THESIS

SAME DATA, SAME FEATURES: MODERN IMAGENET-TRAINED CONVOLUTIONAL NEURAL NETWORKS LEARN THE SAME THING

Submitted by

David G. McNeely-White

Department of Computer Science

In partial fulfillment of the requirements

For the Degree of Master of Science

Colorado State University

Fort Collins, Colorado

Spring 2020

Master's Committee:

Advisor: J. Ross Beveridge

Charles W. Anderson
Carol A. Seger

ABSTRACT

SAME DATA, SAME FEATURES: MODERN IMAGENET-TRAINED CONVOLUTIONAL NEURAL NETWORKS LEARN THE SAME THING

Deep convolutional neural networks (CNNs) are the dominant technology in computer vision today. Much of the recent computer vision literature can be thought of as a competition to find the best architecture for vision within the deep convolutional framework. Despite all the effort invested in developing sophisticated convolutional architectures, however, it's not clear how different from each other the best CNNs really are. This thesis measures the similarity between ten well-known CNNs, in terms of the properties they extract from images. I find that the properties extracted by each of the ten networks are very similar to each other, in the sense that any of their features can be well approximated by an affine transformation of the features of any of the other nine. In particular, there is evidence that each network extracts mostly the same information as each other network, though some do it more robustly.

The similarity between each of these CNNs is surprising. Convolutional neural networks learn complex non-linear features of images, and the architectural differences between systems suggest that these non-linear functions should take different forms. Nonetheless, these ten CNNs which were trained on the same data set seem to have learned to extract similar properties from images. In essence, each CNN's training algorithm hill-climbs in a very different parameter space, yet converges on a similar solution. This suggests that for CNNs, the selection of the training set and strategy may be more important than the selection of the convolutional architecture.

ACKNOWLEDGEMENTS

These past three years at CSU have enabled me to grow and succeed in ways I had never imagined. I am extremely grateful for each opportunity, experience, and friendship which has helped me along the way. Forgetting this chapter in life will be impossible, as this period has forever changed me and redefined my potential. Most importantly, I would never have finished this arduous journey without the support and encouragement of my professors, peers, colleagues, mentors, and loved ones.

First, I would like to thank my two advisors Dr. Bruce Draper, and Dr. Ross Beveridge. Dr. Draper's brilliance and unstoppable work ethic provide a model which any scientist should envy. I am so grateful for his guidance when my research seemed bleak, and for always challenging me to better distill my thoughts and results. Likewise, Dr. Beveridge's insight and vision always challenged me to work harder. He is always able to think of a new and exciting angle to interpret results, and never shied away from the chance to thoroughly critique and refine my writing or speaking. Without either of them and the challenging but rewarding environment they've created, I would be nowhere near as confident and capable as I am today. I would also like to thank my committee members: Dr. Charles Anderson for his excellent introduction to Artificial Intelligence and Machine Learning; and both Dr. Anderson and Dr. Carol Seger for always offering a friendly, patient ear for the odd research discussion. Thank you also to Dr. Francisco Ortega and Dr. Nathaniel Blanchard for your one-on-one support and career advice.

I am also deeply grateful for all my friends in the vision lab, Computer Science department, and greater CSU community for your advice, encouragement, and company. Countless conversations with Matt, Adam, Heting, Rahul, Ben, Joe, and many others helped me so much when I was frustrated or stuck.

Finally, thank you to Kat, my beloved wife. Her love, support, and patience continue to shape me into a better man. I cannot overstate her importance in my life, and my excitement for our future together.

DEDICATION

To my patient counselor,

constant companion,

unflinching advocate,

steadfast confidant,

brilliant adviser,

devoted partner,

dazzling bride,

and closest friend,

Kat.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

What follows is a brief overview of the historical and modern context of deep convolutional neural networks. For readers familiar with these models as they are specifically applied to computer vision, feel free to skip to Section 1.2.

1.1 Context

Since the infamous 2012 AlexNet paper, deep convolutional neural networks (CNNs) have dominated computer vision research [7]. This surge in popularity is also observed economically, with MarketsandMarkets Research predicting the global artificial intelligence market to be valued at $190 billion by 2025 from $16 billion in 2017 [8]. While the recent growth in deep learning is especially strong, the underpinnings of these modern algorithms go back decades.

The first computational model for a neural network was presented in 1943, unsurprisingly inspired by biological neurons [9]. This "logical calculus" provided a framework for representing complex patterns with many interconnected "all-or-none" neurons. Over the next two decades, research into artificial neural networks grew, with a fully functional many-layered (i.e. deep) neural network demonstrated in 1967, though these are still a far-cry from modern nets [10].

The underlying structure of a standard **feed-forward** (i.e. non-cyclical) neural network is relatively straightforward. See Figure 1.1 for an illustration. Essentially, an input vector x (e.g. an image, sound frequencies, some text) is fed into a series of neurons, each multiplying that input by their own vector of learned weights, w_i. Each neuron's sum of those products and a single bias term b are passed into a non-linear activation function $f()$ to then produce that neuron's activation, or

$$h_i(x) = f\left(\sum_j (x_j w_{i,j}) + b_i\right)$$

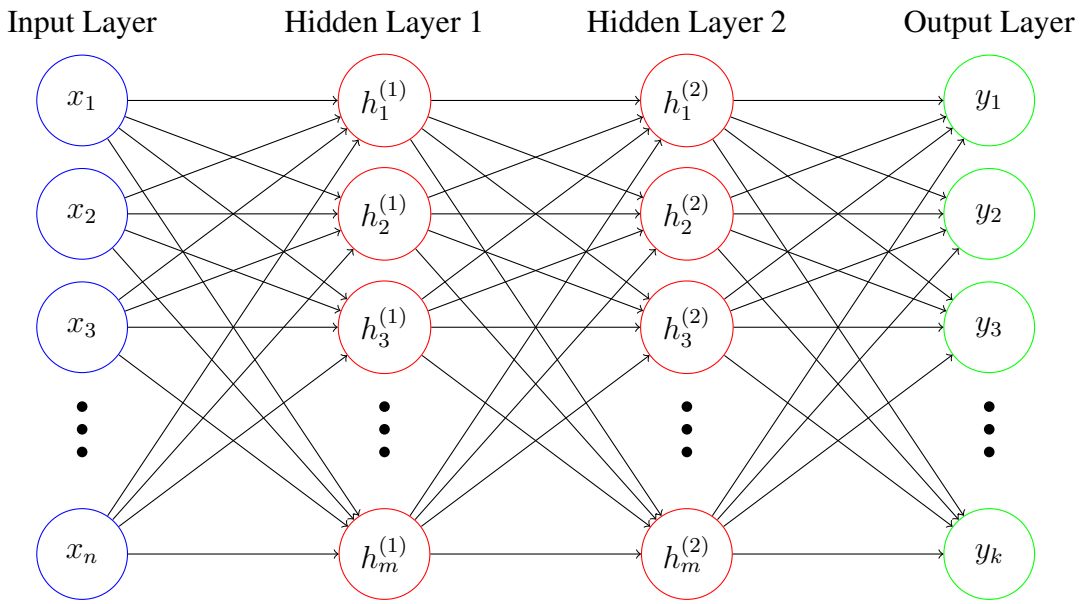


Figure 1.1: Illustration of a fully-connected feed-forward deep artificial neural network.

Many of these neurons are assembled to build a **hidden layer**. The output of a neuron is often called a **feature**, representing the features extracted from the input to be utilized by the model. If multiple hidden layers exist, the network is **deep**. Once feature vectors reach the **output layer** they are used to extract some useful or abstract information about the inputs. So, depending on the size of the input, the number of neurons per layer, and the number of layers in the model, this computation may require only a few hundred multiplications and additions.

This sort of neural network is not what the previously mentioned AlexNet uses, however. AlexNet is a **convolutional** neural network (CNN), as opposed to a fully-connected one. Convolution means that weights are *scanned* over an input image instead of being densely connected to every pixel. This means many fewer connections from inputs to weights, so fewer weights. This also means CNNs treat pixels which are spatially local differently than those which are far apart, similar to receptive fields in the brain's visual cortex [11]. Inspired by those receptive fields, the first CNN (dubbed the Neocognitron) was demonstrated in 1980 [12].

For the purposes of understanding this thesis, let's dig a little deeper. Again, each layer of a CNN consists of a collection of learned filters (i.e. weights) which are *convolved* over the input to

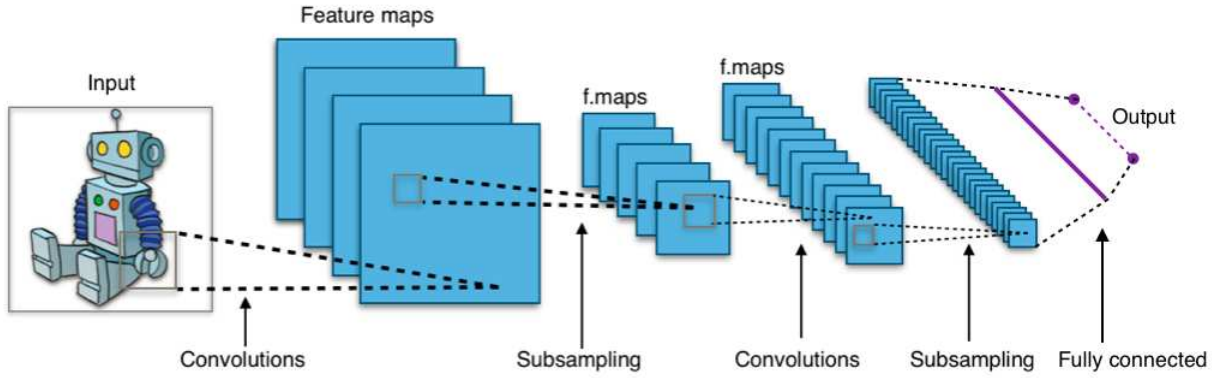


Figure 1.2: Illustration of a typical CNN, including subsampling or pooling operations. Each stack of filters makes up a convolutional layer, with many layers comprising the entire net[1].

produce spatially-sensitive activation maps as input to the next layer. See Figure 1.2 for an illustration. As before, these filters and activation maps exploit spatial locality between features. This allows low-level features extracted by earlier layers to be integrated into more complex or abstract visual information by later layers. The result is a hierarchy of visual features—generally growing more complex or abstract with depth. Most importantly, these inter-layer features represent what visual information is relevant to that model's particular task; they represent what the model has learned.

