THE UNIVERSITY OF CHICAGO

MASTER'S THESIS ANALYSIS OF GENERATIVE ADVERSARIAL MODELS

FOR THE Degree OF
MASTER OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

BY
STEVEN BASART

CHICAGO, ILLINOIS
MONTH 2017
# TABLE OF CONTENTS

LIST OF FIGURES .................................................. iv

ABSTRACT ................................................................. v

1 INTRODUCTION ...................................................... 1
  1.1 Background ................................................... 1
     1.1.1 Objectives .................................................. 1
     1.1.2 Contributions .............................................. 2
  1.2 Outline .......................................................... 3

2 GENERATIVE ADVERSARIAL NETWORKS ......................... 4
  2.1 Overview ....................................................... 4
     2.1.1 Related Work .............................................. 4
     2.1.2 Datasets ................................................... 6
  2.2 Architectural Variants ......................................... 7
  2.3 Training .......................................................... 7
     2.3.1 Loss .......................................................... 9
     2.3.2 F Divergences .............................................. 10
     2.3.3 Training Heuristics ....................................... 11
     2.3.4 GAN Issues ................................................ 12
  2.4 Contributions ................................................... 12
     2.4.1 Architecture ................................................. 13
     2.4.2 Training Procedure ....................................... 14
     2.4.3 Interpretability ............................................ 15
     2.4.4 Semi-Supervision .......................................... 15
     2.4.5 Memorization ............................................... 17

3 EVALUATION OF GENERATIVE MODELS ....................... 18
  3.1 Importance ..................................................... 18
  3.2 Measure .......................................................... 19
  3.3 Embeddings ..................................................... 21
  3.4 Related Works .................................................. 21
  3.5 Experimental Results .......................................... 23

4 CONCLUSIONS ...................................................... 26
  4.1 Conclusion and Future Work ................................... 26
# LIST OF FIGURES

2.1 VAE Diagram .......................................................... 5  
2.2 GAN vs VAE samples .............................................. 6  
2.3 U-net architecture ................................................ 8  
2.4 Pix2Pix Example ................................................... 8  
2.5 Modnet ............................................................... 13  
2.6 Interpretability samples ........................................... 15  
2.7 Semi-supervised .................................................... 16  

3.1 Alternative semantic embeddings ............................... 21  
3.2 Parzen Window Estimation ....................................... 23  
3.3 Improved GAN CIFAR ............................................. 23  
3.4 Our Measure ........................................................ 24  
3.5 Inception scores ................................................... 24  
3.6 Comparison of Measures on Modes ............................ 25
ABSTRACT

We study Generative Adversarial Networks (GANs) and their applications to various tasks as well as the creation of a measure to analyze their performance. Regarding the study of GANs, we create our own novel network that can be used for interpretability, image generation, and augmenting existing datasets via semi-supervision. Last, we create a new measure to score sets of images as opposed to scoring individual images. The measure uses the relative divergence between two pairs of distributions, namely a training set and test set, and a training set and generated set. This measure has several nice properties: tracks image quality, is unbiased, detects missing modes, and captures generalization of the generative model.
CHAPTER 1
INTRODUCTION

1.1 Background

Generative models have been used in such tasks such as information retrieval [Wang et al., 2013], text classification [Antti, 2012], and speech generation [van den Oord et al., 2016]. The problem of generating natural images however, was considered too difficult until recently. Since 2016 researchers have begun using generative models to generate plausible looking images using such methods as autoregressive models such as pixelcnn [Salimans et al., 2017], probabilistic graphical models such as variational autoencoders [Kingma and Welling, 2013], and generative adversarial networks [Goodfellow et al., 2014a].

