DISSERTATION

COMPOUND-GAUSSIAN-REGULARIZED INVERSE PROBLEMS: THEORY, ALGORITHMS, AND NEURAL NETWORKS

Submitted by

Carter Lyons

Department of Mathematics

In partial fulfillment of the requirements

For the Degree of Doctor of Philosophy

Colorado State University

Fort Collins, Colorado

Spring 2024

Doctoral Committee:

Advisor: Margaret Cheney
Co-Advisor: Raghu G. Raj

Mahmood Azimi
Emily King
Jennifer Mueller

ABSTRACT

COMPOUND-GAUSSIAN-REGULARIZED INVERSE PROBLEMS: THEORY, ALGORITHMS, AND NEURAL NETWORKS

Linear inverse problems are frequently encountered in a variety of applications including compressive sensing, radar, sonar, medical, and tomographic imaging. Model-based and data-driven methods are two prevalent classes of approaches used to solve linear inverse problems. Model-based methods incorporate certain assumptions, such as the image prior distribution, into an iterative estimation algorithm, often, as an example, solving a regularized least squares problem. Instead, data-driven methods learn the inverse reconstruction mapping directly by training a neural network structure on actual signal and signal measurement pairs. Alternatively, algorithm unrolling, a recent approach to inverse problems, combines model-based and data-driven methods through the implementation of an iterative estimation algorithm as a deep neural network (DNN). This approach offers a vehicle to embed domain-level and algorithmic insights into the design of neural networks such that the network layers are interpretable. The performance, in reconstructed signal quality, of unrolled DNNs often exceeds that of corresponding iterative algorithms and standard DNNs while doing so in a computationally efficient fashion. In this work, we leverage algorithm unrolling to combine a powerful statistical prior, the compound Gaussian (CG) prior, with the powerful representational ability of machine learning and DNN approaches. Specifically, first we construct a novel iterative CG-regularized least squares algorithm for signal reconstruction and provide a computational theory for this algorithm. Second, using algorithm unrolling, the newly developed CG-based least squares iterative algorithm is transformed into an original DNN in a manner to facilitate the learning of the optimization landscape geometry. Third, a generalization on the newly constructed CG regularized least squares iterative algorithm is developed, theoretically analyzed, and unrolled to yield a novel state-of-the-art DNN that provides a partial learning

of the prior distribution constrained to the CG class of distributions. Fourth, techniques in statistical learning theory are employed for deriving original generalization error bounds on both unrolled DNNs to substantiate theoretical guarantees of each neural network when estimating signals from linear measurements after training. Finally, ample numerical experimentation is conducted for every new CG-based iterative and DNN approach proposed in this paper. Simulation results show our methods outperform previous state-of-the-art iterative signal estimation algorithms and deep-learning-based methods, especially with limited training datasets.

ACKNOWLEDGEMENTS

Sincere gratitude to my advisors Dr. Margaret Cheney and Dr. Raghu G. Raj for providing insightful discussions and relevant feedback throughout this project. This work was sponsored in part by the Office of Naval Research via the NRL base program, under award number N00014-21-1-2145, and under award number N0001423WX01875. Further, this material is based upon research supported in part by the Air Force Office of Scientific Research under award number FA9550-21-1-0169.

DEDICATION

To my wife Sarah for her endless support and encouragement.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Background

In this chapter, we introduce a variety of notations, tools, and techniques from existing literature that are employed throughout the remaining chapters when we construct and evaluate multiple novel compound Gaussian-based iterative and deep learning methods. First, we briefly overview linear inverse problems and common techniques to solve them. Second, we introduce the compound Gaussian prior. Third, we succinctly discuss deep neural networks and the algorithm unrolling technique. Lastly, we overview three common descent methods used in optimization and provided an assortment of theoretical guarantees for each.

1.1 Linear Inverse Problems

Inverse problems in a variety of applications, including compressive sensing (CS) and certain imaging tasks, often assume a linear measurement model. Let $\boldsymbol{c} \in \mathbb{R}^n$ be a vectorized signal observed through measurement matrix $A \in \mathbb{R}^{m \times n}$ with additive zero-mean white Gaussian noise contamination, $\boldsymbol{\nu} \in \mathbb{R}^m$, producing measurements $\boldsymbol{y} \in \mathbb{R}^m$ given as

$$\boldsymbol{y} = A\boldsymbol{c} + \boldsymbol{\nu}. \tag{1.1}$$

In many applications of interest $m \ll n$, which we refer to as the sparse sensing scenario, and the measurements, $\boldsymbol{y}$, are a severe undersampling of the signal, $\boldsymbol{c}$. Signal reconstruction, or estimation, is the linear inverse problem aimed at recovering the signal, $\boldsymbol{c}$, given the measurements, $\boldsymbol{y}$, and sensing matrix, A. Two prevalent classes of approaches are used in practice for image reconstruction, namely model-based and data-driven approaches.

First, model-based approaches incorporate certain assumptions, such as the expected prior density or sparsity level of the signal $\boldsymbol{c}$, into the signal estimation method. Often, model-based methods apply an iterative algorithm designed to minimize an objective function. Examples in-

clude the Iterative Shrinkage and Thresholding Algorithm (ISTA), Bayesian Compressive Sensing (BCS), Basis Pursuit, and Compressive Sampling Matching Pursuit (CoSaMP) [1–4]. Many previous works model the prior density as a generalized Gaussian, $p(\boldsymbol{c}) \propto \exp(-\lambda \|\boldsymbol{c}\|_q^q)$, and solve a corresponding least squares problem with ℓ_q regularization [2, 4–7]. Alternatively, some previous works consider a compound Gaussian (CG) prior [8, 9], which is a class of densities subsuming generalized Gaussian and other densities as special cases, that better captures statistical properties of images [10–12].

Second, data-driven approaches learn the signal reconstruction mapping directly by training a standard deep neural network (DNN) on a dataset of $(\text{measurement, signal})$, i.e. $(\boldsymbol{y}, \boldsymbol{c})$, pairs. Data-driven approaches have been fueled by advances in computing and the success of machine learning in image recognition tasks. Common DNNs in image reconstruction are structured upon a convolutional neural network (CNN), recurrent neural network, or a generative adversarial network (GAN) [13–19].

A more recent approach to signal reconstruction, known as algorithm unrolling or unfolding, stemming from the original work of Gregor and LeCun [20], combines the model-based and data-driven methods by structuring the layers of a DNN upon an iterative algorithm. Unlike a standard DNN, which acts as black-box process, we have an understanding of the inner workings of an unrolled DNN from understanding the original iterative algorithm. Works utilizing algorithm unrolling have shown excellent performance in signal reconstruction while offering simple interpretability of the network layers [21]. Examples of iterative imaging algorithms that have been unrolled include: ISTA [20, 22–26], proximal gradient or gradient descent [27–29], the inertial proximal algorithm for non-convex optimization [30], the primal-dual algorithm [31], alternating direction method of multipliers [32], and half-quadratic splitting [33].

1.1.1 Maximum a Posteriori Estimate

A fruitful way to solve the linear inverse problem to (1.1) is through Bayesian estimation. Let $p(\boldsymbol{c} \,|\, \boldsymbol{y})$ be the conditional probability density of the signal, $\boldsymbol{c}$, given the measurements, $\boldsymbol{y}$.

Furthermore, let $p(\boldsymbol{y} \,|\, \boldsymbol{c})$ be the conditional probability density of the measurements, $\boldsymbol{y}$, given the signal, $\boldsymbol{c}$, and let $p(\boldsymbol{c})$ be the probability density of the signal, $\boldsymbol{c}$. From Bayesian estimation, consider the *maximum a posteriori* (MAP) estimate of the signal in (1.1) given as

$$\begin{aligned}\widehat{\boldsymbol{c}} = \arg\max_{\boldsymbol{c}} \, p(\boldsymbol{c} \,|\, \boldsymbol{y}) &= \arg\max_{\boldsymbol{c}} \, p(\boldsymbol{y} \,|\, \boldsymbol{c}) p(\boldsymbol{c}) \\ &= \arg\max_{\boldsymbol{c}} \, \left\{ \ln\left(p(\boldsymbol{y} \,|\, \boldsymbol{c})\right) + \ln\left(p(\boldsymbol{c})\right) \right\} \\ &= \arg\min_{\boldsymbol{c}} \, \left\{ -\ln\left(p(\boldsymbol{y} \,|\, \boldsymbol{c})\right) - \ln\left(p(\boldsymbol{c})\right) \right\}. \end{aligned} \tag{1.2}$$

Note, for a Gaussian random vector $\boldsymbol{\xi} \in \mathbb{R}^n \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma_X)$, deterministic matrix $M \in \mathbb{R}^{m \times n}$, and deterministic vector $\boldsymbol{b} \in \mathbb{R}^m$

$$M\boldsymbol{\xi} + \boldsymbol{b} \sim \mathcal{N}\left(M\boldsymbol{\mu} + \boldsymbol{b}, M\Sigma_X M^T\right). \tag{1.3}$$

Since $\boldsymbol{\nu} \sim \mathcal{N}(\boldsymbol{0}, \sigma^2 I)$ then, taking $M = I$, $\boldsymbol{\xi} = \boldsymbol{\nu}$, and $\boldsymbol{b} = A\boldsymbol{c}$ in (1.3), we have

$$p(\boldsymbol{y} \,|\, \boldsymbol{c}) = p\left(A\boldsymbol{c} + \boldsymbol{\nu} \,|\, \boldsymbol{c}\right) \sim \mathcal{N}\left(A\boldsymbol{c}, \sigma^2 I\right)$$

and so

$$\begin{aligned} -\ln\left(p(\boldsymbol{y} \,|\, \boldsymbol{c})\right) &= -\ln\left(\frac{1}{\sqrt{(2\pi\sigma^2)^n}} \exp\left(-\frac{\sigma^2}{2}\left(\boldsymbol{y} - A\boldsymbol{c}\right)^T\left(\boldsymbol{y} - A\boldsymbol{c}\right)\right)\right) \\ &= \frac{n}{2}\ln(2\pi\sigma^2) + \frac{\sigma^2}{2}\left(\boldsymbol{y} - A\boldsymbol{c}\right)^T\left(\boldsymbol{y} - A\boldsymbol{c}\right). \end{aligned} \tag{1.4}$$

Combining equation (1.2) and (1.4) gives

$$\begin{aligned}\widehat{\boldsymbol{c}} &= \arg\min_{\boldsymbol{c}} \left\{ \frac{n}{2}\ln(2\pi\sigma^2) + \frac{\sigma^2}{2}(\boldsymbol{y}-A\boldsymbol{c})^T(\boldsymbol{y}-A\boldsymbol{c}) - \ln(p(\boldsymbol{c})) \right\} \\ &= \arg\min_{\boldsymbol{c}} \left\{ \frac{1}{2}(\boldsymbol{y}-A\boldsymbol{c})^T(\boldsymbol{y}-A\boldsymbol{c}) - \frac{1}{\sigma^2}\ln(p(\boldsymbol{c})) \right\} \\ &= \arg\min_{\boldsymbol{c}} \frac{1}{2}\|\boldsymbol{y}-A\boldsymbol{c}\|_2^2 + R(\boldsymbol{c}) \end{aligned} \tag{1.5}$$

where $R(\boldsymbol{c}) = -\frac{1}{\sigma^2}\ln(p(\boldsymbol{c}))$. Equation (1.5) is a regularized least squares estimate where the regularization is informed by the prior distribution of the signal, $\boldsymbol{c}$. Therefore, we equivalently view the MAP estimator as a regularized least squares estimate for properly chosen regularization function. In either case, the prior density assumed for the signal, $\boldsymbol{c}$, is a critical choice to incorporate domain level knowledge into the signal reconstruction process.

1.1.2 Change-of-Basis Representation and Sparsity

Frequently, A is decomposed into the product an observation matrix $\Psi \in \mathbb{R}^{m\times n}$ and a change-of-basis transformation $\Phi \in \mathbb{R}^{n\times n}$. That is, $A = \Psi\Phi$, which corresponds to expressing some original signal $\boldsymbol{x} \in \mathbb{R}^n$ with respect to (w.r.t.) the transformation Φ as $\boldsymbol{x} = \Phi\boldsymbol{c}$. When A is decomposed, the linear measurement model from (1.1) is instead given by

$$\boldsymbol{y} = \Psi\Phi\boldsymbol{c} + \boldsymbol{\nu}. \tag{1.6}$$

The decomposition of A is often employed when $\boldsymbol{x}$ has some desirable properties, such as sparsity, when represented through the change-of-basis.

Recall that a digital signal $\boldsymbol{x} \in \mathbb{R}^n$ is *k-sparse* if at most k entries of $\boldsymbol{x}$ are non-zero. For example, natural images are known to have a sparse representation under a wavelet dictionary or discrete cosine transformation (DCT). In such a case, $\boldsymbol{x}$ is a vectorized image, Φ is the wavelet dictionary or DCT, and $\boldsymbol{c}$ are the sparsity coefficients of the image $\boldsymbol{x}$.

1.2 Compound Gaussian Prior

As shown in Section 1.1.1, the choice of regularization on, or prior density of, the signal of interest, $\boldsymbol{c}$, is a crucial component to incorporate domain level knowledge into a linear inverse problem. Many previous works have employed a generalized Gaussian prior. The generalized Gaussian prior includes a Gaussian prior, corresponding to a Tikhonov regression, [34, 35] and a Laplacian prior, as is predominant in the compressive sensing (CS) framework, [1, 2, 4–6, 36]. Additional regularization, not derived from a specific prior density, have been implemented for image reconstruction including total variation norm [36–39], stochastic based regularization [40], and deep learning based regularization [18, 41, 42].

Through the study of the statistics of image sparsity coefficients, it has been shown that sparse coefficients of natural images exhibit self-similarity, heavy-tailed marginal distributions, and self-reinforcement among local coefficients [11]. Such properties are not encompassed by the generalized Gaussian prior typically assumed for the image sparsity coefficients. Instead, classes of densities, known as CG densities [43] and Gaussian scale mixtures [10, 11], better capture statistical properties of natural images and images from other modalities such as radar [12]. A useful formulation of the CG prior lies in modeling the sparse coefficients of images as

$$\boldsymbol{c} = \boldsymbol{z} \odot \boldsymbol{u} \coloneqq \begin{bmatrix} z_1 u_1 & z_2 u_2 & \cdots & z_n u_n \end{bmatrix}^T \tag{1.7}$$

such that $\boldsymbol{z}$ is a positive random vector, $\boldsymbol{u} \sim \mathcal{N}(\mathbf{0}, \Sigma_u)$, and $\boldsymbol{u}$ and $\boldsymbol{z}$ are independent random vectors. We denote $\boldsymbol{z}$ as the scale variable and $\boldsymbol{u}$ as the Gaussian variable. We remark that Gaussian scale mixtures are a special case of CG densities corresponding to the restriction that $\boldsymbol{z} = z\mathbf{1} \coloneqq z[1, 1, \ldots, 1]^T$. That is, $\boldsymbol{z}$ is not a random vector but rather a scalar random variable. Hence, the CG formulation in (1.7) allows for greater statistical representation ability than the Gaussian scale mixture density.

Finally, we remark that one practical formulation of the scale variable, as discussed in [8, 11], is to take

$$\boldsymbol{z} = h(\boldsymbol{x}), \tag{1.8}$$

where $h : \mathbb{R} \to \mathbb{R}$ is a componentwise, positive, nonlinear function and $\boldsymbol{x}$ follows a multi-scale Gaussian tree process. Previously, the CG prior has been used, with $h(x) = \sqrt{\exp(x/\alpha)}$ for $\alpha \in (0, \infty)$, in a hierarchical Bayesian MAP (HB-MAP) estimate of wavelet and discrete cosine transformation (DCT) coefficients of images [8, 9]. The HB-MAP algorithm produces reconstructed images with superior quality, measured by the structural similarity index (SSIM), over other state-of-the-art CS techniques [8]. Furthermore, the Gaussian scale mixture prior has been successfully used for image denoising [44] and hyperspectral image compressive sensing [45].

1.2.1 Subsumed Distributions

The CG class of distributions subsumes many well-known distributions including the generalized Gaussian that is frequently used in prior works on image reconstruction and linear inverse problems. Thus, the use of the CG prior can be interpreted as a generalization upon these previous works. Next, we state a collection of well-known distributions that are special cases of the CG prior.

Proposition 1.2.1. *The generalized Gaussian, student's t, α-stable, and symmetrized Gamma distributions are special cases of the compound Gaussian distribution.*

The result of Proposition 1.2.1 is found in the literature [10, 11]. Additionally, Proposition 1.2.6 provides the specific nonlinearity, h, producing a Laplace prior allowing an interpretation of the CG prior as a generalization of previous CS work. To prove Proposition 1.2.6 we first show a handful of lemmas to describe the decomposition of a Laplace random variable. First, a lemma on the distribution of the square root of an exponential random variable; the result of which exists in the literature [46].

Lemma 1.2.2 ([46]). *If $Z \sim \text{Exp}(\lambda)$ then $\sqrt{Z} \sim \text{Rayleigh}\left(\frac{1}{\sqrt{2\lambda}}\right)$. That is,*

$$\sqrt{Z} \sim p_{\sqrt{Z}}(z) = 2z\lambda e^{-\lambda z^2}.$$

Proof. Recall the probability density function of Z is $p_Z(z) = \lambda e^{-\lambda z}$ and let $P_Z(z) = 1 - e^{-\lambda z}$ denote the cumulative distribution function. Then observe

$$\mathbb{P}(\sqrt{Z} \leq z) = \mathbb{P}(Z \leq z^2) = P_Z(z^2)$$

and so

$$p_{\sqrt{Z}}(z) = \frac{d}{dz}\mathbb{P}(\sqrt{Z} \leq z) = \frac{d}{dz}P_Z(z^2) = 2zp_Z(z^2) = 2z\lambda e^{-\lambda z^2}. \qquad \boxtimes$$

Second, we define the Inverse Gaussian or Wald distribution, from [47], that is parameterized by $\mu > 0$ and $\lambda > 0$.

Definition 1.2.3 ([47]). *Random variable X is **Inverse Gaussian** of parameters $\mu > 0$ and $\lambda > 0$, written $X \sim \text{IG}(\mu, \lambda)$, if and only if it has probability density function*

$$p_X(x; \mu, \lambda) = \sqrt{\frac{\lambda}{2\pi x^3}} \exp\left(-\frac{\lambda(x-\mu)^2}{2\mu^2 x}\right).$$

Third, a lemma on the mean of an Inverse Gaussian random variable; the results of which exists in the literature [47].

Lemma 1.2.4 ([47]). *Let $X \sim \text{IG}(\mu, \lambda)$. Then $\mathbb{E}(X) = \mu$.*

Proof. We calculate $\mathbb{E}(X)$ using the moment generating function (MGF) of X. Observe,

$$M_X(t) = \int_0^\infty e^{itx} p_X(x; \mu, \lambda)dx = \sqrt{\frac{\lambda}{2\pi}} \int_0^\infty x^{-\frac{3}{2}} \exp\left(itx - \frac{\lambda(x-\mu)^2}{2\mu^2 x}\right) dx.$$

Now, let $\kappa = 2it\mu^2/\lambda$ and note

$$\begin{aligned}
itx - \frac{\lambda(x-\mu)^2}{2\mu^2 x} &= \frac{\lambda x^2}{2\mu^2 x}\kappa - \frac{\lambda(x-\mu)^2}{2\mu^2 x} \\
&= -\frac{\lambda}{2\mu^2 x}\left(x^2(1-\kappa) - 2\mu x + \mu^2\right) \\
&= \frac{\lambda}{\mu} - \frac{\lambda}{2\mu^2 x}(x^2(1-\kappa) + \mu^2) \\
&= \frac{\lambda}{\mu} - \frac{\lambda}{\mu}\sqrt{1-\kappa} + \frac{\lambda}{2\mu^2 x}2x\mu\sqrt{1-\kappa} - \frac{\lambda}{2\mu^2 x}(x^2(1-\kappa) + \mu^2) \\
&= \frac{\lambda}{\mu} - \frac{\lambda}{\mu}\sqrt{1-\kappa} - \frac{\lambda}{2\mu^2 x}(x^2(1-\kappa) - 2\mu\sqrt{1-\kappa} + \mu^2) \\
&= \frac{\lambda}{\mu} - \frac{\lambda}{\mu}\sqrt{1-\kappa} - \frac{\lambda}{2\mu^2 x}(1-\kappa)\left(x^2 - \frac{2\mu}{\sqrt{1-\kappa}} + \frac{\mu^2}{1-\kappa}\right) \\
&= \frac{\lambda}{\mu} - \frac{\lambda}{\mu}\sqrt{1-\kappa} - \frac{\lambda}{2\mu^2 x}(1-\kappa)\left(x - \frac{\mu}{\sqrt{1-\kappa}}\right)^2.
\end{aligned}$$

Next, define $\gamma = \mu/\sqrt{1-\kappa}$. Then

$$itx - \frac{\lambda(x-\mu)^2}{2\mu^2 x} = \frac{\lambda}{\mu} - \frac{\lambda}{\gamma} - \frac{\lambda(x-\gamma)^2}{2\gamma^2 x}.$$

Hence

$$\begin{aligned}
M_X(t) &= \sqrt{\frac{\lambda}{2\pi}}\int_0^\infty x^{-\frac{3}{2}}\exp\left(\frac{\lambda}{\mu} - \frac{\lambda}{\gamma} - \frac{\lambda(x-\gamma)^2}{2\gamma^2 x}\right)dx \\
&= \exp\left(\frac{\lambda}{\mu} - \frac{\lambda}{\gamma}\right)\int_0^\infty \sqrt{\frac{\lambda}{2\pi}}x^{-\frac{3}{2}}\exp\left(-\frac{\lambda(x-\gamma)^2}{2\gamma^2 x}\right)dx \\
&= \exp\left(\frac{\lambda}{\mu} - \frac{\lambda}{\gamma}\right)\int_0^\infty p_X(x;\gamma,\lambda)dx \\
&= \exp\left(\frac{\lambda}{\mu} - \frac{\lambda}{\gamma}\right).
\end{aligned}$$

We calculate the mean via $\mathbb{E}(X) = \frac{1}{i}\frac{d}{dt}M_X(t)\Big|_{t=0}$. Thus, observe

$$\begin{aligned}\frac{d}{dt}M_X(t) &= \frac{d}{dt}\exp\left(\frac{\lambda}{\mu} - \frac{\lambda}{\mu}\left(1 - \frac{2it\mu^2}{\lambda}\right)^{1/2}\right) \\ &= i\mu\left(1 - \frac{2it\mu^2}{\lambda}\right)^{-\frac{1}{2}}\exp\left(\frac{\lambda}{\mu} - \frac{\lambda}{\mu}\left(1 - \frac{2it\mu^2}{\lambda}\right)^{1/2}\right).\end{aligned}$$

Therefore

$$\mathbb{E}(X) = \frac{1}{i}\frac{d}{dt}M_X(t)\Big|_{t=0} = \mu. \qquad \boxtimes$$

Fourth, with the Inverse Gaussian distribution and its mean, we show that a Laplace distribution may be decomposed as a Gaussian scale mixture. An alternative proof of this result is provided an [10].

Lemma 1.2.5 ([10, 11])**.** *Random variable* $X \sim \text{Laplace}(\lambda)$ *if and only if* $X = \sqrt{Z}U$ *for* $Z \sim \text{Exp}\left(\frac{1}{2\lambda^2}\right)$ *and* $U \sim \mathcal{N}(0,1)$.

Proof. Let $Z \sim \text{Exp}\left(\frac{1}{2\lambda^2}\right)$ and $U \sim \mathcal{N}(0,1)$ and define $X = \sqrt{Z}U$. Therefore

$$\begin{aligned}p_X(x) &= \int_0^\infty p_{\sqrt{Z}}(z)p_U(x/z)\frac{1}{z}dz \\ &= \int_0^\infty 2z\frac{1}{2\lambda^2}e^{-\frac{z^2}{2\lambda^2}}\frac{1}{\sqrt{2\pi}}e^{-\frac{1}{2}\frac{x^2}{z^2}}\frac{1}{z}dz \\ &= \frac{1}{\lambda^2\sqrt{2\pi}}\int_0^\infty \exp\left(-\frac{1}{2}\left(\frac{z^4 + \lambda^2x^2}{\lambda^2z^2}\right)\right)dz \\ &= \frac{1}{\lambda^2\sqrt{2\pi}}\int_0^\infty \exp\left(-\frac{1}{2}\left(\frac{z^4 - 2z^2\lambda|x| + \lambda^2|x|^2 + 2z^2\lambda|x|}{\lambda^2z^2}\right)\right)dz \\ &= \frac{1}{\lambda^2\sqrt{2\pi}}e^{-\frac{|x|}{\lambda}}\int_0^\infty \exp\left(-\frac{1}{2}\frac{(z^2 - \lambda|x|)^2}{\lambda^2z^2}\right)dz. \qquad (1.9)\end{aligned}$$

Define $\gamma = |x|^2$ and $\mu = |x|\lambda$. Then

$$\frac{1}{\sqrt{2\pi}}\int_0^\infty \exp\left(-\frac{1}{2}\frac{(z^2 - \lambda|x|)^2}{\lambda^2z^2}\right)dz = \frac{1}{\sqrt{\gamma}}\int_0^\infty \sqrt{\frac{\gamma}{2\pi}}\exp\left(-\frac{\gamma(z^2 - \mu)^2}{2\mu^2z^2}\right)dz \qquad (1.10)$$

Next, define $u = z^2$ then $du = 2zdz$ implying

$$\begin{aligned}
\frac{1}{\sqrt{\gamma}} \int_0^\infty \sqrt{\frac{\gamma}{2\pi}} \exp\left(-\frac{\gamma(z^2-\mu)^2}{2\mu^2 z^2}\right) dz &= \frac{1}{2\sqrt{\gamma}} \int_0^\infty \sqrt{\frac{\gamma}{2\pi}} u^{-\frac{1}{2}} \exp\left(-\frac{\gamma(u-\mu)^2}{2\mu^2 u}\right) du \\
&= \frac{1}{2\sqrt{\gamma}} \int_0^\infty u\sqrt{\frac{\gamma}{2\pi u^3}} \exp\left(-\frac{\gamma(u-\mu)^2}{2\mu^2 u}\right) du \\
&= \frac{1}{2\sqrt{\gamma}} \int_0^\infty u\, p_W(u;\mu,\gamma) du \\
&= \frac{1}{2\sqrt{\gamma}} \mathbb{E}(W) \qquad (1.11)
\end{aligned}$$

where $W \sim \text{IG}(\mu, \gamma)$. Combining equation (1.9), equation (1.10), and equation (1.11) gives

$$\begin{aligned}
p_X(x) &= \frac{1}{\lambda^2\sqrt{2\pi}} e^{-\frac{|x|}{\lambda}} \int_0^\infty \exp\left(-\frac{1}{2}\frac{(z^2-\lambda|x|)^2}{\lambda^2 z^2}\right) dz \\
&= \frac{1}{\lambda^2} e^{-\frac{|x|}{\lambda}} \frac{1}{\sqrt{\gamma}} \int_0^\infty \sqrt{\frac{\gamma}{2\pi}} \exp\left(-\frac{\gamma(z^2-\mu)^2}{2\mu^2 z^2}\right) dz \\
&= \frac{1}{\lambda^2} e^{-\frac{|x|}{\lambda}} \frac{1}{2\sqrt{\gamma}} \mathbb{E}(W).
\end{aligned}$$

Since $\mathbb{E}(W) = \mu = |x|\lambda$ by Lemma 1.2.4 then we have

$$\begin{aligned}
p_X(x) &= \frac{1}{\lambda^2} e^{-\frac{|x|}{\lambda}} \frac{1}{2\sqrt{\gamma}} \mathbb{E}(W) \\
&= \frac{1}{\lambda^2} e^{-\frac{|x|}{\lambda}} \frac{1}{2|x|} |x|\lambda \\
&= \frac{1}{2\lambda} e^{-\frac{|x|}{\lambda}}.
\end{aligned}$$

Implying that $X \sim$ Laplace(λ).

Similarly, when $X \sim$ Laplace(λ) we reverse the steps above to show $X = \sqrt{Z}U$ for $Z \sim$ Exp $\left(\frac{1}{2\lambda^2}\right)$ and $U \sim \mathcal{N}(0,1)$. ⊠

Finally, we prove Proposition 1.2.6, which constructs a nonlinear function, h, such that for multivariate Gaussian random vectors X and U the random variable $h(X) \odot U$ is a multivariate Laplace random vector. This result likely appears in the existing literature already.

Proposition 1.2.6. *Let Υ be the cumulative distribution function (CDF) of a standard Gaussian random variable. Define*

$$h(x) = \left(-2\lambda^2 \ln(1 - \Upsilon(x))\right)^{1/2}.$$

Then $\boldsymbol{c} = h(\boldsymbol{x}) \odot \boldsymbol{u}$, for $\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{0}, I)$ and $\boldsymbol{u} \sim \mathcal{N}(\boldsymbol{0}, I)$, has a Laplace distribution. That is $\boldsymbol{c} \sim \lambda \exp(-\lambda \|\boldsymbol{c}\|_1)/2$.

Proof. By Lemma 1.2.5, a Laplace random variable $X \sim \lambda \exp(-\lambda x)/2$ is decomposed as $X = \sqrt{Z}U$, for $U \sim \mathcal{N}(0, 1)$ and $Z \sim \text{Exp}(1/2\lambda^2)$. Let $F_Z(z) = 1 - \exp(-\frac{1}{2\lambda^2} z)$ be the cumulative distribution function of Z. Due to independence in the components of $\boldsymbol{c}$, we only need to show that $h(x_i)^2 \sim Z$, which is easily observed since

$$\begin{aligned}
\mathbb{P}(h(x_i)^2 \leq x) &= \mathbb{P}\left(-2\lambda^2 \ln(1 - \Upsilon(x_i)) \leq x\right) \\
&= \mathbb{P}\left(\ln(1 - \Upsilon(x_i)) \geq -\frac{1}{2\lambda^2} x\right) \\
&= \mathbb{P}\left(1 - \Upsilon(x_i) \geq \exp\left(-\frac{1}{2\lambda^2} x\right)\right) \\
&= \mathbb{P}\left(\Upsilon(x_i) \leq 1 - \exp\left(-\frac{1}{2\lambda^2} x\right)\right) \\
&= \mathbb{P}\left(x_i \leq \Upsilon^{-1}\left(F_Z(x)\right)\right) \\
&= F_Z(x).
\end{aligned}$$

⊠

1.3 Deep Neural Networks

A DNN may be viewed as a collection of ordered layers, denoted $\boldsymbol{L}_0, \boldsymbol{L}_1, \ldots, \boldsymbol{L}_K$ for $K > 1$, that form a directed acyclic graph starting with the input layer, $\boldsymbol{L}_0$, and ending at the output layer, $\boldsymbol{L}_K$. Intermediate layers $\boldsymbol{L}_1, \ldots, \boldsymbol{L}_{K-1}$ are known as hidden layers. Each layer, $\boldsymbol{L}_k$, contains d_k hidden units [48], or nodes, and for simplicity, we identify the nodes with the computed values assigned to each node when a signal is processed by the DNN.

Real vector-valued signals and functions are considered for everything that follows; however, complex vector-valued signals and functions could be substituted instead. A function $\boldsymbol{f}_k$:

$\mathbb{R}^{d_{i_1(k)}} \times \cdots \times \mathbb{R}^{d_{i_j(k)}} \to \mathbb{R}^{d_k}$, that is parameterized by some $\boldsymbol{\theta}_k$, defines the computation, i.e. signal transmission, at layer $\boldsymbol{L}_k$ where $\mathcal{I}_k \coloneqq \{i_1(k), \ldots, i_j(k)\} \subseteq \{0, 1, \ldots, k-1\}$ are the indices of layers that feed into $\boldsymbol{L}_k$. That is, given an input signal $\overline{\boldsymbol{y}} \in \mathbb{R}^{d_0}$ assigned to $\boldsymbol{L}_0$, a DNN is a composition of parameterized vector input and vector output functions where

$$\boldsymbol{L}_k \equiv \boldsymbol{f}_k \left(\boldsymbol{L}_{i_1(k)}, \ldots, \boldsymbol{L}_{i_j(k)}; \boldsymbol{\theta}_k \right).$$

We now provide two examples of common DNNs and the structure of the parametric functions $\boldsymbol{f}_k$ defining each layer in these DNNs. First, fully-connected or dense neural networks.

Example 1.3.1. ***Fully-Connected Neural Network (FCNN).***

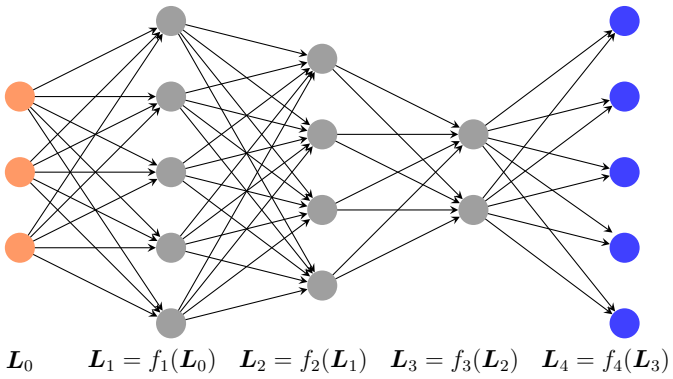


Figure 1.1: A sample deep neural network with the following structure: Input layer, $\boldsymbol{L}_0$, has three hidden units. The first hidden layer, $\boldsymbol{L}_1$, has five nodes (i.e. $f_1 : \mathbb{R}^3 \to \mathbb{R}^5$). The second hidden layer, $\boldsymbol{L}_2$, has four nodes (i.e. $f_2 : \mathbb{R}^5 \to \mathbb{R}^4$). The third hidden layer, $\boldsymbol{L}_3$, has two nodes (i.e. $f_3 : \mathbb{R}^4 \to \mathbb{R}^2$). Finally, the output layer, L_4, has five hidden units (i.e. $f_4 : \mathbb{R}^2 \to \mathbb{R}^5$).

Each layer $\boldsymbol{L}_k$ in a FCNN is a non-linear activation function, σ, composed with an affine transformation of the previous layer $\boldsymbol{L}_{k-1}$. That is, $\mathcal{I}_k = \{k-1\}$, $\boldsymbol{\theta}_k = [W_k, \boldsymbol{b}_k]$ where $W_k \in \mathbb{R}^{d_{k-1} \times d_k}$ is a weight matrix and $\boldsymbol{b}_k \in \mathbb{R}^{d_k}$ is additive bias, and

$$\boldsymbol{L}_k \equiv \boldsymbol{f}_k(\boldsymbol{L}_{k-1}; \boldsymbol{\theta}_k = [W_k, \boldsymbol{b}_k]) = \sigma(W_k \boldsymbol{L}_{k-1} + \boldsymbol{b}_k)$$

for every $k \in \{1, 2, \ldots, K\}$. A sample fully-connected DNN diagram is shown in Figure 1.1 where $K = 4$ and $\boldsymbol{L}_0$, $\boldsymbol{L}_1$, $\boldsymbol{L}_2$, $\boldsymbol{L}_3$, and $\boldsymbol{L}_4$ have 3, 5, 4, 2, and 5 hidden units, respectively.

The second DNN example is a convolutional neural network.

Example 1.3.2. ***Convolutional Neural Network (CNN).***

Each layer $\boldsymbol{L}_k$ in a CNN is a non-linear activation function, σ, composed with a convolution of the previous layer $\boldsymbol{L}_{k-1}$. That is, $\mathcal{I}_k = \{k-1\}$, $\boldsymbol{\theta}_k = [W_k, \boldsymbol{b}_k]$ where W_k is a convolutional weight kernel and $\boldsymbol{b}_k$ is additive bias, and

$$\boldsymbol{L}_k \equiv \boldsymbol{f}_k(\boldsymbol{L}_{k-1}; \boldsymbol{\theta}_k = [W_k, \boldsymbol{b}_k]) = \sigma(W_k \star \boldsymbol{L}_{k-1} + \boldsymbol{b}_k)$$

for every $k \in \{1, 2, \ldots, K\}$ where $\star$ denotes a discrete convolution.

Note that a convolution is a linear transformation and thus is able to be expressed as a matrix multiplication. Hence, a convolutional network is a special case of a fully-connected network where the weight matrix has Toeplitz structure induced by the convolutional operation. Additionally, the convolution in every layer of a CNN may be implemented as a d-dimensional discrete convolution where the input signal to $\boldsymbol{f}_k$ is a discrete signal (possibly reshaped to be) of d inputs. For instance, when a CNN is processing 2-dimensional images, then each convolution is 2-dimensional and the input to each $\boldsymbol{f}_k$ is a matrix representing an image.

We remark that a DNN can be, equivalently, viewed as a directed graph. In this formulations, each hidden unit, or numerical value assigned to the hidden unit, corresponds to a vertex in the graph and the directed edges between vertices are determined by the operations $\boldsymbol{f}_1, \boldsymbol{f}_2, \ldots, \boldsymbol{f}_K$. In a fully connected network, for example, every hidden node in layer $\boldsymbol{L}_{k-1}$ has an edge connecting to every node in layer $\boldsymbol{L}_k$.

A DNN learns, or trains, its parameters, $\boldsymbol{\Theta} = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K)$, by optimizing a loss function $\mathcal{L}(\boldsymbol{\Theta})$ over a training dataset $\mathcal{D} = \{(\overline{\boldsymbol{y}}_i, \overline{\boldsymbol{c}}_i) : i = 1, 2, \ldots, N_s\}$ where each $(\overline{\boldsymbol{y}}_i, \overline{\boldsymbol{c}}_i)$ satisfies equation (1.1). Let $\widehat{\boldsymbol{c}}(\overline{\boldsymbol{y}}_i; \boldsymbol{\Theta})$ denote the DNN output given the input $\overline{\boldsymbol{y}}_i$. Then the DNN loss function is often defined as $\mathcal{L}(\boldsymbol{\Theta}) \coloneqq \frac{1}{N_s} \sum_{i=1}^{N_s} L\left(\widehat{\boldsymbol{c}}(\overline{\boldsymbol{y}}_i; \boldsymbol{\Theta}), \overline{\boldsymbol{c}}_i\right)$ where $L(\boldsymbol{x}_1, \boldsymbol{x}_2) : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ calculates the error between inputs $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$. That is, $L\left(\widehat{\boldsymbol{c}}(\overline{\boldsymbol{y}}_i; \boldsymbol{\Theta}), \overline{\boldsymbol{c}}_i\right)$ is the error between the network output, $\widehat{\boldsymbol{c}}(\overline{\boldsymbol{y}}_i; \boldsymbol{\Theta})$, and the actual coefficients, $\overline{\boldsymbol{c}}_i$. Common selections for the error function $L(\boldsymbol{x}_1, \boldsymbol{x}_2)$ in image reconstruction neural networks include mean-squared error (MSE), mean absolute error (MAE), normalized mean-squared error (NMSE), peak signal-to-noise ratio (PSNR), or SSIM [49]. After training a DNN, the cascade of function compositions can collectively model a desired transformation from an input space to an output space.

1.3.1 Algorithm Unrolling

Algorithm unrolling or unfolding [21] is a technique to create a DNN that is structurally informed by some iterative algorithm. Specifically, the operations from each step k of the iterative algorithm are assigned as the function $\boldsymbol{f}_k$ defining layer k. That is, layer k in the DNN should correspond to the output of k iterations of the original iterative algorithm. Hence, an unrolled DNN with K layers corresponds to the iterative algorithm truncated after K iterations.

Next, any parameters, $\boldsymbol{\theta}_k$, on each step k of the iterative algorithm parameterize $\boldsymbol{f}_k$ in the DNN [20, 21]. Note, some parameters may be artificially introduced through the unrolling process such that these parameters are related to step k of the iterative algorithm. For instance, previous works have introduced a cascade of composed convolutional layers that correspond to applying a proximal operator in an unrolled proximal gradient descent network or unrolled ISTA network [22, 23, 26, 27]. In training the unrolled DNN, each $\boldsymbol{\theta}_k$ is learned, which optimizes the iterative algorithm to produce improved image estimates.

1.4 Steepest Descent

Steepest descent is a general method for numerically optimizing differentiable cost functions that subsumes both gradient descent and Newton's descent as special cases [50]. Broadly, steepest descent generates a sequence of estimates where each estimate minimizes the linear approximation of the underlying cost function centered about the previous estimate. For everything that follows we consider only the vector space $\mathbb{R}^n$ however any n-dimensional vector space V, which is isomorphic to $\mathbb{R}^n$, could be substituted instead. Furthermore, all of the results of this section appear in the literature [50].

To define the iterative update for steepest descent we first require the notation of a dual norm.

Proposition 1.4.1 ([50])**.** *Let $\|\cdot\| : \mathbb{R}^n \to \mathbb{R}$ be any norm on $\mathbb{R}^n$. The function $\|\cdot\|_* : \mathbb{R}^n \to \mathbb{R}$ defined as*

$$\|\boldsymbol{w}\|_* = \sup_{\|\boldsymbol{v}\|=1} \boldsymbol{w}^T\boldsymbol{v} \tag{1.12}$$

*produces a norm on $\mathbb{R}^n$ called the **dual norm** of $\|\cdot\|$.*

Proof. We show $\|\cdot\|_*$ satisfies the properties of subadditivity, absolute homogeneity, positive definiteness, and non-negativity.

1. Subadditivity: $\|\boldsymbol{x}+\boldsymbol{y}\|_* \leq \|\boldsymbol{x}\|_* + \|\boldsymbol{y}\|_*$ for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$.

   Let $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$. By definition of the supremum, for all $\boldsymbol{w}$ such that $\|\boldsymbol{w}\| = 1$ it holds that $\boldsymbol{x}^T\boldsymbol{w} \leq \sup_{\|\boldsymbol{v}\|=1} \boldsymbol{x}^T\boldsymbol{v} = \|\boldsymbol{x}\|_*$ and $\boldsymbol{y}^T\boldsymbol{w} \leq \sup_{\|\boldsymbol{v}\|=1} \boldsymbol{y}^T\boldsymbol{v} = \|\boldsymbol{y}\|_*$. Thus, $\boldsymbol{x}^T\boldsymbol{w} + \boldsymbol{y}^T\boldsymbol{w} = (\boldsymbol{x}+\boldsymbol{y})^T\boldsymbol{w} \leq \|\boldsymbol{x}\|_* + \|\boldsymbol{y}\|_*$ for all $\boldsymbol{w}$ where $\|\boldsymbol{w}\| = 1$. Therefore, for some $\boldsymbol{v}^* \in \mathbb{R}^n$ satisfying $\|\boldsymbol{v}\|^* = 1$

   $$\|\boldsymbol{x}+\boldsymbol{y}\|_* = \sup_{\|\boldsymbol{v}\|=1} (\boldsymbol{x}+\boldsymbol{y})^T\boldsymbol{v} = (\boldsymbol{x}+\boldsymbol{y})^T\boldsymbol{v}^* \leq \|\boldsymbol{x}\|_* + \|\boldsymbol{y}\|_*.$$

2. Absolute Homogeneity: $\|s\boldsymbol{x}\|_* = |s| \, \|\boldsymbol{x}\|_*$ for all $\boldsymbol{x} \in \mathbb{R}^n$ and any scalar s.

Let $\boldsymbol{x} \in \mathbb{R}^n$ and s be a scalar. If $s = 0$ then $\|s\boldsymbol{x}\|_* = \|\mathbf{0}\|_* = \sup_{\|\boldsymbol{v}\|=1} \mathbf{0}^T\boldsymbol{v} = \mathbf{0} = |s|\,\|\boldsymbol{x}\|_*$. For $s \neq 0$ observe

$$\|s\boldsymbol{x}\|_* = \sup_{\|\boldsymbol{v}\|=1} (s\boldsymbol{x})^T\boldsymbol{v} = \sup_{\|s^{-1}\widetilde{\boldsymbol{v}}\|=1} \boldsymbol{x}^T\widetilde{\boldsymbol{v}} = \sup_{\|\widetilde{\boldsymbol{v}}\|=|s|} \boldsymbol{x}^T\widetilde{\boldsymbol{v}} = \sup_{\|\boldsymbol{w}\|=1} \boldsymbol{x}^T|s|\,\|\boldsymbol{w}\| = |s|\,\|\boldsymbol{x}\|_*$$

where we defined $\widetilde{\boldsymbol{v}} = s\boldsymbol{v}$ in the second equality and $\boldsymbol{w} = |s|^{-1}\widetilde{\boldsymbol{v}}$ in the fourth equality.

3. Positive definiteness: $\|\boldsymbol{x}\|_* = 0$ if and only if $\boldsymbol{x} = \mathbf{0}$.

   Take $s = 0$ then $\|\mathbf{0}\|_* = \|s\boldsymbol{x}\|_* = 0$, for any $\boldsymbol{x} \in \mathbb{R}^n$, by absolute homogeneity. Now assume that $\|\boldsymbol{x}\|_* = 0$. For contradiction assume $\boldsymbol{x} \neq \mathbf{0}$. Observe

   $$0 = \|\boldsymbol{x}\|_* = \sup_{\|\boldsymbol{v}\|=1} \boldsymbol{x}^T\boldsymbol{v} \geq \boldsymbol{x}^T\frac{\boldsymbol{x}}{\|\boldsymbol{x}\|} = \frac{\|\boldsymbol{x}\|_2^2}{\|\boldsymbol{x}\|},$$

   implying that $0 = \|\boldsymbol{x}\|_2^2$ or $\boldsymbol{x} = \mathbf{0}$ providing a contradiction.

4. Non-negativity: $||\boldsymbol{x}||_* \geq 0$ for all $\boldsymbol{x} \in \mathbb{R}^n$.

   By positive definiteness $\|\mathbf{0}\|_* = 0$. Let $\boldsymbol{x} \in \mathbb{R}^n$ and assume $\boldsymbol{x} \neq \mathbf{0}$. Then

   $$\|\boldsymbol{x}\|_* = \sup_{\|\boldsymbol{v}\|=1} \boldsymbol{x}^T\boldsymbol{v} \geq \boldsymbol{x}^T\frac{\boldsymbol{x}}{\|\boldsymbol{x}\|} = \frac{\|\boldsymbol{x}\|_2^2}{\|\boldsymbol{x}\|} > 0$$

   as $\|\boldsymbol{x}\|_2 > 0$ and $\|\boldsymbol{x}\| > 0$. ⊠

For illustration, we provide two examples of dual norms. First, the dual norm of the Euclidean norm is presented, which is relevant in reducing steepest descent to gradient descent.

Example 1.4.2. *The Euclidean norm $\|\cdot\|_2$ has the dual norm $\|\cdot\|_{2*} = \|\cdot\|_2$.*

To see this define, for $\boldsymbol{w}, \boldsymbol{v} \in \mathbb{R}^n$, the inner product $\langle \boldsymbol{w}, \boldsymbol{v} \rangle = \boldsymbol{w}^T\boldsymbol{v}$ and note $\|\boldsymbol{v}\|_2^2 = \langle \boldsymbol{v}, \boldsymbol{v} \rangle$. By the Cauchy-Schwartz inequality $|\langle \boldsymbol{w}, \boldsymbol{v} \rangle| \leq \|\boldsymbol{w}\|_2\,\|\boldsymbol{v}\|_2$ and equality holds if and only if $\boldsymbol{w}$ and

$\boldsymbol{v}$ are linearly dependent. Therefore

$$\|\boldsymbol{w}\|_{2*} = \sup_{\|\boldsymbol{v}\|_2=1} \boldsymbol{w}^T\boldsymbol{v} = \sup_{\|\boldsymbol{v}\|_2=1} |\boldsymbol{w}^T\boldsymbol{v}| = \sup_{\|\boldsymbol{v}\|_2=1} |\langle \boldsymbol{w}, \boldsymbol{v}\rangle| = \frac{\boldsymbol{w}^T\boldsymbol{w}}{\|\boldsymbol{w}\|_2} = \|\boldsymbol{w}\|_2$$

where in the second to last equality we use that $\boldsymbol{v}$ is only linearly dependent with $\boldsymbol{w}$ and satisfy $\|\boldsymbol{v}\|_2 = 1$ if $\boldsymbol{v} = \boldsymbol{w}/\|\boldsymbol{w}\|_2$.

Second, the dual norm for a quadratic norm is presented, which is relevant in reducing steepest descent to Newton's descent.

Example 1.4.3. *For a positive definite matrix $B \in \mathbb{R}^{n\times n}$ the quadratic norm defined as $\|\boldsymbol{w}\|_B = (\boldsymbol{w}^T B\boldsymbol{w})^{1/2}$ has the dual norm $\|\boldsymbol{w}\|_{B*} = \|\boldsymbol{w}\|_{B^{-1}}$.*

To show this, first note that there exists an invertible and symmetric matrix $B^{1/2}$ such that $B = B^{1/2}B^{1/2}$ since B is positive definite. Second, note $\|\boldsymbol{w}\|_B = \|B^{1/2}\boldsymbol{w}\|_2$ Then

$$\begin{aligned}
\|\boldsymbol{w}\|_{B*} = \sup_{\|\boldsymbol{v}\|_B=1} \boldsymbol{w}^T\boldsymbol{v} &= \sup_{(\boldsymbol{v}^T B\boldsymbol{v})^{1/2}=1} \boldsymbol{w}^T\boldsymbol{v} \\
&= \sup_{\|\boldsymbol{q}\|_2=1} \boldsymbol{w}^T \left(B^{1/2}\right)^{-1} \boldsymbol{q} \\
&= \sup_{\|\boldsymbol{q}\|_2=1} \left(\left(B^{1/2}\right)^{-1} \boldsymbol{w}\right)^T \boldsymbol{q} \\
&= \|\left(B^{1/2}\right)^{-1} \boldsymbol{w}\|_{2*} = \|\left(B^{1/2}\right)^{-1} \boldsymbol{w}\|_2 = \|\boldsymbol{w}\|_{B^{-1}},
\end{aligned}$$

where we have defined $\boldsymbol{q} = B^{1/2}\boldsymbol{v}$ (or equivalently $\boldsymbol{v} = \left(B^{1/2}\right)^{-1}\boldsymbol{q}$).

Next, we use the dual norm to define an update direction for the steepest descent method. Let $g : \mathbb{R}^n \to \mathbb{R}$ be a cost function and $\|\cdot\|$ be any norm on $\mathbb{R}^n$. The steepest descent direction is a

function $\boldsymbol{d} : \mathbb{R}^n \to \mathbb{R}^n$, induced by g and $\|\cdot\|$, given by

$$\boldsymbol{d}(\boldsymbol{x}) = \|\nabla g(\boldsymbol{x})\|_* \left(\underset{\|\boldsymbol{v}\|=1}{\arg\inf} \ \nabla g(\boldsymbol{x})^T \boldsymbol{v} \right). \tag{1.13}$$

Recall the linear approximation of g centered at $\boldsymbol{x}$ is $\widehat{g}(\boldsymbol{y}) = g(\boldsymbol{x}) + \nabla g(\boldsymbol{x})^T(\boldsymbol{y} - \boldsymbol{x})$. When considering $\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{v}$ the linear approximation reduces to $\widehat{g}(\boldsymbol{x} + \boldsymbol{v}) = g(\boldsymbol{x}) + \nabla g(\boldsymbol{x})^T \boldsymbol{v}$. Therefore, the steepest descent direction (1.13) may be viewed as a scaling of $\arg\inf_{\|\boldsymbol{v}\|=1} \widehat{g}(\boldsymbol{x}+\boldsymbol{v})$ the constrained minimization of the linear approximation of the cost function. That is, the steepest descent direction provides the largest cost function decrease of a unit step, unitized by $\|\cdot\|$, from the point $\boldsymbol{x}$ as measured by the linear approximation. We present the steepest descent directions for the Euclidean norm and the quadratic norm.

Example 1.4.4. *For a cost function $g : \mathbb{R}^n \to \mathbb{R}$, the Euclidean norm $\|\cdot\|_2$ and quadratic norm $\|\cdot\|_B$ induce the respective steepest descent directions*

$$\boldsymbol{d}_2(\boldsymbol{x}) = -\nabla g(\boldsymbol{x})$$
$$\boldsymbol{d}_B(\boldsymbol{x}) = -B^{-1}\nabla g(\boldsymbol{x}).$$

Using the Cauchy-Schwartz inequality we first observe

$$\begin{aligned}
\boldsymbol{d}_2(\boldsymbol{x}) = \|\nabla g(\boldsymbol{x})\|_{2*} \left(\underset{\|\boldsymbol{v}\|_2=1}{\arg\inf} \ \nabla g(\boldsymbol{x})^T \boldsymbol{v} \right) &= -\|\nabla g(\boldsymbol{x})\|_2 \left(\underset{\|\boldsymbol{v}\|_2=1}{\arg\sup} \ \nabla g(\boldsymbol{x})^T \boldsymbol{v} \right) \\
&= -\|\nabla g(\boldsymbol{x})\|_2 \frac{\nabla g(\boldsymbol{x})}{\|\nabla g(\boldsymbol{x})\|_2} \\
&= -\nabla g(\boldsymbol{x}).
\end{aligned}$$

For the quadratic norm, recall $\|\boldsymbol{w}\|_{B*} = \sup_{\|\boldsymbol{v}\|_B=1} \boldsymbol{w}^T\boldsymbol{v} = \|\boldsymbol{w}\|_{B^{-1}} = (\boldsymbol{w}^T B^{-1}\boldsymbol{w})^{1/2}$ implying

$$\underset{\|\boldsymbol{v}\|_B=1}{\arg\sup} \ \boldsymbol{w}^T\boldsymbol{v} = (\boldsymbol{w}^T B^{-1}\boldsymbol{w})^{-1/2} B^{-1}\boldsymbol{w} = B^{-1}\boldsymbol{w}/\|\boldsymbol{w}\|_{B^{-1}}.$$

Therefore

$$\begin{aligned}\boldsymbol{d}_B(\boldsymbol{x}) &= \|\nabla g(\boldsymbol{x})\|_{B*}\left(\underset{\|\boldsymbol{v}\|_B=1}{\arg\inf}\ \nabla g(\boldsymbol{x})^T\boldsymbol{v}\right)\\ &= -\|\nabla g(\boldsymbol{x})\|_{B^{-1}}\left(\underset{\|\boldsymbol{v}\|_B=1}{\arg\sup}\ \nabla g(\boldsymbol{x})^T\boldsymbol{v}\right) = -\|\nabla g(\boldsymbol{x})\|_{B^{-1}}\frac{B^{-1}\nabla g(\boldsymbol{x})}{\|\nabla g(\boldsymbol{x})\|_{B^{-1}}} = -B^{-1}\nabla g(\boldsymbol{x}).\end{aligned}$$

Finally, provided an initial point $\boldsymbol{x}^0 \in \mathbb{R}^n$, steepest descent generates a sequence $\{\boldsymbol{x}^j\}_{j=1}^{\infty}$ given by

$$\boldsymbol{x}^j = \boldsymbol{x}^{j-1} + \eta^{(j)}\boldsymbol{d}(\boldsymbol{x}^{j-1})$$

where $\eta^{(j)}$ is a step size, which can be determined by a backtracking line search as discussed in Section 1.4.1. For further generality, we could assume that each steepest descent direction is determined by a step-specific norm. That is, let $\|\cdot\|_j$ be a norm on $\mathbb{R}^n$ that defines a descent direction $\boldsymbol{d}^j$, which is implemented on steepest descent step j. Then the steepest descent sequence $\{\boldsymbol{x}^j\}_{j=1}^{\infty}$ is given by

$$\boldsymbol{x}^j = \boldsymbol{x}^{j-1} + \eta^{(j)}\boldsymbol{d}^j(\boldsymbol{x}^{j-1}).$$

Newton's descent for convex cost function with non-constant Hessian is a well-known example of steepest descent where a different norm defines the steepest descent direction for each descent step.

1.4.1 Backtracking Line Search

Given a starting point $\boldsymbol{x}$ and steepest descent direction $\boldsymbol{d}$, we desire to choose a step size η to obtain the largest decrease in the cost function, g, with an update of the form $\boldsymbol{x} + \eta\boldsymbol{d}(\boldsymbol{x})$. That is, we want

$$\eta = \arg\min_{t>0} g(\boldsymbol{x} + t\boldsymbol{d}(\boldsymbol{x})).$$

As this line search for the step size is often difficult to solve exactly, inexact iterative methods to determine the step size are instead used in practice. One such method is the backtracking, or Armijo, line search that is a simple and efficient method for approximating the optimal step size. Let $\tau \in (0, 1/2]$ and $\beta \in (0, 1)$. Then the backtracking line search chooses the step size to be $\eta = \beta^r$ where $r = r(\boldsymbol{x})$ is the minimum, non-negative integer such that

$$g(\boldsymbol{x} + \beta^r \boldsymbol{d}(\boldsymbol{x})) \leq g(\boldsymbol{x}) + \tau \beta^r \nabla g(\boldsymbol{x})^T \boldsymbol{d}(\boldsymbol{x}). \tag{1.14}$$

In practice, r is determined iteratively by incrementing r starting from 0 until (1.14) is first satisfied as given in Algorithm 1.

Algorithm 1 Backtracking Line Search

Input: Differentiable cost function g, descent direction $\boldsymbol{d}$, estimate $\boldsymbol{x}$, parameters $\tau \in (0, 1/2]$ and $\beta \in (0, 1)$

1: Set $\eta = 1$
2: **while** $g(\boldsymbol{x} + \eta \boldsymbol{d}(\boldsymbol{x})) > g(\boldsymbol{x}) + \tau \eta \nabla g(\boldsymbol{x})^T \boldsymbol{d}(\boldsymbol{x})$ **do**
3: $\quad \eta = \beta \eta$
4: **end while**

Output: η

1.4.2 Properties of Steepest Descent

Here we show a few important properties of a steepest descent direction $\boldsymbol{d} : \mathbb{R}^n \to \mathbb{R}^n$. First, a simplification of the inner product of the steepest descent direction and the gradient.

Lemma 1.4.5 ([50])**.** *Let $g : \mathbb{R}^n \to \mathbb{R}$ be a differentiable function, $\|\cdot\|$ a norm on $\mathbb{R}^n$ with dual norm $\|\cdot\|_*$, and $\boldsymbol{d} : \mathbb{R}^n \to \mathbb{R}^n$ the induced steepest descent direction. Then*

$$\nabla g(\boldsymbol{x})^T \boldsymbol{d}(\boldsymbol{x}) = -\|\nabla g(\boldsymbol{x})\|_*^2.$$

Proof. Recall $\|\boldsymbol{w}\|_* = \max_{\|\boldsymbol{v}\|=1} \boldsymbol{w}^T\boldsymbol{v}$. Now observe

$$\begin{aligned}
\nabla g(\boldsymbol{x})^T \boldsymbol{d}(\boldsymbol{x}) &= \nabla g(\boldsymbol{x})^T \left(\|\nabla g(\boldsymbol{x})\|_* \left(\arg\min_{\|\boldsymbol{v}\|=1} \nabla g(\boldsymbol{x})^T \boldsymbol{v} \right) \right) \\
&= \|\nabla g(\boldsymbol{x})\|_* \left(\nabla g(\boldsymbol{x})^T \left(\arg\min_{\|\boldsymbol{v}\|=1} \nabla g(\boldsymbol{x})^T \boldsymbol{v} \right) \right) \\
&= \|\nabla g(\boldsymbol{x})\|_* \left(\min_{\|\boldsymbol{v}\|=1} \nabla g(\boldsymbol{x})^T \boldsymbol{v} \right) \\
&= -\|\nabla g(\boldsymbol{x})\|_* \left(\max_{\|\boldsymbol{v}\|=1} \nabla g(\boldsymbol{x})^T \boldsymbol{v} \right) \\
&= -\|\nabla g(\boldsymbol{x})\|_*^2. \qquad \boxtimes
\end{aligned}$$

Second, we show a bound on the norm of the steepest descent direction.

Lemma 1.4.6 ([50])**.** *Let $g : \mathbb{R}^n \to \mathbb{R}$ be a differentiable function, $\|\cdot\|$ a norm on $\mathbb{R}^n$ with dual norm $\|\cdot\|_*$, and $\boldsymbol{d} : \mathbb{R}^n \to \mathbb{R}^n$ the induced steepest descent direction. Then*

$$\|\boldsymbol{d}(\boldsymbol{x})\| = \|\nabla g(x)\|_*.$$

Proof. Observe

$$\begin{aligned}
\|\boldsymbol{d}(\boldsymbol{x})\| &= \left\| \|\nabla g(\boldsymbol{x})\|_* \left(\arg\min_{\|\boldsymbol{v}\|=1} \nabla g(\boldsymbol{x})^T \boldsymbol{v} \right) \right\| \\
&= \|\nabla g(\boldsymbol{x})\|_* \left\| \left(\arg\min_{\|\boldsymbol{v}\|=1} \nabla g(\boldsymbol{x})^T \boldsymbol{v} \right) \right\| = \|\nabla g(\boldsymbol{x})\|_*. \qquad \boxtimes
\end{aligned}$$

1.5 Projected Gradient Descent

Projected gradient descent is a general method for numerically optimizing a differentiable cost function, f, over a closed convex region $\mathcal{X}$. The composition of a projection onto $\mathcal{X}$ and a gradient descent step on f forms the foundation of a projected gradient descent step. As such, we define the projection onto a set and show that this projection is unique for convex sets. First, we provide an optimality condition, from [51], defining stationary points of a differentiable cost function.

Definition 1.5.1 ([51]). *Let $\mathcal{X} \subseteq \mathbb{R}^n$ be convex and $f : \mathcal{X} \to \mathbb{R}$ be a differentiable cost function. Then $\boldsymbol{x}^* \in \mathcal{X}$ is **stationary** if and only if*

$$\langle \nabla f(\boldsymbol{x}^*), \boldsymbol{x} - \boldsymbol{x}^* \rangle \geq 0 \quad \text{for all } \boldsymbol{x} \in \mathcal{X}.$$

We remark that for convex functions, stationary points are global optimizers. Now, we define a projection, or proximity map, onto a closed set as given in [52].

Definition 1.5.2 ([52]). *Let $\mathcal{X} \subseteq \mathbb{R}^n$ be a closed and non-empty set. The **projection onto** $\mathcal{X}$ is denoted by $\mathcal{P}_{\mathcal{X}} : \mathbb{R}^n \to \mathcal{X}$ and defined as*

$$\mathcal{P}_{\mathcal{X}}(\boldsymbol{x}) = \underset{\boldsymbol{z} \in \mathcal{X}}{\arg\min} \, \|\boldsymbol{x} - \boldsymbol{z}\|_2. \tag{1.15}$$

Furthermore, from [52], $\mathcal{P}_{\mathcal{X}}(\boldsymbol{x})$ is a projection of $\boldsymbol{x}$ onto $\mathcal{X}$ if and only if

$$\langle \boldsymbol{x} - \mathcal{P}_{\mathcal{X}}(\boldsymbol{x}), \boldsymbol{v} - \mathcal{P}_{\mathcal{X}}(\boldsymbol{x}) \rangle \leq 0 \quad \text{for all } \boldsymbol{v} \in \mathcal{X}. \tag{1.16}$$

Note that (1.16) is a direct consequence of combining (1.15) with Definition 1.5.1. To see this, first note that minimizing $\|\boldsymbol{x}-\boldsymbol{z}\|_2$ with respect to $\boldsymbol{z}$ is equivalent to minimizing $g(\boldsymbol{z}) \coloneqq \frac{1}{2}\|\boldsymbol{x}-\boldsymbol{z}\|_2^2$. Second, let $\boldsymbol{z}^*$ be any minimizer of $g(\boldsymbol{z})$ then by Definition 1.5.1 for all $\boldsymbol{v} \in \mathcal{X}$ the minimizer $\boldsymbol{z}^*$ satisfies

$$0 \leq \langle \nabla g(\boldsymbol{z}^*), \boldsymbol{v} - \boldsymbol{z}^* \rangle = \langle \boldsymbol{z}^* - \boldsymbol{x}, \boldsymbol{v} - \boldsymbol{z}^* \rangle \qquad \Longrightarrow \qquad \langle \boldsymbol{x} - \boldsymbol{z}^*, \boldsymbol{v} - \boldsymbol{z}^* \rangle \leq 0.$$

Next, we show that the projection onto a convex set is unique; the result of which is found in [52].

Proposition 1.5.3 ([52]). *Let $\mathcal{X} \subseteq \mathbb{R}^n$ be a closed, non-empty, and convex set. Then $\mathcal{P}_{\mathcal{X}}$ is a function. That is, for any $\boldsymbol{x} \in \mathbb{R}^n$ there exists a unique projection $\mathcal{P}_{\mathcal{X}}(\boldsymbol{x})$.*

Proof. Note $\|\boldsymbol{x}-\boldsymbol{z}\|_2$ is continuous in $\boldsymbol{z}$ on $\mathbb{R}^n$ and bounded below by 0. Thus, as $\mathcal{X}$ is closed by the Extreme Value Theorem there exists a $\boldsymbol{z}^* \in \mathcal{X}$ minimizing $\|\boldsymbol{x} - \boldsymbol{z}\|_2$. Next, let $\boldsymbol{x} \in \mathbb{R}^n$ and assume

$\boldsymbol{x}_1^*, \boldsymbol{x}_2^* \in \mathcal{X}$ are two projections of $\boldsymbol{x}$ onto $\mathcal{X}$. Then by (1.16) it holds that $\langle \boldsymbol{x} - \boldsymbol{x}_1^*, \boldsymbol{x}_2^* - \boldsymbol{x}_1^* \rangle \leq 0$ and $\langle \boldsymbol{x} - \boldsymbol{x}_2^*, \boldsymbol{x}_1^* - \boldsymbol{x}_2^* \rangle \leq 0$. Therefore

$$\begin{aligned} 0 &\geq \langle \boldsymbol{x} - \boldsymbol{x}_1^*, \boldsymbol{x}_2^* - \boldsymbol{x}_1^* \rangle + \langle \boldsymbol{x} - \boldsymbol{x}_2^*, \boldsymbol{x}_1^* - \boldsymbol{x}_2^* \rangle \\ &= \langle \boldsymbol{x} - \boldsymbol{x}_1^*, \boldsymbol{x}_2^* - \boldsymbol{x}_1^* \rangle - \langle \boldsymbol{x} - \boldsymbol{x}_2^*, \boldsymbol{x}_2^* - \boldsymbol{x}_1^* \rangle = \langle \boldsymbol{x}_2^* - \boldsymbol{x}_1^*, \boldsymbol{x}_2^* - \boldsymbol{x}_1^* \rangle = \|\boldsymbol{x}_2^* - \boldsymbol{x}_1^*\|_2^2. \end{aligned}$$

Additionally, as $\|\boldsymbol{x}_2^* - \boldsymbol{x}_1^*\|_2^2 \geq 0$ then $\|\boldsymbol{x}_2^* - \boldsymbol{x}_1^*\|_2^2 = 0$ implying $\boldsymbol{x}_1^* = \boldsymbol{x}_2^*$. ☒

Finally, as in [51], for a non-empty, closed, convex set $\mathcal{X}$, a differentiable function $f : \mathcal{X} \to \mathbb{R}$, and initial point $\boldsymbol{x}_0 \in \mathcal{X}$ the projected gradient descent generates the sequence $\{\boldsymbol{x}_k\}_{k=0}^{\infty}$ defined by

$$\boldsymbol{x}_k = \mathcal{P}_{\mathcal{X}}(\boldsymbol{x}_{k-1} - \eta_k \nabla f(\boldsymbol{x}_{k-1})),$$

where $\eta_k > 0$ a step size and $\mathcal{P}_{\mathcal{X}}$ a the projection onto $\mathcal{X}$. When $\mathcal{X} = \mathbb{R}^n$, then projected gradient descent simply reduces to standard gradient descent.

1.5.1 Backtracking Line Search

Similar to Section 1.4.1, a backtracking linesearch to determine η_k in PGD employs two user-chosen parameters $\alpha \in (0, 1/2]$ and $\beta \in (0, 1)$ and sets $\eta_k \leq 1$ as the maximum multiple of β such that

$$f(\boldsymbol{x}_{k+1}) \leq f(\boldsymbol{x}_k) - \alpha \langle \nabla f(\boldsymbol{x}_k), \boldsymbol{x}_k - \boldsymbol{x}_{k+1} \rangle. \tag{1.17}$$

1.5.2 Properties of Projected Gradient Descent

In this section we provide two properties for projected gradient descent, which can be found in [51]. First, we show the following lemma bounding the cost function change for a projected gradient descent step when the cost function is twice continuously differentiable and has Lipschitz continuous gradient.

Lemma 1.5.4 ([51]). *Let $\mathcal{X} \subseteq \mathbb{R}^n$ be a non-empty, closed, convex set. Assume $f : \mathcal{X} \to \mathbb{R}$ is twice continuously differentiable with L-Lipschitz continuous gradient on a compact subset $\mathcal{S} \subseteq \mathcal{X}$. Then for $\boldsymbol{x}_0 \in \mathcal{S}$, the sequence $\{\boldsymbol{x}_k\}_{k=0}^{\infty}$ determined by projected gradient descent with fixed step size η, for $0 < \eta < 2/L$, or step size determined by a backtracking linesearch, satisfies*

$$f(\boldsymbol{x}_k) - f(\boldsymbol{x}_{k+1}) \geq c\|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\|_2^2 \tag{1.18}$$

for positive constant c.

Proof. In (1.16) take $\boldsymbol{x} = \boldsymbol{x}_k - \eta_{k+1}\nabla f(\boldsymbol{x}_k)$ and $\boldsymbol{v} = \boldsymbol{x}_k$, since $\mathcal{P}_{\mathcal{X}}(\boldsymbol{x}_k - \eta_{k+1}\nabla f(\boldsymbol{x}_k)) \equiv \boldsymbol{x}_{k+1}$ we have

$$\begin{aligned}
0 &\geq \langle \boldsymbol{x}_k - \eta_{k+1}\nabla f(\boldsymbol{x}_k) - \mathcal{P}_{\mathcal{X}}(\boldsymbol{x}_k - \eta_{k+1}\nabla f(\boldsymbol{x}_k)), \boldsymbol{x}_k - \mathcal{P}_{\mathcal{X}}(\boldsymbol{x}_k - \eta_{k+1}\nabla f(\boldsymbol{x}_k))\rangle \\
&= \langle \boldsymbol{x}_k - \eta_{k+1}\nabla f(\boldsymbol{x}_k) - \boldsymbol{x}_{k+1}, \boldsymbol{x}_k - \boldsymbol{x}_{k+1}\rangle \\
&= \|\boldsymbol{x}_k - \boldsymbol{x}_{k+1}\|_2^2 - \eta_{k+1}\langle \nabla f(\boldsymbol{x}_k), \boldsymbol{x}_k - \boldsymbol{x}_{k+1}\rangle.
\end{aligned}$$

Implying

$$\langle \nabla f(\boldsymbol{x}_k), \boldsymbol{x}_k - \boldsymbol{x}_{k+1}\rangle \geq \frac{1}{\eta_{k+1}}\|\boldsymbol{x}_k - \boldsymbol{x}_{k+1}\|_2^2 \geq 0$$

and

$$-\langle \nabla f(\boldsymbol{x}_k), \boldsymbol{x}_k - \boldsymbol{x}_{k+1}\rangle \leq -\frac{1}{\eta_{k+1}}\|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\|_2^2. \tag{1.19}$$

Let $H_f(\boldsymbol{x})$ denote the Hessian of f evaluated at $\boldsymbol{x}$. Using Taylor's Theorem [53] to a second-order remainder observe

$$f(\boldsymbol{x}_{k+1}) = f(\boldsymbol{x}_k) + \langle \nabla f(\boldsymbol{x}_k), \boldsymbol{x}_{k+1} - \boldsymbol{x}_k\rangle + \frac{1}{2}(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k)^T H_f(\boldsymbol{\xi})(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k)$$

where $\boldsymbol{\xi} \in \mathcal{X}$ is on the line segment between $\boldsymbol{x}_k$ and $\boldsymbol{x}_{k+1}$. That is, $\boldsymbol{\xi} = c\boldsymbol{x}_{k+1} + (1-c)\boldsymbol{x}_k \in \mathcal{X}$ for some $c \in (0,1)$. As ∇f is L-Lipschitz then the Hessian of f is bounded by L. That is, $\boldsymbol{z}^T H_f(\boldsymbol{x})\boldsymbol{z} \leq L\|\boldsymbol{z}\|_2^2$ for all $\boldsymbol{x}, \boldsymbol{z} \in \mathcal{X}$. Hence

$$f(\boldsymbol{x}_{k+1}) \leq f(\boldsymbol{x}_k) + \langle \nabla f(\boldsymbol{x}_k), \boldsymbol{x}_{k+1} - \boldsymbol{x}_k \rangle + \frac{L}{2}\|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\|_2^2. \tag{1.20}$$

Combining (1.19) and (1.20) produces

$$\begin{aligned} f(\boldsymbol{x}_{k+1}) &\leq f(\boldsymbol{x}_k) + \langle \nabla f(\boldsymbol{x}_k), \boldsymbol{x}_{k+1} - \boldsymbol{x}_k \rangle + \frac{L}{2}\|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\|_2^2 \\ &\leq f(\boldsymbol{x}_k) - \frac{1}{\eta_{k+1}}\|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\|_2^2 + \frac{L}{2}\|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\|_2^2 \\ &= f(\boldsymbol{x}_k) - \left(\frac{1}{\eta_{k+1}} - \frac{L}{2}\right)\|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\|_2^2. \end{aligned}$$

where the second inequality results from using (1.19).

Thus

$$f(\boldsymbol{x}_k) - f(\boldsymbol{x}_{k+1}) \geq \left(\frac{1}{\eta_{k+1}} - \frac{L}{2}\right)\|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\|_2^2. \tag{1.21}$$

For fixed step sizes $\eta_{k+1} = \eta$ where $0 < \eta < 2/L$ we have (1.21) produces (1.18) with $c = \frac{1}{\eta} - \frac{L}{2}$.

Next, combining (1.19) and (1.20) again

$$\begin{aligned} f(\boldsymbol{x}_{k+1}) &\leq f(\boldsymbol{x}_k) + \langle \nabla f(\boldsymbol{x}_k), \boldsymbol{x}_{k+1} - \boldsymbol{x}_k \rangle + \frac{L}{2}\|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\|_2^2 \\ &\leq f(\boldsymbol{x}_k) + \langle \nabla f(\boldsymbol{x}_k), \boldsymbol{x}_{k+1} - \boldsymbol{x}_k \rangle + \frac{L}{2}\eta_{k+1}\langle \nabla f(\boldsymbol{x}_k), \boldsymbol{x}_k - \boldsymbol{x}_{k+1} \rangle \\ f(\boldsymbol{x}_{k+1}) &\leq f(\boldsymbol{x}_k) - \left(1 - \frac{L}{2}\eta_{k+1}\right)\langle \nabla f(\boldsymbol{x}_k), \boldsymbol{x}_k - \boldsymbol{x}_{k+1} \rangle . \end{aligned} \tag{1.22}$$

Since $\langle \nabla f(\boldsymbol{x}_k), \boldsymbol{x}_k - \boldsymbol{x}_{k+1} \rangle \geq 0$ then for $\eta_{k+1} \leq 1/L$ equation (1.22) implies

$$\begin{aligned} f(\boldsymbol{x}_{k+1}) &\leq f(\boldsymbol{x}_k) - \left(1 - \frac{L}{2}\eta_{k+1}\right) \langle \nabla f(\boldsymbol{x}_k), \boldsymbol{x}_k - \boldsymbol{x}_{k+1} \rangle \\ &\leq f(\boldsymbol{x}_k) - \frac{1}{2} \langle \nabla f(\boldsymbol{x}_k), \boldsymbol{x}_k - \boldsymbol{x}_{k+1} \rangle \\ &\leq f(\boldsymbol{x}_k) - \alpha \langle \nabla f(\boldsymbol{x}_k), \boldsymbol{x}_k - \boldsymbol{x}_{k+1} \rangle \end{aligned}$$

for $\alpha \in (0, 1/2]$. That is, (1.22) implies the backtracking linesearch requirement (1.17) when $\eta_{k+1} \leq 1/L$ and thus the backtracking linesearch terminates when $\eta_{k+1} \leq 1/L$. Combining (1.17) and (1.19) with the fact that $\eta_{k+1} \leq 1$ produces (1.18) with $c = \alpha$. ⊠

Finally, we show that a point $\boldsymbol{x}^*$ of projected gradient descent is fixed, i.e. $\boldsymbol{x}^* = \mathcal{P}(\boldsymbol{x}^* - \eta \nabla f(\boldsymbol{x}^*))$, if and only if it is stationary.

Lemma 1.5.5 ([51])**.** *A point is a fixed point of projected gradient descent if and only if it is stationary.*

Proof. Let $\boldsymbol{x}^*$ be a fixed point. That is, for a step size $\eta > 0$ it holds $\boldsymbol{x}^* = \mathcal{P}(\boldsymbol{x}^* - \eta \nabla f(\boldsymbol{x}^*))$. Using (1.16) with $\boldsymbol{x} = \boldsymbol{x}^* - \eta \nabla f(\boldsymbol{x}^*)$ shows for all $\boldsymbol{v} \in \mathcal{X}$

$$\begin{aligned} 0 &\geq \langle \boldsymbol{x} - \mathcal{P}_{\mathcal{X}}(\boldsymbol{x}), \boldsymbol{v} - \mathcal{P}_{\mathcal{X}}(\boldsymbol{x}) \rangle \\ &= \langle \boldsymbol{x}^* - \eta \nabla f(\boldsymbol{x}^*) - \mathcal{P}(\boldsymbol{x}^* - \eta \nabla f(\boldsymbol{x}^*)), \boldsymbol{v} - \mathcal{P}(\boldsymbol{x}^* - \eta \nabla f(\boldsymbol{x}^*)) \rangle \\ &= \langle \boldsymbol{x}^* - \eta \nabla f(\boldsymbol{x}^*) - \boldsymbol{x}^*, \boldsymbol{v} - \boldsymbol{x}^* \rangle \\ &= \langle -\eta \nabla f(\boldsymbol{x}^*), \boldsymbol{v} - \boldsymbol{x}^* \rangle . \end{aligned}$$

Therefore, $\langle \nabla f(\boldsymbol{x}^*), \boldsymbol{v} - \boldsymbol{x}^* \rangle \geq 0$ for all $\boldsymbol{v} \in \mathcal{X}$ and $\boldsymbol{x}^*$ is a stationary point.

Next, let $\boldsymbol{x}^* \in \mathcal{X}$ be a stationary point. Then for all $\boldsymbol{v} \in \mathcal{X}$

$$\begin{aligned} \langle \nabla f(\boldsymbol{x}^*), \boldsymbol{v} - \boldsymbol{x}^* \rangle &\geq 0 \\ \langle -\eta \nabla f(\boldsymbol{x}^*), \boldsymbol{v} - \boldsymbol{x}^* \rangle &\leq 0 \\ \langle \boldsymbol{x}^* - \eta \nabla f(\boldsymbol{x}^*) - \boldsymbol{x}^*, \boldsymbol{v} - \boldsymbol{x}^* \rangle &\leq 0. \end{aligned} \tag{1.23}$$

By (1.16) equation (1.23) implies $\boldsymbol{x}^*$ is the projection of $\boldsymbol{x}^* - \eta \nabla f(\boldsymbol{x}^*)$ onto $\mathcal{X}$. That is, $\boldsymbol{x}^* = \mathcal{P}_{\mathcal{X}}(\boldsymbol{x}^* - \eta \nabla f(\boldsymbol{x}^*))$ and $\boldsymbol{x}^*$ is a fixed point of projected gradient descent. ⊠

1.6 Iterative Shrinkage and Thresholding Algorithm

Projected gradient descent and steepest descent, while being generic optimization methods, are restricted to differentiable functions. Instead, the iterative shrinkage and thresholding algorithm (ISTA), otherwise known as proximal gradient descent, is an optimization method for convex and non-differentiable cost functions that can be written as a sum of a differentiable and a non-differentiable component. At its core, a step of ISTA involves a composition of a proximal operator and a gradient descent step. Accordingly, we define a proximal operator as given in [2].

Definition 1.6.1 ([2])**.** *Let $\mathcal{X}$ be a Hilbert space with norm $\|\cdot\|_{\mathcal{X}}$. For a lower semi-continuous convex function f the **proximal operator** of f is defined as*

$$\text{prox}_f(\boldsymbol{v}) \coloneqq \underset{\boldsymbol{x} \in \mathcal{X}}{\arg\min} \frac{1}{2} \|\boldsymbol{x} - \boldsymbol{v}\|_{\mathcal{X}}^2 + f(\boldsymbol{x}).$$

We remark that for a lower semi-continuous convex function f the function $\frac{1}{2}\|x - v\|_{\mathcal{X}}^2 + f(\boldsymbol{x})$ is convex and thus has unique minimizers. Therefore, the proximal operator is well-defined for lower semi-continuous convex functions. Additionally, the proximal operator is an optimization tool for non-smooth, convex functions as fixed points of $\text{prox}_f(\boldsymbol{v})$ minimize f. In considering the sum of a convex, differentiable function and a convex, non-smooth function, the proximal operator is implemented to optimize the non-smooth function.

Finally, we define the ISTA optimization method as in [2]. Let $\mathcal{X} \subseteq \mathbb{R}^n$, $f : \mathcal{X} \to \mathbb{R}$ be a convex and differentiable function, and $h : \mathcal{X} \to \mathbb{R}$ be convex and possibly non-smooth function. We consider optimizing the function $r \coloneqq f + h$. Thus, for $\boldsymbol{y} \in \mathcal{X}$ let $\widehat{f}_\eta(\boldsymbol{x}, \boldsymbol{y}) \coloneqq f(\boldsymbol{y}) + \langle \nabla f(\boldsymbol{y}), \boldsymbol{x} - \boldsymbol{y} \rangle + \frac{1}{2\eta}\|\boldsymbol{x} - \boldsymbol{y}\|_2^2$ be a quadratic approximation of f centered about $\boldsymbol{y} \in \mathcal{X}$. Now define $Q_\eta(\boldsymbol{x}, \boldsymbol{y}) \coloneqq \widehat{f}_\eta(\boldsymbol{x}, \boldsymbol{y}) + h(\boldsymbol{x})$. Next, define

$$\boldsymbol{p}_\eta(\boldsymbol{y}) \coloneqq \arg\min_{\boldsymbol{x} \in \mathcal{X}} Q_\eta(\boldsymbol{x}, \boldsymbol{y})$$

and observe

$$\begin{aligned}
\boldsymbol{p}_\eta(\boldsymbol{y}) &= \arg\min_{\boldsymbol{x} \in \mathcal{X}} f(\boldsymbol{y}) + \langle \nabla f(\boldsymbol{y}), \boldsymbol{x} - \boldsymbol{y} \rangle + \frac{1}{2\eta}\|\boldsymbol{x} - \boldsymbol{y}\|_2^2 + h(\boldsymbol{x}) \\
&= \arg\min_{\boldsymbol{x} \in \mathcal{X}} \frac{2\eta}{2\eta}\langle \nabla f(\boldsymbol{y}), \boldsymbol{x} - \boldsymbol{y} \rangle + \frac{1}{2\eta}\|\boldsymbol{x} - \boldsymbol{y}\|_2^2 + h(\boldsymbol{x}) \\
&= \arg\min_{\boldsymbol{x} \in \mathcal{X}} \frac{1}{2\eta}\langle \eta\nabla f(\boldsymbol{y}), \eta\nabla f(\boldsymbol{y}) \rangle + \frac{2}{2\eta}\langle \eta\nabla f(\boldsymbol{y}), \boldsymbol{x} - \boldsymbol{y} \rangle + \frac{1}{2\eta}\langle \boldsymbol{x} - \boldsymbol{y}, \boldsymbol{x} - \boldsymbol{y} \rangle + h(\boldsymbol{x}) \\
&= \arg\min_{\boldsymbol{x} \in \mathcal{X}} \frac{1}{2\eta}\langle \eta\nabla f(\boldsymbol{y}), \eta\nabla f(\boldsymbol{y}) + \boldsymbol{x} - \boldsymbol{y} \rangle + \frac{1}{2\eta}\langle \eta\nabla f(\boldsymbol{y}) + \boldsymbol{x} - \boldsymbol{y}, \boldsymbol{x} - \boldsymbol{y} \rangle + h(\boldsymbol{x}) \\
&= \arg\min_{\boldsymbol{x} \in \mathcal{X}} \frac{1}{2\eta}\langle \eta\nabla f(\boldsymbol{y}) + \boldsymbol{x} - \boldsymbol{y}, \eta\nabla f(\boldsymbol{y}) + \boldsymbol{x} - \boldsymbol{y} \rangle + h(\boldsymbol{x}) \\
&= \arg\min_{\boldsymbol{x} \in \mathcal{X}} \frac{1}{2\eta}\|\eta\nabla f(\boldsymbol{y}) + \boldsymbol{x} - \boldsymbol{y}\|_2^2 + h(\boldsymbol{x}) \\
&= \arg\min_{\boldsymbol{x} \in \mathcal{X}} \frac{1}{2}\left\|\boldsymbol{x} - (\boldsymbol{y} - \eta\nabla f(\boldsymbol{y}))\right\|_2^2 + \eta h(\boldsymbol{x}) \\
&= \text{prox}_{\eta h}\left(\boldsymbol{y} - \eta\nabla f(\boldsymbol{y})\right). \qquad (1.24)
\end{aligned}$$

Therefore, given an initial point $\boldsymbol{x}_0 \in \mathcal{X}$, the ISTA sequence $\{\boldsymbol{x}_k\}_{k=0}^\infty$ on $r(\boldsymbol{x})$ is defined as

$$\begin{aligned}
\boldsymbol{x}_k = \boldsymbol{p}_{\eta_k}(\boldsymbol{x}_{k-1}) &= \text{prox}_{\eta_k h}(\boldsymbol{x}_{k-1} - \eta_k \nabla f(\boldsymbol{x}_{k-1})) \\
&= \arg\min_{\boldsymbol{x} \in \mathcal{X}} \frac{1}{2}\|\boldsymbol{x} - (\boldsymbol{x}_{k-1} - \eta_k \nabla f(\boldsymbol{x}_{k-1}))\|_2^2 + \eta_k h(\boldsymbol{x})
\end{aligned}$$

where $\eta_k > 0$ is a step size and $\text{prox}_h(\boldsymbol{v})$ is the proximal operator of h.

1.6.1 Backtracking Line Search

Due to possible non-differentiability of h, a backtracking linesearch to determine η_k in ISTA differs slightly from Section 1.4.1. For a single user-chosen parameter $\beta \in (0, 1)$ the ISTA backtracking linesearch sets $\eta_k \leq 1$ as the maximum multiple of β such that

$$f(\boldsymbol{x}_{k+1}) \leq f(\boldsymbol{x}_k) - \langle \nabla f(\boldsymbol{x}_k), \boldsymbol{x}_k - \boldsymbol{x}_{k+1} \rangle + \frac{1}{2\eta_k} \|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\|_2^2. \tag{1.25}$$

1.6.2 Properties of Iterative Shrinkage and Thresholding Algorithm

In this section we provide two properties of ISTA, which can be found in [2]. First, we show a bound on the cost function change for an ISTA step. For this we require the definition of a subdifferential, as given in [54], and a couple of lemmas.

Definition 1.6.2 ([54])**.** *Let $\mathcal{X} \subseteq \mathbb{R}^n$ and $f : \mathcal{X} \to \mathbb{R}$ be a continuous and possible non-smooth function. A **subgradient** of f at $\boldsymbol{x} \in \mathcal{X}$ is any point $\boldsymbol{g} \in \mathcal{X}$ satisfying*

$$f(\boldsymbol{y}) - f(\boldsymbol{x}) \geq \langle \boldsymbol{g}, \boldsymbol{y} - \boldsymbol{x} \rangle$$

*for all $\boldsymbol{y} \in \mathcal{X}$. The **subdifferential** of f at $\boldsymbol{x}$, denoted $\partial f(\boldsymbol{x})$ is the set of all subgradients given by*

$$\partial f(\boldsymbol{x}) \coloneqq \{\boldsymbol{g} \in \mathcal{X} : \boldsymbol{g} \text{ is a subgradient of } f \text{ at } \boldsymbol{x}\}.$$

We remark that for a convex function f the subdifferential $\partial f(\boldsymbol{x})$ is a non-empty, closed, and convex set for all $\boldsymbol{x}$. Furthermore, for differentiable f it holds that $\partial f(\boldsymbol{x}) = \{\nabla f(\boldsymbol{x})\}$. Hence, subdifferentials generalize the notion of a gradient to convex functions that are possibly non-smooth.

Next, we provide an optimality condition, from [54], for convex functions using the subdifferential.

Lemma 1.6.3 ([54]). *Let $\mathcal{X} \subseteq \mathbb{R}^n$ and $f : \mathcal{X} \to \mathbb{R}$ be a continuous and possible non-smooth function. A point $\boldsymbol{x}^*$ minimizes f if and only if $\mathbf{0} \in \partial f(\boldsymbol{x}^*)$.*

Proof. Let $\boldsymbol{x}^* \in \mathcal{X}$ and assume $\mathbf{0} \in \partial f(\boldsymbol{x}^*)$. Then by Definition 1.6.2 $f(\boldsymbol{y}) - f(\boldsymbol{x}^*) \geq \langle \mathbf{0}, \boldsymbol{y} - \boldsymbol{x}^* \rangle = 0$ for all $\boldsymbol{y} \in \mathcal{X}$. Implying $f(\boldsymbol{y}) \geq f(\boldsymbol{x}^*)$ for all $\boldsymbol{y} \in \mathcal{X}$ or equivalently $\boldsymbol{x}^*$ minimizes f.

Next, let $\boldsymbol{x}^* \in \mathcal{X}$ minimize f. Then $f(\boldsymbol{y}) \geq f(\boldsymbol{x}^*)$ for all $\boldsymbol{y} \in \mathcal{X}$. Implying $f(\boldsymbol{y}) - f(\boldsymbol{x}^*) \geq 0 = \langle \mathbf{0}, \boldsymbol{y} - \boldsymbol{x}^* \rangle$ for all $\boldsymbol{y} \in \mathcal{X}$ or equivalently, by Definition 1.6.2, $\mathbf{0} \in \partial f(\boldsymbol{x}^*)$. ⊠

Now, we provide the following lemma, from [2], which is useful for bounding the change in the cost function during an ISTA step.

Lemma 1.6.4 ([2]). *Let $f : \mathcal{X} \to \mathbb{R}$ be a convex and differentiable function and $h : \mathcal{X} \to \mathbb{R}$ be a convex and possibly non-smooth function. Define $r = f + g$ and $\boldsymbol{p}_\eta(\boldsymbol{v}) = \text{prox}_{\eta h}(\boldsymbol{v} - \eta \nabla f(\boldsymbol{v}))$. For any $\boldsymbol{v} \in \mathbb{R}^n$ and $\eta > 0$ satisfying*

$$f(\boldsymbol{p}_\eta(\boldsymbol{v})) \leq f(\boldsymbol{v}) + \langle \nabla f(\boldsymbol{v}), \boldsymbol{p}_\eta(\boldsymbol{v}) - \boldsymbol{v} \rangle + \frac{1}{2\eta} \|\boldsymbol{p}_\eta(\boldsymbol{v}) - \boldsymbol{v}\|_2^2 \tag{1.26}$$

it holds that

$$r(\boldsymbol{x}) - r(\boldsymbol{p}_\eta(\boldsymbol{v})) \geq \frac{1}{2\eta} \|\boldsymbol{p}_\eta(\boldsymbol{v}) - \boldsymbol{v}\|_2^2 + \frac{1}{\eta} \langle \boldsymbol{v} - \boldsymbol{x}, \boldsymbol{p}_\eta(\boldsymbol{v}) - \boldsymbol{v} \rangle \tag{1.27}$$

for any $\boldsymbol{x} \in \mathbb{R}^n$.

Proof. Define $w(\boldsymbol{x}) = \frac{1}{2} \|\boldsymbol{x} - (\boldsymbol{v} - \eta \nabla f(\boldsymbol{v}))\|_2^2 + \eta h(\boldsymbol{x})$. First note that $\partial w(\boldsymbol{x}) = \boldsymbol{x} - \boldsymbol{v} + \eta \nabla f(\boldsymbol{v}) + \eta \partial h(\boldsymbol{x})$. Thus, by Lemma 1.6.3 and (1.24), $\boldsymbol{z} = \boldsymbol{p}_\eta(\boldsymbol{v})$ if and only if $\mathbf{0} \in \partial w(\boldsymbol{z})$. That is, there exists a $\boldsymbol{g}(\boldsymbol{v}) \in \partial h(\boldsymbol{z})$ such that

$$\boldsymbol{z} - \boldsymbol{v} + \eta \nabla f(\boldsymbol{y}) + \eta \boldsymbol{g}(\boldsymbol{v}) = \nabla f(\boldsymbol{y}) + \frac{1}{\eta}(\boldsymbol{z} - \boldsymbol{v}) + \boldsymbol{g}(\boldsymbol{v}) = \mathbf{0}. \tag{1.28}$$

Since f and h are convex then for all $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}$ there exists a $\boldsymbol{g}(\boldsymbol{y}) \in \partial h(\boldsymbol{y})$ such that

$$f(\boldsymbol{x}) \geq f(\boldsymbol{y}) + \langle \nabla f(\boldsymbol{y}), \boldsymbol{x} - \boldsymbol{y} \rangle \tag{1.29}$$

$$h(\boldsymbol{x}) \geq h(\boldsymbol{y}) + \langle \boldsymbol{g}(\boldsymbol{y}), \boldsymbol{x} - \boldsymbol{y} \rangle. \tag{1.30}$$

Furthermore, by Definition 1.6.2, equation (1.30) holds for every $\boldsymbol{g}(\boldsymbol{y}) \in \partial h(\boldsymbol{y})$. Therefore, taking $\boldsymbol{y} = \boldsymbol{p}_\eta(\boldsymbol{v})$ in (1.30) gives

$$h(\boldsymbol{x}) \geq h(\boldsymbol{p}_\eta(\boldsymbol{v})) + \langle \boldsymbol{g}(\boldsymbol{p}_\eta(\boldsymbol{v})), \boldsymbol{x} - \boldsymbol{p}_\eta(\boldsymbol{v}) \rangle \tag{1.31}$$

where $\boldsymbol{g}(\boldsymbol{p}_\eta(\boldsymbol{v}))$ satisfies (1.28). Now taking $\boldsymbol{y} = \boldsymbol{v}$ in (1.29) and then adding (1.29) and (1.31) together gives

$$f(\boldsymbol{x}) + h(\boldsymbol{x}) \geq f(\boldsymbol{v}) + \langle \nabla f(\boldsymbol{v}), \boldsymbol{x} - \boldsymbol{v} \rangle + h(\boldsymbol{p}_\eta(\boldsymbol{v})) + \langle \boldsymbol{g}(\boldsymbol{p}_\eta(\boldsymbol{v})), \boldsymbol{x} - \boldsymbol{p}_\eta(\boldsymbol{v}) \rangle. \tag{1.32}$$

From (1.26) it holds that

$$\begin{aligned}
-f(\boldsymbol{p}_\eta(\boldsymbol{v})) &\geq -f(\boldsymbol{v}) - \langle \nabla f(\boldsymbol{v}), \boldsymbol{p}_\eta(\boldsymbol{v}) - \boldsymbol{v} \rangle - \frac{1}{2\eta} \|\boldsymbol{p}_\eta(\boldsymbol{v}) - \boldsymbol{v}\|_2^2 \\
f(\boldsymbol{x}) + h(\boldsymbol{x}) - f(\boldsymbol{p}_\eta(\boldsymbol{v})) - h(\boldsymbol{p}_\eta(\boldsymbol{v})) &\geq f(\boldsymbol{x}) + h(\boldsymbol{x}) - f(\boldsymbol{v}) - \langle \nabla f(\boldsymbol{v}), \boldsymbol{p}_\eta(\boldsymbol{v}) - \boldsymbol{v} \rangle \\
&\quad - \frac{1}{2\eta} \|\boldsymbol{p}_\eta(\boldsymbol{v}) - \boldsymbol{v}\|_2^2 - h(\boldsymbol{p}_\eta(\boldsymbol{v})) \\
r(\boldsymbol{x}) - r(\boldsymbol{p}_\eta(\boldsymbol{v})) &\geq f(\boldsymbol{v}) + \langle \nabla f(\boldsymbol{v}), \boldsymbol{x} - \boldsymbol{v} \rangle + h(\boldsymbol{p}_\eta(\boldsymbol{v})) \\
&\quad + \langle \boldsymbol{g}(\boldsymbol{p}_\eta(\boldsymbol{v})), \boldsymbol{x} - \boldsymbol{p}_\eta(\boldsymbol{v}) \rangle - f(\boldsymbol{v}) \\
&\quad - \langle \nabla f(\boldsymbol{v}), \boldsymbol{p}_\eta(\boldsymbol{v}) - \boldsymbol{v} \rangle - \frac{1}{2\eta} \|\boldsymbol{p}_\eta(\boldsymbol{v}) - \boldsymbol{v}\|_2^2 - h(\boldsymbol{p}_\eta(\boldsymbol{v})) \\
&= \langle \nabla f(\boldsymbol{v}), \boldsymbol{x} - \boldsymbol{v} \rangle + \langle \boldsymbol{g}(\boldsymbol{p}_\eta(\boldsymbol{v})), \boldsymbol{x} - \boldsymbol{p}_\eta(\boldsymbol{v}) \rangle \\
&\quad - \langle \nabla f(\boldsymbol{v}), \boldsymbol{p}_\eta(\boldsymbol{v}) - \boldsymbol{v} \rangle - \frac{1}{2\eta} \|\boldsymbol{p}_\eta(\boldsymbol{v}) - \boldsymbol{v}\|_2^2
\end{aligned}$$

where the third inequality results from using (1.32). Simplifying this inequality further gives

$$\begin{aligned} r(\boldsymbol{x}) - r(\boldsymbol{p}_\eta(\boldsymbol{v})) &\geq \langle \nabla f(\boldsymbol{v}), \boldsymbol{x} - \boldsymbol{p}_\eta(\boldsymbol{v}) \rangle + \langle \boldsymbol{g}(\boldsymbol{p}_\eta(\boldsymbol{v})), \boldsymbol{x} - \boldsymbol{p}_\eta(\boldsymbol{v}) \rangle - \frac{1}{2\eta} \|\boldsymbol{p}_\eta(\boldsymbol{v}) - \boldsymbol{v}\|_2^2 \\ &= \langle \nabla f(\boldsymbol{v}) + \boldsymbol{g}(\boldsymbol{p}_\eta(\boldsymbol{v})), \boldsymbol{x} - \boldsymbol{p}_\eta(\boldsymbol{v}) \rangle - \frac{1}{2\eta} \|\boldsymbol{p}_\eta(\boldsymbol{v}) - \boldsymbol{v}\|_2^2 \\ &= \langle -\frac{1}{\eta}(\boldsymbol{p}_\eta(\boldsymbol{v}) - \boldsymbol{v}), \boldsymbol{x} - \boldsymbol{p}_\eta(\boldsymbol{v}) \rangle - \frac{1}{2\eta} \|\boldsymbol{p}_\eta(\boldsymbol{v}) - \boldsymbol{v}\|_2^2 \end{aligned}$$

where the final line results since $\boldsymbol{g}(\boldsymbol{p}_\eta(\boldsymbol{v}))$ satisfies (1.28) that is, $\boldsymbol{g}(\boldsymbol{p}_\eta(\boldsymbol{v})) = -\nabla f(\boldsymbol{y}) - \frac{1}{\eta}(\boldsymbol{p}_\eta(\boldsymbol{v}) - \boldsymbol{v})$. Simplifying this inequality again

$$\begin{aligned} r(\boldsymbol{x}) - r(\boldsymbol{p}_\eta(\boldsymbol{v})) &\geq \frac{1}{\eta} \langle \boldsymbol{v} - \boldsymbol{p}_\eta(\boldsymbol{v}), \boldsymbol{x} - \boldsymbol{p}_\eta(\boldsymbol{v}) \rangle - \frac{1}{2\eta} \langle \boldsymbol{p}_\eta(\boldsymbol{v}) - \boldsymbol{v}, \boldsymbol{p}_\eta(\boldsymbol{v}) - \boldsymbol{v} \rangle \\ &= \frac{1}{\eta} \langle \boldsymbol{v} - \boldsymbol{p}_\eta(\boldsymbol{v}), \boldsymbol{x} - \boldsymbol{p}_\eta(\boldsymbol{v}) \rangle + \frac{1}{\eta} \langle \boldsymbol{v} - \boldsymbol{p}_\eta(\boldsymbol{v}), \boldsymbol{p}_\eta(\boldsymbol{v}) - \boldsymbol{v} \rangle \\ &\quad + \frac{1}{2\eta} \langle \boldsymbol{p}_\eta(\boldsymbol{v}) - \boldsymbol{v}, \boldsymbol{p}_\eta(\boldsymbol{v}) - \boldsymbol{v} \rangle \\ &= \frac{1}{\eta} \langle \boldsymbol{v} - \boldsymbol{p}_\eta(\boldsymbol{v}), \boldsymbol{x} - \boldsymbol{v} \rangle + \frac{1}{2\eta} \langle \boldsymbol{p}_\eta(\boldsymbol{v}) - \boldsymbol{v}, \boldsymbol{p}_\eta(\boldsymbol{v}) - \boldsymbol{v} \rangle \\ &= \frac{1}{2\eta} \|\boldsymbol{p}_\eta(\boldsymbol{v}) - \boldsymbol{v}\|_2^2 + \frac{1}{\eta} \langle \boldsymbol{v} - \boldsymbol{x}, \boldsymbol{p}_\eta(\boldsymbol{v}) - \boldsymbol{v} \rangle. \end{aligned}$$

⊠

Now, we provide a bound on the change in the cost function over a single ISTA step; the result of which is found in [2].

Lemma 1.6.5 ([2])**.** *Let $f : \mathcal{X} \to \mathbb{R}$ be convex, differentiable, and have L-Lipschitz continuous gradient on compact subset $\mathcal{S} \subseteq \mathcal{X}$. Let $h : \mathcal{X} \to \mathbb{R}$ be convex and possibly non-smooth. Then the sequence $\{\boldsymbol{x}_k\}_{k=1}^{\infty}$ determined by ISTA on $r \coloneqq f + g$ with fixed step size η, for $0 < \eta \leq 1/L$, or step size determined by a backtracking linesearch, satisfies*

$$r(\boldsymbol{x}_k) - r(\boldsymbol{x}_{k+1}) \geq c \|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\|_2^2 \tag{1.33}$$

for positive constant c.

Proof. Let $\boldsymbol{p}_\eta(\boldsymbol{v}) = \text{prox}_{\eta h}(\boldsymbol{v} - \eta\nabla f(\boldsymbol{v}))$. As f has L-Lipschitz continuous gradient then (1.26) holds for any $0 < \eta \leq 1/L$. Thus, by Lemma 1.6.4, (1.27) holds for any $0 < \eta \leq 1/L$. In (1.27), let $\boldsymbol{x} = \boldsymbol{v} = \boldsymbol{x}_k$ and note $\boldsymbol{p}_\eta(\boldsymbol{v}) = \boldsymbol{x}_{k+1}$. Then for fixed step size $\eta_{k+1} = \eta$, (1.27) produces (1.33) with $c = \frac{1}{2\eta}$. When the step size is determined by a backtracking linesearch, we observe for $0 < \eta_{k+1} \leq 1/L$ that (1.26) holds and implies the backtracking linesearch requirement (1.25). Combining (1.27) with the fact that $\eta_{k+1} \leq 1$ produces (1.33) with $c = 1/2$. ⊠

Lastly, we show that the fixed points of ISTA are stationary. For this, recall the following optimality conditions, from [51], defining a stationary point of a convex function.

Definition 1.6.6 ([51])**.** *Let $f : \mathcal{X} \to \mathbb{R}$, for convex $\mathcal{X}$, be convex. Let $\partial f(\boldsymbol{x})$ denote the subdifferential of f at $\boldsymbol{x}$. Then $\boldsymbol{x}^* \in \mathcal{X}$ is **stationary** if and only if there exists a $\boldsymbol{d}^* \in \partial f(\boldsymbol{x}^*)$ such that*

$$\langle \boldsymbol{d}^*, \boldsymbol{x} - \boldsymbol{x}^* \rangle \geq 0 \quad \text{for all } \boldsymbol{x} \in \mathcal{X}.$$

We remark that for a convex cost function, f, Definition 1.6.6 implies that any internal stationary point $\boldsymbol{x}^*$ satisfies $\mathbf{0} \in \partial f(\boldsymbol{x}^*)$. Thus, by Lemma 1.6.3 any stationary point is a minimizer.

Finally, we show that fixed points of ISTA are equivalent to stationary points; the result of which appears in the existing literature [51].

Lemma 1.6.7 ([51])**.** *A point is a fixed point of ISTA if and only if it is stationary.*

Proof. Let $\boldsymbol{x}^*$ be a fixed point. That is, for a step size $\eta > 0$ it holds that $\boldsymbol{x}^* = \text{prox}_{\eta h}(\boldsymbol{x}^* - \eta\nabla f(\boldsymbol{x}^*))$. Define $w(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{x} - \boldsymbol{v}\|_2^2 + \eta h(\boldsymbol{x})$ and note

$$\partial w(\boldsymbol{x}) = \boldsymbol{x} - \boldsymbol{v} + \eta\partial h(\boldsymbol{x}).$$

Recall that $\text{prox}_{\eta h}(\boldsymbol{v}) = \underset{\boldsymbol{x}\in\mathcal{X}}{\arg\min}\, w(\boldsymbol{x})$. Let $\boldsymbol{v}^* \in \mathcal{X}$ satisfy $\text{prox}_{\eta h}(\boldsymbol{v}) = \boldsymbol{v}^*$. Then $\boldsymbol{v}^*$ is a minimizer of $w(\boldsymbol{x})$ and thus, since $w(\boldsymbol{x})$ is a convex function, $\boldsymbol{v}^*$ is a stationary point of w from

Definition 1.6.6. Furthermore, using the definition of $\partial w(\boldsymbol{x})$ and Definition 1.6.6, $\text{prox}_{\eta h}(\boldsymbol{v}) = \boldsymbol{v}^*$ if any only if there is a $\boldsymbol{d} \in \partial h(\boldsymbol{v}^*)$ such that for all $\boldsymbol{x} \in \mathcal{X}$

$$\langle \boldsymbol{v}^* - \boldsymbol{v} + \eta \boldsymbol{d}, \boldsymbol{x} - \boldsymbol{v}^* \rangle \geq 0. \tag{1.34}$$

As $\boldsymbol{x}^* = \text{prox}_{\eta h}(\boldsymbol{x}^* - \eta \nabla f(\boldsymbol{x}^*))$ then (1.34) implies that there exists a $\boldsymbol{d}^* \in \partial h(\boldsymbol{x}^*)$

$$\begin{aligned} 0 &\leq \langle \boldsymbol{x}^* - (\boldsymbol{x}^* - \eta \nabla f(\boldsymbol{x}^*)) + \eta \boldsymbol{d}^*, \boldsymbol{x} - \boldsymbol{x}^* \rangle \\ &= \langle \eta \nabla f(\boldsymbol{x}^*) + \eta \boldsymbol{d}^*, \boldsymbol{x} - \boldsymbol{x}^* \rangle \end{aligned}$$

for all $\boldsymbol{x} \in \mathcal{X}$. Hence $\langle \nabla f(\boldsymbol{x}^*) + \boldsymbol{d}^*, \boldsymbol{x} - \boldsymbol{x}^* \rangle \geq 0$ for all $\boldsymbol{x} \in \mathcal{X}$ and it holds that $\boldsymbol{x}^*$ is a stationary point of $r(\boldsymbol{x}) = f(\boldsymbol{x}) + h(\boldsymbol{x})$.

Next, let $\boldsymbol{x}^*$ be a stationary point of $r(\boldsymbol{x}) = f(\boldsymbol{x}) + h(\boldsymbol{x})$. Then there exists a $\boldsymbol{d}^* \in \partial h(\boldsymbol{x}^*)$ such that for all $\boldsymbol{x} \in \mathcal{X}$

$$\begin{aligned} 0 &\leq \langle \nabla f(\boldsymbol{x}^*) + \boldsymbol{d}^*, \boldsymbol{x} - \boldsymbol{x}^* \rangle \\ &= \langle \eta \nabla f(\boldsymbol{x}^*) + \eta \boldsymbol{d}^*, \boldsymbol{x} - \boldsymbol{x}^* \rangle \\ &= \langle \boldsymbol{x}^* - (\boldsymbol{x}^* - \eta \nabla f(\boldsymbol{x}^*)) + \eta \boldsymbol{d}^*, \boldsymbol{x} - \boldsymbol{x}^* \rangle \end{aligned}$$

where $\eta > 0$. Hence, by (1.34), $\boldsymbol{x}^* = \text{prox}_{\eta h}(\boldsymbol{x}^* - \eta \nabla f(\boldsymbol{x}^*))$ and therefore $\boldsymbol{x}^*$ is a fixed point. $\boxtimes$

Chapter 2

Compound Gaussian Informed Inverse Problems

As discussed in Section 1.2, the compound Gaussian (CG) class of distributions has been shown to better capture statistical properties of sparsity coefficients of natural images and images from other modalities such as radar [10–12]. Additionally, many recent works [13–20, 22–26, 30, 32, 33, 55, 56] have shown that deep learning-based approaches, in particular unrolled deep neural networks, can outperform model-based, iterative methods in signal reconstruction quality when solving inverse problems. Inspired by these results, we desire to incorporate the CG prior into a deep learning framework, which we do using the algorithm unrolling technique. First, we develop a CG prior informed iterative signal reconstruction algorithm named compound Gaussian least squares. Subsequently, we unroll the compound Gaussian least squares algorithm converting it into a CG prior informed DNN method named compound Gaussian network.

To start, we discuss the measurement equation we consider throughout this chapter. While many inverse problems are non-linear, we focus on linear inverse problems as this is frequently the assumed measurement model in many applications – including compressive sensing, computed tomography (CT), and magnetic resonance imaging (MRI) – and leave non-linear inverse problem applications to future work. Working from the linear measurement equation (1.6), let $\boldsymbol{x} \in \mathbb{R}^n$ be a vectorized signal that has a representation $\boldsymbol{x} = \Phi \boldsymbol{c}$ with respect to (w.r.t.) a dictionary, $\Phi \in \mathbb{R}^{n \times n}$, and coefficients, $\boldsymbol{c} \in \mathbb{R}^n$. An example, Φ is a wavelet transform and $\boldsymbol{c}$ are the wavelet coefficients of the signal $\boldsymbol{x}$. Consider linear measurements $\boldsymbol{y} \in \mathbb{R}^m$ given by $\boldsymbol{y} = \Psi\Phi\boldsymbol{c} + \boldsymbol{\nu}$ for additive white noise, $\boldsymbol{\nu} \in \mathbb{R}^m$. Now, we decompose $\boldsymbol{c}$ according to the CG prior in Section 1.2. That is, $\boldsymbol{c} = \boldsymbol{z} \odot \boldsymbol{u}$ for $\boldsymbol{u} \sim \mathcal{N}(\boldsymbol{0}, \Sigma_u)$, $\boldsymbol{z}$ and $\boldsymbol{u}$ are independent random vectors, and $\boldsymbol{z} = h(\boldsymbol{\chi})$ where $h : \mathbb{R} \to \mathbb{R}$ is a componentwise, positive, nonlinear function and $\boldsymbol{\chi}$ follows a multi-scale Gaussian tree process. Then the linear measurement model we consider is

$$\boldsymbol{y} = \Psi\Phi(\boldsymbol{z} \odot \boldsymbol{u}) + \boldsymbol{\nu} \tag{2.1}$$

and the linear inverse problem to (2.1) aims to recover $\boldsymbol{z}$ and $\boldsymbol{u}$ given $\boldsymbol{y}$, Ψ, and Φ.

The new contributions of this chapter are succinctly presented in our IEEE Asia Pacific Signal and Information Processing Association conference proceedings [57] and in our IEEE Transactions on Signal Processing journal article [58]. We provide additional explanation, details, and numerical results in this chapter on top of those found in our publications [57,58]. Specifically, in this chapter we have completed:

1. The construction of a novel iterative signal estimation algorithm, named compound Gaussian least squares (CG-LS), to solve the linear inverse problem to (2.1) with general Ψ and Φ matrices. A regularized least squares (RLS) optimization is the foundation for CG-LS where the regularization enforces a CG prior on the signal coefficients. Specifically, the regularization is the sum of two regularization terms; one regularization term is set to enforce normality of $\boldsymbol{u}$ and the second term is set to enforce normality of $\boldsymbol{\chi}$.

2. The development of a fundamental characterization for the existence and location of minimizers of the CG-LS cost function.

3. The derivation of a convergence analysis of CG-LS to stationary points of the cost function under a two-block coordinate descent with one block estimated via steepest descent.

4. A thorough experimental validation of CG-LS illustrating the effectiveness of CG-LS in linear image estimation. We found that in linear tomographic imaging and compressive sensing, CG-LS outperforms many prior state-of-the-art iterative image reconstruction methods.

5. The development of an original DNN through unrolling CG-LS. This new DNN, which we name compound Gaussian Network (CG-Net), is, to the best of our knowledge, the first DNN for general linear inverse problems to be fundamentally informed by a CG prior.

6. A thorough experimental validation of CG-Net that has demonstrated the efficacy of CG-Net to reconstruct images after training. We have shown that CG-Net outperforms other state-of-

the-art deep learning methods for image reconstruction in low measurement noise and low training scenarios.

2.1 Notation and Nomenclature

The set of symbols provided in this section are defined here for ease of referencing and in reading the remainder of this chapter. We note that some have already been defined previously.

$\mathbb{R}$	$=$	Set of real numbers.
$\boldsymbol{v}$	$=$	$[v_i]_{i=1,2,\ldots,n} \in \mathbb{R}^n$. Boldface characters are vectors.
M	$=$	$[M_{ij}]_{i=1,2,\ldots,n}^{j=1,2,\ldots,m} \in \mathbb{R}^{m\times n}$. Uppercase characters are matrices.
$(\cdot)^T$	$=$	Transpose of vector or matrix $(\cdot)$
$\odot$	$=$	Hadamard product.
$D(\boldsymbol{v})$	$=$	Diagonal matrix with entries $v_1, v_2, \ldots, v_n$ on the diagonal.
$M_{\boldsymbol{v}}$	$=$	$MD(\boldsymbol{v})$ for $M \in \mathbb{R}^{m\times n}$ and $\boldsymbol{v} \in \mathbb{R}^n$.
$g(\boldsymbol{v})$	$=$	$[g(v_i)]_{i=1,2,\ldots,n} \in \mathbb{R}^n$ for $\boldsymbol{v} \in \mathbb{R}^n$ and a **componentwise function** $g : \mathbb{R} \to \mathbb{R}$.
$\text{ReLU}(x)$	$=$	$\max\{0, x\}$ is a componentwise function.
$\mathcal{P}_{a,b}(x)$	$=$	$a+\text{ReLU}(x-a)-\text{ReLU}(x-b)$, for $a, b \in \mathbb{R}$, is a **modified ReLU** (mReLU) activation function. This componentwise function projects input x onto the interval $[\min\{a, b\}, \max\{a, b\}]$.
Ψ	$\in$	$\mathbb{R}^{m\times n}$ is a measurement or observation matrix.
Φ	$\in$	$\mathbb{R}^{n\times n}$ is a change-of-basis matrix or dictionary.
A	$=$	$\Psi\Phi$.

2.2 Compound Gaussian Least Squares (CG-LS)

2.2.1 CG-LS Implementation Details

As in the measurement equation (2.1), we decompose the image coefficient as $\boldsymbol{c} = \boldsymbol{z} \odot \boldsymbol{u}$ and let $h : \mathbb{R} \to \mathcal{Z} \subseteq [0, \infty)$ be the componentwise, invertible, and non-linear function as given by the CG prior. Define $f \coloneqq h^{-1}$ and define the regularization function $R : \mathbb{R}^n \times \mathcal{Z}^n \to \mathbb{R}$ as

$$R(\boldsymbol{u}, \boldsymbol{z}) = \lambda||\boldsymbol{u}||_2^2 + \mu||f(\boldsymbol{z})||_2^2$$

to enforce normality of $\boldsymbol{u}$ and $\boldsymbol{\chi} \coloneqq f(\boldsymbol{z})$, as desired from the formulation of the CG prior specified in equation (2.1), where $\mathcal{Z}$ is the domain of f. We consider the novel regularized least squares cost function, from (1.5), for CG-LS given by

$$F(\boldsymbol{u}, \boldsymbol{z}) = ||\boldsymbol{y} - A(\boldsymbol{z} \odot \boldsymbol{u})||_2^2 + \lambda||\boldsymbol{u}||_2^2 + \mu||f(\boldsymbol{z})||_2^2 \tag{2.2}$$

for positive real constants λ and μ. Then CG-LS, detailed in Algorithm 2, aims to obtain the estimator

$$[\boldsymbol{z}^*, \boldsymbol{u}^*] = \underset{\boldsymbol{z} \in \mathcal{Z}^n, \boldsymbol{u} \in \mathbb{R}^n}{\arg\min} \; F(\boldsymbol{u}, \boldsymbol{z}) \tag{2.3}$$

and subsequently estimate the signal coefficients as $\boldsymbol{c}^* = \boldsymbol{z}^* \odot \boldsymbol{u}^*$. We note that $R(\boldsymbol{c}) = R(\boldsymbol{u}, \boldsymbol{z})$ is not exactly proportional to $\log(p(\boldsymbol{c}))$ as specified in the MAP estimate from Section 1.1.1. Instead, the regularization, R, is an approximation capturing important statistical properties of the CG prior while also simplifying the optimization.

For notation we define a function $D : \mathbb{R}^n \to \mathbb{R}^{n \times n}$ as

$$D(\boldsymbol{w}) = D\left(\begin{bmatrix} w_1 & w_2 & \cdots & w_n \end{bmatrix}\right) = \text{diag}(\boldsymbol{w}) = \begin{bmatrix} w_1 & 0 & \cdots & 0 \\ 0 & w_2 & \cdots & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & \cdots & w_n \end{bmatrix}$$

and let $A_{\boldsymbol{w}} = AD(\boldsymbol{w})$.

Due to the explicit joint estimation in (2.3), we optimize by block coordinate descent [59] also known as the nonlinear Gauss-Seidel method. Thus, on iteration k, we desire the block variable estimates

$$\boldsymbol{z}_k = \underset{\boldsymbol{z} \in \mathcal{Z}^n}{\arg\min} \; \left|\left|\boldsymbol{y} - A_{\boldsymbol{u}_{k-1}}\boldsymbol{z}\right|\right|_2^2 + \mu||f(\boldsymbol{z})||_2^2 \tag{2.4}$$

$$\boldsymbol{u}_k = \underset{\boldsymbol{u} \in \mathbb{R}^n}{\arg\min} \; ||\boldsymbol{y} - A_{\boldsymbol{z}_k}\boldsymbol{u}||_2^2 + \lambda||\boldsymbol{u}||_2^2. \tag{2.5}$$

Block coordinate descent generates a monotonically decreasing sequence of cost function values, $\{F(\boldsymbol{u}_k, \boldsymbol{z}_k)\}_{k=0}^{\infty}$, that is bounded below by zero. Thus, it is guaranteed that the sequence of cost function values, $\{F(\boldsymbol{u}_k, \boldsymbol{z}_k)\}_{k=0}^{\infty}$, converges. Convergence rates of block coordinate descent under different requirements on the cost function, such as convexity, have previously been proven [60–62]. We remark that our approach differs from true block coordinate descent as (2.4) does not have an analytical solution for most choices of f and thus we will only be able to approximately, instead of exactly, solve (2.4).

Equation (2.5) is the well-known Tikhonov regularization problem with a solution

$$\boldsymbol{u}_k = (A_{\boldsymbol{z}_k}^T A_{\boldsymbol{z}_k} + \lambda I)^{-1} A_{\boldsymbol{z}_k}^T \boldsymbol{y}. \tag{2.6}$$

Note, in practice, we do not calculate the inverse directly, but instead solve the forward linear system of equations

$$(A_{\boldsymbol{z}_k}^T A_{\boldsymbol{z}_k} + \lambda I)\boldsymbol{u}_k = A_{\boldsymbol{z}_k}^T \boldsymbol{y}.$$

Additionally, using the Woodbury matrix identity [63], we rewrite the Tikhonov solution as

$$\begin{aligned}
\boldsymbol{u}_k &= (A_{\boldsymbol{z}_k}^T A_{\boldsymbol{z}_k} + \lambda I)^{-1} A_{\boldsymbol{z}_k}^T \boldsymbol{y} \\
&= \left[\lambda^{-1} I - \lambda^{-2} A_{\boldsymbol{z}_k}^T (I + \lambda^{-1} A_{\boldsymbol{z}_k} A_{\boldsymbol{z}_k}^T)^{-1} A_{\boldsymbol{z}_k}\right] A_{\boldsymbol{z}_k}^T \boldsymbol{y} \\
&= \lambda^{-1} \left[I - A_{\boldsymbol{z}_k}^T (\lambda I + A_{\boldsymbol{z}_k} A_{\boldsymbol{z}_k}^T)^{-1} A_{\boldsymbol{z}_k}\right] A_{\boldsymbol{z}_k}^T \boldsymbol{y} \\
&= \lambda^{-1} \left[A_{\boldsymbol{z}_k}^T - A_{\boldsymbol{z}_k}^T (\lambda I + A_{\boldsymbol{z}_k} A_{\boldsymbol{z}_k}^T)^{-1} A_{\boldsymbol{z}_k} A_{\boldsymbol{z}_k}^T\right] \boldsymbol{y} \\
&= \lambda^{-1} A_{\boldsymbol{z}_k}^T \left[I - (\lambda I + A_{\boldsymbol{z}_k} A_{\boldsymbol{z}_k}^T)^{-1} A_{\boldsymbol{z}_k} A_{\boldsymbol{z}_k}^T\right] \boldsymbol{y} \\
&= \lambda^{-1} A_{\boldsymbol{z}_k}^T \left[(\lambda I + A_{\boldsymbol{z}_k} A_{\boldsymbol{z}_k}^T)^{-1} (\lambda I + A_{\boldsymbol{z}_k} A_{\boldsymbol{z}_k}^T) - (\lambda I + A_{\boldsymbol{z}_k} A_{\boldsymbol{z}_k}^T)^{-1} A_{\boldsymbol{z}_k} A_{\boldsymbol{z}_k}^T\right] \boldsymbol{y} \\
&= \lambda^{-1} A_{\boldsymbol{z}_k}^T (\lambda I + A_{\boldsymbol{z}_k} A_{\boldsymbol{z}_k}^T)^{-1} \left[\lambda I + A_{\boldsymbol{z}_k} A_{\boldsymbol{z}_k}^T - A_{\boldsymbol{z}_k} A_{\boldsymbol{z}_k}^T\right] \boldsymbol{y} \\
&= \lambda^{-1} A_{\boldsymbol{z}_k}^T (\lambda I + A_{\boldsymbol{z}_k} A_{\boldsymbol{z}_k}^T)^{-1} \left[\lambda I\right] \boldsymbol{y} \\
&= A_{\boldsymbol{z}_k}^T (\lambda I + A_{\boldsymbol{z}_k} A_{\boldsymbol{z}_k}^T)^{-1} \boldsymbol{y}.
\end{aligned} \tag{2.7}$$

As before, to find (2.7) in practice, we do not calculate the inverse directly, but instead perform the following linear solve

$$(\lambda I + A_{\boldsymbol{z}_k} A_{\boldsymbol{z}_k}^T)\widehat{\boldsymbol{u}}_k = \boldsymbol{y}$$

to find $\widehat{\boldsymbol{u}}_k$ and then set $\boldsymbol{u}_k = A_{\boldsymbol{z}_k}^T \widehat{\boldsymbol{u}}_k$.

Next, we solve (2.4) numerically, and approximately, through the steepest descent iterative approach, detailed in Section 1.4. Specifically, we apply steepest descent on the cost function $G_k(\boldsymbol{z}) : \mathcal{Z}^n \to \mathbb{R}$ defined as $G_k(\boldsymbol{z}) = F(\boldsymbol{u}_{k-1}, \boldsymbol{z})$ for $\boldsymbol{u}_{k-1}$ fixed. Let $\boldsymbol{z}_k^j$ be the estimate of $\boldsymbol{z}$ on steepest descent step j of CG-LS iteration k. As in Section 1.4, we allow a different norm $||\cdot||_{(k,j)}$, with dual norm $||\cdot||_{*(k,j)}$, to define the descent direction $\boldsymbol{d}_k^j$ that generates $\boldsymbol{z}_k^j$. That is,

$$\boldsymbol{z}_k^j = \boldsymbol{z}_k^{j-1} + \eta_k^{(j)} \boldsymbol{d}_k^j(\boldsymbol{z}_k^{j-1})$$

for $\eta_k^{(j)}$ determined by the backtracking line search in Algorithm 1. We remark that the descent direction $\boldsymbol{d}_k^j$ is parameterized by the current $\boldsymbol{u}$ estimate, $\boldsymbol{u}_{k-1}$, and the measurements $\boldsymbol{y}$. Thus, we write the descent direction in any of the following three equivalent forms $\boldsymbol{d}_k^j = \boldsymbol{d}_k^j(\boldsymbol{z}) = \boldsymbol{d}_k^j(\boldsymbol{z}; \boldsymbol{u}_{k-1}, \boldsymbol{y})$.

Now, for the initial estimates, $\boldsymbol{z}_0$ and $\boldsymbol{u}_0$, of CG-LS define a modified ReLU (mReLU) activation function $\mathcal{P}_{a,b} : \mathbb{R} \to \mathbb{R}$ as

$$\mathcal{P}_{a,b}(x) = a + \text{ReLU}(x-a) - \text{ReLU}(x-b) = \begin{cases} a & x < a \\ x & a \le x \le b \\ b & b < x \end{cases}$$

for $a, b \in \mathbb{R}$. We remark that $\mathcal{P}_{a,b}(x)$ is a projection operator that projects a given input x on the compact interval $[\min\{a,b\}, \max\{a,b\}]$. The initial estimates for CG-LS then are

$$\begin{aligned} \boldsymbol{z}_0 &= \mathcal{P}_{a,b}(A^T \boldsymbol{y}) \\ \boldsymbol{u}_0 &= (A_{\boldsymbol{z}_0}^T A_{\boldsymbol{z}_0} + \lambda I)^{-1} A_{\boldsymbol{z}_0}^T \boldsymbol{y}. \end{aligned} \tag{2.8}$$

We remark that $\mathcal{P}_{a,b}$ in (2.8) is a componentwise function and thus is applied elementwise to the backprojection vector $A^T\boldsymbol{y}$. Applying $\mathcal{P}_{a,b}$ eliminates negative values of $A^T\boldsymbol{y}$, as $\boldsymbol{z}$ should have positive entries, as well as limits the maximum entries in the initial $\boldsymbol{z}$ estimate for stability in calculating $\boldsymbol{u}_0$.

Next, we determine convergence of CG-LS based on the dual norm, $||\cdot||_{*(k,j)}$, of the gradient, $\nabla_{\boldsymbol{z}}F(\boldsymbol{u},\boldsymbol{z})$, and a user chosen parameter $\delta > 0$. On each steepest descent step, j, of each iteration, k, we check if $||\nabla_{\boldsymbol{z}}F(\boldsymbol{u}_{k-1},\boldsymbol{z}_k^{j-1})||_{*(k,j)} < \delta$. When this holds, we exit the steepest descent steps taking $\boldsymbol{z}_k = \boldsymbol{z}_k^{j-1}$. Since, for all $k \geq 0$, $\nabla_{\boldsymbol{u}}F(\boldsymbol{u}_k,\boldsymbol{z}_k) = \boldsymbol{0}$, then $||\nabla_{\boldsymbol{z}}F(\boldsymbol{u}_{k-1},\boldsymbol{z}_k^0)||_{*(k,1)} < \delta$ implies $||\nabla F(\boldsymbol{u}_{k-1},\boldsymbol{z}_k^0)||_{*(k,1)} < \delta$. Therefore, once $||\nabla_{\boldsymbol{z}}F(\boldsymbol{u}_{k-1},\boldsymbol{z}_k^0)||_{*(k,1)} < \delta$ occurs we say CG-LS has converged and return estimates $\boldsymbol{u}_{k-1}$ and $\boldsymbol{z}_{k-1}$. Let K be a user chosen maximum number of iterations. If no $k \in \{1,2,\ldots,K\}$ exists such that $||\nabla_{\boldsymbol{z}}F(\boldsymbol{u}_{k-1},\boldsymbol{z}_k^0)||_{*(k,1)} < \delta$, then we say CG-LS did not converge.

Algorithm 2 Compound Gaussian Least Squares (CG-LS)

Input: Maximum number of iterations K, number of steepest descent steps J, measurement operator A, (possibly scaled) measurement $\boldsymbol{y}$, modified ReLU function $\mathcal{P}_{a,b}$, steepest descent dual norms and descent directions, and convergence parameter δ.

$\boldsymbol{z}_0 = \mathcal{P}_{a,b}(A^T\boldsymbol{y})$ and $\boldsymbol{u}_0 = (A_{\boldsymbol{z}_0}^T A_{\boldsymbol{z}_0} + \lambda I)^{-1} A_{\boldsymbol{z}_0}^T \boldsymbol{y}$
for $k \in \{1,2,\ldots,K\}$ **do**
    $\underline{\boldsymbol{z}\text{ ESTIMATION}}$:
    $\boldsymbol{z}_k^0 = \boldsymbol{z}_{k-1}$
    for $j \in \{1,2,\ldots,J\}$ **do**
        if $||\nabla_{\boldsymbol{z}}F(\boldsymbol{u}_{k-1},\boldsymbol{z}_k^{j-1})||_{*(k,j)}^2 < \delta$ **then**
            return $\boldsymbol{z}_k = \boldsymbol{z}_k^{j-1}$
        end if
        Compute step size $\eta_k^{(j)}$ (backtracking line search)
        $\boldsymbol{z}_k^j = \boldsymbol{z}_k^{j-1} + \eta_k^{(j)}\boldsymbol{d}_k^j(\boldsymbol{z}_k^{j-1};\boldsymbol{u}_{k-1},\boldsymbol{y})$
    end for
    $\boldsymbol{z}_k = \boldsymbol{z}_k^J$
    $\underline{\boldsymbol{u}\text{ ESTIMATION}}$:
    $\boldsymbol{u}_k = (A_{\boldsymbol{z}_k}^T A_{\boldsymbol{z}_k} + \lambda I)^{-1} A_{\boldsymbol{z}_k}^T \boldsymbol{y}$
end for

Output: $\boldsymbol{c}^* = \boldsymbol{z}_K \odot \boldsymbol{u}_K$

Finally, we remark that the Hierarchical Bayesian maximum a posteriori (HB-MAP) estimate [8, 9] serves as additional inspiration for the development of our CG-LS iterative algorithm proposed in Algorithm 2. Similar to CG-LS, the HB-MAP method uses the compound Gaussian prior, formulated as $\boldsymbol{z} \odot \boldsymbol{u}$ for $\boldsymbol{z} = h(\boldsymbol{\chi})$, in estimating image sparsity coefficients. Empirically shown in [8, 9], the HB-MAP estimator outperforms many other state-of-the-art iterative image estimation algorithms, including sparsity-based approaches, to the reconstruction of images from Radon transform measurements.

For sake of completeness, we briefly discuss the HB-MAP algorithm and the differences between it and CG-LS. First, HB-MAP performs a single estimate of the $\boldsymbol{z}$ field by estimating the $\boldsymbol{\chi}$ field through the optimization of

$$\boldsymbol{y}^T(A_{h(\boldsymbol{x})}\Sigma_u A_{h(\boldsymbol{x})}^T + \Sigma_\nu)^{-1}\boldsymbol{y} + \log\det(A_{h(\boldsymbol{x})}\Sigma_u A_{h(\boldsymbol{x})}^T + \Sigma_\nu) + \boldsymbol{x}^T\Sigma_\chi\boldsymbol{x} \tag{2.9}$$

w.r.t. $\boldsymbol{x}$ where Σ_ν and Σ_χ are covariance matrices for the assumed Gaussian distributions of $\boldsymbol{\nu}$ and $\boldsymbol{\chi}$. Second, with the optimum $\boldsymbol{\chi}^*$ of (2.9) the optimal $\boldsymbol{z}^*$ is estimated as $\boldsymbol{z}^* = h(\boldsymbol{\chi}^*)$. Lastly, using $\boldsymbol{z}^*$, an HB-MAP estimates $\boldsymbol{u}^*$ via a Tikhonov type estimate as in (2.6).

Compared to HB-MAP, our CG-LS algorithm distinctly performs alternating estimates of the $\boldsymbol{z}$ and $\boldsymbol{u}$ fields where each $\boldsymbol{z}$ estimate is produced through the optimization of the cost function in (2.4), which is far simpler and easier to optimize and implement as compared to (2.9). Specifically, calculating the Hessian and gradient of (2.9) is the most computationally intensive part of the HB-MAP algorithm due to the presence of Kronecker operators on high dimensional matrices [8]. Whereas the optimization of the $\boldsymbol{z}$ field in CG-LS requires no such Kronecker operations. Furthermore, it has been shown that the Tikhonov estimate of the $\boldsymbol{u}$ field is the primary component in handling a signal reconstruction [9]. Thus, the simplified optimization for the $\boldsymbol{z}$ field in CG-LS does not hinder the reconstructed signal quality much. That is, despite the simplified optimization for $\boldsymbol{z}$, CG-LS still produces high quality reconstructed signals, as compared to other prior art iterative estimation methods, as shown empirically in Section 2.2.4.

Therefore, as our goal is to apply algorithm unrolling to create a CG prior informed DNN, we developed the CG-LS algorithm, instead of directly unrolling HB-MAP into a DNN, since the HB-MAP estimate suffers from high computational cost in reconstruction time, memory, and data storage [8, 9]. These deficits of HB-MAP are problematic in producing a feasible unrolled DNN: First, the DNN adds additional memory and data storage requirements, on top of those required by HB-MAP, to optimizing the DNN parameters. Second, the unrolled DNN formulation requires the evaluation of HB-MAP over a batch of reconstructions, which is very slow and requires massive amounts of memory. Third, the unrolled DNN would be trained for many epochs where each epoch involves the evaluation of HB-MAP in reconstructing a set of training data signals, which again would be too slow for practical implementation.

2.2.2 Existence and Location of CG-LS Minimizers

Understanding the optimization landscape of a cost function is critical in understanding the success any optimization algorithm will have in finding extrema. For the CG-LS cost function in equation (2.2) we discuss the existence of minimizers along with details on their location that can be ensured through a sufficient scaling on the input measurements, $\boldsymbol{y}$, or proper choices of CG-LS parameters, λ and μ. Throughout the remainder of this chapter, we assume that $f : \mathcal{Z} \to \mathbb{R}$ is a C^2 function defined on $\mathcal{Z} = (z_{\min}, \infty)$ that is coercive. That is,

$$\lim_{z \to z_{\min}} f(z) \to \pm\infty \qquad \text{and} \qquad \lim_{z \to \infty} f(z) \to \pm\infty.$$

While in theory f should be an invertible function, for this section we do not specifically require this.

First, we note that the gradient of (2.2) is

$$\nabla F(\boldsymbol{u}, \boldsymbol{z}) = \begin{bmatrix} \nabla_{\boldsymbol{u}} F(\boldsymbol{u}, \boldsymbol{z}) \\ \nabla_{\boldsymbol{z}} F(\boldsymbol{u}, \boldsymbol{z}) \end{bmatrix} = \begin{bmatrix} -2A_{\boldsymbol{z}}^T(\boldsymbol{y} - A_{\boldsymbol{z}}\boldsymbol{u}) + 2\lambda\boldsymbol{u} \\ -2A_{\boldsymbol{u}}^T(\boldsymbol{y} - A_{\boldsymbol{u}}\boldsymbol{z}) + 2\mu D(f'(\boldsymbol{z}))f(\boldsymbol{z}) \end{bmatrix}. \tag{2.10}$$

Now, let $\boldsymbol{\rho} : \mathbb{R}^n \times \mathcal{Z}^n \to \mathbb{R}^n$ be given by

$$\boldsymbol{\rho}(\boldsymbol{u}, \boldsymbol{z}) = A^T(AD(\boldsymbol{z})\boldsymbol{u} - \boldsymbol{y}).$$

Additionally, let $\boldsymbol{h}_f : \mathcal{Z}^n \to \mathbb{R}^n$ be given by

$$\boldsymbol{h}_f(\boldsymbol{z}) = D(f''(\boldsymbol{z}))f(\boldsymbol{z}) + D(f'(\boldsymbol{z}))f'(\boldsymbol{z}).$$

The Hessian of (2.2), denoted $H_F(\boldsymbol{u}, \boldsymbol{z})$, is then given by

$$\begin{aligned} H_F(\boldsymbol{u}, \boldsymbol{z}) &= \begin{bmatrix} \boldsymbol{J}_{\boldsymbol{u}}(\nabla_{\boldsymbol{u}} F(\boldsymbol{u}, \boldsymbol{z})) & \boldsymbol{J}_{\boldsymbol{u}}(\nabla_{\boldsymbol{z}} F(\boldsymbol{u}, \boldsymbol{z})) \\ \boldsymbol{J}_{\boldsymbol{z}}(\nabla_{\boldsymbol{u}} F(\boldsymbol{u}, \boldsymbol{z})) & \boldsymbol{J}_{\boldsymbol{z}}(\nabla_{\boldsymbol{z}} F(\boldsymbol{u}, \boldsymbol{z})) \end{bmatrix} \\ &= 2 \begin{bmatrix} A_{\boldsymbol{z}}^T A_{\boldsymbol{z}} + \lambda I & A_{\boldsymbol{z}}^T A_{\boldsymbol{u}} + D(\boldsymbol{\rho}(\boldsymbol{u}, \boldsymbol{z})) \\ A_{\boldsymbol{u}}^T A_{\boldsymbol{z}} + D(\boldsymbol{\rho}(\boldsymbol{u}, \boldsymbol{z})) & A_{\boldsymbol{u}}^T A_{\boldsymbol{u}} + \mu D(\boldsymbol{h}_f(\boldsymbol{z})) \end{bmatrix} \end{aligned} \tag{2.11}$$

where $\boldsymbol{J}_{\boldsymbol{u}}$ and $\boldsymbol{J}_{\boldsymbol{z}}$ denote the Jacobian matrix with respect to $\boldsymbol{u}$ and $\boldsymbol{z}$, respectively.

Now, we show that the CG-LS cost function (2.2) is has no maxima.

Proposition 2.2.1. *The CG-LS cost function in equation (2.2) is strongly convex in the $\boldsymbol{u}$ block with parameter at least 2λ.*

Proof. Let $\boldsymbol{v} \in \mathbb{R}^n \setminus \{\boldsymbol{0}\}$. Applying (2.11) we have

$$\begin{bmatrix} \boldsymbol{v}^T & \boldsymbol{0}^T \end{bmatrix} H_F(\boldsymbol{u}, \boldsymbol{z}) \begin{bmatrix} \boldsymbol{v} \\ \boldsymbol{0} \end{bmatrix} = \boldsymbol{v}^T \boldsymbol{J}_{\boldsymbol{u}}(\nabla_{\boldsymbol{u}} F(\boldsymbol{u}, \boldsymbol{z}))\boldsymbol{v} = 2\boldsymbol{v}^T \left(A_{\boldsymbol{z}}^T A_{\boldsymbol{z}} + \lambda I \right) \boldsymbol{v} \geq 2\lambda$$

where the final inequality holds since $A_{\boldsymbol{z}}^T A_{\boldsymbol{z}}$ is a real symmetric matrix for all $\boldsymbol{z}$ and thus satisfies $\boldsymbol{v}^T A_{\boldsymbol{z}}^T A_{\boldsymbol{z}} \boldsymbol{v} \geq 0$ for all $\boldsymbol{v} \in \mathbb{R}^n$. This implies that $\boldsymbol{J}_{\boldsymbol{u}}(\nabla_{\boldsymbol{u}} F(\boldsymbol{u}, \boldsymbol{z})) \succeq 2\lambda I$ and thus (2.2) is strongly convex in the $\boldsymbol{u}$ block with parameter at least 2λ. ⊠

A consequence of Proposition 2.2.1 is that all zero gradient points of the CG-LS cost function (2.2) are local minima or saddle points. Thus, a local maximum of (2.2) only occurs on the

boundary where the gradient may not be zero. That is, define the boundary set $\partial\mathcal{Z}_{\min}$ as

$$\partial\mathcal{Z}_{\min} = \{\boldsymbol{z} \in \mathcal{Z}^n : z_i = z_{\min} \text{ for at least one } i = 1, 2, \ldots, n\}. \tag{2.12}$$

Then any local maximum must occur a boundary point $[\boldsymbol{u}, \boldsymbol{z}] \in \mathbb{R}^n \times \partial\mathcal{Z}_{\min}$. Since $\mathcal{Z} = (z_{\min}, \infty)$ is an open interval than it has no boundary, that is, $\partial\mathcal{Z}_{\min} = \emptyset$. Therefore, the optimization domain $\mathbb{R}^n \times \mathcal{Z}^n$ has no boundary and thus local maxima are inconsequential in implementing CG-LS.