Once these visual features reach the end of the network, they can be used to predict something related to the input. In the case of classification, this means flattening visual features into **feature vectors**, and classifying those features. Nowadays this is done using a simple linear model, though AlexNet used two fully-connected layers. The resulting **logits** have a numeric entry for every label, and are commonly converted into bounded class probabilities using the **softmax** function.

Even with the advances provided by the CNN architecture, these models can still be computationally complex (especially for the 1980s). The next important advancement, then, came in the form of **backpropagation**, a method first applied to neural networks in 1986 [13]. In essence, backpropagation provided an efficient calculation of how every weight in the model contributes to the model's performance on a given input (i.e. the combination of each weight's **gradient**). In

[1]By Aphex34 - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=45679374

1989, the first backpropagation-trained CNN was trained to recognize hand-drawn digits, providing a foundation for modern computer vision [14].

Over the following decades, the availability of cheaper storage and more powerful computational resources finally set the stage for AlexNet. While it wasn't the first GPU implementation of a CNN, nor the first application of a CNN to an image recognition contest, AlexNet marked a massive leap in classification accuracy over previous techniques, thanks in large part to its depth [7]. The 60 million parameter, 7-layer model took 5 to 6 days to train using 2 consumer NVidia GPUs, and the widespread ILSVRC2012[2] dataset's 1.3 million hand-labeled images (i.e. ImageNet 2012). And yet, today AlexNet is far from the best-performing deep convolutional neural network architecture.

Since 2012, deep convolutional neural networks (CNNs) have dominated computer vision research. CNNs are the top performers in object recognition competitions [15] and have been applied to many other visual tasks including object detection and localization [16, 17], segmentation [18], pose estimation [19], gesture recognition [20], and visual saliency [21]. As a cognitive architecture, however, CNNs leave something to be desired, because they aren't so much a single architecture as they are a framework that can be used to instantiate many architectures. These architectures can vary in the number of filters per layer, the size and shape of filters, the connectivity between layers, the use of pooling and regularization layers, and beyond. Over the past 8 years, much of the computer vision literature can be thought of as a competition among labs to find the best architecture for vision within the deep convolutional framework. A snapshot of this effort as it relates to the popular ILSVRC2012 challenge is seen in Figure 1.3. Indeed, CNNs have been demonstrated which have 1001 layers [22], can be run on a smartphone [23], or have their architectures learned via reinforcement learning [4]. Despite all the effort invested in developing convolutional architectures, however, it's not clear how different from each other the best ones really are.

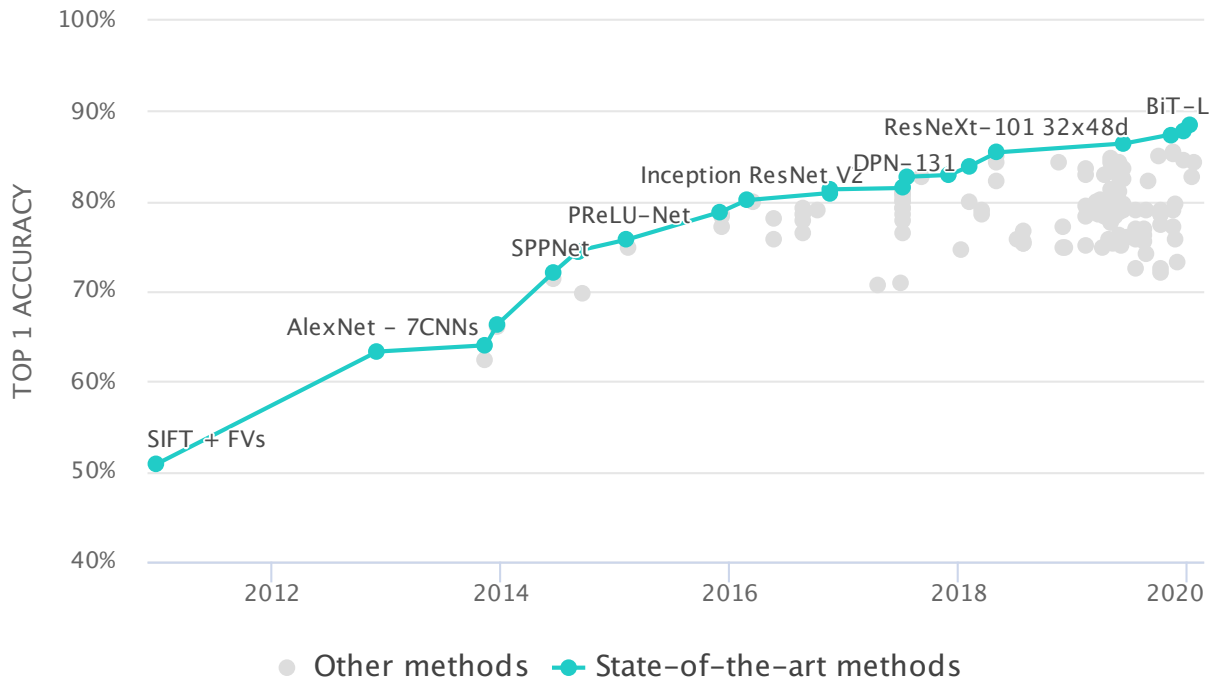


Figure 1.3: State-of-the-art accuracy on the ILSVRC2012 challenge since its inception in 2012. The green dots show the state-of-the-art methods, with other efforts shown in gray (note that not all methods are labelled)[3].

1.2 How different are these CNNs?

Take Inception and ResNet, for example. Inception [2] is a CNN-based architecture that divides processing by scale, merges the results, and repeats. Inception was originally developed by Google in 2014, and refined over the next 2 years [2, 24–26]. ResNet [1], on the other hand, has a simpler, single-scale processing unit with many more layers and a data pass-through between levels (an idea expanded on in DenseNet [27]). ResNet was developed at Microsoft in 2016 and also subsequently refined [1, 22]. In essence, Inception goes wide, while ResNet goes deep. Despite these disjoint pedigrees and conceptual differences in architecture, however, they perform similarly on the ILSVRC2012 image recognition challenge [15]. ResNet-v2 152 labels 78.9% of ILSVRC2012 test images correctly [22], while Inception-v4 labels 80.2% correctly [26]. Since these networks

[2]See Section 2.1 for more information on ImageNet and ILSVRC2012.

[3]By Atlas ML, CC BY-SA 4.0, https://paperswithcode.com/sota/image-classification-on-imagenet

were published in 2016, many have improved upon these numbers, with the state-of-the-art for ILSVRC2012 as of writing belonging to an EfficientNet architecture [28] trained using the Noisy Student method [29] achieving 88.4% accuracy. Still, how do we know that the (voluminous) effort invested into CNN architectures is actually impacting which features these models learn?

1.3 Affine Feature Equivalence

To better understand the differences introduced by different architectures, this thesis addresses the question of how similar different CNN architectures are to each other, not in terms of the number of ILSVRC2012 test images they correctly label, but in terms of the properties they extract from images. Ten popular models are used, all of which use a convolutional architecture to extract features from images, followed by a fully-connected classifier to assign labels to images based on the extracted features. The architectures used by each system to extract features are different, as are the numbers of features extracted (from 1,024 with Inception-v1 to 4,320 with PNASNet). These nets may be compared using various techniques (see Chapter 3 for a discussion of these). For example, visualization techniques have revealed that Inception and ResNet use qualitatively different visual information to discriminate amongst classes [30, 31].

Nonetheless, I find that the properties extracted by every CNN studied are very similar if not equivalent, in the sense that **an affine mapping can be used to predict the features of one CNN from the features of another**. The argument that affine mappings can reveal equivalence is discussed in Chapter 4. Indeed, these mappings are accurate enough that mapped features can even be used to label images without retraining the underlying classifier. This suggests that each of these 10 CNNs, despite their structural differences, exploit essentially the same properties of images. At the same time, a deeper analysis of these mappings suggests that while they may extract the same properties, some appear to do it more robustly than others, and may extract a few more properties.

On one hand, the finding that each of these 10 networks' features are linked by only affine transformations is surprising. CNNs are so powerful precisely because they are able to extract complex, non-linear features from low-level image data and synthesize them into high-level, even abstract

representations. The differences in architecture between these CNNs suggests that the non-linear features learned are quite different, since each CNN must learn using a different configuration of weights in order to succeed at its classification task. Essentially, each CNN must hill-climb in a totally different solution space. However, the results indicate the opposite. Since each one of these CNNs is related to every other by only a linear transformation, they must be extracting the same information.

On the other hand, a second experiment strongly suggests that this relationship has more to do with the layer *after* each feature space studied—the classification layer. In each of the 10 CNNs studied, this layer is also linear, meaning there is a linear relationship from each net's feature vectors to their logits[4]. Using only these classifiers, mappings can be constructed which perfectly translate features from one net's feature space to another's. While this doesn't reduce the degree of linear equivalence between CNNs (in fact it becomes perfect), this does strongly suggests that such equivalence is instead the direct result of each model learning to minimize the same loss. In essence, each net is trained to produce features which best complete the classification task, and each feature space is only a linear transformation from this shared task. Still, feature equivalence by prior layers has not been disproven. Indeed, the work by Lenc et al. demonstrates that prior layers do indeed contain some degree of overlapping information (as discussed in Section 3.3).

Regardless of the mechanism, this work may identify a sort of law of diminishing returns on the effort invested in designing better CNN architectures (at least for the sake of task performance). That is, if architecture differences always produce similar features, then effort should instead be invested on training data and procedure, such as data augmentation. Indeed, the latest two increases in task performance on ILSVRC2012 have been achieved using novel training techniques and data augmentation on established networks rather than adjustments to CNN architecture [29, 32].