The generation of images is an important task for three main reasons: scientific, utilitarian and artistic value. The first scientific reason is that by having a good generator of natural images, one can use the generator as a prior over natural images. A prior can be used for other methods that require a prior over natural images currently used in computer vision. The second reason for the importance in generating images is for utilitarian reasons. There are many tasks that could benefit from generating natural images such as inpainting, colorization, sampling from rare categories, and generating artwork. The last reason is for artistic value. One can consider the task of ”searching for similar images” which is approached as an information retrieval problem, but it could be reformulated as an generation task to become ”generate similar images”.

1.1.1 Objectives

Given the importance of generating images, the work presented in this paper covers a recent research direction namely that of Generative Adversarial Networks or GANs. The main objective of the research involving Generative Adversarial Networks is to explore a novel
approach on utilizing the generator as a basis to query other models. Our first objective aims to provide a solution to the problem that neural networks are considered uninterpretable black box models. In this way we can use a the generator of a generative adversarial network to query what examples would fool another model.

A secondary objective of the GAN research is to explore how well generators can act to supplement or augment a dataset. By modeling the entire distribution it should be possible to generate samples that are harder and can help in generalization. We then explore how diverse and distinct these images are.

The final objective of this work involves the creation of an evaluation measure for generative models. Given the creation of another GAN model in this work, their lacked previously any means to whereby quantitatively compare different generative models. The measure aims to measure generalization error of the generator and ignores the discriminator of GAN. By formulating the measure in this way we can compare different image based generative models outside of GANs.

\[1.1.2 \text{ Contributions} \]

My contributions to GAN research are as follows:

- Creation of a novel GAN architecture. 2.4.1
- Novel approach for Neural Network interpretability. 2.4.3
- Demonstrating usefulness in augmentation of existing datasets. 2.4.4

Related to the creation of the measure the contributions are as follow.

- The measure approximates the relative divergence between generated and real distributions. 3.2
• The measure compares entire distributions rather than comparing on a per-image basis.

3.2

• The measure tracks image quality. 3.5

• The measure penalizes GANs that miss modes. 3.5

1.2 Outline

The paper presents its points in the following order first with what Generative Adversarial Networks are, as well as a sampling of the work in this area, followed by my contributions to this domain. In Section 3 we address an important issue with generative models, which is a way to evaluate their performance. Afterwards we cover my experimental results regarding Generative Adversarial Networks and that of demonstrating our evaluation metric for generative models. Finally we discuss our conclusions and that of potential directions for future work.
CHAPTER 2
GENERATIVE ADVERSARIAL NETWORKS

2.1 Overview

Generative Adversarial Networks (GANs) are a type of generative model that uses neural networks and an adversarial training scheme to train a generative model. First proposed in Goodfellow et al. [2014b] and subsequent work quickly displayed sharp images unlike those seen with other methods see Fig 2.2. At the core of Generative Adversarial Networks is the concept of two competing (adversarial) networks. One network, termed $G$, that takes in random noise as input and generates an image as an output. This network is termed the generator. The other network, termed the discriminator ($D$), has the task to predict the probability a given image is real. Its task is therefore to discriminate between the real and generated images by assigning a low score to generated or fake images while assigning a high score to real images.

The other main variant that still falls under the GAN purview is that of the image to image models first presented in Isola et al. [2017]. These models are also sometimes termed CoGAN for conditional generative adversarial networks. The main difference between CoGANs and a GANs are that the generator network ($G$) in a CoGAN will also take in as input an image to condition its output on. The main focus for the remainder of this chapter will be on the CoGAN model. While we’ll refer to my model as GAN it will be understood that it is specifically a conditional generative adversarial model rather than a plain generative adversarial model.

2.1.1 Related Work

While GANs have become a popular generative model there are other generative models that exist. Variation Autoencoders (VAEs) are a type of autoencoder which are neural network
models that approximate some automorphism. The main difference between an autoencoder and a variational autoencoder are that the bottleneck within the model takes on a specific form in a variational autoencoder and has an additional loss term as seen in Fig 2.1. The Bottleneck is of the form of two vectors where the first vector models the mean and the second vector models the covariance of a multidimensional guassian. The reasoning behind generating these two vectors is because we want to encode our input as a point in some multivariate guassian. To enforce this encoding resembles a multivariate guassian, there is the additional loss of the form $-KL((\mu, \Sigma), \mathcal{N}(0, 1))$. The additional loss term is a penalty for how far the means and variances differ from the standard multidimensional guassian and in practice it is typically scaled by some lambda factor.