Next, we express the minimizer solution, in $\boldsymbol{u}$, for the CG-LS cost function (2.2), with $\boldsymbol{z} \in \mathcal{Z}^n$ fixed, in the following new lemma. For this, we require a function $\boldsymbol{v}(\boldsymbol{z}) : \mathcal{Z}^n \to \mathbb{R}^n$ defined as

$$\boldsymbol{v}(\boldsymbol{z}) = A^T(A_{\boldsymbol{z}}A_{\boldsymbol{z}}^T + \lambda I)^{-1}\boldsymbol{y}. \tag{2.13}$$

Lemma 2.2.2. *Let $\boldsymbol{z} \in \mathcal{Z}^n$ be fixed. The CG-LS cost function (2.2) is minimized, over $\mathbf{u}$, at*

$$\boldsymbol{u}^* = (A_{\boldsymbol{z}}^T A_{\boldsymbol{z}} + \lambda I)^{-1} A_{\boldsymbol{z}}^T \boldsymbol{y} = D(\boldsymbol{z})\boldsymbol{v}(\boldsymbol{z}).$$

Proof. From equation (2.10), $\nabla_{\boldsymbol{u}} F(\boldsymbol{u}, \boldsymbol{z}) = \mathbf{0}$ if and only if $\boldsymbol{u}^* = (A_{\boldsymbol{z}}^T A_{\boldsymbol{z}} + \lambda I)^{-1} A_{\boldsymbol{z}}^T \boldsymbol{y}$. By Proposition 2.2.1, the stationary point $\boldsymbol{u}^*$ is a minimizer. Applying the Woodbury matrix identity [63] observe

$$(A_{\boldsymbol{z}}^T A_{\boldsymbol{z}} + \lambda I)^{-1} = \lambda^{-1}\left(I - A_{\boldsymbol{z}}^T(A_{\boldsymbol{z}}A_{\boldsymbol{z}}^T + \lambda I)^{-1}A_{\boldsymbol{z}}\right).$$

Also observe

$$\begin{aligned} I - (A_{\boldsymbol{z}}A_{\boldsymbol{z}}^T + \lambda I)^{-1}A_{\boldsymbol{z}}A_{\boldsymbol{z}}^T &= (A_{\boldsymbol{z}}A_{\boldsymbol{z}}^T + \lambda I)^{-1}(A_{\boldsymbol{z}}A_{\boldsymbol{z}}^T + \lambda I) - (A_{\boldsymbol{z}}A_{\boldsymbol{z}}^T + \lambda I)^{-1}A_{\boldsymbol{z}}A_{\boldsymbol{z}}^T \\ &= (A_{\boldsymbol{z}}A_{\boldsymbol{z}}^T + \lambda I)^{-1}\left(A_{\boldsymbol{z}}A_{\boldsymbol{z}}^T + \lambda I - A_{\boldsymbol{z}}A_{\boldsymbol{z}}^T\right) \\ &= \lambda(A_{\boldsymbol{z}}A_{\boldsymbol{z}}^T + \lambda I)^{-1}. \end{aligned}$$

Therefore

$$\begin{aligned}
\boldsymbol{u}^* &= (A_{\boldsymbol{z}}^T A_{\boldsymbol{z}} + \lambda I)^{-1} A_{\boldsymbol{z}}^T \boldsymbol{y} \\
&= \lambda^{-1} \left(I - A_{\boldsymbol{z}}^T (A_{\boldsymbol{z}} A_{\boldsymbol{z}}^T + \lambda I)^{-1} A_{\boldsymbol{z}} \right) A_{\boldsymbol{z}}^T \boldsymbol{y} \\
&= \lambda^{-1} A_{\boldsymbol{z}}^T \left(I - (A_{\boldsymbol{z}} A_{\boldsymbol{z}}^T + \lambda I)^{-1} A_{\boldsymbol{z}} A_{\boldsymbol{z}}^T \right) \boldsymbol{y} \\
&= D(\boldsymbol{z}) A^T (A_{\boldsymbol{z}} A_{\boldsymbol{z}}^T + \lambda I)^{-1} \boldsymbol{y} \\
&= D(\boldsymbol{z}) \boldsymbol{v}(\boldsymbol{z}). \qquad \boxtimes
\end{aligned}$$

Lemma 2.2.2 provides an avenue to gain further insight into optimizing the CG-LS cost function (2.2) over $\boldsymbol{z}$. In particular, we derive an insightful equation with roots coinciding to the stationary points, in $\boldsymbol{z}$, of (2.2) as stated in the following new lemma.

Lemma 2.2.3. *Let $\boldsymbol{v}(\boldsymbol{z})$ be given as in (2.13) and define $\boldsymbol{\mathcal{F}} : \mathcal{Z}^n \to \mathbb{R}^n$ by*

$$\boldsymbol{\mathcal{F}}(\boldsymbol{z}) = -\lambda D(\boldsymbol{z}) D(\boldsymbol{v}(\boldsymbol{z})) \boldsymbol{v}(\boldsymbol{z}) + \mu D(f'(\boldsymbol{z})) f(\boldsymbol{z}). \tag{2.14}$$

The CG-LS cost function in equation (2.2) has a stationary point $[\boldsymbol{z}^*, \boldsymbol{u}^*]$ if and only if $\boldsymbol{\mathcal{F}}(\boldsymbol{z}^*) = \mathbf{0}$ and $\boldsymbol{u}^*$ is given as in Lemma 2.2.2 with $\boldsymbol{z} = \boldsymbol{z}^*$.

Proof. The CG-LS cost function (2.2) has a stationary point when $\nabla F(\boldsymbol{u}, \boldsymbol{z}) = \mathbf{0}$, which holds if and only if $\nabla_{\boldsymbol{u}} F(\boldsymbol{u}, \boldsymbol{z}) = \mathbf{0}$ and $\nabla_{\boldsymbol{z}} F(\boldsymbol{u}, \boldsymbol{z}) = \mathbf{0}$. By Lemma 2.2.2 $\nabla_{\boldsymbol{u}} F(\boldsymbol{u}, \boldsymbol{z}) = \mathbf{0}$ if and only if

$\boldsymbol{u} = \boldsymbol{u}^* = D(\boldsymbol{z})\boldsymbol{v}(\boldsymbol{z})$. Then $\nabla F(\boldsymbol{u}, \boldsymbol{z}) = \mathbf{0}$ if and only if $\nabla_{\boldsymbol{z}} F(\boldsymbol{u}^*, \boldsymbol{z}) = \mathbf{0}$. Now, observe

$$\begin{aligned}
A^T(\boldsymbol{y} - A_{\boldsymbol{u}^*}\boldsymbol{z}) &= A^T(\boldsymbol{y} - AD(\boldsymbol{u}^*)\boldsymbol{z}) \\
&= A^T(\boldsymbol{y} - AD(\boldsymbol{z})^2\boldsymbol{v}(\boldsymbol{z})) \\
&= A^T(I - AD(\boldsymbol{z})^2A^T(A_{\boldsymbol{z}}A_{\boldsymbol{z}}^T + \lambda I)^{-1})\boldsymbol{y} \\
&= A^T((A_{\boldsymbol{z}}A_{\boldsymbol{z}}^T + \lambda I)(A_{\boldsymbol{z}}A_{\boldsymbol{z}}^T + \lambda I)^{-1} - AD(\boldsymbol{z})^2A^T(A_{\boldsymbol{z}}A_{\boldsymbol{z}}^T + \lambda I)^{-1})\boldsymbol{y} \\
&= A^T(A_{\boldsymbol{z}}A_{\boldsymbol{z}}^T + \lambda I - AD(\boldsymbol{z})^2A^T)(A_{\boldsymbol{z}}A_{\boldsymbol{z}}^T + \lambda I)^{-1}\boldsymbol{y} \\
&= A^T(A_{\boldsymbol{z}}A_{\boldsymbol{z}}^T + \lambda I - A_{\boldsymbol{z}}A_{\boldsymbol{z}}^T)(A_{\boldsymbol{z}}A_{\boldsymbol{z}}^T + \lambda I)^{-1}\boldsymbol{y} \\
&= \lambda\boldsymbol{v}(\boldsymbol{z}).
\end{aligned}$$

Hence

$$\begin{aligned}
\nabla_{\boldsymbol{z}} F(\boldsymbol{u}^*, \boldsymbol{z}) &= -2D(\boldsymbol{u}^*)A^T(\boldsymbol{y} - A_{\boldsymbol{u}^*}\boldsymbol{z}) + 2\mu D(f'(\boldsymbol{z}))f(\boldsymbol{z}) \\
&= -2\lambda D(\boldsymbol{z})D(\boldsymbol{v}(\boldsymbol{z}))\boldsymbol{v}(\boldsymbol{z}) + 2\mu D(f'(\boldsymbol{z}))f(\boldsymbol{z}) \\
&= 2\boldsymbol{\mathcal{F}}(\boldsymbol{z}).
\end{aligned}$$

Therefore, $\nabla_{\boldsymbol{z}} F(\boldsymbol{u}^*, \boldsymbol{z}) = \mathbf{0}$ if and only if $\boldsymbol{\mathcal{F}}(\boldsymbol{z}) = \mathbf{0}$. ⊠

As $\boldsymbol{z}$ is a positive random vector then each component of $\boldsymbol{w} = \lambda D(\boldsymbol{z})D(\boldsymbol{v}(\boldsymbol{z}))\boldsymbol{v}(\boldsymbol{z})$ is non-negative since $w_i = \lambda z_i v_i(\boldsymbol{z})^2$. Thus, (2.14) only has a root when all of the components of $\mu D(f'(\boldsymbol{z}))f(\boldsymbol{z})$ are non-negative. Define $\mathcal{S}^+(f) = \{z \in \mathcal{Z} : f'(z)f(z) \geq 0\}$. Then for every stationary point $[\boldsymbol{u}^*, \boldsymbol{z}^*]$ of the CG-LS cost function (2.2), each component of $\boldsymbol{z}^*$ must be contained in $\mathcal{S}^+$. Hence, we gain sufficient insight into the location of a minimizer for CG-LS by examining the function $f^2(z)$ and eliminate all regions outside of $\mathcal{S}^+(f)$, which corresponds to eliminating all regions where $f^2(z)$ is strictly decreasing. As an example, if $f^2(z)$ obtains a stationary point at z_0 and is decreasing to left of z_0, such as when f is invertible, then $\boldsymbol{z}^* \in [z_0, \infty)^n$.

We remark that (2.14) may have a root when $\lambda D(\boldsymbol{z})D(\boldsymbol{v}(\boldsymbol{z}))\boldsymbol{v}(\boldsymbol{z}) = \mathbf{0} = \mu D(f'(\boldsymbol{z}))f(\boldsymbol{z})$ but this case, in general, is not applicable. Since $\lambda D(\boldsymbol{z})D(\boldsymbol{v}(\boldsymbol{z}))\boldsymbol{v}(\boldsymbol{z}) = \mathbf{0}$ only when $\boldsymbol{v}(\boldsymbol{z}) = \mathbf{0}$. As-

suming that A is of full rank then $\boldsymbol{v}(\boldsymbol{z}) = \mathbf{0}$ only when $\boldsymbol{y} = \mathbf{0}$, which we assume to have non-trivial measurements and thus $\boldsymbol{y} \neq \mathbf{0}$. Furthermore, if A is not of full rank then $\lambda D(\boldsymbol{z})D(\boldsymbol{v}(\boldsymbol{z}))\boldsymbol{v}(\boldsymbol{z}) = \mathbf{0} = \mu D(f'(\boldsymbol{z}))f(\boldsymbol{z})$ only applies if there exists a vector $\widetilde{\boldsymbol{z}} \in \mathcal{Z}^n$ such that each component of $\widetilde{\boldsymbol{z}}$ is a stationary point of $f^2(z)$ and $(A_{\widetilde{\boldsymbol{z}}}A_{\widetilde{\boldsymbol{z}}}^T + \lambda I)^{-1}\boldsymbol{y}$ lies in the null space of A^T. This specific set of circumstances does not hold, in general, and thus this case is not applicable.

Now, assume that the measurements have been scaled. That is, $\boldsymbol{y} = s\widetilde{\boldsymbol{y}}$ where $\widetilde{\boldsymbol{y}}$ is given by (2.1) and s is a positive real value. With certain choices of scale, s, and CG-LS parameters, λ and μ, we can further refine the location of a stationary point for the CG-LS cost function (2.2) from knowledge of the stationary points of $f^2(z)$ as stated in the following new theorem.

Theorem 2.2.4. *Let $f^2(z)$ obtain a local minimizer at $z_0 \in \mathcal{Z}$. There exists a point $b \in \mathbb{R}$, with $b > z_0$, such that $f'(b)f(b) > 0$. Furthermore, there exists positive s, λ, and μ where the CG-LS cost function in equation (2.2) has a stationary point $[\boldsymbol{u}^*, \boldsymbol{z}^*]$ such that $\boldsymbol{z}^* \in [z_0, b]^n$ and $\boldsymbol{u}^* = (A_{\boldsymbol{z}^*}^T A_{\boldsymbol{z}^*} + \lambda I)^{-1} A_{\boldsymbol{z}^*}^T \boldsymbol{y}$.*

Proof. As z_0 is a local minimizer of $f^2(z)$ then there exits an $\epsilon > 0$ such that $f^2(z)$ is increasing on $(z_0, z_0 + \epsilon)$. That is $\frac{d}{dz}f^2(z) = 2f'(z)f(z) > 0$ on $(z_0, z_0 + \epsilon)$. So, fix b as any point in $(z_0, z_0 + \epsilon)$. Now, we write $\boldsymbol{\mathcal{F}}(\boldsymbol{z}) = [\mathcal{F}_1(\boldsymbol{z}), \mathcal{F}_2(\boldsymbol{z}), \ldots, \mathcal{F}_n(\boldsymbol{z})]^T$, for $\boldsymbol{\mathcal{F}}$ given in equation (2.14), and $\boldsymbol{v}(\boldsymbol{z}) = [v_1(\boldsymbol{z}), v_2(\boldsymbol{z}), \ldots, v_n(\boldsymbol{z})]^T$ for $\boldsymbol{v}$ given in equation (2.13). For $i \in \{1, 2, \ldots, n\}$ define

$$\mathcal{Z}_i(z) = \{\boldsymbol{z} \in [z_0, b]^n \subseteq \mathcal{Z}^n : z_i = z\}.$$

Since $f'(z_0)f(z_0) = 0$ then for any $\boldsymbol{z} \in \mathcal{Z}_i(z_0)$

$$\mathcal{F}_i(\boldsymbol{z}) = -\lambda z_0[v_i(\boldsymbol{z})]^2 \leq 0.$$

Write $\widetilde{\boldsymbol{v}}(\boldsymbol{z}) = A^T(A_{\boldsymbol{z}}A_{\boldsymbol{z}}^T + \lambda I)^{-1}\widetilde{\boldsymbol{y}} \equiv s^{-1}\boldsymbol{v}(\boldsymbol{z})$ and define

$$\widetilde{v}_{\max}(b) = \max_{1 \leq i \leq n} \max_{\boldsymbol{z} \in \mathcal{Z}_i(b)} |\widetilde{v}_i(\boldsymbol{z})|.$$

As $\mathcal{Z}_i(b)$ is a compact set and $\widetilde{\boldsymbol{v}}(\boldsymbol{z})$ a continuous function on $\mathcal{Z}_i(b)$, then $\max_{\boldsymbol{z}\in\mathcal{Z}_i(b)} |\widetilde{v}_i(\boldsymbol{z})|$ is defined and finite. Thus $\widetilde{v}_{\max}(b)$ is finite as it is a maximum over a finite set. Hence, if

$$\frac{\lambda}{\mu}s^2 \leq \frac{1}{\widetilde{v}_{\max}(b)^2}\frac{f'(b)f(b)}{b} \tag{2.15}$$

then

$$\mu f'(b)f(b) \geq \lambda b s^2 \widetilde{v}_{\max}(b)^2 \geq \lambda b s^2 \widetilde{v}_i(\boldsymbol{z})^2 \geq \lambda b v_i(\boldsymbol{z})^2$$

for any $\boldsymbol{z} \in \mathcal{Z}_i(b)$. Thus, for all $i \in \{1, 2, \ldots, n\}$ and any $\boldsymbol{z} \in \mathcal{Z}_i(b)$ we have

$$\mathcal{F}_i(\boldsymbol{z}) = -\lambda b v_i(\boldsymbol{z})^2 + \mu f'(b)f(b) \geq 0.$$

Therefore, by the Poincare-Miranda theorem [64] when s, λ and μ satisfy (2.15) there exists a $\boldsymbol{z}^* \in [z_0, b]^n$ such that $\boldsymbol{\mathcal{F}}(\boldsymbol{z}^*) = \mathbf{0}$. By Lemma 2.2.3, the point $[\boldsymbol{u}^*, \boldsymbol{z}^*]$, where $\boldsymbol{u}^* = (A_{\boldsymbol{z}^*}^T A_{\boldsymbol{z}^*} + \lambda I)^{-1} A_{\boldsymbol{z}^*}^T \boldsymbol{y}$, is a stationary point of the CG-LS cost function (2.2). ⊠

Applying Proposition 2.2.1, any stationary point obtained in Theorem 2.2.4 is a local minimizer or a saddle point of the CG-LS cost function (2.2). We extend Theorem 2.2.4 to ensure the obtained stationary point is a local minimizer with an additional requirement on the scaling, s, and CG-LS parameters μ and λ as well as convexity of $f^2(z)$. To show this extension we require the eigenvalues of a block 2×2 matrix containing diagonal matrices as given in [65].

Lemma 2.2.5 ([65]). *Let $\boldsymbol{a} = [a_i] \in \mathbb{R}^n, \boldsymbol{b} = [b_i] \in \mathbb{R}^n$, and $\boldsymbol{c} = [c_i] \in \mathbb{R}^n$. Define $A = D(\boldsymbol{a}), B = D(\boldsymbol{b})$, and $C = D(\boldsymbol{c})$. The eigenvalues of*

$$M = \begin{bmatrix} A & B \\ B & C \end{bmatrix}$$

are

$$\lambda_i^{\pm} = \frac{1}{2}\left(a_i + c_i \pm \sqrt{(a_i - c_i)^2 + 4b_i^2}\right) \tag{2.16}$$

for $i = 1, 2, \ldots, n$.

Proof. Let $r \in \mathbb{R}$. As B and $C - rI$ are diagonal matrices they commute. Thus, applying Theorem 3 from [65], we have

$$\det(M - rI) = \det((A - rI)(C - rI) - B^2).$$

Since $(A-rI)(C-rI)-B^2$ is a diagonal matrix the determinant is be the product of the diagonal entries. Namely

$$\det((A - rI)(C - rI) - B^2) = \prod_{i=1}^{n} \left[(a_i - r)(c_i - r) - b_i^2\right].$$

Then $\det(M-rI) = \det((A-rI)(C-rI)-B^2) = 0$ if and only if $\prod_{i=1}^{n} [(a_i - r)(c_i - r) - b_i^2] = 0$ implying $(a_i - r)(c_i - r) - b_i^2 = 0$ for $i = 1, 2, \ldots, n$. As $(a_i - r)(c_i - r) - b_i^2 = r^2 - (a_i + c_i) + a_i c_i - b_i^2$ then $(a_i - r)(c_i - r) - b_i^2 = 0$ when

$$\begin{aligned} r &= \frac{1}{2}\left(a_i + c_i \pm \sqrt{(a_i + c_i)^2 - 4(a_i c_i - b_i^2)}\right) \\ &= \frac{1}{2}\left(a_i + c_i \pm \sqrt{(a_i - c_i)^2 + 4b_i^2}\right) \\ &= \lambda_i^{\pm}. \end{aligned}$$

Therefore, the eigenvalues of M are $\lambda_i^{\pm}$ given in (2.16). ⊠

We now have the necessary tools to extend Theorem 2.2.4 from guaranteeing the location of a stationary point to guaranteeing the location of a minimizer of the CG-LS cost function (2.2), as provided in the following new theorem.

Theorem 2.2.6. *Let $f^2(z)$ be strictly convex on $[a, b] \subseteq \mathcal{Z}$ and obtain a local minimizer at $z_0 \in \mathcal{Z}$. Then there exits positive s, λ, and μ such that the CG-LS cost function (2.2) has a non-degenerate local minimizer $[\boldsymbol{u}^*, \boldsymbol{z}^*]$ where $\boldsymbol{z}^* \in [z_0, b]^n$ and $\boldsymbol{u}^* = (A_{\boldsymbol{z}^*}^T A_{\boldsymbol{z}^*} + \lambda I)^{-1} A_{\boldsymbol{z}^*}^T \boldsymbol{y}$.*

Proof. From Theorem 2.2.4, there exists a point $b \in \mathbb{R}$, with $b > z_0$, such that $f'(b)f(b) > 0$. Let s, λ, and μ satisfy (2.15). Then, by Theorem 2.2.4, there exists a stationary point $[\boldsymbol{u}^*, \boldsymbol{z}^*]$ such that

$\boldsymbol{z}^* \in [z_0, b]^n$ and $\boldsymbol{u}^* = (A_{\boldsymbol{z}^*}^T A_{\boldsymbol{z}^*} + \lambda I)^{-1} A_{\boldsymbol{z}^*}^T \boldsymbol{y}$. We now show the Hessian at this stationary point, $H_F(\boldsymbol{u}^*, \boldsymbol{z}^*)$ for H_F given in (2.11), is positive definite under certain choices of s, λ, and μ. Recall from (2.11)

$$\begin{aligned}\frac{H_F(\boldsymbol{u}^*, \boldsymbol{z}^*)}{2} &= \begin{bmatrix} A_{\boldsymbol{z}^*}^T A_{\boldsymbol{z}^*} + \lambda I & A_{\boldsymbol{z}^*}^T A_{\boldsymbol{u}^*} + D(\boldsymbol{\rho}(\boldsymbol{u}^*, \boldsymbol{z}^*)) \\ A_{\boldsymbol{u}^*}^T A_{\boldsymbol{z}^*} + D(\boldsymbol{\rho}(\boldsymbol{u}^*, \boldsymbol{z}^*)) & A_{\boldsymbol{u}^*}^T A_{\boldsymbol{u}^*} + \mu D(\boldsymbol{h}_f(\boldsymbol{z}^*)) \end{bmatrix} \\ &= \begin{bmatrix} A_{\boldsymbol{z}^*}^T \\ A_{\boldsymbol{u}^*}^T \end{bmatrix} \begin{bmatrix} A_{\boldsymbol{z}^*} & A_{\boldsymbol{u}^*} \end{bmatrix} + \begin{bmatrix} \lambda I & D(\boldsymbol{\rho}(\boldsymbol{u}^*, \boldsymbol{z}^*)) \\ D(\boldsymbol{\rho}(\boldsymbol{u}^*, \boldsymbol{z}^*)) & \mu D(\boldsymbol{h}_f(\boldsymbol{z}^*)) \end{bmatrix}\end{aligned}$$

for $\boldsymbol{\rho}(\boldsymbol{u}, \boldsymbol{z}) = A^T(AD(\boldsymbol{z})\boldsymbol{u} - \boldsymbol{y})$. From Lemma 2.2.2, we have $\boldsymbol{u}^* = (A_{\boldsymbol{z}^*}^T A_{\boldsymbol{z}^*} + \lambda I)^{-1} A_{\boldsymbol{z}^*}^T \boldsymbol{y} = D(\boldsymbol{z}^*)\boldsymbol{v}(\boldsymbol{z}^*)$ for $\boldsymbol{v}(\boldsymbol{z}) = A^T(A_{\boldsymbol{z}} A_{\boldsymbol{z}}^T + \lambda I)^{-1}\boldsymbol{y}$. Hence

$$\begin{aligned}\boldsymbol{\rho}(\boldsymbol{u}^*, \boldsymbol{z}^*) &= A^T(AD(\boldsymbol{z}^*)\boldsymbol{u}^* - \boldsymbol{y}) \\ &= A^T(A_{\boldsymbol{z}^*} D(\boldsymbol{z}^*)\boldsymbol{v}(\boldsymbol{z}^*) - \boldsymbol{y}) \\ &= A^T(A_{\boldsymbol{z}^*} A_{\boldsymbol{z}^*}^T (A_{\boldsymbol{z}^*} A_{\boldsymbol{z}^*}^T + \lambda I)^{-1}\boldsymbol{y} - \boldsymbol{y}) \\ &= A^T(A_{\boldsymbol{z}^*} A_{\boldsymbol{z}^*}^T (A_{\boldsymbol{z}^*} A_{\boldsymbol{z}^*}^T + \lambda I)^{-1} - (A_{\boldsymbol{z}^*} A_{\boldsymbol{z}^*}^T + \lambda I)(A_{\boldsymbol{z}^*} A_{\boldsymbol{z}^*}^T + \lambda I)^{-1})\boldsymbol{y} \\ &= A^T(A_{\boldsymbol{z}^*} A_{\boldsymbol{z}^*}^T - A_{\boldsymbol{z}^*} A_{\boldsymbol{z}^*}^T - \lambda I)(A_{\boldsymbol{z}^*} A_{\boldsymbol{z}^*}^T + \lambda I)^{-1}\boldsymbol{y} \\ &= -\lambda A^T(A_{\boldsymbol{z}^*} A_{\boldsymbol{z}^*}^T + \lambda I)^{-1}\boldsymbol{y} \\ &= -\lambda \boldsymbol{v}(\boldsymbol{z}^*).\end{aligned}$$

Thus

$$\begin{bmatrix} \lambda I & D(\boldsymbol{\rho}(\boldsymbol{u}^*, \boldsymbol{z}^*)) \\ D(\boldsymbol{\rho}(\boldsymbol{u}^*, \boldsymbol{z}^*)) & \mu D(\boldsymbol{h}_f(\boldsymbol{z}^*)) \end{bmatrix} = \begin{bmatrix} \lambda I & -\lambda D(\boldsymbol{v}(\boldsymbol{z}^*)) \\ -\lambda D(\boldsymbol{v}(\boldsymbol{z}^*)) & \mu D(\boldsymbol{h}_f(\boldsymbol{z}^*)) \end{bmatrix} \tag{2.17}$$

which, by Lemma 2.2.5, has eigenvalues

$$\lambda_i^{\pm} = \frac{\lambda}{2}\left(1 + c_i \pm \sqrt{(1 - c_i)^2 + 4v_i(\boldsymbol{z}^*)^2}\right)$$

for $c_i = \frac{\mu}{\lambda}(f''(z_i^*)f(z_i^*) + f'(z_i^*)^2)$ where $i = 1, 2, \ldots, n$. Since $f^2(z)$ is strictly convex on $[a, b]$ then $f''(z_i^*)f(z_i^*) + f'(z_i^*)^2 > 0$ implying $c_i > 0$ and thus $\lambda_i^+ > 0$.

Observe,

$$\begin{aligned}
\lambda_i = \frac{\lambda}{2}\left(1 + c_i - \sqrt{(1-c_i)^2 + 4v_i(\boldsymbol{z}^*)^2}\right) > 0 & \\
1 + c_i > \sqrt{(1-c_i)^2 + 4v_i(\boldsymbol{z}^*)^2} & \\
(1+c_i)^2 > (1+c_i)^2 + 4(v_i(\boldsymbol{z}^*)^2 - c_i) & \\
c_i > v_i(\boldsymbol{z}^*)^2. &
\end{aligned}$$

That is, $\lambda_i^- > 0$ if and only if $c_i > v_i(\boldsymbol{z}^*)^2$. Write $\widetilde{\boldsymbol{v}}(\boldsymbol{z}) = A^T(A_{\boldsymbol{z}}A_{\boldsymbol{z}}^T + \lambda I)^{-1}\widetilde{\boldsymbol{y}} \equiv s^{-1}\boldsymbol{v}(\boldsymbol{z})$ and define

$$\begin{aligned}
h_{f\min} &= \min_{z\in[z_0,b]} \left\{f''(z)f(z) + f'(z)^2\right\} \\
\widetilde{v}_{\max} &= \max_{1\le i\le n} \max_{\boldsymbol{z}\in[z_0,b]^n} |\widetilde{v}_i(\boldsymbol{z})|.
\end{aligned}$$

As $[z_0, b]^n$ is a compact set and $\widetilde{\boldsymbol{v}}(\boldsymbol{z})$ is a continuous function on $[z_0, b]^n$, then $\max_{\boldsymbol{z}\in[z_0,b]^n} |\widetilde{v}_i(\boldsymbol{z})|$ is defined and finite. Thus, $\widetilde{v}_{\max}$ is finite as it is a maximum over a finite set. Additionally, as $[z_0, b]$ is compact and $f^2(z)$ is strictly convex on $[z_0, b]$ then $h_{f\min}$ is defined and satisfies $h_{f\min} > 0$. Therefore, if

$$\frac{h_{f\min}}{\widetilde{v}_{\max}^2} > \frac{\lambda}{\mu}s^2 \tag{2.18}$$

then, for all $i = 1, 2, \ldots, n$, we have

$$\frac{f''(z_i^*)f(z_i^*) + f'(z_i^*)^2}{\widetilde{v}_{\max}^2} \ge \frac{h_{f\min}}{\widetilde{v}_{\max}^2} > \frac{\lambda}{\mu}s^2.$$

This implies

$$c_i = \frac{\mu}{\lambda}\left(f''(z_i^*)f(z_i^*) + f'(z_i^*)^2\right) > s^2\widetilde{v}_{\max}^2 \geq s^2\widetilde{v}_i(\boldsymbol{z}^*)^2 = v_i(\boldsymbol{z}^*)^2.$$

Thus, when s, λ, and μ satisfy (2.18) then $\lambda_i^{\pm} > 0$ implying (2.17) is a positive definite matrix. Therefore, when s, λ, and μ satisfy (2.18) the Hessian $H_F(\boldsymbol{u}^*, \boldsymbol{z}^*)$ is the sum of a positive semi-definite matrix and a positive definite matrix, implying it is positive definite and $[\boldsymbol{u}^*, \boldsymbol{z}^*]$ is a local minimizer of the CG-LS cost function (2.2). ⊠

Combining the two requirements on s, λ, and μ given in (2.15) and (2.18) we have the single requirement

$$\frac{\lambda}{\mu}s^2 < \min\left\{\frac{h_{f\min}}{\widetilde{v}_{\max}^2}, \frac{f'(b)f(b)}{b\widetilde{v}_{\max}(b)^2}\right\}.$$

As $\widetilde{v}_{\max} \geq \widetilde{v}_{\max}(b)$ we alternatively express this requirement to guarantee a local minimizer as

$$\frac{\lambda}{\mu}s^2 < \frac{1}{\widetilde{v}_{\max}^2}\min\left\{h_{f\min}, \frac{f'(b)f(b)}{b}\right\}. \tag{2.19}$$

Since f a single variable function obtaining $\min\left\{h_{f\min}, \frac{f'(b)f(b)}{b}\right\}$ is practical. On the other hand, obtaining $\widetilde{v}_{\max}^2$ is far more difficult as our optimization variable $\boldsymbol{z}$ is contained within a matrix inversion. Although a rough numerical approximation of $\widetilde{v}_{\max}^2$ may be sufficient to choose a scale s for the implementation of CG-LS. For our numerical experiments, we choose a scaling s empirically instead of attempting to approximate $\widetilde{v}_{\max}^2$.

We remark that Theorem 2.2.6 extends optimization insights from the single variable function $f^2(z)$, which is far easier to analyze, to the multivariate CG-LS cost function (2.2) that involves $\boldsymbol{z}$ and $\boldsymbol{u}$. In particular, knowing that $\boldsymbol{z}^* \in [z_0, b]^n$, which is informed by intervals of convexity and roots of $f^2(z)$, we can choose the mReLU interval defining $\boldsymbol{z}_0$ to be $[z_0, b]$, which may increase the convergence rate for CG-LS. Furthermore, we may localize the optimization in CG-LS by applying the mReLU function, $\mathcal{P}_{[z_0,b]}$, after each update in line 10 of Algorithm 2 to further increase the convergence rate. Finally, the convexity of $f^2(z)$ in conjunction with the scaling law revealed in

Theorem 2.2.6 guarantees a region of convexity for the cost function in (2.4) and the existence of a unique stationary point within this convex region.

2.2.3 Convergence of CG-LS

For convergence analysis, we assume that f is a C^2 function, defined on $\mathcal{Z} = (z_{\min}, \infty)$, and is coercive. That is,

$$\lim_{z \to z_{\min}} f(z) \to \pm\infty \qquad \text{and} \qquad \lim_{z \to \infty} f(z) \to \pm\infty.$$

Now, given initial estimates, $\boldsymbol{u}_0$ and $\boldsymbol{z}_0$, define a sublevel set

$$S(\boldsymbol{u}_0, \boldsymbol{z}_0) = \{(\boldsymbol{u}, \boldsymbol{z}) \in \mathbb{R}^n \times \mathcal{Z}^n : F(\boldsymbol{u}, \boldsymbol{z}) \leq F(\boldsymbol{u}_0, \boldsymbol{z}_0)\}$$

for F the CG-LS cost function (2.2). We note that $S(\boldsymbol{u}_0, \boldsymbol{z}_0) \subseteq \mathbb{R}^n \times \mathcal{Z}^n$ is closed, since F is continuous, and bounded, since F is coercive in both $\boldsymbol{u}$ and $\boldsymbol{z}$. Thus, $S(\boldsymbol{u}_0, \boldsymbol{z}_0)$ is compact by the Heine-Borel theorem [53]. Now, we show that F has a Lipschitz continuous gradient on $S(\boldsymbol{u}_0, \boldsymbol{z}_0)$. For this, we use a proposition about the convex hull of a compact set dependent on the following lemma; a result which appears in [66].

Lemma 2.2.7 ([66]). *The set $\mathcal{W} = \{(w_0, w_1, \ldots, w_d) : w_i \geq 0, \sum_{i=0}^{d} w_i = 1\} \subset \mathbb{R}^{d+1}$ is compact.*

Proof. For any $\boldsymbol{w} \in \mathcal{W}$ we note that $1 = \|\boldsymbol{w}\|_1 \geq \|\boldsymbol{w}\|_2$ and thus $\mathcal{W}$ is bounded. Now let the sequence $\{\boldsymbol{w}_i\}_{i \in \mathbb{N}} \subseteq \mathcal{W}$ converge to $\boldsymbol{w}$. For notation let $[\boldsymbol{w}_i]_j$ be the jth component of the vector $\boldsymbol{w}_i$. Assume for contradiction that $\boldsymbol{w} \notin \mathcal{W}$ then $w_j < 0$ for some j or $\sum_{i=0}^{d} w_i \neq 1$. First, if $w_j < 0$ for some j then for all $i \in \mathbb{N}$

$$\|\boldsymbol{w}_i - \boldsymbol{w}\|_2 \geq |[\boldsymbol{w}_n]_j - w_j| = [\boldsymbol{w}_n]_j - w_j \geq |w_j| > 0$$

implying that $\{\boldsymbol{w}_i\}_{i \in \mathbb{N}}$ does not converge to $\boldsymbol{w}$, which is a contradiction. Hence, all components of $\boldsymbol{w}$ must be non-negative. Second, assume $\sum_{i=0}^{d} w_i = a \neq 1$ for all $w_i \geq 0$. Then, using the

reverse triangle inequality, for all $i \in \mathbb{N}$

$$\|\boldsymbol{w}_i - \boldsymbol{w}\|_2 \geq \frac{1}{\sqrt{d+1}}\|\boldsymbol{w}_i - \boldsymbol{w}\|_1 \geq |\,\|\boldsymbol{w}_i\|_1 - \|\boldsymbol{w}\|_1\,| = \left|\sum_{j=0}^{d}[\boldsymbol{w}_i]_j] - \sum_{j=0}^{d}\boldsymbol{w}_j\right| = |1-a| > 0$$

implying that $\{\boldsymbol{w}_i\}_{i\in\mathbb{N}}$ does not converge to $\boldsymbol{w}$, which is a contradiction. Thus, $\boldsymbol{w} \in \mathcal{W}$ implying that $\mathcal{W}$ contains all its limit points and is closed. Therefore, $\mathcal{W}$ is compact. ⊠

Now we show the required proposition that the convex hull of a compact set is compact as provided in [66].

Proposition 2.2.8 ([66])**.** *Let $\mathcal{S} \subset R^d$ be compact. Then the convex hull of $\mathcal{S}$, denoted Conv$(\mathcal{S})$, is compact.*

Proof. Define $\mathcal{W} = \{(w_0, w_1, \ldots, w_d) : w_i \geq 0, \sum_{i=0}^{d} w_i = 1\} \subseteq \mathbb{R}^{d+1}$ and let $G : \mathcal{S}^{d+1} \times \mathcal{W} \to \mathbb{R}^d$ be a convex combination function given by

$$G(\boldsymbol{x}, \boldsymbol{w}) = \sum_{i=0}^{d} w_i \boldsymbol{x}_i$$

where $\boldsymbol{x} = (\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_d) \in \mathcal{S}^{d+1}$ and $\boldsymbol{w} = (w_0, w_1, \ldots, w_d) \in \mathbb{R}^{d+1}$.

Let $G(\mathcal{S}^{d+1} \times \mathcal{W})$ be the image of G. That is, $G(\mathcal{S}^{d+1} \times \mathcal{W}) = \{G(\boldsymbol{x}, \boldsymbol{w}) : \boldsymbol{x} \in \mathcal{S}^{d+1}, \boldsymbol{w} \in \mathcal{W}\}$. Note that since $\mathcal{S}$ is compact, Cartesian products of compact sets are compact, and, by Lemma 2.2.7, $\mathcal{W}$ is compact then $\mathcal{S}^{d+1} \times \mathcal{W}$ is compact. Next, let $[\boldsymbol{x}_i]_j$ denote the jth component of vector $\boldsymbol{x}_i$ and $\boldsymbol{\delta}_i \in \mathbb{R}^d$ be a vector with 1 in the ith component and zeros in all other components. We observe that $\nabla_{[\boldsymbol{x}_i]_j} G(\boldsymbol{x}, \boldsymbol{w}) = w_i \boldsymbol{\delta}_j$. Similarly, $\nabla_{\boldsymbol{w}} G(\boldsymbol{x}, \boldsymbol{w}) = \boldsymbol{x}$. Hence, G is a differentiable and thus continuous function. Therefore, as the continuous image of a compact set is compact, $G(\mathcal{S}^{d+1} \times \mathcal{W})$ is a compact set.

Now, by definition of a convex hull, $G(\boldsymbol{x}, \boldsymbol{w}) \in \text{Conv}(\mathcal{S})$ for all $\boldsymbol{x} \in \mathcal{S}^{d+1}$ and $\boldsymbol{w} \in \mathcal{W}$. Hence, $G(\mathcal{S}^{d+1} \times \mathcal{W}) \subseteq \text{Conv}(\mathcal{S})$. By Carathéodary's Theorem [67,68], for any $\boldsymbol{v} \in \text{Conv}(\mathcal{S})$ there exists a $\boldsymbol{x} \in \mathcal{S}^{d+1}$ and $\boldsymbol{w} \in \mathcal{W}$ such that $\boldsymbol{v} = G(\boldsymbol{x}, \boldsymbol{w})$. Hence, $\text{Conv}(\mathcal{S}) \subseteq G(\mathcal{S}^{d+1} \times \mathcal{W})$. Therefore, $G(\mathcal{S}^{d+1} \times \mathcal{W}) = \text{Conv}(\mathcal{S})$ and thus the convex hull $\text{Conv}(\mathcal{S})$ is compact. ⊠

Using Proposition 2.2.8 we show that the CG-LS cost function (2.2) has Lipschitz continuous gradient on the sublevel set $S(\boldsymbol{u}_0, \boldsymbol{z}_0)$ as given in the following new corollary.

Corollary 2.2.9. *The CG-LS cost function (2.2) has Lipschitz continuous gradient on the sublevel set* $S(\boldsymbol{u}_0, \boldsymbol{z}_0)$.

Proof. As f is a twice continuously differentiable function on $\mathcal{Z}$ then the CG-LS cost function $F(\boldsymbol{u}, \boldsymbol{z})$ is twice continuously differentiable on $\mathbb{R}^n \times \mathcal{Z}^n$. Hence, the Hessian of F, denoted $H_F(\boldsymbol{u}, \boldsymbol{z})$ and given in (2.11), is a continuous function of $\boldsymbol{u}$ and $\boldsymbol{z}$ on $\mathbb{R}^n \times \mathcal{Z}^n$. Thus, $\|H_F(\boldsymbol{u}, \boldsymbol{z})\|_2$ is a continuous function on $\mathbb{R}^n \times \mathcal{Z}^n$. As $S(\boldsymbol{u}_0, \boldsymbol{z}_0) \subseteq \mathbb{R}^n \times \mathcal{Z}^n$ is compact then by Proposition 2.2.8 the convex hull of $S(\boldsymbol{u}_0, \boldsymbol{z}_0)$, denoted as $\text{Conv}(S(\boldsymbol{u}_0, \boldsymbol{z}_0))$, is compact. Furthermore, as $\mathbb{R}^n \times \mathcal{Z}^n$ is convex and $\text{Conv}(S(\boldsymbol{u}_0, \boldsymbol{z}_0))$ is the smallest convex set containing $S(\boldsymbol{u}_0, \boldsymbol{z}_0)$ then $\text{Conv}(S(\boldsymbol{u}_0, \boldsymbol{z}_0)) \subseteq \mathbb{R}^n \times \mathcal{Z}^n$. Thus $F(\boldsymbol{u}, \boldsymbol{z})$, $H_F(\boldsymbol{u}, \boldsymbol{z})$, and $\|H_F(\boldsymbol{u}, \boldsymbol{z})\|_2$ are defined on $\text{Conv}(S(\boldsymbol{u}_0, \boldsymbol{z}_0))$. Hence, by the Extreme Value Theorem [53], since $\|H_F(\boldsymbol{u}, \boldsymbol{z})\|_2$ is a continuous function on the compact set $\text{Conv}(S(\boldsymbol{u}_0, \boldsymbol{z}_0))$ then $\|H_F(\boldsymbol{u}, \boldsymbol{z})\|_2$ obtains a bounded maximum on $\text{Conv}(S(\boldsymbol{u}_0, \boldsymbol{z}_0))$ that we denote by $H_{F,\max}$. Therefore, by the Mean Value Theorem [53], for all $(\boldsymbol{u}_i, \boldsymbol{z}_i) \in \text{Conv}(S(\boldsymbol{u}_0, \boldsymbol{z}_0))$ where $i \in \{1, 2\}$

$$\begin{aligned}\|\nabla F(\boldsymbol{u}_1, \boldsymbol{z}_1) - \nabla F(\boldsymbol{u}_2, \boldsymbol{z}_2)\|_2 &\leq \left(\max_{\boldsymbol{u}, \boldsymbol{z} \in \text{Conv}(S(\boldsymbol{u}_0, \boldsymbol{z}_0))} \|H_F(\boldsymbol{u}, \boldsymbol{z})\|_2 \right) \|(\boldsymbol{u}_1, \boldsymbol{z}_1) - (\boldsymbol{u}_2, \boldsymbol{z}_2)\|_2 \\ &\leq H_{F,\max} \|(\boldsymbol{u}_1, \boldsymbol{z}_1) - (\boldsymbol{u}_2, \boldsymbol{z}_2)\|_2\end{aligned}$$

and thus F has Lipschitz continuous gradient on $\text{Conv}(S(\boldsymbol{u}_0, \boldsymbol{z}_0))$. Similarly, F has Lipschitz continuous gradient on $S(\boldsymbol{u}_0, \boldsymbol{z}_0)$, with Lipschitz constant at most $H_{F,\max}$, since $S(\boldsymbol{u}_0, \boldsymbol{z}_0) \subseteq \text{Conv}(S(\boldsymbol{u}_0, \boldsymbol{z}_0))$. ⊠

Next, we remark that a Euclidean bound exists for any norm, $||\cdot||$, on $\mathbb{R}^n$. That is, there exists a constant $0 < \gamma \leq 1$ such that $||\cdot|| \geq \gamma||\cdot||_2$. So, for every CG-LS steepest descent norm $||\cdot||_{(k,j)}$ with dual norm $||\cdot||_{*(k,j)}$ we let $\gamma_{(k,j)}$ and $\gamma_{*(k,j)}$ be the respective Euclidean bound constants. To show CG-LS, in Algorithm 2, converges, we first give a lower bound on the change in the cost function for a steepest descent step on $\boldsymbol{z}$ in the following new proposition.

Proposition 2.2.10. *Define $G_k(\boldsymbol{z}) = F(\boldsymbol{u}_{k-1}, \boldsymbol{z})$ for $\boldsymbol{u}_{k-1}$ fixed and $F(\boldsymbol{u}, \boldsymbol{z})$ given in (2.2). For every $k, j \in \mathbb{N}$, there exists a positive constant, $c_{(k,j)}$, such that CG-LS in Algorithm 2 satisfies*

$$G_k\left(\boldsymbol{z}_k^{j-1}\right) - G_k\left(\boldsymbol{z}_k^{j}\right) \geq c_{(k,j)} \left\|\nabla G_k\left(\boldsymbol{z}_k^{j-1}\right)\right\|_{*(k,j)}^2.$$

Proof. By definition $(\boldsymbol{u}_0, \boldsymbol{z}_0) \in S(\boldsymbol{u}_0, \boldsymbol{z}_0)$. For any fixed $k, j \in \mathbb{N}$ assume that $\left(\boldsymbol{u}_{k-1}, \boldsymbol{z}_k^{j-1}\right) \in S(\boldsymbol{u}_0, \boldsymbol{z}_0)$. As $\boldsymbol{z}_k^j = \boldsymbol{z}_k^{j-1} + \eta_k^{(j)} \boldsymbol{d}_k^j\left(\boldsymbol{z}_k^{j-1}\right)$ and $\boldsymbol{d}_k^j\left(\boldsymbol{z}_k^{j-1}\right)$ is a constrained minimizer of the linear approximation of G_k centered at $\boldsymbol{z}_k^{j-1}$, then, for sufficiently small $\eta_k^{(j)} > 0$, it holds that $G_k(\boldsymbol{z}_k^j) \leq G_k(\boldsymbol{z}_k^{j-1})$. Hence, $\left(\boldsymbol{u}_{k-1}, \boldsymbol{z}_k^j\right) \in S(\boldsymbol{u}_0, \boldsymbol{z}_0)$ and thus, by induction on j, $\left(\boldsymbol{u}_{k-1}, \boldsymbol{z}_k^j\right) \in S(\boldsymbol{u}_0, \boldsymbol{z}_0)$ for all $j \in \mathbb{N}$. Implying $(\boldsymbol{u}_{k-1}, \boldsymbol{z}_k) = \left(\boldsymbol{u}_{k-1}, \boldsymbol{z}_k^J\right) \in S(\boldsymbol{u}_0, \boldsymbol{z}_0)$. As $\boldsymbol{u}_k$ is the global minimizer of $F(\boldsymbol{u}, \boldsymbol{z}_k)$ then $F(\boldsymbol{u}_k, \boldsymbol{z}_k) \leq F(\boldsymbol{u}_{k-1}, \boldsymbol{z}_k)$. Hence, $(\boldsymbol{u}_k, \boldsymbol{z}_k) \in S(\boldsymbol{u}_0, \boldsymbol{z}_0)$ and thus, by induction on k, $(\boldsymbol{u}_k, \boldsymbol{z}_k) \in S(\boldsymbol{u}_0, \boldsymbol{z}_0)$ for all $k \in \mathbb{N}$. Therefore, the sequence of estimates generated by CG-LS satisfies $\{(\boldsymbol{u}_k, \boldsymbol{z}_k^j)\}_{k\in\mathbb{N}}^{j\in\mathbb{N}} \subseteq S(\boldsymbol{u}_0, \boldsymbol{z}_0)$.

By Corollary 2.2.9, the function $G_k(\boldsymbol{z})$ has Lipschitz continuous gradient on $S(\boldsymbol{u}_0, \boldsymbol{z}_0)$. Let $L \coloneqq L_{\boldsymbol{z}}(\boldsymbol{u}_0, \boldsymbol{z}_0)$ be the smallest Lipschitz constant of ∇G_k on $S(\boldsymbol{u}_0, \boldsymbol{z}_0)$. Thus, for any $\boldsymbol{x}$ and $\boldsymbol{z}$ such that the points $(\boldsymbol{u}_{k-1}, \boldsymbol{x})$ and $(\boldsymbol{u}_{k-1}, \boldsymbol{z})$ satisfy $(\boldsymbol{u}_{k-1}, \boldsymbol{x}), (\boldsymbol{u}_{k-1}, \boldsymbol{z}) \in S(\boldsymbol{u}_0, \boldsymbol{z}_0)$ it holds that

$$G_k(\boldsymbol{z}) \leq G_k(\boldsymbol{x}) + \langle \nabla G_k(\boldsymbol{x}), \boldsymbol{z} - \boldsymbol{x}\rangle + \frac{L}{2}\|\boldsymbol{z} - \boldsymbol{x}\|_2^2.$$

Set $\boldsymbol{x} = \boldsymbol{z}_k^{j-1}$ and $\boldsymbol{z} = \boldsymbol{z}_k^j = \boldsymbol{z}_k^{j-1} + \eta_k^{(j)} \boldsymbol{d}_k^j\left(\boldsymbol{z}_k^{j-1}\right)$ for small enough step size, $\eta_k^{(j)} > 0$, such that $(\boldsymbol{u}_{k-1}, \boldsymbol{z}_k^j) \in S(\boldsymbol{u}_0, \boldsymbol{z}_0)$. Then

$$G_k\left(\boldsymbol{z}_k^j\right) \leq G_k\left(\boldsymbol{z}_k^{j-1}\right) + \eta_k^{(j)} \nabla G_k\left(\boldsymbol{z}_k^{j-1}\right)^T \boldsymbol{d}_k^j\left(\boldsymbol{z}_k^{j-1}\right) + \frac{L}{2}\left\|\eta_k^{(j)} \boldsymbol{d}_k^j\left(\boldsymbol{z}_k^{j-1}\right)\right\|_2^2. \tag{2.20}$$

Applying Lemma 1.4.5, we have

$$\nabla G_k\left(\boldsymbol{z}_k^{j-1}\right)^T \boldsymbol{d}_k^j(\boldsymbol{z}_k^{j-1}) = -\|\nabla G_k(\boldsymbol{z}_k^{j-1})\|_{*(k,j)}^2.$$

Since $\gamma_{(k,j)}^2\|\cdot\|_2^2 \leq \|\cdot\|_{(k,j)}^2$ then applying Lemma 1.4.6,

$$\left\|\boldsymbol{d}_k^j(\boldsymbol{z}_k^{j-1})\right\|_2^2 \leq \frac{1}{\gamma_{(k,j)}^2}\left\|\boldsymbol{d}_k^j(\boldsymbol{z}_k^{j-1})\right\|_{(k,j)}^2 = \frac{1}{\gamma_{(k,j)}^2}\left\|\nabla G_k(\boldsymbol{z}_k^{j-1})\right\|_{*(k,j)}^2.$$

Combining these two properties with (2.20) gives

$$G_k\left(\boldsymbol{z}_k^j\right) \leq G_k\left(\boldsymbol{z}_k^{j-1}\right) - \eta_k^{(j)}\left(1 - \frac{\eta_k^{(j)}L}{2\gamma_{(k,j)}^2}\right)\left|\left|\nabla G_k\left(\boldsymbol{z}_k^{j-1}\right)\right|\right|_{*(k,j)}^2. \tag{2.21}$$

The backtracking line search simultaneously requires

$$G_k\left(\boldsymbol{z}_k^j\right) \leq G_k\left(\boldsymbol{z}_k^{j-1}\right) - \tau\eta_k^{(j)}\left|\left|\nabla G_k\left(\boldsymbol{z}_k^{j-1}\right)\right|\right|_{*(k,j)}^2. \tag{2.22}$$

Since $\tau \in (0, 1/2]$, then (2.21) implies (2.22) when $\eta_k^{(j)} \leq \gamma_{(k,j)}^2/L$. Hence, the backtracking line search step size satisfies $1 \geq \eta_k^{(j)} \geq \min\left\{1, \beta\gamma_{(k,j)}^2/L\right\} > 0$, which combined with (2.21) and (2.22) gives

$$G_k\left(\boldsymbol{z}_k^{j-1}\right) - G_k\left(\boldsymbol{z}_k^j\right) \geq \tau\min\left\{1, \beta\gamma_{(k,j)}^2/L\right\}\left|\left|\nabla G_k\left(\boldsymbol{z}_k^{j-1}\right)\right|\right|_{*(k,j)}^2.$$

Therefore $c_{(k,j)} = \tau\min\left\{1, \beta\gamma_{(k,j)}^2/L\right\} > 0$. ⊠

From Proposition 2.2.10, the sequence of cost function values, $\{G_k(\boldsymbol{z}_k^j)\}_{j=0}^{\infty}$, monotonically decreases, and when the gradient is large, we expect a large decrease in the cost function. Since $G_k(\boldsymbol{z}) \geq 0$ for all $\boldsymbol{z}$, we know the sequence $\{G_k(\boldsymbol{z}_k^j)\}_{j=0}^{\infty}$ converges by the monotone convergence theorem. That is, each steepest descent process updating $\boldsymbol{z}$ in CG-LS does converge.

Now, to show convergence of CG-LS to a stationary point of the CG-LS cost function (2.2), we apply Proposition 2.2.10 and similarly bound the change in the CG-LS cost function over an iteration of CG-LS as in the following new theorem.

Theorem 2.2.11. *Assume the Euclidean bound sequences, given by* $\{\gamma_{(k,1)}\}_{k=1}^{\infty}$ *and* $\{\gamma_{*(k,1)}\}_{k=1}^{\infty}$, *are bounded below by* $\gamma > 0$. *Then CG-LS converges to a zero gradient stationary point of* $F(\boldsymbol{u}, \boldsymbol{z})$

given in (2.2). Furthermore, the sequence of minimum gradient dual norms

$$\left\{ \min_{1\leq k\leq K} ||\nabla F(\boldsymbol{u}_k, \boldsymbol{z}_k)||^2_{*(k,1)} \right\}_{K=1}^{\infty}$$

satisfies

$$\min_{1\leq k\leq K} ||\nabla F(\boldsymbol{u}_k, \boldsymbol{z}_k)||^2_{*(k,1)} \leq \mathcal{O}\left(\frac{1}{K}\right).$$

Proof. For all $k \geq 1$, we have $\boldsymbol{z}_k^0 = \boldsymbol{z}_{k-1}$ and $\boldsymbol{z}_k^J = \boldsymbol{z}_k$. Thus, using a telescoping sum observe that

$$\begin{aligned} F(\boldsymbol{u}_{k-1}, \boldsymbol{z}_{k-1}) - F(\boldsymbol{u}_{k-1}, \boldsymbol{z}_k) &= F(\boldsymbol{u}_{k-1}, \boldsymbol{z}_k^0) - F(\boldsymbol{u}_{k-1}, \boldsymbol{z}_k^J) \\ &= \sum_{j=1}^{J} \left(F\left(\boldsymbol{u}_{k-1}, \boldsymbol{z}_k^{j-1}\right) - F\left(\boldsymbol{u}_{k-1}, \boldsymbol{z}_k^{j}\right) \right). \end{aligned}$$

Now, summing the conclusion of Proposition 2.2.10 over the J steepest descent steps produces

$$\begin{aligned} \sum_{j=1}^{J} \left(F\left(\boldsymbol{u}_{k-1}, \boldsymbol{z}_k^{j-1}\right) - F\left(\boldsymbol{u}_{k-1}, \boldsymbol{z}_k^{j}\right) \right) &\geq \sum_{j=1}^{J} c_{(k,j)} \left\| \nabla_{\boldsymbol{z}} F\left(\boldsymbol{u}_{k-1}, \boldsymbol{z}_k^{j-1}\right) \right\|^2_{*(k,j)} \\ &\geq c_{(k,1)} \left|\left| \nabla_{\boldsymbol{z}} F(\boldsymbol{u}_{k-1}, \boldsymbol{z}_{k-1}) \right|\right|^2_{*(k,1)}. \end{aligned}$$

Therefore, summing the telescoping series on the left hand side of the above inequality, we have

$$F(\boldsymbol{u}_{k-1}, \boldsymbol{z}_{k-1}) - F(\boldsymbol{u}_{k-1}, \boldsymbol{z}_k) \geq c_{(k,1)} \left|\left| \nabla_{\boldsymbol{z}} F(\boldsymbol{u}_{k-1}, \boldsymbol{z}_{k-1}) \right|\right|^2_{*(k,1)}. \tag{2.23}$$

First, note that $\nabla_{\boldsymbol{u}} F(\boldsymbol{u}_{k-1}, \boldsymbol{z}_{k-1}) = \mathbf{0}$ since $\boldsymbol{u}_{k-1}$ is a global minimizer of $F(\boldsymbol{u}, \boldsymbol{z})$ when $\boldsymbol{z}$ is fixed at $\boldsymbol{z}_{k-1}$. Hence, $||\nabla_{\boldsymbol{z}} F(\boldsymbol{u}_{k-1}, \boldsymbol{z}_{k-1})||^2_{*(k,1)} = ||\nabla F(\boldsymbol{u}_{k-1}, \boldsymbol{z}_{k-1})||^2_{*(k,1)}$. Second, note that $F(\boldsymbol{u}_k, \boldsymbol{z}_k) \leq F(\boldsymbol{u}_{k-1}, \boldsymbol{z}_k)$ since $\boldsymbol{u}_k$ is a global minimizer of $F(\boldsymbol{u}, \boldsymbol{z})$ when $\boldsymbol{z}$ is fixed at $\boldsymbol{z}_k$. Hence, $-F(\boldsymbol{u}_k, \boldsymbol{z}_k) \geq -F(\boldsymbol{u}_{k-1}, \boldsymbol{z}_k)$. Third, define

$$c \coloneqq \tau \min\left\{1, \beta\gamma^2/L\right\} > 0 \tag{2.24}$$

and note, since $\gamma_{(k,1)} \geq \gamma$, then

$$c_{(k,1)} = \tau \min\left\{1, \beta\gamma_{(k,1)}^2/L\right\} \geq c.$$

Combining these three results with (2.23) gives

$$F(\boldsymbol{u}_{k-1}, \boldsymbol{z}_{k-1}) - F(\boldsymbol{u}_k, \boldsymbol{z}_k) \geq c||\nabla F(\boldsymbol{u}_{k-1}, \boldsymbol{z}_{k-1})||_{*(k,1)}^2. \tag{2.25}$$

Thus, CG-LS generates a monotonic decreasing sequence of cost function values, which are bounded below by 0. Hence, $\{F(\boldsymbol{u}_k, \boldsymbol{z}_k)\}_{k=0}^{\infty}$ converges to a value F^* by the monotone convergence theorem. Hence, using a telescoping series, observe

$$\sum_{k=1}^{\infty} (F(\boldsymbol{u}_{k-1}, \boldsymbol{z}_{k-1}) - F(\boldsymbol{u}_k, \boldsymbol{z}_k)) = F(\boldsymbol{u}_0, \boldsymbol{z}_0) - \lim_{k\to\infty} F(\boldsymbol{u}_k, \boldsymbol{z}_k) = F(\boldsymbol{u}_0, \boldsymbol{z}_0) - F^*.$$

Now taking an infinite sum of (2.25) and using that $\|\cdot\|_{*(k,1)} \geq \gamma_{*(k,1)}\|\cdot\|)_2$ and $\gamma_{*(k,1)} \geq \gamma$ gives

$$\begin{aligned}
\sum_{k=1}^{\infty} (F(\boldsymbol{u}_{k-1}, \boldsymbol{z}_{k-1}) - F(\boldsymbol{u}_k, \boldsymbol{z}_k)) &\geq \sum_{k=1}^{\infty} c\|\nabla F(\boldsymbol{u}_{k-1}, \boldsymbol{z}_{k-1})\|_{*(k,1)}^2 \\
&\geq \gamma_{*(k,1)}^2 c \sum_{k=1}^{\infty} ||\nabla F(\boldsymbol{u}_{k-1}, \boldsymbol{z}_{k-1})||_2^2 \\
&\geq \gamma^2 c \sum_{k=1}^{\infty} ||\nabla F(\boldsymbol{u}_{k-1}, \boldsymbol{z}_{k-1})||_2^2.
\end{aligned}$$

Therefore

$$F(\boldsymbol{u}_0, \boldsymbol{z}_0) - F^* \geq \gamma^2 c \sum_{k=1}^{\infty} ||\nabla F(\boldsymbol{u}_{k-1}, \boldsymbol{z}_{k-1})||_2^2. \tag{2.26}$$

Equation (2.26) implies that $\sum_{k=0}^{\infty} ||\nabla F(\boldsymbol{u}_k, \boldsymbol{z}_k)||_2^2$ converges. Thus, $\lim_{k\to\infty} ||\nabla F(\boldsymbol{u}_k, \boldsymbol{z}_k)||_2^2 \to 0$ and so $\lim_{k\to\infty} \nabla F(\boldsymbol{u}_k, \boldsymbol{z}_k) \to \boldsymbol{0}$. Hence, since ∇F is a continuous function then $\lim_{k\to\infty}(\boldsymbol{u}_k, \boldsymbol{z}_k) \to (\boldsymbol{u}^*, \boldsymbol{z}^*)$ where $\nabla F(\boldsymbol{u}^*, \boldsymbol{z}^*) = \boldsymbol{0}$ implying that CG-LS converges to zero gradient stationary points.

Finally, taking an average of (2.25) over K iterations gives

$$\frac{F(\boldsymbol{u}_0, \boldsymbol{z}_0) - F(\boldsymbol{u}_K, \boldsymbol{z}_K)}{c} \frac{1}{K} \geq \min_{1 \leq k \leq K} ||\nabla F(\boldsymbol{u}_k, \boldsymbol{z}_K)||^2_{*(k,1)}. \qquad \boxtimes$$

Theorem 2.2.11 shows that CG-LS generates a monotonic decreasing sequence of cost function values, $\{F(\boldsymbol{u}_k, \boldsymbol{z}_k)\}_{k=0}^{\infty}$. Combining Theorem 2.2.11 with the fact that the CG-LS cost function (2.2) has no local maxima, then CG-LS, in Algorithm 2, is guaranteed convergence to a local minimum or a saddle point. As (2.2) is, in general, not convex, we cannot guarantee that Algorithm 2 converges to a local nor global minimum. However, if Theorem 2.2.6 is also satisfied, we guarantee convergence to a global minimizer of (2.2) relative to the interval $\boldsymbol{z} \in [z_0, b]^n$

Lastly, we further strengthen our convergence results when Theorem 2.2.6 is satisfied through the following new theorem.

Theorem 2.2.12. *Let Theorem 2.2.6 be satisfied with local minimizer $[\boldsymbol{u}^*, \boldsymbol{z}^*]$ corresponding to minimum value F^*. Then there exists a region $\mathcal{C} \subseteq \mathbb{R}^n \times \mathcal{Z}^n$ such that $[\boldsymbol{u}^*, \boldsymbol{z}^*] \in \mathcal{C}$ and $F(\boldsymbol{u}, \boldsymbol{z})$, given in (2.2), is strongly convex on $\mathcal{C}$ with constant ℓ. Furthermore, for any $[\boldsymbol{u}_0, \boldsymbol{z}_0] \in \mathcal{C}$ we have for all $k \geq 0$*

$$F(\boldsymbol{u}_k, \boldsymbol{z}_k) - F^* \leq (1 - 2\ell\gamma^2 c)^k (F(\boldsymbol{u}_0, \boldsymbol{z}_0) - F^*) \tag{2.27}$$

where $c > 0$ is given in equation (2.24) and $\gamma > 0$ is as given in Theorem 2.2.11

Proof. For $\delta > 0$, $\boldsymbol{u}' \in \mathbb{R}^n$, and $\boldsymbol{z}' \in \mathcal{Z}^n$ define

$$\mathcal{B}_\delta(\boldsymbol{u}', \boldsymbol{z}') = \{(\boldsymbol{u}, \boldsymbol{z}) \in \mathbb{R}^n \times \mathcal{Z}^n : \|(\boldsymbol{u}, \boldsymbol{z}) - (\boldsymbol{u}', \boldsymbol{z}')\|_2 < \delta\}.$$

Note that the non-degenerate local minimizer $(\boldsymbol{u}^*, \boldsymbol{z}^*)$ in Theorem 2.2.6 satisfies, for all $\boldsymbol{w} \in \mathbb{R}^{2n}$, $\boldsymbol{w}^T H_F(\boldsymbol{u}^*, \boldsymbol{z}^*)\boldsymbol{w} = \ell_0(\boldsymbol{w}) \geq \lambda_{\min} > 0$ where $\lambda_{\min}$ is the minimum eigenvalue of the Hessian $H_F(\boldsymbol{u}^*, \boldsymbol{z}^*)$, for $H_F(\boldsymbol{u}, \boldsymbol{z})$ is given in (2.11). Since $F(\boldsymbol{u}, \boldsymbol{z})$ is twice continuously differentiable then, for fixed $\boldsymbol{w} \in \mathbb{R}^{2n}$, $Q(\boldsymbol{u}, \boldsymbol{z}) \coloneqq \boldsymbol{w}^T H_F(\boldsymbol{u}, \boldsymbol{z})\boldsymbol{w}$ is a continuous function. Fix $\ell \in (0, \lambda_{\min})$ and

let $\epsilon = \lambda_{\min} - \ell > 0$. By continuity of Q there exists a $\delta > 0$ such that for all $(\boldsymbol{u}, \boldsymbol{z}) \in \mathcal{B}_\delta(\boldsymbol{u}^*, \boldsymbol{z}^*)$ we have $|Q(\boldsymbol{u}, \boldsymbol{z}) - Q(\boldsymbol{u}^*, \boldsymbol{z}^*)| < \epsilon = \lambda_{\min} - \ell$, which implies $Q(\boldsymbol{u}, \boldsymbol{z}) \geq \ell$ since $Q(\boldsymbol{u}^*, \boldsymbol{z}^*) \geq \lambda_{\min}$. Therefore, $F(\boldsymbol{u}, \boldsymbol{z})$ is strongly convex on $\mathcal{C} = \mathcal{B}_\delta(\boldsymbol{u}^*, \boldsymbol{z}^*)$ with constant ℓ.

Next, strong convexity of $F(\boldsymbol{u}, \boldsymbol{z})$ on $\mathcal{C}$, with constant ℓ, and $(\boldsymbol{u}_0, \boldsymbol{z}_0) \in \mathcal{C}$ implies that for all $k \geq 1$

$$F(\boldsymbol{u}_{k-1}, \boldsymbol{z}_{k-1}) - F^* \leq \frac{1}{2\ell} \|\nabla F(\boldsymbol{u}_{k-1}, \boldsymbol{z}_{k-1})\|_2^2. \tag{2.28}$$

From (2.25), using $\|\cdot\|_{*(k,1)} \geq \gamma \|\cdot\|_2$ observe

$$\begin{aligned} F(\boldsymbol{u}_{k-1}, \boldsymbol{z}_{k-1}) - F(\boldsymbol{u}_k, \boldsymbol{z}_k) &\geq c||\nabla F(\boldsymbol{u}_{k-1}, \boldsymbol{z}_{k-1})||_{*(k,1)}^2 \\ F(\boldsymbol{u}_{k-1}, \boldsymbol{z}_{k-1}) - F(\boldsymbol{u}_k, \boldsymbol{z}_k) - F^* &\geq \gamma^2 c||\nabla F(\boldsymbol{u}_{k-1}, \boldsymbol{z}_{k-1})||_2^2 - F^* \\ F(\boldsymbol{u}_{k-1}, \boldsymbol{z}_{k-1}) - F^* &\geq F(\boldsymbol{u}_k, \boldsymbol{z}_k) - F^* + \gamma^2 c||\nabla F(\boldsymbol{u}_{k-1}, \boldsymbol{z}_{k-1})||_2^2. \end{aligned}$$

Combining with (2.28) produces

$$\begin{aligned} F(\boldsymbol{u}_{k-1}, \boldsymbol{z}_{k-1}) - F^* &\geq F(\boldsymbol{u}_k, \boldsymbol{z}_k) - F^* + 2\ell\gamma^2 c(F(\boldsymbol{u}_{k-1}, \boldsymbol{z}_{k-1}) - F^*) \\ (1 - 2\ell\gamma^2 c)(F(\boldsymbol{u}_{k-1}, \boldsymbol{z}_{k-1}) - F^*) &\geq F(\boldsymbol{u}_k, \boldsymbol{z}_k) - F^*. \end{aligned} \tag{2.29}$$

Applying (2.29) recursively gives $F(\boldsymbol{u}_k, \boldsymbol{z}_k) - F^* \leq (1 - 2\ell\gamma^2 c)^k (F(\boldsymbol{u}_0, \boldsymbol{z}_0) - F^*)$. ⊠

Theorem 2.2.12 shows that there exists a region around the local minimizer where (2.2) is strongly convex and thus we guarantee a linear rate of convergence of the cost function values within this region.

2.2.4 Numerical Results

We test the CG-LS algorithm using gradient descent, which we denote by gCG-LS, and Newton descent, which we denote by nCG-LS, as the steepest descent method for updating $\boldsymbol{z}$. That is,

$$\boldsymbol{d}_k^j(\boldsymbol{z}) = \begin{cases} -\nabla_{\boldsymbol{z}} F(\boldsymbol{u}_{k-1}, \boldsymbol{z}) & \text{gCG-LS} \\ -(\boldsymbol{J}_{\boldsymbol{z}}(\nabla_{\boldsymbol{z}} F(\boldsymbol{u}_{k-1}, \boldsymbol{z})))^{-1} \nabla_{\boldsymbol{z}} F(\boldsymbol{u}_{k-1}, \boldsymbol{z}) & \text{nCG-LS} \end{cases}$$

where $\nabla_{\boldsymbol{z}} F$ and $\boldsymbol{J}_{\boldsymbol{z}}(\nabla_{\boldsymbol{z}} F)$ are given in (2.10) and (2.11), respectively.

Using nCG-LS requires $\boldsymbol{J}_{\boldsymbol{z}}(\nabla_{\boldsymbol{z}} F(\boldsymbol{u}_{k-1}, \boldsymbol{z}))$ to be a positive definite matrix for any $\boldsymbol{z}$ in the optimization domain of interest. Assume the chosen $f(z)$ satisfies the requirements of Theorem 2.2.6 for interval $[a, b]$ and minimizer z_0. Then $\boldsymbol{J}_{\boldsymbol{z}}(\nabla_{\boldsymbol{z}} F(\boldsymbol{u}_{k-1}, \boldsymbol{z}))$ is positive definite for all $\boldsymbol{z} \in [a, b]^n$ and, informed by Theorem 2.2.6, we guarantee a local minimizer lies in $[z_0, b]^n \subset [a, b]^n$ under sufficient scaling of the input data. Therefore, we can properly scale the input measurements to nCG-LS and apply the mReLU function $\mathcal{P}_{z_0,b}$ at each $\boldsymbol{z}$ update in line 10 of Algorithm 2 to ensure $\boldsymbol{J}_{\boldsymbol{z}}(\nabla_{\boldsymbol{z}} F(\boldsymbol{u}_{k-1}, \boldsymbol{z}))$ is positive definite. Alternatively, or in addition to scaling the input data, to find $\boldsymbol{z}_k^j$ we may perform an eigendecomposition on $\boldsymbol{J}_{\boldsymbol{z}}(\nabla_{\boldsymbol{z}} F(\boldsymbol{u}_{k-1}, \boldsymbol{z}_k^{j-1}))$ to find the closest positive definite matrix that is then used in the Newton descent step. That is, let Λ_k^j, Q_k^j be the eigendecomposition of $\boldsymbol{J}_{\boldsymbol{z}}(\nabla_{\boldsymbol{z}} F(\boldsymbol{u}_{k-1}, \boldsymbol{z}_k^{j-1}))$ satisfying $\boldsymbol{J}_{\boldsymbol{z}}(\nabla_{\boldsymbol{z}} F(\boldsymbol{u}_{k-1}, \boldsymbol{z}_k^{j-1})) = Q_k^j \Lambda_k^j (Q_k^j)^T$ for Λ_k^j a diagonal matrix. Then instead of using $\boldsymbol{J}_{\boldsymbol{z}}(\nabla_{\boldsymbol{z}} F(\boldsymbol{u}_{k-1}, \boldsymbol{z}_k^{j-1}))$ to find the Newton descent step we use $Q_k^j (\Lambda_k^j)^{\epsilon+} (Q_k^j)^T$ where, for a small positive parameters $\epsilon > 0$, the diagonal matrix $(\Lambda_k^j)^{\epsilon+}$ is defined as $[(\Lambda_k^j)^{\epsilon+}]_{ii} = \max\{[\Lambda_k^j]_{ii}, \epsilon\}$. We remark that $Q_k^j (\Lambda_k^j)^{\epsilon+} (Q_k^j)^T$ is the closest matrix with a minimum eigenvalue of ϵ to $\boldsymbol{J}_{\boldsymbol{z}}(\nabla_{\boldsymbol{z}} F(\boldsymbol{u}_{k-1}, \boldsymbol{z}_k^{j-1}))$ as measured by the Frobenius norm.

For numerical evaluation, we test CG-LS on the linear inverse problems of image reconstruction from tomographic and compressed sensing measurements. The datasets we employ include: 32×32 images from the CIFAR10 image dataset [69], 64×64 images from the CalTech101 image dataset [70], and 128×128 images from the Set11 image dataset [22]. Each image from every dataset has been converted to a single-channel grayscale image, scaled down by the maximum pixel value and vectorized. For tomographic measurements, a Radon transform, at a specified number of

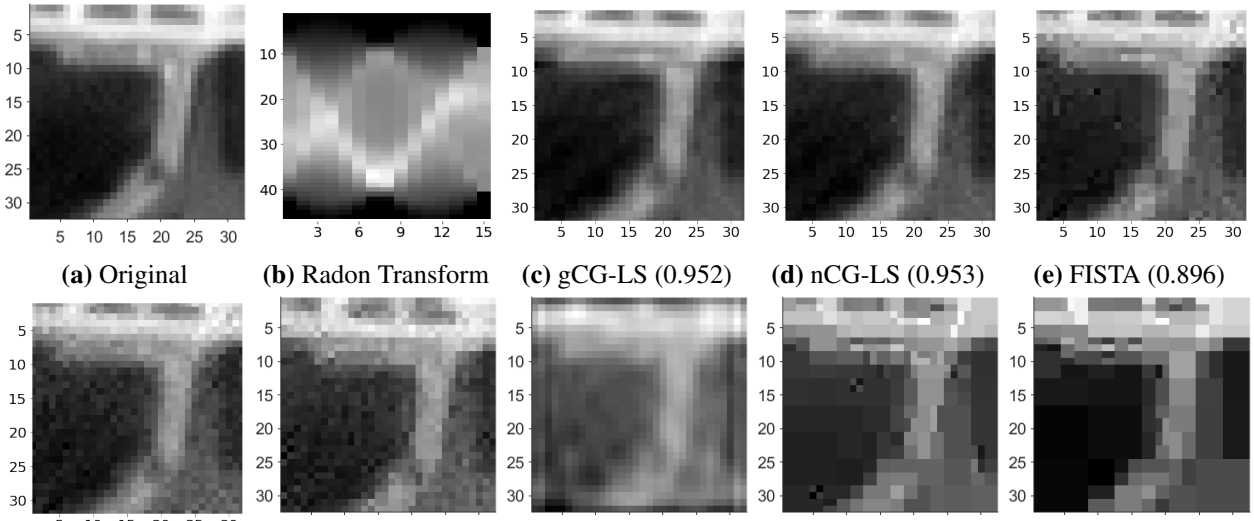
angles, is performed on each image. Instead, for compressed sensing measurements a fixed random Gaussian matrix is applied to each vectorized image. After applying the measurement matrix to each image, white noise is added producing noisy measurements, $\boldsymbol{y}$, at a specified signal-to-noise ratio (SNR). Finally, a biorthogonal wavelet transformation is applied to each image to produce the sparsity coefficients, $\boldsymbol{c}$.

We compare CG-LS against six prior art iterative algorithms for solving linear inverse problems: Fast Iterative Shrinkage and Thresholding Algorithm (FISTA) [2], ℓ_1-least squares (ℓ_1-LS) [6], Fourier backprojection (FBP) [71], Bayesian compressive sensing (BCS) [3], Compressive Sampling Matching Pursuit (CoSaMP) [4], and Kalman Filtering (KF) [72]. Note, FISTA, ℓ_1-LS, BCS, and CoSaMP are sparsity based approaches to signal reconstruction. For signal prior FISTA and ℓ_1-LS assume a Laplace distribution and BCS assumes a student's t distribution. FBP is a method specifically for Radon inversion and dependent on the structure of the sinogram measurements produced by the Radon transform. The hyperparameters for each prior art method were chosen to maximize each algorithms performance.

For reconstructing 32×32 images we use $f(z) = \ln(z)$, $\mathcal{Z} = (0, \infty)$, $\mu = 2$, $K = 1000$, $J = 1$, and $\delta = 10^{-6}$. For tomographic measurements at an SNR of 60dB and 40dB, we take $\lambda = 0.3$ and $\lambda = 2$, respectively. For compressed sensing measurements at an SNR of 60dB we set $\lambda = 150$. Using nCG-LS requires $\boldsymbol{J}_{\boldsymbol{z}}(\nabla_{\boldsymbol{z}} F(\boldsymbol{u}_{k-1}, \boldsymbol{z}))$ to be positive definite in the $\boldsymbol{z}$ variable, which for $f(z) = \ln(z)$, is guaranteed when $\boldsymbol{z} \in (0, e)^n$. Informed by Theorem 2.2.6, we guarantee a local minimizer lies in $[1, e)^n$ under sufficient scaling of the input data. Therefore, in each nCG-LS test in reconstructing 32×32 images, we scale the input measurement by a factor, chosen empirically, of e^{-4}. Additionally, we use an eigendecomposition on $\boldsymbol{J}_{\boldsymbol{z}}(\nabla_{\boldsymbol{z}} F(\boldsymbol{u}_{k-1}, \boldsymbol{z}))$ to find the closest positive semi-definite matrix that is then used in the Newton descent step. Alternatively, we remark that the mReLU function $\mathcal{P}_{1,e}$ may be applied at each $\boldsymbol{z}$ update in line 10 of Algorithm 2 to ensure $\boldsymbol{J}_{\boldsymbol{z}}(\nabla_{\boldsymbol{z}} F(\boldsymbol{u}_{k-1}, \boldsymbol{z}))$ is positive semi-definite. Finally, for the initial $\boldsymbol{z}$ estimate, we choose $\mathcal{P}_{a,b} = \mathcal{P}_{1,e}$ for nCG-LS whereas for gCG-LS we use $\mathcal{P}_{a,b} = \mathcal{P}_{1,e^2}$.

A visualized reconstruction of a 32×32 Barbara snippet, from a 60dB SNR Radon transform at 15 uniformly spaced angles, is shown in Figure 2.1. We see by visual inspection and SSIM that both gCG-LS and nCG-LS produce superior reconstructions to the other iterative algorithms with nCG-LS slightly outperforming gCG-LS. Table 2.1 provides the average image reconstruction SSIM and PSNR along with 99% confidence intervals for our gCG-LS, our nCG-LS, and each of the six comparative methods when every method reconstructs 200, 32×32 CIFAR10 images from Radon transform measurements. Note, larger SSIM and PSNR values correspond to image reconstructions that are visually closer to the original image of interest. Thus, as seen in Table 2.1, CG-LS performs best in all noise and measurement scenarios with nCG-LS slightly outperforming gCG-LS.

The superiority of CG-LS in 32×32 image reconstruction is further highlighted in Table 2.2. Table 2.2 displays the average SSIM and PSNR, with 99% confidence intervals, from the reconstruction 200, 32×32 CIFAR10 images from random Gaussian measurements. That is, the measurement matrix $\Psi \in \mathbb{R}^{m \times n}$ has entries drawn i.i.d. from a Gaussian distribution, which is typical



(a) Original **(b)** Radon Transform **(c)** gCG-LS (0.952) **(d)** nCG-LS (0.953) **(e)** FISTA (0.896)

(f) KF (0.887) **(g)** ℓ_1-LS (0.849) **(h)** FBP (0.844) **(i)** BCS (0.827) **(j)** CoSaMP (0.838)

Figure 2.1: Image reconstructions, with SSIM value in parenthesis, using our gCG-LS, our nCG-LS, FISTA, KF, ℓ_1-LS, FBP, BCS, and CoSaMP. The input to each algorithm is a vectorized (2.1b), which is a Radon transform of (2.1a) at 15 uniformly spaced angles with an SNR of 60dB. We observe that CG-LS outperforms all methods with nCG-LS slightly outperforming gCG-LS.

Table 2.1: Average SSIM ($\times 10^2$) and PSNR with 99% confidence intervals for our gCG-LS and nCG-LS and six comparative methods. Each algorithm estimated two hundred, 32×32 CIFAR10 images from Radon transform measurements taken at 15, 10, or 6 uniformly spaced angles. Each measurement has an SNR of 60dB or 40dB. We find that CG-LS performs best in all noise and sparse scenarios with nCG-LS, in **bold**, slightly outperforming gCG-LS, in *italics*.

Angles	15			
SNR	60 dB		40 dB	
	SSIM ($\times 10^2$)	PSNR	SSIM ($\times 10^2$)	PSNR
nCG-LS	$\mathbf{91.3 \pm .59}$	$\mathbf{28.0 \pm .39}$	$\mathbf{87.0 \pm .97}$	$\mathbf{26.2 \pm .28}$
gCG-LS	$\mathit{90.1 \pm .57}$	$\mathit{27.1 \pm .38}$	$\mathit{85.9 \pm .93}$	$\mathit{25.5 \pm .28}$
KF	$89.8 \pm .67$	$27.9 \pm .34$	85.2 ± 1.0	$26.0 \pm .26$
FISTA	86.6 ± 1.0	$26.6 \pm .29$	78.6 ± 1.1	$23.6 \pm .27$
ℓ_1-LS	$86.6 \pm .58$	$26.0 \pm .42$	72.7 ± 1.6	$22.1 \pm .35$
FBP	$85.7 \pm .77$	$20.6 \pm .36$	81.2 ± 1.2	$20.2 \pm .35$
BCS	$77.4 \pm .96$	$22.7 \pm .43$	$74.1 \pm .93$	$22.1 \pm .34$
CoSaMP	75.9 ± 1.0	$22.4 \pm .48$	$71.2 \pm .98$	$21.3 \pm .34$

Angles	10			
SNR	60 dB		40 dB	
	SSIM ($\times 10^2$)	PSNR	SSIM ($\times 10^2$)	PSNR
nCG-LS	$\mathbf{83.6 \pm .82}$	$\mathbf{24.8 \pm .39}$	$\mathbf{80.6 \pm .94}$	$\mathbf{24.0 \pm .32}$
gCG-LS	$\mathit{82.3 \pm .78}$	$\mathit{24.3 \pm .40}$	$\mathit{80.1 \pm .86}$	$\mathit{23.8 \pm .34}$
KF	$81.9 \pm .84$	$24.9 \pm .39$	$78.0 \pm .95$	$23.9 \pm .31$
FISTA	$81.4 \pm .89$	$24.8 \pm .39$	$69.4 \pm .97$	$21.7 \pm .31$
ℓ_1-LS	$74.5 \pm .93$	$22.8 \pm .42$	67.2 ± 1.1	$21.2 \pm .29$
FBP	72.4 ± 1.1	$14.5 \pm .42$	68.4 ± 1.3	$14.3 \pm .43$
BCS	62.4 ± 1.2	$19.9 \pm .44$	60.4 ± 1.2	$19.6 \pm .42$
CoSaMP	61.7 ± 1.2	$19.5 \pm .44$	58.6 ± 1.1	$18.9 \pm .36$

Angles	6			
SNR	60 dB		40 dB	
	SSIM ($\times 10^2$)	PSNR	SSIM ($\times 10^2$)	PSNR
nCG-LS	$\mathbf{70.0 \pm 1.1}$	$\mathbf{21.7 \pm .37}$	$\mathbf{68.0 \pm 1.2}$	$\mathbf{21.3 \pm .35}$
gCG-LS	$\mathit{68.9 \pm 1.1}$	$\mathit{21.4 \pm .42}$	$\mathit{67.9 \pm 1.1}$	$\mathit{21.3 \pm .40}$
KF	67.9 ± 1.2	$21.8 \pm .38$	64.6 ± 1.2	$21.3 \pm .34$
FISTA	67.9 ± 1.2	$21.4 \pm .39$	53.3 ± 1.2	$19.1 \pm .38$
ℓ_1-LS	55.2 ± 1.3	$19.4 \pm .42$	52.5 ± 1.2	$18.9 \pm .36$
FBP	53.4 ± 1.1	$18.8 \pm .46$	50.2 ± 1.2	$18.7 \pm .45$
BCS	44.6 ± 1.5	$17.3 \pm .45$	44.1 ± 1.5	$17.2 \pm .44$
CoSaMP	35.5 ± 1.5	$14.8 \pm .48$	32.4 ± 1.3	$14.2 \pm .43$

in compressed sensing applications. The reconstructions in Table 2.2 are from measurements at a compressed or sampling ratio – defined as the fraction of the measurement matrix dimensions $\frac{m}{n}$ –

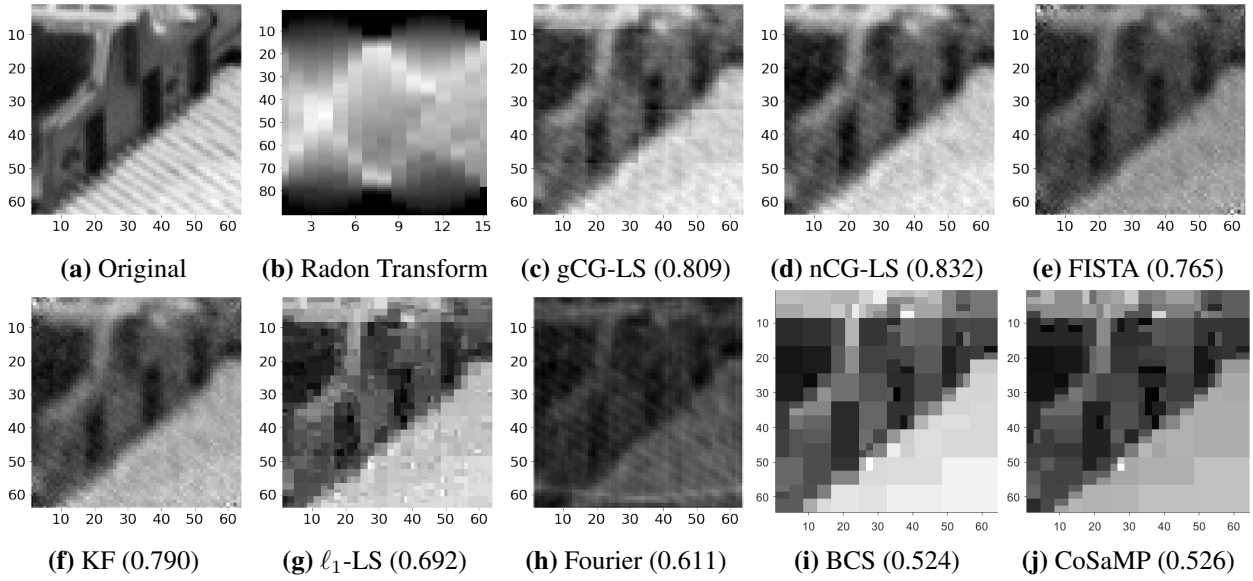
of $0.5, 0.3,$ or 0.1. We observe that nCG-LS and gCG-LS outperform all five comparison methods for each sampling ratio. Furthermore, while Kalman Filtering was comparable to prior art in the tomographic imaging problem of Radon inversion, it performs underperforms appreciably in these compressive sensing experiments. Thus, a thorough examination of Kalman based approaches to general linear inverse problems is a subject of future work.

Table 2.2: Average SSIM ($\times 10^2$) and PSNR with 99% confidence intervals for our gCG-LS and nCG-LS and five comparative iterative compressive sensing methods. Each algorithm estimated two hundred, 32×32 CIFAR10 [69] images from random Gaussian measurements taken at a sampling ratio of 0.5, 0.3, or 0.1. Each measurement has an SNR of 60dB. We find that CG-LS performs best in all noise and sparse scenarios where nCG-LS is in **bold** and gCG-LS is in *italics*.

Ratio	0.5		0.3		0.1	
SNR	60 dB		60 dB		60 dB	
	SSIM ($\times 10^2$)	PSNR	SSIM ($\times 10^2$)	PSNR	SSIM ($\times 10^2$)	PSNR
nCG-LS	$\mathbf{76.2 \pm .87}$	$\mathbf{23.1 \pm .40}$	$\mathbf{57.1 \pm 1.2}$	$\mathbf{20.0 \pm .43}$	$\mathbf{25.2 \pm 1.4}$	$\mathbf{15.3 \pm .43}$
gCG-LS	$\mathit{76.9 \pm .91}$	$\mathit{23.3 \pm .42}$	$\mathit{57.5 \pm 1.3}$	$\mathit{20.0 \pm .46}$	$\mathit{24.7 \pm 1.5}$	$\mathit{15.1 \pm .44}$
KF	$12.4 \pm .96$	$7.62 \pm .29$	6.80 ± 0.56	$6.68 \pm .34$	$1.57 \pm .14$	$5.66 \pm .31$
FISTA	75.1 ± 1.1	$22.8 \pm .44$	54.2 ± 1.3	$19.4 \pm .46$	20.6 ± 0.97	$14.2 \pm .40$
ℓ_1-LS	74.6 ± 1.5	$22.7 \pm .58$	52.4 ± 2.0	$18.8 \pm .85$	21.4 ± 1.0	$14.6 \pm .36$
BCS	72.4 ± 1.6	$22.3 \pm .56$	53.0 ± 1.9	$19.1 \pm .56$	23.5 ± 1.6	$14.8 \pm .43$
CoSaMP	69.0 ± 1.7	$21.3 \pm .56$	49.4 ± 1.9	18.4 ± 0.58	0.038 ± 1.4	$1.0 \pm .95$

For reconstructing 64×64 images we use the same parameter setup for CG-LS as in the 32×32 image reconstructions except that in each nCG-LS test the input measurements are scaled by a factor, chosen empirically, of e^{-6}. A visualized reconstruction of a 64×64 Barbara snippet, from a 60dB SNR Radon transform at 15 uniformly spaced angles, is shown in Figure 2.2. We see by visual inspection and SSIM that both gCG-LS and nCG-LS produce superior reconstructions to the other iterative algorithms with nCG-LS slightly outperforming gCG-LS.

In Table 2.3 is the average image reconstruction SSIM and PSNR along with 99% confidence intervals for our gCG-LS, our nCG-LS, and each of the six comparative iterative methods when every method reconstructs 200, 64×64 CalTech101 images from Radon transform measurements. As seen in Table 2.3, CG-LS performs best in all noise and measurement scenarios with nCG-LS slightly outperforming gCG-LS.



(a) Original **(b)** Radon Transform **(c)** gCG-LS (0.809) **(d)** nCG-LS (0.832) **(e)** FISTA (0.765)

(f) KF (0.790) **(g)** ℓ_1-LS (0.692) **(h)** Fourier (0.611) **(i)** BCS (0.524) **(j)** CoSaMP (0.526)

Figure 2.2: Image reconstructions, with SSIM value in parenthesis, using our gCG-LS, nCG-LS, FISTA, KF, ℓ_1-LS, Fourier backprojection, Bayesian Compressive Sensing, and CoSaMP. The input to each algorithm is a vectorized (2.2b), which is the Radon transform of the original 64×64 image, (2.2a), at 15 uniformly spaced angles with a noise level of 60dB SNR. We observe that CG-LS outperforms all methods with nCG-LS slightly outperforming gCG-LS.

For reconstructing 128×128 images we use $K = 100$ and $J = 1$ as the maximum iteration choices for CG-LS. With tomographic measurements at an SNR of 60dB and 40dB, we take $\lambda = 0.3$ and $\lambda = 30$, respectively. Both the maximum number of iterations and the regularization parameter λ, for reconstructing 128×128 images, where chosen empirically to maximize the trade-off between CG-LS performance and required computational time. All other parameters are setup exactly the same as in the case of reconstructing 64×64 images with CG-LS.

Table 2.4 displays the average reconstructed image SSIM and PSNR quality along with 99% confidence intervals for our gCG-LS, our nCG-LS, and five of the comparative methods when every method reconstructs all Set11, 128×128 images from Radon transform measurements. As seen in Table 2.4, CG-LS performs best, or comparably, in all noise and measurement scenarios with nCG-LS slightly outperforming gCG-LS. A visualized reconstruction of a 128×128 boat image, from a 60dB SNR Radon transform at 15 uniformly spaced angles, is shown in Figure 2.3. We see by visual inspection and SSIM that both gCG-LS and nCG-LS produce superior, or

Table 2.3: Average SSIM ($\times 10^2$) and PSNR with 99% confidence intervals for our gCG-LS and nCG-LS and six comparative methods. Each algorithm estimated two hundred, 64×64 CalTech101 images from Radon transform measurements taken at 15, 10, or 6 uniformly spaced angles. Each measurement has an SNR of 60dB or 40dB. We find that CG-LS performs best in all noise and sparse scenarios with nCG-LS, in **bold**, slightly outperforming gCG-LS, in *italics*.

Angles	15			
SNR	60 dB		40 dB	
	SSIM ($\times 10^2$)	PSNR	SSIM ($\times 10^2$)	PSNR
nCG-LS	$\mathbf{67.5 \pm 1.2}$	$\mathbf{22.7 \pm .45}$	$\mathbf{63.2 \pm 1.4}$	$\mathbf{22.0 \pm .41}$
gCG-LS	$\mathit{65.5 \pm 1.1}$	$\mathit{22.0 \pm .43}$	$\mathit{61.3 \pm 1.3}$	$\mathit{21.4 \pm .39}$
KF	64.5 ± 1.2	$22.7 \pm .45$	59.7 ± 1.3	$21.9 \pm .40$
FISTA	63.5 ± 1.2	$22.6 \pm .44$	51.2 ± 1.3	$19.8 \pm .40$
ℓ_1-LS	61.6 ± 1.3	$21.5 \pm .49$	45.3 ± 1.7	$18.3 \pm .46$
FBP	54.6 ± 1.3	$15.3 \pm .57$	48.1 ± 1.4	$14.9 \pm .56$
BCS	54.3 ± 1.9	$19.4 \pm .49$	52.3 ± 1.8	$19.1 \pm .47$
CoSaMP	54.2 ± 2.0	$19.2 \pm .49$	51.5 ± 1.7	$18.6 \pm .45$

Angles	10			
SNR	60 dB		40 dB	
	SSIM ($\times 10^2$)	PSNR	SSIM ($\times 10^2$)	PSNR
nCG-LS	$\mathbf{58.3 \pm 1.3}$	$\mathbf{21.0 \pm .43}$	$\mathbf{56.0 \pm 1.3}$	$\mathbf{20.7 \pm .41}$
gCG-LS	$\mathit{56.9 \pm 1.2}$	$\mathit{20.5 \pm .44}$	$\mathit{54.4 \pm 1.2}$	$\mathit{20.3 \pm .42}$
KF	56.6 ± 1.3	$21.1 \pm .44$	53.1 ± 1.2	$20.7 \pm .41$
FISTA	56.5 ± 1.3	$21.1 \pm .44$	42.9 ± 1.0	$18.5 \pm .39$
ℓ_1-LS	48.7 ± 1.4	$19.4 \pm .46$	40.7 ± 1.1	$18.0 \pm .40$
FBP	41.5 ± 1.1	$9.33 \pm .66$	36.2 ± 1.1	$9.13 \pm .65$
BCS	40.7 ± 1.9	$17.2 \pm .47$	39.6 ± 1.9	$17.0 \pm .46$
CoSaMP	41.1 ± 2.1	$16.8 \pm .48$	40.0 ± 1.9	$16.6 \pm .46$

Angles	6			
SNR	60 dB		40 dB	
	SSIM ($\times 10^2$)	PSNR	SSIM ($\times 10^2$)	PSNR
nCG-LS	$\mathbf{46.8 \pm 1.3}$	$\mathbf{19.0 \pm .42}$	$\mathbf{45.7 \pm 1.3}$	$\mathbf{18.9 \pm .42}$
gCG-LS	$\mathit{45.4 \pm 1.4}$	$\mathit{18.5 \pm .47}$	$\mathit{44.0 \pm 1.2}$	$\mathit{18.4 \pm .46}$
KF	47.2 ± 1.5	$19.2 \pm .44$	43.7 ± 1.3	$18.9 \pm .42$
FISTA	45.5 ± 1.5	$19.1 \pm .44$	31.1 ± 1.0	$16.5 \pm .40$
ℓ_1-LS	33.7 ± 1.4	$16.7 \pm .44$	30.0 ± 1.0	$16.2 \pm .43$
FBP	$27.6 \pm .81$	$14.4 \pm .71$	$23.6 \pm .76$	$14.1 \pm .70$
BCS	29.1 ± 1.9	$15.2 \pm .49$	28.9 ± 1.9	$15.2 \pm .48$
CoSaMP	24.8 ± 1.6	$13.5 \pm .47$	24.3 ± 1.5	$13.4 \pm .46$

comparable, reconstructions to the other iterative algorithms with nCG-LS slightly outperforming gCG-LS.

Table 2.4: Average SSIM ($\times 10^2$) and PSNR with 99% confidence intervals for our gCG-LS and nCG-LS and five comparative methods. Each algorithm estimated all 11, 128×128 Set11 images from Radon transform measurements taken at 15, 10, or 6 uniformly spaced angles. Each measurement has an SNR of 60dB or 40dB. We find that CG-LS performs best, or comparably, in all noise and sparse scenarios with nCG-LS, in **bold**, slightly outperforming gCG-LS, in *italics*.

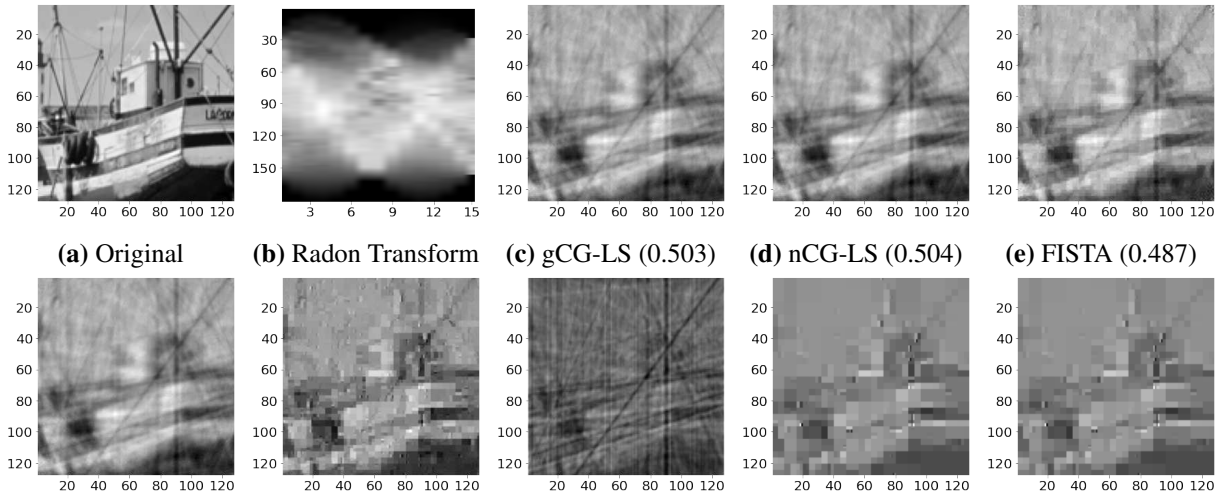
Angles	15			
SNR	60 dB		40 dB	
	SSIM ($\times 10^2$)	PSNR	SSIM ($\times 10^2$)	PSNR
nCG-LS	$\mathbf{51.3 \pm 7.7}$	$\mathbf{20.7 \pm 2.0}$	$\mathbf{46.9 \pm 7.6}$	$\mathbf{19.9 \pm 1.7}$
gCG-LS	$\mathit{51.3 \pm 7.8}$	$\mathit{20.7 \pm 2.0}$	$\mathit{46.9 \pm 7.6}$	$\mathit{19.9 \pm 1.7}$
FISTA	50.9 ± 7.4	20.6 ± 1.9	33.0 ± 3.9	17.4 ± 1.3
ℓ_1-LS	42.6 ± 7.3	19.2 ± 1.9	32.9 ± 4.0	17.4 ± 1.3
FBP	35.3 ± 4.0	$11.2 \pm .69$	27.4 ± 2.9	$10.6 \pm .62$
BCS	39.1 ± 11	17.7 ± 2.1	37.6 ± 11	17.5 ± 1.8
CoSaMP	38.7 ± 11	17.3 ± 2.0	38.0 ± 11	17.2 ± 1.9

Angles	10			
SNR	60 dB		40 dB	
	SSIM ($\times 10^2$)	PSNR	SSIM ($\times 10^2$)	PSNR
nCG-LS	$\mathbf{44.3 \pm 7.4}$	$\mathbf{19.2 \pm 1.5}$	$\mathbf{40.4 \pm 7.1}$	$\mathbf{18.6 \pm 1.3}$
gCG-LS	$\mathit{44.3 \pm 7.4}$	$\mathit{19.2 \pm 1.5}$	$\mathit{40.3 \pm 7.1}$	$\mathit{18.5 \pm 1.4}$
FISTA	43.4 ± 7.1	19.1 ± 1.5	27.8 ± 4.0	16.6 ± 1.2
ℓ_1-LS	33.5 ± 6.8	17.5 ± 1.6	27.3 ± 4.1	16.5 ± 1.3
FBP	25.0 ± 3.2	$5.46 \pm .55$	18.7 ± 2.0	$5.23 \pm .55$
BCS	31.7 ± 10	16.3 ± 1.4	31.4 ± 9.7	16.3 ± 1.4
CoSaMP	32.2 ± 10	15.6 ± 1.3	32.0 ± 10	15.6 ± 1.4

Angles	6			
SNR	60 dB		40 dB	
	SSIM ($\times 10^2$)	PSNR	SSIM ($\times 10^2$)	PSNR
nCG-LS	$\mathbf{37.8 \pm 7.9}$	$\mathbf{17.8 \pm 1.4}$	$\mathbf{35.1 \pm 6.7}$	$\mathbf{17.6 \pm 1.3}$
gCG-LS	$\mathit{37.8 \pm 7.9}$	$\mathit{17.8 \pm 1.4}$	$\mathit{35.1 \pm 6.7}$	$\mathit{17.6 \pm 1.3}$
FISTA	37.3 ± 8.3	17.8 ± 1.4	26.4 ± 4.3	$16.0 \pm .97$
ℓ_1-LS	25.6 ± 6.4	15.7 ± 1.4	21.7 ± 4.5	15.1 ± 1.2
FBP	15.7 ± 2.5	$0.85 \pm .59$	11.2 ± 1.2	$0.68 \pm .59$
BCS	28.7 ± 10	15.1 ± 1.1	28.5 ± 10	15.1 ± 1.1
CoSaMP	28.9 ± 10	14.6 ± 1.0	28.8 ± 10	$14.5 \pm .95$

2.2.5 Comparison of Computational Time

Table 2.5 lists the average computational time per image, in milliseconds, across 200 test image reconstructions for CG-LS and each of the compared prior art iterative algorithms. For fair comparison, each algorithm was run on the same 64-bit Intel(R) Xeon(R) CPU E5-2690 machine.



(a) Original **(b)** Radon Transform **(c)** gCG-LS (0.503) **(d)** nCG-LS (0.504) **(e)** FISTA (0.487)

(f) KF (0.491) **(g)** ℓ_1-LS (0.401) **(h)** FBP (0.348) **(i)** BCS (0.342) **(j)** CoSaMP (0.342)

Figure 2.3: Image reconstructions (SSIM) using our gCG-LS, our nCG-LS, FISTA, KF, ℓ_1-LS, FBP, BCS, and CoSaMP. The input to each algorithm is a vectorized (2.3b), which is the Radon transform of (2.3a) at 15 uniformly spaced angles with an SNR of 60dB. We observe that CG-LS outperforms all methods with nCG-LS slightly outperforming gCG-LS.

We observe that the high performance of CG-LS is at the expense of a significant computational time. Although, while CG-LS is slower than all of the comparative iterative methods, it, however, has yet to be optimized for computational efficiency and speed. As such, the required computational time for CG-LS could likely be significantly improved upon either through improving the base code implementation or by implementing CG-LS in C or C++.

Table 2.5: Average reconstruction time of 32×32 images from Radon transform measurements at 15 uniform angles for our nCG-LS, our gCG-LS, and five comparison iterative algorithms.

Method	nCG-LS	gCG-LS	FISTA	ℓ_1-LS	FBP	BCS	CoSaMP
Time (ms)	1.5×10^5	8.7×10^4	410	180	1.5	622	407

2.2.6 Comparison Between Tikhonov and CG-LS Estimator

Here, we discuss the relationship between the CG-LS estimator and a general Tikhonov regularization solution to the inverse problem of (1.1). We specifically compare against the Tikhonov solution as this is a core component of the CG-LS estimator since the $\boldsymbol{u}$ field is a Tikhonov esti-

mate. Consider the general Tikhonov regularization problem and corresponding solution

$$\begin{aligned}\boldsymbol{c}^*_{\text{Tik}} &= \arg\min_{\boldsymbol{x}} \ ||\boldsymbol{y} - A\boldsymbol{c}||_2^2 + ||\Gamma\boldsymbol{c}||_2^2 \\ &= (A^TA + \Gamma^T\Gamma)^{-1}A^T\boldsymbol{y}\end{aligned}$$

where Γ is known as the Tikhonov matrix. Typically, $\Gamma = \alpha I$ for some scalar $\alpha > 0$. Assume that Γ is the Cholesky factorization of some matrix Q. That is, $Q = \Gamma^T\Gamma$. Then

$$\boldsymbol{c}^*_{\text{Tik}} = (A^TA + Q)^{-1}A^T\boldsymbol{y} = Q^{-1}A^T(I + AQ^{-1}A^T)^{-1}\boldsymbol{y}$$

where the second equality results from the Woodbury matrix identity and holds only if Q is invertible.

Now, consider a generalized CG-LS problem

$$\begin{aligned}&\boldsymbol{c}^*_{\text{CG}} = \boldsymbol{z}^* \odot \boldsymbol{u}^* \qquad \text{where} \\ &\boldsymbol{z}^*, \boldsymbol{u}^* = \arg\min_{\boldsymbol{z},\boldsymbol{u}} \ \frac{1}{2}||\boldsymbol{y} - A(\boldsymbol{z} \odot \boldsymbol{u})||_2^2 + \frac{1}{2}\boldsymbol{u}^T P_u^{-1}\boldsymbol{u} + \mathcal{R}(\boldsymbol{z})\end{aligned}$$

for any regularization function $\mathcal{R}(\boldsymbol{z})$ and P_u a covariance matrix of $\boldsymbol{u}$. Note, for CG-LS, $P_u = \lambda^{-1}I$ and $\mathcal{R}(\boldsymbol{z}) = \mu\|f(\boldsymbol{z})\|_2^2$ and thus the above cost function may be viewed as a generalization of the CG-LS cost function. As the $\boldsymbol{u}$ estimate is a Tikhonov solution we have

$$\boldsymbol{u}^* = (A^T_{\boldsymbol{z}^*}A_{\boldsymbol{z}^*} + P_u^{-1})^{-1}A^T_{\boldsymbol{z}^*}\boldsymbol{y}$$

and thus, for $D(\boldsymbol{z}) = \text{Diag}(\boldsymbol{z})$

$$\begin{aligned}\boldsymbol{c}^*_{\text{CG}} &= \boldsymbol{z}^* \odot (A^T_{\boldsymbol{z}^*}A_{\boldsymbol{z}^*} + P_u^{-1})^{-1}A^T_{\boldsymbol{z}^*}\boldsymbol{y} \\ &= \boldsymbol{z}^* \odot P_uA^T_{\boldsymbol{z}^*}(I + A_{\boldsymbol{z}^*}P_uA^T_{\boldsymbol{z}^*})^{-1}\boldsymbol{y} \\ &= D(\boldsymbol{z}^*)P_uD(\boldsymbol{z}^*)A^T(I + A(\boldsymbol{z}^*)P_uD(\boldsymbol{z}^*)A^T)^{-1}\boldsymbol{y}\end{aligned}$$

where the second equality results from the Woodbury matrix identity and holds for all $\boldsymbol{z}^*$ and invertible P_u. Take $B(\boldsymbol{z}^*) = D(\boldsymbol{z}^*)P_u D(\boldsymbol{z}^*)$ then the generalized CG-LS solution is

$$\boldsymbol{c}^*_{\text{CG}} = B(\boldsymbol{z}^*)A^T(I + AB(\boldsymbol{z}^*)A^T)^{-1}\boldsymbol{y}.$$

Thus, summarizing the solutions

$$\boldsymbol{c}^* = \begin{cases} (A^TA + Q)^{-1}A^T\boldsymbol{y} & \text{Tikhonov} \\ \boldsymbol{z}^* \odot (A^T_{\boldsymbol{z}^*}A_{\boldsymbol{z}^*} + P_u^{-1})^{-1}A^T_{\boldsymbol{z}^*}\boldsymbol{y} & \text{CG-LS} \end{cases}$$
$$= \begin{cases} Q^{-1}A^T(I + AQ^{-1}A^T)^{-1}\boldsymbol{y} & \text{Tikhonov} \\ B(\boldsymbol{z}^*)A^T(I + AB(\boldsymbol{z}^*)A^T)^{-1}\boldsymbol{y} & \text{CG-LS} \end{cases}$$

where the second equality again is from the Woodbury matrix identity and holds only for invertible Q. Therefore, the Tikhonov solution is equivalent to the CG-LS solution when $Q^{-1} = B(\boldsymbol{z}^*)$. From this perspective, we can view the CG-LS algorithm as a method determining the optimal Tikhonov matrix, by estimating $\boldsymbol{z}^*$, for each linear inverse problem of interest. As the Tikhonov matrix is typically determined a priori and fixed for all estimates in the standard Tikhonov solution, CG-LS has the benefit of removing these required user chosen parameters to allow for improved reconstructions over the standard Tikhonov solution. Additionally, the equivalence between CG-LS and the standard Tikhonov solution only occurs for invertible Q where $Q^{-1} = B(\boldsymbol{z}^*)$. For sparse reconstructions, $\boldsymbol{z}^*$ controls the sparsity of the final reconstruction $\boldsymbol{c}^* = \boldsymbol{z}^* \odot \boldsymbol{u}^*$ and clearly if any entry of $\boldsymbol{z}^*$ is zero then $B(\boldsymbol{z}^*)$ is not invertible. Therefore, the CG-LS solution is distinct from a standard Tikhonov solution for reconstructing sparse signals of interest.

We empirically evaluate the standard Tikhonov solution to further show that the CG-LS solution is distinct from, and outperforms, the Tikhonov solution. Table 2.6 shows the average SSIM and PSNR from gCG-LS, nCG-LS, and a standard Tikhonov solution to the linear inverse problem of recovering an image from Radon transform measurements at 15, 10, or 6 uniformly spaced

angles. We observe that our nCG-LS and gCG-LS methods outperform the standard Tikhonov by SSIM while performing comparably in PSNR. Furthermore, noise has a greater impact on the standard Tikhonov solution as compared to CG-LS. This is observed by inspecting the difference in the average SSIM or PSNR for reconstructing from 40dB SNR measurements versus reconstructing from 60dB SNR measurements. The decrease in SSIM and PSNR is greater for the Tikhonov solution as compared to CG-LS.

Table 2.6: Average SSIM ($\times 10^2$) and PSNR with 99% confidence intervals for our gCG-LS, our nCG-LS, and a standard Tikhonov solution. Each algorithm estimated two hundred, 32×32 CIFAR10 images from Radon transform measurements taken at 15, 10, or 6 uniformly spaced angles. Each measurement has an SNR of 60dB or 40dB. We find that CG-LS performs best in all noise and sparse scenarios where nCG-LS is in **bold** and gCG-LS is in *italics*.

Angles	15		10		6	
SNR	60 dB		60 dB		60 dB	
	SSIM ($\times 10^2$)	PSNR	SSIM ($\times 10^2$)	PSNR	SSIM ($\times 10^2$)	PSNR
nCG-LS	$\mathbf{91.3 \pm .59}$	$\mathbf{28.0 \pm .39}$	$\mathbf{83.6 \pm .82}$	$\mathbf{24.8 \pm .39}$	$\mathbf{70.0 \pm 1.1}$	$\mathbf{21.7 \pm .37}$
gCG-LS	$\textit{90.1} \pm \textit{.57}$	$\textit{27.1} \pm \textit{.38}$	$\textit{82.3} \pm \textit{.78}$	$\textit{24.3} \pm \textit{.40}$	$\textit{68.9} \pm \textit{1.1}$	$\textit{21.4} \pm \textit{.42}$
Tikhonov	$90.0 \pm .55$	$28.1 \pm .39$	$81.9 \pm .84$	$24.7 \pm .39$	67.9 ± 1.2	$21.6 \pm .39$
Ratio	15		10		6	
SNR	40 dB		40 dB		40 dB	
	SSIM ($\times 10^2$)	PSNR	SSIM ($\times 10^2$)	PSNR	SSIM ($\times 10^2$)	PSNR
nCG-LS	$\mathbf{87.0 \pm .97}$	$\mathbf{26.2 \pm .28}$	$\mathbf{80.6 \pm .94}$	$\mathbf{24.0 \pm .32}$	$\mathbf{68.0 \pm 1.2}$	$\mathbf{21.3 \pm .35}$
gCG-LS	$\textit{85.9} \pm \textit{.93}$	$\textit{25.5} \pm \textit{.28}$	$\textit{80.1} \pm \textit{.86}$	$\textit{23.8} \pm \textit{.34}$	$\textit{67.9} \pm \textit{1.1}$	$\textit{21.3} \pm \textit{.40}$
Tikhonov	85.2 ± 1.0	$26.0 \pm .27$	$78.0 \pm .96$	$23.7 \pm .31$	65.5 ± 1.2	$21.3 \pm .34$

Similarly, Table 2.7 shows the average SSIM and PSNR from gCG-LS, nCG-LS, and a standard Tikhonov solution to image reconstruction from random Gaussian measurements at a sampling ratio of 0.5, 0.3, or 0.1. Table 2.7 emphasizes the difference in reconstruction quality that is obtained from CG-LS over a standard Tikhonov solution as CG-LS significantly outperforms a standard Tikhonov solution in these compressive sensing examples. Lastly, we comment that in each Tikhonov experiment, the Tikhonov matrix was set as $\Gamma = \alpha I$ where α was chosen to produce the best reconstructions possible.

Table 2.7: Average SSIM ($\times 10^2$) and PSNR with 99% confidence intervals for our gCG-LS, our nCG-LS, and a standard Tikhonov estimate. Each algorithm estimated two hundred, 32×32 CIFAR10 [69] images from random Gaussian measurements taken at a sampling ratio of 0.5, 0.3, or 0.1. Each measurement has an SNR of 60dB. We find that CG-LS performs best in all noise and sparse scenarios where nCG-LS is in **bold** and gCG-LS is in *italics*.

Ratio	0.5		0.3		0.1	
SNR	60 dB		60 dB		60 dB	
	SSIM ($\times 10^2$)	PSNR	SSIM ($\times 10^2$)	PSNR	SSIM ($\times 10^2$)	PSNR
nCG-LS	$\mathbf{76.2 \pm .87}$	$\mathbf{23.1 \pm .40}$	$\mathbf{57.1 \pm 1.2}$	$\mathbf{20.0 \pm .43}$	$\mathbf{25.2 \pm 1.4}$	$\mathbf{15.3 \pm .43}$
gCG-LS	$\mathit{76.9 \pm .91}$	$\mathit{23.3 \pm .42}$	$\mathit{57.5 \pm 1.3}$	$\mathit{20.0 \pm .46}$	$\mathit{24.7 \pm 1.5}$	$\mathit{15.1 \pm .44}$
Tikhonov	$12.4 \pm .96$	$7.62 \pm .29$	$6.80 \pm .56$	$6.68 \pm .34$	1.57 ± 0.14	$5.66 \pm .31$

2.2.7 Impact of Sparsity on Reconstruction Quality

One reason our CG-LS method outperforms the comparative methods may be attributed to the relatively low sparsity level of the wavelet coefficients, $\boldsymbol{c}$. That is, only a small number of the wavelet coefficients are zero, which CG-LS manages while the sparsity based approaches of FISTA, ℓ_1-LS, BCS, and CoSaMP on the other hand require a sufficiently high signal sparsity for superb performance. Additionally, we remark that while a small number of the wavelet coefficients are zero, many of the wavelet coefficients are small, e.g. on the order of 10^{-4}, compared to the maximum wavelet coefficient that is on the order of 10^1. These small, but nonzero, wavelet coefficients appear to be particularly troublesome for the sparsity based iterative reconstruction approaches.

To illustrate the impact of the small, nonzero wavelet coefficients, let $\boldsymbol{I}$ and $\boldsymbol{c} = \Phi \boldsymbol{I}$ be a vectorized image and corresponding biorthogonal wavelet coefficients, respectively. Let $\mathcal{H}_\delta : \mathbb{R} \to \mathbb{R}$ be the **Hard Thresholding** componentwise function defined as, for $\delta > 0$ a fixed parameter,

$$\mathcal{H}_\delta(x) = \begin{cases} 0 & |x| \leq \delta \\ x & |x| > \delta. \end{cases}$$

We sparsify the image wavelet coefficients by applying the Hard thresholding function, that is, $\boldsymbol{c}_\delta = \mathcal{H}_\delta(\boldsymbol{c})$. Then we define a "sparsified" image formulation, the image is not necessarily sparse but formed from the sparsified wavelet coefficients, $\boldsymbol{I}_\delta = \Phi^T \boldsymbol{c}_\delta$. The "sparsified" image is measured as in (1.6), that is, $\boldsymbol{y} = \Psi \Phi \boldsymbol{c}_\delta + \boldsymbol{\nu}$ and we consider the linear inverse problem of recovering $\boldsymbol{c}_\delta$ from

$\boldsymbol{y}$. With the estimated sparse coefficients $\widehat{\boldsymbol{c}}_\delta$ we consider the estimated image $\widehat{\boldsymbol{I}}_\delta = \Phi^T \widehat{\boldsymbol{c}}_\delta$, which is compared against $\boldsymbol{I}_\delta$ to assess reconstruction quality in terms of SSIM and PSNR.

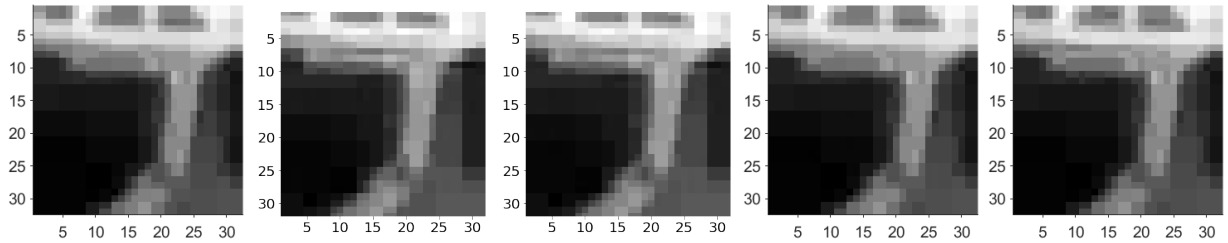
Table 2.8 displays the average SSIM and PSNR from estimating 200 "sparsified" 32×32 CIFAR10 images using FISTA, ℓ_1-LS, BCS, and CoSaMP. In comparing Table 2.1 and Table 2.8 we see that truly sparse coefficients significantly improve the reconstruction quality of each sparsity-based iterative algorithm. In particular, the small but nonzero components that exist in the wavelet coefficients of natural images, and images from other modalities, is detrimental to the success of each sparsity-based iterative algorithm. Note, for each image, δ was chosen such that $\mathcal{H}_\delta(\boldsymbol{c})$ sets all but the absolute largest 20% of the wavelet coefficients, i.e. 820 of the 1024 coefficients, to zero. Additionally, the results of Table 2.8 are for no additive measurement noise, i.e. $\boldsymbol{\nu} = \boldsymbol{0}$, and adding measurement noise to the "sparsified" image measurements produces results for each of the iterative algorithms on par with those appearing in Table 2.1. That is, adding small nonzero deviations in the measurements, via the measurement noise, is also detrimental to the success of each sparsity-based iterative algorithm.

For visualization a sample "sparsified" reconstruction is displayed in Figure 2.4. Figure 2.4 shows the reconstruction of a "sparsified" 32×32 Barbara snippet from a 15 angle Radon transform with no additive noise. Comparing Figure 2.1 and Figure 2.4 we again see that sparsity in

Table 2.8: Study on the impact of sparsity for FISTA, ℓ_1-LS, BCS, and CoSaMP reconstructions. Each method reconstructed two hundred, 32×32 CIFAR10 images from Radon transform measurements taken at 15, 10, or 6 uniformly spaced angles with no additive noise. The wavelet coefficients of each image were sparsified before measuring by setting all but the absolute largest 20% of the entries, i.e. 820 of the 1024 entries, to zero. Provided is the average SSIM $(\times 10^2)$ and PSNR with 99% confidence intervals for each reconstruction algorithm. We find that a truly sparse signal greatly benefits each of the methods in terms of reconstruction quality and many small but nonzero components in the image wavelet coefficients is detrimental to each methods success.

Angles	15		10		6	
	SSIM $(\times 10^2)$	PSNR	SSIM $(\times 10^2)$	PSNR	SSIM $(\times 10^2)$	PSNR
FISTA	$97.5 \pm .35$	$34.3 \pm .71$	$84.8 \pm .77$	$25.6 \pm .41$	61.6 ± 1.3	$20.6 \pm .42$
ℓ_1-LS	$98.5 \pm .31$	39.1 ± 1.3	$85.2 \pm .95$	$25.9 \pm .45$	61.8 ± 1.2	$20.6 \pm .42$
BCS	$95.5 \pm .86$	33.3 ± 3.3	71.5 ± 1.1	$21.5 \pm .40$	49.5 ± 1.6	$18.1 \pm .43$
CoSaMP	$97.0 \pm .63$	36.1 ± 2.1	$74.3 \pm .98$	$21.9 \pm .42$	49.2 ± 1.1	$18.9 \pm .44$

the wavelet coefficients has monumental impact on the quality of the estimate image for each of the four sparsity-based iterative algorithms. While these experiments highlight the importance of sparse signals for FISTA, ℓ_1-LS, BCS, and CoSaMP, these experiments have little practical application as often only the noisy measurement $\boldsymbol{y}$ is known upfront and we cannot force sparsity of the image coefficients as we have done for illustration in this subsection.



(a) Original **(b)** FISTA (0.933) **(c)** ℓ_1-LS (0.943) **(d)** BCS (0.957) **(e)** CoSaMP (0.947)

Figure 2.4: Sparsified image reconstructions, with SSIM in parenthesis, using four sparsity based iterative reconstruction methods: FISTA, ℓ_1-LS, BCS, and CoSaMP. The input to each algorithm is noiseless 15 angle Radon transform of (2.4a). Compared to Figure 2.1, we observe that measuring truly sparse wavelet coefficients produces significantly improved reconstructions from each method.

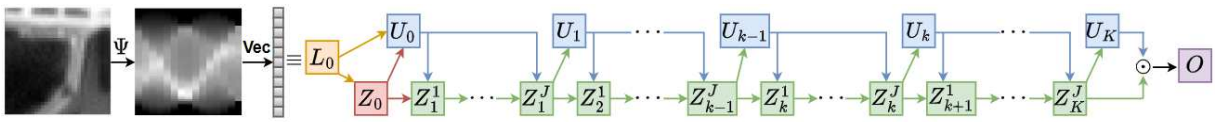
2.3 Compound Gaussian Network (CG-Net)

2.3.1 Network Structure

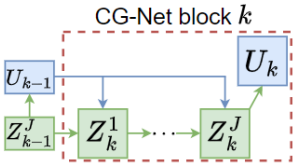
We create a novel DNN, named compound Gaussian Network (CG-Net), by applying algorithm unrolling to CG-LS in Algorithm 2. CG-Net has the structure shown in Figure 2.5, which consists of an input layer, L_0, and initial $\boldsymbol{z}$ layer, Z_0, and initial $\boldsymbol{u}$ layer U_0, K CG-Net blocks given in Figure 2.5b, and an output layer, O.

Two main functions, $\boldsymbol{f}_k : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ and $\boldsymbol{g}_k^j : \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$, define the CG-Net layers. Each $\boldsymbol{f}_k(\boldsymbol{z}, \boldsymbol{y}) \equiv \boldsymbol{f}_k(\boldsymbol{z}, \boldsymbol{y}; \lambda_k)$, which is parameterized by a scalar λ_k, corresponds to updating $\boldsymbol{u}$ as

$$\boldsymbol{f}_k(\boldsymbol{z}, \boldsymbol{y}) \coloneqq A_{\boldsymbol{z}}^T (A_{\boldsymbol{z}} A_{\boldsymbol{z}}^T + \lambda_k I)^{-1} \boldsymbol{y}. \tag{2.30}$$



(a) End-to-end network structure of CG-Net.



(b) Block k of CG-Net

$$L_0 = \mathbf{y} \qquad U_k = (A_{Z_k^J}^T A_{Z_k^J} + \lambda_k I)^{-1} A_{Z_k^J}^T L_0 \qquad O = U_K \odot Z_K^J$$

$$Z_0 = \mathcal{P}_{a_0,b_0}(A^T L_0) \qquad Z_k^j = \mathcal{P}_{a_k^j,b_k^j}\left(Z_k^{j-1} + \eta_k^j \mathbf{d}_k^j(Z_k^{j-1}; U_{k-1}, L_0)\right)$$

(c) Mathematical descriptions of the CG-Net layers.

Figure 2.5: End-to-end network structure for CG-Net, the unrolled deep neural network of Algorithm 2, is shown in (2.5a). A mathematical description of each layer is given in (2.5c). CG-Net consists of an input layer, L_0, initialization layer, Z_0, output layer, O, and K CG-Net blocks (2.5b). Each CG-Net block, k, contains J steepest descent layers, $Z_k^1, \ldots, Z_k^J$, and a single Tikhonov layer, U_k. Every layer takes the measurements, $L_0 \equiv \boldsymbol{y}$, as an input so these connections are omitted for clarity.

That is, each $\boldsymbol{f}_k$ is the Tikhonov solution (2.7) with parameter λ_k. Next, each $\boldsymbol{g}_k^j(\boldsymbol{z}, \boldsymbol{u}, \boldsymbol{y}) \equiv \boldsymbol{g}_k^j(\boldsymbol{z}, \boldsymbol{u}, \boldsymbol{y}; a_k^j, b_k^j, \eta_k^{(j)}, \boldsymbol{d}_k^j)$, which is parameterized by a descent direction, $\boldsymbol{d}_k^j$, step size, $\eta_k^{(j)}$ and mReLU parameters a_k^j and b_k^j, corresponds to updating $\boldsymbol{z}$ as

$$\boldsymbol{g}_k^j(\boldsymbol{z}, \boldsymbol{u}, \boldsymbol{y}) \coloneqq \mathcal{P}_{a_k^j,b_k^j}\left(\boldsymbol{z} + \eta_k^{(j)} \boldsymbol{d}_k^j(\boldsymbol{z}; \boldsymbol{u}, \boldsymbol{y})\right). \tag{2.31}$$

That is, each $\boldsymbol{g}_k^j$ is a projected steepest descent update of $\boldsymbol{z}$.

In CG-LS, the step size, $\eta_k^{(j)}$, is found by a backtracking line search, which we cannot implement in CG-Net. Thus, the application of the mReLU activation function, $\mathcal{P}_{a,b}$, at each steepest descent step serves to guarantee the next step is not too large and stays within $\mathcal{Z}$. Implementing the mReLU activation function at each steepest descent step is further justified by Theorem 2.2.6 where we know that a minimizer lies within a hypercube and that $\mathcal{P}_{a,b}$ projects onto this hypercube. Finally, the application of the mReLU function additionally assists in providing generalization error bounds for CG-Net.

Now, we mathematically detail the CG-Net layers:

$L_0 = \boldsymbol{y}$ is the input measurements to the network

$Z_0 = \mathcal{P}_{a_0,b_0}(\hat{A}^T\boldsymbol{y})$, for $\hat{A} = A/\|A\|_2$, is the initial estimate of $\boldsymbol{z}$ from line 1 of Algorithm 2.

$U_0 = \boldsymbol{f}_0(Z_0, \boldsymbol{y})$ is the initial estimate of $\boldsymbol{u}$ corresponding to line 1 of Algorithm 2.

The kth CG-Net block, shown in Figure 2.5b, consists of layers:

$Z_k^j = \boldsymbol{g}_k^j(Z_k^{j-1}, U_{k-1}, \boldsymbol{y})$ is $\boldsymbol{z}$ on steepest descent step j of iteration k, corresponding to line 10 in Algorithm 2.

$U_k = \boldsymbol{f}_k(Z_k^J, \boldsymbol{y})$ is $\boldsymbol{u}$ on iteration k, corresponding to line 14 in Algorithm 2.

$O = U_K \odot Z_K^J$ is the estimated wavelet coefficients produced by CG-Net.

Note, to simplify notation, we let $Z_k^0 = Z_{k-1}^J$ for every $k \in \{2, \dots, K\}$ and $Z_1^0 = Z_0$. CG-Net contains $K+1$ estimation layers for $\boldsymbol{u}$, KJ steepest descent step layers for $\boldsymbol{z}$, one input, one output and one initialization layer. Thus, CG-Net contains $K(J+1)+4$ layers in total.

By incorporating the CG prior, CG-Net differentiates from previous unrolled DNNs in two key ways. First, the Tikhonov updates of $\boldsymbol{u}$ provide non-linear transformation layers of $\boldsymbol{z}$. These layers impose a significant structure via data consistency in equating measurements and measured estimated signals. Second, instead of estimating the original signal of interest directly, CG-Net simultaneously estimates two separate signals and formulates its output as a Hadamard product, which may be viewed as a unique output activation function. These network attributes, inspired by the theory in Sections 2.2.2 and 2.2.3, are shown to be advantageous empirically in Section 2.3.3.

2.3.2 Network Parameters and Loss Functions

We further detail the parameters that will be learned by CG-Net. First, let $\epsilon > 0$ be a small, positive real number parameter, which is not updated during the training of CG-Net. This stabilizing parameter ϵ acts as a minimum value threshold for certain parameters in CG-Net. Now, for every $k = 0, 1, \dots, K$, the layer U_k is parameterized by regularization scalar, λ_k, as given in (2.30). While in CG-LS, given in Algorithm 2, every λ_k is taken to be the same constant, λ from (2.2), we instead increase the trainability of CG-Net by allowing different λ_k at each layer updating $\boldsymbol{u}$.

Furthermore, as each λ_k must be positive for (2.30) to guarantee a unique solution, we replace λ_k in (2.30) with $\max\{\lambda_k, \epsilon\}$.

Next, for each Z_k^j layer, defined by (2.31), we implement the steepest descent direction

$$\boldsymbol{d}_k^j(\boldsymbol{z}; \boldsymbol{u}) = -B_k^j \nabla_{\boldsymbol{z}} F(\boldsymbol{z}; \boldsymbol{u})$$

for a positive definite matrix B_k^j that is learned in CG-Net. Note, this descent direction is steepest descent based upon the quadratic norm $||\cdot||^2_{(B_k^j)^{-1}}$. Thus, CG-Net can be understood as learning the quadratic norm defining every steepest descent step that optimally traverses the optimization landscape of the cost function in (2.4). Alternatively, CG-Net can be understood to be learning the geometry of the optimization landscape of the cost function in (2.4) to produce the best updates of $\boldsymbol{z}$. For matrix L, let Q and Λ be the eigendecomposition of $(L + L^T)/2$. That is, $(L + L^T)/2 = Q\Lambda Q^T$. Define, for small $\epsilon > 0$, the diagonal matrix Λ_ϵ as $[\Lambda_\epsilon]_{ii} = \max\{\Lambda_{ii}, \epsilon\}$ and let

$$P_\epsilon(L) = Q\Lambda_\epsilon Q^T. \tag{2.32}$$

That is, $P_\epsilon(L)$ is the closest symmetric, real-valued matrix with minimum eigenvalue of ϵ to $(L + L^T)/2$ as measured by the Frobenius norm.

We enforce B_k^j to be positive definite by learning a lower triangular matrix L_k^j and setting $B_k^j = P_\epsilon(L_k^j)$. Therefore, CG-Net layer Z_k^j is parameterized by a lower triangular matrix L_k^j defining the steepest descent vector

$$\boldsymbol{d}_k^j(\boldsymbol{z}; \boldsymbol{u}) = -P_\epsilon(L_k^j) \nabla_{\boldsymbol{z}} F(\boldsymbol{u}, \boldsymbol{z}).$$

Note, while the entire matrix L_k^j could be learned, we set CG-Net to learn only the diagonal and sub-diagonal in L_k^j constraining B_k^j to be a tridiagonal matrix. An alternative, for further restriction, would be to only learn the diagonal of L_k^j.

Additionally, as $\nabla_{\boldsymbol{z}} F(\boldsymbol{u}, \boldsymbol{z})$, given in (2.10), depends on the regularization scalar μ, the CG-Net layer Z_k^j is parameterized by regularization scalar μ_k^j. As before, while in CG-LS every μ_k^j

is taken to be the same constant, μ from (2.2), we instead increase the trainability of CG-Net by allowing different μ_k^j at each layer updating $\boldsymbol{z}$. Furthermore, layer Z_k^j is parameterized by the step size $\eta_k^{(j)}$ which we take to be a diagonal matrix to allow each steepest descent update to scale the step length in each component separately.

Finally, layer Z_k^j learns real numbers $a_k^j > z_{\min}$ and $b_k^j > z_{\min}$, which are applied through the mReLU activation function, $\mathcal{P}_{a_k^j,b_k^j}$, in (2.31). As we require that $a_k^j > z_{\min}$ and $b_k^j > z_{\min}$ we set $a_k^j = \max\{a_k^j, z_{\min} + \epsilon\}$ and $b_k^j = \max\{b_k^j, z_{\min} + \epsilon\}$. Similarly, the initial $\boldsymbol{z}$ layer, Z_0, is parameterized by two real numbers a_0 and b_0, defining the mReLU function, $\mathcal{P}_{a_0,b_0}$, where we set $a_0 = \max\{a_0, z_{\min} + \epsilon\}$ and $b_0 = \max\{b_0, z_{\min} + \epsilon\}$ for learned a_0 and b_0.

At a maximum, for each L_k^j a full lower triangular matrix, CG-Net has $K\left(\frac{J}{2}\left(n(n+3)+6\right)\right)+3$ parameters

$$\boldsymbol{\Theta} = \left\{\lambda_0, a_0, b_0, \lambda_k, \mu_k^j, L_k^j, \eta_k^{(j)}, a_k^j, b_k^j\right\}_{k=1,2,\ldots,K}^{j=1,2,\ldots,J}$$

for n is the reconstructed signal size. For our application of CG-Net, only the diagonal and sub-diagonal in L_k^j are learned and thus CG-Net has a total of $K(J(3n+2)+1)+3$ parameters. The CG-Net parameters are trained by minimizing a loss function involving the SSIM image quality metric [49], namely

$$\mathcal{L}_{\mathcal{B}}(\boldsymbol{\Theta}) = \frac{1}{|\mathcal{B}|}\sum_{(\overline{\boldsymbol{y}}_i,\overline{\boldsymbol{c}}_i)\in\mathcal{B}}\left(1 - \text{SSIM}(\Phi\widehat{\boldsymbol{c}}(\overline{\boldsymbol{y}}_i;\boldsymbol{\Theta}), \Phi\overline{\boldsymbol{c}}_i)\right)$$

where $\mathcal{B} \subset \mathcal{D} = \{(\overline{\boldsymbol{y}}_i, \overline{\boldsymbol{c}}_i) : i = 1, 2, \ldots, N_s\}$ is a batch of data points and $\widehat{\boldsymbol{c}}(\overline{\boldsymbol{y}}_i;\boldsymbol{\Theta})$ is the estimate of $\boldsymbol{c}$ from CG-Net given input $\overline{\boldsymbol{y}}_i$ and network parameterization $\boldsymbol{\Theta}$. We note that the mean absolute error loss function is an adequate alternative providing nearly identical results for CG-Net as produced by the SSIM loss function.

The CG-Net loss function is optimized through adaptive moment estimation (Adam) [73], which is a stochastic gradient-based optimizer. Other common optimization methods include: stochastic gradient descent, RMSprop, and Adadelta [73]. The gradient, $\nabla_{\boldsymbol{\Theta}}\mathcal{L}_{\mathcal{B}}$, used in Adam is

calculated via backpropagation through the network, which we implement with automatic differentiation [74] using TensorFlow [75].

2.3.3 Numerical Results

Given a sensing matrix, Ψ, sparsity transformation, Φ, and noise level in SNR, we create a set of training and testing measurement-coefficient pairs, $(\boldsymbol{y}, \boldsymbol{c})$, as in Section 2.2.4, that we use to train and evaluate a CG-Net. Measurements are formed from 32×32 CIFAR10 [69] images and 64×64 CalTech101 [70] images. For network size, CG-Net running on 32×32 or 64×64 image measurements use $(K, J) = (20, 1)$ or $(K, J) = (5, 1)$, respectively. These network sizes were chosen empirically such that the time to complete one image reconstruction was reasonably quick while still producing excellent reconstructions on a validation set of test images.

For all experiments we set $f(z) = \ln(z)$ and initialize $a_0 = 1, b_0 = \exp(2)$, every $\eta_k^{(j)} = \frac{1}{2}I$, $\mu_k^j = 2$, $a_k^j = \frac{8}{10}$, $b_k^j = \exp(3)$, and $L_k^j = I$. Finally, we initialize $\lambda_k = 0.3$ for measurements with an SNR of 60dB and $\lambda_k = 2$ all other measurements. Every CG-Net was trained for 30 epochs using a learning rate of 10^{-3} for Adam with early stopping implemented as necessary to prevent overfitting.

We compare CG-Net against nine state-of-the-art methods: memory augmented deep unfolding network (MADUN) [22], ISTA-Net$^+$ [23], FISTA-Net [24], iPiano-Net [30], ReconNet [13], LEARN^{++} [29], Learned Primal-Dual (LPD) [31], FBPConvNet [56], and iRadonMAP [55]. Additionally, memory augmented proximal unrolled network (MAPUN) [26] was considered, but due to the similarity in performance to MADUN only MADUN results are shown. Note, LEARN^{++}, LPD, FBPConvNet, and iRadonMAP are CT-specific reconstruction methods relying on the structure of the CT sinogram measurements. Instead MADUN, ISTA-Net$^+$, FISTA-Net, iPiano-Net, and ReconNet are linear inverse problem reconstruction methods with particular application in image compressive sensing. Furthermore, we remark that MADUN, ISTA-Net$^+$, FISTA-Net, iPiano-Net, LEARN^{++}, and LPD are DNNs formed by algorithm unrolling while ReconNet, iRadonMAP, and FBPConvNet are instead standard DNNs.