[4]Recall that logits are the final features used for classification, often converted to bounded class probabilities using softmax.

Chapter 2

Background

This section gives the background information on ImageNet and each of the 10 CNNs studied which is necessary to fully understand this paper. Nothing in this section is novel, but it is included for completeness. Readers who are already familiar with this popular dataset and CNNs may choose to skip to Chapter 3.

2.1 ImageNet

The ImageNet project is a large, open database of images and videos for facilitating computer vision research [33]. The database consists of roughly 14 million images belonging to about 22,000 categories organized in a hierarchical fashion using WordNet [33]. About 1 million of these images include bounding boxes as well.

Since 2010, the project has provided a subset of the dataset for the annual ImageNet Large Scale Visual Recognition Challenge. The challenge now consists of object classification, detection, localization, and more [15]. Since AlexNet's splash in 2012, that year's challenge dataset (ILSVRC2012) has drawn continuous attention as a benchmark (see Figure 1.3) [15]. That particular dataset consists of 1000 classes, including animals, plants, foods, and various instruments. Notably, 120 of these classes are dog breeds. See Figure 2.1 for a few examples. While ILSVRC2012 remains a popular benchmark for new CNN architectures and training methods, it has also become a common starting point for the field of transfer learning (i.e. pretraining), thanks in part to the breadth of its features [34, 35].

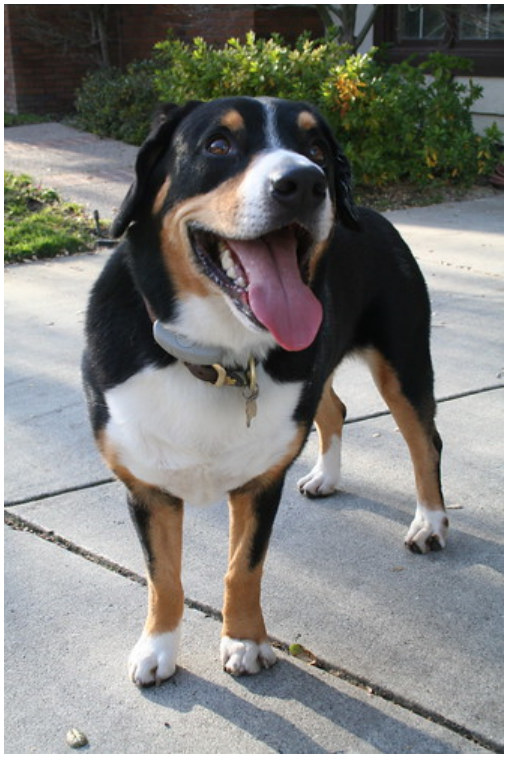
2.2 ResNet

In 2015, He, et al. introduced ResNet and the residual block (Fig. 2.2), consisting of two convolutional layers and a non-parameterized shortcut connection which passes the previous block's output to the next block [1]. This provided a leap in state-of-the-art performance on the ILSVRC2012

(a) Cardoon

(b) Cheeseburger

(c) Appenzeller



(d) Entlebucher

Figure 2.1: Examples of images in ILSVRC2012, including 2 of the 120 dog breeds.

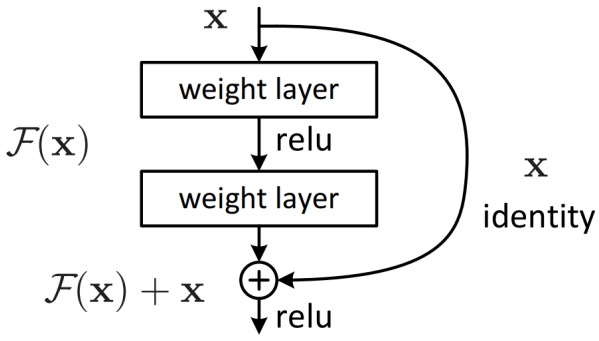


Figure 2.2: ResNet-v1 shortcut connections (taken from [1])

challenge using a 152-layer ResNet, and established the property that having more layers tends to produce higher recognition accuracy [22]. Continued improvements can be demonstrated with even $1,000$ layers—a previously unattainable feat [1]. Following this result, they modify the residual block so that ReLU activation is not applied to the shortcut connection, providing further improvement with ResNet-v2 in 2016 [22]. This simple structural feature has made its way into many other deep CNNs, providing broad success (e.g. DenseNet [27], Inception-ResNet-v2 [26]). ResNet-v2 152 labels 78.9% of samples correctly using a single model and 320x320 single-crop on the ILSVRC2012 challenge's 50k validation samples, remaining a high-performer today. In this thesis both versions of the 152-layer ResNet are studied: ResNet-v1-152 and ResNet-v2-152.

2.3 Inception

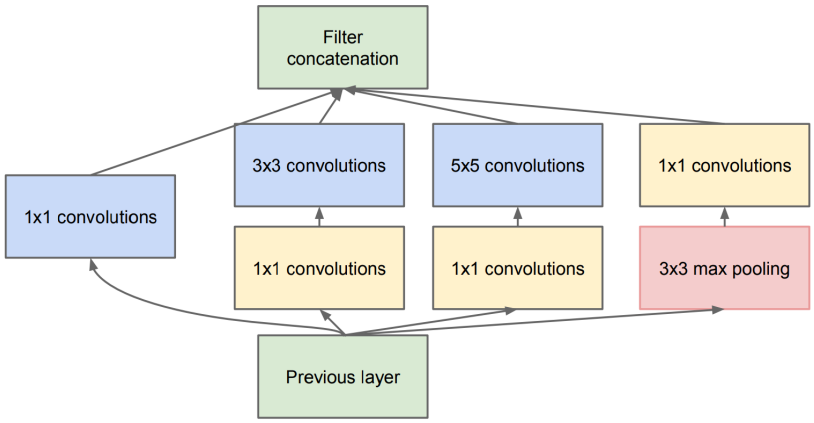


Figure 2.3: Inception-v1 module (taken from [2])

Szegedy et al. took a different approach, beginning with a core architecture for addressing scale variation, known as the Inception module (Fig. 2.3) [2]. These modules use a combination of convolutional layers at different scales and max pooling computed on the same input and concatenated together. Many of these modules are stacked together to create GoogLeNet (later Inception-v1). While the modular architecture was never abandoned, numerous refinements were made to these modules as the authors optimized for classification performance, training time, and computational

footprint [24–26]. Though many architectural features contribute to Inception's success, notable strategies include the use of 1×1 convolutions (a form of learned pooling for dimensionality reduction), factoring $n \times n$ convolutional layers into stacked $n \times 1$ and $1 \times n$ layers (a computational footprint reduction), and batch normalization (a technique for reducing covariance shift). The most recent Inception variant (Inception-v4) is still a high-performer on the ILSVRC2012 challenge dataset, correctly labelling 80.2% of ILSVRC2012 validation samples using a single model and 299x299 single-crop [26]. In this thesis all 5 versions of Inception are used: Inception-v1 through Inception-v4, and the hybrid Inception-ResNet-v2.

2.4 MobileNet

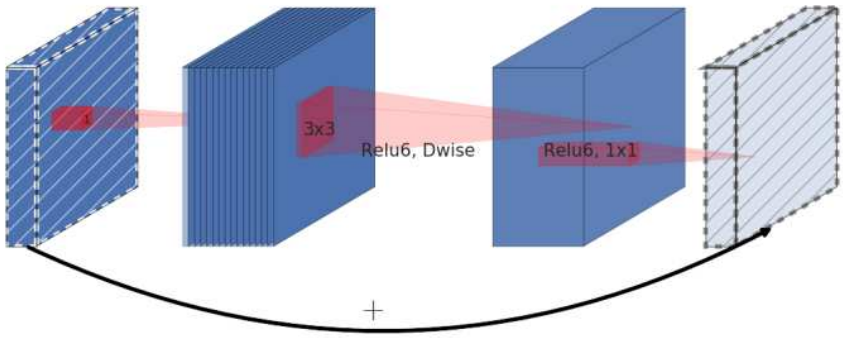


Figure 2.4: Illustration of convolutional layer used in MobileNet-v2 including depthwise separable convolutions, a shortcut connection, and linear bottleneck (hatched layers have no non-linearities), taken from [3].

In 2017, Howard et al. focused on a different challenge facing CNNs: efficiency [23]. They present the low-resource MobileNet-v1 architecture which takes advantage of multiple optimizations. The key architectural feature for this work is called a depthwise-separable convolution. Instead of a standard convolution which simultaneously filters and combines inputs at different depths (e.g. channel, for the first layer), a depthwise-separable convolution splits this operation into two steps, drastically reducing computational costs. This is achieved through "shallow" single-channel filters being convolved over each input channel, then combined with 1×1 convolutions (also called pointwise convolutions) as illustrated by the second and third blocks in Figure 2.4.

These models are further tuned for specific scenarios by use of two hyperparameters: a width multiplier for reducing both model size and cost, and a resolution multiplier for further reducing computational cost. Among many other experimental successes, a variant of MobileNet-v1 with a depth multiplier of 1.0 and a resolution of 224x224 beat Inception-v1 in performance, model size, and computational cost (MobileNet-v1-1.0-224).

This architecture was enhanced by Sandler et al. for even greater performance and efficiency as MobileNet-v2 [3]. The key changes made in MobileNet-v2 were the introduction of "linear bottleneck" layers and the addition of residual connections popularized by ResNet [1]. These additional features are shown in Figure 2.4. With these and other enhancements, a MobileNet-v2 variant outperforms the best MobileNet-v1 while using fewer parameters and computational power, achieving 74.7% accuracy on ILSVRC2012 [3]. In this thesis only this top-performing MobileNet-v2-1.4-224 variant is used.

In 2019, further optimizations were presented as MobileNet-v3, though these models are neither covered nor studied here [36].