The main benefit of this model is that once trained it becomes easy and efficient to sample from the model. The generation of the image happens by initializing the bottleneck with parameters sampled from a multivariate guassian and running the model beginning from the bottleneck layer through the rest of the network.

Another recent generative model is known as Pixel CNN [Salimans et al., 2017]. There have been a few variants of this model such as Pixel RNN and Pixel CNN++. The model predicts the next pixel given the previous context which are all of the previous pixels in raster order. As a generative model one just needs to initialize the first pixel the model sees and then it can be used to generate all of the other pixels in raster order. It has had its
greatest success as a means for compression.

Finally there are older generative models that have existed for a while such as guassian mixture models. However previous approaches utilizing these models failed to generate anything resembling natural images.

### 2.1.2 Datasets

The datasets used for the experiments presented in this paper include: MNIST, SVHN, CIFAR, Char74k.

**MNIST.** The MNIST dataset consists of black and white handwritten digits has a training set of 60,000 examples, and a test set of 10,000 examples.

**SVHN.** The Street View House Numbers (SVHN) dataset is of cropped 32 by 32 digits found through Google Street View. There are 73,257 images in the training set, 26,032 images in the test set, and 531,131 images for additional training.

**CIFAR.** The two CIFAR datasets consist of colored natural scene images, with 32x32 pixels each. CIFAR-10 consists of images drawn from 10 and CIFAR-100 consists of images drawn from 100 classes. For both datasets there are 50,000 training images and 10,000 test
Char74k. We used the subset of English characters that consists of 12,503 images. We randomly shuffled the dataset and split it into 10503 for training and 2000 for testing. Even on a dataset with this few examples per class ModNet is able to generate compelling examples.

2.2 Architectural Variants

The architectural space that has been explored by the research community relating to GAN research has covered the majority of the architectural variants that have won the ImageNet Russakovsky et al. [2014] challenge. The first paper on the subject used only fully connected layers and then the next several architectures followed a fully convolutional approaches. There have been several attempts to use more powerful architectures such as ResNet or others but all of the results thus far have been poor.

For our experiments we use a modification of the U-Net architecture (also presented in some literature as V-Net). Our architectural modifications are described in more detail in section 2.4.1. The U-Net architecture was first used for GAN research in Isola et al. [2017]. The symmetric shape of the U-Net architecture is constructed in such a way that allows features from earlier layers to get concatenated onto a later layer in the network. Examples of using the U-Net as a generator in be seen in Fig 2.4.

2.3 Training

An important question that remains is how to train a GAN. The Discriminator is trained like a binary classifier where the positive labels correspond to real images and the negative labels correspond to the generated images. The Generator is trained as an adversary to the discriminator. It is trained to maximize the loss of the Discriminator and is updated by
Figure 2.3: U-net architecture
Example for 32x32 pixels in the lowest resolution. Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations.

Figure 2.4: Pix2Pix Example

back-propagating the loss through the discriminator network to the generator network. The loss function is covered in more detail in 2.3.1.

There are a few differences in training a CoGAN as compared to a GAN. The main
difference is that the loss function changes by the addition of an $L_1$ norm penalty between the generated image and the target image. The additional loss term is sometimes weighted.