For every set of training data, each method was trained using early stopping. That is, as shown in Fig. 2.6, training was conducted until the model initially overfits as compared to a validation dataset. In doing so, we ensure every model is sufficiently trained while also not being over trained thereby presenting the best performance for each model for the provided set of training data.

Shown in Fig. 2.7 is the average SSIM quality, over 200 test image reconstructions, for CG-Net and the nine comparison methods when each is trained on a varying amount of training data. We consider small sets of training data specifically of size 1000, 500, 100, and 20 measurement, coefficient pairs. Note, ReconNet, iRadonMAP, and some instances of FBPConvNet perform significantly lower are thus omitted from Fig. 2.7. In Fig. 2.7a, 2.7b, and 2.7c we see for the reconstruction of images from Radon transforms – at 15, 10, or 6 uniformly spaced angles, respectively

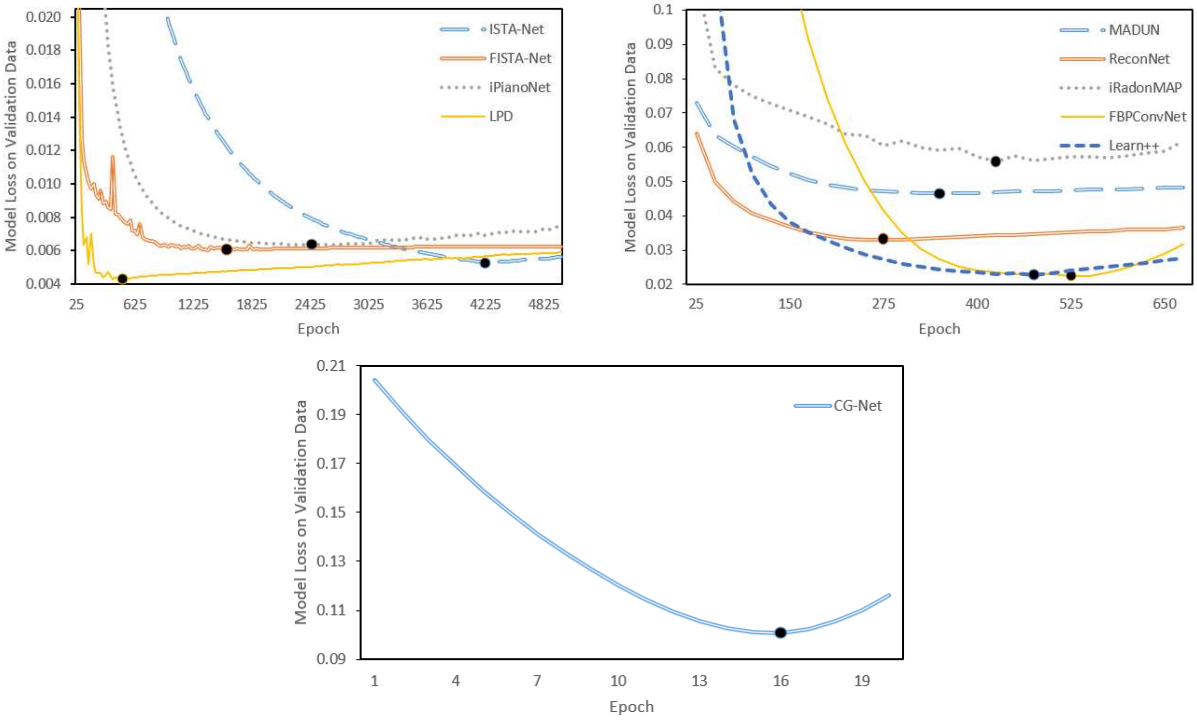
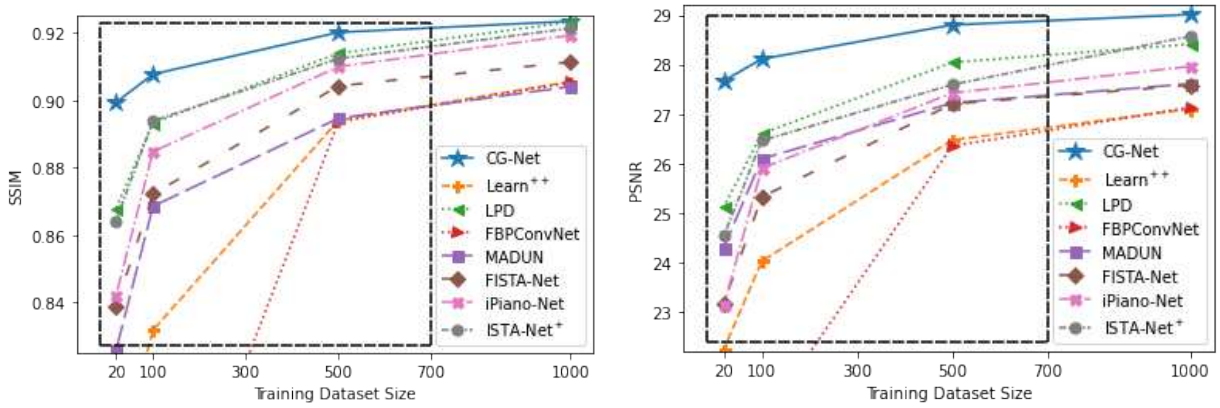
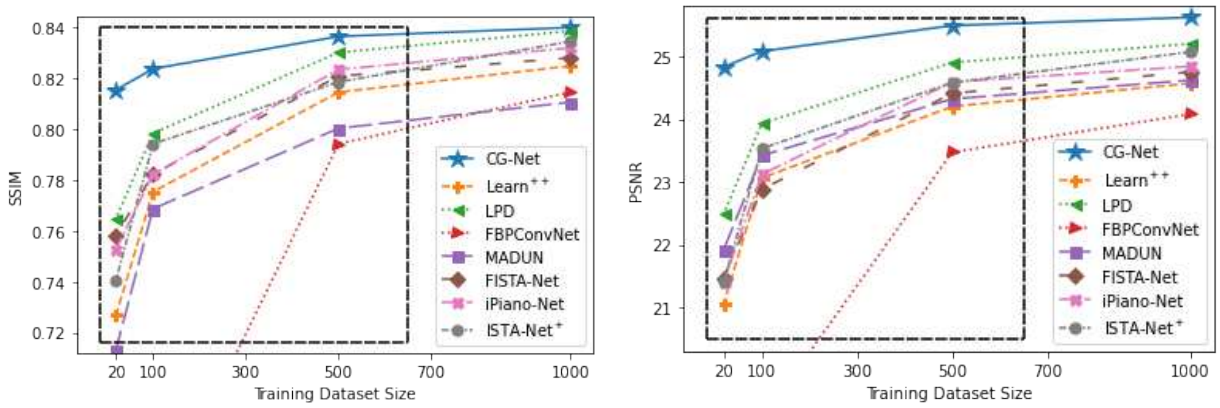


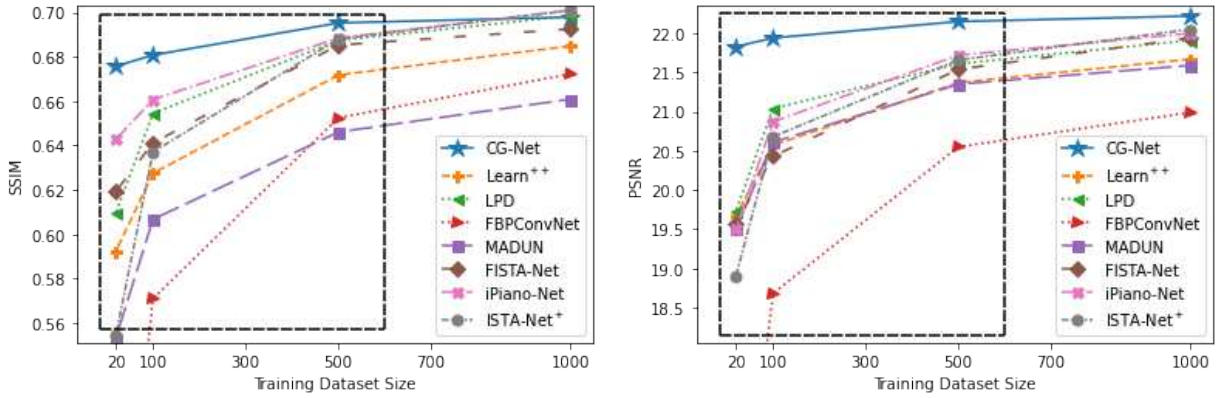
Figure 2.6: Model loss curves on a validation dataset for training each deep learning-based method on Radon inversion from 15 uniformly spaced angles with 20 training samples. The point on each model's loss curve represents the epoch in which that model achieved its best loss on a validation dataset and overfits on subsequent epochs. Only these best loss epoch results are presented for each method. Note, for the plots above, CG-Net uses an SSIM loss function, MADUN uses a mean absolute error loss function, and all other methods use a mean square error loss function (possibly with some additional regularization). As the network loss functions are not equivalent, these plots do not contribute a comparison between the methods and only illustrate that each method is maximally trained for every supplied training dataset.



(a) Reconstructions from fifteen uniformly spaced angle Radon transforms.



(b) Reconstructions from ten uniformly spaced angle Radon transforms.



(c) Reconstructions from six uniformly spaced angle Radon transforms.

Figure 2.7: Average test image reconstruction SSIM (left) and PSNR (right) when we vary the amount of CIFAR10 [69] data in training eight machine learning-based image reconstruction methods. Here, the sensing matrices, Ψ, are a Radon transform at 15, 10, or 6 uniformly spaced angles and the sparsity dictionary, Φ, is a biorthogonal wavelet transform. Measurements in training and testing have an SNR of 60dB. Our method, CG-Net, significantly outperforms all comparative methods, as highlighted in the boxed region, in the low training regime.

– with an SNR of 60dB that CG-Net outperforms all comparative methods and does so appreciably in low training, which is highlighted in the boxed in regions.

Results from training with the smallest training dataset, of size 20, are further detailed in Table 2.9 and Table 2.10, which displays the average SSIM and PSNR plus 99% confidence intervals over 200 test image reconstructions for CG-Net and each comparison method. The results of Table 2.9 are for reconstructing 32×32 CIFAR10 [69] images whereas the results of Table 2.10 are for reconstruction 64×64 CalTech101 [70] images. We see in both Table 2.9 and Table 2.10 that CG-Net significantly outperforms all compared prior art methods in this low training scenario. CG-Net provides, roughly, a 0.03 improvement in SSIM and 1.9dB improvement in PSNR for

Table 2.9: Average SSIM ($\times 10^2$) and PSNR with 99% confidence intervals for ten machine learning-based image reconstruction methods. Each method reconstructed two hundred, 32×32 CIFAR10 [69] images, after training on a **set of only 20 samples**. Sensing matrix, Ψ, is a Radon transform at 15, 10, or 6 uniformly spaced angles and the dictionary, Φ, is a biorthogonal wavelet transform. Each measurement is noisy with an SNR of 60dB or 40dB. In all our method CG-Net, highlighted in **bold**, outperforms other approaches.

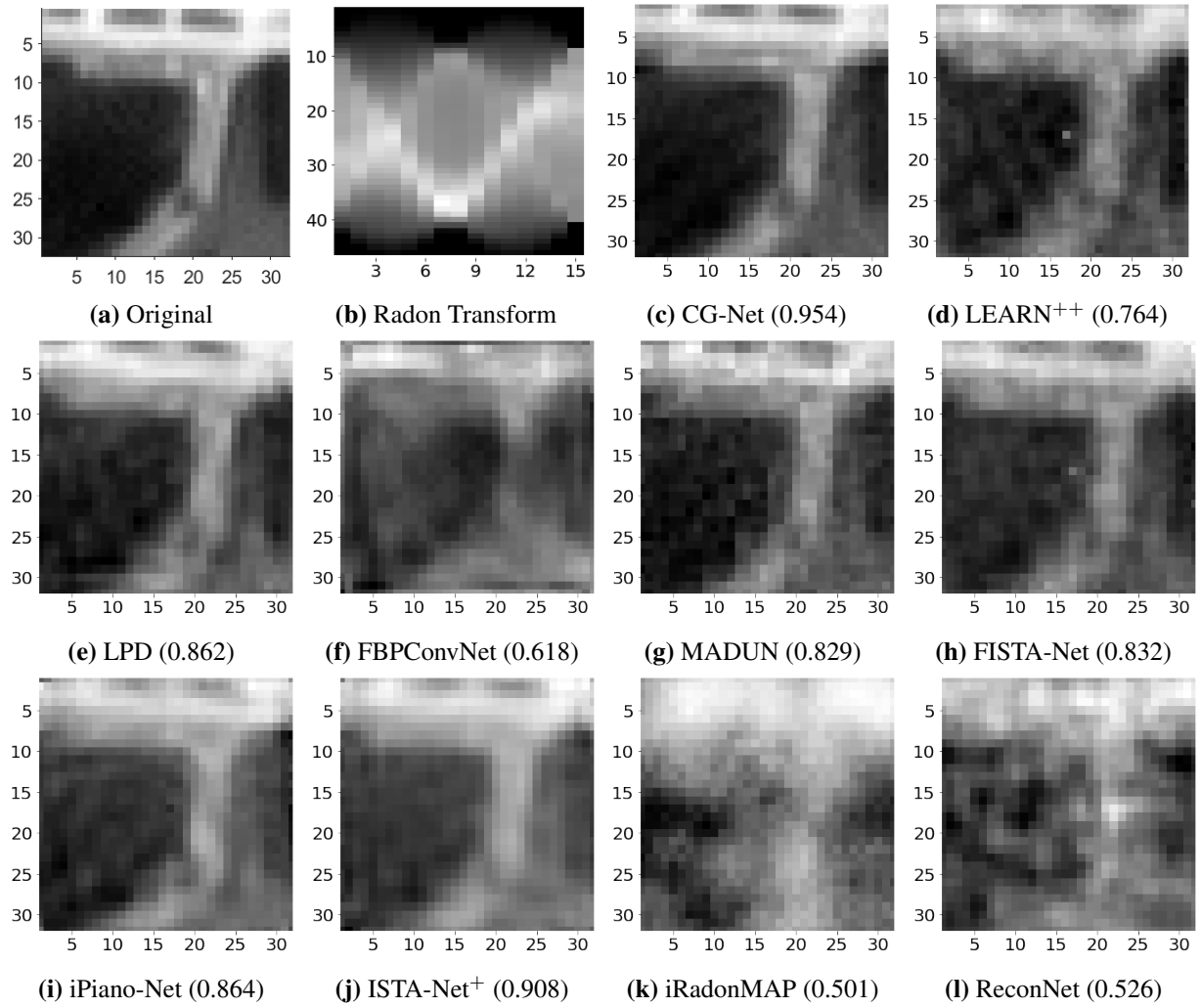
Angles	15		10		6	
SNR	60 dB		60 dB		60 dB	
	SSIM ($\times 10^2$)	PSNR	SSIM ($\times 10^2$)	PSNR	SSIM ($\times 10^2$)	PSNR
CG-Net	$\mathbf{89.9 \pm .56}$	$\mathbf{27.7 \pm .38}$	$\mathbf{81.5 \pm .83}$	$\mathbf{24.8 \pm .40}$	$\mathbf{67.6 \pm 1.2}$	$\mathbf{21.8 \pm .40}$
LEARN^{++}	$79.1 \pm .98$	$22.2 \pm .32$	72.7 ± 1.1	$21.1 \pm .33$	59.2 ± 1.4	$19.7 \pm .33$
LPD	$86.8 \pm .64$	$25.1 \pm .34$	$76.5 \pm .93$	$22.5 \pm .36$	61.0 ± 1.2	$19.7 \pm .32$
FBPConvNet	62.6 ± 2.2	$17.2 \pm .41$	54.1 ± 1.6	$16.3 \pm .31$	41.9 ± 1.4	$14.8 \pm .31$
MADUN	$82.6 \pm .74$	$24.3 \pm .33$	$71.3 \pm .86$	$21.9 \pm .35$	55.3 ± 1.2	$19.5 \pm .36$
FISTA-Net	$83.8 \pm .81$	$23.2 \pm .30$	$75.8 \pm .94$	$21.5 \pm .29$	61.9 ± 1.3	$19.6 \pm .29$
iPiano-Net	$84.2 \pm .63$	$23.1 \pm .44$	$75.3 \pm .91$	$21.4 \pm .43$	64.3 ± 1.2	$19.5 \pm .46$
ISTA-Net^{+}	$86.4 \pm .62$	$24.5 \pm .39$	74.1 ± 1.0	$21.4 \pm .39$	55.4 ± 1.4	$18.9 \pm .33$
iRadonMAP	21.4 ± 1.5	$14.6 \pm .33$	20.8 ± 1.4	$14.4 \pm .35$	18.4 ± 1.3	$14.1 \pm .37$
ReconNet	19.6 ± 1.4	$15.8 \pm .32$	18.7 ± 1.6	$14.9 \pm .32$	17.7 ± 1.5	$14.6 \pm .32$
Angles	15		10		6	
SNR	40 dB		40 dB		40 dB	
	SSIM ($\times 10^2$)	PSNR	SSIM ($\times 10^2$)	PSNR	SSIM ($\times 10^2$)	PSNR
CG-Net	$\mathbf{84.4 \pm .86}$	$\mathbf{25.5 \pm .28}$	$\mathbf{77.4 \pm .83}$	$\mathbf{23.7 \pm .34}$	$\mathbf{66.1 \pm 1.2}$	$\mathbf{21.6 \pm .37}$
LEARN^{++}	78.7 ± 1.0	$22.2 \pm .32$	72.4 ± 1.0	$21.1 \pm .31$	59.1 ± 1.3	$19.6 \pm .32$
LPD	$84.3 \pm .77$	$24.4 \pm .30$	$75.9 \pm .90$	$22.4 \pm .33$	59.6 ± 1.2	$19.4 \pm .31$
FBPConvNet	56.5 ± 1.7	$15.4 \pm .32$	53.6 ± 2.1	$16.3 \pm .37$	40.7 ± 1.6	$14.5 \pm .31$
MADUN	$80.2 \pm .89$	$23.8 \pm .29$	$69.9 \pm .90$	$21.7 \pm .33$	54.6 ± 1.2	$19.4 \pm .35$
FISTA-Net	81.9 ± 1.0	$22.8 \pm .31$	74.9 ± 1.0	$21.4 \pm .29$	61.6 ± 1.3	$19.5 \pm .28$
iPiano-Net	$82.1 \pm .72$	$22.8 \pm .40$	$74.2 \pm .91$	$21.2 \pm .42$	61.8 ± 1.3	$19.4 \pm .43$
ISTA-Net^{+}	$83.6 \pm .83$	$24.1 \pm .36$	$73.6 \pm .99$	$21.4 \pm .37$	55.3 ± 1.4	$18.7 \pm .33$
iRadonMAP	20.7 ± 1.4	$13.9 \pm .37$	20.5 ± 1.4	$14.3 \pm .34$	18.2 ± 1.3	$14.0 \pm .37$
ReconNet	19.0 ± 1.5	$15.2 \pm .31$	18.3 ± 1.6	$15.0 \pm .32$	17.1 ± 1.6	$14.5 \pm .31$

Table 2.10: Average SSIM ($\times 10^2$) and PSNR with 99% confidence intervals for ten machine learning-based image reconstruction methods. Each method reconstructed two hundred, 64×64 CalTech101 [70] images, after training on **only 20 samples**. Sensing matrix, Ψ, is a Radon transform at 15, 10, or 6 uniformly spaced angles and the dictionary, Φ, is a biorthogonal wavelet transform. Each measurement is noisy with an SNR of 60dB or 40dB. In all our method CG-Net, highlighted in **bold**, outperforms other approaches.

Angles	15		10		6	
SNR	60 dB		60 dB		60 dB	
	SSIM ($\times 10^2$)	PSNR	SSIM ($\times 10^2$)	PSNR	SSIM ($\times 10^2$)	PSNR
CG-Net	$\mathbf{65.4 \pm 1.2}$	$\mathbf{22.8 \pm .46}$	$\mathbf{56.6 \pm 1.3}$	$\mathbf{21.1 \pm .44}$	$\mathbf{45.7 \pm 1.3}$	$\mathbf{19.1 \pm .43}$
LEARN^{++}	53.4 ± 1.4	$19.3 \pm .36$	49.8 ± 1.5	$19.0 \pm .37$	38.6 ± 1.3	$17.1 \pm .35$
LPD	64.4 ± 1.2	$21.0 \pm .37$	53.9 ± 1.2	$19.5 \pm .36$	44.2 ± 1.3	$17.5 \pm .36$
FBPConvNet	45.0 ± 1.6	$15.9 \pm .50$	36.1 ± 1.5	$14.2 \pm .38$	30.4 ± 1.2	$12.6 \pm .45$
MADUN	57.8 ± 1.2	$21.0 \pm .46$	49.1 ± 1.2	$19.8 \pm .42$	38.8 ± 1.4	$18.1 \pm .42$
FISTA-Net	63.7 ± 1.1	$21.7 \pm .38$	55.2 ± 1.3	$20.2 \pm .36$	45.6 ± 1.3	$17.9 \pm .38$
iPiano-Net	63.9 ± 1.3	$20.4 \pm .49$	56.3 ± 1.6	$19.1 \pm .56$	45.7 ± 1.6	$17.7 \pm .49$
ISTA-Net^{+}	60.0 ± 1.6	$20.3 \pm .42$	54.5 ± 1.7	$19.6 \pm .45$	45.6 ± 1.5	$18.3 \pm .42$
iRadonMAP	12.9 ± 1.1	$12.5 \pm .34$	12.4 ± 1.1	$12.3 \pm .35$	$11.5 \pm .93$	$11.4 \pm .36$
ReconNet	$12.9 \pm .97$	$14.0 \pm .38$	$11.5 \pm .94$	$13.7 \pm .37$	$10.6 \pm .96$	$13.4 \pm .38$
Angles	15		10		6	
SNR	40 dB		40 dB		40 dB	
	SSIM ($\times 10^2$)	PSNR	SSIM ($\times 10^2$)	PSNR	SSIM ($\times 10^2$)	PSNR
CG-Net	$\mathbf{60.6 \pm 1.3}$	$\mathbf{22.0 \pm .41}$	$\mathbf{53.1 \pm 1.2}$	$\mathbf{20.7 \pm .41}$	$\mathbf{43.1 \pm 1.2}$	$\mathbf{18.9 \pm .41}$
LEARN^{++}	52.3 ± 1.4	$19.2 \pm .35$	48.7 ± 1.5	$18.9 \pm .37$	37.1 ± 1.3	$17.0 \pm .35$
LPD	60.7 ± 1.1	$20.7 \pm .36$	51.1 ± 1.2	$19.1 \pm .35$	43.5 ± 1.3	$17.6 \pm .35$
FBPConvNet	42.2 ± 1.5	$15.6 \pm .43$	32.3 ± 1.4	$13.7 \pm .39$	24.4 ± 1.1	$11.9 \pm .40$
MADUN	56.5 ± 1.2	$20.8 \pm .43$	47.9 ± 1.1	$19.6 \pm .40$	38.2 ± 1.4	$18.0 \pm .42$
FISTA-Net	59.6 ± 1.2	$21.0 \pm .35$	53.4 ± 1.2	$19.4 \pm .34$	44.4 ± 1.3	$17.9 \pm .38$
iPiano-Net	61.0 ± 1.2	$19.7 \pm .44$	54.0 ± 1.6	$18.7 \pm .54$	45.1 ± 1.6	$17.5 \pm .48$
ISTA-Net^{+}	58.9 ± 1.5	$20.2 \pm .41$	53.9 ± 1.7	$19.4 \pm .44$	44.5 ± 1.4	$18.1 \pm .42$
iRadonMAP	12.8 ± 1.1	$12.4 \pm .34$	12.3 ± 1.1	$12.3 \pm .35$	$11.3 \pm .92$	$11.4 \pm .35$
ReconNet	$11.3 \pm .92$	$13.5 \pm .36$	$11.0 \pm .91$	$13.6 \pm .38$	$9.40 \pm .90$	$13.1 \pm .38$

reconstructing 32×32 images on average. Similarly, CG-Net provides a 0.97dB improvement in PSNR for reconstructing 64×64 images on average. While the values in Table 2.9 and Table 2.10 may seem low, they are a result of training the models only on a very small training dataset. To this point, while CG-Net does not appreciably outperform CG-LS in the lowest training scenario it does so when provided enough training data. In particular, comparing the CG-LS results in Table 2.1 and the results in Fig. 2.7 and in Table 2.11 we see that with over 100 training samples CG-Net outperforms CG-LS. Nevertheless, as highlighted in Section 2.3.6, CG-Net provides a significant reconstruction time improvement over CG-LS while still providing comparable reconstruction quality even in low training.

For a visual comparison of the methods, Fig. 2.8 and Fig. 2.9 respectively show the reconstructions, plus SSIM values, of a 32×32 and 64×64 Barbara image snippet from a 60dB SNR Radon transform at 15 uniformly spaced angles. Additionally, Fig. 2.10 shows the reconstructions, plus SSIM values, of a 32×32 dog image from a 60dB SNR Radon transform at 10 uniformly spaced angles. Finally, Fig. 2.11 shows the reconstructions, plus SSIM values, of a 32×32 truck image from a 60dB SNR Radon transform at 6 uniformly spaced angles. Every test reconstruction in Fig. 2.8, Fig. 2.9, Fig. 2.10, and Fig. 2.11 are performed after training each method on a dataset of

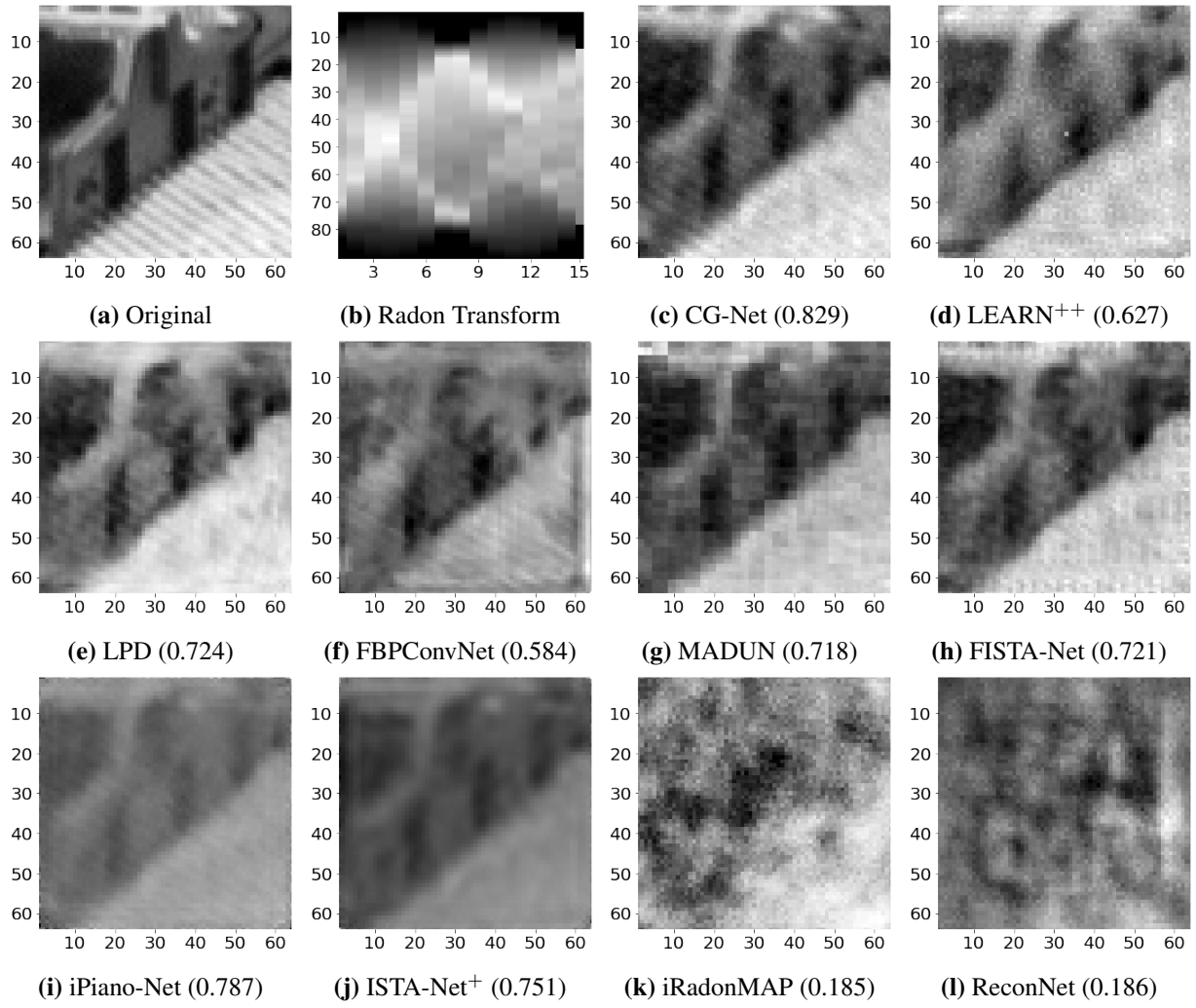


(a) Original **(b)** Radon Transform **(c)** CG-Net (0.954) **(d)** LEARN^{++} (0.764)

(e) LPD (0.862) **(f)** FBPConvNet (0.618) **(g)** MADUN (0.829) **(h)** FISTA-Net (0.832)

(i) iPiano-Net (0.864) **(j)** ISTA-Net^{+} (0.908) **(k)** iRadonMAP (0.501) **(l)** ReconNet (0.526)

Figure 2.8: Image reconstructions with SSIM in parentheses using CG-Net and nine other deep learning methods for 32×32 Barbara images. The sensing matrix, Ψ, is a Radon transform at 15 uniformly spaced angles, the dictionary, Φ, is a biorthogonal wavelet transformation, and each measurement has an SNR of 60dB. Our method CG-Net, shown in (2.8c) performs best. All test reconstructions were performed after training each method on a dataset of only 20 samples.



(a) Original **(b)** Radon Transform **(c)** CG-Net (0.829) **(d)** LEARN^{++} (0.627)

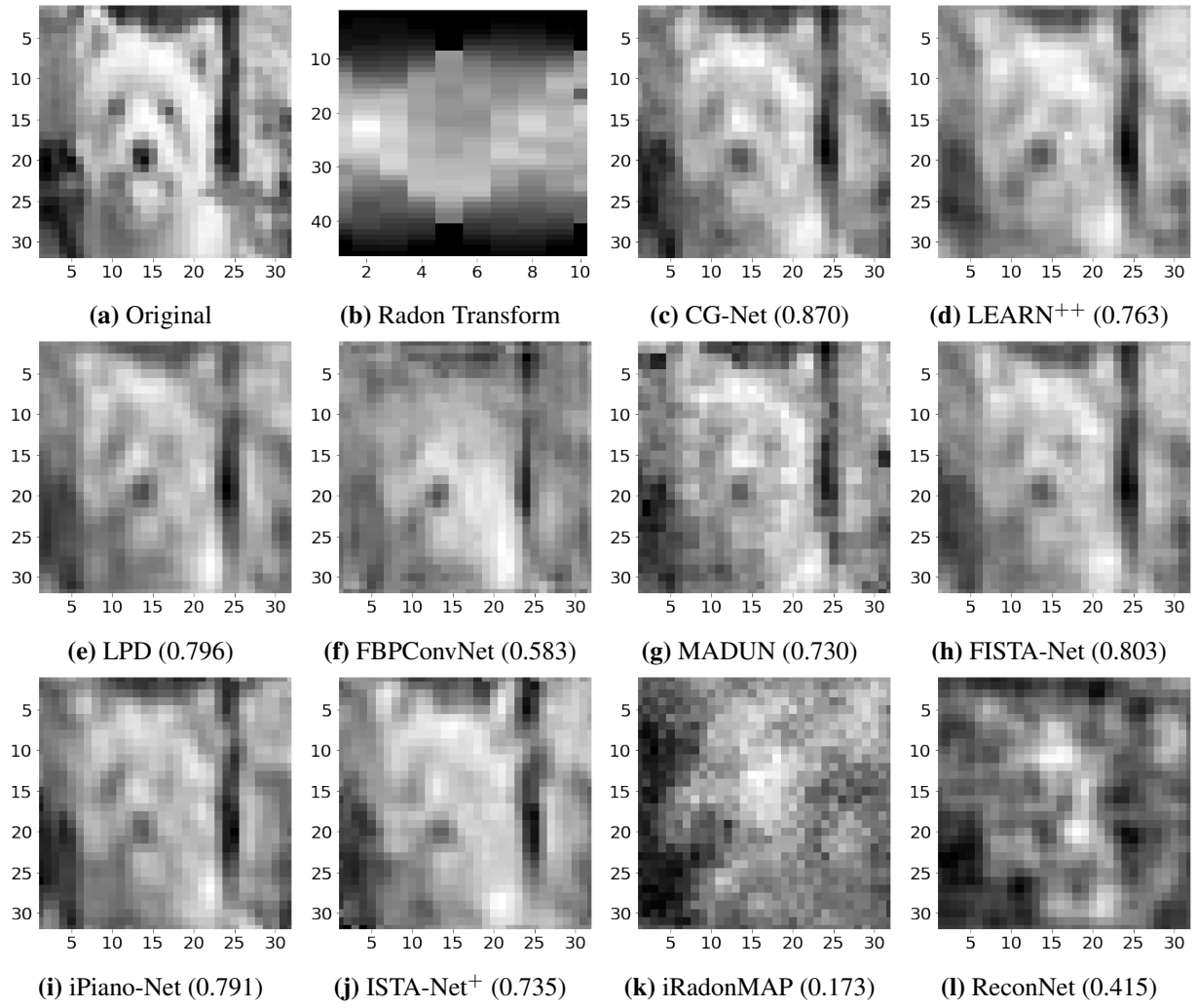
(e) LPD (0.724) **(f)** FBPConvNet (0.584) **(g)** MADUN (0.718) **(h)** FISTA-Net (0.721)

(i) iPiano-Net (0.787) **(j)** ISTA-Net^{+} (0.751) **(k)** iRadonMAP (0.185) **(l)** ReconNet (0.186)

Figure 2.9: Image reconstructions with SSIM in parentheses using CG-Net and nine other deep learning methods for 64×64 Barbara images. The sensing matrix, Ψ, is a Radon transform at 15 uniformly spaced angles, the dictionary, Φ, is a biorthogonal wavelet transformation, and each measurement has an SNR of 60dB. Our method CG-Net, shown in (2.9c), performs best. All test reconstructions were performed after training each method on a dataset of only 20 samples.

only 20 samples. In all figures, we see CG-Net producing higher quality images both visually and by SSIM over each of the nine prior art comparison methods.

We believe the high performance of CG-Net in low training scenarios is due to a couple of factors. First, the initialization of CG-Net corresponds precisely to CG-LS and, unlike other algorithm unrolling methods, we do not replace any part of the optimization with a CNN or other subnetwork structure. As these optimization-replacing subnetworks are learned completely from scratch, they often overfit quickly in low training. Second, CG-Net naturally incorporates the powerful statistical



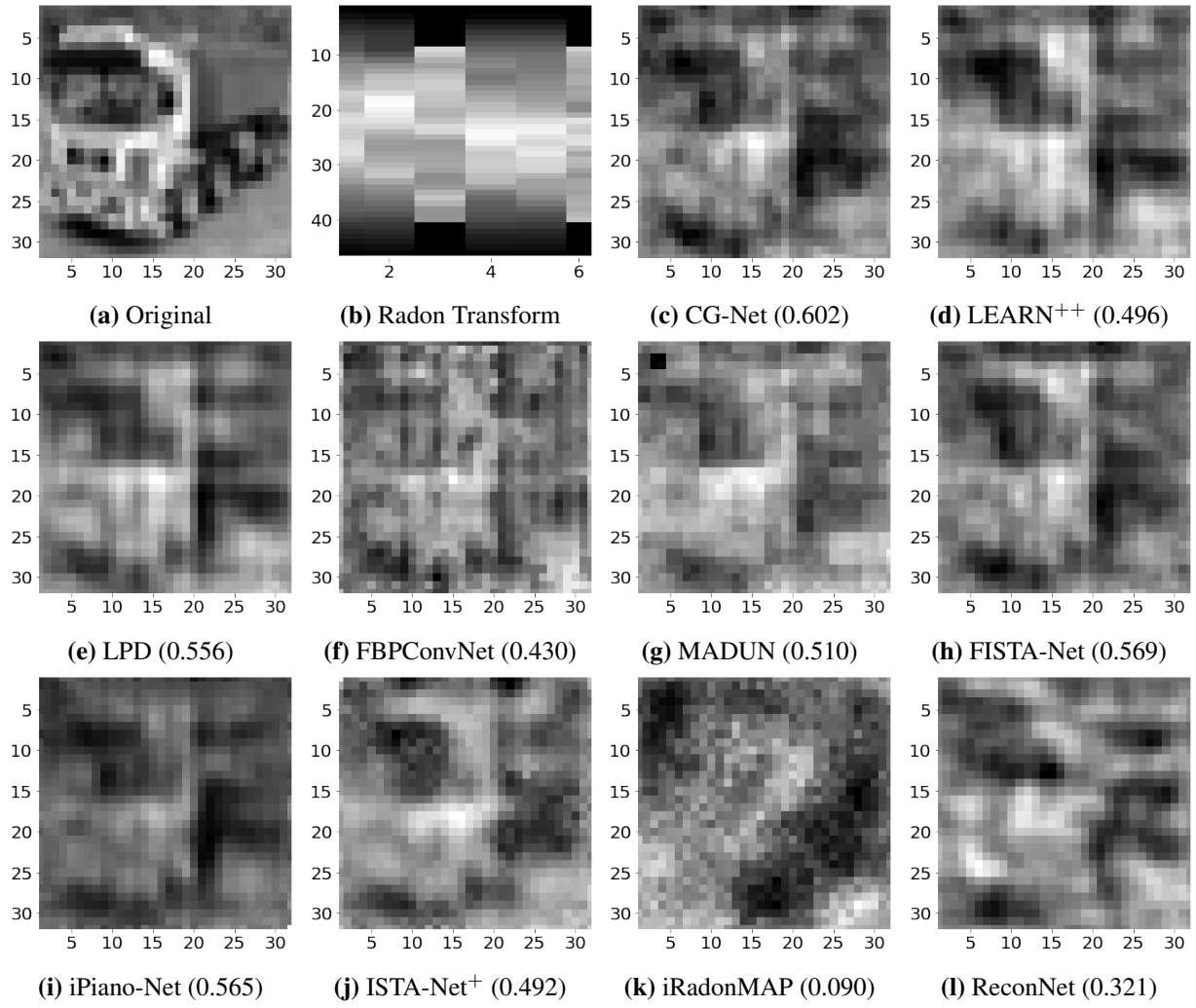
(a) Original **(b)** Radon Transform **(c)** CG-Net (0.870) **(d)** LEARN^{++} (0.763)

(e) LPD (0.796) **(f)** FBPConvNet (0.583) **(g)** MADUN (0.730) **(h)** FISTA-Net (0.803)

(i) iPiano-Net (0.791) **(j)** ISTA-Net^{+} (0.735) **(k)** iRadonMAP (0.173) **(l)** ReconNet (0.415)

Figure 2.10: Test image reconstructions, with SSIM value in parentheses, of a 32×32 dog image. Here, Ψ is a Radon transform at 10 uniformly spaced angles, Φ is a biorthogonal wavelet transformation, and each measurement has an SNR of 60dB. Our method CG-Net (2.10c) performs best. All test reconstructions were performed after training each method on a dataset of only 20 samples.

CG prior through the Tikhonov estimates and Hadamard product output, which provides beneficial data-consistency solution structure for low training. However, as discussed in Section 2.3.4, with greater noise and higher training, CG-Net may perform lower than the best performing comparative methods, possibly due to the enforced structure that makes CG-Net so successful in low training. Another consideration, as shown in Table 2.14, is that the model complexity of CG-Net may need to be increased in higher training by increasing the number of unrolled iterations.



(a) Original **(b)** Radon Transform **(c)** CG-Net (0.602) **(d)** LEARN^{++} (0.496)

(e) LPD (0.556) **(f)** FBPConvNet (0.430) **(g)** MADUN (0.510) **(h)** FISTA-Net (0.569)

(i) iPiano-Net (0.565) **(j)** ISTA-Net^{+} (0.492) **(k)** iRadonMAP (0.090) **(l)** ReconNet (0.321)

Figure 2.11: Test image reconstructions, with SSIM value in parentheses, for a 32×32 truck image. Here, Ψ is a 6 uniformly spaced angle Radon transform, Φ is a biorthogonal wavelet transformation, and measurement have an SNR of 60dB. Our method CG-Net (2.11c) performs best. All test reconstructions were performed after training each method on a dataset of only 20 samples.

To further highlight the applicability of CG-Net we consider alternative sensing matrices, Ψ, and dictionaries, Φ as summarized in Table 2.11. As typical in compressive sensing applications, we consider $\Psi \in \mathbb{R}^{m \times n}$ as a Gaussian matrix with a fixed sampling ratio defined as the ratio of measurement matrix dimensions $\frac{m}{n}$. Furthermore, we consider a discrete cosine dictionary (DCT) as an alternative representation dictionary to the biorthogonal wavelet dictionary. The training dataset for each experiment in Table 2.11 consists of 2000 measurement, coefficient pairs. Finally,

note that for each compressive sensing problem, we initialize every $\lambda_k = 10^2$ and otherwise retain the same setup and initialization for CG-Net as used in the Radon inversion simulations.

Table 2.11 provides the average SSIM and PSNR, across 200 test image reconstructions, with 99% confidence intervals for various choices of Ψ and Φ. As LEARN^{++}, LPD, FBPConvNet, and iRadonMAP are reconstruction methods specifically for computed tomography (CT) the compressive sensing problem is not applicable and thus these comparisons are omitted from Table 2.11. Again, CG-Net outperforms, or performs comparably to, each of the competitive methods in these alternative sensing matrices and dictionary schemes. We remark that an advantage of CG-Net is its applicability to any general linear inverse problem while many of the comparison methods are only for the specific problem of image estimation. It remains a point of future work to study CG-Net and the comparative methods for non-image estimation tasks.

Lastly, as shown in Figure 2.7, Table 2.9, Table 2.10 and Table 2.11 all of the unrolled DNN methods have a clear advantage over standard DNN methods in low training scenarios. This is, perhaps, unsurprising given that the unrolled DNN methods enforce some solution structure, through data consistency layers (typically a gradient step on a data fidelity measure), on the estimated output produced by the network. More specifically, by solution structure, or data consistency, we mean that the signal estimate produced by a DNN, $\widehat{\boldsymbol{c}}(\boldsymbol{y};\boldsymbol{\Theta})$, satisfies a degree of consistency with the input measurement $\boldsymbol{y}$ in that $\|\boldsymbol{y} - A\widehat{\boldsymbol{c}}(\boldsymbol{y};\boldsymbol{\Theta})\|_2 \leq \delta$ for some $\delta > 0$. This data consistency provides better initialization over standard DNN methods with no such required structure. That is, with no training the reconstructions from an unrolled DNN method, generally, are closer to the actual signal of interest as compared to the reconstructions from a standard DNN method. This likely leads to unrolled DNN methods requiring less learning for ample performance and thus succeeding in low training scenarios. To this point, the excellent performance of our CG-Net method in low training, as compared to state-of-the-art unrolled DNN and standard DNN methods, is expected as CG-Net enforces an appreciable solution structure, i.e. δ is a small value, by incorporating the CG prior through the Tikhonov update layers and Hadamard product output. However, given enough training data, standard DNN methods could outperform unrolled methods, which may now have

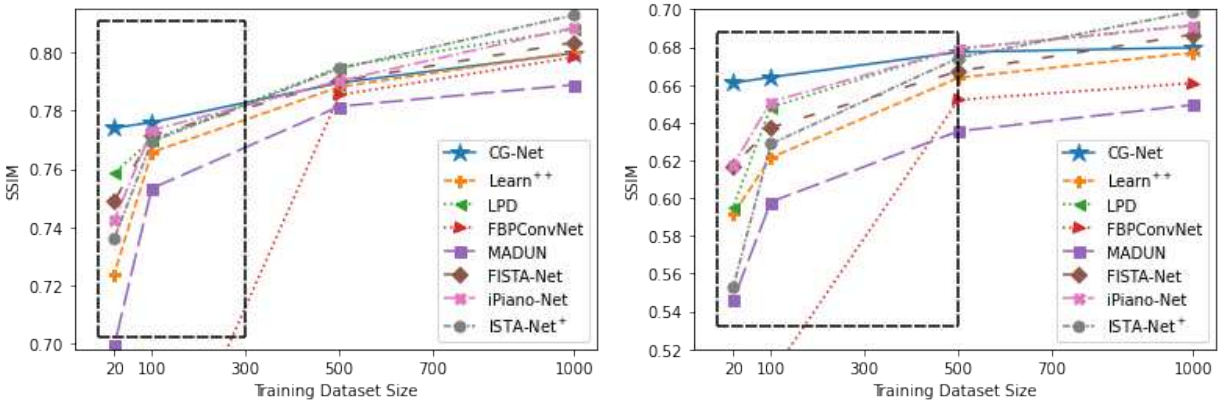
Table 2.11: Average SSIM $(\times 10^2)$ and PSNR with 99% confidence intervals for six deep learning based image reconstruction methods for training and testing on alternative sensing matrices, Ψ, and dictionaries, Φ. Here, $\Psi \in \mathbb{R}^{m \times n}$ is a Radon transform, at 15 or 10 uniformly spaced angles, or a Gaussian matrix, with 0.5 or 0.3 sampling ratio $\big($defined as $\frac{m}{n}\big)$, as in compressive sensing. Additionally, Φ is either a biorthogonal wavelet dictionary or discrete cosine transformation. Each measurement is corrupted with noise at an SNR of 60dB. Every method reconstructed two hundred, 32×32 CIFAR10 images after training on a set of 2000 samples. The samples in training and testing are produced from the same measurement and representation matrices. In all cases, our method CG-Net, highlighted in **bold**, performs better or comparably to all other approaches.

Φ	Discrete Cosine Transformation			
Ψ	15 Angles		10 Angles	
	SSIM $(\times 10^2)$	PSNR	SSIM $(\times 10^2)$	PSNR
CG-Net	$\mathbf{92.6 \pm .44}$	$\mathbf{29.2 \pm .44}$	$\mathbf{84.3 \pm .80}$	$\mathbf{25.7 \pm .44}$
MADUN	$90.9 \pm .53$	$28.1 \pm .42$	$82.5 \pm .84$	$25.1 \pm .42$
FISTA-Net	$91.6 \pm .53$	$27.8 \pm .41$	$83.5 \pm .92$	$25.0 \pm .41$
iPiano-Net	$92.9 \pm .57$	$28.9 \pm .49$	$85.0 \pm .88$	$25.5 \pm .42$
ISTA-Net$^+$	$92.7 \pm .48$	$28.7 \pm .44$	$85.2 \pm .85$	$25.6 \pm .42$
ReconNet	68.0 ± 1.3	$22.0 \pm .35$	64.2 ± 1.5	$21.5 \pm .38$
Φ	Biorthogonal Wavelet Dictionary			
Ψ	0.5 sampling ratio		0.3 sampling ratio	
	SSIM $(\times 10^2)$	PSNR	SSIM $(\times 10^2)$	PSNR
CG-Net	$\mathbf{84.4 \pm .80}$	$\mathbf{25.3 \pm .49}$	$\mathbf{70.7 \pm 1.1}$	$\mathbf{22.2 \pm .48}$
MADUN	$72.0 \pm .98$	$22.2 \pm .38$	62.8 ± 1.2	$20.9 \pm .41$
FISTA-Net	75.8 ± 1.1	$23.0 \pm .39$	66.5 ± 1.4	$21.4 \pm .42$
iPiano-Net	$88.7 \pm .64$	$26.5 \pm .39$	79.5 ± 1.0	$24.1 \pm .44$
ISTA-Net$^+$	$85.5 \pm .72$	$25.6 \pm .39$	$80.1 \pm .99$	$24.2 \pm .43$
ReconNet	77.4 ± 1.0	$23.4 \pm .40$	65.3 ± 1.3	$21.2 \pm .39$
Φ	Discrete Cosine Transformation			
Ψ	0.5 sampling ratio		0.3 sampling ratio	
	SSIM $(\times 10^2)$	PSNR	SSIM $(\times 10^2)$	PSNR
CG-Net	$\mathbf{89.4 \pm .72}$	$\mathbf{26.9 \pm .37}$	$\mathbf{78.8 \pm 1.1}$	$\mathbf{23.2 \pm .36}$
MADUN	$79.2 \pm .95$	$23.5 \pm .37$	70.6 ± 1.2	$21.9 \pm .41$
FISTA-Net	75.8 ± 1.1	$23.0 \pm .39$	66.5 ± 1.4	$21.4 \pm .42$
iPiano-Net	$88.7 \pm .64$	$26.5 \pm .39$	79.5 ± 1.0	$24.1 \pm .44$
ISTA-Net$^+$	$85.5 \pm .72$	$25.6 \pm .39$	$80.1 \pm .99$	$24.2 \pm .43$
ReconNet	77.4 ± 1.0	$23.4 \pm .40$	65.3 ± 1.3	$21.2 \pm .39$

hindered learning capacities due to the required solution structure. Full coverage of a comparison between unrolled and standard DNN methods is a topic we leave for future study.

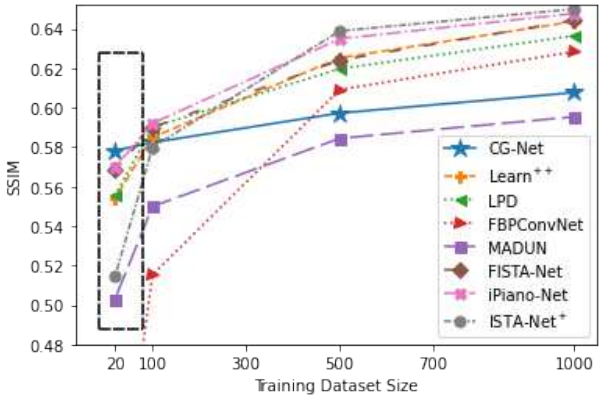
2.3.4 Impact of Measurement Noise

Shown in Section 2.3.3, CG-Net significantly outperforms many state-of-the-art deep learning methods in image reconstruction in the presence of low measurement noise. However, when the amount of measurement noise is increased, the performance of CG-Net may falter as compared to the state-of-the-art deep learning methods. Figure 2.12 shows the average reconstruction SSIM for CG-Net and all nine comparison methods when every method reconstructs two hundred, 32×32 CIFAR10 [69] images from 10 or 6 uniformly spaced Radon transform measurements where each measurement is corrupted by noise at an SNR of 40dB or 30dB. We observe that in the



(a) Ten uniformly spaced angles and 40dB SNR.

(b) Six uniformly spaced angles and 40dB SNR.



(c) Six uniformly spaced angles and 30dB SNR.

Figure 2.12: Study of the impact that higher measurement noise has on CG-Net. Displayed is the average test image reconstruction SSIM when we vary the amount of CIFAR10 [69] data in training eight machine learning-based image reconstruction methods. Here, the sensing matrices, Ψ, are a Radon transform at 10 or 6 uniformly spaced angles and the sparsity dictionary, Φ, is a biorthogonal wavelet transform. Measurements in training and testing have an SNR of 40dB or 30dB.

lowest training scenarios, of 20 and 100 training data samples, CG-Net still outperforms all of the comparison deep learning methods. However, with more than 100 training samples, CG-Net is outperformed, sometimes appreciably, by the comparison deep learning methods.

A partial reason that CG-Net is adversely affected by measurement noise may be attributed to the strong solution structure or data consistency imposed by the Tikhonov update layers. Specifically, the Tikhonov update layers closely match the measured estimated signal to the actual measurements, which may be disadvantageous when the actual measurements have a higher amount of additive noise. As such, CG-Net may be attempting to fit to the measurement noise rather than learning how to remove it from the estimate signal, as is done in alternative deep learning methods, especially in higher training scenarios. Nevertheless, as shown in Table 2.14, CG-Net has, by far, the fewest number of trainable parameters of the compared deep learning methods. Thus, increasing the model complexity of CG-Net, by increasing the number of unrolled iterations, may be necessary in these higher noise and higher training data scenarios as then CG-Net would have greater learning capacities to better capture underlying properties of the signals-of-interest.

2.3.5 Ablation Study

We consider the effect of removing the learned steepest descent matrix B_k^j from CG-Net by fixing it during training. Two possibilities are employed, a gradient CG-Net (gCG-Net) where $B_k^j = I$ and a Newton CG-Net (nCG-Net) where $B_k^j = \left(\boldsymbol{J}_{\boldsymbol{z}}(\nabla_{\boldsymbol{z}} F(\boldsymbol{u}_{k-1}, \boldsymbol{z}_k^{j-1}))\right)^{-1}$ where $\boldsymbol{J}_{\boldsymbol{z}}(\nabla_{\boldsymbol{z}} F(\boldsymbol{u}, \boldsymbol{z}))$ is given in equation (2.11). Note, the nCG-Net was first proposed in [57] where it was empirically shown to outperform many prior art iterative image reconstruction methods. All other aspects of training gCG-Net and nCG-Net are identical to that of training CG-Net in Section 2.3.3 except for nCG-Net we scale the input measurements by e^{-4} and initialize $a_0 = a_k^{(j)} = 1$ and $b_0 = b_k^{(j)} = e$.

Shown in Table 2.12 is the average SSIM of 200 test image reconstructions – from Radon transforms at 15, 10, and 6 uniformly spaced angles with an SNR of 60dB or 40dB – when we vary the amount of training data. With fewer than 100 training data samples both gCG-Net and nCG-Net structures perform comparably or outperform the fully general version of CG-Net. Likely, this

is due to gCG-Net and nCG-Net having fewer parameters to be fit for these cases and thus avoiding overfitting. However, with more than 100 training data samples CG-Net outperforms both gCG-Net and nCG-Net. Again, this is likely due to gCG-Net and nCG-Net having fewer parameters to be fit and thus being unable to extract as much information from larger training datasets as compared to CG-Net.

Table 2.12: Ablation study for CG-Net. Average SSIM ($\times 10^2$) and PSNR for 32×32 image reconstructions from a Radon transform, at several different amounts of uniformly spaced angles, with a set SNR. We find that gCG-Net and nCG-Net outperform the full version of CG-Net in the lowest training scenario since fewer parameters must be fit for these cases.

Method	(Angles,SNR)	Training Dataset Size					
		20		100		500	
		SSIM ($\times 10^2$)	PSNR	SSIM ($\times 10^2$)	PSNR	SSIM ($\times 10^2$)	PSNR
gCG-Net	(15,60)	90.0	27.7	90.8	28.2	91.2	28.5
nCG-Net		90.7	28.1	90.7	28.2	90.8	28.2
CG-Net		89.9	27.7	90.8	28.1	92.0	28.8
gCG-Net	(10,60)	81.6	24.9	82.4	25.1	82.8	25.3
nCG-Net		81.7	24.9	81.8	24.9	82.0	25.0
CG-Net		81.5	24.8	82.4	25.1	83.6	25.5
gCG-Net	(6,60)	67.6	21.8	68.1	21.8	68.7	22.0
nCG-Net		67.1	21.7	67.2	21.7	67.9	21.8
CG-Net		67.6	21.8	68.1	21.9	69.5	22.1
gCG-Net	(6,40)	66.1	21.6	66.4	21.6	66.7	21.6
nCG-Net		65.5	21.5	65.6	21.5	66.2	21.6
CG-Net		66.1	21.6	66.4	21.6	67.7	21.7

2.3.6 Comparison of Computational Time and Parameters

Table 2.13 lists the average computational time per image, in milliseconds, across 200 test image reconstructions for CG-Net and all nine comparison deep learning methods. For fair comparison, the reconstructions for each method were run on the same 64-bit Intel(R) Xeon(R) CPU E5-2690 machine. In comparing Table 2.5 and Table 2.13, we observed that the required computational time for CG-LS is reduced by more than a factor of 100 for CG-Net while producing the same or slightly improved quality image reconstructions as compared to CG-LS. This reduced

Table 2.13: Average reconstruction time of 32×32 images from Radon transform measurements at 15 uniform angles for our CG-Net and nine comparison deep learning based image reconstruction methods.

Method	CG-Net	ReconNet	LPD	LEARN^{++}	iRadonMAP
Time (ms)	765	0.65	4.7	10.6	4.5
Method	iPiano-Net	FISTA-Net	MADUN	ISTA-Net^{+}	FBPConvNet
Time (ms)	13.0	6.4	26.0	7.9	2.0

computational time of CG-Net is one of the primary advantages in using CG-Net over using CG-LS.

Nevertheless, CG-Net lags in computational time against the comparative DNN methods that take a fraction of the time to reconstruct an image. This is likely an outcome of the required linear solver used to calculate the inverse in (2.6) to update $\boldsymbol{u}$ and the required eigendecomposition in (2.32). Instead, the comparative methods solely consist of convolutions or matrix, vector products that are appreciably faster to implement than linear solvers.

Improving the speed of CG-Net serves as one direction of future work for which we suggest a technique to remove some costly computations in CG-Net after training. For training CG-Net, the eigendecomposition in equation (2.32) must be implemented within each Z_k^j layer call as the entries of L_k^j are actively being updated and thus, we need to actively ensure $(L_k^j + (L_k^j)^T)/2$ stays positive definite. Using CG-Net post-training, the eigendecomposition only needs to be implemented once upon instantiating the model with pre-trained network parameters, thereby removing many eigendecomposition calculations and thus increasing the speed of CG-Net.

Finally, Table 2.14 provides the number of parameters for CG-Net and all nine comparative deep learning-based methods for linear inverse problems. We remark that CG-Net has the fewest parameters to be trained out of the compared methods, which may be a contributing factor to the success of CG-Net when small training datasets are used. Increasing the number of unrolled iterations in CG-Net to produce a DNN with a model complexity matching the average of the compared methods remains a point of future work. In particular, with larger training datasets, CG-Net can be outperformed by the best-performing comparative method. Thus, in these scenarios, we

Table 2.14: Parameter count for our CG-Net and nine comparison deep learning methods in the reconstruction of a 32×32 image from Radon transform measurements at 15 uniformly spaced angles.

Method	Parameters ($\times 10^5$)	Method	Parameters ($\times 10^5$)
CG-Net	0.62	ISTA-Net$^+$	3.37
LPD	2.53	MADUN	29.7
LEARN^{++}	12.0	FISTA-Net	0.75
iRadonMAP	8.33	iPiano-Net	19.3
FBPConvNet	7.09	ReconNet	7.30

may be required to increase the number of unrolled iterations in CG-Net, thereby increasing the number of learned parameters, to more adequately match the results of the comparative methods.

We remark that, for each each deep learning method, the ratio of unknowns to data points is proportional to the values presented in Table 2.14. That is, we can obtain the ratio of unknowns to measurements by dividing each entry of Table 2.14 by the number of data points used in training the deep learning methods. As an example, the number of data points used when we train with 20 CIFAR10 images to reconstruct from Radon transforms at 15 uniformly spaced angles is $20 \times$ (image size+Radon transform size) $= 20 \times (1024 + 690) = 3.428 \times 10^4$. Therefore, the ratio of unknowns to measurements for CG-Net is $0.62 \times 10^5 / 3.428 \times 10^4 \approx 1.81$.

2.4 Conclusion

Informed by the powerful statistical representation of signals through the compound Gaussian prior, we have developed two novel approaches to signal reconstruction in linear inverse problems. The first is an iterative signal reconstruction algorithm, named CG-LS, that enforces the CG prior. The second is a deep neural network implementation of CG-LS, constructed through algorithm unrolling, named CG-Net. Both approaches have been published in our conference proceedings of the IEEE Asia Pacific Signal and Information Processing Association conference [57] and in our IEEE Transactions on Signal Processing journal article [58]. Furthermore, a provisional patent of our two innovative methods has been filed.

The CG-LS algorithm is based upon a regularized least squares estimate of the signal change-of-basis coefficients where the regularization, equivalent to the negative log prior from a MAP

estimate, is chosen to capture the fundamental statistics of the CG prior. We have conducted a rigorous theoretical characterization of CG-LS, which gave important insights into the implementation of the algorithm. We completed ample numerical validation of CG-LS, which showed a significant improvement over other state-of-the-art image reconstruction algorithms.

Next, we have applied algorithm unrolling to CG-LS, creating a new deep neural network named CG-Net. To the best of our knowledge, CG-Net is the first unrolled DNN for natural and tomographic image reconstruction to be fundamentally informed by a CG prior. Multiple datasets were used to train and test CG-Net where, after each training, we have shown that CG-Net outperforms other state-of-the-art unrolled DNNs and standard DNNs. In particular, CG-Net significantly outperforms prior art deep learning methods in low training contexts for both tomographic imaging and compressive sensing applications. However, it was also demonstrated that in the presence of higher measurement noise or larger training datasets, CG-Net is outperformed by the best performing deep learning-based signal reconstruction method. Finally, we completed a comparison study on the computational time to reconstruct a single image from each iterative and DNN method. We have shown that CG-Net significantly improved upon the necessary time to reconstruct an image over CG-LS, but still falls short of the speed other DNN methods achieve for an image reconstruction.

In the next chapter, we expand upon the generalizability of the two CG-based methodologies we have proposed in this chapter by replacing aspects of the CG-based optimization with relevant neural network structures. In particular, CG-LS is expanded by a generalized regularization such that greater information can be supplied into the Gaussian variable covariance and scale variable prior distribution on a problem specific basis. Subsequently, unrolling this generalized CG-LS method, replaces the scale variable prior information, e.g. the choice of nonlinearity h, with a subnetwork embedded inside of the unrolled deep neural network. This extension of CG-Net expands upon its applicability as it no longer requires a user-specified function for h and provides a greater learning capacity.

Chapter 3

Deep Compound Gaussian Regularized Inverse Problems

Inspired by the statistical richness of the compound Gaussian (CG) class of distributions in representing images [10–12] and by the success of deep learning-based approaches in solving inverse problems [13–20, 22–26, 30, 32, 33, 55, 56] we developed a novel unrolled-compound-Gaussian-prior-based deep neural network in Chapter 2. This compound Gaussian network (CG-Net), while outperforming many previous state-of-the-art methods in low measurement noise and small training dataset scenarios can be outperformed, sometimes considerably, when higher measurement noise is present or larger training datasets are available. Part of problem may be attributed to the small parameter count of CG-Net, but increasing the parameters through unrolling a larger number of iterations exacerbates the, already problematic, reconstruction time requirements for CG-Net. In this chapter, we have proposed, theoretically evaluated, and empirically evaluated a novel modification to the iterative compound Gaussian least squares (CG-LS) algorithm and unrolled CG-Net that assists in the high measurement noise, large training datasets, and significant reconstruction time problems present in CG-Net.

First, we name the novel modified CG-LS algorithm as the generalized compound Gaussian least squares (G-CG-LS) algorithm. Instead of constraining the scale variable, $\boldsymbol{z}$, to be a non-linear function of a Gaussian, e.g. a log-normal variable in CG-LS, we now have implemented an implicit regularization function to be specified on a problem-specific basis. The “generalized" in G-CG-LS denotes the fact that, with defining the scale variable regularization to be log-normal, G-CG-LS reduces to CG-LS as a special case.

Second, we have applied algorithm unrolling to G-CG-LS creating a new DNN we name deep regularized compound Gaussian network (DR-CG-Net). Unlike CG-Net, where the scale variable distribution continues to be fixed as log-norm, the implicit regularization of G-CG-LS is intro-

duced as a subnetwork within DR-CG-Net and is learned during the training process. That is, the regularization and thus distribution of the scale variable are learned by the unrolled DR-CG-Net. The “deep regularization" naming of DR-CG-Net is attributed to the representation of the scale variable regularization as another deep neural network.

In this chapter, we consider the same linear measurement equation used in Chapter 2. For sake of chapter completeness, we restate the measurement equation here. While many inverse problems are non-linear, we focus on linear inverse problems as this is frequently the assumed measurement model in many applications – including compressive sensing, computed tomography (CT), and magnetic resonance imaging (MRI) – and leave non-linear inverse problem applications to future work. Working from the linear measurement equation (1.6), let $\boldsymbol{x} \in \mathbb{R}^n$ be a vectorized signal that has a representation $\boldsymbol{x} = \Phi \boldsymbol{c}$ with respect to (w.r.t.) a dictionary, $\Phi \in \mathbb{R}^{n \times n}$, and coefficients, $\boldsymbol{c} \in \mathbb{R}^n$. An example, Φ is a wavelet transform and $\boldsymbol{c}$ the wavelet coefficients of signal $\boldsymbol{x}$. Consider linear measurements $\boldsymbol{y} \in \mathbb{R}^m$ given by $\boldsymbol{y} = \Psi\Phi\boldsymbol{c} + \boldsymbol{\nu}$ for additive white noise, $\boldsymbol{\nu} \in \mathbb{R}^m$. Now, we decompose $\boldsymbol{c}$ according to the CG prior; that is, $\boldsymbol{c} = \boldsymbol{z} \odot \boldsymbol{u}$ for $\boldsymbol{u} \sim \mathcal{N}(\boldsymbol{0}, \Sigma_u)$, $\boldsymbol{z}$ and $\boldsymbol{u}$ are independent random vectors, and $\boldsymbol{z} \sim p_{\boldsymbol{z}}$ – for some density $p_{\boldsymbol{z}}$ – is a positive random vector. The linear measurement model we consider then is

$$\boldsymbol{y} = \Psi\Phi(\boldsymbol{z} \odot \boldsymbol{u}) + \boldsymbol{\nu} \tag{3.1}$$

and the linear inverse problem to (3.1) aims to recover the product $\boldsymbol{z} \odot \boldsymbol{u}$, through the estimation of $\boldsymbol{z}$ and $\boldsymbol{u}$, given $\boldsymbol{y}$, Ψ, and Φ.

The new contributions of this chapter are succinctly presented in our proceedings of the IEEE Asilomar Conference on Signals, Systems, and Computers [76] and in our IEEE Transactions on Computational Imaging journal article [77]. We present additional explanation, details, and numerical results in this chapter to those found in our publications [76, 77]. Specifically, in this chapter we have completed:

1. The construction of a novel generalization on the CG prior informed iterative signal estimation algorithm from [57]. Our algorithm, called generalized compound Gaussian least squares (G-CG-LS), is a regularized least squares optimization that solves the linear inverse problem to (3.1) with general Ψ and Φ matrices. The regularization of G-CG-LS consists of a Gaussian term and an implicitly defined term, which together enforce a CG prior.

2. The derivation of new encompassing theoretical convergence analysis of G-CG-LS to stationary points for two distinct scale variable descent methods; namely, projected gradient descent (PGD) and the iterative shrinkage and thresholding algorithm (ISTA).

3. The construction of novel unrolled DNN versions of G-CG-LS, called the deep regularized compound Gaussian network (DR-CG-Net), such that the implicit portion of the regularization in G-CG-LS is trainable. A PGD DR-CG-Net and ISTA DR-CG-Net have been proposed, which correspond to a PGD or ISTA scale variable descent method, respectively. Our unrolled DR-CG-Nets allow for learning of the prior density within the framework of the powerful CG class of distributions.

4. A presentation of copious new empirical results for DR-CG-Net on tomographic imaging and compressive sensing problems as the size of the training dataset is altered. The effectiveness of DR-CG-Net to reconstruct images after training has been evaluated and shown to outperform other state-of-the-art imaging algorithms, including CG-Net, especially in low-training scenarios.

3.1 Notation and Nomenclature

The set of symbols provided in this section are defined here for ease of referencing and in reading the remainder of this chapter. We note that some have already been defined previously.

$\mathbb{R}$	$=$	Set of real numbers.
$\boldsymbol{v}$	$=$	$[v_i]_{i=1,2,\ldots,n} \in \mathbb{R}^n$. Boldface characters are vectors.
M	$=$	$[M_{ij}]_{i=1,2,\ldots,n}^{j=1,2,\ldots,m} \in \mathbb{R}^{m\times n}$. Uppercase characters are matrices.
$(\cdot)^T$	$=$	Transpose of vector or matrix $(\cdot)$
$\odot$	$=$	Hadamard product.
$D(\boldsymbol{v})$	$=$	Diagonal matrix with entries $v_1, v_2, \ldots, v_n$ on the diagonal.
$M_{\boldsymbol{v}}$	$=$	$MD(\boldsymbol{v})$ for $M \in \mathbb{R}^{m\times n}$ and $\boldsymbol{v} \in \mathbb{R}^n$.
$f(\boldsymbol{v})$	$=$	$[f(v_i)]_{i=1,2,\ldots,n} \in \mathbb{R}^n$ for $\boldsymbol{v} \in \mathbb{R}^n$ and a **componentwise function** $f : \mathbb{R} \to \mathbb{R}$.
$\text{ReLU}(x)$	$=$	$\max\{0, x\}$ is a componentwise function.
$\mathcal{P}_{\mathcal{C}}$	$=$	Unique project operator onto convex set $\mathcal{C}$.
Ψ	$\in$	$\mathbb{R}^{m\times n}$ is a measurement or observation matrix.
Φ	$\in$	$\mathbb{R}^{n\times n}$ is a change-of-basis matrix or dictionary.
A	$=$	$\Psi\Phi$.
$A_{\boldsymbol{v}}$	$=$	$AD(\boldsymbol{v})$.

3.2 Generalized Compound Gaussian Least Squares (G-CG-LS)

3.2.1 G-CG-LS Implementation Details

Inspired by the compound Gaussian least squares (CG-LS) algorithm in Chapter 2, and work in [57, 58, 76], we consider a maximum a posteriori (MAP) estimate of the scale variable, $\boldsymbol{z}$, and Gaussian variable, $\boldsymbol{u}$, from equation (3.1). For this let $p_{\boldsymbol{x}}(\boldsymbol{x})$ be the probability density function of random vector $\boldsymbol{x}$ and let $p_{\boldsymbol{x}|\boldsymbol{v}}(\boldsymbol{x}|\boldsymbol{v})$ be the conditional probability density given $\boldsymbol{v}$. Additionally, let $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$ be a multivariate normal distribution of mean $\boldsymbol{\mu}$ and covariance Σ. As $\boldsymbol{\nu}$, from equation (3.1), is a vector of white Gaussian noise then $\boldsymbol{\nu} \sim \mathcal{N}(\boldsymbol{0}, \sigma^2 I)$ for some $\sigma \geq 0$. Thus

$$p_{\boldsymbol{y}|\boldsymbol{u},\boldsymbol{z}}(\boldsymbol{y}|\boldsymbol{u},\boldsymbol{z}) = p_{\boldsymbol{y}|\boldsymbol{u},\boldsymbol{z}}(A(\boldsymbol{z}\odot\boldsymbol{u}) + \boldsymbol{\nu}|\boldsymbol{u},\boldsymbol{z}) \sim \mathcal{N}(A(\boldsymbol{z}\odot\boldsymbol{u}), \sigma^2 I)$$

for $A = \Psi\Phi$. Since $\boldsymbol{u} \sim \mathcal{N}(\mathbf{0}, \Sigma_u)$ then using independence of $\boldsymbol{u}$ and $\boldsymbol{z}$ the joint MAP estimate of $\boldsymbol{z}$ and $\boldsymbol{u}$ from equation (3.1) is

$$\begin{aligned}
\arg\max_{\boldsymbol{u},\boldsymbol{z}} \, p_{\boldsymbol{u},\boldsymbol{z}|\boldsymbol{y}}(\boldsymbol{u},\boldsymbol{z}|\boldsymbol{y}) &= \arg\max_{\boldsymbol{u},\boldsymbol{z}} \, \frac{p_{\boldsymbol{y}|\boldsymbol{u},\boldsymbol{z}}(\boldsymbol{y}|\boldsymbol{u},\boldsymbol{z}) p_{\boldsymbol{u},\boldsymbol{z}}(\boldsymbol{u},\boldsymbol{z})}{p_{\boldsymbol{y}}(\boldsymbol{y})} \\
&= \arg\max_{\boldsymbol{u},\boldsymbol{z}} \, p_{\boldsymbol{y}|\boldsymbol{u},\boldsymbol{z}}(\boldsymbol{y}|\boldsymbol{u},\boldsymbol{z}) p_{\boldsymbol{u},\boldsymbol{z}}(\boldsymbol{u},\boldsymbol{z}) \\
&= \arg\max_{\boldsymbol{u},\boldsymbol{z}} \, \ln\left(p_{\boldsymbol{y}|\boldsymbol{u},\boldsymbol{z}}(\boldsymbol{y}|\boldsymbol{u},\boldsymbol{z})\right) + \ln\left(p_{\boldsymbol{u},\boldsymbol{z}}(\boldsymbol{u},\boldsymbol{z})\right) \\
&= \arg\min_{\boldsymbol{u},\boldsymbol{z}} \, -\ln(p_{\boldsymbol{y}|\boldsymbol{u},\boldsymbol{z}}(\boldsymbol{y}|\boldsymbol{u},\boldsymbol{z})) - \ln(p_{\boldsymbol{u},\boldsymbol{z}}(\boldsymbol{u},\boldsymbol{z})) \\
&= \arg\min_{\boldsymbol{u},\boldsymbol{z}} \, -\ln(p_{\boldsymbol{y}|\boldsymbol{u},\boldsymbol{z}}(\boldsymbol{y}|\boldsymbol{u},\boldsymbol{z})) - \ln(p_{\boldsymbol{u}}(\boldsymbol{u})) - \ln(p_{\boldsymbol{z}}(\boldsymbol{z})) \\
&= \arg\min_{\boldsymbol{u},\boldsymbol{z}} \, \frac{1}{2}||\boldsymbol{y} - A(\boldsymbol{z} \odot \boldsymbol{u})||_2^2 + \frac{1}{2}\boldsymbol{u}^T P_u^{-1}\boldsymbol{u} + \mathcal{R}(\boldsymbol{z})
\end{aligned}$$

where $\mathcal{R}(\boldsymbol{z}) = -\sigma^2 \ln(p_{\boldsymbol{z}}(\boldsymbol{z}))$ and $P_u = \sigma^{-2}\Sigma_u$. Therefore, defining the cost function

$$F(\boldsymbol{u},\boldsymbol{z}) = \frac{1}{2}\|\boldsymbol{y} - A(\boldsymbol{z} \odot \boldsymbol{u})\|_2^2 + \frac{1}{2}\boldsymbol{u}^T P_u^{-1}\boldsymbol{u} + \mathcal{R}(\boldsymbol{z}), \tag{3.2}$$

we consider the estimates

$$\begin{bmatrix}\boldsymbol{u}^* & \boldsymbol{z}^*\end{bmatrix} = \underset{(\boldsymbol{u}\ \boldsymbol{z})\in\mathbb{R}^n\times\mathfrak{Z}}{\arg\min} \; F(\boldsymbol{u},\boldsymbol{z}) \tag{3.3}$$

where $\mathfrak{Z} \subseteq [0,\infty)^n$ is the domain of $\mathcal{R}$, which we assume to be convex. Our solution for the linear inverse problem to (1.1) is then given as

$$\boldsymbol{c}^* = \boldsymbol{z}^* \odot \boldsymbol{u}^*.$$

We remark that the term $\frac{1}{2}\|\boldsymbol{y} - A(\boldsymbol{z} \odot \boldsymbol{u})\|_2^2$ in equation (3.2) is the data fidelity term while the remaining two terms are regularization terms. Our G-CG-LS algorithm, given in Algorithm 3, is an iterative method to approximately solve (3.3) through block coordinate descent [59].

To further detail Algorithm 3, define $A_{\boldsymbol{z}} = A\text{Diag}(\boldsymbol{z})$ and note that the minimization of (3.2) w.r.t. $\boldsymbol{u}$ is a Tikhonov regularization, or ridge regression, problem with minimizer $\mathcal{T}(\boldsymbol{z}) \equiv \mathcal{T}(\boldsymbol{z}; P_u)$, which we call the Tikhonov solution, defined as

$$\begin{aligned}
\mathcal{T}(\boldsymbol{z}) \coloneqq \mathcal{T}(\boldsymbol{z}; P_u) &= (A_{\boldsymbol{z}}^T A_{\boldsymbol{z}} + P_u^{-1})^{-1} A_{\boldsymbol{z}}^T \boldsymbol{y} && (3.4)\\
&= \left(P_u - P_u A_{\boldsymbol{z}}^T (I + A_{\boldsymbol{z}} P_u A_{\boldsymbol{z}}^T)^{-1} A_{\boldsymbol{z}} P_u\right) A_{\boldsymbol{z}}^T \boldsymbol{y} \\
&= P_u A_{\boldsymbol{z}}^T \left(I - (I + A_{\boldsymbol{z}} P_u A_{\boldsymbol{z}}^T)^{-1} A_{\boldsymbol{z}} P_u A_{\boldsymbol{z}}^T\right) \boldsymbol{y} \\
&= P_u A_{\boldsymbol{z}}^T (I + A_{\boldsymbol{z}} P_u A_{\boldsymbol{z}}^T)^{-1} \left(I + A_{\boldsymbol{z}} P_u A_{\boldsymbol{z}}^T - A_{\boldsymbol{z}} P_u A_{\boldsymbol{z}}^T\right) \boldsymbol{y} \\
&= P_u A_{\boldsymbol{z}}^T (I + A_{\boldsymbol{z}} P_u A_{\boldsymbol{z}}^T)^{-1} \boldsymbol{y} && (3.5)
\end{aligned}$$

where we used the Woodbury matrix identity in the second line. In practice, we do not calculate the inverse matrix in equation (3.4) and instead solve a system of linear equations. Furthermore, equation (3.4) is a $n \times n$ system of linear equations, where n is the signal of interest size, which is reduced to a $m \times m$ system of linear equation in equation (3.5), where m is the measurement size. Typically, $m \ll n$ and thus equation (3.5) is significantly faster to implement than equation (3.4) in constructing the Tikhonov solution.

Next, for minimizing (3.2) w.r.t. $\boldsymbol{z}$, a number, J, of descent steps are iteratively applied. Let $g(\boldsymbol{z}, \boldsymbol{u}) : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n$ be a descent function of (3.2) w.r.t. $\boldsymbol{z}$, which we call the scale variable update function. Two possibilities for g that we consider are discussed in Section 3.2.2. Then, the G-CG-LS estimate of $\boldsymbol{z}$ on descent step j of iteration k and the G-CG-LS estimate of $\boldsymbol{u}$ on iteration k are given respectively by

$$\boldsymbol{z}_k^{(j)} = g(\boldsymbol{z}_k^{(j-1)}, \boldsymbol{u}_{k-1}) \quad \text{and} \quad \boldsymbol{u}_k = \mathcal{T}(\boldsymbol{z}_k; P_u).$$

Note that we define $\boldsymbol{z}_k^{(J)} = \boldsymbol{z}_{k+1}^{(0)} \equiv \boldsymbol{z}_k$ and that g is additionally parameterized by the measurement $\boldsymbol{y}$ but this is omitted for simplicity.

Algorithm 3 Generalized Compound Gaussian Least Squares (G-CG-LS)

Input: Measurement $\boldsymbol{y}$, measurement operator A, iterations K, scale variable update steps J, and projection operator $\mathcal{P}_{[0,b]^n}$

$\boldsymbol{z}_0 = \mathcal{P}_{[0,b]^n}(A^T\boldsymbol{y})$ and $\boldsymbol{u}_0 = \mathcal{T}(\boldsymbol{z}_0)$ (or $\widetilde{\mathcal{T}}(\boldsymbol{0}, \boldsymbol{z}_0)$ for NAGD)
for $k \in \{1, 2, \ldots, K\}$ **do**
  $\underline{\boldsymbol{z} \text{ ESTIMATION}}$:
  $\boldsymbol{z}_k^{(0)} = \boldsymbol{z}_{k-1}$
  for $j \in \{1, 2, \ldots, J\}$ **do**
    $\boldsymbol{z}_k^{(j)} = g(\boldsymbol{z}_k^{(j-1)}, \boldsymbol{u}_{k-1})$
  end for
  $\boldsymbol{z}_k = \boldsymbol{z}_k^{(J)}$
  $\underline{\boldsymbol{u} \text{ ESTIMATION}}$:
  $\boldsymbol{u}_k = \mathcal{T}(\boldsymbol{z}_k)$ (or $\widetilde{\mathcal{T}}(\boldsymbol{u}_{k-1}, \boldsymbol{z}_k)$ for NAGD)
end for

Output: $\boldsymbol{c}^* = \boldsymbol{z}_K \odot \boldsymbol{u}_K$

For succinctness in representing the estimation of $\boldsymbol{z}$, we define the complete scale variable mapping $\mathcal{Z} : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n$ as

$$\mathcal{Z}(\boldsymbol{z}, \boldsymbol{u}) = \underbrace{g \circ \cdots g \circ g(\boldsymbol{z}, \boldsymbol{u})}_{J \text{ times}} \approx \underset{\boldsymbol{z} \in \mathfrak{Z}}{\arg\min}\ F(\boldsymbol{u}, \boldsymbol{z}) \tag{3.6}$$

where the J compositions occur over the first component of g. Then on iteration k of G-CG-LS we have the estimates

$$\boldsymbol{z}_k = \mathcal{Z}(\boldsymbol{z}_{k-1}, \boldsymbol{u}_{k-1}) \quad \text{and} \quad \boldsymbol{u}_k = \mathcal{T}(\boldsymbol{z}_k; P_u).$$

Now, Algorithm 3 is initialized as $\boldsymbol{z}_0 = \mathcal{P}_{[0,b]^n}(A^T\boldsymbol{y})$ and $\boldsymbol{u}_0 = \mathcal{T}(\boldsymbol{z}_0)$. Using the rectified linear unit (ReLU) activation function, we remark that $\mathcal{P}_{[0,b]^n}(\boldsymbol{x}) = [\text{ReLU}(x_i) - \text{ReLU}(x_i - b)]_{i=1}^n$ and is applied to eliminate negative values in $\boldsymbol{z}_0$, since $\boldsymbol{z}$ should be a positive vector, and limit the maximum value in $\boldsymbol{z}_0$ for numerical stability in calculating $\boldsymbol{u}_0$.

Next, we remark that (3.2) is strongly convex w.r.t. $\boldsymbol{u}$ and thus the Tikhonov solution is efficiently approximated by Nesterov accelerated gradient descent (NAGD) [78]. Hence, for sufficiently high-dimensional signals of interest, such that the linear solve in (3.5) is infeasible, we

replace the exact calculation of (3.5) by a NAGD approximation. To detail the NAGD approximation, define for $\eta > 0$

$$r_u(\boldsymbol{u}, \boldsymbol{z}; \eta) \equiv r_u(\boldsymbol{u}, \boldsymbol{z}) \coloneqq \boldsymbol{u} - \eta \nabla_u F(\boldsymbol{u}, \boldsymbol{z}) \\ = \boldsymbol{u} - \eta \left(A_{\boldsymbol{z}}^T (A_{\boldsymbol{z}} \boldsymbol{u} - \boldsymbol{y}) + P_u^{-1} \boldsymbol{u} \right). \tag{3.7}$$

Note that r_u is a gradient descent step on (3.2) w.r.t. $\boldsymbol{u}$. The NAGD estimate of $\boldsymbol{u}$ on descent step $j+1$ of iteration k is then given by

$$\boldsymbol{u}_k^{(j+1)} = r_u(\boldsymbol{u}_k^{(j)}, \boldsymbol{z}_k) + \beta(r_u(\boldsymbol{u}_k^{(j)}, \boldsymbol{z}_k) - r_u(\boldsymbol{u}_k^{(j-1)}, \boldsymbol{z}_k))$$

where $\beta \geq 0$ is a momentum control parameter and $\boldsymbol{u}_k^{(0)} = \boldsymbol{u}_k^{(-1)} = \boldsymbol{u}_{k-1}$. Let J_u denote the number of NAGD steps, then the estimate of $\boldsymbol{u}$ on iteration k is given as $\boldsymbol{u}_k = u_k^{(J_u)}$. Note that we denote this composition of all J_u NAGD steps as $\widetilde{\mathcal{T}}(\boldsymbol{u}, \boldsymbol{z}) \equiv \widetilde{\mathcal{T}}(\boldsymbol{u}, \boldsymbol{z}; P_u)$ to emphasize that the NAGD estimate approximates the Tikhonov solution, $\mathcal{T}$. That is, $\boldsymbol{u}_k = \boldsymbol{u}_{k-1}^{(J_u)} = \widetilde{\mathcal{T}}(\boldsymbol{u}_{k-1}, \boldsymbol{z}_k)$.

Lastly, as our G-CG-LS method generalizes on previous CG-based iterative algorithms [57,58, 76] we briefly explain key differences between these methods. Previous work [57,58,76] similarly decomposes $\boldsymbol{c}$ via (1.7), but restricts $\boldsymbol{z} = h(\boldsymbol{x})$ where $\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{0}, I)$ and h is an invertible nonlinear function. For this choice of $\boldsymbol{z}$, $\mathcal{R}$ in (3.2) is set at $\mathcal{R}(\boldsymbol{z}) = \mu \|h^{-1}(\boldsymbol{z})\|_2^2$, for scalar $\mu > 0$, to enforce normality of $\boldsymbol{x}$. Furthermore, previous work [57,58,76] restricts P_u in (3.2) to be a scaled identity matrix. Our G-CG-LS algorithm naturally generalizes these prior CG-based iterative algorithm cost functions to a cost function with implicit scale regularization, $\mathcal{R}$, and general covariance matrix structure. By applying algorithm unrolling to G-CG-LS, as shown in Section 3.3, $\mathcal{R}$ and P_u are optimally learned to produce a DNN with greater representational capacity and applicability than the previous unrolled CG-based DNNs proposed in [57, 58, 76].

3.2.2 Scale Variable Update Methods

Two standard methods we employ in optimizing (3.2) w.r.t. $\boldsymbol{z}$ are detailed below. Note that η is a positive real value and denotes a step size for each scale variable update method. Additionally, we define $A_{\boldsymbol{u}} = A\text{Diag}(\boldsymbol{u})$.

Projected Gradient Descent (PGD). When $\mathcal{R}$ is a differentiable function we apply a PGD step to update $\boldsymbol{z}$. That is, we perform a gradient step on $F(\boldsymbol{u}, \boldsymbol{z})$ w.r.t. $\boldsymbol{z}$ and then project onto the convex domain $\mathfrak{Z}$. Mathematically, the PGD step is given by

$$\begin{aligned} g(\boldsymbol{z}, \boldsymbol{u}) &\coloneqq \mathcal{P}_{\mathfrak{Z}}\left(\boldsymbol{z} - \eta \nabla_{\boldsymbol{z}} F(\boldsymbol{u}, \boldsymbol{z})\right) \\ &= \mathcal{P}_{\mathfrak{Z}}\left(\boldsymbol{z} - \eta\left(A_{\boldsymbol{u}}^T(A_{\boldsymbol{u}}\boldsymbol{z} - \boldsymbol{y}) + \nabla\mathcal{R}(\boldsymbol{z})\right)\right) \end{aligned}$$

where $\mathcal{P}_{\mathfrak{Z}}$ is the projection onto $\mathfrak{Z}$.

Iterative Shrinkage and Thresholding (ISTA). When $\mathcal{R}$ is a convex function and possibly non-smooth, i.e. possibly not differentiable, then we apply an ISTA step to update $\boldsymbol{z}$. That is, we perform a gradient step on $\frac{1}{2}\|\boldsymbol{y} - A(\boldsymbol{z} \odot \boldsymbol{u})\|_2^2$, which is the data fidelity term of $F(\boldsymbol{u}, \boldsymbol{z})$, w.r.t. $\boldsymbol{z}$ and then apply a proximal operator of $\mathcal{R}$ to incorporate optimization of the regularization. Mathematically, the ISTA step is given by

$$\begin{aligned} g(\boldsymbol{z}, \boldsymbol{u}) &\coloneqq \text{prox}_{\eta\mathcal{R}}(\boldsymbol{z} - \eta A_{\boldsymbol{u}}^T(A_{\boldsymbol{u}}\boldsymbol{z} - \boldsymbol{y})) \\ &= \arg\min_{\boldsymbol{\zeta}\in\mathfrak{Z}} \frac{1}{2}\|\boldsymbol{\zeta} - (\boldsymbol{z} - \eta A_{\boldsymbol{u}}^T(A_{\boldsymbol{u}}\boldsymbol{z} - \boldsymbol{y}))\|_2^2 + \eta\mathcal{R}(\boldsymbol{\zeta}) \end{aligned}$$

where prox_f is the proximal operator of f and is well-defined for convex f. We remark that for f a non-smooth, convex function the proximal operator is an optimization tool as fixed points of prox_f minimize f. The general ISTA method [2] above, which is equivalent to proximal gradient descent, is an optimization method for the sum of a convex, differentiable function and a convex,

non-smooth function. From the use of the proximal operator on the non-smooth piece, fixed points of ISTA are optimality points of the original sum of functions (see Section 1.6).

The PGD and ISTA methods were chosen for their simplicity to implement, provable convergence properties, and the previous success of unrolling ISTA for linear inverse problems [20, 22–24]. Additional details about PGD and ISTA, including backtracking linesearch methods to determine η and descent bounds, are given in Sections 1.5 and 1.6, respectively.

3.2.3 Convergence of G-CG-LS

We first remark that, with $\boldsymbol{z}$ fixed, equation (3.2) is a Tikhonov regularization problem over $\boldsymbol{u}$ and thus is strongly convex in $\boldsymbol{u}$ with strength controlled by the spectral radius of P_u [34]. Hence, equation (3.2) has no local maxima. Now, we derive convergence of G-CG-LS under various combinations of the following regularization assumptions:

($\mathcal{A}1$) $\mathcal{R}$ is bounded below and satisfies $\lim_{z_i \to \infty} \mathcal{R}(\boldsymbol{z}) \to \infty$ for all $i = 1, 2, \ldots, n$.

($\mathcal{A}2$) $\mathcal{R}$ is convex and possibly non-smooth.

($\mathcal{A}3$) $\mathcal{R}$ is twice continuously differentiable.

First, a new proposition on the convergence of G-CG-LS cost function values.

Proposition 3.2.1. *Let $\mathcal{R}$ satisfy ($\mathcal{A}1$). If $\mathcal{R}$ satisfies ($\mathcal{A}2$) for ISTA G-CG-LS or ($\mathcal{A}3$) for PGD G-CG-LS, then the sequence of cost function values $\{F(\boldsymbol{u}_k, \boldsymbol{z}_k)\}_{k=1}^{\infty}$ converges.*

To prove Proposition 3.2.1 we use Lemma 1.5.4 and Lemma 1.6.5 providing lower bounds on the change in cost function values for a PGD and ISTA step, respectively. Additionally, Proposition 2.2.8 is employed to guarantee that the convex hull of the sublevel set for the G-CG-LS cost function is compact.

Proof of Proposition 3.2.1. For initial estimates $\boldsymbol{u}_0$ and $\boldsymbol{z}_0$, define the sublevel set

$$S(\boldsymbol{u}_0, \boldsymbol{z}_0) = \{(\boldsymbol{u}, \boldsymbol{z}) \in \mathbb{R}^n \times \mathfrak{Z} : F(\boldsymbol{u}, \boldsymbol{z}) \leq F(\boldsymbol{u}_0, \boldsymbol{z}_0)\}$$

and note that $S(\boldsymbol{u}_0, \boldsymbol{z}_0) \subseteq \mathbb{R}^n \times \mathfrak{Z}$. When either $(\mathcal{A}2)$ or $(\mathcal{A}3)$ hold then F is continuous and thus $S(\boldsymbol{u}_0, \boldsymbol{z}_0)$ is closed. Furthermore, as $(\mathcal{A}1)$ holds then F is coercive in both $\boldsymbol{u}$ and $\boldsymbol{z}$ and thus $S(\boldsymbol{u}_0, \boldsymbol{z}_0)$ is bounded. Hence, $S(\boldsymbol{u}_0, \boldsymbol{z}_0)$ is compact by the Heine-Borel theorem [53]. Lastly, by Proposition 2.2.8 the convex hull of $S(\boldsymbol{u}_0, \boldsymbol{z}_0)$, denoted $\text{Conv}(S(\boldsymbol{u}_0, \boldsymbol{z}_0))$, is compact. Additionally, as $\mathbb{R}^n \times \mathfrak{Z}$ is convex and $\text{Conv}(S(\boldsymbol{u}_0, \boldsymbol{z}_0))$ is the smallest convex set containing $S(\boldsymbol{u}_0, \boldsymbol{z}_0)$ then $\text{Conv}(S(\boldsymbol{u}_0, \boldsymbol{z}_0)) \subseteq \mathbb{R}^n \times \mathfrak{Z}$.

Now, let $f(\boldsymbol{u}, \boldsymbol{z}) = \frac{1}{2}\|\boldsymbol{y} - A(\boldsymbol{z} \odot \boldsymbol{u})\|_2^2$. Note f is twice continuously differentiable in $\boldsymbol{z}$. When $(\mathcal{A}3)$ holds then $F(\boldsymbol{u}, \boldsymbol{z})$ is also twice continuously differentiable in $\boldsymbol{z}$. Hence, $H_{f;\boldsymbol{z}}(\boldsymbol{u}, \boldsymbol{z})$ and $H_{F;\boldsymbol{z}}(\boldsymbol{u}, \boldsymbol{z})$, the $\boldsymbol{z}$ Hessians of f and F respectively, are continuous. Thus, $\|H_{f;\boldsymbol{z}}(\boldsymbol{u}, \boldsymbol{z})\|_2$ and $\|H_{F;\boldsymbol{z}}(\boldsymbol{u}, \boldsymbol{z})\|_2$ are also continuous. As $\text{Conv}(S(\boldsymbol{u}_0, \boldsymbol{z}_0)) \subseteq \mathbb{R}^n \times \mathfrak{Z}$ then $\|H_{f;\boldsymbol{z}}(\boldsymbol{u}, \boldsymbol{z})\|_2$ and $\|H_{F;\boldsymbol{z}}(\boldsymbol{u}, \boldsymbol{z})\|_2$ are defined on $\text{Conv}(S(\boldsymbol{u}_0, \boldsymbol{z}_0))$. Since $\|H_{f;\boldsymbol{z}}(\boldsymbol{u}, \boldsymbol{z})\|_2$ and $\|H_{F;\boldsymbol{z}}(\boldsymbol{u}, \boldsymbol{z})\|_2$ are continuous functions on the compact set $\text{Conv}(S(\boldsymbol{u}_0, \boldsymbol{z}_0))$ then, by the Extreme Value Theorem [53], each obtains a bounded maximum on $\text{Conv}(S(\boldsymbol{u}_0, \boldsymbol{z}_0))$. Therefore, by the Mean Value Theorem [53], f and F (when $(\mathcal{A}3)$ holds) have Lipschitz continuous gradient on $\text{Conv}(S(\boldsymbol{u}_0, \boldsymbol{z}_0))$. Similarly, since $S(\boldsymbol{u}_0, \boldsymbol{z}_0) \subseteq \text{Conv}(S(\boldsymbol{u}_0, \boldsymbol{z}_0))$ then f and F (when $(\mathcal{A}3)$ holds) have Lipschitz continuous gradient on $S(\boldsymbol{u}_0, \boldsymbol{z}_0)$.

Hence, for PGD G-CG-LS where $(\mathcal{A}3)$ holds, Lemma 1.5.4 holds. Similarly, for ISTA G-CG-LS where $(\mathcal{A}2)$ holds, Lemma 1.6.5 (taking $r = \mathcal{R}$) holds. Consequently, for all $k \in \mathbb{N}$, any $j = 1, 2, \ldots, J$, it holds for either PGD or ISTA G-CG-LS that

$$F(\boldsymbol{u}_{k-1}, \boldsymbol{z}_k^{(j-1)}) - F(\boldsymbol{u}_{k-1}, \boldsymbol{z}_k^{(j)}) \geq c\|\boldsymbol{z}_k^{(j)} - \boldsymbol{z}_k^{(j-1)}\|_2^2 \tag{3.8}$$

for some $c > 0$. Then

$$F(\boldsymbol{u}_{k-1}, \boldsymbol{z}_{k-1}) = F(\boldsymbol{u}_{k-1}, \boldsymbol{z}_k^0) \geq F(\boldsymbol{u}_{k-1}, \boldsymbol{z}_k^1) \geq \cdots \geq F(\boldsymbol{u}_{k-1}, \boldsymbol{z}_k^J) = F(\boldsymbol{u}_{k-1}, \boldsymbol{z}_k) \geq F(\boldsymbol{u}_k, \boldsymbol{z}_k)$$

holds for all $k \in \mathbb{N}$. Hence, $\{F(\boldsymbol{u}_k, \boldsymbol{z}_k)\}_{k=1}^{\infty}$ is a monotonic decreasing sequence. As f is bounded below by 0, $\boldsymbol{u}^T P_u^{-1} \boldsymbol{u}$ is bounded below by 0, and $\mathcal{R}(\boldsymbol{z})$ is bounded below by $(\mathcal{A}1)$ then $F(\boldsymbol{u}_k, \boldsymbol{z}_k)$

is bounded below for all k. Therefore, $\{F(\boldsymbol{u}_k, \boldsymbol{z}_k)\}_{k=1}^{\infty}$ is a monotonic decreasing sequence that is bounded below and thus converges. $\boxtimes$

Using Proposition 3.2.1 we show, in the following new theorem, that G-CG-LS converges to stationary points satisfying the optimality conditions of Definition 1.5.1 and Definition 1.6.6.

Theorem 3.2.2. *Let $\mathcal{R}$ satisfy $(\mathcal{A}1)$. If $\mathcal{R}$ satisfies $(\mathcal{A}2)$ for ISTA G-CG-LS or $(\mathcal{A}3)$ for PGD G-CG-LS, then every limit point of the sequence $\{(\boldsymbol{u}_k, \boldsymbol{z}_k)\}_{k=1}^{\infty}$ is a stationary point of the G-CG-LS cost function (3.2).*

The proof of Theorem 3.2.2 is inspired by ideas from [2, 51, 79], which consider the convergence of PGD and ISTA-type updates. Where [2, 51] consider a single block update, which can transfer to individual updates of $\boldsymbol{z}$ when $\boldsymbol{u}$ is fixed, we distinctly have two block updates defining each iteration. While [79] considers multiblock updates with each block being updated by a single step of an identical method, we instead consider multiple steps and unique methods defining each block update. To prove Theorem 3.2.2 we use Proposition 3.2.1, Lemma 1.5.5, and Lemma 1.6.7.

Proof of Theorem 3.2.2. Summing (3.8), from the proof of Proposition 3.2.1, over j, and using $\boldsymbol{z}_k^{(0)} = \boldsymbol{z}_{k-1}$ and $\boldsymbol{z}_k^{(J)} = \boldsymbol{z}_k$ shows

$$\sum_{j=1}^{J} \left(F(\boldsymbol{u}_{k-1}, \boldsymbol{z}_k^{(j-1)}) - F(\boldsymbol{u}_{k-1}, \boldsymbol{z}_k^{(j)}) \right) \geq \sum_{j=1}^{J} c \|\boldsymbol{z}_k^{(j)} - \boldsymbol{z}_k^{(j-1)}\|_2^2$$

$$F(\boldsymbol{u}_{k-1}, \boldsymbol{z}_{k-1}) - F(\boldsymbol{u}_{k-1}, \boldsymbol{z}_k) \geq c \sum_{j=1}^{J} \|\boldsymbol{z}_k^{(j)} - \boldsymbol{z}_k^{(j-1)}\|_2^2$$

where the left hand side results from the cancellation of a telescoping series. As $F(\boldsymbol{u}_{k-1}, \boldsymbol{z}_k) \geq F(\boldsymbol{u}_k, \boldsymbol{z}_k)$ we have $-F(\boldsymbol{u}_{k-1}, \boldsymbol{z}_k) \leq -F(\boldsymbol{u}_k, \boldsymbol{z}_k)$. Therefore

$$F(\boldsymbol{u}_{k-1}, \boldsymbol{z}_{k-1}) - F(\boldsymbol{u}_k, \boldsymbol{z}_k) \geq F(\boldsymbol{u}_{k-1}, \boldsymbol{z}_{k-1}) - F(\boldsymbol{u}_{k-1}, \boldsymbol{z}_k) \geq c \sum_{j=1}^{J} \|\boldsymbol{z}_k^{(j)} - \boldsymbol{z}_k^{(j-1)}\|_2^2. \quad (3.9)$$

By Proposition 3.2.1, $\{F(\boldsymbol{u}_\ell, \boldsymbol{z}_\ell)\}_{\ell=1}^{\infty}$ converges and we let F^* be the limit point. Hence, summing (3.9) over $k \in \mathbb{N}$ and using that the left hand side of (3.9) is a telescoping sum we have

$$F(\boldsymbol{u}_0, \boldsymbol{z}_0) - F^* = \sum_{k=1}^{\infty} F(\boldsymbol{u}_{k-1}, \boldsymbol{z}_{k-1}) - F(\boldsymbol{u}_k, \boldsymbol{z}_k) \geq c \sum_{k=1}^{\infty} \sum_{j=1}^{J} \|\boldsymbol{z}_k^{(j)} - \boldsymbol{z}_k^{(j-1)}\|_2^2. \tag{3.10}$$

Therefore, the series $\sum_{k=1}^{\infty} a_k^{(J)}$ for $a_k^{(J)} = \sum_{j=1}^{J} \|\boldsymbol{z}_k^{(j)} - \boldsymbol{z}_k^{(j-1)}\|_2^2$ converges. Hence, $\lim_{k\to\infty} a_k^{(J)} \to 0$. Since every $a_k^{(J)}$ is the sum of J non-negative terms then $\lim_{k\to\infty} a_k^{(J)} \to 0$ implies that each term in the summation must converge to zero. Thus, it holds that $\lim_{k\to\infty} \|\boldsymbol{z}_k^{(j)} - \boldsymbol{z}_k^{(j-1)}\|_2^2 \to 0$ for every $j = 1, 2, \ldots, J$.

We make two remarks. First, note that for any real-valued sequence $\{x_k\}_{k=1}^{\infty}$ the Cauchy-Schwarz inequality implies

$$\sum_{j=1}^{J} x_j^2 \geq \frac{1}{J} \left(\sum_{j=1}^{J} x_j \right)^2.$$

Second, using a telescoping sum and the triangle inequality note that

$$\|\boldsymbol{z}_k - \boldsymbol{z}_{k-1}\|_2 = \left\| \boldsymbol{z}_k^{(J)} - \boldsymbol{z}_k^{(0)} \right\|_2 = \left\| \sum_{j=1}^{J} (\boldsymbol{z}_k^{(j)} - \boldsymbol{z}_k^{(j-1)}) \right\|_2 \leq \sum_{j=1}^{J} \|\boldsymbol{z}_k^{(j)} - \boldsymbol{z}_k^{(j-1)}\|_2.$$

Combining these two notes we have

$$\sum_{j=1}^{J} \|\boldsymbol{z}_k^{(j)} - \boldsymbol{z}_k^{(j-1)}\|_2^2 \geq \frac{1}{J} \left(\sum_{j=1}^{J} \|\boldsymbol{z}_k^{(j)} - \boldsymbol{z}_k^{(j-1)}\|_2 \right)^2 \geq \frac{1}{J} \|\boldsymbol{z}_k - \boldsymbol{z}_{k-1}\|_2^2. \tag{3.11}$$

Further combining (3.10) and (3.11) produces

$$F(\boldsymbol{u}_0, \boldsymbol{z}_0) - F^* \geq \frac{c}{J} \sum_{k=1}^{\infty} \|\boldsymbol{z}_k - \boldsymbol{z}_{k-1}\|_2^2.$$

Hence, it also holds that the series $\sum_{k=1}^{\infty}\|\boldsymbol{z}_k-\boldsymbol{z}_{k-1}\|_2^2$ converges and thus $\lim_{k\to\infty}\|\boldsymbol{z}_k-\boldsymbol{z}_{k-1}\|_2^2\to 0$. By continuity of the Tikhonov solution, $\mathcal{T}(\boldsymbol{z})$, it similarly holds that $\lim_{k\to\infty}\|\boldsymbol{u}_k-\boldsymbol{u}_{k-1}\|_2^2=\lim_{k\to\infty}\|\mathcal{T}(\boldsymbol{z}_k)-\mathcal{T}(\boldsymbol{z}_{k-1})\|_2^2\to 0$.

Next, let $(\boldsymbol{u}^*,\boldsymbol{z}^*)$ be any limit point of the sequence $\{(\boldsymbol{u}_k,\boldsymbol{z}_k)\}_{k=1}^{\infty}$ and $\{(\boldsymbol{u}_{k_i},\boldsymbol{z}_{k_i})\}_{i=1}^{\infty}\subseteq\{(\boldsymbol{u}_k,\boldsymbol{z}_k)\}_{k=1}^{\infty}$ be a subsequence converging to $(\boldsymbol{u}^*,\boldsymbol{z}^*)$. By Lemma 1.5.4 and Lemma 1.6.5 the sequence of step sizes $\left\{\eta_{k_i}^{(1)}\right\}_{i=1}^{\infty}$ is bounded and thus there exists a convergent subsequence $\left\{\eta_{k_{i_\ell}}^{(1)}\right\}_{\ell=1}^{\infty}\subseteq\left\{\eta_{k_i}^{(1)}\right\}_{i=1}^{\infty}$. Let η^* be the limit point of $\left\{\eta_{k_{i_\ell}}^{(1)}\right\}_{\ell=1}^{\infty}$. As $\{(\boldsymbol{u}_{k_i},\boldsymbol{z}_{k_i})\}_{i=1}^{\infty}$ converges every subsequence converges to the same limit point implying $\{(\boldsymbol{u}_{k_{i_\ell}},\boldsymbol{z}_{k_{i_\ell}})\}_{\ell=1}^{\infty}$ converges to $(\boldsymbol{u}^*,\boldsymbol{z}^*)$. As $\lim_{k\to\infty}\|\boldsymbol{z}_k-\boldsymbol{z}_{k-1}\|_2^2\to 0$ and $\lim_{k\to\infty}\|\boldsymbol{u}_k-\boldsymbol{u}_{k-1}\|_2^2\to 0$ then $\{(\boldsymbol{u}_{k_{i_\ell}-1},\boldsymbol{z}_{k_{i_\ell}-1})\}_{\ell=1}^{\infty}$ also converges to $(\boldsymbol{u}^*,\boldsymbol{z}^*)$. Similarly, as $\lim_{k\to\infty}\|\boldsymbol{z}_k^{(j)}-\boldsymbol{z}_k^{(j-1)}\|_2^2\to 0$ for every $j=1,2,\ldots,J$ we have

$$\lim_{\ell\to\infty}\left\|\boldsymbol{z}_{k_{i_\ell}}^{(1)}-\boldsymbol{z}_{k_{i_\ell}}^{(0)}\right\|_2^2=\lim_{\ell\to\infty}\left\|\boldsymbol{z}_{k_{i_\ell}}^{(1)}-\boldsymbol{z}_{k_{i_\ell}-1}\right\|_2^2\to 0,$$

which implies $\left\{\boldsymbol{z}_{k_{i_\ell}}^{(0)}\right\}_{\ell=1}^{\infty}$ and $\left\{\boldsymbol{z}_{k_{i_\ell}}^{(1)}\right\}_{\ell=1}^{\infty}$ converge to $\boldsymbol{z}^*$. In PGD G-CG-LS, by continuity of the projection $\mathcal{P}_{\mathfrak{Z}}$ [52] and continuity of the gradient $\nabla_{\boldsymbol{z}}F$, observe

$$\begin{aligned}\boldsymbol{z}^*&=\lim_{\ell\to\infty}\boldsymbol{z}_{k_{i_\ell}}^{(1)}\\&=\lim_{\ell\to\infty}\mathcal{P}_{\mathfrak{Z}}\left(\boldsymbol{z}_{k_{i_\ell}}^{(0)}-\eta_{k_{i_\ell}}^{(1)}\nabla_{\boldsymbol{z}}F(\boldsymbol{u}_{k_{i_\ell}-1},\boldsymbol{z}_{k_{i_\ell}}^{(0)})\right)\\&=\mathcal{P}_{\mathfrak{Z}}(\boldsymbol{z}^*-\eta^*\nabla_{\boldsymbol{z}}F(\boldsymbol{u}^*,\boldsymbol{z}^*)).\end{aligned}$$

Therefore, by Lemma 1.5.5 it holds that

$$\langle\nabla_{\boldsymbol{z}}F(\boldsymbol{u}^*,\boldsymbol{z}^*),\boldsymbol{z}-\boldsymbol{z}^*\rangle\geq 0\qquad\text{for all }\boldsymbol{z}\in\mathfrak{Z}.\tag{3.12}$$

In ISTA G-CG-LS, by continuity of the proximal operator [80]

$$\begin{aligned}\boldsymbol{z}^* &= \lim_{\ell\to\infty} \boldsymbol{z}_{k_{i_\ell}}^{(1)} \\ &= \lim_{\ell\to\infty} \text{prox}_{\eta_{k_{i_\ell}}^{(1)}\mathcal{R}}\left(\boldsymbol{z}_{k_{i_\ell}}^{(0)} - \eta_{k_{i_\ell}}^{(1)} A_{\boldsymbol{u}_{k_{i_\ell}-1}}^T (A_{\boldsymbol{u}_{k_{i_\ell}-1}}\boldsymbol{z}_{k_{i_\ell}}^{(0)} - \boldsymbol{y})\right) \\ &= \text{prox}_{\eta^*\mathcal{R}}\left(\boldsymbol{z}^* - \eta^* A_{\boldsymbol{u}^*}^T (A_{\boldsymbol{u}^*}\boldsymbol{z}^* - \boldsymbol{y})\right).\end{aligned}$$

Therefore, by Lemma 1.6.7, since $\partial_{\boldsymbol{z}} F(\boldsymbol{u},\boldsymbol{z}) = \{\nabla_{\boldsymbol{z}} f(\boldsymbol{u},\boldsymbol{z}) + \boldsymbol{d} : \boldsymbol{d} \in \partial\mathcal{R}(\boldsymbol{z})\}$ there exists a $\boldsymbol{d}^* \in \partial\mathcal{R}(\boldsymbol{z}^*)$, where $\partial\mathcal{R}(\boldsymbol{z}^*)$ is the subdifferential of $\mathcal{R}$ a $\boldsymbol{z}^*$, such that

$$\langle \nabla_{\boldsymbol{z}} f(\boldsymbol{u}^*,\boldsymbol{z}^*) + \boldsymbol{d}^*, \boldsymbol{z} - \boldsymbol{z}^*\rangle \geq 0 \text{ for all } \boldsymbol{z} \in \mathfrak{Z}. \tag{3.13}$$

As $\boldsymbol{u}^*$ is the global minimizer of $F(\boldsymbol{u},\boldsymbol{z}^*)$ and $\nabla_{\boldsymbol{u}} F(\boldsymbol{u},\boldsymbol{z}) = \nabla_{\boldsymbol{u}} f(\boldsymbol{u},\boldsymbol{z}) + P_u^{-1}\boldsymbol{u}$ then

$$\langle \nabla_{\boldsymbol{u}} F(\boldsymbol{u}^*,\boldsymbol{z}^*), \boldsymbol{u} - \boldsymbol{u}^*\rangle = \langle \nabla_{\boldsymbol{u}} f(\boldsymbol{u}^*,\boldsymbol{z}^*) + P_u^{-1}\boldsymbol{u}^*, \boldsymbol{u} - \boldsymbol{u}^*\rangle \geq 0 \tag{3.14}$$

holds for all $\boldsymbol{u} \in \mathbb{R}^n$. For PGD G-CG-LS, adding together (3.12) and (3.14)

$$\left\langle \nabla F(\boldsymbol{u}^*,\boldsymbol{z}^*), \begin{bmatrix}\boldsymbol{u} & \boldsymbol{z}\end{bmatrix}^T - \begin{bmatrix}\boldsymbol{u}^* & \boldsymbol{z}^*\end{bmatrix}^T \right\rangle \geq 0$$

holds for all $(\boldsymbol{u},\boldsymbol{z}) \in \mathbb{R}^n \times \mathfrak{Z}$. Therefore $(\boldsymbol{u}^*,\boldsymbol{z}^*)$ is a stationary point of $F(\boldsymbol{u},\boldsymbol{z})$. Similarly, for ISTA G-CG-LS, adding together (3.13) and (3.14)

$$\left\langle \nabla f(\boldsymbol{u}^*,\boldsymbol{z}^*) + \begin{bmatrix}P_{\boldsymbol{u}}^{-1}\boldsymbol{u}^* & \boldsymbol{d}^*\end{bmatrix}^T, \begin{bmatrix}\boldsymbol{u} & \boldsymbol{z}\end{bmatrix}^T - \begin{bmatrix}\boldsymbol{u}^* & \boldsymbol{z}^*\end{bmatrix}^T \right\rangle \geq 0$$

holds for all $(\boldsymbol{u},\boldsymbol{z}) \in \mathbb{R}^n \times \mathfrak{Z}$ and thus, $(\boldsymbol{u}^*,\boldsymbol{z}^*)$ is a stationary point of $F(\boldsymbol{u},\boldsymbol{z})$. ⊠

Next, we discuss the convergence of the G-CG-LS sequence of estimates. For this, we say a sequence $\{\boldsymbol{x}_k\}_{k=1}^{\infty} \subseteq \mathbb{R}^n$ converges to a set $\mathcal{S}$ if the sequence values become arbitrarily close to elements of $\mathcal{S}$.

Definition 3.2.3. *A sequence $\{\boldsymbol{x}_k\}_{k=1}^{\infty} \subseteq \mathbb{R}^n$ converges to set $\mathcal{S}$ if for every $\epsilon > 0$ there exists a $k_0 \in \mathbb{N}$ where for all $k \geq k_0$ it holds that $\boldsymbol{x}_k \in \bigcup_{\boldsymbol{x} \in \mathcal{S}} \{\boldsymbol{z} \in \mathbb{R}^n : \|\boldsymbol{x} - \boldsymbol{z}\|_2 \leq \epsilon\}$.*

For example, the sequence $\{(-1)^k\}_{k=1}^{\infty}$ converges to the set $\mathcal{S} = \{-1, 1\}$. We now show that the G-CG-LS estimates converge to a connected set of stationary points that have constant cost function value in the following new proposition.

Proposition 3.2.4. *Let $\mathcal{R}$ satisfy $(\mathcal{A}1)$ and satisfy $(\mathcal{A}2)$ for ISTA G-CG-LS or $(\mathcal{A}3)$ for PGD G-CG-LS. Then the sequence $\{(\boldsymbol{u}_k, \boldsymbol{z}_k)\}_{k=1}^{\infty}$ generated by ISTA G-CG-LS or PGD G-CG-LS converges to a closed and connected set of stationary points of constant cost function value.*

To prove Proposition 3.2.4 we use Proposition 3.2.1, Theorem 3.2.2, the following two lemmas from the literature [81].

Lemma 3.2.5 ([81], Proposition 12.4.1)**.** *Let $\{\boldsymbol{x}_k\}_{k=1}^{\infty} \subseteq \mathbb{R}^n$ be a bounded sequence satisfying $\lim_{k\to\infty} \|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\|_2 \to 0$. Then the limits points of $\{\boldsymbol{x}_k\}_{k=1}^{\infty}$ form a connected set.*

Lemma 3.2.6 ([81])**.** *A bounded sequence $\{\boldsymbol{x}_k\}_{k=1}^{\infty} \subseteq \mathbb{R}^n$ converges to its set of limit points.*

Proof. Let $\mathcal{L}$ be the limit points of $\{\boldsymbol{x}_k\}_{k=1}^{\infty}$. Assume for contradiction $\{\boldsymbol{x}_k\}_{k=1}^{\infty}$ does not converge to $\mathcal{L}$. Then there is a $\epsilon > 0$ and subsequence $\{\boldsymbol{x}_{k_i}\}_{i=1}^{\infty}$ such that $\boldsymbol{x}_{k_i} \notin \cup_{\boldsymbol{x} \in \mathcal{L}} \{\widehat{\boldsymbol{x}} : \|\boldsymbol{x} - \widehat{\boldsymbol{x}}\|_2 \leq \epsilon\}$ for every $i \in \mathbb{N}$. As $\{\boldsymbol{x}_{k_i}\}_{i=1}^{\infty}$ is bounded then by the Bolzano-Weierstrass Theorem [53] there exists a convergence subsequence $\{\boldsymbol{x}_{k_{i_\ell}}\}_{\ell=1}^{\infty}$. Let $\boldsymbol{x}^* \in \mathcal{L}$ be the limit point and note that there exists a $\ell_0 \in \mathbb{N}$ such that for all $\ell \geq \ell_0$ it holds that $\|\boldsymbol{x}^* - \boldsymbol{x}_{k_{i_\ell}}\|_2 \leq \epsilon$. Thus, $\boldsymbol{x}_{k_{i_\ell}} \in \{\widehat{\boldsymbol{x}} : \|\boldsymbol{x}^* - \widehat{\boldsymbol{x}}\|_2 \leq \epsilon\} \subseteq \cup_{\boldsymbol{x} \in \mathcal{L}} \{\widehat{\boldsymbol{x}} : \|\boldsymbol{x} - \widehat{\boldsymbol{x}}\|_2 \leq \epsilon\}$ for all $\ell \geq \ell_0$ producing a contradiction. Therefore, $\{\boldsymbol{x}_k\}_{k=1}^{\infty}$ converges to $\mathcal{L}$. ⊠

Now, we prove Proposition 3.2.4.

Proof of Proposition 3.2.4. Let $\mathcal{S}$ be the set of stationary points of the G-CG-LS cost function in equation (3.2). Let $\{(\boldsymbol{u}_k, \boldsymbol{z}_k)\}_{k=0}^{\infty}$ be the G-CG-LS sequence of estimates with the set of limit points $\mathcal{L}$. By [53], $\mathcal{L}$ is a closed set. By Theorem 3.2.2, $\mathcal{L} \subseteq \mathcal{S}$ and $\{(\boldsymbol{u}_k, \boldsymbol{z}_k)\}_{k=0}^{\infty}$ satisfies

$\lim_{k\to\infty}\|(\boldsymbol{u}_{k+1},\boldsymbol{z}_{k+1})-(\boldsymbol{u}_k,\boldsymbol{z}_k)\|_2 \to 0$. As $\{(\boldsymbol{u}_k,\boldsymbol{z}_k)\}_{k=0}^{\infty} \subseteq \mathcal{S}(\boldsymbol{u}_0,\boldsymbol{z}_0)$ and the sublevel set $\mathcal{S}(\boldsymbol{u}_0,\boldsymbol{z}_0)$ is compact by Proposition 3.2.1 then $\{(\boldsymbol{u}_k,\boldsymbol{z}_k)\}_{k=0}^{\infty}$ is bounded. Hence, $\mathcal{L}$ is a connected set by Lemma 3.2.5. Additionally, by Proposition 3.2.1, $\{F(\boldsymbol{u}_k,\boldsymbol{z}_k)\}_{k=1}^{\infty}$ converges to some value F^*. Thus, for every subsequence $\{(\boldsymbol{u}_{k_i},\boldsymbol{z}_{k_i})\}_{i=1}^{\infty} \subseteq \{(\boldsymbol{u}_k,\boldsymbol{z}_k)\}_{k=0}^{\infty}$ it holds that the sequence $\{F(\boldsymbol{u}_{k_i},\boldsymbol{z}_{k_i})\}_{i=1}^{\infty}$ converges to F^*. Therefore, the G-CG-LS cost function (3.2) is constant on $\mathcal{L}$ at value F^*. Finally, by Lemma 3.2.6, the sequence $\{(\boldsymbol{u}_k,\boldsymbol{z}_k)\}_{k=0}^{\infty}$ converges to $\mathcal{L}$. ⊠

Lastly, we provide a condition where the G-CG-LS sequence of estimates converges to a single point as stated in the following new corollary.

Corollary 3.2.7. *Let $\mathcal{R}$ satisfy $(\mathcal{A}1)$ and satisfy $(\mathcal{A}2)$ for ISTA G-CG-LS or $(\mathcal{A}3)$ for PGD G-CG-LS. Additionally, if $\mathcal{R}$ is chosen such that the G-CG-LS cost function (3.2) has non-degenerate stationary points then the ISTA G-CG-LS and PGD G-CG-LS sequences $\{(\boldsymbol{u}_k,\boldsymbol{z}_k)\}_{k=1}^{\infty}$ converge.*

Proof. By Proposition 3.2.4 the G-CG-LS sequence $\{(\boldsymbol{u}_k,\boldsymbol{z}_k)\}_{k=0}^{\infty}$ converges to its set of limit points $\mathcal{L}$ that is connected. Since each stationary point is non-degenerate then it is isolated. Hence, $\mathcal{L}$ is a set containing a single single point, which $\{(\boldsymbol{u}_k,\boldsymbol{z}_k)\}_{k=0}^{\infty}$ converges to. ⊠

We remark in choosing a scale variable regularization $\mathcal{R}(\boldsymbol{z})$ to be positively defined on an open domain, for example $\mathcal{R}(\boldsymbol{z}) = \|\log(\boldsymbol{z})\|_2^2$ defined on $(0,\infty)^n$ as in CG-LS, then all stationary points are internal points and, consequently, the assumptions of Theorem 3.2.2 and Corollary 3.2.7 hold, implying G-CG-LS converges to a limit point of zero gradient (or zero subdifferential when $\mathcal{R}$ is convex). In choosing $\mathcal{R}(\boldsymbol{z}) \propto -\log(p_{\boldsymbol{z}}(\boldsymbol{z}))$, as in a MAP estimate, many prior distributions $p_{\boldsymbol{z}}$ produce $\mathcal{R}(\boldsymbol{z})$ defined only on the open domain $(0,\infty)^n$. Specifically, any twice continuously differentiable distribution satisfying $\lim_{z_i\to 0} p_{\boldsymbol{z}}(\boldsymbol{z}) \to 0$ will produce $\mathcal{R}$ only defined on $(0,\infty)^n$, thus satisfying the PGD assumptions of Theorem 3.2.2. Examples of such $\boldsymbol{z}$ distributions useful in modeling image coefficients include log-normal and Gamma$(\alpha,1)$ for $\alpha > 1$ distributions [8, 10, 11]. As CG-LS is a special case of G-CG-LS, Section 2.2 in Chapter 2 provides encompassing numerical results for our G-CG-LS method showing that it outperforms many prior art iterative image reconstruction algorithms.

3.3 Deep Regularized Compound Gaussian Network (DR-CG-Net)

3.3.1 Network Structure

We create a novel DNN named DR-CG-Net, with end-to-end structure shown in Figure 3.1, by applying algorithm unrolling to G-CG-LS in Algorithm 3. Instead of requiring a specified $\mathcal{R}$, DR-CG-Net learns $\mathcal{R}$ through a subnetwork representing either $\nabla\mathcal{R}$ or $\text{prox}_{\eta\mathcal{R}}$. Thus, in training DR-CG-Net, the regularization, and thus scale variable prior distribution, are learned.

Let $\mathcal{V}_k^{(j)} : \mathbb{R}^n \to \mathbb{R}^n$, for $k = 1, 2, \ldots, K$ and $j = 1, 2, \ldots, J$, be a subnetwork. That is, each $\mathcal{V}_k^{(j)}$ is a collection of layers mapping from $\mathbb{R}^n$ to $\mathbb{R}^n$. For example, we use convolutional layers making each $\mathcal{V}_k^{(j)}$ subnetwork a CNN as detailed in Section 3.3.2. Following the scale variable updates of Section 3.2.2 we define the intermediate scale variable mapping, $g_k^{(j)}$, as

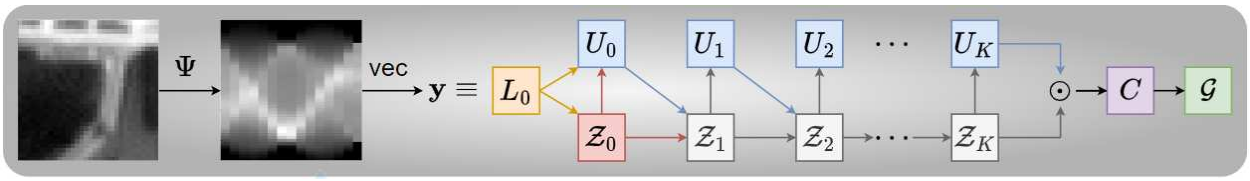
$$g_k^{(j)}(\boldsymbol{z}, \boldsymbol{u}) = \begin{cases} \text{ReLU}\left(r_k^{(j)}(\boldsymbol{u}, \boldsymbol{z}) + \mathcal{V}_k^{(j)}(\boldsymbol{z})\right) & \text{PGD} \\ \text{ReLU}\left(\mathcal{V}_k^{(j)}\left(r_k^{(j)}(\boldsymbol{u}, \boldsymbol{z})\right)\right) & \text{ISTA.} \end{cases} \tag{3.15}$$

where $r_k^{(j)}$ is a mapping of the data fidelity gradient descent step of $F(\boldsymbol{u}, \boldsymbol{z})$ w.r.t. $\boldsymbol{z}$,

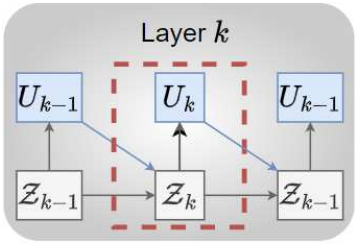
$$r_k^{(j)}(\boldsymbol{u}, \boldsymbol{z}) = \boldsymbol{z} - \eta_k^{(j)}(A_{\boldsymbol{u}}^T(A_{\boldsymbol{u}}\boldsymbol{z} - \boldsymbol{y})) \tag{3.16}$$

for a step size $\eta_k^{(j)} > 0$. Note that $g_k^{(j)}$ corresponds to the update methods in Section 3.2.2 where $\mathcal{V}_k^{(j)}$ replaces $\nabla\mathcal{R}$ or $\text{prox}_{\eta\mathcal{R}}$ in PGD or ISTA, respectively. We apply the ReLU activation function to ensure all intermediate $\boldsymbol{z}$ estimates in DR-CG-Net maintain positive entries. Finally, recall the Tikhonov update defined in (3.4), and let $\mathcal{T}_k(\boldsymbol{z}) = \mathcal{T}(\boldsymbol{z}; P_k)$ for a covariance matrix, P_k.

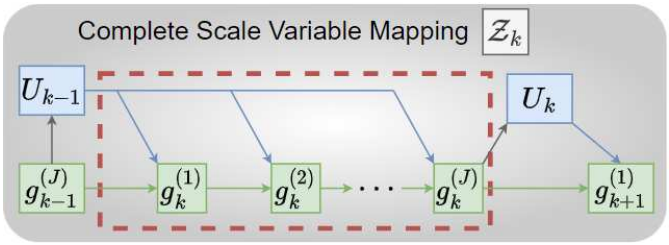
Each layer k of DR-CG-Net, shown in Figure 3.1b, is broken down into a complete scale variable mapping block, $\mathcal{Z}_k$ shown in Figure 3.1c, and a Tikhonov update block, U_k, so that layer k corresponds to iteration k of Algorithm 3. As in Algorithm 3, every complete scale variable



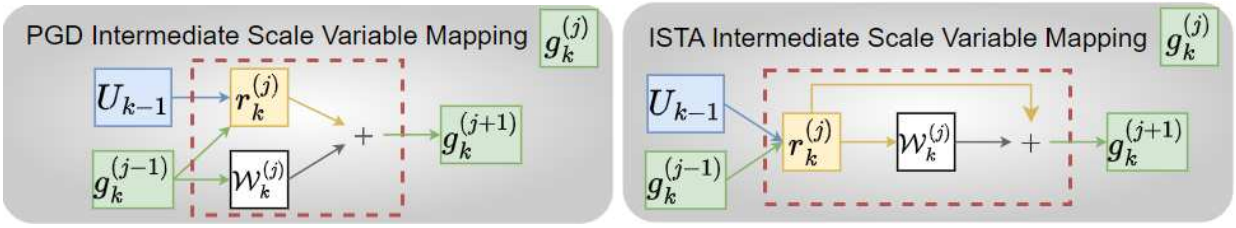
(a) End-to-end network structure of DR-CG-Net.



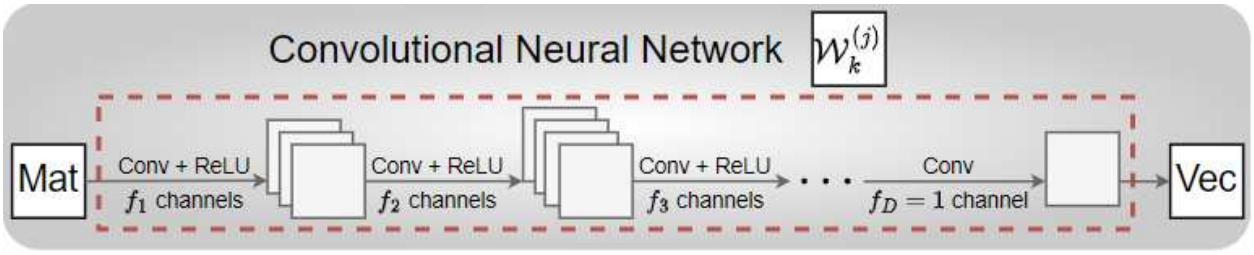
(b) Layer k analogous to iteration k of Algorithm 3.



(c) Complete scale variable mapping, $\mathcal{Z}_k$, producing estimate $\boldsymbol{z}_k$ in Algorithm 3.



(d) Intermediate scale variable mapping, $g_k^{(j)}$, for PGD (left) and ISTA (right) which produces the estimate, $\boldsymbol{z}_k^{(j)}$, in Algorithm 2.



(e) Network representing $\nabla\mathcal{R}$ or $\text{prox}_{\eta\mathcal{R}}$ for PGD or ISTA, respectively. No bias is added, ReLU activation functions are employed, and $f_1, f_2, \ldots,$ and f_D denote the integer number of output filter channels. *Mat* converts a vector into a matrix and *Vec* inverts this process.

Figure 3.1: End-to-end network structure for DR-CG-Net, the unrolled deep neural network of Algorithm 2, is shown in (3.1a). DR-CG-Net consists of an input block, L_0, initialization block, Z_0, $K+1$ Tikhonov blocks, U_k, K complete scale variable mappings, $\mathcal{Z}_k$, a Hadamard product block, C, and an optional refinement block, $\mathcal{G}$. Each $\mathcal{Z}_k$, with structure in (3.1c), consists of J scale variable updates $g_k^{(j)}$ further detailed in (3.1d). Each $g_k^{(k)}$ consists of a data fidelity gradient descent step, $r_k^{(j)}$, added to a convolutional neural network, $\mathcal{W}_k^{(j)}$ in (3.1e), and corresponds to an intermediate update of the $\boldsymbol{z}$ variable.

mapping $\mathcal{Z}_k$ consists of a composition of the J scale variable update blocks $g_k^{(1)}, \ldots, g_k^{(J)}$ shown in Figure 3.1d for PGD and ISTA.

Mathematically detailing the DR-CG-Net blocks we have:

$L_0 = \boldsymbol{y}$ is the input measurements to the network

$\mathcal{Z}_0 = \mathcal{P}_{[0,b]^n}(\hat{A}^T\boldsymbol{y})$, for $\hat{A} = A/||A||_2$, is the initial estimate of $\boldsymbol{z}$ from line 1 of Algorithm 3.

$U_k = \mathcal{T}_k(g_k^{(J)}; P_k)$ (or $\widetilde{\mathcal{T}}(U_{k-1}, g_k^{(J)}; P_k)$ for NAGD), for covariance matrix P_k, is the $\boldsymbol{u}$ Tikhonov estimate corresponding to line 1 and line 10 of Algorithm 3.

The kth complete scale variable mapping $\mathcal{Z}_k$ contains:

$g_k^{(j)} = g_k^{(j)}(g_k^{(j-1)}, U_{k-1})$ is the intermediate scale variable mapping analogous to $\boldsymbol{z}_k^j$ on line 6 in Algorithm 3.

$r_k^{(j)} = r_k^{(j)}(U_{k-1}, g_k^{(j-1)})$ is the data fidelity gradient step.

$\mathcal{W}_k^{(j)} = D$ layer convolutional neural network.

$C = U_K \odot \mathcal{Z}_K$ is the estimate signal, $\boldsymbol{c}$.

$\mathcal{G} = g_{K+1}^{(1)}(C, \mathbf{1})$ is an optional refinement of the estimated signal, $\boldsymbol{c}$, producing the DR-CG-Net output. Note $\mathbf{1} \in \mathbb{R}^n$ is a vector of ones.

Note, to simplify notation, we let $g_k^{(0)} = g_{k-1}^{(J)}$ and $U_{-1} = \mathbf{0}$. We remark that the optional refinement block $\mathcal{G}$ deviates from the structure of G-CG-LS and instead corresponds to an unrolled PGD or ISTA step on the cost function $\widetilde{F}(\boldsymbol{c}) \coloneqq \frac{1}{2}\|\boldsymbol{y} - A\boldsymbol{c}\|_2^2 + \widetilde{\mathcal{R}}(\boldsymbol{c})$ w.r.t. the overall signal, $\boldsymbol{c}$, where $\widetilde{\mathcal{R}}(\boldsymbol{c})$ a learned implicit regularization on $\boldsymbol{c}$. We further remark that minimizing $\widetilde{F}$ may be viewed as finding a solution to the inverse problem of (1.1) directly for the total signal, $\boldsymbol{c}$, rather than through a CG decomposition as in G-CG-LS. As (3.2) is a special case of $\widetilde{F}$, where the CG decomposition from (1.7) is used to split $\boldsymbol{c}$, the refinement block simultaneously updates both $\boldsymbol{z}$ and $\boldsymbol{u}$ such that the interrelationship between these fields is best exploited for the signal of interest.

The refinement block is motivated, in part, by prior art approaches, such as MADUN [22] and ISTA-Net$^+$ [23], that model each DNN layer as an ISTA-type step on the cost function $\widetilde{F}$. As such, our DR-CG-Net approach first exploits the powerful CG prior through minimizing (3.2) to obtain a high quality estimate for the inverse problem to (1.1), and subsequently fine-tunes this estimate through the optional refinement block that mimics a single layer of these prior art methods. Empirically shown in Section 3.3.5, the unrolled G-CG-LS portion of DR-CG-Net, that

is, up to and including block C, is the core component in signal reconstruction performance and that the refinement block marginally refines, e.g. denoises, the final estimate C.

Assume each subnetwork, $\mathcal{V}_k^{(j)}$, uses D layers. Then DR-CG-Net contains $K+1$ estimation layers for $\boldsymbol{u}$, $KJ(D+1)$ layers for updating $\boldsymbol{z}$, one input, one output, and one initialization layer. Thus, DR-CG-Net is a DNN with $K(J(D+1)+1)+4$ layers.

Finally, we provide a slight variation on the DR-CG-Net structure that can be used to reconstruct significantly higher-dimensional signals. Specifically, we use this alternative DR-CG-Net structure for reconstructing images larger than size 64×64 where complications occur in calculation of the Tikhonov solution via (3.5). For significantly high-dimensional signals of interest, the calculation of the Tikhonov solution requires large matrices to be stored and subsequently used in a large linear solver. For example, in the problem of reconstructing a $d \times d$ image, the signal of interest dimension is d^2 and the image sinogram of a Radon transform at s uniformly spaced angles, as calculated by the Python package *scikit-image*, is a $\left(d + \left\lceil(\sqrt{2}-1)d\right\rceil\right) \times s$ image. Thus, the measurement dimension is $s\left(d + \left\lceil(\sqrt{2}-1)d\right\rceil\right)$. Hence, to calculate the Tikhonov solution, a minimum of one $d^2 \times s\left(d + \left\lceil(\sqrt{2}-1)d\right\rceil\right)$ matrix-matrix product and a $s\left(d + \left\lceil(\sqrt{2}-1)d\right\rceil\right) \times s\left(d + \left\lceil(\sqrt{2}-1)d\right\rceil\right)$ linear solve are required. Therefore, a significant computational cost and memory overhead are required to calculate the Tikhonov solution for large d. We empirically found that, with our hardware constraints, often all available working memory would be allocated before the calculation of the Tikhonov solution finished. Furthermore, when the Tikhonov solution can be calculated, more working memory is still required to backpropagate through the Tikhonov solution as multiple higher order tensors, with dimensions dependent on the signal and measurement dimensions, must be calculated and stored.

To accommodate the complications introduced by calculating the Tikhonov solution for significantly high-dimensional signals of interest, we, as discussed in Section 3.2, approximate the Tikhonov solution with J_u gradient descent steps on (3.2) w.r.t. $\boldsymbol{u}$ that are accelerated via Nesterov momentum [78] as shown in Figure 3.2. By replacing the Tikhonov solution with a NAGD approximation, we avoid overloading the available memory capacity to allow the training and testing

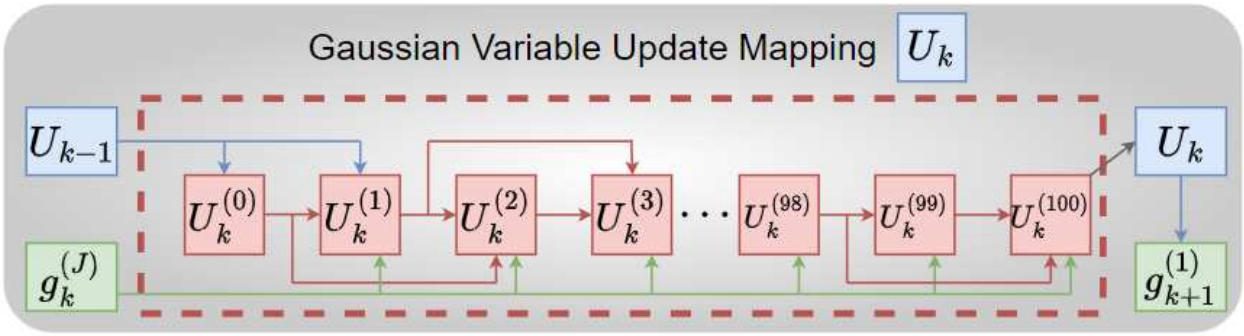


Figure 3.2: Structure of the $\boldsymbol{u}$ update block, U_k, for using gradient descent steps with Nesterov momentum. This block provides an approximate estimate of the Tikhonov solution.

of DR-CG-Net on higher-dimensional signals of interest. We empirically chose $J_u = 100$ NAGD steps as we found this to be the minimum number of steps that provided a sufficient approximation of the Tikhonov solution.

To mathematically detail the replacement of the Tikhonov solution with NAGD steps, we use (3.7) to define the Gaussian variable update method, denoted by $g_u : \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n$, as

$$g_u(\boldsymbol{u}, \boldsymbol{w}, \boldsymbol{z}; \eta, \beta) \equiv g_u(\boldsymbol{u}, \boldsymbol{w}, \boldsymbol{z}) \coloneqq r_u(\boldsymbol{u}, \boldsymbol{z}; \eta) + \beta \left(r_u(\boldsymbol{u}, \boldsymbol{z}; \eta) - r_u(\boldsymbol{w}, \boldsymbol{z}; \eta) \right)$$

where β is a positive, real-valued parameter controlling the Nesterov momentum. That is, a larger β corresponds to a greater amount of momentum allowed in the NAGD update. We adjust DR-CG-Net such that the Tikhonov solution in each U_k block is replaced by the NAGD approximation block shown in Figure 3.2. From Figure 3.2, we set $U_k^{(0)}$ as the current $\boldsymbol{u}$ estimate, i.e. $U_k^{(0)} \equiv U_{k-1}$, and for $j \in \{1, \ldots, 100\}$ we define

$U_k^{(j)} = g_u\left(U_k^{(j-1)}, U_k^{(j-2)}, g_k^{(J)}; \eta, 1 - \frac{3}{6+j}\right)$ is a NAGD update of (3.2) with respect to $\boldsymbol{u}$.

