DISSERTATION

TRANSFORMER, DIFFUSION, AND GAN-BASED AUGMENTATIONS FOR CONTRASTIVE LEARNING OF VISUAL REPRESENTATIONS

Submitted by

Samuel Armstrong

Department of Computer Science

In partial fulfillment of the requirements

For the Degree of Doctor of Philosophy

Colorado State University

Fort Collins, Colorado

Spring 2024

Doctoral Committee:

Advisor: Sangmi Pallickara
Co-Advisor: Shrideep Pallickara

Sudipto Ghosh
F. Jay Breidt

ABSTRACT

TRANSFORMER, DIFFUSION, AND GAN-BASED AUGMENTATIONS FOR CONTRASTIVE LEARNING OF VISUAL REPRESENTATIONS

Generative modeling and self-supervised learning have emerged as two of the most prominent fields of study in machine learning in recent years. Generative models are able to learn detailed visual representations that can then be used to generate synthetic data. Modern self-supervised learning methods are able to extract high-level visual information from images in an unsupervised manner and then apply this information to downstream tasks such as object detection and segmentation. As generative models become more and more advanced, we want to be able to extract their learned knowledge and then apply it to downstream tasks. In this work, we develop Generative Contrastive Learning (GCL), a methodology that uses contrastive learning to extract information from modern generative models. We define GCL's high-level components: an encoder, feature map augmenter, decoder, handcrafted augmenter, and contrastive learning model and demonstrate how to apply GCL to the three major types of large generative models: GANs, Diffusion Models, and Image Transformers. Due to the complex nature of generative models and the near-infinite number of unique images they can produce, we have developed several methodologies to synthesize images in a manner that compliments the augmentation-based learning that is used in contrastive learning frameworks. Our work shows that applying these large generative models to self-supervised learning can be done in a computationally viable manner without the use of large clusters of high-performance GPUs. Finally, we show the clear benefit of leveraging generative models in a contrastive learning setting using standard self-supervised learning benchmarks.

ACKNOWLEDGEMENTS

This research was supported by the National Science Foundation (OAC-1931363, CNS-2312319), and an NSF/NIFA Artificial Intelligence (AI) Institutes AI-CLIMATE Award [2023-03616].

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF EQUATIONS

1. Introduction

As large generative machine learning models become more and more sophisticated, we are starting to see the emergence of Artificial General Intelligence (AGI), i.e., a ML model that can be applied across a large array of tasks and data. In particular, Large Language Models (LLMs) such as the GPTs, LLaMA, and BERT are able to verbally communicate in a manner similar to humans and can solve a large range of language problems across multiple domains [1, 64, 65, 80]. However, the development of an AGI for computer vision models lags behind current language models. This is due to the high complexity of spatial relationships in computer vision data such as images, videos, and 3D representations compared to natural language. Due to this complexity, a self-supervised learning encoder is needed to create representations (a.k.a. embeddings) from computer vision data, thereby reducing this computational complexity. These representations are 1D arrays that can represent vision data in a compressed form. When representations are effectively extracted, they are easier for ML models to learn from compared to learning directly from the raw data itself. These representations can be used for a large range of downstream tasks, such as classification, object detection, and object segmentation, across many domains, such as photographs, medical imaging, and satellite imagery. How to extract these representations efficiently and in a self-supervised manner is a prominent and ongoing area of ML research.

Recently, contrastive learning models have had much success in encoding vision data in a completely unsupervised manner. Contrastive learning is able to extract information from data using only the data and without the use of labels or any other supervision. Contrastive learning takes inspiration from human vision and its ability to visually compare two objects to learn abstract representations of complex objects and scenes.

Generative modeling is very useful in the context of contrastive learning in that it is able to generate positive pairs, which are groups of images that contain similar visual features, such as similar objects or patterns. Typically, in contrastive learning, a positive pairing is done by randomly applying handcrafted

(a.k.a. traditional) computer vision augmentations, such as flipping, rotating, cropping, color jitter, Gaussian noise, etc., to an image.

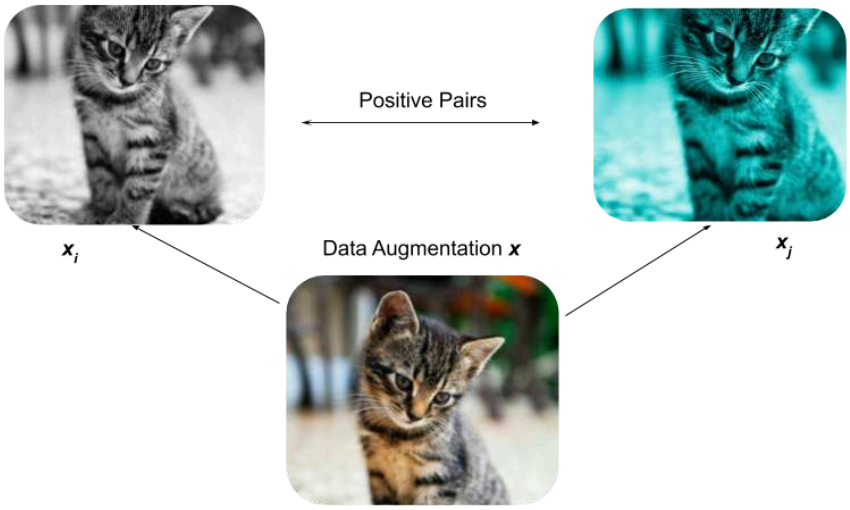

Figure 1: The image x is randomly augmented twice, creating a positive pair (x_i, x_j).

These augmentations are randomly applied to a single image twice, creating a positive pairing between the two augmented images *(***Figure 1***)*. Ideally, the positive pairing of images is as different from each other as possible while still preserving important visual features. Although difficult to define, the extraction of high-level features can be measured by the representations' accuracy on downstream tasks. In contrastive learning, typically, the more different the positively paired images are, the better performance of the representations on downstream tasks. However, if the positively paired images are too different, the contrastive learning encoder will be unable to identify important visual features in both images, and the representations will perform poorly on downstream tasks. Finding the right balance of strong augmentations that also preserve important visual features is key to a well-performing contrastive learning model, and we posit generative machine learning models are well suited to this task.

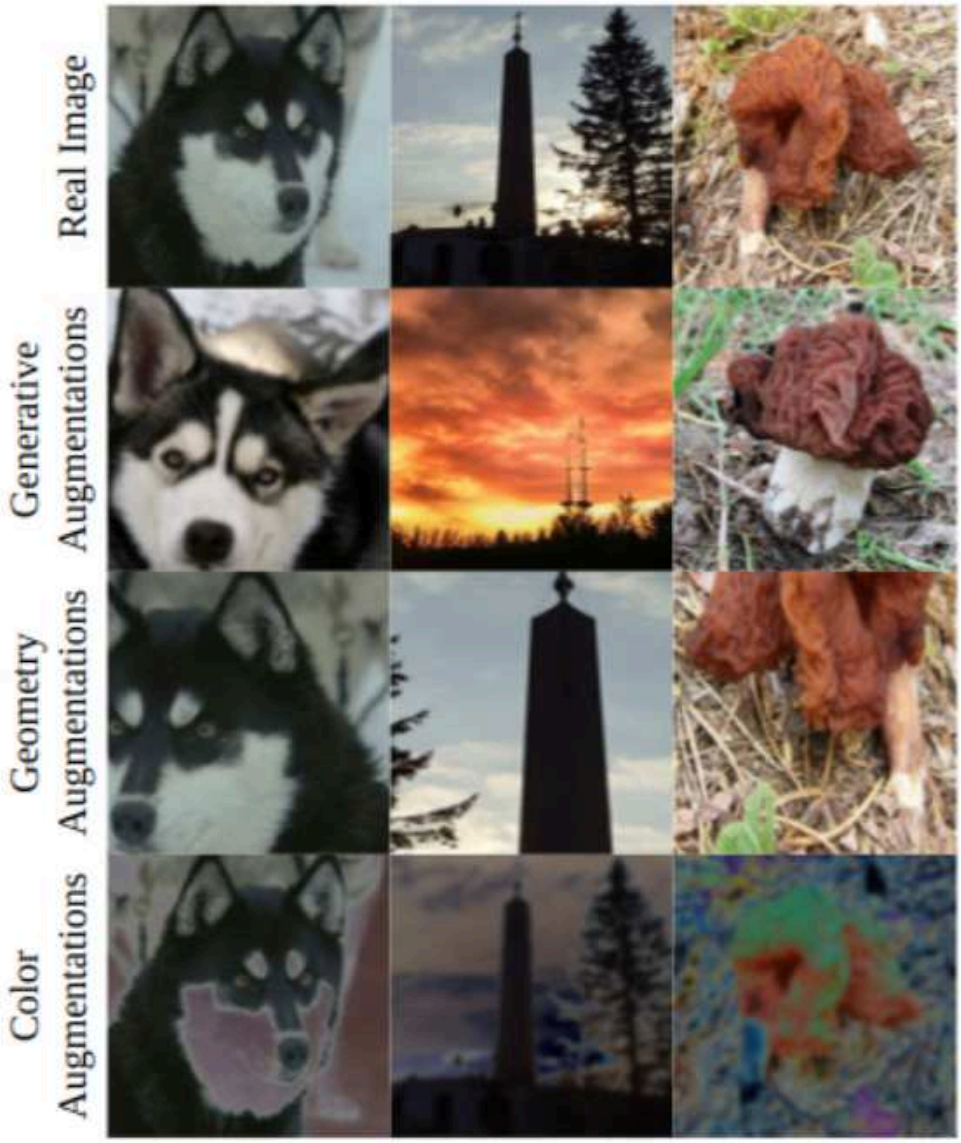

Figure 2: Examples of generative augmentations and handcrafted (geometry and color) augmentations. The first row is the original, non-augmented image. Column two contains examples of augmentations using our noise-based generative augmentation strategy. Columns three and four are examples of the handcrafted augmentations cropping and color jitter, respectively.

Generative ML models are able to augment visual features in a diverse assortment of sophisticated ways compared to handcrafted augmentations. These generative model-based augmentations greatly increase the amount of change we can apply when forming positive pairs, resulting in improved downstream accuracy when applied to contrastive learning. Unfortunately, the current literature on how

best to apply generative modes to contrastive learning is lacking. In our research, we develop a GCL framework that allows us to leverage the three major types of generative models to directly improve contrastive learning.

1.1 Problem Formulation

Our GCL framework leverages generative models in a way that can then be used to extract more meaningful contrastive learning representations in an efficient and effective manner. Our GCL framework capabilities should include the ability to:

PF1) Map Real Images to a Generative Model's Feature Map

PF2) Augment the Feature Map

PF3) Synthesize Realistic Generetively Augmented Images

PF4) Leverage Generated Images in a Contrastive Learning Setting

PF5) Improve Downstream Accuracies

1.1.1 Encoding Images (PF1)

Understanding images is a difficult task for ML models, and how human vision learns to understand images is not fully understood. The oversimplified goal of computer vision can be stated as the ability to map any image to an efficient and effective 1D representation that can then easily be applied to any task, downstream, generative, or otherwise. This general ability to apply dimension reduction to any given image is clearly quite difficult. There are two methods for encoding images in an unsupervised manner: generative modeling and self-supervised learning. Although generative models encode visual data effectively, these representations often contain too much information about low-level visual features, which are not useful for high-level downstream tasks like classification, object detection, and object

segmentation [50]. Finding ML methods that extract high-level visual information from generative models while filtering out low-level information would allow us to apply the knowledge learned by generative models to downstream tasks. To do this, we use GCL, which minimizes the difference between similar high-level visual features in the generative model's feature map space.

1.1.2 Feature Map (PF2)

How generative models map and organize image representations they have learned is often referred to as the generative model's feature map, a.k.a. latent space. This feature map is typically represented as a single 1D vector of values that follow a probability distribution. A feature map is different from a self-supervised representation in that it represents all visual features for generative purposes, while a self-supervised representation represents solely high-level information for downstream tasks. Images with similar visual features are mapped to similar feature maps, while dissimilar images are mapped to dissimilar feature maps. Measuring the visual similarities between images is a non-trivial task and is difficult to define. However, measuring the difference between feature maps is typically straightforward as they are represented by numerical representations. These feature maps either explicitly or implicitly follow probability distributions when the generative model is trained. Because feature maps follow a probability distribution, we can now sample from that distribution to generate a feature map. This feature map can then be inputted into a generative model to generate a synthetic image. Generative models can have more than one feature map representing an image. When there are multiple feature maps, they either represent visual features in an image in a hierarchical order (typically the size of the visual feature) or their spatial placement. These multiple feature maps are highly interconnected to one another. This can cause randomly sampling the feature maps' probability distribution to produce static or low-quality images. Because of the complex relationship between these interconnected feature maps, a second generative model is needed to sample the feature maps. Feature maps can also be conditioned on other data, such as text or labels. When using class labels for generative models, each class has its own separate feature map probability distribution. This greatly reduces the complexity of mapping images to the feature

map space and allows generative models to scale to larger, more complex datasets. For our GCL framework, ideally, we want a generative model that uses a single feature map with as few conditions on labels as possible. This will make sampling realistic, non-static images easy and straightforward while also allowing our generative model to scale to larger datasets like ImageNet. Although GCL essentially minimizes the difference between the feature map of images with similar visual features, our final contrastive learning model must also be able to encode images. For this reason, we map these feature maps back to images before training the encoder.

1.1.3 Synthesizing Images (PF3)

Similar to humans, ML models should be able to synthesize completely unique data that resembles previously seen data. This can be done by both interpolating and extrapolating a generative model's feature maps. Generative ML models can be grouped into five categories: Variational AutoEncoders (VAEs), Normalizing Flows (NF), AutoRegressive Models (ARMs), Generative Adversarial Networks (GANs), and Diffusion Models (DMs). Each of these model architectures has its own advantages and disadvantages when used to augment images for positive pairings. These generated images must be realistic images that represent both important high-level features and negligible low-level features in regard to downstream tasks. Mapping feature maps to images is a hard problem as it is learning the inverse function of the encoder. Training models to decode data in generative modeling requires more computation than encoding as it struggles more with scaling to large datasets and complex data. After synthesizing an image, we can then use it in the context of contrastive learning.

1.1.4 Augmentation Strategies (PF4)

Modern contrastive learning frameworks leverage handcrafted image augmentations such as cropping, flipping, Gaussian noise, etc. These handcrafted augmentations are quick and easy to apply but lack the diversity of changes that generative models can apply. Modern generative computer vision

models are able to augment images using global visual features learned from other images in the data, while handcrafted augmentations are bound to local visual features. Because generative models map images to multi-dimensional distributions, the number of ways you can synthesize unique images is many times the size of the training data and virtually unlimited. Our work requires us to traverse this large feature map space to find feature maps that augment images in a way that increases downstream accuracy.

Using generative models in a contrastive learning setting is difficult because the only true way to determine if a generative augmentation is effective is to then train a contrastive learning and benchmark its downstream accuracy. Like generative models, contrastive learning is computationally expensive. This, compounded with the sheer size of the feature map space, makes developing GCL computationally expensive. Our work relies heavily on previous contrastive learning research, specifically handcrafted augmentations and why they are effective, as well as common self-supervised learning benchmarks.

1.1.5 Downstream Accuracy (PF5)

The goal of self-supervised learning in computer vision is to extract representations from images that can then be used to train ML models for various downstream tasks. These representations should be easy for ML models to understand, meaning a simple shallow neural network model should be able to use them for downstream tasks with minimal training. In addition, the number of samples needed to train a ML model on these representations should be significantly decreased compared to the number of real images required for training a fully supervised ML model. These downstream tasks include applications that extract high-level information from the images, such as classification, object detection, and image segmentation. Measuring the effectiveness of our GCL implementation is mainly based on the accuracy of its representations on downstream tasks. For evaluating self-supervised research, a protocol has been established that we follow throughout our research.

1.2 Scientific Challenges

Reproducing human vision's ability to extract abstract representations of complex visual scenes in ML models has proven difficult as both mapping images to a feature map space and mapping the feature map space to images are non-trivial problems. Many of the problems of developing a cohesive architecture between generative models and contrastive learning stem from the use of generative models. Generative models are often slow in both training and inference speed and require the use of large clusters of powerful GPUs. Contrastive learning models also suffer from long training times but to a much lesser extent. The scalability of generative models has likely been the most prominent issue in their development since their inception. Only in the past four years have we seen generative models that are able to scale to large vision datasets of one million or more images [2]. Comparatively, contrastive learning is easy to scale to large, complex datasets. The use of labels and natural language has been used heavily to scale these large generative vision models; however, contrastive learning functions in an unsupervised environment. Finally, image quality can be measured by the amount of information a generated image contains compared to a real image and is the defining metric for generative models. GCL is heavily dependent on the image quality of generative models as this translates to the information that is then learned by our contrastive learning model.

SC1) Speed

SC2) Scalability

SC3) Unsupervised Learning

SC4) Image Quality

1.2.1 Training and Inference Speed (SC1)

SOTA generative image models such as GANs, image transformers, and DMs have successfully been scaled to large real-world datasets [3, 6]. These models provide the highest quality of synthetic images, to the point where it is difficult to discern real images from generated ones. However, these models often struggle with image synthesis and/or image editing at scale due to iterative inference times. An iterative inference time means the model must be called multiple times sequentially to generate a single image. These iterative inference generative models either generate parts of the image or refine the quality of the image in each iteration. Iterative inference times often make generating large datasets of millions of images infeasible as the amount of time to generate all the images can take weeks if not months. An inference time of a second would require 165 days to synthesize the 14,197,122 images found in the largest version of the ImageNet dataset, ImageNet-21K [4], using a batch size of one. Developing ML methods that speed up the inference times of generative models is an ongoing and active area of ML research. A generative model's slow inference speed affects GCL as we need to first sample the generative model before training our contrastive learning model.

Both generative models and contrastive learning models suffer from long training times. This issue is compounded by the fact that our training environment is limited to 4 3090 24GB NVIDIA GPUs. In contrastive learning research, an environment of 16-32 A100 80GB NVIDIA GPUS is the standard for benchmarking [41, 42, 58]. When factoring in the model weights and how they must be replicated on each GPU, the training time for a contrastive learning model for 1,000 epochs can be done in ~12 hours on 32 A100s, while in our environment, it takes ~16 days. Generative models are even more computationally hungry than contrastive learning models and would require months to train on large datasets like ImageNet in our environment. We rely heavily on using pretrained generative models as well as reducing epochs and using subsets of datasets throughout our work.

1.2.2 Scalability (SC2)

Scalability is another key issue with generative models. Often, generative ML models struggle with high-resolution images, large datasets, and images with complex scenes. Typically, a high resolution for generative models is at least 256x256 pixels. When generative ML models were first introduced, they could only produce images with at most 28x28 pixels [5]. Large ML datasets are usually in the magnitude of hundreds of thousands, if not millions or even billions, of images, while smaller datasets are typically in the tens of thousands. Computer vision models, in general, struggle with complex visual scenes. A common practice when creating computer vision datasets is cropping the image around the point of interest. When the positioning of visual features in images is consistent across the entire dataset, it makes it much easier for a ML model to learn. In contrast, generative ML models struggle to learn from in-the-wild images where there is no cropping and there are complex unstructured scenes.

Many ML methods have been developed and successfully applied to address these scalability issues. It is common to see a multitude of different scaling methods being applied to a single model, especially in SOTA generative models such as Diffusion Transformers (DiTs) and Masked Diffusion Transformers (MDTs) [3, 61, 67]. There are several types of generative models that struggle to scale to large datasets, like ImageNet, and therefore cannot be used directly in GCL. Many scaling techniques use a divide-and-conquer methodology, dividing the task of image generation into many steps, which slows the inference speed down, again limiting the pool of usable generative models in GCL. Contrastive learning methods scale easily, so finding a generative model that can match this scalability can be a challenge. Another issue with many of these scaling methods is that they often rely on some form of labeling or class conditioning.

1.2.3 Unsupervised Training (SC3)

SOTA big data generative models use labeled images in order to scale to large datasets [67, 78, 3]. As self-supervised learning applications are often used in a semi-supervised setting, our generative model

should preferably be trained in an unsupervised manner. There are two approaches to scaling generative models in an unsupervised manner. The first is the use of an encoder. This encoder compresses an image into the generative model's feature map and then inputs it into the generator, which recreates the image. Adding an encoder network can be computationally expensive as we are adding another network to our generative model. Using an encoder can also destabilize training as it requires an additional loss function when training the model. However, in GCL, we require the use of an encoder model to map a real image to the feature map of our generative model. The second, less common approach to scaling without labels is clustering. A clustering model can group images into subsets that have similar visual features. These clusters can then be used as pseudo labels. This helps the generative model scale to large sets of images in a divide-and-conquer method. However, training generative models solely with an encoder or clustering results in worse image quality than training with true labels [7]. This is why using labels for image generation models has become the norm, as it directly affects the image quality metric that all generative models are evaluated on. Finding modern pretrained generative models that have been trained in an unsupervised or at least semi-supervised manner is difficult, and the models that are available are often outdated and provide poor image quality.

1.2.4 Image Quality (SC4)

The success of a generative image model is based on the quality of its images. There are a few common measures for the quality of synthetic images. The current standard is the Frechet Inception Distance (FID) [82]. FID uses a pretrained Inception V3 model to judge the accuracy of a group of generated images to a group of real images. Each image in both groups is inputted into the Inception model, but instead of taking the outputted class probabilities, we use the outputted representation from the second to last layer of the neural network. The means m and covariance matrices C of this representation are taken for both the generated images G and real images R and are then plugged into the FID formula, see **Formula 1**.

$$FID((m_R, C_R), (m_G, C_G)) = ||m_R - m_G||_2^2 + Tr(C_R + C_G - 2(C_R C_G)^{1/2})$$

Formula 1: Frechet Inception Distance Formula

In the FID formula Tr is the trace operator, i.e., the sum of the elements along the main diagonal of the matrix. The lower the FID, the higher the image quality of the generated images. An FID of zero would mean that a generated image has the same quality as a real image. The current SOTA generative model on the ImageNet data is able to achieve an FID of around 1.58 with the Masked Diffusion Transformers V2 (MDTv2) [67]. In our work, we find that the image quality of generative models affects downstream accuracy, so using SOTA generative models in our GCL framework translates to higher downstream accuracy.

Another method for comparing two images is the Structural Similarity Index (SSIM). SSIM is used less as an image quality metric and more to determine the similarity between the structure of two images [81]. SSIM uses a sliding window between two images (x and y) measuring the mean (μ) and variance (σ) of pixels in the window as well as the covariance between pixels between the two windows (σ_{xy}). Constants C are also used to prevent division by zero. SSIM provides a better metric for comparing the structure rather than the image quality itself of two images compared to FID. SSIM is also computationally lighter to compute than FID as it does not rely on any ML models. Training a contrastive learning model with a positive pair where the images are too different can make learning difficult for the encoder model and negatively impact its performance. The SSIM metric can be used to assess how different the images in a positive pair are, giving a good indication if the augmentations are too strong.

$$SSIM(x, y) = ((2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2))/((\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2))$$

Formula 2: Structural Similarity Index Formula

1.3 Research Questions

The end goal of our GCL framework is to train an encoder that can encode images into representations that achieve higher downstream accuracy compared to contrastive learning methods that do not utilize generative models. In order to develop an efficient and effective GCL framework, we address the following research questions:

RQ1) How can generative models be leveraged to augment images in a manner that maintains important visual information?

RQ2) How do we efficiently incorporate generative models into a contrastive learning architecture without increasing computation significantly?

RQ3) How can we construct a generative augmentation architecture that boosts the downstream accuracy of learning representations in a self-supervised manner?

1.3.1 Augmentation-Based Learning (RQ1)

Generative models learn image representations that perform poorly on high-level downstream tasks because their learned representations contain too much detail of irrelevant low-level visual information. Finding methods to distill this knowledge so that it can then be applied to downstream tasks will allow us to apply transfer learning between generative models and task-oriented ML models. As current large generative models have no native method for disassociating low-level and high-level visual features, we must implement our own outside methodology for doing so. Contrastive learning, the most common and effective form of self-supervised learning, learns representations of high-level visual features while simultaneously filtering out low-level ones. Contrastive learning uses augmentation-based learning, where augmentations are data transformations that imprecisely alter the low-level features in data. Not only are current data transformations, i.e., handcrafted augmentations, imprecise, but they are also severely limited

in the amount of low-level visual features that can be introduced while training a contrastive learning model. As generative models learn global low-level features and are able to add them to real images, we posit that these augmentations will improve augmentation-based learning methods like contrastive learning. However, the problem still remains: How can generative models change low-level visual features while preserving high-level visual features when they are unable to differentiate the two?

1.3.2 Efficient Generative Augmentations (RQ2)

In any ML application, there is always a trade-off between computation and accuracy. Due to the large size and complexity of modern generative models and the nearly unlimited number of unique data points they can create, generated models are rarely used in tandem with other ML methods. With the development of simple data generation frameworks, like GCL, the ability to apply generative models to other domains of ML becomes much easier and more straightforward. Finding efficient generative models for generating data is key to this framework. This means a model that can scale to large datasets and generate data quickly and efficiently. Finding generative models well-suited to this is non-trivial as most modern research of generative models focuses on improving image quality, leaving developing computational efficiencies as more of an afterthought. Additionally, the work surrounding generative models is extensive, and the speed at which new, better-performing generative models are being produced has increased steadily over time, making future-proofing our GCL framework difficult. This requires not just knowledge of current SOTA generative model implementations but also an understanding of the future direction of generative models themselves. Contrastive learning is also one of the more computationally intense ML methodologies. Contrastive learning models require long training times and benefit greatly from the use of larger neural networks. Like generative models, understanding current contrastive learning research and its direction will help us combine the two in a computationally efficient manner. It is difficult to judge what an appropriate trade-off between computation and accuracy is in regard to GCL. Throughout our work, we develop computational efficiencies and apply cost-saving methods to maintain the usability of our GCL architecture.