### 2.3.1 Loss

Let us now define some notation used hereafter. Discriminator output $D$, and generator output $G$ are from our conditional generator. Input image $x$ is a real image, and label $y$ is the corresponding label of image $x$. Let $y_t$ be the target label. Finally, let $\sigma$ represent the logistic sigmoid function. We can summarize our loss as follows

\[
L_{GAN} = L_G + L_D
\]  
\[
L_G = -\log(\sigma(D(G(y_t|x))))
\]  
\[
L_D = -\log(\sigma(D(G(x)))) - \log(1 - \sigma(D(G(y_t|x))))
\]

Our first loss is the typical GAN loss. The only thing to note here is that $L_D$ sometimes has an alternative form $L_D = -\log(\sigma(D(G(x)))) + \log(\sigma(D(G(y_t|x))))$. We found that the alternative form tended to be much less stable for our experiments.

\[
L_{label} = y \log(\text{softmax}(D)) + (1 - y) \log(\text{softmax}(D))
\]

The next loss that we employ is the standard cross entropy loss. Unlike many other GAN papers such as Salimans et al. [2016] we wish to encourage the discriminator to classify the real images $G(x)$ with true label $y$. At the same time we also encourage $D$ to classify generated images $D(G(y_t))$ with the target label $y_t$. Next we define our modification penalty

\[
L_1 = \|x - G(y_t|x)\|_1.
\]
This last loss is an $\ell_1$ penalty between the generated image and that the image it is conditioned on. In this way we try to enforce that the changes applied to an image remain minimal. Now, we have a combined loss

$$L = L_{GAN} + L_{label} + \lambda L_1.$$  \hspace{1cm} (2.6)

By combining all three losses, we can train our model to produce natural fooling examples. We add a hyper-parameter $\lambda$ to the $L_1$ loss to control the strength of preservation over class adjustment.

### 2.3.2 F Divergences

This section is a comment on the loss function that is typically used and about our own loss function that we used. The typical loss function used in GAN literature derived a relation of the Jenson Shannon Divergence (JSD) to the Adversarial Loss however the Adversarial loss is then modified from

$$-\log(1 - D(G(x)))$$  \hspace{1cm} (2.7)

to

$$\log(D(G(x)))$$  \hspace{1cm} (2.8)

The authors of the original paper Goodfellow et al. [2014b] by performing this modification lose any connection to JSD. This new loss function has no known theoretical implications.

Due to this new loss function having no known divergence counterpart another work Nowozin et al. [2016] has shown that the specific divergence chosen in Goodfellow et al. [2014b] is not necessary to make GANs work. In their work they test out a variety of divergences and term their GAN f-GAN showing that the family of f-divergences can all work to train a GAN. Similarly there have been other loss functions applied such as the Wasserstein distance. In these papers though they compared the GANs using the "easiest"
datasets for GANs which are LSUN and MNIST. LSUN is the large scale scene understanding dataset, however when used in the GAN literature they typically restrict the dataset to only include bedrooms. Ultimately though there is no theoretical or even practical best loss function as each paper seems to use a slight modification.

This lack of a best function is highlighted by the literature where some papers either delay showing results or fail to show any comparative results and instead highlight their own hand picked images. In my own experiments we have found that loss function does affect the time to convergence and even the final qualitative outputs.

### 2.3.3 Training Heuristics

In addition to what was mentioned in section 2.3, the first GAN paper [Goodfellow et al., 2014a] employed a differential or varied update schedule between the discriminator and that of the generator. The generator is updated twice as often as the discriminator as in for every update step that the discriminator undergoes it will be frozen for the next two while only the generator is updated. There are a few commonly held intuitions behind this heuristic. One is that the discriminator has an easier time learning to discriminate. Another is that the generator has a harder time learning to generate a new distribution that can fool the discriminator. The above training scheme more often than not ends up with a collapsed model that produces output akin to white noise. Several heuristics are employed to deal with model collapse described below.

Other heuristics employed also appear to target the discriminator by making it weaker. A heuristic termed label flipping adds noise to the positive target labels. This asymmetric targeting tries to smooth the two distributions that the discriminator learns. Similarly another technique adds noise directly to the real and generated inputs fed into the discriminator. The level of noise is dropped on an exponentially decaying schedule.