We remark that $U_k^{(-1)}$, used in calculating $U_k^{(1)}$, is set as $U_k^{(-1)} \equiv U_{k-1}$. Furthermore, in calculating the U_0 block, we set $U_0^{(-1)} \equiv U_0^{(0)} = \mathbf{0}$ to be a vector of zeros. Additionally, we choose an adaptive Nesterov momentum parameter of $1 - \frac{3}{6+j}$, where $j \in \{1, \ldots, 100\}$ is the NAGD step count, as proposed in [78]. This adaptive momentum parameter gradually increases the amount of allowed momentum used in subsequent NAGD steps, as, due to (3.2) being convex,

the momentum term $r_u(\boldsymbol{u}, \boldsymbol{z}; \eta) - r_u(\boldsymbol{w}, \boldsymbol{z}; \eta)$ decreases in subsequent NAGD steps. Finally, the step size η is determined on a signal-by-signal basis through a backtracking linesearch algorithm. The backtracking linesearch is calculated outside of the DNN and thus is assumed to be a constant by the network, which will not be backpropagated through.

Note that when we replace the Tikhonov solution with J_u NAGD steps, each U_k block, which previously contained a single layer, now contains J_u layers. We assume that each subnetwork $\mathcal{V}_k^{(j)}$ uses D layers. Then DR-CG-Net contains $J_u(K+1)$ estimation layers for $\boldsymbol{u}$, $KJ(D+1)$ layers for updating $\boldsymbol{z}$, one input, one output, and one initialization layer. Thus, DR-CG-Net with NAGD steps approximating the Tikhonov solution is an extremely deep neural network with $K(J(D+1)+J_u)+J_u+3$ layers.

3.3.2 Network Parameters and Subnetwork

For every $k = 0, 1, \ldots, K$, the layer U_k is parameterized by a covariance matrix, P_k. To reduce the number of parameters learned by the network and for consistency in P_k representing the covariance matrix of $\boldsymbol{u}$, DR-CG-Net learns a single covariance matrix P and constrains $P_1 = \cdots = P_K = P$. Furthermore, we consider the possibility of a structured covariance matrix where P is either a scaled identity, diagonal, tridiagonal, or full matrix. Imposing a covariance structure may be desirable or advantageous, for example to ensure only local reinforcement among entries of the estimated signal $\boldsymbol{c}$.

We remark that to ensure P is a covariance matrix, i.e. symmetric and positive definite, we impose for $\epsilon > 0$ a small fixed real number, one of the following structures

$$P = \begin{cases} \max\{\lambda, \epsilon\} I & \text{Scaled Identity} \\ \text{Diag}([\max\{\lambda_i, \epsilon\}]_{i=1}^n) & \text{Diagonal} \\ L_{\text{tri}} L_{\text{tri}}^T + \epsilon I & \text{Tridiagonal} \\ LL^T + \epsilon I & \text{Full.} \end{cases}$$

In the scaled identity case, only a constant λ is learned. In the diagonal case, a vector $\boldsymbol{\lambda} = [\lambda_i]_{i=1}^n$ is learned. In the tridiagonal case, two vectors $\boldsymbol{\lambda}_1 \in \mathbb{R}^n$ and $\boldsymbol{\lambda}_2 \in \mathbb{R}^{n-1}$ are learned such that the

lower triangular matrix component L_{tri} is formed by placing $\boldsymbol{\lambda}_1$ on the diagonal and $\boldsymbol{\lambda}_2$ on the first subdiagonal. Finally, in the full case, a lower triangular matrix, L, is learned.

Next, each $\mathcal{Z}_k$ block is parameterized by both a collection of step sizes $\{\eta_k^{(1)}, \ldots, \eta_k^{(J)}\}$ (as $\eta_k^{(j)}$ parameterizes the data fidelity gradient step, $r_k^{(j)}$) and by the parameters of the subnetworks $\mathcal{V}_k^{(1)}, \ldots, \mathcal{V}_k^{(J)}$. Similarly, the refinement block is parameterized by a step size $\eta_{K+1}^{(1)}$ and the parameters of the subnetwork $\mathcal{V}_{K+1}^{(1)}$.

We remark that for fixed constant $\gamma_{\max} > 0$ and fixed $\boldsymbol{z}$ and $\boldsymbol{u}$ we take $\eta_k^{(j)}$ from (3.16) as

$$\eta_k^{(j)} = \delta_k^{(j)} \begin{cases} 1 & \|A_{\boldsymbol{u}}^T(A_{\boldsymbol{u}}\boldsymbol{z} - \boldsymbol{y})\|_2 \leq \gamma_{\max} \\ \frac{\gamma_{\max}}{\|A_{\boldsymbol{u}}^T(A_{\boldsymbol{u}}\boldsymbol{z} - \boldsymbol{y})\|_2} & \text{else} \end{cases}$$

and let the network learn $\delta_k^{(j)}$, a real-valued parameter. This is a slight variation on normalized gradient descent [78, 82] for the data fidelity term of (2.2), which we empirically find to provide stability. In particular, normalized gradient descent supplies sample specific step size adjustments to prevent a large and counterproductive gradient descent step.

For detailing the subnetworks, we let $\mathcal{W}_k^{(j)}$ be a CNN of depth D using ReLU activation functions where the convolutions in layer d use a kernel of $k_d \times k_d$ size and produce f_d output filter channels as shown in Fig. 3.1e. Note, zero padding is applied to each filter channel of the input such that that the output at each filter channel is the same size as the input. Furthermore, we take $f_D = 1$ such that for an input image $X \in \mathbb{R}^{n \times n}$ we have $\mathcal{W}_k^{(j)}(X) \in \mathbb{R}^{n \times n}$. Next, for $\boldsymbol{x} \in \mathbb{R}^{n^2}$ we define $\text{mat}(\boldsymbol{x})$ to be $\boldsymbol{x}$ reshaped into a $n \times n$ matrix. Similarly, for $X \in \mathbb{R}^{n \times n}$, we define $\text{vec}(X)$ to be X reshaped into a vector of size n^2 such that $\text{vec}(\text{mat}(\boldsymbol{x})) = \boldsymbol{x}$. Then we set

$$\mathcal{V}_k^{(j)}(\boldsymbol{x}) = \begin{cases} \text{vec}(\mathcal{W}_k^{(j)}(\text{mat}(\boldsymbol{x}))) & \text{PGD} \\ \boldsymbol{x} + \text{vec}(\mathcal{W}_k^{(j)}(\text{mat}(\boldsymbol{x}))) & \text{ISTA.} \end{cases}$$

Define $W_{k,d}^{(j)}$, for $d = 1, 2, \ldots, D$, as the convolutional weight kernels of $\mathcal{W}_k^{(j)}$, which, additionally, parameterize the subnetwork $\mathcal{V}_k^{(j)}$.

From the above, the DR-CG-Net parameters are

$$\boldsymbol{\Theta} = \{P\} \cup \left\{ \delta_k^{(j)}, W_{k,d}^{(j)}, \delta_{K+1}^{(1)}, W_{K+1,d}^{(1)} \right\}_{\substack{k=1,2,\ldots,K \\ d=1,2,\ldots,D}}^{j=1,2,\ldots,J}.$$

Let $f_0 = 1$, $p = \sum_{d=1}^{D} f_{d-1} f_d k_d^2$, and

$$\dim(\mathcal{P}) = \begin{cases} 1 & \text{Scaled Identity Covariance Matrix} \\ n & \text{Diagonal Covariance Matrix} \\ 2n-1 & \text{Tridiagonal Covariance Matrix} \\ \frac{n(n+1)}{2} & \text{Full Covariance Matrix.} \end{cases}$$

As each U_k block shares the same learned covariance matrix, then all U_k blocks contribute $\dim(\mathcal{P})$ parameters to DR-CG-Net. Every $\mathcal{Z}_k$ block, for $k \in \{1, 2, \ldots, K\}$, has J scale variable update blocks $g_k^{(j)}$, for $j \in \{1, 2, \ldots, J\}$, each contributing a single step size parameter and p parameters from the CNN subnetwork, $\mathcal{V}_k^{(j)}$. Similarly, the optional refinement block contributes a single stpe size parameter and p parameters from the CNN subnetwork, $\mathcal{V}_{K+1}^{(1)}$. Thus, DR-CG-Net has $\dim(\mathcal{P}) + (KJ+1)(p+1)$ total parameters.

3.3.3 Loss Function

The DR-CG-Net parameters are trained by minimizing the mean absolute error loss function given as

$$\mathcal{L}_{\mathcal{B}}(\boldsymbol{\Theta}) = \frac{1}{|\mathcal{B}|} \frac{1}{n} \sum_{(\overline{\boldsymbol{y}}_i, \overline{\boldsymbol{c}}_i) \in \mathcal{B}} \|\widehat{\boldsymbol{c}}(\overline{\boldsymbol{y}}_i; \boldsymbol{\Theta}) - \overline{\boldsymbol{c}}_i\|_1 \tag{3.17}$$

where $\mathcal{B}$ is a batch of data points. The mean absolute error loss function is optimized with adaptive moment estimation (Adam) [73], which is a stochastic, gradient-based optimizer. The gradient $\nabla_{\boldsymbol{\Theta}} \mathcal{L}_{\mathcal{B}}$, used by Adam, was calculated via backpropagation through the network where the back-propagation was calculated using automatic differentiation in TensorFlow [75].

3.3.4 Data

We use 32×32 CIFAR10 images [69], 64×64 CalTech101 images [70], and 128×128 LoDoPaB-CT images [83]. Each image has been converted to a single-channel grayscale image, scaled down by the maximum pixel value, and vectorized. We apply a given sensing method Ψ to each image, after which white noise is added, producing noisy measurements $\boldsymbol{y}$ at a specified SNR. This produces a dataset $\mathcal{D}_{N_s} = \{(\overline{\boldsymbol{y}}_i, \overline{\boldsymbol{c}}_i) : i = 1, 2, \ldots, N_s\}$ where N_s is the number of data samples i.e. the number of signals used. For a given sensing method and noise, we create three such datasets: a training dataset, validation dataset, and testing dataset. The training dataset is used for training a deep-learning-based method, the validation dataset is used to determine overfitting, and the testing dataset is used after training to assess the performance of a deep-learning-based method. Note that we randomly sample across all classes of images from CIFAR10 and CalTech101 in forming the training, validation, and testing datasets.

We consider two types of sensing matrices:

1. Radon Transform: Typical in tomography, specifically X-Ray computed tomography, we form sensing matrices Ψ that correspond to Radon transforms at a number of uniformly spaced angles. For 32×32 images we use 15, 10, and 6 uniformly spaced angles. For 64×64 images we use 30, 22, and 15 uniformly spaced angles. Lastly, for 128×128 images we use 76 and 60 uniformly spaced angles.

2. Random Gaussian Matrix: Typical in CS, $\Psi \in \mathbb{R}^{m \times n}$ has entries sampled from a standard Gaussian distribution with a given sampling ratio defined at $\frac{m}{n}$. We form three sensing matrices corresponding to sampling ratios of $0.5, 0.3,$ and 0.1.

Finally, we consider measurements with two different SNRs of 60dB or 40dB.

3.3.5 Numerical Results

Two types of DR-CG-Nets, called PGD DR-CG-Net and ISTA DR-CG-Net, are considered and corresponding, respectively, to using the PGD or ISTA intermediate scale variable mapping given

in (3.15). For unrolled iterations we set $(K, J) = (3, 4)$ for 32×32 image reconstructions and $(K, J) = (1, 24)$ for 64×64 and 128×128 image reconstructions. Furthermore, each $\mathcal{V}_k^{(j)}$ is taken to be a CNN with depth $D = 8$. These network size parameters where chosen empirically such that the time to complete a signal reconstruction was reasonably quick while still producing excellent reconstructions on a validation dataset. Every convolution uses a 3×3 kernel initialized according to the Glorot Uniform distribution [84] with ReLU activation functions and $f_1 = \cdots = f_7 = 32$ and $f_8 = 1$ filter channels.

We initialize each step size as $\delta_k^{(j)} = 1$, set $\gamma_{\max} = 1$, and fix $\epsilon = 10^{-4}$. Each $\boldsymbol{u}$ covariance matrix structure, P_u, is initialized as a diagonal matrix with 0.1 or 10 on the diagonal for Radon transform or Gaussian measurements, respectively. Additionally, for the initial $\boldsymbol{z}$ estimate, we set $\mathcal{P}_{[0,b]^n} = \mathcal{P}_{[0,10]^n}$. We train all DR-CG-Nets using a learning rate of 10^{-4} for 2000 epochs implementing early stopping as necessary to prevent overfitting.

We compare our DR-CG-Net method against ten state-of-the-art deep-learning-based methods: (i) compound Gaussian network (CG-Net) [57, 58], (ii) memory augmented deep unfolding network (MADUN) [22], (iii) ISTA-Net$^+$ [23], (iv) FISTA-Net [24], (v) iPiano-Net [30], (vi) ReconNet [13], (vii) LEARN^{++} [29], (viii) Learned Primal-Dual (LPD) [31], (ix) FBPConvNet [56], and (x) iRadonMAP [55]. Although memory augmented proximal unrolled network (MAPUN) [26] was considered, due to the similarity in performance to MADUN only MADUN results are shown. Finally, two model-based approaches, filtered backprojection (FBP) [71] and super-voxel model-based iterative reconstruction (svMBIR) [85], are compared. As these model-based approaches are generally not as competitive as the top deep-learning methods, their results are relegated to the single plot of Fig. 3.12 to provide a baseline.

Note that LEARN^{++}, LPD, FBPConvNet, and iRadonMAP are CT-specific reconstruction methods relying on the structure of the CT sinogram measurements. On the other hand, MADUN, ISTA-Net$^+$, FISTA-Net, iPiano-Net, and ReconNet are reconstruction methods with particular application in image CS. Furthermore, we remark that CG-Net, MADUN, ISTA-Net$^+$, FISTA-Net, iPiano-Net, LEARN^{++}, and LPD are DNNs formed by algorithm unrolling while ReconNet,

iRadonMAP, and FBPConvNet are instead standard DNNs. Lastly, as with DR-CG-Net, a refinement block is added to, and trained with, each CG-Net.

Main Results

For each set of data discussed in Section 3.3.4, we train an ISTA DR-CG-Net, PGD DR-CG-Net, and each of the ten comparison methods using varying amounts of training dataset sizes. For CIFAR10 images, the training datasets consist of $N_s = 20, 100, 500, 1000,$ and 2000 data samples and, after training, 8000 test data samples are provided to every network to assess its performance. From varying the amount of training data, we find that a core utility of DR-CG-Net is its superior performance on small training datasets, i.e. a highly underdetermined system in (3.17). As such, for CalTech101 and LoDoPaB-CT images, we use training datasets with $N_s = 20$ samples, to focus on evaluating low training scenarios in greater detail, and 200 test data samples are provided to every network after training.

Average PSNR and SSIM quality metrics on the test dataset reconstructions are used to evaluate network performance where higher values of these metrics correspond to reconstructed images that more closely match the original images. Note, the test data samples are formed from the same sensing matrix and measurement SNR that produced the training data, but are unseen by the network during the training process.

Next, we remark that every method was trained using early stopping. That is, as shown in Figure 3.3, training was conducted until the model initially overfits as compared to a validation dataset. We define initial overfitting as the point after which the performance of the model on a validation dataset no longer improves with further training of the models' parameters on the training dataset. By implementing early stopping, we ensure every model is sufficiently trained while also not being over trained thereby presenting the best performance for each model given the provided set of training data.

For training datasets of size $N_s = 20$ and $N_s = 100$, we set P_u as a scaled identity covariance matrix structure. A tridiagonal covariance matrix structure for P_u is used for all other training datasets. Next, for images larger than 64×64, due to memory constraints in backpropagating the

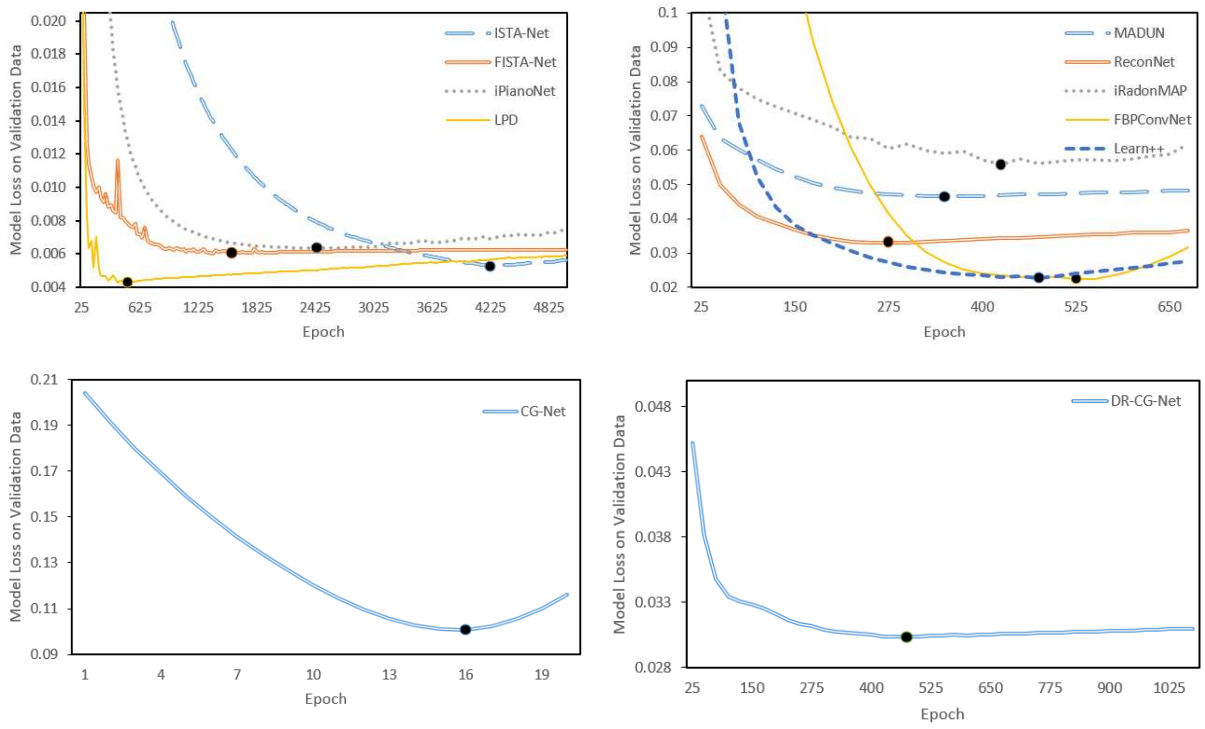
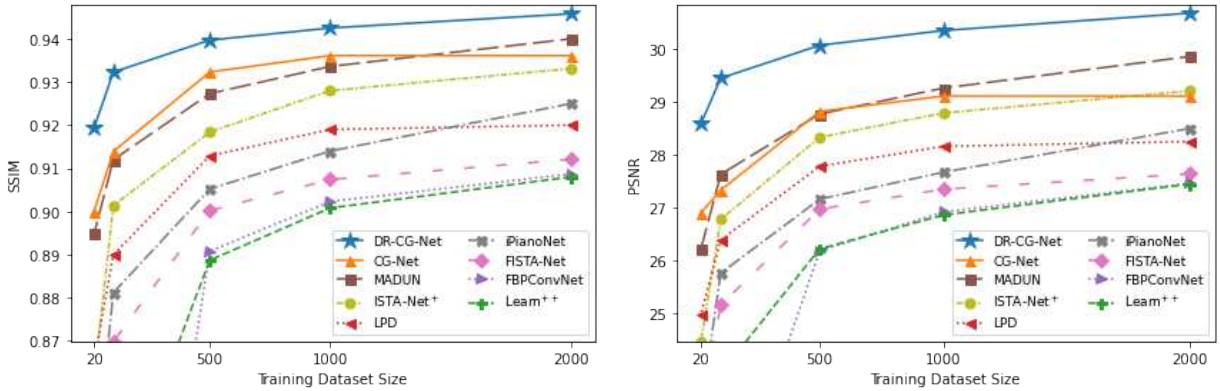


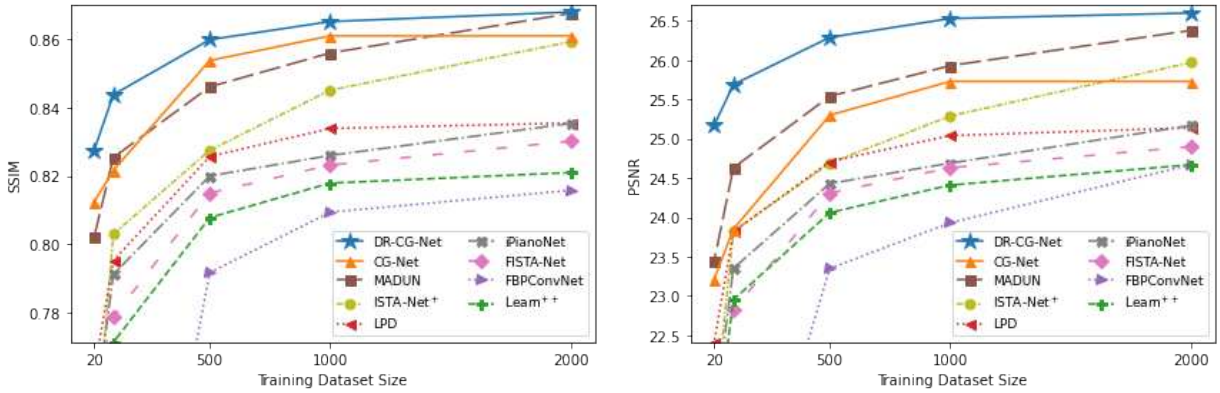
Figure 3.3: Model loss curves on a validation dataset for training each deep learning-based method on Radon inversion from 15 uniformly spaced angles with 20 training samples. The point on each model's loss curve represents the epoch in which that model achieved its best loss on a validation dataset and overfits on subsequent epochs. Only these best loss epoch results are presented for each method. Note, for the plots above, CG-Net uses an SSIM loss function, MADUN and DR-CG-Net use a mean absolute error loss function, and all other methods use a mean square error loss function (possibly with some additional regularization). As the network loss functions are not equivalent, these plots do not contribute a comparison between the methods and only illustrate that each method is maximally trained for every supplied training dataset.

Tikhonov solution, we approximate the Tikhonov solution with 100 NAGD steps as detailed in Fig. 3.2. Lastly, for Radon inversion experiments, we take $\Phi = I$ allowing DR-CG-Net to intrinsically learn the optimal signal representation basis. Instead, when reconstructing from Gaussian measurements, we take $\Phi =$ discrete cosine transformation.

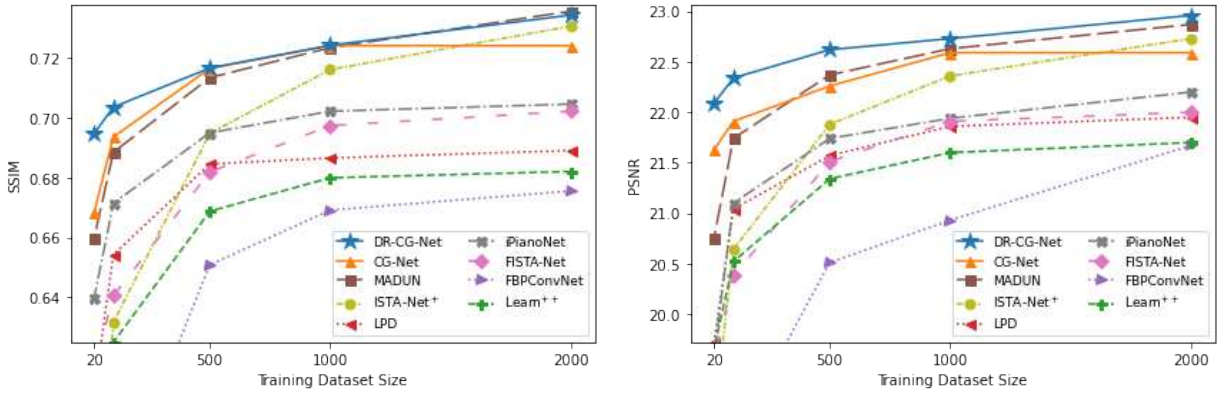
Shown in Figure 3.4 and Figure 3.5 are the average SSIM and PSNR quality metrics, over a set of 8000 test CIFAR10 image reconstructions, from ISTA DR-CG-Net and eight state-of-the-art deep-learning-based comparison methods where each method is trained on a varying amount of training data. The CIFAR10 images for training and testing are reconstructed from Radon transform measurements, at 15, 10, or 6 uniformly spaced angles, with an SNR of 60dB and 40dB



(a) Reconstructions from fifteen uniformly spaced angle Radon transforms.



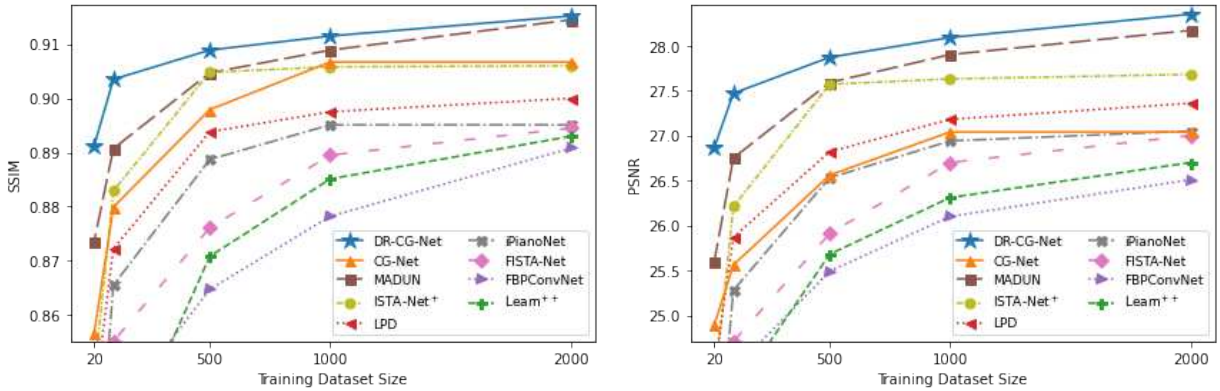
(b) Reconstructions from ten uniformly spaced angle Radon transforms.



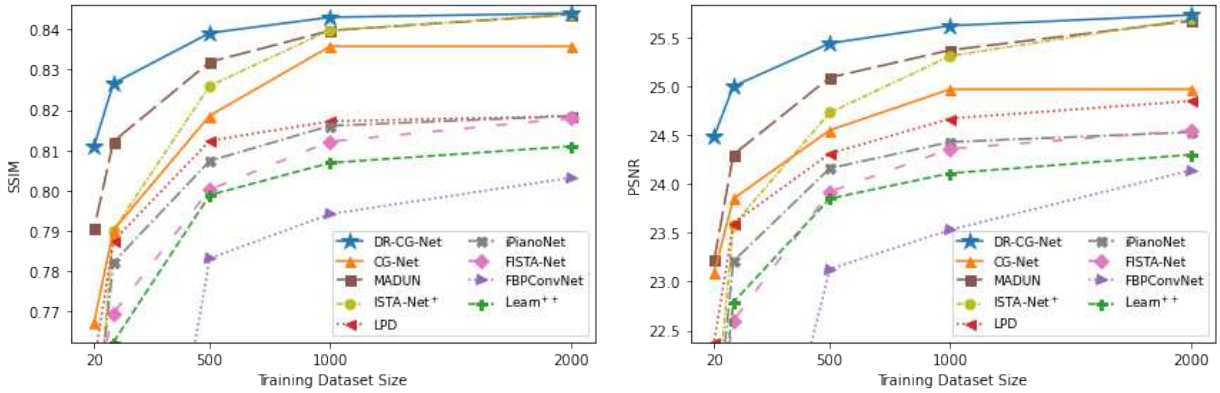
(c) Reconstructions from six uniformly spaced angle Radon transforms.

Figure 3.4: Average test image reconstruction SSIM and PSNR when we vary the amount of CIFAR10 data in training nine machine learning-based image reconstruction methods. Here, the sensing matrices, Ψ, are a Radon transform at 15, 10, or 6 uniformly spaced angles, $\Phi = I$, and the measurement SNR is 60dB. **Our DR-CG-Net method outperforms the compared prior art methods, in all scenarios, and does so appreciably in low training**.

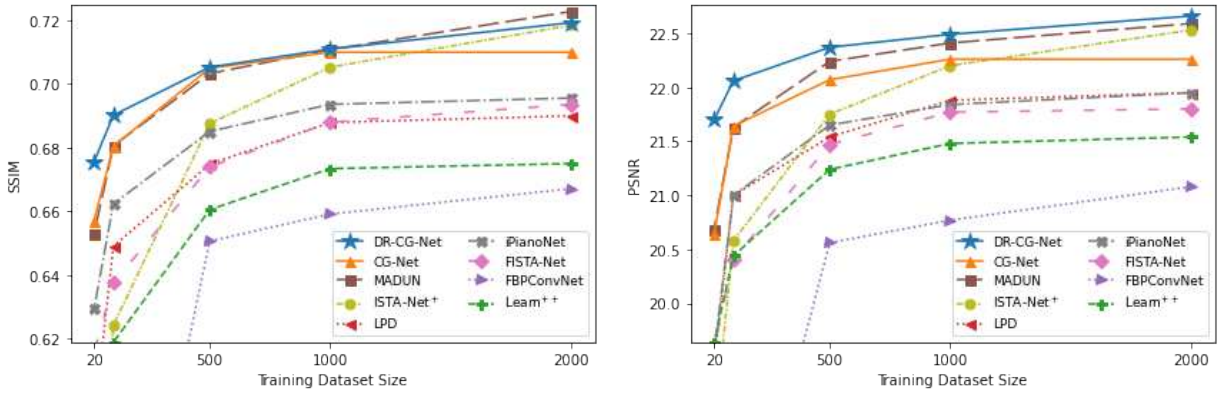
for Figure 3.4 and Figure 3.5, respectively. Note that the ReconNet and iRadonMAP methods perform significantly lower and are thus omitted from Figure 3.4 and Figure 3.5. Additionally, as



(a) Reconstructions from fifteen uniformly spaced angle Radon transforms.



(b) Reconstructions from ten uniformly spaced angle Radon transforms.



(c) Reconstructions from six uniformly spaced angle Radon transforms.

Figure 3.5: Average test image reconstruction SSIM and PSNR when we vary the amount of CIFAR10 data in training nine machine learning-based image reconstruction methods. Here, the sensing matrices, Ψ, are a Radon transform at 15, 10, or 6 uniformly spaced angles, $\Phi = I$, and the measurement SNR is 40dB. **Our DR-CG-Net method outperforms the compared prior art methods, in all scenarios, and does so appreciably in low training**.

PGD DR-CG-Net performs equivalently to ISTA DR-CG-Net, it too was omitted from Figure 3.4 and Figure 3.5. From Figure 3.4 and Figure 3.5, we see that our DR-CG-Net method outperforms,

or performs comparably to, each compared prior art method in every training scenario and excels significantly in the lowest training scenarios of $N_s = 20$ and $N_s = 100$.

With low training and low measurement noise, CG-Net is the second highest performing method as measured by SSIM, which generally corresponds to better image quality from a human perspective than PSNR. This is expected since, as shown in the results of Chapter 2, CG-Net performs exceptionally well in low noise and low training scenarios. Otherwise, in general, MADUN is the second highest performing method behind our DR-CG-Net. Comparing to the results presented in Chapter 2, MADUN performs significantly higher here, which is attributed to setting the sparsity transform matrix to an identity matrix and allowing the network to learn the optimal sparsity representation. Similarly, for DR-CG-Net in the Radon inversion experiments, using a sparsity transformation reduces performance as compared to using an identity matrix where the network is allowed to learn the optimal sparsity representation basis. We believe this, in part, to be a consequence of the convolutions present in both DR-CG-Net and MADUN, as many previous works have shown convolutional layers to be excellent in processing and learning details from images. While setting the sparsity transformation to be the identity matrix forces the MADUN convolutions to act directly on the image, in DR-CG-Net the convolutions act on the $\boldsymbol{z}$ field of the image rather than the $\boldsymbol{z}$ field of the image sparsity coefficients.

For higher training, roughly around 2000 data samples, and a lower number of angles in the Radon transform, MADUN and our DR-CG-Net perform about equivalently as shown in Fig. 3.5. We speculate that the lower performance of DR-CG-Net with lower numbers of angles in the Radon transform, as compared to MADUN, may be attributed to the Tikhonov solution which struggles to backproject from a measurement space of much lower dimension than that of the signal-of-interest space. Further note that the discrepancy between DR-CG-Net and MADUN decreases as the amount of measurement noise increases. We speculate that this also may be attributed to the Tikhonov solution, which closely fits the measured estimated signal to the measurements and thus overfits to undesirable measurement noise. Similarly, we conjecture that the Tikhonov solution may attribute to the diminishing improvement of DR-CG-Net, over the comparison methods, as a

greater amount of training data is used. The Tikhonov solution, which closely fits the measured estimated signal to the true measurements, may be restricting the capacity that DR-CG-Net can learn when significant training data is available. Although we have stated a few speculated limitations of the Tikhonov solution, we still emphasize that the inclusion of the Tikhonov solution, from the CG prior decomposition, is a crucial component to the success of DR-CG-Net, especially in low-training scenarios. Further empirical investigation into all three of these properties and studying possible remedies, such as expanding the refinement block for DR-CG-Net when higher measurement noise is present, is a subject of future work.

Visual comparisons of all methods, in reconstructing 32×32 images from Radon transform measurements, are displayed in Figure 3.6, Figure 3.7, and Figure 3.8. Displayed in Figure 3.6 are the reconstructions of a 32×32 test cat image, from a Radon transform at 15 uniformly spaced angles with 60dB SNR, after training with only 20 samples. Similarly, displayed in Figure 3.7 are the reconstructions of a 32×32 test car image, from a Radon transform at 10 uniformly spaced

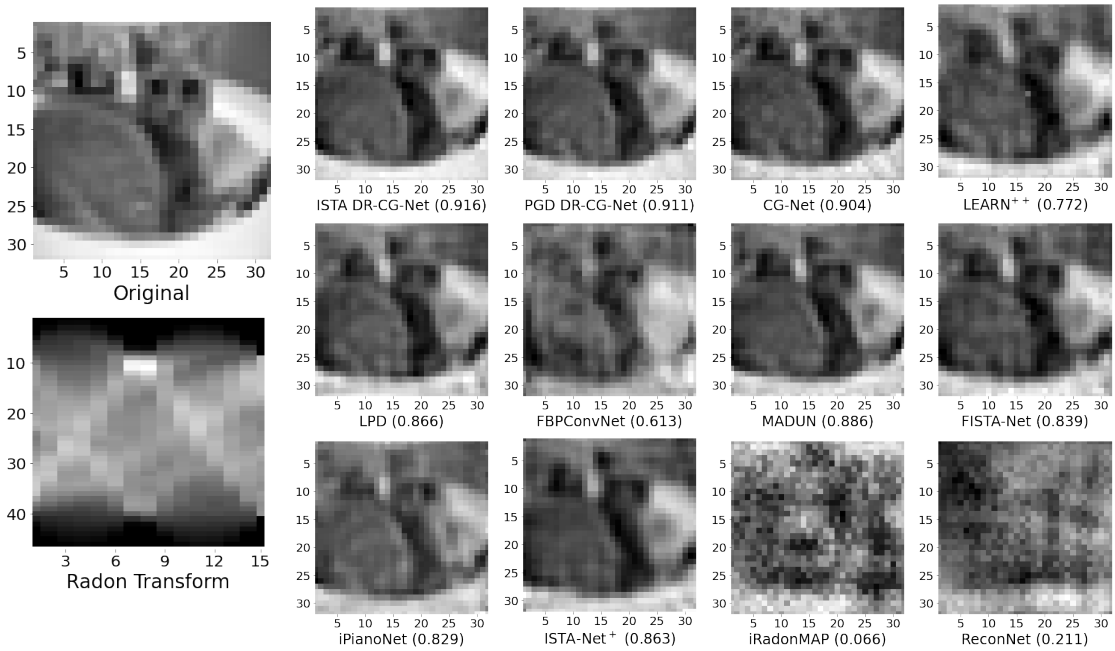


Figure 3.6: Image reconstructions (SSIM) using our DR-CG-Nets and ten comparative deep learning methods on a 32×32 cat image after training on only 20 samples. The sensing matrix, Ψ, is a Radon transform at 15 uniformly spaced angles, $\Phi = I$, and the measurement has an SNR of 60dB. **Our DR-CG-Net methods perform best visually and by SSIM.**

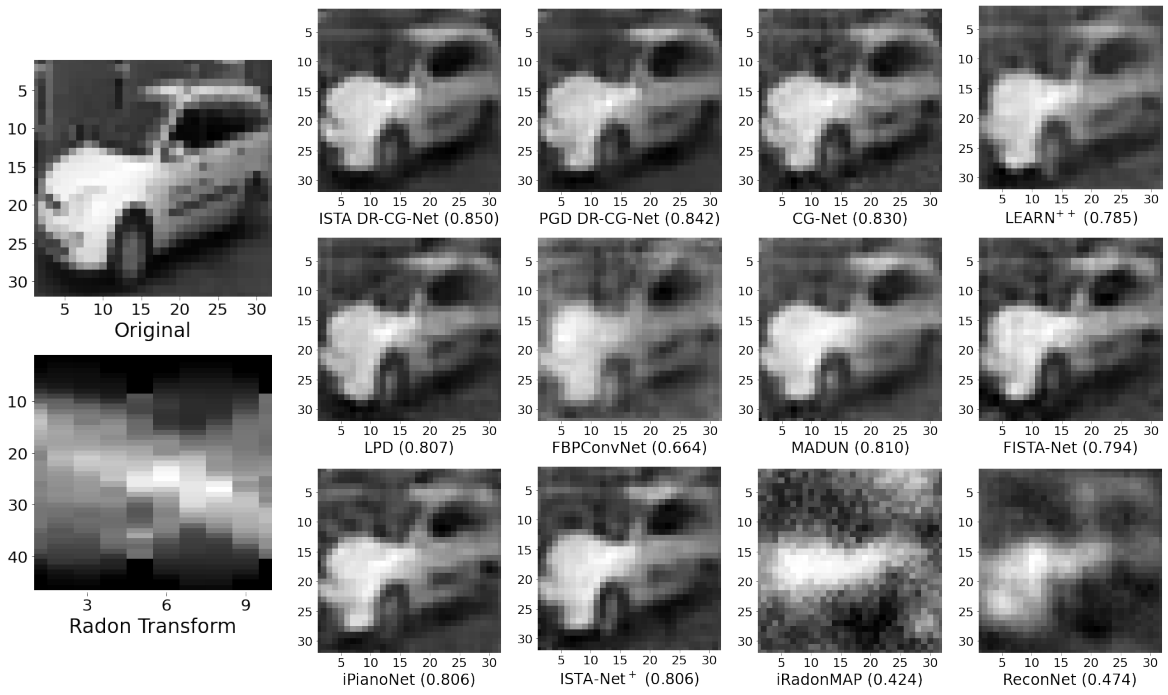


Figure 3.7: Image reconstructions (SSIM) using our DR-CG-Nets and ten comparative deep learning methods on a 32×32 car image after training on only 100 samples. The sensing matrix, Ψ, is a Radon transform at 10 uniformly spaced angles, $\Phi = I$, and the measurement has an SNR of 60dB. **Our DR-CG-Net methods perform best visually and by SSIM.**

angles with 60dB SNR, after training with only 100 samples. Finally, displayed in Figure 3.8 are the reconstructions of a 32×32 test horse image, from a Radon transform at 6 uniformly spaced angles with 60dB SNR, after training with 1000 samples. From each visualization, we see ISTA and PGD DR-CG-Net performing comparably and producing superior reconstructions, both visually and by SSIM, to all ten comparison methods.

Average test image reconstruction SSIM and PSNR from training and testing with the CalTech101 and LoDoPaB-CT datasets are detailed in Figure 3.9. In particular, Figure 3.9b provides bar plots showing the mean SSIM and PSNR with 99% confidence intervals for our ISTA DR-CG-Net, CG-Net, MADUN, ISTA-Net^{+}, iPiano-Net, FISTA-Net, LPD, LEARN++, and FBP-ConvNet. Each method reconstructed 200 test CalTech101 images after training with a dataset of only 20 samples where the measurements are Radon transforms at 30, 22, or 15 uniformly spaced angles with an SNR of 60dB or 40dB. Additionally, Figure 3.9a provides bar plots showing the mean SSIM and PSNR with 99% confidence intervals for our ISTA DR-CG-Net, MADUN,

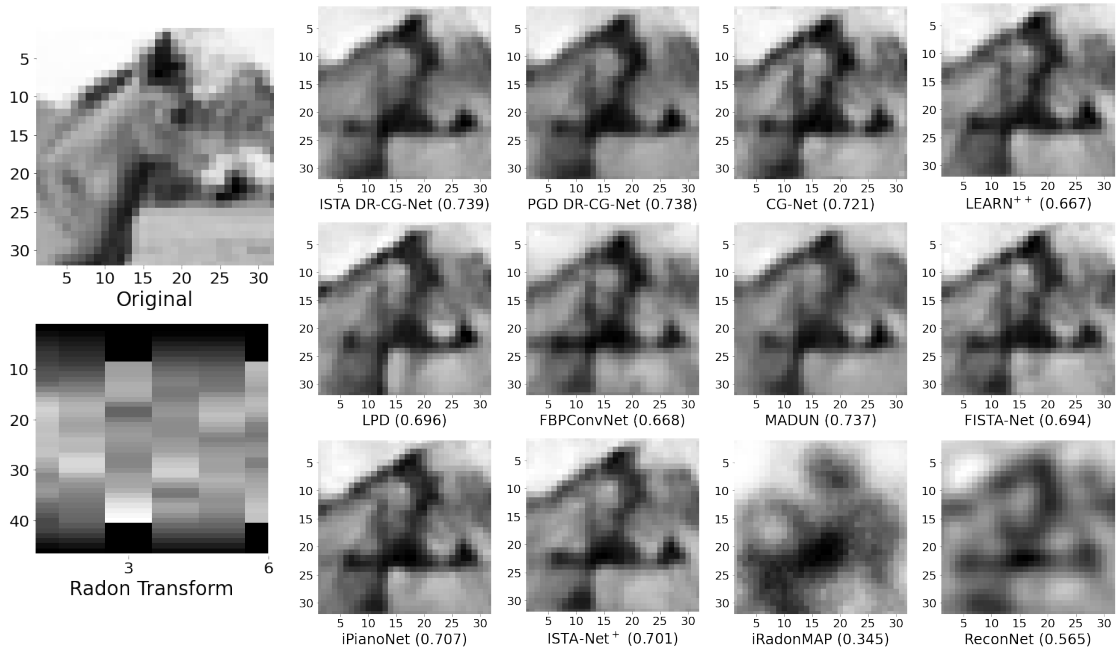
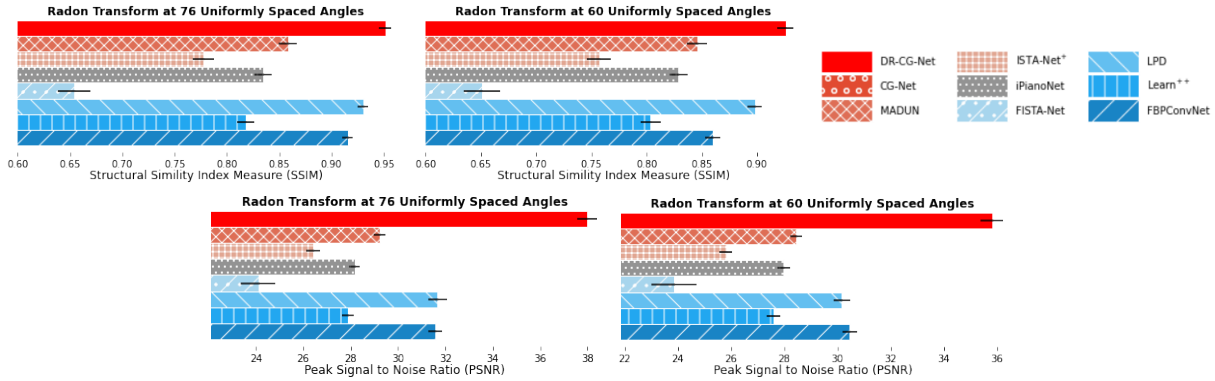


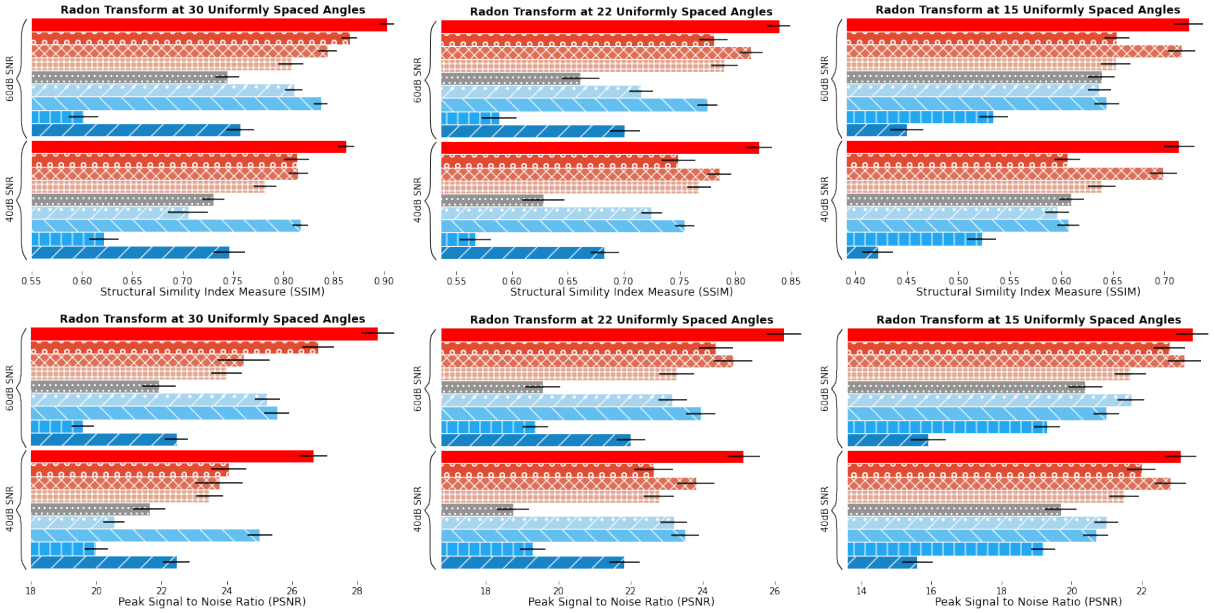
Figure 3.8: Image reconstructions (SSIM) using our DR-CG-Nets and ten comparative deep learning methods on a 32×32 horse image after training with 1000 samples. The sensing matrix, Ψ, is a Radon transform at 6 uniformly spaced angles, $\Phi = I$, and the measurement has an SNR of 60dB. **Our DR-CG-Net methods perform best visually and by SSIM.**

ISTA-Net$^+$, iPiano-Net, FISTA-Net, LPD, LEARN++, and FBPConvNet. Each method reconstructed 200 test LoDoPaB-CT images after training with a dataset of only 20 samples where the measurements are Radon transforms at 76 or 60 uniformly spaced angles with an SNR of 60dB. As ReconNet and iRadonMAP perform much lower they omitted from Figure 3.9. Additionally, as PGD DR-CG-Net performs nearly identical to ISTA DR-CG-Net only the ISTA DR-CG-Net results are shown in Figure 3.9. From Figure 3.9 we observe that in reconstructing larger images from Radon transform measurements DR-CG-Net still appreciably outperforms all ten comparison methods in low-training scenarios.

Sample reconstructions of CalTech101 and LoDoPaB-CT images from Radon transform measurements, after training with only 20 samples, are displayed in Figure 3.10, Figure 3.11, Figure 3.12, and Figure 3.13. In particular, Figure 3.10 and Figure 3.11 each show the reconstructions of a test LoDoPaB-CT scan image from a Radon transform at 76 and 60 uniformly spaced angles, respectively. Instead, Figure 3.12 shows the reconstructions of a test CalTech101 spider image



(a) Average SSIM and PSNR, with 99% confidence intervals, for nine deep learning-based image estimation methods reconstructing 200, 128×128 LoDoPaB-CT images [83] from Radon transform measurements. Each measurement has an SNR of 60dB and $\Phi = I$.



(b) Average SSIM and PSNR, with 99% confidence intervals, for nine deep learning-based image estimation methods reconstructing 200, 64×64 CalTech101 images [70] from Radon transform measurements. Each measurement has an SNR of 60dB or 40dB and $\Phi = I$.

Figure 3.9: Test image reconstruction quality for deep learning-based image estimation methods trained on only 20 samples. **In all cases, our method, DR-CG-Net given by the top red bar, outperforms the other approaches.**

from a Radon transform at 30 uniformly spaced angles. Lastly, Figure 3.13 shows the reconstructions of a test CalTech101 jar image from a Radon transform at 15 uniformly spaced angles. Each Radon transform measurement had an SNR of 60dB and every reconstruction shown is produced from testing a deep learning method after training it with only 20 samples. Again, from each visualization we see ISTA and PGD DR-CG-Net performing comparably and producing superior

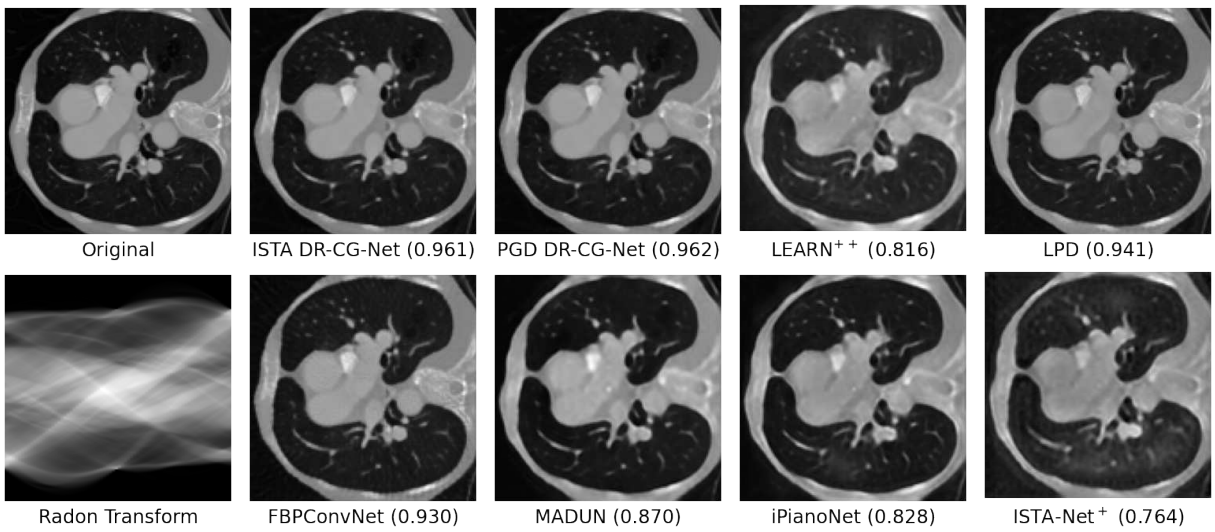


Figure 3.10: Image reconstructions (SSIM) using our DR-CG-Net and six competitive deep learning methods on a 128×128 test scan after training with only 20 samples. The sensing matrix, Ψ, is a Radon transform at 76 uniformly spaced angles, $\Phi = I$, and each measurement has an SNR of 60dB. Our DR-CG-Net methods perform best visually and by SSIM.

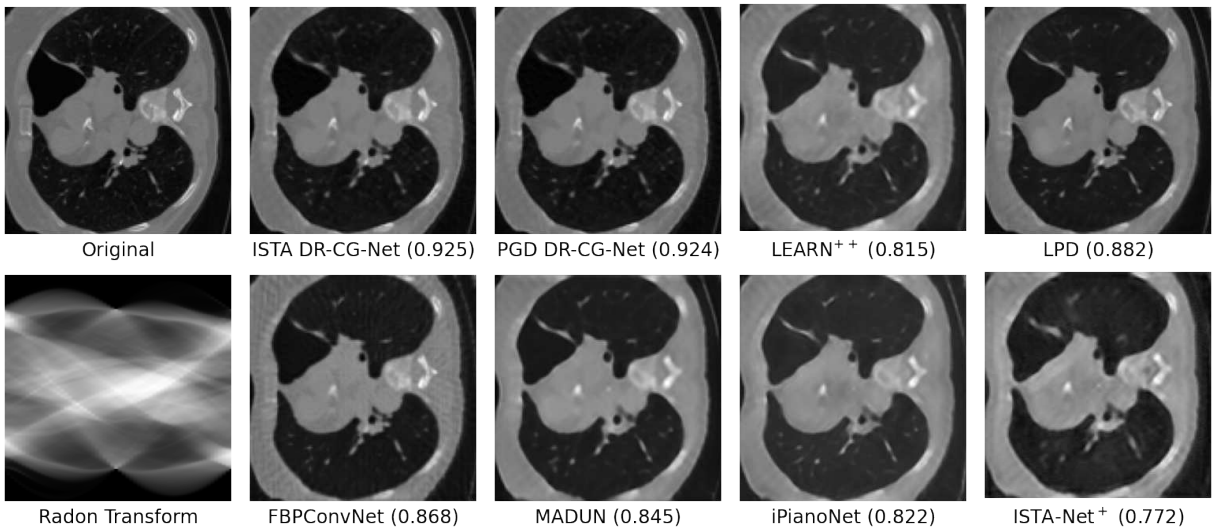


Figure 3.11: Image reconstructions (SSIM) using our DR-CG-Net and six competitive deep learning methods on a 128×128 test scan after training with only 20 samples. The sensing matrix, Ψ, is a Radon transform at 76 uniformly spaced angles, $\Phi = I$, and each measurement has an SNR of 60dB. Our DR-CG-Net methods perform best visually and by SSIM.

reconstructions, both visually and by SSIM, to all ten comparison methods. We particularly highlight, from Figure 3.10 and Figure 3.11, that DR-CG-Net can recover small features present in the original image that are of importance in medical imaging, e.g. X-ray CT and MRI, for identifying and diagnosing aliments.

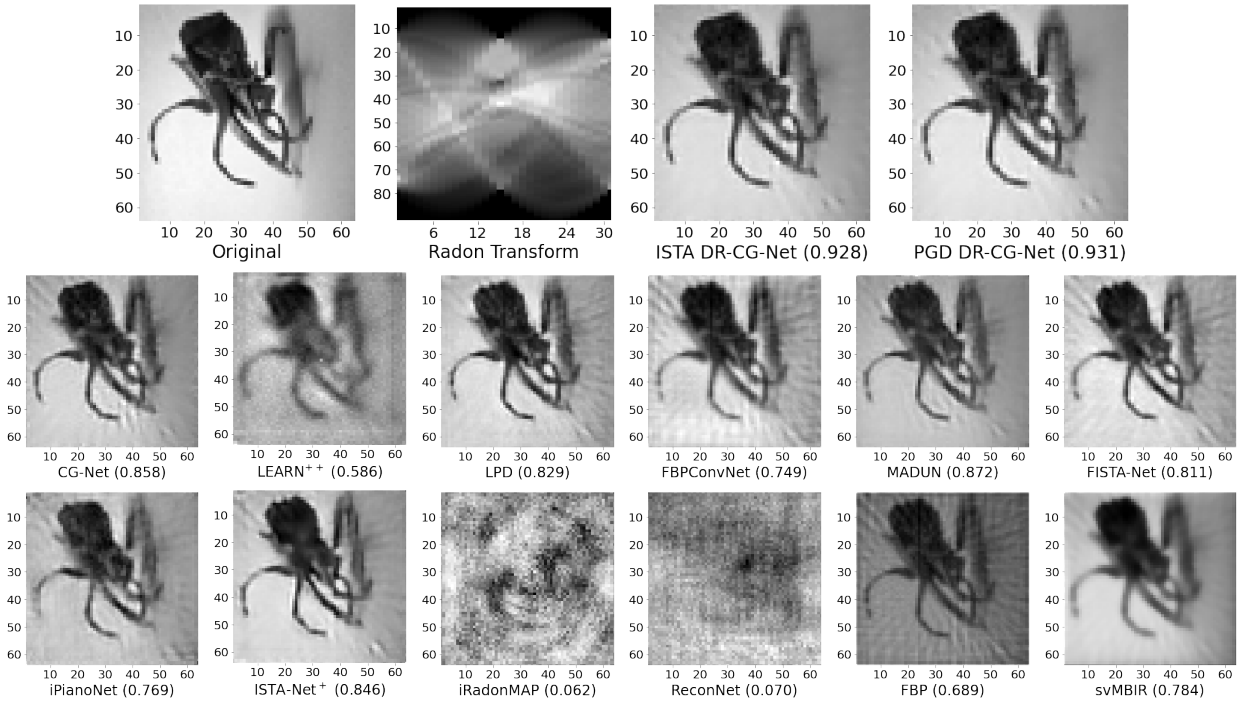


Figure 3.12: Image reconstructions (SSIM) using our DR-CG-Nets, ten comparative deep learning methods, and two baseline iterative-based methods on a 64×64 spider image. The sensing matrix, Ψ, is a Radon transform at 30 uniformly spaced angles, $\Phi = I$, and each measurement has an SNR of 60dB. Our DR-CG-Net methods perform best visually and by SSIM.

To further highlight the applicability of our DR-CG-Net methods, we reconstruct CIFAR10 and CalTech101 images from random Gaussian measurements as in the field of compressed sensing. Figure 3.14 shows the average SSIM and PSNR, with 99% confidence intervals, from reconstructing test CIFAR10 and CalTech101 images from Gaussian measurements at a sampling ratio of 0.5, 0.3, or 0.1. Each method displayed in the bar plot of Figure 3.14a reconstructed 8000 test images after training on a set of only 20 samples. Similarly, each method displayed in the bar plot of Figure 3.14b reconstructed 200 test images after training on a set of only 20 samples. As LPD, LEARN^{++}, iRadonMAP, and FBPConvNet are CT-specific reconstruction methods relying on the structure of the CT sinogram measurements, the compressive sensing problem is not applicable and thus these comparisons are omitted from Figure 3.14. As ReconNet is significantly the lowest performing method, the results of this method are omitted from Figure 3.14. Additionally, only ISTA DR-CG-Net results are shown in Figure 3.14 since PGD DR-CG-Net performs nearly identically. Note that for all Gaussian measurement reconstructions, a discrete cosine transform matrix

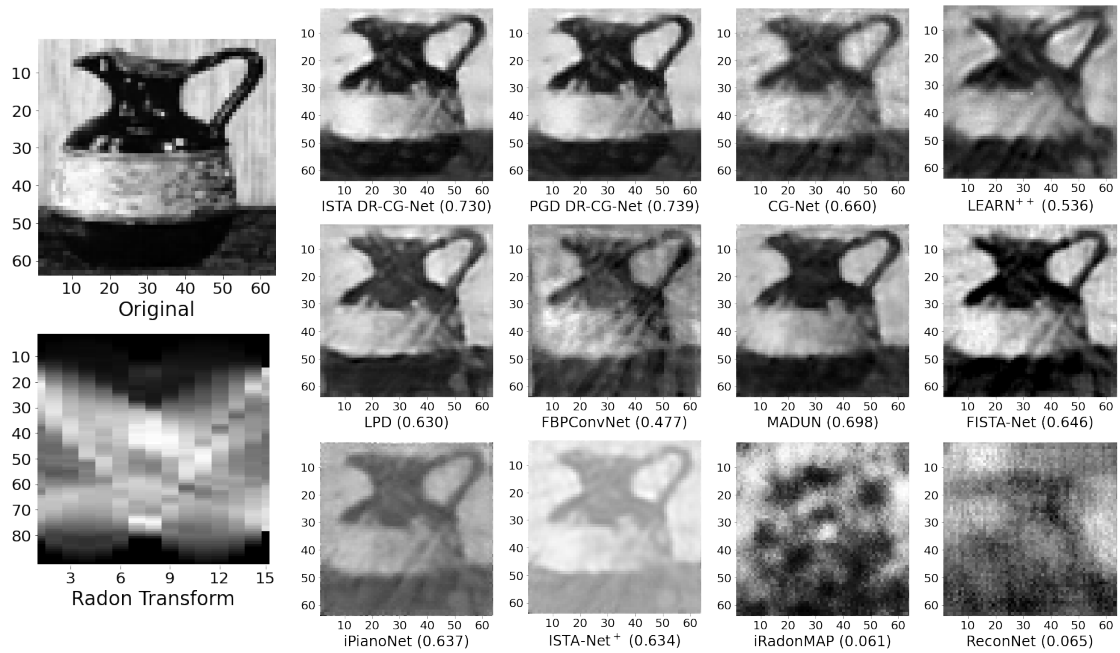
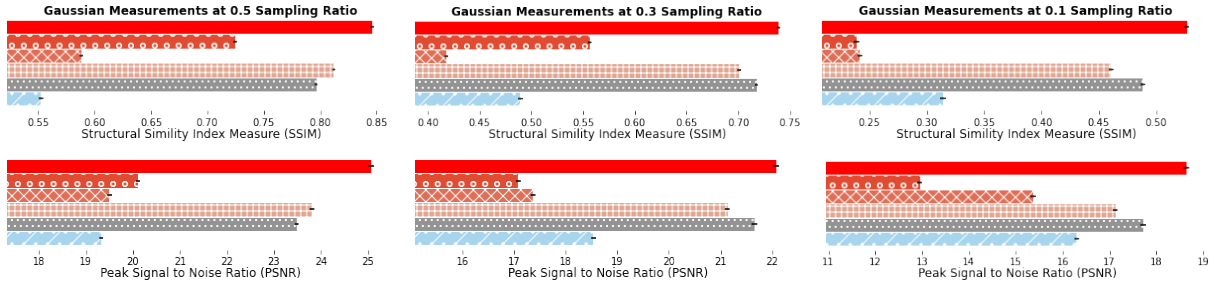


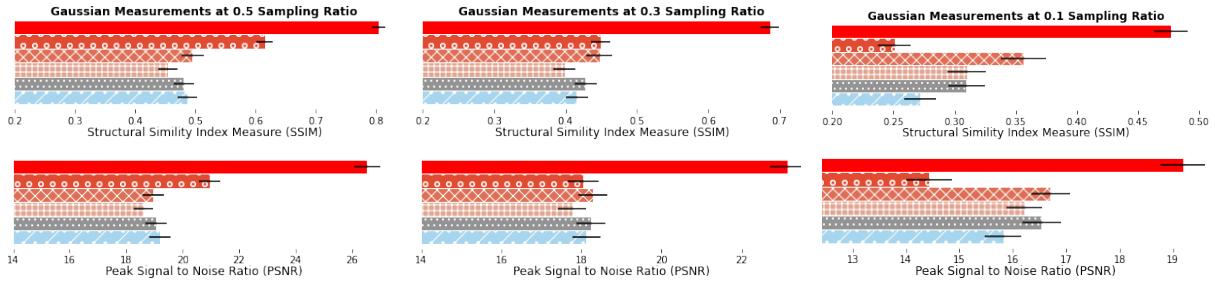
Figure 3.13: Image reconstructions (SSIM) using our DR-CG-Nets and ten comparative deep learning methods on 64×64 Barbara images after training on only 20 samples. The sensing matrix, Ψ, is a Radon transform at 15 uniformly spaced angles, $\Phi = I$, and each measurement has an SNR of 60dB. Our DR-CG-Net methods perform best visually and by SSIM.

is used for Φ. From Figure 3.14 we observe that even with alternative measurement matrices, our DR-CG-Net method still appreciably outperforms all ten comparison methods in low-training scenarios. We remark that our DR-CG-Net method is not limited to the Radon transform and random Gaussian measurement cases specifically considered here. Likely, DR-CG-Net has a greater widespread applicability to linear inverse problems, focused on image estimation, with any general measurement matrix. Further empirical study of DR-CG-Net in reconstructing images from other measurement types is an ongoing subject of future work.

Sample reconstructions of CIFAR10 and CalTech101 images from Gaussian transform measurements are displayed in Figure 3.15 and Figure 3.16. In particular, Figure 3.15 and Figure 3.16 respectively display a test CIFAR10 plane image and test CalTech101 strawberry image from a Gaussian measurement at a 0.5 sampling ratio. Each Gaussian measurement had an SNR of 60dB and every reconstruction shown is produced from testing a deep-learning method after the method is trained with only 20 samples. From each visualization we see ISTA and PGD DR-CG-Net per-



(a) Average SSIM and PSNR, with 99% confidence intervals, for six deep learning-based image estimation methods reconstructing 8000, 32×32 CIFAR10 images.



(b) Average SSIM and PSNR, with 99% confidence intervals, for six deep learning-based image estimation methods reconstructing 200, 64×64 CalTech101 images.

Figure 3.14: Test image reconstruction quality for six deep learning-based image estimation methods trained on only 20 samples. Every Gaussian measurement has an SNR of 60dB and Φ = discrete cosine transformation. **In all cases, our method, DR-CG-Net given by the top red bar, outperforms the other approaches.**

forming comparably and both producing superior reconstructions, visually and by SSIM, to all six compared methods.

An advantage of DR-CG-Net, as highlighted by our CIFAR10 and CalTech101 results in Figure 3.4, Figure 3.5, Figure 3.6, Figure 3.7, Figure 3.8, Figure 3.9b, Figure 3.12, Figure 3.13, Figure 3.14, Figure 3.15 and Figure 3.16 is its ability to perform well on non-uniform data. Both CIFAR10 and CalTech101 contain many distinct classes of images from which we uniformly sample to create our training and testing datasets. In training with only 20 samples, few if any samples from each class will be seen by the network. Despite this, DR-CG-Net is able to recover high quality images in the test dataset that contains many more samples from both the classes present in the training data and from classes not present in the training data. That is, DR-CG-Net has the

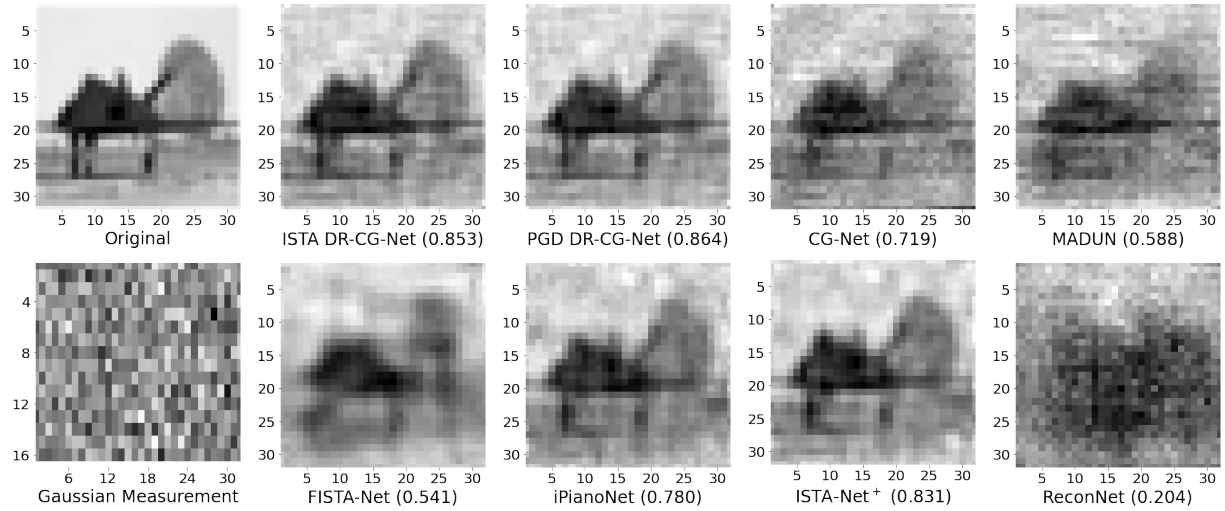


Figure 3.15: Image reconstructions (SSIM) using our DR-CG-Net and six comparative deep learning methods on a 32×32 plane image after training on only 20 samples. The sensing matrix, Ψ, is a Gaussian matrix at 0.5 sampling ratio, $\Phi =$ discrete cosine transformation, and each measurement has an SNR of 60dB. Our DR-CG-Net methods perform best visually and by SSIM.

ability to learn only from a handful of samples in a class and generalize well to unseen classes of samples.

We posit that the high performance of DR-CG-Net, especially in low-training scenarios, is due to the natural incorporation of the powerful CG prior through unrolling G-CG-LS. Specifically, the Tikhonov estimation layers and Hadamard product layer provide significant data-consistency structure by closely matching input measurements and measured estimated signals. Through this data consistency, a natural regularization for the DNN optimization is enforced by restricting the possible generated signals to a CG class. A thorough examination of the interplay between the iterative algorithm regularization and regularization for the DNN optimization is an intriguing subject of future work for unrolled DNN methods as a whole. Finally, the ability to learn the scale variable distribution, unlike in CG-Net where it is fixed as log-normal, provides DR-CG-Net with greater training capacity and flexibility. Thus, DR-CG-Net is able to significantly outperform competitive state-of-the-art deep learning-based methods in image reconstruction quality for training with small datasets. Furthermore, DR-CG-Net edges out CG-Net in test reconstruction performance in low-training scenarios, where CG-Net has shown exceptional performance, and significantly outperform CG-Net in test reconstruction performance in high-training scenarios.

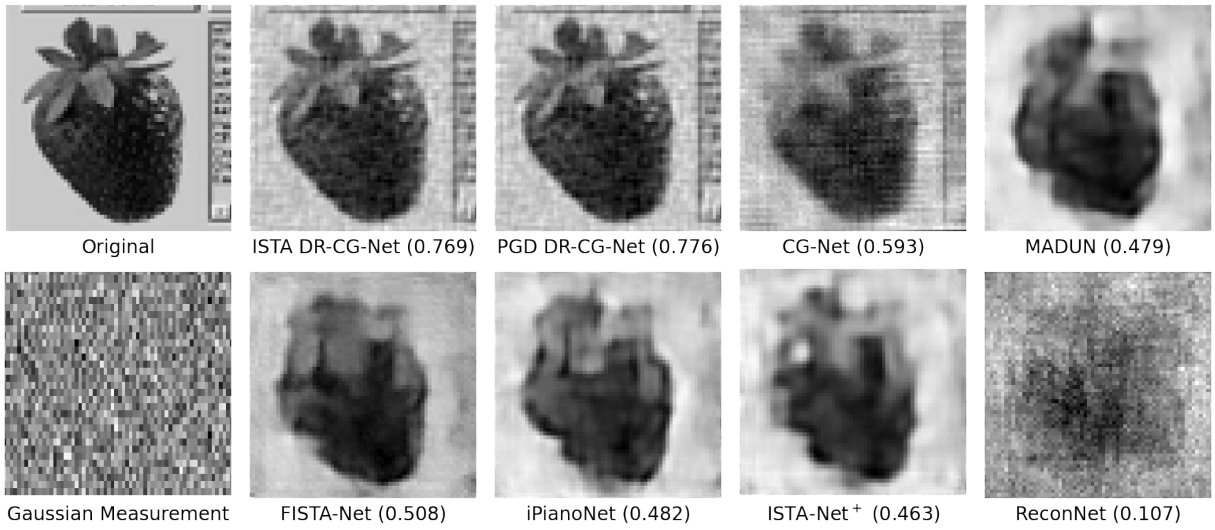


Figure 3.16: Image reconstructions (SSIM) using our DR-CG-Net and six comparative deep learning methods on a 64×64 strawberry image after training on only 20 samples. $\Psi =$ Gaussian matrix at 0.5 sampling ratio, $\Phi =$ discrete cosine transformation, and measurements have an SNR of 60dB. DR-CG-Net method performs best visually, particularly the leaf detail, and by SSIM.

Next, we remark that PGD DR-CG-Net and ISTA DR-CG-Net perform comparably in all measurement, measurement noise, and training dataset size scenarios considered for numerical experimentation in this section. This is perhaps unsurprising given that both methods are similar in implementation and that the learned subnetwork replaces a first-order optimization in both methods. As PGD DR-CG-Net has empirically shown a marginal image quality improvement over ISTA DR-CG-Net and reconstructs images slightly faster than ISTA DR-CG-Net, we recommended to default to PGD DR-CG-Net for any desired usage.

We reiterate that the results of Figure 3.6, Figure 3.7, Figure 3.9, Figure 3.10, Figure 3.11, Figure 3.12, Figure 3.13, Figure 3.14, Figure 3.15, and Figure 3.16 are for training on limited datasets, which is of interest in many applications in tomographic imaging. Consequently, the results presented are not representative of the performance of these methods with unlimited training data where our method is likely to be matched by the comparison methods as indicated by the results in Figure 3.4 and Figure 3.5.

Lastly, displayed in Appendix A are histogram density plots of SSIM values from 8000 reconstructed test CIFAR10 images from DR-CG-Net and seven prior art deep learning methods. The

solid vertical line on each histogram plot is the mean SSIM value and the dashed vertical line on each histogram plot is the median SSIM value. These histogram plots provide further insight into the performance of DR-CG-Net compared against the competitive prior-art deep-learning methods. In particular, we observe that the median SSIM value, for every method, is always greater than the mean SSIM value. Furthermore, the DR-CG-Net histograms visually display a narrower spread of SSIM values about the mean SSIM value than the seven compared methods. Finally, each SSIM histogram density appears to be roughly a truncated Gaussian distribution or a beta distribution where the covariance of the distribution decreases with greater angles in the Radon transform measurements and increases with more measurement noise and a greater amount of data samples used in training.

Refinement Block Study

Here, we investigate the impact that the refinement block $\mathcal{G}$ has on the quality of the reconstructed signals produced by DR-CG-Net. To this end, we empirically evaluate two alternative formulations for DR-CG-Net where $\mathcal{G}$ is removed before and after training as shown in Table 3.1. When $\mathcal{G}$ is removed prior to training, a fresh DR-CG-Net is trained using the setup of Section 3.3.5 with the C block now as the DR-CG-Net output. Instead, when $\mathcal{G}$ is removed after training, a trained DR-CG-Net from Section 3.3.5 is truncated to return the C block as output and no additional training is conducted.

In Table 3.1, the **T** and **F** in the $\mathcal{G}$ column indicates if the refinement block is used and b.t. or a.t. denote if $\mathcal{G}$ is removed before or after training. For both instances of removing the refinement block, the reconstruction quality of DR-CG-Net is still significant, which indicates that the unrolled G-CG-LS portion of DR-CG-Net is the core component in the superb image reconstructions produced by DR-CG-Net. Additionally, higher noise, i.e. 40dB measurement SNR, results in a larger discrepancy in the DR-CG-Net performance when the refinement block is or is not implemented as compared to the lower noise case. This suggests that the primary function for the refinement block is to denoise the estimate image from the unrolled G-CG-LS portion of DR-CG-Net.

Table 3.1: Refinement block study for ISTA DR-CG-Net. Displayed is the average SSIM ($\times 10^2$) and PSNR for CIFAR10 image reconstructions from a Radon transform, at several different amounts of uniformly spaced angles, with a set SNR. **T** and **F** indicate if the refinement block is or is not used, respectively. Further, b.t. and a.t. denote if the refinement block is removed before or after training, respectively.

$\mathcal{G}$	(Angles,SNR)	Training Dataset Size					
		20		100		500	
		SSIM ($\times 10^2$)	PSNR	SSIM ($\times 10^2$)	PSNR	SSIM ($\times 10^2$)	PSNR
F b.t.	(15,60)	0.901	28.22	0.922	28.96	0.937	29.89
F a.t.		0.900	27.97	0.912	28.35	0.917	28.76
T		0.919	28.59	0.932	29.45	0.940	30.07
F b.t.	(15,40)	0.852	25.98	0.882	26.94	0.897	27.71
F a.t.		0.857	26.10	0.862	26.24	0.869	26.30
T		0.891	26.86	0.903	27.47	0.909	27.87
F b.t.	(10,60)	0.821	25.13	0.840	25.60	0.855	26.12
F a.t.		0.820	25.03	0.829	25.30	0.839	25.63
T		0.828	25.17	0.844	25.69	0.860	26.29
F b.t.	(10,40)	0.775	23.84	0.794	24.34	0.826	25.15
F a.t.		0.775	23.87	0.787	24.12	0.789	24.27
T		0.811	24.49	0.827	25.00	0.839	25.44

Computational Time and Complexity

Table 3.2 lists the average computational time per image, in milliseconds, across 8000 test image reconstructions running on a 64-bit Intel(R) Xeon(R) CPU E5-2690. We see that DR-CG-Net is comparable in reconstruction time to MADUN, which is the comparison method providing the most competitive reconstruction results. While CG-Net was the second highest performing comparison method, in reconstructed signal quality, for low training scenarios, we see from Table 3.2 that DR-CG-Net is, by comparison, at least 20 times faster on average in reconstruction an image from Radon transform measurements.

The speed increase of DR-CG-Net over CG-Net is from three main sources: First, each update of $\boldsymbol{z}$ in CG-Net implements an eigendecomposition calculation, which is slower than each $\boldsymbol{z}$ update in DR-CG-Net requiring only matrix-vector products. Second, CG-Net was formed by unrolling a larger number of iterations, specifically $(K, J) = (20, 1)$, whereas DR-CG-Net only requires $(K, J) = (3, 4)$ for excellent performance. Third, the Woodbury matrix identity is employed in DR-CG-Net to accelerate the calculation of the Tikhonov solution.

Table 3.2: Average reconstruction time of 32×32 images from Radon transform measurements at 15 uniform angles.

Method	PGD DR-CG-Net	ISTA DR-CG-Net	CG-Net	MADUN
Time (ms)	22.0	52.7	765	26.0
Method	ReconNet	iRadonMAP	LEARN^{++}	LPD
Time (ms)	0.65	4.5	10.6	4.7
Method	FBPConvNet	iPiano-Net	FISTA-Net	ISTA-Net^{+}
Time (ms)	2.0	13.0	6.4	7.9

To explore the scalability of DR-CG-Net we consider the computational complexity of a forward pass through the network, in floating-point operations (FLOPs), with respect to the signal-of-interest size, n, and measurement size, m. We assume a batch size of one as the complexity simply grows proportionally to batch size. The initialization block, $\mathcal{Z}_0$, requires one matrix-vector product of size $n \times m$ by $m \times 1$ using $2mn$ FLOPs. Next, for $i \geq 1$, block $\mathcal{Z}_i$ requires one data fidelity gradient update, $r_k^{(j)}$, via (3.16) and D convolutions with 3×3 kernels and $f_1, \ldots, f_D$ filter channels. Each evaluation of the data fidelity gradient update, $r_k^{(j)}$, uses $4mn + 4n + m$ FLOPs as follows:

1. $\boldsymbol{u} \odot \boldsymbol{z}$: Multiplication of two length-n vectors requiring n FLOPs

2. $A_{\boldsymbol{u}}\boldsymbol{z} = A(\boldsymbol{u} \odot \boldsymbol{z})$: Multiplication of a $m \times n$ matrix by a $n \times 1$ vector requiring $2mn$ FLOPs (technically $m(2n-1)$ but this scales as $2mn$)

3. $A_{\boldsymbol{u}}\boldsymbol{z} - \boldsymbol{y}$: Subtraction of two length-m vectors requiring m FLOPs

4. $A^T(A_{\boldsymbol{u}}\boldsymbol{z} - \boldsymbol{y})$: Multiplication of a $n \times m$ matrix by a $m \times 1$ vector requiring $2mn$ FLOPs

5. $A_{\boldsymbol{u}}^T(A_{\boldsymbol{u}}\boldsymbol{z} - \boldsymbol{y}) = \boldsymbol{u} \odot (A^T(A_{\boldsymbol{u}}\boldsymbol{z} - \boldsymbol{y}))$: Multiplication of two length-n vectors requiring n FLOPs

6. $\eta A_{\boldsymbol{u}}^T(A_{\boldsymbol{u}}\boldsymbol{z} - \boldsymbol{y})$: Multiplying each entry of a length-n vector by a constant η requiring n FLOPs

7. $\boldsymbol{z} - \eta A_{\boldsymbol{u}}^T(A_{\boldsymbol{u}}\boldsymbol{z} - \boldsymbol{y})$: Subtraction of two length-n vectors requiring n FLOPs

Note that the number of FLOPs in evaluating a convolution using a kernel of size $k \times k$ with f output filter channels is $2 \times H \times W \times k \times k \times f$ where H and W denote the height and width of the output image at each filter channel. Thus, the D convolutions of block $\mathcal{Z}_i$, which each output images of size $\sqrt{n} \times \sqrt{n}$, use $18n \sum_{d=1}^{D} f_d$ FLOPs. Finally, block $\mathcal{Z}_i$, has an addition of two length-n vectors, those being $r_k^{(j)}$ and the vectorized output from the D convolutions, which uses n FLOPs. Therefore, in total, block $\mathcal{Z}_i$ uses $4mn + 18n \sum_{d=1}^{D} f_d + \mathcal{O}(n+m)$ FLOPs.

In the exact calculation of the Tikhonov solution via (3.5), first $I + A_{\boldsymbol{z}} P_u A_{\boldsymbol{z}}^T$ is calculated using $2mn^2 + 2m^2 n + mn + m$ FLOPs as follows:

1. $A_{\boldsymbol{z}}$: Multiplication of each entry of A by an entry of $\boldsymbol{z}$ requiring mn FLOPs
2. $A_{\boldsymbol{z}} P_u$: Multiplication of an $m \times n$ matrix by a $n \times n$ matrix (assuming naively that P_u is a full matrix) requiring $2mn^2$ FLOPs
3. $A_{\boldsymbol{z}} P_u A_{\boldsymbol{z}}^T$: Multiplication of a $m \times n$ matrix by a $n \times m$ matrix requiring $2m^2 n$ FLOPs
4. $I + A_{\boldsymbol{z}} P_u A_{\boldsymbol{z}}^T$: Addition of m diagonal entries requiring m FLOPs

Second, $(I + A_{\boldsymbol{z}} P_u A_{\boldsymbol{z}}^T)^{-1} \boldsymbol{y}$ is calculated by solving a $m \times m$ system of equations that naively uses $\frac{2}{3}m^3 + \mathcal{O}(m^2)$ FLOPs. Third, $A_{\boldsymbol{z}}^T (I + A_{\boldsymbol{z}} P_u A_{\boldsymbol{z}}^T)^{-1} \boldsymbol{y}$ is calculated by the multiplication of a $n \times m$ matrix with a $m \times 1$ vector using $2mn$ FLOPs. Finally, $P_u A_{\boldsymbol{z}}^T (I + A_{\boldsymbol{z}} P_u A_{\boldsymbol{z}}^T)^{-1} \boldsymbol{y}$ is computed by the multiplication of a $n \times n$ matrix with a $n \times 1$ vector using $2n^2$ FLOPs. Thus, a total of $\frac{2}{3}m^3 + 2mn^2 + \mathcal{O}(m^2 n + n^2)$ FLOPs are used in calculating (3.5). Therefore, when we calculate the Tikhonov solution exactly, the total FLOPs for DR-CG-Net scales as

$$\mathcal{O}\left(KJ\left[m + \textstyle\sum_{d=1}^{D} f_d\right] n + Km^3 + Kmn^2\right).$$

When we consider, instead, the approximation of the Tikhonov solution by J_u NAGD steps, the first two evaluations of r_u via (3.7) are computed, using $2n^2 + 4mn + 5n + m$ FLOPs each as follows:

1. $\boldsymbol{u} \odot \boldsymbol{z}$: Multiplication of two length-n vectors requiring n FLOPs

2. $A_{\boldsymbol{z}}\boldsymbol{u} = A(\boldsymbol{u} \odot \boldsymbol{z})$: Multiplication of a $m \times n$ matrix by a $n \times 1$ vector requiring $2mn$ FLOPs

3. $A_{\boldsymbol{z}}\boldsymbol{u} - \boldsymbol{y}$: Subtraction of two length-m vectors required m FLOPs

4. $A^T(A_{\boldsymbol{z}}\boldsymbol{u} - \boldsymbol{y})$: Multiplication of a $n \times m$ matrix by a $m \times 1$ vector requiring $2mn$ FLOPs

5. $A_{\boldsymbol{z}}^T(A_{\boldsymbol{z}}\boldsymbol{u} - \boldsymbol{y}) = \boldsymbol{z} \odot (A^T(A_{\boldsymbol{z}}\boldsymbol{u} - \boldsymbol{y}))$: Multiplication of two length-n vectors requiring n FLOPs

6. $P_u^{-1}\boldsymbol{u}$: Multiplication of a $n \times n$ matrix by a $n \times 1$ vector requiring $2n^2$ FLOPs (assuming P_u^{-1} is a full matrix and is known, which, in our applications, it is known as we will learn P_u^{-1} in DR-CG-Net directly when we approximate the Tikhonov solution by NAGD steps)

7. $A_{\boldsymbol{z}}^T(A_{\boldsymbol{z}}\boldsymbol{u} - \boldsymbol{y}) + P_u^{-1}\boldsymbol{u}$: Addition of two length-n vectors requiring n FLOPs

8. $\eta(A_{\boldsymbol{z}}^T(A_{\boldsymbol{z}}\boldsymbol{u} - \boldsymbol{y}) + P_u^{-1}\boldsymbol{u})$: Multiply each entry of a length-n vector by a constant η requiring n FLOPs

9. $u - \eta(A_{\boldsymbol{z}}^T(A_{\boldsymbol{z}}u - y) + P_u^{-1}u)$: Subtraction of two length-n vectors requiring n FLOPs

Second, a length-n vector subtraction, scalar multiplication, and length-n vector addition are calculated using, collectively, $3n$ FLOPs. Therefore, when we approximate the Tikhonov solution with NAGD steps, the total FLOPs for DR-CG-Net scales as

$$\mathcal{O}\left(KJ\left[m + \sum_{d=1}^{D} f_d\right] n + KJ_u n^2\right).$$

Thus, the computational complexity of DR-CG-Net is reduced by a power of m for the replacement of the exact Tikhonov solution with NAGD steps. Still, both versions have non-linear polynomial growth in computational complexity and thus, further computational reductions are required for scaling DR-CG-Net to signals of significantly higher dimension.

3.3.6 Network Parameters

Table 3.3 displays the number of parameters, i.e. unknowns, for DR-CG-Net and all ten compared deep learning-based methods for linear inverse problems. Note that the ratio of unknowns to data point measurements can be obtained by dividing each entry of Table 3.3 by the number of data points. For example, when training with 20 CIFAR10 images and 15 angle Radon transforms we use $20 \times (\text{image size} + \text{Radon transform size}) = 20 \times (1024 + 690) = 3.428 \times 10^4$ points of data, giving a ratio of unknowns to measurements of $7.26 \times 10^5/(3.428 \times 10^4) \approx 21.2$ for DR-CG-Net. Instead, when training with 20 LoDoPaB-CT images and 60 angle Radon transforms, we have a ratio of unknowns to measurements of $7.26 \times 10^5/(5.46 \times 10^5) \approx 1.33$ for DR-CG-Net. These ratios are on par with FBPConvNet and ReconNet that have a similar number of parameters to DR-CG-Net.

While the ratio of unknowns to measurements is high in low-training scenarios, our empirical results demonstrate that DR-CG-Net handles this situation well. In particular, we note that despite DR-CG-Net having a significantly greater ratio of unknowns to measurements than CG-Net, DR-CG-Net still outperforms CG-Net when both are trained on small training datasets.

Table 3.3: Parameter count for our DR-CG-Nets and ten comparison deep learning methods when each method is reconstructing a 32×32 image from Radon transform measurements at 15 uniformly spaced angles.

Method	Parameters ($\times 10^5$)	Method	Parameters ($\times 10^5$)
ISTA DR-CG-Net	7.26	PGD DR-CG-Net	7.26
CG-Net	1.17	ISTA-Net$^+$	3.37
LPD	2.53	MADUN	29.7
LEARN^{++}	12.0	FISTA-Net	0.75
iRadonMAP	8.33	iPiano-Net	19.3
FBPConvNet	7.09	ReconNet	7.30

3.4 Deep Scale Regularized Compound Gaussian Network

In this section, we discuss the deep scale regularized compound Gaussian network (DSR-CG-Net), which is a variation of DR-CG-Net. The deep scale regularized compound Gaussian network, originally presented in our IEEE Asilomar conference proceedings [76], was proposed before the DR-CG-Net method and is a hybrid between the CG-Net method and the DR-CG-Net method. Specifically, the scale regularization is not learned in its entirety, as in DR-CG-Net, but instead is a learned function of a Gaussian random variable.

A critical component of the compound Gaussian prior is the choice of distribution, $p(\boldsymbol{z})$, for the scale variable $\boldsymbol{z}$. In CG-LS and CG-Net the scale variable distribution is captured by the nonlinearity h (with inverse nonlinearity f), as given in (1.8) and (2.2), and choices of regularization, namely the Euclidean norm, on the vector $f(\boldsymbol{z})$. As images have wide variability, different scale variable distributions, $p_i(\boldsymbol{z})$ and $p_j(\boldsymbol{z})$, may statistically represent the sparsity coefficients of two distinct classes of images, $\mathcal{I}_i$ and $\mathcal{I}_j$. That is, it may be useful to make CG-Net adaptable to the choice of scale variable distribution by providing it the capability to learn either the inverse nonlinearity f or regularization on the $\boldsymbol{z}$ variable.

As in Chapter 2, we use algorithm unrolling to incorporate the CG prior, from an iterative algorithm, into the machine learning framework where, now, we include the learning of the CG scale variable regularization. Thus, we first require an iterative algorithm to apply algorithm unrolling to. Similar to (2.2) we consider a regularized least squares minimizer

$$\underset{[\boldsymbol{z},\boldsymbol{u}]}{\arg\min}\ ||\boldsymbol{y} - A(\boldsymbol{z} \odot \boldsymbol{u})||_2^2 + \lambda||\boldsymbol{u}||_2^2 + \mathcal{R}(f(\boldsymbol{z})) \tag{3.18}$$

where the regularization $\lambda||\boldsymbol{u}||_2^2$ enforces normality of $\boldsymbol{u}$ and $\mathcal{R} : \mathbb{R}^n \to \mathbb{R}$ is an implicit regularization function with gradient $r = \nabla\mathcal{R}$ and Hessian $H_{\mathcal{R}} = \nabla r$. In unrolling an iterative algorithm to solve (3.18), the gradient and Hessian will be learned by the deep neural network.

3.4.1 Iterative Algorithm Implementation

Due to the implicit joint estimation in (3.18), we optimize by block coordinate descent, which on iteration k is given by

$$\boldsymbol{z}_k = \arg\min_{\boldsymbol{z}} ||\boldsymbol{y} - A_{\boldsymbol{u}_{k-1}}\boldsymbol{z}||_2^2 + \mathcal{R}(h^{-1}(\boldsymbol{z})), \tag{3.19}$$

$$\boldsymbol{u}_k = \arg\min_{\boldsymbol{u}} ||\boldsymbol{y} - A_{\boldsymbol{z}_k}\boldsymbol{u}||_2^2 + \lambda||\boldsymbol{u}||_2^2 \tag{3.20}$$

Recall $A_{\boldsymbol{w}} = AD(\boldsymbol{w})$ and $D(\boldsymbol{w})$ is a diagonal matrix formed from placing the entries of $\boldsymbol{w}$ along the diagonal. Note (3.20) is a Tikhonov regularization problem with the solution

$$\boldsymbol{u}_k = (A_{\boldsymbol{z}_k}^T A_{\boldsymbol{z}_k} + \lambda I)^{-1} A_{\boldsymbol{z}_k}^T \boldsymbol{y}. \tag{3.21}$$

To solve (3.19), we use a steepest descent method, which on step j of iteration k is given by

$$\boldsymbol{z}_k^j = \boldsymbol{z}_k^{j-1} - \eta_k^{(j)} \boldsymbol{d}(\boldsymbol{z}_k^{j-1}; \boldsymbol{u}_{k-1}, \boldsymbol{y}) \tag{3.22}$$

where $\boldsymbol{d}$ is a function giving a steepest descent search direction and $\eta_k^{(j)}$ is a step size chosen by a backtracking line search. We specifically consider two steepest descent approaches: a gradient descent approach, in which $\boldsymbol{d}$ is a gradient vector, and a Newton descent approach, in which $\boldsymbol{d}$ is an inverse Hessian times a gradient vector. Algorithm 4 details pseudocode for the generalized scale compound Gaussian least squares (GS-CG-LS) estimate of $\boldsymbol{u}$ and $\boldsymbol{z}$.

Note, the gradient of the cost function in (3.19) is

$$\boldsymbol{v}_g(\boldsymbol{z}; \boldsymbol{u}_{k-1}, \boldsymbol{y}) = 2A_{\boldsymbol{u}_{k-1}}^T (A_{\boldsymbol{u}_{k-1}}\boldsymbol{z} - \boldsymbol{y}) + D(f'(\boldsymbol{z}))r(f(\boldsymbol{z})) \tag{3.23}$$

and thus, the Hessian, $\mathcal{H}(\boldsymbol{z}; \boldsymbol{u}_{k-1}, \boldsymbol{y}) \equiv \boldsymbol{J}_{\boldsymbol{z}}(\boldsymbol{v}_g(\boldsymbol{z}; \boldsymbol{u}_{k-1}, \boldsymbol{y}))$, is

$$\mathcal{H}(\boldsymbol{z}; \boldsymbol{u}_{k-1}, \boldsymbol{y}) = 2A_{\boldsymbol{u}_{k-1}}^T A_{\boldsymbol{u}_{k-1}} + P(\boldsymbol{z}) \tag{3.24}$$

where, for $H_{\mathcal{R}}$ is the Hessian of $\mathcal{R}$,

$$P(\boldsymbol{z}) = D(f''(\boldsymbol{z}))D(r(f(\boldsymbol{z}))) + D(f'(\boldsymbol{z}))H_{\mathcal{R}}(f(\boldsymbol{z}))D(f'(\boldsymbol{z})).$$

For step j of iteration k, we take the steepest descent direction in (3.22) to be

$$\boldsymbol{d}(\boldsymbol{z}_k^{j-1}; \boldsymbol{u}_{k-1}, \boldsymbol{y}) = \begin{cases} -\boldsymbol{v}_g(\boldsymbol{z}_k^{j-1}; \boldsymbol{u}_{k-1}, \boldsymbol{y}) & \text{gradient descent} \\ -(\mathcal{H}(\boldsymbol{z}_k^{j-1}; \boldsymbol{u}_{k-1}, \boldsymbol{y}))^{-1}\boldsymbol{v}_g(\boldsymbol{z}_k^{j-1}; \boldsymbol{u}_{k-1}, \boldsymbol{y}) & \text{Newton descent.} \end{cases}$$

For the initial estimate of $\boldsymbol{z}$ we take $\boldsymbol{z}_0 = \mathcal{P}_{a,b}(A^T\boldsymbol{y})$ where, for $0 \leq a < b$

$$\mathcal{P}_{a,b}(x) = a + \text{ReLU}(x - a) - \text{ReLU}(x - b)$$

and $\mathcal{P}_{a,b}$ is applied componentwise to $A^T\boldsymbol{y}$. We remark that $\mathcal{P}_{a,b}$, which we call a modified ReLU activation function, is a projection operator onto the interval $[a, b]$, which eliminates negative val-

Algorithm 4 Generalized Scale Compound Gaussian Least Squares (GS-CG-LS)

Input: Measurement $\boldsymbol{y}$, measurement operator A, number of iterations K, number of steepest descent steps J, regularization parameter λ, operator $\mathcal{P}_{a,b}$, and regularization function $\mathcal{R}$

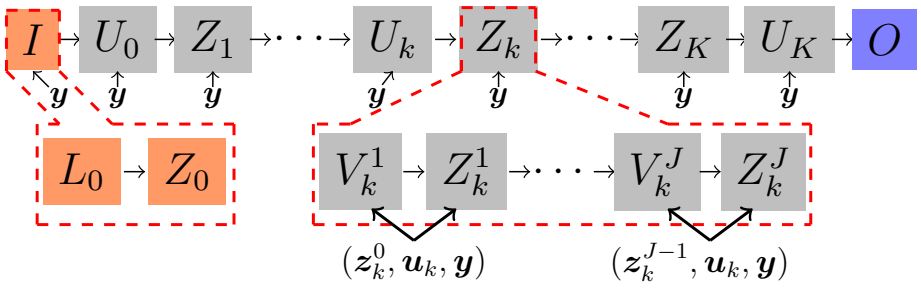
Calculate initial $\boldsymbol{z}_0 = \mathcal{P}_{a,b}(A^T\boldsymbol{y})$ and $\boldsymbol{u}_0 = (A_{\boldsymbol{z}_0}^T A_{\boldsymbol{z}_0} + \lambda I)^{-1} A_{\boldsymbol{z}_0}^T \boldsymbol{y}$
for $k \in \{1, 2, \ldots, K\}$ **do**
    $\boldsymbol{z}$ ESTIMATION:
    $\boldsymbol{z}_k^0 = \boldsymbol{z}_{k-1}$
    for $j \in \{1, 2, \ldots, J\}$ **do**
        Compute step size η_k^{j-1} (Armijo line search)
        Compute descent direction $\boldsymbol{d}(\boldsymbol{z}_k^{j-1}; \boldsymbol{u}_{k-1}, \boldsymbol{y})$
            ▷ ($\boldsymbol{d} = \boldsymbol{v}_g$ for gradient descent and
            ▷ $\boldsymbol{d} = \mathcal{H}^{-1}\boldsymbol{v}_g$ for Newton descent)
        $\boldsymbol{z}_k^j = \boldsymbol{z}_k^{j-1} - \eta_k^{(j)}\boldsymbol{d}(\boldsymbol{z}_k^{j-1}; \boldsymbol{u}_{k-1}, \boldsymbol{y})$
    end for
        $\boldsymbol{z}_k = \boldsymbol{z}_k^J$
    $\boldsymbol{u}$ ESTIMATION:
    $\boldsymbol{u}_k = (A_{\boldsymbol{z}_k}^T A_{\boldsymbol{z}_k} + \lambda I)^{-1} A_{\boldsymbol{z}_k}^T \boldsymbol{y}$
end for

Return: $\boldsymbol{c}^* = \boldsymbol{z}_K \odot \boldsymbol{u}_K$.

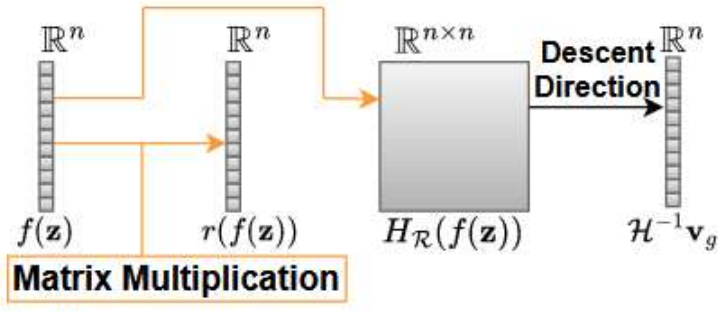
ues, as $\boldsymbol{z}$ should have positive components, and limits the maximum values in the initial $\boldsymbol{z}$ estimate. Then the initial estimate of $\boldsymbol{u}$ is given by (3.21).

3.4.2 Deep Scale Regularization Compound Gaussian Network Structure

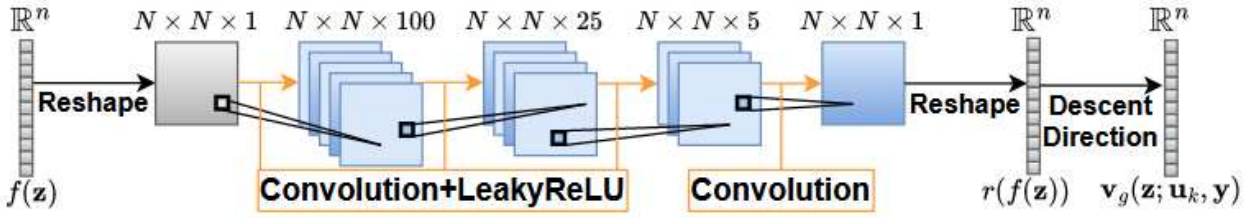
We apply algorithm unrolling to GS-CG-LS in Algorithm 4 to create a deep neural network we name deep scale regularization compound Gaussian network (DSR-CG-Net), which has the structure shown in Figure 3.17. Unlike the GS-CG-LS iterative algorithm, a regularization func-



(a) End-to-end structure of DSR-CG-Net.



(b) V_k^j subnetwork for approximating the steepest descent direction in Newton DSR-CG-Net.



(c) V_k^j subnetwork for approximating the steepest descent direction in gradient DSR-CG-Net.

Figure 3.17: Network structure for DSR-CG-Net, the unfolded deep neural network of GS-CG-LS in Algorithm 4.

tion $\mathcal{R}$ need not be specified and instead is learned, via the gradient or Hessian of $\mathcal{R}$, through a subnetwork of layers in DSR-CG-Net.

Two principal operations, $\boldsymbol{f}_k : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ and $\boldsymbol{g}_k^j : \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$, define the DSR-CG-Net layers. Using (3.21), each $\boldsymbol{f}_k$ is defined as

$$\boldsymbol{f}_k(\boldsymbol{z}, \boldsymbol{y}) = (A_{\boldsymbol{z}}^T A_{\boldsymbol{z}} + \lambda_k I)^{-1} A_{\boldsymbol{z}}^T \boldsymbol{y}, \tag{3.25}$$

for $\lambda_k > 0$, and corresponds to updating $\boldsymbol{u}$ on iteration k in Algorithm 4. Next, let $\boldsymbol{d}_k^j$ be a steepest descent direction, i.e. $\boldsymbol{d}_k^j = \boldsymbol{v}_g$ or $\boldsymbol{d}_k^j = \mathcal{H}^{-1}\boldsymbol{v}_g$, then, using (3.22), each $\boldsymbol{g}_k^j$ is defined as

$$\boldsymbol{g}_k^j(\boldsymbol{z}, \boldsymbol{u}, \boldsymbol{y}; \boldsymbol{d}_k^j) = \mathcal{P}_{a,b}\left(\boldsymbol{z} - \eta_k^{(j)} \boldsymbol{d}_k^j(\boldsymbol{z}; \boldsymbol{u}, \boldsymbol{y})\right), \tag{3.26}$$

and corresponds to updating $\boldsymbol{z}$ on the jth steepest descent step of iteration k in Algorithm 4. Note, in Algorithm 4 the step size $\eta_k^{(j)}$ is found by a backtracking line search and serves as a guarantee that $\boldsymbol{z}_k^j$ maintains positive components and that the steepest descent step is not too large. This is not guaranteed by the step size $\eta_k^{(j)}$ learned by DSR-CG-Net during training. Thus, as given in (3.26), we additionally apply the modified ReLU activation function $\mathcal{P}_{a,b}$ after each steepest descent step to guarantee the step is not too large and each $\boldsymbol{z}$ estimate maintains positive components.

We now detail the DSR-CG-Net layers and layer blocks:

I The input block I consists of the input layer $L_0 = \boldsymbol{y}$ and an initialization layer $Z_0 = \mathcal{P}_{a,b}(A^T L_0)$ giving the initial $\boldsymbol{z}$ estimate corresponding to line 1 in Algorithm 4.

U_k The kth U block contains a single layer giving the kth estimate of $\boldsymbol{u}$ from line 12 in Algorithm 4. That is, $U_k = \boldsymbol{f}_k(Z_{k-1}, L_0)$.

Z_k The kth Z block contains J updates of $\boldsymbol{z}$ from lines 3-10 of Algorithm 4. Update j contains the following blocks:

V_k^j A subnetwork of layers that approximate the gradient or Hessian of regularization function $\mathcal{R}$ and subsequently calculates the jth steepest descent direction on iteration k given by $\boldsymbol{d}_k^j(Z_k^{j-1}; U_{k-1}, L_0)$. Block V_k^j corresponds to line 7 in Algorithm 4.

Z_k^j A single layer containing the jth steepest descent step update of $\boldsymbol{z}$ on iteration k from line 8 of Algorithm 4. That is $Z_k^j = \boldsymbol{g}_k^j(Z_k^{j-1}, U_{k-1}, L_0; V_k^j)$. Note, for simplicity of notation we define $Z_k^0 \equiv Z_{k-1}^J$.

O The output block consists of the output layer that is the Hadamard product of the final U and Z layers. That is, $O = Z_K^J \odot U_K$ and gives the estimated wavelet coefficients.

Assume that each subnetwork V_k^j uses L layers. Then DSR-CG-Net contains K estimation layers for $\boldsymbol{u}$, KJL layers to approximate the steepest descent directions, KJ steepest descent $\boldsymbol{z}$ updates, one input, one output, and one initialization layer. So, in total, DSR-CG-Net contains $K(J(L+1)+1)+3$ layers.

3.4.3 DSR-CG-Net Parameters

Each U_k layer is parameterized by a regularization scalar $\lambda_k > 0$ and each Z_k^j layer is parameterized by a steepest descent step size $\eta_k^{(j)}$. The majority of the DSR-CG-Net parameters are contained in the subnetworks V_k^j that depend on whether a gradient or Newton descent method is implemented.

First, we consider applying algorithm unrolling when gradient descent is implemented in Algorithm 4 and label the resulting DNN as gradient DSR-CG-Net (gDSR-CG-Net). In gDSR-CG-Net the steepest descent directions $\boldsymbol{d}_k^j$ are taken to be $\boldsymbol{v}_g$, given in (3.23), and an approximation of $r = \nabla\mathcal{R}$ is required. Each subnetwork V_k^j for gDSR-CG-Net is shown in Figure 3.17c and consists of four convolutional layers using a convolutional kernel of size 5×5 with 100, 25, 5, and 1 filter channels, respectively. All convolutions are performed using a unit stride with zero padding applied so that the input and output from each filter channel are the same size. Note that the number of filter channels was chosen empirically, with a gradual reduction over the convolutional layers to a single final filter channel, so that the output dimension from and input dimension to the four convolutions are equivalent. A bias matrix is added to the output from each convolutional filter channel, which, excluding the final convolutional layer, is passed componentwise through a Leaky ReLU activation function defined, for slope coefficient $\beta \geq 0$, as

$$\text{LeakyReLU}(x) = \text{ReLU}(x) - \beta\text{ReLU}(-x).$$

Therefore, each subnetwork, V_k^j, contains the convolutional kernels and biases as parameters.

We remark that V_k^j approximates the required steepest descent direction $\boldsymbol{d}_k^j(Z_k^{j-1}; U_{k-1}, L_0)$ as follows: the vector $f(Z_k^{j-1})$ is reshaped into a matrix and sent through the four convolutional layers, the output from the final convolution is reshaped into a vector that approximates $r(f(Z_k^{j-1}))$, and finally $\boldsymbol{d}_k^j(Z_k^{j-1}; U_{k-1}, L_0) = \boldsymbol{v}_g(Z_k^{j-1}; U_{k-1}, L_0)$ is calculated using (3.23) with the approximated $r(f(Z_k^{j-1}))$.

Second, we consider applying algorithm unfolding when Newton descent is implemented in Algorithm 4 and label the resulting DNN as Newton DSR-CG-Net (nDSR-CG-Net). In nDSR-CG-Net the steepest descent directions $\boldsymbol{d}_k^j$ are taken to be $\mathcal{H}^{-1}\boldsymbol{v}_g$, using (3.23) and (3.24), and an approximation of both $r = \nabla\mathcal{R}$ and $H_\mathcal{R} = \nabla r$ are required. Thus, we require a collection of analytically differentiable network layers to approximate r and use the same layer weights to also approximate $H_\mathcal{R}$. As it is difficult to derive analytical gradients of convolutional layers, we instead

consider a fully connected layer for r given by

$$r(\boldsymbol{\zeta}) = \sigma(W\boldsymbol{\zeta} + \boldsymbol{b}) \tag{3.27}$$

for $\boldsymbol{\zeta} \in \mathbb{R}^n, W \in \mathbb{R}^{n\times n}, \boldsymbol{b} \in \mathbb{R}^n$, and σ, a componentwise activation function. Then the Hessian, $H_{\mathcal{R}}$, is given by

$$H_{\mathcal{R}}(\boldsymbol{\zeta}) \equiv \nabla r(\boldsymbol{\zeta}) = W^T D\{\sigma'(W\boldsymbol{\zeta} + \boldsymbol{b})\}. \tag{3.28}$$

Therefore, each subnetwork for nDSR-CG-Net, shown in Figure 3.17b, contains a weight matrix W_k^j and bias $\boldsymbol{b}_k^j$ as parameters that are used as W and $\boldsymbol{b}$ in (3.27) and (3.28).

We remark that V_k^j approximates the required steepest descent direction $\boldsymbol{d}_k^j(Z_k^{j-1}; U_{k-1}, L_0)$ as follows: calculate $r(f(Z_k^{j-1})) = \sigma\left(W_k^j f(Z_k^{j-1}) + \boldsymbol{b}_k^j\right)$ as given by (3.27), similarly calculate $H_{\mathcal{R}}(f(Z_k^{j-1}))$ using (3.28), and finally use $r(f(Z_k^{j-1}))$ and $H_{\mathcal{R}}(f(Z_k^{j-1}))$ in (3.23) and (3.24) to calculate $\boldsymbol{d}_k^j(Z_k^{j-1}; U_{k-1}, L_0) = \left(\mathcal{H}(Z_k^{j-1}; U_{k-1}, L_0)\right)^{-1} \boldsymbol{v}_g(Z_k^{j-1}; U_{k-1}, L_0)$. As $\mathcal{H}$ must be invertible, in implementation we perform an eigendecomposition on each $\mathcal{H}(Z_k^{j-1}; U_{k-1}, L_0)$ and, for some real number $\epsilon > 0$, set all eigenvalues with magnitude less than ϵ to ϵ.

Lastly, we let $f(z) \equiv h^{-1}(z) = 2\alpha \ln(z)$ and thus each subnetwork V_k^j is also parameterized by a constant α_k^j defining the inverse nonlinearity $f(z)$. Note that we assume each subnetwork has an independent set of parameters, e.g. different convolutional kernels and biases for each V_k^j in gDSR-CG-Net, for generality and ease of implementation. Alternatively, the same set of parameters can define all subnetworks V_k^j, which would decrease the total parameters in DSR-CG-Net significantly and likely result in less required training while also reducing the amount that could be learned by the network.

3.4.4 DSR-CG-Net Loss Function

To learn the DSR-CG-Net parameters, we use a loss function based on the SSIM quality metric [49] given by

$$\mathcal{L}_{\mathcal{B}}(\boldsymbol{\Theta}) = \frac{1}{|\mathcal{B}|} \sum_{(\overline{\boldsymbol{y}}_i, \overline{\boldsymbol{c}}_i) \in \mathcal{B}} (1 - \text{SSIM}(\Phi \widehat{\boldsymbol{c}}(\overline{\boldsymbol{y}}_i; \boldsymbol{\Theta}), \Phi \overline{\boldsymbol{c}}_i))$$

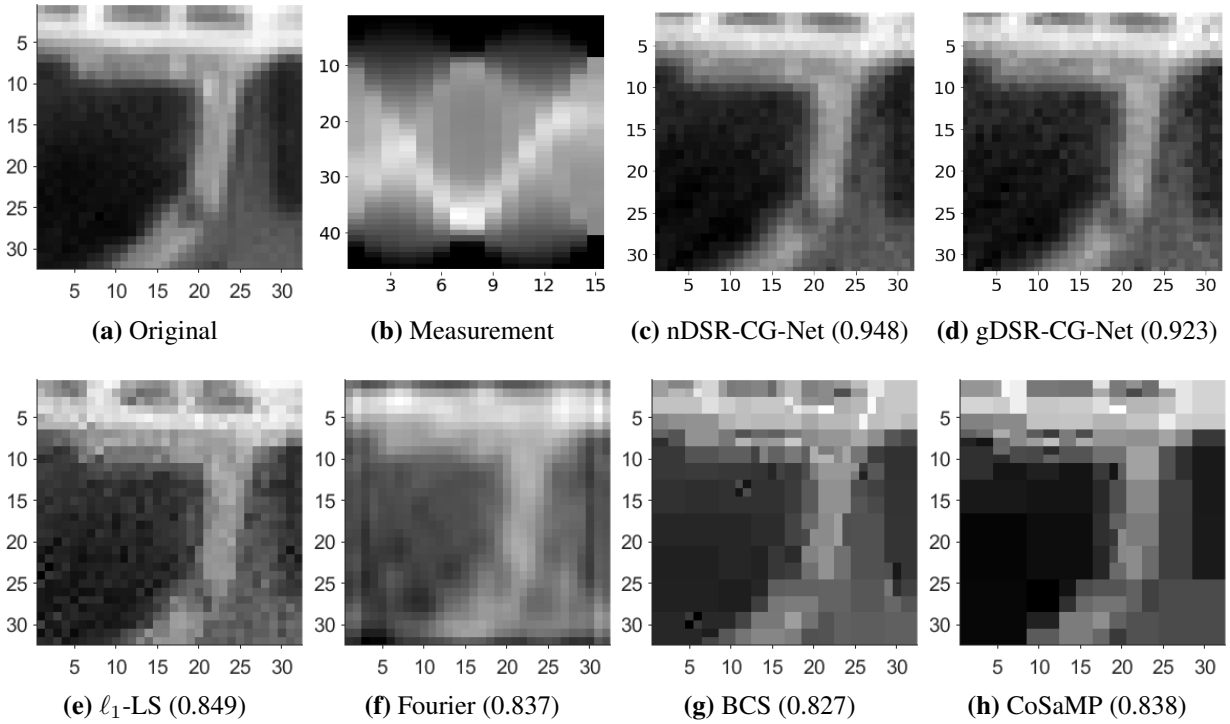
for $\mathcal{B} \subset \mathcal{D}$, a batch of measurement-and-wavelet-coefficient data pairs, $(\overline{\boldsymbol{y}}_i, \overline{\boldsymbol{c}}_i)$, and $\widehat{\boldsymbol{c}}(\overline{\boldsymbol{y}}_i; \boldsymbol{\Theta})$, the wavelet coefficients estimated from DSR-CG-Net. Note that Φ, which is the change-of-basis matrix, is fixed as a biorthogonal wavelet dictionary such that $\Phi \widehat{\boldsymbol{c}}(\overline{\boldsymbol{y}}_i; \boldsymbol{\Theta})$ and $\Phi \overline{\boldsymbol{c}}_i$ denoted the estimated and true image, respectively. We optimize $\mathcal{L}_{\mathcal{B}}(\boldsymbol{\Theta})$ using adaptive moment estimation (Adam) [73], which is a stochastic gradient-based optimizer. Other common optimization methods include: stochastic gradient descent, RMSprop, and Adadelta [73]. The gradient $\nabla_{\boldsymbol{\Theta}} \mathcal{L}_{\mathcal{B}}$ used in Adam is calculated via backpropagation through the network, which we implement using automatic differentiation [74] in TensorFlow.

3.4.5 DSR-CG-Net Numerical Results

We run experimental tests using the CIFAR10 image dataset [69]. Each image is converted from three-channel RGB to single-channel grayscale and the CIFAR10 classification label is disregarded. In Chapter 2, we considered $\mathcal{R}$, in (3.18) and Algorithm 4, to be the ℓ_2 norm for which (3.19) is convex when each entry of $\boldsymbol{z}$ is contained within the interval $(0, e)$. Additionally, it can be guaranteed that each $\boldsymbol{z}$ estimate lies within $(0, e)$ by scaling down the input measurements. Thus, we scale down each image, I, by a factor of e^{-4}/I_{max}, chosen emperically, where I_{max} is the max pixel value of I. We apply a Radon transform, at 15 and 6 uniformly spaced angles, to each image and add white noise to produce measurements $\boldsymbol{y}$ at a set signal-to-noise ratio (SNR). Finally, we apply a biorthogonal wavelet transform to each image to obtain the sparse coefficients $\boldsymbol{c}$. Thus, the training datasets, $\mathcal{D}_{15}$ and $\mathcal{D}_6$, consists of Radon transform measurements, at 15 or 6 angles respectively, and wavelet coefficient data pairs $(\overline{\boldsymbol{y}}_i, \overline{\boldsymbol{c}}_i)$ corresponding to an image I_i.

For gDSR-CG-Net and nDSR-CG-Net, we choose $(K, J) = (10, 2)$ and $(K, J) = (10, 1)$, respectively. Every Leaky ReLU activation function, in gDSR-CG-Net, utilizes a slope coefficient of 0.1 and we take σ in (3.27), for nDSR-CG-Net, to be the identity operation. We take $\mathcal{P}_{a,b} = \mathcal{P}_{10^{-2},e^4}$ for gDSR-CG-Net and $\mathcal{P}_{a,b} = \mathcal{P}_{1,e}$ for nDSR-CG-Net. We initialize all $\lambda_k = 0.3$, all $\eta_k^{(j)} = 0.5$, all biases as zeros, all fully connected weight matrices for nDSR-CG-Net as $W_k^j = 4I$, and each convolutional kernel according to the Glorot uniform initialization [84]. We train both DSR-CG-Nets on 4000 data pairs, i.e. $|\mathcal{D}_{15}| = |\mathcal{D}_6| = 4000$, for 20 epochs with a learning rate of 10^{-3}. Note that all of these hyperparameter configurations were chosen on an empirical basis.

An example test reconstruction of a 32×32 Barbara image snippet is shown in Figure 3.18. After training both gDSR-CG-Net and nDSR-CG-Net on $\mathcal{D}_{15}$, a Radon transform at 15 uniformly spaced angles of the Barbara snippet with an SNR of 60dB is provided to each network. We



(a) Original **(b)** Measurement **(c)** nDSR-CG-Net (0.948) **(d)** gDSR-CG-Net (0.923)

(e) ℓ_1-LS (0.849) **(f)** Fourier (0.837) **(g)** BCS (0.827) **(h)** CoSaMP (0.838)

Figure 3.18: Test image reconstructions using Newton DSR-CG-Net, gradient DSR-CG-Net, ℓ_1-LS, filtered backprojection, Bayesian Compressive Sensing, and CoSaMP. Each method operates on a vectorized (3.18b) which is the Radon transform of the original image, (3.18a), at 15 uniformly spaced angles with an SNR 60dB. Newton DSR-CG-Net outperforms gradient DSR-CG-Net and both outperform all compared methods.

see in Figure 3.18c and 3.18d noisy artifacts in the estimated images, but the main structure is easily identifiable. Figure 3.18 also contains the Barbara image reconstruction for four iterative algorithms: ℓ_1-least squares (ℓ_1-LS) [6], filtered backprojection (FBP) [71], BCS [3], and CoSaMP [4]. We see by visual inspection and SSIM value that nDSR-CG-Net outperforms gDSR-CG-Net and both produce improved reconstructions over the compared iterative algorithms.

Table 3.4 lists the average values and 99% confidence intervals for SSIM, PSNR, and reconstruction time over 8000 test image reconstructions from 15 uniformly spaced angle Radon transforms with 60dB SNR. Larger SSIM and PSNR values correspond to image reconstructions that are visually closer to the original image. From Table 3.4 we observe that DSR-CG-Net appreciably outperforms each of the four compared iterative algorithms in SSIM and PSNR image quality while being on par to ℓ_1-LS, BCS, and CoSaMP in reconstruction time. Additionally, nDSR-CG-Net provides, on average, a 0.43 SSIM and 0.17 PSNR increase in reconstructed image quality at the cost of 0.233 seconds per reconstructed image over gDSR-CG-Net. We remark that many of the test image reconstructions from gDSR-CG-Net display noisy artifacts similar to Figure 3.18d, which are less pronounced in the image estimations from nDSR-CG-Net.

Benefits of gDSR-CG-Net and nDSR-CG-Net include the fast reconstruction time, independence from image-specific parameter tuning (as required by ℓ_1-LS, BCS, and CoSaMP), and simultaneous multi-image reconstructions. These factors could allow for image reconstructions in

Table 3.4: Average SSIM ($\times 10^2$), PSNR, and reconstruction time with 99% confidence interval over 8000 test image reconstructions, from Radon transforms at 15 uniformly spaced angles with 60dB SNR, for each reconstruction algorithm.

	SSIM ($\times 10^2$)	PSNR (dB)	Time (s)
nDSR-CG-Net	90.56 ± 0.09	28.00 ± 0.06	$0.662 \pm (3.10 \times 10^{-4})$
gDSR-CG-Net	90.13 ± 0.09	27.83 ± 0.06	$0.435 \pm (1.37 \times 10^{-3})$
ℓ_1-LS	84.61 ± 0.10	24.93 ± 0.06	$0.631 \pm (2.56 \times 10^{-3})$
FBP	82.68 ± 0.14	22.53 ± 0.06	$0.0015 \pm (8.09 \times 10^{-5})$
BCS	77.72 ± 0.16	22.75 ± 0.07	$0.616 \pm (1.68 \times 10^{-2})$
CoSaMP	77.25 ± 0.17	22.58 ± 0.08	$0.400 \pm (7.58 \times 10^{-3})$

real-time. Furthermore, using convolutional layers in gDSR-CG-Net removes the network dependency on the image size. Hence, the number of network parameters remains the same for larger images and so the same trained network is able to be directly applied to larger images. Since nDSR-CG-Net uses a fully connected layer to approximate the regularization gradient, the number of parameters does depend on image size. Thus, larger images require retraining of nDSR-CG-Net along with greater data storage capacities. A drawback of gDSR-CG-Net is that noise has a greater influence on and presence in the estimated images from this network. This noise sensitivity may hinder the success of gDSR-CG-Net when provided measurements that are heavily corrupted by noise.

Additional simulation results are displayed in Table 3.5, which shows the average values and 99% confidence intervals for SSIM, PSNR, and reconstruction time over 8000 test image reconstructions from Radon transforms at 6 uniformly spaced angles with an SNR of 60dB. Again, nDSR-CG-Net and gDSR-CG-Net both outperform the four compared iterative methods in SSIM and PSNR quality while being on par to ℓ_1-LS and BCS in reconstruction time. Furthermore, nDSR-CG-Net provides, on average, a 0.81 SSIM and 0.14 PSNR improvement over gDSR-CG-Net at the cost of 0.199 seconds per image reconstruction as compared to gDSR-CG-Net.

Choosing to use nDSR-CG-Net versus gDSR-CG-Net will depend on user specific considerations. As highlighted in Table 3.4 and Table 3.5, if core user interest is strictly to produce the

Table 3.5: Average SSIM ($\times 10^2$), PSNR, and reconstruction time with 99% confidence interval over 8000 image reconstructions, from Radon transforms at 6 uniformly spaced angles with 60dB SNR, for each reconstruction algorithm.

	SSIM ($\times 10^2$)	PSNR (dB)	Time (s)
nDSR-CG-Net	67.96 ± 0.19	21.84 ± 0.06	$0.599 \pm (1.33 \times 10^{-3})$
gDSR-CG-Net	67.15 ± 0.18	21.70 ± 0.06	$0.400 \pm (5.94 \times 10^{-4})$
ℓ_1-LS	53.99 ± 0.23	19.63 ± 0.07	$0.594 \pm (3.82 \times 10^{-3})$
FBP	53.18 ± 0.18	18.74 ± 0.07	$0.0010 \pm (6.77 \times 10^{-5})$
BCS	40.72 ± 0.24	16.68 ± 0.08	$0.533 \pm (2.62 \times 10^{-3})$
CoSaMP	40.89 ± 0.21	15.77 ± 0.07	$0.244 \pm (2.59 \times 10^{-3})$

highest quality image reconstruction possible then nDSR-CG-Net should always be prioritized over gDSR-CG-Net. However, since the improvement in image quality, of nDSR-CG-Net over gDSR-CG-Net, is relatively small as compared to the increased computational time, e.g. (from Table 3.4) nDSR-CG-Net provides a 0.477% increase in SSIM and 0.611% increase in PSNR at the cost of a 52.2% increase in computational time, gDSR-CG-Net could be the preferred method in general.

Finally, we remark that DSR-CG-Net underperforms both CG-Net, from Chapter 2, and DR-CG-Net, from Section 3.3. This may be due, in part, to the constraint of the inverse non-linearity h^{-1} that exists in both the CG-Net and DSR-CG-Net approaches. From a statistical point-of-view, in implementing the CG prior, the inverse non-linearity applied to the $\boldsymbol{z}$ scale variable is expected to generate a Gaussian random vector. This property is exactly captured in the ℓ_2 norm choice of $\mathcal{R}$ used in CG-Net. Perhaps deviating from this choice, by allowing the network to learn $\mathcal{R}$, is a factor in the decreased image reconstruction performance of DSR-CG-Net. Additionally, in DR-CG-Net, the entirety of the scale variable regularization is learned allowing it to take a form outside of a non-linearity applied to a Gaussian, which DSR-CG-Net is forced to conform to. This increased generalizability of DR-CG-Net is a contributing factor to it outperforming the DSR-CG-Net method.

3.5 Conclusion

Using a powerful learned CG class of densities, that subsume many commonly used priors in imaging and CS, this chapter has presented new fundamental techniques for linear inverse problems. We developed a novel CG-based iterative estimation algorithm, named G-CG-LS, that allows for problem-specific choices of the scale variable distribution by generalizing prior CG-based methods. Applying algorithm unrolling to G-CG-LS, we constructed a novel CG-based deep neural network, named DR-CG-Net, that optimally learns the scale variable portion and the Gaussian covariance portion of the CG distribution. Hence, DR-CG-Net has the flexibility to learn the prior while constraining to the informative CG class of distributions.

We have conducted a theoretical characterization of G-CG-LS and a fundamental numerical validation of DR-CG-Net in tomographic imaging and CS problems. Across multiple datasets, we have empirically demonstrated that our DR-CG-Net significantly outperforms competitive state-of-the-art deep learning-based methods in tomographic imaging and CS scenarios, especially in the difficult case of low training. While CG-Net, which DR-CG-Net expands upon, is the closest comparative method in signal reconstruction quality for the low-training scenarios, it is still appreciably outperformed by DR-CG-Net both in estimated signal quality and in the computational time required to train and reconstruct a signal.

Both the novel G-CG-LS and DR-CG-Net approaches presented in this chapter have been published in our conference proceedings of the IEEE Asilomar Conference on Signals, Systems, and Computers [76] and in our IEEE Transactions on Computational Imaging journal article [77]. Furthermore, a provisional patent for our two innovative methods has been filed.

In the following chapter, we tackle generalization error bounds, which are an important theoretical question for CG-Net, DR-CG-Net, and CG-based deep neural networks as a whole. Specifically, we develop the framework for a generalized CG deep neural network that encompasses both of CG-Net and DR-CG-Net as special cases. For this framework, approaches from statistical learning theory literature are employed, under reasonable assumptions, to construct a generalization error bound. Subsequently, this generalization error bound is applied to both the CG-Net and DR-CG-Net approaches and asymptotic forms of these error bounds are derived.

Chapter 4

Generalization Error Bounds for Deep Compound Gaussian Neural Networks

Algorithm unfolding or unrolling is the technique of constructing a deep neural network (DNN) from an iterative algorithm. Unrolled DNNs often provide better interpretability and superior empirical performance over standard DNNs in signal estimation tasks. An important theoretical question, which has only recently received attention, is the development of generalization error bounds for unrolled DNNs. These bounds deliver theoretical and practical insights into the performance of a DNN on empirical datasets that are distinct from, but sampled from the same probability density as, the DNN training data. In this chapter, we develop novel generalization error bounds for a class of unrolled DNNs that are informed by a compound Gaussian prior. These compound Gaussian networks have been shown to outperform comparative standard and unrolled deep neural networks in compressive sensing and tomographic imaging problems. The developed generalization error bounds are formulated by bounding the Rademacher complexity of the class of compound Gaussian network estimates with Dudley's integral. Under realistic conditions, we show that, at worst, the generalization error scales $\mathcal{O}(n\sqrt{\ln(n)})$ in the signal dimension and $\mathcal{O}((\text{Network Size})^{3/2})$ in network size.

The success of machine learning in image classification has recently spurred its application, in particular with deep neural networks (DNN), in signal estimation tasks. Signal estimation is one example of an inverse problem where we desire a reconstructed signal from some undersampled measurements. Particular applications of interest include X-Ray computed tomography (CT), magnetic resonance imaging (MRI), and compressive sensing.

While DNNs have been shown to outperform iterative approaches in estimated signal quality and computational time, this typically requires a significant amount of available training data as well as training time. Certain problems of interest, such as MRI, do not have large training

datasets readily available and thus the performance of DNNs on these problems can falter. Additionally, standard DNNs, e.g. convolutional neural networks, do not incorporate any outside prior information into the estimation model, as is included in iterative approaches.

Algorithm unfolding or unrolling is a technique, introduced by Gregor and LeCun [20], which combines the iterative approaches and deep learning methods by structuring the layers of a DNN such that each layer corresponds to an iteration from the iterative algorithm. While a standard DNN acts as a black-box process, we understand the inner workings of an unrolled DNN from understanding the original iterative algorithm. Furthermore, algorithm unrolling allows for the incorporation of prior information into the deep learning framework. Example iterative algorithms that have been unrolled into DNNs include: iterative shrinkage and thresholding (ISTA) [20, 22, 23, 25] and proximal gradient descent [27, 28]. These unrolled DNNs have shown excellent performance in image estimation while offering simple interpretability of the network layers [21], although still requiring large training datasets.

In Chapter 2 and Chapter 3 we have introduced and evaluated two compound Gaussian (CG) informed DNNs that were developed through the use of algorithm unrolling. These CG unrolled DNNs have empirically shown to produce superior estimated signals, over comparative iterative and DNN methods in linear inverse problems, especially in the low training data scenario [57, 58, 76, 77]. While this success has been shown empirically, no generalization guarantees, i.e. guarantees on the ability of a DNN to perform well on unseen test data, have been provided; a gap which this has chapter filled. Specifically, in this chapter we have completed:

1. The establishment and proving of a Lipschitz property of the outputs from unrolled, CG-informed DNNs with respect to the DNN parameters.

2. The development of an encompassing generalization error bound (GEB) for unrolled, CG-informed DNNs. The developed GEB results from bounding Rademacher complexities with integrals of covering numbers through Dudley's inequality and subsequently bounding these integrals using a Lipschitz property of CG-informed DNNs.

3. The application of the developed GEB for two distinct formulations of unrolled, CG-based DNNs named compound Gaussian network (CG-Net) and deep regularized compound Gaussian network (DR-CG-Net) [58, 77] proposed in Chapter 2 and Chapter 3, respectively.

4. The development of asymptotic forms for the CG-Net and DR-CG-Net generalization error bounds that depend on reconstructed signal dimension and DNN size. Theoretically, we demonstrate that DR-CG-Net exhibits a tighter GEB than CG-Net, which was empirically observed in [58, 77] and in Chapter 3.

The GEB we have developed in this chapter is formulated using techniques inspired by those discussed in [86] which produces a GEB for an ISTA unrolled DNN with a learned sparsity transformation. Such techniques are similarly used in [87] for an unrolled DNN employing a learned analysis sparsity transformation and in [88] for an unrolled ℓ_1-ℓ_1 recurrent neural network.

4.1 Notation and Nomenclature

The set of symbols provided in this section are defined here for ease of referencing and in reading the remainder of this chapter. We note that some have already been defined previously.

$\mathbb{R}$	$=$	Set of real numbers.
$\boldsymbol{v}$	$=$	$[v_i]_{i=1,2,\ldots,d} \in \mathbb{R}^d$. Boldface characters are vectors.
$(\cdot)^T$	$=$	Transpose of vector or matrix $(\cdot)$
$\odot$	$=$	Hadamard product.
$g(\boldsymbol{v})$	$=$	$[g(v_i)]_{i=1,2,\ldots,d} \in \mathbb{R}^d$ for $\boldsymbol{v} \in \mathbb{R}^d$ and a **componentwise function** $g : \mathbb{R} \to \mathbb{R}$.
$\mathbb{N}[d]$	$=$	$\{1, 2, \ldots, d\}$ is the set of the first d natural numbers.
$\text{ReLU}(x)$	$=$	$\max\{0, x\}$ is a componentwise function.
$\mathcal{P}_{a,b}(x)$	$=$	a+ReLU$(x-a)-$ReLU$(x-b)$, for $a, b \in \mathbb{R}$, is a **modified ReLU** (mReLU) activation function. This componentwise function projects input x onto the interval $[\min\{a,b\}, \max\{a,b\}]$.
$\rho_b(\boldsymbol{v})$	$=$	$\boldsymbol{v}/\max\{1, b^{-1}\|\boldsymbol{v}\|_2\}$ for $b > 0$, is the b-Ball mapping, i.e. the projection onto the Euclidean ball of radius b.
n	$=$	Signal-of-interest dimension or reconstructed signal dimension.
m	$=$	Measurement dimension.
A	$\in$	$\mathbb{R}^{m \times n}$ is a measurement, observation, or sensing matrix.

$\boldsymbol{\nu}$	$\in$	$\mathbb{R}^m$ additive measurement noise.
$\boldsymbol{y}$	$=$	$A\boldsymbol{c} + \boldsymbol{\nu}$ forward measurement model.
$\boldsymbol{z}$	$=$	The scale variable of a compound Gaussian distribution.
$\boldsymbol{u}$	$=$	The Gaussian variable of a compound Gaussian distribution.
$A_{\boldsymbol{z}}$	$=$	$A\text{Diag}(\boldsymbol{z}) \in \mathbb{R}^{m\times n}$ for any vector $\boldsymbol{z} \in \mathbb{R}^n$.
$\mathcal{T}_{\boldsymbol{y}}(\boldsymbol{z})$	$\equiv$	$\mathcal{T}_{\boldsymbol{y}}(\boldsymbol{z}; P_u) \coloneqq (A_{\boldsymbol{z}}^T A_{\boldsymbol{z}} + P_u^{-1})^{-1} A_{\boldsymbol{z}}^T \boldsymbol{y} = P_u A_{\boldsymbol{z}}^T (I + A_{\boldsymbol{z}} P_u A_{\boldsymbol{z}}^T)^{-1} \boldsymbol{y}$.
N_s	$=$	The number of training data points.
$\mathcal{S}$	$=$	$\{(\overline{\boldsymbol{y}}_i, \overline{\boldsymbol{c}}_i)\}_{i\in\mathbb{N}[N_s]}$ is a training dataset.
L	$=$	Neural network loss function.
$\widehat{\boldsymbol{c}}(\boldsymbol{y}; \boldsymbol{\Theta})$	$=$	Neural network output given input $\boldsymbol{y}$ and parameterization $\boldsymbol{\Theta}$.
$\mathcal{L}_{\mathcal{S}}(\widehat{\boldsymbol{c}})$	$=$	Empirical loss of a hypothesis $\widehat{\boldsymbol{c}}$ over training dataset $\mathcal{S}$.
$\mathcal{L}(\widehat{\boldsymbol{c}})$	$=$	Actual loss of a hypothesis $\widehat{\boldsymbol{c}}$.
$\mathcal{R}_{\mathcal{S}}$	$=$	Empirical Rademacher complexity over the training dataset $\mathcal{S}$.
$\mathcal{N}(M, d, \epsilon)$	$=$	ϵ-covering number of metric space (M, d).
$(X_t)_{t\in T}$	$=$	Stochastic process over state space S and index set T.
$\Delta(T)$	$=$	$\sup_{t\in T} \sqrt{\mathbb{E}\lvert X_t\rvert^2}$ is the radius of T (or radius of $(X_t)_{t\in T}$).
$\mathcal{H}_{\text{CG}}^{(1)}$	$=$	Hypothesis class, i.e. space of possible outputs, for generalized compound Gaussian network (G-CG-Net).
$\mathcal{H}_{\text{CG}}^{(2)}$	$=$	Hypothesis class for CG-Net.
$\mathcal{H}_{\text{CG}}^{(3)}$	$=$	Hypothesis class for DR-CG-Net.
$\mathbb{E}$	$=$	Expected value operator.
$\lVert\cdot\rVert_p$	$=$	Standard finite-dimensional vector space p-norm or induced matrix operator p-norm.
$\mathcal{O}$	$=$	Big O notation. That is, $f(x) = \mathcal{O}(g(x))$ as $x \to \infty$ implies there exists a $c > 0$ and $x_0 \geq 0$ such that $\lvert f(x)\rvert \leq cg(x)$ for all $x \geq x_0$.
o	$=$	Little o notation. That is, $f(x) = o(g(x))$ implies $\lim_{x\to\infty} f(x)/g(x) \to 0$.
$\lesssim$	$=$	Scaling symbol. That is, $a \lesssim b$ implies $a \leq cb$ for some constant $c > 0$.
$c_{\max}$	$=$	Euclidean norm bound on $\boldsymbol{c}$, the original signals-of-interest.
z_∞	$=$	Maximum entry bound, i.e. ∞-norm bound, on the scale variables $\boldsymbol{z}$.
$\mathcal{P}$	$=$	$\{P \in \mathbb{R}^{n\times n} :$ symmetric, positive definite, $\lVert P\rVert_2 \leq p_{\max}, \lVert P^{-1}\rVert_2 \leq p_{\min}^{-1}\}$.
$p_{\max}$	$=$	Positive upper spectral bound on covariance matrices $P \in \mathcal{P}$.
$p_{\min}$	$=$	Positive lower spectral bound on covariance matrices $P \in \mathcal{P}$.
$\mathcal{P}_{\text{full}}$	$\subseteq$	$\mathcal{P}$ is a set of full covariance matrices.
$\mathcal{P}_{\text{tri}}$	$\subseteq$	$\mathcal{P}_{\text{full}}$ is a set of tridiagonal covariance matrices.
$\mathcal{P}_{\text{diag}}$	$\subseteq$	$\mathcal{P}_{\text{tri}}$ is a set of diagonal covariance matrices.
$\mathcal{P}_{\text{const}}$	$\subseteq$	$\mathcal{P}_{\text{diag}}$ is a set of scaled identity covariance matrices.
$\dim(V)$	$=$	Dimension of vector space V.
$\boldsymbol{\theta}_{k,d}^{(j)}$	$=$	Scale-variable update parameters.
$\lVert\cdot\rVert_{(d)}$	$=$	A norm on the scale-variable update parameter $\boldsymbol{\theta}_{k,d}^{(j)}$.
ω_d	$=$	Bound on the scale-variable update parameters, i.e. $\lVert\boldsymbol{\theta}_{k,d}^{(j)}\rVert_{(d)} \leq \omega_d$.

α_d	=	Dimension of the scale-variable update parameter $\boldsymbol{\theta}_{k,d}^{(j)}$.
$B_k^{(j)}$	=	Learned positive definite steepest descent matrix in CG-Net.
$\mu_k^{(j)}$	=	Learned regularization scalar parameter in CG-Net.
μ	=	Positive bound on the learned CG-Net regularization scalar parameter.
ξ	=	Positive, real-valued scalar parameter that defines the b-Ball mapping used in each CG-Net and DR-CG-Net scale-variable update.
$\delta_k^{(j)}$	=	Learned scalar step size parameter in DR-CG-Net.
δ	=	Positive bound on the learned DR-CG-Net step size parameter.
$\mathcal{V}_k^{(j)}$	=	Convolutional neural network implemented as a subnetwork in DR-CG-Net.
L_c	=	Number of convolutional layers in $\mathcal{V}_k^{(j)}$.
f_ℓ	=	Number of output filter channels from convolutional layer ℓ in $\mathcal{V}_k^{(j)}$.
k_ℓ	=	Kernel size of the convolution in layer ℓ of $\mathcal{V}_k^{(j)}$.
$W_{k,\ell}^{(j)}$	=	Weight matrix representation of the convolution in layer ℓ of $\mathcal{V}_k^{(j)}$.
w_ℓ	=	Bound on the convolutional kernel in layer ℓ of $\mathcal{V}_k^{(j)}$. That is, $\|W_{k,\ell}^{(j)}\|_2 \leq w_\ell$.