2.5 (Progressive) Neural Architecture Search

Instead of hand-picking architectural features, Zoph et al. sought to use reinforcement learning to search for a better CNN [37]. Using a recurrent neural network (RNN), reusable "cell" structures are proposed before being placed into a predefined architecture for evaluation, consisting of fitting and evaluating the model on a smaller dataset (in this case CIFAR-10). After searching for 2,000 GPU-hours, cell structures are placed into a larger model, which resulted in a NASNet variant achieving state-of-the-art performance on ILSVRC2012 (82.7%) while using many fewer parameters than previous methods [37].

Also in 2018, Liu et al. took the same cell-learning approach from NASNet but achieved much higher efficiency during search [4]. This was thanks to two major optimizations: cell configurations were searched in order of increasing complexity, and the search was guided by a separate "surrogate" function trained alongside the search. With these enhancements, PNASNet achieved

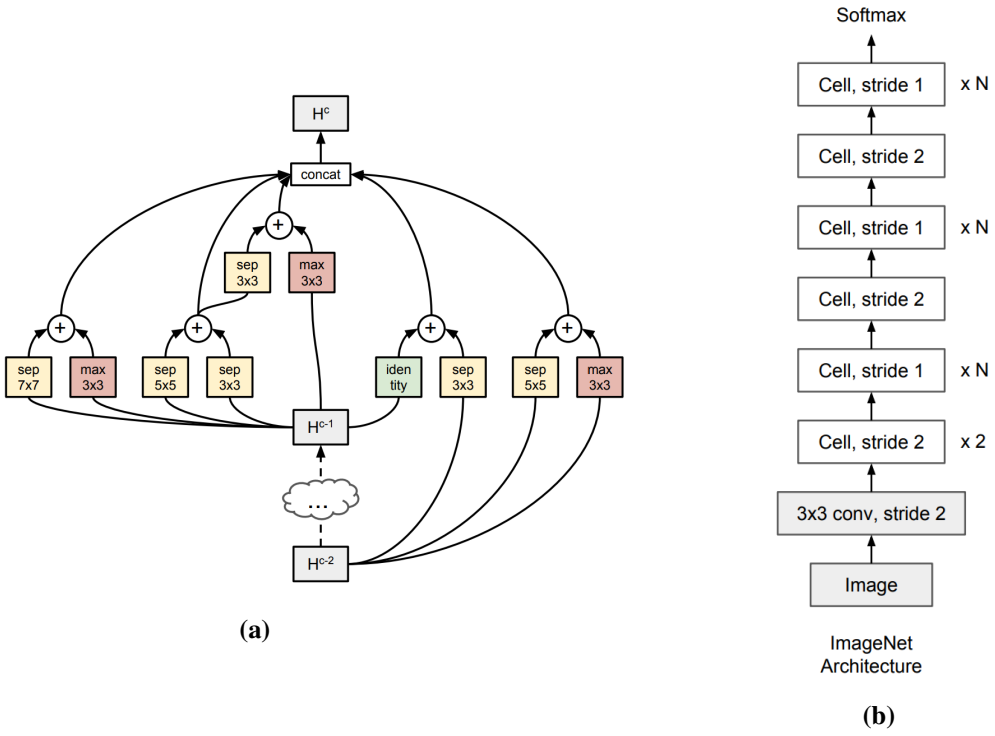


Figure 2.5: PNASNet cell (left) and complete ImageNet architecture (right), taken from [4]

state-of-the-art accuracy on ILSVRC2012 (82.9%) all while reducing the computation necessary for the search by at least 5x. See Figure 2.5 for the resulting cell and architecture found by [4].

Besides achieving state-of-the-art task performance, both papers also present a low-resource, "mobile" variant. The mobile variants are not studied in this thesis. Instead, both state-of-the-art models are used: NASNet-Large and PNASNet-Large (also called NASNet-A and PNASNet-5, respectively).

Chapter 3

Related Work

To deal with a 14-dimensional space, visualize a 3-D space and say 'fourteen' to yourself very loudly. Everyone does it. –Geoffrey Hinton

Alongside the strong interest in more powerful, lower resource CNNs, is a growing interest in comparing and understanding them. What follows is an overview of common methods of comparing and understanding CNNs, and how they relate to the work presented in this thesis.

3.1 Black box evaluation

The most common method for comparing deep learning systems is black box evaluation. A task is found for which there is a common dataset, and networks are evaluated to see which produces the lower error rate. Care must be taken in this mode of comparison to make sure that the finding is repeatable by others and not overfit to the particular dataset. In particular, one cannot simply take the entire dataset and train until error is minimized on the whole, since the model may have memorized the features in that dataset rather than the more desirable generic features common to a particular class. This "overfitting" issue is commonly mitigated by splitting the dataset into *test* and *train* partitions. The model is trained on the train set, using the test set for an unbiased evaluation [5]. Selection of the training and test sets are also important, then, since either partition may not be representative of the other. For datasets which are not pre-partitioned, there are multiple methods of achieving an unbiased indication of performance, such as k-fold cross-validation.

As mentioned before though, ImageNet—or ILSVRC2012 in particular—is the most common dataset for comparing image classification systems [15]. Recall Figure 1.3 and its source for notable submissions and state-of-the-art advances since the 2012 challenge. In the case of Ima-

[5]Sometimes a third *validation* set is used as well to guide training processes, such as selection of hyperparameters. The test set, however, remains unseen during any training/tuning processes.

geNet, the dataset is pre-partitioned into 1.3 million training images, and 50k testing[6] images [15]. Accordingly, image classification models are typically trained on the 1.3 million images in the training set to minimize incorrect predictions (i.e. classification error). Then, they are evaluated by predicting labels for each of the 50k test set images to produce a rate of correct prediction (i.e. classification accuracy). While significant gains in accuracy alone are often enough cause to publish, many model designers may choose to dive deeper.

For example, another common metric for comparison is top-n accuracy, or the rate at which the correct label occurred within the best n predictions for the model. In the case of ILSVRC2012, top-5 accuracy/error is typically reported [15]. Others may provide a confusion matrix, or a breakdown of class predictions by class label, identifying classes which the model commonly confuses.

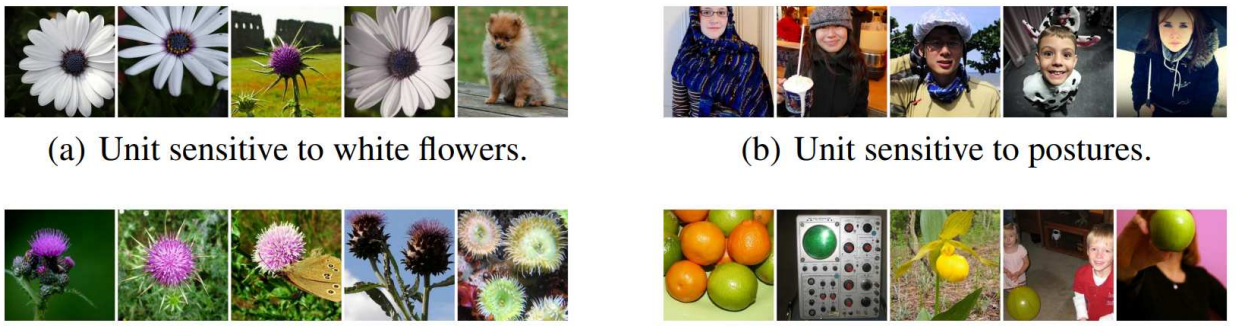
Black box evaluations are appropriate when the goal is to pick a system to minimize error rates, but they provide little information about the relative strengths and weaknesses of systems. When two systems perform similarly, as with many of the CNNs studied here, black box evaluations fail to disclose whether the two systems are doing essentially the same thing, or instead are very different and just happen to label roughly the same number of images correctly. Even when performance is changed, the underlying reason is not clear. Still, they can be useful for summarizing these highly complex models.

3.2 Visualization

While black box methods are useful for comparing otherwise mysterious, immensely complex models by abstracting away their millions of parameters, finer-grained analyses are possible. Visualization techniques enable researchers to understand what visual structures are relevant and how they are extracted. What follows is a brief overview of these techniques split into two major categories: those which ultimately use image **examples** to examine the model's learned features, and those which **generate** images which represent those learned features.

[6]Called the *validation* set by the ImageNet project. To be clear, the 50k *validation* samples are used for evaluation and model comparison, not training (so more similar to the *test* set described prior).

3.2.1 Correlation and Attribution



(a) Unit sensitive to white flowers.

(b) Unit sensitive to postures.

(c) Unit senstive to round, spiky flowers.

(d) Unit senstive to round green or yellow objects.

Figure 3.1: Images stimulating an individual neuron most strongly. Taken from [5].

One straightforward method for understanding what visual information a particular neuron responds to is to simply find which samples maximize its activation. These examples are often visually related to each other, giving some insight into the visual information relevant to a particular neuron. See Figure 3.1 and its source, [5], for examples and more analysis. However, this method does not reveal what *caused* the activation, only which inputs are *correlated* with it. Accordingly, others have produced saliency maps which show which particular parts of an image contribute most to a neuron's activation (or class prediction) by systematically occluding sections of the image as in [38]. This technique has been extended by instead correlating image regions with semantic concepts [30, 39–41] or channels (i.e. filters) [41, 42]. These methods allow for much greater intuition behind a CNN's class decision, providing for greater model interpretability.

Interestingly, Bau et al. used a method called Network Dissection (i.e. saliency maps sensitive to semantic concepts) to compare two of the CNNs studied in this thesis [30]. When comparing Inception-v1 and ResNet-v2-152, they found the former discriminates fewer semantic concepts than the latter.[7].

[7]Let me note here that Bau et al.'s result only adds to the surprise that all 10 CNNs studied in this thesis end up extracting the same features.