Inspired by these techniques we tested freezing a layer or several layers of a discrimina-
tor. This acts as a stronger level of corruption due to the tensor undergoing a nonlinear combination. We found this technique to outperform adding noise or label flipping and can be augmented with label flipping. This leads to hypothesis that weak discriminators are optimal at least in the beginning of training to make GANs work. It still does not illucidate how to improve GAN training however.

The success or failure of a particular heuristic is still hard to determine apriori under what conditions they work well. However a common theme among the heuristics which we have noticed is that they all aim to weaken the discriminator.

2.3.4 GAN Issues

The section 2.3.3 highlighted some of the issues dealing with model collapse for GANs. Specifically that refers to when the generator stops learning or only produces one output. However there are other issues that are encountered when training GANs. One of the issues is with the training of GANs, a lot work in the literature has focused on how to stabilize the training.

Another issue is that of having a meaningful loss function. Currently the loss function can at best be a reflection as to how well the generator is "fooling" or the discriminator is successfully discriminating. However performing at an optimum for this metric does not necessarily correspond to realistic generations. The optimum itself is also typically not reached when training GANs.

2.4 Contributions

While in some previous sections we have highlighted a few of the contributions, in this section we will directly speak to some of the more prominent contributions of my recent work. We will discuss the new architecture, new method for interpretability, results in semi-supervision and memorization.
For our work we further modified the U-Net to instead produce an image mask that would be applied to an image. In this way the network termed Mod-Net modifies the original image to produce a new image. The generator of Mod-Net will generate an image in the range $[-1, 1]$. This image termed a "delta image" will be added point-wise to the original image to highlight or mask out features present in the original image. We direct the label of the label of the generated image by conditioning the generator on a one-hot vector that is fed in along with the original image. An example is shown in Fig 2.5.

The Architecture dubbed ModNet varies from its sibling network the U-Net in three key ways. In utilizing the UNet architecture to serve as a generator the network has to learn the complex task of generating images. However because the network is conditioned on natural images it’s task is easier than regular GAN which have to map from the space of a uniform gaussian to natural images. Our network termed ModNet could be argued has an easier task than the UNet architecture. The task has changed from generating an entire image to
generating the deltas required to be applied to the original image.

The other modification is that now the final nonlinear activation has to be changed from a sigmoid unit to a tanh. The reasoning behind this change is that if the deltas are bounded in \([0,1]\) instead of \([-1,1]\) the network is constrained to only learn to add to the image. When testing under this setting we observed poor results mostly just adding a bright filter or white mask over areas instead of learning a proper transformation.

Lastly by adding for the additional classification loss we now can create two variants of our architecture. One in which the discriminator acts as both discriminator and classifier as seen in Odena et al. [2016]. The other new variant is where we can split our classifier from our discriminator this allows for two different gradient signals to be received by the generator.

There are two variants to the discriminator we use depending on the use case. The first variant is where we have our discriminator output the probability score \([0, 1]\) and a classification score. This is similar to the discriminator used in Odena et al. [2016] where they attach an extra output layer to the discriminator. The other variant shown in Fig 2.5 is where we can separate the discriminator into two parts. One part is the standard discriminator that outputs the \([0,1]\) loss and the other is a separate classifier. The classifier has the option of being pretrained in this manner and optionally frozen.

\[2.4.2 \text{ Training Procedure}\]

Based on the use case of our model there two distinct training procedures that could be employed. The two use cases are for interpreting a pretrained model or for augmentation. The training for the latter follows the training procedure described in section 2.3. The training for interpretability requires that the discriminator be separate from the classifier. The classifier is never updated during training but it’s error signal is used to update the the generator.
A sample of CIFAR-10 images that fool a weak classifier. We can see that the model learns to associate wheels with trucks by putting them on a cat and switching the label. Target labels of generated images are as follows. 1st row: Deer, Deer, Truck, Cat. 2nd row: Cat, Horse, Ship, Plane. 3rd row: Truck, Truck, Deer, Truck. 4th row: Car, Dog, Bird, Horse.

