A Comparison of Conditional Generative Adversarial Networks

And a Brief Look at Image-to-Image Translation

Peter Bromley
Middlebury College
Middlebury, VT
pbromley@middlebury.edu

ABSTRACT

Generative Adversarial Networks (GANs), introduced by Goodfellow *et al.*[6], are a popular new deep learning tool that has produced impressive results in the task of generative modeling. In the past few years, many papers have documented a plethora of variants of the "vanilla" GAN framework that either improve upon the original approach or accomplish different tasks. One subset of these variants is Conditional GANs, initially proposed by Mirza *et al.*[15], which uses class labels to influence the learning process and create class-conditioned networks. Another subset attempts to solve the problem of image-to-image translation, in which images from one domain are stylistically transferred to another domain. In this paper, we perform a direct comparison between three conditional models documented in the literature (cGAN[15], ACGAN[16], and InfoGAN[3]) as well as a model inspired by a comment online which will be referred to as 2NGAN. Toy datasets (MNIST[12], Fashion-MNIST[22], CIFAR10[11]) are used in the comparison for consistency. Models and images are compared conceptually (how labels are utilized and what that may affect), subjectively, and empirically (Inception Score). We find that our 2NGAN model outperforms the others on the widely used Inception Score metric[19] and also exhibits the most stability in training. Additionally, we briefly look into image-to-image translation with CycleGAN[24] and examine potential limitations of the model, specifically a lack of structural manipulation that limits the possible image domain pairings used for translation.

Keywords

Generative Adversarial Networks; Generative Modeling, Conditional Generative Adversarial Networks, Inception Score, Image-to-Image Translation

1 Introduction

The human brain boasts an impressive image generation ability that allows for the instantaneous visualization of the world. When asked to imagine an elephant, the brain has little trouble piecing together an image of a gray animal with large ears, coarse skin, and a long trunk. This elephant is not "real", but is instead most likely an approximation based on previous images that have been processed and classified by the brain. This ability requires a fairly detailed understanding of an elephant and is a potentially crucial skill for an intelligent machine to possess. A technique that attempts to address this problem is generative modeling.

Generative modeling is a challenging but exciting field of machine learning that has been heavily studied in recent years. Broadly, generative modeling refers to the mapping of a latent, often random input to a point on a high dimensional probability distribution. The goal is for the generated distribution to be similar to the real data distribution. The production of synthetic data using a learned generative function allows for the exploration of structural components and relationships in the real data distribution. We can use a good generative model to get a more comprehensive understanding of a high dimensional probability distribution by investigating the way in which the learned function maps various latent inputs to the data manifold. Some generative modeling processes have been shown to produce results that can be applied to practical problems such as image super-resolution[13], stylistic image translation[24][10], and in-painting[4], but the potential of deep generative models stretches to a more general use-case as a replication of the imaginative ability of the human mind[1]. This application could be an exciting opportunity to teach computers how to structurally view the world around them.

Recently, Generative Adversarial Networks[6] were discovered to be an effective way to approach generative modeling. A deep learning framework that utilizes a min-max "adversarial" training approach, GANs have produced impressive results generating highly realistic synthetic images. Since the inception of GANs, many researchers have experimented with the model and come up with variants that produce higher quality or novel output. One common category of these variants is defined by the use of conditional labels as an extra input, originally introduced by Mirza *et al.*[15]. Labels can be incorporated in the GAN framework in a number of creative ways. Mirza *et al.*[15], Chen *et al.*[3], and Odena *et al.*[16] explore some of these methods and though the papers are well-cited, we have not yet seen a comprehensive comparison of these models and their results. Looking further into the advantages and disadvantages of each of these methods could inform when and how we should choose to use conditioning labels when using GANs to complete specific tasks.

Another interesting application of GANs is stylistic image-to-image translation. An example of this is seen in Zhu *et al.*[24], where images are translated from a domain X to a domain Y and back. This framework has resulted in intriguing results for many image domains, and we wanted to investigate both the types of image data that produced quality output and the types of input that failed.