4.2 Preliminaries

In this section, we define the generalization error of a DNN and provide some preliminary statistical learning theory notation and results from the existing literature [1, 86–89] that we have employed in the development of a generalization error bound for unrolled, CG-based DNNs. For completeness of this chapter, we first provide a brief mathematical overview of a DNN.

A DNN can be viewed as a collection of ordered layers, denoted $\boldsymbol{L}_0, \boldsymbol{L}_1, \ldots, \boldsymbol{L}_K$ for $K > 1$, where layers feed into one another from the input layer, $\boldsymbol{L}_0$, to the output layer, $\boldsymbol{L}_K$. Intermediate layers $\boldsymbol{L}_1, \ldots, \boldsymbol{L}_{K-1}$ are known as hidden layers. Each layer, $\boldsymbol{L}_k$, contains d_k hidden units [48], which are assigned a computed value when a signal is transmitted through the DNN. For simplicity, we identify the nodes with the computed values assigned to each node when the DNN is processing a signal.

A function $\boldsymbol{f}_k : \mathbb{R}^{d_{i_1(k)}} \times \cdots \times \mathbb{R}^{d_{i_j(k)}} \to \mathbb{R}^{d_k}$, that is parameterized by some $\boldsymbol{\theta}_k$, defines the computation, i.e. signal transmission, at layer $\boldsymbol{L}_k$ where $\mathcal{I}_k \coloneqq \{i_1(k), \ldots, i_j(k)\} \subseteq \{0, 1, \ldots, K-1\}$ are the indices of layers that feed into $\boldsymbol{L}_k$. That is, given an input signal $\overline{\boldsymbol{y}} \in \mathbb{R}^{d_0}$ assigned to

$\boldsymbol{L}_0$, a DNN is the composition of parameterized vector input and vector output functions where

$$\boldsymbol{L}_k \equiv \boldsymbol{f}_k\left(\boldsymbol{L}_{i_1(k)}, \ldots, \boldsymbol{L}_{i_j(k)}; \boldsymbol{\theta}_k\right).$$

A DNN learns, or trains, its parameters, $\boldsymbol{\Theta} = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K)$, by optimizing a loss function $\mathcal{L}(\boldsymbol{\Theta})$ over a training dataset $\mathcal{S} = \{(\overline{\boldsymbol{y}}_i, \overline{\boldsymbol{c}}_i) : i = 1, 2, \ldots, N_s\}$ where each $(\overline{\boldsymbol{y}}_i, \overline{\boldsymbol{c}}_i)$ is a pair of a signal measurement and corresponding true signal. Let $\widehat{\boldsymbol{c}}(\overline{\boldsymbol{y}}_i; \boldsymbol{\Theta})$ denote the DNN output given the input $\overline{\boldsymbol{y}}_i$. Then the loss function is often defined as $\mathcal{L}(\boldsymbol{\Theta}) \coloneqq \frac{1}{N_s} \sum_{i=1}^{N_s} L\left(\widehat{\boldsymbol{c}}(\overline{\boldsymbol{y}}_i; \boldsymbol{\Theta}), \overline{\boldsymbol{c}}_i\right)$ where $L(\boldsymbol{x}_1, \boldsymbol{x}_2) : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ calculates the error between inputs $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$. That is, $L\left(\widehat{\boldsymbol{c}}(\overline{\boldsymbol{y}}_i; \boldsymbol{\Theta}), \overline{\boldsymbol{c}}_i\right)$ is the error between the network output, $\widehat{\boldsymbol{c}}(\overline{\boldsymbol{y}}_i; \boldsymbol{\Theta})$, and the actual coefficients, $\overline{\boldsymbol{c}}_i$.

4.2.1 Generalization Error

Let $\mathcal{C}$ and $\mathcal{Y}$ be the set of possible finite-dimensional signals of interest, e.g. images, and possible finite-dimensional signal measurements, e.g. image measurements, respectively. Further, let $(\mathcal{C}, \mathcal{A}_c, \mathcal{D}_c)$ be a probability space where $\mathcal{A}_c$ and $\mathcal{D}_c$ are a σ-algebra and unknown probability density on $\mathcal{C}$, respectively. Similarly, define the probability space $(\mathcal{Y}, \mathcal{A}_y, \mathcal{D}_y)$. Now, consider the probability space $(\mathcal{Y} \times \mathcal{C}, \mathcal{A}_y \otimes \mathcal{A}_c, \mathcal{D})$ where $\mathcal{D}$ is an unknown joint probability density with marginal distributions $\mathcal{D}_c$ and $\mathcal{D}_y$.

Let $\mathcal{S} = \{(\overline{\boldsymbol{y}}_i, \overline{\boldsymbol{c}}_i)\}_{i \in \mathbb{N}[N_s]}$ be a training dataset where each pair $(\overline{\boldsymbol{y}}_i, \overline{\boldsymbol{c}}_i)$ is drawn i.i.d. from $\mathcal{D}$. We denote the network parameters as $\boldsymbol{\Theta} = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K)$ and let Ω_k be the space of $\boldsymbol{\theta}_k$ that can be learned by the DNN. Further, we write the network output, i.e. signal at $\boldsymbol{L}_K$ that is dependent on network parameters and input measurements $\boldsymbol{y}$, as $\widehat{\boldsymbol{c}}(\boldsymbol{y}; \boldsymbol{\Theta})$. We remark that $\widehat{\boldsymbol{c}}(\boldsymbol{y}; \boldsymbol{\Theta})$ is not a single output, but actually a parameterized function with input $\boldsymbol{y}$ and parameterization $\boldsymbol{\Theta}$.

Now, we provide a few required definitions, from the literature [86], for which we let $L\left(\boldsymbol{x}_1, \boldsymbol{x}_2\right)$ be a loss function that measures the distance between samples $\boldsymbol{x}_1 \in \mathbb{R}^{d_K}$ and $\boldsymbol{x}_2 \in \mathbb{R}^{d_K}$. Common loss functions include mean-squared error and mean-absolute error [49].

Definition 4.2.1 ([86]). *The **hypothesis space**, $\mathcal{H}$, of a DNN is the space of possible parameterized functions that can be generated for the DNN. That is*

$$\mathcal{H} = \{\widehat{\boldsymbol{c}}(\cdot;\boldsymbol{\Theta}) : \boldsymbol{\Theta} \in \Omega_1 \times \cdots \times \Omega_K\}.$$

Definition 4.2.2 ([86]). *The **empirical loss** of a hypothesis $\widehat{\boldsymbol{c}} \in \mathcal{H}$, over a training dataset $\mathcal{S}$, is given by*

$$\mathcal{L}_{\mathcal{S}}(\widehat{\boldsymbol{c}}) = \frac{1}{N_s}\sum_{i=1}^{N_s} L\left(\widehat{\boldsymbol{c}}(\overline{\boldsymbol{y}}_i;\boldsymbol{\Theta}), \overline{\boldsymbol{c}}_i\right).$$

Definition 4.2.3 ([86]). *The **actual loss** of a hypothesis $\widehat{\boldsymbol{c}} \in \mathcal{H}$ is defined as*

$$\mathcal{L}(\widehat{\boldsymbol{c}}) = \mathbb{E}_{(\boldsymbol{y},\boldsymbol{c})\sim\mathcal{D}}\left[L(\widehat{\boldsymbol{c}}(\boldsymbol{y};\boldsymbol{\Theta}), \boldsymbol{c})\right]$$

where $\mathbb{E}_{(\boldsymbol{y},\boldsymbol{c})\sim\mathcal{D}}\left[L(\widehat{\boldsymbol{c}}(\boldsymbol{y};\boldsymbol{\Theta}), \boldsymbol{c})\right]$ is the expected value or average of $L(\widehat{\boldsymbol{c}}(\boldsymbol{y};\boldsymbol{\Theta}), \boldsymbol{c})$ over samples $(\boldsymbol{y}, \boldsymbol{c})$ drawn from the distribution $\mathcal{D}$.

Finally, we define the generalization error for a DNN.

Definition 4.2.4 ([86]). *The **generalization error**, denoted $GE_{\mathcal{S}}$, of a hypothesis $\widehat{\boldsymbol{c}} \in \mathcal{H}$ is the difference between the empirical and actual loss. That is,*

$$\mathrm{GE}_{\mathcal{S}}(\widehat{\boldsymbol{c}}) = |\mathcal{L}(\widehat{\boldsymbol{c}}) - \mathcal{L}_{\mathcal{S}}(\widehat{\boldsymbol{c}})|.$$

We remark that the training of a DNN is equivalent to iteratively minimizing the empirical loss, i.e. by choosing a hypothesis $\widehat{\boldsymbol{c}} \in \mathcal{H}$ that minimizes $\mathcal{L}_{\mathcal{S}}(\widehat{\boldsymbol{c}})$. A hypothesis that minimizes the empirical loss produces estimates $\widehat{\boldsymbol{c}}(\overline{\boldsymbol{y}}_i;\boldsymbol{\Theta})$ that closely match the signals of interest $\overline{\boldsymbol{c}}_i$ where $(\overline{\boldsymbol{y}}_i, \overline{\boldsymbol{c}}_i)$ is a training data pair. It is critical in applications, however, for a DNN to similarly generate excellent results when provided any new data sample drawn from $\mathcal{D}$ that is not contained in the training dataset. This is tantamount to minimizing the generalization error and thus, obtaining an

estimate or bound on $\text{GE}_{\mathcal{S}}$ is of significant interest. A tight bound on the generalization error, for a hypothesis $\widehat{\boldsymbol{c}} \in \mathcal{H}$, implies that the network generalizes well from the training data to the testing data. That is, the reconstruction performance of the DNN $\widehat{\boldsymbol{c}}$ on training data accurately represents the reconstruction performance on any arbitrary test data drawn from the same distribution.

4.2.2 Rademacher Complexity

A Rademacher variable is a real valued and discrete random variable γ taking values ± 1 with equal probability. Using Rademacher variables we provide a definition of an empirical Rademacher complexity as given in [1].

Definition 4.2.5 ([1]). *Let $\mathcal{G}$ be a set of functions $g : \mathcal{X} \to \mathbb{R}$ and $\mathcal{S} = \{\boldsymbol{x}_i\}_{i=1}^{N_s} \subseteq \mathcal{X}$. The **empirical Rademacher complexity** of $\mathcal{G}$ is*

$$\mathcal{R}_{\mathcal{S}}(\mathcal{G}) = \mathbb{E}_{\boldsymbol{\gamma}}\left[\sup_{g\in\mathcal{G}} \frac{1}{N_s}\sum_{i=1}^{N_s} \gamma_i g(\boldsymbol{x}_i)\right]$$

for $\boldsymbol{\gamma} = [\gamma_1, \ldots, \gamma_{N_s}]$ a vector of i.i.d. Rademacher variables.

For a DNN with hypothesis space $\mathcal{H}$ using loss function L, we are interested in the Rademacher complexity of the real-valued set of functions $\mathcal{G} = L \circ \mathcal{H} = \{L \circ h : h \in \mathcal{H}\}$ as we apply the loss function L to the DNN outputs to assess performance. The following theorem, from [90], establishes a bound on the actual loss in terms of the empirical loss and Rademacher complexity of $L \circ \mathcal{H}$.

Theorem 4.2.6 ([90], Theorem 26.5). *Let $\mathcal{H}$ be a set of functions, $\mathcal{S} = \{(\overline{\boldsymbol{y}}_i, \overline{\boldsymbol{c}}_i)\}_{i=1}^{N_s}$ a training set drawn i.i.d. from $\mathcal{D}$, and L a real-valued loss function satisfying $|L(h(\boldsymbol{y}), \boldsymbol{c})| \leq c$ for all $h \in \mathcal{H}$ and $(\boldsymbol{y}, \boldsymbol{c}) \sim \mathcal{D}$. Then, for $\varepsilon \in (0, 1)$ with probability at least $1 - \varepsilon$ we have for all $h \in \mathcal{H}$*

$$\mathcal{L}(h) \leq \mathcal{L}_{\mathcal{S}}(h) + 2\mathcal{R}_{\mathcal{S}}(L \circ \mathcal{H}) + 4c\sqrt{2\ln(4/\varepsilon)/N_s}.$$

Next, we provide a contraction lemma, from [86, 91], allowing us to ignore the loss function and only consider the hypothesis class.

Lemma 4.2.7 ([91] Corollary 4, [86] Lemma 2)**.** *Let $\mathcal{H}$ be a set of functions $h : \mathcal{X} \to \mathbb{R}^d$ and $\mathcal{S} = \{\boldsymbol{x}_i\}_{i=1}^{N_s} \subseteq \mathcal{X}$. Then for any τ-Lipschitz functions $g_i : \mathbb{R}^d \to \mathbb{R}$, where $i \in \mathbb{N}[N_s]$,*

$$\mathbb{E}_{\boldsymbol{\gamma}}\left[\sup_{h\in\mathcal{H}}\sum_{i=1}^{N_s}\gamma_i g_i \circ h(\boldsymbol{x}_i)\right] \leq \sqrt{2}\tau\mathbb{E}_{\Gamma}\left[\sup_{h\in\mathcal{H}}\sum_{i=1}^{N_s}\sum_{k=1}^{d}\gamma_{ik}h_k(\boldsymbol{x}_i)\right]$$

where $\boldsymbol{\gamma} = [\gamma_i]_{i\in\mathbb{N}[N_s]}$ and $\Gamma = [\gamma_{ik}]_{i\in\mathbb{N}[N_s]}^{k\in\mathbb{N}[d]}$ are collections of i.i.d. Rademacher variables.

The left hand side of the inequality in Lemma 4.2.7 is the Rademacher complexity of $L \circ \mathcal{H}$ when we set each $g_i = L$, whereas the right hand side of the inequality in Lemma 4.2.7 is a scaled version of the Rademacher complexity on $\mathcal{H}$. Therefore, Lemma 4.2.7 allows for the removal of the loss function from consideration so long as the loss function is Lipschitz.

4.2.3 Dudley's Inequality

To define Dudley's inequality, which is employed to obtain a bound on $\mathcal{R}_{\mathcal{S}}(L \circ \mathcal{H})$ in Theorem 4.2.6, we require some terminology from [1, 89]. First, we define the covering of a metric space.

Definition 4.2.8 ([89])**.** *Let (M, d) be a metric space. An ϵ-**covering** of M is a subset $C_M(\epsilon) \subseteq M$ satisfying for every $x \in M$ there exists a $c \in C_M(\epsilon)$ such that $d(x, c) \leq \epsilon$.*

Second, we define the covering number of a metric space.

Definition 4.2.9 ([89])**.** *Let (M, d) be a metric space. The ϵ-**covering number** of M, denoted $\mathcal{N}(M, d, \epsilon)$, is a natural number giving the minimum cardinality of all ϵ-coverings $C_M(\epsilon)$. Equivalently, $\mathcal{N}(M, d, \epsilon)$ is the minimum number of ϵ radius balls, as measured by d, to contain M.*

Third, we define a sub-Gaussian stochastic process. For this we first define a stochastic process.

Definition 4.2.10 ([1])**.** *Let S and T be sets called the **state space** and **index set**, respectively. A **stochastic process** or **random process** is a collection of random variables $(X_t)_{t\in T}$, each defined on a common probability space $(\Omega, \mathcal{F}, \mathbb{P})$, such that X_t takes values in S for each $t \in T$.*

Now, we define a sub-Gaussian stochastic process.

Definition 4.2.11 ([1]). *A real-valued stochastic process $(X_t)_{t\in T}$ is called a **sub-Gaussian process** if $\mathbb{E}(X_t) = 0$ and for all $s, t \in T$ and $\theta > 0$*

$$\mathbb{E}\left[\exp(\theta(X_s - X_t))\right] \leq \exp\left(\theta^2 \widetilde{d}(X_s, X_t)^2/2\right)$$

where $\widetilde{d}(X_s, X_t) = \left(\mathbb{E}\left|X_s - X_t\right|^2\right)^{1/2}$ is a pseudo-metric.

Fourth, we define the radius of a stochastic process or, equivalently, the radius of the index set.

Definition 4.2.12 ([1]). *Let $(X_t)_{t\in T}$ be a real-valued stochastic process. The **radius** of X_T, or the radius of T, is denoted by $\Delta(T)$ and defined as*

$$\Delta(T) = \sup_{t\in T} \sqrt{\mathbb{E}|X_t|^2}.$$

With these four definitions we provide Dudley's Inequality.

Lemma 4.2.13 (Dudley's Inequality, [89], Theorem 8.23 [1]). *For a sub-Gaussian stochastic process $(X_t)_{t\in T}$ with pseudo-metric $\widetilde{d}$*

$$\mathbb{E}\left(\sup_{t\in T} X_t\right) \leq 4\sqrt{2} \int_0^{\Delta(T)/2} \sqrt{\ln\left(\mathcal{N}(T, \widetilde{d}, \epsilon)\right)} d\epsilon.$$

In plain language, Dudley's Inequality bounds the expected maximum of a stochastic process by an integral over the root log covering number of the stochastic process index set.

4.2.4 Rademacher Process

A Rademacher process is a stochastic process where the stochastic properties of the process are driven entirely by Rademacher random variables. To formally define a Rademacher process note that for a vector space V and set D we write V^D to denote the set of all functions that map from D to V. That is, $V^D = \{f \text{ such that } f : D \to V\}$.

Definition 4.2.14 ([1]). *Let V be a vector space, T an index set, and $\mathcal{X} = V^T$. A **Rademacher process**, $(X_t)_{t\in T}$, is a stochastic process of the form $X_t = \sum_{k=1}^n \gamma_k x_k(t)$ where $x_k \in \mathcal{X}$ for each $k = 1, 2, \ldots, n$ and $\boldsymbol{\gamma} = [\gamma_1, \ldots, \gamma_n]$ are i.i.d. Rademacher variables.*

We remark that the state space of a Rademacher process is the vector space V. Additionally, the functions x_k are deterministic and thus the Rademacher variables $\boldsymbol{\gamma}$ drive the stochastic nature for a Rademacher process. Now, we highlight two useful properties, from [1], of scalar valued Rademacher processes, i.e. a Rademacher process such that $V \subseteq \mathbb{C}$.

Lemma 4.2.15 ([1]). *Let $(X_t)_{t\in T}$ be a Rademacher process on state space $V \subseteq \mathbb{C}$. That is $X_t = \sum_{k=1}^n \gamma_k x_k(t)$ for $x_k \in V^T$ for each $k = 1, 2, \ldots, n$ and $\boldsymbol{\gamma} = [\gamma_1, \ldots, \gamma_n]$ are i.i.d. Rademacher variables. Then*

$$\mathbb{E}_{\boldsymbol{\gamma}}(|X_t|^2) = \sum_{k=1}^n |x_k(t)|^2$$

and

$$\widetilde{d}(X_s, X_t)^2 \coloneqq \mathbb{E}_{\boldsymbol{\gamma}}|X_s - X_t|^2 = \sum_{k=1}^n (|x_k(s) - x_k(t)|)^2.$$

Proof. For $a \in \mathbb{C}$, let $R(a) \in \mathbb{R}$ and $C(a) \in \mathbb{R}$ respectively denote the real and complex parts of a, i.e. $a = R(a) + iC(a)$, and recall that $|a| = \sqrt{R(a)^2 + C(a)^2}$. Observe

$$\begin{aligned}
\mathbb{E}_{\boldsymbol{\gamma}}(|X_t|^2) &= \mathbb{E}_{\boldsymbol{\gamma}}\left(\left|\sum_{k=1}^n \gamma_k x_k(t)\right|^2\right) \\
&= \mathbb{E}_{\boldsymbol{\gamma}}\left(\left|\sum_{k=1}^n \gamma_k R(x_k(t)) + i\sum_{k=1}^n \gamma_k C(x_k(t))\right|^2\right) \\
&= \mathbb{E}_{\boldsymbol{\gamma}}\left(\left(\sum_{k=1}^n \gamma_k R(x_k(t))\right)^2 + \left(\sum_{k=1}^n \gamma_k C(x_k(t))\right)^2\right). \qquad (4.1)
\end{aligned}$$

Now, for any $c_k \in \mathbb{R}$ where $k = 1, 2, \ldots, n$, using the linearity of the expected value operator, note

$$\mathbb{E}_{\boldsymbol{\gamma}}\left(\left(\sum_{k=1}^{n}\gamma_k c_k\right)^2\right) = \sum_{k=1}^{n}\mathbb{E}_{\boldsymbol{\gamma}}\left(\gamma_k^2 c_k^2\right) + \sum_{k=1}^{n}\sum_{\substack{j=1\\ j\neq k}}^{n}\mathbb{E}_{\boldsymbol{\gamma}}\left(\gamma_j\gamma_k c_j c_k\right) = \sum_{k=1}^{n} c_k^2 \tag{4.2}$$

where in the final equality we used that $\gamma_k^2 = 1$ and $\mathbb{E}_{\boldsymbol{\gamma}}(\gamma_j\gamma_k) = \mathbb{E}_{\boldsymbol{\gamma}}(\gamma_i)\mathbb{E}_{\boldsymbol{\gamma}}(\gamma_k) = 0$ since γ_j and γ_k are independent for any $j \neq k$ and $\mathbb{E}_{\boldsymbol{\gamma}}(\gamma_k) = 0$ for all k. Combining equation (4.1) and equation (4.2) produces

$$\begin{aligned}
\mathbb{E}_{\boldsymbol{\gamma}}(|X_t|^2) &= \mathbb{E}_{\boldsymbol{\gamma}}\left(\left(\sum_{k=1}^{n}\gamma_k R(x_k(t))\right)^2\right) + \mathbb{E}_{\boldsymbol{\gamma}}\left(\left(\sum_{k=1}^{n}\gamma_k C(x_k(t))\right)^2\right)\\
&= \sum_{k=1}^{n} R(x_k(t))^2 + \sum_{k=1}^{n} C(x_k(t))^2\\
&= \sum_{k=1}^{n} R(x_k(t))^2 + C(x_k(t))^2\\
&= \sum_{k=1}^{n} |x_k(t)|^2.
\end{aligned}$$

The second property $\widetilde{d}(X_s, X_t)^2 \coloneqq \mathbb{E}_{\boldsymbol{\gamma}}|X_s - X_t|^2 = \sum_{k=1}^{n}(|x_k(s) - x_k(t)|)^2$ is an immediate consequence of the first property by noting that $X_s - X_t = \sum_{k=1}^{n}\gamma_k(x_k(s) - x_k(t))$ is a Rademacher process. ⊠

Finally, we demonstrate that real-valued Rademacher processes are sub-Gaussian.

Lemma 4.2.16 ([1])**.** *Any real-valued Rademacher process, $(X_t)_{t\in T}$ for $X_t \in \mathbb{R}$, is a sub-Gaussian process.*

Proof. Let $(X_t)_{t\in T}$ be a Rademacher process. Using the linearity of the expected value operator and that the expected value of a Rademacher variable is 0, observe for all $t \in T$

$$\mathbb{E}_{\boldsymbol{\gamma}}(X_t) = \mathbb{E}_{\boldsymbol{\gamma}}\left(\sum_{k=1}^{n}\gamma_k x_k(t)\right) = \sum_{k=1}^{n}\mathbb{E}_{\gamma_k}(\gamma_k)x_k(t) = 0.$$

First, note that, for any Rademacher variable, γ, and $x \in \mathbb{R}$

$$\mathbb{E}_\gamma(\exp(\gamma x)) = \frac{1}{2}\exp(x) + \frac{1}{2}\exp(-x). \tag{4.3}$$

Second, we show that $(2k)! \geq 2^k k!$ for all $k \in \mathbb{N} \cup \{0\}$. The base case $k = 0$ holds trivially as $0! = 1 = 2^0 0!$. Now assume that $(2k)! \geq 2^k k!$ holds for some fixed $k > 0$ and observe

$$\begin{aligned}(2(k+1))! = (2k+2)! &= (2k+2)(2k+1)(2k)! \\ &\geq (2k+2)(2k+1)2^k k! \\ &= 2(k+1)(2k+1)2^k k! = (2k+1)2^{k+1}(k+1)! \geq 2^{k+1}(k+1)!\end{aligned}$$

where the final inequality holds since $(2k+1) \geq 1$ for all $k \geq 0$. Thus, $(2k)! \geq 2^k k!$ for all $k \in \mathbb{N} \cup \{0\}$.

Third, note that for all $x \in \mathbb{R}$, using the Taylor series of $\exp(x)$ and that $(2k)! \geq 2^k k!$ for all $k \in \mathbb{N} \cup \{0\}$

$$\begin{aligned}\frac{1}{2}\left(\exp(x) + \exp(-x)\right) &= \frac{1}{2}\left(\sum_{k=0}^{\infty}\frac{x^k}{k!} + \sum_{k=0}^{\infty}\frac{(-x)^k}{k!}\right) \\ &= \frac{1}{2}\sum_{k=0}^{\infty}\frac{x^{2k}}{(2k)!} \\ &\leq \frac{1}{2}\sum_{k=0}^{\infty}\frac{x^{2k}}{2^k k!} = \frac{1}{2}\sum_{k=0}^{\infty}\frac{\left(\frac{x^2}{2}\right)^k}{k!} = \frac{1}{2}\exp\left(\frac{x^2}{2}\right) \leq \exp\left(\frac{x^2}{2}\right).\end{aligned} \tag{4.4}$$

Now, recall for any independent random variables X and Y that $\mathbb{E}(XY) = \mathbb{E}(X)\mathbb{E}(Y)$. Additionally, $f_1(X)$ and $f_2(Y)$ are independent random variables for any functions f_1 and f_2 defined on the domains on which X and Y take values in. Using these two facts along with equation (4.3)

and equation (4.4) observe for any $s, t \in T$ and $\theta > 0$

$$\begin{aligned}
\mathbb{E}_{\boldsymbol{\gamma}}(\exp(\theta(X_s - X_t))) &= \mathbb{E}_{\boldsymbol{\gamma}}\left(\exp\left(\theta \sum_{k=1}^{n} \gamma_k(x_k(s) - x_k(t))\right)\right) \\
&= \mathbb{E}_{\boldsymbol{\gamma}}\left(\prod_{k=1}^{n} \exp\left(\theta\gamma_k(x_k(s) - x_k(t))\right)\right) \\
&= \prod_{k=1}^{n} \mathbb{E}_{\gamma_k}\left(\exp\left(\theta\gamma_k(x_k(s) - x_k(t))\right)\right) \\
&= \prod_{k=1}^{n} \frac{1}{2}\left(\exp\left(\theta(x_k(s) - x_k(t))\right) + \exp\left(-\theta(x_k(s) - x_k(t))\right)\right) \\
&\leq \prod_{k=1}^{n} \exp\left(\theta^2(x_k(s) - x_k(t))^2/2\right) \\
&= \exp\left(\theta^2 \sum_{k=1}^{n} (x_k(s) - x_k(t))^2/2\right) \\
&= \exp\left(\theta^2 \sum_{k=1}^{n} (|x_k(s) - x_k(t)|)^2/2\right) \\
&= \exp(\theta^2 \widetilde{d}(X_s, X_t)^2/2)
\end{aligned}$$

where in the final inequality we used Lemma 4.2.15. ⊠

As a consequence of Lemma 4.2.16, any Rademacher process satisfies Dudley's Inequality in Lemma 4.2.13.

4.2.5 Covering Number Bounds

In this section, we provide a series of bounds on the covering number of various generic classes of sets. Of key importance is a bound on the covering number for a set of parameterized functions. First, we discuss a covering number bound on a subset of a unit ball as given in [1, 89].

Lemma 4.2.17 ([89], Proposition C.3 [1]). *For any norm $\|\cdot\|$ on $\mathbb{R}^n$ and subset $U \subseteq \{\boldsymbol{x} \in \mathbb{R}^n : \|\boldsymbol{x}\| \leq 1\}$ it holds that*

$$\mathcal{N}(U, \|\cdot\|, \epsilon) \leq (1 + 2/\epsilon)^n .$$

Proof. For a metric space (M, d) let $\mathcal{N}^{\text{pack}}(M, d, \epsilon)$ denote the ϵ-packing number. That is $\mathcal{N}^{\text{pack}}$ is the maximum integer number of points $\{\boldsymbol{x}_k\}_{k=1,2,\ldots,\mathcal{N}^{\text{pack}}} \subseteq M$ that are ϵ-separated, i.e. $d(\boldsymbol{x}_k, \boldsymbol{x}_j) > \epsilon$ for all $k, j \in \{1, 2, \ldots, \mathcal{N}^{\text{pack}}\}$ where $k \neq j$. Next, let $X = \{\boldsymbol{x}_k\}_{k=1,2,\ldots,\mathcal{N}^{\text{pack}}}$ be the maximal ϵ-packing points of (M, d) . Assume that X is not an ϵ-covering of M. Then there exists a $\boldsymbol{v} \in M$ such that $d(\boldsymbol{x}_k, \boldsymbol{v}) > \epsilon$ for all $k \in \{1, 2, \ldots, \mathcal{N}^{\text{pack}}\}$. Hence, $X \cup \{\boldsymbol{v}\}$ is an ϵ-packing of M, which contradicts that X is the maximum ϵ-packing. Therefore, every maximal ϵ-packing of M is an ϵ-covering of M and thus $\mathcal{N}(M, d, \epsilon) \leq \mathcal{N}^{\text{pack}}(M, d, \epsilon)$.

Now, let $X = \{\boldsymbol{x}_k\}_{k=1,2,\ldots,\mathcal{N}^{\text{pack}}}$ be the maximal ϵ-packing points of the metric space $(U, \|\cdot\|)$. Define the radius δ ball as $B_\delta(\boldsymbol{x}) = \{\boldsymbol{z} \in \mathbb{R}^n : \|\boldsymbol{x} - \boldsymbol{z}\| \leq \delta\}$ and let $\text{vol}(B)$ denote the standard volume of a set $B \subseteq \mathbb{R}^n$. By definition of an ϵ-packing, $B_{\epsilon/2}(\boldsymbol{x}_k) \cap B_{\epsilon/2}(\boldsymbol{x}_j) = \emptyset$ for all $k, j \in \{1, 2, \ldots, \mathcal{N}^{\text{pack}}\}$ where $k \neq j$. Furthermore, as $\text{vol}\big(B_{\epsilon/2}(\boldsymbol{x}_k)\big) = \text{vol}\left(B_{\epsilon/2}(\boldsymbol{0})\right)$ for every k, then $\text{vol}\Big(\bigcup_{k=1}^{\mathcal{N}^{\text{pack}}} B_{\epsilon/2}(\boldsymbol{x}_k)\Big) = \mathcal{N}^{\text{pack}}\text{vol}(B_{\epsilon/2}(\boldsymbol{0}))$. As every $\boldsymbol{x}_k \in U \subseteq B_1(\boldsymbol{0})$ then $\bigcup_{k=1}^{\mathcal{N}^{\text{pack}}} B_{\epsilon/2}(\boldsymbol{x}_k) \subseteq B_{1+\epsilon/2}(\boldsymbol{0})$ since $\boldsymbol{x}_k$ could be a boundary point of $B_1(\boldsymbol{0})$ where $B_{\epsilon/2}(\boldsymbol{x}_k)$ is then outside of $B_1(\boldsymbol{0})$ by a maximum distance of $\epsilon/2$. Therefore, it holds that $\mathcal{N}^{\text{pack}}\text{vol}(B_{\epsilon/2}(\boldsymbol{0})) = \text{vol}\left(\bigcup_{k=1}^{\mathcal{N}^{\text{pack}}} B_{\epsilon/2}(\boldsymbol{x}_k)\right) \leq \text{vol}(B_{1+\epsilon/2}(\boldsymbol{0}))$. On $\mathbb{R}^n$, volume satisfies a homogeneity relationships given by $\text{vol}(B_\delta(\boldsymbol{0})) = \delta^n \text{vol}(B_1(\boldsymbol{0}))$, which implies

$$\mathcal{N}^{\text{pack}}(\epsilon/2)^n \text{vol}(B_1(\boldsymbol{0})) = \mathcal{N}^{\text{pack}}\text{vol}(B_{\epsilon/2}(\boldsymbol{0})) \leq \text{vol}(B_{1+\epsilon/2}(\boldsymbol{0})) = (1 + \epsilon/2)^n \text{vol}(B_1(\boldsymbol{0}))$$

or $\mathcal{N}^{\text{pack}}(U, \|\cdot\|, \epsilon) \leq (1 + \epsilon/2)^n/(\epsilon/2)^n = (1 + 2/\epsilon)^n$. Therefore, $\mathcal{N}(M, d, \epsilon) \leq \mathcal{N}^{\text{pack}}(M, d, \epsilon) \leq (1 + 2/\epsilon)^n$. ⊠

Second, we provide a covering number bound for a space of parametric function satisfying a Lipschitz criterion.

Lemma 4.2.18 ([89])**.** *Let $(\Theta, \|\cdot\|_\vartheta)$ be a non-empty, bounded, finite-dimensional, and normed vector space. Define $\mathcal{F} = \{f_\theta : \mathcal{X} \to \mathcal{Y} \,|\, \theta \in \Theta\}$ and let $\|\cdot\|_\mathcal{F}$ be any norm on $\mathcal{F}$. Assume that for any $\theta, \widetilde{\theta} \in \Theta$*

$$\|f_\theta(\boldsymbol{x}) - f_{\widetilde{\theta}}(\boldsymbol{x})\|_\mathcal{F} \leq \Gamma(\boldsymbol{x})\|\theta - \widetilde{\theta}\|_\vartheta$$

and let $\Gamma = \sup_{\boldsymbol{x}\in\mathcal{X}} \Gamma(\boldsymbol{x})$. *Then*

$$\mathcal{N}(\mathcal{F}, \|\cdot\|_{\mathcal{F}}, \epsilon) \leq \mathcal{N}(\Theta, \|\cdot\|_{\vartheta}, \epsilon/\Gamma).$$

Proof. Let C_Θ be any ϵ/Γ-covering of Θ and $C_{\mathcal{F}} = \{f_\theta : \theta \in C_\Theta\}$. Note $|C_{\mathcal{F}}| \leq |C_\Theta|$ as it is possible for $\theta, \widetilde{\theta} \in C_\Theta$ with $\theta \neq \widetilde{\theta}$ to generate $f_\theta = f_{\widetilde{\theta}}$ and equality occurs if and only if every $\theta \in C_\Theta$ uniquely defines a function $f_\theta \in C_{\mathcal{F}}$. Further, by definition of an ϵ-covering, for any $f_\theta \in \mathcal{F}$ there exists a $c \in C_\Theta$ such that $\|\theta - c\| \leq \epsilon/\Gamma$. Thus $f_c \in C_{\mathcal{F}}$ satisfies

$$\|f_\theta(\boldsymbol{x}) - f_c(\boldsymbol{x})\|_{\mathcal{F}} \leq \Gamma(\boldsymbol{x})\|\theta - c\|_{\vartheta} \leq \Gamma\|\theta - c\|_{\vartheta} \leq \epsilon.$$

Hence, every ϵ/Γ-covering of Θ generates an ϵ-covering of $\mathcal{F}$. Take C_Θ to be the minimal cardinality ϵ/Γ-covering of Θ, then $\mathcal{N}(\mathcal{F}, \|\cdot\|_{\mathcal{F}}, \epsilon) \leq |C_{\mathcal{F}}| \leq |C_\Theta| = \mathcal{N}(\Theta, \|\cdot\|_{\vartheta}, \epsilon/\Gamma)$. ⊠

Finally, we generalize Lemma 4.2.18 to a class of functions parameterized by a sequence of parameters; the result of which is a direct consequence of Lemma 4.2.18 and likely exists in the literature.

Corollary 4.2.19. *Let* $(\Theta_i, \|\cdot\|_{\vartheta_i})$, *for* $i \in \mathbb{N}[n]$, *be a sequence of non-empty, bounded, finite-dimensional, and normed vector spaces. Define* $\mathcal{F} = \{f_{\theta_1,\ldots,\theta_n} : \mathcal{X} \to \mathcal{Y} \,|\, \theta_i \in \Theta_i\}$ *and let* $\|\cdot\|_{\mathcal{F}}$ *be a norm on* $\mathcal{F}$. *Assume that*

$$\|f_{\theta_1,\ldots,\theta_n}(\boldsymbol{x}) - f_{\widetilde{\theta}_1,\ldots,\widetilde{\theta}_n}(\boldsymbol{x})\|_{\mathcal{F}} \leq \sum_{i=1}^{n} \Gamma_i(\boldsymbol{x})\|\theta_i - \widetilde{\theta}_i\|_{\vartheta_i}$$

and let $\Gamma_i = \sup_{\boldsymbol{x}\in\mathcal{X}} \Gamma_i(\boldsymbol{x})$. *Then*

$$\mathcal{N}(\mathcal{F}, \|\cdot\|_{\mathcal{F}}, \epsilon) \leq \prod_{i=1}^{n} \mathcal{N}(\Theta_i, \|\cdot\|_{\vartheta_i}, \epsilon/(n\Gamma_i)).$$

Proof. Let (S_k, d_k), for $k \in \mathbb{N}[n]$, be metric spaces and define the product metric space $(\mathcal{S}, d)$ where $\mathcal{S} \coloneqq S_1 \times \cdots \times S_n$ and $d(\boldsymbol{s}, \boldsymbol{c}) \coloneqq \sum_{k=1}^{n} a_k d_k(s_k, c_k)$ for $\boldsymbol{s}, \boldsymbol{c} \in \mathcal{S}$ and fixed $a_k \in (0, \infty)$.

Let C_k be any $\epsilon/(a_k n)$-covering of S_k and define $\mathcal{C} \coloneqq C_1 \times \cdots \times C_n$. Then for any $\boldsymbol{s} \in \mathcal{S}$ there exists a $\boldsymbol{c} \in \mathcal{C}$ such that $d(\boldsymbol{s}, \boldsymbol{c}) = \sum_{k=1}^{n} a_k d_k(s_k, c_k) \leq \sum_{k=1}^{n} a_k \epsilon/(n a_k) = \epsilon$, which implies that $\mathcal{C}$ is an ϵ-covering of $\mathcal{S}$. Take every C_k to be the minimal cardinality $\epsilon/(a_k n)$-covering of S_k then

$$\mathcal{N}(\mathcal{S}, d, \epsilon) \leq |\mathcal{C}| = \prod_{k=1}^{n} |C_k| = \prod_{k=1}^{n} \mathcal{N}\left(S_k, d_k, \frac{\epsilon}{a_k n}\right). \tag{4.5}$$

Next, let $(\Theta, \|\cdot\|_\Theta)$ be the finite-dimensional and normed vector space where $\Theta = \Theta_1 \times \cdots \times \Theta_n$ and for $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_n) \in \Theta$ we define $\|\boldsymbol{\theta}\|_\Theta = \sum_{i=1}^{n} \Gamma_i \|\theta_i\|_{\vartheta_i}$. Then for any $\boldsymbol{\theta}, \widetilde{\boldsymbol{\theta}} \in \Theta$ where $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_n)$ and $\widetilde{\boldsymbol{\theta}} = (\widetilde{\theta}_1, \ldots, \widetilde{\theta}_n)$ we have

$$\|f_{\theta_1,\ldots,\theta_n}(\boldsymbol{x}) - f_{\widetilde{\theta}_1,\ldots,\theta_n}(\boldsymbol{x})\|_{\mathcal{F}} \leq \sum_{i=1}^{n} \Gamma_i(\boldsymbol{x}) \|\theta_i - \widetilde{\theta}_i\|_{\vartheta_i} \leq \sum_{i=1}^{n} \Gamma_i \|\theta_i - \widetilde{\theta}_i\|_{\vartheta_i} = \|\boldsymbol{\theta} - \widetilde{\boldsymbol{\theta}}\|_\Theta.$$

Thus, by Lemma 4.2.18

$$\mathcal{N}(\mathcal{F}, \|\cdot\|_{\mathcal{F}}, \epsilon) \leq \mathcal{N}(\Theta, \|\cdot\|_\Theta, \epsilon)$$

As $(\Theta, \|\cdot\|_\Theta)$ is a product space then applying equation (4.5) gives

$$\mathcal{N}(\mathcal{F}, \|\cdot\|_{\mathcal{F}}, \epsilon) \leq \mathcal{N}(\Theta, \|\cdot\|_\Theta, \epsilon) \leq \prod_{i=1}^{n} \mathcal{N}\left(\Theta_i, \|\cdot\|_{\vartheta_i}, \epsilon/(n\Gamma_i)\right)$$

producing the desired result. ⊠

4.2.6 Properties of Bounding Functions

In this section, we provide a handful of functions, with a corresponding Lipschitz property, that map an input vector onto a bounded space. These functions and their Lipschitz properties are beneficial in analyzing the GEBs for unrolled, CG-based DNNs and all likely exist in the literature.

Mapping to an Approximately Normalized Vector

Definition 4.2.20. *Let $\mathcal{B} = \{\boldsymbol{x} \in \mathbb{R}^n : ||\boldsymbol{x}||_2 < 1\}$ be the open unit ball. For fixed $\epsilon > 0$, the **approximately normalized vector mapping** $f_\epsilon : \mathbb{R}^n \to \mathcal{B}$ is defined as*

$$f_\epsilon(\boldsymbol{x}) = \frac{\boldsymbol{x}}{||\boldsymbol{x}||_2 + \epsilon}.$$

Now, we prove a Lipschitz property of the approximately normalized vector mapping.

Lemma 4.2.21. *The approximately normalized vector mapping satisfies*

$$||f_\epsilon(\boldsymbol{x}_1) - f_\epsilon(\boldsymbol{x}_2)||_2 \leq \frac{1}{\epsilon}||\boldsymbol{x}_1 - \boldsymbol{x}_2||_2$$

for all $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{R}^n$ and every $\epsilon > 0$.

Proof. First recall that for any differentiable function $f : \mathbb{R}^n \to \mathbb{R}^m$

$$||f(\boldsymbol{x}_1) - f(\boldsymbol{x}_2)||_2 \leq L||\boldsymbol{x}_1 - \boldsymbol{x}_2||_2 \text{ for all } \boldsymbol{x}_1, \boldsymbol{x}_2 \quad \iff \quad \sup_{\boldsymbol{x}} ||\nabla f(\boldsymbol{x})||_{2\to 2} \leq L.$$

Note

$$\nabla f(\boldsymbol{x}) = \frac{1}{||\boldsymbol{x}||_2 + \epsilon} I - \frac{1}{(||\boldsymbol{x}||_2 + \epsilon)^2 ||\boldsymbol{x}||_2} \boldsymbol{x}\boldsymbol{x}^T.$$

Let $(\lambda_i(\boldsymbol{x}), \boldsymbol{v}_i(\boldsymbol{x}))$ be the eigenpairs of $\boldsymbol{x}\boldsymbol{x}^T$. As the outer product $\boldsymbol{x}\boldsymbol{x}^T$ is a rank 1 matrix then $n-1$ of the eigenvalues are zero, which we assume WLOG that $\lambda_2(\boldsymbol{x}) = \cdots = \lambda_n(\boldsymbol{x}) = 0$. Observe

$$\boldsymbol{x}^T ||\boldsymbol{x}||_2^2 \boldsymbol{v}_1(\boldsymbol{x}) = ||\boldsymbol{x}||_2^2 \boldsymbol{x}^T \boldsymbol{v}_1(\boldsymbol{x}) = \boldsymbol{x}^T \boldsymbol{x}\boldsymbol{x}^T \boldsymbol{v}_1(\boldsymbol{x}) = \boldsymbol{x}^T \lambda_1(\boldsymbol{x}) \boldsymbol{v}_1(\boldsymbol{x})$$

implying that $\lambda_1(\boldsymbol{x}) = ||\boldsymbol{x}||_2^2$. Also observe

$$\nabla f(\boldsymbol{x}) \boldsymbol{v}_i(\boldsymbol{x}) = \left(\frac{1}{||\boldsymbol{x}||_2 + \epsilon} - \frac{1}{(||\boldsymbol{x}||_2 + \epsilon)^2 ||\boldsymbol{x}||_2} \lambda_i(\boldsymbol{x}) \right) \boldsymbol{v}_i(\boldsymbol{x}),$$

implying that

$$\begin{aligned}&\text{eigenpairs}(\nabla f(\boldsymbol{x}))\\&= \left\{\left(\frac{1}{||\boldsymbol{x}||_2+\epsilon} - \frac{||\boldsymbol{x}||_2^2}{(||\boldsymbol{x}||_2+\epsilon)^2||\boldsymbol{x}||_2}, \boldsymbol{v}_1(\boldsymbol{x})\right)\right\} \bigcup \left\{\left(\left(\frac{1}{||\boldsymbol{x}||_2+\epsilon}, \boldsymbol{v}_i(\boldsymbol{x})\right)\right)\right\}_{i=2}^{n}.\end{aligned}$$

Note

$$\frac{1}{||\boldsymbol{x}||_2+\epsilon} - \frac{||\boldsymbol{x}||_2^2}{(||\boldsymbol{x}||_2+\epsilon)^2||\boldsymbol{x}||_2} = \frac{1}{||\boldsymbol{x}||_2+\epsilon}\left(1 - \frac{||\boldsymbol{x}||_2}{||\boldsymbol{x}||_2+\epsilon}\right) \leq \frac{1}{||\boldsymbol{x}||_2+\epsilon}$$

where the last inequality holds since $0 \leq \frac{||\boldsymbol{x}||_2}{||\boldsymbol{x}||_2+\epsilon} \leq 1$ implying $0 \leq 1 - \frac{||\boldsymbol{x}||_2}{||\boldsymbol{x}||_2+\epsilon} \leq 1$. Therefore

$$\sup_{\boldsymbol{x}} ||\nabla f(\boldsymbol{x})||_{2\to 2} \leq \sup_{\boldsymbol{x}} \frac{1}{||\boldsymbol{x}||_2+\epsilon} = \frac{1}{\epsilon}$$

and thus, the approximately normalized vector mapping is $\frac{1}{\epsilon}$-Lipschitz. ⊠

Mapping to Ball of Radius B

Definition 4.2.22. *Let $\mathcal{B}_B = \{\boldsymbol{x} \in \mathbb{R}^n : ||\boldsymbol{x}||_2 \leq B\}$ be the ball of radius B. For fixed $B > 0$, the B-**Ball mapping** $\rho_B : \mathbb{R}^n \to \mathcal{B}_B$ is defined as*

$$\rho_B(\boldsymbol{x}) = \begin{cases}\boldsymbol{x} & ||\boldsymbol{x}||_2 \leq B\\ B\frac{\boldsymbol{x}}{||\boldsymbol{x}||_2} & ||\boldsymbol{x}||_2 > B\end{cases} = \frac{\boldsymbol{x}}{\max\{1, ||\boldsymbol{x}||_2/B\}}.$$

We remark that the B-Ball mapping is the projection onto the Euclidean ball of radius B. Next, we show that the B-Ball mapping is 1-Lipschitz.

Lemma 4.2.23. *The B-Ball mapping satisfies*

$$||\rho_B(\boldsymbol{x}_1) - \rho_B(\boldsymbol{x}_2)||_2 \leq ||\boldsymbol{x}_1 - \boldsymbol{x}_2||_2$$

for every $B > 0$ and $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{R}^n$.

Proof. This is trivially true for $\boldsymbol{x}_1, \boldsymbol{x}_2$ satisfying $||\boldsymbol{x}_1||, ||\boldsymbol{x}_2|| \leq B$. Following the exact same steps as in the proof of Lemma 4.2.21 shows the result holds for $\boldsymbol{x}_1, \boldsymbol{x}_2$ satisfying $||\boldsymbol{x}_1||, ||\boldsymbol{x}_2|| > B$. Now let $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ satisfy $||\boldsymbol{x}_1|| \leq B$ and $||\boldsymbol{x}_2|| > B$. Without loss of generality we assume $\boldsymbol{x}_1 = \mathbf{0}$. Then

$$||\rho_B(\boldsymbol{x}_1) - \rho_B(\boldsymbol{x}_2)||_2 = \left|\left|\frac{B}{||\boldsymbol{x}_2||_2}\boldsymbol{x}_2\right|\right|_2 \leq ||\boldsymbol{x}_2||_2 = ||\boldsymbol{x}_1 - \boldsymbol{x}_2||_2$$

proving the desired result. ⊠

Lastly, we remark that the B-Ball mapping is bounded in norm by B as

$$\|\rho_B(\boldsymbol{x})\|_2 = \left\|\frac{\boldsymbol{x}}{\max\{1, \|\boldsymbol{x}\|_2/B\}}\right\|_2 \leq B. \tag{4.6}$$

Componentwise Mapping to Ball of Radius B

Definition 4.2.24. *Let $\mathcal{B}_b = \{\boldsymbol{x} \in \mathbb{R}^n : ||\boldsymbol{x}||_2 \leq b\}$ be the Euclidean ball of radius b. For fixed $B > 0$, the **componentwise $\sqrt{n}B$-ball mapping** $f_B : \mathbb{R} \to [-B, B]$ is a componentwise function defined as*

$$f_B(x) = \begin{cases} x & |x| \leq B \\ B\frac{x}{|x|} = B\text{sign}(x) & x > B. \end{cases}$$

Then for any vector $\boldsymbol{x} \in \mathbb{R}^k$ we define $f_B(\boldsymbol{x}) = [f_B(x_i)]_{i=1}^n : \mathbb{R}^n \to \mathcal{B}_{\sqrt{n}B}$.

To show the componentwise ball mapping is Lipschitz we first show that any componentwise function that is L-Lipschitz over $\mathbb{R}$ is similarly L-Lipschitz over $\mathbb{R}^n$.

Lemma 4.2.25. *Let $f : \mathbb{R} \to \mathbb{R}$ be a componentwise function that is L-Lipschitz. Then $f(\boldsymbol{x}) = [f(x_i)]_{i=1}^n : \mathbb{R}^n \to \mathbb{R}^n$ is L-Lipschitz over $\mathbb{R}^n$.*

Proof. Observe

$$||f(\boldsymbol{x}_1) - f(\boldsymbol{x}_2)||_2 = \left(\sum_{i=1}^{n}(f(x_{1i}) - f(x_{2i}))^2\right)^{1/2} = \left(\sum_{i=1}^{n}(|f(x_{1i}) - f(x_{2i})|)^2\right)^{1/2}$$
$$\leq \left(\sum_{i=1}^{n}(L\,|x_{1i} - x_{2i}|)^2\right)^{1/2}$$
$$= L||\boldsymbol{x}_1 - \boldsymbol{x}_2||_2. \qquad \boxtimes$$

Now, we show that the componentwise ball mapping is 1-Lipschitz.

Lemma 4.2.26. *The componentwise $\sqrt{n}B$-ball mapping satisfies*

$$||f_B(\boldsymbol{x}_1) - f_B(\boldsymbol{x}_2)||_2 \leq ||\boldsymbol{x}_1 - \boldsymbol{x}_2||_2$$

for every $B > 0$ and $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{R}^n$.

Proof. Note

$$\frac{d}{dx}f_B(x) = \begin{cases} 1 & |x| \leq B \\ 0 & \text{else} \end{cases}$$

implying that $f_B(x)$ is 1-Lipschitz over $\mathbb{R}$. Applying Lemma 4.2.25 produces the desired result. $\boxtimes$

Modified ReLU Activation Function

Recall the ReLU activation function is a componentwise function defined as $\text{ReLU}(x) = \max\{0, x\}$.

Definition 4.2.27. *Let $a, b \in \mathbb{R}$. The **modified ReLU** (mReLU) activation function, $\mathcal{P}_{a,b}(x) : \mathbb{R} \to \mathbb{R}$, is a componentwise function defined as*

$$\mathcal{P}_{a,b}(x) = a + \text{ReLU}(x - a) - \text{ReLU}(x - b).$$

We remark that the mReLU activation function projects an input $x \in \mathbb{R}$ onto the interval $[\min\{a, b\}, \max\{a, b\}]$. Now we show that the mReLU activation function is 1-Lipschitz.

Lemma 4.2.28. *The modified ReLU activation function satisfies*

$$\|\mathcal{P}_{a,b}(\boldsymbol{x}_1) - \mathcal{P}_{a,b}(\boldsymbol{x}_2)\|_2 \leq \|\boldsymbol{x}_1 - \boldsymbol{x}_2\|$$

for every $a, b \in \mathbb{R}$ *and* $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{R}$.

Proof. We show the real-valued modified ReLU function is $1-$Lipschitz and then apply Lemma 4.2.25. For this we consider cases for the locations of inputs x_1 and x_2 with respect to the interval $[\min\{a, b\}, \max\{a, b\}]$.

Case 1: Let $x_1, x_2 \in [\min\{a, b\}, \max\{a, b\}]$. If $a \leq b$ then

$$\begin{aligned}
|\mathcal{P}_{a,b}(x_1) - \mathcal{P}_{a,b}(x_2)| &= |\text{ReLU}(x_1 - a) - \text{ReLU}(x_1 - b) - (\text{ReLU}(x_2 - a) - \text{ReLU}(x_2 - b))| \\
&= |x_1 - a - (x_2 - a)| \\
&= |x_1 - x_2|.
\end{aligned}$$

Instead, if $b \leq a$

$$\begin{aligned}
|\mathcal{P}_{a,b}(x_1) - \mathcal{P}_{a,b}(x_2)| &= |\text{ReLU}(x_1 - a) - \text{ReLU}(x_1 - b) - (\text{ReLU}(x_2 - a) - \text{ReLU}(x_2 - b))| \\
&= |x_2 - b - (x_1 - b)| \\
&= |x_1 - x_2|.
\end{aligned}$$

Case 2: Let $x_1 \in [\min\{a, b\}, \max\{a, b\}]$ and $x_2 \notin [\min\{a, b\}, \max\{a, b\}]$. If $a \leq b$ and $x_2 > b$ then

$$|\mathcal{P}_{a,b}(x_1) - \mathcal{P}_{a,b}(x_2)| = |x_1 - a - (x_2 - a - (x_2 - b))| = |x_1 - b| < |x_1 - x_2|.$$

Similarly, if $a \leq b$ and $x_2 < a$ then

$$|\mathcal{P}_{a,b}(x_1) - \mathcal{P}_{a,b}(x_2)| = |x_1 - a| < |x_1 - x_2|.$$

Instead, if $b \leq a$ and $x_2 > a$ then

$$|\mathcal{P}_{a,b}(x_1) - \mathcal{P}_{a,b}(x_2)| = |-(x_1 - b) - (x_2 - a - (x_2 - b))| = |x_1 - a| < |x_1 - x_2|.$$

Similarly, if $b \leq a$ and $x_2 < b$ then

$$|\mathcal{P}_{a,b}(x_1) - \mathcal{P}_{a,b}(x_2)| = |x_1 - b| < |x_1 - x_2|.$$

Case 3: Let $x_2 \in [\min\{a, b\}, \max\{a, b\}]$ and $x_1 \notin [\min\{a, b\}, \max\{a, b\}]$. This case follows the exact same manner as Case 2.

Case 4: Let $x_1, x_2 \notin [\min\{a, b\}, \max\{a, b\}]$. If x_1 and x_2 are both left or both right of the interval $[\min\{a, b\}, \max\{a, b\}]$ then $|\mathcal{P}_{a,b}(x_1) - \mathcal{P}_{a,b}(x_2)| = 0 \leq |x_1 - x_2|$. Now, let x_1 and x_2 be on opposite sides of the interval $[\min\{a, b\}, \max\{a, b\}]$. Then $|\mathcal{P}_{a,b}(x_1) - \mathcal{P}_{a,b}(x_2)| \leq |a - b| \leq |x_1 - x_2|$.

As x_1 and x_2 fall into one of the four cases above for any $x_1, x_2 \in \mathbb{R}$, then $\mathcal{P}_{a,b}(x)$ is a 1-Lipschitz function over $\mathbb{R}$. Therefore, by Lemma 4.2.25 the modified ReLU function $\mathcal{P}_{a,b}(\boldsymbol{x})$ is 1-Lipschitz over $\mathbb{R}^n$. ⊠

4.2.7 Integral Bounds

We conclude the preliminaries section with bounds on certain integrals that will manifest in the derivation of generalization error bounds for compound Gaussian networks. First, a bound on the integral of a function of the form $\ln(1 + 1/x)$, which exists in the literature [86].

Lemma 4.2.29 ([86]). *For all $\nu, \beta > 0$*

$$\int_0^\beta \sqrt{\ln(1 + \nu/x)}dx \leq \beta\sqrt{\ln(e(1 + \nu/\beta))}. \tag{4.7}$$

Proof. Let $\mathcal{L}^2$ be the inner product space of square-integrable real-valued functions on $[0, \beta]$ for $\beta > 0$. Then for $f, g \in \mathcal{L}^2$ define the inner product $\langle \cdot, \cdot \rangle : \mathcal{L}^2 \times \mathcal{L}^2 \to \mathbb{R}$ as $\langle f, g \rangle = \int_0^\beta f(x)g(x)dx$.

By the Cauchy-Schwartz inequality

$$\left|\int_0^\beta f(x)g(x)dx\right| = \langle f, g\rangle \leq \sqrt{\langle f, f\rangle\langle g, g\rangle} = \sqrt{\int_0^\beta f(x)^2dx \int_0^\beta g(x)^2dx}.$$

Take $f(x) \equiv 1$ and $g(x) = \sqrt{\ln(1+\nu/x)}$ for $\nu > 0$. Then

$$\begin{aligned}\int_0^\beta \sqrt{\ln(1+\nu/x)}dx &= \left|\int_0^\beta \sqrt{\ln(1+\nu/x)}dx\right| \\ &\leq \sqrt{\int_0^\beta 1dx \int_0^\beta \ln(1+\nu/x)dx} = \sqrt{\beta\int_0^\beta \ln(1+\nu/x)dx}. \end{aligned} \tag{4.8}$$

Take $u = \nu/x$ then $x = \nu/u$, $dx = -\nu/u^2du$, and $u \to \infty$ when $x \to 0$ and $u = \nu/\beta$ when $x = \beta$. Thus

$$\int_0^\beta \ln(1+\nu/x)dx = \int_\infty^{\nu/\beta} -\frac{\nu}{u^2}\ln(1+u)du = \nu\int_{\nu/\beta}^\infty \frac{1}{u^2}\ln(1+u)du.$$

Using integration by parts

$$\begin{aligned}\int_0^\beta \ln(1+\nu/x)dx &= \nu\int_{\nu/\beta}^\infty \frac{1}{u^2}\ln(1+u)du \\ &= \nu\left[-u^{-1}\ln(1+u)\Big|_{\nu/\beta}^\infty - \int_{\nu/\beta}^\infty -\frac{1}{u}\frac{1}{1+u}\right] \\ &= \nu\left[\frac{\beta}{\nu}\ln(1+\nu/\beta) + \int_{\nu/\beta}^\infty \frac{1}{u}\frac{1}{1+u}du\right] \\ &= \beta\ln(1+\nu/\beta) + \nu\int_{\nu/\beta}^\infty \frac{1}{u}\frac{1}{1+u}du \end{aligned} \tag{4.9}$$

where in the third equality we used that $\lim_{u\to\infty} u^{-1}\ln(1+u) = 0$. Since $\frac{1}{1+x} \leq \frac{1}{x}$ for all $x > 0$ then

$$\int_{\nu/\beta}^\infty \frac{1}{u}\frac{1}{1+u}du \leq \int_{\nu/\beta}^\infty \frac{1}{u^2}du = \frac{\beta}{\nu}. \tag{4.10}$$

Thus

$$\int_0^\beta \ln(1+\nu/x)dx \le \beta\ln(1+\nu/\beta) + \nu\frac{\beta}{\nu} = \beta(1+\ln(1+\nu/\beta)) = \beta\ln(e(1+\nu/\beta)),$$

which combined with (4.8) produces

$$\int_0^\beta \sqrt{\ln(1+\nu/x)}dx \le \sqrt{\beta^2\ln(e(1+\nu/\beta))} = \beta\sqrt{\ln(e(1+\nu/\beta))},$$

which holds for all $\nu, \beta > 0$. ⊠

Second, we improve the bound of Lemma 4.2.29, which likely already exists in the literature.

Lemma 4.2.30. *For all* $\nu, \beta > 0$

$$\int_0^\beta \sqrt{\ln(1+\nu/x)}dx \le \sqrt{\beta^2\ln(1+\nu/\beta) + \nu\beta\ln(1+\beta/\nu)}. \tag{4.11}$$

Proof. Observe

$$\begin{aligned}\int_{\nu/\beta}^\infty \frac{1}{u}\frac{1}{1+u}du = \ln\left(\frac{u}{1+u}\right)\Big|_{\nu/\beta}^\infty &= -\ln\left(\frac{\nu/\beta}{1+\nu/\beta}\right)\\ &= -\ln\left(\frac{\nu}{\beta+\nu}\right) = -\ln\left(\frac{1}{1+\beta/\nu}\right) = \ln\left(1+\beta/\nu\right).\end{aligned}$$

Combining with (4.9) gives

$$\int_0^\beta \ln(1+\nu/x)dx = \beta\ln(1+\nu/\beta) + \nu\ln(1+\beta/\nu),$$

which combined with (4.8) produces for all $\nu, \beta > 0$

$$\int_0^\beta \sqrt{\ln(1+\nu/x)}dx \le \sqrt{\beta^2\ln(1+\nu/\beta) + \nu\beta\ln(1+\beta/\nu)}.$$ ⊠

4.3 Generalized Compound Gaussian Network

In this section, we provide the developmental setup for a generalized compound Gaussian network (G-CG-Net) shown in Fig. 4.1. G-CG-Net is an unrolled, CG-based DNN for solving linear inverse problems where the use of “generalized" in the naming of G-CG-Net denotes the fact that this network encompasses the compound Gaussian network (CG-Net) [57, 58] and deep regularized compound Gaussian network (DR-CG-Net) [76, 77] as special cases. Recall that CG-Net and DR-CG-Net are detailed in Chapter 2 and Chapter 3, respectively. In contrast, the use of “generalized" in terms of network error, i.e. generalization error bound, denotes a DNNs ability to transfer from training data to testing data. In developing G-CG-Net, we first establish a CG-based iterative algorithm for solving linear inverse problems, named generalized compound Gaussian least squares (G-CG-LS). Subsequently, we apply algorithm unrolling to G-CG-LS to produce G-CG-Net.

Through the study of image statistics, it has been shown that sparsity coefficients of natural images exhibit self-similarity, heavy-tailed marginal distributions, and self-reinforcement among local coefficients [11]. These properties are encompassed by the class of CG densities [10, 11, 43]. Thus, the CG prior better captures statistical properties of natural images as well as images from other modalities such as radar [12, 92]. A useful formulation of the CG prior is modeling a signal as the Hadamard product $\boldsymbol{c} = \boldsymbol{z} \odot \boldsymbol{u}$ such that $\boldsymbol{u} \sim \mathcal{N}(\boldsymbol{0}, \Sigma_u)$, $\boldsymbol{z} \sim p_{\boldsymbol{z}}$, and $\boldsymbol{u}$ and $\boldsymbol{z}$ are independent random variables [8, 11]. We call $\boldsymbol{z}$ the scale variable and $\boldsymbol{u}$ the Gaussian variable. The linear measurement model we consider then is

$$\boldsymbol{y} = A\boldsymbol{c} + \boldsymbol{\nu} \equiv A(\boldsymbol{z} \odot \boldsymbol{u}) + \boldsymbol{\nu}. \tag{4.12}$$

G-CG-Net is a method that recovers $\boldsymbol{c}$, by estimating $\boldsymbol{z}$ and $\boldsymbol{u}$, when given $\boldsymbol{y}$ and A.

4.3.1 Iterative Algorithm

Algorithm 5 provides pseudocode for the iterative algorithm generalized compound Gaussian least squares (G-CG-LS) to be unrolled into G-CG-Net. While G-CG-LS was first introduced in

Chapter 3, we briefly restate it here for completeness of this chapter. Consider the cost function

$$F(\boldsymbol{u}, \boldsymbol{z}) = \frac{1}{2}\|\boldsymbol{y} - A(\boldsymbol{z} \odot \boldsymbol{u})\|_2^2 + \frac{1}{2}\boldsymbol{u}^T P_u^{-1}\boldsymbol{u} + \mathcal{R}(\boldsymbol{z}). \tag{4.13}$$

Note that a maximum-a-posteriori (MAP) estimate of $\boldsymbol{z}$ and $\boldsymbol{u}$ from (4.12) is a special case of (4.13) when $P_u \propto \Sigma_u$ and $\mathcal{R} \propto \log(p_{\boldsymbol{z}}(\boldsymbol{z}))$.

Using block coordinate descent [59], G-CG-LS alternatively minimizes (4.13) over $\boldsymbol{z}$ and $\boldsymbol{u}$. The minimum of (4.13) in $\boldsymbol{u}$ is a Tikhonov solution, $\mathcal{T}_{\boldsymbol{y}}(\boldsymbol{z}) \equiv \mathcal{T}_{\boldsymbol{y}}(\boldsymbol{z}; P_u)$, given by

$$\mathcal{T}_{\boldsymbol{y}}(\boldsymbol{z}) \coloneqq (A_{\boldsymbol{z}}^T A_{\boldsymbol{z}} + P_u^{-1})^{-1} A_{\boldsymbol{z}}^T \boldsymbol{y} = P_u A_{\boldsymbol{z}}^T (I + A_{\boldsymbol{z}} P_u A_{\boldsymbol{z}}^T)^{-1} \boldsymbol{y}$$

where the second equality results from using the Woodbury matrix identity.

As the minimum of (4.13) in $\boldsymbol{z}$ does not have a closed form solution, for general regularization $\mathcal{R}(\boldsymbol{z})$, we iteratively minimize by performing J descent steps on $\boldsymbol{z}$. That is, for $g : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n$ a scale variable descent update, the estimate $\boldsymbol{z}$ on descent step j of iteration k, denoted as $\boldsymbol{z}_k^{(j)}$ is given by $\boldsymbol{z}_k^{(j)} = g(\boldsymbol{z}_k^{(j-1)}, \boldsymbol{u}_{k-1}; \boldsymbol{y})$ where $\boldsymbol{z}_{k+1}^{(0)} = \boldsymbol{z}_k^{(J)}$. For instance, in Chapter 2 and [57, 58, 76] take g as a steepest descent step. Instead, in Chapter 3 and [77] take g as an iterative shrinkage and thresholding step and as a projected gradient descent step.

Algorithm 5 Generalized Compound Gaussian Least Squares (G-CG-LS)

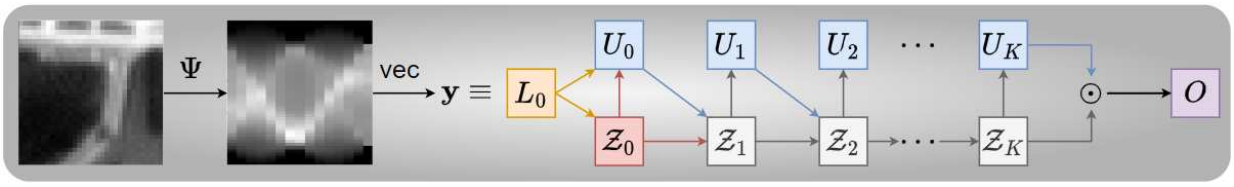
Input: Maximum number of iterations K, number of steepest descent steps J, measurement operator A, (possibly scaled) measurement $\boldsymbol{y}$, initial estimate $\boldsymbol{z}_1^{(0)}$.

Choose a $\boldsymbol{u}_0$ (e.g. $\boldsymbol{u}_0 = \mathcal{T}_{\boldsymbol{y}}(\boldsymbol{z}_1^{(0)})$)
for $k \in \{1, 2, \ldots, K\}$ **do**
    $\boldsymbol{z}$ ESTIMATION:
    for $j \in \{1, 2, \ldots, J\}$ **do**
        $\boldsymbol{z}_k^{(j)} = g(\boldsymbol{z}_k^{(j-1)}, \boldsymbol{u}_{k-1}) \equiv g(\boldsymbol{z}_k^{(j-1)}, \boldsymbol{u}_{k-1}; \boldsymbol{y})$
    end for
    $\boldsymbol{z}_{k+1}^{(0)} = \boldsymbol{z}_k^{(J)}$
    $\boldsymbol{u}$ ESTIMATION:
    $\boldsymbol{u}_k = \mathcal{T}_{\boldsymbol{y}}(\boldsymbol{z}_k^{(J)})$
end for

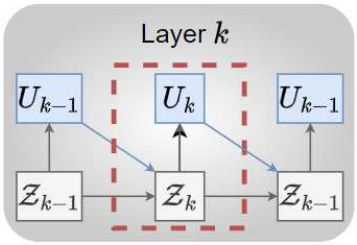
Output: $\boldsymbol{c}^* = \boldsymbol{z}_K^{(J)} \odot \boldsymbol{u}_K$

4.3.2 Unrolled Deep Neural Network (G-CG-Net)

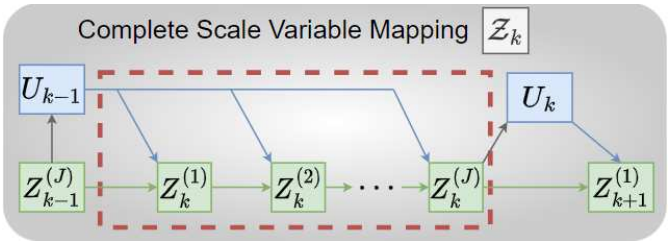
Applying algorithm unrolling to Algorithm 5, we create G-CG-Net with end-to-end structure shown in Fig. 4.1. G-CG-Net is a generalization on CG-Net introduced in Chapter 2 and DR-CG-Net introduced in Chapter 3; reducing to each with under specific choices of the scale variable descent update and $\boldsymbol{u}$ covariance matrix structure. Each layer k of G-CG-Net, shown in the dashed box of Fig. 4.1b, corresponds to iteration k of Algorithm 5 and implements a complete scale variable mapping, $\mathcal{Z}_k$ shown in Fig. 4.1c, that updates $\boldsymbol{z}$ and a Tikhonov update of $\boldsymbol{u}$. Each $\mathcal{Z}_k$ consists of J scale variable updates $Z_k^{(1)}, \ldots, Z_k^{(J)}$ where every $Z_k^{(j)}$ updates $\boldsymbol{z}$ once and is the output of a scale variable descent update, denoted $g_k^{(j)}$, as given in line 5 of Algorithm 5.



(a) End-to-end network structure of G-CG-Net.



(b) Layer k analogous to iteration k in Algorithm 5.



(c) Complete scale mapping module $\mathcal{Z}_k$ producing estimate $\boldsymbol{z}_k^{(J)}$ in Algorithm 5.

Figure 4.1: End-to-end network structure for G-CG-Net, the unrolled deep neural network of Algorithm 5, is shown in (4.1a). G-CG-Net consists of an input block, L_0, initialization block, $\mathcal{Z}_0$, $K+1$ Tikhonov blocks, U_k, output block, O, and K complete scale variable mappings, $\mathcal{Z}_k$, with structure in (4.1c). Each $\mathcal{Z}_k$ consists of J scale variable updates $Z_k^{(j)}$.

Mathematically detailing the G-CG-Net blocks we have:

$L_0 = \boldsymbol{y}$ is the input measurements to the network

$\mathcal{Z}_0 = \mathcal{P}_{0,b}(\hat{A}^T\boldsymbol{y})$, for $\hat{A} = \frac{A}{\|A\|_2}$, is an initial estimate of $\boldsymbol{z}$.

$U_k = \mathcal{T}_{\boldsymbol{y}}(Z_k^{(J)})$ is the Tikhonov estimate of $\boldsymbol{u}$ corresponding to line 1 and 9 of Algorithm 5.

The kth complete scale variable mapping, $\mathcal{Z}_k$, contains:

$Z_k^{(j)} = \mathcal{P}_{0,z_\infty}(g_k^{(j)}(Z_k^{(j-1)}, U_{k-1}))$ is the scale variable update corresponding to line 5 in Algorithm 5.

$O = \rho_{c_{\max}}(\mathcal{Z}_K \odot U_K)$ is the estimated signal output. produced by G-CG-Net.

Note that $\rho_{c_{\max}}$ and $\mathcal{P}_{0,z_\infty}$, for $c_{\max} \geq 0$ and $z_\infty \geq 0$, are applied for technical reasons in developing the GEB and are further discussed in Section 4.4.1. Furthermore, to simplify notation, we let $Z_{k+1}^{(0)} = Z_k^{(J)}$ for $k \in \{1, 2, \ldots, K-1\}$ and $Z_1^{(0)} = \mathcal{Z}_0$.

Every layer U_k is parameterized by the same covariance matrix P_u and each scale variable update $Z_k^{(j)}$ we assume, generally, to be parameterized by D arbitrary weights $\boldsymbol{\theta}_{k,1}^{(j)}, \ldots, \boldsymbol{\theta}_{k,D}^{(j)}$. We additionally assume that $\boldsymbol{\theta}_{k,d}^{(j)} \in \Omega_d$ for $k \in \mathbb{N}[K]$, $j \in \mathbb{N}[J]$, and some finite-dimensional vector space Ω_d. For instance, if a fully connected layer, mapping from $\mathbb{R}^n \to \mathbb{R}^n$, is implemented in each $g_k^{(j)}$, then every $\boldsymbol{\theta}_{k,1}^{(j)}$ could be the weight matrix from the layer while $\boldsymbol{\theta}_{k,2}^{(j)}$ could be the additive bias of the layer. Hence, the G-CG-Net parameters are

$$\boldsymbol{\Theta} = \{P_u\} \cup \{\boldsymbol{\theta}_{k,d}^{(j)}\}_{k \in \mathbb{N}[K],\ d \in \mathbb{N}[D].}^{j \in \mathbb{N}[J]} \tag{4.14}$$

We remark that a structured form may be imposed on the covariance matrix P_u. In particular, to ensure that P_u is symmetric and positive definite (SPD), we consider, for $\epsilon > 0$ a small fixed real number,

$$P_u = \begin{cases} \max\{\lambda, \epsilon\} I & \text{Scaled Identity} \\ \text{diag}([\max\{\lambda_i, \epsilon\}]_{i=1}^n) & \text{Diagonal} \\ L_{\text{tri}} L_{\text{tri}}^T + \epsilon I & \text{Tridiagonal} \\ LL^T + \epsilon I & \text{Full.} \end{cases}$$

In the scaled identity case, only a constant λ is learned. In the diagonal case, a vector $\boldsymbol{\lambda} = [\lambda_i]_{i=1}^n$ is learned. In the tridiagonal case, two vectors $\boldsymbol{\lambda}_1 \in \mathbb{R}^n$ and $\boldsymbol{\lambda}_2 \in \mathbb{R}^{n-1}$ are learned such that the lower triangular matrix component L_{tri} is formed by placing $\boldsymbol{\lambda}_1$ on the diagonal and $\boldsymbol{\lambda}_2$ on the first subdiagonal. Finally, in the case of a full covariance matrix, an entire lower triangular matrix L is learned.

4.3.3 Realizations

Two specific forms of G-CG-Net are detailed here, namely CG-Net from Chapter 2 and DR-CG-Net from Chapter 3. Note that we slightly adjust a few details of the implementation of CG-Net and DR-CG-Net to assist in the generalization error bound analysis.

Compound Gaussian Network (CG-Net) [57]

For this method, the scale variable is formulated as $\boldsymbol{z} = h(\boldsymbol{\chi})$ for $\boldsymbol{\chi} \sim \mathcal{N}(\mathbf{0}, I)$ and h is a componentwise, non-linear, twice continuously differentiable, and invertible function. Accordingly, the scale variable regularization is set as $\mathcal{R}(\boldsymbol{z}) = \mu \|h^{-1}(\boldsymbol{z})\|_2^2$, for a constant $\mu > 0$, to enforce normality of $\boldsymbol{\chi} = h^{-1}(\boldsymbol{z})$.

The scale variable descent update, g, is a projected steepest descent step based on a learned quadratic norm. We consider a slightly adjusted update, to assist in the analysis, given as

$$g_k^{(j)}(\boldsymbol{z}, \boldsymbol{u}) = \mathcal{P}_{a,b}(\boldsymbol{z} - B_k^{(j)} \rho_\xi(\nabla_{\boldsymbol{z}} F(\boldsymbol{u}, \boldsymbol{z}; \mu_k^{(j)}))) \tag{4.15}$$

where $a, b, \xi > 0$ are fixed, $B_k^{(j)}$ is a learned $n \times n$ positive definite matrix, and

$$\nabla_{\boldsymbol{z}} F(\boldsymbol{u}, \boldsymbol{z}; \mu_k^{(j)}) = A_{\boldsymbol{u}}^T (A_{\boldsymbol{u}} \boldsymbol{z} - \boldsymbol{y}) + \mu_k^{(j)} [h^{-1}]'(\boldsymbol{z}) \odot h^{-1}(\boldsymbol{z}) \tag{4.16}$$

for $\mu_k^{(j)}$ a learned scalar. Note, the application of Euclidean ball projection mapping, ρ_ξ, ensures a sufficiently small step size is used in each gradient update for numerical stability.

For parameters, P_u is structured as a scaled identity matrix and $D = 2$ where $\boldsymbol{\theta}_{k,1}^{(j)} = B_k^{(j)}$ and $\boldsymbol{\theta}_{k,2}^{(j)} = \mu_k^{(j)}$. A structural similarity index measure (SSIM) loss function is used to train CG-Net. For two images or matrices I_1 and I_2 of equivalent size, the SSIM loss function is given by $L(I_1, I_2) = 1 - \text{SSIM}(I_1, I_2)$.

Deep Regularized Compound Gaussian Network (DR-CG-Net) [76]

In this method, the scale variable regularization is left as an implicit function that is learned, through its gradient, in the unrolled deep neural network. In DR-CG-Net, a projected gradient

descent (PGD) or ISTA step are employed as scale variable descent updates. These scale variable descent update methods are given by

$$g_k^{(j)}(\boldsymbol{z}, \boldsymbol{u}) = \begin{cases} v_k^{(j)}(\boldsymbol{z}, \boldsymbol{u}; \delta_k^{(j)}) + \mathcal{V}_k^{(j)}(\boldsymbol{z}) & \text{PGD} \\ v_k^{(j)}(\boldsymbol{z}, \boldsymbol{u}; \delta_k^{(j)}) + \mathcal{V}_k^{(j)}\left(v_k^{(j)}(\boldsymbol{z}, \boldsymbol{u}; \delta_k^{(j)})\right) & \text{ISTA} \end{cases} \tag{4.17}$$

where, for a step size $\delta_k^{(j)} > 0$ and fixed real number $\xi > 0$,

$$v_k^{(j)}(\boldsymbol{z}, \boldsymbol{u}; \delta_k^{(j)}) = \boldsymbol{z} - \delta_k^{(j)} \rho_\xi(A_{\boldsymbol{u}}^T(A_{\boldsymbol{u}}\boldsymbol{z} - \boldsymbol{y})) \tag{4.18}$$

is a gradient update of $\boldsymbol{z}$ over the data fidelity term of (4.13) and $\mathcal{V}_k^{(j)} : \mathbb{R}^n \to \mathbb{R}^n$ an embedded subnetwork. We remark that the PGD update in (4.17) is a gradient descent step on (4.13) with respect to (w.r.t.) $\boldsymbol{z}$ when $\mathcal{V}_k^{(j)} = \nabla\mathcal{R}$. Instead, the ISTA update of (4.17) is a proximal gradient descent step on (4.13) with respect to (w.r.t.) $\boldsymbol{z}$ when $\text{prox}_{\mathcal{R}}(\boldsymbol{z}) = \boldsymbol{z} + \mathcal{V}_k^{(j)}(\boldsymbol{z})$. Hence, in training $\mathcal{V}_k^{(j)}$ the regularization, or equivalently prior distribution, for the scale variable is learned. Finally, ρ_ξ is applied for numerical stability as in CG-Net.

Each subnetwork, $\mathcal{V}_k^{(j)}$, consists of L_c convolutional layers using ReLU activation functions. That is, layer ℓ consists of f_ℓ convolutions, i.e. filter channels, using kernel size $k_\ell \times k_\ell$ with unit stride. Note, zero padding is applied to each filter channel of the input such that the output, at any filter channel, is the same size as the input from any filter channel. Furthermore, we take $f_{L_c} = 1$ so that given a single channel image input to $\mathcal{V}_k^{(j)}$ the output is also single channel image of equivalent dimension. For our analysis we assume, without loss of generality, that convolutional layer ℓ of $\mathcal{V}_k^{(j)}$ is implemented as a matrix-vector product with weight matrix $W_{k,\ell}^{(j)}$. As each weight matrix $W_{k,\ell}^{(j)}$ is structured upon a convolutional layer mapping from $f_{\ell-1}$ to f_ℓ filter channels using kernel sizes of $k_\ell \times k_\ell$ then $f_{\ell-1} f_\ell k_\ell^2$ parameters define $W_{k,\ell}^{(j)}$.

For parameters, P_u is structured as a tridiagonal matrix and $D = L_c + 1$ where $\boldsymbol{\theta}_{k,\ell}^{(j)} = W_{k,\ell}^{(j)}$ for $1 \leq \ell \leq L_c$ and $\boldsymbol{\theta}_{k,L_c+1}^{(j)} = \delta_k^{(j)}$. A mean absolute loss function, $L(\boldsymbol{x}_1, \boldsymbol{x}_2) = \frac{1}{n}\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_1$, is used to train DR-CG-Net.

4.4 Generalization Error Bound for G-CG-Net

In order to estimate the generalization error, similarly to [86–88], we derive an upper bound on $\mathcal{L}(\widehat{\boldsymbol{c}})$ in terms of $\mathcal{L}_{\mathcal{S}}(\widehat{\boldsymbol{c}})$ and a Dudley's inequality bound of the Rademacher complexity for the hypothesis space generated by G-CG-Net. The Dudley's inequality bound is evaluated using a covering number argument dependent on a Lipschitz property of the G-CG-Net outputs w.r.t. the G-CG-Net parameters. Our key contribution here is showing the G-CG-Net outputs are indeed Lipschitz w.r.t. to the network parameters and applying our derived GEB to the specific CG-Net and DR-CG-Net structures.

The remainder of this section is structured as follows. Section 4.4.1 explicates boundedness assumptions, a Lipschitz assumption, and loss function requirements that underlie our GEBs for deep compound Gaussian networks. We derive a Lipschitz property of G-CG-Net outputs w.r.t. the G-CG-Net parameters in section 4.4.2. Finally, section 4.4.3 details the GEB for G-CG-Net and provides a proof of this bound.

4.4.1 Assumptions for the Generalization Error Bound

A common assumption in machine learning literature and implementation is that the input data to a neural network is bounded. Specific bounds are often guaranteed through preprocessing of the data, which has been shown to assist in the performance of DNN models [93]. For instance, in [57, 76], which use images as the signals of interest, each image is scaled down to be bounded in the Euclidean unit ball. Furthermore, bounded data implies that the possible parameters to be learned by a DNN are similarly bounded as DNNs are trained only for a finite number of epochs using a small learning rate to estimate bounded signals from bounded inputs.

Assumption 4.4.1. *The following bounds hold almost surely:*

1. *Original signals,* $\boldsymbol{c}$*, satisfy* $\|\boldsymbol{c}\|_2 \leq c_{\max}$.

2. *Scale variables,* $\boldsymbol{z}$*, satisfy* $\|\boldsymbol{z}\|_\infty \leq z_\infty$.

3. *Covariance matrices, P_u, satisfy $P_u \in \mathcal{P}$ where, for $0 < p_{\min} \leq p_{\max}$,*

$$\begin{aligned}\mathcal{P} &= \{n \times n \text{ SPD matrices with bounded spectrum}\} \\ &= \{P \in \mathbb{R}^{n\times n} : P \text{ is SPD}, \|P\|_2 \leq p_{\max}, \|P^{-1}\|_2 \leq p_{\min}^{-1}\}. \end{aligned} \tag{4.19}$$

4. *Each scale variable update parameter, $\boldsymbol{\theta}_{k,d}^{(j)}$, satisfies $\|\boldsymbol{\theta}_{k,d}^{(j)}\|_{(d)} \leq \omega_d$ for $\omega_d \geq 0$ and some norm $\|\cdot\|_{(d)}$.*

We remark that assuming the scale variables are bounded is reasonable since we assume the original signals are bounded and the original signals are a product of the scale variable and a Gaussian variable. The boundedness of the scale variables is enforced in G-CG-Net by the mReLU activation function, $\mathcal{P}_{0,z_\infty}$, in each scale variable update layer. Finally, as the original signals are bounded, we force the estimates from G-CG-Net to, equivalently, be bounded by applying the projection operator $\rho_{c_{\max}}$ to the estimate $\mathcal{Z}_K \odot U_K$ produced by G-CG-Net.

Next, we require the following assumptions on the DNN loss function.

Assumption 4.4.2. *For a finite-dimensional vector space V, the network loss function $L(\boldsymbol{x}_1, \boldsymbol{x}_2) : V \to \mathbb{R}$ satisfies:*

1. *Bounded:* $\quad |L(\boldsymbol{x}_1, \boldsymbol{x}_2)| \leq c$
2. *τ-Lipschitz:* $\quad \|L(\boldsymbol{x}_1, \boldsymbol{x}) - L(\boldsymbol{x}_2, \boldsymbol{x})\|_2 \leq \tau \|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_2$

for $c \geq 0$, $\tau \geq 0$ and all $\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x} \in V$.

The final assumption we require is a Lipschitz property for the scale variable descent update.

Assumption 4.4.3. *For every $\boldsymbol{z}_j \in \mathbb{R}^n$ satisfying $\|\boldsymbol{z}_j\|_\infty \leq z_\infty$ and $\boldsymbol{u}_j = \mathcal{T}_{\boldsymbol{y}}(\boldsymbol{z}_j; P_j)$ where $P_j \in \mathcal{P}$, each scale variable descent update method $g_k^{(j)}(\boldsymbol{z}, \boldsymbol{u})$, which is parameterized by some $\vartheta_k^{(j)} =$*

$(\boldsymbol{\theta}_{k,1}^{(j)}, \ldots, \boldsymbol{\theta}_{k,D}^{(j)})$, *satisfies*

$$\|g_k^{(j)}(\boldsymbol{z}_1, \boldsymbol{u}_1; \vartheta_k^{(j)}) - g_k^{(j)}(\boldsymbol{z}_2, \boldsymbol{u}_2; \widetilde{\vartheta}_k^{(j)})\|_2 \leq r_{k,1}^{(j-1)}\|\boldsymbol{z}_1 - \boldsymbol{z}_2\|_2 + r_{k,2}^{(j-1)}\|\boldsymbol{u}_1 - \boldsymbol{u}_2\|_2 + \sum_{d=1}^{D} r_{k,d,3}^{(j-1)}\|\boldsymbol{\theta}_{k,d}^{(j)} - \widetilde{\boldsymbol{\theta}}_{k,d}^{(j)}\|_{(d)}$$

for non-negative constants $r_{k,1}^{(j-1)}, r_{k,2}^{(j-1)}, r_{k,d,3}^{(j-1)}$, *and norms* $\{\|\cdot\|_{(d)}\}_{d=1}^{D}$.

While the Lipschitz property of Assumption 4.4.3 may seem arbitrarily restrictive we show it holds for CG-Net and DR-CG-Net in Section 4.5 and Section 4.6, respectively.

4.4.2 Lipschitz Property of G-CG-Net

In this section, we show that G-CG-Net is Lipschitz w.r.t. its parameters $\boldsymbol{\Theta}$ in (4.14). This result is a cornerstone in proving the GEB of G-CG-Net provided in this chapter and is dependent on a Lipschitz and bounded property of the Tikhonov solution along with Assumption 4.4.3. The following lemma from the literature will be frequently required.

Lemma 4.4.4. *For* $1 \leq p \leq \infty$ *let* $\|\cdot\|_p$ *be the standard* p*-norm on* $\mathbb{R}^d$. *Then for all* $1 \leq q \leq \ell$ *and all* $\boldsymbol{x} \in \mathbb{R}^d$ *it holds that*

$$\|\boldsymbol{x}\|_\ell \leq \|\boldsymbol{x}\|_q \qquad \text{and} \qquad \|\boldsymbol{x}\|_q \leq d^{\frac{1}{q}-\frac{1}{\ell}}\|\boldsymbol{x}\|_\ell.$$

Proof. First, we show that $\|\boldsymbol{x}\|_\ell \leq \|\boldsymbol{x}\|_q$ for all $\boldsymbol{x} \in \mathbb{R}^d$ and all $1 \leq q \leq \ell$. Let $\boldsymbol{y} \in \mathbb{R}^d$ satisfy $\|\boldsymbol{y}\|_q = 1$. Thus, $\|\boldsymbol{y}\|_q^p = 1$ for every $p \geq 1$ and each component of $\boldsymbol{y}$ satisfies $|y_i| \leq 1$. Hence, for any $1 \leq q \leq \ell$ it holds that $|y_i|^\ell \leq |y_i|^q$. Therefore

$$\|\boldsymbol{y}\|_\ell^\ell = \sum_{i=1}^{d} |y_i|^\ell \leq \sum_{i=1}^{d} |y_i|^p = \|\boldsymbol{y}\|_q^q = 1 = \|\boldsymbol{y}\|_q^\ell$$

implying $\|\boldsymbol{y}\|_\ell \leq \|\boldsymbol{y}\|_q$. Now, let $\boldsymbol{x} \in \mathbb{R}^n$. As $\frac{\boldsymbol{x}}{\|\boldsymbol{x}\|_q} = 1$ then using homogeneity of norms

$$\frac{1}{\|\boldsymbol{x}\|_q}\|\boldsymbol{x}\|_\ell = \left\|\frac{\boldsymbol{x}}{\|\boldsymbol{x}\|_q}\right\|_\ell \leq \left\|\frac{\boldsymbol{x}}{\|\boldsymbol{x}\|_q}\right\|_q = \frac{1}{\|\boldsymbol{x}\|_q}\|\boldsymbol{x}\|_q$$

implying $\|\boldsymbol{x}\|_\ell \leq \|\boldsymbol{x}\|_q$.

Second, we show that $\|\boldsymbol{x}\|_q \leq n^{\frac{1}{q}-\frac{1}{\ell}}\|\boldsymbol{x}\|_\ell$ for all $\boldsymbol{x} \in \mathbb{R}^d$ and all $1 \leq q \leq \ell$. Clearly, equality holds if $q = \ell$. Now consider $q < \ell$. Note, Hölder's Inequality states that for all $\boldsymbol{a} \in \mathbb{R}^d$, $\boldsymbol{b} \in \mathbb{R}^d$, and all $p, s > 1$ satisfying $1/p + 1/s = 1$ it holds that

$$\sum_{i=1}^{d} |a_i b_i| \leq \left(\sum_{i=1}^{d} |a_i|^p\right)^{\frac{1}{p}} \left(\sum_{i=1}^{d} |b_i|^s\right)^{\frac{1}{s}}.$$

For $r > 1$ set $p = r$ and $s = \frac{r}{r-1}$ then

$$\sum_{i=1}^{d} |a_i b_i| \leq \left(\sum_{i=1}^{d} |a_i|^r\right)^{\frac{1}{r}} \left(\sum_{i=1}^{d} |b_i|^{\frac{r}{r-1}}\right)^{1-\frac{1}{r}}.$$

Set $a_i = |x_i|^q$ and $b_i = 1$ for $i = 1, 2, \ldots, d$. Then for $r = \frac{\ell}{q} > 1$ observe

$$\begin{aligned}
\sum_{i=1}^{d} |x_i|^q &\leq \left(\sum_{i=1}^{d} |x_i|^{q\left(\frac{\ell}{q}\right)}\right)^{\frac{q}{\ell}} \left(\sum_{i=1}^{d} 1\right)^{1-\frac{q}{\ell}} \\
\|\boldsymbol{x}\|_q^q &\leq \left(\sum_{i=1}^{d} |x_i|^\ell\right)^{\frac{q}{\ell}} d^{1-\frac{q}{\ell}} \\
\|\boldsymbol{x}\|_q &\leq \left(\sum_{i=1}^{d} |x_i|^\ell\right)^{\frac{1}{\ell}} d^{\frac{1}{q}-\frac{1}{\ell}} \\
\|\boldsymbol{x}\|_q &\leq d^{\frac{1}{q}-\frac{1}{\ell}}\|\boldsymbol{x}\|_\ell.
\end{aligned}$$

⊠

Next, we use Lemma 4.4.4 to derive a similar matrix norm inequality.

Corollary 4.4.5. *For $1 \leq p \leq \infty$ let $\|\cdot\|_p$ be the standard vector p-norm or induced matrix p-norm. Then for all $1 \leq q \leq \ell$ and all matrices M it holds that*

$$\|M\|_\ell \leq \|M\|_q.$$

Proof. Recall

$$\|M\|_p \coloneqq \sup_{\|\boldsymbol{x}\|_p \leq 1} \|M\boldsymbol{x}\|_p.$$

By Lemma 4.4.4 it holds that $\{\boldsymbol{x} \in \mathbb{R}^d : \|\boldsymbol{x}\|_\ell \leq 1\} \subseteq \{\boldsymbol{x} \in \mathbb{R}^d : \|\boldsymbol{x}\|_q \leq 1\}$ and thus

$$\|M\|_\ell = \sup_{\|\boldsymbol{x}\|_\ell \leq 1} \|M\boldsymbol{x}\|_\ell \leq \sup_{\|\boldsymbol{x}\|_\ell \leq 1} \|M\boldsymbol{x}\|_q \leq \sup_{\|\boldsymbol{x}\|_q \leq 1} \|M\boldsymbol{x}\|_q = \|M\|_q. \qquad \boxtimes$$

Lipschitz of Complete Scale Variable Mappings

For notation we write $\mathcal{Z}_k^{(J)} : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n$ as the complete scale variable mapping consisting of J scale variable updates. That is, $\mathcal{Z}_k^{(J)} = Z_k^{(J)} \circ \cdots \circ Z_k^{(1)}$. As each scale variable update, $Z_k^{(j)}$, is parameterized by some $\boldsymbol{\theta}_{k,1}^{(j)}, \ldots, \boldsymbol{\theta}_{k,D}^{(j)}$ then the complete scale variable mapping, $\mathcal{Z}_k^{(J)}$, is parameterized by $\Theta_k^{(J)} = (\boldsymbol{\theta}_{k,1}^{(j)}, \ldots, \boldsymbol{\theta}_{k,D}^{(j)})_{j \in \mathbb{N}[J]}$. We remark that this notation is introduced since we will show $\mathcal{Z}_k^{(J)}$ is Lipschitz w.r.t. $\Theta_k^{(J)}$ by induction over J. Additionally, this notation emphasizes that the number of unrolled scale variable updates, J, is a parameter that can be varied at implementation and so the following Lipschitz property holds for all $J \in \mathbb{N}$.

Proposition 4.4.6. *Let the Lipschitz property in Assumption 4.4.3 hold. Then the complete scale variable mapping $\mathcal{Z}_k^{(J)}(\boldsymbol{z}, \boldsymbol{u})$, which is parameterized by some $\Theta_k^{(J)} = (\boldsymbol{\theta}_{k,1}^{(j)}, \ldots, \boldsymbol{\theta}_{k,D}^{(j)})_{j \in \mathbb{N}[J]}$, satisfies*

$$\begin{aligned}\left\| \mathcal{Z}_k^{(J)}(\boldsymbol{z}_1, \boldsymbol{u}_1; \Theta_k^{(J)}) - \mathcal{Z}_k^{(J)}(\boldsymbol{z}_2, \boldsymbol{u}_2; \widetilde{\Theta}_k^{(J)}) \right\|_2 &\leq \widehat{r}_{k,1}^{(J)} \|\boldsymbol{z}_1 - \boldsymbol{z}_2\|_2 + \widehat{r}_{k,2}^{(J)} \|\boldsymbol{u}_1 - \boldsymbol{u}_2\|_2 \\ &\quad + \sum_{j=1}^{J} \sum_{d=1}^{D} \widehat{r}_{k,d,3}^{(j,J)} \left\| \boldsymbol{\theta}_{k,d}^{(j)} - \widetilde{\boldsymbol{\theta}}_{k,d}^{(j)} \right\|_{(d)}\end{aligned}$$

for $\widehat{r}_{k,1}^{(J)} = \prod_{j=1}^{J} r_{k,1}^{(j-1)}$, $\widehat{r}_{k,2}^{(J)} = \sum_{j=1}^{J} r_{k,2}^{(j-1)} \prod_{\ell=j}^{J-1} r_{k,1}^{(\ell)}$, *and* $\widehat{r}_{k,d,3}^{(j,J)} = r_{k,d,3}^{(j-1)} \prod_{\ell=j}^{J-1} r_{k,1}^{(\ell)}$.

Proof. Using induction on J, the base case, $J = 1$, holds by Assumption 4.4.3 where we set $\prod_{j=J+1}^{J} r_{k,1}^{(j)} = 1$. Let the induction hypothesis hold for fixed $J > 1$. Observe that, using Assumption 4.4.3, the induction hypothesis, and that $\mathcal{P}_{0,z_\infty}$ is 1-Lipschitz by Lemma 4.2.28

$$
\begin{aligned}
&\left\| \mathcal{Z}_k^{(J+1)}(\boldsymbol{z}_1, \boldsymbol{u}_1; \Theta_k^{(J+1)}) - \mathcal{Z}_k^{(J+1)}(\boldsymbol{z}_2, \boldsymbol{u}_2; \widetilde{\Theta}_k^{(J+1)}) \right\|_2 \\
&\qquad = \Big\| \mathcal{P}_{0,z_\infty}(g_k^{(J+1)}(\mathcal{Z}_k^{(J)}(\boldsymbol{z}_1, \boldsymbol{u}_1; \Theta_k^{(J)}), \boldsymbol{u}_1; \vartheta_k^{(J+1)})) \\
&\qquad\quad - \mathcal{P}_{0,z_\infty}(g_k^{(J+1)}(\mathcal{Z}_k^{(J)}(\boldsymbol{z}_2, \boldsymbol{u}_2; \widetilde{\Theta}_k^{(J)}), \boldsymbol{u}_2; \widetilde{\vartheta}_k^{(J+1)})) \Big\|_2 \\
&\qquad = \Big\| g_k^{(J+1)}(\mathcal{Z}_k^{(J)}(\boldsymbol{z}_1, \boldsymbol{u}_1; \Theta_k^{(J)}), \boldsymbol{u}_1; \vartheta_k^{(J+1)}) \\
&\qquad\quad - g_k^{(J+1)}(\mathcal{Z}_k^{(J)}(\boldsymbol{z}_2, \boldsymbol{u}_2; \widetilde{\Theta}_k^{(J)}), \boldsymbol{u}_2; \widetilde{\vartheta}_k^{(J+1)}) \Big\|_2 \\
&\qquad \le r_{k,1}^{(J)} \left\| \mathcal{Z}_k^{(J)}(\boldsymbol{z}_1, \boldsymbol{u}_1; \Theta_k^{(J)}) - \mathcal{Z}_k^{(J)}(\boldsymbol{z}_2, \boldsymbol{u}_2; \widetilde{\Theta}_k^{(J)}) \right\|_2 \\
&\qquad\quad + r_{k,2}^{(J)} \|\boldsymbol{u}_1 - \boldsymbol{u}_2\|_2 + \sum_{d=1}^{D} r_{k,d,3}^{(J)} \|\boldsymbol{\theta}_{k,d}^{(J+1)} - \widetilde{\boldsymbol{\theta}}_{k,d}^{(J+1)}\|_{(d)} \\
&\qquad \le r_{k,1}^{(J)} \left(\widehat{r}_{k,1}^{(J)} \|\boldsymbol{z}_1 - \boldsymbol{z}_2\|_2 + \widehat{r}_{k,2}^{(J)} \|\boldsymbol{u}_1 - \boldsymbol{u}_2\|_2 \right) \\
&\qquad\quad + r_{k,1}^{(J)} \sum_{j=1}^{J} \sum_{d=1}^{D} \widehat{r}_{k,d,3}^{(j,J)} \|\boldsymbol{\theta}_{k,d}^{(j)} - \widetilde{\boldsymbol{\theta}}_{k,d}^{(j)}\|_{(d)} + r_{k,2}^{(J)} \|\boldsymbol{u}_1 - \boldsymbol{u}_2\|_2 \\
&\qquad\quad + \sum_{d=1}^{D} r_{k,d,3}^{(J)} \|\boldsymbol{\theta}_{k,d}^{(J+1)} - \widetilde{\boldsymbol{\theta}}_{k,d}^{(J+1)}\|_{(d)}.
\end{aligned} \tag{4.20}
$$

First note

$$
r_{k,1}^{(J)} \widehat{r}_{k,1}^{(J)} = r_{k,1}^{(J)} \prod_{j=1}^{J} r_{k,1}^{(j-1)} = \prod_{j=1}^{J+1} r_{k,1}^{(j-1)} = \widehat{r}_{k,1}^{(J+1)}.
$$

Second note

$$\begin{aligned} r_{k,1}^{(J)}\,\widehat{r}_{k,2}^{(J)} + r_{k,2}^{(J)} &= r_{k,1}^{(J)} \sum_{j=1}^{J} r_{k,2}^{(j-1)} \prod_{\ell=j}^{J-1} r_{k,1}^{(\ell)} + r_{k,2}^{(J)} \\ &= \sum_{j=1}^{J} r_{k,2}^{(j-1)} \prod_{\ell=j}^{J} r_{k,1}^{(\ell)} + r_{k,2}^{(J)} \\ &= \sum_{j=1}^{J} r_{k,2}^{(j-1)} \prod_{\ell=j}^{J} r_{k,1}^{(\ell)} + r_{k,2}^{(J)} \prod_{\ell=J+1}^{J} r_{k,1}^{(\ell)} \\ &= \sum_{j=1}^{J+1} r_{k,2}^{(j-1)} \prod_{\ell=j}^{J} r_{k,1}^{(\ell)} \\ &= \widehat{r}_{k,2}^{(J+1)}. \end{aligned}$$

Third note

$$r_{k,1}^{(J)}\,\widehat{r}_{k,d,3}^{(j,J)} = r_{k,1}^{(J)} r_{k,d,3}^{(j-1)} \prod_{\ell=j}^{J-1} r_{k,1}^{(\ell)} = r_{k,d,3}^{(j-1)} \prod_{\ell=j}^{J} r_{k,1}^{(\ell)} = \widehat{r}_{k,d,3}^{(j,J+1)}$$

and

$$r_{k,d,3}^{(J)} = r_{k,d,3}^{(J)} \prod_{\ell=J+1}^{J} r_{k,1}^{(\ell)} = \widehat{r}_{k,d,3}^{(J+1,J+1)}.$$

Combining these three notes with (4.20) gives

$$\begin{aligned} \left\| \mathcal{Z}_k^{(J+1)}(\boldsymbol{z}_1, \boldsymbol{u}_1; \Theta_k^{(J+1)}) - \mathcal{Z}_k^{(J+1)}(\boldsymbol{z}_2, \boldsymbol{u}_2; \widetilde{\Theta}_k^{(J+1)}) \right\|_2 &\leq \widehat{r}_{k,1}^{(J+1)} \|\boldsymbol{z}_1 - \boldsymbol{z}_2\|_2 + \widehat{r}_{k,2}^{(J+1)} \|\boldsymbol{u}_1 - \boldsymbol{u}_2\|_2 \\ &\quad + \sum_{j=1}^{J+1} \sum_{d=1}^{D} \widehat{r}_{k,d,3}^{(j,J+1)} \|\boldsymbol{\theta}_{k,d}^{(j)} - \boldsymbol{\theta}_{k,d}^{(j)}\|_{(d)}. \end{aligned}$$ ⊠

Properties of the Tikhonov Solution

Recall, for square matrix M the induced matrix 2-norm, $\|M\|_2$, also known as the spectral norm, is the largest absolute eigenvalue of M. First, we provide a few lemmas necessary to derive a Lipschitz condition for the Tikhonov solution.

Lemma 4.4.7. *For any invertible and symmetric matrix $P \in \mathbb{R}^{n\times n}$, and any scale variable $\boldsymbol{z} \in \mathbb{R}^n$ it holds that*

$$\|(A_{\boldsymbol{z}}^T A_{\boldsymbol{z}} + P^{-1})^{-1}\|_2 \leq \|P\|_2.$$

Proof. Let $0 < \lambda_1 \leq \cdots \leq \lambda_n$ be the eigenvalues of $A_{\boldsymbol{z}}^T A_{\boldsymbol{z}} + P^{-1}$, $0 \leq \gamma_1 \leq \cdots \leq \gamma_n$ be the eigenvalues of $A_{\boldsymbol{z}}^T A_{\boldsymbol{z}}$, and $0 < \kappa_1 \leq \cdots \leq \kappa_n$ the eigenvalues of P. Note these eigenvalues are all non-negative as $A_{\boldsymbol{z}}^T A_{\boldsymbol{z}}$, P, and $A_{\boldsymbol{z}}^T A_{\boldsymbol{z}} + P^{-1}$ are real, symmetric matrices. Then $0 < \lambda_n^{-1} \leq \cdots \leq \lambda_1^{-1}$ are the eigenvalues of $(A_{\boldsymbol{z}}^T A_{\boldsymbol{z}} + P^{-1})^{-1}$ and $0 < \kappa_n^{-1} \leq \cdots \leq \kappa_1^{-1}$ are the eigenvalues of P^{-1}. From Weyl's inequality [94]

$$\gamma_1 + \kappa_n^{-1} \leq \lambda_1$$

and thus

$$\|(A_{\boldsymbol{z}}^T A_{\boldsymbol{z}} + P^{-1})^{-1}\|_2 = \lambda_1^{-1} \leq \frac{1}{\gamma_1 + \kappa_n^{-1}} \leq \frac{1}{\kappa_n^{-1}} = \kappa_n = \|P\|_2. \qquad \boxtimes$$

Next, we provide a Lipschitz bound for the difference of matrices $\|A_{\boldsymbol{z}_2}^T A_{\boldsymbol{z}_2} - A_{\boldsymbol{z}_1}^T A_{\boldsymbol{z}_1}\|_2$.

Lemma 4.4.8. *Let $\boldsymbol{z}_1 \in \mathbb{R}^n$ and $\boldsymbol{z}_2 \in \mathbb{R}^n$ satisfy $\|\boldsymbol{z}_1\|_\infty, \|\boldsymbol{z}_2\|_\infty \leq z_\infty$. Then*

$$\|A_{\boldsymbol{z}_2}^T A_{\boldsymbol{z}_2} - A_{\boldsymbol{z}_1}^T A_{\boldsymbol{z}_1}\|_2 \leq 2z_\infty \|A\|_2^2 \|\boldsymbol{z}_1 - \boldsymbol{z}_2\|_\infty.$$

Proof. Using the triangle inequality observe

$$\begin{aligned}
\|A_{\boldsymbol{z}_2}^T A_{\boldsymbol{z}_2} - A_{\boldsymbol{z}_1}^T A_{\boldsymbol{z}_1}\|_2 &= \|A_{\boldsymbol{z}_2}^T A_{\boldsymbol{z}_2} - A_{\boldsymbol{z}_2}^T A_{\boldsymbol{z}_1} + A_{\boldsymbol{z}_2}^T A_{\boldsymbol{z}_1} - A_{\boldsymbol{z}_1}^T A_{\boldsymbol{z}_1}\|_2 \\
&\leq \|A_{\boldsymbol{z}_2}^T \left(A_{\boldsymbol{z}_2} - A_{\boldsymbol{z}_1}\right)\|_2 + \|\left(A_{\boldsymbol{z}_2}^T - A_{\boldsymbol{z}_1}^T\right) A_{\boldsymbol{z}_1}\|_2 \\
&= \|\text{Diag}(\boldsymbol{z}_2) A^T A \left(\text{Diag}(\boldsymbol{z}_2) - \text{Diag}(\boldsymbol{z}_1)\right)\|_2 \\
&\quad + \|\left(\text{Diag}(\boldsymbol{z}_2) - \text{Diag}(\boldsymbol{z}_1)\right) A^T A \text{Diag}(\boldsymbol{z}_1)\|_2 \\
&\leq \|\text{Diag}(\boldsymbol{z}_2)\|_2 \|A^T A\|_2 \|\text{Diag}(\boldsymbol{z}_2) - \text{Diag}(\boldsymbol{z}_1)\|_2 \\
&\quad + \|\text{Diag}(\boldsymbol{z}_2) - \text{Diag}(\boldsymbol{z}_1)\|_2 \|A^T A\|_2 \|\text{Diag}(\boldsymbol{z}_1)\|_2 \\
&= \left(\|\text{Diag}(\boldsymbol{z}_1)\|_2 + \|\text{Diag}(\boldsymbol{z}_2)\|_2\right) \|A^T A\|_2 \|\text{Diag}(\boldsymbol{z}_1) - \text{Diag}(\boldsymbol{z}_2)\|_2 \\
&\leq \left(\|\boldsymbol{z}_1\|_\infty + \|\boldsymbol{z}_2\|_\infty\right) \|A^T A\|_2 \|\boldsymbol{z}_1 - \boldsymbol{z}_2\|_\infty \\
&\leq 2 z_\infty \|A\|_2^2 \|\boldsymbol{z}_1 - \boldsymbol{z}_2\|_\infty. \qquad \boxtimes
\end{aligned}$$

Now, we show a simple bound on the difference of the inverses of two invertible matrices in terms of the difference in the original matrices.

Lemma 4.4.9. *For any invertible matrices $P \in \mathbb{R}^{k \times k}$ and $\widetilde{P} \in \mathbb{R}^{k \times k}$*

$$\|\widetilde{P}^{-1} - P^{-1}\|_2 \leq \|P^{-1}\|_2 \|\widetilde{P}^{-1}\|_2 \|P - \widetilde{P}\|_2.$$

Proof. Observe

$$\|\widetilde{P}^{-1} - P^{-1}\|_2 = \|\widetilde{P}^{-1}\left(P - \widetilde{P}\right) P^{-1}\|_2 \leq \|P^{-1}\|_2 \|\widetilde{P}^{-1}\|_2 \|P - \widetilde{P}\|_2. \qquad \boxtimes$$

Next, we bound the spectral norm on the difference of two invertible portions of the Tikhonov solution. For an invertible matrix, $M \in \mathbb{R}^{k \times k}$, recall that the condition number of M is defined as

$$\text{Cond}(M) \coloneqq \|M\|_2 \|M^{-1}\|_2.$$

Corollary 4.4.10. *Let $P, \widetilde{P} \in \mathbb{R}^{n\times n}$ be any invertible and symmetric matrices. For any $\boldsymbol{z}_1 \in \mathbb{R}^n$ and $\boldsymbol{z}_2 \in \mathbb{R}^n$ satisfying $\|\boldsymbol{z}_1\|_\infty, \|\boldsymbol{z}_2\|_\infty \leq z_\infty$ the following bound holds*

$$\begin{aligned}\|(A_{\boldsymbol{z}_1}^T A_{\boldsymbol{z}_1} + P^{-1})^{-1} - (A_{\boldsymbol{z}_2}^T A_{\boldsymbol{z}_2} + \widetilde{P}^{-1})^{-1}\|_2 \leq 2z_\infty \|A\|_2^2 \|P\|_2 \|\widetilde{P}\|_2 \|\boldsymbol{z}_1 - \boldsymbol{z}_2\|_\infty \\ + \text{Cond}(P)\text{Cond}(\widetilde{P})\|P - \widetilde{P}\|_2.\end{aligned}$$

Proof. Using Lemma 4.4.9 and then Lemma 4.4.7 note

$$\begin{aligned}&\|(A_{\boldsymbol{z}_1}^T A_{\boldsymbol{z}_1} + P^{-1})^{-1} - (A_{\boldsymbol{z}_2}^T A_{\boldsymbol{z}_2} + \widetilde{P}^{-1})^{-1}\|_2 \\ &\quad \leq \|(A_{\boldsymbol{z}_1}^T A_{\boldsymbol{z}_1} + P^{-1})^{-1}\|_2 \|(A_{\boldsymbol{z}_2}^T A_{\boldsymbol{z}_2} + \widetilde{P}^{-1})^{-1}\|_2 \|A_{\boldsymbol{z}_2}^T A_{\boldsymbol{z}_2} + \widetilde{P}^{-1} - A_{\boldsymbol{z}_1}^T A_{\boldsymbol{z}_1} - P^{-1}\|_2 \\ &\quad \leq \|P\|_2 \|\widetilde{P}\|_2 \|A_{\boldsymbol{z}_2}^T A_{\boldsymbol{z}_2} + \widetilde{P}^{-1} - A_{\boldsymbol{z}_1}^T A_{\boldsymbol{z}_1} - P^{-1}\|_2.\end{aligned}$$

Next, using the triangle inequality, Lemma 4.4.8, and Lemma 4.4.9 observe

$$\begin{aligned}\|A_{\boldsymbol{z}_2}^T A_{\boldsymbol{z}_2} + \widetilde{P}^{-1} - A_{\boldsymbol{z}_1}^T A_{\boldsymbol{z}_1} - P^{-1}\|_2 &\leq \|A_{\boldsymbol{z}_2}^T A_{\boldsymbol{z}_2} - A_{\boldsymbol{z}_1}^T A_{\boldsymbol{z}_1}\|_2 + \|\widetilde{P}^{-1} - P^{-1}\|_2 \\ &\leq 2z_\infty \|A\|_2^2 \|\boldsymbol{z}_1 - \boldsymbol{z}_2\|_\infty + \|P^{-1}\|_2 \|\widetilde{P}^{-1}\|_2 \|P - \widetilde{P}\|_2.\end{aligned}$$

Combining these two inequalities gives

$$\begin{aligned}&\|(A_{\boldsymbol{z}_1}^T A_{\boldsymbol{z}_1} + P^{-1})^{-1} - (A_{\boldsymbol{z}_2}^T A_{\boldsymbol{z}_2} + \widetilde{P}^{-1})^{-1}\|_2 \\ &\quad \leq \|P\|_2 \|\widetilde{P}\|_2 \|A_{\boldsymbol{z}_2}^T A_{\boldsymbol{z}_2} + \widetilde{P}^{-1} - A_{\boldsymbol{z}_1}^T A_{\boldsymbol{z}_1} - P^{-1}\|_2 \\ &\quad \leq \|P\|_2 \|\widetilde{P}\|_2 \left(2z_\infty \|A\|_2^2 \|\boldsymbol{z}_1 - \boldsymbol{z}_2\|_\infty + \|P^{-1}\|_2 \|\widetilde{P}^{-1}\|_2 \|P - \widetilde{P}\|_2\right) \\ &\quad \leq 2z_\infty \|A\|_2^2 \|P\|_2 \|\widetilde{P}\|_2 \|\boldsymbol{z}_1 - \boldsymbol{z}_2\|_\infty + \text{Cond}(P)\text{Cond}(\widetilde{P})\|P - \widetilde{P}\|_2. \qquad \boxtimes\end{aligned}$$

With all of the previous properties of the Tikhonov solution, we now prove that the Tikhonov solution satisfies a Lipschitz property.

Proposition 4.4.11. *Let $\boldsymbol{z}_1 \in \mathbb{R}^n$ and $\boldsymbol{z}_2 \in \mathbb{R}^n$ satisfy $\|\boldsymbol{z}_1\|_\infty, \|\boldsymbol{z}_2\|_\infty \leq z_\infty$. Then the Tikhonov solution $\mathcal{T}_{\boldsymbol{y}}$, which is parameterized by a SPD matrix $P \in \mathbb{R}^{n\times n}$, satisfies*

$$\|\mathcal{T}_{\boldsymbol{y}}(\boldsymbol{z}_1; P) - \mathcal{T}_{\boldsymbol{y}}(\boldsymbol{z}_2; \widetilde{P})\|_2 \leq c_1(\boldsymbol{y})\|\boldsymbol{z}_1 - \boldsymbol{z}_2\|_\infty + c_2(\boldsymbol{y})\|P - \widetilde{P}\|_2$$

for

$$\begin{aligned} c_1(\boldsymbol{y}) &= \|P\|_2\|A\|_2\|\boldsymbol{y}\|_2 \left(1 + 2z_\infty^2\|\widetilde{P}\|_2\|A\|_2^2\right) \\ c_2(\boldsymbol{y}) &= z_\infty\|A\|_2\|\boldsymbol{y}\|_2 \ \text{Cond}(P)\text{Cond}(\widetilde{P}). \end{aligned}$$

Proof. Using the triangle inequality observe

$$\begin{aligned} \|\mathcal{T}_{\boldsymbol{y}}(\boldsymbol{z}_1; P) - \mathcal{T}_{\boldsymbol{y}}(\boldsymbol{z}_2; \widetilde{P})\|_2 &= \|(A_{\boldsymbol{z}_1}^T A_{\boldsymbol{z}_1} + P^{-1})^{-1} A_{\boldsymbol{z}_1}^T \boldsymbol{y} - (A_{\boldsymbol{z}_2}^T A_{\boldsymbol{z}_2} + \widetilde{P}^{-1})^{-1} A_{\boldsymbol{z}_2}^T \boldsymbol{y}\|_2 \\ &= \big\|(A_{\boldsymbol{z}_1}^T A_{\boldsymbol{z}_1} + P^{-1})^{-1} A_{\boldsymbol{z}_1}^T \boldsymbol{y} - (A_{\boldsymbol{z}_1}^T A_{\boldsymbol{z}_1} + P^{-1})^{-1} A_{\boldsymbol{z}_2}^T \boldsymbol{y} \\ &\qquad + (A_{\boldsymbol{z}_1}^T A_{\boldsymbol{z}_1} + P^{-1})^{-1} A_{\boldsymbol{z}_2}^T \boldsymbol{y} - (A_{\boldsymbol{z}_2}^T A_{\boldsymbol{z}_2} + \widetilde{P}^{-1})^{-1} A_{\boldsymbol{z}_2}^T \boldsymbol{y}\big\|_2 \\ &= \big\|(A_{\boldsymbol{z}_1}^T A_{\boldsymbol{z}_1} + P^{-1})^{-1} \left(A_{\boldsymbol{z}_1}^T - A_{\boldsymbol{z}_2}^T\right) \boldsymbol{y} \\ &\qquad + \left[(A_{\boldsymbol{z}_1}^T A_{\boldsymbol{z}_1} + P^{-1})^{-1} - (A_{\boldsymbol{z}_2}^T A_{\boldsymbol{z}_2} + \widetilde{P}^{-1})^{-1}\right] A_{\boldsymbol{z}_2}^T \boldsymbol{y}\big\|_2 \\ &\leq \big\|(A_{\boldsymbol{z}_1}^T A_{\boldsymbol{z}_1} + P^{-1})^{-1} \left(A_{\boldsymbol{z}_1}^T - A_{\boldsymbol{z}_2}^T\right) \boldsymbol{y}\big\|_2 \\ &\qquad + \left\|\left[(A_{\boldsymbol{z}_1}^T A_{\boldsymbol{z}_1} + P^{-1})^{-1} - (A_{\boldsymbol{z}_2}^T A_{\boldsymbol{z}_2} + \widetilde{P}^{-1})^{-1}\right] A_{\boldsymbol{z}_2}^T \boldsymbol{y}\right\|_2 \\ &\leq \big\|(A_{\boldsymbol{z}_1}^T A_{\boldsymbol{z}_1} + P^{-1})^{-1}\big\|_2 \|\boldsymbol{y}\|_2 \|A_{\boldsymbol{z}_1}^T - A_{\boldsymbol{z}_2}^T\|_2 \\ &\qquad + z_\infty\|A\|_2 \|\boldsymbol{y}\|_2 \|(A_{\boldsymbol{z}_1}^T A_{\boldsymbol{z}_1} + P^{-1})^{-1} - (A_{\boldsymbol{z}_2}^T A_{\boldsymbol{z}_2} + \widetilde{P}^{-1})^{-1}\|_2 \\ &\leq \|P\|_2\|A\|_2\|\boldsymbol{y}\|_2 \ \|\boldsymbol{z}_1 - \boldsymbol{z}_2\|_\infty \\ &\qquad + z_\infty\|A\|_2\|\boldsymbol{y}\|_2\|(A_{\boldsymbol{z}_1}^T A_{\boldsymbol{z}_1} + P^{-1})^{-1} - (A_{\boldsymbol{z}_2}^T A_{\boldsymbol{z}_2} + \widetilde{P}^{-1})^{-1}\|_2 \end{aligned}$$

where in the last inequality we used Lemma 4.4.7. Combining the above inequality with Corollary 4.4.10 produces

$$\begin{aligned}
\|\mathcal{T}_{\boldsymbol{y}}(\boldsymbol{z}_1;P)-\mathcal{T}_{\boldsymbol{y}}(\boldsymbol{z}_2;\widetilde{P})\|_2 &\leq \|P\|_2\|A\|_2\|\boldsymbol{y}\|_2\ \|\boldsymbol{z}_1-\boldsymbol{z}_2\|_\infty \\
&\quad + z_\infty\|A\|_2\|\boldsymbol{y}\|_2\|(A_{\boldsymbol{z}_1}^TA_{\boldsymbol{z}_1}+P^{-1})^{-1}-(A_{\boldsymbol{z}_2}^TA_{\boldsymbol{z}_2}+\widetilde{P}^{-1})^{-1}\|_2 \\
&\leq \|P\|_2\|A\|_2\|\boldsymbol{y}\|_2\ \|\boldsymbol{z}_1-\boldsymbol{z}_2\|_\infty \\
&\quad + z_\infty\|A\|_2\|\boldsymbol{y}\|_2\Big(2z_\infty\|A\|_2^2\|P\|_2\|\widetilde{P}\|_2\|\boldsymbol{z}_1-\boldsymbol{z}_2\|_\infty \\
&\qquad\qquad +\text{Cond}(P)\text{Cond}(\widetilde{P})\|P-\widetilde{P}\|_2\Big) \\
&= \|P\|_2\|A\|_2\|\boldsymbol{y}\|_2\left(1+2z_\infty^2\|\widetilde{P}\|_2\|A\|_2^2\right)\|\boldsymbol{z}_1-\boldsymbol{z}_2\|_\infty \\
&\quad + z_\infty\|A\|_2\|\boldsymbol{y}\|_2\ \text{Cond}(P)\text{Cond}(\widetilde{P})\|P-\widetilde{P}\|_2 \\
&= c_1(\boldsymbol{y})\|\boldsymbol{z}_1-\boldsymbol{z}_2\|_\infty+c_2(\boldsymbol{y})\|P-\widetilde{P}\|_2. \qquad \boxtimes
\end{aligned}$$

Lastly, we derive a bound on the norm of the Tikhonov solution.

Proposition 4.4.12. *For any SPD matrix $P\in\mathbb{R}^{n\times n}$, any $\boldsymbol{z}\in\mathbb{R}^n$, $\boldsymbol{y}\in\mathbb{R}^m$, and $2\leq p\leq\infty$ it holds that*

$$\|\mathcal{T}_{\boldsymbol{y}}(\boldsymbol{z};P)\|_p\leq\|\boldsymbol{z}\|_\infty\|P\|_2\|A\|_p\|\boldsymbol{y}\|_p.$$

Proof. Using Lemma 4.4.7 and Lemma 4.4.4 observe

$$\begin{aligned}
\|\mathcal{T}_{\boldsymbol{y}}(\boldsymbol{z};P)\|_p &= \|(A_{\boldsymbol{z}}^TA_{\boldsymbol{z}}+P^{-1})^{-1}A_{\boldsymbol{z}}^T\boldsymbol{y}\|_p \\
&\leq \|(A_{\boldsymbol{z}}^TA_{\boldsymbol{z}}+P^{-1})^{-1}\|_p\|\text{Diag}(\boldsymbol{z})\|_p\|A\|_p\|\boldsymbol{y}\|_p \\
&\leq \|(A_{\boldsymbol{z}}^TA_{\boldsymbol{z}}+P^{-1})^{-1}\|_2\|\text{Diag}(\boldsymbol{z})\|_2\|A\|_p\|\boldsymbol{y}\|_p \\
&\leq \|\boldsymbol{z}\|_\infty\|P\|_2\|A\|_p\|\boldsymbol{y}\|_p. \qquad \boxtimes
\end{aligned}$$

Lipschitz Property of G-CG-Net Outputs

Using the above Lipschitz property of the complete scale variable mappings and the Lipschitz and bounded properties of the Tikhonov solution, we provide a Lipschitz property of the G-CG-

Net outputs w.r.t. the network parameters. For this section we define $\boldsymbol{\zeta}_k$ and $\widetilde{\boldsymbol{\zeta}}_k$ as the G-CG-Net scale variable estimates on iteration k when G-CG-Net is parameterized by $\boldsymbol{\Theta}$ or $\widetilde{\boldsymbol{\Theta}}$, from (4.14), respectively. That is, $\boldsymbol{\zeta}_k$ is recursively defined by $\boldsymbol{\zeta}_k = \mathcal{Z}_k^{(J)}(\boldsymbol{\zeta}_{k-1}, \mathcal{T}_{\boldsymbol{y}}(\boldsymbol{\zeta}_{k-1}; P_u); \Theta_k^{(J)})$

First, we show a Lipschitz property of the final scale variable estimate in the following proposition.

Proposition 4.4.13. *If Assumption 4.4.3 holds then*

$$\|\boldsymbol{\zeta}_K - \widetilde{\boldsymbol{\zeta}}_K\|_2 \leq \widehat{c}_1^{(K,J)}(\boldsymbol{y})\|P_u - \widetilde{P}_u\|_2 + \sum_{k=1}^{K}\sum_{j=1}^{J}\sum_{d=1}^{D} \widehat{c}_{k,j,d,2}^{(K,J)}(\boldsymbol{y})\|\boldsymbol{\theta}_{k,d}^{(j)} - \widetilde{\boldsymbol{\theta}}_{k,d}^{(j)}\|_{(d)}$$

for

$$\widehat{c}_1^{(K,J)}(\boldsymbol{y}) = c_2(\boldsymbol{y})\sum_{k=1}^{K}\widehat{r}_{k,2}^{(J)}\prod_{\ell=k+1}^{K}(\widehat{r}_{\ell,1}^{(J)} + \widehat{r}_{\ell,2}^{(J)}c_1(\boldsymbol{y}))$$
$$\widehat{c}_{k,j,d,2}^{(K,J)}(\boldsymbol{y}) = \widehat{r}_{k,d,3}^{(j,J)}\prod_{\ell=k+1}^{K}(\widehat{r}_{\ell,1}^{(J)} + \widehat{r}_{\ell,2}^{(J)}c_1(\boldsymbol{y}))$$

where $\widehat{r}_{k,1}^{(J)}, \widehat{r}_{k,2}^{(J)}$, and $\widehat{r}_{k,d,3}^{(j,J)}$ are given in Proposition 4.4.6 and $c_1(\boldsymbol{y})$ and $c_2(\boldsymbol{y})$ are given in Proposition 4.4.11.

Proof. Combining Proposition 4.4.6 and 4.4.11 for any k it holds that

$$\begin{aligned}
\|\boldsymbol{\zeta}_k - \widetilde{\boldsymbol{\zeta}}_k\|_2 &= \left\|\mathcal{Z}_k^{(J)}(\boldsymbol{\zeta}_{k-1}, \mathcal{T}_{\boldsymbol{y}}(\boldsymbol{\zeta}_{k-1}; P_u); \Theta_k^{(J)}) - \mathcal{Z}_k^{(J)}(\widetilde{\boldsymbol{\zeta}}_{k-1}, \mathcal{T}_{\boldsymbol{y}}(\widetilde{\boldsymbol{\zeta}}_{k-1}; \widetilde{P}_u); \widetilde{\Theta}_k^{(J)})\right\|_2 \\
&\leq \widehat{r}_{k,1}^{(J)}\|\boldsymbol{\zeta}_{k-1} - \widetilde{\boldsymbol{\zeta}}_{k-1}\|_2 + \widehat{r}_{k,2}^{(J)}\|\mathcal{T}_{\boldsymbol{y}}(\boldsymbol{\zeta}_{k-1}; P_u) - \mathcal{T}_{\boldsymbol{y}}(\widetilde{\boldsymbol{\zeta}}_{k-1}; \widetilde{P}_u)\|_2 \\
&\quad + \sum_{j=1}^{J}\sum_{d=1}^{D}\widehat{r}_{k,d,3}^{(j,J)}\|\boldsymbol{\theta}_{k,d}^{(j)} - \widetilde{\boldsymbol{\theta}}_{k,d}^{(j)}\|_{(d)} \\
&\leq (\widehat{r}_{k,1}^{(J)} + \widehat{r}_{k,2}^{(J)}c_1(\boldsymbol{y}))\|\boldsymbol{\zeta}_{k-1} - \widetilde{\boldsymbol{\zeta}}_{k-1}\|_2 + \widehat{r}_{k,2}^{(J)}c_2(\boldsymbol{y})\|P_u - \widetilde{P}_u\|_2 \\
&\quad + \sum_{j=1}^{J}\sum_{d=1}^{D}\widehat{r}_{k,d,3}^{(j,J)}\|\boldsymbol{\theta}_{k,d}^{(j)} - \widetilde{\boldsymbol{\theta}}_{k,d}^{(j)}\|_{(d)}. \qquad (4.21)
\end{aligned}$$

Now, we use induction on K. The base case $K = 1$ holds by (4.21) as $\boldsymbol{\zeta}_0 = \widetilde{\boldsymbol{\zeta}}_0 \equiv \mathcal{Z}_0$ and $\prod_{\ell=K+1}^{K}(\widehat{r}_{\ell,1}^{(J)} + \widehat{r}_{\ell,2}^{(J)} c_1(\boldsymbol{y})) = 1$. Assume the induction hypothesis holds for fixed $K-1$ where $K > 2$. By (4.21) and the induction hypothesis observe

$$
\begin{aligned}
\|\boldsymbol{\zeta}_K - \widetilde{\boldsymbol{\zeta}}_K\|_2 &\leq (\widehat{r}_{K,1}^{(J)} + \widehat{r}_{K,2}^{(J)} c_1(\boldsymbol{y}))\|\boldsymbol{\zeta}_{K-1} - \widetilde{\boldsymbol{\zeta}}_{K-1}\|_2 + \widehat{r}_{K,2}^{(J)} c_2(\boldsymbol{y})\|P_u - \widetilde{P}_u\|_2 \\
&\quad + \sum_{j=1}^{J}\sum_{d=1}^{D} \widehat{r}_{K,d,3}^{(j,J)} \|\boldsymbol{\theta}_{K,d}^{(j)} - \widetilde{\boldsymbol{\theta}}_{K,d}^{(j)}\|_{(d)} \\
&\leq \left[(\widehat{r}_{K,1}^{(J)} + \widehat{r}_{K,2}^{(J)} c_1(\boldsymbol{y}))\widehat{c}_1^{(K-1,J)}(\boldsymbol{y}) + \widehat{r}_{K,2}^{(J)} c_2(\boldsymbol{y})\right] \|P_u - \widetilde{P}_u\|_2 \\
&\quad + (\widehat{r}_{K,1}^{(J)} + \widehat{r}_{K,2}^{(J)} c_1(\boldsymbol{y})) \sum_{k=1}^{K-1}\sum_{j=1}^{J}\sum_{d=1}^{D} \widehat{c}_{k,j,d,2}^{(K-1,J)}(\boldsymbol{y}) \|\boldsymbol{\theta}_{k,d}^{(j)} - \widetilde{\boldsymbol{\theta}}_{k,d}^{(j)}\|_{(d)} \\
&\quad + \sum_{j=1}^{J}\sum_{d=1}^{D} \widehat{r}_{K,d,3}^{(j,J)} \|\boldsymbol{\theta}_{K,d}^{(j)} - \widetilde{\boldsymbol{\theta}}_{K,d}^{(j)}\|_{(d)}. \qquad (4.22)
\end{aligned}
$$

First note,

$$
\begin{aligned}
&(\widehat{r}_{K,1}^{(J)} + \widehat{r}_{K,2}^{(J)} c_1(\boldsymbol{y}))\widehat{c}_1^{(K-1,J)}(\boldsymbol{y}) + \widehat{r}_{K,2}^{(J)} c_2(\boldsymbol{y}) \\
&\qquad = (\widehat{r}_{K,1}^{(J)} + \widehat{r}_{K,2}^{(J)} c_1(\boldsymbol{y}))c_2(\boldsymbol{y}) \sum_{k=1}^{K-1} \widehat{r}_{k,2}^{(J)} \prod_{\ell=k+1}^{K-1} (\widehat{r}_{\ell,1}^{(J)} + \widehat{r}_{\ell,2}^{(J)} c_1(\boldsymbol{y})) \\
&\qquad\quad + \widehat{r}_{K,2}^{(J)} c_2(\boldsymbol{y}) \\
&\qquad = c_2(\boldsymbol{y}) \sum_{k=1}^{K-1} \widehat{r}_{k,2}^{(J)} \prod_{\ell=k+1}^{K} (\widehat{r}_{\ell,1}^{(J)} + \widehat{r}_{\ell,2}^{(J)} c_1(\boldsymbol{y})) \\
&\qquad\quad + \widehat{r}_{K,2}^{(J)} c_2(\boldsymbol{y}) \prod_{\ell=K+1}^{K} (\widehat{r}_{\ell,1}^{(J)} + \widehat{r}_{\ell,2}^{(J)} c_1(\boldsymbol{y})) \\
&\qquad = c_2(\boldsymbol{y}) \sum_{k=1}^{K} \widehat{r}_{k,2}^{(J)} \prod_{\ell=k+1}^{K} (\widehat{r}_{\ell,1}^{(J)} + \widehat{r}_{\ell,2}^{(J)} c_1(\boldsymbol{y})) \\
&\qquad = \widehat{c}_1^{(K,J)}(\boldsymbol{y}).
\end{aligned}
$$

Similarly, note

$$(\widehat{r}_{K,1}^{(J)} + \widehat{r}_{K,2}^{(J)} c_1(\boldsymbol{y}))\widehat{c}_{k,j,d,2}^{(K-1,J)}(\boldsymbol{y}) = \widehat{c}_{k,j,d,2}^{(K,J)}(\boldsymbol{y})$$

and

$$\widehat{r}_{K,d,3}^{(j,J)} = \widehat{r}_{K,d,3}^{(j,J)} \prod_{\ell=K+1}^{K} (\widehat{r}_{\ell,1}^{(J)} + \widehat{r}_{\ell,2}^{(J)} c_1(\boldsymbol{y})) = \widehat{c}_{K,j,d,2}^{(K,J)}(\boldsymbol{y}).$$

Combining these two notes with (4.22) produces

$$\begin{aligned}
\|\boldsymbol{\zeta}_K - \widetilde{\boldsymbol{\zeta}}_K\|_2 \leq & \left[(\widehat{r}_{K,1}^{(J)} + \widehat{r}_{K,2}^{(J)} c_1(\boldsymbol{y}))\widehat{c}_1^{(K-1,J)}(\boldsymbol{y}) + \widehat{r}_{K,2}^{(J)} c_2(\boldsymbol{y})\right] \|P_u - \widetilde{P}_u\|_2 \\
& + (\widehat{r}_{K,1}^{(J)} + \widehat{r}_{K,2}^{(J)} c_1(\boldsymbol{y})) \sum_{k=1}^{K-1}\sum_{j=1}^{J}\sum_{d=1}^{D} \widehat{c}_{k,j,d,2}^{(K-1,J)}(\boldsymbol{y}) \|\boldsymbol{\theta}_{k,d}^{(j)} - \widetilde{\boldsymbol{\theta}}_{k,d}^{(j)}\|_{(d)} \\
& + \sum_{j=1}^{J}\sum_{d=1}^{D} \widehat{r}_{K,d,3}^{(j,J)} \|\boldsymbol{\theta}_{K,d}^{(j)} - \widetilde{\boldsymbol{\theta}}_{K,d}^{(j)}\|_{(d)} \\
= & \, \widehat{c}_1^{(K,J)}(\boldsymbol{y}) \|P_u - \widetilde{P}_u\|_2 + \sum_{k=1}^{K-1}\sum_{j=1}^{J}\sum_{d=1}^{D} \widehat{c}_{k,j,d,2}^{(K,J)}(\boldsymbol{y}) \|\boldsymbol{\theta}_{k,d}^{(j)} - \widetilde{\boldsymbol{\theta}}_{k,d}^{(j)}\|_{(d)} \\
& + \sum_{j=1}^{J}\sum_{d=1}^{D} \widehat{c}_{K,j,d,2}^{(K,J)}(\boldsymbol{y}) \|\boldsymbol{\theta}_{K,d}^{(j)} - \widetilde{\boldsymbol{\theta}}_{K,d}^{(j)}\|_{(d)} \\
= & \, \widehat{c}_1^{(K,J)}(\boldsymbol{y}) \|P_u - \widetilde{P}_u\|_2 + \sum_{k=1}^{K}\sum_{j=1}^{J}\sum_{d=1}^{D} \widehat{c}_{k,j,d,2}^{(K,J)}(\boldsymbol{y}) \|\boldsymbol{\theta}_{k,d}^{(j)} - \widetilde{\boldsymbol{\theta}}_{k,d}^{(j)}\|_{(d)}. \qquad \boxtimes
\end{aligned}$$

Finally, we show that G-CG-Net estimates, $\widehat{\boldsymbol{c}}(\boldsymbol{y}; \boldsymbol{\Theta})$, are Lipschitz w.r.t. the G-CG-Net parameters.