2.4.3 Interpretability

In this section we highlight how with this new approach we can better understand what the networks themselves are learning. This can be used to attempt to debug, or make them better. To run the interpretability experiments we needed to use the variant of our architecture where the classifier and discriminator are separate from one another. In this way our goal is to generate images with a high classification score that fool a classifier.

For Fig 2.6 we show what happens when you take a shallow classifier and use our method to fool it. We are able to show some small and also later image changes that will lead to a high classification score.

2.4.4 Semi-Supervision

Previous work such as Salimans et al. [2016] used GANs as a tool to expand a small dataset. They created several thousand images from a GAN and used a fraction of the training data to train a new classifier. They reported the total error as opposed to accuracy as we suspect
it didn’t perform that well compared to other data augmentation methods. Another thing that should be noted is that their model could be memorizing the training data. Some indications that this could be true are that they use a per pixel l2 loss during their GAN training.

We were inspired by their idea though to use a generator to create extra images that can be used for semi supervised learning. We ran two different semi supervised experiments. The first experiment involves creating an entirely generated conditional dataset. From this we train a new classifier and measure it’s performance. Using only generated images with unbalanced label counts we can achieve 88% which is rather low accuracy compared to regular dataset. This implies that using the generator causes some of the features in the training data to be removed and not allow for as great generalization as the original dataset.

The second experiment involves taking a subset of the original training data and using only those images as the conditional images to generate the other labels in the dataset. Then a new classifier is trained on both the generated and real subset of the images and is compared to a classifier trained on only real data. The results for this experiment are shown
Lastly because we were under the suspicion that the traditional GANs might be memorizing the training data, we wanted to test if this was also the case for the CoGAN. We computed the nearest neighbors of all of the training images and looked up the nearest neighbor of all the generated images against training images. We observed that all of the generated images were always closest to image they were conditioned on rather than any other image. This led us to the conclusion that our training scheme was not retrieving another image within the training data that was of the target label. Even while this might be the case for my particular CoGAN it could still be true of GANs in general though.
CHAPTER 3

EVALUATION OF GENERATIVE MODELS

In this section we address the concerns about memorization and a lack of any evaluation metrics or measures. To address these concerns we have to ask the fundamental question what is the purpose of a generative model? This question is important because in the literature there are various different goals that papers propose for GANs including as a pretraining scheme and for creating a good prior. We would like to argue that the true purpose of generative models is to create a model that captures the full joint distribution.

3.1 Importance

As many papers have released on the topic of GANs, some have even gone so far as to claim that they are state of the art in generation. Crucially though there was no way to say if any model is any ”good” or even better than a different model. Claims such as state of the art and better require some form of comparison of which none are currently used. The creation of some measure or metric is therefore of importance to be able to substantiate current claims from published works.

My contributions are as follows

- The measure approximates the relative divergence between generated and real distributions. 3.2
- The measure compares entire distributions rather than comparing on a per-image basis. 3.2
- The measure tracks image quality. 3.5
- The measure penalizes GANs that miss modes. 3.5
3.2 Measure

Let’s define some of the notation used in our measure. Let T be our test distribution, R be the training distribution, G be the generated distribution. Let N(X, Y) be a matrix that consists of the nearest neighbors from each point in X to Y.

We are utilizing the relative relative entropy which is defined below.

\[
D((A||B)||(C||D)) = \sum_x P(A_x)ln \frac{P(A_x)P(D_x)}{P(B_x)P(C_x)}
\]  

(3.1)

Relative relative entropy measures the relative divergence between two pairs of distributions. In our case we wish to measure the relative divergence between the pair of R and T relative to G and T. Due to the potentially large space that all the distributions live in instead of dealing with G, R, and T directly we will deal with a common embedding space for all of them. In this way we can avoid the curse of dimensionality and as long as the embedding dimension is large enough we can still capture all the significant variation present between the distributions.