1.3.3 Self-Supervised Accuracy (RQ3)

Contrastive learning research has provided the best downstream accuracy in the context of self-supervised learning research [41, 42, 46]. Using self-supervised learning provides several advantages over supervised learning. Supervised learning uses labeled data, which requires people to hand-label each data point. Self-supervised learning does not require labels when training the encoder and, therefore, requires a smaller amount of labeled data when applied to downstream tasks. Because self-supervised learning does not rely on labeled data, it is not as susceptible to overfitting as supervised learning, meaning larger models can be used to greater effect. The representations learned by self-supervised learning can be reused for various downstream tasks, while supervised learning methods require separate models to be trained for each task. The only place where self-supervised learning methods lag behind supervised methods is downstream accuracy. Although the difference between the downstream accuracy of self-supervised methods and supervised methods is minute, only recently have a handful of self-supervised learning methods been able to produce accuracies that are comparable to or better than supervised methods [39, 40, 59]. These breakthroughs in self-supervised learning accuracy would position it as a superior alternative to supervised learning in future ML research. In our work, we present a framework that can be used in tandem with modern contrastive learning methods to help boost accuracy on downstream tasks.

2. Background

Our proposed GCL framework relies heavily on previous work done on generative models and contrastive learning. The literature on generative modeling, in particular, is extensive. As our GCL model's performance relies heavily on the performance of the generative model, we provide a taxonomy of current generative models. We also analyze the advantages and disadvantages of different SOTA generative model implementations. For contrastive learning, we give a taxonomy of commonly used augmentations. We also give a detailed explanation of contrastive learning and discuss several prominent contrastive learning frameworks. Finally, we briefly discuss two previous implementations of GCL.

2.1 Generative Modeling

Over the last decade, the number of generative models in computer vision has increased exponentially [9]. The SOTA of these models is able to produce images so realistic that it becomes difficult for a person to tell the difference between real images and synthesized ones. Typically, the realism of these generated images is measured using the FID. The current standard dataset for measuring generative computer vision models is the ImageNet dataset, a dataset that consists of a thousand classes and 1,431,167 images [4]. The images found in the ImageNet dataset have not had any image-cleaning techniques applied to them. While the ImageNet dataset is the most common and widely used generative computer vision dataset, other common image datasets are MNIST, CIFAR-10, CelebA, LSUN, FFHQ, and AFHQ. These other datasets have been cleaned, specifically cropped around the focus of the image, making them generally easier for generative models to learn. In regards to GCL, we are looking for a generative model with a fast inference time, good scalability, and high image quality.

2.1.1 Generative Model Taxonomy

VAEs are a fast, lightweight generative model introduced in 2013 by Kingma et al. [5]. The VAE is composed of two neural networks: an encoder and a decoder network. The encoder compresses the data down into a feature map, and then the decoder reconstructs the inputted data from the feature map (**Figure 3**). The loss for VAEs typically consists of two loss functions: a reconstruction loss and Evidence Lower Bound loss (ELBO) with Mean Squared Error (MSE) and Kullback–Leibler Divergence (KLD) being the most common loss functions used, respectively. The reconstruction loss ensures the VAE creates images that are similar to the training data, while the ELBO loss implicitly maps the feature map to a probability distribution. The ELBO loss allows us to sample and perturb the feature map space of the training data. The decoder can then input samples from this probability distribution to produce synthetic images.

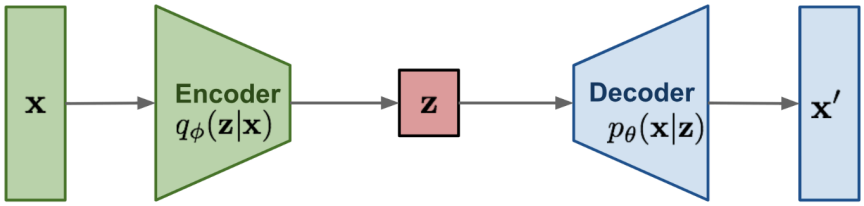

Figure 3: VAE architecture, where x is the input image, z is the feature map, and x' is the reconstructed image.

The VAE is computationally the least expensive of all the generative models in computer vision. This is due to the fact that the encoder and decoder networks are simple feedforward convolutional neural networks that share the same loss and backpropagation process. VAEs also have the fastest inference time of all the computer vision generative models because the encoder is relatively lightweight and generates images in a non-iterative manner. This makes VAEs well-suited to producing large sets of images quickly and efficiently. Unfortunately, VAEs produce blurry, low-quality images. This issue is especially noticeable in higher-resolution images where the blurriness and poor image quality become more

pronounced as the size of the image increases. Improving the quality of the images produced by VAEs is an ongoing area of research that has produced several VAE variations that can reduce the blurriness and increase the quality of its images [10, 11, 12].

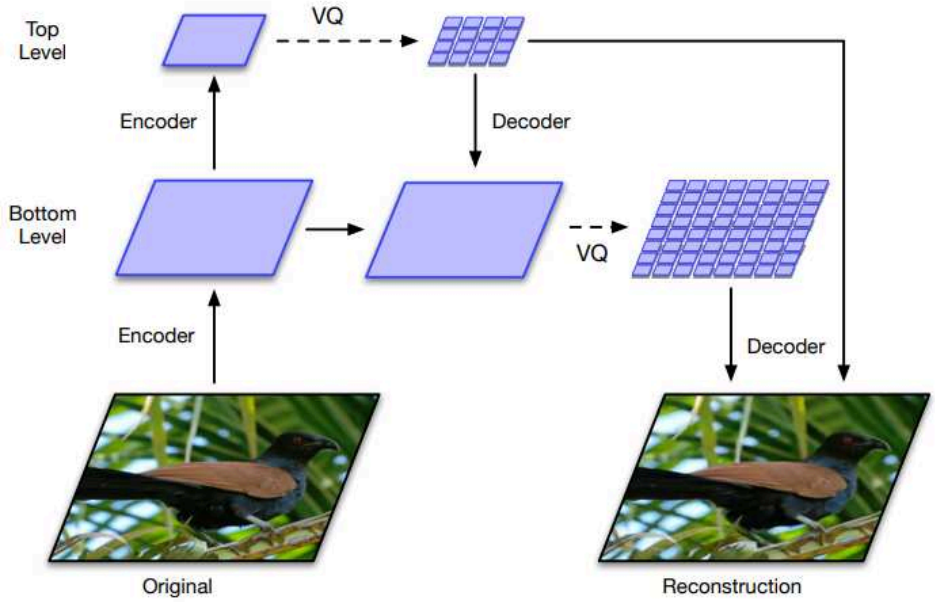

Figure 4: VQVAE2's hierarchical feature map architecture.

The current SOTA VAE on the ImageNet dataset is the Vector Quantized Variational AutoEncoder (VQVAE2) [13]. The VQVAE2 differs from the original VAE architecture in two ways. Firstly, the VQVAE2 uses a hierarchical architecture for its encoding. This means that the encoder in this architecture encodes the image and then passes the encoding to both a decoder and another encoder. Multiple encoders and decoders are then stacked on one another in an architecture similar to a U-Net (**Figure 4**). Using hierarchical encodings helps increase the quality of images produced in VAEs, and it allows it to scale to higher-resolution images. The second difference between the VQVAE2 architecture and the original VAE is that it encodes its feature map into a 2D discrete probability distribution instead of a 1D continuous distribution (**Figure 5**).

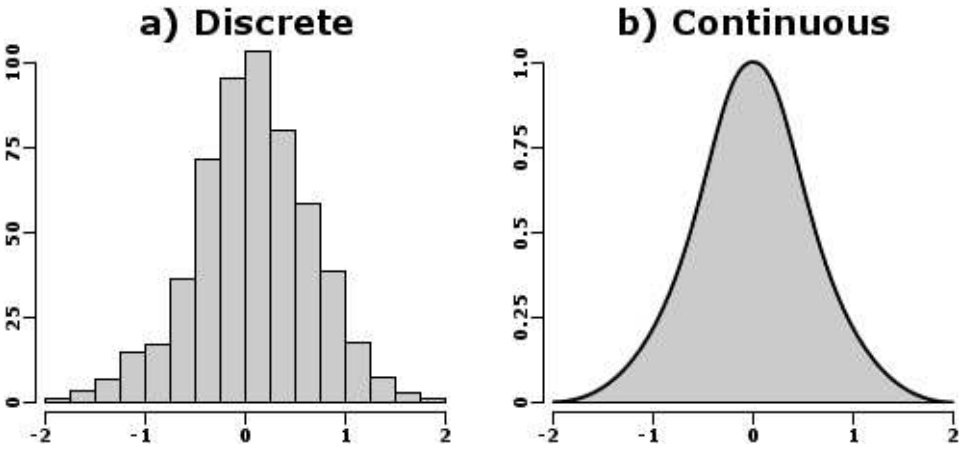

Figure 5: Example of discrete and continuous distributions.

This process of mapping an image to a discrete feature map in ML is known as vector quantization. In VAEs and other generative models, discrete feature maps have been shown to produce higher-quality images in smaller, more efficient feature maps compared to feature maps that follow continuous distributions [13]. The downside to using both hierarchical and discrete feature maps is that it makes sampling and perturbing the feature map space difficult. This is due to the feature maps in a hierarchical VAE being highly interconnected to one another. This means that randomly sampling these feature maps fails to reproduce these connections and ends up producing static or low-quality images. The same is true for discrete feature maps as the discrete variables also have complex interconnectivity, which randomly sampling the discrete values fails to produce, again resulting in static or low-quality images. In order to sample from the hierarchical discrete feature maps used by the VQVAE2, we need to train another generative model on its feature map space. Training generative models on these highly-dimension discrete feature maps is difficult and requires expensive generative models such as ARMs and DMs that are well suited to modeling highly-dimensional data [14, 15]. Training these generative models on the feature maps requires more computation than training the VQVAE2 model itself. Also, the fast non-iterative inference we normally have from VAEs is now compromised due to the slow, iterative inference speed common to ARMs and DMs. This makes synthesizing large datasets with generative models that use

hierarchical and/or discrete feature maps temporally and computationally expensive. Also, despite the use of hierarchical and discrete feature maps, VQVAE2's FID of ~31 on the ImageNet dataset is still significantly lower compared to other SOTA generative models, which can FID as low as 1.58 [67]. This has largely resulted in other generative models like GANs, DMs, and ARMs being used instead of VAEs in generative applications. One benefit to using VAEs is they have an encoder, which can help them scale to large datasets without the use of supervised learning. Overall, SOTA VAEs are scalable and can be trained in an unsupervised manner but lack fast inference speeds and high-quality images. Other than VAEs, the only other generative model type that has both an encoder and a non-iterative inference speed is NFs.

The first NF image model was the Deep Latent Gaussian Model (DPGM), which was introduced in 2015 by Rezende et al. [17]. NF models consist of a single multi-layered network that explicitly maps feature maps to a probability distribution. These NF models use invertible neural network layers that bijectively map data to a feature map (**Figure 6**). These invertible layers are computationally expensive as they require the use of a Jacobian Matrix and require very large feature maps. Because NF models explicitly map feature maps to a probability distribution, they can use Negative Log-Likelihood (NLL) as their loss function when training. In comparison, VAEs implicitly map feature maps to a probability distribution using an ELBO loss. NFs are generally easier and more stable to train than other generative models due to the use of the NLL loss. Learning the exact distribution also means that these feature maps are highly representative of the training data relative to other generative models, creating a large diversity of images. In comparison, other generative models, like GANs, do not model the feature map space as well, causing its generated image to have less diversity in visual features. In order to compress an image down into a smaller feature map, NF layers lose small details of an image, which degrades the image quality. Unfortunately, NF models do not scale well and struggle with large, high-resolution datasets like ImageNet. NF models like GLOW, Flow++, and DenseNet work well on low-resolution versions of the ImageNet dataset (64x64 pixels) and on domain-specific datasets like CelebA and CIFAR-10 [18, 19, 20].

NF models are the least popular of the generative models due to their inability to scale well and low image quality. Current NF research focuses on improving both scalability and image quality.

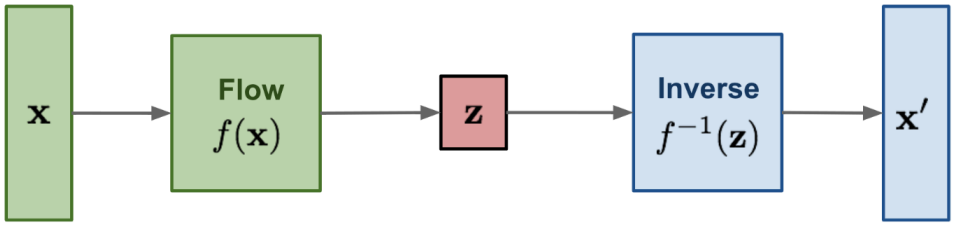

Figure 6: NF architecture where x is the input image, z is the feature map, and x' is the reconstructed image.

The current SOTA NF model on a low-resolution version of the ImageNet dataset is DenseFlow [21]. DenseFlow uses the same GLOW modules introduced in [18], which use activation normalization layers, 1x1 convolutional layers, and affine coupling layers to create an efficient, invertible neural network layer called a GLOW module. DenseFlow improves upon the GLOW module by adding densely connected blocks, which have skip connections from one neural network layer to every preceding neural network layer. DenseFlow also uses fast self-attention layers inside the GLOW modules (**Figure 7**). Adding both the densely connected blocks and self-attention layers to the GLOW module results in higher-quality images, faster convergence, and a small computational footprint compared to previous NF models [21]. Although easier to train, explicitly mapping feature maps directly into a probability distribution is often not computationally efficient due to large image datasets that have too much variability. Other generative models often use a divide-and-conquer method to break the image up into smaller feature maps, resulting in better scalability but also a slower iterative inference time.

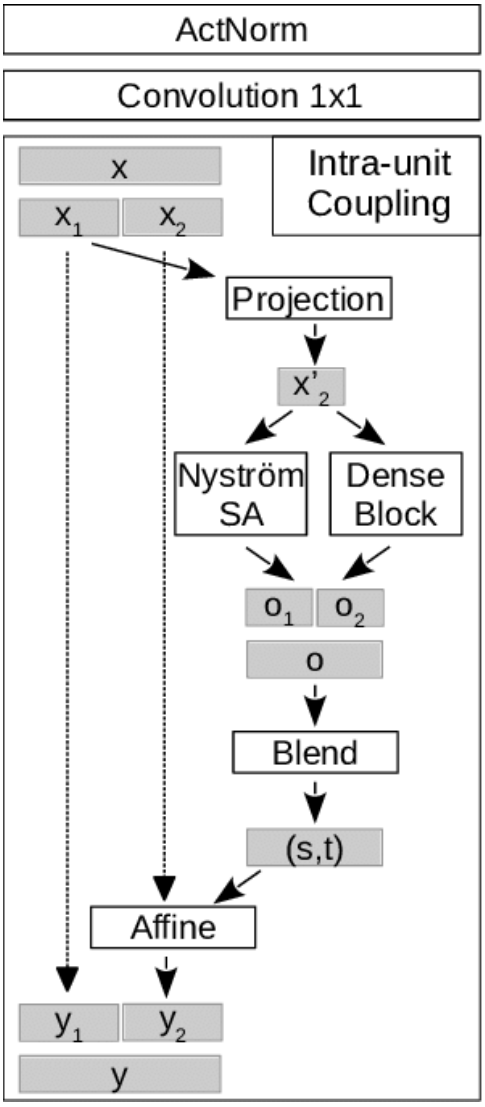

Figure 7: DenseFlow Block Architecture

In 2016, the Pixel Recurrent Neural Network (PixelRNN) was introduced as one of the first ARMs for image generation [22]. The PixelRNN and other ARMs, like the Pixel Convolutional Neural Network (PixelCNN), Pixel Simple Neural AttentIve Learner (PixelSNAIL), and Generative Image Transformers (GIT), are able to synthesize high-quality images on large, high-resolution datasets like ImageNet [14, 23 24]. Unlike other generative models in computer vision, ARMs either generate a single pixel or small groups of pixels in a sequential order. This significantly slows down the inference time of generating entire images, as the ARM must be called multiple times. This results in slow and highly iterative

inference speeds. Current ML regarding ARMs focuses on speeding up this slow inference time and increasing image quality.

The introduction of the transformer model has fundamentally changed ML research and is the new standard architecture for ARMs. Previously, recurrent neural network (RNN) models like long short-term memories (LSTMs) and gated recurrent units (GRUs) were the standard architecture for image-based ARMs [25, 26]. Transformers differ from LSTMs and GRUs in the way they map relationships between sequences. For transformer language models, tokens are created for each word and position in a text. For image transformers, pixel values (0-255) or small groups of pixels replace the word tokens. Transformers use a very efficient form of attention for mapping the relationship between each token. This attention mechanism uses three matrices to accomplish this: a query, key, and value matrix. The multi-headed attention is then able to predict the relationship between any two tokens at any given position.

When generating a pixel with an ARM, previously generated pixels are inputted into an ARM, and the values for each possible pixel are outputted in the form of a probability distribution. This allows us to calculate the exact distribution for each pixel and use the pixel value with the highest likelihood as the generated pixel. Because we are explicitly mapping pixels to a probability distribution, we can use a loss function such as NLL, similar to NF models. The transformer's way of modeling attention requires the relationships between each pair of tokens to be stored in the model, making the model's size equivalent to $O(N^2)$, where N is the number of unique tokens. This differs from LSTMs and GRUs, which only store a subset of important relationships between tokens but use a constant amount of space that does not require being scaled up when the number of tokens in a sequence is increased. Because of this, transformers outperform previous ARMs, especially in regard to long-term dependencies.

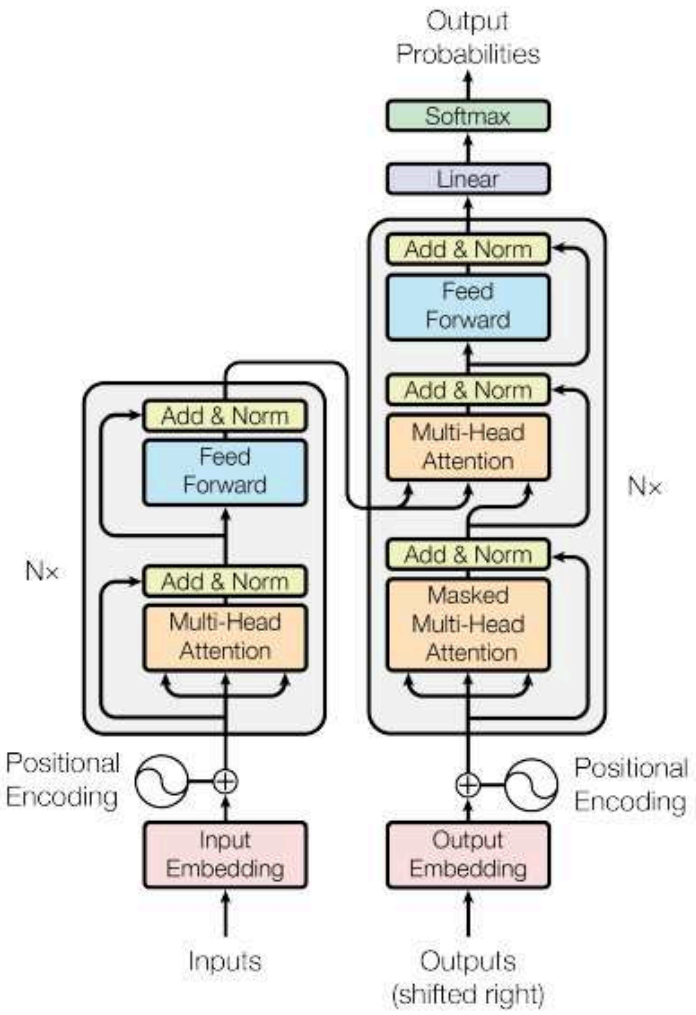

Figure 8: The transformer architecture.

ARMs are highly iterative in their inference speed, making synthesizing and augmenting large numbers of images computationally and temporally expensive. Increasing the number of pixels generated in an inference iteration can help reduce this in a process known as parallel decoding. With parallel decodings, we can increase the number of pixels generated in parallel and even generate all the pixels at once. However, the more pixels you generate at once, i.e., the fewer iterations needed to generate an entire image, the lower the quality of the image [27]. This creates a tradeoff between inference speed and image quality. Reducing the number of iterations during inference can cause the image quality to deteriorate rapidly, and generating a high-resolution image in single-digit iterations results in blurry,

poor-quality images. Overall, ARMs, specifically image transformers, have gained much popularity as generative models in recent years. In fact, the current SOTA model for image generating on the ImageNet dataset uses a hybrid of both ARMs and DMs [67].

DMs are often regarded as the newest of the generative computer vision models, but the first DM was actually created in 2015 by Sohl-Dickstein et al. [28]. DMs were largely ignored until 2020 when Ho et al. published [29] showing SOTA results on generating the CIFAR10 dataset. The diffusion process starts by taking an image and adding a small amount of Gaussian noise to it. This noise is then removed by inputting the image into a U-Net image-to-image mapper, which maps the noisy image back to the original image. Noise is added many times in the diffusion process, where each step makes the image appear more and more like a static image. The U-Net model is trained on each of these steps, each time removing small amounts of noise from the image. During training, the DM, which typically uses a U-Net architecture as its backbone but can also use an image transformer, removes noise over many steps (**Figure 9**). The DM is conditioned on each step, meaning both the noisy image and the amount of noise added are inputted into the DM. When adding Gaussian noise iteratively to an image, it will eventually become a completely static image whose pixels follow a random distribution. After training, a DM can generate an image by inputting a static image into the DM, which denoises it iteratively. With each iteration, the DM reduces the amount of noise in the image until it becomes a real, completely noiseless image. This makes the DM inference speed highly iterative, even more so than ARMs. The quality of the images created through DMs is very high, and DMs have made up a large majority of the SOTA generative models on the ImageNet dataset in recent years [6, 61, 67]. Variations of DMs focus on increasing image quality, decreasing inference time, and mapping text-to-image.

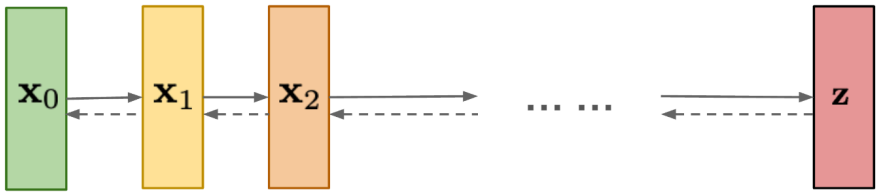

Figure 9: DM architecture, where x_0 is the imputed image, each preceding $x_1, x_2, \ldots$ represents a step where noise is added to the image, and z is a static image.

Due to the highly iterative nature of DMs, it is difficult to augment large numbers of images quickly. To generate a positive pair of images with DM, we can take an inputted image and add enough noise to it to distort some visual features while preserving others. Then, once we iteratively denoise the image, it will share some visual features with the original inputted image. Another common way of augmenting images with DMs is to condition them on labels or natural language. It has been shown that guiding DMs with class conditions and natural language prompts significantly increases their image quality [30]. We can create positive pairs by inputting the same label or text and two different static images into the DM for denoising. This will create two images that have the same visual features associated with the inputted text/prompt, but other visual features that are unassociated with the text/prompt will have some variation. DM's success in producing high-quality images is fairly recent, and before 2021, GANs largely dominated other generative models in image quality.

The first GAN was introduced in 2014 by Goodfellow et al. [2]. GANs consist of two networks: the generator and the discriminator. Instead of using an encoder, the generator randomly samples from a probability distribution and then outputs an image using a decoder architecture. The discriminator is inputted with both the generated and real images. The discriminator loss function is based on how accurately it classifies real images as real and generated images as generated. The generator's loss function is based on whether its generated images are classified as real by the discriminator (**Figure 10**). These networks' loss functions are directly opposed to one another in a zero-sum competition, i.e., adversarial learning. GANs are able to produce high-quality images with fast, non-iterative inference.

However, due to the complex nature of adversarial learning, GANs require highly computational training, and the training itself is unstable, often failing with training issues such as vanishing gradient, mode collapse, or failure to converge. There are many variations of GANs, as GANs have been the standard for generative models for many years prior to DMs, with the majority of these variations focused on stabilizing training, improving image quality, and scaling to larger datasets and higher resolution images [2, 31, 32].