3.2.2 Optimization

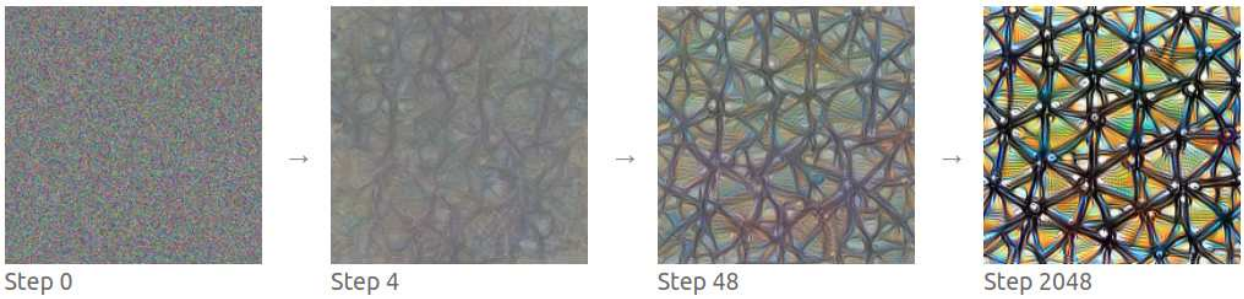


Figure 3.2: Illustration of an optimization-based technique. After many iterations, an activation-maximizing image is generated which reveals visual features learned by a particular channel (i.e. filter). Taken from [6].

Recall that backpropagation relies on a model whose weights are differentiable with respect to the error for a given input (i.e. how each weight contributes to the error function). For another popular class of techniques, this error gradient provides a foundation for visual analysis of CNNs via optimization. In essence, random noise is fed into a model which produces an error-sensitive gradient. By tweaking the input noise iteratively, the activation of a particular neuron (or class prediction) can be controlled in some way. Eventually, the input image settles on an often vivid, even abstract depiction of visual information relevant to that neuron [6].

This method was first proposed by Erhan et al. [43], and subsequently refined by Simonyan et al. [44]. Many have extended this method to produce higher-quality visualizations by using regularization [45, 46], or enforcing prior constraints (i.e. statistical similarity to realistic images) [47], among other techniques. This iterative optimization-based visualization technique is illustrated in Figure 3.2[8], which is taken from [6]. Still, while this analysis can reveal learned visual features, it does not clearly indicate whether two networks are using the same visual features for their task.

[8]This and many more excellent visualizations are available at https://distill.pub/2017/feature-visualization/appendix/googlenet/4a.html#4a-11, the appendix of [6].

3.3 Affine Feature Mappings

To the best of my knowledge, affine equivalence via CNN feature spaces mappings as presented in this thesis (and in my previous paper, [48]) is a new finding. However, the work by Lenc et al. contains many similarities [49]. In that work, as in this thesis, affine transformations are learned between CNNs trained on ILSVRC2012. In their case, the networks are AlexNet [7], VGG-16 [44] and ResNet-v1-50 [1]. They train mappings between the *convolutional* layers of these networks to determine feature compatibility. Because these mappings are dealing with spatially-sensitive convolutional layer output, they sometimes require interpolation. The transformations presented in this thesis, however, map between the output of the final convolutional layer of one network to the fully-connected classification layer of the other. Still, a varying degree of feature compatibility is found, and these features are not obviously related. More significantly, Lenc et al. used supervision in the form of image labels to train the mappings, effectively creating newly-trained networks. In this thesis, affine transformations are trained to predict one feature set from the other, **without supervision and independent of any semantic image labels**. Nonetheless, Lenc et al. indeed found linear similarity between the different CNNs.

In the field of face recognition, another relevant work is by Dong et al. [50]. By encoding faces into a hand-designed 50-dimensional parameter space (25 encoding face geometry, and 25 encoding face "appearance"), they are able to create affine maps to deep CNN feature vectors. While they also demonstrate surprising linear relationships to the output of presumably nonlinear CNN features, they do not study the linearity *between* CNN feature representations. Still, such linearity as I present may be suggested by their result, since if a linear relationship exists between their custom parameter space and different deep CNN feature spaces, then some linear relationship exists *between* CNN feature spaces (since the composition of linear transformations is a linear transformation). They also conclude that demonstrating this finding for general object-related tasks would be doubtful and very difficult, since their process involves hand-picked features.

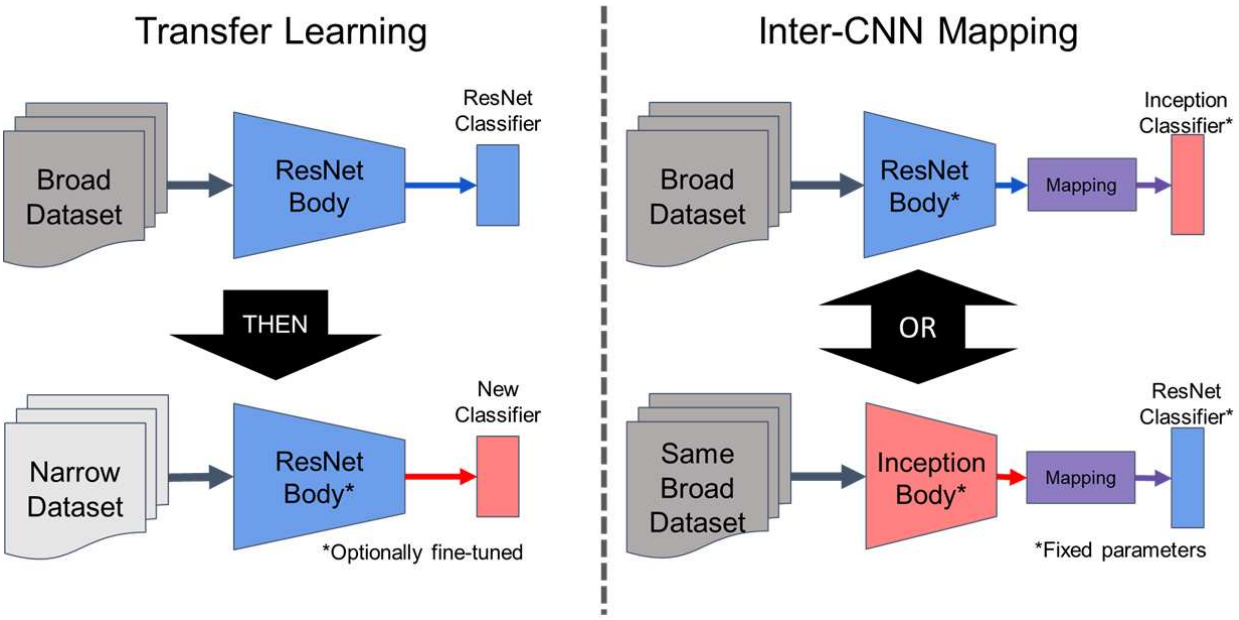


Figure 3.3: Transfer learning (left) contrasted with affine equivalence testing (right)

3.4 Transfer Learning

Transfer learning begins with a network trained on a large dataset covering a broad domain, and a new dataset which requires model adaptation. This approach is particularly useful for domains where data is scarce, as it has been observed that deep CNNs pretrained on ImageNet can identify features which are useful for narrower domains [35, 51]. These patterns can either be extracted as-is and correlated with new labels, as in fixed feature extraction, or fine-tuned to a narrower domain along with a new classifier, as in warm-starting. In contrast, my experimental architecture is a mapping *across* networks, not across domains and thus not transfer learning. To be clear, I only modify weights in the mapping, reusing the existing body and classifier, in contrast to building a new classifier from scratch and optionally fine-tuning the feature extractor. I also do not train using any ground truth classification labels, instead using a comparison of two feature vectors as loss. See Fig. 3.3 for a comparison of common transfer learning paradigms with my experimental approach.

Still, some use transfer learning to compare and contrast different CNN architectures. Indeed, initialization-based comparisons are appropriate for comparing CNNs with the same architecture

that were trained for different tasks. The idea is that if the tasks (and therefore the network parameters) are similar, one network should train more quickly and accurately when initialized using the trained weights from the other, similar task. In essence, CNNs are compared by the benefits of a warm start, as in [35], where CNNs which performed better on ImageNet tend to perform better when transferred to other datasets. While each CNN can be evaluated using this metric, doing so does **not** clearly reveal whether the underlying features used by those CNNs are necessarily *similar*.

Chapter 4

CNN Affine Equivalence[9]

How can we determine whether two CNNs are encoding the same information? As described in Chapter 3, task performance scores are often compared, though they do not reveal similarities in terms of the features extracted from images. Similarly, visualization techniques can reveal quite a lot, though finding equivalence is not trivial. Indeed, in the case of ResNet and Inception architectures, visualization techniques of early layers have revealed qualitatively different features [30, 31]. Therefore, a complete comparison of all features is not what is attempted here.

But the key question, and focus of this thesis, is to compare CNNs by the features they actually use to make decisions. For this, the feature vector is used, i.e. the pooled output of the final convolutional layer. In the task of image classification, these feature vectors are passed into one or more fully-connected layers known as the classifier to produce image class predictions. Unfortunately, feature vectors are not trivially interpretable nor comparable across different CNNs, with some represented in over 4000 dimensions. Further, even when feature spaces have the same number of dimensions, they are not compatible between networks (see Section 5.6). Accordingly, my aim is to compare CNNs using a complete representation of their learned features.

4.1 Proposing Affine Equivalence

Given CNNs A and B, then, how can a pair of feature vectors x_A and x_B generated from the same image be compared? How can it be determined whether they are encoding the same information? Consider the idea of overlapping information in those feature vectors—i.e. information is stored in both representations, even if the way that information is stored is not equivalent between the two. If the feature vectors did not contain overlapping information, then finding a mapping between the two would require additional information. Thus, if a mapping can be created

[9] Not to be confused with proper affine equivalence in the context of boolean functions.

which accurately predicts x_B given a general x_A without additional information, they must encode overlapping information.