Some questions still remain in terms of how to represent images as distributions. One approach is to consider each image as a delta function around which some mass exists. This creates a point-wise mass distribution that can be used in common for each distribution. Even with this representation though computing the actual probability distribution of space is still intractable. From here we further relax the distributions to be modeled via a gaussian mixture. Gaussian mixtures are still intractable in their general form but we can approximate them with a variational approximation which will provide us with a lower bound on the true distribution Goldberger et al. [2003].

We arrive at our variational approximation by simplifying Goldberger et al. [2003] work in approximating Gaussian mixtures. The following derivation begins from said work.

The likelihood \( L_f(g) = E_{f(x)}[log g(x)] \) relates to the KL divergence by \( D(f||g) = \)
\( L_f(f) - L_f(g) \). Therefore any estimate of likelihood can be related back to the KL.

By Jensen’s inequality we have

\[
L_f(g) = E_{f(x)} \log g(x) \\
= E_{f(x)} \log \sum \omega_b g_b(x) \\
= E_{f(x)} \log \sum \phi_{b|a} \frac{\omega_b g(x)}{\phi_{b|a}} \\
\geq E_{f(x)} \log \sum \phi_{b|a} \log \frac{\omega_b g(x)}{\phi_{b|a}} \\
= \mathcal{L}_f(g, \phi) \tag{3.2}
\]

This being a lower bound on \( L_f(g) \), we get the best bound by maximizing \( \mathcal{L}_f(g, \phi) \) with respect to \( \phi \).

\[
\hat{\phi}_{b|a} = \frac{\omega_b e^{-D(f_a || g_b)}}{\sum_{b'} \omega_{b'} e^{-D(f_a || g_{b'})}} \tag{3.3}
\]

A similar bound can be achieved for \( L_f(f) \). Finally we define the variational approximation by substituting \( \phi \) into \( L_f(g) \) and the corresponding \( \psi \) into \( L_f(f) \).

\[
D(f || g) = \sum_a \pi_a \log \frac{\sum_{a'} \pi_{a'} e^{-D(f_a || f_{a'})}}{\sum_{b'} \omega_{b'} e^{-D(f_a || g_{b'})}} \tag{3.4}
\]

In our specific case we can simplify the variational approximation to what’s below due to some assumptions. Namely we assume an equal weighting of each Guassian in the mixture. Thus the \( \pi \) and \( \omega \) become a normalization parameter equal to the number of examples in each respective mixture.

\[
\left[ \log \left( \sum_{t \in T} e^{-N(R,T)} \right) + \log \left( \frac{|G|}{|T|} \right) - \log \left( \sum_{g \in G} e^{-N(R,G)} \right) \right] \cdot \frac{1}{|R|} \tag{3.5}
\]
3.3 Embeddings

The embedding was left purposefully vague in the definition of the measure because we have tested a few embeddings and found that they all seem to work. However given a large enough dataset might show that one embedding doesn’t work well enough.

Random Projections: Inspired by the work in compressed sensing we tried to take random 1D sparse projections of each image.

PCA compression: For this embedding we take the top principal components that account for 90% of the variation in the data and pca compress all of the images to their principal eigen vectors.

![Figure 3.1: Alternative semantic embeddings](image)

SVHN results. We show that other semantic representations track each other with our measure.

Semantic representations: Lastly the embedding scheme that we believe to be the most robust is a semantic embedding. For this approach we take a state of the art network for that dataset and represent each image with its logits. We compared this to other semantically informed embeddings in Figure 3.1. In the Figure we can see that other semantic embeddings produced similar results such as hypercolumns and the layer before logits.