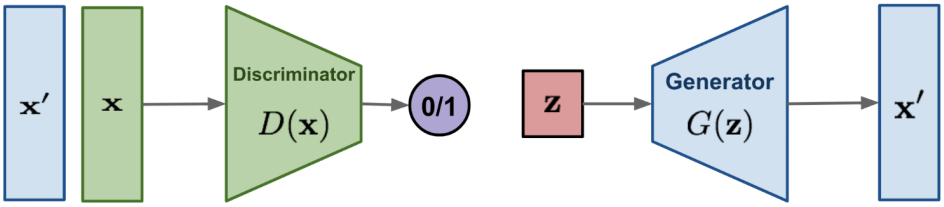

Figure 10: GAN architecture, where x' is the generated image x is the real image and z is a random feature map.

The current SOTA GAN for the ImageNet dataset is StyleGAN-XL [3], which uses the StyleGAN3 [33] architecture for its backbone. StyleGANs are able to extract feature maps for both the style and the content of images (**Figure 11**).

Figure 11: Example of styles and content in images.

StyleGAN-XL uses several ML methods, such as progressive growing and class conditioning, to scale to large datasets like ImageNet. Progressively growing GANs (ProGANs) start training by generating images at very low resolution. Then, as training continues, the image resolution increases. When the image resolution increases, new neural network layers are added to the ProGAN. Progressive growing increases the speed of training while also stabilizing training [32]. GANs have received a lot of attention relative to other generative models, and the number of GAN variations is extensive. In particular, image-to-image mapping applications are where GANs have been exceedingly successful for tasks such as super-resolution and image editing. When generating positive pairs for images, the GAN is very apt due to its fast, non-iterative inference speed, high-quality images, and scalability. The main downside to current SOTA GANs is they rely heavily on supervised learning to scale to large datasets, do not include an encoder model in their architecture, and have a slightly higher FID compared to some image transformers and DMs.

2.1.2 Hybrid Models

As more and more generative computer vision models are developed, we see the lines between VAEs, NFs, ARMs, DMs, and GANs start to blur. Using hybrids of these generative model's neural network architectures often results in higher-quality images and can help address known issues that singular generative model types suffer from.

The VQVAE and VQGAN, in particular, have often been paired with other generative models that have slow inference speeds, like ARMs and DMs. This is because vector quantization can be used to condense data down into 2D discrete feature representations. Because these 2D discrete feature maps are smaller in size than the raw images themselves, they require less training for generative models, which can also generate the feature maps faster. Vector quantization has seen a lot of recent attention and has successfully been used in combination with both image transformers and DMs, resulting in reduced inference time and increased FID scores [15, 34, 35, 36].

As mentioned earlier, using a transformer as the backbone for a DM instead of a convolution-based model can significantly increase the image quality. Replacing the convolutional backbone of GANs with a transformer has also had some success [37]. One way to increase image quality in generative models is to use the adversarial loss found in GANs. Using a discriminator network that is fed real and generated images can often improve the quality of the generated images at the cost of increasing computation and decreasing training stability. The popularity of generative models has exploded over the last year, especially in regard to foundational models like LLMs and large vision models. LLMs can easily be applied to a large domain of NLP tasks in an AGI-like manner. Applying large vision models to a wide variety of downstream tasks is not as straightforward, and more ML methods like GCL are needed to bridge this gap.

2.1.3 Performance Overview

When evaluating which generative model is most appropriate for GCL, there are four key attributes to consider: inference speed (**SC1**), scalability (**SC2**), unsupervised training (**SC3**), and image quality (**SC4**). SOTA VAEs such as VQVAE2 are able to scale to large datasets like ImageNet through the use of methods like vector quantization and hierarchical feature maps. However, the inference time of SOTA VAEs is slow due to the fact they often require a second generative model to synthesize multiple discrete feature maps. Also, the images produced by VAEs have low FID scores relative to the other generative models (much lower than ARMS, DMs, and GANs). Typically, VAEs do not require labels during training as generative models that have an encoder built into their architecture and do not typically require additional conditional inputs such as class labels. Overall, SOTA VAEs like VQVAE2 are a poor choice for GCL on large datasets due to their low-quality images and slow inference time.

NFs are the only generative model that has not been successfully applied to the ImageNet dataset at 256x256 resolution. This inability to scale to large, high-resolution datasets makes NFs intractable and difficult to use. They are better suited to low-resolution datasets and domain-specific datasets, e.g., CelebA, FFHQ, AFHQ. Their inference speed is typically non-iterative and relatively fast compared to other generative models. The quality of images produced by NFs is very low, making using them hard to justify. NF models do not require labels to train and have relatively stable training due to the explicit feature mapping to a probability distribution. The fact that NF models have a single 1D feature map that follows a well-mapped probability distribution makes them apt for augmenting images. However, their lack of scalability to larger datasets and resolutions makes NF models unusable for our GCL implementation.

ARMs scale well to large datasets like ImageNet and to other highly-dimensional data like discrete feature maps. Their inference speed is highly iterative, making them slow to sample and produce positive pairs of images. When paired with parallel decoding, the inference time of ARMs is increased substantially and can then be efficiently applied to a large image dataset. The image quality of ARMS is

SOTA when used in a hybrid approach with DMs. SOTA computer vision ARMs tend to use labels or some other conditional input, such as natural language, when training and producing samples, as do most SOTA image generation models. In order to use ARMs efficiently for GCL, we would have to make excessive use of parallel decoding.

DMs, similar to ARMs, scale well to large datasets and highly-dimensional data. DMs suffer from the slowest inference times of generative models due to their iterative nature, but in regards to GCL, this could potentially be overcome through the hybridization with another generative model type, specifically GANS. The quality of images produced by DMs is very high, and DMs are a very popular area of ongoing generative modeling research. DMs benefit greatly from the use of class-guided synthesis, and all of the SOTA DMs have used class labels on the ImageNet dataset. However, the first implementations of DMs did not use labels while training, so a completely unsupervised DM that produces high-quality images on the ImageNet dataset is available but has fairly low image quality [30]. Overall, DMs would work well for GCL when the issue of inference speed is addressed.

GANs are the clearcut choice for generating large synthetic datasets. GANs are the only generative model that can produce high-quality images (the other two being image transformers and DMs) that can also synthesize images in a non-iterative manner. GANs scale well to large datasets, like ImageNet, when certain ML techniques are applied. These ML techniques increase the GAN's ability to scale immensely. In recent years, ARMs, DMs, and GANs have been very competitive over SOTA image quality on the ImageNet dataset. These models also typically use labels to improve their scalability. The use of labels or even pseudo labels helps generative models divide up large amounts of data, allowing them to map groups of images to different feature maps instead of just one. When labels are not used, these generative models' FID scores drop significantly. Unfortunately, developing unsupervised generative models is an area of research that has been neglected in favor of developing supervised generative models that are easier to scale and produce higher image quality. This poses a major impediment to the development of GCL.

Table 1: An analysis of each generative model type in regards to attributes needed for GCL.

	VAE	NF	ARM	DM	GAN
Scalability	✓	✗	✓	✓	✓
Inference	✗	✓	✗	✗	✓
Quality	✗	✗	✓	✓	✓
Unsupervised	✓	✓	✗	✗	✗

2.2 Contrastive Learning

Since the advent of backpropagation, ML has relied heavily on supervised learning to train models. Supervised learning requires data that is hand-labeled, where the data is the input and the labels are the target when training the supervised learning model. The amount of data in the world grows exponentially, but the majority of this data is unlabelled [38]. Hand labeling data is slow, tedious work that cannot keep up with the current growth of unlabeled data. Therefore, new ML methods that require either no labels (unsupervised learning) or only small amounts of labeled data (semi-supervised learning) must be developed. Unsupervised learning is able to extract information from unlabeled data but cannot apply this information to solve downstream tasks. Common forms of unsupervised learning are clustering and anomaly detection. Self-supervised learning is similar to unsupervised learning, with the exception that it allows us to extract information from the data in a way that allows us to later apply it to downstream

tasks, i.e., learned representations. In computer vision, image encoders are the most prevalent type of self-supervised model, as they allow us to encode images into meaningful representations we can then use for downstream tasks. In particular, contrastive learning approaches have shown promising results in generating meaningful, high-performance self-supervised representations. The barebones framework for contrastive learning consists of three parts: an image augmenter, an encoder, and a loss function. The augmenter is inputted an image, which is then augmented using a set of augmentations such as cropping, flipping, Gaussian noise, etc. The augmenter is used to randomly change low-level features in the image while preserving high-level ones. Typically, the augmenter produces two augmented images to form a positive pairing. The encoder then takes the positive pair and outputs a 1D continuous representation for each of them. Because both images contain the same high-level features, despite having minor distortions from the applied augmentations, they should have approximately the same representations. A contrastive learning loss function is used to calculate the similarity between the two representations while preventing the training from collapsing. A common problem when training contrastive learning models is that the encoder may learn to output a constant representation, as outputting the same representation for every image will ensure the perfect similarity between representations. There is a large range of contrastive learning loss functions, but each of them calculates the similarity between the positive pair's representations. Negative pairs are commonly used alongside positive pairs as these can prevent mapping to a constant representation. A negative pair is simply two images that contain different high-level features and, therefore, should have different representations. The negatively paired images are encoded, and the similarity between their representations is increased by the contrastive learning loss function. Negative pairs are used to stabilize training, as simply using the similarity between representations will cause the encoder to output constant representations. Contrastive learning methods that do not use negative sampling are referred to as non-contrastive learning methods. One of the downsides to negative sampling is ensuring that the negative samples are not the same class. For example, two different images of an apple may be sampled, and then the difference between their representations is maximized even though they are images of the same object. Finding suitable sets of negative samples for each data point is

a common research topic in contrastive learning but does add to the overall computation of the contrastive learning model. Another method for preventing the encoder model from outputting a constant representation is the use of asymmetric encoders. This is done by using two different encoders; the first encoder, typically referred to as the student or online network, is trained using backpropagation. The second encoder, typically referred to as the teacher or target network, does not update its weights using backpropagation but instead uses an exponential moving average of the student's weights. This exponential moving average ensures that the output of the student and teacher models are always different and, therefore, prevents the encoder from mapping to a constant. The downside of using the student-teacher model is that the weights of the teacher model must be stored, although the updating of these weights is computationally small compared to backpropagation as they do not require storing their gradients. Using two differing models when training a contrastive learning framework is known as asymmetric contrastive learning. Other asymmetric methodologies exist besides exponential moving average; however, these methodologies are not as computationally efficient and have yet to produce a higher downstream accuracy [58]. Typically, either negative samples or asymmetric training is used in contrastive learning, although symmetric and non-contrastive contrastive learning frameworks do exist, e.g., Barlow Twins.

The encoder of a contrastive learning framework is comprised of two networks: the backbone and the projector. The backbone is typically a ResNet model such as a ResNet50, ResNet152, etc. The projector is a shallow neural network that is typically comprised of three neural network layers. Once the images are encoded with the backbone and then the projector, we apply a contrastive learning loss to the outputted representations. Using a projector has proven essential to contrastive learning, as using the output from the backbone directly will result in low-quality encodings [40]. Once training the encoder is complete, the projector is typically discarded. Exactly why using a projector when training a contrastive learning model is not fully understood, but adding several linear layers to the backbone model in the form of a projector model does affect downstream accuracy significantly.

2.2.1 Self-Supervised Benchmarks

Self-supervised learning representation quality is typically measured using two different evaluations on the ImageNet dataset: a linear evaluation and a semi-supervised evaluation. For both evaluations, typically, only the backbone is used, and the augmenter and projector are discarded. For the linear evaluation, we freeze the weights of the encoder and then train on 100% of the labeled training data. The outputted representations are then inputted into a single-layer neural network, a.k.a. the linear classifier. The linear classifier then classifies the representations into the hand-labeled classes of the training dataset. After training the encoder and linear classifier, they are then evaluated on the hand-labeled validation data. This evaluation uses top-1 and top-5 accuracies. Top-1 accuracy means that the class of the highest probability must exactly match the true label from the validation data to be considered correct. The top-5 accuracy is when the true label from the validation dataset is one of the five highest probabilities outputted by the linear classifier. Different backbones can be used for the encoder architecture, with larger backbone models providing better results. The standard backbone of contrastive learning is the ResNet-50 model, which is also the smallest of the ResNet models. However, larger variations of the ResNet model, e.g., ResNet-152 and ResNet-200, have been found to increase accuracy on downstream tasks while also requiring fewer training samples. The use of vision transformers as a backbone instead of convolutional-based models has increased in popularity as they provide comparable accuracies and computation. Due to the transformer's native attention mechanism, they can be used to extract saliency maps in an unsupervised manner when trained with contrastive learning. Increasing the model size in self-supervised learning is an interesting dynamic because, typically, in supervised learning, using a larger model, especially on smaller datasets, will deteriorate the performance due to overfitting. This does not seem to be the case in self-supervised learning, where larger, more complex backbones significantly increase the performance of contrastive learning without suffering from overfitting (**Figure 12**).

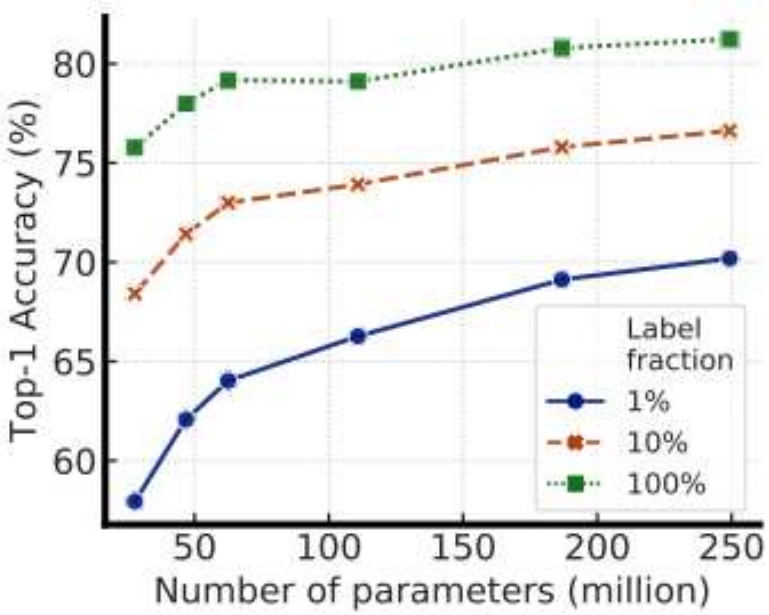

Figure 12: Accuracy plot comparing the size of the backbone with the Top-1 accuracy. This experiment is done using semi-supervised evaluation on 1%, 10%, and 100% of the data and is taken from [41].

Linear evaluations are useful to compare self-supervised learning methods directly to supervised learning. The current SOTA contrastive learning framework for linear evaluation on the ImageNet dataset with a ResNet50 backbone is ReLICv2 for both top-1 and top-5 accuracies [39]. Unfortunately, neither the code nor the pretrained encoder is available for ReLICv2, making it difficult to compare to other contrastive learning frameworks. ReLICv2 is the first self-supervised learning method to outperform SOTA-supervised learning methods on the ImageNet dataset. This shows that self-supervised learning can be competitive in performance and that training the encoder in an unsupervised manner can result in higher accuracy than training in a supervised manner. Likely, the fact that overfitting affects supervised models more than it affects self-supervised models plays a key role in this.

Although linear evaluations are useful for measuring the information held by representations, having 100% of the dataset being labeled is atypical in real-world applications. Real-world applications usually only have small amounts of the data labeled. In semi-supervised evaluation, only 1% or 10% of the training data is used for training a single linear layer. Also, during semi-supervised evaluation, the weights of the encoder are not frozen but instead fine-tuned with a small learning rate. The single-layer

neural network is appended to the encoder to predict the classes from the validation data using both top-1 and top-5 accuracy. The top-1 and top-5 SOTA for both 1% and 10% of the ImageNet training data using a ResNet50 backbone are achieved using the SimCLRv2 contrastive learning framework [40].

The ImageNet dataset is the standard for evaluating self-supervised learning representations, as well as computer vision classification and generative models. The most current ImageNet dataset created in 2017 is the ImageNet Large-scale Visual Recognition Challenge (ILSVRC), which contains 1,000 classes and 1,281,167 training images, 50,000 validation images, and 100,000 test images [4]. The standard resolution for this dataset is 256x256; however, a 512x512 version is also available. In addition to the ILSVRC version of the ImageNet dataset, an ImageNet21K version also exists, which was created in 2021. The ImageNet21K dataset is much larger as it has 21,841 classes and 14,197,122 images. However, the ImageNet21K dataset is not typically used due to its higher complexity and large memory footprint, making it difficult to train ML models, especially in regard to computationally hungry applications such as generative modeling and contrastive learning. We will likely see the use of larger datasets like ImageNet21K for evaluating future applications of contrastive learning as processing power increases over time.

2.2.2 Augmentation Taxonomy

There are a wide variety of different augmentations in computer vision. Typically, these augmentations are domain-specific and cannot be applied to other data types outside of computer vision, e.g., NLP and graphs. In contrastive learning, augmentations take an image and remove some visual features from it. The contrastive learning model is then tasked with learning these removed features. This aligns well with the goals of self-supervised learning, which is to learn about data from the data itself, which is typically done by masking or removing parts of the data. In computer vision, we can group augmentations into three types: color augmentations, geometry augmentations, and generative augmentations. It is difficult to gauge the effectiveness of augmentations without applying them directly to contrastive learning and calculating the accuracy of downstream tasks. This can be exceedingly

difficult to do as contrastive learning is computationally expensive, and the sheer number of combinations and adjustments we can make to augmentations is combinatorially intensive. Augmentations are typically applied in some random fashion where we can adjust the threshold of how often an augmentation is randomly applied. Also, many augmentations have a level of strength associated with them, e.g., with color jitter, we can adjust the amount of brightness, contrast, saturation, and hue that changes in an image. In contrastive learning, having too few or too weak augmentations will result in poor downstream accuracy, but so will having too many or too strong augmentations. Developing methods to find the optimal configurations for augmentations in contrastive learning is an ongoing area of research and a recurring theme throughout our work.

Color augmentations have proven essential to contrastive learning, and all the SOTA contrastive learning models on the ImageNet dataset distort the color of augmented images [41, 42]. It is difficult to say for certain why color distortions are so important in downstream tasks like object detection as color can be very useful when identifying objects, but some likely reasons are: objects typically have a large range of colors, colors are not unique to any one object type, and object shapes are often more useful to ML models in the context of object detection. Some common color augmentations are color jitter, Gaussian blur, solarization, greyscale, and Sobel filter. Color jitter randomly changes the brightness, contrast, saturation, and hue of an image (**Figure 13**). Typically, in contrastive learning, the brightness and contrast are heavily distorted, while the saturation and hue are less distorted. Gaussian blur is a common contrastive learning augmentation that randomly blurs the color between pixels in a kernel of a set size (**Figure 13**). Gaussian blurs are particularly useful as they often remove smaller, low-level visual features that are typically not very useful for high-level computer vision tasks like classification and object detection. Gaussian blurs typically have a low randomness threshold and are therefore applied often in contrastive learning augmentations. The size of the kernel, as well as the range of the standard deviation for the Gaussian distribution, can be increased to increase the overall blurriness applied to an image. Another color augmentation, solarization, inverts the pixel values above a certain threshold. Solarization is common in contrastive learning, but the randomness threshold is kept fairly high so that it

is not applied often. The solarization pixel threshold is typically equal to half the max pixel value but can be decreased for stronger augmentation. Grayscale is an image augmentation that changes the three bands of an RGB image to a single band for greyscale (**Figure 13**). Greyscale is a very common augmentation in contrastive learning, although the random threshold is typically kept fairly high, so it is not applied very often. A Sobel filter is an image augmentation that measures the normalized gradient magnitude of pixels in an image. This creates a grayscale image where the edges of objects are depicted in lighter shades while the rest of the image is depicted in darker shades (**Figure 13**). Sobel is not as common in contrastive learning as the previously mentioned color augmentations but is commonly used in computer vision ML models for edge detection. Generative models that are able to transfer style from one image to another could also be potentially used for color augmentations. Color augmentations have proven essential to contrastive learning on the ImageNet dataset; however, they may not be well suited to other downstream tasks or datasets where color may play a more important role, such as facial recognition and satellite imagery.

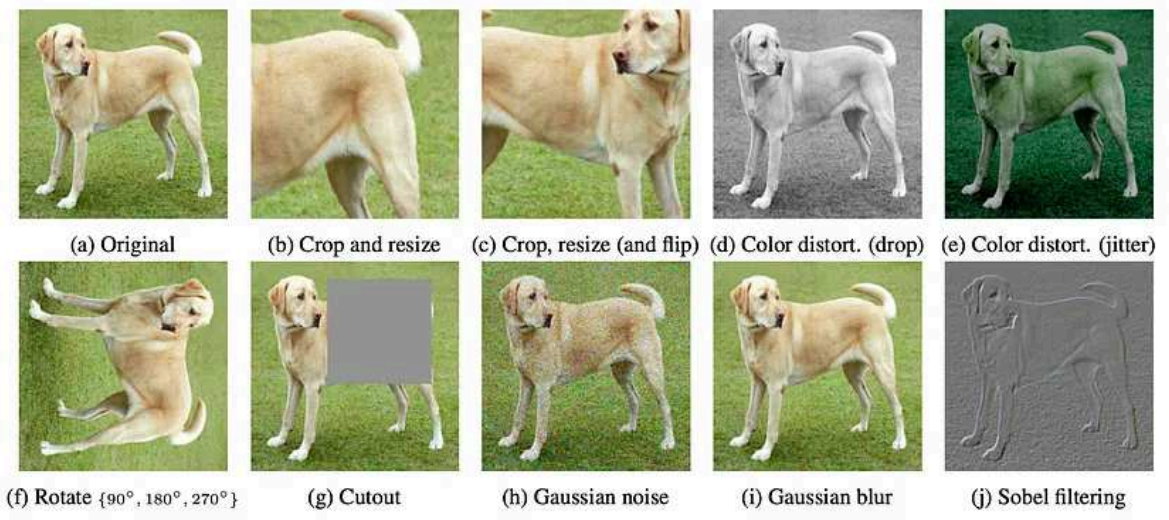

Figure 13: Examples of the augmentations, cropping, flipping, grayscale, color jitter, rotation, cutout, Gaussian noise, Gaussian blur, and Sobel filtering.

Geometry augmentations are image transformations that change the 2D or 3D location of visual features in images. The most common geometry augmentations are cropping, cutout, flipping, rotating, and perspective augmentations. Cropping is one of the most common and best-performing augmentations when used in contrastive learning [41]. It takes a small area of the image and then resizes it to the original image size (**Figure 13**). Typically, in contrastive learning, the random threshold for cropping is low, so it is applied frequently. When image datasets contain images of varying resolutions, cropping is used one hundred percent of the time as it normalizes the size of the image. There are several different cropping methods, but the size of the cropped image can be adjusted in each of them. Very similar to cropping is cutout, which randomly erases a small area of the image, turning all the pixels to black (**Figure 13**). Contrastive learning is often used in NLP in the form of masking and in graph data in the form of edge and vertice masking. This makes cutout, i.e., removing parts of the data, an interesting option for cross-modal data such as text and image datasets. The flipping augmentation swaps the xy coordinates of pixel values across either axes, i.e., horizontal or vertical flipping (**Figure 13**). Horizontal flipping is often used in high-level computer vision tasks, while vertical flipping is not commonly used. The randomness threshold for horizontal flipping in contrastive learning is typically set to 50%. Rotating augmentations are image augmentations that rotate the image around the center point by a randomly sampled amount of degrees (**Figure 13**). After rotating an image, any areas that are now visible but were not present in the original images typically have their pixel values set to zero, i.e., when the image is rotated by a value between 90, 180, 270, and 360 degrees. Rotation is not commonly used in contrastive learning augmentations. Random rotation augmentations can adjust the strength of the augmentation by changing the range of degrees that an image can be rotated. Perspective augmentations move the point of view of the flat 2D image around in a 3D space. The point of view can move the image directly away and/or at a 180-degree angle on both xy axes. Similar to rotation, any visible areas that were not in the original image typically have their pixel values set to zero. Also, like rotation, the range of degrees and distance can also be adjusted for random perspective augmentations. Perspective augmentations are not commonly used in contrastive learning. 3D-aware generative models can also be used to apply geometry

augmentations. Typically, neural radiance fields (NeRFs) models or mesh-based generative models can be used to map an image to a 3D representation and can then adjust the coordinates of the 3D objects or point-of-view. Geometry augmentations are heavily used across computer vision and allow contrastive learning to learn visual features from images by moving or removing them from the image.

Although not as common or as easy to implement as color or geometry augmentations, generative augmentations can also improve the accuracy of contrastive learning on downstream tasks. Generative augmentations require the use of a generative model that can sample and perturb the feature map space to create an image. There are two methods for creating positive pairs in generative modeling. The first is sampling a feature map, creating a duplicate of the feature map, and then slightly changing the values of the duplicate feature map, a.k.a. augmenting the feature map. We take both the sampled and augmented feature maps and have the generator synthesize images for the positive pairing. The more changes we add to the augmented feature map, the more visual changes we see between the positive pair. The second method requires the generative model to have an encoder. We first take a real image and encode it into a feature map, which we then augment, similar to the first feature augmentation method. We then use the generator to map this augmented feature map to the augmented image, which we use with the inputted image as a positive pairing. The key difference between these two methods is that in the former, we synthesize two images for the positive pairing, while in the latter, we use a real image and a synthetic image. Typically, synthesizing both images to create a positive pair has been used in past GCL implementations, as most SOTA generative models either do not have an encoder or the encoder has a slow iterative inference time.