Theorem 4.4.14. *Let Assumption 4.4.3 hold. Then for any parameterizations $\boldsymbol{\Theta}$ and $\widetilde{\boldsymbol{\Theta}}$ of G-CG-Net, the G-CG-Net estimates satisfy the following Lipschitz property:*

$$\|\widehat{\boldsymbol{c}}(\boldsymbol{y}; \boldsymbol{\Theta}) - \widehat{\boldsymbol{c}}(\boldsymbol{y}; \widetilde{\boldsymbol{\Theta}})\|_2 \leq \kappa(\boldsymbol{y}) \|P_u - \widetilde{P}_u\|_2 + \sum_{k=1}^{K}\sum_{j=1}^{J}\sum_{d=1}^{D} \kappa_{k,d}^{(j)}(\boldsymbol{y}) \|\boldsymbol{\theta}_{k,d}^{(j)} - \widetilde{\boldsymbol{\theta}}_{k,d}^{(j)}\|_{(d)}$$

for

$$\kappa(\boldsymbol{y}) = z_\infty(c_1(\boldsymbol{y}) + \|P_u\|_2\|A\|_\infty\|\boldsymbol{y}\|_\infty)\widehat{c}_1^{(K,J)}(\boldsymbol{y}) + z_\infty c_2(\boldsymbol{y})$$
$$\kappa_{k,d}^{(j)}(\boldsymbol{y}) = z_\infty(c_1(\boldsymbol{y}) + \|P_u\|_2\|A\|_\infty\|\boldsymbol{y}\|_\infty)\widehat{c}_{k,j,d,2}^{(K,J)}(\boldsymbol{y})$$

where $\widehat{c}_1^{(K,J)}(\boldsymbol{y})$, $\widehat{c}_{k,j,d,2}^{(K,J)}(\boldsymbol{y})$ *are given in Proposition 4.4.13 and* $c_1(\boldsymbol{y}), c_2(\boldsymbol{y})$ *are given in Proposition 4.4.11.*

Proof. As $\rho_{c_{\max}}$ is 1-Lipschitz by Lemma 4.2.23 then using the triangle inequality

$$\begin{aligned}
&\|\widehat{\boldsymbol{c}}(\boldsymbol{y};\boldsymbol{\Theta}) - \widehat{\boldsymbol{c}}(\boldsymbol{y};\widetilde{\boldsymbol{\Theta}})\|_2 \\
&= \|\rho_{c_{\max}}(\boldsymbol{\zeta}_K \circ \mathcal{T}_{\boldsymbol{y}}(\boldsymbol{\zeta}_K; P_u)) - \rho_{c_{\max}}(\widetilde{\boldsymbol{\zeta}}_K \circ \mathcal{T}_{\boldsymbol{y}}(\widetilde{\boldsymbol{\zeta}}_K; \widetilde{P}_u))\|_2 \\
&\leq \|\boldsymbol{\zeta}_K \circ \mathcal{T}_{\boldsymbol{y}}(\boldsymbol{\zeta}_K; P_u) - \widetilde{\boldsymbol{\zeta}}_K \circ \mathcal{T}_{\boldsymbol{y}}(\widetilde{\boldsymbol{\zeta}}_K; \widetilde{P}_u)\|_2 \\
&\leq \|\boldsymbol{\zeta}_K \circ \mathcal{T}_{\boldsymbol{y}}(\boldsymbol{\zeta}_K; P_u) - \widetilde{\boldsymbol{\zeta}}_K \circ \mathcal{T}_{\boldsymbol{y}}(\boldsymbol{\zeta}_K; P_u) + \widetilde{\boldsymbol{\zeta}}_K \circ \mathcal{T}_{\boldsymbol{y}}(\boldsymbol{\zeta}_K; P_U) - \widetilde{\boldsymbol{\zeta}}_K \circ \mathcal{T}_{\boldsymbol{y}}(\widetilde{\boldsymbol{\zeta}}_K; \widetilde{P}_u)\|_2 \\
&\leq \|\mathcal{T}_{\boldsymbol{y}}(\boldsymbol{\zeta}_K; P_u)\|_\infty\|\boldsymbol{\zeta}_K - \widetilde{\boldsymbol{\zeta}}_K\|_2 + z_\infty\|\mathcal{T}_{\boldsymbol{y}}(\boldsymbol{\zeta}_K; P_u) - \mathcal{T}_{\boldsymbol{y}}(\widetilde{\boldsymbol{\zeta}}_K; \widetilde{P}_u)\|_2.
\end{aligned}$$

Using Proposition 4.4.11

$$\begin{aligned}
&\|\widehat{\boldsymbol{c}}(\boldsymbol{y};\boldsymbol{\Theta}) - \widehat{\boldsymbol{c}}(\boldsymbol{y};\widetilde{\boldsymbol{\Theta}})\|_2 \\
&\leq \|\mathcal{T}_{\boldsymbol{y}}(\boldsymbol{\zeta}_K; P_u)\|_\infty\|\boldsymbol{\zeta}_K - \widetilde{\boldsymbol{\zeta}}_K\|_2 + z_\infty\left(c_1(\boldsymbol{y})\|\boldsymbol{\zeta}_K - \widetilde{\boldsymbol{\zeta}}_K\|_\infty + c_2(\boldsymbol{y})\|P_u - \widetilde{P}_u\|_2\right) \\
&\leq \|\mathcal{T}_{\boldsymbol{y}}(\boldsymbol{\zeta}_K; P_u)\|_\infty\|\boldsymbol{\zeta}_K - \widetilde{\boldsymbol{\zeta}}_K\|_2 + z_\infty\left(c_1(\boldsymbol{y})\|\boldsymbol{\zeta}_K - \widetilde{\boldsymbol{\zeta}}_K\|_2 + c_2(\boldsymbol{y})\|P_u - \widetilde{P}_u\|_2\right) \\
&\leq \left(\|\mathcal{T}_{\boldsymbol{y}}(\boldsymbol{\zeta}_K; P_u)\|_\infty + z_\infty c_1(\boldsymbol{y})\right)\|\boldsymbol{\zeta}_K - \widetilde{\boldsymbol{\zeta}}_K\|_2 + z_\infty c_2(\boldsymbol{y})\|P_u - \widetilde{P}_u\|_2.
\end{aligned}$$

Now, using Proposition 4.4.12

$$\begin{aligned}
&\|\widehat{\boldsymbol{c}}(\boldsymbol{y};\boldsymbol{\Theta}) - \widehat{\boldsymbol{c}}(\boldsymbol{y};\widetilde{\boldsymbol{\Theta}})\|_2 \\
&\leq \left(z_\infty\|P_u\|_2\|A\|_\infty\|\boldsymbol{y}\|_\infty + z_\infty c_1(\boldsymbol{y})\right)\|\boldsymbol{\zeta}_K - \widetilde{\boldsymbol{\zeta}}_K\|_2 + z_\infty c_2(\boldsymbol{y})\|P_u - \widetilde{P}_u\|_2 \\
&= z_\infty\left(\|P_u\|_2\|A\|_\infty\|\boldsymbol{y}\|_\infty + c_1(\boldsymbol{y})\right)\|\boldsymbol{\zeta}_K - \widetilde{\boldsymbol{\zeta}}_K\|_2 + z_\infty c_2(\boldsymbol{y})\|P_u - \widetilde{P}_u\|_2
\end{aligned}$$

Finally, using Proposition 4.4.13

$$
\begin{aligned}
&\|\widehat{\boldsymbol{c}}(\boldsymbol{y};\boldsymbol{\Theta}) - \widehat{\boldsymbol{c}}(\boldsymbol{y};\widetilde{\boldsymbol{\Theta}})\|_2 \\
&\leq z_\infty \left(\|P_u\|_2\|A\|_\infty\|\boldsymbol{y}\|_\infty + c_1(\boldsymbol{y})\right)\left(\widehat{c}_1^{(K,J)}(\boldsymbol{y})\|P_u - \widetilde{P}_u\|_2 + \sum_{k=1}^{K}\sum_{j=1}^{J}\sum_{d=1}^{D}\widehat{c}_{k,j,d,2}^{(K,J)}(\boldsymbol{y})\|\boldsymbol{\theta}_{k,d}^{(j)} - \widetilde{\boldsymbol{\theta}}_{k,d}^{(j)}\|_{(d)}\right) \\
&\quad + z_\infty c_2(\boldsymbol{y})\|P_u - \widetilde{P}_u\|_2 \\
&= \left(z_\infty \left(\|P_u\|_2\|A\|_\infty\|\boldsymbol{y}\|_\infty + c_1(\boldsymbol{y})\right)\widehat{c}_1^{(K,J)} + z_\infty c_2(\boldsymbol{y})\right)\|P_u - \widetilde{P}_u\|_2 \\
&\quad + z_\infty \left(\|P_u\|_2\|A\|_\infty\|\boldsymbol{y}\|_\infty + c_1(\boldsymbol{y})\right)\sum_{k=1}^{K}\sum_{j=1}^{J}\sum_{d=1}^{D}\widehat{c}_{k,j,d,2}^{(K,J)}(\boldsymbol{y})\|\boldsymbol{\theta}_{k,d}^{(j)} - \widetilde{\boldsymbol{\theta}}_{k,d}^{(j)}\|_{(d)} \\
&= \kappa(\boldsymbol{y})\|P_u - \widetilde{P}_u\|_2 + \sum_{k=1}^{K}\sum_{j=1}^{J}\sum_{d=1}^{D}\kappa_{k,d}^{(j)}(\boldsymbol{y})\|\boldsymbol{\theta}_{k,d}^{(j)} - \widetilde{\boldsymbol{\theta}}_{k,d}^{(j)}\|_{(d)}. \qquad \boxtimes
\end{aligned}
$$

Theorem 4.4.14 provides that the difference in the G-CG-Net outputs given two different parameterizations depends directly on the discrepancy in the parameters. This property allows for representing the covering numbers of the G-CG-Net hypothesis class in terms of covering numbers for the parameter space by Corollary 4.2.19. As the parameter spaces are subsets of finite-dimensional and real-valued vector spaces, the covering numbers for the parameter spaces are more tractable to calculate and bound than the covering numbers for hypothesis class of functions.

4.4.3 Constructing a Generalization Error Bound

Define $\mathcal{P}_{\text{const}} \subset \mathcal{P}_{\text{diag}} \subset \mathcal{P}_{\text{tri}} \subset \mathcal{P}_{\text{full}}$ to be the options for $\mathcal{P}$, in Assumption 4.4.1, corresponding to the vector spaces of constant, diagonal, tridiagonal, and full covariance matrices with bounded spectrum, respectively. Additionally, let each scale variable update parameter, $\boldsymbol{\theta}_{k,d}^{(j)}$, be of dimension $\alpha_d \geq 0$ (i.e. $\boldsymbol{\theta}_{k,d}^{(j)} \in \mathbb{R}^{\alpha_d}$) and define the sets

$$
\Omega_d = \{\boldsymbol{\theta} \in \mathbb{R}^{\alpha_d} : \|\boldsymbol{\theta}\|_{(d)} \leq \omega_d\} \tag{4.23}
$$

for ω_d given in Assumption 4.4.1. Then the hypothesis class for G-CG-Net is

$$\mathcal{H}_{\text{CG}}^{(1)} = \left\{ \widehat{\boldsymbol{c}}\left(\cdot; \{P_u, \boldsymbol{\theta}_{k,d}^{(j)}\}_{\substack{k\in\mathbb{N}[K]\\ d\in\mathbb{N}[D]}}^{j\in\mathbb{N}[J]}\right) : P_u \in \mathcal{P}, \boldsymbol{\theta}_{k,d}^{(j)} \in \Omega_d \right\}.$$

Theorem 4.4.15 (Generalization Error Bound for G-CG-Net)**.** *Let $\mathcal{S} = \{(\overline{\boldsymbol{y}}_i, \overline{\boldsymbol{c}}_i)\}_{i=1}^{N_s}$ be a training dataset where each $(\overline{\boldsymbol{c}}_i, \overline{\boldsymbol{y}}_i)$ is given by (4.12) and define $y_{\max} = \max_{1\leq i\leq N_s} \|\overline{\boldsymbol{y}}_i\|_2$. If Assumption 4.4.1, 4.4.2, and 4.4.3 hold then with probability at least $1-\varepsilon$, for all $\widehat{\boldsymbol{c}} \in \mathcal{H}_{\text{CG}}^{(1)}$, the generalization error of G-CG-Net is bounded as*

$$\begin{aligned}\mathcal{L}(\widehat{\boldsymbol{c}}) \leq \mathcal{L}_{\mathcal{S}}(\widehat{\boldsymbol{c}}) &+ \frac{8\tau c_{\max}}{\sqrt{N_s}}\sqrt{\dim(\mathcal{P})}\sqrt{\ln\left(e\left(1+\frac{4p_{\max}(KJD+1)\kappa}{c_{\max}}\right)\right)}\\ &+ \frac{8\tau c_{\max}}{\sqrt{N_s}}\sum_{k=1}^{K}\sum_{j=1}^{J}\sum_{d=1}^{D}\sqrt{\alpha_d \ln\left(e\left(1+\frac{4\omega_d(KJD+1)\kappa_{k,d}^{(j)}}{c_{\max}}\right)\right)}\\ &+ 4c\sqrt{2\ln(4/\varepsilon)/N_s}\end{aligned}$$

for $\dim(\mathcal{P}) = 1, n, 2n-1$, *or* $n(n+1)/2$ *when* $\mathcal{P} = \mathcal{P}_{\text{const}}$, $\mathcal{P} = \mathcal{P}_{\text{diag}}$, $\mathcal{P} = \mathcal{P}_{\text{tri}}$, *or* $\mathcal{P} = \mathcal{P}_{\text{full}}$, *respectively. Additionally,*

$$\kappa = z_\infty(c_1 + p_{\max}y_{\max}\|A\|_\infty)\widehat{c}_1^{(K,J)} + z_\infty c_2 \tag{4.24}$$

$$\kappa_{k,d}^{(j)} = z_\infty(c_1 + p_{\max}y_{\max}\|A\|_\infty)\widehat{c}_{k,j,d,2}^{(K,J)} \tag{4.25}$$

where

$$\begin{aligned} c_1 &= p_{\max}y_{\max}\ \|A\|_2\left(1+2z_\infty^2 p_{\max}\ \|A\|_2^2\right)\\ c_2 &= z_\infty y_{\max}\|A\|_2\left(p_{\max}/p_{\min}\right)^2\\ \widehat{c}_1^{(K,J)} &= c_2\sum_{k=1}^{K}\widehat{r}_{k,2}^{(J)}\prod_{\ell=k+1}^{K}(\widehat{r}_{\ell,1}^{(J)} + \widehat{r}_{\ell,2}^{(J)}c_1)\\ \widehat{c}_{k,j,d,2}^{(K,J)} &= \widehat{r}_{k,d,3}^{(j,J)}\prod_{\ell=k+1}^{K}(\widehat{r}_{\ell,1}^{(J)} + \widehat{r}_{\ell,2}^{(J)}c_1)\end{aligned}$$

for $\widehat{r}_{k,1}^{(J)} = \prod_{j=1}^{J} r_{k,1}^{(j-1)}, \quad \widehat{r}_{k,2}^{(J)} = \sum_{j=1}^{J} r_{k,2}^{(j-1)} \prod_{\ell=j}^{J-1} r_{k,1}^{(\ell)}, \quad$ *and* $\quad \widehat{r}_{k,d,3}^{(j,J)} = r_{k,d,3}^{(j-1)} \prod_{\ell=j}^{J-1} r_{k,1}^{(\ell)} \quad$ *are as given in Proposition 4.4.6.*

As an overview, proving Theorem 4.4.15 consists of the following steps: **Step (1)**, we establish a Rademacher process generated by the hypothesis class, $\mathcal{H}_{\text{CG}}^{(1)}$. **Step (2)**, using Lemma 4.2.7, we express the Rademacher complexity of $L \circ \mathcal{H}_{\text{CG}}^{(1)}$, for L the G-CG-Net loss function, as the expected value of the supremum of the established Rademacher process. **Step (3)**, invoking Dudley's inequality we bound the expected value of the supremum of the established Rademacher process in terms of an integral of some covering numbers. **Step (4)**, we use the Lipschitz property of Theorem 4.4.14 together with Corollary 4.2.19 to bound the covering numbers from Dudley's inequality by covering numbers of the G-CG-Net parameter spaces. **Step (5)**, we use Lemma 4.2.17 to further bound the covering numbers of the G-CG-Net parameter spaces. **Step (6)**, using the bounds of Step (5), we bound Dudley's inequality by evaluable integrals where evaluation by Lemma 4.2.29, further simplification, and then the use of Theorem 4.2.6 produces the desired GEB for G-CG-Net.

Proof of Theorem 4.4.15. Let

$$\overline{Y} \coloneqq [\overline{\boldsymbol{y}}_1, \ldots, \overline{\boldsymbol{y}}_{N_s}] \in \mathbb{R}^{m \times N_s}$$

and

$$\begin{aligned} \mathcal{M}_{\text{CG}}^{(1)} &\coloneqq \{M_{\widehat{\boldsymbol{c}}} = [\widehat{\boldsymbol{c}}(\overline{\boldsymbol{y}}_1), \ldots, \widehat{\boldsymbol{c}}(\overline{\boldsymbol{y}}_{N_s})] : \widehat{\boldsymbol{c}} \in \mathcal{H}_{\text{CG}}^{(1)}\} \\ &= \left\{ \widehat{\boldsymbol{c}}\left(\overline{Y}; \boldsymbol{\Theta} = \left\{P, \boldsymbol{\theta}_{k,d}^{(j)}\right\}_{\substack{k \in \mathbb{N}[K] \\ d \in \mathbb{N}[D]}}^{j \in \mathbb{N}[J]}\right) : P \in \mathcal{P}_{\text{full}}, \boldsymbol{\theta}_{k,d}^{(j)} \in \Omega_d \right\} \end{aligned}$$

where

$$\widehat{\boldsymbol{c}}\left(\overline{Y}; \boldsymbol{\Theta}\right) \coloneqq [\widehat{\boldsymbol{c}}\left(\overline{\boldsymbol{y}}_1; \boldsymbol{\Theta}\right), \ldots, \widehat{\boldsymbol{c}}\left(\overline{\boldsymbol{y}}_{N_s}; \boldsymbol{\Theta}\right)] = \begin{bmatrix} \widehat{c}_1(\overline{\boldsymbol{y}}_1) & \cdots & \widehat{c}_1(\overline{\boldsymbol{y}}_{N_s}) \\ \vdots & \ddots & \vdots \\ \widehat{c}_n(\overline{\boldsymbol{y}}_1) & \cdots & \widehat{c}_n(\overline{\boldsymbol{y}}_{N_s}) \end{bmatrix} \in \mathbb{R}^{n \times N_s}.$$

Step (1). Define the Rademacher process $(X_{M_{\widehat{\boldsymbol{c}}}})_{M_{\widehat{\boldsymbol{c}}}\in\mathcal{M}_{\text{CG}}^{(1)}}$ as

$$X_{M_{\widehat{\boldsymbol{c}}}} \coloneqq \sum_{i=1}^{N_s}\sum_{k=1}^{n}\gamma_{ik}[M_{\widehat{\boldsymbol{c}}}]_{ki} = \sum_{i=1}^{N_s}\sum_{k=1}^{n}\gamma_{ik}\widehat{c}_k(\overline{\boldsymbol{y}}_i) \tag{4.26}$$

for Rademacher variables $\Gamma = [\gamma_{ik}]_{i\in\mathbb{N}[N_s]}^{k\in\mathbb{N}[n]}$.

Step (2). Let L be any G-CG-Net loss function satisfying Assumption 4.4.2 for bound c and Lipschitz constant τ (e.g. SSIM loss, mean absolute error, etc.). Given a hypothesis $\widehat{\boldsymbol{c}} \in \mathcal{H}_{\text{CG}}^{(1)}$ and data point $(\overline{\boldsymbol{y}}_i, \overline{\boldsymbol{c}}_i) \in \mathcal{S}$, we write $L\circ\widehat{\boldsymbol{c}}(\overline{\boldsymbol{y}}_i) = L(\widehat{\boldsymbol{c}}(\overline{\boldsymbol{y}}_i), \overline{\boldsymbol{c}}_i)$ to be the evaluation of the loss function at the estimate $\widehat{\boldsymbol{c}}(\overline{\boldsymbol{y}}_i)$. Now, the empirical Rademacher complexity, $\mathcal{R}_{\mathcal{S}}$, of the set of functions $L\circ\mathcal{H}_{\text{CG}}^{(1)}$ is

$$\mathcal{R}_{\mathcal{S}}(L\circ\mathcal{H}_{\text{CG}}^{(1)}) = \mathbb{E}_{\boldsymbol{\gamma}}\left[\sup_{\widehat{\boldsymbol{c}}\in\mathcal{H}_{\text{CG}}^{(1)}}\frac{1}{N_s}\sum_{i=1}^{N_s}\gamma_i L\circ\widehat{\boldsymbol{c}}(\overline{\boldsymbol{y}}_i)\right] = \frac{1}{N_s}\mathbb{E}_{\boldsymbol{\gamma}}\left[\sup_{\widehat{\boldsymbol{c}}\in\mathcal{H}_{\text{CG}}^{(1)}}\sum_{i=1}^{N_s}\gamma_i L\circ\widehat{\boldsymbol{c}}(\overline{\boldsymbol{y}}_i)\right]$$

for $\boldsymbol{\gamma} = [\gamma_i]_{i\in\mathbb{N}[N_s]}$ a vector of Rademacher variables. Next, using (4.26) and Lemma 4.2.7 with $\mathcal{H} = \mathcal{H}_{\text{CG}}^{(1)}$ and each $g_i(\boldsymbol{x}) = L(\boldsymbol{x}, \overline{\boldsymbol{c}}_i)$ we have

$$\mathcal{R}_{\mathcal{S}}(L\circ\mathcal{H}_{\text{CG}}^{(1)}) \leq \frac{\sqrt{2}\tau}{N_s}\mathbb{E}_{\Gamma}\left[\sup_{\widehat{\boldsymbol{c}}\in\mathcal{H}_{\text{CG}}^{(1)}}\sum_{i=1}^{N_s}\sum_{k=1}^{n}\gamma_{ik}\widehat{c}_k(\overline{\boldsymbol{y}}_i)\right] = \frac{\sqrt{2}\tau}{N_s}\mathbb{E}_{\Gamma}\left(\sup_{\widehat{\boldsymbol{c}}\in\mathcal{H}_{\text{CG}}^{(1)}} X_{M_{\widehat{\boldsymbol{c}}}}\right). \tag{4.27}$$

Step (3). As $(X_{M_{\widehat{\boldsymbol{c}}}})_{M_{\widehat{\boldsymbol{c}}}\in\mathcal{M}_{\text{CG}}^{(1)}}$ is a real-valued Rademacher process then note by Lemma 4.2.15

$$\widetilde{d}(X_{M_{\widehat{\boldsymbol{c}}_1}}, X_{M_{\widehat{\boldsymbol{c}}_2}})^2 = \sum_{i=1}^{N_s}\sum_{k=1}^{n}([M_{\widehat{\boldsymbol{c}}_1}]_{ki} - [M_{\widehat{\boldsymbol{c}}_2}]_{ki})^2 = \|M_{\widehat{\boldsymbol{c}}_1} - M_{\widehat{\boldsymbol{c}}_2}\|_F^2$$

for $\|\cdot\|_F$ the Frobenius norm. Observe for any $M_{\widehat{\boldsymbol{c}}} \in \mathcal{M}_{\text{CG}}^{(1)}$

$$\|M_{\widehat{\boldsymbol{c}}}\|_F = \|\widehat{\boldsymbol{c}}(\overline{Y};\boldsymbol{\Theta})\|_F = \sqrt{\sum_{i=1}^{N_s}\|\widehat{\boldsymbol{c}}(\overline{\boldsymbol{y}}_i;\boldsymbol{\Theta})\|_2^2} \leq \sqrt{N_s}c_{\max}$$

where we used that any output from G-CG-Net is bounded, in Euclidean norm, by $c_{\max}$. Hence, by Lemma 4.2.15, the radius of $\mathcal{M}_{\text{CG}}^{(1)}$, denoted $\Delta(\mathcal{M}_{\text{CG}}^{(1)})$ and defined in Definition 4.2.12, is

$$\Delta(\mathcal{M}_{\text{CG}}^{(1)}) = \sup_{M_{\widehat{\boldsymbol{c}}} \in \mathcal{M}_{\text{CG}}^{(1)}} \sqrt{\mathbb{E}|X_{M_{\widehat{\boldsymbol{c}}}}|^2} = \sup_{\widehat{\boldsymbol{c}} \in \mathcal{H}_{\text{CG}}^{(1)}} \sqrt{\sum_{i=1}^{N_s} \sum_{k=1}^{n} |[M_{\widehat{\boldsymbol{c}}_1}]_{ki}|^2} = \sup_{\widehat{\boldsymbol{c}} \in \mathcal{H}_{\text{CG}}^{(1)}} \|M_{\widehat{\boldsymbol{c}}}\|_F \leq \sqrt{N_s} c_{\max}.$$

Therefore, using Dudley's Inequality, from Lemma 4.2.13, and (4.27)

$$\begin{aligned}
\mathcal{R}_{\mathcal{S}}(L \circ \mathcal{H}_{\text{CG}}^{(1)}) &\leq \frac{\sqrt{2}\tau}{N_s} \mathbb{E}_{\Gamma}\Big(\sup_{\widehat{\boldsymbol{c}} \in \mathcal{H}_{\text{CG}}^{(1)}} X_{M_{\widehat{\boldsymbol{c}}}} \Big) \\
&\leq \frac{8\tau}{N_s} \int_0^{\Delta(\mathcal{M}_{\text{CG}}^{(1)})/2} \sqrt{\ln\Big(\mathcal{N}(\mathcal{M}_{\text{CG}}^{(1)}, \|\cdot\|_F, \epsilon)\Big)} d\epsilon \\
&\leq \frac{8\tau}{N_s} \int_0^{\frac{\sqrt{N_s} c_{\max}}{2}} \sqrt{\ln\Big(\mathcal{N}(\mathcal{M}_{\text{CG}}^{(1)}, \|\cdot\|_F, \epsilon)\Big)} d\epsilon. \qquad (4.28)
\end{aligned}$$

Step (4). Let $\boldsymbol{\Theta} = \{P\} \cup \{\boldsymbol{\theta}_{k,d}^{(j)}\}_{k \in \mathbb{N}[K], d \in \mathbb{N}[D]}^{j \in \mathbb{N}[J]}$ and $\widetilde{\boldsymbol{\Theta}} = \{\widetilde{P}\} \cup \{\widetilde{\boldsymbol{\theta}}_{k,d}^{(j)}\}_{k \in \mathbb{N}[K], d \in \mathbb{N}[D]}^{j \in \mathbb{N}[J]}$ be any two possible parameterizations of G-CG-Net. By Theorem 4.4.14 for any $i \in \mathbb{N}[N_s]$

$$\|\widehat{\boldsymbol{c}}(\overline{\boldsymbol{y}}_i; \boldsymbol{\Theta}) - \widehat{\boldsymbol{c}}(\overline{\boldsymbol{y}}_i; \widetilde{\boldsymbol{\Theta}})\|_2 \leq \kappa \|P_u - \widetilde{P}_u\|_2 + \sum_{k=1}^{K} \sum_{j=1}^{J} \sum_{d=1}^{D} \kappa_{k,d}^{(j)} \|\boldsymbol{\theta}_{k,d}^{(j)} - \widetilde{\boldsymbol{\theta}}_{k,d}^{(j)}\|_{(d)}.$$

As κ and $\kappa_{k,d}^{(j)}$, given respectively in (4.24) and (4.25), are independent of $\overline{\boldsymbol{y}}_i$ then

$$\|\widehat{\boldsymbol{c}}(\overline{Y}; \boldsymbol{\Theta}) - \widehat{\boldsymbol{c}}(\overline{Y}; \widetilde{\boldsymbol{\Theta}})\|_F = \sqrt{\sum_{i=1}^{N_s} \|\boldsymbol{c}(\overline{\boldsymbol{y}}_i; \boldsymbol{\Theta}) - \boldsymbol{c}(\overline{\boldsymbol{y}}_i; \widetilde{\boldsymbol{\Theta}})\|_2^2} \leq \sqrt{N_s} \|\widehat{\boldsymbol{c}}(\overline{\boldsymbol{y}}_j; \boldsymbol{\Theta}) - \widehat{\boldsymbol{c}}(\overline{\boldsymbol{y}}_j; \widetilde{\boldsymbol{\Theta}})\|_2$$

for any $j \in \mathbb{N}[N_s]$. Thus

$$\|\widehat{\boldsymbol{c}}(\overline{Y}; \boldsymbol{\Theta}) - \widehat{\boldsymbol{c}}(\overline{Y}; \widetilde{\boldsymbol{\Theta}})\|_F \leq \sqrt{N_s} \kappa \|P_u - \widetilde{P}_u\|_2 + \sqrt{N_s} \sum_{k=1}^{K} \sum_{j=1}^{J} \sum_{d=1}^{D} \kappa_{k,d}^{(j)} \|\boldsymbol{\theta}_{k,d}^{(j)} - \widetilde{\boldsymbol{\theta}}_{k,d}^{(j)}\|_{(d)}.$$

Define $q(k, j, d) : \mathbb{N}[K] \times \mathbb{N}[J] \times \mathbb{N}[D] \to \{2, 3, \ldots, KJD + 1\}$ as $q(k, j, d) = (k-1)JD + (j-1)D + d + 1$, which coverts three input indices to a single unique index. As $P_u, \widetilde{P}_u \in \mathcal{P}$

and $\boldsymbol{\theta}_{k,d}^{(j)}, \widetilde{\boldsymbol{\theta}}_{k,d}^{(j)} \in \Omega_d$, then from Corollary 4.2.19 where $n = KJD + 1, \Theta_1 = \mathcal{P}, \vartheta_1 = \|\cdot\|_2$, $\Theta_{q(k,j,d)} = \Omega_d, \vartheta_{q(k,j,d)} = \|\cdot\|_{(d)}, \mathcal{F} = \mathcal{M}_{\text{CG}}^{(1)}, \|\cdot\|_{\mathcal{F}} = \|\cdot\|_F, \Gamma_1 = \sqrt{N_s}\kappa$, and $\Gamma_{q(k,j,d)} = \sqrt{N_s}\kappa_{k,d}^{(j)}$ we have

$$\begin{aligned}
&\mathcal{N}\left(\mathcal{M}_{\text{CG}}^{(1)}, \|\cdot\|_F, \epsilon\right) \\
&\leq \mathcal{N}\left(\Theta_1, \vartheta_1, \frac{\epsilon}{(KJD+1)\Gamma_1}\right) \prod_{k=1}^{K}\prod_{j=1}^{J}\prod_{d=1}^{D} \mathcal{N}\left(\Theta_{q(k,j,d)}, \vartheta_{q(k,j,d)}, \frac{\epsilon}{(KJD+1)\Gamma_{q(k,j,d)}}\right) \\
&= \mathcal{N}\left(\mathcal{P}, \|\cdot\|_2, \frac{\epsilon}{\sqrt{N_s}\kappa(KJD+1)}\right) \prod_{k=1}^{K}\prod_{j=1}^{J}\prod_{d=1}^{D} \mathcal{N}\left(\Omega_d, \|\cdot\|_{(d)}, \frac{\epsilon}{\sqrt{N_s}\kappa_{k,d}^{(j)}(KJD+1)}\right).
\end{aligned}$$

Hence

$$\begin{aligned}
\ln\left(\mathcal{N}\left(\mathcal{M}_{\text{CG}}^{(1)}, \|\cdot\|_F, \epsilon\right)\right) \leq \ln\left(\mathcal{N}\left(\mathcal{P}, \|\cdot\|_2, \frac{\epsilon}{\sqrt{N_s}\kappa(KJD+1)}\right)\right) \\
+ \sum_{k=1}^{K}\sum_{j=1}^{J}\sum_{d=1}^{D} \ln\left(\mathcal{N}\left(\Omega_d, \|\cdot\|_{(d)}, \frac{\epsilon}{\sqrt{N_s}\kappa_{k,d}^{(j)}(KJD+1)}\right)\right). \quad (4.29)
\end{aligned}$$

Step (5). As $\mathcal{P}$, defined in (4.19), contains symmetric $n \times n$ matrices then for $\mathcal{P} = \mathcal{P}_{\text{full}}$ only $n(n+1)/2$ entries are required to uniquely define a matrix and $\dim(\mathcal{P}_{\text{full}}) = n(n+1)/2$. Similarly, $\dim(\mathcal{P}_{\text{tri}}) = 2n - 1$, $\dim(\mathcal{P}_{\text{diag}}) = n$, and $\dim(\mathcal{P}_{\text{const}}) = 1$. Furthermore, as $\mathcal{P}/p_{\max} = \{P/p_{\max} : P \in \mathcal{P}\}$, where $p_{\max}$ is given in (4.19), is contained in the $\|\cdot\|_2$ unit ball, then, using Lemma 4.2.17, observe

$$\mathcal{N}\left(\mathcal{P}, \|\cdot\|_2, \epsilon\right) = \mathcal{N}\left(\frac{\mathcal{P}}{p_{\max}}, \|\cdot\|_2, \frac{\epsilon}{p_{\max}}\right) \leq \left(1 + \frac{2p_{\max}}{\epsilon}\right)^{\dim(\mathcal{P})}.$$

Similarly, as Ω_d/ω_d is contained in the $\|\cdot\|_{(d)}$ unit ball then, using Lemma 4.2.17, we have

$$\mathcal{N}\left(\Omega_d, \|\cdot\|_{(d)}, \epsilon\right) \leq \left(1 + \frac{2\omega_d}{\epsilon}\right)^{\alpha_d}.$$

Combining these two observations with (4.29) gives

$$\ln\left(\mathcal{N}\left(\mathcal{M}_{\text{CG}}^{(1)}, \|\cdot\|_F, \epsilon\right)\right) \leq \dim(\mathcal{P}) \ln\left(1 + \frac{2p_{\max}\sqrt{N_s}\kappa(KJD+1)}{\epsilon}\right) + \sum_{k=1}^{K}\sum_{j=1}^{J}\sum_{d=1}^{D} \alpha_d \ln\left(1 + \frac{2\omega_d\sqrt{N_s}\kappa_{k,d}^{(j)}(KJD+1)}{\epsilon}\right). \tag{4.30}$$

Step (6). Taking the square root of and then integrating over both sides of the inequality in equation (4.30) then applying Lemma 4.2.29 and the subadditivity of square roots produces, for any $\beta > 0$,

$$\begin{aligned}
&\int_0^\beta \sqrt{\ln\left(\mathcal{N}\left(\mathcal{M}_{\text{CG}}^{(1)}, \|\cdot\|_F, \epsilon\right)\right)}d\epsilon \\
&\quad\leq \sqrt{\dim(\mathcal{P})}\int_0^\beta \sqrt{\ln\left(1 + \frac{2p_{\max}\sqrt{N_s}\kappa(KJD+1)}{\epsilon}\right)}d\epsilon \\
&\qquad + \sum_{k=1}^{K}\sum_{j=1}^{J}\sum_{d=1}^{D}\sqrt{\alpha_d}\int_0^\beta \sqrt{\ln\left(1 + \frac{2\omega_d\sqrt{N_s}\kappa_{k,d}^{(j)}(KJD+1)}{\epsilon}\right)}d\epsilon \\
&\quad\leq \sqrt{\dim(\mathcal{P})}\beta\sqrt{\ln\left(e\left(1 + \frac{2p_{\max}\sqrt{N_s}\kappa(KJD+1)}{\beta}\right)\right)} \\
&\qquad + \sum_{k=1}^{K}\sum_{j=1}^{J}\sum_{d=1}^{D}\sqrt{\alpha_d}\beta\sqrt{\ln\left(e\left(1 + \frac{2\omega_d\sqrt{N_s}\kappa_{k,d}^{(j)}(KJD+1)}{\beta}\right)\right)}.
\end{aligned} \tag{4.31}$$

Combining equation (4.31) and equation (4.28) produces

$$
\begin{aligned}
&\mathcal{R}_{\mathcal{S}}(L \circ \mathcal{H}_{\text{CG}}^{(1)}) \\
&\quad \leq \frac{8\tau}{N_s} \int_0^{\frac{\sqrt{N_s}c_{\max}}{2}} \sqrt{\ln\left(\mathcal{N}(\mathcal{M}_{\text{CG}}^{(1)}, \|\cdot\|_F, \epsilon\right)} d\epsilon \\
&\quad \leq \frac{8\tau}{N_s} \sqrt{\dim(\mathcal{P})} \frac{\sqrt{N_s}c_{\max}}{2} \sqrt{\ln\left(e\left(1 + \frac{2p_{\max}\sqrt{N_s}\kappa(KJD+1)}{\frac{\sqrt{N_s}c_{\max}}{2}}\right)\right)} \\
&\qquad + \frac{8\tau}{N_s} \sum_{k=1}^{K}\sum_{j=1}^{J}\sum_{d=1}^{D} \sqrt{\alpha_d} \frac{\sqrt{N_s}c_{\max}}{2} \sqrt{\ln\left(e\left(1 + \frac{2\omega_d\sqrt{N_s}\kappa_{k,d}^{(j)}(KJD+1)}{\frac{\sqrt{N_s}c_{\max}}{2}}\right)\right)} \\
&\quad = \frac{4\tau c_{\max}}{\sqrt{N_s}} \sqrt{\dim(\mathcal{P})} \sqrt{\ln\left(e\left(1 + \frac{4p_{\max}\kappa(KJD+1)}{c_{\max}}\right)\right)} \\
&\qquad + \frac{4\tau c_{\max}}{\sqrt{N_s}} \sum_{k=1}^{K}\sum_{j=1}^{J}\sum_{d=1}^{D} \sqrt{\alpha_d} \sqrt{\ln\left(e\left(1 + \frac{4\omega_d\kappa_{k,d}^{(j)}(KJD+1)}{c_{\max}}\right)\right)}.
\end{aligned} \tag{4.32}
$$

Finally, inserting (4.32) into Theorem 4.2.6, then for $\varepsilon \in (0,1)$ with probability at least $1-\varepsilon$ we have for any $\widehat{\boldsymbol{c}} \in \mathcal{H}_{\text{CG}}^{(1)}$

$$
\begin{aligned}
\mathcal{L}(\widehat{\boldsymbol{c}}) &\leq \mathcal{L}_{\mathcal{S}}(\widehat{\boldsymbol{c}}) + 2\mathcal{R}_{\mathcal{S}}(L \circ \mathcal{H}_{\text{CG}}^{(1)}) + 4c\sqrt{2\ln(4/\varepsilon)/N_s} \\
&\leq \mathcal{L}_{\mathcal{S}}(\widehat{\boldsymbol{c}}) + \frac{8\tau c_{\max}}{\sqrt{N_s}} \sqrt{\dim(\mathcal{P})} \sqrt{\ln\left(e\left(1 + \frac{4p_{\max}\kappa(KJD+1)}{c_{\max}}\right)\right)} \\
&\quad + \frac{8\tau c_{\max}}{\sqrt{N_s}} \sum_{k=1}^{K}\sum_{j=1}^{J}\sum_{d=1}^{D} \sqrt{\alpha_d} \sqrt{\ln\left(e\left(1 + \frac{4\omega_d\kappa_{k,d}^{(j)}(KJD+1)}{c_{\max}}\right)\right)} \\
&\quad + 4c\sqrt{2\ln(4/\varepsilon)/N_s}
\end{aligned}
$$

producing the desired generalization error bound. ⊠

Discussion

Ideally, through training G-CG-Net a hypothesis $\widehat{\boldsymbol{c}} \in \mathcal{H}_{\text{CG}}^{(1)}$ is found such that the empirical loss, $\mathcal{L}_{\mathcal{S}}(\widehat{\boldsymbol{c}})$, is minimized, but any $\widehat{\boldsymbol{c}}$ possibly generated by G-CG-Net could be used in the GEB of Theorem 4.4.15. Early stopping while training is of particular interest as the generated $\widehat{\boldsymbol{c}}$ does not

optimize $\mathcal{L}_{\mathcal{S}}$, but the GEB of Theorem 4.4.15 would still apply to $\widehat{\boldsymbol{c}}$. For instance, as discussed in Chapter 2 and Chapter 3, both CG-Net and DR-CG-Net are training using early stopping and for other DNNs it may be advantageous to used early stopping to prevent overfitting especially when small sets of training data are available.

Additionally, we remark for noiseless measurements, that is, $\overline{\boldsymbol{y}}_i = A\overline{\boldsymbol{c}}_i$, that $y_{\max} \leq c_{\max}\|A\|_2$ for any set of training data. For white noise measurements, that is, $\overline{\boldsymbol{y}}_i = A\overline{\boldsymbol{c}}_i + \boldsymbol{\nu}$ where $\boldsymbol{\nu} \sim \mathcal{N}(0, \sigma^2 I)$, then $y_{\max} \leq c_{\max}\|A\|_2 + z_p\sigma$ with high probability for large z_p. For instance, using the quantile function of a normal distribution, $y_{\max} \leq c_{\max}\|A\|_2 + 6.11\sigma$ with probability $(1 - 2 \times 10^{-9})^{mN_s} \approx 1 - 2mN_s \times 10^{-9}$. Furthermore, as discussed in Chapter 2 for the pure Newton formulation of CG-Net, denoted nCG-Net, there are advantages to scaling the input measurements provided to CG-Net. Hence, the use of scaling may be an additional tool to bound $y_{\max}$.

To determine the other parameters of the GEB in Theorem 4.4.15:

- τ and c can typically be easily calculated for any chosen loss function used to train G-CG-Net.
- N_s is fixed by the size of the training data set.
- $c_{\max}$ may be determined by the preprocessing implemented on the training and testing data. Alternatively, $c_{\max}$ can be set as $c_{\max} = \max_{1 \leq i \leq N_s} \|\overline{\boldsymbol{c}}_i\|_2$.
- z_∞ could be taken equal to $c_{\max}$ or chosen empirically, through a series of short training experiments of G-CG-Net with varying choices of z_∞, to optimize the performance of G-CG-Net.
- $\|A\|_2$ and $\|A\|_\infty$ can easily be calculated once the measurement model used for the experimentation is specified.
- $\dim(\mathcal{P})$ is fixed by the chosen covariance matrix structure used in training G-CG-Net.
- α_d is fixed by the choice of scale variable update method. Specifically, α_d is equal to the dimension of the parameters that are present in each unrolled scale variable update method.

- $p_{\max}$, $p_{\min}$, and ω_d can be set by providing any maximum norm bound requirement on the parameters when training. For example, after updating $\boldsymbol{\theta}_{k,d}^{(j)}$ via stochastic gradient descent we can project $\boldsymbol{\theta}_{k,d}^{(j)}$ onto the ball of radius ω_d as measured by a norm $\|\cdot\|_{(d)}$. Note, by implementation of the covariance matrix $p_{\min}$ is set at ϵ where $\epsilon > 0$ is the stabilizing parameter guaranteeing a positive definite covariance matrix.

- $r_{k,1}^{(j)}, r_{k,2}^{(j)}$, and $r_{k,d,3}^{(j,J)}$ would need to be determined from the choice of unrolled scale variable update method. Specifically, showing Assumption 4.4.3 holds for the desired unrolled scale variable update method provides upper bounds on the parameters $r_{k,1}^{(j)}, r_{k,2}^{(j)}$, and $r_{k,d,3}^{(j,J)}$. Likely these parameters depend on the other parameters already defined above.

- κ and $\kappa_{k,d}^{(j)}$ can both be readily calculated given all other parameters already defined above.

4.4.4 Alternative Generalization Error Bounds

Tighter Logarithmic Integral Bound

The GEB provided here is a tighter bound than that of Theorem 4.4.15, which results from removing the simplifying bound in equation (4.10) of Lemma 4.2.29. That is, we will use the tighter, but slightly more complex, integral bound of Lemma 4.2.30 rather than the integral bound of Lemma 4.2.29. Specifically, examining equation (4.10) we note that

$$\int_{\nu/\beta}^{\infty} \frac{1}{u} \frac{1}{1+u} du = \ln\left(1 + \beta/\nu\right).$$

Hence, in exactly evaluating the integral of $\frac{1}{u}\frac{1}{1+u}$ over $[\nu/\beta, \infty)$, as in Lemma 4.2.30, we obtain logarithmic growth in β/ν versus the linear growth provided in the upper bound of (4.10) and a logarithmically growing bound is notably tighter than a linearly growing bound.

Theorem 4.4.16 (Generalization Error Bound for G-CG-Net)**.** *Let $\mathcal{S} = \{(\overline{\boldsymbol{y}}_i, \overline{\boldsymbol{c}}_i)\}_{i=1}^{N_s}$ be a training dataset such that Assumption 4.4.1 holds for each $\overline{\boldsymbol{c}}_i$, $\overline{\boldsymbol{y}}_i$ is given by (4.12), and $y_{\max} = \max_{1\leq i\leq N_s} \|\overline{\boldsymbol{y}}_i\|_2$. If Assumption 4.4.2 and Assumption 4.4.3 hold then with probability at least*

$1-\varepsilon$*, for all* $\widehat{\boldsymbol{c}} \in \mathcal{H}_{\text{CG}}^{(1)}$*, the generalization error of G-CG-Net is bounded as*

$$
\begin{aligned}
\mathcal{L}(\widehat{\boldsymbol{c}}) &\leq \mathcal{L}_{\mathcal{S}}(\widehat{\boldsymbol{c}}) + 2\mathcal{R}_{\mathcal{S}}(L \circ \mathcal{H}_{\text{CG}}^{(1)}) + 4c\sqrt{2\ln(4/\varepsilon)/N_s} \\
&\leq \mathcal{L}_{\mathcal{S}}(\widehat{\boldsymbol{c}}) + \frac{8\tau c_{\max}}{\sqrt{N_s}}\sqrt{\dim(\mathcal{P})}\sqrt{\ln\left(1 + \frac{4p_{\max}\kappa(KJD+1)}{c_{\max}}\right)} \\
&\quad + \frac{16\tau}{\sqrt{N_s}}\sqrt{\dim(\mathcal{P})}\sqrt{p_{\max}\kappa(KJD+1)c_{\max}\ln\left(1 + \frac{c_{\max}}{4p_{\max}\kappa(KJD+1)}\right)} \\
&\quad + \frac{8\tau c_{\max}}{\sqrt{N_s}}\sum_{k=1}^{K}\sum_{j=1}^{J}\sum_{d=1}^{D}\sqrt{\alpha_d}\sqrt{\ln\left(1 + \frac{4\omega_d\kappa_{k,d}^{(j)}(KJD+1)}{c_{\max}}\right)} \\
&\quad + \frac{16\tau}{\sqrt{N_s}}\sum_{k=1}^{K}\sum_{j=1}^{J}\sum_{d=1}^{D}\sqrt{\alpha_d}\sqrt{\omega_d\kappa_{k,d}^{(j)}(KJD+1)c_{\max}\ln\left(1 + \frac{c_{\max}}{4\omega_d\kappa_{k,d}^{(j)}(KJD+1)}\right)} \\
&\quad + 4c\sqrt{2\ln(4/\varepsilon)/N_s}
\end{aligned}
$$

for $\dim(\mathcal{P}) = 1, n, 2n-1$, *or* $n(n+1)/2$ *when* $\mathcal{P} = \mathcal{P}_{\text{const}}$, $\mathcal{P} = \mathcal{P}_{\text{diag}}$, $\mathcal{P} = \mathcal{P}_{\text{tri}}$, *or* $\mathcal{P} = \mathcal{P}_{\text{full}}$, *respectively. Additionally,* κ *and* $\kappa_{k,d}^{(j)}$ *are given in equation (4.24) and equation (4.25) from Theorem 4.4.15, respectively.*

Proof. Follow the proof of Theorem 4.4.15 up to **Step (6)**. Next, taking the square root of and then integrating over both sides of the inequality in equation (4.30) then applying Lemma 4.2.30 and the subadditivity of square roots produces, for any $\beta > 0$,

$$
\begin{aligned}
&\int_0^\beta \sqrt{\ln\left(\mathcal{N}\left(\mathcal{M}_{\mathrm{CG}}^{(1)}, \|\cdot\|_F, \epsilon\right)\right)} d\epsilon \\
&\quad \leq \sqrt{\dim(\mathcal{P})} \int_0^\beta \sqrt{\ln\left(1 + \frac{2p_{\max}\sqrt{N_s}\kappa(KJD+1)}{\epsilon}\right)} d\epsilon \\
&\qquad + \sum_{k=1}^{K}\sum_{j=1}^{J}\sum_{d=1}^{D} \sqrt{\alpha_d} \int_0^\beta \sqrt{\ln\left(1 + \frac{2\omega_d\sqrt{N_s}\kappa_{k,d}^{(j)}(KJD+1)}{\epsilon}\right)} d\epsilon \\
&\quad \leq \sqrt{\dim(\mathcal{P})}\beta \sqrt{\ln\left(1 + \frac{2p_{\max}\sqrt{N_s}\kappa(KJD+1)}{\beta}\right)} \\
&\qquad + \sqrt{\dim(\mathcal{P})} \sqrt{2p_{\max}\sqrt{N_s}\kappa(KJD+1)\beta \ln\left(1 + \frac{\beta}{2p_{\max}\sqrt{N_s}\kappa(KJD+1)}\right)} \\
&\qquad + \sum_{k=1}^{K}\sum_{j=1}^{J}\sum_{d=1}^{D} \sqrt{\alpha_d}\beta \sqrt{\ln\left(1 + \frac{2\omega_d\sqrt{N_s}\kappa_{k,d}^{(j)}(KJD+1)}{\beta}\right)} \\
&\qquad + \sum_{k=1}^{K}\sum_{j=1}^{J}\sum_{d=1}^{D} \sqrt{\alpha_d} \sqrt{2\omega_d\sqrt{N_s}\kappa_{k,d}^{(j)}(KJD+1)\beta \ln\left(1 + \frac{\beta}{2\omega_d\sqrt{N_s}\kappa_{k,d}^{(j)}(KJD+1)}\right)}.
\end{aligned}
\tag{4.33}
$$

Combining equation (4.33) and equation (4.28) produces

$$
\begin{aligned}
&\mathcal{R}_{\mathcal{S}}(L \circ \mathcal{H}_{\text{CG}}^{(1)}) \\
&\leq \frac{8\tau}{N_s} \int_0^{\frac{\sqrt{N_s}c_{\max}}{2}} \sqrt{\ln\left(\mathcal{N}(\mathcal{M}_{\text{CG}}^{(1)}, \|\cdot\|_F, \epsilon\right)} d\epsilon \\
&\leq \frac{8\tau}{N_s} \sqrt{\dim(\mathcal{P})} \frac{\sqrt{N_s}c_{\max}}{2} \sqrt{\ln\left(1 + \frac{2p_{\max}\sqrt{N_s}\kappa(KJD+1)}{\frac{\sqrt{N_s}c_{\max}}{2}}\right)} \\
&+ \frac{8\tau}{N_s} \sqrt{\dim(\mathcal{P})} \sqrt{2p_{\max}\sqrt{N_s}\kappa(KJD+1)\frac{\sqrt{N_s}c_{\max}}{2} \ln\left(1 + \frac{\frac{\sqrt{N_s}c_{\max}}{2}}{2p_{\max}\sqrt{N_s}\kappa(KJD+1)}\right)} \\
&+ \frac{8\tau}{N_s} \sum_{k=1}^{K}\sum_{j=1}^{J}\sum_{d=1}^{D} \sqrt{\alpha_d} \frac{\sqrt{N_s}c_{\max}}{2} \sqrt{\ln\left(1 + \frac{2\omega_d\sqrt{N_s}\kappa_{k,d}^{(j)}(KJD+1)}{\frac{\sqrt{N_s}c_{\max}}{2}}\right)} \\
&+ \frac{8\tau}{N_s} \sum_{k=1}^{K}\sum_{j=1}^{J}\sum_{d=1}^{D} \sqrt{\alpha_d} \sqrt{2\omega_d\sqrt{N_s}\kappa_{k,d}^{(j)}(KJD+1)\frac{\sqrt{N_s}c_{\max}}{2} \ln\left(1 + \frac{\frac{\sqrt{N_s}c_{\max}}{2}}{2\omega_d\sqrt{N_s}\kappa_{k,d}^{(j)}(KJD+1)}\right)} \\
&= \frac{4\tau c_{\max}}{\sqrt{N_s}} \sqrt{\dim(\mathcal{P})} \sqrt{\ln\left(1 + \frac{4p_{\max}\kappa(KJD+1)}{c_{\max}}\right)} \\
&+ \frac{8\tau}{\sqrt{N_s}} \sqrt{\dim(\mathcal{P})} \sqrt{p_{\max}\kappa(KJD+1)c_{\max} \ln\left(1 + \frac{c_{\max}}{4p_{\max}\kappa(KJD+1)}\right)} \\
&+ \frac{4\tau c_{\max}}{\sqrt{N_s}} \sum_{k=1}^{K}\sum_{j=1}^{J}\sum_{d=1}^{D} \sqrt{\alpha_d} \sqrt{\ln\left(1 + \frac{4\omega_d\kappa_{k,d}^{(j)}(KJD+1)}{c_{\max}}\right)} \\
&+ \frac{8\tau}{\sqrt{N_s}} \sum_{k=1}^{K}\sum_{j=1}^{J}\sum_{d=1}^{D} \sqrt{\alpha_d} \sqrt{\omega_d\kappa_{k,d}^{(j)}(KJD+1)c_{\max} \ln\left(1 + \frac{c_{\max}}{4\omega_d\kappa_{k,d}^{(j)}(KJD+1)}\right)}. \qquad (4.34)
\end{aligned}
$$

Finally, inserting (4.34) into Theorem 4.2.6, then for $\varepsilon \in (0,1)$ with probability at least $1-\varepsilon$ we have for any $\widehat{\boldsymbol{c}} \in \mathcal{H}_{\text{CG}}^{(1)}$

$$
\begin{aligned}
\mathcal{L}(\widehat{\boldsymbol{c}}) &\leq \mathcal{L}_{\mathcal{S}}(\widehat{\boldsymbol{c}}) + 2\mathcal{R}_{\mathcal{S}}(L \circ \mathcal{H}_{\text{CG}}^{(1)}) + 4c\sqrt{2\ln(4/\varepsilon)/N_s} \\
&\leq \mathcal{L}_{\mathcal{S}}(\widehat{\boldsymbol{c}}) + \frac{8\tau c_{\max}}{\sqrt{N_s}}\sqrt{\dim(\mathcal{P})}\sqrt{\ln\left(1+\frac{4p_{\max}\kappa(KJD+1)}{c_{\max}}\right)} \\
&\quad + \frac{16\tau}{\sqrt{N_s}}\sqrt{\dim(\mathcal{P})}\sqrt{p_{\max}\kappa(KJD+1)c_{\max}\ln\left(1+\frac{c_{\max}}{4p_{\max}\kappa(KJD+1)}\right)} \\
&\quad + \frac{8\tau c_{\max}}{\sqrt{N_s}}\sum_{k=1}^{K}\sum_{j=1}^{J}\sum_{d=1}^{D}\sqrt{\alpha_d}\sqrt{\ln\left(1+\frac{4\omega_d\kappa_{k,d}^{(j)}(KJD+1)}{c_{\max}}\right)} \\
&\quad + \frac{16\tau}{\sqrt{N_s}}\sum_{k=1}^{K}\sum_{j=1}^{J}\sum_{d=1}^{D}\sqrt{\alpha_d}\sqrt{\omega_d\kappa_{k,d}^{(j)}(KJD+1)c_{\max}\ln\left(1+\frac{c_{\max}}{4\omega_d\kappa_{k,d}^{(j)}(KJD+1)}\right)} \\
&\quad + 4c\sqrt{2\ln(4/\varepsilon)/N_s}
\end{aligned}
$$

producing the desired generalization error bound. ⊠

Discussion

While the GEB of Theorem 4.4.16 is a tighter bound than that of Theorem 4.4.15, we show in the large-network-size limit that the GEBs are equal. Note that the G-CG-Net network size is roughly equal to the product KJD as for each iteration K there are J scale variable updates each of which use a subnetwork consisting of D layers.

Proposition 4.4.17. *The generalization error bounds of Theorem 4.4.15 and Theorem 4.4.16 are equal in the limit $KJD \to \infty$.*

Proof. We write GEB_O and GEB_N to denote the right hand side of the generalization error bounds from Theorem 4.4.15 and Theorem 4.4.16, respectively. Observe

$$
\lim_{x\to\infty} x\ln\left(1+\frac{a}{x}\right) = \lim_{x\to\infty}\frac{\ln\left(1+\frac{a}{x}\right)}{x^{-1}} = \lim_{x\to\infty}\frac{\frac{1}{1+\frac{a}{x}}\left(-\frac{a}{x^2}\right)}{-x^{-2}} = a\lim_{x\to\infty}\frac{x}{x+a} = a
$$

where in the third equality we use L'Hôpital's rule. Thus, for Theorem 4.4.16

$$\begin{aligned}
&\lim_{KJD\to\infty} \frac{16\tau}{\sqrt{N_s}}\sqrt{\dim(\mathcal{P})}\sqrt{p_{\max}\kappa(KJD+1)c_{\max}\ln\left(1+\frac{c_{\max}}{4p_{\max}\kappa(KJD+1)}\right)} \\
&\quad= \lim_{x\to\infty} \frac{16\tau}{\sqrt{N_s}}\sqrt{\dim(\mathcal{P})}\sqrt{p_{\max}\kappa c_{\max} x\ln\left(1+\frac{c_{\max}}{4p_{\max}\kappa}\frac{1}{x}\right)} \\
&\quad= \frac{16\tau}{\sqrt{N_s}}\sqrt{\dim(\mathcal{P})}\sqrt{p_{\max}\kappa c_{\max}\frac{c_{\max}}{4p_{\max}\kappa}} \\
&\quad= \frac{8\tau c_{\max}}{\sqrt{N_s}}\sqrt{\dim(\mathcal{P})}.
\end{aligned}$$

Similarly,

$$\begin{aligned}
&\lim_{KDJ\to\infty} \frac{16\tau}{\sqrt{N_s}}\sum_{k=1}^{K}\sum_{j=1}^{J}\sum_{d=1}^{D}\sqrt{\alpha_d}\sqrt{\omega_d\kappa_{k,d}^{(j)}(KJD+1)c_{\max}\ln\left(1+\frac{c_{\max}}{4\omega_d\kappa_{k,d}^{(j)}(KJD+1)}\right)} \\
&\quad= \frac{8\tau c_{\max}}{\sqrt{N_s}}\sum_{k=1}^{K}\sum_{j=1}^{J}\sum_{d=1}^{D}\sqrt{\alpha_d}.
\end{aligned}$$

Therefore

$$\begin{aligned}
&\lim_{KJD\to\infty} \mathrm{GEB}_N \\
&\quad= \mathcal{L}_{\mathcal{S}}(\widehat{\boldsymbol{c}}) + \lim_{KJD\to\infty} \frac{8\tau c_{\max}}{\sqrt{N_s}}\sqrt{\dim(\mathcal{P})}\left(1+\sqrt{\ln\left(1+\frac{4p_{\max}\kappa(KJD+1)}{c_{\max}}\right)}\right) \\
&\qquad+ \lim_{KJD\to\infty} \frac{8\tau c_{\max}}{\sqrt{N_s}}\sum_{k=1}^{K}\sum_{j=1}^{J}\sum_{d=1}^{D}\sqrt{\alpha_d}\left(1+\sqrt{\ln\left(1+\frac{4\omega_d\kappa_{k,d}^{(j)}(KJD+1)}{c_{\max}}\right)}\right) \\
&\qquad+ 4c\sqrt{2\ln(4/\varepsilon)/N_s} \\
&\quad= \mathcal{L}_{\mathcal{S}}(\widehat{\boldsymbol{c}}) + \lim_{KJD\to\infty} \frac{8\tau c_{\max}}{\sqrt{N_s}}\sqrt{\dim(\mathcal{P})}\sqrt{\ln\left(1+\frac{4p_{\max}\kappa(KJD+1)}{c_{\max}}\right)} \\
&\qquad+ \lim_{KJD\to\infty} \frac{8\tau c_{\max}}{\sqrt{N_s}}\sum_{k=1}^{K}\sum_{j=1}^{J}\sum_{d=1}^{D}\sqrt{\alpha_d}\sqrt{\ln\left(1+\frac{4\omega_d\kappa_{k,d}^{(j)}(KJD+1)}{c_{\max}}\right)} \\
&\qquad+ 4c\sqrt{2\ln(4/\varepsilon)/N_s}.
\end{aligned}$$

Similarly,

$$
\begin{aligned}
&\lim_{KJD\to\infty} \text{GEB}_O \\
&\quad = \mathcal{L}_{\mathcal{S}}(\widehat{\boldsymbol{c}}) + \lim_{KJD\to\infty} \frac{8\tau c_{\max}}{\sqrt{N_s}}\sqrt{\dim(\mathcal{P})}\sqrt{\ln\left(e\left(1+\frac{4p_{\max}(KJD+1)\kappa}{c_{\max}}\right)\right)} \\
&\qquad + \lim_{KJD\to\infty} \frac{8\tau c_{\max}}{\sqrt{N_s}}\sum_{k=1}^{K}\sum_{j=1}^{J}\sum_{d=1}^{D}\sqrt{\alpha_d \ln\left(e\left(1+\frac{4\omega_d(KJD+1)\kappa_{k,d}^{(j)}}{c_{\max}}\right)\right)} \\
&\qquad + 4c\sqrt{2\ln(4/\varepsilon)/N_s} \\
&\quad = \mathcal{L}_{\mathcal{S}}(\widehat{\boldsymbol{c}}) + \lim_{KJD\to\infty} \frac{8\tau c_{\max}}{\sqrt{N_s}}\sqrt{\dim(\mathcal{P})}\sqrt{1+\ln\left(1+\frac{4p_{\max}(KJD+1)\kappa}{c_{\max}}\right)} \\
&\qquad + \lim_{KJD\to\infty} \frac{8\tau c_{\max}}{\sqrt{N_s}}\sum_{k=1}^{K}\sum_{j=1}^{J}\sum_{d=1}^{D}\sqrt{\alpha_d}\sqrt{1+\ln\left(1+\frac{4\omega_d(KJD+1)\kappa_{k,d}^{(j)}}{c_{\max}}\right)} \\
&\qquad + 4c\sqrt{2\ln(4/\varepsilon)/N_s} \\
&\quad = \mathcal{L}_{\mathcal{S}}(\widehat{\boldsymbol{c}}) + \lim_{KJD\to\infty} \frac{8\tau c_{\max}}{\sqrt{N_s}}\sqrt{\dim(\mathcal{P})}\sqrt{\ln\left(1+\frac{4p_{\max}(KJD+1)\kappa}{c_{\max}}\right)} \\
&\qquad + \lim_{KJD\to\infty} \frac{8\tau c_{\max}}{\sqrt{N_s}}\sum_{k=1}^{K}\sum_{j=1}^{J}\sum_{d=1}^{D}\sqrt{\alpha_d}\sqrt{\ln\left(1+\frac{4\omega_d(KJD+1)\kappa_{k,d}^{(j)}}{c_{\max}}\right)} \\
&\qquad + 4c\sqrt{2\ln(4/\varepsilon)/N_s}
\end{aligned}
$$

and thus $\lim_{KJD\to\infty} \text{GEB}_O = \lim_{KJD\to\infty} \text{GEB}_N$. ⊠

Since we are concerned about the generalization error as the size of the network grows, Proposition 4.4.17 shows that the improvement on the GEB present in Theorem 4.4.16 is actually negligible and the GEB from Theorem 4.4.15 is an equivalent bound. While this may seem surprising as the GEB from Theorem 4.4.16 was produced by using a logarithmic bound, $\ln(1+\beta/\nu)$, in place of a linear bound, β/ν, the difference in these two bounds is actually negligible since for G-CG-Net the ratio $\beta/\nu = \mathcal{O}((KJD)^{-1}) = \mathcal{O}((\text{Network Size})^{-1})$. Then for large network sizes $\beta/\nu \to 0$ implying $\ln(1+\beta/\nu) \to 0$. In other words, the second term $\nu\int_{\nu/\beta}^{\infty}\frac{1}{u}\frac{1}{1+u}du$ in equation (4.9) is negligible in comparison to the first term $\beta\ln(1+\nu/\beta)$ for the G-CG-Net GEB application.

4.5 Generalization Error Bound for CG-Net

In this section we apply the generalization error bound derived for G-CG-Net to the CG-Net realization, discussed in Section 4.3.3, producing the GEB for CG-Net in Theorem 4.5.1. For CG-Net, the scale variable update parameter spaces, Ω_d from (4.23), are

$$\Omega_d = \begin{cases} \mathcal{P}_{\text{full}} & d = 1 \\ [-\mu, \mu] & d = 2 \end{cases}$$

for a constant $\mu > 0$. Thus, the hypothesis class for CG-Net is

$$\mathcal{H}_{\text{CG}}^{(2)} = \left\{ \widehat{\boldsymbol{c}}\left(\cdot; \left\{P_u, B_k^{(j)}, \mu_k^{(j)}\right\}_{k\in\mathbb{N}[K]}^{j\in\mathbb{N}[J]}\right) : P_u \in \mathcal{P}_{\text{const}}, B_k^{(j)} \in \Omega_1, \mu_k^{(j)} \in \Omega_2 \right\}.$$

Theorem 4.5.1 (Generalization Error Bound for CG-Net)**.** *Let $\mathcal{S} = \{(\overline{\boldsymbol{y}}_i, \overline{\boldsymbol{c}}_i)\}_{i=1}^{N_s}$ be a training dataset where each $(\overline{\boldsymbol{c}}_i, \overline{\boldsymbol{y}}_i)$ is given by (4.12) and define $y_{\max} = \max_{1\leq i\leq N_s} \|\overline{\boldsymbol{y}}_i\|_2$. If Assumption 4.4.1 holds then with probability at least $1-\varepsilon$, for all $\widehat{\boldsymbol{c}} \in \mathcal{H}_{\text{CG}}^{(2)}$, the generalization error of CG-Net is bounded as*

$$\begin{aligned}
\mathcal{L}(\widehat{\boldsymbol{c}}) \leq \mathcal{L}_{\mathcal{S}}(\widehat{\boldsymbol{c}}) &+ \frac{8\tau c_{\max}}{\sqrt{N_s}} \sqrt{\ln\left(e\left(1 + \frac{4p_{\max}(2KJ+1)\kappa}{c_{\max}}\right)\right)} \\
&+ \frac{8\tau c_{\max}}{\sqrt{N_s}} \sum_{k=1}^{K}\sum_{j=1}^{J} \sqrt{\frac{n(n+1)}{2}} \sqrt{\ln\left(e\left(1 + \frac{4p_{\max}(2KJ+1)\kappa_{k,1}^{(j)}}{c_{\max}}\right)\right)} \\
&+ \frac{8\tau c_{\max}}{\sqrt{N_s}} \sum_{k=1}^{K}\sum_{j=1}^{J} \sqrt{\ln\left(e\left(1 + \frac{4\mu(2KJ+1)\kappa_{k,2}^{(j)}}{c_{\max}}\right)\right)} \\
&+ 8\sqrt{2\ln(4/\varepsilon)/N_s}
\end{aligned}$$

for τ a Lipschitz constant of the SSIM loss function and

$$\begin{aligned}
\kappa &= z_\infty(c_1 + p_{\max} y_{\max} \|A\|_\infty)\widehat{c}_1^{(K,J)} + z_\infty c_2 \\
\kappa_{k,d}^{(j)} &= z_\infty(c_1 + p_{\max} y_{\max} \|A\|_\infty)\widehat{c}_{k,j,d,2}^{(K,J)}
\end{aligned}$$

where

$$
\begin{aligned}
c_1 &= p_{\max} y_{\max} \ \|A\|_2 \left(1 + 2z_\infty^2 p_{\max} \ \|A\|_2^2\right) \\
c_2 &= z_\infty y_{\max} \|A\|_2 \left(p_{\max}/p_{\min}\right)^2 \\
\widehat{c}_1^{(K,J)} &= c_2 r_2 \left(\frac{1 - r_1^J}{1 - r_1}\right) \left(\frac{1 - \left(r_1^J + c_1 r_2 \frac{1 - r_1^J}{1 - r_1}\right)^K}{1 - \left(r_1^J + c_1 r_2 \frac{1 - r_1^J}{1 - r_1}\right)}\right) \\
\widehat{c}_{k,j,d,2}^{(K,J)} &= r_1^{J-j} \left(r_1^J + c_1 r_2 \frac{1 - r_1^J}{1 - r_1}\right)^{K-k} \begin{cases} \xi & d = 1 \\ p_{\max} h_{\max} & d = 2 \end{cases}
\end{aligned}
$$

for

$$
\begin{aligned}
r_1 &= 1 + p_{\max} \left(\left(z_\infty p_{\max} y_{\max} \|A\|_2 \|A\|_\infty\right)^2 + \mu \tau_h\right) \\
r_2 &= p_{\max} y_{\max} \|A\|_2 \left(1 + z_\infty^2 p_{\max} \|A\|_2 (\|A\|_2 + \|A\|_\infty)\right) \\
h_{\max} &= \max_{z \in [a,b]} \ [h^{-1}]'(z) h^{-1}(z) \\
\tau_h &= \max_{z \in [a,b]} \ [h^{-1}]''(z) h^{-1}(z) + [h^{-1}]'(z)^2.
\end{aligned}
$$

That is, the GEB for CG-Net is as given in Theorem 4.4.15 where τ a Lipschitz constant of the SSIM loss function, $\dim(\mathcal{P}) = 1$, $c = 2$, $D = 2$,

$$
(\alpha_d, \omega_d) = \begin{cases} (\frac{n(n+1)}{2}, p_{\max}) & d = 1 \\ (1, \mu) & d = 2, \end{cases}
$$

$\widehat{r}_{k,1}^{(J)} = r_1^J, \quad \widehat{r}_{k,2}^{(J)} = r_2 \dfrac{1 - r_1^J}{1 - r_1}$, and

$$
\widehat{r}_{k,d,3}^{(j,J)} = r_1^{J-j} \begin{cases} \xi & d = 1 \\ p_{\max} h_{\max} & d = 2. \end{cases}
$$

As an overview, proving the GEB for CG-Net in Theorem 4.5.1 consists of the following steps: First, we show Assumption 4.4.2 holds for the SSIM loss function used in CG-Net. Second, we

invoke Proposition 4.4.12 to show that the Lipschitz condition of Assumption 4.4.3 holds for each CG-Net scale variable update method. Finally, we apply Theorem 4.4.15 to produce the GEB of Theorem 4.5.1.

Proof of Theorem 4.5.1. As SSIM returns a value in $[-1, 1]$ then the SSIM loss function is bounded by 2. From [49] SSIM is a differentiable function and thus continuous. As the CG-Net outputs are bounded, in the $\|\cdot\|_2$ ball of radius $c_{\max}$, then the gradient of the SSIM loss function is bounded. Hence, by the mean value theorem [53], there exists a Lipschitz constant for the SSIM loss function, on the $\|\cdot\|_2$ ball of radius $c_{\max}$, which we denote by τ. Therefore, Assumption 4.4.2 holds.

Next, for $i \in \{1, 2\}$, let $\boldsymbol{z}_i \in \mathbb{R}^n$ satisfy $\|\boldsymbol{z}_i\|_\infty \leq z_\infty$ and $\boldsymbol{u}_i = \mathcal{T}_{\overline{\boldsymbol{y}}_p}(\boldsymbol{z}_i; P_i)$ for some $P_i \in \mathcal{P}_{\text{const}}$ and $p \in \mathbb{N}[N_s]$. As $\mathcal{P}_{a,b}$ is 1-Lipschitz by Lemma 4.2.28, observe that the scale variable update method of CG-Net, given in equation (4.15), satisfies

$$\begin{aligned}
&\|g_k^{(j)}(\boldsymbol{z}_1, \boldsymbol{u}_1; B_k^{(j)}, \mu_k^{(j)}) - g_k^{(j)}(\boldsymbol{z}_2, \boldsymbol{u}_2; \widetilde{B}_k^{(j)}, \widetilde{\mu}_k^{(j)})\|_2 \\
&\quad = \left\| \mathcal{P}_{a,b}(\boldsymbol{z}_1 - B_k^{(j)} \rho_\xi(\nabla_{\boldsymbol{z}} F(\boldsymbol{u}_1, \boldsymbol{z}_1; \mu_k^{(j)}))) - \mathcal{P}_{a,b}(\boldsymbol{z}_2 - \widetilde{B}_k^{(j)} \rho_\xi(\nabla_{\boldsymbol{z}} F(\boldsymbol{u}_2, \boldsymbol{z}_2; \widetilde{\mu}_k^{(j)}))) \right\|_2 \\
&\quad \leq \left\| \boldsymbol{z}_1 - B_k^{(j)} \rho_\xi(\nabla_{\boldsymbol{z}} F(\boldsymbol{u}_1, \boldsymbol{z}_1; \mu_k^{(j)})) - \boldsymbol{z}_2 - \widetilde{B}_k^{(j)} \rho_\xi(\nabla_{\boldsymbol{z}} F(\boldsymbol{u}_2, \boldsymbol{z}_2; \widetilde{\mu}_k^{(j)})) \right\|_2 \\
&\quad \leq \|\boldsymbol{z}_1 - \boldsymbol{z}_2\|_2 + \left\| \widetilde{B}_k^{(j)} \rho_\xi(\nabla_{\boldsymbol{z}} F(\boldsymbol{u}_2, \boldsymbol{z}_2; \widetilde{\mu}_k^{(j)})) - B_k^{(j)} \rho_\xi(\nabla_{\boldsymbol{z}} F(\boldsymbol{u}_1, \boldsymbol{z}_1; \mu_k^{(j)})) \right\|_2 \qquad (4.35)
\end{aligned}$$

where in the final line we used the triangle inequality.

First, since ρ_ξ is 1-Lipschitz by Lemma 4.2.23 and bounded by ξ by equation (4.6) then using the triangle inequality

$$
\begin{aligned}
&\|\widetilde{B}_k^{(j)}\rho_\xi(\nabla_{\boldsymbol{z}}F(\boldsymbol{u}_2,\boldsymbol{z}_2;\widetilde{\mu}_k^{(j)})) - B_k^{(j)}\rho_\xi(\nabla_{\boldsymbol{z}}F(\boldsymbol{u}_1,\boldsymbol{z}_1;\mu_k^{(j)}))\|_2 \\
&\quad= \Big\|\widetilde{B}_k^{(j)}\rho_\xi(\nabla_{\boldsymbol{z}}F(\boldsymbol{u}_2,\boldsymbol{z}_2;\widetilde{\mu}_k^{(j)})) - \widetilde{B}_k^{(j)}\rho_\xi(\nabla_{\boldsymbol{z}}F(\boldsymbol{u}_1,\boldsymbol{z}_1;\mu_k^{(j)})) \\
&\qquad+\widetilde{B}_k^{(j)}\rho_\xi(\nabla_{\boldsymbol{z}}F(\boldsymbol{u}_1,\boldsymbol{z}_1;\mu_k^{(j)})) - B_k^{(j)}\rho_\xi(\nabla_{\boldsymbol{z}}F(\boldsymbol{u}_1,\boldsymbol{z}_1;\mu_k^{(j)}))\Big\|_2 \\
&\quad\le \Big\|\widetilde{B}_k^{(j)}\rho_\xi(\nabla_{\boldsymbol{z}}F(\boldsymbol{u}_2,\boldsymbol{z}_2;\widetilde{\mu}_k^{(j)})) - \widetilde{B}_k^{(j)}\rho_\xi(\nabla_{\boldsymbol{z}}F(\boldsymbol{u}_1,\boldsymbol{z}_1;\mu_k^{(j)}))\Big\|_2 \\
&\qquad+ \Big\|\widetilde{B}_k^{(j)}\rho_\xi(\nabla_{\boldsymbol{z}}F(\boldsymbol{u}_1,\boldsymbol{z}_1;\mu_k^{(j)})) - B_k^{(j)}\rho_\xi(\nabla_{\boldsymbol{z}}F(\boldsymbol{u}_1,\boldsymbol{z}_1;\mu_k^{(j)}))\Big\|_2 \\
&\quad\le \Big\|\widetilde{B}_k^{(j)}\Big\|_2 \Big\|\rho_\xi(\nabla_{\boldsymbol{z}}F(\boldsymbol{u}_2,\boldsymbol{z}_2;\widetilde{\mu}_k^{(j)})) - \rho_\xi(\nabla_{\boldsymbol{z}}F(\boldsymbol{u}_1,\boldsymbol{z}_1;\mu_k^{(j)}))\Big\|_2 \\
&\qquad+ \|\rho_\xi(\nabla_{\boldsymbol{z}}F(\boldsymbol{u}_1,\boldsymbol{z}_1;\mu_k^{(j)}))\|_2\,\|\widetilde{B}_k^{(j)} - B_k^{(j)}\|_2 \\
&\quad\le p_{\max}\|\rho_\xi(\nabla_{\boldsymbol{z}}F(\boldsymbol{u}_2,\boldsymbol{z}_2;\widetilde{\mu}_k^{(j)})) - \rho_\xi(\nabla_{\boldsymbol{z}}F(\boldsymbol{u}_1,\boldsymbol{z}_1;\mu_k^{(j)}))\|_2 + \xi\|\widetilde{B}_k^{(j)} - B_k^{(j)}\|_2 \\
&\quad\le p_{\max}\|\nabla_{\boldsymbol{z}}F(\boldsymbol{u}_2,\boldsymbol{z}_2;\widetilde{\mu}_k^{(j)}) - \nabla_{\boldsymbol{z}}F(\boldsymbol{u}_1,\boldsymbol{z}_1;\mu_k^{(j)})\|_2 + \xi\|\widetilde{B}_k^{(j)} - B_k^{(j)}\|_2. \qquad (4.36)
\end{aligned}
$$

Second, using equation (4.16) and the triangle inequality for any $j \in \mathbb{N}[N_s]$

$$
\begin{aligned}
&\|\nabla_{\boldsymbol{z}}F(\boldsymbol{u}_2,\boldsymbol{z}_2;\widetilde{\mu}_k^{(j)}) - \nabla_{\boldsymbol{z}}F(\boldsymbol{u}_1,\boldsymbol{z}_1;\mu_k^{(j)})\|_2 \\
&\quad= \Big\|A_{\boldsymbol{u}_2}^T(A_{\boldsymbol{u}_2}\boldsymbol{z}_2 - \overline{\boldsymbol{y}}_j) + \widetilde{\mu}_k^{(j)}[h^{-1}]'(\boldsymbol{z}_2)\odot h^{-1}(\boldsymbol{z}_2) \\
&\qquad-A_{\boldsymbol{u}_1}^T(A_{\boldsymbol{u}_1}\boldsymbol{z}_1 - \overline{\boldsymbol{y}}_j) - \mu_k^{(j)}[h^{-1}]'(\boldsymbol{z}_1)\odot h^{-1}(\boldsymbol{z}_1)\Big\|_2 \\
&\quad\le \big\|A_{\boldsymbol{u}_1}^T(A_{\boldsymbol{u}_1}\boldsymbol{z}_1 - \overline{\boldsymbol{y}}_j) - A_{\boldsymbol{u}_2}^T(A_{\boldsymbol{u}_2}\boldsymbol{z}_2 - \overline{\boldsymbol{y}}_j)\big\|_2 \\
&\qquad+ \Big\|\mu_k^{(j)}[h^{-1}]'(\boldsymbol{z}_1)\odot h^{-1}(\boldsymbol{z}_1) - \widetilde{\mu}_k^{(j)}[h^{-1}]'(\boldsymbol{z}_2)\odot h^{-1}(\boldsymbol{z}_2)\Big\|_2. \qquad (4.37)
\end{aligned}
$$

Third, using the triangle inequality

$$\begin{aligned}
&\left\|\mu_k^{(j)}[h^{-1}]'(\boldsymbol{z}_1)\odot h^{-1}(\boldsymbol{z}_1)-\widetilde{\mu}_k^{(j)}[h^{-1}]'(\boldsymbol{z}_2)\odot h^{-1}(\boldsymbol{z}_2)\right\|_2\\
&\quad=\Big\|\mu_k^{(j)}[h^{-1}]'(\boldsymbol{z}_1)\odot h^{-1}(\boldsymbol{z}_1)-\widetilde{\mu}_k^{(j)}[h^{-1}]'(\boldsymbol{z}_1)\odot h^{-1}(\boldsymbol{z}_1)\\
&\qquad+\widetilde{\mu}_k^{(j)}[h^{-1}]'(\boldsymbol{z}_1)\odot h^{-1}(\boldsymbol{z}_1)-\widetilde{\mu}_k^{(j)}[h^{-1}]'(\boldsymbol{z}_2)\odot h^{-1}(\boldsymbol{z}_2)\Big\|_2\\
&\quad\le\left\|\mu_k^{(j)}[h^{-1}]'(\boldsymbol{z}_1)\odot h^{-1}(\boldsymbol{z}_1)-\widetilde{\mu}_k^{(j)}[h^{-1}]'(\boldsymbol{z}_1)\odot h^{-1}(\boldsymbol{z}_1)\right\|_2\\
&\qquad+\left\|\widetilde{\mu}_k^{(j)}[h^{-1}]'(\boldsymbol{z}_1)\odot h^{-1}(\boldsymbol{z}_1)-\widetilde{\mu}_k^{(j)}[h^{-1}]'(\boldsymbol{z}_2)\odot h^{-1}(\boldsymbol{z}_2)\right\|_2\\
&\quad\le\left\|[h^{-1}]'(\boldsymbol{z}_1)\odot h^{-1}(\boldsymbol{z}_1)\right\|_2\,|\mu_k^{(j)}-\widetilde{\mu}_k^{(j)}|\\
&\qquad+|\widetilde{\mu}_k^{(j)}|\,\left\|[h^{-1}]'(\boldsymbol{z}_1)\odot h^{-1}(\boldsymbol{z}_1)-[h^{-1}]'(\boldsymbol{z}_2)\odot h^{-1}(\boldsymbol{z}_2)\right\|_2\\
&\quad\le h_{\max}\,|\mu_k^{(j)}-\widetilde{\mu}_k^{(j)}|+\mu\tau_h\,\|\boldsymbol{z}_1-\boldsymbol{z}_2\|_2
\end{aligned}\tag{4.38}$$

where in the final inequality we used that τ_h is an upper bound on $\nabla_z[h^{-1}]'(z)h^{-1}(z)$ implying τ_h is a Lipschitz constant of $[h^{-1}]'(z)h^{-1}(z)$ and thus, by Lemma 4.2.25, τ_h is a Lipschitz constant of $[h^{-1}]'(\boldsymbol{z})\odot h^{-1}(\boldsymbol{z})$.

Fourth, assume that $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$ are both Tikhonov solutions. That is,

$$\boldsymbol{u}_i=\mathcal{T}_{\overline{\boldsymbol{y}}_j}(\boldsymbol{z}_i;P_i)$$

for a measurement $\overline{\boldsymbol{y}}_j$, some $\boldsymbol{z}_i\in[0,\infty)^n$ satisfying $\|\boldsymbol{z}_i\|_\infty\le z_\infty$, and some covariance matrix $P_i\in\mathcal{P}_{\text{const}}$ where $i=1,2$ and $j\in\mathbb{N}[N_s]$. By Proposition 4.4.12 for $i=1,2$ and a measurement $\overline{\boldsymbol{y}}_j$ where $j\in\mathbb{N}[N_s]$, it holds that

$$\|\boldsymbol{u}_i\|_p\le z_\infty p_{\max}\|A\|_p\|\overline{\boldsymbol{y}}_j\|_p.$$

Hence, using the triangle inequality

$$
\begin{aligned}
&\left\|A_{\boldsymbol{u}_1}^T A_{\boldsymbol{u}_1}\boldsymbol{z}_1 - A_{\boldsymbol{u}_2}^T A_{\boldsymbol{u}_2}\boldsymbol{z}_2\right\|_2 \\
&= \left\|A_{\boldsymbol{u}_1}^T A_{\boldsymbol{u}_1}\boldsymbol{z}_1 - A_{\boldsymbol{u}_1}^T A_{\boldsymbol{u}_2}\boldsymbol{z}_1 + A_{\boldsymbol{u}_1}^T A_{\boldsymbol{u}_2}\boldsymbol{z}_1 - A_{\boldsymbol{u}_2}^T A_{\boldsymbol{u}_2}\boldsymbol{z}_1 + A_{\boldsymbol{u}_2}^T A_{\boldsymbol{u}_2}\boldsymbol{z}_1 - A_{\boldsymbol{u}_2}^T A_{\boldsymbol{u}_2}\boldsymbol{z}_2\right\|_2 \\
&\leq \left\|A_{\boldsymbol{u}_1}^T A_{\boldsymbol{u}_1}\boldsymbol{z}_1 - A_{\boldsymbol{u}_1}^T A_{\boldsymbol{u}_2}\boldsymbol{z}_1\right\|_2 + \left\|A_{\boldsymbol{u}_1}^T A_{\boldsymbol{u}_2}\boldsymbol{z}_1 - A_{\boldsymbol{u}_2}^T A_{\boldsymbol{u}_2}\boldsymbol{z}_1\right\|_2 + \left\|A_{\boldsymbol{u}_2}^T A_{\boldsymbol{u}_2}\boldsymbol{z}_1 - A_{\boldsymbol{u}_2}^T A_{\boldsymbol{u}_2}\boldsymbol{z}_2\right\|_2 \\
&\leq \left\|A_{\boldsymbol{u}_1}^T A_{\boldsymbol{z}_1}\boldsymbol{u}_1 - A_{\boldsymbol{u}_1}^T A_{\boldsymbol{z}_1}\boldsymbol{u}_2\right\|_2 + \left\|\mathrm{Diag}(\boldsymbol{u}_1 - \boldsymbol{u}_2)\left(A^T A_{\boldsymbol{z}_1}\boldsymbol{u}_2\right)\right\|_2 + \left\|A_{\boldsymbol{u}_2}^T A_{\boldsymbol{u}_2}\left(\boldsymbol{z}_1 - \boldsymbol{z}_2\right)\right\|_2 \\
&\leq \left\|A_{\boldsymbol{u}_1}^T A_{\boldsymbol{z}_1}\right\|_2 \|\boldsymbol{u}_1 - \boldsymbol{u}_2\|_2 + \left\|A^T A_{\boldsymbol{z}_1}\boldsymbol{u}_2\right\|_2 \|\boldsymbol{u}_1 - \boldsymbol{u}_2\|_\infty + \left\|A_{\boldsymbol{u}_2}^T A_{\boldsymbol{u}_2}\right\|_2 \|\boldsymbol{z}_1 - \boldsymbol{z}_2\|_2 \\
&\leq \|\boldsymbol{u}_1\|_\infty \|\boldsymbol{z}_1\|_\infty \|A\|_2^2 \|\boldsymbol{u}_1 - \boldsymbol{u}_2\|_2 + \|A\|_2^2 \|\boldsymbol{z}_1\|_\infty \|\boldsymbol{u}_2\|_2 \|\boldsymbol{u}_1 - \boldsymbol{u}_2\|_2 + \|A\|_2^2 \|\boldsymbol{u}_2\|_\infty^2 \|\boldsymbol{z}_1 - \boldsymbol{z}_2\|_2 \\
&\leq z_\infty^2 p_{\max}\|A\|_\infty \|\overline{\boldsymbol{y}}_j\|_\infty \|A\|_2^2 \|\boldsymbol{u}_1 - \boldsymbol{u}_2\|_2 + z_\infty^2 p_{\max}\|A\|_2 \|\overline{\boldsymbol{y}}_j\|_2 \|A\|_2^2 \|\boldsymbol{u}_2\|_2 \|\boldsymbol{u}_1 - \boldsymbol{u}_2\|_2 \\
&\quad + \left(z_\infty p_{\max}\|A\|_\infty \|\overline{\boldsymbol{y}}_j\|_\infty\right)^2 \|A\|_2^2 \|\boldsymbol{z}_1 - \boldsymbol{z}_2\|_2 \\
&\leq z_\infty^2 p_{\max}\|A\|_2^2 \|\overline{\boldsymbol{y}}_j\|_2 (\|A\|_2 + \|A\|_\infty) \|\boldsymbol{u}_1 - \boldsymbol{u}_2\|_2 + z_\infty^2 p_{\max}^2 \|A\|_2^2 \|A\|_\infty^2 \|\overline{\boldsymbol{y}}_j\|_2^2 \|\boldsymbol{z}_1 - \boldsymbol{z}_2\|_2.
\end{aligned}
$$

Thus, for any $j \in \mathbb{N}[N_s]$

$$
\begin{aligned}
&\|A_{\boldsymbol{u}_1}^T(A_{\boldsymbol{u}_1}\boldsymbol{z}_1 - \overline{\boldsymbol{y}}_j) - A_{\boldsymbol{u}_2}^T(A_{\boldsymbol{u}_2}\boldsymbol{z}_2 - \overline{\boldsymbol{y}}_j)\|_2 \\
&\qquad \leq \|A_{\boldsymbol{u}_1}^T A_{\boldsymbol{u}_1}\boldsymbol{z}_1 - A_{\boldsymbol{u}_2}^T A_{\boldsymbol{u}_2}\boldsymbol{z}_2\|_2 + \|(A_{\boldsymbol{u}_1}^T - A_{\boldsymbol{u}_2}^T)\overline{\boldsymbol{y}}_j\|_2 \\
&\qquad \leq \left(z_\infty p_{\max} y_{\max}\|A\|_2\|A\|_\infty\right)^2 \|\boldsymbol{z}_1 - \boldsymbol{z}_2\|_2 \\
&\qquad\quad + z_\infty^2 p_{\max}\|A\|_2^2\, y_{\max}(\|A\|_2 + \|A\|_\infty) \|\boldsymbol{u}_1 - \boldsymbol{u}_2\|_2 + \|A\|_2\|\overline{\boldsymbol{y}}_j\|_2 \|\boldsymbol{u}_1 - \boldsymbol{u}_2\|_\infty \\
&\qquad \leq \left(z_\infty p_{\max} y_{\max}\|A\|_2\|A\|_\infty\right)^2 \|\boldsymbol{z}_1 - \boldsymbol{z}_2\|_2 \\
&\qquad\quad + \left(z_\infty^2 p_{\max}\|A\|_2^2\, y_{\max}(\|A\|_2 + \|A\|_\infty) + \|A\|_2 y_{\max}\right) \|\boldsymbol{u}_1 - \boldsymbol{u}_2\|_2 \\
&\qquad \leq \left(z_\infty p_{\max} y_{\max}\|A\|_2\|A\|_\infty\right)^2 \|\boldsymbol{z}_1 - \boldsymbol{z}_2\|_2 \\
&\qquad\quad + y_{\max}\|A\|_2 \left(1 + z_\infty^2 p_{\max}\|A\|_2(\|A\|_2 + \|A\|_\infty)\right) \|\boldsymbol{u}_1 - \boldsymbol{u}_2\|_2
\end{aligned} \tag{4.39}
$$

Next, combining equation (4.39), equation (4.38), equation (4.37), equation (4.36), and equation (4.35)

$$
\begin{aligned}
&\|g_k^{(j)}(\boldsymbol{z}_1, \boldsymbol{u}_1; B_k^{(j)}, \mu_k^{(j)}) - g_k^{(j)}(\boldsymbol{z}_2, \boldsymbol{u}_2; \widetilde{B}_k^{(j)}, \widetilde{\mu}_k^{(j)})\|_2 \\
&\quad \leq \|\boldsymbol{z}_1 - \boldsymbol{z}_2\|_2 + \left\| \widetilde{B}_k^{(j)} \rho_\xi(\nabla_{\boldsymbol{z}} F(\boldsymbol{u}_2, \boldsymbol{z}_2; \widetilde{\mu}_k^{(j)})) - B_k^{(j)} \rho_\xi(\nabla_{\boldsymbol{z}} F(\boldsymbol{u}_1, \boldsymbol{z}_1; \mu_k^{(j)})) \right\|_2 \\
&\quad \leq \|\boldsymbol{z}_1 - \boldsymbol{z}_2\|_2 + p_{\max} \|\nabla_{\boldsymbol{z}} F(\boldsymbol{u}_2, \boldsymbol{z}_2; \widetilde{\mu}_k^{(j)}) - \nabla_{\boldsymbol{z}} F(\boldsymbol{u}_1, \boldsymbol{z}_1; \mu_k^{(j)})\|_2 + \xi \|\widetilde{B}_k^{(j)} - B_k^{(j)}\|_2 \\
&\quad \leq \|\boldsymbol{z}_1 - \boldsymbol{z}_2\|_2 + \xi \|\widetilde{B}_k^{(j)} - B_k^{(j)}\|_2 + p_{\max} \left\| A_{\boldsymbol{u}_1}^T (A_{\boldsymbol{u}_1} \boldsymbol{z}_1 - \overline{\boldsymbol{y}}_j) - A_{\boldsymbol{u}_2}^T (A_{\boldsymbol{u}_2} \boldsymbol{z}_2 - \overline{\boldsymbol{y}}_j) \right\|_2 \\
&\qquad + p_{\max} \left\| \mu_k^{(j)} [h^{-1}]'(\boldsymbol{z}_1) \odot h^{-1}(\boldsymbol{z}_1) - \widetilde{\mu}_k^{(j)} [h^{-1}]'(\boldsymbol{z}_2) \odot h^{-1}(\boldsymbol{z}_2) \right\|_2 \\
&\quad \leq \|\boldsymbol{z}_1 - \boldsymbol{z}_2\|_2 + \xi \|\widetilde{B}_k^{(j)} - B_k^{(j)}\|_2 + p_{\max} \left\| A_{\boldsymbol{u}_1}^T (A_{\boldsymbol{u}_1} \boldsymbol{z}_1 - \overline{\boldsymbol{y}}_j) - A_{\boldsymbol{u}_2}^T (A_{\boldsymbol{u}_2} \boldsymbol{z}_2 - \overline{\boldsymbol{y}}_j) \right\|_2 \\
&\qquad + p_{\max} h_{\max} |\mu_k^{(j)} - \widetilde{\mu}_k^{(j)}| + p_{\max} \mu \tau_h \|\boldsymbol{z}_1 - \boldsymbol{z}_2\|_2 \\
&\quad \leq \|\boldsymbol{z}_1 - \boldsymbol{z}_2\|_2 + \xi \|\widetilde{B}_k^{(j)} - B_k^{(j)}\|_2 + p_{\max} h_{\max} |\mu_k^{(j)} - \widetilde{\mu}_k^{(j)}| + p_{\max} \mu \tau_h \|\boldsymbol{z}_1 - \boldsymbol{z}_2\|_2 \\
&\qquad + p_{\max} \left(z_\infty p_{\max} y_{\max} \|A\|_2 \|A\|_\infty \right)^2 \|\boldsymbol{z}_1 - \boldsymbol{z}_2\|_2 \\
&\qquad + p_{\max} y_{\max} \|A\|_2 \left(1 + z_\infty^2 p_{\max} \|A\|_2 (\|A\|_2 + \|A\|_\infty) \right) \|\boldsymbol{u}_1 - \boldsymbol{u}_2\|_2 \\
&\quad = \left(1 + p_{\max} \left(\left(z_\infty p_{\max} y_{\max} \|A\|_2 \|A\|_\infty \right)^2 + \mu \tau_h \right) \right) \|\boldsymbol{z}_1 - \boldsymbol{z}_2\|_2 \\
&\qquad + p_{\max} y_{\max} \|A\|_2 \left(1 + z_\infty^2 p_{\max} \|A\|_2 (\|A\|_2 + \|A\|_\infty) \right) \|\boldsymbol{u}_1 - \boldsymbol{u}_2\|_2 \\
&\qquad + \xi \|\widetilde{B}_k^{(j)} - B_k^{(j)}\|_2 + p_{\max} h_{\max} |\mu_k^{(j)} - \widetilde{\mu}_k^{(j)}|. \qquad (4.40)
\end{aligned}
$$

Therefore, by equation (4.40), Assumption 4.4.3 holds specifically with $r_{k,1}^{(j-1)} = r_1, r_{k,2}^{(j-1)} = r_2$ and

$$
(\boldsymbol{\theta}_{k,d}^{(j)}, r_{k,d,3}^{(j-1)}, \|\cdot\|_{(d)}) = \begin{cases} (B_k^{(j)}, \xi, \|\cdot\|_2) & d = 1 \\ (\mu_k^{(j)}, p_{\max} h_{\max}, |\cdot|) & d = 2. \end{cases}
$$

Hence, the constants $\widehat{r}_{k,1}^{(J)}, \widehat{r}_{k,2}^{(J)}$, and $\widehat{r}_{k,d,3}^{(j,J)}$ from Proposition 4.4.6 are given as

$$\begin{aligned}
\widehat{r}_{k,1}^{(J)} &= \prod_{j=1}^{J} r_{k,1}^{(j-1)} = r_1^J \\
\widehat{r}_{k,2}^{(J)} &= \sum_{j=1}^{J} r_{k,2}^{(j-1)} \prod_{\ell=j}^{J-1} r_{k,1}^{(\ell)} = r_2 \sum_{j=1}^{J} r_1^{J-j} = r_2 \frac{1-r_1^J}{1-r_1},
\end{aligned}$$

and

$$\widehat{r}_{k,d,3}^{(j,J)} = r_{k,d,3}^{(j-1)} \prod_{\ell=j}^{J-1} r_{k,1}^{(\ell)} = r_{k,d,3}^{(j-1)} r_1^{J-j} = r_1^{J-j} \begin{cases} \xi & d=1 \\ p_{\max} h_{\max} & d=2. \end{cases}$$

Thus, the constants $\widehat{c}_1^{(K,J)}$ and $\widehat{c}_{k,j,d,2}^{(K,J)}$ from Theorem 4.4.15 are given as

$$\begin{aligned}
\widehat{c}_1^{(K,J)} &= c_2 \sum_{k=1}^{K} \widehat{r}_{k,2}^{(J)} \prod_{\ell=k+1}^{K} (\widehat{r}_{\ell,1}^{(J)} + \widehat{r}_{\ell,2}^{(J)} c_1) \\
&= c_2 \sum_{k=1}^{K} r_2 \frac{1-r_1^J}{1-r_1} \prod_{\ell=k+1}^{K} \left(r_1^J + r_2 \frac{1-r_1^J}{1-r_1} c_1 \right) \\
&= c_2 r_2 \frac{1-r_1^J}{1-r_1} \sum_{k=1}^{K} \left(r_1^J + r_2 \frac{1-r_1^J}{1-r_1} c_1 \right)^{K-k} \\
&= c_2 r_2 \frac{1-r_1^J}{1-r_1} \sum_{k=0}^{K-1} \left(r_1^J + r_2 \frac{1-r_1^J}{1-r_1} c_1 \right)^{k} \\
&= c_2 r_2 \left(\frac{1-r_1^J}{1-r_1} \right) \left(\frac{1 - \left(r_1^J + r_2 \frac{1-r_1^J}{1-r_1} c_1 \right)^K}{1 - \left(r_1^J + r_2 \frac{1-r_1^J}{1-r_1} c_1 \right)} \right)
\end{aligned}$$

and

$$\begin{aligned}
\widehat{c}_{k,j,d,2}^{(K,J)} = \widehat{r}_{k,d,3}^{(j,J)} \prod_{\ell=k+1}^{K} (\widehat{r}_{\ell,1}^{(J)} + \widehat{r}_{\ell,2}^{(J)} c_1) &= \widehat{r}_{k,d,3}^{(j,J)} \prod_{\ell=k+1}^{K} \left(r_1^J + r_2 \frac{1-r_1^J}{1-r_1} c_1 \right) \\
&= \widehat{r}_{k,d,3}^{(j,J)} \left(r_1^J + r_2 \frac{1-r_1^J}{1-r_1} c_1 \right)^{K-k} \\
&= r_1^{J-j} \left(r_1^J + r_2 \frac{1-r_1^J}{1-r_1} c_1 \right)^{K-k} \begin{cases} \xi & d=1 \\ p_{\max} h_{\max} & d=2 \end{cases}
\end{aligned}$$

for

$$c_1 = p_{\max} y_{\max} \, \|A\|_2 \left(1 + 2z_\infty^2 p_{\max} \, \|A\|_2^2\right)$$
$$c_2 = z_\infty y_{\max} \|A\|_2 \left(p_{\max}/p_{\min}\right)^2 .$$

Noting that each steepest descent matrix $B_k^{(j)} \in \mathcal{P}_{\text{full}}$, each step size $\mu_k^{(j)} \in [-\mu, \mu]$, and any covariance matrix $P_u \in \mathcal{P}_{\text{full}}$ then

$$(\alpha_d, \omega_d) = \begin{cases} (\frac{n(n+1)}{2}, p_{\max}) & d = 1 \\ (1, \mu) & d = 2. \end{cases}$$

and $\sqrt{\dim(\mathcal{P}_{\text{const}})} = 1$. Therefore, applying Theorem 4.4.15 produces the desired generalization error bound for CG-Net. ⊠

We remark that in Theorem 4.5.1 we only state a Lipschitz constant for the SSIM loss function exists as deriving one is not illuminating. This is, in part, due to the fact that the function SSIM(I_1, I_2), from the SSIM loss function, is implemented as the average structural similarity over a set of patches from the input images where in each patch a Gaussian weighting filter is used [49].

For examples of the parameters in the GEB of Theorem 4.5.1, all numerical experiments in [57, 76] and Chapter 2 use $h(z) = \exp(z)$ on $[a, b] = [1, \exp(3)]$ and $\xi = 1$. Thus $h_{\max} = \exp(-1)$, $\tau_h = 1$, and $z_\infty = \exp(3)$. Furthermore, preprocessing is used such that $c_{\max} = 1$ and for $\epsilon > 0$, a small stabilizing parameter, $p_{\max} = 1/\epsilon$, $p_{\min} = \epsilon$, and $|\mu| \leq 1/\epsilon$. The remaining constants $\|A\|_2$, $\|A\|_\infty$, and $y_{\max}$ can be calculated given the measurement model and training dataset.

4.5.1 Asymptotic Forms

In this section, we provide asymptotic forms for the CG-Net generalization error bound given in Theorem 4.5.1. We consider asymptotic bounds as the size of the network – i.e. K and J, the number of unrolled iterations of CG-LS – grows, as the dimension of the measurements and

reconstructed signal – i.e. m and n, respectively – grow, and as the number of training data samples – i.e. N_s – grows.

Recall the values $\tau, c_{\max}, p_{\max}, p_{\min}, \mu, z_\infty, y_{\max}, \|A\|_2, \|A\|_\infty, \xi, h_{\max}$, and τ_h that are utilized by Theorem 4.5.1. Note that τ is a Lipschitz constant on the SSIM loss function and is thus independent of network size, measurement dimension, reconstructed signal dimension, and training dataset size. Similarly, $h_{\max}$ and τ_h are constant bounds determined solely by the choice of inverse nonlinearity h and are independent of network size,measurement dimension, reconstructed signal dimension, and training dataset size. As $p_{\max}, p_{\min}, \mu, z_\infty$, and ξ are all parameters chosen and fixed during the constructing of CG-Net before training, we assume that each is independent of network size, measurement dimension, reconstructed signal dimension, and training dataset size. Thus, we establish asymptotic forms for the CG-Net generalization error the rely on whether $c_{\max}, y_{\max}, \|A\|_2$, and $\|A\|_\infty$ depend on the measurement and reconstructed signal dimension.

Finally, let $a \lesssim b$ denote that $a \leq s_c b$ for some constant $s_c > 0$. Then $a \lesssim b$ implies $a = \mathcal{O}(b)$.

Dependent on Signal Dimension

For the following asymptotic generalization error bound of CG-Net, we assume that each of $c_{\max}, y_{\max}, \|A\|_2$, and $\|A\|_\infty$ depend on the measurement and reconstructed signal dimension in that $\lim_{m,n\to\infty} c_{\max} \to \infty$, $\lim_{m,n\to\infty} y_{\max} \to \infty$, $\lim_{m,n\to\infty} \|A\|_p \to \infty$ for any $1 \leq p \leq \infty$. Note for any $\boldsymbol{x} \in \mathbb{R}^k$ that $\|\boldsymbol{x}\|_2 \leq \sqrt{k}\|\boldsymbol{x}\|_\infty$. As $c_{\max}$ and $y_{\max}$ are, respectively, the maximum Euclidean norm bounds on the signals of interest, of dimension n, and measurements, of dimension m, then $c_{\max} \lesssim \sqrt{n}$ and $y_{\max} \lesssim \sqrt{m}$; where we have reasonably assumed that the maximum entry in any measurement or reconstructed signal is independent of the signal dimension. Furthermore, as $\|A\|_2 \leq \sqrt{m}\|A\|_\infty$ and $\|A\|_\infty \leq n\,(\max_{i,j} |A_{i,j}|)$, since $\|A\|_\infty$ is the max absolute row sum, then $\|A\|_2 \lesssim n\sqrt{m}$ and $\|A\|_\infty \leq n$; where again we have reasonably assumed that the maximum entry of the measurement matrix is independent of the matrix dimensions.

Corollary 4.5.2. *Let $c_{\max}, y_{\max}, \|A\|_2$, and $\|A\|_\infty$ depend on the measurement and reconstructed signal dimension. The generalization error for CG-Net scales as*

$$|\mathcal{L}(\widehat{\boldsymbol{c}}) - \mathcal{L}_{\mathcal{S}}(\widehat{\boldsymbol{c}})| \lesssim c_{\max}\sqrt{\frac{n^2(KJ)^3(\ln(y_{\max}) + \ln(\|A\|_2) + \ln(\|A\|_\infty))}{N_s}}.$$

Furthermore, as, at most, $c_{\max} = \mathcal{O}(\sqrt{n}), y_{\max} = \mathcal{O}(\sqrt{m}), \|A\|_2 = \mathcal{O}(n\sqrt{m})$, and $\|A\|_\infty = \mathcal{O}(n)$ then generalization error for CG-Net scales at most as

$$|\mathcal{L}(\widehat{\boldsymbol{c}}) - \mathcal{L}_{\mathcal{S}}(\widehat{\boldsymbol{c}})| \lesssim \sqrt{\frac{n^3(\ln(m) + \ln(n))(KJ)^3}{N_s}}.$$

A proof of Corollary 4.5.2 is provided in Appendix B.2 and results from ignoring constants to consider how the GEB from Theorem 4.5.1 scales in network size and signal dimension. From Corollary 4.5.2, once the amount of training data satisfies

$$N_s \sim c_{\max}^2 n^2 (KJ)^3 (\ln(y_{\max}) + \ln(\|A\|_2) + \ln(\|A\|_\infty))$$

or

$$N_s \sim n^3(\ln(m) + \ln(n))(KJ)^3$$

then the GEB of CG-Net is guaranteed to be small with high probability. Furthermore, Corollary 4.5.2 shows that, at worst, the CG-Net generalization error grows $\mathcal{O}(\sqrt{n^3 \ln(n)})$ in reconstructed signal dimension, $\mathcal{O}(\sqrt{(KJ)^3}) = \mathcal{O}((\text{Network Size})^{3/2})$ in network size, and $\mathcal{O}(\sqrt{\ln(m)})$ in measurement dimension, and $\mathcal{O}(N_s^{-1/2})$ in training dataset size.

Independent Measurement Matrix Norm

Now we assume that $\|A\|_2$ and $\|A\|_\infty$ are independent of m and n defining the dimension of A. For instance, the measurement matrix could be normalized through preprocessing such that $\|A\|_2 = 1$.

Corollary 4.5.3. *Let $c_{\max}$ and $y_{\max}$ depend on the measurement and reconstructed signal dimension. Furthermore, assume $\|A\|_2$ and $\|A\|_\infty$ are independent of the measurement and reconstructed signal dimension. The generalization error for CG-Net scales as*

$$|\mathcal{L}(\widehat{\boldsymbol{c}}) - \mathcal{L}_{\mathcal{S}}(\widehat{\boldsymbol{c}})| \lesssim c_{\max}\sqrt{\frac{n^2(KJ)^3\ln(y_{\max})}{N_s}}.$$

Furthermore, as, at most, $c_{\max} = \mathcal{O}(\sqrt{n})$ and $y_{\max} = \mathcal{O}(\sqrt{m})$ then generalization error for CG-Net scales at most as

$$|\mathcal{L}(\widehat{\boldsymbol{c}}) - \mathcal{L}_{\mathcal{S}}(\widehat{\boldsymbol{c}})| \lesssim \sqrt{\frac{n^3(KJ)^3\ln(m)}{N_s}}.$$

Corollary 4.5.3 is a direct consequence of Corollary 4.5.2, as shown in Appendix B.3, where the terms involving $\|A\|_2$ and $\|A\|_\infty$ have been removed as they are of lower order, in m and n, than the other terms, i.e. $\ln(y_{\max})$, which are being summed. From Corollary 4.5.3, once the amount of training data satisfies

$$N_s \sim c_{\max}^2 n^2 (KJ)^3 \ln(y_{\max})$$

or

$$N_s \sim n^3 \ln(m)(KJ)^3$$

then the GEB of CG-Net is guaranteed to be small with high probability. Furthermore, Corollary 4.5.3 shows that, at worst, the CG-Net generalization error grows $\mathcal{O}(n^{3/2})$ in reconstructed signal dimension, $\mathcal{O}(\sqrt{(KJ)^3}) = \mathcal{O}((\text{Network Size})^{3/2})$ in network size, $\mathcal{O}(\sqrt{\ln(m)})$ in measurement dimension, and $\mathcal{O}(N_s^{-1/2})$ in training dataset size.

Independence of Signal Dimension

Here, we assume that $c_{\max}$, $y_{\max}$, $\|A\|_2$, and $\|A\|_\infty$ are independent of m and n defining the measurement and reconstructed signal dimension. As an example, CG-Net and DR-CG-Net from Chapter 2 and Chapter 3 implement preprocessing to scale all images of interest to be in the unit ball. That is, $c_{\max} \leq 1$. Additionally, if the measurement matrix is normalized through preprocessing such that $\|A\|_2 = 1$ then $y_{\max} \leq \|A\|_2 c_{\max} \leq 1$.

Corollary 4.5.4. *Let $c_{\max}$, $y_{\max}$, $\|A\|_2$, and $\|A\|_\infty$ be independent of the measurement and reconstructed signal dimension. The generalization error for CG-Net scales, at most, as*

$$|\mathcal{L}(\widehat{\boldsymbol{c}}) - \mathcal{L}_{\mathcal{S}}(\widehat{\boldsymbol{c}})| \lesssim \sqrt{\frac{n^2(KJ)^3}{N_s}}.$$

Corollary 4.5.4 is a direct consequence of Corollary 4.5.2, as shown in Appendix B.4, where the terms involving $c_{\max}$, $y_{\max}$, $\|A\|_2$ and $\|A\|_\infty$ have been removed as they are now of constant order, i.e. do not scale with m and n. From Corollary 4.5.4, once the amount of training data satisfies

$$N_s \sim n^2(KJ)^3$$

then the GEB of CG-Net is guaranteed to be small with high probability. Furthermore, Corollary 4.5.4 shows that, at worst, the CG-Net generalization error grows $\mathcal{O}(n)$ in reconstructed signal dimension, $\mathcal{O}(\sqrt{(KJ)^3}) = \mathcal{O}((\text{Network Size})^{3/2})$ in network size, and $\mathcal{O}(N_s^{-1/2})$ in training dataset size.

4.6 Generalization Error Bound for DR-CG-Net without Bias

In this section, we apply the generalization error bound derived for G-CG-Net to the DR-CG-Net realizations discussed in Section 4.3.3. Recall that the two formulations of DR-CG-Net considered implement either a projected gradient descent (PGD) or an iterative shrinkage and thresh-

olding (ISTA) scale-variable-update method as given in equation (4.17). We denote the implementations of DR-CG-Net using a PGD or ISTA scale-variable-update method as PGD DR-CG-Net and ISTA DR-CG-Net, respectively.

To prove a GEB for DR-CG-Net we first require a bounded and Lipschitz property for fully-connected networks.

4.6.1 Properties of Fully-Connected Neural Networks

In this section, we provide a series of informative properties of fully-connected neural networks; all of which likely exist in the literature. Let $\mathcal{G}_t^{(i)} : \mathbb{R}^{d_t} \to \mathbb{R}^{d_{t+1}}$ denote a single fully-connected layer that is defined by

$$\mathcal{G}_t^{(i)}(\boldsymbol{x}) \coloneqq W_t^{(i)}\boldsymbol{x} \tag{4.41}$$

for a weight matrix $W_t^{(i)} \in \mathbb{R}^{d_{t+1}\times d_t}$. Next, for componentwise activation function σ, define a fully-connected network $\mathcal{G}^{(i,T)} : \mathbb{R}^{d_1} \to \mathbb{R}^{d_{T+1}}$ as

$$\mathcal{G}^{(i,T)}(\boldsymbol{x}) = \mathcal{G}_T^{(i)} \circ \sigma \circ \mathcal{G}_{T-1}^{(i)} \circ \cdots \circ \sigma \circ \mathcal{G}_1^{(i)}(\boldsymbol{x}). \tag{4.42}$$

That is, the fully-connected network is a composition of fully-connected layers where the output of each layer is provided to an activation function.

First, we record a bound on the output from a fully connected network.

Lemma 4.6.1. *Let σ be a componentwise activation function satisfying $\|\sigma(\boldsymbol{x})\|_2 \leq \|\boldsymbol{x}\|_2$ and $\mathcal{G}^{(i,T)}$ be given as in (4.42). If $\|W_t^{(i)}\|_2 \leq \varpi_t$ for all $t \in \mathbb{N}[T]$, then*

$$\|\sigma(\mathcal{G}^{(i,T)}(\boldsymbol{x}))\|_2 \leq \prod_{t=1}^{T} \varpi_t \|\boldsymbol{x}\|_2.$$

Proof. We use induction on T. The base case $T = 1$ holds trivially. Assume the induction hypothesis holds for fixed T where $T > 1$. Observe

$$\begin{aligned}\|\sigma(\mathcal{G}^{(i,T+1)}(\boldsymbol{x}))\|_2 &= \|\sigma(W_{T+1}^{(i)}\sigma(\mathcal{G}^{(i,T)}(\boldsymbol{x})))\|_2 \\ &\leq \varpi_{T+1}\|\sigma(\mathcal{G}^{(i,T)}(\boldsymbol{x}))\|_2 \leq \prod_{t=1}^{T+1}\varpi_t\|\boldsymbol{x}\|_2. \end{aligned}$$ ⊠

Now, we show that fully-connected networks are Lipschitz.

Lemma 4.6.2. *In addition to the assumptions of Lemma 4.6.1 assume σ is τ-Lipschitz. Then*

$$\begin{aligned}\|\mathcal{G}^{(1,T)}(\boldsymbol{x}_1) - \mathcal{G}^{(2,T)}(\boldsymbol{x}_2)\|_2 \leq \tau^{T-1}\prod_{t=1}^{T}\varpi_t\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_2 \\ + \sum_{t=1}^{T}\left(\tau^{T-t}\prod_{\substack{t'=1\\t'\neq t}}^{T}\varpi_{t'}\|\boldsymbol{x}_1\|_2\right)\|W_t^{(1)} - W_t^{(2)}\|_2.\end{aligned}$$

Proof. We use induction on T. First, observe for any t

$$\begin{aligned}\|\mathcal{G}_t^{(1)}(\boldsymbol{x}_1) - \mathcal{G}_t^{(2)}(\boldsymbol{x}_2)\|_2 &= \|W_t^{(1)}\boldsymbol{x}_1 - W_t^{(2)}\boldsymbol{x}_2\|_2 \\ &= \|W_t^{(1)}\boldsymbol{x}_1 - W_t^{(2)}\boldsymbol{x}_1 + W_t^{(2)}\boldsymbol{x}_1 - W_t^{(2)}\boldsymbol{x}_2\|_2 \\ &\leq \varpi_t\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_2 + \|\boldsymbol{x_1}\|_2\,\|W_t^{(1)} - W_t^{(2)}\|_2.\end{aligned}$$

Thus, the base case $T = 1$ holds. Additionally, for any $t > 1$

$$\begin{aligned}\|\mathcal{G}^{(1,t)}(\boldsymbol{x}_1) - \mathcal{G}^{(2,t)}(\boldsymbol{x}_2)\|_2 &\leq \tau\varpi_t\|\mathcal{G}^{(1,t-1)}(\boldsymbol{x}_1) - \mathcal{G}^{(2,t-1)}(\boldsymbol{x}_2)\|_2 \\ &\quad + \|\sigma(\mathcal{G}^{(1,t-1)}(\boldsymbol{x}_1))\|_2\,\|W_t^{(1)} - W_t^{(2)}\|_2 \\ &\leq \tau\varpi_t\|\mathcal{G}^{(1,t-1)}(\boldsymbol{x}_1) - \mathcal{G}^{(2,t-1)}(\boldsymbol{x}_2)\|_2 + \prod_{t'=1}^{t-1}\varpi_{t'}\|\boldsymbol{x}_1\|_2\,\|W_t^{(1)} - W_t^{(2)}\|_2\end{aligned} \tag{4.43}$$

where we used Lemma 4.6.1 in the final inequality.

Assume the induction hypothesis holds for fixed $T > 1$. Then using equation (4.43)

$$
\begin{aligned}
&\|\mathcal{G}^{(1,T+1)}(\boldsymbol{x}_1) - \mathcal{G}^{(2,T+1)}(\boldsymbol{x}_2)\|_2 \\
&\quad \leq \tau\varpi_{T+1}\|\mathcal{G}^{(1,T)}(\boldsymbol{x}_1) - \mathcal{G}^{(2,T)}(\boldsymbol{x}_2)\|_2 + \prod_{t=1}^{T}\varpi_t\|\boldsymbol{x}_1\|_2\,\|W_{T+1}^{(1)} - W_{T+1}^{(2)}\|_2 \\
&\quad \leq \tau\varpi_{T+1}\tau^{T-1}\prod_{t=1}^{T}\varpi_t\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_2 + \tau\varpi_{T+1}\sum_{t=1}^{T}\left(\tau^{T-t}\prod_{\substack{t'=1\\t'\neq t}}^{T}\varpi_{t'}\|\boldsymbol{x}_1\|_2\right)\|W_t^{(1)} - W_t^{(2)}\|_2 \\
&\qquad + \prod_{t=1}^{T}\varpi_t\|\boldsymbol{x}_1\|_2\,\|W_{T+1}^{(1)} - W_{T+1}^{(2)}\|_2 \\
&\quad = \tau^T\prod_{t=1}^{T+1}\varpi_t\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_2 + \sum_{t=1}^{T}\left(\tau^{T+1-t}\prod_{\substack{t'=1\\t'\neq t}}^{T+1}\varpi_{t'}\|\boldsymbol{x}_1\|_2\right)\|W_t^{(1)} - W_t^{(2)}\|_2 \\
&\qquad + \tau^{T+1-(T+1)}\prod_{\substack{t'=1\\t'\neq T+1}}^{T+1}\varpi_t\|\boldsymbol{x}_1\|_2\,\|W_{T+1}^{(1)} - W_{T+1}^{(2)}\|_2 \\
&\quad = \tau^T\prod_{t=1}^{T+1}\varpi_t\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_2 + \sum_{t=1}^{T+1}\left(\tau^{T+1-t}\prod_{\substack{t'=1\\t'\neq t}}^{T+1}\varpi_{t'}\|\boldsymbol{x}_1\|_2\right)\|W_t^{(1)} - W_t^{(2)}\|_2
\end{aligned}
$$

producing the desired result. ⊠

4.6.2 Generalization Error Bound for PGD DR-CG-Net

In this section, we state and prove the generalization error bound for PGD DR-CG-Net where a PGD scale-variable update is implemented in DR-CG-Net. Furthermore, asymptotic forms of the GEB for PGD DR-CG-Net are derived.

Note that, for PGD DR-CG-Net, the scale-variable parameter spaces, Ω_d from (4.23), are

$$
\Omega_d = \begin{cases} \{W \in \mathbb{R}^{nf_d \times nf_{d-1}} : \|W\|_2 \leq w_d\} & d \in \mathbb{N}[L_c] \\ [-\delta, \delta] & d = L_c + 1. \end{cases} \tag{4.44}
$$

for real valued constants $w_d > 0$ and $\delta > 0$. Note, each $W \in \Omega_d$, for $d \in \mathbb{N}[L_c]$, corresponds to a convolutional layer mapping a $\sqrt{n} \times \sqrt{n}$ image from f_{d-1} to f_d filter channels using convolutional kernels of size $k_d \times k_d$. Thus, the hypothesis class for PGD DR-CG-Net is

$$\mathcal{H}_{\text{CG}}^{(3)} = \left\{ \boldsymbol{c}\left(\cdot; \{P_u, \delta_k^{(j)}, W_{k,\ell}^{(j)}\}_{\substack{k\in\mathbb{N}[K] \\ \ell\in\mathbb{N}[L_c]}}^{j\in\mathbb{N}[J]}\right) : P_u \in \mathcal{P}_{\text{tri}}, W_{k,\ell}^{(j)} \in \Omega_\ell, \delta_k^{(j)} \in \Omega_{L_c+1} \right\}.$$

Theorem 4.6.3 (Generalization Error Bound for PGD DR-CG-Net)**.** *Let $\mathcal{S} = \{(\overline{\boldsymbol{y}}_i, \overline{\boldsymbol{c}}_i)\}_{i=1}^{N_s}$ be a training dataset such that Assumption 4.4.1 holds for each $\overline{\boldsymbol{c}}_i$, $\overline{\boldsymbol{y}}_i$ is given by (4.12), and $y_{\max} = \max_{1\le i\le N_s} \|\overline{\boldsymbol{y}}_i\|_2$. Then for all $\widehat{\boldsymbol{c}} \in \mathcal{H}_{\text{CG}}^{(3)}$, with probability at least $1-\varepsilon$, the generalization error of PGD DR-CG-Net is bounded as*

$$\begin{aligned}
\mathcal{L}(\widehat{\boldsymbol{c}}) \le \mathcal{L}_{\mathcal{S}}(\widehat{\boldsymbol{c}}) &+ \frac{8c_{\max}}{\sqrt{nN_s}}\sqrt{2n-1}\sqrt{\ln\left(e\left(1+\frac{4p_{\max}(KJ(L_c+1)+1)\kappa}{c_{\max}}\right)\right)} \\
&+ \frac{8c_{\max}}{\sqrt{nN_s}}\sum_{k=1}^{K}\sum_{j=1}^{J}\sum_{d=1}^{L_c}\sqrt{f_{d-1}f_d k_d^2}\sqrt{\ln\left(e\left(1+\frac{4w_d(KJ(L_c+1)+1)\kappa_{k,d}^{(j)}}{c_{\max}}\right)\right)} \\
&+ \frac{8c_{\max}}{\sqrt{nN_s}}\sum_{k=1}^{K}\sum_{j=1}^{J}\sqrt{\ln\left(e\left(1+\frac{4\delta(KJ(L_c+1)+1)\kappa_{k,L_c+1}^{(j)}}{c_{\max}}\right)\right)} \\
&+ 4c_{\max}\sqrt{2\ln(4/\varepsilon)/(nN_s)}
\end{aligned}$$

for

$$\begin{aligned}
\kappa &= z_\infty(c_1 + p_{\max}y_{\max}\|A\|_\infty)\widehat{c}_1^{(K,J)} + z_\infty c_2 \\
\kappa_{k,d}^{(j)} &= z_\infty(c_1 + p_{\max}y_{\max}\|A\|_\infty)\widehat{c}_{k,j,d,2}^{(K,J)}
\end{aligned}$$

where

$$
\begin{aligned}
c_1 &= p_{\max} y_{\max} \ \|A\|_2 \left(1 + 2 z_\infty^2 p_{\max} \ \|A\|_2^2\right) \\
c_2 &= z_\infty y_{\max} \|A\|_2 \left(p_{\max}/p_{\min}\right)^2 \\
\widehat{c}_1^{(K,J)} &= c_2 r_2 \left(\frac{1 - r_1^J}{1 - r_1}\right) \left(\frac{1 - \left(r_1^J + c_1 r_2 \frac{1 - r_1^J}{1 - r_1}\right)^K}{1 - \left(r_1^J + c_1 r_2 \frac{1 - r_1^J}{1 - r_1}\right)}\right) \\
\widehat{c}_{k,j,d,2}^{(K,J)} &= r_1^{J-j} \left(r_1^J + c_1 r_2 \frac{1 - r_1^J}{1 - r_1}\right)^{K-k} \begin{cases} \left(\sqrt{n} z_\infty \prod_{\substack{\ell=1 \\ \ell \neq d}}^{L_c} w_\ell\right) & d = 1, 2, \ldots, L_c \\ \xi & d = L_c + 1 \end{cases}
\end{aligned}
$$

for

$$
\begin{aligned}
r_1 &= 1 + \delta (z_\infty p_{\max} y_{\max} \|A\|_2 \|A\|_\infty)^2 + \prod_{\ell=1}^{L_c} w_\ell \\
r_2 &= \delta y_{\max} \|A\|_2 \left(1 + z_\infty^2 p_{\max} \|A\|_2 (\|A\|_2 + \|A\|_\infty)\right).
\end{aligned}
$$

That is, the generalization error of PGD DR-CG-Net is bounded as in Theorem 4.4.15 with $\tau = 1/\sqrt{n}$, $\dim(\mathcal{P}) = 2n - 1$, $c = c_{\max}/\sqrt{n}$, $D = L_c + 1$,

$$
(\alpha_d, \omega_d) = \begin{cases} (f_{d-1} f_d k_d^2, w_d) & d = 1, 2, \ldots, L_c \\ (1, \delta) & d = L_c + 1, \end{cases}
$$

$\widehat{r}_{k,1}^{(J)} = r_1^J, \quad \widehat{r}_{k,2}^{(J)} = r_2 \dfrac{1 - r_1^J}{1 - r_1}$, and

$$
\widehat{r}_{k,d,3}^{(j,J)} = r_1^{J-j} \begin{cases} \left(\sqrt{n} z_\infty \prod_{\substack{\ell=1 \\ \ell \neq d}}^{L_c} w_\ell\right) & d = 1, 2, \ldots, L_c \\ \xi & d = L_c + 1. \end{cases}
$$

As an overview, proving Theorem 4.6.3 consists of the following steps: First, we show Assumption 4.4.2 holds for the mean absolute error loss function used in DR-CG-Net. Second, we invoke Lemma 4.6.2 to show that the Lipschitz condition of Assumption 4.4.3 holds for each PGD

DR-CG-Net scale variable update method in (4.17). Finally, we apply Theorem 4.4.15 to produce Theorem 4.6.3.

Proof of Theorem 4.6.3. As DR-CG-Net employs the mean absolute loss function, $L(\boldsymbol{x}_1, \boldsymbol{x}_2) = \frac{1}{n}\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_1$, for any $\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x} \in \mathbb{R}^n$ using the reverse triangle inequality and Lemma 4.4.4 results in

$$|L(\boldsymbol{x}_1, \boldsymbol{x}) - L(\boldsymbol{x}_2, \boldsymbol{x})| \leq L(\boldsymbol{x}_1, \boldsymbol{x}_2) = \frac{1}{n}\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_1 \leq \frac{1}{\sqrt{n}}\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_2 \leq \frac{c_{\max}}{\sqrt{n}}.$$

Hence, Assumption 4.4.2 holds for $c = c_{\max}/\sqrt{n}$ and $\tau = 1/\sqrt{n}$.

Let $\mathcal{V}_k^{(j)}$ and $\widetilde{\mathcal{V}}_k^{(j)}$ be the convolutional subnetworks of DR-CG-Net, which are parameterized by $\{W_{k,\ell}^{(j)}\}_{\ell \in \mathbb{N}[L_c]}$ and $\{\widetilde{W}_{k,\ell}^{(j)}\}_{\ell \in \mathbb{N}[L_c]}$, respectively. Using (4.17) and the triangle inequality observe

$$\begin{aligned}
&\left\| g_k^{(j)}\left(\boldsymbol{z}_1, \boldsymbol{u}_1; \{\delta_k^{(j)}, W_{k,\ell}^{(j)}\}_{\ell \in \mathbb{N}[L_c]}\right) - g_k^{(j)}\left(\boldsymbol{z}_2, \boldsymbol{u}_2; \{\widetilde{\delta}_k^{(j)}, \widetilde{W}_{k,\ell}^{(j)}\}_{\ell \in \mathbb{N}[L_c]}\right)\right\|_2 \\
&\quad = \left\| v_k^{(j)}(\boldsymbol{z}_1, \boldsymbol{u}_1; \delta_k^{(j)}) + \mathcal{V}_k^{(j)}(\boldsymbol{z}_1) - v_k^{(j)}(\boldsymbol{z}_2, \boldsymbol{u}_2; \widetilde{\delta}_k^{(j)}) - \widetilde{\mathcal{V}}_k^{(j)}(\boldsymbol{z}_2)\right\|_2 \\
&\quad \leq \|v_k^{(j)}(\boldsymbol{z}_1, \boldsymbol{u}_1; \delta_k^{(j)}) - v_k^{(j)}(\boldsymbol{z}_2, \boldsymbol{u}_2; \widetilde{\delta}_k^{(j)})\|_2 + \|\mathcal{V}_k^{(j)}(\boldsymbol{z}_1) - \widetilde{\mathcal{V}}_k^{(j)}(\boldsymbol{z}_2)\|_2. \qquad (4.45)
\end{aligned}$$

For any $\overline{\boldsymbol{y}}_p$, where $p \in \mathbb{N}[N_s]$, define $\boldsymbol{d}_i = A_{\boldsymbol{u}_i}^T(A_{\boldsymbol{u}_i}\boldsymbol{z}_i - \overline{\boldsymbol{y}}_p)$ for $i = 1, 2$. Using (4.18) and the triangle inequality note

$$\begin{aligned}
\|v_k^{(j)}(\boldsymbol{z}_1, \boldsymbol{u}_1; \delta_k^{(j)}) - v_k^{(j)}(\boldsymbol{z}_2, \boldsymbol{u}_2; \widetilde{\delta}_k^{(j)})\|_2 &= \|\boldsymbol{z}_1 - \delta_k^{(j)}\rho_\xi(\boldsymbol{d}_1) - \boldsymbol{z}_2 + \widetilde{\delta}_k^{(j)}\rho_\xi(\boldsymbol{d}_2)\|_2 \\
&\leq \|\boldsymbol{z}_1 - \boldsymbol{z}_2\|_2 + \|\delta_k^{(j)}\rho_\xi(\boldsymbol{d}_1) - \widetilde{\delta}_k^{(j)}\rho_\xi(\boldsymbol{d}_2)\|_2. \qquad (4.46)
\end{aligned}$$

As ρ_ξ is 1-Lipschitz by Lemma 4.2.23 and bounded, in norm $\|\cdot\|_2$, by ξ then

$$\begin{aligned}
\|\delta_k^{(j)}\rho_\xi(\boldsymbol{d}_1) - \widetilde{\delta}_k^{(j)}\rho_\xi(\boldsymbol{d}_2)\|_2 &= \|\delta_k^{(j)}\rho_\xi(\boldsymbol{d}_1) - \delta_k^{(j)}\rho_\xi(\boldsymbol{d}_2) + \delta_k^{(j)}\rho_\xi(\boldsymbol{d}_2) - \widetilde{\delta}_k^{(j)}\rho_\xi(\boldsymbol{d}_2)\|_2 \\
&\leq |\delta_k^{(j)}|\ \|\rho_\xi(\boldsymbol{d}_1) - \rho_\xi(\boldsymbol{d}_2)\|_2 + \|\rho_\xi(\boldsymbol{d}_2)\|_2\ |\delta_k^{(j)} - \widetilde{\delta}_k^{(j)}| \\
&\leq \delta\|\boldsymbol{d}_1 - \boldsymbol{d}_2\|_2 + \xi|\delta_k^{(j)} - \widetilde{\delta}_k^{(j)}|. \qquad (4.47)
\end{aligned}$$

Now, let $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$ be Tikhonov solutions. That is, $\boldsymbol{u}_i = \mathcal{T}_{\overline{\boldsymbol{y}}_p}(\boldsymbol{z}_i; P_i)$ for a measurement $\overline{\boldsymbol{y}}_p$, some $\boldsymbol{z}_i \in [0,\infty)^n$ satisfying $\|\boldsymbol{z}_i\|_\infty \leq z_\infty$, and some covariance matrix $P_i \in \mathcal{P}_{\text{tri}}$ where $i = 1, 2$ and $p \in \mathbb{N}[N_s]$. Then equation (4.39) gives

$$\begin{aligned}\|\boldsymbol{d}_1 - \boldsymbol{d}_2\|_2 \leq (z_\infty p_{\max} y_{\max} \|A\|_2 \|A\|_\infty)^2 \|\boldsymbol{z}_1 - \boldsymbol{z}_2\|_2 \\ + y_{\max}\|A\|_2 \left(1 + z_\infty^2 p_{\max}\|A\|_2(\|A\|_2 + \|A\|_\infty)\right) \|\boldsymbol{u}_1 - \boldsymbol{u}_2\|_2,\end{aligned}$$

which combined with (4.46) with (4.47) produces

$$\begin{aligned}&\|v_k^{(j)}(\boldsymbol{z}_1, \boldsymbol{u}_1; \delta_k^{(j)}) - v_k^{(j)}(\boldsymbol{z}_2, \boldsymbol{u}_2; \widetilde{\delta}_k^{(j)})\|_2 \\ &\quad \leq \|\boldsymbol{z}_1 - \boldsymbol{z}_2\|_2 + \delta\|\boldsymbol{d}_1 - \boldsymbol{d}_2\|_2 + \xi|\delta_k^{(j)} - \widetilde{\delta}_k^{(j)}| \\ &\quad \leq \|\boldsymbol{z}_1 - \boldsymbol{z}_2\|_2 + \delta(z_\infty p_{\max} y_{\max}\|A\|_2\|A\|_\infty)^2 \|\boldsymbol{z}_1 - \boldsymbol{z}_2\|_2 \\ &\qquad + \delta y_{\max}\|A\|_2 \left(1 + z_\infty^2 p_{\max}\|A\|_2(\|A\|_2 + \|A\|_\infty)\right) \|\boldsymbol{u}_1 - \boldsymbol{u}_2\|_2 + \xi|\delta_k^{(j)} - \widetilde{\delta}_k^{(j)}| \\ &\quad = \left(1 + \delta(z_\infty p_{\max} y_{\max}\|A\|_2\|A\|_\infty)^2\right) \|\boldsymbol{z}_1 - \boldsymbol{z}_2\|_2 + r_2 \|\boldsymbol{u}_1 - \boldsymbol{u}_2\|_2 + \xi|\delta_k^{(j)} - \widetilde{\delta}_k^{(j)}|. \quad (4.48)\end{aligned}$$

Next, as $W_{k,d}^{(j)}, \widetilde{W}_{k,d}^{(j)} \in \Omega_d$, for $d \in \mathbb{N}[L_c]$ then $W_{k,d}^{(j)}$, and $\widetilde{W}_{k,d}^{(j)}$ are defined by $f_{d-1}f_d k_d^2$ parameters and satisfy $\|W_{k,d}^{(j)}\|_2 \leq w_d$ and $\|\widetilde{W}_{k,d}^{(j)}\|_2 \leq w_d$. Additionally, $\delta_k^{(j)} \in \Omega_{L_c+1}$ is a positive real number satisfying $|\delta_k^{(j)}| \leq \delta$. Hence, the dimension of the parameter spaces α_d and bounds on the parameter spaces ω_d are given by

$$(\alpha_d, \omega_d) = \begin{cases} (f_{d-1}f_d k_d^2, w_d) & d = 1, 2, \ldots, L_c \\ (1, \delta) & d = L_c + 1. \end{cases}$$

Additionally, note that every Tikhonov solution covariance matrix has tridiagonal structure and $\sqrt{\dim(\mathcal{P}_{\text{tri}})} = \sqrt{2n-1}$.

Now, as the ReLU activation function is 1-Lipschitz and $||\boldsymbol{z}_1||_2 \leq \sqrt{n} z_\infty$ then from Lemma 4.6.2,

$$\|\mathcal{V}_k^{(j)}(\boldsymbol{z}_1) - \widetilde{\mathcal{V}}_k^{(j)}(\boldsymbol{z}_2)\|_2 \leq \prod_{\ell=1}^{L_c} w_\ell \|\boldsymbol{z}_1 - \boldsymbol{z}_2\|_2 + \sum_{\ell=1}^{L_c} \left(\sqrt{n} z_\infty \prod_{\substack{\ell'=1 \\ \ell' \neq \ell}}^{L_c} w_{\ell'} \right) \left\| W_{k,\ell}^{(j)} - \widetilde{W}_{k,\ell}^{(j)} \right\|_2. \quad (4.49)$$

Combining (4.49) and (4.48) with (4.45) produces

$$\begin{aligned}
&\left\| g_k^{(j)} \left(\boldsymbol{z}_1, \boldsymbol{u}_1; \{\delta_k^{(j)}, W_{k,\ell}^{(j)}\}_{\ell \in \mathbb{N}[L_c]} \right) - g_k^{(j)} \left(\boldsymbol{z}_2, \boldsymbol{u}_2; \{\widetilde{\delta}_k^{(j)}, \widetilde{W}_{k,\ell}^{(j)}\}_{\ell \in \mathbb{N}[L_c]} \right) \right\|_2 \\
&\quad \leq \|v_k^{(j)}(\boldsymbol{z}_1, \boldsymbol{u}_1; \delta_k^{(j)}) - v_k^{(j)}(\boldsymbol{z}_2, \boldsymbol{u}_2; \widetilde{\delta}_k^{(j)})\|_2 + \|\mathcal{V}_k^{(j)}(\boldsymbol{z}_1) - \widetilde{\mathcal{V}}_k^{(j)}(\boldsymbol{z}_2)\|_2 \\
&\quad \leq \left(1 + \delta(z_\infty p_{\max} y_{\max} \|A\|_2 \|A\|_\infty)^2\right) \|\boldsymbol{z}_1 - \boldsymbol{z}_2\|_2 + r_2 \|\boldsymbol{u}_1 - \boldsymbol{u}_2\|_2 + \xi |\delta_k^{(j)} - \widetilde{\delta}_k^{(j)}| \\
&\qquad + \prod_{\ell=1}^{L_c} w_\ell \|\boldsymbol{z}_1 - \boldsymbol{z}_2\|_2 + \sum_{\ell=1}^{L_c} \left(\sqrt{n} z_\infty \prod_{\substack{\ell'=1 \\ \ell' \neq \ell}}^{L_c} w_{\ell'} \right) \left\| W_{k,\ell}^{(j)} - \widetilde{W}_{k,\ell}^{(j)} \right\|_2 \\
&= r_1 \|\boldsymbol{z}_1 - \boldsymbol{z}_2\|_2 + r_2 \|\boldsymbol{u}_1 - \boldsymbol{u}_2\| + \sum_{\ell=1}^{L_c} \left(\sqrt{n} z_\infty \prod_{\substack{\ell'=1 \\ \ell' \neq \ell}}^{L_c} w_{\ell'} \right) \left\| W_{k,\ell}^{(j)} - \widetilde{W}_{k,\ell}^{(j)} \right\|_2 + \xi |\delta_k^{(j)} - \widetilde{\delta}_k^{(j)}|.
\end{aligned}$$

Therefore, Assumption 4.4.3 holds with $r_{k,1}^{(j-1)} = r_1$, $r_{k,2}^{(j-1)} = r_2$, and

$$(\boldsymbol{\theta}_{k,d}^{(j)}, r_{k,d,3}^{(j-1)}, \|\cdot\|_{(d)}) = \begin{cases} \left(W_{k,d}^{(j)}, \sqrt{n} z_\infty \prod_{\substack{\ell=1 \\ \ell \neq d}}^{L_c} w_\ell, \|\cdot\|_2 \right) & d = 1, 2, \ldots, L_c \\ (\delta_k^{(j)}, \xi, |\cdot|) & d = L_c + 1. \end{cases}$$

Hence, the constants $\widehat{r}_{k,1}^{(J)}, \widehat{r}_{k,2}^{(J)}$, and $\widehat{r}_{k,d,3}^{(j,J)}$ from Proposition 4.4.6 are given as

$$\begin{aligned}
\widehat{r}_{k,1}^{(J)} &= \prod_{j=1}^{J} r_{k,1}^{(j-1)} = r_1^J \\
\widehat{r}_{k,2}^{(J)} &= \sum_{j=1}^{J} r_{k,2}^{(j-1)} \prod_{\ell=j}^{J-1} r_{k,1}^{(\ell)} = r_2 \sum_{j=1}^{J} r_1^{J-j} = r_2 \frac{1 - r_1^J}{1 - r_1},
\end{aligned}$$

and

$$\widehat{r}_{k,d,3}^{(j,J)} = r_{k,d,3}^{(j-1)} \prod_{\ell=j}^{J-1} r_{k,1}^{(\ell)} = r_{k,d,3}^{(j-1)} r_1^{J-j} = r_1^{J-j} \begin{cases} \sqrt{n} z_\infty \prod_{\substack{\ell=1 \\ \ell \neq d}}^{L_c} w_\ell & d = 1, 2, \ldots, L_c \\ \xi & d = L_c + 1. \end{cases}$$

Thus, the constants $\widehat{c}_1^{(K,J)}$ and $\widehat{c}_{k,j,d,2}^{(K,J)}$ from Theorem 4.4.15 are given as

$$\begin{aligned} \widehat{c}_1^{(K,J)} &= c_2 \sum_{k=1}^{K} \widehat{r}_{k,2}^{(J)} \prod_{\ell=k+1}^{K} (\widehat{r}_{\ell,1}^{(J)} + \widehat{r}_{\ell,2}^{(J)} c_1) \\ &= c_2 \sum_{k=1}^{K} r_2 \frac{1 - r_1^J}{1 - r_1} \prod_{\ell=k+1}^{K} \left(r_1^J + r_2 \frac{1 - r_1^J}{1 - r_1} c_1 \right) \\ &= c_2 r_2 \frac{1 - r_1^J}{1 - r_1} \sum_{k=1}^{K} \left(r_1^J + r_2 \frac{1 - r_1^J}{1 - r_1} c_1 \right)^{K-k} \\ &= c_2 r_2 \frac{1 - r_1^J}{1 - r_1} \sum_{k=0}^{K-1} \left(r_1^J + r_2 \frac{1 - r_1^J}{1 - r_1} c_1 \right)^{k} \\ &= c_2 r_2 \left(\frac{1 - r_1^J}{1 - r_1} \right) \left(\frac{1 - \left(r_1^J + r_2 \frac{1 - r_1^J}{1 - r_1} c_1 \right)^K}{1 - \left(r_1^J + r_2 \frac{1 - r_1^J}{1 - r_1} c_1 \right)} \right) \end{aligned}$$

and

$$\begin{aligned} \widehat{c}_{k,j,d,2}^{(K,J)} &= \widehat{r}_{k,d,3}^{(j,J)} \prod_{\ell=k+1}^{K} (\widehat{r}_{\ell,1}^{(J)} + \widehat{r}_{\ell,2}^{(J)} c_1) \\ &= \widehat{r}_{k,d,3}^{(j,J)} \prod_{\ell=k+1}^{K} \left(r_1^J + r_2 \frac{1 - r_1^J}{1 - r_1} c_1 \right) \\ &= \widehat{r}_{k,d,3}^{(j,J)} \left(r_1^J + r_2 \frac{1 - r_1^J}{1 - r_1} c_1 \right)^{K-k} \\ &= r_1^{J-j} \left(r_1^J + r_2 \frac{1 - r_1^J}{1 - r_1} c_1 \right)^{K-k} \begin{cases} \sqrt{n} z_\infty \prod_{\substack{\ell=1 \\ \ell \neq d}}^{L_c} w_\ell & d = 1, 2, \ldots, L_c \\ \xi & d = L_c + 1 \end{cases} \end{aligned}$$

for

$$c_1 = p_{\max} y_{\max} \, \|A\|_2 \left(1 + 2z_\infty^2 p_{\max} \, \|A\|_2^2\right)$$
$$c_2 = z_\infty y_{\max} \|A\|_2 \left(p_{\max}/p_{\min}\right)^2 .$$

Lastly, applying Theorem 4.4.15 produces the desired generalization error bound for PGD DR-CG-Net. ⊠

4.6.3 Asymptotic Forms of PGD DR-CG-Net Generalization Error Bounds

In this section we provide asymptotic forms for the PGD DR-CG-Net generalization error bound given in Theorem 4.6.3. We consider asymptotic bounds as the size of the network – i.e. K and J, the number of unrolled iterations of CG-LS – grows, as the dimension of the measurements and reconstructed signal – i.e. m and n, respectively – grow, and as the number of training data samples – i.e. N_s – grows.