That said, the complexity of that mapping may enhance or degrade this conclusion. If the mapping is highly nonlinear, for example, one could say that the feature spaces are correlated, but don't share a representation. After all, each feature vector is a highly nonlinear transformation of the input image. However if the mapping is linear, then these feature spaces are very similar, since a linear mapping is homomorphic—i.e. it preserves the structure between points. In this work then, affine mappings (i.e. linear with a constant bias) between feature spaces are fit and studied.

To summarize, I argue that two sets of vectors which are related by only an affine mapping are, for the purpose of comparing CNN feature representations, equivalent. Further, I propose that if an affine transformation can be constructed which can accurately predict x_B given a general x_A, then the degree of that accuracy is an indicator of the similarity between the encoded features used by both CNNs. To be precise, an affine mapping is

$$\begin{bmatrix} \tilde{x}_B \\ \mathbf{1} \end{bmatrix} = M_{A \to B} \begin{bmatrix} x_A \\ \mathbf{1} \end{bmatrix} \tag{4.1}$$

where $M_{A \to B}$ is the mapping matrix and $\tilde{x}_B$ is the prediction generated from x_A.

4.2 Demonstrating Affine Equivalence

So far the calculation of this mapping accuracy has been left undefined. It could be calculated as simply as a distance between $\tilde{x}_B$ and x_B , though that may be misleading. At least in the case of image classification, CNNs are not necessarily trained to produce feature vectors which are related by distance. Recall that for this task, an image is passed into a feature extractor F, producing a feature vector x which is passed to a classifier C consisting of one or more fully-connected layers and softmax. The output of that classifier, $C(x) = y$, is compared to the ground truth image label $\hat{y}$ (the actual source of training loss). The number of correct labels evaluated over a large validation set is then used to produce classification accuracy, a. See the first two rows of Figure 4.1 for

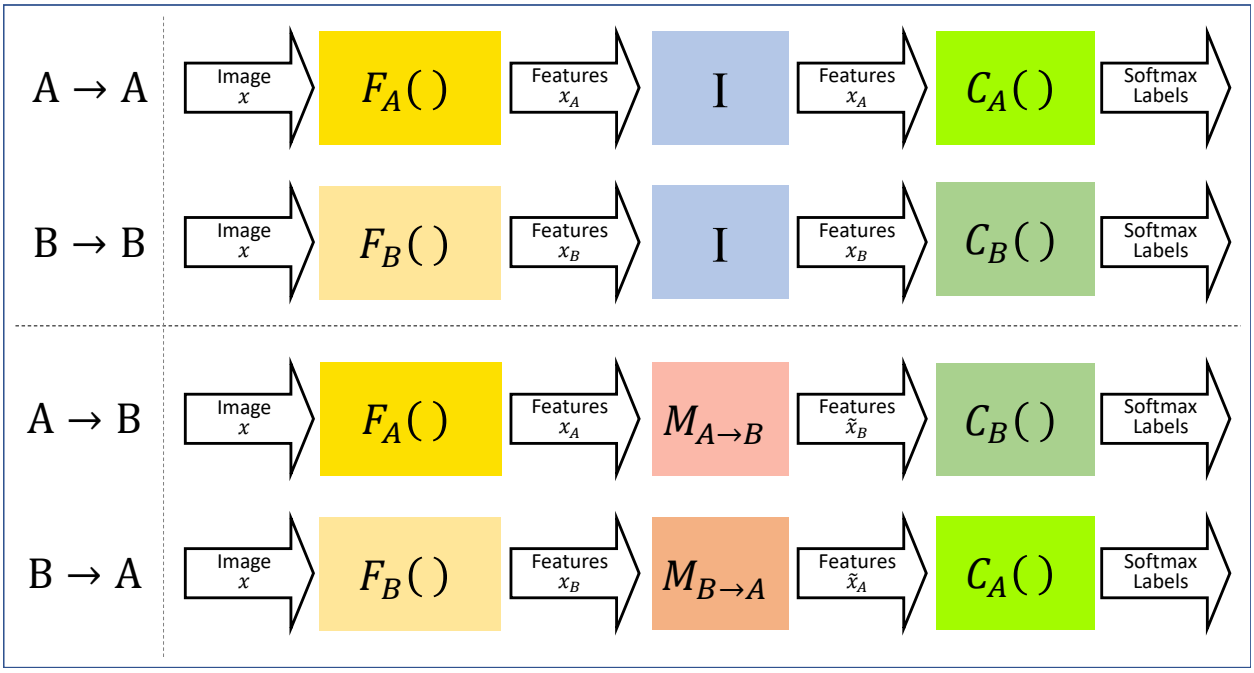


Figure 4.1: Illustration of how a pair of standard CNNs can be used to create two alternative CNNs where the features from one are affine mapped to the classifier of another. The notation used in this figure, in particular $F()$ and $C()$ to capture the mapping from input to the feature space and then feature space to final classification softmax is further described below. The first two rows show CNNs A and B without alteration; the mapping from $F()$ to $C()$ is the identity mapping I. The next two illustrate the swapping of classifiers and the introduction of affine mappings $M_{A\to B}$ and $M_{B\to A}$.

this standard configuration, which should be familiar to most readers, minus the *identity* mappings represented by blue rectangle labeled "I".

So, mapping accuracy is calculated in a similar fashion. For a given predicted feature vector $\tilde{x}_B$, an image label prediction is produced by passing the predicted feature vector through the target CNN's classifier, or $C_B(\tilde{x}_B)$. This predicted label $\tilde{y}_B$ is then compared to the ground truth image label $\hat{y}$ to produce the classification accuracy of mapped feature vectors, or $\tilde{a}_B$. See the lower two rows of Figure 4.1 for this configuration. This can then be compared to the classification accuracy of the vanilla, unmapped feature vectors as a_A or a_B (the top two rows of Figure 4.1). This mapping evaluation process is concretely described in Section 5.3. Then, I argue if the classification performance of one CNN's classifier using another CNN's features is near the classification performance of either vanilla, unmapped CNN, then those two CNNs are extracting effectively equivalent visual information from the input images.

Chapter 5

Affine Equivalence of 10 CNNs

Using the method described in Chapter 4, mappings were fit between 90 unique pairs of 10 CNN architectures (100 pairs are possible, though 10 of those would produce identity mappings).

5.1 Sourcing and Validating Pre-trained CNNs

Each of these models were trained by Google on the 1.3 million training samples in ILSVRC2012, and obtained from TensorFlow Hub[10] with the exception of Inception-v4, which was obtained from the TensorFlow-Slim GitHub repository[11]. While there are many notable others, these architectures were selected for their generally 1) distinct pedigrees and 2) ease of availability through TensorFlow Hub. See Table 5.1 for their respective top-1 single-crop classification accuracies on the 50k ILSVRC2012 validation samples both reported by Google and observed by me. Note: a crop size of 331x331 is used (except in the case of MobileNet-v2 which is only available pretrained with a maximum crop size of 224x224) to minimize the difference in features encoded into each network's feature vectors. This means most networks actually perform better than reported by Google, and none perform worse (again except, marginally, MobileNet-v2).

5.2 Solving for Affine Maps

With pretrained models validated, affine maps can be trained. Let X_T be the set of 1.3 million ILSVRC2012 training samples, and y_t be their respective image class labels. Similarly, let X_V and y_v be the 50k ILSVRC2012 validation[12] samples and labels. Given "source" CNN A composed of feature extractor F_A and classifier C_A, compute the set of training feature vectors T_A by feeding training images X_T into CNN A's feature extractor F_A. Given "target" CNN B similarly compute

[10] https://tfhub.dev/

[11] https://github.com/tensorflow/models/tree/master/research/slim

[12] Again, this is typically used as the *test* set, despite being named the *validation* set by the ImageNet project.

Table 5.1: Classification accuracies of the 10 CNNs studied on the ILSVRC2012 validation set, both reported by Google and independently observed locally by me (with respective crop sizes). Also included for reference are the number of dimensions used in each CNN's feature space, and the total number of parameters present in the model.

CNN	Reported	Reported Crop	Local	Local Crop	# dims	# params
Inception-v1	69.8%	224x224	71.1%	331x331	1024	5M
Inception-v2	73.9%	224x224	73.9%	331x331	1024	11M
MobileNet-v2-1.4-224	74.9%	224x224	74.6%	224x224	1792	7M
ResNet-v1-152	76.8%	224x224	78.8%	331x331	2048	60M
ResNet-v2-152	77.8%	224x224	78.7%	331x331	2048	60M
Inception-v3	78.0%	299x299	78.9%	331x331	2048	24M
Inception-v4	80.1%	299x299	80.4%	331x331	1536	43M
Inception-ResNet-v2	80.4%	299x299	81.2%	331x331	1536	56M
NASNet-Large	82.7%	331x331	82.7%	331x331	4032	89M
PNASNet-Large	82.9%	331x331	82.9%	331x331	4320	86M

T_B by feeding X_T into F_A. Then, compute an (augmented) affine mapping matrix $M_{A\to B}$ by ordinary (linear) least squares regression, minimizing the sum of distances between targets T_B and mapping predictions. Or, given $t_{A,i} \in T_A, t_{B,i} \in T_B$, the feature vectors corresponding to training sample i for network A and B, respectively, find the $M_{A\to B}$ such that

$$\sum_{i=1}^{|X_T|} \left\| M_{A\to B} \begin{bmatrix} t_{A,i} \\ 1 \end{bmatrix} - \begin{bmatrix} t_{B,i} \\ 1 \end{bmatrix} \right\|^2 \tag{5.1}$$

is minimized. This completes the training of an affine mapping between CNN A's feature space to CNN B's. This process is completed for each permutation of those 10 pre-trained CNNs to produce 90 pairs or 90 mappings (again omitting 10 identity mappings). Observe that the ground truth image labels, y_T are never used in the mapping training process. Also note that the number of dimensions in each CNN's feature space is not necessarily equal.