Augmenting the feature map is key to getting high accuracy on downstream tasks, as it allows us to control the strength of the augmentations. The feature map space is typically a 1D continuous encoding, although 2D and discrete encodings are sometimes used, where each of its elements is a float that follows a probability distribution. In many generative models, there is also a conditional input that is used alongside the feature map, such as a class label or text prompt. One of the more common methods for augmenting the feature map space is to add a small amount of Gaussian noise to the feature map. When

sampling Gaussian noise, the standard deviation can be adjusted to increase the strength of the augmentation. Knowing what standard deviation is most effective for creating positive pairs is difficult as it requires the training of a contrastive learning model for different values of standard deviation. Also, the standard deviation of the noise likely does not have a linear relationship to the amount of visual features in the image. The standard deviation will need to be adjusted when applied to different datasets. Another method for augmenting the feature map space is linear interpolation. For this, we sample two feature maps from the feature map space and then take the linear interpolation between the two. When using linear interpolation, we can change the weight of the interpolation, which lets us adjust how similar the outputted feature map is to either of the two inputted feature maps. For example, with a weight of zero, the outputted feature map would be identical to the first inputted feature map; with a weight of 0.5, the outputted feature map would be the mean between each corresponding element of the two inputted feature maps, and with a weight 1.0 the outputted feature map would be identical to the second inputted feature map. We then take the first inputted feature map and the interpolated feature map and input both of them into the generator to create the images for our positive pairing. The strength of the feature augmentation can be adjusted by increasing the weight of the linear interpolation. Another possible direction for generatively augmenting images is the use of style and content models [33], which have a separate feature map for the content and style of an image. The style feature map can then be randomly sampled while the content feature map is kept constant, likely preserving important visual features in the generated image.

Steering is another method for augmenting feature maps in generative models. Steering trains a single-layer linear model to find changes in the feature map space that minimizes the L2 loss between the original and augmented images. The goal of steering is to manipulate the feature map space of generative models in a way that mimics augmentations that adjust the brightness of an image, shift the images along the xy axes, and zoom in on the image (**Figure 14**). Because steering is using the L2 loss function, the resulting augmentation should retain many of the visual features of the original image. Steering is well-suited to creating positive pairs as it retains much of the visual features, making the likelihood that the augmentations are too strong relatively low.

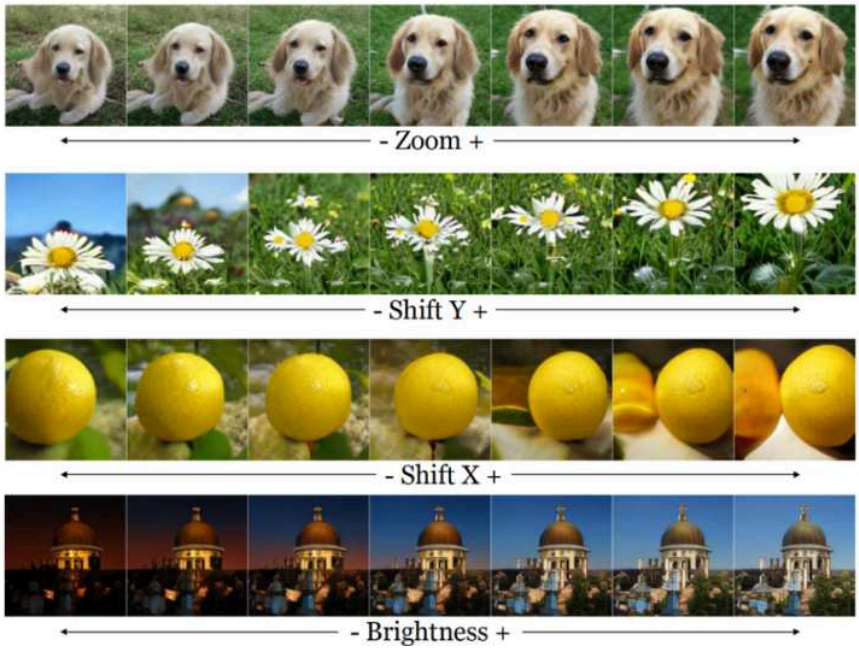

Figure 14: Images produced by steering the feature space that mimics the effects of zooming, shifting the xy axes, and adjusting brightness.

The last generative augmentation strategy is a navigator. Similar to steering, a navigator also uses a single-layer neural network model to augment the feature map across the feature map space. The main difference between steering and the navigator is that instead of minimizing an L2 loss, it trains in an adversarial learning method with a contrastive learning encoder. This works by first sampling a feature map and then inputting it into the navigator, which outputs the augmented feature map. Both these maps are inputted into the generator, creating a positive pair. The positive pair is then inputted into the contrastive learning model, which outputs the contrastive learning loss. The contrastive learning model is then backpropagated against this loss, while the navigator is backpropagated against the inverse of the contrastive learning loss [43]. After training a navigator, it should then produce an optimal feature map augmentation for contrastive learning. However, the current literature on training a navigator and its effectiveness is lacking. This is likely due to the fact that generative augmentations are rarely used with contrastive learning as they require the training of a computationally expensive generator model. However, on smaller datasets, where using generative models is not as expensive, feature augmentations

have shown promising results both in generating data for sparse datasets and creating positive pairs for contrastive learning [44, 45].

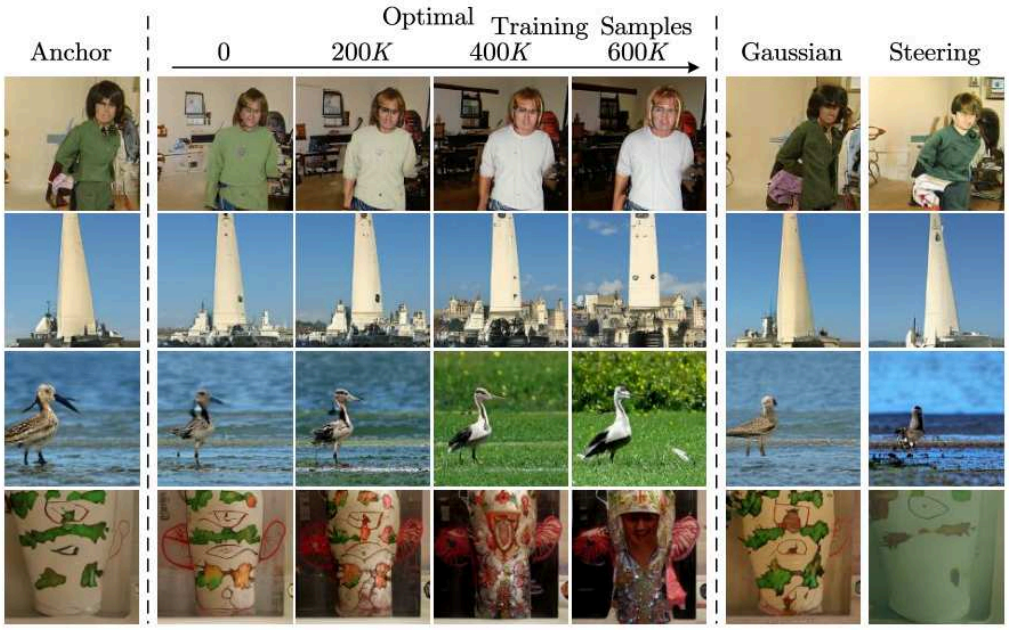

Figure 15: Examples of Gaussian noise, steering, and the navigator (optimal) as generative augmentation strategies.

2.2.3 Contrastive Learning Frameworks

There have been many contrastive learning frameworks over the years; however, two of the most commonly used ones are a Simple Framework for Contrastive Learning of Visual Representations (SimCLR) and Bootstrap Your Own Latent (BYOL). SimCLR uses a symmetric encoder and uses negative pairings in addition to positive pairings. On the other hand, BYOL uses only positive pairings but also uses asymmetric encoders. Some other notable contrastive learning frameworks include Barlow Twins, Variance Invariance Covariance Regularization (VICReg), Momentum Contrast (MoCo), Swapping Assignments between multiple Views (SwAV), Representation Learning via Invariant Causal Mechanisms (ReLIC), Simple Siamese Representation (SimSiam), self-DIstillation with NO labels

(DINO), and Online Bag-of-Visual-Words (OBoW) (**Figure 16**) [46, 54, 57, 56, 68, 69, 58, 70]. We will explore the contrastive learning frameworks SimCLR and BYOL as they are the first two approaches to use negative pairing and asymmetric encoders. We will also define three more modern contrastive learning frameworks: Barlow Twins, VICReg, and DINO, as they provide SOTA or near SOTA accuracy on contrastive learning benchmarks and are the contrastive learning frameworks we use in our GCL implementations [41, 42, 46].

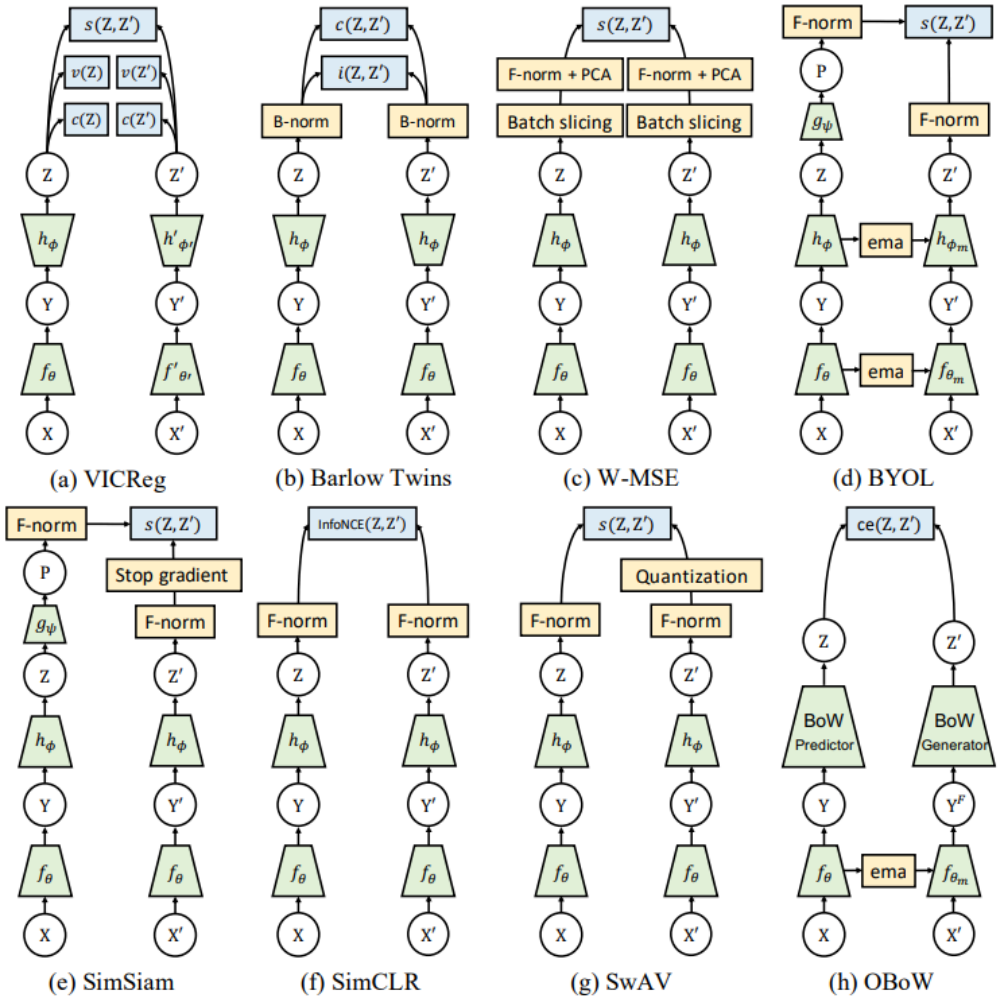

Figure 16: Architecture of common contrastive learning frameworks.

SimCLR follows the standard augmentation schema used in contrastive learning to form positive pairs. This is done by taking a single image and randomly applying the augmentations: cropping, Gaussian blur, horizontal flipping, color jitter, solarization, and grayscale to create a positive pairing of augmented images (z_i, z_j). Negative pairing is done by randomly sampling another image from the datasets z_k and then applying the same set of augmentations used in the positive pair to the negative pair (z_i, z_k). Once the positive pairing and negative pairing are inputted into the backbone and projector, the encodings are inputted into the Normalized Temperature-scaled Cross Entropy Loss (NT-Xent) (**Formula 3**).

$$NTXent = -log(\frac{exp(sim(z_i z_j)/\tau}{\Sigma_{k=1}^{2N} 1_{[k\neq i]} exp(sim(z_i z_k))/\tau})$$

Formula 3: Normalized Temperature-scaled Cross Entropy Loss

NT-Xent minimizes the difference between the positive pair's representations while maximizing the difference between the negative pair's representations. One issue with this is that randomly sampling images from the dataset may produce an image that is similar to the one used in the positive pairing. This is a common issue in contrastive learning, and there are contrastive learning frameworks that have developed systems to pick images that are different from the positive pairing image, specifically in a process known as hard negative sampling, which requires increased computation. Regardless, random sampling of negative pairs works quite well with SimCLR and other contrastive frameworks and prevents the encoder from mapping to a constant function. Pretraining SimCLR is a computationally intensive process on the ImageNet dataset, as the paper used 128 TPUs over 800 epochs [41]. Typically, in contrastive learning frameworks, pretraining is done with at least 16 GPUs or TPUs for 800-1,000 epochs, which would take approximately 16 days for us to complete in our current training environment. For ablation studies, contrastive learning papers typically lower the epochs to one hundred epochs to decrease computation, which is what we use throughout our benchmarks.

Once the pretraining is complete, fine-tuning can begin, which uses the backbone and a single neural network layer for outputting class probabilities. Fine-tuning, unlike pretraining, uses labeled data. The amount of labeled data used is typically 1%, 10%, or 100% of the training data, with the more labels used, the higher the downstream accuracy. The labeled images are inputted into the encoder, which then outputs the class probabilities, which can then be compared to the true labels. Fine-tuning takes significantly less time to train than pretraining. This is due to the smaller amount of weights needing backpropagation in the linear evaluation and the reduced amount of training data in the semi-supervised evaluation. Also, supervised models like classifiers take fewer epochs to train when the backbone has been pretrained using contrastive learning, typically requiring one hundred epochs to train. Once the fine-tuning is complete, the model can be used for classification and evaluated with linear or semi-supervised benchmarks.

There are several key points from the work done on SimCLR that can be applied to any other contrastive learning framework [41]. The first is that the number of parameters the backbone uses heavily influences the model's performance on downstream tasks. A ResNet50 backbone achieves an accuracy of 57.9% when trained on 1% of the labeled data, while a ResNet152x3 achieves an accuracy of 74.9% using the same training schema. The batch size during training also heavily influences the accuracy of contrastive learning frameworks, with the difference in accuracy being approximately 3.5% between batch sizes of 256 and 4,096 using the same model and training schema. Based on the SimCLR paper [41], it seems that models that are trained in an unsupervised manner do not suffer from overfitting in the same way supervised models do. This is significant to ML as supervised training with larger models, especially on small amounts of data, negatively impacts the model's ability to generalize on new unseen data.

As SimCLR is the first and among the most common frameworks for contrastive learning that uses negative sampling, BYOL is the first and among the most common for asymmetric encoders. BYOL uses the same set of augmentations that are commonly used in SimCLR and other contrastive learning frameworks. Instead of using the same encoder (backbone and projector) for its positive pairing, it has two asymmetric encoders known as the online and target networks. The online and target networks are

identical in architecture in BYOL, but in non-contrastive frameworks, the target model and online models can differ in size. The online network is trained using an optimizer, while the target network is an exponential moving average (EMA) of the online network. The target network is updated once every n iteration so that the weights of the target are updated much slower than that of the online model. Once the first image of the positive pair is inputted into the online network and the second image is inputted into the target network, a shallow neural network called the predictor tries to predict the target network's outputted representation from the inputted online network's outputted representation. Based on the difference between the predicted and actual outputted representation, the predictor and online network are updated using backpropagation, while the target network is not (**Figure 17**). Why this training schema prevents mapping encodings to a constant is not fully understood, but having an asymmetry between the two encoders accomplishes the same training stability as using negative pairs. In recent contrastive learning research, it does appear that asymmetric encoders are being used more often than negative sampling methods [58, 71, 72, 73].

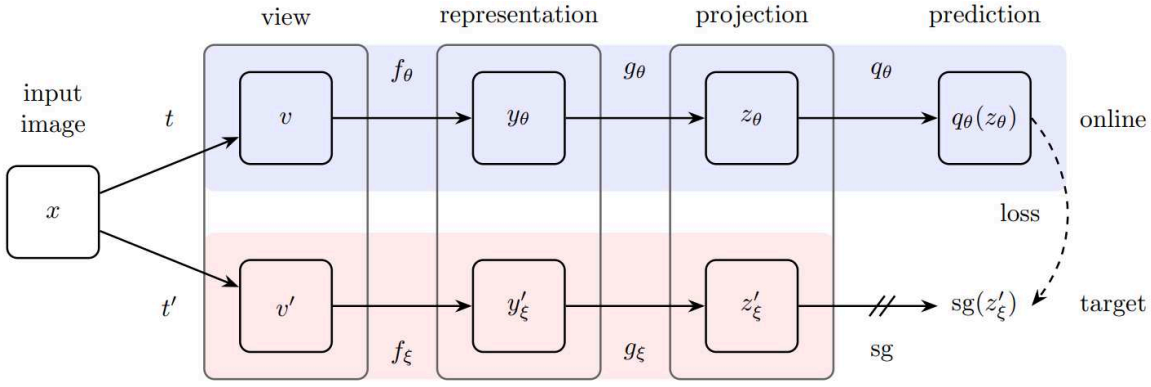

Figure 17: Architecture of BYOL.

BYOL is trained for 1,000 epochs using 64 TPUs [42]. BYOL is less dependent on batch size than SimCLR, but its downstream accuracy is still negatively affected by batch sizes smaller than 1,028. Also, BYOL is still heavily affected by the size of its backbone, as is the norm in contrastive learning

frameworks. BYOL provided SOTA performance on semi-supervised learning when [42] was published, significantly outperforming the first version of SimCLR. However, the second version of SimCLR, SimCLRv2, now outperforms BYOL, indicating that both contrastive learning frameworks will continue to develop and finetune over time. SimCLR and BYOL are among the most commonly used applications of self-supervised learning in computer vision. However, both are computationally expensive and when either the size of the backbone or the batch size is decreased, the performance decreases significantly.

DINO, similar to BYOL, uses asymmetric learning in the form of updating a second encoder with an exponential moving average of the first. Unlike BYOL, DINO does not require the use of a predictor model when calculating the loss. DINO instead applies the softmax function to the outputs of the teacher and student model and applies a cross-entropy loss function to them. Before the cross-entropy loss is applied to the teacher's outputs, they are first sharpened and centered, which are normalization methods that function similarly to the exponential moving average methodology. DINO's downstream accuracy is relatively high compared to similar contrastive learning frameworks while having a relatively low computational footprint compared to previous asymmetric contrastive learning methods. DINO scales particularly well to large datasets and when using large vision transformers.

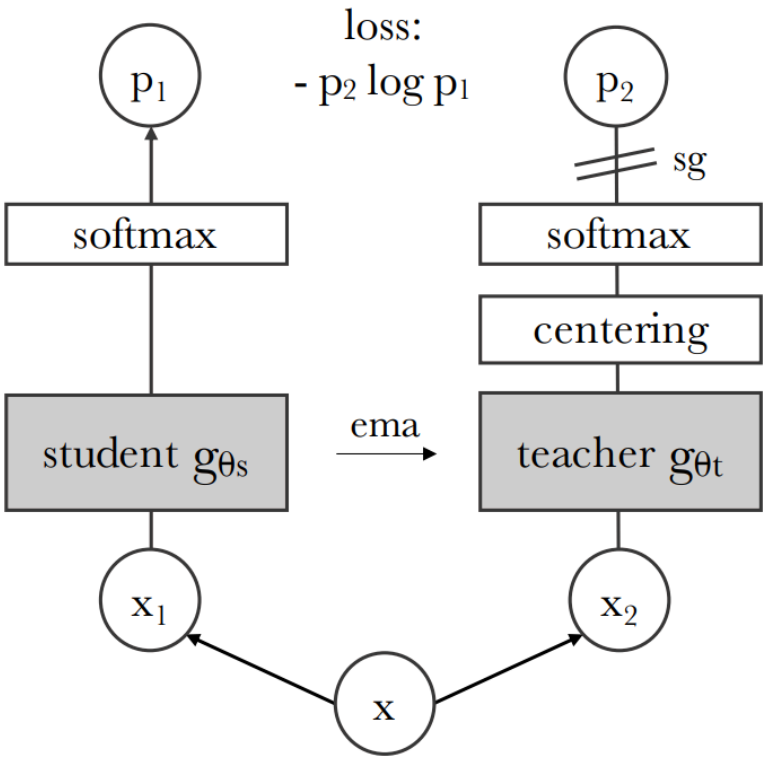

Figure 18: Architecture of DINO.

Barlow Twins is a non-contrastive framework that uses a single symmetric encoder, similar to SimCLR. The loss used in Barlow Twins has shown to be less dependent on batch size than SimCLR and other contrastive learning frameworks [46].

This loss is calculated by taking the cross-correlation matrix of the representations from the positively paired images. The difference between this matrix and an identity matrix is then used to calculate the loss. This is done in two functions: the invariance loss function and the redundancy reduction loss function (**Formula 4**). The invariance loss function takes the mean squared error between an array of ones and the cross-correlation matrix's diagonal values. The redundancy reduction loss function minimizes the squared values of all non-diagonal matrix values. The redundancy loss is typically multiplied by a constant value λ typically set to 0.005.

$$Loss_{Barlow\ Twins} = \sum_i (1 - C_{ii})^2 + \lambda \sum_i \sum_{j \neq i} C_{ij}^2$$

Formula 4: Barlow Twins Loss Function

In past contrastive learning frameworks, either a symmetric encoder or negative sampling needed to be used to prevent the encoder from mapping to a constant. Barlow Twins provides the best of both worlds as the encoder models can be symmetric while also not requiring negative pairs. Also, Barlow Twins' backbone scales down well compared to other contrastive learning frameworks and is less reliant on large batch sizes. This, plus its high performance on downstream tasks, makes Barlow Twins an ideal contrastive learning framework for GCL. However, in our benchmarks, we found that while training Barlow Twins, the gradient will sometimes explode; for this reason, we also use the VICReg contrastive learning framework in our work.

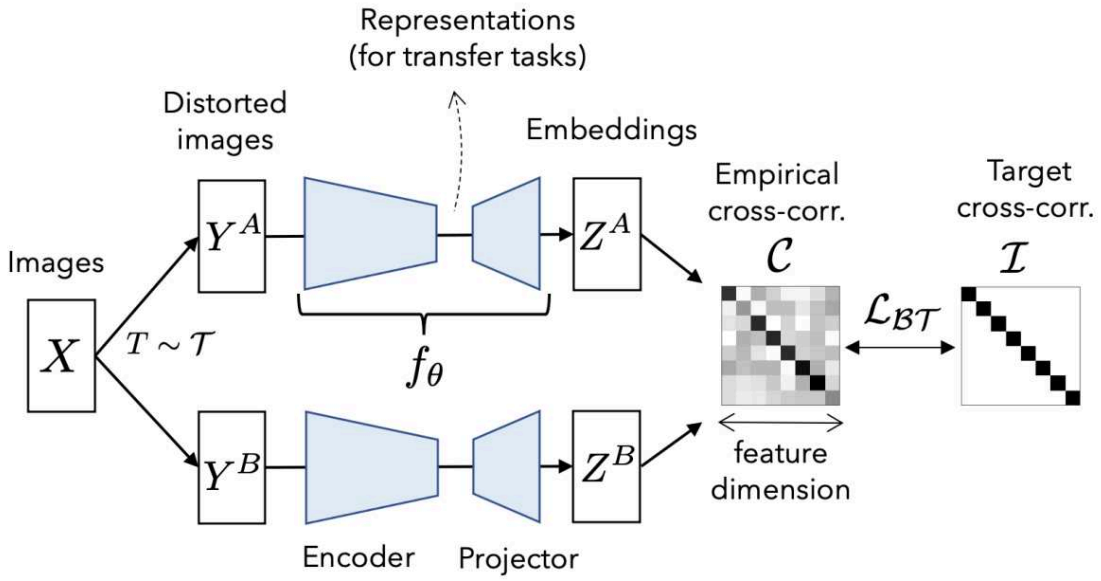

Figure 19: The Barlow Twins' architecture.

The VICReg contrastive learning framework uses negative sampling and a symmetric encoder. The idea behind VICReg is intuitive in that the variance between negative pairings is kept constant, the invariance between positive pairings is minimized, and the covariance between elements in the representations is also minimized. The invariance loss uses the mean squared distance between the positive pair's representations. The variance loss uses a hinge function to keep the variance between the negative pair's representations above a certain threshold. The covariance is calculated between every element in a representation and minimized. VICReg's negative sampling is computationally inexpensive and allows some flexibility for negative samples possibly being the same class with the use of the hinge loss. VICReg's accuracy is very similar, if slightly worse, to Barlow Twins'. However, we found that VICReg's training is much more stable than Barlow Twins and that the training gradient never exploded or vanished. Also, the interpretability and intuitiveness of the variance, invariance, and covariance loss functions gave much insight into how the model progressed during training, which the other contrastive learning frameworks lacked. For this reason, we used VICReg for a large majority of our benchmarks while occasionally using Barlow Twins and DINO.