Recall the values $c_{\max}, p_{\max}, p_{\min}, f_d, k_d, w_d, \delta, z_\infty, y_{\max}, \|A\|_2, \|A\|_\infty$, and ξ that are utilized by Theorem 4.6.3. As $p_{\max}, p_{\min}, f_d, k_d, w_d, \delta, z_\infty$, and ξ are all parameters chosen and fixed during the construction of PGD DR-CG-Net before training, we assume that each is independent of network size, measurement dimension, reconstructed signal dimension, and training dataset size. Thus, we establish asymptotic forms for the PGD DR-CG-Net generalization error that rely on whether $c_{\max}, y_{\max}, \|A\|_2$, and $\|A\|_\infty$ depend on the measurement and reconstructed signal dimension.

Finally, recall that we use the notation $a \lesssim b$ to mean that $a \leq s_c b$ for some constant $s_c > 0$; that is, $a \lesssim b$ implies $a = \mathcal{O}(b)$.

Dependent on Signal Dimension

For the following asymptotic generalization error bound of PGD DR-CG-Net, we assume that each of $c_{\max}, y_{\max}, f_d, \|A\|_2$, and $\|A\|_\infty$ depend on the measurement and reconstructed signal dimension in that $\lim_{m,n\to\infty} c_{\max} \to \infty$, $\lim_{m,n\to\infty} y_{\max} \to \infty$, and $\lim_{m,n\to\infty} \|A\|_p \to \infty$ for any $1 \leq p \leq \infty$. Note for any $\boldsymbol{x} \in \mathbb{R}^k$ that $\|\boldsymbol{x}\|_2 \leq \sqrt{k}\|\boldsymbol{x}\|_\infty$. As $c_{\max}$ and $y_{\max}$ are, respectively, the

maximum Euclidean norm bounds on the signals of interest, of dimension n, and measurements, of dimension m, then $c_{\max} \lesssim \sqrt{n}$ and $y_{\max} \lesssim \sqrt{m}$; where we have reasonably assumed that the maximum entry in any measurement or reconstructed signal is independent of the signal dimension. Furthermore, as $\|A\|_2 \leq \sqrt{m}\|A\|_\infty$ and $\|A\|_\infty \leq n\left(\max_{i,j}|A_{i,j}|\right)$, since $\|A\|_\infty$ is the max absolute row sum, then $\|A\|_2 \lesssim n\sqrt{m}$ and $\|A\|_\infty \leq n$; where again we have reasonably assumed that the maximum entry of the measurement matrix is independent of the matrix dimensions.

Corollary 4.6.4. *Let $c_{\max}, y_{\max}, \|A\|_2$, and $\|A\|_\infty$ depend on the measurement and reconstructed signal dimension. The generalization error for PGD DR-CG-Net scales as*

$$|\mathcal{L}(\widehat{\boldsymbol{c}}) - \mathcal{L}_{\mathcal{S}}(\widehat{\boldsymbol{c}})| \lesssim c_{\max}\left(\sqrt{n} + KJ\left(1 + \sum_{d=1}^{L_c}\sqrt{f_{d-1}f_d k_d^2}\right)\right)\sqrt{\frac{\mathcal{F}(K,J,L_c)}{nN_s}} + c_{\max}KJ\left(\sum_{d=1}^{L_c}\sqrt{f_{d-1}f_d k_d^2}\right)\sqrt{\frac{\ln(n)}{nN_s}}$$

for

$$\mathcal{F}(K,J,L_c) = \ln(KJL_c) + KJ\left(\ln(y_{\max}) + \ln(\|A\|_2) + \ln(\|A\|_\infty) + \ln\left(1 + \prod_{\ell=1}^{L_c} w_\ell\right)\right).$$

Furthermore, as, at most, $c_{\max} = \mathcal{O}(\sqrt{n})$, $y_{\max} = \mathcal{O}(\sqrt{m})$, $\|A\|_2 = \mathcal{O}(n\sqrt{m})$, and $\|A\|_\infty = \mathcal{O}(n)$ then generalization error for PGD DR-CG-Net scales at most as

$$|\mathcal{L}(\widehat{\boldsymbol{c}}) - \mathcal{L}_{\mathcal{S}}(\widehat{\boldsymbol{c}})| \lesssim \left(\sqrt{n} + KJL_c\right)\sqrt{\frac{KJ\left(\ln(m) + \ln(n) + L_c\right)}{N_s}}.$$

A proof of Corollary 4.6.4 is provided in Appendix C and results from ignoring constants to consider how the GEB from Theorem 4.6.3 scales in network size and signal dimension. From Corollary 4.6.4, once the amount of training data satisfies

$$N_s \sim \frac{c_{\max}^2\left[\left(\sqrt{n} + KJ\left(1 + \sum_{d=1}^{L_c}\sqrt{f_{d-1}f_d k_d^2}\right)\right)\sqrt{\mathcal{F}(K,J,L_c)} + KJ\left(\sum_{d=1}^{L_c}\sqrt{f_{d-1}f_d k_d^2}\right)\sqrt{\ln(n)}\right]^2}{n}$$

or

$$N_s \sim \left(\sqrt{n} + KJL_c\right)^2 KJ\left(\ln(m) + \ln(n) + L_c\right)$$

then the GEB of PGD DR-CG-Net is guaranteed to be small with high probability. Furthermore, Corollary 4.6.4 shows that, at worst, the PGD DR-CG-Net generalization error grows as $\mathcal{O}(\sqrt{n\ln(n)})$ in reconstructed signal dimension, $\mathcal{O}(\sqrt{(KJL_c)^3}) = \mathcal{O}((\text{Network Size})^{3/2})$ in network size, $\mathcal{O}(\sqrt{\ln(m)})$ in measurement dimension, and $\mathcal{O}(N_s^{-1/2})$ in training dataset size.

Independent Measurement Matrix Norm

Now we assume that $\|A\|_2$ and $\|A\|_\infty$ are independent of m and n defining the dimension of A. For instance, the measurement matrix could be normalized through preprocessing such that $\|A\|_2 = 1$.

Corollary 4.6.5. *Let $c_{\max}$ and $y_{\max}$ depend on the measurement and reconstructed signal dimension. Furthermore, assume $\|A\|_2$ and $\|A\|_\infty$ are independent of the measurement and reconstructed signal dimension. The generalization error for PGD DR-CG-Net scales as*

$$\begin{aligned}|\mathcal{L}(\widehat{\boldsymbol{c}}) - \mathcal{L}_{\mathcal{S}}(\widehat{\boldsymbol{c}})| \lesssim\ & c_{\max}\left(\sqrt{n} + KJ\left(1 + \sum_{d=1}^{L_c}\sqrt{f_{d-1}f_d k_d^2}\right)\right)\sqrt{\frac{\mathcal{F}(K,J,L_c)}{nN_s}} \\ & + c_{\max}KJ\left(\sum_{d=1}^{L_c}\sqrt{f_{d-1}f_d k_d^2}\right)\sqrt{\frac{\ln(n)}{nN_s}}\end{aligned}$$

for

$$\mathcal{F}(K,J,L_c) = \ln(KJL_c) + KJ\left(\ln(y_{\max}) + \ln\left(1 + \prod_{\ell=1}^{L_c} w_\ell\right)\right).$$

Furthermore, as, at most, $c_{\max} = \mathcal{O}(\sqrt{n})$ *and* $y_{\max} = \mathcal{O}(\sqrt{m})$ *then generalization error for PGD DR-CG-Net scales at most as*

$$|\mathcal{L}(\widehat{\boldsymbol{c}}) - \mathcal{L}_{\mathcal{S}}(\widehat{\boldsymbol{c}})| \lesssim \left(\sqrt{n} + KJL_c\right) \sqrt{\frac{KJ\left(\ln(m) + L_c\right)}{N_s}} + KJL_c \sqrt{\frac{\ln(n)}{N_s}}.$$

Corollary 4.6.5 is a direct consequence of Corollary 4.6.4 where the $\|A\|_p$ terms, for $p = 2, \infty$, are removed as they are now of order $\mathcal{O}(1)$. Furthermore, as in Corollary 4.6.4, Corollary 4.6.5 results from ignoring constants to consider how the GEB from Theorem 4.6.3 scales in network size and signal dimension. From Corollary 4.6.5, once the amount of training data satisfies

$$N_s \sim \frac{c_{\max}^2 \left[\left(\sqrt{n} + KJ\left(1 + \sum_{d=1}^{L_c} \sqrt{f_{d-1} f_d k_d^2}\right)\right) \sqrt{\mathcal{F}(K, J, L_c)} + KJ\left(\sum_{d=1}^{L_c} \sqrt{f_{d-1} f_d k_d^2}\right) \sqrt{\ln(n)}\right]^2}{n}$$

or

$$N_s \sim \left[\left(\sqrt{n} + KJL_c\right) \sqrt{KJ\left(\ln(m) + L_c\right)} + KJL_c \sqrt{\ln(n)}\right]^2$$

then the GEB of PGD DR-CG-Net is guaranteed to be small with high probability. Furthermore, Corollary 4.6.5 shows that, at worst, the PGD DR-CG-Net generalization error grows as $\mathcal{O}(\sqrt{(KJL_c)^3}) = \mathcal{O}((\text{Network Size})^{3/2})$ in network size, $\mathcal{O}(\sqrt{n})$ in reconstructed signal dimension, $\mathcal{O}(\sqrt{\ln(m)})$ in measurement dimension, and $\mathcal{O}(N_s^{-1/2})$ in training dataset size.

Independence of Signal Dimension

Here, we assume that $c_{\max}$, $y_{\max}$, $\|A\|_2$, and $\|A\|_\infty$ are independent of m and n defining the measurement and reconstructed signal dimension. As an example, CG-Net and DR-CG-Net from Chapter 2 and Chapter 3 implement preprocessing to scale all images of interest to be in the unit ball. That is, $c_{\max} \leq 1$. Additionally, if the measurement matrix is normalized through preprocessing such that $\|A\|_2 = 1$ then $y_{\max} \leq \|A\|_2 c_{\max} \leq 1$.

Corollary 4.6.6. *Let $c_{\max}$, $y_{\max}$, $\|A\|_2$, and $\|A\|_\infty$ are independent of the measurement and reconstructed signal dimension. The generalization error for PGD DR-CG-Net scales as*

$$|\mathcal{L}(\widehat{\boldsymbol{c}}) - \mathcal{L}_{\mathcal{S}}(\widehat{\boldsymbol{c}})| \lesssim \left(\sqrt{n} + KJ\left(1 + \sum_{d=1}^{L_c}\sqrt{f_{d-1}f_d k_d^2}\right)\right)\sqrt{\frac{\mathcal{F}(K,J,L_c)}{nN_s}} + KJ\left(\sum_{d=1}^{L_c}\sqrt{f_{d-1}f_d k_d^2}\right)\sqrt{\frac{\ln(n)}{nN_s}}$$

for

$$\mathcal{F}(K,J,L_c) = \ln(KJL_c) + KJ\ln\left(1 + \prod_{\ell=1}^{L_c} w_\ell\right).$$

Furthermore, the generalization error for PGD DR-CG-Net scales at most as

$$|\mathcal{L}(\widehat{\boldsymbol{c}}) - \mathcal{L}_{\mathcal{S}}(\widehat{\boldsymbol{c}})| \lesssim \left(\sqrt{n} + KJL_c\right)\sqrt{\frac{KJL_c}{nN_s}} \lesssim \sqrt{\frac{(KJL_c)^3}{N_s}}.$$

Corollary 4.6.6 is a direct consequence of Corollary 4.6.4 where the $\|A\|_p$ terms, for $p = 2, \infty$, and $c_{\max}$ and $y_{\max}$ terms are removed as they are now of order $\mathcal{O}(1)$. Furthermore, as in Corollary 4.6.4, Corollary 4.6.6 results from ignoring constants to consider how the GEB from Theorem 4.6.3 scales in network size and reconstructed signal dimension. From Corollary 4.6.6, once the amount of training data satisfies

$$N_s \sim \frac{c_{\max}^2\left[\left(\sqrt{n} + KJ\left(1 + \sum_{d=1}^{L_c}\sqrt{f_{d-1}f_d k_d^2}\right)\right)\sqrt{\mathcal{F}(K,J,L_c)} + KJ\left(\sum_{d=1}^{L_c}\sqrt{f_{d-1}f_d k_d^2}\right)\sqrt{\ln(n)}\right]^2}{n}$$

or

$$N_s \sim (KJL_c)^3$$

then the GEB of PGD DR-CG-Net is guaranteed to be small with high probability. Finally, Corollary 4.6.5 shows that, at worst, the PGD DR-CG-Net generalization error grows in network size as $\mathcal{O}(\sqrt{(KJL_c)^3}) = \mathcal{O}((\text{Network Size})^{3/2})$ and in training dataset size as $\mathcal{O}(N_s^{-1/2})$.

4.6.4 Generalization Error Bound for ISTA DR-CG-Net

In this section, we state and prove the generalization error bound for ISTA DR-CG-Net where an ISTA scale-variable update is implemented in DR-CG-Net. Furthermore, asymptotic forms of the GEB for ISTA DR-CG-Net are derived.

Note that, the scale-variable parameter spaces, Ω_d from (4.23), and hypothesis class, $\mathcal{H}^{(3)}_{\text{CG}}$, for ISTA DR-CG-Net are equivalent to those for PGD DR-CG-Net from Section 4.6.2. For convenience, we restate both the scale-variable parameter spaces and hypothesis class here. The scale-variable parameter spaces, Ω_d from (4.23), are

$$\Omega_d = \begin{cases} \{W \in \mathbb{R}^{nf_d \times nf_{d-1}} : \|W\|_2 \leq w_d\} & d \in \mathbb{N}[L_c] \\ [-\delta, \delta] & d = L_c + 1. \end{cases} \tag{4.50}$$

for real valued constants $w_d > 0$ and $\delta > 0$. Note, each $W \in \Omega_d$, for $d \in \mathbb{N}[L_c]$, corresponds to a convolutional layer mapping a $\sqrt{n} \times \sqrt{n}$ image from f_{d-1} to f_d filter channels using convolutional kernels of size $k_d \times k_d$. Thus, the hypothesis class for ISTA DR-CG-Net is

$$\mathcal{H}^{(3)}_{\text{CG}} = \left\{ \boldsymbol{c}\left(\cdot; \{P_u, \delta_k^{(j)}, W_{k,\ell}^{(j)}\}_{\substack{\ell \in \mathbb{N}[L_c]}}^{\substack{j \in \mathbb{N}[J] \\ k \in \mathbb{N}[K]}}\right) : P_u \in \mathcal{P}_{\text{tri}}, W_{k,\ell}^{(j)} \in \Omega_\ell, \delta_k^{(j)} \in \Omega_{L_c+1} \right\}.$$

Theorem 4.6.7 (Generalization Error Bound for ISTA DR-CG-Net)**.** *Let $\mathcal{S} = \{(\overline{\boldsymbol{y}}_i, \overline{\boldsymbol{c}}_i)\}_{i=1}^{N_s}$ be a training dataset such that Assumption 4.4.1 holds for each $\overline{\boldsymbol{c}}_i$, $\overline{\boldsymbol{y}}_i$ is given by (4.12), and $y_{\max} = \max_{1 \leq i \leq N_s} \|\overline{\boldsymbol{y}}_i\|_2$. Then for all $\widehat{\boldsymbol{c}} \in \mathcal{H}^{(3)}_{\text{CG}}$, with probability at least $1 - \varepsilon$, the generalization error of*

ISTA DR-CG-Net is bounded as

$$\begin{aligned}\mathcal{L}(\widehat{\boldsymbol{c}}) \leq \mathcal{L}_{\mathcal{S}}(\widehat{\boldsymbol{c}}) &+ \frac{8c_{\max}}{\sqrt{nN_s}}\sqrt{2n-1}\sqrt{\ln\left(e\left(1+\frac{4p_{\max}(KJ(L_c+1)+1)\kappa}{c_{\max}}\right)\right)} \\ &+ \frac{8c_{\max}}{\sqrt{nN_s}}\sum_{k=1}^{K}\sum_{j=1}^{J}\sum_{d=1}^{L_c}\sqrt{f_{d-1}f_d k_d^2}\sqrt{\ln\left(e\left(1+\frac{4w_d(KJ(L_c+1)+1)\kappa_{k,d}^{(j)}}{c_{\max}}\right)\right)} \\ &+ \frac{8c_{\max}}{\sqrt{nN_s}}\sum_{k=1}^{K}\sum_{j=1}^{J}\sqrt{\ln\left(e\left(1+\frac{4\delta(KJ(L_c+1)+1)\kappa_{k,L_c+1}^{(j)}}{c_{\max}}\right)\right)} \\ &+ 4c_{\max}\sqrt{2\ln(4/\varepsilon)/(nN_s)}\end{aligned}$$

for

$$\begin{aligned}\kappa &= z_\infty(c_1 + p_{\max}y_{\max}\|A\|_\infty)\widehat{c}_1^{(K,J)} + z_\infty c_2 \\ \kappa_{k,d}^{(j)} &= z_\infty(c_1 + p_{\max}y_{\max}\|A\|_\infty)\widehat{c}_{k,j,d,2}^{(K,J)}\end{aligned}$$

where

$$\begin{aligned}c_1 &= p_{\max}y_{\max}\ \|A\|_2\left(1+2z_\infty^2 p_{\max}\ \|A\|_2^2\right) \\ c_2 &= z_\infty y_{\max}\|A\|_2\left(p_{\max}/p_{\min}\right)^2 \\ \widehat{c}_1^{(K,J)} &= c_2 r_2\left(\frac{1-r_1^J}{1-r_1}\right)\left(\frac{1-\left(r_1^J + c_1 r_2\frac{1-r_1^J}{1-r_1}\right)^K}{1-\left(r_1^J + c_1 r_2\frac{1-r_1^J}{1-r_1}\right)}\right) \\ \widehat{c}_{k,j,d,2}^{(K,J)} &= r_1^{J-j}\left(r_1^J + c_1 r_2\frac{1-r_1^J}{1-r_1}\right)^{K-k}\begin{cases}(\sqrt{n}z_\infty + \delta\xi)\prod_{\substack{\ell=1\\ \ell\neq d}}^{L_c} w_\ell & d = 1,2,\ldots,L_c \\ \left(1+\prod_{\ell=1}^{L_c} w_\ell\right)\xi & d = L_c+1\end{cases}\end{aligned}$$

for

$$r_1 = \left(1 + \prod_{\ell=1}^{L_c} w_\ell\right)\left(1 + \delta(z_\infty p_{\max} y_{\max}\|A\|_2\|A\|_\infty)^2\right)$$
$$r_2 = \left(1 + \prod_{\ell=1}^{L_c} w_\ell\right)\delta y_{\max}\|A\|_2\left(1 + z_\infty^2 p_{\max}\|A\|_2(\|A\|_2 + \|A\|_\infty)\right)$$

That is, the generalization error of ISTA DR-CG-Net is bounded as in Theorem 4.4.15 with $\tau = 1/\sqrt{n}$, $\dim(\mathcal{P}) = 2n-1$, $c = c_{\max}/\sqrt{n}$, $D = L_c + 1$,

$$(\alpha_d, \omega_d) = \begin{cases} (f_{d-1} f_d k_d^2, w_d) & d = 1, 2, \ldots, L_c \\ (1, \delta) & d = L_c + 1, \end{cases}$$

$\widehat{r}_{k,1}^{(J)} = r_1^J, \quad \widehat{r}_{k,2}^{(J)} = r_2 \dfrac{1 - r_1^J}{1 - r_1}$, and

$$\widehat{r}_{k,d,3}^{(j,J)} = r_1^{J-j} \begin{cases} (\sqrt{n} z_\infty + \delta\xi) \prod_{\substack{\ell=1 \\ \ell \neq d}}^{L_c} w_\ell & d = 1, 2, \ldots, L_c \\ \left(1 + \prod_{\ell=1}^{L_c} w_\ell\right) \xi & d = L_c + 1. \end{cases}$$

As an overview, proving Theorem 4.6.7 consists of the following steps: First, we show Assumption 4.4.2 holds for the mean absolute error loss function used in DR-CG-Net. Second, we invoke Lemma 4.6.2 to show that the Lipschitz condition of Assumption 4.4.3 holds for each ISTA DR-CG-Net scale variable update method in (4.17). Finally, we apply Theorem 4.4.15 to produce Theorem 4.6.7.

Proof of Theorem 4.6.7. As DR-CG-Net employs the mean absolute loss function, $L(\boldsymbol{x}_1, \boldsymbol{x}_2) = \frac{1}{n}\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_1$, for any $\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x} \in \mathbb{R}^n$ using the reverse triangle inequality results in

$$|L(\boldsymbol{x}_1, \boldsymbol{x}) - L(\boldsymbol{x}_2, \boldsymbol{x})| \leq L(\boldsymbol{x}_1, \boldsymbol{x}_2) = \frac{1}{n}\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_1 \leq \frac{1}{\sqrt{n}}\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_2 \leq \frac{c_{\max}}{\sqrt{n}}.$$

Hence, Assumption 4.4.2 holds for $c = c_{\max}/\sqrt{n}$ and $\tau = 1/\sqrt{n}$.

Let $\mathcal{V}_k^{(j)}$ and $\widetilde{\mathcal{V}}_k^{(j)}$ be the convolutional subnetworks of DR-CG-Net, which are parameterized by $\{W_{k,\ell}^{(j)}\}_{\ell\in\mathbb{N}[L_c]}$ and $\{\widetilde{W}_{k,\ell}^{(j)}\}_{\ell\in\mathbb{N}[L_c]}$, respectively. Using (4.17) and the triangle inequality we observe

$$\begin{aligned}
&\left\| g_k^{(j)}\left(\boldsymbol{z}_1,\boldsymbol{u}_1;\{\delta_k^{(j)},W_{k,\ell}^{(j)}\}_{\ell\in\mathbb{N}[L_c]}\right) - g_k^{(j)}\left(\boldsymbol{z}_2,\boldsymbol{u}_2;\{\widetilde{\delta}_k^{(j)},\widetilde{W}_{k,\ell}^{(j)}\}_{\ell\in\mathbb{N}[L_c]}\right)\right\|_2 \\
&= \left\| v_k^{(j)}(\boldsymbol{z}_1,\boldsymbol{u}_1;\delta_k^{(j)}) + \mathcal{V}_k^{(j)}\left(v_k^{(j)}(\boldsymbol{z}_1,\boldsymbol{u}_1;\delta_k^{(j)})\right) - v_k^{(j)}(\boldsymbol{z}_2,\boldsymbol{u}_2;\widetilde{\delta}_k^{(j)}) - \widetilde{\mathcal{V}}_k^{(j)}\left(v_k^{(j)}(\boldsymbol{z}_2,\boldsymbol{u}_2;\widetilde{\delta}_k^{(j)})\right)\right\|_2 \\
&\leq \left\| v_k^{(j)}(\boldsymbol{z}_1,\boldsymbol{u}_1;\delta_k^{(j)}) - v_k^{(j)}(\boldsymbol{z}_2,\boldsymbol{u}_2;\widetilde{\delta}_k^{(j)})\right\|_2 \\
&\quad + \left\| \mathcal{V}_k^{(j)}\left(v_k^{(j)}(\boldsymbol{z}_1,\boldsymbol{u}_1;\delta_k^{(j)})\right) - \widetilde{\mathcal{V}}_k^{(j)}\left(v_k^{(j)}(\boldsymbol{z}_2,\boldsymbol{u}_2;\widetilde{\delta}_k^{(j)})\right)\right\|_2 . \qquad (4.51)
\end{aligned}$$

Next, as $W_{k,d}^{(j)}, \widetilde{W}_{k,d}^{(j)} \in \Omega_d$, for $d \in \mathbb{N}[L_c]$ then $W_{k,d}^{(j)}$, and $\widetilde{W}_{k,d}^{(j)}$ are defined by $f_{d-1}f_d k_d^2$ parameters and satisfy $\|W_{k,d}^{(j)}\|_2, \|\widetilde{W}_{k,d}^{(j)}\|_2 \leq w_d$. Additionally, $\delta_k^{(j)} \in \Omega_{L_c+1}$ is a positive real number satisfying $|\delta_k^{(j)}| \leq \delta$. Hence, the dimension of the parameter spaces α_d and bounds on the parameter spaces ω_d are given by

$$(\alpha_d,\omega_d) = \begin{cases} (f_{d-1}f_d k_d^2, w_d) & d = 1,2,\ldots,L_c \\ (1,\delta) & d = L_c+1. \end{cases}$$

Additionally, note that every Tikhonov solution covariance matrix has tridiagonal structure and $\sqrt{\dim(\mathcal{P}_{\text{tri}})} = \sqrt{2n-1}$.

Now, as $\|\rho_\xi(\boldsymbol{x})\|_2 \leq \xi$ for all $\boldsymbol{x} \in \mathbb{R}^n$ and $\|\boldsymbol{z}\|_1 \leq z_\infty$ then, for any measurement $\overline{\boldsymbol{y}}_j$, by the triangle inequality

$$\begin{aligned}
\|v_k^{(j)}(\boldsymbol{z}_1,\boldsymbol{u}_1;\delta_k^{(j)})\|_2 &= \|\boldsymbol{z}_1 - \delta_k^{(j)}\rho_\xi(A_{\boldsymbol{u}_1}^T(A_{\boldsymbol{u}_1}\boldsymbol{z}_1 - \overline{\boldsymbol{y}}_j))\|_2 \\
&\leq \|\boldsymbol{z}_1\|_2 + \|\delta_k^{(j)}\rho_\xi(A_{\boldsymbol{u}_1}^T(A_{\boldsymbol{u}_1}\boldsymbol{z}_1 - \overline{\boldsymbol{y}}_j))\|_2 \\
&\leq \sqrt{n}z_\infty + \delta\xi.
\end{aligned}$$

Hence, as the ReLU activation function is 1-Lipschitz then using Lemma 4.6.2

$$\begin{aligned}
&\left\|\mathcal{V}_k^{(j)}\left(v_k^{(j)}(\boldsymbol{z}_1,\boldsymbol{u}_1;\delta_k^{(j)})\right)-\widetilde{\mathcal{V}}_k^{(j)}\left(v_k^{(j)}(\boldsymbol{z}_2,\boldsymbol{u}_2;\widetilde{\delta}_k^{(j)})\right)\right\|_2\\
&\quad\le\prod_{\ell=1}^{L_c}w_\ell\left\|v_k^{(j)}(\boldsymbol{z}_1,\boldsymbol{u}_1;\delta_k^{(j)})-v_k^{(j)}(\boldsymbol{z}_2,\boldsymbol{u}_2;\widetilde{\delta}_k^{(j)})\right\|_2\\
&\qquad+\sum_{\ell=1}^{L_c}\left(\|v_k^{(j)}(\boldsymbol{z}_1,\boldsymbol{u}_1;\delta_k^{(j)})\|_2\prod_{\substack{\ell'=1\\\ell'\neq\ell}}^{L_c}w_{\ell'}\right)\left\|W_{k,\ell}^{(j)}-\widetilde{W}_{k,\ell}^{(j)}\right\|_2\\
&\quad\le\prod_{\ell=1}^{L_c}w_\ell\left\|v_k^{(j)}(\boldsymbol{z}_1,\boldsymbol{u}_1;\delta_k^{(j)})-v_k^{(j)}(\boldsymbol{z}_2,\boldsymbol{u}_2;\widetilde{\delta}_k^{(j)})\right\|_2\\
&\qquad+\left(\sqrt{n}z_\infty+\delta\xi\right)\sum_{\ell=1}^{L_c}\prod_{\substack{\ell'=1\\\ell'\neq\ell}}^{L_c}w_{\ell'}\left\|W_{k,\ell}^{(j)}-\widetilde{W}_{k,\ell}^{(j)}\right\|_2.
\end{aligned}\tag{4.52}$$

By equation (4.48) it holds that

$$\begin{aligned}
&\|v_k^{(j)}(\boldsymbol{z}_1,\boldsymbol{u}_1;\delta_k^{(j)})-v_k^{(j)}(\boldsymbol{z}_2,\boldsymbol{u}_2;\widetilde{\delta}_k^{(j)})\|_2\\
&\quad=\left(1+\delta(z_\infty p_{\max}y_{\max}\|A\|_2\|A\|_\infty)^2\right)\|\boldsymbol{z}_1-\boldsymbol{z}_2\|_2\\
&\qquad+\delta y_{\max}\|A\|_2\left(1+z_\infty^2p_{\max}\|A\|_2(\|A\|_2+\|A\|_\infty)\right)\|\boldsymbol{u}_1-\boldsymbol{u}_2\|_2+\xi|\delta_k^{(j)}-\widetilde{\delta}_k^{(j)}|.
\end{aligned}$$

Combining equation (4.48), equation (4.52), and equation (4.51) produces

$$
\begin{aligned}
&\left\| g_k^{(j)}\left(\boldsymbol{z}_1, \boldsymbol{u}_1; \{\delta_k^{(j)}, W_{k,\ell}^{(j)}\}_{\ell\in\mathbb{N}[L_c]}\right) - g_k^{(j)}\left(\boldsymbol{z}_2, \boldsymbol{u}_2; \{\widetilde{\delta}_k^{(j)}, \widetilde{W}_{k,\ell}^{(j)}\}_{\ell\in\mathbb{N}[L_c]}\right)\right\|_2 \\
&\quad\leq \left\| v_k^{(j)}(\boldsymbol{z}_1, \boldsymbol{u}_1; \delta_k^{(j)}) - v_k^{(j)}(\boldsymbol{z}_2, \boldsymbol{u}_2; \widetilde{\delta}_k^{(j)})\right\|_2 \\
&\qquad + \left\| \mathcal{V}_k^{(j)}\left(v_k^{(j)}(\boldsymbol{z}_1, \boldsymbol{u}_1; \delta_k^{(j)})\right) - \widetilde{\mathcal{V}}_k^{(j)}\left(v_k^{(j)}(\boldsymbol{z}_2, \boldsymbol{u}_2; \widetilde{\delta}_k^{(j)})\right)\right\|_2 \\
&\quad\leq \left(1 + \prod_{\ell=1}^{L_c} w_\ell\right) \left\| v_k^{(j)}(\boldsymbol{z}_1, \boldsymbol{u}_1; \delta_k^{(j)}) - v_k^{(j)}(\boldsymbol{z}_2, \boldsymbol{u}_2; \widetilde{\delta}_k^{(j)})\right\|_2 \\
&\qquad + \left(\sqrt{n} z_\infty + \delta\xi\right) \sum_{\ell=1}^{L_c} \prod_{\substack{\ell'=1 \\ \ell'\neq\ell}}^{L_c} w_{\ell'} \left\| W_{k,\ell}^{(j)} - \widetilde{W}_{k,\ell}^{(j)}\right\|_2 \\
&\quad\leq \left(1 + \prod_{\ell=1}^{L_c} w_\ell\right) \left(1 + \delta(z_\infty p_{\max} y_{\max} \|A\|_2 \|A\|_\infty)^2\right) \|\boldsymbol{z}_1 - \boldsymbol{z}_2\|_2 \\
&\qquad + \left(1 + \prod_{\ell=1}^{L_c} w_\ell\right) \delta y_{\max} \|A\|_2 \left(1 + z_\infty^2 p_{\max} \|A\|_2 (\|A\|_2 + \|A\|_\infty)\right) \|\boldsymbol{u}_1 - \boldsymbol{u}_2\|_2 \\
&\qquad + \left(1 + \prod_{\ell=1}^{L_c} w_\ell\right) \xi |\delta_k^{(j)} - \widetilde{\delta}_k^{(j)}| + \left(\sqrt{n} z_\infty + \delta\xi\right) \sum_{\ell=1}^{L_c} \prod_{\substack{\ell'=1 \\ \ell'\neq\ell}}^{L_c} w_{\ell'} \left\| W_{k,\ell}^{(j)} - \widetilde{W}_{k,\ell}^{(j)}\right\|_2.
\end{aligned}
$$

Therefore, Assumption 4.4.3 holds with $r_{k,1}^{(j-1)} = r_1$, $r_{k,2}^{(j-1)} = r_2$, and

$$
(\boldsymbol{\theta}_{k,d}^{(j)}, r_{k,d,3}^{(j-1)}, \|\cdot\|_{(d)}) = \begin{cases} \left(W_{k,d}^{(j)}, \left(\sqrt{n} z_\infty + \delta\xi\right) \prod_{\substack{\ell=1 \\ \ell\neq d}}^{L_c} w_\ell, \|\cdot\|_2\right) & d = 1, 2, \ldots, L_c \\ \left(\delta_k^{(j)}, \left(1 + \prod_{\ell=1}^{L_c} w_\ell\right) \xi, |\cdot|\right) & d = L_c + 1. \end{cases}
$$

Hence, the constants $\widehat{r}_{k,1}^{(J)}, \widehat{r}_{k,2}^{(J)}$, and $\widehat{r}_{k,d,3}^{(j,J)}$ from Proposition 4.4.6 are given as

$$
\begin{aligned}
\widehat{r}_{k,1}^{(J)} &= \prod_{j=1}^{J} r_{k,1}^{(j-1)} = r_1^J \\
\widehat{r}_{k,2}^{(J)} &= \sum_{j=1}^{J} r_{k,2}^{(j-1)} \prod_{\ell=j}^{J-1} r_{k,1}^{(\ell)} = r_2 \sum_{j=1}^{J} r_1^{J-j} = r_2 \frac{1 - r_1^J}{1 - r_1},
\end{aligned}
$$

and

$$\widehat{r}_{k,d,3}^{(j,J)} = r_{k,d,3}^{(j-1)} \prod_{\ell=j}^{J-1} r_{k,1}^{(\ell)} = r_{k,d,3}^{(j-1)} r_1^{J-j} = r_1^{J-j} \begin{cases} (\sqrt{n} z_\infty + \delta\xi) \prod_{\substack{\ell=1 \\ \ell \neq d}}^{L_c} w_\ell & d = 1, 2, \ldots, L_c \\ \left(1 + \prod_{\ell=1}^{L_c} w_\ell\right) \xi & d = L_c + 1. \end{cases}$$

Thus, the constants $\widehat{c}_1^{(K,J)}$ and $\widehat{c}_{k,j,d,2}^{(K,J)}$ from Theorem 4.4.15 are given as

$$\begin{aligned} \widehat{c}_1^{(K,J)} &= c_2 \sum_{k=1}^{K} \widehat{r}_{k,2}^{(J)} \prod_{\ell=k+1}^{K} (\widehat{r}_{\ell,1}^{(J)} + \widehat{r}_{\ell,2}^{(J)} c_1) \\ &= c_2 \sum_{k=1}^{K} r_2 \frac{1 - r_1^J}{1 - r_1} \prod_{\ell=k+1}^{K} \left(r_1^J + r_2 \frac{1 - r_1^J}{1 - r_1} c_1 \right) \\ &= c_2 r_2 \frac{1 - r_1^J}{1 - r_1} \sum_{k=1}^{K} \left(r_1^J + r_2 \frac{1 - r_1^J}{1 - r_1} c_1 \right)^{K-k} \\ &= c_2 r_2 \frac{1 - r_1^J}{1 - r_1} \sum_{k=0}^{K-1} \left(r_1^J + r_2 \frac{1 - r_1^J}{1 - r_1} c_1 \right)^{k} \\ &= c_2 r_2 \left(\frac{1 - r_1^J}{1 - r_1} \right) \left(\frac{1 - \left(r_1^J + r_2 \frac{1 - r_1^J}{1 - r_1} c_1 \right)^K}{1 - \left(r_1^J + r_2 \frac{1 - r_1^J}{1 - r_1} c_1 \right)} \right) \end{aligned}$$

and

$$\begin{aligned} \widehat{c}_{k,j,d,2}^{(K,J)} &= \widehat{r}_{k,d,3}^{(j,J)} \prod_{\ell=k+1}^{K} (\widehat{r}_{\ell,1}^{(J)} + \widehat{r}_{\ell,2}^{(J)} c_1) \\ &= \widehat{r}_{k,d,3}^{(j,J)} \prod_{\ell=k+1}^{K} \left(r_1^J + r_2 \frac{1 - r_1^J}{1 - r_1} c_1 \right) \\ &= \widehat{r}_{k,d,3}^{(j,J)} \left(r_1^J + r_2 \frac{1 - r_1^J}{1 - r_1} c_1 \right)^{K-k} \\ &= r_1^{J-j} \left(r_1^J + r_2 \frac{1 - r_1^J}{1 - r_1} c_1 \right)^{K-k} \begin{cases} (\sqrt{n} z_\infty + \delta\xi) \prod_{\substack{\ell=1 \\ \ell \neq d}}^{L_c} w_\ell & d = 1, 2, \ldots, L_c \\ \left(1 + \prod_{\ell=1}^{L_c} w_\ell\right) \xi & d = L_c + 1 \end{cases} \end{aligned}$$

for

$$c_1 = p_{\max} y_{\max} \, \|A\|_2 \left(1 + 2z_\infty^2 p_{\max} \, \|A\|_2^2\right)$$
$$c_2 = z_\infty y_{\max} \|A\|_2 \left(p_{\max}/p_{\min}\right)^2 .$$

Lastly, applying Theorem 4.4.15 produces the desired generalization error bound for ISTA DR-CG-Net. ⊠

4.6.5 Asymptotic Forms of ISTA DR-CG-Net Generalization Error Bounds

In this section we provide asymptotic forms for the ISTA DR-CG-Net generalization error bound given in Theorem 4.6.7. We consider asymptotic bounds as the size of the network – i.e. K and J, the number of unrolled iterations of CG-LS – grows, as the dimension of the measurements and reconstructed signal – i.e. m and n, respectively – grow, and as the number of training data samples – i.e. N_s – grows.

Recall the values $c_{\max}, p_{\max}, p_{\min}, f_d, k_d, w_d, \delta, z_\infty, y_{\max}, \|A\|_2, \|A\|_\infty$, and ξ that are utilized by Theorem 4.6.7. As $p_{\max}, p_{\min}, f_d, k_d, w_d, \delta, z_\infty$, and ξ are all parameters chosen and fixed during the construction of ISTA DR-CG-Net before training, we assume that each is independent of network size, measurement dimension, reconstructed signal dimension, and training dataset size. Thus, we establish asymptotic forms for the ISTA DR-CG-Net generalization error that rely on whether $c_{\max}, y_{\max}, \|A\|_2$, and $\|A\|_\infty$ depend on the measurement and reconstructed signal dimension.

Finally, recall that we use the notation $a \lesssim b$ to denote that $a \leq s_c b$ for some constant $s_c > 0$; that is, $a \lesssim b$ implies $a = \mathcal{O}(b)$.

Dependent on Signal Dimension

For the following asymptotic generalization error bound of ISTA DR-CG-Net, we assume that each of $c_{\max}, y_{\max}, \|A\|_2$, and $\|A\|_\infty$ depend on the measurement and reconstructed signal dimension in that $\lim_{m,n\to\infty} c_{\max} \to \infty$, $\lim_{m,n\to\infty} y_{\max} \to \infty$, and $\lim_{m,n\to\infty} \|A\|_p \to \infty$ for any $1 \leq p \leq \infty$. Note for any $\boldsymbol{x} \in \mathbb{R}^k$ that $\|\boldsymbol{x}\|_2 \leq \sqrt{k}\|\boldsymbol{x}\|_\infty$. As $c_{\max}$ and $y_{\max}$ are, respectively, the

maximum Euclidean norm bounds on the signals of interest, of dimension n, and measurements, of dimension m, then $c_{\max} \lesssim \sqrt{n}$ and $y_{\max} \lesssim \sqrt{m}$; where we have reasonably assumed that the maximum entry in any measurement or reconstructed signal is independent of the signal dimension. Furthermore, as $\|A\|_2 \leq \sqrt{m}\|A\|_\infty$ and $\|A\|_\infty \leq n\left(\max_{i,j}|A_{i,j}|\right)$, since $\|A\|_\infty$ is the max absolute row sum, then $\|A\|_2 \lesssim n\sqrt{m}$ and $\|A\|_\infty \leq n$; where again we have reasonably assumed that the maximum entry of the measurement matrix is independent of the matrix dimensions.

Corollary 4.6.8. *Let $c_{\max}, y_{\max}, \|A\|_2$, and $\|A\|_\infty$ depend on the measurement and reconstructed signal dimension. The generalization error for ISTA DR-CG-Net scales as*

$$|\mathcal{L}(\widehat{\boldsymbol{c}}) - \mathcal{L}_{\mathcal{S}}(\widehat{\boldsymbol{c}})| \lesssim c_{\max}\left(\sqrt{n} + KJ\left(1 + \sum_{d=1}^{L_c}\sqrt{f_{d-1}f_d k_d^2}\right)\right)\sqrt{\frac{\mathcal{F}(K,J,L_c)}{nN_s}} + c_{\max}KJ\left(\sum_{d=1}^{L_c}\sqrt{f_{d-1}f_d k_d^2}\right)\sqrt{\frac{\ln(n)}{nN_s}}$$

for

$$\mathcal{F}(K,J,L_c) = \ln(KJL_c) + KJ\left(\ln(y_{\max}) + \ln(\|A\|_2) + \ln(\|A\|_\infty) + \ln\left(1 + \prod_{\ell=1}^{L_c} w_\ell\right)\right).$$

Furthermore, as, at most, $c_{\max} = \mathcal{O}(\sqrt{n})$, $y_{\max} = \mathcal{O}(\sqrt{m})$, $\|A\|_2 = \mathcal{O}(n\sqrt{m})$, and $\|A\|_\infty = \mathcal{O}(n)$ then generalization error for ISTA DR-CG-Net scales at most as

$$|\mathcal{L}(\widehat{\boldsymbol{c}}) - \mathcal{L}_{\mathcal{S}}(\widehat{\boldsymbol{c}})| \lesssim \left(\sqrt{n} + KJL_c\right)\sqrt{\frac{KJ\left(\ln(m) + \ln(n) + L_c\right)}{N_s}}.$$

A proof of Corollary 4.6.8 is provided in Appendix D and results from ignoring constants to consider how the GEB from Theorem 4.6.7 scales in network size and signal dimension. From Corollary 4.6.8, once the amount of training data satisfies

$$N_s \sim \frac{c_{\max}^2\left[\left(\sqrt{n} + KJ\left(1 + \sum_{d=1}^{L_c}\sqrt{f_{d-1}f_d k_d^2}\right)\right)\sqrt{\mathcal{F}(K,J,L_c)} + KJ\left(\sum_{d=1}^{L_c}\sqrt{f_{d-1}f_d k_d^2}\right)\sqrt{\ln(n)}\right]^2}{n}$$

or

$$N_s \sim \left(\sqrt{n} + KJL_c\right)^2 KJ\left(\ln(m) + \ln(n) + L_c\right)$$

then the GEB of ISTA DR-CG-Net is guaranteed to be small with high probability. Furthermore, Corollary 4.6.8 shows that, at worst, the ISTA DR-CG-Net generalization error grows as $\mathcal{O}(\sqrt{n\ln(n)})$ in reconstructed signal dimension, $\mathcal{O}(\sqrt{(KJL_c)^3}) = \mathcal{O}((\text{Network Size})^{3/2})$ in network size, $\mathcal{O}(\sqrt{\ln(m)})$ in measurement dimension, and $\mathcal{O}(N_s^{-1/2})$ in training dataset size.

Independent Measurement Matrix Norm

Now we assume that $\|A\|_2$ and $\|A\|_\infty$ are independent of m and n defining the dimension of A. For instance, the measurement matrix could be normalized through preprocessing such that $\|A\|_2 = 1$.

Corollary 4.6.9. *Let $c_{\max}$ and $y_{\max}$ depend on the measurement and reconstructed signal dimension. Furthermore, assume $\|A\|_2$ and $\|A\|_\infty$ are independent of the measurement and reconstructed signal dimension. The generalization error for ISTA DR-CG-Net scales as*

$$\begin{aligned}|\mathcal{L}(\widehat{\boldsymbol{c}}) - \mathcal{L}_{\mathcal{S}}(\widehat{\boldsymbol{c}})| \lesssim\ & c_{\max}\left(\sqrt{n} + KJ\left(1 + \sum_{d=1}^{L_c}\sqrt{f_{d-1}f_d k_d^2}\right)\right)\sqrt{\frac{\mathcal{F}(K,J,L_c)}{nN_s}} \\ & + c_{\max}KJ\left(\sum_{d=1}^{L_c}\sqrt{f_{d-1}f_d k_d^2}\right)\sqrt{\frac{\ln(n)}{nN_s}}\end{aligned}$$

for

$$\mathcal{F}(K,J,L_c) = \ln(KJL_c) + KJ\left(\ln(y_{\max}) + \ln\left(1 + \prod_{\ell=1}^{L_c} w_\ell\right)\right).$$

Furthermore, as, at most, $c_{\max} = \mathcal{O}(\sqrt{n})$ *and* $y_{\max} = \mathcal{O}(\sqrt{m})$ *then generalization error for ISTA DR-CG-Net scales at most as*

$$|\mathcal{L}(\widehat{\boldsymbol{c}}) - \mathcal{L}_{\mathcal{S}}(\widehat{\boldsymbol{c}})| \lesssim \left(\sqrt{n} + KJL_c\right) \sqrt{\frac{KJ\left(\ln(m) + L_c\right)}{N_s}} + KJL_c \sqrt{\frac{\ln(n)}{N_s}}.$$

Corollary 4.6.9 is a direct consequence of Corollary 4.6.8 where the $\|A\|_p$ terms, for $p = 2, \infty$, are removed as they are now of order $\mathcal{O}(1)$. Furthermore, as in Corollary 4.6.8, Corollary 4.6.9 results from ignoring constants to consider how the GEB from Theorem 4.6.7 scales in network size and signal dimension. From Corollary 4.6.9, once the amount of training data satisfies

$$N_s \sim \frac{c_{\max}^2 \left[\left(\sqrt{n} + KJ\left(1 + \sum_{d=1}^{L_c} \sqrt{f_{d-1} f_d k_d^2}\right)\right) \sqrt{\mathcal{F}(K, J, L_c)} + KJ\left(\sum_{d=1}^{L_c} \sqrt{f_{d-1} f_d k_d^2}\right) \sqrt{\ln(n)}\right]^2}{n}$$

or

$$N_s \sim \left[\left(\sqrt{n} + KJL_c\right) \sqrt{KJ\left(\ln(m) + L_c\right)} + KJL_c \sqrt{\ln(n)}\right]^2$$

then the GEB of ISTA DR-CG-Net is guaranteed to be small with high probability. Furthermore, Corollary 4.6.9 shows that, at worst, the ISTA DR-CG-Net generalization error grows as $\mathcal{O}(\sqrt{(KJL_c)^3}) = \mathcal{O}((\text{Network Size})^{3/2})$ in network size, $\mathcal{O}(\sqrt{n})$ in reconstructed signal dimension, $\mathcal{O}(\sqrt{\ln(m)})$ in measurement dimension, and $\mathcal{O}(N_s^{-1/2})$ in training dataset size.

Independence of Signal Dimension

Here, we assume that $c_{\max}$, $y_{\max}$, $\|A\|_2$, and $\|A\|_\infty$ are independent of m and n defining the measurement and reconstructed signal dimension. As an example, CG-Net and DR-CG-Net from Chapter 2 and Chapter 3 implement preprocessing to scale all images of interest to be in the unit ball. That is, $c_{\max} \leq 1$. Additionally, if the measurement matrix is normalized through preprocessing such that $\|A\|_2 = 1$ then $y_{\max} \leq \|A\|_2 c_{\max} \leq 1$.

Corollary 4.6.10. *Let $c_{\max}$, $y_{\max}$, $\|A\|_2$, and $\|A\|_\infty$ are independent of the measurement and reconstructed signal dimension. The generalization error for ISTA DR-CG-Net scales as*

$$|\mathcal{L}(\widehat{\boldsymbol{c}}) - \mathcal{L}_{\mathcal{S}}(\widehat{\boldsymbol{c}})| \lesssim \left(\sqrt{n} + KJ\left(1 + \sum_{d=1}^{L_c}\sqrt{f_{d-1}f_d k_d^2}\right)\right)\sqrt{\frac{\ln(KJL_c) + KJ\ln\left(1+\prod_{\ell=1}^{L_c} w_\ell\right)}{nN_s}} + KJ\left(\sum_{d=1}^{L_c}\sqrt{f_{d-1}f_d k_d^2}\right)\sqrt{\frac{\ln(n)}{nN_s}}.$$

Furthermore, the generalization error for ISTA DR-CG-Net scales at most as

$$|\mathcal{L}(\widehat{\boldsymbol{c}}) - \mathcal{L}_{\mathcal{S}}(\widehat{\boldsymbol{c}})| \lesssim \left(\sqrt{n} + KJL_c\right)\sqrt{\frac{KJL_c}{nN_s}} \lesssim \sqrt{\frac{(KJL_c)^3}{N_s}}.$$

Corollary 4.6.10 is a direct consequence of Corollary 4.6.8 where the $\|A\|_p$ terms, for $p = 2, \infty$, and $c_{\max}$ and $y_{\max}$ terms are removed as they are now of order $\mathcal{O}(1)$. Furthermore, as in Corollary 4.6.8, Corollary 4.6.10 results from ignoring constants to consider how the GEB from Theorem 4.6.7 scales in network size and signal dimension. From Corollary 4.6.10, once the amount of training data satisfies

$$N_s \sim \frac{c_{\max}^2\left[\left(\sqrt{n} + KJ\left(1+\sum_{d=1}^{L_c}\sqrt{f_{d-1}f_d k_d^2}\right)\right)\sqrt{\mathcal{F}(K,J,L_c)} + KJ\left(\sum_{d=1}^{L_c}\sqrt{f_{d-1}f_d k_d^2}\right)\sqrt{\ln(n)}\right]^2}{n}$$

for, $\mathcal{F}(K,J,L_c) = \ln(KJL_c) + KJ\ln\left(1+\prod_{\ell=1}^{L_c} w_\ell\right)$, or

$$N_s \sim (KJL_c)^3$$

then the GEB of ISTA DR-CG-Net is guaranteed to be small with high probability. Finally, Corollary 4.6.9 shows that, at worst, the ISTA DR-CG-Net generalization error grows in network size as $\mathcal{O}(\sqrt{(KJL_c)^3}) = \mathcal{O}((\text{Network Size})^{3/2})$ and in training dataset size as $\mathcal{O}(N_s^{-1/2})$.

4.7 Generalization Error Bound Comparison

It is readily observed in Section 4.6.3 and Section 4.6.5 that PGD DR-CG-Net and ISTA DR-CG-Net have equivalent asymptotic generalization error bound forms. This is supported by the numerical results of Chapter 3 and the numerical results of [77], which show that PGD DR-CG-Net and ISTA DR-CG-Net have nearly equal performance on test datasets after training. As PGD DR-CG-Net and ISTA DR-CG-Net have equivalent generalization error bounds, we simply refer to DR-CG-Net as a whole for the remainder of this section instead of specifying which scale variable update method is used.

We remark that the network size of CG-Net is roughly equal to KJ whereas the network size of DR-CG-Net is roughly equal to KJL_c. Next, we compare the asymptotic generalization error bound forms of CG-Net and DR-CG-Net under the constraint that both networks have equal network size. For the comparison we consider the following three cases: (1) $c_{\max}, y_{\max}, \|A\|_2$, and $\|A\|_\infty$ depend on the signal dimension, (2) $c_{\max}$ and $y_{\max}$ and depend on the signal dimension while $\|A\|_2$ and $\|A\|_\infty$ are independent of the signal dimension, and (3) $c_{\max}, y_{\max}, \|A\|_2$, and $\|A\|_\infty$ are all independent of the signal dimension.

Dependent on Signal Dimension

Here we assume that each of $c_{\max}, y_{\max}, \|A\|_2$, and $\|A\|_\infty$ depend on the measurement and reconstructed signal dimension in that $c_{\max} = \mathcal{O}(\sqrt{n})$, $y_{\max} = \mathcal{O}(\sqrt{m})$, $\|A\|_2 = \mathcal{O}(n\sqrt{m})$, and $\|A\|_\infty = \mathcal{O}(n)$. By Corollary 4.5.2 and Corollary 4.6.8 the GEB of CG-Net and DR-CG-Net scale at most as

$$|\mathcal{L}(\widehat{\boldsymbol{c}}) - \mathcal{L}_{\mathcal{S}}(\widehat{\boldsymbol{c}})| \lesssim \begin{cases} \sqrt{\frac{n^3(\ln(m)+\ln(n))(\text{Network Size})^3}{N_s}} & \text{CG-Net} \\ (\sqrt{n} + \text{Network Size})\sqrt{\frac{(\text{Network Size})(\ln(m)+\ln(n))}{N_s}} & \text{DR-CG-Net.} \end{cases}$$

We are able to further upper bound the DR-CG-Net GEB as

$$|\mathcal{L}(\widehat{\boldsymbol{c}}) - \mathcal{L}_{\mathcal{S}}(\widehat{\boldsymbol{c}})| \lesssim \begin{cases} \sqrt{\frac{n^3(\ln(m)+\ln(n))(\text{Network Size})^3}{N_s}} & \text{CG-Net} \\ \sqrt{\frac{n(\text{Network Size})^3(\ln(m)+\ln(n))}{N_s}} & \text{DR-CG-Net.} \end{cases}$$

Therefore, the ratio of the CG-Net and DR-CG-Net GEB asymptotic forms is

$$\frac{\text{GEB CG-Net}}{\text{GEB DR-CG-Net}} = n.$$

That is, the GEB for CG-Net grows to an extra order of the reconstructed signal dimension as compared to the GEB for DR-CG-Net. Hence, DR-CG-Net produces a tighter GEB, which is supported by the numerical experiments of Chapter 2 and Chapter 3 – and additionally the numerical experiments of [58, 77] – showing that DR-CG-Net produces improved test reconstructions after training as compared to CG-Net.

Independent Measurement Matrix Norm

Here we assume that $c_{\max}$ and $y_{\max}$ depend on the measurement and reconstructed signal dimension in that $y_{\max} = \mathcal{O}(\sqrt{m})$, and $c_{\max} = \mathcal{O}(\sqrt{n})$. Furthermore, we assume that both $\|A\|_2$ and $\|A\|_\infty$ are independent of the measurement and reconstructed signal dimension. By Corollary 4.5.3 and Corollary 4.6.9 the GEB of CG-Net and DR-CG-Net scale at most as

$$|\mathcal{L}(\widehat{\boldsymbol{c}}) - \mathcal{L}_{\mathcal{S}}(\widehat{\boldsymbol{c}})| \lesssim \begin{cases} \sqrt{\frac{n^3 \ln(m)(\text{Network Size})^3}{N_s}} & \text{CG-Net} \\ (\sqrt{n} + \text{Network Size})\sqrt{\frac{(\text{Network Size})\ln(m)}{N_s}} + \sqrt{\frac{(\text{Network Size})^2 \ln(n)}{N_s}} & \text{DR-CG-Net.} \end{cases}$$

We are able to further upper bound the DR-CG-Net GEB as

$$|\mathcal{L}(\widehat{\boldsymbol{c}}) - \mathcal{L}_{\mathcal{S}}(\widehat{\boldsymbol{c}})| \lesssim \begin{cases} \sqrt{\frac{n^3 \ln(m)(\text{Network Size})^3}{N_s}} & \text{CG-Net} \\ \sqrt{\frac{n(\text{Network Size})^3 \ln(m)}{N_s}} & \text{DR-CG-Net.} \end{cases}$$

Therefore, the ratio of the CG-Net and DR-CG-Net GEB asymptotic forms is

$$\frac{\text{GEB CG-Net}}{\text{GEB DR-CG-Net}} = n.$$

Again, the GEB for CG-Net grows with an extra order of the reconstructed signal dimension as compared to the GEB for DR-CG-Net. So, DR-CG-Net produces a tighter GEB which is supported by the numerical experiments of [58, 77], Chapter 2, and Chapter 3 that show DR-CG-Net producing improved test reconstructions after training as compared to CG-Net.

Independence of Signal Dimension

Here we assume that $c_{\max}$, $y_{\max}$, $\|A\|_2$, and $\|A\|_\infty$ are all independent of the measurement and reconstructed signal dimension. By Corollary 4.5.4 and Corollary 4.6.10 the GEB of CG-Net and DR-CG-Net scale at most as

$$|\mathcal{L}(\widehat{\boldsymbol{c}}) - \mathcal{L}_{\mathcal{S}}(\widehat{\boldsymbol{c}})| \lesssim \begin{cases} \sqrt{\frac{n^2(\text{Network Size})^3}{N_s}} & \text{CG-Net} \\ \sqrt{\frac{(\text{Network Size})^3}{N_s}} & \text{DR-CG-Net.} \end{cases}$$

Therefore, the ratio of the CG-Net and DR-CG-Net GEB asymptotic forms is

$$\frac{\text{GEB CG-Net}}{\text{GEB DR-CG-Net}} = n.$$

Similarly to the previous to cases, the GEB for CG-Net grows to an extra order of the reconstructed signal dimension as compared to the GEB for DR-CG-Net. Therefore, DR-CG-Net produces a tighter GEB as compared to CG-Net, which is supported by the numerical experiments in [58, 77], Chapter 2, and Chapter 3, which show that DR-CG-Net produces improved test reconstructions after training as compared to CG-Net.

4.8 Conclusion

In this chapter, we have derived a generalization error bound for a class of compound-Gaussian-prior-based DNNs that solve linear inverse problems. Subsequently, this generalization error bound was applied to two realizations, namely, CG-Net from Chapter 2 and DR-CG-Net from Chapter 3, showing bounds for these two cases. The developed generalization error bound was produced by bounding the Rademacher complexity of the network hypothesis class by Dudley's inequality, which is further bounded using a Lipschitz property to estimate covering numbers of the network hypothesis class. A key contribution of this chapter was our proof that parameters of compound Gaussian DNNs satisfy a Lipschitz condition under reasonable assumptions, thereby allowing us to produce generalization bounds for CG-based DNNs. Lastly, we derived asymptotic forms for the CG-Net and DR-CG-Net generalization error bounds that show each bound scales $\mathcal{O}((\text{Network Size})^{3/2})$ in the network size, scales $\mathcal{O}(N_s^{-1/2})$ in the training dataset size, and scales $\mathcal{O}(n)$ in signal-of-interest dimension for CG-Net.

Chapter 5

Summary and Future Work

5.1 Summary

5.1.1 Compound Gaussian Network

In Chapter 2, we have proposed, theoretically studied, and empirically validated two new methods for solving linear inverse problems. Both methods have been published in our conference proceedings of the IEEE Asia Pacific Signal and Information Processing Association conference [57] and in our IEEE Transactions on Signal Processing journal article [58]. Furthermore, a provisional patent of our two innovative methods has been filed.

Our goal was to develop a new method that approached inverse problems from the perspective of interweaving the powerful compound Gaussian (CG) class of densities with the powerful representational abilities of deep neural networks (DNNs). The choice of these two components was inspired by prior work showing that CG densities better capture statistical properties of natural images and images from other modalities such as radar [10–12, 44, 95] and by prior work showing that DNNs outperform standard iterative reconstruction methods to inverse problems [15–26,28–33,41,42,45,55–58,76,77,87,96–104]. To accomplish our goal, we employed the algorithm unrolling technique to create a DNN from an iterative signal reconstruction algorithm. Therefore, the first of the two new methods proposed in Chapter 2 is a CG-informed iterative reconstruction algorithm and the second new method is an unrolled DNN version of the iterative algorithm that inherently incorporates a CG prior.

The novel CG-informed iterative reconstruction algorithm we have developed in Chapter 2 is named compound Gaussian least squares (CG-LS) and is a block coordinate descent method on a regularized least squares cost function. The two-term regularization of the CG-LS cost function consists of a Euclidean norm term, to incorporate the Gaussian component of the CG prior, and

a Euclidean norm composed with an inverse non-linearity, to incorporate a non-linear Gaussian transformation for the scaling, i.e. compounding, in the CG prior.

For the CG-LS cost function, we have provided a theoretical study on the existence and location of stationary points and minimizers. Specifically, using the Poincare-Miranda theorem, we first have derived measurement scaling and choices of the regularization scaling parameters to guarantee the existence of a stationary point where the scale variable lies within a compact hypercube and the optimal Gaussian variable is given by a Tikhonov solution. Second, we have derived measurement scaling and choices of the regularization scaling parameters to guarantee that the previous stationary point is an isolated minimizer of the CG-LS cost function.

For the CG-LS algorithm, we first have proven a lower bound on the change in the CG-LS cost function over a single steepest descent update of the scale variable. Second, with the derived lower bound, we have proven that the sequence of estimates produced by CG-LS converges linearly to a stationary point of zero gradient of the CG-LS cost function. Lastly, under the assumption of proper measurement scaling and choices of the regularization scaling parameters in the CG-LS cost function, we have proven that the CG-LS cost function has an isolated minimizer to which CG-LS will converge linearly given a sufficiently close initialization.

Our novel CG-LS algorithm was empirically compared against six state-of-the-art iterative reconstruction algorithms in tomographic imaging and compressive sensing problems. In the tomographic imaging tests, images were estimated from Radon transform measurements taken at a varying number of uniformly spaced angles. For each number of angles in the Radon transform, our CG-LS algorithm outperformed all six comparison methods in reconstructed image quality as measured by SSIM, PSNR, and visual inspection. In the compressive sensing tests, images were estimated from random Gaussian measurements taken at a sampling ratio of $50\%, 30\%$, and 10%. Again our CG-LS algorithm outperformed all six comparison methods appreciably in estimated image quality. Despite the excellent estimated image quality produced by CG-LS, it was observed in Chapter 2 that the CG-LS performance comes at the cost of a significant computational complexity as compared to the other six iterative algorithms. Finally, we discussed a connection

between the CG-LS estimator and a standard Tikhonov estimator. Specifically, we showed that the Tikhonov estimator is a special case of the CG-LS estimator. Inasmuch, the empirical evaluation we presented has highlighted the benefits of improved estimated signal quality produced by CG-LS over a standard Tikhonov estimate.

With the theoretical guarantees and empirical success of CG-LS, we have transformed CG-LS into a DNN through the algorithm unrolling technique. This DNN, which we name compound Gaussian network (CG-Net), is inherently informed by a CG prior through the fundamental basis of the CG prior within the CG-LS algorithm. Furthermore, our implementation of CG-Net allows the DNN to optimally learn the geometry of the CG-LS optimization landscape for efficient traversal. As such, we have empirically shown that fewer iterations are required in the unrolled CG-Net than are necessary in CG-LS. Notably, this permits CG-Net to reconstruct signals roughly 100 times faster than CG-LS.

Further numerical experimentation has been completed, which compares CG-Net against ten state-of-the-art deep-learning-based methods for linear inverse problems in imaging. In particular, by varying the amount of training data used for inverting a Radon transform, we have shown that CG-Net not only outperforms each comparison method in image reconstruction quality, but does so by a significant margin when a limited amount of training data is available. Additionally, we analyzed the impact of using alternative sparsity transformation bases, i.e. wavelet and discrete cosine transformations, and alternative measurement models, i.e. Radon and random Gaussian transformations, and have shown that CG-Net still outperforms or performs comparably to all ten prior art methods for all alternative measurements and sparsity transformations considered. We have further demonstrated, however, that in the presence of ample measurement noise and with larger amounts of training data, the CG-Net method has faltering performance and may be outperformed by the best-performing prior art method. Finally, we have conducted an ablation study of the CG-Net architecture and shown that removing certain trainable portions of CG-Net is advantageous for extremely small training datasets, but the full CG-Net structure is, in general, the best performing.

On the theoretical front, in Chapter 4 we have developed a generalization error bound for CG-Net. This bound provides that, with sufficient training data, the excellent CG-Net performance on training data is guaranteed to extend to excellent performance of CG-Net on unseen test data. Specifically, under realistic conditions, the test performance of CG-Net is guaranteed when the amount of training data scales quadratically in the signal-of-interest size and cubically in the network size.

5.1.2 Deep Regularized Compound Gaussian Network

In Chapter 3, we have proposed, theoretically studied, and empirically validated two new CG-based methods for solving linear inverse problems that are novel extensions upon the CG-LS and CG-Net structures from Chapter 2. Both methods have been published in our proceedings of the IEEE Asilomar Conference on Signals, Systems, and Computers [76] and in our IEEE Transactions on Computational Imaging journal article [77]. Furthermore, a provisional patent of our two innovative approaches has been filed.

Our goal in developing these two novel extensions was twofold. First, we wanted to alleviate some of the complications and downsides occurring with CG-Net. Specifically, CG-Net was computationally intensive with the impactful calculations occurring within the Tikhonov solution and within a required eigendecomposition calculation for each scale variable update. Furthermore, the high performance of CG-Net falters, comparatively to other state-of-the-art deep learning methods, with high measurement noise and larger amounts of training data. Second, we wanted to incorporate some learning of the prior distribution within the unrolled network as many works with learned regularization have shown superb reconstruction performance in inverse problems [14, 22, 27, 28, 105–122] and since the scale variable distribution is rigidly fixed in CG-Net as log-normal. To accomplish our goal, we first modified the CG-LS cost function and algorithm to accommodate problem specific choices of the scale variable distribution and Gaussian variable covariance. Afterwards, we again have invoked the algorithm unrolling technique to create a DNN

that now permits a learning of the signal-of-interest prior, through a quickly calculable CNN embedded in the DNN, while constraining to the CG class of densities.

The new iterative algorithm we have proposed in Chapter 3, named generalized compound Gaussian least squares (G-CG-LS), is a novel generalization of the CG-LS algorithm from Chapter 2. As with CG-LS, the G-CG-LS algorithm is a block coordinate descent method on a regularized least squares cost function or equivalently is a MAP estimate of the scale and Gaussian variables. In CG-LS, the scale variable regularization is fixed to the Euclidean norm of an inverse non-linearity and corresponds to a log-normal prior distribution. Instead, for G-CG-LS the scale variable regularization is generalized to an implicit function defined on a compact set. Likewise, the Gaussian variable covariance is generalized from a scaled identity matrix in CG-LS, to any symmetric positive definite matrix in G-CG-LS. As opposed to CG-LS, where only a steepest descent approach is used to estimate the scale variable, both a projected gradient descent (PGD) approach and an iterative shrinkage and thresholding algorithm (ISTA) approach have been implemented for estimating the scale variable. The usage of two distinct methods allows G-CG-LS to handle both differentiable and non-smooth scale variable regularization choices.

For the G-CG-LS algorithm, we have leveraged existing literature on the PGD and ISTA approaches to show that, for both scale variable update methods, the sequence of cost function values produces by G-CG-LS converges. With the convergence of the cost function values, we have next proven that, with both scale variable update methods, the G-CG-LS sequence of estimates converges to a closed and connected set of stationary points of the G-CG-LS cost function where the cost function is constant on this set. Lastly, under the condition that the scale variable regularization is chosen such that the G-CG-LS cost function has non-degenerate stationary points, we have proven that the G-CG-LS sequence of estimates converge to a single stationary point of the G-CG-LS cost function.

Next, we have transformed the G-CG-LS algorithm into an unrolled DNN with algorithm unrolling. This DNN, which we name deep regularized compound Gaussian network (DR-CG-Net), learns the scale variable regularization, and thus scale variable distribution, through a CNN sub-

network replacing the regularization gradient in PGD or replacing the proximal operator of the regularization in ISTA. Additionally, the Gaussian variable covariance matrix is trainable by DR-CG-Net. Hence, the new DNN we have proposed learns the prior distribution, by learning the scale variable distribution and Gaussian variable covariance, accomplishing one of our goals. Furthermore, the usage of CNNs in updating the scale variable, which are significantly faster than implementing an eigendecomposition as is done in CG-Net, helps alleviate a portion of the computational burden thereby accomplishing another part of our goals.

We have completed ample numerical experimentation of DR-CG-Net, which compares DR-CG-Net against CG-Net and ten other state-of-the-art deep learning based methods for solving linear inverse problems in imaging. By varying the amount of training data used for inverting Radon transform measurements, we have shown that DR-CG-Net not only retains the impactful property of providing outstanding performance in image reconstruction quality, with a small amount of training data available, but continues to outperform other state-of-the-art methods when larger training datasets are used. This accomplished another portion of our goals. In addition, despite DR-CG-Net having a far greater number of parameters and learning capacity than CG-Net, DR-CG-Net provides a remarkable performance increase over CG-Net in the small training dataset scenarios. Beyond Radon inversion, we have empirically studied the compressive sensing problem of reconstructing a signal from random Gaussian measurements. The results of this study similarly showed that DR-CG-Net provides a significant improvement over CG-Net and all ten comparison methods in image reconstruction quality as measured by SSIM, PSNR, and visual inspection. Finally, larger sized computed tomography scan image reconstructions were performed, where the Tikhonov solution layer in DR-CG-Net was replaced with 100 gradient descent layers accelerated with Nesterov momentum. From these reconstructions, we have visually seen that DR-CG-Net reconstructs all of the fine grain details of the original scan that may be of importance in diagnosing a patient and produces images with higher SSIM and PSNR than all ten comparison methods.

The only goal we did not entirely accomplish with our novel DR-CG-Net modification of CG-Net is the handling of measurement noise. DR-CG-Net suffers from the same complication as

CG-Net where in the presence of higher measurement noise the network performance will falter as compared to the best-performing prior-art method. Although, DR-CG-Net is able to handle a moderate amount of measurement noise, specifically measurements with an SNR of 40dB, while CG-Net only effectively handles low measurement noise, measurements with an SNR of 60dB. As we have stated in Chapter 2 and Chapter 3, the main problem in handling measurement noise may be the Tikhonov solution, which is a structure that is shared between both CG-Net and DR-CG-Net. Likely, the Tikhonov solution is overfitting the measured estimated signal to the original measurement, which is harmful when a lot of noise is present in the measurement. Finally, on the numerical front, we conducted an empirical study on the refinement block added to the end of the DR-CG-Net structure, which highlighted that the main functionality of the refinement block is to assist in the denoising of the estimated image from the unrolled G-CG-LS portion of DR-CG-Net.

On the theoretical front, in Chapter 4 we have developed a generalization error bound for DR-CG-Net. This bound provides that, with sufficient training data, the excellent DR-CG-Net performance on training data is guaranteed to extend to excellent performance of DR-CG-Net on unseen test data. In particular, under realistic conditions, the test reconstruction performance of DR-CG-Net is guaranteed when the amount of training data scales cubically with network size. Therefore, the generalization error bound for DR-CG-Net is tighter than the CG-Net bound, which agrees with all of our empirical study that has shown DR-CG-Net to consistently outperform CG-Net.

5.1.3 Generalization Error for Compound Gaussian Based Networks

In Chapter 4, we developed a general framework for compound-Gaussian-based deep neural networks and studied the theoretical property of generalization error for this framework. Our general CG-based DNN framework and the corresponding generalization error bounds have been documented in a manuscript to be submitted to the IEEE Transactions on Information Theory [123].

The generalized compound Gaussian network (G-CG-Net) framework we developed is a class of deep neural networks for linear inverse problems, which was constructed by applying algo-

rithm unrolling to the G-CG-LS algorithm of Chapter 3. Instead of specifying a projected gradient descent or iterative shrinkage and thresholding algorithm update of the scale variable, as is implemented in DR-CG-Net, we employ a generic and implicit scale-variable descent function defined by a set of the network parameters. Structuring the G-CG-Net framework in this way permits the reduction of G-CG-Net to both CG-Net and DR-CG-Net as special cases when the scale-variable descent function is properly chosen.

Employing tools from statistical learning theory literature, we have derived a generalization error bound for the G-CG-Net framework of DNNs. The developed generalization error bound was produced by bounding the Rademacher complexity of the network hypothesis class by Dudley's inequality, which is further bounded using a Lipschitz property to estimate covering numbers of the network hypothesis class. A key contribution of our work in deriving the generalization error was our proof that parameters of G-CG-Net DNNs satisfy a Lipschitz condition under reasonable assumptions. This Lipschitz property enabled us to construct bounds on Dudley's inequality and thereby produce a generalization error bound.

Finally, we applied the G-CG-Net generalization error bound to the CG-Net and DR-CG-Net realizations, showing bounds for these two cases. For both CG-Net and DR-CG-Net, we derived asymptotic forms of the generalization error bounds under certain combinations of assumptions on the dependence of some network hyperparameters on the reconstructed signal dimension. Under realistic assumptions, we have shown that each bound scales in network size as $\mathcal{O}((\text{Network Size})^{3/2})$, scales in training dataset size as $\mathcal{O}(N_s^{-1/2})$, and, only for CG-Net, scales in reconstructed signal dimension as $\mathcal{O}(n)$.

5.2 Future Work

We conclude this chapter with a discussion on future work for our CG-Net, our DR-CG-Net, theoretical details of compound-Gaussian-prior-based deep neural networks, and additionally propose new methods for compound Gaussian-based generative adversarial networks. Note that, while our CG-LS, CG-Net, G-CG-LS, and DR-CG-Net methods are formulated specifically for solv-

ing linear inverse problems, these techniques could be extended to general bivariate optimization problems. Specifically, the block coordinate technique we employ could similarly be implemented for the development of an iterative bivariate optimization algorithm. Subsequently, an unrolled deep neural network could be constructed from the developed iterative optimization algorithm that implements a similar learning of the optimization landscape, through the learning of an optimal steepest descent step, as in CG-Net or implements a similar learning of the regularization, through an embedded subnetwork, as in DR-CG-Net.

5.2.1 Compound Gaussian Network

From Chapter 2, specifically Table 2.5 and Table 2.13, we empirically demonstrated that the improved reconstructed image quality produced by CG-LS and CG-Net was at a high computational cost. To this point, one direction of future work is to improve the speed of both CG-LS and CG-Net. As discussed in Section 2.2.5, a possibility for increasing the speed of CG-LS is to implement the method in a low-level programming language such as C or C++ rather than in Python and Jupyter Notebooks as we have done. Additionally, as discussed in Section 2.3.6, one option for decreasing the test reconstruction time of CG-Net is to implement the eigendecomposition, from equation (2.32), in each scale variable update to only compute once upon instantiating the CG-Net model with pretrained network parameters. Currently, CG-Net computes the eigendecomposition for every new batch of signals the network processes as this is required during training to actively ensure positive definiteness is maintained while updating the network weights. Since the CG-Net weights are fixed after training, the repetitious eigendecompositions for actively ensuring positive definiteness are no longer necessary.

Analyzing the performance of CG-Net for larger image reconstructions also serves as a key point of future work. In Chapter 2 we presented fundamental development, theory, and results for a CG-inspired iterative reconstruction algorithm and unrolled DNN. As CG-LS continues to outperform competitive iterative methods for reconstructing larger images, we anticipate CG-Net to similarly outperform or perform comparably to competitive methods in low-training scenarios

when applied to larger image reconstructions. For larger images, the training time will be costly, however, so optimization of the CG-Net implementation may be required. Alternatively, a technique, common in compressive sensing applications, of splitting an image into disjoint blocks of small size and then measuring and reconstructing separately [22, 26] could be employed for future experimental evaluation of CG-Net.

As we focused, in Chapter 2, on laying the fundamental groundwork for solving linear inverse problems with a CG prior, further future work could extend CG-LS and CG-Net to non-linear inverse problems where the matrix A is replaced by a non-linear function $\mathcal{F}_A$. Alternatively, a linearization of a non-linear forward operator, which is an adequate approximation for many non-linear inverse problems under suitable conditions (e.g. radar imaging [124]), could be utilized. Therefore, the theoretical and empirical groundwork laid in Chapter 2 serves as the basis for future applications of our CG-based inverse problem methodology to computed tomography, radar, or other experimental data.

Finally, a thorough empirical comparison between standard DNN approaches and algorithm unrolling-based DNN approaches for solving inverse problems stands as another crucial goal of future work for deep-learning-based inverse problems as a whole. Briefly discussed in Section 2.3.3, unrolling approaches, such as CG-Net, appear to have a clear advantage when small training data sets are available. However, given enough training data, standard DNN approaches could have an advantage over unrolled approaches in a couple of ways. One is that unrolling approaches have required solution structure and restriction through data consistency layers, which may limit the total learning capacity of an unrolled DNN whereas standard DNN approaches have no such restrictions. Additionally, unrolled approaches often share trained weights across network layers leading to a recurrent network structure, which may suffer from vanishing and exploding gradient problems.

5.2.2 Deep Regularized Compound Gaussian Network

Leveraging the new foundations established in Chapter 3, numerous opportunities for future investigation of the G-CG-LS and DR-CG-Net approaches are illuminated. However, we note that many such opportunities for future work align with those presented for CG-Net. Specifically, one opportunity would be to extend G-CG-LS and DR-CG-Net to non-linear inverse problems by replacing the matrix A with a non-linear function for the forward measurement operator $\mathcal{F}_A$. Other opportunities are to analyze the performance of DR-CG-Net for larger image reconstructions, for data corrupted different noise models, and for training with larger datasets. In particular, studying each of these future objectives will provide greater insight into the practical, real-world applicability of our method.

We foresee the main complication in reconstructing larger images to be hardware constraints in the memory required to store matrices for and calculate the Tikhonov solution. One workaround is to replace the Tikhonov solution with gradient descent steps as we did for our 128×128 image results. While the gradient descent steps may not exactly produce the minimizer as the Tikhonov solution does, we have empirically shown that our DR-CG-Net method still successful reconstructs images. For Radon transform measurements, A is a sparse matrix and can be stored in a memory-efficient way (i.e. as a SparseTensor object in TensorFlow). Thus, with the development of a sparse linear solver through TensorFlow, only a slight modification of our code is necessary to efficiently deploy DR-CG-Net, with the exact Tikhonov solution, on larger images. Additionally, for Radon transform measurements, a matrix-free method, such as conjugate gradient, may be substituted to approximate the Tikhonov solution without constructing and storing the forward operator matrix. For CS applications, it is common to split larger images into disjoint blocks of small size and then measure and reconstruct the blocks separately [22, 26], which may be employed for future experimental evaluation of DR-CG-Net.

Another intriguing line of work is empirically studying DR-CG-Net, and each comparison deep learning method, for training and testing on a single category of images. The training and testing datasets employed in this paper contain images from multiple categories, e.g. CIFAR10 has 10

categories and CalTech101 has 101 categories, creating a significant statistical variability in the datasets. For some methods, such as FBPConvNet, the statistical variability in the training and testing datasets can distort performance. As such, it would be of interest to observe if DR-CG-Net continues to provide state-of-the-art performance on single modal datasets or if other deep learning methods are better suited when homogeneity in the training and testing datasets is known upfront. Furthermore, it would be of interest to observe how DR-CG-Net and all the other deep learning methods generalize to images from alternative categories after training on a single category. We conjecture that DR-CG-Net will generalize well to images outside of the training class in a similar manner to how well it performs on the CalTech101 dataset.

Exploring alternative scale-variable-update methods, rather than PGD and ISTA, such as the alternating direction method of multipliers is an additional opening for future work. The unrolled implementation for DR-CG-Net would require careful construction for a different scale-variable-update method, but may provide performance benefits such as in reconstructed image quality. Furthermore, proving convergence properties of G-CG-LS for alternative scale-variable-update methods stands as a crucial theoretical study when investigating this line of future work.

Additionally, the version of DR-CG-Net that uses convolutional layers as the subnetwork and learns a scaled identity covariance matrix is independent of signal dimension. Thus, instead of training a unique DR-CG-Net for different signal dimensions or measurement techniques, future work may explore modifying DR-CG-Net so that a single network is able to reconstruct varying sized signals from multiple measurement models. Modifications could include creating a bank of measurement devices, as some prior works have done [101, 102], which DR-CG-Net accesses and learns to interpolate for choosing an optimal measurement model when given an observed measurement.

Finally, a thorough empirical evaluation of the G-CG-LS method on a wide variety of linear inverse problems would be a fruitful point of future work. As CG-LS is a special case of G-CG-LS, we know that G-CG-LS will be successful in inverse problems for tomographic imaging and compressive sensing with a log-normal prior for the scale variable. However, as G-CG-LS permits

alternative scale variable regularizations, it would be illuminating to observe how the performance of G-CG-LS is affected by a total variation norm regularization or stochastic based regularization. Such choices may improve the performance of G-CG-LS, in reconstructed image quality, above the state-of-the-art performance demonstrated by CG-LS. Furthermore, a hybrid approach between G-CG-LS and DR-CG-Net is inviting to explore where the learned regularization from DR-CG-Net is plugged into the G-CG-LS algorithm for the scale variable regularization.

5.2.3 Theoretical Properties of Compound Gaussian Based Networks

While the generalization error bounds we have derived for CG-based deep neural networks show that with sufficient training data, roughly scaling quadratically in signal dimension and cubically in network size, small generalization error guarantees can be met, it still remains to be shown that such a property is true in low-training scenarios. Such a property is desirable as CG-Net and DR-CG-Net significantly outperform comparative methods in image estimation problems when trained with a small training dataset. Likely, aspects of the iterative algorithm, in particular the Tikhonov solution, would need to be further leveraged to provide insight into a small-training generalization-error bound. Furthermore, PAC-Bayes generalization bounds [125] for CG-based DNNs, extending the derived generalization error bounds in this paper, is another open line of future work, which can provide greater insights into the generalization for low-training scenarios.

Additionally, the theoretical study of statistical minimax bounds for the CG-LS and G-CG-LS estimators may help quantify the success that these two algorithms have shown empirically. Furthermore, a lower minimax bound could illuminate potential pitfalls and worst-case scenarios for the performance of either CG-LS or G-CG-LS. One possibility is to explore a sparsity-based approach, as in [126], where we would take the sparse object to be the scale variable field rather than the entire signal or image of interest.

5.2.4 Compound Gaussian Generative Adversarial Networks

Generative adversarial networks (GANs) are another deep learning technique that recently have been implemented as a learned prior information for inverse problems. Originally proposed by

Goodfellow et. al. in 2014 [127], GANs pit two separate deep neural networks, denoted as the generator and the discriminator, against each other in a zero-sum game. The generator is trained to produce counterfeit signals, i.e. images, while the discriminator is trained to distinguish authentic signals from counterfeit signals. Currently, generative adversarial networks are a premier approach in machine learning, gaining significant traction and attention across widespread fields and applications including image generation [128–133, 133–142], video generation [143–157], and text generation [158–167]. Due to the success of GANs in capturing the statistical properties of natural images, the generator of a GAN has been used as the prior information for linear inverse problems [14, 105–122]. That is, an inverse problem is solved where the reconstructed signal is constrained onto the range of a GAN generator.

Informed by the usage of GANs in inverse problems and by the excellent empirical results from Chapter 2 and Chapter 3, potential future work could create a CG-based GAN method to solve inverse problems. For instance, instead of constraining the estimated signal onto the range of a GAN generator, there may be an advantage gained in constraining the CG scale variable, Gaussian variable, or both onto the range of a generator. Additionally, a CG-based GAN structure for generating images could be a fruitful point of future work where independent parallel information processing generators are utilized in a way to resemble the CG prior structure where the Hadamard product of two independent random variables is applied.

Bibliography

[1] Simon Foucart and Holger Rauhut. *A Mathematical Introduction to Compressive Sensing*. Springer New York, 2013.

[2] Amir Beck and Marc Teboulle. A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, Jan 2009.

[3] Shihao Ji, Ya Xue, and Lawrence Carin. Bayesian Compressive Sensing. *IEEE Transactions on Signal Processing*, 56(6):2346–2356, 2008.

[4] Deanna Needell and Joel A. Tropp. CoSaMP: Iterative signal recovery from incomplete and inaccurate samples. *Applied and Computational Harmonic Analysis*, 26(3):301–321, May 2009.

[5] Ingrid Daubechies, Michel Defrise, and Christine De Mol. An Iterative Thresholding Algorithm for Linear Inverse Problems with a Sparsity Constraint. *Commun. Pure Appl. Math*, 57(11):1413–1457, Nov 2004.

[6] Seung-Jean Kim, Kwangmoo Koh, Michael Lustig, Stephen Boyd, and Dimitry Gorinevsky. An Interior-Point Method for Large-Scale ℓ_1-Regularized Least Squares. *IEEE Journal of Selected Topics in Signal Processing*, 1(4):606–617, Dec 2007.

[7] Deanna Needell and Roman Vershynin. Signal Recovery From Incomplete and Inaccurate Measurements Via Regularized Orthogonal Matching Pursuit. *IEEE Journal of Selected Topics in Signal Processing*, 4(2):310–316, 2010.

[8] Raghu G Raj. A hierarchical Bayesian-MAP approach to inverse problems in imaging. *Inverse Problems*, 32(7):075003, Jul 2016.

[9] John McKay, Raghu G Raj, and Vishal Monga. Fast stochastic hierarchical Bayesian map for tomographic imaging. In *51st Asilomar Conference on Signals, Systems, and Computers*, pages 223–227. IEEE, Oct 2017.

[10] Martin J Wainwright and Eero P Simoncelli. Scale Mixtures of Gaussians and the Statistics of Natural Images. In *Advances in Neural Information Processing Systems*, volume 12, pages 855–861. MIT Press, 1999.

[11] Martin J. Wainwright, Eero P. Simoncelli, and Alan S. Willsky. Random Cascades on Wavelet Trees and Their Use in Analyzing and Modeling Natural Images. *Applied and Computational Harmonic Analysis*, 11(1):89–123, 2001.

[12] Zachary Chance, Raghu G Raj, and David J Love. Information-theoretic structure of multi-static radar imaging. In *IEEE RadarCon (RADAR)*, pages 853–858, 2011.

[13] Kuldeep Kulkarni, Suhas Lohit, Pavan Turaga, Ronan Kerviche, and Amit Ashok. ReconNet: Non-Iterative Reconstruction of Images From Compressively Sensed Measurements. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 449–458, 2016.

[14] Ashish Bora, Ajil Jalal, Eric Price, and Alexandros G Dimakis. Compressed Sensing using Generative Models. In *Proceedings of the International Conference on Machine Learning*, volume 70, pages 537–546, 2017.

[15] Chen Qin, Jo Schlemper, Jose Caballero, Anthony N Price, Joseph V Hajnal, and Daniel Rueckert. Convolutional Recurrent Neural Networks for Dynamic MR Image Reconstruction. *IEEE Transactions on Medical Imaging*, 38(1):280–290, Jan 2019.

[16] Dong Liang, Jing Cheng, Ziwen Ke, and Leslie Ying. Deep Magnetic Resonance Image Reconstruction: Inverse Problems Meet Neural Networks. *IEEE Signal Processing Magazine*, 37(1):141–151, 2020.

[17] Alice Lucas, Michael Iliadis, Rafael Molina, and Aggelos K Katsaggelos. Using Deep Neural Networks for Inverse Problems in Imaging: Beyond Analytical Methods. *IEEE Signal Processing Magazine*, 35(1):20–36, 2018.

[18] Ge Wang, Jong Chul Ye, and Bruno De Man. Deep learning for tomographic image reconstruction. *Nature Machine Intelligence*, 2(12):737–748, Dec 2020.

[19] Shekhar S Chandra, Marlon Bran Lorenzana, Xinwen Liu, Siyu Liu, Steffen Bollmann, and Stuart Crozier. Deep learning in magnetic resonance image reconstruction. *Journal of Medical Imaging and Radiation Oncology*, 65(5):564–577, 2021.

[20] Karol Gregor and Yann LeCun. Learning fast approximations of sparse coding. In *Proceedings of the 27th International Conference on Machine Learning*, pages 399–406, 2010.

[21] Vishal Monga, Yuelong Li, and Yonina C Eldar. Algorithm Unrolling: Interpretable, efficient deep learning for signal and image processing. *IEEE Signal Processing Magazine*, 38(2):18–44, Mar 2021.

[22] Jiechong Song, Bin Chen, and Jian Zhang. Memory-Augmented Deep Unfolding Network for Compressive Sensing. In *Proceedings of the 29th ACM International Conference on Multimedia*, pages 4249–4258, Oct 2021.

[23] Jian Zhang and Bernard Ghanem. ISTA-Net: Interpretable Optimization-Inspired Deep Network for Image Compressive Sensing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1828–1837, 2018.

[24] Jinxi Xiang, Yonggui Dong, and Yunjie Yang. FISTA-Net: Learning a Fast Iterative Shrinkage Thresholding Network for Inverse Problems in Imaging. *IEEE Transactions on Medical Imaging*, 40(5):1329–1339, May 2021.

[25] Kyong Hwan Jin, Michael T McCann, Emmanuel Froustey, and Michael Unser. Deep Convolutional Neural Network for Inverse Problems in Imaging. *IEEE Transactions on Image Processing*, 26(9):4509–4522, 2017.

[26] Jiechong Song, Bin Chen, and Jian Zhang. Deep Memory-Augmented Proximal Unrolling Network for Compressive Sensing. *International Journal of Computer Vision*, 131(6):1477–1496, Jun 2023.

[27] Tim Meinhardt, Michael Moller, Caner Hazirbas, and Daniel Cremers. Learning Proximal Operators: Using Denoising Networks for Regularizing Inverse Imaging Problems. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1781–1790, 2017.

[28] Steven Diamond, Vincent Sitzmann, Felix Heide, and Gordon Wetzstein. Unrolled Optimization with Deep Priors. *arXiv preprint arXiv:1705.08041*, 2019.

[29] Yi Zhang, Hu Chen, Wenjun Xia, Yang Chen, Baodong Liu, Yan Liu, Huaiqiang Sun, and Jiliu Zhou. LEARN++: Recurrent Dual-Domain Reconstruction Network for Compressed Sensing CT. *IEEE Transactions on Radiation and Plasma Medical Sciences*, 2022.

[30] Yueming Su and Qiusheng Lian. iPiano-Net: Nonconvex optimization inspired multi-scale reconstruction network for compressed sensing. *Signal Processing: Image Communication*, 89:115989, 2020.

[31] Jonas Adler and Ozan Öktem. Learned Primal-Dual Reconstruction. *IEEE Transactions on Medical Imaging*, 37(6):1322–1332, 2018.

[32] Yan Yang, Jian Sun, Huibin Li, and Zongben Xu. ADMM-CSNet: A Deep Learning Approach for Image Compressive Sensing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(3):521–538, 2020.

[33] Uwe Schmidt and Stefan Roth. Shrinkage Fields for Effective Image Restoration. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2774–2781, 2014.

[34] Mario Bertero, Christine De Mol, and Edward Roy Pike. Linear inverse problems with discrete data: II. Stability and regularisation. *Inverse Problems*, 4(3):573–594, Aug 1988.

[35] Leslie Ying, Dan Xu, and Z-P Liang. On Tikhonov regularization for image reconstruction in parallel MRI. In *Proceedings of the 26th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, volume 1, pages 1056–1059. IEEE, 2004.

[36] José M Bioucas-Dias and Mário A T Figueiredo. A New TwIST: Two-Step Iterative Shrinkage/Thresholding Algorithms for Image Restoration. *IEEE Transactions on Image Processing*, 16(12):2992–3004, 2007.

[37] Yilun Wang, Junfeng Yang, Wotao Yin, and Yin Zhang. A New Alternating Minimization Algorithm for Total Variation Image Reconstruction. *SIAM Journal on Imaging Sciences*, 1(3):248–272, 2008.

[38] Joachim Dahl, Per Christian Hansen, Søren Holdt Jensen, and Tobias Lindstrøm Jensen. Algorithms and software for total variation image reconstruction via first-order methods. *Numerical Algorithms*, 53(1):67–92, Jan 2010.

[39] Yunhai Xiao, Junfeng Yang, and Xiaoming Yuan. Alternating algorithms for total variation image reconstruction from random projections. *Inverse Problems & Imaging*, 6(3):547, 2012.

[40] Ye Zhang and Chuchu Chen. Stochastic asymptotical regularization for linear inverse problems. *Inverse Problems*, 39(1):015007, 2022.

[41] Jiaming Liu, Yu Sun, Xiaojian Xu, and Ulugbek S Kamilov. Image Restoration Using Total Variation Regularized Deep Image Prior. In *2019 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 7715–7719. IEEE, 2019.

[42] Jiaming Liu, Yu Sun, Cihat Eldeniz, Weijie Gan, Hongyu An, and Ulugbek S Kamilov. RARE: Image Reconstruction Using Deep Priors Learned Without Groundtruth. *IEEE Journal of Selected Topics in Signal Processing*, 14(6):1088–1099, 2020.

[43] Jian Wang, Aleksandar Dogandžić, and Arye Nehorai. Maximum Likelihood Estimation of Compound-Gaussian Clutter and Target Parameters. *IEEE Transactions on Signal Processing*, 54(10):3884–3898, 2006.

[44] Javier Portilla, Vasily Strela, Martin J Wainwright, and Eero P Simoncelli. Image Denoising Using Scale Mixtures of Gaussians in the Wavelet Domain. *IEEE Transactions on Image Processing*, 12(11):1338–1351, Nov 2003.

[45] Tao Huang, Weisheng Dong, Xin Yuan, Jinjian Wu, and Guangming Shi. Deep Gaussian Scale Mixture Prior for Spectral Compressive Imaging. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16216–16225, 2021.

[46] Athanasios Papoulis and S Unnikrishna Pillai. *Probability, Random Variables and Stochastic Processes*. McGraw Hill, 1981.

[47] Raj S Chhikara and J Leroy Folks. *The Inverse Gaussian Distribution: Theory, Methodology, and Applications*. Marcel Dekker, Inc, 1989.

[48] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT press, 2016.

[49] Hang Zhao, Orazio Gallo, Iuri Frosio, and Jan Kautz. Loss Functions for Image Restoration With Neural Networks. *IEEE Transactions on Computational Imaging*, 3(1):47–57, Mar 2017.

[50] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, Mar 2004.

[51] Dimitri P. Bertsekas. *Nonlinear Programming*. Athena Scientific, 2016.

[52] Ward Cheney and Allen A Goldstein. Proximity Maps For Convex Sets. *Proceedings of the American Mathematical Society*, 10(3):448–450, 1959.

[53] Walter Rudin. *Principles of Mathematical Analysis*. McGraw Hill Education, 1953.

[54] R Tyrrell Rockafellar. *Convex Analysis*. Princeton university press, 1997.

[55] Ji He, Yongbo Wang, and Jianhua Ma. Radon Inversion via Deep Learning. *IEEE Transactions on Medical Imaging*, 39(6):2076–2087, 2020.

[56] Kyong Hwan Jin, Michael T McCann, Emmanuel Froustey, and Michael Unser. Deep Convolutional Neural Network for Inverse Problems in Imaging. *IEEE Transactions on Image Processing*, 26(9):4509–4522, 2017.

[57] Carter Lyons, Raghu G Raj, and Margaret Cheney. CG-Net: A Compound Gaussian Prior Based Unrolled Imaging Network. In *2022 IEEE Asia-Pacific Signal and Information Processing Association Annual Summit and Conference*, pages 623–629, 2022.

[58] Carter Lyons, Raghu G. Raj, and Margaret Cheney. A Compound Gaussian Least Squares Algorithm and Unrolled Network for Linear Inverse Problems. *IEEE Transactions on Signal Processing*, 71:4303–4316, 2023.

[59] Stephen J Wright. Coordinate descent algorithms. *Mathematical Programming*, 151(1):3–34, Jun 2015.

[60] Luigi Grippo and Marco Sciandrone. On the convergence of the block nonlinear Gauss–Seidel method under convex constraints. *Operations Research Letters*, 26(3):127–136, 2000.

[61] Luigi Grippof and Marco Sciandrone. Globally convergent block-coordinate techniques for unconstrained optimization. *Optimization Methods and Software*, 10(4):587–637, 1999.

[62] Amir Beck and Luba Tetruashvili. On the Convergence of Block Coordinate Descent Type Methods. *SIAM Journal on Optimization*, 23(4):2037–2060, Jan 2013.

[63] Max A Woodbury. *Inverting modified matrices*. Department of Statistics, Princeton University, 1950.

[64] Wladyslaw Kulpa. The Poincaré-Miranda Theorem. *The American Mathematical Monthly*, 104(6):545–550, 1997.

[65] John R Silvester. Determinants of Block Matrices. *The Mathematical Gazette*, 84(501):460–467, 2000.

[66] Walter Rudin. *Functional Analysis*. McGraw Hill Education, 1991.

[67] Constantin Carathéodory. Über den variabilitätsbereich der fourier'schen konstanten von positiven harmonischen funktionen. *Rendiconti Del Circolo Matematico di Palermo (1884-1940)*, 32(1):193–217, 1911.

[68] Ernst Steinitz. Bedingt konvergente reihen und konvexe systeme. *J. Reine Angew. Math.*, 1913.

[69] Alex Krizhevsky. Learning Multiple Layers of Features from Tiny Images. Technical report, University of Toronto, 2009.

[70] Li Fei-Fei, Rob Fergus, and Pietro Perona. Learning Generative Visual Models from Few Training Examples: An Incremental Bayesian Approach Tested on 101 Object Categories. In *Conference on Computer Vision and Pattern Recognition Workshop*, pages 178–178, 2004.

[71] Stanley R Deans. *The Radon Transform and Some of Its Applications*. Dover, 2007.

[72] Marco A Iglesias, Kui Lin, Shuai Lu, and Andrew M Stuart. Filter Based Methods for Statistical Linear Inverse Problems. *arXiv preprint arXiv:1512.01955*, 2022.

[73] Diederik P Kingma and Jimmy L Ba. Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations*, 2015.

[74] Atilim Gunes Baydin, Barak A Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. Automatic Differentiation in Machine Learning: a Survey. *Journal of Machine Learning Research*, 18, 2018.

[75] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, et al. TensorFlow: A System for Large-Scale Machine Learning. In *12th USENIX Symposium on Operating Systems Design and Implementation*, volume 16, pages 265–283, 2016.

[76] Carter Lyons, Raghu G. Raj, and Margaret Cheney. A Deep Compound Gaussian Regularized Unfoled Imaging Network. In *2022 56th Asilomar Conference on Signals, Systems, and Computers*, pages 940–947, 2022.

[77] Carter Lyons, Raghu G. Raj, and Margaret Cheney. Deep Regularized Compound Gaussian Network for Solving Linear Inverse Problems. *IEEE Transactions on Computational Imaging*, 2024.

[78] Yurii Nesterov. A method of solving a convex programming problem with convergence rate $O(1/k^2)$. In *Doklady Akademii Nauk*, volume 269, pages 543–547. Russian Academy of Sciences, 1983.

[79] Yangyang Xu and Wotao Yin. A Block Coordinate Descent Method for Regularized Multiconvex Optimization with Applications to Nonnegative Tensor Factorization and Completion. *SIAM Journal on Imaging Sciences*, 6(3):1758–1789, 2013.

[80] R Tyrrell Rockafellar. Monotone Operators and the Proximal Point Algorithm. *SIAM Journal on Control and Optimization*, 14(5):877–898, 1976.

[81] Kenneth Lange. *Optimization*, volume 95. Springer Science & Business Media, 2013.

[82] Ryan Murray, Brian Swenson, and Soummya Kar. Revisiting Normalized Gradient Descent: Fast Evasion of Saddle Points. *IEEE Transactions on Automatic Control*, 64(11):4818–4824, 2019.

[83] Johannes Leuschner, Maximilian Schmidt, Daniel Otero Baguer, and Peter Maass. LoDoPaB-CT, a benchmark dataset for low-dose computed tomography reconstruction. *Scientific Data*, 8(1):109, 2021.

[84] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

[85] SVMBIR Development Team. Super-Voxel Model Based Iterative Reconstruction (SVMBIR). Software library available from https://github.com/cabouman/svmbir, 2020.

[86] Arash Behboodi, Holger Rauhut, and Ekkehard Schnoor. Compressive Sensing and Neural Networks from a Statistical Learning Perspective. In *Compressed Sensing in Information Processing*, pages 247–277. Springer, 2022.

[87] Vicky Kouni and Yannis Panagakis. DECONET: An Unfolding Network for Analysis-Based Compressed Sensing With Generalization Error Bounds. *IEEE Transactions on Signal Processing*, 71:1938–1951, 2023.

[88] Boris Joukovsky, Tanmoy Mukherjee, Huynh Van Luong, and Nikos Deligiannis. Generalization Error Bounds for Deep Unfolding RNNs. In *Uncertainty in Artificial Intelligence*, pages 1515–1524. PMLR, 2021.

[89] Luc Devroye, László Györfi, and Gábor Lugosi. *A probabilistic theory of pattern recognition*, volume 31. Springer Science & Business Media, 2013.

[90] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge Univeristy Press, 2014.

[91] Andreas Maurer. A vector-contraction inequality for Rademacher complexities. In *Interantional Conference on Algorithmic Learning Theory*, pages 3–17. Springer, 2016.

[92] Zacharie Idriss, Raghu G Raj, and Ram M Narayanan. Waveform Optimization for Multistatic Radar Imaging Using Mutual Information. *IEEE Transactions on Aerospace and Electronics Systems*, 57(4):2410–2425, Aug 2021.

[93] Jorge Sola and Joaquin Sevilla. Importance of input data normalization for the application of neural networks to complex industrial problems. *IEEE Transactions on Nuclear Science*, 44(3):1464–1468, 1997.

[94] Joel N Franklin. *Matrix Theory*. Dover, 1993.

[95] Yihong Wu and Pengkun Yang. Optimal estimation of Gaussian mixtures via denoised method of moments. *The Annals of Statistics*, 48(4):1981 – 2007, 2020.

[96] Daniel Otero Baguer, Johannes Leuschner, and Maximilian Schmidt. Computed tomography reconstruction using deep image prior and learned reconstruction methods. *Inverse Problems*, 36(9):094004, Sep 2020.

[97] Dongwoon Hyun, Jeremy J Dahl, Kevin T Looby, and Leandra L Brickson. Ultrasound Speckle Reduction and Image Reconstruction using Deep Learning Techniques, U.S. Patent 011030780, Jun. 2021.

[98] Argyrou Maria, Maintas Dimitris, Tsoumpas Charalampos, and Stiliaris Efstathios. Tomographic Image Reconstruction based on Artificial Neural Network (ANN) Techniques. In *IEEE Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC)*, pages 3324–3327, 2012.

[99] Weiwen Wu, Dianlin Hu, Chuang Niu, Hengyong Yu, Varut Vardhanabhuti, and Ge Wang. DRONE: Dual-Domain Residual-Based Optimization NEtwork for Sparse-View CT Reconstruction. *IEEE Transactions on Medical Imaging*, 40(11):3002–3014, 2021.

[100] Siwang Zhou, Yan He, Yonghe Liu, Chengqing Li, and Jianming Zhang. Multi-channel deep networks for block-based image compressive sensing. *IEEE Transactions on Multimedia*, 23:2627–2640, 2021.

[101] Di You, Jian Zhang, Jingfen Xie, Bin Chen, and Siwei Ma. COAST: COntrollable Arbitrary-Sampling NeTwork for Compressive Sensing. *IEEE Transactions on Image Processing*, 30:6066–6080, 2021.

[102] Di You, Jingfen Xie, and Jian Zhang. ISTA-Net++: Flexible Deep Unfolding Network for Compressive Sensing. In *2021 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6. IEEE, 2021.

[103] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising. *IEEE transactions on image processing*, 26(7):3142–3155, 2017.

[104] Singanallur V Venkatakrishnan, Charles A Bouman, and Brendt Wohlberg. Plug-and-play priors for model based reconstruction. In *2013 IEEE Global Conference on Signal and Information Processing*, pages 945–948. IEEE, 2013.

[105] Ajil Jalal, Liu Liu, Alexandros G Dimakis, and Constantine Caramanis. Robust Compressed Sensing using Generative Models. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 713–727, 2020.

[106] Muhammad Asim, Max Daniels, Oscar Leong, Ali Ahmed, and Paul Hand. Invertible generative models for inverse problems: mitigating representation error and dataset bias. In *International Conference on Machine Learning*, pages 399–409. PMLR, 2020.

[107] Muhammad Asim, Fahad Shamshad, and Ali Ahmed. Blind Image Deconvolution Using Deep Generative Priors. *IEEE Transactions on Computational Imaging*, 6:1493–1506, 2020.

[108] Muhammad Asim, Fahad Shamshad, and Ali Ahmed. Solving Bilinear Inverse Problems using Deep Generative Priors. *CoRR*, 2018.

[109] Benjamin Aubin, Bruno Loureiro, Antoine Baker, Florent Krzakala, and Lenka Zdeborová. Exact asymptotics for phase retrieval and compressed sensing with random generative priors. In Jianfeng Lu and Rachel Ward, editors, *Proceedings of The First Mathematical and Scientific Machine Learning Conference*, volume 107 of *Proceedings of Machine Learning Research*, pages 55–73. PMLR, 20–24 Jul 2020.

[110] Paul Hand and Babhru Joshi. Global Guarantees for Blind Demodulation with Generative Priors. *Advances in Neural Information Processing Systems (NeurIPS)*, 32, 2019.

[111] Paul Hand, Oscar Leong, and Vlad Voroninski. Phase Retrieval Under a Generative Prior. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31, 2018.

[112] Gauri Jagatap and Chinmay Hegde. Phase Retrieval using Untrained Neural Network Priors. In *NeurIPS 2019 Workshop on Solving Inverse Problems with Deep Networks*, 2019.

[113] Zhaoqiang Liu, Selwyn Gomes, Avtansh Tiwari, and Jonathan Scarlett. Sample Complexity Bounds for 1-bit Compressive Sensing and Binary Stable Embeddings with Generative Priors. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 6216–6225. PMLR, 13–18 Jul 2020.

[114] Shuang Qiu, Xiaohan Wei, and Zhuoran Yang. Robust One-Bit Recovery via ReLU Generative Networks: Improved Statistical Rates and Global Landscape Analysis. *arXiv preprint arXiv:1908.05368*, 2019.

[115] Manik Dhar, Aditya Grover, and Stefano Ermon. Modeling Sparse Deviations for Compressed Sensing using Generative Models. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1214–1223. PMLR, 10–15 Jul 2018.

[116] Maya Kabkab, Pouya Samangouei, and Rama Chellappa. Task-Aware Compressed Sensing with Generative Adversarial Networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[117] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Deep Image Prior. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[118] Reinhard Heckel and Mahdi Soltanolkotabi. Compressive sensing with un-trained neural networks: Gradient descent finds a smooth approximation. In *Proceedings of the 37th*

International Conference on Machine Learning, volume 119 of *Proceedings of Machine Learning Research*, pages 4149–4158. PMLR, 13–18 Jul 2020.

[119] Morteza Mardani, Enhao Gong, Joseph Y. Cheng, Shreyas S. Vasanawala, Greg Zaharchuk, Lei Xing, and John M. Pauly. Deep Generative Adversarial Neural Networks for Compressive Sensing MRI. *IEEE Transactions on Medical Imaging*, 38(1):167–179, 2019.

[120] Ganlin Song, Zhou Fan, and John Lafferty. Surfing: Iterative Optimization Over Incrementally Trained Deep Networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, 2019.

[121] Viraj Shah and Chinmay Hegde. Solving Linear Inverse Problems Using GAN Priors: An Algorithm with Provable Guarantees. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4609–4613, 2018.

[122] Fabian Latorre, Armin Eftekhari, and Volkan Cevher. Fast and Provable ADMM for Learning with Generative Priors. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, 2019.

[123] Carter Lyons, Raghu G. Raj, and Margaret Cheney. On Generalization Bounds for Deep Compound Gaussian Networks. In preparation.

[124] David C Munson, James D O'Brien, and W Kenneth Jenkins. A Tomographic Formulation of Spotlight-Mode Synthetic Aperture Radar. *Proceedings of the IEEE*, 71(8):917–925, Aug 1983.

[125] Pierre Alquier. User-friendly introduction to pac-bayes bounds. *arXiv preprint arXiv:2110.11216*, 2021.

[126] David L Donoho, Iain M Johnstone, Jeffrey C Hoch, and Alan S Stern. Maximum Entropy and the Nearly Black Object. *Journal of the Royal Statistical Society: Series B (Methodological)*, 54(1):41–67, 1992.

[127] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems 27 (NIPS 2014)*, volume 27, 2014.

[128] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-Image Translation with Conditional Adversarial Networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

[129] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired Image-To-Image Translation Using Cycle-Consistent Adversarial Networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[130] Yongyi Lu, Yu-Wing Tai, and Chi-Keung Tang. Attribute-Guided Face Generation Using Conditional CycleGAN. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[131] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4681–4690, 2017.

[132] Xintao Wang, Ke Yu, Shixiang Wu, Jinjin Gu, Yihao Liu, Chao Dong, Yu Qiao, and Chen Change Loy. ESRGAN: Enhanced Super-Resolution Generative Adversarial Networks. In *Proceedings of the European conference on computer vision (ECCV) workshops*, 2018.

[133] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive Growing of GANs for Improved Quality, Stability, and Variation. *arXiv preprint arXiv:1710.10196*, 2017.

[134] Tero Karras, Samuli Laine, and Timo Aila. A Style-Based Generator Architecture for Generative Adversarial Networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4401–4410, 2019.

[135] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and Improving the Image Quality of StyleGAN. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8110–8119, 2020.

[136] Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Training Generative Adversarial Networks with Limited Data. *Advances in neural information processing systems*, 33:12104–12114, 2020.

[137] Tero Karras, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Alias-Free Generative Adversarial Networks. *Advances in Neural Information Processing Systems*, 34:852–863, 2021.

[138] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-Attention Generative Adversarial Networks. In *International Conference on Machine Learning*, pages 7354–7363. PMLR, 2019.

[139] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large Scale GAN Training for High Fidelity Natural Image Synthesis. In *International Conference on Learning Representations*, 2018.

[140] Axel Sauer, Katja Schwarz, and Andreas Geiger. StyleGAN-XL: Scaling StyleGAN to Large Diverse Datasets. In *ACM SIGGRAPH 2022 conference proceedings*, pages 1–10, 2022.

[141] Or Patashnik, Zongze Wu, Eli Shechtman, Daniel Cohen-Or, and Dani Lischinski. StyleCLIP: Text-Driven Manipulation of StyleGAN Imagery. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2085–2094, 2021.

[142] Katherine Crowson, Stella Biderman, Daniel Kornis, Dashiell Stander, Eric Hallahan, Louis Castricato, and Edward Raff. VQGAN-CLIP: Open Domain Image Generation and Editing with Natural Language Guidance. In *European Conference on Computer Vision*, pages 88–105. Springer, 2022.

[143] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Generating Videos with Scene Dynamics. *Advances in neural information processing systems*, 29, 2016.

[144] Sergey Tulyakov, Ming-Yu Liu, Xiaodong Yang, and Jan Kautz. MoCoGAN: Decomposing Motion and Content for Video Generation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1526–1535, 2018.

[145] Masaki Saito, Eiichi Matsumoto, and Shunta Saito. Temporal Generative Adversarial Nets with Singular Value Clipping. In *Proceedings of the IEEE international conference on computer vision*, pages 2830–2839, 2017.

[146] Katsunori Ohnishi, Shohei Yamamoto, Yoshitaka Ushiku, and Tatsuya Harada. Hierarchical Video Generation from Orthogonal Information: Optical Flow and Texture. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[147] Yaohui Wang, Piotr Bilinski, Francois Bremond, and Antitza Dantcheva. G^3AN: Disentangling Appearance and Motion for Video Generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5264–5273, 2020.

[148] Gaurav Mittal and Baoyuan Wang. Animating Face using Disentangled Audio Representations. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3290–3298, 2020.

[149] Yitong Li, Zhe Gan, Yelong Shen, Jingjing Liu, Yu Cheng, Yuexin Wu, Lawrence Carin, David Carlson, and Jianfeng Gao. StoryGAN: A Sequential Conditional GAN for Story Visualization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6329–6338, 2019.

[150] Yogesh Balaji, Martin Renqiang Min, Bing Bai, Rama Chellappa, and Hans Peter Graf. Conditional GAN with Discriminative Filter Generation for Text-to-Video Synthesis. In *IJCAI*, volume 1, page 2, 2019.

[151] Junting Pan, Chengyu Wang, Xu Jia, Jing Shao, Lu Sheng, Junjie Yan, and Xiaogang Wang. Video Generation from Single Semantic Label Map. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3733–3742, 2019.

[152] Ya-Liang Chang, Zhe Yu Liu, Kuan-Ying Lee, and Winston Hsu. Free-Form Video Inpainting with 3D Gated Convolution and Temporal PatchGAN. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9066–9075, 2019.

[153] Yanhong Zeng, Jianlong Fu, and Hongyang Chao. Learning Joint Spatial-Temporal Transformations for Video Inpainting. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVI 16*, pages 528–543. Springer, 2020.

[154] Haoye Cai, Chunyan Bai, Yu-Wing Tai, and Chi-Keung Tang. Deep Video Generation, Prediction and Completion of Human Action Sequences. In *Proceedings of the European conference on computer vision (ECCV)*, pages 366–382, 2018.

[155] Hyeongwoo Kim, Pablo Garrido, Ayush Tewari, Weipeng Xu, Justus Thies, Matthias Niessner, Patrick Pérez, Christian Richardt, Michael Zollhöfer, and Christian Theobalt. Deep Video Portraits. *ACM transactions on graphics (TOG)*, 37(4):1–14, 2018.

[156] Aliaksandr Siarohin, Stéphane Lathuilière, Sergey Tulyakov, Elisa Ricci, and Nicu Sebe. Animating Arbitrary Objects via Deep Motion Transfer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2377–2386, 2019.

[157] Lingjie Liu, Weipeng Xu, Michael Zollhoefer, Hyeongwoo Kim, Florian Bernard, Marc Habermann, Wenping Wang, and Christian Theobalt. Neural Rendering and Reenactment of Human Actor Videos. *ACM Transactions on Graphics (TOG)*, 38(5):1–14, 2019.

[158] Yizhe Zhang, Zhe Gan, and Lawrence Carin. Generating Text via Adversarial Training. In *NIPS workshop on Adversarial Training*, volume 21, pages 21–32. academia. edu, 2016.

[159] Weili Nie, Nina Narodytska, and Ankit Patel. RelGAN: Relational Generative Adversarial Networks for Text Generation. In *International conference on learning representations*, 2018.

[160] Haiyan Yin, Dingcheng Li, Xu Li, and Ping Li. Meta-CoTGAN: A Meta Cooperative Training Paradigm for Improving Adversarial Text Generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 9466–9473, 2020.

[161] Heng Wang, Zengchang Qin, and Tao Wan. Text Generation Based on Generative Adversarial Nets with Latent Variables. In *Advances in Knowledge Discovery and Data Mining: 22nd Pacific-Asia Conference, PAKDD 2018, Melbourne, VIC, Australia, June 3-6, 2018, Proceedings, Part II 22*, pages 92–103. Springer, 2018.

[162] William Fedus, Ian Goodfellow, and Andrew M Dai. MaskGAN: Better Text Generation via Filling in the _. In *International Conference on Learning Representations*, 2018.

[163] Yang Li, Quan Pan, Suhang Wang, Tao Yang, and Erik Cambria. A Generative Model for Category Text Generation. *Information Sciences*, 450:301–315, 2018.

[164] Jinyin Chen, Yangyang Wu, Chengyu Jia, Haibin Zheng, and Guohan Huang. Customizable Text Generation via Conditional Text Generative Adversarial Network. *Neurocomputing*, 416:125–135, 2020.

[165] Ke Wang and Xiaojun Wan. Automatic Generation of Sentimental Texts via Mixture Adversarial Networks. *Artificial Intelligence*, 275:540–558, 2019.

[166] Dayiheng Liu, Jie Fu, Qian Qu, and Jiancheng Lv. BFGAN: Backward and Forward Generative Adversarial Networks for Lexically Constrained Sentence Generation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 27(12):2350–2361, 2019.

[167] Liqun Chen, Shuyang Dai, Chenyang Tao, Haichao Zhang, Zhe Gan, Dinghan Shen, Yizhe Zhang, Guoyin Wang, Ruiyi Zhang, and Lawrence Carin. Adversarial Text Generation via Feature-Mover's Distance. *Advances in Neural Information Processing Systems*, 31, 2018.

Appendix A

Additional DR-CG-Net Numerical Results

A.1 Histograms

This section of the appendix provides histogram density plots of SSIM values from 8000 reconstructed test CIFAR10 images using DR-CG-Net and seven competitive deep learning methods. The solid vertical line and dashed vertical line on each histogram plot mark the mean and median SSIM value, respectively. We remark that for each method and every training setup considered in the histogram plots below, the median SSIM value is greater than the mean SSIM value. This signifies that the majority of reconstructed images are of higher quality than what is indicated by the mean and that a handful of outlier images exist with significantly lower quality than the mean. Although, this may also be an artifact of the SSIM function having a maximum value of 1 and so many high quality reconstructed images will have an SSIM value that bunches up near this maximum.

Additionally, while there is a significant variability among the histogram densities across the methods and across the different training setups, the histogram densities appear to resemble either a truncated normal distribution or a beta distribution (with parameter $a > b$ i.e. $a = 5$ and $b = 2$). Furthermore, the covariance of the reconstructed image SSIM values appears to decrease as the number of angles in the Radon transform increases and appears to increase with greater measurement noise and more data samples used in training. Lastly, in every histogram plot our DR-CG-Net method visually displays the highest mean and median SSIM value and, additionally, displays the smallest covariance out of the eight compared methods.

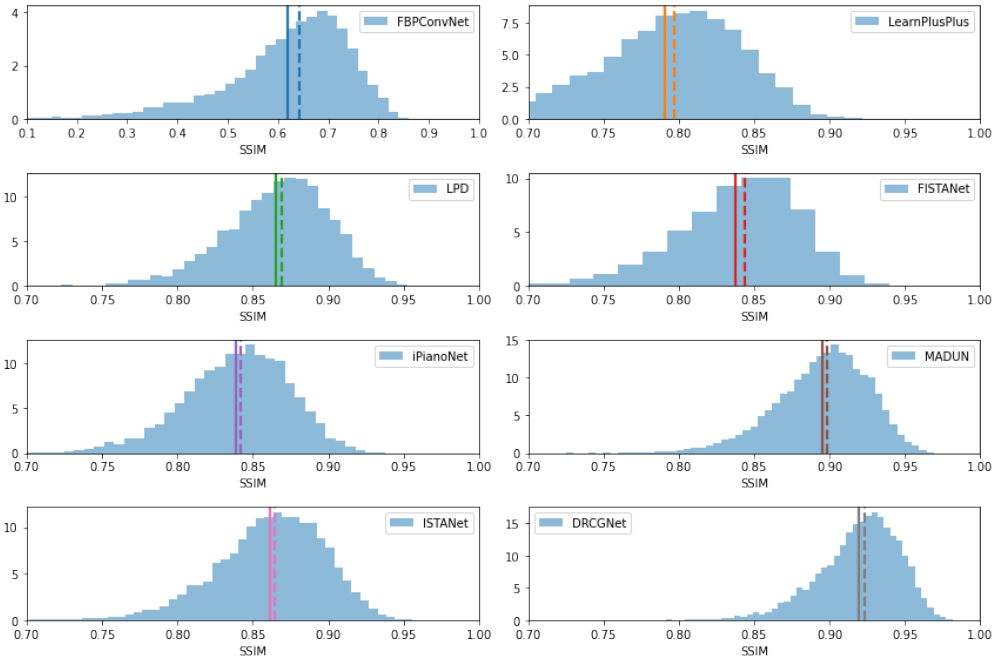


Figure A.1: Histogram density plots of SSIM values from 8000 test CIFAR10 images, reconstructed from Radon transforms at 15 uniformly spaced angles with a SNR of 60dB, using DR-CG-Net and seven competitive deep learning methods. Each method is trained with a dataset of only 20 samples. The solid vertical line and dashed vertical line on each histogram plot marks the mean and median SSIM value, respectively.

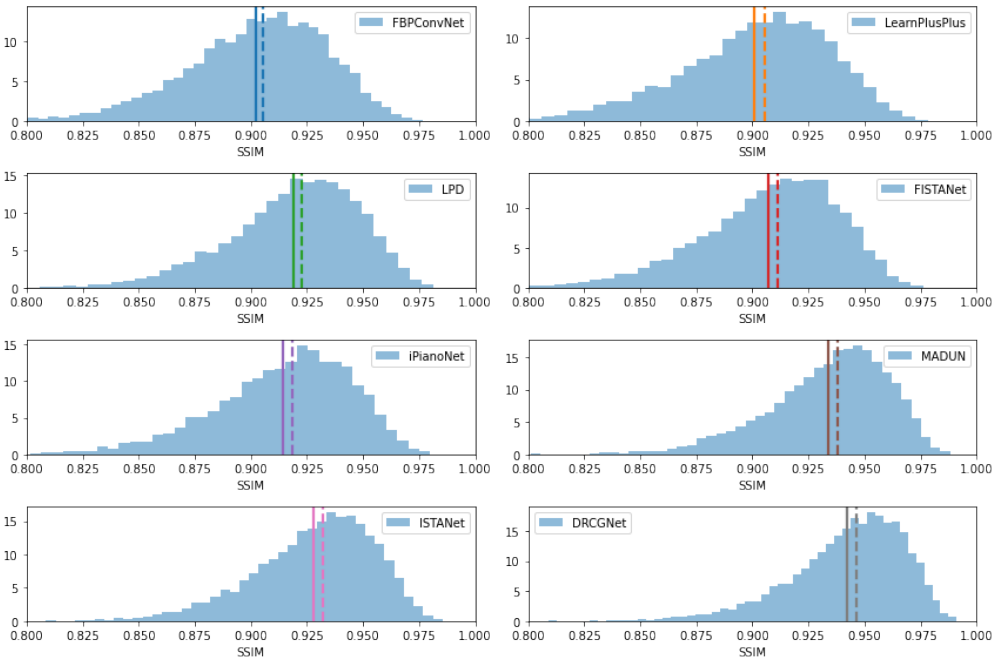


Figure A.2: Histogram density plots of SSIM values from 8000 test CIFAR10 images, reconstructed from Radon transforms at 15 uniformly spaced angles with a SNR of 60dB, using DR-CG-Net and seven competitive deep learning methods. Each method is trained with a dataset of 1000 samples. The solid vertical line and dashed vertical line on each histogram plot marks the mean and median SSIM value, respectively.

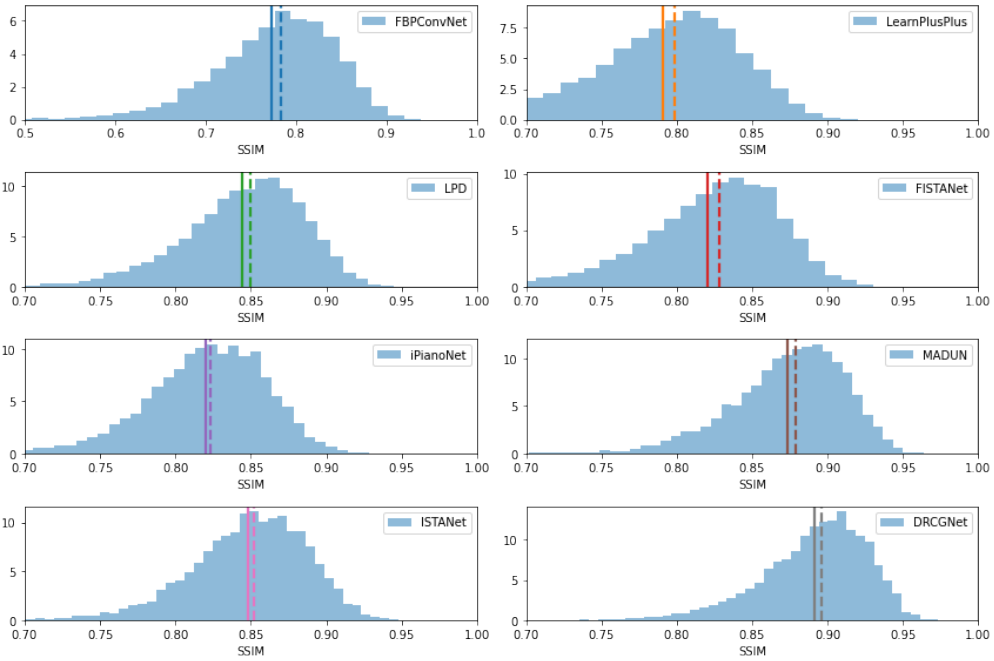


Figure A.3: Histogram density plots of SSIM values from 8000 test CIFAR10 images, reconstructed from Radon transforms at 15 uniformly spaced angles with a SNR of 40dB, using DR-CG-Net and seven competitive deep learning methods. Each method is trained with a dataset of only 20 samples. The solid vertical line and dashed vertical line on each histogram plot marks the mean and median SSIM value, respectively.

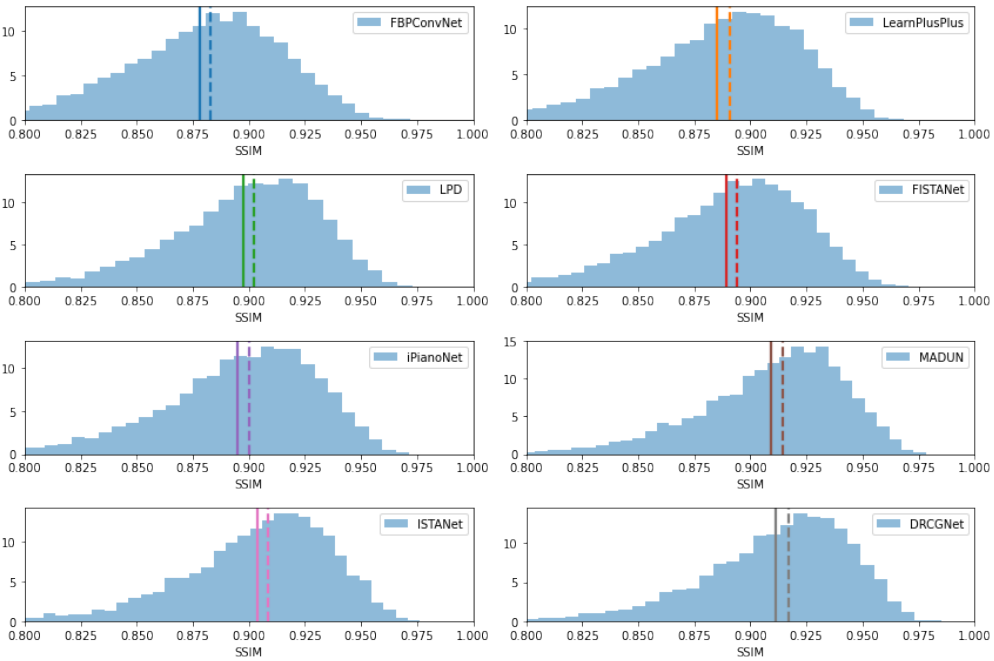


Figure A.4: Histogram density plots of SSIM values from 8000 test CIFAR10 images, reconstructed from Radon transforms at 15 uniformly spaced angles with a SNR of 40dB, using DR-CG-Net and seven competitive deep learning methods. Each method is trained with a dataset of 1000 samples. The solid vertical line and dashed vertical line on each histogram plot marks the mean and median SSIM value, respectively.

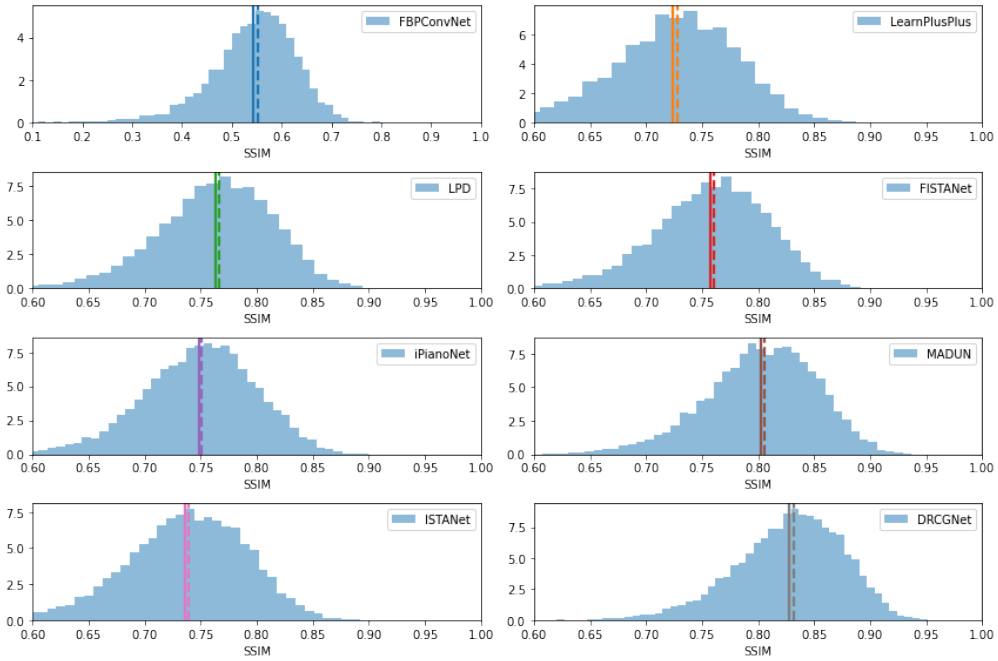


Figure A.5: Histogram density plots of SSIM values from 8000 test CIFAR10 images, reconstructed from Radon transforms at 10 uniformly spaced angles with a SNR of 60dB, using DR-CG-Net and seven competitive deep learning methods. Each method is trained with a dataset of only 20 samples. The solid vertical line and dashed vertical line on each histogram plot marks the mean and median SSIM value, respectively.

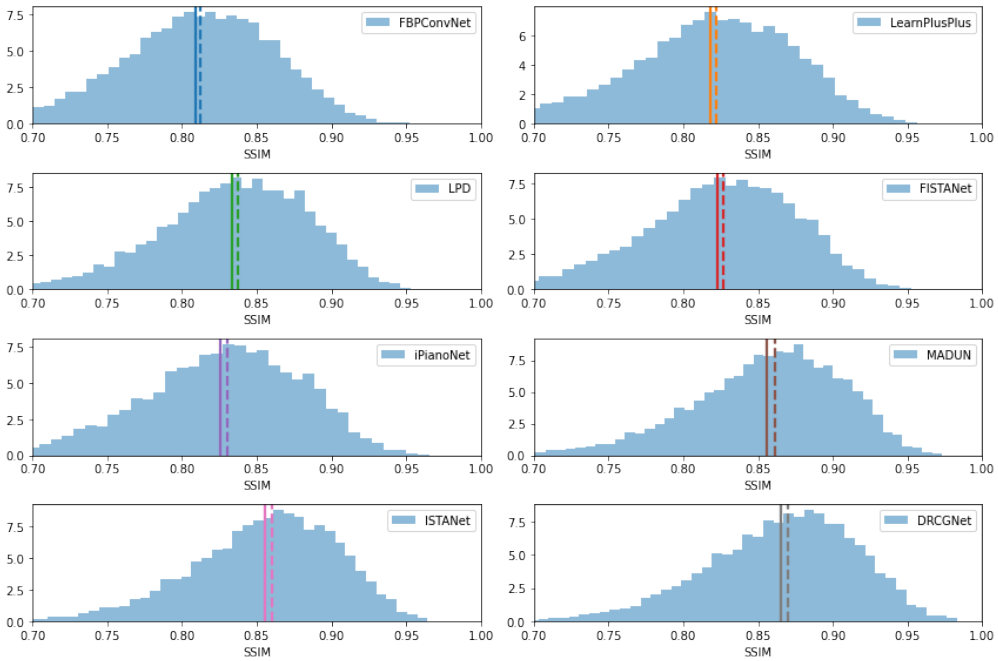


Figure A.6: Histogram density plots of SSIM values from 8000 test CIFAR10 images, reconstructed from Radon transforms at 10 uniformly spaced angles with a SNR of 60dB, using DR-CG-Net and seven competitive deep learning methods. Each method is trained with a dataset of 1000 samples. The solid vertical line and dashed vertical line on each histogram plot marks the mean and median SSIM value, respectively.

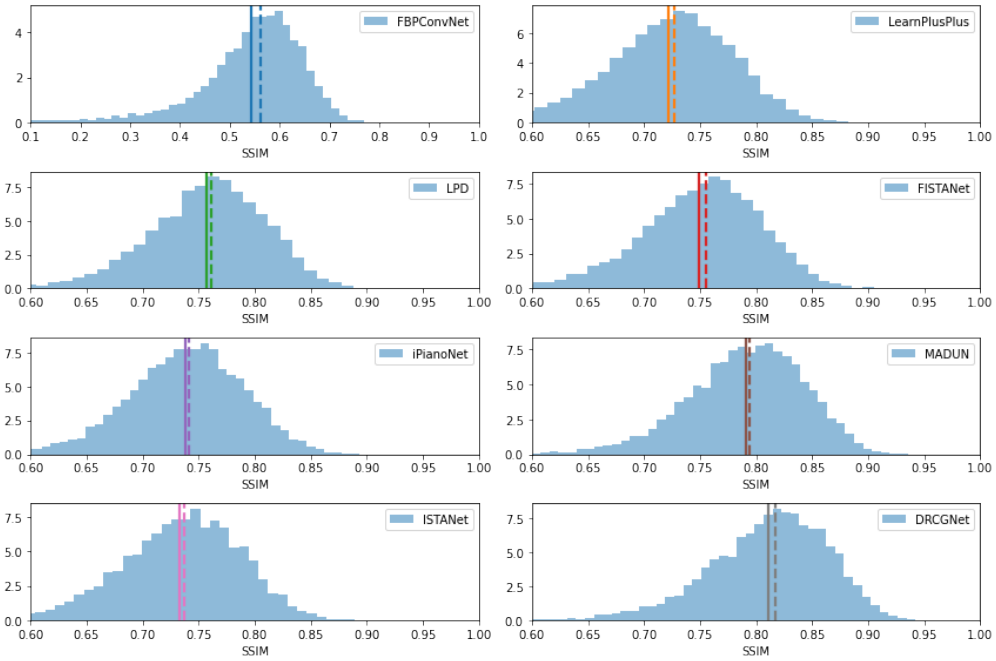


Figure A.7: Histogram density plots of SSIM values from 8000 test CIFAR10 images, reconstructed from Radon transforms at 10 uniformly spaced angles with a SNR of 40dB, using DR-CG-Net and seven competitive deep learning methods. Each method is trained with a dataset of only 20 samples. The solid vertical line and dashed vertical line on each histogram plot marks the mean and median SSIM value, respectively.

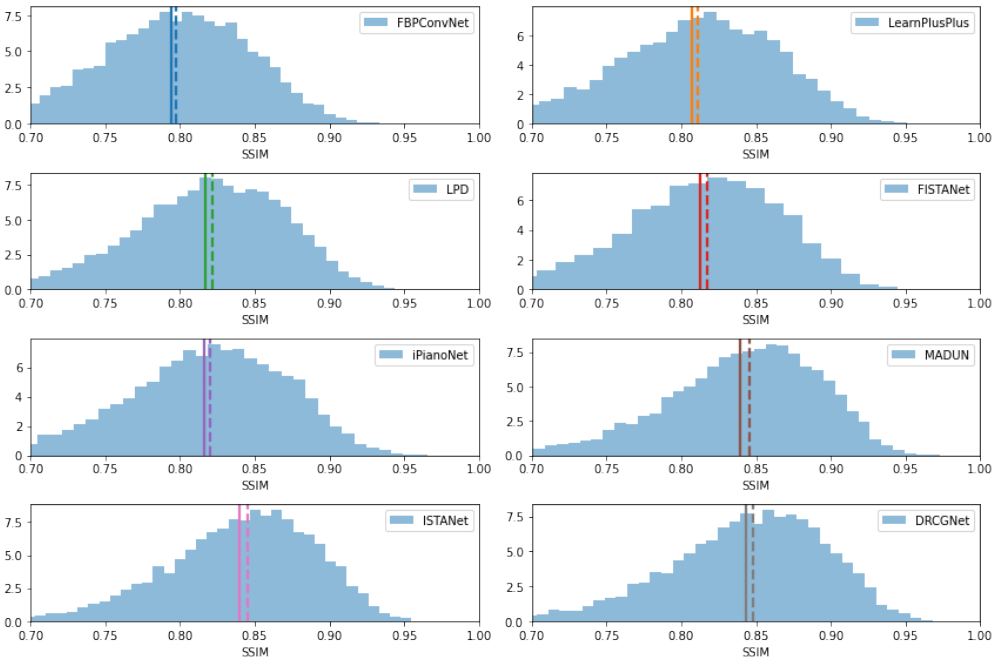


Figure A.8: Histogram density plots of SSIM values from 8000 test CIFAR10 images, reconstructed from Radon transforms at 10 uniformly spaced angles with a SNR of 40dB, using DR-CG-Net and seven competitive deep learning methods. Each method is trained with a dataset of 1000 samples. The solid vertical line and dashed vertical line on each histogram plot marks the mean and median SSIM value, respectively.

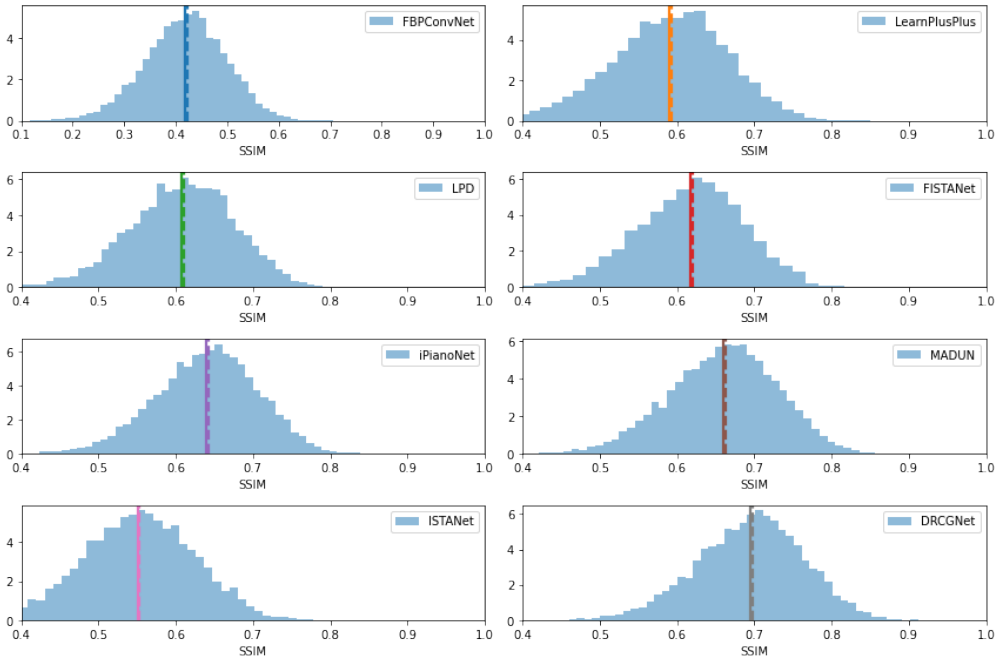


Figure A.9: Histogram density plots of SSIM values from 8000 test CIFAR10 images, reconstructed from Radon transforms at 6 uniformly spaced angles with a SNR of 60dB, using DR-CG-Net and seven competitive deep learning methods. Each method is trained with a dataset of only 20 samples. The solid vertical line and dashed vertical line on each histogram plot marks the mean and median SSIM value, respectively.