5.3 Evaluating Affine Maps

With mappings trained, I can now evaluate them and the greater affine equivalence between these 10 CNNs. To evaluate the mapping from source CNN A to target CNN B, validation images

X_V are fed into both feature extractors to produce feature vector sets V_A and V_B. Then, each of V_A can be mapped into CNN B's feature space by simply multiplying it by the trained mapping $M_{A \to B}$ to produce $\tilde{V}_B$, or CNN A's features mapped into CNN B's feature space. As discussed in Chapter 4, mapped feature vectors (and thus mappings) are evaluated using the same method as a typical image classification CNN's feature vectors. That is, they are passed through the final classification layer of that CNN, or in this case C_B. The resulting logits are passed through softmax, from which the maximum value is used as prediction. These predictions are compared to ground truth image labels y_V to produce a mapped classification accuracy $\tilde{a}_{A \to B}$. Each of the 90 mappings are evaluated this way.

5.4 Mapping Performance

See the performance of each mapping (and unmapped CNN) in Table 5.2. Row headings indicate the model whose feature extractor is used as the feature vector source, while column headings indicate the model whose classifier is used as destination. Each cell contains the classification accuracy achieved by the independently trained mapping between feature extractor and classifier. So, values along the diagonal are identity mappings, and indicate the accuracy of the unmodified source CNN (i.e. the same values from Figure 5.1). The small number indicates the percent change in accuracy from the unmodified **source** CNN (i.e. the value in that row which belongs to the diagonal). Cells are shaded according to this percent change, so that the darker the shade, the greater performance penalty introduced by the mapping. All accuracies are reported using the ILSVRC2012 validation set, which was unseen by all CNNs and mappings during all training procedures.

While the reduction in classification accuracy introduced by each mapping is never zero, the greatest is only an 11.6% reduction. Comparing across feature extractors (row-wise), mappings from the Inception-v1 and MobileNet-v2-1.4-224 feature vectors incur a consistently larger penalty (though no greater than 11.6%). This may be because these two models are the smallest, consisting 5 million and 7 million parameters, respectively. Still, every mapping incurs relatively little penalty

Table 5.2: Classification accuracies and penalties of 100 inter-CNN affine maps (i.e. 10 CNNs mapped to each of 10 CNNs). Each cell represents a single, independent affine mapping between the feature extractor from the **row CNN** and the classifier from the **column CNN**. The number in large font in each cell indicates the classification accuracy of this hybrid CNN on the 50k ILSVRC2012 validation set. Note: this means the diagonal shows the performance of an identity mapping, equivalent to the performance of unmapped CNNs in Table 5.1. The number in small font indicates the percent change from the unmapped row/source CNN (i.e. the value in that row which belongs to the diagonal). The darker the shade of the cell, the greater the performance penalty introduced by the mapping, relative to the feature extractor's own classifier. For example, the cell in the Inception-v4 row and ResNet-v2-152 column represents a mapping between Inception-v4's features and ResNet-v2-152's classifier, which produces 79.57% accuracy on the validation set, a -1.01% change from Inception-v4 alone.

		Target (classifier)									
Source (feature extractor)		Inception V1	Inception V2	MobileNet V2 1.4 224	ResNet V1 152	ResNet V2 152	Inception V3	Inception V4	Inception ResNet V2	NASNet Large	PNASNet Large
	Inception V1	71.06% 0.0%	62.82% -11.6%	68.17% -4.1%	68.45% -3.7%	68.31% -3.9%	67.36% -5.21%	66.77% -6.04%	65.86% -7.32%	64.97% -8.58%	65.40% -7.97%
	Inception V2	69.91% -5.44%	73.94% 0.0%	72.65% -1.74%	72.73% -1.63%	72.66% -1.73%	72.34% -2.17%	72.04% -2.57%	71.55% -3.24%	71.01% -3.97%	71.24% -3.65%
	MobileNet V2 1.4 224	68.17% -8.62%	66.27% -11.16%	74.60% 0.0%	71.57% -4.06%	71.25% -4.49%	70.56% -5.42%	69.93% -6.25%	69.43% -6.93%	69.16% -7.30%	69.72% -6.54%
	ResNet V1 152	73.98% -6.09%	73.42% -6.80%	76.94% -2.33%	78.78% 0.0%	77.04% -2.21%	76.47% -2.93%	76.08% -3.42%	75.86% -3.70%	75.26% -4.46%	75.12% -4.65%
	ResNet V2 152	74.79% -4.97%	74.42% -5.45%	77.19% -1.92%	77.79% -1.16%	78.70% 0.0%	76.75% -2.48%	76.35% -2.98%	76.10% -3.31%	75.38% -4.22%	75.44% -4.15%
	Inception V3	75.57% -4.19%	75.55% -4.22%	77.91% -1.22%	77.9% -1.24%	77.62% -1.59%	78.88% 0.0%	77.63% -1.58%	77.50% -1.75%	77.12% -2.23%	77.17% -2.17%
	Inception V4	78.29% -2.61%	78.49% -2.37%	79.84% -0.69%	79.75% -0.79%	79.57% -1.01%	79.79% -0.74%	80.39% 0.0%	79.68% -0.88%	79.64% -0.93%	79.61% -0.97%
	Inception ResNet V2	79.63% -1.89%	79.63% -1.89%	80.71% -0.55%	80.80% -0.44%	80.52% -0.78%	80.69% -0.57%	80.82% -0.41%	81.16% 0.0%	80.77% -0.48%	80.69% -0.58%
	NASNet Large	81.00% -2.01%	81.30% -1.65%	82.43% 0.28%	82.32% -0.41%	82.25% -0.50%	82.42% -0.29%	82.61% -0.06%	82.58% -0.10%	82.66% 0.0%	82.65% -0.01%
	PNASNet Large	81.16% -2.15%	81.40% -1.86%	82.62% -0.39%	82.52% -0.52%	82.46% -0.59%	82.72% -0.27%	82.80% -0.18%	82.77% -0.20%	82.84% -0.13%	82.94% 0.0%

when used to transform between different feature spaces. These mappings are effective enough to retain the vast majority of features' expressiveness, even when used for classification.

Comparing across classifiers (column-wise), mapped feature vectors which are classified by Inception-v1's and Inception-v2's classifiers incur the largest and most consistent penalties. Again there may be many explanations, though these two models uniquely use the smallest feature space at 1024 dimensions. More interestingly, when comparing column-wise (feeding different features into the same classifier), accuracies are sometimes *increased* when compared to the classifier CNN's own features. That is, even though the mappings were trained to reproduce the classifier CNN's less expressive features, features of greater discriminative power still get through. As an example, see ResNet-v2-152's column and unmapped accuracy of 78.70%. This unmapped accuracy is taking ResNet-v2's features and passing them into ResNet-v2's classifier. When instead fed a linear transformation of PNASNet's features, ResNet-v2's classifier produces 82.46% accuracy. This is an increase in the ResNet-v2 classifier's accuracy of 3.76%! Of course, when analyzing the effects of mappings on the *source* CNN's performance (i.e. the fashion that cells were shaded in Table 5.2), no feature vectors actually become more discriminative when mapped. Still, the important conclusion is that much of the information required by ResNet-v2's classifier is present in PNASNet's features, though more information is likely captured in PNASNet's features.

The central question of this thesis is whether the features learned by each CNN studied are equivalent. In every case, the percent change in classification accuracy introduced by using another CNN's features is no worse than -11.6% (and in most cases, much better). Indeed, the median percent change over all mappings is only -1.90%. This strongly suggests that the features learned by one CNN are also learned by every other, though some have learned these features more robustly. Indeed, some CNNs seem to have learned to express features nearly *identically*, such as NASNet and PNASNet which are penalized by only 0.01% when using the former's features in the latter's classifier.

Table 5.3: Classification accuracies and penalties of 100 inter-CNN affine maps (i.e. 10 CNNs mapped to each of 10 CNNs). The table format is identical to Table 5.2. The only difference between the two is in the creation of the mappings, as outlined in Section 5.5. Note that this table presents mappings which produce no performance penalty compared to unmapped features.

		Target (classifier)									
Source (feature extractor)		Inception V1	Inception V2	MobileNet V2 1.4 224	ResNet V1 152	ResNet V2 152	Inception V3	Inception V4	Inception ResNet V2	NASNet Large	PNASNet Large
	Inception V1	71.06%	71.06%	71.06%	71.06%	71.06%	71.06%	71.06%	71.06%	71.06%	71.06%
	Inception V2	73.94%	73.94%	73.94%	73.94%	73.94%	73.94%	73.94%	73.94%	73.94%	73.94%
	MobileNet V2 1.4 224	74.60%	74.60%	74.60%	74.60%	74.60%	74.60%	74.60%	74.60%	74.60%	74.60%
	ResNet V1 152	78.78%	78.78%	78.78%	78.78%	78.78%	78.78%	78.78%	78.78%	78.78%	78.78%
	ResNet V2 152	78.70%	78.70%	78.70%	78.70%	78.70%	78.70%	78.70%	78.70%	78.70%	78.70%
	Inception V3	78.88%	78.88%	78.88%	78.88%	78.88%	78.88%	78.88%	78.88%	78.88%	78.88%
	Inception V4	80.39%	80.39%	80.39%	80.39%	80.39%	80.39%	80.39%	80.39%	80.39%	80.39%
	Inception ResNet V2	81.16%	81.16%	81.16%	81.16%	81.16%	81.16%	81.16%	81.16%	81.16%	81.16%
	NASNet Large	82.66%	82.66%	82.66%	82.66%	82.66%	82.66%	82.66%	82.66%	82.66%	82.66%
	PNASNet Large	82.94%	82.94%	82.94%	82.94%	82.94%	82.94%	82.94%	82.94%	82.94%	82.94%

5.5 Logit Compatibility

As mentioned in the introduction, another experiment was conducted. This experiment was based on an idea proposed recently by one of my committee members, Dr. Charles Anderson. To start, mappings are created which encode features into logits using their typical classifier (C_A). Recall that in every CNN studied here, a classifier is simply a 2D matrix of weights—a linear transformation. These logits are then decoded using the pseudoinverse of another CNN's classifier (C_B^{-1}) to produce mapped features. More precisely, a mapping $M_{A \to B}$ between CNN A and CNN B can be constructed simply as

$$M_{A \to B} = C_A C_B^{-1}.$$

Those mapped features can be classified by C_B to produce an accuracy as described previously.