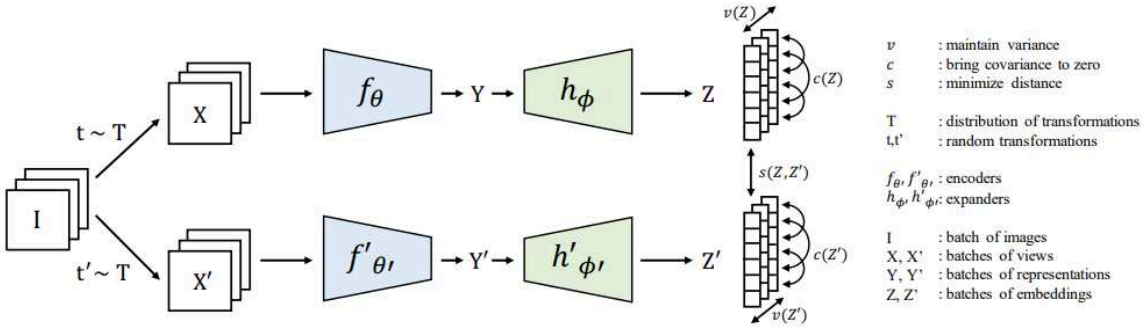

Figure 20: VICReg architecture.

There have been many contrastive learning frameworks over the years. The difference between these frameworks, however, appears to be relatively minor. Similar to supervised learning, many minor ML techniques are applied to these contrastive learning frameworks to slightly increase accuracy. As the

development of contrastive learning frameworks continues, the increases in accuracy seem to be suffering from diminishing returns. In the future, more focus will likely be applied to augmentation strategies and gathering larger datasets, as these also increase contrastive learning accuracy. Instead of focusing on the contrastive learning frameworks themselves, GCL focuses on optimizing the augmentation strategies that they use.

2.3 Previous Generative Contrastive Learning

There have been previous attempts at creating a GCL framework. Typically, these have focused on smaller datasets where data points are sparse, and the use of generative models can help alleviate this sparseness. Several generative models have been developed specifically for this approach [47, 48, 49]. However, few papers have focused on applying GCL to large datasets such as the ImageNet dataset, where instead of sparseness, scalability becomes an issue.

2.3.1 Generative Representations

The first attempt to implement a GCL for the ImageNet dataset, albeit at a lower resolution (128x128 pixels) than is typically used for self-supervised learning benchmarks, was Generative Representations (GenRep). GenRep generates synthetic images using a pretrained Big Bidirectional Generative Adversarial Network (BigBiGAN). BigBiGAN is a variation of BigGAN that uses an encoder instead of class labels, making its training completely unsupervised. However, BigBiGAN is outdated, with a relatively low FID score of 20.32 at a resolution of 128x128 on the ImageNet dataset. In the work done in [50], the BigBiGAN model was not trained to 256x256 resolution due to the author's computational restraints. In GenRep, the authors use two different generative augmentation strategies. The first augmentation strategy simply adds Gaussian noise directly to the BigBiGAN's feature map (**Formula 5**).

$$W_{Gaussian} \sim N^{t}(\mu, \sigma, t)$$

Formula 5: Gaussian Noise Feature Augmentation

$$Z_{Augmented} = Z_{Original} + W_{Steer}$$

Formula 6: Steering Feature Augmentation

The second augmentation strategy was based on the steering method developed in [51] (**Formula 6**). The authors found that using Gaussian noise performed worse than steering. They also found that using handcrafted augmentations outperformed both of their generative augmentations, i.e., their generative augmentations did not improve downstream accuracy. They used the SimCLR framework for all of their experiments. One inconsistency with their results is that their real images' linear Top-1 accuracy only achieved an accuracy of 43.90%. In the first SimCLR paper, the authors were able to achieve an accuracy of 69.3% using the same ResNet50 backbone as GenRep. Likely, the large difference between these two scores is due to the GenRep author's lack of computational resources or inconsistencies in replicating the original SimCLR framework. Regardless, the authors found that using their SimCLR implication on synthetic data resulted in slightly worse results and concluded that this was likely due to the low quality of the images created by the BigBiGAN. The authors speculate that using a generative model with a better FID would likely result in better contrastive learning downstream accuracy, although no results were given to confirm this.

2.3.2 COP-GEN

The second and last paper to apply GCL to the ImageNet dataset, again at 128x128 resolution, was Contrastive Optimal Positives (COP-GEN) for self-supervised contrastive learning [43]. This paper repeated the experiments from GenRep and added an extra feature augmentation strategy, the navigator.

The navigator is a shallow neural network that learns how to optimally augment the feature map space of the BigBiGAN model (**Formula 7**).

$$Z_{Augmented} = Z_{Original} + Navigator(Z_{Original})$$

Formula 7: Navigator Feature Augmentation

During the navigator's training, the weights of the BigBiGAN are frozen, and the sampled feature map $Z_{Original}$ and the augmented feature map $Z_{augmented}$ are passed into the generator. The outputted positive pair is then inputted in the SimCLR framework, which outputs the loss. This loss is then used to update the SimCLR model, while the negative of the SimCLR loss is used to update the navigator. Interestingly, the results of COP-GEN find that both the generative augmentations for steering and the navigator provide better downstream accuracy than using solely handcrafted augmentations (**Table 2**). This directly contradicts the results found in GenRep that found the generative augmentation steering underperforms using solely handcrafted augmentations.

Table 2: Linear evaluation of using synthetic images produced by Gaussian noise, steering, and navigator (COP-GEN) and solely handcrafted augmentations on SimCLR framework taken from [43].

Method	$T_{\mathbf{z}}$	$T_{\mathbf{x}}$	ImageNet Linear Evaluation	
			Top-1	Top-5
Training on ImageNet-1K real images:				
SimCLR	N/A	SimCLR Aug.	49.44	75.58
Training on BigBiGAN synthetic images:				
GenRep	Gaussian	SimCLR Aug.	48.73	73.13
GenRep	Steering	SimCLR Aug.	51.19	74.97
COP-Gen	Optimal	SimCLR Aug.	**53.25**	**77.16**

Again, the results they produced are far from the results found in the original SimCLR paper and other contrastive learning framework papers. The accuracy for using real images linear Top-1 accuracy on the SimCLR framework was 49.44%, making these results questionable and difficult to interpret. Although COP-GEN's results indicate that GCL can outperform using real images, due to the low accuracies achieved in their experiments, it's difficult to tell how accurate these results are. Overall, GenRep and COP-GEN both implemented GCL frameworks but found conflicting results. Due to the low accuracies achieved on their benchmarks, it is difficult to conclude the effectiveness of these past GCL frameworks.

3. Overview

In the previous chapters, we have covered the foundations of generative models, contrastive learning, and previous attempts at GCL. We have shown that generative models possess the ability to augment images in virtually unlimited ways, but finding methods that also maintain important visual features in images is difficult. However, once augmentation methods that preserve important visual features are developed, they can then be directly applied to contrastive learning and used for downstream tasks. As generative models and contrastive learning are both computationally intensive tasks, finding computational efficiencies in using both in a single ML architecture is key to the usability of GCL. Our contributions to GCL can be categorized as the feature augmentation strategies we develop (**RQ1**), the study and optimization of using generatively augmented images in a contrastive learning setting (**RQ2**), and the computational efficiencies that can be used in a GCL architecture (**RQ3**).

3.1 Feature Augmentation Strategy (RQ1)

Current generative models do not possess the ability to identify important high-level visual features when learning to synthesize data. In order to augment images while preserving these high-level features, we make the assumption that the more similar two images' feature maps are, the more likely they contain the same high-level features. In our work, we augment the feature maps using direct and indirect methods. Direct methods do not require a second ML model to augment the feature map of an image and instead make use of the continuous distribution the feature map follows or native feature map augmentation features to the generative model (e.g., the noise vector in GANs). Indirect feature map augmentation methods require the use of a second model to augment the feature map space. Indirect feature map augmentations are able to utilize outside information, such as natural language or masks, to maintain important visual features while augmenting an image. Indirect feature augmentation methods are

particularly useful when generative models have complex or multiple feature maps or when using feature maps that make use of discrete distributions.

Being able to quantify the strength of an augmentation applied to a feature map is key when preserving high-level visual features. This allows us to adjust the strength of a generative augmentation depending on the accuracy of downstream tasks. Ideally, the strength of a generative augmentation should be normalized between zero and one, but this is not always the case, particularly when adding Gaussian noise directly to a feature map. When normalized, a generative augmentation strength of zero returns an image with a feature map identical to the original image's, while a strength of one generates a completely unique feature map that may or may not have any similar visual features to the original image. In this context, the strength is similar to sampling a unique feature map and interpolating between it and the original image's feature map. The strength of our generative augmentations is functionally the same as the strength of color and geometry augmentations.

In our GCL architecture, we also make the assumption using a single real image and a single synthetic image in the positive pair provides better results than using two synthetic images. We base this on the accuracy of previous attempts at GCL and the fact that the current generative models cannot produce images that are perfectly photorealistic. Having one real image in the positive pair increases their aggregated image quality and ensures that at least one of the images will not contain hallucinations produced by the generative model. Note that both images in the positive pair do not have to contain the same visual feature for the contrastive learning model to learn, but that important visual features should be inferable from one image to the other. Implementing a generative augmentation strategy that augments a real image requires the use of an encoder model. This severely limits the number of generative models we can use in our GCL implementation. Another constraint is the use of the inference time of generative models. As modern generative models often learn multi-dimensional iteratively, learning the data in parts, we are again limited to a small set of generative models that have fast enough inference to make GCL viable in our current environment. Finally, in our work, we select generative models that are GANs, image transformers, and diffusion models, as these are the only modern generative models that scale and

have achieved SOTA or near SOTA image quality. In our methodology, we provide implementations for each of the three types of generative models as well as self-supervised benchmarks.

For our GCL implementation of GANs, we make use of a pretrained encoder model and the noise input vector. We map an image to the GAN's feature map and then sample noise, which is inputted into the GAN's noise vector. Encoding the image ensures the generatively augmented image has some similar visual features to the original image, while the noise vector ensures that there are also some variations. This implementation of GCL is the most straightforward to implement and does not require a second ML to augment the image, but it also does not have an easily controllable strength value. In our implementation, we make the assumption that the encoder model is preserving the visual feature we want, which we validate later using self-supervised learning benchmarks. This GAN-based augmentation strategy demonstrates the trade-off between using direct feature augmentations, which do not require a second ML model but make the feature augmentations imprecise, and indirect feature augmentations, which require a second ML model but make adjusting the augmentation strength more precise.

For our second GCL implementation, we use the image transformer architecture, specifically a bidirectional transformer. The bidirectional transformer is well-suited to generative augmentations due to its fast inference times and its native ability for in-painting images. We make use of this in-painting in our GCL implementation by masking parts of the image and having the image transformer predict the masked values. The more masked values applied to the image, the larger the difference in visual similarities between the original image and the augmented one. Generating a completely random mask, where each pixel has the same probability of being masked, is the most straightforward implementation of masking and makes adjusting the strength augmentation the same as adjusting the probability that a pixel is masked. However, using domain-specific information to form the mask can also be used to help improve the masking strategy. We make use of the cropping bias when implementing our transformer-based augmentations on the ImageNet dataset by using masks that focus on the center of the image where important visual features are often located. In addition, we also implement an out-painting augmentation schema that predicts an area around the image. We also apply a uniformly random masking to a satellite

image dataset. Our transformer-based augmentations are indirect feature augmenters as they require the use of a VQGAN and a transformer model to augment; however, they make adjusting the strength of the augmentations precise and easy.

For our third and final GCL implementation, we make use of DMs as the generative model. Because so much previous work on DMs has focused on mapping natural language to images [74,75,76], we have developed a GCL architecture that utilizes natural language inputs. Again, we are using an indirect feature augmenter in the form of a captioning model. Captioning models are ML models used for generating text on images and videos. By applying a captioning model to the images in a dataset, we are extracting important visual information in the form of text. We then input the text alongside the image into the DM, which outputs the generatively augmented image. Similar to transformers, DMs learn data in a divide-and-conquer methodology, learning different parts of the image at different steps during inference. As our transformer-based augmentations start with part of the real image and then generate the remaining, we do the same with our DM-based augmentations. However, with DMs, instead of masking, we are applying varying levels of Gaussian noise to the image, and how many steps are then used in the denoising process. In this DM-based approach, we use the number of steps as our strength adjustment variable. One could potentially also adjust the strength of these augmentations by adding variations to the inputted text; however, this seems to add an unnecessary level of complexity. Instead, we use the captioning model as a way to add noise to the DM model's feature map, causing variations in the visual features. Potentially, a DM that does not use natural language could be used, but to our knowledge, none currently exist that have fast enough inference to be used in the context of GCL. Unlike GANs and image transformers, many large pretrained DMs are open-source, allowing the use of foundational models (e.g., Stable Diffusion, Sora, DALL-E) to be used in our GCL research.

In our work, we have outlined feature augmentation strategies for the three main types of generative image models. Our augmentations make use of the architectures and feature mapping native to these generative models. We believe this work lays a framework for other generative augmentation strategies to build on and outlines the concept of direct and indirect feature augmentations in the context of GCL.

3.2 Leveraging Augmented Images (RQ2)

So far, we have focused on the development of generative augmentation strategies; however, we have found throughout our research that using generative augmentations in tandem with handcrafted augmentations boosts downstream accuracy. There are two ways of using handcrafted augmentations with generative augmentations: mixing and pairing.

Mixing refers to using either generative augmentations or handcrafted augmentations. Mixing can be defined using random sampling, where the mixing rate is a number between zero and one. If a number randomly sampled from the zero and one range is less than the mixing rate, a generative augmentation is applied, and if it is greater, a set of handcrafted augmentations is applied instead. When the mixing rate is set to zero, no generative augmentations are applied, and when the mixing rate is set to one, only generative augmentations are applied. In our research, the optimal mixing rate varies with both the generative augmentation strategy and the dataset used. So, finding the mixing rate will have to be fine-tuned for each application. However, we have consistently found that using a mixing rate between 0.25 and 0.75 consistently performs the best across our benchmarks. When applying mixing, we reuse the same set of handcrafted augmentations that are standard in contrastive learning.

Pairing refers to applying a set of handcrafted augmentations after an image is generatively augmented. Pairing can be defined by the set of handcrafted augmentations used. Each of these handcrafted has its own sampling rates, but we reuse the same parameters that are typically used on contrastive learning applications. We have found that using the full set of handcrafted augmentations typically used in contrastive learning actually degrades performance when paired with generative augmentations. Specifically, the use of color augmentations negatively impacts downstream performance, while the use of geometry augmentations increases downstream accuracy. For this reason, we recommend adjusting the set of augmentations used in different GCL applications.

Based on our benchmarks, it appears that two factors affect mixing and pairing: image quality and structural similarity. When the image quality of the generative augmentations is higher, using a higher

mixing rate increases performance. Inversely, when the image quality is higher, using fewer handcrafted augmentations in pairing increases downstream accuracy. We find a similar pattern with structural similarity where generative augmentations with a higher structural similarity benefit from a higher mixing rate and a smaller set of pairing augmentations. In our work, we defined two interactions between generatively augmented images and handcrafted augmentations and discussed patterns that we found that can be beneficial when implementing GCL.

3.3 Computational Efficiencies (RQ3)

Generative models are among the most computationally expensive and data-hungry ML models. Our work makes excessive use of pretrained generative models and combines them with another computation-hungry ML methodology, contrastive learning. As contrastive learning models are sensitive to batch size during training, it is important to keep additional computation during training as minimal as possible. It is also important that the generative augmentations are not overly computation-heavy, especially in relation to the amount of computation spent on training the contrastive learning model itself, as this affects the overall usability. In our work, we rely heavily on precomputing augmentations and efficient sampling methods.

Minimizing the GPU memory consumption during training allows for larger batch sizes, resulting in a higher downstream accuracy. When training the contrastive learning model, the model weights, as well as both images in the positive pair, must be in the GPU memory. Generatievly augmenting images requires the model weights as well as the inputted image to be in the GPU memory. Unlike the contrastive learning model, the generative model's weights are not being trained and do not require the gradient to be stored in the GPU memory, decreasing GPU consumption significantly. Preprocessing needs to be done when generatively augmenting images, where the generative model is run on the GPU and the augmented images are stored on disk. Storing a single generatively augmented image for each training image doubles the amount of storage. However, when considering the price and relative size of GPU memory compared

to on-disk memory, this preprocessing is the most cost-effective manner to implement generative augmentations. When actually training the model, moving both the original and generatively augmented image from the storage to the RAM is slower than simply retrieving the original augmentation. However, this retrieval process is always parallelized, utilizing multiple CPU threads, when training ML models and is typically not a bottleneck during training. When using solely handcrafted augmentations, the CPU duplicates the image and then applies the augmentations before storing both images in the RAM. This means that the computation of moving the positive pair from the RAM to the GPU for handcrafted and generative augmentations is identical. The movement of data from the RAM to the GPU is one of the largest bottlenecks when training ML models and is unaffected by the use of generative augmentations. Throughout our benchmarks, we have found virtually no difference in the training times of the contrastive learning model when using generative augmentations.

This leaves the majority of the increased computation in implementing GCL in the preprocessing of the generative augmentations. We have previously discussed how two of the three major generative models, DMs and image transformers, use iterative inferences when generating data. This creates the largest bottleneck in regard to GCL, as handcrafted augmentations are near instantaneous. To cut down this time, we only create a single generative augmentation per image, which differs from handcrafted augmentations, which create a number of unique augmentations equal to the number of epochs during training. Interestingly, we find that using multiple generative augmentations per image does not increase downstream accuracy significantly. Having to generate only a single generatively augmented image reduces computation significantly, as generating two generatively augmented images doubles our preprocessing time and storage. When selecting generative models for GCL, any GAN can be used as long as the encoder model is non-iterative, e.g., does not use GAN inversion, as they can augment images in a single iteration. For image transformers, only bidirectional transformers should be used as they can utilize parallel decoding, reducing the number of iterations during inference to the 8-12 range. As the majority of SOTA image transformers use bidirectional transformers by default [27, 63, 78], this does not necessarily limit the pool of image transformers that can be used in GCL. DMs are significantly slower

than any other generative models, making it difficult to justify their use in GCL. In our work, we make use of a DM and GAN hybrid architecture that first trains a DM, which uses hundreds of iterations during inference, as a teacher model and then trains a GAN-like model as a student. Once trained, the GAN-like model is able to generate images in as little as 1-4 iterations, depending on the image resolution. This type of hybrid architecture is, to our knowledge, among the first DMs that are able to generate images in real-time and enable us to use DMs in GCL. It is difficult to put an exact value on the amount of time that is appropriate for preprocessing generatively augmented images relative to training the contrastive learning model itself. Spending more time on the preprocessing than the training itself seems excessive and likely to deter others from using GCL. In our work, the image transformer is the slowest generative model when generatively augmenting images, using 12 iterations when generating an image. Generating a single set of images with these transformer-based augmentations takes approximately a quarter of the time to train a contrastive learning model for 100 epochs or 2.5% of the time to train a contrastive learning model for 1,000 epochs. In different environments, preprocessing times will vary, especially when GPUs with more than 24GB of memory are used, which will speed up the preprocessing significantly (our batch size for preprocessing was typically around 16 per GPU, which is relatively small). Although not computationally free, GCL takes only a fraction of the time it takes to train the contrastive learning model itself while boosting downstream accuracy. Our work has shown the costs and benefits of using preprocessing in regard to GCL and outlined the types of generative models computationally appropriate for GCL.

4. Methodology

In our methodology, we outline the high-level components of our GCL framework and then provide three GCL implementations using GANs, image transformers, and DMs. For each architecture, we outline the needed model characteristics as well as the methodology to augment images efficiently and

effectively. We then evaluate our GCL framework using standard self-supervised learning protocol and analyze its performance.

4.1 Generative Contrastive Learning Framework

The GCL framework is comprised of five parts: an encoder, a feature map augmenter, a decoder, a handcrafted augmenter, and a contrastive learning model.

4.1.1 Encoder

For a generative model to augment an image, it must first be mapped to its feature map, a.k.a. an encoding. Encoding images and other multi-dimensional data is typically done using either a convolutional or transformer-based model. The outputted encodings can be either single or multi-dimensional and follow either discrete or continuous distributions. Single-dimension encodings provide the most compact representation of the image but do not preserve the location of the encoded features. Multi-dimensional encodings preserve the locations of image features; e.g., features from the top left of the image will be represented by the values in the top left of the encoding. Multi-dimensional encodings allow generative augmentations to be applied to specific areas of the image while preserving features in other areas of the image. This is beneficial to creating positive pairs as it allows us to control the strength of augmentations in a more precise manner. Encodings represented using continuous distributions are straightforward to create and use and are common to most generative models. Because they follow a continuous distribution, we are able to randomly sample them and add noise to them in a straightforward manner. Encodings that are represented using discrete distributions are often more efficient and require less memory to store compared to continuous distributions. Vector quantization is the most common methodology used to create discrete distributions in computer vision. Typically, either a VQGAN or VQVAE model is used when computing discrete distributions for generative modeling in computer vision. VQGANs are able to encode larger areas of the image into a single token, typically

16x16 or 32x32 patches, compared to VQVAE, which typically uses patches of 8x8 or smaller. VQGANs require significantly more training iteration than a VQVAE while also requiring the use of a discriminator model, making training them more computationally expensive. Unlike continuous distributions, we are unable to randomly sample the discrete distributions as doing so will create static images. To augment encodings that follow discrete distributions, a second generative model must be trained in order to sample discrete values that are cohesive enough to produce a realistic image. Generative models that are well-suited to learning discrete representations include DMs, ARMs, and NFs. Overall, there are four types of encodings that can be used in generative models: single-dimension continuous, multi-dimensional continuous, single-dimension discrete, and multi-dimension discrete, with single-dimension continuous being the most common, followed by multi-dimension discrete.

4.1.2 Feature Map Augmenter

Finding the optimal amount of change to an image's feature map is the main challenge when applying generative augmentations. For continuous feature maps, simply adding Gaussian noise to the feature map values can effectively augment the inputted image. However, oftentimes, simply adding Gaussian noise to the feature map will result in either too many changes to the image or too few. Increasing or decreasing the standard deviation of the noise can be used to control the amount of change in the augmented image. Finding this magic number for the standard deviation can be accomplished by applying different levels of noise to a few samples and visually judging the changes to the image. For a more precise approach, a dataset should be generated for each standard deviation value and then evaluated using a contrastive learning model and standard self-supervised learning protocol. This approach requires intensive computational resources as well as storage for the generated datasets. Using a smaller dataset, a subset of the current dataset, reducing the image resolution, or reducing the number of training epochs can help alleviate these costs, although they all also reduce accuracy. Some generative models, specifically GAN models, have a noise vector input that is used during training. When this native noise vector is present, a standard deviation value for the Gaussian noise is predefined during the generative model’s training and

should be reused when augmenting images. Adjusting the standard deviation value for generative models with a noise vector will not increase the similarity between the inputted and generated images. The noise vector is mapped to global visual features, and inputting Gaussian noise with a standard deviation of zero (vector of constants) will return a set of random visual features. Noise vector differs from direct feature augmentations where inputting Gaussian noise with a standard deviation of zero will output a generative image identical to the input image.

Multi-dimension continuous feature maps are similar to single-dimension continuous feature maps in that noise can be directly added to the feature map values. However, because multi-dimension feature maps preserve the location of features, augmentations can be applied to specific areas of the image. This creates the problem of which areas should be generatively augmented in order to increase downstream accuracy. There are many potential approaches to this, which can vary greatly in complexity. In our work on the ImageNet dataset, we have found that augmenting the center of the image works best (see section 4); however, we have attempted augmenting the outer edges of the image, as well as augmenting areas based on generated saliency maps. The location of augmentations can either be decided in a uniformly random manner, an informed manner, or a mixture of the two. A purely random augmentation placement would select one or more areas of the image, each with varying sizes similar to the cutout augmentation, and augment those areas. An informed augmentation placement would use information about the image to decide which areas should be augmented, such as cropped images, often placing important features closer to the center of the image. Because contrastive learning benefits from augmentations that have some degree of randomness, we recommend either using a mixture of random and informed augmentation placement or using an informed augmentation placement strategy and adding random handcrafted augmentations to the generatively augmented image instead of the feature map directly.

The main difference between single and multi-dimension discrete feature maps is that augmentation placement cannot be applied to single-dimension encodings. Whenever discrete representations are used in generative modeling, a second generative model must be trained on the feature map. We have found that randomly sampling even a single value in an image's discrete feature map results in the

corresponding area of the image becoming static. Our findings also found that applying weighted sampling based on the discrete distribution from the first generative model blurs the image and does not augment any visual features beyond that. For the second generative model, transformer-based models are typically used as they are well suited to handling multi-dimensional data and discrete tokens. This second model can increase the overall training time significantly. However, due to the high quality of images produced by generative models that utilize discrete representations and their relatively fast inference time, they are well suited to GCL. Comparatively, continuous feature maps are simpler and easier to augment, as they make direct feature augmentation possible; however, the image quality of models that use continuous feature maps currently lags behind generative models that use discrete representations. As discrete representations are more efficient at representing images and scaling to large datasets, using them may be beneficial to environments with more strict space requirements.