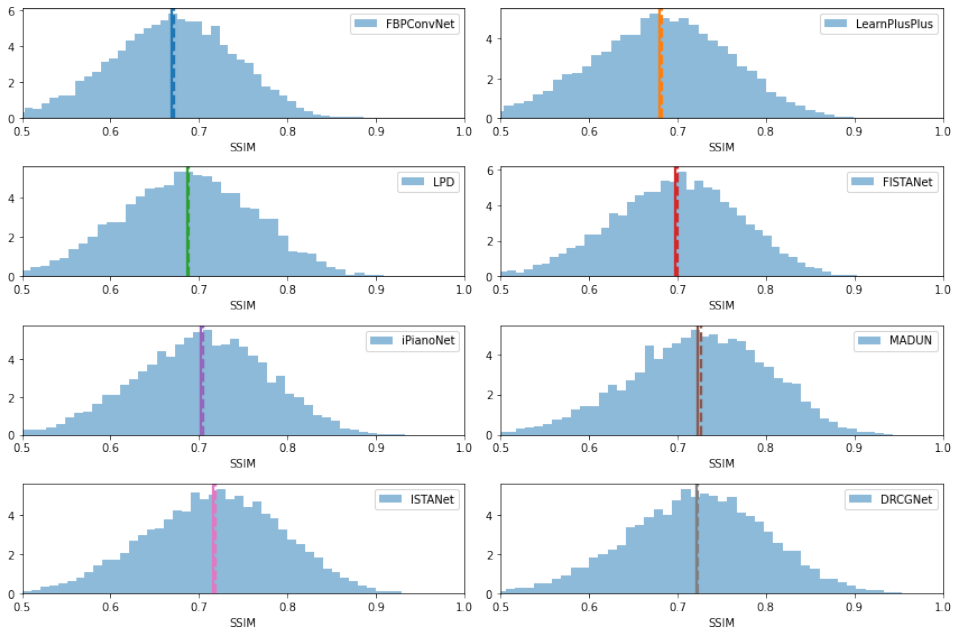


Figure A.10: Histogram density plots of SSIM values from 8000 test CIFAR10 images, reconstructed from Radon transforms at 6 uniformly spaced angles with a SNR of 60dB, using DR-CG-Net and seven competitive deep learning methods. Each method is trained with a dataset of 1000 samples. The solid vertical line and dashed vertical line on each histogram plot marks the mean and median SSIM value, respectively.

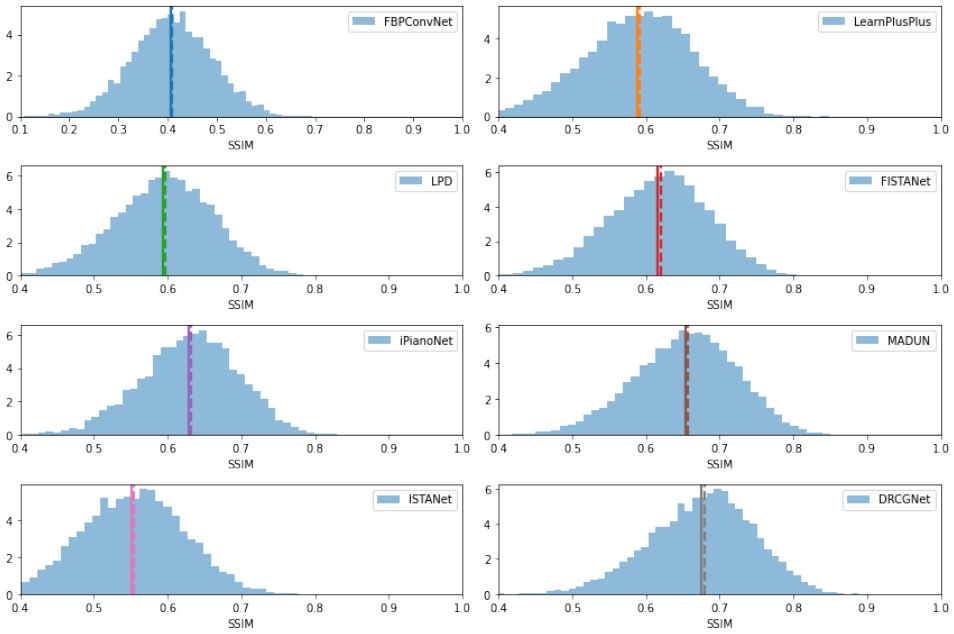


Figure A.11: Histogram density plots of SSIM values from 8000 test CIFAR10 images, reconstructed from Radon transforms at 6 uniformly spaced angles with a SNR of 40dB, using DR-CG-Net and seven competitive deep learning methods. Each method is trained with a dataset of only 20 samples. The solid vertical line and dashed vertical line on each histogram plot marks the mean and median SSIM value, respectively.

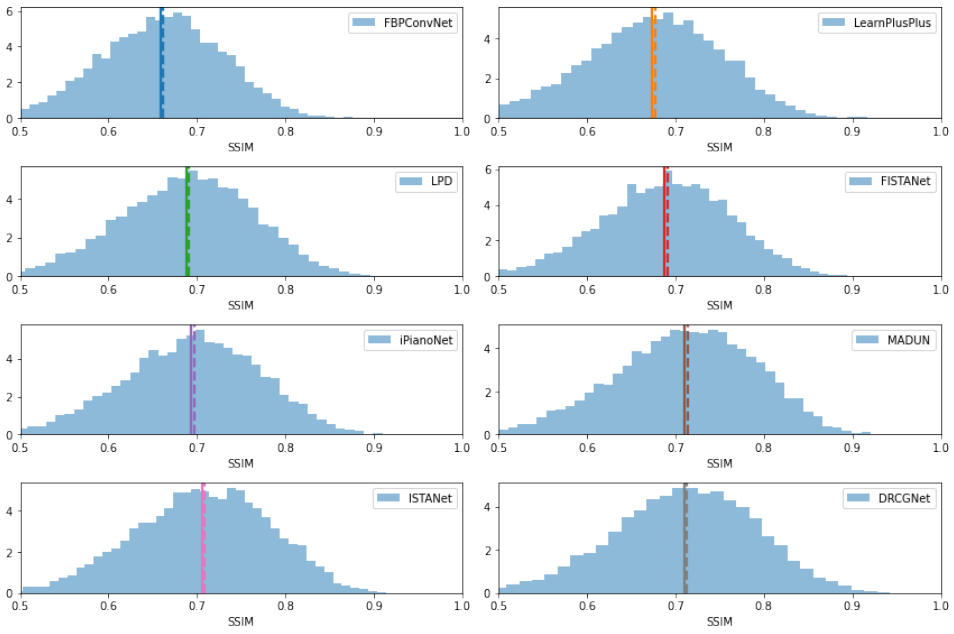


Figure A.12: Histogram density plots of SSIM values from 8000 test CIFAR10 images, reconstructed from Radon transforms at 6 uniformly spaced angles with a SNR of 40dB, using DR-CG-Net and seven competitive deep learning methods. Each method is trained with a dataset of 1000 samples. The solid vertical line and dashed vertical line on each histogram plot marks the mean and median SSIM value, respectively.

Appendix B

CG-Net Asymptotic Generalization Error

In this appendix we supply a detailed construction of the asymptotic generalization error bound forms for CG-Net, which are stated in section 4.5.1.

B.1 Properties of Big O Notation

We provide a handful of properties of the Big O notation, which will be useful in producing asymptotic forms for the unrolled, CG-based DNN generalization error bounds.

Lemma B.1.1. *Let* $f_1 = \mathcal{O}(g_1)$ *and* $f_2 = \mathcal{O}(g_2)$ *then* $f_1 + f_2 = \mathcal{O}\left(\max\left\{g_1, g_2\right\}\right)$.

Proof. As $f_i = \mathcal{O}(g_i)$ then there exists constants M_i and x_i such that $|f_i| \leq M_i g_I(x)$ for all $x \geq x_i$. Thus, using the triangle inequality, for all $x \geq \max\{x_1, x_2\}$

$$\begin{aligned} |f_1 + f_2| \leq |f_1| + |f_2| &\leq M_1 g_1(x) + M_2 g_2(x) \\ &\leq M_1 \max\{g_1(x), g_2(x)\} + M_2 \max\{g_1(x), g_2(x)\} \\ &= (M_1 + M_2) \max\{g_1(x), g_2(x)\} \end{aligned}$$

implying that $f_1 + f_2 = \mathcal{O}\left(\max\left\{g_1, g_2\right\}\right)$. ⊠

B.2 Dependent on Signal Size

We provide a proof of Corollary 4.5.2 where $c_{\max}, y_{\max}, \|A\|_2$, and $\|A\|_\infty$ depend on the measurement and signal of interest size.

Proof of Corollary 4.5.2. First recall that the CG-Net GEB given as

$$\begin{aligned}\mathcal{L}(\widehat{\boldsymbol{c}}) \leq \mathcal{L}_{\mathcal{S}}(\widehat{\boldsymbol{c}}) &+ \frac{8\tau c_{\max}}{\sqrt{N_s}}\sqrt{\ln\left(e\left(1+\frac{4p_{\max}(2KJ+1)\kappa}{c_{\max}}\right)\right)} \\ &+ \frac{8\tau c_{\max}}{\sqrt{N_s}}\sum_{k=1}^{K}\sum_{j=1}^{J}\sqrt{\frac{n(n+1)}{2}}\sqrt{\ln\left(e\left(1+\frac{4p_{\max}(2KJ+1)\kappa_{k,1}^{(j)}}{c_{\max}}\right)\right)} \\ &+ \frac{8\tau c_{\max}}{\sqrt{N_s}}\sum_{k=1}^{K}\sum_{j=1}^{J}\sqrt{\ln\left(e\left(1+\frac{4\mu(2KJ+1)\kappa_{k,2}^{(j)}}{c_{\max}}\right)\right)} \\ &+ 8\sqrt{2\ln(4/\varepsilon)/N_s}\end{aligned}$$

for τ a Lipschitz constant of the SSIM loss function and

$$\begin{aligned}\kappa &= z_\infty(c_1 + p_{\max}y_{\max}\|A\|_\infty)\widehat{c}_1^{(K,J)} + z_\infty c_2 \\ \kappa_{k,d}^{(j)} &= z_\infty(c_1 + p_{\max}y_{\max}\|A\|_\infty)\widehat{c}_{k,j,d,2}^{(K,J)}\end{aligned}$$

where

$$\begin{aligned}c_1 &= p_{\max}y_{\max}\ \|A\|_2\left(1+2z_\infty^2 p_{\max}\ \|A\|_2^2\right) \\ c_2 &= z_\infty y_{\max}\|A\|_2\left(p_{\max}/p_{\min}\right)^2 \\ \widehat{c}_1^{(K,J)} &= c_2 r_2\left(\frac{1-r_1^J}{1-r_1}\right)\left(\frac{1-\left(r_1^J + c_1 r_2\frac{1-r_1^J}{1-r_1}\right)^K}{1-\left(r_1^J + c_1 r_2\frac{1-r_1^J}{1-r_1}\right)}\right) \\ \widehat{c}_{k,j,d,2}^{(K,J)} &= r_1^{J-j}\left(r_1^J + c_1 r_2\frac{1-r_1^J}{1-r_1}\right)^{K-k}\begin{cases}\xi & d=1\\ p_{\max}h_{\max} & d=2\end{cases}\end{aligned}$$

for

$$
\begin{aligned}
r_1 &= 1 + p_{\max}\left((z_\infty p_{\max} y_{\max}\|A\|_2\|A\|_\infty)^2 + \mu\tau_h\right) \\
r_2 &= p_{\max} y_{\max}\|A\|_2\left(1 + z_\infty^2 p_{\max}\|A\|_2(\|A\|_2 + \|A\|_\infty)\right) \\
h_{\max} &= \max_{z\in[a,b]} [h^{-1}]'(z)h^{-1}(z) \\
\tau_h &= \max_{z\in[a,b]} [h^{-1}]''(z)h^{-1}(z) + [h^{-1}]'(z)^2.
\end{aligned}
$$

We will consider each term on the right hand side of the CG-Net GEB separately and provide a Big O bound for each. Subsequently, using Lemma B.1.1 we can combine each of the three separate bounds into a single bound by taking a maximum. Note that the empirical loss term $\mathcal{L}_{\mathcal{S}}(\widehat{\boldsymbol{c}})$ is independent of the network size of CG-Net and we will assume it is independent of the reconstructed signal size allowing us to ignore this term for the asymptotic analysis. Similarly, we ignore the error term $8\sqrt{2\ln(4/\varepsilon)/N_s}$ as it is independent of the network size of CG-Net as well as independent of the reconstructed signal size. Hence, we have four terms from the right hand side of the CG-Net GEB to consider.

Term 1:

Note that $\sqrt{1+x} \leq 1 + \sqrt{x}$ and $\ln(1+x) \leq \frac{\ln(3)}{\ln(2)}\ln(x)$ for all $x \geq 2$. Thus, taking n, K, and J large enough such that $c_{\max} \geq 4p_{\max}$ and $4p_{\max}(2KJ+1)\kappa \geq 2c_{\max}$

$$
\begin{aligned}
&\sqrt{\ln\left(e\left(1 + \frac{4p_{\max}(2KJ+1)\kappa}{c_{\max}}\right)\right)} \\
&\quad= \sqrt{1 + \ln\left(1 + \frac{4p_{\max}(2KJ+1)\kappa}{c_{\max}}\right)} \\
&\quad\leq 1 + \sqrt{\frac{\ln(3)}{\ln(2)}}\sqrt{\ln\left(\frac{4p_{\max}(2KJ+1)\kappa}{c_{\max}}\right)} \\
&\quad= 1 + \sqrt{\frac{\ln(3)}{\ln(2)}}\left(\sqrt{\ln\left(\frac{4p_{\max}}{c_{\max}}\right)} + \sqrt{\ln\left(2KJ+1\right)} + \sqrt{\ln(\kappa)}\right) \\
&\quad\leq 1 + \sqrt{\frac{\ln(3)}{\ln(2)}}\left(\sqrt{\ln\left(4p_{\max}\right)} + \sqrt{\ln\left(2KJ+1\right)} + \sqrt{\ln(\kappa)}\right). \qquad \text{(B.1)}
\end{aligned}
$$

As $(1-x^n)/(1-x) \geq 1$ for all $x \geq 1$ and $n \in \mathbb{N}$ and $r_1 \geq 1$ then

$$\begin{aligned}
\ln(\kappa) &= \ln\left(z_\infty(c_1 + p_{\max} y_{\max} ||A||_\infty) c_2 r_2 \frac{1-r_1^J}{1-r_1} \frac{1-(r_1^J + r_2 \frac{1-r_1^J}{1-r_1} c_1)^K}{1-(r_1^J + r_2 \frac{1-r_1^J}{1-r_1} c_1)} + z_\infty c_2 \right) \\
&\leq \ln\left(\frac{1-r_1^J}{1-r_1} \frac{1-(r_1^J + r_2 \frac{1-r_1^J}{1-r_1} c_1)^K}{1-(r_1^J + r_2 \frac{1-r_1^J}{1-r_1} c_1)} \left(z_\infty(c_1 + p_{\max} y_{\max} ||A||_\infty) c_2 r_2 + z_\infty c_2 \right) \right) \\
&= \ln\left(\frac{1-r_1^J}{1-r_1} \frac{1-(r_1^J + r_2 \frac{1-r_1^J}{1-r_1} c_1)^K}{1-(r_1^J + r_2 \frac{1-r_1^J}{1-r_1} c_1)} \right) + \ln(z_\infty(c_1 + p_{\max} y_{\max} ||A||_\infty) c_2 r_2 + z_\infty c_2).
\end{aligned} \tag{B.2}$$

Next, we will show that $\frac{1-x^n}{1-x} \leq \frac{n}{2}(1+x^{n-1})$ for all $x \geq 0$ and $n \in \mathbb{N}$. For this, define the rational function $Q_n(x) = \frac{1-x^n}{(1-x)(1+x^{n-1})}$ for $n \in \mathbb{N}$. First, note that $1-x^n = (1-x)(1+x+x^2+\cdots+x^{n-1})$ and thus $Q_n = \frac{1+x+x^2+\cdots+x^{n-1}}{1+x^{n-1}}$. For $n=1$, note that $Q_1(x) = \frac{1}{2} \leq \frac{n}{2}$. Now, let $n > 1$ and observe

$$Q_n'(x) = \frac{(1+x^{n-1})(1+2x+3x^2+\cdots+(n-1)x^{n-2}) - (1+x+x^2+\cdots+x^{n-1})((n-1)x^{n-2})}{(1+x^{n-1})^2}.$$

Next, observe

$$\begin{aligned}
&(1+x^{n-1})(1+2x+3x^2+\cdots+(n-1)x^{n-2}) - (1+x+x^2+\cdots+x^{n-1})((n-1)x^{n-2}) \\
&= 1+2x+3x^2+\cdots+(n-1)x^{n-2} + x^{n-1} + 2x^n + 3x^{n+1} + \cdots + (n-1)x^{2n-3} \\
&\quad - (n-1)(x^{n-2} + x^{n-1} + x^n + \cdots + x^{2n-3}) \\
&= 1+2x+3x^2 + (n-2)x^{n-3} \\
&\quad + (1-(n-1))x^{n-1} + (2-(n-1))x^n + \cdots + ((n-2)-(n-1))x^{2n-4} \\
&= 1+2x+3x^2 + (n-2)x^{n-3} - (n-2)x^{n-1} - (n-3)x^n - \cdots - x^{2n-4} \\
&= 1 - x^{2(n-2)} + 2x(1-x^{2(n-3)}) + 3x^2(1-x^{2(n-4)}) + \cdots (n-2)x^{n-3}(1-x^2) \\
&= \sum_{k=1}^{n-2} k x^{k-1}(1-x^{2(n-(k+1))}),
\end{aligned}$$

implying that

$$Q_n'(x) = \frac{\sum_{k=1}^{n-2} kx^{k-1}(1 - x^{2(n-(k+1))})}{(1 + x^{n-1})^2}.$$

Clearly $Q_n'(0) = 1$ and $Q'(1) = 0$. Now, for $0 < x < 1$ note that $x^{k-1} > 0$ and $x^{2(n-(k+1))} < 1$ for all $n > 1$ and $k \in \{1, 2, \ldots, n-2\}$. Thus $Q_n'(x) > 0$ for all $x \in [0, 1)$. Next, for $x > 1$, note that $x^{k-1} > 0$ and $x^{2(n-(k+1))} > 1$ for all $n > 1$ and $k \in \{1, 2, \ldots, n-2\}$. Thus $Q_n'(x) < 0$ for all $x > 1$. Therefore, on $[0, \infty)$, the function $Q_n(x)$ has a single maximizer $x = 1$ producing the maximum value of $Q_n(1) = \frac{n}{2}$, implying that for all $x \geq 0$ and $n \in \mathbb{N}$

$$\frac{1 - x^n}{1 - x} \leq \frac{n}{2}(1 + x^{n-1}). \tag{B.3}$$

By (B.3), for all $x \geq 0$ and $n \in \mathbb{N}$ it holds that

$$\ln\left(\frac{1 - x^n}{1 - x}\right) \leq \ln\left(\frac{n}{2}\right) + \ln(1 + x^{n-1}).$$

Now, as $\ln(1 + x^{n-1}) \leq \ln(x^{n-1} + x^{n-1}) = \ln(2) + \ln(x^{n-1})$ for all $x \geq 1$ then for all $x \geq 1$

$$\begin{aligned}
\ln\left(\frac{1 - x^n}{1 - x}\right) &\leq \ln\left(\frac{n}{2}\right) + \ln(1 + x^{n-1}) \\
&\leq \ln\left(\frac{n}{2}\right) + \ln(2) + \ln(x^{n-1}) \\
&= \ln(n) + \ln(x^{n-1}) = \ln(n) + (n-1)\ln(x).
\end{aligned} \tag{B.4}$$

Since $r_1 \geq 1$ then $r_1^J + r_2 \frac{1-r_1^J}{1-r_1} c_1 \geq 1$ and thus using equation (B.4)

$$\begin{aligned}
&\ln\left(\frac{1-r_1^J}{1-r_1}\frac{1-(r_1^J + r_2\frac{1-r_1^J}{1-r_1}c_1)^K}{1-(r_1^J + r_2\frac{1-r_1^J}{1-r_1}c_1)}\right) \\
&\quad = \ln\left(\frac{1-r_1^J}{1-r_1}\right) + \ln\left(\frac{1-(r_1^J + r_2\frac{1-r_1^J}{1-r_1}c_1)^K}{1-(r_1^J + r_2\frac{1-r_1^J}{1-r_1}c_1)}\right) \\
&\quad \leq \ln(J) + (J-1)\ln(r_1) + \ln(K) + (K-1)\ln\left(r_1^J + r_2\frac{1-r_1^J}{1-r_1}c_1\right).
\end{aligned}$$

Next, using that $r_1 \geq 1$ and equation (B.3), observe

$$\begin{aligned}
\ln\left(r_1^J + r_2\frac{1-r_1^J}{1-r_1}c_1\right) &\leq \ln\left(r_1^J + r_2\frac{J}{2}(1+r_1^{J-1})c_1\right) \\
&= \ln\left(r_1^{J-1}\left(r_1 + r_2 c_1\frac{J}{2}\right) + r_2\frac{J}{2}c_1\right) \\
&\leq \ln\left(r_1^{J-1}\left(\left(r_1 + r_2 c_1\frac{J}{2}\right) + r_2\frac{J}{2}c_1\right)\right) \\
&= (J-1)\ln(r_1) + \ln\left(r_1 + r_2 c_1 J\right). \qquad \text{(B.5)}
\end{aligned}$$

Hence

$$\begin{aligned}
&\ln\left(\frac{1-r_1^J}{1-r_1}\frac{1-(r_1^J + r_2\frac{1-r_1^J}{1-r_1}c_1)^K}{1-(r_1^J + r_2\frac{1-r_1^J}{1-r_1}c_1)}\right) \\
&\quad \leq \ln(J) + (J-1)\ln(r_1) + \ln(K) + (K-1)\ln\left(r_1^J + r_2\frac{1-r_1^J}{1-r_1}c_1\right) \\
&\quad \leq \ln(J) + (J-1)\ln(r_1) + \ln(K) + (K-1)\left((J-1)\ln(r_1) + \ln\left(r_1 + r_2 c_1 J\right)\right) \\
&\quad = \ln(J) + \ln(K) + K(J-1)\ln(r_1) + (K-1)\ln\left(r_1 + r_2 c_1 J\right). \qquad \text{(B.6)}
\end{aligned}$$

Therefore, using equation (B.2), equation (B.6), and subadditivity of square roots

$$\begin{aligned}
&\sqrt{\ln(\kappa)} \\
&\leq \sqrt{\ln\left(\frac{1-r_1^J}{1-r_1}\frac{1-(r_1^J+r_2\frac{1-r_1^J}{1-r_1}c_1)^K}{1-(r_1^J+r_2\frac{1-r_1^J}{1-r_1}c_1)}\right)} + \sqrt{\ln(q_1)} \\
&\leq \sqrt{\ln(J)+\ln(K)+K(J-1)\ln(r_1)+(K-1)\ln\left(r_1+r_2c_1J\right)} + \sqrt{\ln(q_1)} \\
&\leq \sqrt{\ln(J)} + \sqrt{\ln(K)} + \sqrt{K(J-1)\ln(r_1)} + \sqrt{(K-1)\ln\left(r_1+r_2c_1J\right)} + \sqrt{\ln(q_1)} \qquad \text{(B.7)}
\end{aligned}$$

where $q_1 = z_\infty(c_1 + p_{\max}y_{\max}||A||_\infty)c_2r_2 + z_\infty c_2$ is a constant.

Combining equation (B.1) and equation (B.7)

$$\begin{aligned}
&\sqrt{\ln\left(e\left(1+\frac{4p_{\max}(2KJ+1)\kappa}{c_{\max}}\right)\right)} \\
&\qquad \leq 1 + \sqrt{\frac{\ln(3)}{\ln(2)}}\Big(\sqrt{\ln\left(4p_{\max}\right)} + \sqrt{\ln\left(2KJ+1\right)} + \sqrt{\ln(J)} + \sqrt{\ln(K)} \\
&\qquad\qquad + \sqrt{K(J-1)}\sqrt{\ln(r_1)} + \sqrt{K-1}\sqrt{\ln\left(r_1+r_2c_1J\right)} + \sqrt{\ln(q_1)}\Big).
\end{aligned}$$

Next, observe

$$\begin{aligned} c_1 &= p_{\max} y_{\max} \ \|A\|_2 \left(1 + 2z_\infty^2 p_{\max} \ \|A\|_2^2\right) \\ &\leq p_{\max} \max\{1, 2z_\infty^2 p_{\max}\} \left[y_{\max}\|A\|_2 \left(1 + \|A\|_2^2\right)\right] \\ &\lesssim y_{\max}\|A\|_2^3, \end{aligned} \tag{B.8}$$

$$c_2 = z_\infty y_{\max}\|A\|_2 \left(p_{\max}/p_{\min}\right)^2 \lesssim y_{\max}\|A\|_2, \tag{B.9}$$

$$\begin{aligned} r_1 &= 1 + p_{\max}\left(\left(z_\infty p_{\max} y_{\max}\|A\|_2\|A\|_\infty\right)^2 + \mu\tau_h\right) \\ &\leq 1 + p_{\max}(z_\infty^2 p_{\max} + \mu\tau_h)\left[\left(y_{\max}\|A\|_2\|A\|_\infty\right)^2\right] \\ &\lesssim \left(y_{\max}\|A\|_2\|A\|_\infty\right)^2, \end{aligned} \tag{B.10}$$

and

$$\begin{aligned} r_2 &= p_{\max} y_{\max}\|A\|_2 \left(1 + z_\infty^2 p_{\max}\|A\|_2(\|A\|_2 + \|A\|_\infty)\right) \\ &\leq p_{\max} \max\{1, z_\infty^2 p_{\max}\}\left[y_{\max}\|A\|_2 \left(1 + \|A\|_2 \left(\|A\|_2 + \|A\|_\infty\right)\right)\right] \\ &\lesssim y_{\max}\|A\|_2^2 \left(\|A\|_2 + \|A\|_\infty\right). \end{aligned} \tag{B.11}$$

Equation (B.10) implies that there exists a constant $a_{r_1} > 0$ such that

$$\begin{aligned} \ln(r_1) &\leq \ln\left(a_{r_1}\left(y_{\max}\|A\|_2\|A\|_\infty\right)^2\right) \\ &= \ln(a_{r_1}) + 2\left[\ln(y_{\max}) + \ln(\|A\|_2) + \ln(\|A\|_\infty)\right] \\ &\lesssim \ln(y_{\max}) + \ln(\|A\|_2) + \ln(\|A\|_\infty). \end{aligned} \tag{B.12}$$

Equation (B.8), equation (B.10), and equation (B.11) imply that there exists a constant $a_1 > 0$ such that

$$\begin{aligned}
\ln(r_1 + r_2 c_1 J) &\leq \ln\left(a_1\left[(y_{\max}\|A\|_2\|A\|_\infty)^2 + y_{\max}^2\|A\|_2^5\left(\|A\|_2 + \|A\|_\infty\right)J\right]\right) \\
&= \ln(a_1) + \ln\left(y_{\max}^2\|A\|_2^2\left[\|A\|_\infty^2 + \|A\|_2^3\left(\|A\|_2 + \|A\|_\infty\right)J\right]\right) \\
&= \ln(a_1) + 2\ln(y_{\max}) + 2\ln(\|A\|_2) \\
&\quad + \ln\left(\|A\|_\infty^2 + \|A\|_2^3\left(\|A\|_2 + \|A\|_\infty\right)J\right) \\
&\leq \ln(a_1) + 2\ln(y_{\max}) + 2\ln(\|A\|_2) + 2\ln(\|A\|_\infty) + 3\ln(\|A\|_2) \\
&\quad + \ln(\|A\|_2 + \|A\|_\infty) + \ln(J) \qquad \text{(B.13)} \\
&\leq \ln(a_1) + 2\ln(y_{\max}) + 6\ln(\|A\|_2) + 3\ln(\|A\|_\infty) + \ln(J) \qquad \text{(B.14)} \\
&\lesssim \ln(y_{\max}) + \ln(\|A\|_2) + \ln(\|A\|_\infty) + \ln(J) \qquad \text{(B.15)}
\end{aligned}$$

where in (B.13) and (B.14) we used that $\ln(x+y) \leq \ln(x) + \ln(y)$ for $x, y \geq 2$ and took n, m, and J large enough such that $\|A\|_\infty^2 \geq 2$, $\|A\|_2^3\left(\|A\|_2 + \|A\|_\infty\right)J \geq 2$, $\|A\|_2 \geq 2$, and $\|A\|_\infty \geq 2$.

Furthermore, equation (B.8), equation (B.9), and equation (B.11) imply that there exists a constant $a_{q_1} > 0$ such that

$$\begin{aligned}
q_1 &= z_\infty(c_1 + p_{\max}y_{\max}||A||_\infty)c_2 r_2 + z_\infty c_2 \\
&\leq a_{q_1} z_\infty\left(\left(y_{\max}\|A\|_2^3 + p_{\max}y_{\max}\|A\|_\infty\right)y_{\max}^2\|A\|_2^3\left(\|A\|_2 + \|A\|_\infty\right) + y_{\max}\|A\|_2\right) \\
&\leq a_{q_1} z_\infty y_{\max}\|A\|_2\left[\max\{1, p_{\max}\}y_{\max}^3\|A\|_2^2\left(\|A\|_2^3 + \|A\|_\infty\right)\left(\|A\|_2 + \|A\|_\infty\right) + 1\right] \\
&\leq a_{q_1} z_\infty\left(1 + \max\{1, p_{\max}\}\right)y_{\max}^4\|A\|_2^3\left(\|A\|_2^3 + \|A\|_\infty\right)\left(\|A\|_2 + \|A\|_\infty\right). \qquad \text{(B.16)}
\end{aligned}$$

Therefore, equation (B.16) implies that

$$\begin{aligned}
\ln(q_1) &\leq \ln(a_{q_1}) + \ln\left(z_\infty\left(1+\max\{1,p_{\max}\}\right)\right) + 4\ln(y_{\max}) + 3\ln(\|A\|_2) \\
&\quad + \ln\left(\|A\|_2^3 + \|A\|_\infty\right) + \ln\left(\|A\|_2 + \|A\|_\infty\right) \\
&\leq \ln(a_{q_1}) + \ln\left(z_\infty\left(1+\max\{1,p_{\max}\}\right)\right) + 4\ln(y_{\max}) + 3\ln(\|A\|_2) \\
&\quad + 3\ln(\|A\|_2) + \ln(\|A\|_\infty) + \ln(\|A\|_2) + \ln(\|A\|_\infty) \\
&\lesssim \ln(y_{\max}) + \ln(\|A\|_2) + \ln(\|A\|_\infty) \qquad \text{(B.17)}
\end{aligned}$$

where in the second inequality we used that $\ln(x+y) \leq \ln(x) + \ln(y)$ for $x, y \geq 2$ and took n and m large enough such that $\|A\|_2 \geq 2$ and $\|A\|_\infty \geq 2$.

Note that

$$\begin{aligned}
1 + \sqrt{\frac{\ln(3)}{\ln(2)}}\sqrt{\ln\left(4p_{\max}\right)} &= \mathcal{O}(1) \\
\sqrt{\frac{\ln(3)}{\ln(2)}}\sqrt{\ln(2KJ+1)} &= \mathcal{O}\left(\sqrt{\ln(KJ)}\right) \\
\sqrt{\frac{\ln(3)}{\ln(2)}}\sqrt{\ln(J)} &= \mathcal{O}\left(\sqrt{\ln(J)}\right) \\
\sqrt{\frac{\ln(3)}{\ln(2)}}\sqrt{\ln(K)} &= \mathcal{O}\left(\sqrt{\ln(K)}\right).
\end{aligned}$$

Furthermore, by equation (B.12)

$$\sqrt{\frac{\ln(3)}{\ln(2)}}\sqrt{K(J-1)}\sqrt{\ln(r_1)} = \mathcal{O}\left(\sqrt{K(J-1)\left(\ln(y_{\max}) + \ln(\|A\|_2) + \ln(\|A\|_\infty)\right)}\right)$$

and by equation (B.15)

$$\begin{aligned}
&\sqrt{\frac{\ln(3)}{\ln(2)}}\sqrt{K-1}\sqrt{\ln\left(r_1 + r_2 c_1 J\right)} \\
&\qquad = \mathcal{O}\left(\sqrt{(K-1)\left(\ln(y_{\max}) + \ln(\|A\|_2) + \ln(\|A\|_\infty) + \ln(J)\right)}\right).
\end{aligned}$$

Lastly, by equation (B.17)

$$\sqrt{\frac{\ln(3)}{\ln(2)}}\sqrt{\ln(q_1)} = \mathcal{O}\left(\sqrt{\ln(y_{\max}) + \ln(\|A\|_2) + \ln(\|A\|_\infty)}\right)$$

Thus, by Lemma B.1.1

$$\begin{aligned}
&\mathcal{O}\left(\sqrt{\ln\left(e\left(1+\frac{4p_{\max}(2KJ+1)\kappa}{c_{\max}}\right)\right)}\right) \\
&\quad= \mathcal{O}\Big(\max\Big\{1, \sqrt{\ln(KJ)}, \sqrt{\ln(J)}, \sqrt{\ln(K)}, \\
&\qquad\qquad \sqrt{K(J-1)\left(\ln(y_{\max}) + \ln(\|A\|_2) + \ln(\|A\|_\infty)\right)}, \\
&\qquad\qquad \sqrt{(K-1)\left(\ln(y_{\max}) + \ln(\|A\|_2) + \ln(\|A\|_\infty) + \ln(J)\right)}, \\
&\qquad\qquad \sqrt{\ln(y_{\max}) + \ln(\|A\|_2) + \ln(\|A\|_\infty)}\Big\}\Big) \\
&\quad= \mathcal{O}\left(\sqrt{K(J-1)\left(\ln(y_{\max}) + \ln(\|A\|_2) + \ln(\|A\|_\infty)\right)}\right). \qquad \text{(B.18)}
\end{aligned}$$

Term 2:

As in the analysis of Term 1, taking n, K, and J large enough such that $c_{\max} \geq 4p_{\max}$ and $4p_{\max}(2KJ+1)\kappa_{k,1}^{(j)} \geq 2c_{\max}$

$$\begin{aligned}
&\sqrt{\ln\left(e\left(1+\frac{4p_{\max}(2KJ+1)\kappa_{k,1}^{(j)}}{c_{\max}}\right)\right)} \\
&\quad= \sqrt{1+\ln\left(1+\frac{4p_{\max}(2KJ+1)\kappa_{k,1}^{(j)}}{c_{\max}}\right)} \\
&\quad\leq 1+\sqrt{\frac{\ln(3)}{\ln(2)}}\sqrt{\ln\left(\frac{4p_{\max}(2KJ+1)\kappa_{k,1}^{(j)}}{c_{\max}}\right)} \\
&\quad= 1+\sqrt{\frac{\ln(3)}{\ln(2)}}\left(\sqrt{\ln\left(\frac{4p_{\max}}{c_{\max}}\right)} + \sqrt{\ln(2KJ+1)} + \sqrt{\ln\left(\kappa_{k,1}^{(j)}\right)}\right) \\
&\quad\leq 1+\sqrt{\frac{\ln(3)}{\ln(2)}}\left(\sqrt{\ln\left(4p_{\max}\right)} + \sqrt{\ln(2KJ+1)} + \sqrt{\ln\left(\kappa_{k,1}^{(j)}\right)}\right).
\end{aligned}$$

By equation (B.5), note that

$$\begin{aligned}
\ln\left(\kappa_{k,1}^{(j)}\right) &= \ln\left(z_\infty(c_1 + p_{\max}y_{\max}||A||_\infty)\xi r_1^{J-j}\left(r_1^J + r_2\frac{1-r_1^J}{1-r_1}c_1\right)^{K-k}\right) \\
&= \ln(z_\infty(c_1 + p_{\max}y_{\max}||A||_\infty)\xi) + (J-j)\ln(r_1) \\
&\quad + (K-k)\ln\left(r_1^J + r_2\frac{1-r_1^J}{1-r_1}c_1\right) \\
&\leq \ln(z_\infty(c_1 + p_{\max}y_{\max}||A||_\infty)\xi) + (J-j)\ln(r_1) \\
&\quad + (K-k)\left((J-1)\ln(r_1) + \ln\left(r_1 + r_2c_1J\right)\right).
\end{aligned}$$

Therefore, using subadditivity of square roots

$$\begin{aligned}
&\sum_{k=1}^{K}\sum_{j=1}^{J}\sqrt{\ln\left(e\left(1+\frac{4p_{\max}(2KJ+1)\kappa_{k,1}^{(j)}}{c_{\max}}\right)\right)} \\
&\leq \sum_{k=1}^{K}\sum_{j=1}^{J}\left(1+\sqrt{\frac{\ln(3)}{\ln(2)}}\left(\sqrt{\ln\left(4p_{\max}\right)}+\sqrt{\ln(2KJ+1)}\right)\right) \\
&\quad + \sum_{k=1}^{K}\sum_{j=1}^{J}\sqrt{\frac{\ln(3)}{\ln(2)}}\left(\sqrt{\ln(z_\infty(c_1 + p_{\max}y_{\max}||A||_\infty)\xi)}+\sqrt{J-j}\sqrt{\ln(r_1)}\right) \\
&\quad + \sum_{k=1}^{K}\sum_{j=1}^{J}\sqrt{\frac{\ln(3)}{\ln(2)}}\sqrt{K-k}(\sqrt{J-1}\sqrt{\ln(r_1)}+\sqrt{\ln(r_1+r_2c_1J)}).
\end{aligned} \tag{B.19}$$

Since $\sqrt{x}$ is a convex function on $[0,\infty)$ then any left Riemann sum approximation of $\int_a^b \sqrt{x}dx$, for $a, b \geq 0$ is an under estimate. Hence, $\sum_{m=1}^{M}\sqrt{m} \leq \int_1^M \sqrt{x}dx$ and thus

$$\sum_{m=1}^{M}\sqrt{M-m} = \sum_{m=1}^{M-1}\sqrt{m} \leq \int_1^M \sqrt{x}dx = \frac{2}{3}M^{3/2} - \frac{3}{2} \leq \frac{2}{3}M^{3/2}. \tag{B.20}$$

Let $q_2 = 1+\sqrt{\frac{\ln(3)}{\ln(2)}}\left(\sqrt{\ln\left(4p_{\max}\right)}+\sqrt{\ln(z_\infty(c_1+p_{\max}y_{\max}||A||_\infty)\xi)}\right)$ be a constant. Combining equation (B.19) and equation (B.20) produces

$$\begin{aligned}
&\sum_{k=1}^{K}\sum_{j=1}^{J}\sqrt{\ln\left(e\left(1+\frac{4p_{\max}(2KJ+1)\kappa_{k,1}^{(j)}}{c_{\max}}\right)\right)} \\
&\quad\leq KJq_2 + KJ\sqrt{\frac{\ln(3)}{\ln(2)}}\sqrt{\ln(2KJ+1)} + KJ^{3/2}\frac{2}{3}\sqrt{\frac{\ln(3)}{\ln(2)}}\sqrt{\ln(r_1)} \\
&\quad+ K^{3/2}J\sqrt{J-1}\frac{2}{3}\sqrt{\frac{\ln(3)}{\ln(2)}}\sqrt{\ln(r_1)} + K^{3/2}J\frac{2}{3}\sqrt{\frac{\ln(3)}{\ln(2)}}\sqrt{\ln(r_1+r_2c_1J)}.
\end{aligned}$$

Defining $\widehat{q}_2 = 1+\sqrt{\frac{\ln(3)}{\ln(2)}}\sqrt{\ln\left(4p_{\max}\right)}$ Using equation (B.8) there exists a constant a_{q_2} such that

$$\begin{aligned}
q_2 &= \widehat{q}_2 + \sqrt{\frac{\ln(3)}{\ln(2)}}\sqrt{\ln(z_\infty(c_1+p_{\max}y_{\max}\|A\|_\infty)\xi)} \\
&= \widehat{q}_2 + \sqrt{\frac{\ln(3)}{\ln(2)}}\sqrt{\ln(z_\infty\xi)+\ln(c_1+p_{\max}y_{\max}\|A\|_\infty)} \\
&\leq \widehat{q}_2 + \sqrt{\frac{\ln(3)}{\ln(2)}}\sqrt{\ln(z_\infty\xi)+\ln(a_{q_2}y_{\max}\|A\|_2^3+p_{\max}y_{\max}\|A\|_\infty))} \\
&\leq \widehat{q}_2 + \sqrt{\frac{\ln(3)}{\ln(2)}}\sqrt{\ln(z_\infty\xi)+\ln(\max\{a_{q_2},p_{\max}\}y_{\max}(\|A\|_2^3+\|A\|_\infty))} \\
&\leq \widehat{q}_2 + \sqrt{\frac{\ln(3)}{\ln(2)}}\sqrt{\ln(z_\infty\xi)} \\
&\quad+ \sqrt{\frac{\ln(3)}{\ln(2)}}\sqrt{\ln(\max\{a_{q_2},p_{\max}\})+\ln(y_{\max})+\ln(\|A\|_2^3+\|A\|_\infty)} \\
&\leq \widehat{q}_2 + \sqrt{\frac{\ln(3)}{\ln(2)}}\left(\sqrt{\ln(z_\infty\xi)}+\sqrt{\ln(\max\{a_{q_2},p_{\max}\})}\right) \\
&\quad+ \sqrt{\frac{\ln(3)}{\ln(2)}}\sqrt{\ln(y_{\max})+3\ln(\|A\|_2)+\ln(\|A\|_\infty)} \qquad \text{(B.21)} \\
&\lesssim \sqrt{\ln(y_{\max})+\ln(\|A\|_2)+\ln(\|A\|_\infty)} \qquad \text{(B.22)}
\end{aligned}$$

where (B.21) results from using that $\ln(x+y) \leq \ln(x) + \ln(y)$ for $x, y \geq 2$ and taking n and m large enough such that $\|A\|_2^3 \geq 2$ and $\|A\|_\infty \geq 2$.

Therefore, by equation (B.22),

$$KJq_2 = \mathcal{O}(KJ\sqrt{\ln(y_{\max}) + \ln(\|A\|_2) + \ln(\|A\|_\infty)}).$$

Note that

$$KJ\sqrt{\frac{\ln(3)}{\ln(2)}}\sqrt{\ln(2KJ+1)} = \mathcal{O}\left(KJ\sqrt{\ln(KJ)}\right).$$

Furthermore, by equation (B.12)

$$\begin{aligned} KJ^{3/2}\frac{2}{3}\sqrt{\frac{\ln(3)}{\ln(2)}}\sqrt{\ln(r_1)} &= \mathcal{O}\left(KJ^{3/2}\sqrt{\ln(y_{\max}) + \ln(\|A\|_2) + \ln(\|A\|_\infty)}\right) \\ K^{3/2}J\sqrt{J-1}\frac{2}{3}\sqrt{\frac{\ln(3)}{\ln(2)}}\sqrt{\ln(r_1)} &= \mathcal{O}\left((KJ)^{3/2}\sqrt{\ln(y_{\max}) + \ln(\|A\|_2) + \ln(\|A\|_\infty)}\right). \end{aligned}$$

Finally, using equation (B.15)

$$\begin{aligned} &K^{3/2}J\frac{2}{3}\sqrt{\frac{\ln(3)}{\ln(2)}}\sqrt{\ln(r_1 + r_2c_1J)} \\ &\quad = \mathcal{O}\left(K^{3/2}J\sqrt{\ln(y_{\max}) + \ln(\|A\|_2) + \ln(\|A\|_\infty) + \ln(J)}\right). \end{aligned}$$

Thus, by Lemma B.1.1

$$\begin{aligned}
&\mathcal{O}\left(\sum_{k=1}^{K}\sum_{j=1}^{J}\sqrt{\ln\left(e\left(1+\frac{4p_{\max}(2KJ+1)\kappa_{k,1}^{(j)}}{c_{\max}}\right)\right)}\right)\\
&\quad=\mathcal{O}\Big(\max\Big\{KJ\sqrt{\ln(y_{\max})+\ln(\|A\|_2)+\ln(\|A\|_\infty)},\\
&\qquad\qquad KJ\sqrt{\ln(KJ)},\\
&\qquad\qquad KJ^{3/2}\sqrt{\ln(y_{\max})+\ln(\|A\|_2)+\ln(\|A\|_\infty)},\\
&\qquad\qquad (KJ)^{3/2}\sqrt{\ln(y_{\max})+\ln(\|A\|_2)+\ln(\|A\|_\infty)},\\
&\qquad\qquad K^{3/2}J\sqrt{\ln(y_{\max})+\ln(\|A\|_2)+\ln(\|A\|_\infty)+\ln(J)}\Big\}\Big)\\
&\quad=\mathcal{O}\left((KJ)^{3/2}\sqrt{\ln(y_{\max})+\ln(\|A\|_2)+\ln(\|A\|_\infty)}\right). \qquad \text{(B.23)}
\end{aligned}$$

Term 3:

Following the same steps shown in evaluating Term 2 of the GEB, where ξ is replaced by $p_{\max}h_{\max}$, we have

$$\begin{aligned}
&\sum_{k=1}^{K}\sum_{j=1}^{J}\sqrt{\ln\left(e\left(1+\frac{4\mu(2KJ+1)\kappa_{k,2}^{(j)}}{c_{\max}}\right)\right)}\\
&\quad\le KJq_3+KJ\sqrt{\frac{\ln(3)}{\ln(2)}}\sqrt{\ln(2KJ+1)}+KJ^{3/2}\frac{2}{3}\sqrt{\frac{\ln(3)}{\ln(2)}}\sqrt{\ln(r_1)}\\
&\quad+K^{3/2}J\sqrt{J-1}\frac{2}{3}\sqrt{\frac{\ln(3)}{\ln(2)}}\sqrt{\ln(r_1)}+K^{3/2}J\frac{2}{3}\sqrt{\frac{\ln(3)}{\ln(2)}}\sqrt{\ln(r_1+r_2c_1J)}
\end{aligned}$$

for $q_3=1+\sqrt{\frac{\ln(3)}{\ln(2)}}\left(\sqrt{\ln(4\mu)}+\sqrt{\ln(z_\infty(c_1+p_{\max}y_{\max}||A||_\infty)p_{\max}h_{\max})}\right)$. Therefore, again following the steps in evaluating Term 2, the third term of the GEB scales as

$$\begin{aligned}
&\mathcal{O}\left(\sum_{k=1}^{K}\sum_{j=1}^{J}\sqrt{\ln\left(e\left(1+\frac{4\mu(2KJ+1)\kappa_{k,2}^{(j)}}{c_{\max}}\right)\right)}\right)\\
&\quad=\mathcal{O}\left((KJ)^{3/2}\sqrt{\ln(y_{\max})+\ln(\|A\|_2)+\ln(\|A\|_\infty)}\right). \qquad \text{(B.24)}
\end{aligned}$$

Full Generalization Error Bound:

Combining the bound on each of the three terms – i.e. equation (B.18), equation (B.23), and equation (B.24) – we have

$$\begin{aligned}
\mathcal{L}(\widehat{\boldsymbol{c}}) &\leq \mathcal{L}_{\mathcal{S}}(\widehat{\boldsymbol{c}}) + \frac{8\tau c_{\max}}{\sqrt{N_s}}\sqrt{\ln\left(e\left(1+\frac{4p_{\max}(2KJ+1)\kappa}{c_{\max}}\right)\right)} \\
&\quad + \frac{8\tau c_{\max}}{\sqrt{N_s}}\sum_{k=1}^{K}\sum_{j=1}^{J}\sqrt{\frac{n(n+1)}{2}}\sqrt{\ln\left(e\left(1+\frac{4p_{\max}(2KJ+1)\kappa_{k,1}^{(j)}}{c_{\max}}\right)\right)} \\
&\quad + \frac{8\tau c_{\max}}{\sqrt{N_s}}\sum_{k=1}^{K}\sum_{j=1}^{J}\sqrt{\ln\left(e\left(1+\frac{4\mu(2KJ+1)\kappa_{k,2}^{(j)}}{c_{\max}}\right)\right)} \\
&\quad + 8\sqrt{2\ln(4/\varepsilon)/N_s} \\
&= \mathcal{L}_{\mathcal{S}}(\widehat{\boldsymbol{c}}) + \mathcal{O}\left(c_{\max}\sqrt{\frac{K(J-1)\left(\ln(y_{\max})+\ln(\|A\|_2)+\ln(\|A\|_\infty)\right)}{N_s}}\right) \\
&\quad + \mathcal{O}\left(c_{\max}\sqrt{\frac{n(n+1)}{2}}\sqrt{\frac{(KJ)^3(\ln(y_{\max})+\ln(\|A\|_2)+\ln(\|A\|_\infty))}{N_s}}\right) \\
&\quad + \mathcal{O}\left(c_{\max}\sqrt{\frac{(KJ)^3(\ln(y_{\max})+\ln(\|A\|_2)+\ln(\|A\|_\infty))}{N_s}}\right) \\
&\quad + 4c\sqrt{2\ln(4/\varepsilon)/N_s}.
\end{aligned}$$

As $\sqrt{x+y} \leq \sqrt{x}+\sqrt{y} \leq 2\sqrt{x+y}$ for all $x, y \geq 0$ then the CG-Net generalization error bound is given asymptotically by

$$\begin{aligned}
&|\mathcal{L}(\widehat{\boldsymbol{c}}) - \mathcal{L}_{\mathcal{S}}(\widehat{\boldsymbol{c}})| \\
&\quad \lesssim c_{\max}\sqrt{\frac{\left(K(J-1)+\left(1+\frac{n(n+1)}{2}\right)(KJ)^3\right)\left(\ln(y_{\max})+\ln(\|A\|_2)+\ln(\|A\|_\infty)\right)}{N_s}}.
\end{aligned} \tag{B.25}$$

Using Lemma B.1.1 and removing all but the highest order terms – i.e. highest power terms of the variables K, J, m, or n – in (B.25)

$$|\mathcal{L}(\widehat{\boldsymbol{c}}) - \mathcal{L}_{\mathcal{S}}(\widehat{\boldsymbol{c}})| \lesssim c_{\max}\sqrt{\frac{n^2(KJ)^3(\ln(y_{\max}) + \ln(\|A\|_2) + \ln(\|A\|_\infty))}{N_s}}$$
$$= \mathcal{O}\left(\sqrt{\frac{c_{\max}^2 n^2(KJ)^3(\ln(y_{\max}) + \ln(\|A\|_2) + \ln(\|A\|_\infty))}{N_s}}\right). \tag{B.26}$$

Finally, let $c_{\max} = \mathcal{O}(\sqrt{n}), y_{\max} = \mathcal{O}(\sqrt{m})$, $\|A\|_2 = \mathcal{O}(\sqrt{mn})$, and $\|A\|_\infty = \mathcal{O}(n)$ then equation (B.26) reduces to

$$|\mathcal{L}(\widehat{\boldsymbol{c}}) - \mathcal{L}_{\mathcal{S}}(\widehat{\boldsymbol{c}})| \lesssim \sqrt{n}\sqrt{\frac{n^2(KJ)^3(\ln(\sqrt{m}) + \ln(\sqrt{m}n) + \ln(n))}{N_s}}$$
$$\lesssim \sqrt{\frac{n^3(KJ)^3(\ln(m) + \ln(n))}{N_s}}$$
$$= \mathcal{O}\left(\sqrt{\frac{n^3(\ln(m) + \ln(n))(KJ)^3}{N_s}}\right). \tag{B.27}$$

producing the desired asymptotic forms for the CG-Net generalization error. ⊠

B.3 Independent Measurement Matrix Norm

We provide a proof of Corollary 4.5.3 where $c_{\max}$ and $y_{\max}$ depend on the measurement and signal of interest size. Furthermore, $\|A\|_2$ and $\|A\|_\infty$ are independent of measurement and signal of interest size.

Proof of Corollary 4.5.3. Note that $c_1 \lesssim y_{\max}$, $c_2 \lesssim y_{\max}$, $r_1 \lesssim y_{\max}^2$, and $r_2 \lesssim y_{\max}$. Therefore $\ln(r_1) \lesssim \ln(y_{\max})$ and $\ln(r_1 + r_2 c_1 J) \lesssim \ln(y_{\max}) + \ln(J)$. Further, by equation (B.16) and equation (B.17), q_1 from the proof of Corollary 4.5.2 satisfies $\ln(q_1) \lesssim \ln(y_{\max})$. Next, by equation (B.22, q_2 from the proof of Corollary 4.5.2 satisfies $q_2 \lesssim \sqrt{\ln(y_{\max})}$. Similarly, q_3 from the proof of Corollary 4.5.2 satisfies $q_3 \lesssim \sqrt{\ln(y_{\max})}$.

Hence, following the proof of Corollary 4.5.2 but replacing the bounds on $c_1, c_2, r_1, r_2, \ln(r_1 + r_2 c_1 J), q_1, q_2$, and q_3 with those in the above paragraph gives

$$\mathcal{O}\left(\sqrt{\ln\left(e\left(1+\frac{4p_{\max}(2KJ+1)\kappa}{c_{\max}}\right)\right)}\right) = \mathcal{O}\left(\sqrt{K(J-1)\ln(y_{\max})}\right)$$

$$\mathcal{O}\left(\sum_{k=1}^{K}\sum_{j=1}^{J}\sqrt{\ln\left(e\left(1+\frac{4p_{\max}(2KJ+1)\kappa_{k,d}^{(j)}}{c_{\max}}\right)\right)}\right) = \mathcal{O}\left((KJ)^{3/2}\sqrt{\ln(y_{\max})}\right)$$

for $d = 1, 2$.

Combining these bounds with the CG-Net generalization error in Theorem 4.5.1 gives

$$\begin{aligned}
\mathcal{L}(\widehat{\boldsymbol{c}}) \leq \mathcal{L}_{\mathcal{S}}(\widehat{\boldsymbol{c}}) &+ \frac{8\tau c_{\max}}{\sqrt{N_s}}\sqrt{\ln\left(e\left(1+\frac{4p_{\max}(2KJ+1)\kappa}{c_{\max}}\right)\right)} \\
&+ \frac{8\tau c_{\max}}{\sqrt{N_s}}\sum_{k=1}^{K}\sum_{j=1}^{J}\sqrt{\frac{n(n+1)}{2}}\sqrt{\ln\left(e\left(1+\frac{4p_{\max}(2KJ+1)\kappa_{k,1}^{(j)}}{c_{\max}}\right)\right)} \\
&+ \frac{8\tau c_{\max}}{\sqrt{N_s}}\sum_{k=1}^{K}\sum_{j=1}^{J}\sqrt{\ln\left(e\left(1+\frac{4\mu(2KJ+1)\kappa_{k,2}^{(j)}}{c_{\max}}\right)\right)} \\
&+ 8\sqrt{2\ln(4/\varepsilon)/N_s} \\
\mathcal{L}(\widehat{\boldsymbol{c}}) \leq \mathcal{L}_{\mathcal{S}}(\widehat{\boldsymbol{c}}) &+ \mathcal{O}\left(c_{\max}\sqrt{\frac{K(J-1)\ln(y_{\max})}{N_s}}\right) + \mathcal{O}\left(c_{\max}\sqrt{\frac{n(n+1)}{2}}\sqrt{\frac{(KJ)^3\ln(y_{\max})}{N_s}}\right) \\
&+ \mathcal{O}\left(c_{\max}\sqrt{\frac{(KJ)^3\ln(y_{\max})}{N_s}}\right) + 8\sqrt{2\ln(4/\varepsilon)/N_s}.
\end{aligned}$$

As $\sqrt{x+y} \leq \sqrt{x} + \sqrt{y} \leq 2\sqrt{x+y}$ for all $x, y \geq 0$ then the CG-Net generalization error bound is given asymptotically by

$$|\mathcal{L}(\widehat{\boldsymbol{c}}) - \mathcal{L}_{\mathcal{S}}(\widehat{\boldsymbol{c}})| \lesssim c_{\max}\sqrt{\frac{\left(K(J-1) + \left(1+\frac{n(n+1)}{2}\right)(KJ)^3\right)\ln(y_{\max})}{N_s}}. \tag{B.28}$$

Using Lemma B.1.1 and removing all but the highest order terms – i.e. highest power terms of the variables K, J, m, or n – in (B.28)

$$|\mathcal{L}(\widehat{\boldsymbol{c}}) - \mathcal{L}_{\mathcal{S}}(\widehat{\boldsymbol{c}})| \lesssim c_{\max}\sqrt{\frac{n^2(KJ)^3\ln(y_{\max})}{N_s}} = \mathcal{O}\left(\sqrt{\frac{c_{\max}^2 n^2(KJ)^3\ln(y_{\max})}{N_s}}\right). \tag{B.29}$$

Finally, let $c_{\max} = \mathcal{O}(\sqrt{n})$ and $y_{\max} = \mathcal{O}(\sqrt{m})$ then equation (B.29) reduces to

$$\begin{aligned}|\mathcal{L}(\widehat{\boldsymbol{c}}) - \mathcal{L}_{\mathcal{S}}(\widehat{\boldsymbol{c}})| &\lesssim \sqrt{n}\sqrt{\frac{n^2(KJ)^3\ln(\sqrt{m})}{N_s}} \\ &\lesssim \sqrt{n}\sqrt{\frac{n^2(KJ)^3\ln(m)}{N_s}} = \mathcal{O}\left(\sqrt{\frac{n^3(KJ)^3\ln(m)}{N_s}}\right).\end{aligned} \tag{B.30}$$

producing the desired asymptotic forms for the CG-Net generalization error. ⊠

B.4 Independence of Signal Size

We provide a proof of Corollary 4.5.4 where $c_{\max}$, $y_{\max}$, $\|A\|_2$, and $\|A\|_\infty$ are independent on the measurement and signal of interest size.

Proof of Corollary 4.5.4. Note that $c_1 \lesssim 1$, $c_2 \lesssim 1$, $r_1 \lesssim 1$, and $r_2 \lesssim 1$. Therefore $\ln(r_1) \lesssim 1$ and $\ln(r_1 + r_2c_1J) \lesssim \ln(J)$. Further, by equation (B.16) and equation (B.17), q_1 from the proof of Corollary 4.5.2 satisfies $\ln(q_1) \lesssim 1$. Next, by equation (B.22, q_2 from the proof of Corollary 4.5.2 satisfies $q_2 \lesssim 1$. Similarly, q_3 from the proof of Corollary 4.5.2 satisfies $q_3 \lesssim 1$.

Hence, following the proof of Corollary 4.5.2 but replacing the bounds on $c_1, c_2, r_1, r_2, \ln(r_1 + r_2c_1J), q_1, q_2$, and q_3 with those in the above paragraph gives

$$\mathcal{O}\left(\sqrt{\ln\left(e\left(1 + \frac{4p_{\max}(2KJ+1)\kappa}{c_{\max}}\right)\right)}\right) = \mathcal{O}\left(\sqrt{K(J-1)}\right)$$

$$\mathcal{O}\left(\sum_{k=1}^{K}\sum_{j=1}^{J}\sqrt{\ln\left(e\left(1 + \frac{4p_{\max}(2KJ+1)\kappa_{k,d}^{(j)}}{c_{\max}}\right)\right)}\right) = \mathcal{O}\left((KJ)^{3/2}\right)$$

for $d = 1, 2$.

Combining these bounds with the CG-Net generalization error in Theorem 4.5.1 gives

$$\begin{aligned}
\mathcal{L}(\widehat{\boldsymbol{c}}) \leq \mathcal{L}_{\mathcal{S}}(\widehat{\boldsymbol{c}}) &+ \frac{8\tau c_{\max}}{\sqrt{N_s}} \sqrt{\ln\left(e\left(1 + \frac{4p_{\max}(2KJ+1)\kappa}{c_{\max}}\right)\right)} \\
&+ \frac{8\tau c_{\max}}{\sqrt{N_s}} \sum_{k=1}^{K}\sum_{j=1}^{J} \sqrt{\frac{n(n+1)}{2}} \sqrt{\ln\left(e\left(1 + \frac{4p_{\max}(2KJ+1)\kappa_{k,1}^{(j)}}{c_{\max}}\right)\right)} \\
&+ \frac{8\tau c_{\max}}{\sqrt{N_s}} \sum_{k=1}^{K}\sum_{j=1}^{J} \sqrt{\ln\left(e\left(1 + \frac{4\mu(2KJ+1)\kappa_{k,2}^{(j)}}{c_{\max}}\right)\right)} \\
&+ 8\sqrt{2\ln(4/\varepsilon)/N_s} \\
\mathcal{L}(\widehat{\boldsymbol{c}}) \leq \mathcal{L}_{\mathcal{S}}(\widehat{\boldsymbol{c}}) &+ \mathcal{O}\left(\sqrt{\frac{K(J-1)}{N_s}}\right) + \mathcal{O}\left(\sqrt{\frac{n(n+1)}{2}}\sqrt{\frac{(KJ)^3}{N_s}}\right) + \mathcal{O}\left(\sqrt{\frac{(KJ)^3}{N_s}}\right) \\
&+ 8\sqrt{2\ln(4/\varepsilon)/N_s}.
\end{aligned}$$

As $\sqrt{x+y} \leq \sqrt{x} + \sqrt{y} \leq 2\sqrt{x+y}$ for all $x, y \geq 0$ then the CG-Net generalization error bound is given asymptotically by

$$|\mathcal{L}(\widehat{\boldsymbol{c}}) - \mathcal{L}_{\mathcal{S}}(\widehat{\boldsymbol{c}})| \lesssim \sqrt{\frac{K(J-1) + \left(1 + \frac{n(n+1)}{2}\right)(KJ)^3}{N_s}}. \tag{B.31}$$

Using Lemma B.1.1 and removing all but the highest order terms – i.e. highest power terms of the variables K, J, m, or n – in (B.31)

$$|\mathcal{L}(\widehat{\boldsymbol{c}}) - \mathcal{L}_{\mathcal{S}}(\widehat{\boldsymbol{c}})| \lesssim \sqrt{\frac{n^2(KJ)^3}{N_s}} = \mathcal{O}\left(\sqrt{\frac{n^2(KJ)^3}{N_s}}\right) \tag{B.32}$$

producing the desired asymptotic forms for the CG-Net generalization error. ⊠

Appendix C

PGD DR-CG-Net Asymptotic Generalization Error

In this appendix, we supply a detailed construction of the asymptotic form for the generalization error bound of PGD DR-CG-Net stated in Corollary 4.6.4 where $c_{\max}, y_{\max}, \|A\|_2$, and $\|A\|_\infty$ depend on the measurement and signal of interest size. To prove Corollary 4.6.5 the exact steps in the proof of Corollary 4.6.4 below can be followed with $\|A\|_2$ and $\|A\|_\infty$ set to 1. Similarly, to prove Corollary 4.6.6 the steps to prove Corollary 4.6.4 below can be followed with $c_{\max}, y_{\max}, \|A\|_2$, and $\|A\|_\infty$ set at 1.

Proof of Corollary 4.6.4. First recall that the GEB is

$$\begin{aligned}\mathcal{L}(\widehat{\boldsymbol{c}}) \leq \mathcal{L}_{\mathcal{S}}(\widehat{\boldsymbol{c}}) &+ \frac{8c_{\max}}{\sqrt{nN_s}}\sqrt{2n-1}\sqrt{\ln\left(e\left(1+\frac{4p_{\max}(KJ(L_c+1)+1)\kappa}{c_{\max}}\right)\right)} \\ &+ \frac{8c_{\max}}{\sqrt{nN_s}}\sum_{k=1}^{K}\sum_{j=1}^{J}\sum_{d=1}^{L_c}\sqrt{f_{d-1}f_d k_d^2}\sqrt{\ln\left(e\left(1+\frac{4w_d(KJ(L_c+1)+1)\kappa_{k,d}^{(j)}}{c_{\max}}\right)\right)} \\ &+ \frac{8c_{\max}}{\sqrt{nN_s}}\sum_{k=1}^{K}\sum_{j=1}^{J}\sqrt{\ln\left(e\left(1+\frac{4\delta(KJ(L_c+1)+1)\kappa_{k,L_c+1}^{(j)}}{c_{\max}}\right)\right)} \\ &+ 4c_{\max}\sqrt{2\ln(4/\varepsilon)/(nN_s)}\end{aligned}$$

for

$$\begin{aligned}\kappa &= z_\infty(c_1 + p_{\max}y_{\max}\|A\|_\infty)\widehat{c}_1^{(K,J)} + z_\infty c_2 \\ \kappa_{k,d}^{(j)} &= z_\infty(c_1 + p_{\max}y_{\max}\|A\|_\infty)\widehat{c}_{k,j,d,2}^{(K,J)}\end{aligned}$$

where

$$
\begin{aligned}
c_1 &= p_{\max} y_{\max} \ \|A\|_2 \left(1 + 2 z_\infty^2 p_{\max} \ \|A\|_2^2\right) \\
c_2 &= z_\infty y_{\max} \|A\|_2 \left(p_{\max}/p_{\min}\right)^2 \\
\widehat{c}_1^{(K,J)} &= c_2 r_2 \left(\frac{1 - r_1^J}{1 - r_1}\right) \left(\frac{1 - \left(r_1^J + c_1 r_2 \frac{1 - r_1^J}{1 - r_1}\right)^K}{1 - \left(r_1^J + c_1 r_2 \frac{1 - r_1^J}{1 - r_1}\right)}\right) \\
\widehat{c}_{k,j,d,2}^{(K,J)} &= r_1^{J-j} \left(r_1^J + c_1 r_2 \frac{1 - r_1^J}{1 - r_1}\right)^{K-k} \begin{cases} \left(\sqrt{n} z_\infty \prod_{\substack{\ell=1 \\ \ell \neq d}}^{L_c} w_\ell\right) & d = 1, 2, \ldots, L_c \\ \xi & d = L_c + 1 \end{cases}
\end{aligned}
$$

for

$$
\begin{aligned}
r_1 &= 1 + \delta (z_\infty p_{\max} y_{\max} \|A\|_2 \|A\|_\infty)^2 + \prod_{\ell=1}^{L_c} w_\ell \\
r_2 &= \delta y_{\max} \|A\|_2 \left(1 + z_\infty^2 p_{\max} \|A\|_2 (\|A\|_2 + \|A\|_\infty)\right).
\end{aligned}
$$

We will consider each term on the right hand side of the PGD DR-CG-Net GEB separately and provide a Big O bound for each. Subsequently, using Lemma B.1.1 we can combine each of the three separate bounds into a single bound by taking a maximum. Note that the empirical loss term $\mathcal{L}_{\mathcal{S}}(\widehat{c})$ is independent of the network size of PGD DR-CG-Net and we will assume it is independent of the reconstructed signal size allowing us to ignore this term for the asymptotic analysis. Hence, we have four terms to consider individually; three terms structured in the form of $\ln(e(1 + \widetilde{c}(KJ(L_c + 1) + 1)\widetilde{\kappa}$, for constant $\widetilde{c}$ and variable $\widetilde{\kappa}$ dependent on signal and network size, and the error term $4c_{\max}\sqrt{2\ln(4/\varepsilon)/(nN_s)}$.

Define the function $\mathcal{G} : \mathbb{R}^{m \times n} \times \mathbb{R} \to \mathbb{R}$ by

$$
\mathcal{G}(A, L_c) \coloneqq \ln(y_{\max}) + \ln(\|A\|_2) + \ln(\|A\|_\infty) + \ln\left(1 + \prod_{d=1}^{L_c} w_d\right). \tag{C.1}
$$

As the function $\mathcal{G}$ will be present in the first three terms we analyze below, it will be useful in simplifying the asymptotic bounds for each term.

Term 1:

As in the proof of Corollary 4.5.2 it holds that

$$\begin{aligned}
&\sqrt{\ln\left(e\left(1+\frac{4p_{\max}(KJ(L_c+1)+1)\kappa}{c_{\max}}\right)\right)} \\
&\quad\leq 1+\sqrt{\frac{\ln(3)}{\ln(2)}}\Big(\sqrt{\ln\left(4p_{\max}\right)}+\sqrt{\ln\left(KJ(L_c+1)+1\right)}+\sqrt{\ln(J)}+\sqrt{\ln(K)} \\
&\qquad+\sqrt{K(J-1)}\sqrt{\ln(r_1)}+\sqrt{K-1}\sqrt{\ln\left(r_1+r_2c_1J\right)}+\sqrt{\ln(q_1)}\Big),
\end{aligned}$$

when taking n, K, J, and L_c large enough such that $c_{\max} \geq 4p_{\max}$ and $4p_{\max}(KJ(L_c+1)+1)\kappa \geq 2c_{\max}$, and for $q_1 = z_\infty(c_1 + p_{\max}y_{\max}||A||_\infty)c_2r_2 + z_\infty c_2$.

Now, let $a = 1+\delta(z_\infty p_{\max}y_{\max}\|A\|_2\|A\|_\infty)^2$. As $a \geq 1$

$$\ln(r_1) = \ln\left(a+\prod_{d=1}^{L_c}w_d\right) = \ln(a)+\ln\left(1+\prod_{d=1}^{L_c}w_d/a\right) \leq \ln(a)+\ln\left(1+\prod_{d=1}^{L_c}w_d\right)$$

and

$$\ln\left(r_1+r_2c_1J\right) = \ln\left(a+r_2c_1J+\prod_{d=1}^{L_c}w_d\right) \leq \ln(a+r_2c_1J)+\ln\left(1+\prod_{d=1}^{L_c}w_d\right).$$

By equation (B.12), $\ln(a) \lesssim \ln(y_{\max})+\ln(\|A\|_2)+\ln(\|A\|_\infty)$ and thus

$$\ln(r_1) \lesssim \ln(y_{\max})+\ln(\|A\|_2)+\ln(\|A\|_\infty)+\ln\left(1+\prod_{d=1}^{L_c}w_d\right) = \mathcal{G}(A,L_c). \tag{C.2}$$

Similarly, by equation (B.15), $\ln(a+r_2c_1J) \lesssim \ln(y_{\max})+\ln(\|A\|_2)+\ln(\|A\|_\infty)+\ln(J)$ and thus

$$\begin{aligned}
\ln\left(r_1+r_2c_1J\right) &\lesssim \ln(y_{\max})+\ln(\|A\|_2)+\ln(\|A\|_\infty)+\ln(J)+\ln\left(1+\prod_{d=1}^{L_c}w_d\right) \\
&= \mathcal{G}(A,L_c)+\ln(J).
\end{aligned} \tag{C.3}$$

Additionally, by equation (B.17)

$$\ln(q_1) \lesssim \ln(y_{\max}) + \ln(\|A\|_2) + \ln(\|A\|_\infty). \tag{C.4}$$

Note that

$$1 + \sqrt{\frac{\ln(3)}{\ln(2)}}\sqrt{\ln\left(4p_{\max}\right)} = \mathcal{O}(1)$$

$$\sqrt{\frac{\ln(3)}{\ln(2)}}\sqrt{\ln(KJ(L_c+1)+1)} = \mathcal{O}\left(\sqrt{\ln(KJL_c)}\right)$$

$$\sqrt{\frac{\ln(3)}{\ln(2)}}\sqrt{\ln(J)} = \mathcal{O}\left(\sqrt{\ln(J)}\right)$$

$$\sqrt{\frac{\ln(3)}{\ln(2)}}\sqrt{\ln(K)} = \mathcal{O}\left(\sqrt{\ln(K)}\right).$$

Furthermore, by equation (C.2)

$$\sqrt{\frac{\ln(3)}{\ln(2)}}\sqrt{K(J-1)}\sqrt{\ln(r_1)} = \mathcal{O}\left(\sqrt{K(J-1)\mathcal{G}(A,L_c)}\right)$$

and by equation (C.3)

$$\sqrt{\frac{\ln(3)}{\ln(2)}}\sqrt{K-1}\sqrt{\ln(r_1+r_2c_1J)} = \mathcal{O}\left(\sqrt{(K-1)\left(\mathcal{G}(A,L_c)+\ln(J)\right)}\right).$$

Lastly, by equation (C.4)

$$\sqrt{\frac{\ln(3)}{\ln(2)}}\sqrt{\log(q_1)} = \mathcal{O}\left(\sqrt{\ln(y_{\max})+\ln(\|A\|_2)+\ln(\|A\|_\infty)}\right).$$

Thus, using Lemma B.1.1

$$\begin{aligned}
&\sqrt{\ln\left(e\left(1+\frac{4p_{\max}(KJ(L_c+1)+1)\kappa}{c_{\max}}\right)\right)} \\
&\qquad = \mathcal{O}\left(\max\left\{1, \sqrt{\ln(KJL_c)}, \sqrt{\ln(J)}, \sqrt{\ln(K)}, \sqrt{K(J-1)\mathcal{G}(A,L_c)},\right.\right. \\
&\qquad\qquad \left.\left.\sqrt{(K-1)\left(\mathcal{G}(A,L_c)+\ln(J)\right)}, \sqrt{\ln(y_{\max})+\ln(\|A\|_2)+\ln(\|A\|_\infty)}\right\}\right) \\
&\qquad = \mathcal{O}\left(\sqrt{\max\left\{\ln(KJL_c), K(J-1)\mathcal{G}(A,L_c)\right\}}\right). \qquad \text{(C.5)}
\end{aligned}$$

Furthermore, let $w_{\max} = \max_{1\le d\le L_c} w_d$. Without loss of generality, we assume that $w_{\max} > 0$, otherwise every convolutional sub-network in DR-CG-Net would be using a matrix kernel of all zeros thereby mapping all inputs to zero. By the binomial theorem for any $x, y \in \mathbb{R}$ and any $k \in \mathbb{N}$

$$(x+y)^k = \sum_{j=0}^{k}\binom{k}{j}x^{k-j}y^j \ge \binom{k}{0}x^k + \binom{k}{k}y^k = x^k + y^k$$

implying $1 + x^k \le (1+x)^k$. Thus

$$\ln\left(1+\prod_{d=1}^{L_c} w_d\right) \le \ln\left(1+w_{\max}^{L_c}\right) \le \ln((1+w_{\max})^{L_c}) = L_c\ln(1+w_{\max}) \lesssim L_c,$$

which implies

$$\mathcal{G}(A,L_c) \lesssim \ln(y_{\max}) + \ln(\|A\|_2) + \ln(\|A\|_\infty) + L_c.$$

Combining with equation (C.5) and noting that $\ln(KJL_c) \le K(J-1)(\ln(y_{\max}) + \ln(\|A\|_2) + \ln(\|A\|_\infty) + L_c)$ produces

$$\begin{aligned}
&\mathcal{O}\left(\sqrt{\ln\left(e\left(1+\frac{4p_{\max}(KJ(L_c+1)+1)\kappa}{c_{\max}}\right)\right)}\right) \\
&\qquad = \mathcal{O}\left(\sqrt{K(J-1)\left(\ln(y_{\max})+\ln(\|A\|_2)+\ln(\|A\|_\infty)+L_c\right)}\right). \qquad \text{(C.6)}
\end{aligned}$$

Term 2:

Take n large enough such that $c_{\max} \geq 1$ and K, J and L_c large enough such that $4w_d(KJ(L_c+1)+1) \geq 1$. Then, using subadditivity of the square root function, observe

$$\begin{aligned}
&\sqrt{\ln\left(e\left(1+\frac{4w_d(KJ(L_c+1)+1)\kappa_{k,d}^{(j)}}{c_{\max}}\right)\right)} \\
&\qquad \leq \sqrt{\ln\left(e\left(1+4w_d(KJ(L_c+1)+1)\kappa_{k,d}^{(j)}\right)\right)} \\
&\qquad = \sqrt{1+\ln\left(1+4w_d(KJ(L_c+1)+1)\kappa_{k,d}^{(j)}\right)} \\
&\qquad \leq 1+\sqrt{\ln\left(1+4w_d(KJ(L_c+1)+1)\kappa_{k,d}^{(j)}\right)} \\
&\qquad \leq 1+\sqrt{\ln\left(4w_d(KJ(L_c+1)+1)\left(1+\kappa_{k,d}^{(j)}\right)\right)} \\
&\qquad = 1+\sqrt{\ln(4w_d)+\ln(KJ(L_c+1)+1)+\ln(1+\kappa_{k,d}^{(j)})} \\
&\qquad \leq 1+\sqrt{\ln(4w_d)}+\sqrt{\ln(KJ(L_c+1)+1)}+\sqrt{\ln(1+\kappa_{k,d}^{(j)})}.
\end{aligned}$$

Define $\eta_d = \prod_{\substack{\ell=1 \\ \ell\neq d}}^{L_c} w_\ell$. Take n, K, and J large enough such that

$$\sqrt{n}z_\infty^2(c_1+p_{\max}y_{\max}\|A\|_\infty)r_1^{J-j}\left(r_1^J+c_1r_2\frac{1-r_1^J}{1-r_1}\right)^{K-k} \geq 1.$$

Then for every $d \in \mathbb{N}[L_c]$, by equation (B.5), it holds that

$$\begin{aligned}
&\ln\left(1+\kappa_{k,d}^{(j)}\right) \\
&= \ln\left(1+\sqrt{n}z_\infty^2(c_1+p_{\max}y_{\max}\|A\|_\infty)r_1^{J-j}\left(r_1^J+c_1r_2\frac{1-r_1^J}{1-r_1}\right)^{K-k}\eta_d\right) \\
&\leq \ln\left(\left(\sqrt{n}z_\infty^2(c_1+p_{\max}y_{\max}\|A\|_\infty)r_1^{J-j}\left(r_1^J+c_1r_2\frac{1-r_1^J}{1-r_1}\right)^{K-k}\right)(1+\eta_d)\right) \\
&= \ln(\sqrt{n})+\ln(q_2)+(J-j)\ln(r_1)+(K-k)\ln\left(r_1^J+c_1r_2\frac{1-r_1^J}{1-r_1}\right)+\ln(1+\eta_d) \\
&\leq \ln(\sqrt{n})+\ln(q_2)+(J-j)\ln(r_1)+(K-k)\left((J-1)\ln(r_1)+\ln(r_1+r_2c_1J)\right)+\ln(1+\eta_d)
\end{aligned}$$

where $q_2 = z_\infty^2(c_1 + p_{\max} y_{\max} \|A\|_\infty)$.

Hence

$$\begin{aligned}
&\sqrt{\ln\left(e\left(1+\frac{4w_d(KJ(L_c+1)+1)\kappa_{k,d}^{(j)}}{c_{\max}}\right)\right)} \\
&\quad\le 1+\sqrt{\ln(4w_d)}+\sqrt{\ln(KJ(L_c+1)+1)}+\sqrt{\ln(\sqrt{n})}+\sqrt{\ln(q_2)}+\sqrt{(J-j)}\sqrt{\ln(r_1)} \\
&\qquad+\sqrt{(K-k)}\sqrt{(J-1)}\sqrt{\ln(r_1)}+\sqrt{(K-k)}\sqrt{\ln(r_1+r_2c_1J)}+\sqrt{\ln(1+\eta_d)}.
\end{aligned}$$

Therefore, by equation (B.20)

$$\begin{aligned}
&\sum_{k=1}^{K}\sum_{j=1}^{J}\sqrt{\ln\left(e\left(1+\frac{4w_d(KJ(L_c+1)+1)\kappa_{k,d}^{(j)}}{c_{\max}}\right)\right)} \\
&\quad\le KJ\left(1+\sqrt{\ln(4w_d)}\right)+KJ\sqrt{\ln(KJ(L_c+1)+1)}+KJ\sqrt{\ln(\sqrt{n})}+KJ\sqrt{\ln(q_2)} \\
&\qquad+\frac{2}{3}KJ^{3/2}\sqrt{\ln(r_1)}+\frac{2}{3}K^{3/2}J\sqrt{J-1}\sqrt{\ln(r_1)} \\
&\qquad+\frac{2}{3}K^{3/2}J\sqrt{\ln\left(r_1+r_2c_1J\right)}+KJ\sqrt{\ln\left(1+\eta_d\right)}.
\end{aligned}$$

Define $s_{L_c} = \sum_{d=1}^{L_c}\sqrt{f_{d-1}f_d k_d^2}$, then

$$\begin{aligned}
&\sum_{k=1}^{K}\sum_{j=1}^{J}\sum_{d=1}^{L_c}\sqrt{f_{d-1}f_dk_d^2}\sqrt{\ln\left(e\left(1+\frac{4w_d(KJ(L_c+1)+1)\kappa_{k,d}^{(j)}}{c_{\max}}\right)\right)} \\
&\quad\le KJ\left(s_{L_c}+\sum_{d=1}^{L_c}\sqrt{f_{d-1}f_dk_d^2}\sqrt{\ln(4w_d)}\right)+KJs_{L_c}\sqrt{\ln(KJ(L_c+1)+1)} \\
&\qquad+KJs_{L_c}\sqrt{\ln(\sqrt{n})}+KJs_{L_c}\sqrt{\ln(q_2)} \\
&\qquad+\frac{2}{3}KJ^{3/2}s_{L_c}\sqrt{\ln(r_1)}+\frac{2}{3}K^{3/2}Js_{L_c}\sqrt{J-1}\sqrt{\ln(r_1)} \\
&\qquad+\frac{2}{3}K^{3/2}Js_{L_c}\sqrt{\ln\left(r_1+r_2c_1J\right)}+KJ\sum_{d=1}^{L_c}\sqrt{f_{d-1}f_dk_d^2}\sqrt{\ln\left(1+\eta_d\right)}.
\end{aligned}$$

Now, define $\widetilde{w}_{\max} = \max\{1, \max_{1\le d\le L_c} w_d\}$. Note that $\sqrt{x+y} \le \sqrt{x}+\sqrt{y} \le 2\sqrt{x+y}$ for all $x, y \ge 0$. Thus, for sufficiently large m, n, K and J such that $4\widetilde{w}_{\max} \le y_{\max}\|A\|_2\|A\|_\infty$ and

$4 \leq KJ$, observe that

$$
\begin{aligned}
&\sum_{d=1}^{L_c} \sqrt{f_{d-1} f_d k_d^2 \ln(4w_d)} + \sum_{d=1}^{L_c} \sqrt{f_{d-1} f_d k_d^2 \ln\left(1+\eta_d\right)} \\
&\quad = \sum_{d=1}^{L_c} \sqrt{f_{d-1} f_d k_d^2 \ln(4w_d)} + \sum_{d=1}^{L_c} \sqrt{f_{d-1} f_d k_d^2 \ln\left(1 + \prod_{\substack{\ell=1 \\ \ell \neq d}}^{L_c} w_\ell\right)} \\
&\quad \leq 2 \sum_{d=1}^{L_c} \sqrt{f_{d-1} f_d k_d^2 \left(\ln(4w_d) + \ln\left(1 + \prod_{\substack{\ell=1 \\ \ell \neq d}}^{L_c} w_\ell\right)\right)} \\
&\quad \leq 2 \sum_{d=1}^{L_c} \sqrt{f_{d-1} f_d k_d^2 \left(\ln(4) + \ln\left(w_d + \prod_{\ell=1}^{L_c} w_\ell\right)\right)} \\
&\quad \leq 2 \sqrt{\left(\ln(4) + \ln\left(\widetilde{w}_{\max} + \prod_{\ell=1}^{L_c} w_\ell\right)\right)} \sum_{d=1}^{L_c} \sqrt{f_{d-1} f_d k_d^2} \\
&\quad \leq 2 \sqrt{\left(\ln(4) + \ln(\widetilde{w}_{\max}) + \ln\left(1 + \prod_{\ell=1}^{L_c} w_\ell\right)\right)} \sum_{d=1}^{L_c} \sqrt{f_{d-1} f_d k_d^2} \\
&\quad \leq \left(\sum_{d=1}^{L_c} \sqrt{f_{d-1} f_d k_d^2}\right) \sqrt{KJ \left(\ln(y_{\max}) + \ln(\|A\|_2) + \ln(\|A\|_\infty) + \ln\left(1 + \prod_{d=1}^{L_c} w_d\right)\right)} \\
&\quad = \left(\sum_{d=1}^{L_c} \sqrt{f_{d-1} f_d k_d^2}\right) \sqrt{KJ \mathcal{G}(A, L_c)}.
\end{aligned}
$$

Next, using equation (B.8), there exists a constant $a_{c_1} > 0$ such that

$$
\begin{aligned}
q_2 = z_\infty^2 (c_1 + p_{\max} y_{\max} \|A\|_\infty) &\leq z_\infty^2 (a_{c_1} y_{\max} \|A\|_2^3 + p_{\max} y_{\max} \|A\|_\infty) \\
&\leq z_\infty^2 y_{\max} \max\{a_{c_1}, p_{\max}\} \left(\|A\|_2^3 + \|A\|_\infty\right).
\end{aligned}
$$

Implying that

$$\begin{aligned}\ln(q_2) &\leq \ln(z_\infty^2) + \ln(\max\{a_{c_1}, p_{\max}\}) + \ln(y_{\max}) + \ln\left(\|A\|_2^3 + \|A\|_\infty\right) \\ &\leq \ln(z_\infty^2) + \ln(\max\{a_{c_1}, p_{\max}\}) + \ln(y_{\max}) + 3\ln\left(\|A\|_2\right) + \ln\left(\|A\|_\infty\right) \\ &\lesssim \ln(y_{\max}) + \ln(\|A\|_2) + \ln(\|A\|_\infty) \end{aligned} \tag{C.7}$$

where in the second inequality we used that $\ln(x+y) \leq \ln(x) + \ln(y)$ for $x, y \geq 2$ and took n and m large enough such that $\|A\|_2 \geq 2$ and $\|A\|_\infty \geq 2$.

Note that

$$\begin{aligned} KJs_{L_c} &= \mathcal{O}\left(KJ\sum_{d=1}^{L_c}\sqrt{f_{d-1}f_d k_d^2}\right) \\ KJ\sum_{d=1}^{L_c}\sqrt{f_{d-1}f_d k_d^2}\left(\sqrt{\ln(4w_d)} + \sqrt{\ln\left(1+\eta_d\right)}\right) &= \mathcal{O}\left((KJ)^{3/2}\sqrt{\mathcal{G}(A, L_c)}\sum_{d=1}^{L_c}\sqrt{f_{d-1}f_d k_d^2}\right) \\ KJs_{L_c}\sqrt{\ln(KJ(L_c+1)+1)} &= \mathcal{O}\left(KJ\sqrt{\ln(KJL_c)}\sum_{d=1}^{L_c}\sqrt{f_{d-1}f_d k_d^2}\right) \\ KJs_{L_c}\sqrt{\ln(\sqrt{n})} &= \mathcal{O}\left(KJ\sqrt{\ln(n)}\sum_{d=1}^{L_c}\sqrt{f_{d-1}f_d k_d^2}\right). \end{aligned}$$

By equation (C.7)

$$KJs_{L_c}\sqrt{\ln(q_2)} = \mathcal{O}\left(KJ\sqrt{\ln(y_{\max}) + \ln(\|A\|_2) + \ln(\|A\|_\infty)}\sum_{d=1}^{L_c}\sqrt{f_{d-1}f_d k_d^2}\right).$$

Similarly, by equation (C.2)

$$\begin{aligned} \frac{2}{3}KJ^{3/2}s_{L_c}\sqrt{\ln(r_1)} &= \mathcal{O}\left(KJ^{3/2}\sqrt{\mathcal{G}(A, L_c)}\sum_{d=1}^{L_c}\sqrt{f_{d-1}f_d k_d^2}\right) \\ \frac{2}{3}K^{3/2}Js_{L_c}\sqrt{J-1}\sqrt{\ln(r_1)} &= \mathcal{O}\left((KJ)^{3/2}\sqrt{\mathcal{G}(A, L_c)}\sum_{d=1}^{L_c}\sqrt{f_{d-1}f_d k_d^2}\right). \end{aligned}$$

Lastly, using equation (C.3)

$$\frac{2}{3}K^{3/2}Js_{L_c}\sqrt{\ln\left(r_1+r_2c_1J\right)}=\mathcal{O}\left(K^{3/2}J\sqrt{\mathcal{G}(A,L_c)+\ln(J)}\sum_{d=1}^{L_c}\sqrt{f_{d-1}f_dk_d^2}\right).$$

Therefore, by Lemma B.1.1,

$$\begin{aligned}
&\mathcal{O}\left(\sum_{k=1}^{K}\sum_{j=1}^{J}\sum_{d=1}^{L_c}\sqrt{f_{d-1}f_dk_d^2}\sqrt{\ln\left(e\left(1+\frac{4w_d(KJ(L_c+1)+1)\kappa_{k,d}^{(j)}}{c_{\max}}\right)\right)}\right)\\
&=\mathcal{O}\left(KJ\left(\sum_{d=1}^{L_c}\sqrt{f_{d-1}f_dk_d^2}\right)\max\Big\{1,\sqrt{KJ\mathcal{G}(A,L_c)},\ \sqrt{\ln(KJL_c)},\ \sqrt{\ln(n)},\ \sqrt{J\mathcal{G}(A,L_c)},\right.\\
&\qquad\left.\sqrt{\ln(y_{\max})+\ln(\|A\|_2)+\ln(\|A\|_\infty)},\ \sqrt{KJ\mathcal{G}(A,L_c)},\ \sqrt{K\left(\mathcal{G}(A,L_c)+\ln(J)\right)}\Big\}\right)\\
&=\mathcal{O}\left(KJ\left(\sum_{d=1}^{L_c}\sqrt{f_{d-1}f_dk_d^2}\right)\sqrt{\max\left\{KJ\mathcal{G}(A,L_c),\ln(KJL_c),\ln(n)\right\}}\right) \qquad \text{(C.8)}
\end{aligned}$$

where the final equality holds since

$$\ln(y_{\max})+\ln(\|A\|_2)+\ln(\|A\|_\infty)\leq J\mathcal{G}(A,L_c)\leq KJ\mathcal{G}(A,L_c)$$

and, for all $K,A,L_c,J\in\mathbb{N}$

$$K\left(\mathcal{G}(A,L_c)+\ln(J)\right)\leq KJ\mathcal{G}(A,L_c).$$

Finally, in equation (C.8), bound each f_d by $f_{\max}=\max_{1\leq d\leq L_c}f_d$ and each k_d by $k_{\max}=\max_{1\leq d\leq L_c}k_d$. Additionally, recall that $\mathcal{G}(A,L_c)\lesssim\ln(y_{\max})+\ln(\|A\|_2)+\ln(\|A\|_\infty)+L_c$. As $\ln(KJL_c)\leq KJ(\ln(y_{\max})+\ln(\|A\|_2)+\ln(\|A\|_\infty)+L_c)$, then equation (C.8) reduces to

$$\begin{aligned}
&\mathcal{O}\left(\sum_{k=1}^{K}\sum_{j=1}^{J}\sum_{d=1}^{L_c}\sqrt{f_{d-1}f_dk_d^2}\sqrt{\ln\left(e\left(1+\frac{4w_d(KJ(L_c+1)+1)\kappa_{k,d}^{(j)}}{c_{\max}}\right)\right)}\right)\\
&\qquad=\mathcal{O}\left(KJL_c\sqrt{KJ\left(\ln(y_{\max})+\ln(\|A\|_2)+\ln(\|A\|_\infty)+L_c\right)+\ln(n)}\right). \qquad \text{(C.9)}
\end{aligned}$$

Term 3:

Following the analysis of Term 2 in Corollary 4.5.2, the inequality

$$\sqrt{\ln\left(e\left(1+\frac{4\delta(KJ(L_c+1)+1)\kappa^{(j)}_{k,L_c+1}}{c_{\max}}\right)\right)} \leq 1+\sqrt{\frac{\ln(3)}{\ln(2)}}\left(\sqrt{\ln(4\delta)}+\sqrt{\ln(KJ(L_c+1)+1)}+\sqrt{\ln\left(\kappa^{(j)}_{k,L_c+1}\right)}\right)$$

holds when taking n, K, J, and L_c large enough such that $c_{\max} \geq 4\delta$ and $4\delta(KJ(L_c+1)+1)\kappa^{(j)}_{k,L_c+1} \geq 2c_{\max}$. Furthermore,

$$\begin{aligned}\ln\left(\kappa^{(j)}_{k,L_c+1}\right) \leq{}& \ln(z_\infty(c_1+p_{\max}y_{\max}||A||_\infty)\xi)+(J-j)\ln(r_1)\\ &+(K-k)\left((J-1)\ln(r_1)+\ln\left(r_1+r_2c_1J\right)\right).\end{aligned}$$

Therefore, again following the analysis of Term 2 in Corollary 4.5.2, by using equation (C.2), equation (C.3), and Lemma B.1.1 it holds that

$$\begin{aligned}&\mathcal{O}\left(\sum_{k=1}^{K}\sum_{j=1}^{J}\sqrt{\ln\left(e\left(1+\frac{4\delta(KJ(L_c+1)+1)\kappa^{(j)}_{k,L_c+1}}{c_{\max}}\right)\right)}\right)\\ &=\mathcal{O}\Big(\max\Big\{KJ\sqrt{\ln(y_{\max})+\ln(\|A\|_2)+\ln(\|A\|_\infty)},\ KJ\sqrt{\ln(KJL_c)},\ KJ^{3/2}\sqrt{\mathcal{G}(A,L_c)},\\ &\qquad (KJ)^{3/2}\sqrt{\mathcal{G}(A,L_c)},\ K^{3/2}J\sqrt{\mathcal{G}(A,L_c)+\ln(J)}\Big\}\Big)\\ &=\mathcal{O}\left(KJ\sqrt{\max\left\{\ln(KJL_c),\ KJ\mathcal{G}(A,L_c)\right\}}\right) \qquad \text{(C.10)}\end{aligned}$$

where the final equality holds since

$$\ln(y_{\max})+\ln(\|A\|_2)+\ln(\|A\|_\infty) \leq J\mathcal{G}(A,L_c) \leq KJ\mathcal{G}(A,L_c)$$

and, for all $K, A, L_c, J \in \mathbb{N}$

$$K\left(\mathcal{G}(A, L_c) + \ln(J)\right) \leq KJ\mathcal{G}(A, L_c).$$

Finally, recall that $\mathcal{G}(A, L_c) \lesssim \ln(y_{\max}) + \ln(\|A\|_2) + \ln(\|A\|_\infty) + L_c$. Hence, using that $\ln(KJL_c) \leq KJ(\ln(y_{\max}) + \ln(\|A\|_2) + \ln(\|A\|_\infty) + L_c)$, equation (C.10) reduces to

$$\begin{aligned}\mathcal{O}\left(\sum_{k=1}^{K}\sum_{j=1}^{J}\sqrt{\ln\left(e\left(1+\frac{4\delta(KJ(L_c+1)+1)\kappa^{(j)}_{k,L_c+1}}{c_{\max}}\right)\right)}\right)\\ = \mathcal{O}\left((KJ)^{3/2}\sqrt{\ln(y_{\max}) + \ln(\|A\|_2) + \ln(\|A\|_\infty) + L_c}\right)\end{aligned} \tag{C.11}$$

Term 4:

It immediately holds that

$$4c_{\max}\sqrt{2\ln(4/\varepsilon)/(nN_s)} = \mathcal{O}\left(\frac{c_{\max}}{\sqrt{nN_s}}\right). \tag{C.12}$$

Full Generalization Error Bound:

Combing the bound on each of the four terms – i.e. equation (C.5), equation (C.8), equation (C.10), and equation (C.12) – we have

$$\begin{aligned}
\mathcal{L}(\widehat{\boldsymbol{c}}) \leq \mathcal{L}_{\mathcal{S}}(\widehat{\boldsymbol{c}}) &+ \frac{8c_{\max}}{\sqrt{nN_s}}\sqrt{2n-1}\sqrt{\ln\left(e\left(1+\frac{4p_{\max}(KJ(L_c+1)+1)\kappa}{c_{\max}}\right)\right)} \\
&+ \frac{8c_{\max}}{\sqrt{nN_s}}\sum_{k=1}^{K}\sum_{j=1}^{J}\sum_{d=1}^{L_c}\sqrt{f_{d-1}f_d k_d^2}\sqrt{\ln\left(e\left(1+\frac{4w_d(KJ(L_c+1)+1)\kappa_{k,d}^{(j)}}{c_{\max}}\right)\right)} \\
&+ \frac{8c_{\max}}{\sqrt{nN_s}}\sum_{k=1}^{K}\sum_{j=1}^{J}\sqrt{\ln\left(e\left(1+\frac{4\delta(KJ(L_c+1)+1)\kappa_{k,L_c+1}^{(j)}}{c_{\max}}\right)\right)} \\
&+ 4c_{\max}\sqrt{2\ln(4/\varepsilon)/(nN_s)} \\
= \mathcal{L}_{\mathcal{S}}(\widehat{\boldsymbol{c}}) &+ \mathcal{O}\left(c_{\max}\sqrt{\frac{\max\left\{\ln(KJL_c), K(J-1)\mathcal{G}(A,L_c)\right\}}{N_s}}\right) \\
&+ \mathcal{O}\left(c_{\max}KJ\left(\sum_{d=1}^{L_c}\sqrt{f_{d-1}f_d k_d^2}\right)\sqrt{\frac{\max\left\{KJ\mathcal{G}(A,L_c),\, \ln(KJL_c),\, \ln(n)\right\}}{nN_s}}\right) \\
&+ \mathcal{O}\left(c_{\max}KJ\sqrt{\frac{\max\left\{\ln(KJL_c),\ KJ\left(\mathcal{G}(A,L_c)\right)\right\}}{nN_s}}\right) \\
&+ \mathcal{O}\left(\frac{c_{\max}}{\sqrt{nN_s}}\right).
\end{aligned}$$

Note that for positive functions f_1 and f_2, $g = \mathcal{O}(\max\{f_1, f_2\})$ if and only if $|g| \lesssim f_1 + f_2$. Thus, the PGD DR-CG-Net generalization error bound is given asymptotically by

$$\begin{aligned}
|\mathcal{L}(\widehat{\boldsymbol{c}}) - \mathcal{L}_{\mathcal{S}}(\widehat{\boldsymbol{c}})| \lesssim c_{\max}&\left(\sqrt{n} + KJ\left(1+\sum_{d=1}^{L_c}\sqrt{f_{d-1}f_d k_d^2}\right)\right)\sqrt{\frac{\mathcal{F}(K,J,L_c)}{nN_s}} \\
&+ c_{\max}KJ\left(\sum_{d=1}^{L_c}\sqrt{f_{d-1}f_d k_d^2}\right)\sqrt{\frac{\ln(n)}{nN_s}}
\end{aligned} \tag{C.13}$$

for

$$\begin{aligned}\mathcal{F}(K, J, L_c) &= \ln(KJL_c) + KJ\mathcal{G}(A, L_c) \\ &= \ln(KJL_c) + KJ\left(\ln(y_{\max}) + \ln(\|A\|_2) + \ln(\|A\|_\infty) + \ln\left(1 + \prod_{\ell=1}^{L_c} w_\ell\right)\right).\end{aligned}$$

Next, combining equation (C.6), equation (C.9), and equation (C.11), where each f_d, k_d, and w_d are bounded respectively by $f_{\max}, k_{\max}$, and $w_{\max}$, then equation (C.13) reduces to

$$\begin{aligned}&|\mathcal{L}(\widehat{\boldsymbol{c}}) - \mathcal{L}_{\mathcal{S}}(\widehat{\boldsymbol{c}})| \\ &\quad\lesssim c_{\max}\left(\sqrt{n} + KJ\left(1 + L_c\right)\right)\sqrt{\frac{KJ\left(\ln(y_{\max}) + \ln(\|A\|_2) + \ln(\|A\|_\infty) + L_c\right)}{nN_s}} \\ &\quad\quad + c_{\max}KJL_c\sqrt{\frac{\ln(n)}{nN_s}}.\end{aligned} \tag{C.14}$$

Lastly, let $c_{\max} = \mathcal{O}(\sqrt{n}), y_{\max} = \mathcal{O}(\sqrt{m})$, $\|A\|_2 = \mathcal{O}(\sqrt{mn})$, and $\|A\|_\infty = \mathcal{O}(n)$. Then equation (C.14) reduces to

$$|\mathcal{L}(\widehat{\boldsymbol{c}}) - \mathcal{L}_{\mathcal{S}}(\widehat{\boldsymbol{c}})| \lesssim \left(\sqrt{n} + KJL_c\right)\sqrt{\frac{KJ\left(\ln(m) + \ln(n) + L_c\right)}{N_s}} \tag{C.15}$$

giving the desired asymptotic forms form the PGD DR-CG-Net generalization error. ⊠

Appendix D

ISTA DR-CG-Net Asymptotic Generalization Error

In this appendix, we supply a detailed construction of the asymptotic form for the generalization error bound of ISTA DR-CG-Net stated in Corollary 4.6.8 where $c_{\max}, y_{\max}, \|A\|_2$, and $\|A\|_\infty$ depend on the measurement and signal of interest size. To prove Corollary 4.6.9 the exact steps in the proof of Corollary 4.6.8 below can be followed with $\|A\|_2$ and $\|A\|_\infty$ set to 1. Similarly, to prove Corollary 4.6.10 the steps to prove Corollary 4.6.8 below can be followed with $c_{\max}, y_{\max}, \|A\|_2$, and $\|A\|_\infty$ set at 1.

Proof of Corollary 4.6.8. First recall that the GEB is

$$\begin{aligned}\mathcal{L}(\widehat{\boldsymbol{c}}) \leq \mathcal{L}_{\mathcal{S}}(\widehat{\boldsymbol{c}}) &+ \frac{8c_{\max}}{\sqrt{nN_s}}\sqrt{2n-1}\sqrt{\ln\left(e\left(1+\frac{4p_{\max}(KJ(L_c+1)+1)\kappa}{c_{\max}}\right)\right)} \\ &+ \frac{8c_{\max}}{\sqrt{nN_s}}\sum_{k=1}^{K}\sum_{j=1}^{J}\sum_{d=1}^{L_c}\sqrt{f_{d-1}f_d k_d^2}\sqrt{\ln\left(e\left(1+\frac{4w_d(KJ(L_c+1)+1)\kappa_{k,d}^{(j)}}{c_{\max}}\right)\right)} \\ &+ \frac{8c_{\max}}{\sqrt{nN_s}}\sum_{k=1}^{K}\sum_{j=1}^{J}\sqrt{\ln\left(e\left(1+\frac{4\delta(KJ(L_c+1)+1)\kappa_{k,L_c+1}^{(j)}}{c_{\max}}\right)\right)} \\ &+ 4c_{\max}\sqrt{2\ln(4/\varepsilon)/(nN_s)}\end{aligned}$$

for

$$\begin{aligned}\kappa &= z_\infty(c_1 + p_{\max}y_{\max}\|A\|_\infty)\widehat{c}_1^{(K,J)} + z_\infty c_2 \\ \kappa_{k,d}^{(j)} &= z_\infty(c_1 + p_{\max}y_{\max}\|A\|_\infty)\widehat{c}_{k,j,d,2}^{(K,J)}\end{aligned}$$

where

$$
\begin{aligned}
c_1 &= p_{\max} y_{\max} \ \|A\|_2 \left(1 + 2z_\infty^2 p_{\max} \ \|A\|_2^2\right) \\
c_2 &= z_\infty y_{\max} \|A\|_2 \left(p_{\max}/p_{\min}\right)^2 \\
\widehat{c}_1^{(K,J)} &= c_2 r_2 \left(\frac{1 - r_1^J}{1 - r_1}\right) \left(\frac{1 - \left(r_1^J + c_1 r_2 \frac{1 - r_1^J}{1 - r_1}\right)^K}{1 - \left(r_1^J + c_1 r_2 \frac{1 - r_1^J}{1 - r_1}\right)}\right) \\
\widehat{c}_{k,j,d,2}^{(K,J)} &= r_1^{J-j} \left(r_1^J + c_1 r_2 \frac{1 - r_1^J}{1 - r_1}\right)^{K-k} \begin{cases} (\sqrt{n} z_\infty + \delta\xi) \prod_{\substack{\ell=1 \\ \ell \neq d}}^{L_c} w_\ell & d = 1, 2, \ldots, L_c \\ \left(1 + \prod_{\ell=1}^{L_c} w_\ell\right) \xi & d = L_c + 1 \end{cases}
\end{aligned}
$$

for

$$
\begin{aligned}
r_1 &= \left(1 + \prod_{\ell=1}^{L_c} w_\ell\right) \left(1 + \delta (z_\infty p_{\max} y_{\max} \|A\|_2 \|A\|_\infty)^2\right) \\
r_2 &= \left(1 + \prod_{\ell=1}^{L_c} w_\ell\right) \delta y_{\max} \|A\|_2 \left(1 + z_\infty^2 p_{\max} \|A\|_2 (\|A\|_2 + \|A\|_\infty)\right)
\end{aligned}
$$

We will consider each term on the right hand side of the ISTA DR-CG-Net GEB separately and provide a Big O bound for each. Subsequently, using Lemma B.1.1 we can combine each of the three separate bounds into a single bound by taking a maximum. Note that the empirical loss term $\mathcal{L}_{\mathcal{S}}(\widehat{c})$ is independent of the network size of ISTA DR-CG-Net and we will assume it is independent of the reconstructed signal size allowing us to ignore this term for the asymptotic analysis. Hence, we have four terms to consider individually; three terms structured in the form of $\ln(e(1 + \widetilde{c}(KJ(L_c + 1) + 1)\widetilde{\kappa}$, for constant $\widetilde{c}$ and variable $\widetilde{\kappa}$ dependent on signal and network size, and the error term $4c_{\max}\sqrt{2\ln(4/\varepsilon)/(nN_s)}$.

Define the function $\mathcal{G} : \mathbb{R}^{m \times n} \times \mathbb{R} \to \mathbb{R}$ by

$$
\mathcal{G}(A, L_c) \coloneqq \ln(y_{\max}) + \ln(\|A\|_2) + \ln(\|A\|_\infty) + \ln\left(1 + \prod_{d=1}^{L_c} w_d\right).
$$

As the function $\mathcal{G}$ will be present in the first three terms we analyze below, it will be useful in simplifying the asymptotic bounds for each term.

Term 1:

As in the proof of Corollary 4.5.2 it holds that

$$\begin{aligned}
&\sqrt{\ln\left(e\left(1+\frac{4p_{\max}(KJ(L_c+1)+1)\kappa}{c_{\max}}\right)\right)} \\
&\quad\leq 1+\sqrt{\frac{\ln(3)}{\ln(2)}}\left(\sqrt{\ln\left(4p_{\max}\right)}+\sqrt{\ln\left(KJ(L_c+1)+1\right)}+\sqrt{\ln(J)}+\sqrt{\ln(K)}\right. \\
&\qquad\left.+\sqrt{K(J-1)}\sqrt{\ln(r_1)}+\sqrt{K-1}\sqrt{\ln\left(r_1+r_2c_1J\right)}+\sqrt{\ln(q_1)}\right),
\end{aligned}$$

when taking n, K, J, and L_c large enough such that $c_{\max}\geq 4p_{\max}$ and $4p_{\max}(KJ(L_c+1)+1)\kappa\geq 2c_{\max}$, and for $q_1=z_\infty(c_1+p_{\max}y_{\max}||A||_\infty)c_2r_2+z_\infty c_2$.

Now, let

$$\begin{aligned}
a_1&=1+\delta(z_\infty p_{\max}y_{\max}\|A\|_2\|A\|_\infty)^2\\
a_2&=\delta y_{\max}\|A\|_2\left(1+z_\infty^2p_{\max}\|A\|_2(\|A\|_2+\|A\|_\infty)\right).
\end{aligned}$$

Observe

$$\ln(r_1)=\ln\left(\left(1+\prod_{\ell=1}^{L_c}w_\ell\right)a_1\right)=\ln(a_1)+\ln\left(1+\prod_{d=1}^{L_c}w_d\right)$$

and

$$\ln\left(r_1+r_2c_1J\right)=\ln\left(\left(1+\prod_{\ell=1}^{L_c}w_\ell\right)(a_1+a_2c_1J)\right)=\ln(a_1+a_2c_1J)+\ln\left(1+\prod_{d=1}^{L_c}w_d\right).$$

By equation (B.12), $\ln(a_1)\lesssim\ln(y_{\max})+\ln(\|A\|_2)+\ln(\|A\|_\infty)$ and thus

$$\ln(r_1)\lesssim\ln(y_{\max})+\ln(\|A\|_2)+\ln(\|A\|_\infty)+\ln\left(1+\prod_{d=1}^{L_c}w_d\right)=\mathcal{G}(A,L_c). \tag{D.1}$$

Similarly, by equation (B.12) and (B.15), $\ln(a_1+a_2c_1J) \lesssim \ln(y_{\max})+\ln(\|A\|_2)+\ln(\|A\|_\infty)+\ln(J)$ and thus

$$\begin{aligned}\ln\left(r_1 + r_2c_1J\right) &\lesssim \ln(y_{\max}) + \ln(\|A\|_2) + \ln(\|A\|_\infty) + \ln(J) + \ln\left(1 + \prod_{d=1}^{L_c} w_d\right) \\ &= \mathcal{G}(A, L_c) + \ln(J). \end{aligned} \tag{D.2}$$

Furthermore, by equation (B.11)

$$r_2 \lesssim \left(1 + \prod_{d=1}^{L_c} w_d\right) y_{\max}\|A\|_2^2 \left(\|A\|_2 + \|A\|_\infty\right). \tag{D.3}$$

Lastly, equation (B.8), equation (B.9), and equation (D.3) imply that there exists a constant $a_{q_1} > 0$ such that

$$\begin{aligned} & q_1 \\ & = z_\infty(c_1 + p_{\max}y_{\max}||A||_\infty)c_2r_2 + z_\infty c_2 \\ & \leq a_{q_1}z_\infty \left(\left(y_{\max}\|A\|_2^3 + p_{\max}y_{\max}\|A\|_\infty\right)\left(1 + \prod_{d=1}^{L_c} w_d\right) y_{\max}^2\|A\|_2^3 \left(\|A\|_2 + \|A\|_\infty\right) + y_{\max}\|A\|_2\right) \\ & \leq a_{q_1}z_\infty y_{\max}\|A\|_2 \left[\max\{1, p_{\max}\}\left(1 + \prod_{d=1}^{L_c} w_d\right) y_{\max}^3\|A\|_2^2 \left(\|A\|_2^3 + \|A\|_\infty\right)\left(\|A\|_2 + \|A\|_\infty\right) + 1\right] \\ & \leq a_{q_1}z_\infty \left(1 + \max\{1, p_{\max}\}\right)\left(1 + \prod_{d=1}^{L_c} w_d\right) y_{\max}^4\|A\|_2^3 \left(\|A\|_2^3 + \|A\|_\infty\right)\left(\|A\|_2 + \|A\|_\infty\right). \end{aligned} \tag{D.4}$$

Therefore, equation (D.4) implies that

$$\begin{aligned}
\ln(q_1) &\leq \ln(a_{q_1}) + \ln\left(z_\infty\left(1+\max\{1,p_{\max}\}\right)\right) + 4\ln(y_{\max}) + 3\ln(\|A\|_2) \\
&\quad + \ln\left(\|A\|_2^3 + \|A\|_\infty\right) + \ln\left(\|A\|_2 + \|A\|_\infty\right) + \ln\left(1+\prod_{d=1}^{L_c} w_d\right) \\
&\leq \ln(a_{q_1}) + \ln\left(z_\infty\left(1+\max\{1,p_{\max}\}\right)\right) + 4\ln(y_{\max}) + 3\ln(\|A\|_2) \\
&\quad + 3\ln(\|A\|_2) + \ln(\|A\|_\infty) + \ln(\|A\|_2) + \ln(\|A\|_\infty) + \ln\left(1+\prod_{d=1}^{L_c} w_d\right) \\
&\lesssim \ln(y_{\max}) + \ln(\|A\|_2) + \ln(\|A\|_\infty) + \ln\left(1+\prod_{d=1}^{L_c} w_d\right) \\
&= \mathcal{G}(A, L_c) \qquad \text{(D.5)}
\end{aligned}$$

where in the second inequality we used that $\ln(x+y) \leq \ln(x) + \ln(y)$ for $x, y \geq 2$ and took n and m large enough such that $\|A\|_2 \geq 2$ and $\|A\|_\infty \geq 2$.

Hence, note that

$$\begin{aligned}
1 + \sqrt{\frac{\ln(3)}{\ln(2)}}\sqrt{\ln\left(4p_{\max}\right)} &= \mathcal{O}(1) \\
\sqrt{\frac{\ln(3)}{\ln(2)}}\sqrt{\ln(KJ(L_c+1)+1)} &= \mathcal{O}\left(\sqrt{\ln(KJL_c)}\right) \\
\sqrt{\frac{\ln(3)}{\ln(2)}}\sqrt{\ln(J)} &= \mathcal{O}\left(\sqrt{\ln(J)}\right) \\
\sqrt{\frac{\ln(3)}{\ln(2)}}\sqrt{\ln(K)} &= \mathcal{O}\left(\sqrt{\ln(K)}\right).
\end{aligned}$$

Furthermore, by equation (D.1)

$$\sqrt{\frac{\ln(3)}{\ln(2)}}\sqrt{K(J-1)}\sqrt{\ln(r_1)} = \mathcal{O}\left(\sqrt{K(J-1)\mathcal{G}(A, L_c)}\right)$$

and by equation (D.2)

$$\sqrt{\frac{\ln(3)}{\ln(2)}}\sqrt{K-1}\sqrt{\ln(r_1+r_2c_1J)} = \mathcal{O}\left(\sqrt{(K-1)\left(\mathcal{G}(A,L_c)+\ln(J)\right)}\right).$$

Lastly, by equation (D.5)

$$\sqrt{\frac{\ln(3)}{\ln(2)}}\sqrt{\log(q_1)} = \mathcal{O}\left(\sqrt{\mathcal{G}(A,L_c)}\right).$$

Thus, using Lemma B.1.1

$$\begin{aligned}
&\sqrt{\ln\left(e\left(1+\frac{4p_{\max}(KJ(L_c+1)+1)\kappa}{c_{\max}}\right)\right)} \\
&\qquad = \mathcal{O}\Big(\max\Big\{1, \sqrt{\ln(KJL_c)}, \sqrt{\ln(J)}, \sqrt{\ln(K)}, \sqrt{K(J-1)\mathcal{G}(A,L_c)}, \\
&\qquad\qquad \sqrt{(K-1)\left(\mathcal{G}(A,L_c)+\ln(J)\right)}, \sqrt{\mathcal{G}(A,L_c)}\Big\}\Big) \\
&\qquad = \mathcal{O}\left(\sqrt{\max\left\{\ln(KJL_c), K(J-1)\mathcal{G}(A,L_c)\right\}}\right). \qquad \text{(D.6)}
\end{aligned}$$

Furthermore, let $w_{\max} = \max_{1\leq d\leq L_c} w_d$. Without loss of generality, we assume that $w_{\max} > 0$, otherwise every convolutional sub-network in DR-CG-Net would be using a matrix kernel of all zeros thereby mapping all inputs to zero. By the binomial theorem for any $x, y \in \mathbb{R}$ and any $k \in \mathbb{N}$

$$(x+y)^k = \sum_{j=0}^{k}\binom{k}{j}x^{k-j}y^j \geq \binom{k}{0}x^k + \binom{k}{k}y^k = x^k + y^k$$

implying $1 + x^k \leq (1+x)^k$. Thus

$$\ln\left(1+\prod_{d=1}^{L_c} w_d\right) \leq \ln\left(1+w_{\max}^{L_c}\right) \leq \ln((1+w_{\max})^{L_c}) = L_c\ln(1+w_{\max}) \lesssim L_c,$$

which implies

$$\mathcal{G}(A, L_c) \lesssim \ln(y_{\max}) + \ln(\|A\|_2) + \ln(\|A\|_\infty) + L_c.$$

Combining with equation (D.6) and noting that $\ln(KJL_c) \leq K(J-1)(\ln(y_{\max}) + \ln(\|A\|_2) + \ln(\|A\|_\infty) + L_c)$ produces

$$\begin{aligned}
&\mathcal{O}\left(\sqrt{\ln\left(e\left(1 + \frac{4p_{\max}(KJ(L_c+1)+1)\kappa}{c_{\max}}\right)\right)}\right) \\
&\qquad = \mathcal{O}\left(\sqrt{K(J-1)\left(\ln(y_{\max}) + \ln(\|A\|_2) + \ln(\|A\|_\infty) + L_c\right)}\right). \qquad \text{(D.7)}
\end{aligned}$$

Term 2:

Following the analysis of Term 2 in the proof of Corollary 4.6.8 it holds that

$$\begin{aligned}
&\sqrt{\ln\left(e\left(1 + \frac{4w_d(KJ(L_c+1)+1)\kappa_{k,d}^{(j)}}{c_{\max}}\right)\right)} \\
&\qquad \leq 1 + \sqrt{\ln(4w_d)} + \sqrt{\ln(KJ(L_c+1)+1)} + \sqrt{\ln(1+\kappa_{k,d}^{(j)})}
\end{aligned}$$

where we take n large enough such that $c_{\max} \geq 1$ and K, J and L_c large enough such that $4w_d(KJ(L_c+1)+1) \geq 1$.

Define $\eta_d = \prod_{\substack{\ell=1 \\ \ell \neq d}}^{L_c} w_\ell$. Take n, K, and J large enough such that

$$z_\infty(\sqrt{n}z_\infty + \delta\xi)(c_1 + p_{\max}y_{\max}\|A\|_\infty)r_1^{J-j}\left(r_1^J + c_1 r_2 \frac{1-r_1^J}{1-r_1}\right)^{K-k} \geq 1.$$

Then for every $d \in \mathbb{N}[L_c]$, by equation (B.5), it holds that

$$
\begin{aligned}
&\ln\left(1+\kappa_{k,d}^{(j)}\right)\\
&=\ln\left(1+z_\infty(\sqrt{n}z_\infty+\delta\xi)(c_1+p_{\max}y_{\max}\|A\|_\infty)r_1^{J-j}\left(r_1^J+c_1r_2\frac{1-r_1^J}{1-r_1}\right)^{K-k}\eta_d\right)\\
&\le\ln\left(\left(z_\infty(\sqrt{n}z_\infty+\delta\xi)(c_1+p_{\max}y_{\max}\|A\|_\infty)r_1^{J-j}\left(r_1^J+c_1r_2\frac{1-r_1^J}{1-r_1}\right)^{K-k}\right)(1+\eta_d)\right)\\
&=\ln(\sqrt{n}z_\infty+\delta\xi)+\ln(q_2)+(J-j)\ln(r_1)+(K-k)\ln\left(r_1^J+c_1r_2\frac{1-r_1^J}{1-r_1}\right)+\ln(1+\eta_d)\\
&\le\ln(\sqrt{n}z_\infty+\delta\xi)+\ln(q_2)+(J-j)\ln(r_1)+(K-k)\left((J-1)\ln(r_1)+\ln(r_1+r_2c_1J)\right)+\ln(1+\eta_d)
\end{aligned}
$$

where $q_2 = z_\infty(c_1 + p_{\max}y_{\max}\|A\|_\infty)$.

Hence

$$
\begin{aligned}
&\sqrt{\ln\left(e\left(1+\frac{4w_d(KJ(L_c+1)+1)\kappa_{k,d}^{(j)}}{c_{\max}}\right)\right)}\\
&\le 1+\sqrt{\ln(4w_d)}+\sqrt{\ln(KJ(L_c+1)+1)}+\sqrt{\ln(\sqrt{n}z_\infty+\delta\xi)}+\sqrt{(J-j)}\sqrt{\ln(r_1)}\\
&\quad+\sqrt{\ln(q_2)}+\sqrt{(K-k)}\sqrt{(J-1)}\sqrt{\ln(r_1)}+\sqrt{(K-k)}\sqrt{\ln(r_1+r_2c_1J)}+\sqrt{\ln(1+\eta_d)}.
\end{aligned}
$$

Therefore, by equation (B.20)

$$
\begin{aligned}
&\sum_{k=1}^{K}\sum_{j=1}^{J}\sqrt{\ln\left(e\left(1+\frac{4w_d(KJ(L_c+1)+1)\kappa_{k,d}^{(j)}}{c_{\max}}\right)\right)}\\
&\qquad\le KJ\left(1+\sqrt{\ln(4w_d)}\right)+KJ\sqrt{\ln(KJ(L_c+1)+1)}+KJ\sqrt{\ln(\sqrt{n}z_\infty+\delta\xi)}\\
&\qquad\quad+KJ\sqrt{\ln(q_2)}+\frac{2}{3}KJ^{3/2}\sqrt{\ln(r_1)}+\frac{2}{3}K^{3/2}J\sqrt{J-1}\sqrt{\ln(r_1)}\\
&\qquad\quad+\frac{2}{3}K^{3/2}J\sqrt{\ln\left(r_1+r_2c_1J\right)}+KJ\sqrt{\ln\left(1+\eta_d\right)}.
\end{aligned}
$$

Define $s_{L_c} = \sum_{d=1}^{L_c} \sqrt{f_{d-1} f_d k_d^2}$, then

$$\begin{aligned}
&\sum_{k=1}^{K}\sum_{j=1}^{J}\sum_{d=1}^{L_c} \sqrt{f_{d-1} f_d k_d^2}\sqrt{\ln\left(e\left(1+\frac{4w_d(KJ(L_c+1)+1)\kappa_{k,d}^{(j)}}{c_{\max}}\right)\right)} \\
&\quad \leq KJ\left(s_{L_c} + \sum_{d=1}^{L_c}\sqrt{f_{d-1} f_d k_d^2}\sqrt{\ln(4w_d)}\right) + KJ s_{L_c}\sqrt{\ln(KJ(L_c+1)+1)} \\
&\quad + KJ s_{L_c}\sqrt{\ln(\sqrt{n}z_\infty + \delta\xi)} + KJ s_{L_c}\sqrt{\ln(q_2)} \\
&\quad + \frac{2}{3}KJ^{3/2} s_{L_c}\sqrt{\ln(r_1)} + \frac{2}{3}K^{3/2} J s_{L_c}\sqrt{J-1}\sqrt{\ln(r_1)} \\
&\quad + \frac{2}{3}K^{3/2} J s_{L_c}\sqrt{\ln\left(r_1 + r_2 c_1 J\right)} + KJ\sum_{d=1}^{L_c}\sqrt{f_{d-1} f_d k_d^2}\sqrt{\ln\left(1+\eta_d\right)}.
\end{aligned}$$

Recall that for sufficiently large m, n, K and J such that $4\widetilde{w}_{\max} \leq y_{\max}\|A\|_2\|A\|_\infty$ and $4 \leq KJ$

$$\sum_{d=1}^{L_c}\sqrt{f_{d-1} f_d k_d^2 \ln(4w_d)} + \sum_{d=1}^{L_c}\sqrt{f_{d-1} f_d k_d^2 \ln\left(1+\eta_d\right)} \leq \left(\sum_{d=1}^{L_c}\sqrt{f_{d-1} f_d k_d^2}\right)\sqrt{KJ\mathcal{G}(A, L_c)}.$$

Next, using equation (B.8), there exists a constant $a_{c_1} > 0$ such that

$$\begin{aligned}
q_2 = z_\infty(c_1 + p_{\max} y_{\max}\|A\|_\infty) &\leq z_\infty(a_{c_1} y_{\max}\|A\|_2^3 + p_{\max} y_{\max}\|A\|_\infty) \\
&\leq z_\infty y_{\max}\max\{a_{c_1}, p_{\max}\}\left(\|A\|_2^3 + \|A\|_\infty\right).
\end{aligned}$$

Implying that

$$\begin{aligned}
\ln(q_2) &\leq \ln(z_\infty) + \ln(\max\{a_{c_1}, p_{\max}\}) + \ln(y_{\max}) + \ln\left(\|A\|_2^3 + \|A\|_\infty\right) \\
&\leq \ln(z_\infty) + \ln(\max\{a_{c_1}, p_{\max}\}) + \ln(y_{\max}) + 3\ln\left(\|A\|_2\right) + \ln\left(\|A\|_\infty\right) \\
&\lesssim \ln(y_{\max}) + \ln(\|A\|_2) + \ln(\|A\|_\infty) \qquad \text{(D.8)}
\end{aligned}$$

where in the second inequality we used that $\ln(x+y) \leq \ln(x) + \ln(y)$ for $x, y \geq 2$ and took n and m large enough such that $\|A\|_2 \geq 2$ and $\|A\|_\infty \geq 2$.

Note that

$$
\begin{aligned}
KJs_{L_c} &= \mathcal{O}\left(KJ\sum_{d=1}^{L_c}\sqrt{f_{d-1}f_dk_d^2}\right)\\
KJ\sum_{d=1}^{L_c}\sqrt{f_{d-1}f_dk_d^2}\left(\sqrt{\ln(4w_d)}+\sqrt{\ln\left(1+\eta_d\right)}\right)&\\
&= \mathcal{O}\left((KJ)^{3/2}\sqrt{\mathcal{G}(A,L_c)}\sum_{d=1}^{L_c}\sqrt{f_{d-1}f_dk_d^2}\right)\\
KJs_{L_c}\sqrt{\ln(KJ(L_c+1)+1)} &= \mathcal{O}\left(KJ\sqrt{\ln(KJL_c)}\sum_{d=1}^{L_c}\sqrt{f_{d-1}f_dk_d^2}\right)\\
KJs_{L_c}\sqrt{\ln(\sqrt{n}z_\infty+\delta\xi)} &= \mathcal{O}\left(KJ\sqrt{\ln(n)}\sum_{d=1}^{L_c}\sqrt{f_{d-1}f_dk_d^2}\right).
\end{aligned}
$$

By equation (D.8)

$$
KJs_{L_c}\sqrt{\ln(q_2)} = \mathcal{O}\left(KJ\sqrt{\ln(y_{\max})+\ln(\|A\|_2)+\ln(\|A\|_\infty)}\sum_{d=1}^{L_c}\sqrt{f_{d-1}f_dk_d^2}\right).
$$

Similarly, by equation (D.1)

$$
\begin{aligned}
\frac{2}{3}KJ^{3/2}s_{L_c}\sqrt{\ln(r_1)} &= \mathcal{O}\left(KJ^{3/2}\sqrt{\mathcal{G}(A,L_c)}\sum_{d=1}^{L_c}\sqrt{f_{d-1}f_dk_d^2}\right)\\
\frac{2}{3}K^{3/2}Js_{L_c}\sqrt{J-1}\sqrt{\ln(r_1)} &= \mathcal{O}\left((KJ)^{3/2}\sqrt{\mathcal{G}(A,L_c)}\sum_{d=1}^{L_c}\sqrt{f_{d-1}f_dk_d^2}\right).
\end{aligned}
$$

Lastly, using equation (D.2)

$$
\frac{2}{3}K^{3/2}Js_{L_c}\sqrt{\ln\left(r_1+r_2c_1J\right)} = \mathcal{O}\left(K^{3/2}J\sqrt{\mathcal{G}(A,L_c)+\ln(J)}\sum_{d=1}^{L_c}\sqrt{f_{d-1}f_dk_d^2}\right).
$$

Therefore, by Lemma B.1.1,

$$\mathcal{O}\left(\sum_{k=1}^{K}\sum_{j=1}^{J}\sum_{d=1}^{L_c}\sqrt{f_{d-1}f_d k_d^2}\sqrt{\ln\left(e\left(1+\frac{4w_d(KJ(L_c+1)+1)\kappa_{k,d}^{(j)}}{c_{\max}}\right)\right)}\right)$$

$$= \mathcal{O}\left(KJ\left(\sum_{d=1}^{L_c}\sqrt{f_{d-1}f_d k_d^2}\right)\max\Big\{1, \sqrt{KJ\mathcal{G}(A,L_c)}, \sqrt{\ln(KJL_c)}, \sqrt{\ln(n)}, \sqrt{J\mathcal{G}(A,L_c)},\right.$$

$$\left.\sqrt{\ln(y_{\max})+\ln(\|A\|_2)+\ln(\|A\|_\infty)}, \sqrt{KJ\mathcal{G}(A,L_c)}, \sqrt{K\left(\mathcal{G}(A,L_c)+\ln(J)\right)}\Big\}\right)$$

$$= \mathcal{O}\left(KJ\left(\sum_{d=1}^{L_c}\sqrt{f_{d-1}f_d k_d^2}\right)\sqrt{\max\left\{KJ\mathcal{G}(A,L_c), \ln(KJL_c), \ln(n)\right\}}\right) \tag{D.9}$$

where the final equality holds since

$$\ln(y_{\max})+\ln(\|A\|_2)+\ln(\|A\|_\infty) \le J\mathcal{G}(A,L_c) \le KJ\mathcal{G}(A,L_c)$$

and, for all $K, A, L_c, J \in \mathbb{N}$

$$K\left(\mathcal{G}(A,L_c)+\ln(J)\right) \le KJ\mathcal{G}(A,L_c).$$

Finally, in equation (D.9), bound each f_d by $f_{\max} = \max_{1\le d\le L_c} f_d$ and each k_d by $k_{\max} = \max_{1\le d\le L_c} k_d$. Additionally, recall that $\mathcal{G}(A,L_c) \lesssim \ln(y_{\max})+\ln(\|A\|_2)+\ln(\|A\|_\infty)+L_c$. As $\ln(KJL_c) \le KJ(\ln(y_{\max})+\ln(\|A\|_2)+\ln(\|A\|_\infty)+L_c)$, then equation (D.9) reduces to

$$\mathcal{O}\left(\sum_{k=1}^{K}\sum_{j=1}^{J}\sum_{d=1}^{L_c}\sqrt{f_{d-1}f_d k_d^2}\sqrt{\ln\left(e\left(1+\frac{4w_d(KJ(L_c+1)+1)\kappa_{k,d}^{(j)}}{c_{\max}}\right)\right)}\right)$$

$$= \mathcal{O}\left(KJL_c\sqrt{KJ\left(\ln(y_{\max})+\ln(\|A\|_2)+\ln(\|A\|_\infty)+L_c\right)+\ln(n)}\right). \tag{D.10}$$

Term 3:

Following the analysis of Term 2 in Corollary 4.5.2, the inequality

$$\sqrt{\ln\left(e\left(1+\frac{4\delta(KJ(L_c+1)+1)\kappa^{(j)}_{k,L_c+1}}{c_{\max}}\right)\right)} \leq 1+\sqrt{\frac{\ln(3)}{\ln(2)}}\left(\sqrt{\ln(4\delta)}+\sqrt{\ln(KJ(L_c+1)+1)}+\sqrt{\ln\left(\kappa^{(j)}_{k,L_c+1}\right)}\right)$$

holds when taking n, K, J, and L_c large enough such that $c_{\max} \geq 4\delta$ and $4\delta(KJ(L_c+1)+1)\kappa^{(j)}_{k,L_c+1} \geq 2c_{\max}$. Furthermore,

$$\ln\left(\kappa^{(j)}_{k,L_c+1}\right) \leq \ln\left(z_\infty \xi(c_1 + p_{\max} y_{\max}||A||_\infty)\left(1+\prod_{\ell=1}^{L_d} w_\ell\right)\right) + (J-j)\ln(r_1) + (K-k)\left((J-1)\ln(r_1)+\ln\left(r_1+r_2c_1J\right)\right).$$

By equation (D.8) there exists a constant $a_{q_2} > 0$ such that

$$\ln\left(z_\infty \xi(c_1 + p_{\max} y_{\max}||A||_\infty)\left(1+\prod_{\ell=1}^{L_d} w_\ell\right)\right) \leq a_{q_2}\left(\ln(y_{\max})+\ln(\|A\|_2)+\ln(\|A\|_\infty)+\ln\left(1+\prod_{\ell=1}^{L_d} w_\ell\right)\right) \leq a_{q_2}\mathcal{G}(A, L_c).$$

Hence,

$$\ln\left(\kappa^{(j)}_{k,L_c+1}\right) \leq a_{q_2}\mathcal{G}(A, L_c) + (J-j)\ln(r_1) + (K-k)\left((J-1)\ln(r_1)+\ln\left(r_1+r_2c_1J\right)\right).$$

Therefore, again following the analysis of Term 2 in Corollary 4.5.2, by using equation (D.1), equation (D.2), and Lemma B.1.1 it holds that

$$\begin{aligned}
&\mathcal{O}\left(\sum_{k=1}^{K}\sum_{j=1}^{J}\sqrt{\ln\left(e\left(1+\frac{4\delta(KJ(L_c+1)+1)\kappa^{(j)}_{k,L_c+1}}{c_{\max}}\right)\right)}\right)\\
&=\mathcal{O}\Big(\max\Big\{KJ\sqrt{\mathcal{G}(A,L_c)},\ KJ\sqrt{\ln(KJL_c)},\\
&\qquad\qquad KJ^{3/2}\sqrt{\mathcal{G}(A,L_c)},\ (KJ)^{3/2}\sqrt{\mathcal{G}(A,L_c)},\ K^{3/2}J\sqrt{\mathcal{G}(A,L_c)+\ln(J)}\Big\}\Big)\\
&=\mathcal{O}\left(KJ\sqrt{\max\left\{\ln(KJL_c),\ KJ\left(\mathcal{G}(A,L_c)\right)\right\}}\right) \qquad \text{(D.11)}
\end{aligned}$$

where the final equality holds since

$$\ln(y_{\max})+\ln(\|A\|_2)+\ln(\|A\|_\infty)\leq J\mathcal{G}(A,L_c)\leq KJ\mathcal{G}(A,L_c)$$

and, for all $K, A, L_c, J\in\mathbb{N}$

$$K\left(\mathcal{G}(A,L_c)+\ln(J)\right)\leq KJ\mathcal{G}(A,L_c).$$

Finally, recall that $\mathcal{G}(A,L_c)\lesssim \ln(y_{\max})+\ln(\|A\|_2)+\ln(\|A\|_\infty)+L_c$. Hence, using that $\ln(KJL_c)\leq KJ(\ln(y_{\max})+\ln(\|A\|_2)+\ln(\|A\|_\infty)+L_c)$, equation (D.11) reduces to

$$\begin{aligned}
&\mathcal{O}\left(\sum_{k=1}^{K}\sum_{j=1}^{J}\sqrt{\ln\left(e\left(1+\frac{4\delta(KJ(L_c+1)+1)\kappa^{(j)}_{k,L_c+1}}{c_{\max}}\right)\right)}\right)\\
&\qquad=\mathcal{O}\left((KJ)^{3/2}\sqrt{\ln(y_{\max})+\ln(\|A\|_2)+\ln(\|A\|_\infty)+L_c}\right) \qquad \text{(D.12)}
\end{aligned}$$

Term 4:

It immediately holds that

$$4c_{\max}\sqrt{2\ln(4/\varepsilon)/(nN_s)}=\mathcal{O}\left(\frac{c_{\max}}{\sqrt{nN_s}}\right). \qquad \text{(D.13)}$$

Full Generalization Error Bound:

Combing the bound on each of the four terms – i.e. equation (D.6), equation (D.9), equation (D.11), and equation (D.13) – we have

$$\begin{aligned}
\mathcal{L}(\widehat{\boldsymbol{c}}) \leq \mathcal{L}_{\mathcal{S}}(\widehat{\boldsymbol{c}}) &+ \frac{8c_{\max}}{\sqrt{nN_s}}\sqrt{2n-1}\sqrt{\ln\left(e\left(1+\frac{4p_{\max}(KJ(L_c+1)+1)\kappa}{c_{\max}}\right)\right)} \\
&+ \frac{8c_{\max}}{\sqrt{nN_s}}\sum_{k=1}^{K}\sum_{j=1}^{J}\sum_{d=1}^{L_c}\sqrt{f_{d-1}f_d k_d^2}\sqrt{\ln\left(e\left(1+\frac{4w_d(KJ(L_c+1)+1)\kappa_{k,d}^{(j)}}{c_{\max}}\right)\right)} \\
&+ \frac{8c_{\max}}{\sqrt{nN_s}}\sum_{k=1}^{K}\sum_{j=1}^{J}\sqrt{\ln\left(e\left(1+\frac{4\delta(KJ(L_c+1)+1)\kappa_{k,L_c+1}^{(j)}}{c_{\max}}\right)\right)} \\
&+ 4c_{\max}\sqrt{2\ln(4/\varepsilon)/(nN_s)} \\
= \mathcal{L}_{\mathcal{S}}(\widehat{\boldsymbol{c}}) &+ \mathcal{O}\left(c_{\max}\sqrt{\frac{\max\left\{\ln(KJL_c), K(J-1)\mathcal{G}(A,L_c)\right\}}{N_s}}\right) \\
&+ \mathcal{O}\left(c_{\max}KJ\left(\sum_{d=1}^{L_c}\sqrt{f_{d-1}f_d k_d^2}\right)\sqrt{\frac{\max\left\{KJ\mathcal{G}(A,L_c),\, \ln(KJL_c),\, \ln(n)\right\}}{nN_s}}\right) \\
&+ \mathcal{O}\left(c_{\max}KJ\sqrt{\frac{\max\left\{\ln(KJL_c),\;\; KJ\left(\mathcal{G}(A,L_c)\right)\right\}}{nN_s}}\right) \\
&+ \mathcal{O}\left(\frac{c_{\max}}{\sqrt{nN_s}}\right).
\end{aligned}$$

Note that for positive functions f_1 and f_2, $g = \mathcal{O}(\max\{f_1, f_2\})$ if and only if $|g| \lesssim f_1 + f_2$. Thus, the ISTA DR-CG-Net generalization error bound is given asymptotically by

$$\begin{aligned}
|\mathcal{L}(\widehat{\boldsymbol{c}}) - \mathcal{L}_{\mathcal{S}}(\widehat{\boldsymbol{c}})| \lesssim c_{\max}&\left(\sqrt{n} + KJ\left(1+\sum_{d=1}^{L_c}\sqrt{f_{d-1}f_d k_d^2}\right)\right)\sqrt{\frac{\mathcal{F}(K,J,L_c)}{nN_s}} \\
&+ c_{\max}KJ\left(\sum_{d=1}^{L_c}\sqrt{f_{d-1}f_d k_d^2}\right)\sqrt{\frac{\ln(n)}{nN_s}} \qquad \text{(D.14)}
\end{aligned}$$

for

$$\begin{aligned}\mathcal{F}(K, J, L_c) &= \ln(KJL_c) + KJ\mathcal{G}(A, L_c) \\ &= \ln(KJL_c) + KJ\left(\ln(y_{\max}) + \ln(\|A\|_2) + \ln(\|A\|_\infty) + \ln\left(1 + \prod_{\ell=1}^{L_c} w_\ell\right)\right).\end{aligned}$$

Next, combining equation (D.7), equation (D.10), and equation (D.12), where each f_d, k_d, and w_d are bounded respectively by $f_{\max}, k_{\max}$, and $w_{\max}$, then equation (D.14) reduces to

$$\begin{aligned}&|\mathcal{L}(\widehat{\boldsymbol{c}}) - \mathcal{L}_{\mathcal{S}}(\widehat{\boldsymbol{c}})| \\ &\quad\lesssim c_{\max}\left(\sqrt{n} + KJ\left(1 + L_c\right)\right)\sqrt{\frac{KJ\left(\ln(y_{\max}) + \ln(\|A\|_2) + \ln(\|A\|_\infty) + L_c\right)}{nN_s}} \\ &\quad\quad + c_{\max}KJL_c\sqrt{\frac{\ln(n)}{nN_s}}.\end{aligned} \tag{D.15}$$

Lastly, let $c_{\max} = \mathcal{O}(\sqrt{n}), y_{\max} = \mathcal{O}(\sqrt{m})$, $\|A\|_2 = \mathcal{O}(\sqrt{mn})$, and $\|A\|_\infty = \mathcal{O}(n)$. Then equation (D.15) reduces to

$$|\mathcal{L}(\widehat{\boldsymbol{c}}) - \mathcal{L}_{\mathcal{S}}(\widehat{\boldsymbol{c}})| \lesssim \left(\sqrt{n} + KJL_c\right)\sqrt{\frac{KJ\left(\ln(m) + \ln(n) + L_c\right)}{N_s}} \tag{D.16}$$

giving the desired asymptotic forms form the ISTA DR-CG-Net generalization error. ⊠