3.4 Related Works

There are few prior quantitative measures to evaluate GANs. The few that do exist have either been a poor measure or far too time consuming to actually be put into practice.
For example, Im et al. [2016] proposes a relative measure between two GANs that essentially compares the ability of the discriminator to discriminate between real and fake which seems to miss the intended purpose of GANs as described earlier. In a similar vein the work by Lopez-Paz and Oquab [2017] provides a parametric approach to compare distributions through a discriminator. By using a parametric approach it measures the capacity of another model to distinguish between real and generated, instead of measuring the capacity of the generator.

The work from [Salimans et al., 2016] introduced several approaches to evaluate their GAN model. The first of which is the **Inception score** which is a “rough guideline.” This useful guideline is unfortunately ineffective should the GAN learn to constantly generate a single compelling example, and this measure is limited to images like CIFAR-10. The other approach proposed in the paper is to use GANs to generate data that can be used in a **semi-supervised learning** setting. This approach is a useful quality of a measure for a GAN to measure its performance on some downstream task. However, it remains to be seen if the performance is task specific and there does not appear to be a standard benchmark for the task.

Another technique to evaluate GANs is **Parzen window estimation**. This non-parametric method can be used to compute the log-likelihood of GAN samples. However as Theis et al. [2015] show Parzen window estimates can be a poor approximation when the data is high-dimensional. We show that this is indeed the case for a complex datasets like CIFAR-10. Our measure addresses the concerns facing kernel density estimators by embedding the samples into a semantically meaningful space and by using KL-divergence instead of computing log-likelihoods.

Other measures such as total variation and calculating the number of missing modes can be useful features but assuredly do not capture most generative modeling desiderata.
Figure 3.2: Parzen Window Estimation
Parzen window estimation on CIFAR-10. The samples from the model without Batch Normalization never achieve good image quality, yet they improve beyond the normal model with respect to log-likelihood.

3.5 Experimental Results

Using the MNIST, SVHN, and cifar datasets, we trained a regular GAN and a weakened GAN version created by removing the first batchnorm from the generator. Without the batchnorm the samples become lower quality and also produce redundant results as shown in Fig 3.3. The results of our measure are shown in Fig 3.4.

We show the results of the inception score for the same datasets in Fig 3.5. The results for MNIST seem to follow a curious trend in the inception score but it does preserve the correct ordering. Finally we compare our measure against the inception score when addressing the
Our measure plotted over time for the datasets. The baseline for all of the experiments is zero.

issue of modes. We simulate the generation of modes by sampling real images from the training set and repeating those samples until we attain the same number of samples as the size of the training data. The results of this experiment are shown in Fig 3.6.

Figure 3.5: Inception scores
Inception scores plotted for the datasets. Larger values indicate better score.
For this experiment we sampled 10, 20, 100, 1000, 10000 images from the training set and replicated the samples up to the size of the original dataset of 50000. The inception score saturates around 1000 images or a replication amount of 5. For our measure since we also compare to a test set it does not saturate but instead approaches the relative divergence.

For both measure the orange line represents the best performance on each respective measure.
CHAPTER 4
CONCLUSIONS

4.1 Conclusion and Future Work

In this work we created both a novel GAN architecture, and approach that we demonstrate can be utilized for both interpretability and augmentation. However this method does not solve the issue of instability in GAN training. This approach still fails to produce interpretable results on image datasets that GANs still fail on such as Imagenet or COCO.

With regards to our measure, we believe it to be quite robust as to the settings we have tested against. It is also a relatively fast measure to compute. The current main downside of the method is the space complexity grows as the square of the dataset. We are currently exploring ways to mitigate this impact perhaps through clustering of the representations before computing the pairwise distances. However assuming the dataset size isn’t very large our approach outperforms all other methods.

For future work we plan to examine curriculum learning or training schedules. This can be approached in two ways: first start with easy examples then make it harder. The other approach can be to progressively make the discriminator stronger over time and allow the generator to consistently "win". Neither of these have been studied partially because coming up with a good curriculum is rather difficult. The idea of a discriminator curriculum has never been proposed to the best of my knowledge.
REFERENCES