4.1.3 Decoder

The decoder of generative models is much larger and more compute-intensive than the decoder. The inference time of the decoder model is essential to applying GCL efficiently as it is the largest computational bottleneck in the generative model. To measure the effectiveness of the decoder model, we compare the inference time to handcrafted augmentations as well as the total training time of the contrastive learning model. Because the decoder models of ARMs and DMs are highly iterative, the number of iterations is a key factor in the inference speed. As reducing the number of iterations in these models negatively affects the quality of the generatively augmented image, making the iterations too small will negatively impact the downstream accuracy and effectiveness of GCL. Due to computational restrictions, our work has been confined to generative models that have 1-12 iterations when decoding an image from a feature map. In addition to the feature map, some generative model encoders have other inputs. GANs have an inputted noise vector, and DMs commonly use natural language as an input. Like the feature maps themselves, these inputs can be adjusted to change visual features in the outputted images. However, these additional inputs add unneeded complexity to the GCL process, as we have found

using solely feature map augmentations to be sufficient enough to improve our GCL's downstream accuracy.

4.1.4 Handcrafted Augmentations

Handcrafted augmentations like cropping, flipping, and color jitter are commonplace when training computer vision models. Ideally, generative augmentations would be able to change the visual features in an image enough that using handcrafted augmentations would no longer benefit the downstream accuracy. However, we found this not to be the case and that both mixing and pairing handcrafted augmentations with generative augmentations directly increases the learned representations' downstream accuracy. When we apply only handcrafted augmentations, we use the typical suite of augmentations: cropping, flipping, color jitter, gaussian blur, solarization, and greyscale. For pairing handcrafted augmentations, we apply a set of handcrafted augmentations to the generatively augmented image. Our results show that after generatively augmenting an image, a much smaller set of handcrafted augmentations, typically only geometry augmentations, is needed and that applying too many handcrafted augmentations degrades downstream performance.

4.1.5 Contrastive Learning Model

Generative augmentations are beneficial to most contrastive learning frameworks. For most of our work, we use three different contrastive learning frameworks: VICReg, Barlow Twins, and DINO. Of these three, Barlow Twins performs the best without the use of multi-crop augmentation. Multi-crop is an augmentation strategy where six additional crops of size 96x96 are inputted alongside the two 224x224 crops typically used for benchmarking contrastive learning. Multi-crop is efficient during training as the smaller images take up little space on the GPU's memory. Multi-crop has been shown to increase downstream accuracy by approximately 4% on ResNet-50 self-supervised linear benchmarks [56]. For our work, when using contrastive learning frameworks that also use multi-crop, we simply crop the

original image. We leave studying the effects of pairing multi-crop with generative augmentations to future work. In addition to providing the highest accuracy, Barlow Twins is non-contrastive (does not require negative sampling) and symmetric (uses a single encoder model), making it lightweight and simple to use and modify. Unfortunately, we found the Barlow Twins loss function to be unstable at times during training, i.e., the gradient exploding. For this reason, we do much of our benchmarking using the VICReg framework. VICReg provides accuracy similar to, albeit slightly worse than, Barlow Twins and does use negative sampling. However, VICReg's training has remained stable throughout our experiments while having a computational footprint similar to that of Barlow Twins. The third model in our work in DINO, which is non-contrastive, is asymmetric and uses multi-cropping natively. DINO benefits from using a transformer-based encoder instead of a ResNet-based one. All three frameworks, Barlow Twins, VICReg, and DINO, were implemented by META AI and have similar codebases, making them easy to implement and compare directly. Although most contrastive learning frameworks seem to benefit from GCL, there are several that are likely incompatible. The contrastive learning framework, SMoG (Synchronous Momentum Grouping) and NNCLR (Nearest-Neighbor Contrastive Learning of visual Representations), both cluster images together during training and pair the current image with their nearest neighbor to form a positive pair [59, 77]. Likely, SMoG and NNCLR would not benefit from generative augmentations as the nearest neighbor image contains changes in visual features similar to the effects of generative augmentations. Unlike generative augmentations, both SMoG and NNCLR require additional computation during training to cluster images. The clustering used in these contrastive learning frameworks is also susceptible to issues relating to outliers and imbalanced data, issues that do not exist in the ImageNet dataset. The ReLICv2 contrastive learning framework may also not benefit from generative augmentations as it already uses a ML model to erase the background in images; however, more work is needed to confirm this. Although SMoG, NNCLR, and ReLICv2 are likely not compatible with generative augmentations, each has shown that using ML models to improve the formation of positive pairs is beneficial to downstream accuracy in contrastive learning applications.

4.2 Noise-Based Augmentations

GANs are particularly well-suited to GCL due to their non-iterative inference and the random noise vector native to the GAN architecture. Our first GCL implementation uses the ICGAN as the pretrained generative model. This model uses the BigGAN architecture as its backbone, which was one of the first generative models able to scale to the ImageNet dataset successfully. The ICGAN's architecture also includes a native encoder model, which can be used to map images to a single-dimension continuous feature map. The encoder backbone is a ResNet model and has a non-iterative inference. The ICGAN's encoder helps the model scale to larger datasets in an unsupervised manner. As most GANs struggle to scale to large datasets in an unsupervised manner, the ICGAN architecture is apt for sparsely labeled datasets or applications that require semi-supervised learning. Unfortunately, the ICGAN suffers from poor image quality, with an FID score of 15.6 on the ImageNet dataset; the supervised version of the ICGAN was 7.4 [2]. Since the ICGAN was introduced, no other GAN models that scale to large image datasets in an unsupervised manner have been introduced, making it SOTA in unsupervised GAN architectures. Once the image is fed into the encoder and outputs a feature map, a random noise vector is sampled and inputted into the ICGAN's generator model. The generator then outputs a generative image that has similar visual features to the inputted image.

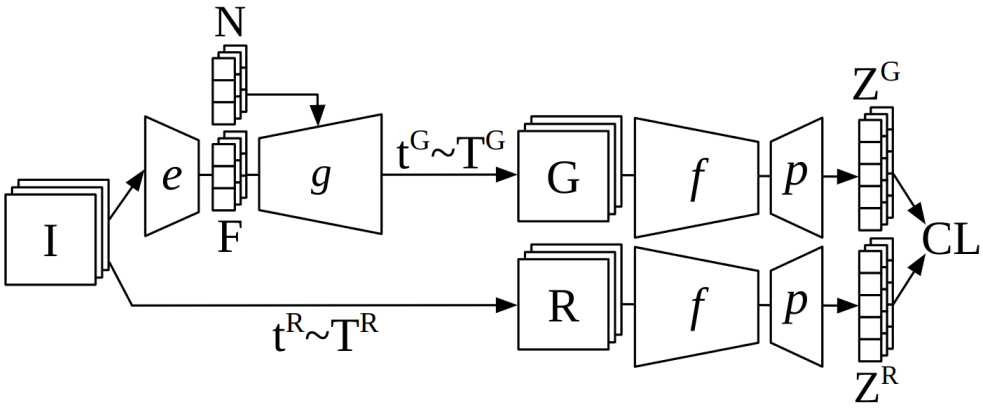

Figure 21: Architecture of noise-based augmentations in a contrastive learning framework.

As the ICGAN follows a GAN architecture, it can add variance to the visual features of an image through an inputted noise vector. The architecture of our noise-based generative augmentations within a contrastive learning framework can be seen in **Figure 21**. Given a batch of images **I**, two views are produced. The first view is fed into the feature extractor ***e***, which outputs the corresponding feature maps **F**. Noise vectors **N** are then sampled, paired with the feature maps **F**, and fed into the generator model ***g***, which outputs the generatively augmented images. A subset of augmentations $\mathbf{t}^{\mathbf{G}}$ is then sampled from a set of handcrafted augmentations $\mathbf{T}^{\mathbf{G}}$ and applied to the generatively augmented images, creating a batch of images **G**. A second subset of augmentations $\mathbf{t}^{\mathbf{R}}$ is then sampled from another set of handcrafted augmentations $\mathbf{T}^{\mathbf{R}}$ and applied to the second view, creating a batch of images **R**. The positive pairs (**G**, **R**) are then fed into the contrastive learning model's encoder ***f*** and projector ***p***, respectively. The outputted encodings ($\mathbf{Z}^{\mathbf{G}}$, $\mathbf{Z}^{\mathbf{R}}$) are then fed into the contrastive learning function.

Figure 22: Samples from Noise-Based Augmentations on the ImageNet dataset.

The inputted noise vector allows us to change visual features in the outputted image with each random sampling. Noise vectors that have similar values produce images that contain similar visual features. If the contrastive learning model is struggling to learn the similarities between the original image and the generatively augmented image, the use of GAN inversion techniques can be used to increase the similarity between the images. The GAN inversion process uses the inputted noise vector as the training parameter and the difference between the original and generated images as the loss function. This can be a

highly iterative process, where each iteration increases the similarity between the two images. As we want there to be some variance in visual features, it's important not to train using too many iterations of the GAN inversion, as the images' visual features will become too similar. For the ImageNet dataset, we found the GAN inversion was unneeded to boost downstream accuracy.

As with any type of data augmentations in ML, there is an optimal strength at which to apply them, resulting in better downstream accuracy. Although our noise-based augmentations are fast and straightforward to apply, it is difficult to control the strength of the augmentations themselves. The only inputs we are able to manipulate to adjust the strength of the augmentations are the noise vector and the input image. The effects of the noise vector on each generated image's visual features are not clear and would require training a second ML model to adjust the changes. The input image is also difficult to directly manipulate without compromising the image quality of the generated image. The two main weaknesses of our GAN-based augmentations are the lack of control over the strength of the augmentations and the low quality of the generated images. Other generative models, specifically image transformers and DMs, do not suffer from these flaws, as we demonstrate in the following sections.

4.3 Transformer-Based Augmentations

Transformers have become the standard backbone for the majority of new computer vision models. Particularly in generative modeling, image transformers are now being used instead of convolutional-based architectures, as they produce higher-quality images and scale well to complex data. However, image transformers, like other ARMs, suffer from an iterative inference time when generating an image. Unidirectional image transformers predict parts of an image in sequential order. Bidirectional image transformers can predict parts of an image in any order, making them ideal for in-painting and out-painting applications. Also, because bidirectional image transformers do not have to predict image tokens in any particular order, their inference time can be sped up significantly. This is done by first generating an entire image in a single inference and keeping the parts of the image with the highest

quality. The low-quality parts of the image are then resampled by the transformer, which again preserves the highest-quality image parts while resampling the low-quality parts. This process is repeated until the entire image is comprised of high-quality parts. The main parameter for this inference is the amount of image parts retained after each inference, which affects the inference speed as well as the image quality. Bidirectional image transformer models like MaskGIT and MAGVIT have been able to generate SOTA image quality in 8-12 iterations [27,63].

For our transformer-based GCL, we use the MaskGIT model, which uses a discrete 2D feature map. To change visual features in the encoded image, we mask parts of the feature map and use a bidirectional transformer to predict the missing values. Due to the incomplete information the transformer is receiving, the predicted missing values will not be identical to the original feature map. This discrepancy between the original feature map's values and the predicted feature maps results in the decoded image having visual features that are different from the inputted. Increasing the size of the mask will result in more changes in the visual features as the difference between feature maps will increase. We found that, in addition to increasing mask size, increasing the adjacency of masked values also increases the change in visual features. Randomly changing the discrete values produces a static image, while sampling the discrete values from a likelihood-based model, like a transformer, produces realistic visual features. Finding the optimal mask for each dataset is too computationally expensive as it requires training on many mask permutations. For our work, we experimented with two masking strategies: in-painting and out-painting.

Both our in-painting and out-painting generative augmentation implementations use a pretrained Masked Generative Image Transformer as the generative model (MaskGIT). The MaskGIT model provides a better image quality than the ICGAN, achieving an FID of 4.02 on the ImageNet dataset. The MaskGIT model does have a slower inference time than the ICGAN. This is due to MaskGIT's bidirectional transformer, which generates an image over 12 iterations, where each iteration improves the image quality. The MaskGIT model scales to 256x256 image resolution but trains in a supervised manner. Because MaskGIT does not use a GAN architecture, it does not have an inputted noise vector to add

visual variances to an image. Instead, we use the MaskGIT's innate in-painting and out-painting ability to add visual variances to an image.

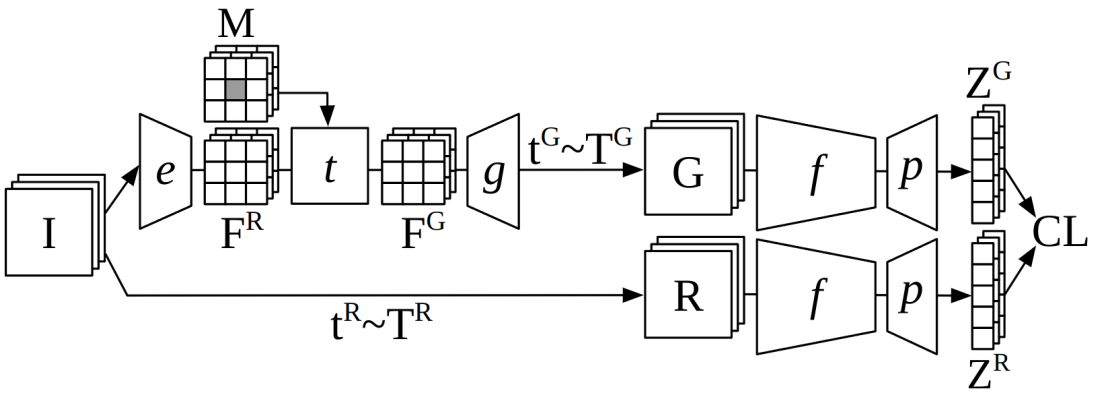

Figure 23: Architecture of in-painting augmentations in a contrastive learning framework.

Our in-painting augmentation architecture, seen in **Figure 23**, is identical to our noise-based architecture, except for how the generatively augmented images **G** are synthesized. Instead of mapping the first view of the images **I** into a feature map, we instead map them into 16x16 discrete vector quantizations $\mathbf{F^R}$ using MaskGIT's vector quantized encoder ***e***. All values not on the outer edge of the 16x16 vector quantization are masked with a mask **M**. Then, MaskGIT's bidirectional transformer ***t*** predicts all the masked values. The predicted 16x16 vector quantization $\mathbf{F^G}$ is then decoded back into the generatively augmented image using MaskGIT's vector quantized decoder ***g***.

Figure 24: Samples from In-Painting Augmentations on the ImageNet dataset.

Like our in-painting augmentation strategy, our out-painting augmentation strategy uses the pretrained MaskGIT model to predict values masked in the vector quantization $\mathbf{F^R}$. However, instead of predicting masked values on the inner portion of the vector quantization, the out-painting augmentation strategy instead predicts values on the outer portion. To achieve this, we make two changes to the in-painting architecture, which can be seen in **Figure 25**.

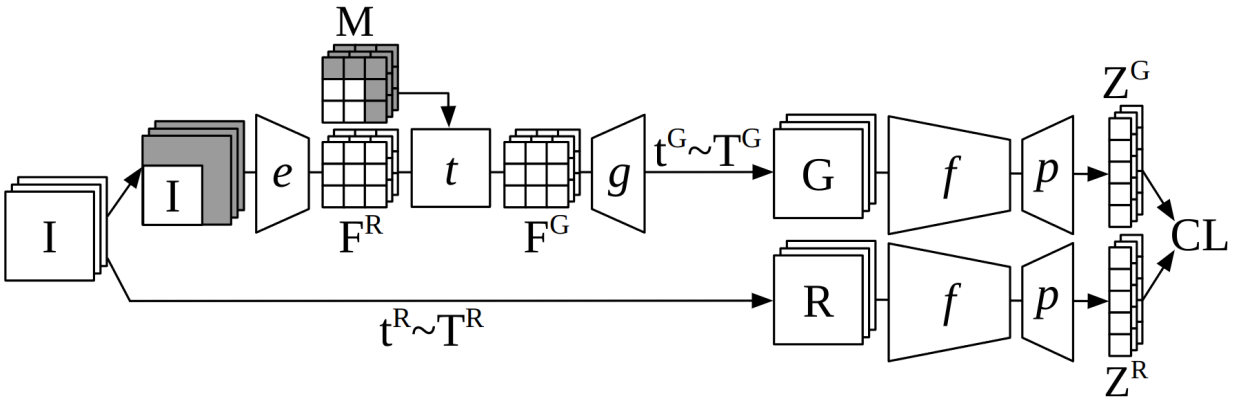

Figure 25: Architecture of out-painting augmentations in a contrastive learning framework.

Before inputting the first view of images **I** into ***e***, we decrease the image's resolution to 196x196 and overlay it onto a blank 256x256 image in nine different positions, which can be seen in **Figure 27**. The other change between the in-painting and out-painting augmentation implementations is the mask **M**. The masked areas in our out-painting implementation cover the parts of the image that are blank, which depends on the positioning of the overlayed image. As the out-painting augmentations augment the backgrounds of images and the in-painting augmentations augment the foreground of images, they allow us to see the effects of augmenting visual features in the foreground vs the background in regards to downstream accuracy.

Figure 26: Samples from In-Painting Augmentations on the ImageNet dataset

.

Figure 27: Overlay positioned in the upper left, upper center, upper right, center left, center, center right, lower left, lower center, and lower right of the blank image, followed by the non-overlayed view. The first row contains the masked input, and the second row contains the generative images produced by our out-painting augmentation.

4.4 Diffusion-Based Augmentations

Because DMs have the slowest inference speed compared to other generative models, they are difficult to use in a GCL setting. At the root of this problem is the high number of iterations during inference. Stability AI has introduced a series of SDXL (Stable Diffusion Extra Large) Turbo models, which are DMs that can produce photorealistic images in 1-4 iterations at resolutions of 256x256 and higher [64]. This fast inference speed is achieved through knowledge distillation, where a U–Net-based DM is first trained on a large dataset of captioned text. Next, a second identical DM is then trained as a student model using the pretrained DM as the teacher model. During training, the number of iterations the student model uses is set to 4, and an adversarial loss is used. The use of the adversarial loss allows the

student model to train similarly to a GAN, i.e., fast, non-iterative inference. The use of the teacher model allows the student model to produce images at the high-quality DMs for which they are known. This hybrid between DMs and GANs is ideal for GCL. The SDXL Turbo model can generate images from solely natural language prompts as well as image and natural language prompts. The latter of these allows us to augment existing images with natural language. However, finding the optimal natural language input for these generative augmentations is computationally expensive due to the high number of possible permutations in natural language.

For our work, we use the pretrained captioning model BLIP (Bootstrapping Language-Image Pre-training) to create natural language captions for each image in the ImageNet dataset. This is similar to how large text-to-video models like OpenAI's Sora use captioning models to add artificial text inputs to videos before training the generative model [62]. The text prompts created by BLIP are single sentences that typically describe the main object as well as the background (see **Figure 28**).

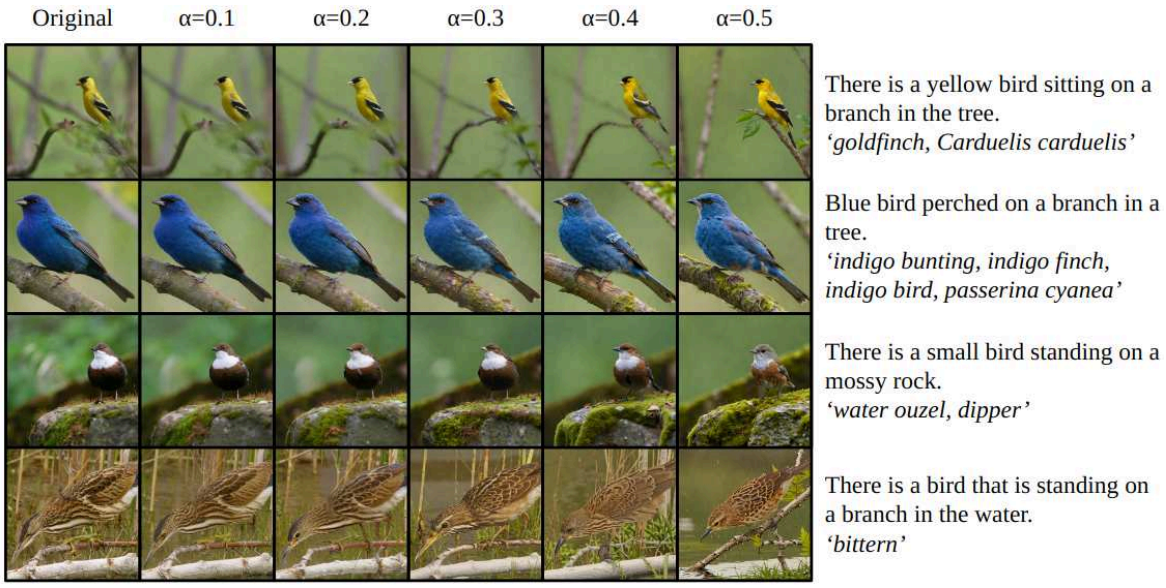

Figure 28: Generatively augmented images of birds from different labels on the ImageNet100 dataset. The first column contains the original image, followed by generatively augmented images from the SDXL Turbo model using increasing strengths. On the right are each image's generated captions, followed by the label of the image.

There are more highly descriptive captioning models that could also be used, but we found these LLMs to be more computationally expensive and will leave investigating the effects of more detailed captions to future research. Once the natural language prompts are extracted for each image in the dataset, we input them along with their corresponding image into the SDXL Turbo generator. In addition to the image and natural language inputs, the SDXL Turbo model also requires a strength input **α**. This strength metric controls the amount of noise added to the input image before the denoising process is used to generate an image. An **α** of zero would produce an image identical to the input image as no denoising is applied. An **α** of one would produce an image with no similar visual features to the input image as the input image is completely replaced with static noise. Similar to our masking augmentations, **α** must be set to a value strong enough to create variations between the images that cause our contrastive learning model to learn. However, if these visual variations are too strong, the contrastive learning model will struggle to find similar visual features between the positive pair, resulting in poor downstream accuracy. The number of iterations used by the SDXL Turbo model must be greater or equal to one when multiplied by **α**, e.g., an **α** of 0.5 must have at least two iterations, and an **α** of 0.1 must have at least ten iterations.

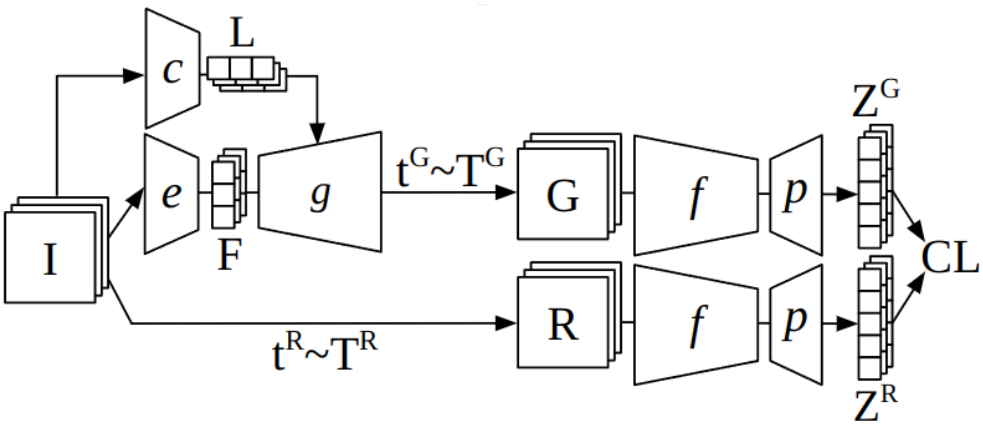

Figure 29: Architecture of diffusion-based augmentations in a contrastive learning framework.

Our diffusion-based augmentation architecture, seen in **Figure 29**, is identical to our noise-based architecture, except for the inputs to **G**. Instead of generating a random noise vector and inputting it into

the model, we instead take the original image and input it into a captioning model $\boldsymbol{c}$. The outputted natural language prompts $\mathbf{L}$ are then inputted into the generator model $\boldsymbol{g}$. The visual variances synthesized by our diffusion-based augmentations are based on the natural language inputs being imperfect and the noise being added to the input during the diffusion process. Like our masking augmentations, the generator is recreating an image with an incomplete amount of information, resulting in an image that has differing visual features that are also realistic. The natural language input could likely be excluded from our diffusion-based augmentation architecture, as the noise added during the diffusion process is likely sufficient enough to add visual variances. However, to our knowledge, large diffusion models that also have fast inference speeds, like SDXL Turbo, all use images paired with natural language, as these have been shown to help diffusion models scale to large datasets and improve image quality. We leave training a fast inference DM with no natural language inputs to future work, as this would require significant computational resources. In our methodology, we have outlined the high-level components needed for our overall GCL framework as well as the components for the GCL architecture using the three different types of large generative models: GANs, image transformers, and DMs. We have also given a detailed explanation of how we have implemented GCL for each of the generative model types.