Simply put, this mapping technique tests how much relevant information might be lost by converting features to and from logits[13]. See Table 5.3 for the results of this alternative mapping technique presented in the same format as Table 5.2. In every case, these mappings are able to perfectly convert features from one feature space to another. Since each classifier was found to be full-rank, converting to and from the logit representation is lossless. In turn, this enables lossless feature space mappings as described above, essentially converting the source features to logits, then to the target feature space. Intuitively, converting between features via logits is bound to work well when those features are trained for classification as logits, and those mapped features are only evaluated as logits (i.e. evaluated via classification accuracy).

This result strongly suggests that these feature spaces are equivalent because of their equivalent tasks—their equivalent loss functions during training. Each network is trained to extract visual features which are relevant to a particular class (the feature space), but those features are all eventually converted into class probabilities (softmaxed logits). Because each network is trained to produce the same logits for the same inputs, and feature spaces are a lossless encoding of logits, it's much less surprising that these feature spaces are equivalent.

[13]Recall that logits are the final output of a classification model, represented as a sequence of "scores" for each class. Logits are usually converted into bounded class probabilities via softmax.

5.6 Just Checking

For the sake of completeness, I have conducted another experiment. Instead of mapping between feature spaces using an affine transformation, feature vectors were simply passed directly from one CNN's feature extractor to another CNN's classifier. When the number of dimensions didn't match (see Table 5.1), vectors were naively truncated or padded with zeros. The best accuracy achieved in this fashion is 0.27%, which is only slightly better than random guessing ($1/1000 = 0.1\%$).

This makes clear that these feature vectors are not directly compatible, since no hybrid configuration achieves performance much better than random. Even among very similar architecture pairs like ResNet-v1 and ResNet-v2, or Inception-v1 and Inception-v2 where dimensionality is unchanged, feature vectors are not directly interchangeable. Indeed, each CNN consists of millions of stochastically initialized and trained parameters, so each independent training will almost certainly produce a unique feature space.

Still, despite naive incompatibility, the discovery of a structure-preserving unsupervised affine mapping has revealed that feature vectors obtained from different CNNs are very much related to each other. In the extreme case where the mapping is computed directly via shared logit space (Section 5.5), the mapping is precise. In the case where the mapping is derived from comparisons between features for known common images, the affine equivalence is not perfect, but it is an excellent approximation in practice (Section 5.4). One reason this finding is so important is that it reveals the extent to which feature spaces from very different CNNs trained on the same data end up representing essentially the same information relative to object classes.

Chapter 6

Conclusion

Convolutional neural networks have catalyzed the field of computer vision, providing the basis for models which have made massive gains in performance over previous methods [15]. The application of these massive models to large datasets continues to facilitate growth in the abilities of AI, propelling a burgeoning multi-billion dollar industry [8]. While the size and complexity of CNN-based models enables their power, they can also obscure their crucial underlying features.

Equivalence between CNN features was sought in order to see whether CNNs capture different but roughly equally discriminative information, or the same information but using a different architecture and encoding to do it. This thesis has demonstrated two relatively simple methods for CNN comparison which reveal a fundamental property of modern ImageNet-trained CNN-based models: linear equivalence. Linear mappings can be constructed between 10 CNNs of varying architecture, pedigree, and performance with zero penalty in classification accuracy. These findings suggest that these networks are, in fact, extracting qualitatively the same features. A latter experiment also demonstrates this equivalence, but with an emphasis on the linear equivalence of each feature space to a common logit space. Even so, linear equivalence between CNN outputs is, to the best of my knowledge, new.

I speculate that this has implications for other CNNs as well. As they grow more complex, there may be a law of diminishing returns if all nets end up extracting similar features, though that may be what happens because that is what the training process and dataset supports. As mentioned in Section 1.2, the highest-performing CNNs on ImageNet are indeed the result of advances in training processes and data preprocessing using previously-published architectures [29, 32].

6.1 Future Directions

If these findings are indeed general for ImageNet-trained CNNs, it also suggests there is some common subset of features intrinsic to ImageNet data. More experiments and analysis could be

conducted to determine if linear equivalence remains when CNNs are not trained with a common loss function. Regardless of the source of linear equivalence, further analysis may reveal a reduced-dimension "canonical space", which effectively expresses the bulk of these ImageNet features. Others have indeed found a reduced "intrinsic space" of fewer than 20 dimensions for CNNs trained on face identification datasets and an ImageNet subset [52]. A systematic analysis of these affine maps may be useful for characterizing ImageNet-trained CNN feature spaces.

In the same vein, variations in the training data could have an effect on linear equivalence. Each network studied here was trained on the same dataset. It's not clear then whether the finding is consistent in other image classification domains, or even other problem domains. Additionally, the CNNs used in this work were all trained using similar or equivalent processes. Again, recent advances in ImageNet classification performance have been achieved without changes to prior architectures, instead relying on extra data and novel training techniques [29, 32]. A future investigation could take this into account, seeking the tipping point at which variations in training input and strategy prevent similarity between feature spaces as observed here.

As Lenc et al. have demonstrated by creating linear mappings between convolutional layers, there is much more work to be done to characterize and explain these intrinsic linear relationships.

Bibliography

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[2] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2016.

[3] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[4] Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 19–34, 2018.

[5] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[6] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2017. https://distill.pub/2017/feature-visualization.

[7] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[8] MarketsandMarkets Research Private Ltd. Artificial intelligence market by offering (hardware, software, services), technology (machine learning, natural language process-

ing, context-aware computing, computer vision), end-user industry, and geography - global forecast to 2025. Available at https://www.marketsandmarkets.com/Market-Reports/artificial-intelligence-market-74851580.html (2020/01/21).

[9] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[10] Alekseĭ Grigor'evich Ivakhnenko and Valentin Grigor'evich Lapa. Cybernetics and forecasting techniques. 1967.

[11] David H Hubel and Torsten N Wiesel. Receptive fields and functional architecture of monkey striate cortex. *The Journal of physiology*, 195(1):215–243, 1968.

[12] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 36(4):193–202, 1980.

[13] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[14] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[15] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[16] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[17] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[18] Fayao Liu, Guosheng Lin, and Chunhua Shen. Crf learning with cnn features for image segmentation. *Pattern Recognition*, 48(10):2983–2992, 2015.

[19] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[20] Pradyumna Narayana, Ross Beveridge, and Bruce A Draper. Gesture recognition: Focus on the hands. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5235–5244, 2018.

[21] Guanbin Li and Yizhou Yu. Visual saliency detection based on multiscale deep cnn features. *IEEE Transactions on Image Processing*, 25(11):5012–5024, 2016.

[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016.

[23] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[24] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[25] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[26] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[27] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[28] Mingxing Tan and Quoc V Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*, 2019.

[29] Qizhe Xie, Eduard Hovy, Minh-Thang Luong, and Quoc V Le. Self-training with noisy student improves imagenet classification. *arXiv preprint arXiv:1911.04252*, 2019.

[30] David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6541–6549, 2017.

[31] Maximilian Alber, Sebastian Lapuschkin, Philipp Seegerer, Miriam Hägele, Kristof T Schütt, Grégoire Montavon, Wojciech Samek, Klaus-Robert Müller, Sven Dähne, and Pieter-Jan Kindermans. iNNvestigate neural networks! *Journal of Machine Learning Research*, 20(93):1–8, 2019.

[32] Hugo Touvron, Andrea Vedaldi, Matthijs Douze, and Hervé Jégou. Fixing the train-test resolution discrepancy. *arXiv preprint arXiv:1906.06423*, 2019.

[33] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[34] Minyoung Huh, Pulkit Agrawal, and Alexei A Efros. What makes imagenet good for transfer learning? *arXiv preprint arXiv:1608.08614*, 2016.

[35] Simon Kornblith, Jonathon Shlens, and Quoc V Le. Do better imagenet models transfer better? *arXiv preprint arXiv:1805.08974*, 2018.

[36] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1314–1324, 2019.

[37] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8697–8710, 2018.

[38] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

[39] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2921–2929, 2016.

[40] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

[41] Chris Olah, Arvind Satyanarayan, Ian Johnson, Shan Carter, Ludwig Schubert, Katherine Ye, and Alexander Mordvintsev. The building blocks of interpretability. *Distill*, 2018. https://distill.pub/2018/building-blocks.

[42] Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas, and Rory Sayres. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). *arXiv preprint arXiv:1711.11279*, 2017.

[43] Dumitru Erhan, Yoshua Bengio, Aaron Courville, and Pascal Vincent. Visualizing higher-layer features of a deep network. *University of Montreal*, 1341(3):1, 2009.

[44] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

[45] Aravindh Mahendran and Andrea Vedaldi. Understanding deep image representations by inverting them. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5188–5196, 2015.

[46] Alexey Dosovitskiy and Thomas Brox. Inverting visual representations with convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4829–4837, 2016.

[47] Alexander Mordvintsev, Christopher Olah, and Mike Tyka. Inceptionism: Going deeper into neural networks, 2015.

[48] David McNeely-White, J. Beveridge, and Bruce Draper. Inception and resnet features are (almost) equivalent. *Cognitive Systems Research*, 59, 10 2019.

[49] Karel Lenc and Andrea Vedaldi. Understanding image representations by measuring their equivariance and equivalence. *International Journal of Computer Vision*, 127(5):456–476, May 2019.

[50] Qiulei Dong, Jiayin Sun, and Zhanyi Hu. Face representation by deep learning: a linear encoding in a parameter space? *arXiv preprint arXiv:1910.09768*, 2019.

[51] Maxime Oquab, Leon Bottou, Ivan Laptev, and Josef Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1717–1724, 2014.

[52] Sixue Gong, Vishnu Naresh Boddeti, and Anil K Jain. On the intrinsic dimensionality of image representations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3987–3996, 2019.