5. Results

To evaluate our GCL architectures, we first need to optimize the different training parameters used by our generative augmentation strategies. We do this using several ablations to find what handcrafted augmentations pair well with generative augmentations, how often we should apply generative augmentations (mixing), and what strength we should apply the generative augmentations. As generative augmentations are computationally expensive, we examine the effects of using more than one generative image per training image on downstream accuracy. In addition to applying standard self-supervised learning protocol to evaluate our generative augmentations in a contrastive learning setting, we also apply them to transfer learning applications. We follow this with a cost-benefit analysis. In addition to our analyses of photograph-based datasets like ImageNet, we also apply our GCL framework to the domain of satellite imagery. Finally, we summarize the best accuracies for each of our generative augmentation strategies in a final analysis.

5.1 Pairing Ablation

Handcrafted augmentations in computer vision can be grouped into two categories: geometry and color. Geometry augmentations change pixel locations in an image, and color augmentations change the values of pixels. As generative augmentations change visual features' positioning and color, they can be considered both geometry and color augmentations. For this reason, applying additional handcrafted augmentations like cropping and color jitter may be redundant when applied after the generative augmentations. To study the effects of pairing generative and handcrafted augmentations, we train a Variance Invariance Covariance Regularization (VICReg) contrastive learning model on our noise-based (ICGAN) and in-painting (MaskGIT) augmentations paired with each of the six handcrafted augmentations typically found in contrastive learning: cropping, flipping, color jitter, Gaussian blur, solarization, and grayscale. To establish a baseline, we train a contrastive learning model using generative

augmentations without any handcrafted augmentations. Each contrastive learning model is trained on the ImageNet dataset for 100 epochs with a batch size of 800 using the Layer-wise Adaptive Rate Scaling (LARS) optimizer and a ResNet50 backbone. We evaluate each model by applying standard self-supervised linear, i.e., linear and semi-supervised evaluation.

Table 3: Pairing Augmentations Ablatio's Top-1 and Top-5 accuracies obtained with our generative augmentations. The linear classification uses frozen representations on the full ImageNet dataset. The semi-supervised classification uses fine-tuned representations on 1% and 10% of the ImageNet dataset.

Augmentations	Linear		Semi-Supervised			
	Top-1	Top-5	Top-1		Top-5	
			1%	10%	1%	10%
ICGAN	54.4	78.4	32.9	49.3	60.5	75.1
ICGAN + Crop	62.8	84.0	40.6	58.3	68.3	82.0
ICGAN + Flip	57.4	80.5	35.7	52.6	63.8	77.8
ICGAN + Solarization	55.8	79.7	32.9	50.5	61.1	76.2
ICGAN + Color Jitter	55.7	79.3	32.5	50.1	60.5	76.0
ICGAN + Blur	54.2	78.3	32.1	49.3	60.0	75.2
ICGAN + Greyscale	52.7	77.3	30.6	48.4	58.5	74.6
MaskGIT	67.8	87.7	48.3	63.0	74.3	85.1
MaskGIT + Crop	**67.9**	87.7	47.9	62.9	73.9	85.2
MaskGIT + Flip	67.8	**87.8**	47.8	62.8	73.9	85.0
MaskGIT + Solarization	67.6	87.7	48.3	62.9	**74.4**	**85.3**
MaskGIT + Color Jitter	67.5	87.6	47.9	62.8	73.8	85.0
MaskGIT + Blur	67.8	**87.8**	48.0	62.8	74.2	85.2
MaskGIT + Greyscale	67.6	87.6	**48.6**	**63.1**	**74.4**	85.1

Based on the linear top-1 results, pairing generative augmentations with cropping outperformed pairings with any of the other handcrafted augmentations. The accuracy of the ICGAN augmentations increased when paired with cropping, flipping, solarization, and color jitter but decreased when paired with blurring and greyscale. Interestingly, the MaskGIT augmentations only increased in accuracy when paired with cropping and did not benefit from being paired with any of the other handcrafted augmentations. The use of the flipping augmentation resulted in approximately the same downstream

accuracy as the baseline model, while all the color augmentations decreased the accuracy. These results indicate that geometry augmentations pair well with generative augmentations and that color augmentations do not. The models trained with MaskGIT's augmentations significantly outperformed those trained with the ICGAN's augmentations, increasing accuracy anywhere from 5.1% to 13.4%, showing a clear benefit to using our transformer-based augmentation implementation compared to our noise-based augmentation implementation.

5.2 Mixing Ablation

In contrastive learning, handcrafted augmentations are randomly sampled, where the sampling threshold is represented by γ. The sampling threshold for each handcrafted augmentation varies; for example, on the ImageNet dataset, the sampling rate for solarization is typically 0.2, while flipping's sampling rate is typically 0.5. For this reason, we apply our noise-based, in-painting, and out-painting augmentations using varying sampling rates of 0.0 (baseline), 0.25, 0.5, 0.75, and 1.0. When a generative augmentation is applied to an image, the cropping augmentation is also applied, as we found this increases downstream accuracy in our pairing ablation. When a generative augmentation is not applied, each of the six handcrafted augmentations is instead applied following the same augmentation strategy used in [46, 54]. When an out-painting augmentation is sampled, one of the nine different views is randomly sampled. As a baseline, we train a single model where the sampling rate for the generative augmentations is set to zero. The training of these models is identical to the training schema used in our pairing ablation. Also included in **Table 4** are the results from the GenRep and COP-GEN generative augmentation strategies [43, 55].

Table 4: Mixing Augmentation Ablation, where γ represents the sampling rate of our generative augmentation strategies. The linear classification uses frozen representations on the ImageNet dataset. The semi-supervised classification uses fine-tuned representations on 1% and 10% of the ImageNet dataset.

Approach	γ	Linear		Semi-Supervised			
		Top-1	Top-5	Top-1		Top-5	
				1%	10%	1%	10%
Non-Generative	0%	65.8	86.6	42.9	60.7	70.2	84.1
Noise-Based	25%	66.3	87.0	43.9	61.1	71.4	84.3
Noise-Based	50%	65.9	86.7	42.9	60.5	70.4	83.6
Noise-Based	75%	65.3	86.1	41.7	59.6	69.5	82.9
Noise-Based	100%	64.0	85.0	40.7	58.9	68.3	82.5
In-Painting	25%	67.9	88.0	47.0	62.9	73.7	85.3
In-Painting	50%	**68.7**	**88.3**	48.3	**63.5**	74.3	**85.7**
In-Painting	75%	68.5	88.2	**48.5**	63.3	**74.5**	85.5
In-Painting	100%	67.9	87.7	47.9	62.9	73.9	85.2
Out-Painting	25%	66.8	87.2	44.5	61.7	71.5	84.6
Out-Painting	50%	66.8	87.1	45.1	61.9	71.8	84.6
Out-Painting	75%	66.7	87.1	45.1	61.8	71.7	84.7
Out-Painting	100%	66.4	86.8	45.2	61.7	71.6	84.6
GenRep [17]	100%	51.2	75.0	-	-	-	-
COP-Gen [19]	100%	53.3	77.2	-	-	-	-

The linear top-1 results show that a sampling rate of 0.25 provides the best results for noise-based augmentations, while a sampling rate of 0.5 provides the best results for our in-painting augmentations. The accuracy for the sampling rates of 0.25 and 0.5 are identical for the out-painting augmentation. Our in-painting augmentation strategy provided the best results, outperforming both the noise-based and out-painting strategies. Our in-painting strategy also outperformed the GenRep and COP-GEN augmentation strategies by at least 15%. Our out-painting augmentation strategy underperformed our in-painting strategy significantly, showing that generatively augmenting the visual features in the center

of an image outperforms augmenting the visual features surrounding the center of the image, likely due to the cropping bias of photographs.

5.3 Diffusion Strength Ablation

To benchmark the effects of our diffusion-based augmentations, we train a contrastive learning model on the generatively augmented images of the ImageNet100 subset of the ImageNet dataset. We used DINO as our contrastive learning framework with a ResNet50 backbone, which we trained for 100 epochs. We then train a single linear layer using standard self-supervised learning protocol and evaluate the downstream accuracy. Because our diffusion-based augmentations can train at different strengths, we trained a separate contrastive learning model at the strengths of 0.0, 0.1, 0.2, 0.3, 0.4, and 0.5. Strength 0.0 is the baseline, as no changes to the image are being made by the diffusion-based augmentations. For our mixing augmentations, we apply the flipping and cropping augmentations as these provided the best results from our mixing ablation, albeit using our in-painting augmentation. We used a sampling rate of 0.5 as this provided the best accuracy in our pairing ablation.

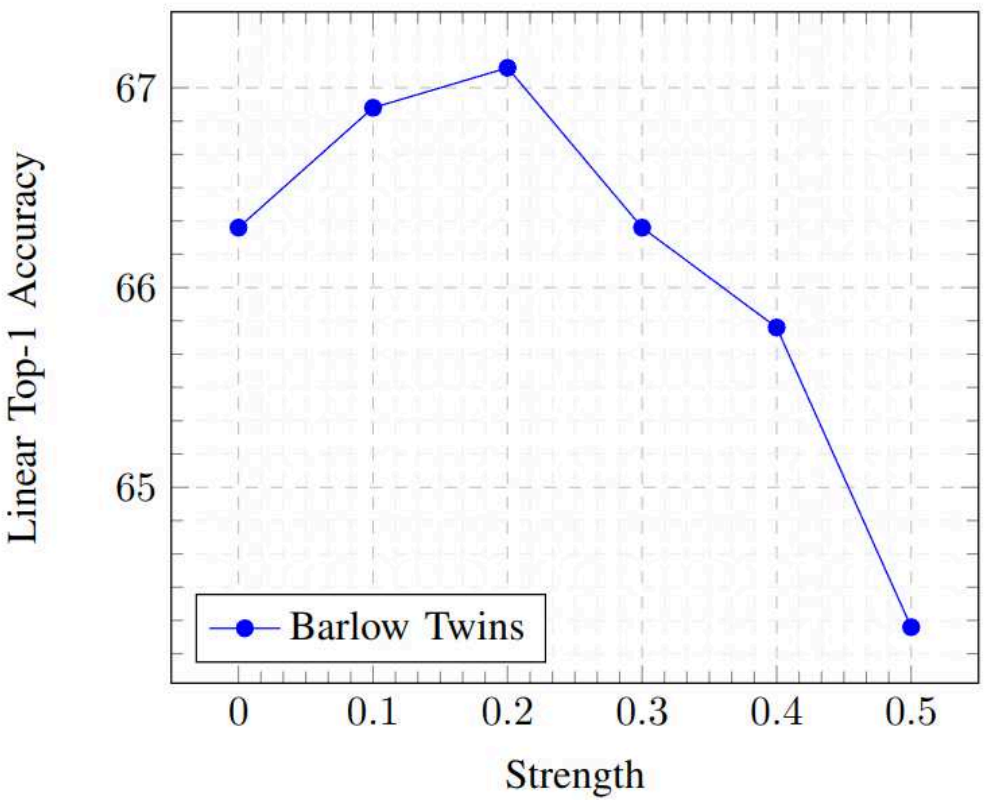

Figure 30: Diffusion-Based Augmentation Strength Ablation at strengths 0.0, 0.1, 0.2, 0.3, 0.4, and 0.5. The linear classification uses frozen representations on the ImageNet100 dataset.

Our results show that the accuracy of using diffusion-based augmentations increased when the strength was set to 0.1 and 0.2, while a strength of 0.3 had an accuracy approximately the same as our baseline. When a strength of 0.4 or higher was used, the accuracy of our model dropped below the baseline. The increase in accuracy from our best-performing model, which used a strength of 0.2, was only 0.8 increase over the baseline, a relatively smaller increase compared to our other generative augmentations. Adjusting training parameters, such as the sampling rate, may increase accuracy further; however, further work is needed. Likely, the reason for this relatively smaller increase in accuracy is the diffusion-based augmentations are losing too much information in the natural language inputs, which are generated by the BLIP captioning model. Applying the Adversarial Diffusion Distillation methodology to a DM that does not use natural language inputs could potentially remedy this but would require significant computational resources to train. The SDXL-Turbo model used for our diffusion-based models

is not trained on the ImageNet dataset, showing that it is able to generalize well to unseen datasets like ImageNet. Also, the SDXL-Turbo model shows that diffusion-based augmentations can be performed at the same speed as generative augmentations that use GANs or image transformers through the use of transfer learning methods like Adversarial Diffusion Distillation that hybridizes the DM and GAN architectures.

5.4 Generative Views Ablation

Multi-crop is an augmentation strategy introduced in [56] where multiple augmented images are paired with a single image. By using multiple augmented views instead of the single view typically used in contrastive learning, the authors were able to increase the downstream accuracy significantly. Essentially, multi-crop increases the number of images in positive pairs. Similarly, we implement a multi-view augmentation strategy to improve our out-painting augmentation strategy. This implementation pairs each image with one to three samples from our out-painting augmentation strategy. As we increase the number of images used per batch, we adjust the batch size accordingly, using a batch of 768 for a single generatively augmented view (two images), a batch size of 512 for two generatively augmented views (three images), and a batch size of 384 for three generatively augmented views (four images). We also train using zero generative views, meaning the same handcrafted augmentation strategy from our previous ablations is applied to views of the original image (single view). The out-painting augmentations are paired with the cropping augmentation and applied using a sampling rate of one, along with other parameters typically used in contrastive learning. In addition to using the VICReg contrastive learning framework, we also use the Barlow Twins, SwaV, BYOL, and SimCLR frameworks. To reduce training times, we train using the ImageNet100 dataset, a subset of the ImageNet dataset. All other training parameters are identical to our previous experiments.

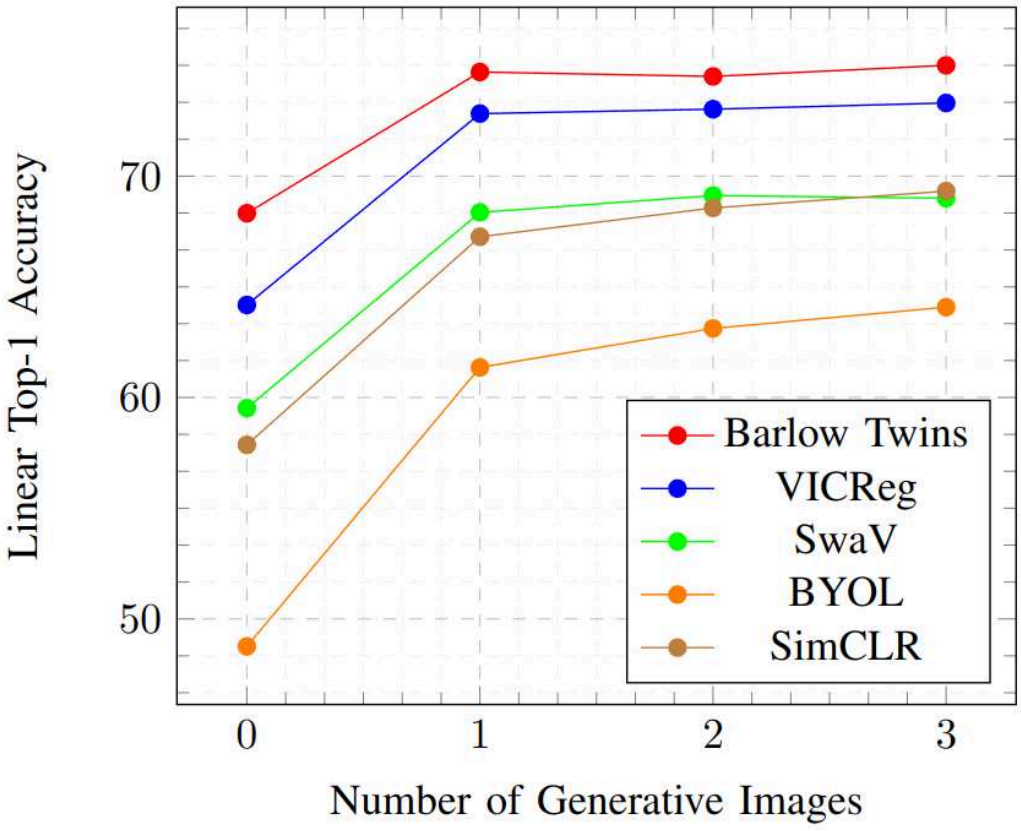

Figure 31: Generative Views Ablation, where the number of generated images per real image increases from zero to three, and the linear top-1 accuracy is measured. Five contrastive learning frameworks, Barlow Twins, VICReg, SwaV, BYOL, and SimCLR, are used. The linear classification uses frozen representations on the ImageNet100 dataset.

Our results show that using multiple generated views does increase linear top-1 accuracy. However, this increase suffers from diminishing returns. For the best-performing model, Barlow Twins, the accuracy increased by 0.3 percent when the number of views increased from one to three. Compared to the results from multi-crop, which increased linear top-1 accuracy by over 3% [56], the increase in accuracy from using multiple generatively augmented views on the Barlow Twins and VICReg models is significantly less. Each of the five contrastive learning framework's accuracy increased significantly when out-painting augmentations were used instead of handcrafted augmentations, indicating any contrastive learning framework benefits from the use of generative augmentations.

5.5 Transfer Learning Analysis

One of the benefits of using self-supervised methods, like contrastive learning, compared to supervised methods is that it is less dependent on labeled data and performs well in a semi-supervised setting. However, unlike ICGAN, MaskGIT is trained in a supervised manner, as are the majority of current DMs. Most of the state-of-the-art generative models on the ImageNet dataset have been trained in a supervised manner [2, 16, 30, 60, 3, 6, 61]. This may lead to bias when applying transformer-based augmentations to semi-supervised benchmarks, as the generative model has been exposed to 100% of the data instead of the 1% and 10% used in the supervised benchmarks. To remove any bias, we apply our pretrained contrastive learning models to the Places365, VOC07, and COCO datasets following standard self-supervised learning linear evaluation protocol. We use the best-performing generative augmentations from our mixing ablation as pretrained contrastive learning models, i.e., our noise-based augmentations paired with cropping and a sampling rate of 0.25 and in-painting and out-painting augmentations, each paired with cropping and a sampling rate of 0.5. Additionally, we use the contrastive learning model trained using only handcrafted augmentations as a baseline.

Table 5: Transfer Learning Analysis where the best performing contrastive learning models for each augmentation strategy are applied to the Places365, VOC07, and COCO datasets, and the linear top-1 accuracy is measured. The linear classification uses frozen representations on each respective dataset.

Augmentations	Places365	VOC07	COCO
Hand-Crafted	47.6	83.8	71.4
Noise-Based	47.9	84.9	72.4
Out-Painting	47.6	84.4	71.9
In-Painting	**49.1**	**85.2**	**72.7**

Of the three pretrained contrastive learning models, the model trained using our in-painting augmentations performs the best on each of the three datasets. The next best-performing model was

trained using noise-based augmentations, followed by the models trained using out-painting and handcrafted augmentations, respectively.

5.6 Cost-Benefit Analysis

Applying generative augmentations incurs a significantly greater computational cost compared to handcrafted augmentations. State-of-the-art generative models require substantial computational resources and often have highly iterative inference times, making large-scale applications difficult. However, we posit certain generative models, like the ICGAN and MaskGIT, are particularly well-suited to augmenting large numbers of images efficiently. To show this, we profiled the performance of our noise-based, in-painting, and out-painting augmentations as well as the cropping, color jitter, and blurring augmentations by measuring each augmentation's linear top-1 accuracy, FID, Structural Similarity Index (SSIM), inference time (milliseconds), and number of parameters used (see **Table 6**). For the generative augmentations' linear top-1 accuracy, we use the same results from our pairing ablation. The linear top-1 accuracy of the crop, color jitter, and Gaussian blur augmentations are taken from [41] indicated by †. For each image in ImageNet's training dataset, we calculate the FID and SSIM between the original and augmented image and aggregate the results. To calculate inference time, we average the time taken to augment 1,000 images from the ImageNet dataset using a batch size of one. We also measure the number of parameters the ICGAN and MaskGIT models use when augmenting an image.

Table 6: Cost-Benefit Analysis's linear Top-1 accuracy, FID, SSIM, inference time measured in milliseconds, and number of parameters for the noise-based (ICGAN), in-painting (MaskGIT), out-painting (MaskGIT), crop, color jitter, and blur augmentations on the ImageNet dataset.

	Linear Top-1	FID	SSIM	Inference	Param.
Noise-Based	54.4	21.13	0.21	19.82ms	116M
In-Painting	67.8	3.70	0.32	98.93ms	227M
Out-Painting	66.5	7.29	0.27	100.91ms	227M
Crop	33.1†	2.48	0.27	0.58ms	N/A
Color Jitter	18.8†	0.23	0.87	3.70ms	N/A
Gaussian Blur	2.6†	3.72	0.88	0.64ms	N/A

Due to the use of MaskGIT instead of ICGAN, the FID produced by the in-painting augmented images is significantly improved compared to the noise-based augmented images. Interestingly, the FID of our in-painting augmentations is also higher than the FID of the handcrafted augmentation Gaussian blur and is comparable to the FID of the other handcrafted augmentations' FIDs. This higher FID likely gives our transformer-based augmentations a significant bias over noise-based augmentations. Training an ICGAN with the StyleGAN-XL backbone instead of the BigGAN backbone would likely help alleviate this disparity, but it is outside the scope of our work. The time it takes to generate the augmentations with MaskGIT is 5x longer than ICGAN and 26x longer than the color jitter augmentation. Creating a single view with our in-painting augmentation for each training image in the ImageNet dataset takes approximately 25% of the time to train a contrastive learning model for 100 epochs using 4 NVIDIA 3090 GPUs. For this reason, we recommend generating a single generatively augmented image and reusing it for each epoch during training, as this reduces computation and does not affect downstream accuracy significantly. This differs from handcrafted augmentations, which create a unique augmented image for every training epoch. The number of parameters MaskGIT uses is approximately double ICGAN's, which will also reduce augmentation speed as it increases the GPU consumption, thus requiring a reduction in batch size. In these results, there is a clear trade-off between the amount of computation and downstream accuracy between our noise-based and transformer-based augmentations.

5.7 Satellite Image Study

In addition to applying our in-painting augmentations to photograph datasets like ImageNet, we also apply them to the satellite image domain, specifically the Wildfire Prediction Dataset [79]. The Wildfire Prediction Dataset consists of 22,710 images containing where wildfires will occur and 20,140 images where no wildfires will occur. The dataset uses a 70%/15%/15% training, testing, and validation split. The satellite images themselves use only the three RGB bands and have a resolution of 350x350 but are resized to a 128x128 resolution due to computational restraints. To train the MaskGIT model, we first train a VQGAN on the satellite images. We train this model for 300 epochs using a batch size of 32. The codebook used for the VQGAN represents patches of 8x8 pixels that represent an image in a 16x16 discrete representation, where the discrete values range from 0-1023. Once the VQGAN is finished training, we then train a bidirectional transformer on the encoded images. Unlike the transformer model used in the original MaskGIT model, ours does not use label inputs and is trained in a completely unsupervised manner. Other than this, the bidirectional transformers from our work and MaskGIT are identical. The bidirectional transformer is trained for approximately 3,000 epochs using a batch size of 48. Once training is complete we then generate the inpainted augmentations for each satellite image. We do this using a uniformly random mask where the probability of a pixel being masked is 0.25, 0.5, and 0.75. We do this instead of masking the center of the images as the wildfire pixels can occur at any position in satellite images, unlike photographs where the focus of the image is typically located in the center. We apply each of the masking percentages of 0.0, 0.25, 0.5, and 0.75 at mixing rates of 0%, 25%, 50%, 75%, and 100%, where 0% is the baseline as it uses no generatively augmented images (**Figure 32**).

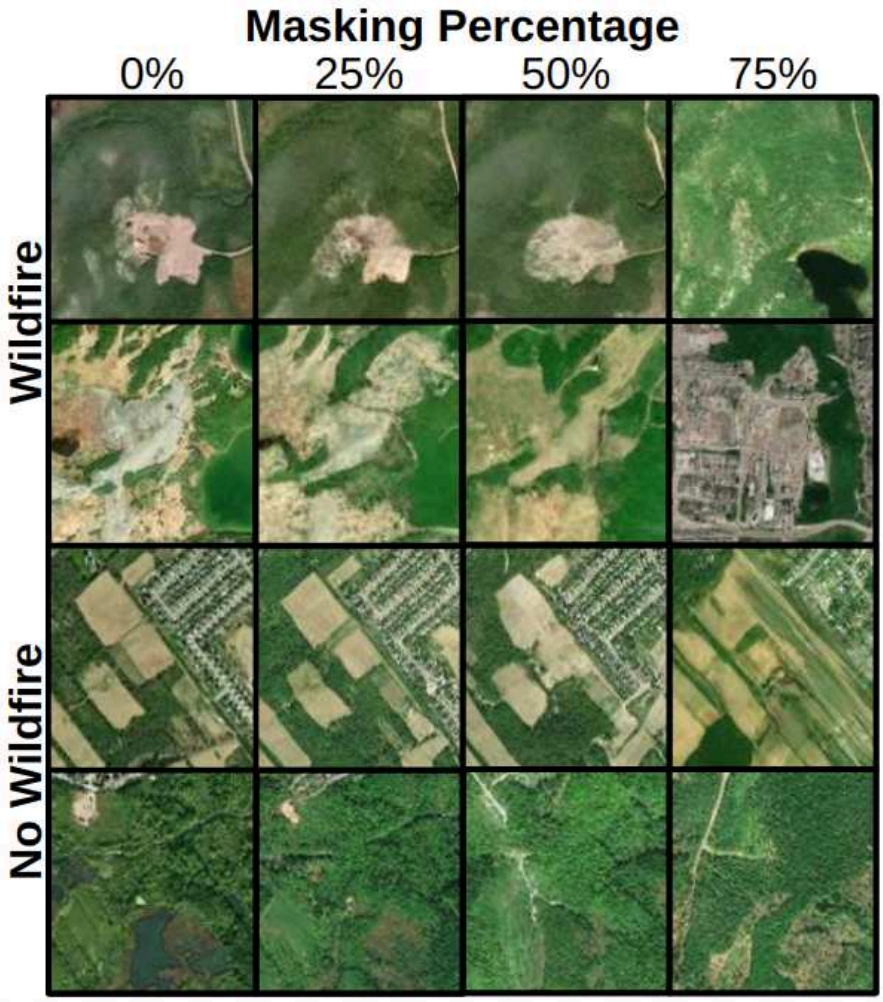

Figure 32: Generatively augmented satellite image samples from the Wildfire Prediction Dataset. The images were taken from both the wildfire and no wildfire categories and augmented using our inpainting augmentation strategy, where the number of masked pixels is 0%, 25%, 50%, and 75%, respectively.

We use the VICReg model as our contrastive learning framework and a ResNet50 model as our backbone, which is trained for 100 epochs. We then evaluate the trained contrastive learning model using top-1 linear accuracy. Our results (**Figure 33**) show that using a masking probability of 0.25 performs the best when used with a mixing percentage of 75% and gives an accuracy of 98.37%; in comparison, the baseline is 97.96%, and a fully supervised model is 98.68%. When we train the contrastive learning model for 1,000 epochs using our GCL implementation, the downstream accuracy increases to 99.22%. When we increase the training epochs to 1,000 for the fully supervised method, the accuracy increases to

99.20%. This is the first time in our research we have been able to train our contrastive learning for the full 1,000 epochs due to the reduced size of the Wildfire Prediction Dataset compared to the ImageNet dataset. When the number of epochs is increased from 100 to 1,000, our GCL outperforms the supervised benchmark using the same ResNet50 backbone.

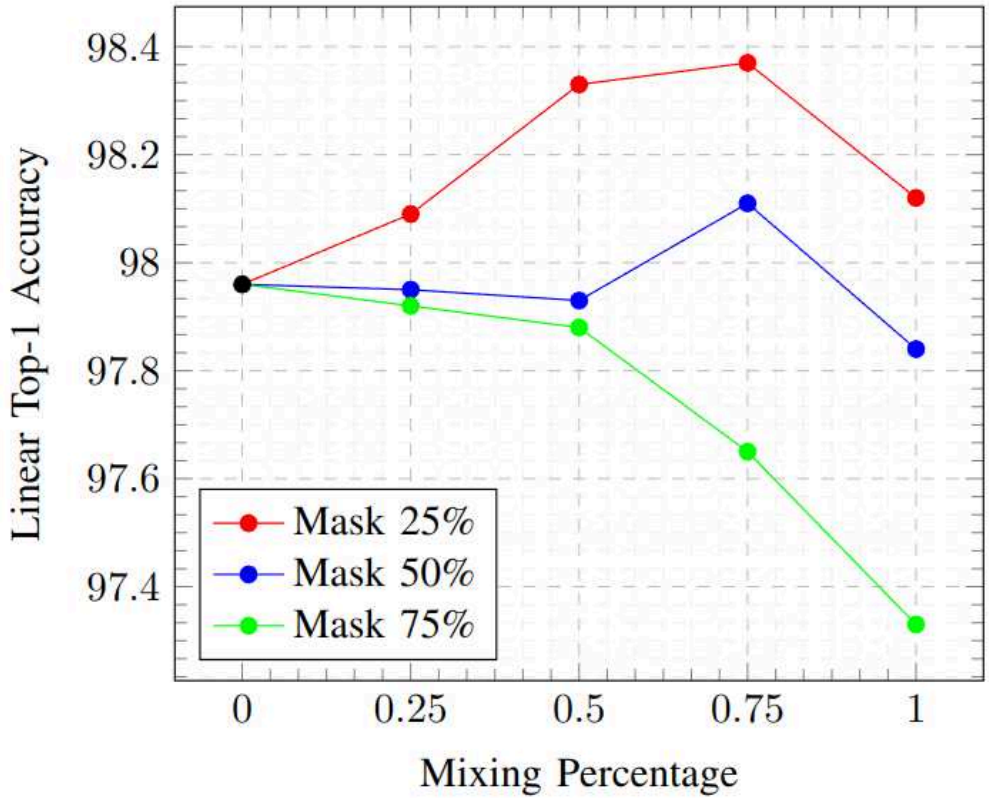

Figure 33: Linear Top-1 accuracy of in-painting generative augmentations on Wildfire Prediction Dataset using 100 epochs.

The results from using satellite imagery differ from using photographs in that using a smaller masking probability and a higher mixing percentage benefits downstream accuracy significantly. This is likely because the signals for wildfire prediction are weaker than ImageNet's object detection, and stronger generative augmentation results in too much of this information being lost. In our satellite image results, we are able to train a contrastive learning model for the full 1,000 epochs typically used in contrastive learning benchmarks due to the smaller size of the Wildfire Prediction Dataset as well as the decreased

resolution of the images. The accuracy of this self-supervised model is able to outperform the supervised model trained using similar training parameters. These results show that generative augmentations can be applied to satellite images as well as photographs.

5.8 Final Analysis

The best linear accuracies achieved by each of our generative augmentation strategies on the ImageNet dataset can be seen in **Table 7.** Our best-performing generative augmentation strategy was the transformer-based in-painting strategy, followed by our diffusion-based, noise-based, and out-painting strategies. For a baseline, we use a contrastive learning framework trained using traditional handcrafted augmentations. Each generative augmentation strategy outperformed the baseline while not adding any additional computation while training the contrastive learning model. These increases in accuracy are comparable to the increases in accuracy achieved through the development of other contrastive learning and augmentation research [41, 57, 42, 46, 54, 58, 56, 59]. Our results from the Wildfire Prediction Dataset indicate that using our GCL framework when scaled to 1,000 epochs, can outperform equivalent supervised learning methods.

Table 7: Best accuracy achieved by each generative augmentation strategy on the ImageNet dataset.

Augmentation	Traditional	Noise	Out-Painting	In-Painting	Diffusion
Linear Top-1	65.8	66.8	66.3	**68.7**	67.1

6. Conclusion

In our work, we have presented GCL, a framework that can be used to improve the augmentations used in contrastive learning through the use of generative models. GCL is comprised of five separate components: the encoder, feature map augmenter, decoder, handcrafted augmenter, and contrastive learning model. The encoder is used to map a real image to a generative model's feature map space (**PF1**). The feature augmenter is able to take an image's feature map and augment it in a way that causes variations in low-level visual features while preserving important high-level visual features (**PF2**). A decoder is then used to map the augmented feature map back to a realistic synthetic image that can be paired with the original image to form a positive pair (**PF3**). We then apply further handcrafted augmentations to the positive pair, providing further optimization to our proposed generative augmentation strategy (**PF4**). Finally, we train an encoder model using contrastive learning that can map images to representations that can then be applied to downstream tasks (**PF5**). As there are three major types of modern generative models that can be used at scale: GANs, image transformers, and DMs, our work develops methodologies that can be applied to each of them.

For GANs, we have implemented noise-based augmentations, which utilize the noise vector native to the GAN architecture. Our noise-based augmentations are the fastest and most efficient of our generative augmentations due to the non-iterative inference time of GANs and the relatively small computational footprint of the ICGAN. However, the downstream accuracy of our noise-based augmentations underperformed our transformer-based augmentations as well as our DM-based augmentations. This is likely due to the ICGAN's comparatively lower image quality and the lack of a direct method for controlling the strength of our noise-based augmentations.

To utilize image transformers in our GCL framework, we make use of their innate in-painting and out-painting abilities. We found that our in-painting augmentations perform the best of any of the generative augmentations we implemented in our work. Although our out-painting augmentations provided lower boosts to downstream accuracy, they demonstrated that using multiple generative

augmentations per training image does not significantly boost downstream accuracy, allowing for a crucial computation efficiency when implementing GCL. As the vast majority of modern generative models are trained in a supervised manner, so was the pretrained MaskGIT model we used in our transformer-based augmentation. However, we found that when applying the contrastive learning model trained with our transformer-based augmentations to transfer learning tasks, which the MaskGIT model had not been exposed to, it still increased downstream accuracy. In addition, we trained a completely unsupervised version of the MaskGIT model on a smaller, more computationally friendly satellite image dataset and found that it increased downstream accuracy beyond supervised learning benchmarks.

For our third and final implementation of GCL, we implemented DM-based augmentations. Although DM's inference is the most highly iterative of generative models, we found that using the SDXL Turbo model significantly cut down this inference speed. Unlike our noise-based and transformer-based augmentations, our DM-based augmentation makes use of a generative model not trained on the ImageNet dataset. Instead, the SDXL Turbo model was trained on a much larger real-world dataset similar to other foundational models. Despite not being trained on the ImageNet dataset directly, the SDXL Turbo model still provided photo-realistic augmentations and demonstrated how large foundational generative models can be applied to contrastive learning. Our DM-based augmentations utilize DM's innate ability to denoise images by adding noise to the original image and then iteratively denoising it to form our generatively augmented image.

Throughout our research, we have faced several scientific challenges. The training and inference speed of generative models has severely limited the number of generative models we were able to use in our GCL framework (**SC1**). Also, due to the long training times of contrastive learning models, we were unable to train our contrastive learning benchmarks for the full 1,000 epochs typically used in contrastive learning. The scalability of generative models also limited the pool of generative models we were able to apply to the ImageNet dataset, specifically limiting us to GANs, image transformers, and DMs (**SC2**). As the encoder of a contrastive learning model is trained in an unsupervised manner, ideally, the generative model used in GCL should also be unsupervised (**SC3**). This problem is accentuated by the fact that

almost all SOTA modern generative models use labels, as this boosts their image quality. Due to all of the previously mentioned scientific challenges, we were not able to use current SOTA generative models in our GCL framework. This likely degraded the performance of our benchmarks, as we found using generative models with higher image quality resulted in higher downstream accuracy (**SC4**).

In our work on GCL, we have developed an efficient data generation framework that generates synthetic images that have similar visual features to images in the original dataset (**RQ1**). This form of data generation is especially useful for models that make use of augmentation-based learning. To ensure that the generative models produce images that maintain important high-level visual features while adding variations to unimportant low-level visual features, we rely heavily on the assumption that similar feature maps are more likely to contain the same high-level features than low-level features. Although difficult to prove, the results from increasing the strength of our feature map augmentations and their corresponding downstream accuracies indicate that increasing the difference between feature maps to a certain point does preserve high-level visual features while augmenting low-level ones.

Despite the large computational footprint of generative models, we have successfully incorporated them into our GCL framework without significantly increasing training time and computation (**RQ2**). We accomplish this through both preprocessing the generative augmentations and efficient sampling methods. By computing our generative augmentations beforehand, the additional training costs when training our contrastive learning model are insignificant. By using generative models with relatively fast inference times, we are able to generatively augment images at a fraction of the time it takes to train our contrastive learning model.

Finally, throughout our work, we have shown that using generative augmentations increases the downstream performance of contrastive learning in a self-supervised learning setting (**RQ3**). Although not computationally free, GCL consistently boosted downstream accuracy in every benchmark in our results. Our work demonstrates that the knowledge learned by generative models can be efficiently and effectively applied to downstream tasks in a self-supervised manner through the use of GCL.

References

[1] OpenAI. GPT-4 Technical Report. arXiv, 27 Mar. 2023. arXiv.org, https://doi.org/10.48550/arXiv.2303.08774.

[2] Brock, Andrew, Jeff Donahue, and Karen Simonyan. "Large scale GAN training for high fidelity natural image synthesis." arXiv preprint arXiv:1809.11096 (2018).

[3] Sauer, Axel, Katja Schwarz, and Andreas Geiger. "Stylegan-xl: Scaling stylegan to large, diverse datasets." ACM SIGGRAPH 2022 conference proceedings. 2022.

[4] Deng, Jia, et al. "Imagenet: A large-scale hierarchical image database." 2009 IEEE conference on computer vision and pattern recognition. Ieee, 2009.

[5] Kingma, Diederik P., and Max Welling. "Auto-encoding variational bayes." arXiv preprint arXiv:1312.6114 (2013).

[6] Peebles, William, and Saining Xie. "Scalable Diffusion Models with Transformers." arXiv preprint arXiv:2212.09748 (2022).

[7] Lučić, Mario, et al. "High-fidelity image generation with fewer labels." International conference on machine learning. PMLR, 2019.

[8] Van Gansbeke, Wouter, et al. "Scan: Learning to classify images without labels." Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part X. Cham: Springer International Publishing, 2020.

[9] Iglesias, Guillermo, Edgar Talavera, and Alberto Díaz-Álvarez. "A survey on GANs for computer vision: Recent research, analysis and taxonomy." Computer Science Review 48 (2023): 100553.

[10] Vahdat, Arash, and Jan Kautz. "NVAE: A deep hierarchical variational autoencoder." Advances in neural information processing systems 33 (2020): 19667-19679.

[11] Child, Rewon. "Very deep vaes generalize autoregressive models and can outperform them on images." arXiv preprint arXiv:2011.10650 (2020).

[12] Van Den Oord, Aaron, and Oriol Vinyals. "Neural discrete representation learning." Advances in neural information processing systems 30 (2017).

[13] Razavi, Ali, Aaron Van den Oord, and Oriol Vinyals. "Generating diverse high-fidelity images with vq-vae-2." Advances in neural information processing systems 32 (2019).

[14] Chen, Xi, et al. "Pixelsnail: An improved autoregressive generative model." International Conference on Machine Learning. PMLR, 2018.

[15] Cohen, Max, et al. "Diffusion bridges vector quantized Variational AutoEncoders." arXiv preprint arXiv:2202.04895 (2022).

[16] Yu, Jiahui, et al. "Vector-quantized image modeling with improved VQGAN." arXiv preprint arXiv:2110.04627 (2021).

[17] Rezende, Danilo Jimenez, Shakir Mohamed, and Daan Wierstra. "Stochastic backpropagation and variational inference in deep latent Gaussian models." International conference on machine learning. Vol. 2. 2014.

[18] Kingma, Durk P., and Prafulla Dhariwal. "Glow: Generative flow with invertible 1x1 convolutions." Advances in neural information processing systems 31 (2018).

[19] Ho, Jonathan, et al. "Flow++: Improving flow-based generative models with variational dequantization and architecture design." International Conference on Machine Learning. PMLR, 2019.

[20] Dinh, Laurent, Jascha Sohl-Dickstein, and Samy Bengio. "Density estimation using real nvp." arXiv preprint arXiv:1605.08803 (2016).

[21] Grcić, Matej, Ivan Grubišić, and Siniša Šegvić. "Densely connected normalizing flows." Advances in Neural Information Processing Systems 34 (2021): 23968-23982.

[22] Van Den Oord, Aäron, Nal Kalchbrenner, and Koray Kavukcuoglu. "Pixel recurrent neural networks." International conference on machine learning. PMLR, 2016.

[23] Van den Oord, Aaron, et al. "Conditional image generation with pixelcnn decoders." Advances in neural information processing systems 29 (2016).

[24] Nash, Charlie, et al. "Generating images with sparse representations." arXiv preprint arXiv:2103.03841 (2021).

[25] Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." Neural computation 9.8 (1997): 1735-1780.

[26] Cho, Kyunghyun, et al. "Learning phrase representations using RNN encoder-decoder for statistical machine translation." arXiv preprint arXiv:1406.1078 (2014).

[27] Chang, Huiwen, et al. "Maskgit: Masked generative image transformer." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2022.

[28] Sohl-Dickstein, Jascha, et al. "Deep unsupervised learning using nonequilibrium thermodynamics." International Conference on Machine Learning. PMLR, 2015.

[29] Ho, Jonathan, Ajay Jain, and Pieter Abbeel. "Denoising diffusion probabilistic models." Advances in Neural Information Processing Systems 33 (2020): 6840-6851.

[30] Dhariwal, Prafulla, and Alexander Nichol. "Diffusion models beat gans on image synthesis." Advances in Neural Information Processing Systems 34 (2021): 8780-8794.

[31] Saxena, Divya, and Jiannong Cao. "Generative adversarial networks (GANs) challenges, solutions, and future directions." ACM Computing Surveys (CSUR) 54.3 (2021): 1-42.

[32] Karras, Tero, et al. "Progressive growing of gans for improved quality, stability, and variation." arXiv preprint arXiv:1710.10196 (2017).

[33] Karras, Tero, et al. "Alias-free generative adversarial networks." Advances in Neural Information Processing Systems 34 (2021): 852-863.

[34] Lee, Doyup, et al. "Autoregressive image generation using residual quantization." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2022.

[35] Rombach, Robin, et al. "High-resolution image synthesis with latent diffusion models." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2022.

[36] Esser, Patrick, Robin Rombach, and Bjorn Ommer. "Taming transformers for high-resolution image synthesis." Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2021.

[37] Jiang, Yifan, Shiyu Chang, and Zhangyang Wang. "Transgan: Two pure transformers can make one strong gan, and that can scale up." Advances in Neural Information Processing Systems 34 (2021): 14745-14758.

[38] Phiri, Mwalimu. “Exponential Growth of Data.” Medium, Medium, 9 Mar. 2023, https://medium.com/@mwaliph/exponential-growth-of-data-2f53df89124.

[39] Tomasev, Nenad, et al. "Pushing the limits of self-supervised ResNets: Can we outperform supervised learning without labels on ImageNet?." arXiv preprint arXiv:2201.05119 (2022).

[40] Chen, Ting, et al. "Big self-supervised models are strong semi-supervised learners." Advances in neural information processing systems 33 (2020): 22243-22255.

[41] Chen, Ting, et al. "A simple framework for contrastive learning of visual representations." International conference on machine learning. PMLR, 2020.

[42] Grill, Jean-Bastien, et al. "Bootstrap your own latent-a new approach to self-supervised learning." Advances in neural information processing systems 33 (2020): 21271-21284.

[43] Li, Yinqi, et al. "Optimal Positive Generation via Latent Transformation for Contrastive Learning." Advances in Neural Information Processing Systems 35 (2022): 18327-18342.

[44] Kim, Youngsung, et al. "Deep generative-contrastive networks for facial expression recognition." arXiv preprint arXiv:1703.07140 (2017).

[45] Chen, Hao, et al. "Joint generative and contrastive learning for unsupervised person re-identification." Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2021.

[46] Zbontar, Jure, et al. "Barlow twins: Self-supervised learning via redundancy reduction." International Conference on Machine Learning. PMLR, 2021.

[47] Liu, Bingchen, et al. "Towards faster and stabilized gan training for high-fidelity few-shot image synthesis." International Conference on Learning Representations. 2021.

[48] Zhang, Yuxuan, et al. "Datasetgan: Efficient labeled data factory with minimal human effort." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2021.

[49] Li, Daiqing, et al. "BigDatasetGAN: Synthesizing ImageNet with pixel-wise annotations." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2022.

[50] Donahue, Jeff, and Karen Simonyan. "Large scale adversarial representation learning." Advances in neural information processing systems 32 (2019).

[51] Jahanian, Ali, Lucy Chai, and Phillip Isola. "On the" steerability" of generative adversarial networks." arXiv preprint arXiv:1907.07171 (2019).

[52] Casanova, Arantxa, et al. "Instance-conditioned gan." Advances in Neural Information Processing Systems 34 (2021): 27517-27529.

[53] Ci, Yuanzheng, et al. "Fast-MoCo: Boost Momentum-Based Contrastive Learning with Combinatorial Patches." Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXVI. Cham: Springer Nature Switzerland, 2022.

[54] Bardes, Adrien, Jean Ponce, and Yann LeCun. "Vicreg: Variance-invariance-covariance regularization for self-supervised learning." arXiv preprint arXiv:2105.04906 (2021).

[55] Jahanian, Ali, et al. "Generative models as a data source for multiview representation learning." arXiv preprint arXiv:2106.05258 (2021).

[56] Caron, Mathilde, et al. "Unsupervised learning of visual features by contrasting cluster assignments." Advances in neural information processing systems 33 (2020): 9912-9924.

[57] He, K., Fan, H., Wu, Y., Xie, S., & Girshick, R. (2020). Momentum contrast for unsupervised visual representation learning. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 9729-9738).

[58] Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P., & Joulin, A. (2021). Emerging properties in self-supervised vision transformers. In Proceedings of the IEEE/CVF international conference on computer vision (pp. 9650-9660).

[59] Pang, B., Zhang, Y., Li, Y., Cai, J., & Lu, C. (2022, October). Unsupervised visual representation learning by synchronous momentum grouping. In European Conference on Computer Vision (pp. 265-282). Cham: Springer Nature Switzerland.

[60] Ganz, R., & Elad, M. (2021). Bigroc: Boosting image generation via a robust classifier. arXiv preprint arXiv:2108.03702.

[61] Gao, S., Zhou, P., Cheng, M. M., & Yan, S. (2023). Masked diffusion transformer is a strong image synthesizer. arXiv preprint arXiv:2303.14389.\

[62] Video Generation Models as World Simulators, openai.com/research/video-generation-models-as-world-simulators. Accessed 1 Mar. 2024.

[63] Yu, L., Cheng, Y., Sohn, K., Lezama, J., Zhang, H., Chang, H., ... & Jiang, L. (2023). Magvit: Masked generative video transformer. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 10459-10469).

[64] Sauer, A., Lorenz, D., Blattmann, A., & Rombach, R. (2023). Adversarial diffusion distillation. arXiv preprint arXiv:2311.17042.

[65] Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., ... & McGrew, B. (2023). Gpt-4 technical report. arXiv preprint arXiv:2303.08774.

[66] Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M. A., Lacroix, T., ... & Lample, G. (2023). Llama: Open and efficient foundation language models. arXiv preprint arXiv:2302.13971.

[67] Gao, S., Zhou, P., Cheng, M. M., & Yan, S. (2023). Masked diffusion transformer is a strong image synthesizer. arXiv preprint arXiv:2303.14389.

[68] Mitrovic, J., McWilliams, B., Walker, J., Buesing, L., & Blundell, C. (2020). Representation learning via invariant causal mechanisms. arXiv preprint arXiv:2010.07922.

[69] Chen, X., & He, K. (2021). Exploring simple siamese representation learning. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 15750-15758).

[70] Gidaris, S., Bursuc, A., Puy, G., Komodakis, N., Cord, M., & Pérez, P. (2021). Obow: Online bag-of-visual-words generation for self-supervised learning. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 6830-6840).

[71] Wang, W., Shen, J., Xie, J., Cheng, M. M., Ling, H., & Borji, A. (2019). Revisiting video saliency prediction in the deep learning era. IEEE transactions on pattern analysis and machine intelligence, 43(1), 220-237.

[72] Assran, M., Duval, Q., Misra, I., Bojanowski, P., Vincent, P., Rabbat, M., ... & Ballas, N. (2023). Self-supervised learning from images with a joint-embedding predictive architecture. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 15619-15629).

[73] Baevski, A., Hsu, W. N., Xu, Q., Babu, A., Gu, J., & Auli, M. (2022, June). Data2vec: A general framework for self-supervised learning in speech, vision and language. In International Conference on Machine Learning (pp. 1298-1312). PMLR.

[74] Meng, C., He, Y., Song, Y., Song, J., Wu, J., Zhu, J. Y., & Ermon, S. (2021). Sdedit: Guided image synthesis and editing with stochastic differential equations. arXiv preprint arXiv:2108.01073.

[75] Rombach, R., Blattmann, A., Lorenz, D., Esser, P., & Ommer, B. (2022). High-resolution image synthesis with latent diffusion models. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 10684-10695).

[76] Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., ... & Sutskever, I. (2021, July). Zero-shot text-to-image generation. In International Conference on Machine Learning (pp. 8821-8831). PMLR.

[77] Dwibedi, D., Aytar, Y., Tompson, J., Sermanet, P., & Zisserman, A. (2021). With a little help from my friends: Nearest-neighbor contrastive learning of visual representations. In Proceedings of the IEEE/CVF International Conference on Computer Vision (pp. 9588-9597).

[78] Yu, L., Lezama, J., Gundavarapu, N. B., Versari, L., Sohn, K., Minnen, D., ... & Jiang, L. (2023). Language Model Beats Diffusion--Tokenizer is Key to Visual Generation. arXiv preprint arXiv:2310.05737.

[79] Video Generation Models as World Simulators, openai.com/research/video-generation-models-as-world-simulators. Accessed 1 Mar. 2024.

[80] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

[81] Brunet, D., Vrscay, E. R., & Wang, Z. (2011). On the mathematical properties of the structural similarity index. IEEE Transactions on Image Processing, 21(4), 1488-1499.

[82] Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., & Hochreiter, S. (2017). Gans trained by a two time-scale update rule converge to a local nash equilibrium. Advances in neural information processing systems, 30.

[83] Oquab, M., Darcet, T., Moutakanni, T., Vo, H., Szafraniec, M., Khalidov, V., ... & Bojanowski, P. (2023). Dinov2: Learning robust visual features without supervision. arXiv preprint arXiv:2304.07193.