In this paper, we conduct a thorough comparison of the three conditional models discussed above, as well as a fourth model referred to as 2NGAN that utilizes labels in a way that has not been heavily documented in the literature. We compare the models conceptually (a discussion of the ways in which labels are used and what implications might result), subjectively (based on our opinion of image quality), and empirically. Benchmark datasets (MNIST[12], Fashion-MNIST[22], CIFAR10[11]) are exclusively used for the purpose of consistency and measurability. We look into factors such as training stability and quality and diversity of generated samples throughout the analysis. We also explore unpaired image-to-image translation using the CycleGAN model[24], analyzing the effectiveness of the model applied to various image domains.

We start with a problem statement (Section 2) where we outline some necessary background information and motivations for the project. Section 3 gives examples of related work in generative modeling, image-to-image translation, and the empirical analysis of GANs. In Section 4 we detail the models, data, and technologies that we used to complete our investigations. Here we provide a link to our GitHub. Section 5 shows the results of our direct comparison between established conditional GAN models and the impressive performance of our 2NGAN model, as well as a look at some of the shortcomings of the CycleGAN[24] model. Finally, we describe our overall conclusions in Section 6.

2 Problem Statement

The inspiration for our project came from an initial reading of the original GAN paper written by Goodfellow *et al.*[6]. We were drawn in by the intuitive yet intriguing training method and the impressive results. Having worked with sparse convolutional autoencoders, we noticed the vast improvement of GANs over the mean-squared-error loss method.

2.1 Background

Consider a set of real images of animals and objects. These images are defined as three matrices of RGB pixel values, which could be thought of as a point in a very high dimension. We can extrapolate this to the whole dataset and conclude that there is a high dimensional probability distribution (p_{real}) that, if we sample from it, gives us "realistic" images of animals and objects. The goal of generative modeling is to find a function with an output space (p_{fake}) that is a close approximation of p_{real}. In other words, by finding a function that maps random noise to a point (approximately) on the real data manifold, we can inspect the way in which these inputs result in different outputs to get a better understanding of the structural properties of the real data.

2.1.1 Generative Adversarial Networks

To understand the different conditional variants of GANs, it is essential to first define the "vanilla" GAN training framework. The process involves two neural networks: a generator network G and a discriminator network D. The generator network G is a parameterized function that takes an input z (usually a noise vector sampled from a random normal distribution in a relatively low dimension) and maps it to a point in the dimension of the real data. We can perform this generation process in multiple ways. The "vanilla" GAN uses a series of fully connected neural network layers to upsample, while the Deep Convolutional GAN[18] uses deep convolutional layers instead of fully connected layers. Most implementations on image data use the convolutional framework, and upsampling is often performed with "deconvolutions" (or "transposed convolutions"). The resulting generated images approximately define p_{fake} (they are essentially sampled from the "fake" probability distribution), and the goal is to make p_{fake} as close to p_{real} as possible.

We use the discriminator network D to achieve this goal. The discriminator D tells us how close it thinks p_{fake} is to p_{real} by classifying images as real or fake using a single sigmoid activated node. It is trained on labelled data to become an increasingly powerful metric that defines the distance between the fake image distribution and the real image distribution. We use binary cross-entropy as a loss function for this supervised classification problem.

The key here is that we can leverage the loss function of D to teach G to create better looking photos. G starts by generating completely random noise, which is easy for D to discriminate. However, G utilizes the gradient of the loss function of D to step in a direction that will make it more likely to fool D. Since the discriminator is optimized to differentiate between real and fake photos, "fooling" D more easily should result in more realistic photos. We are left with a game that pits two networks against each other, defined by the loss function[6]:

$$\underset{G}{\min}\underset{D}{\max} V(D,G) = \mathbf{E}_{x \backsim p_{real}}[log(D(x)] + \mathbf{E}_{z \backsim noise}[log(1 - D(G(z))] \quad (1)$$

This, of course, is just binary cross-entropy, but notice that we minimize with respect to G and maximize with respect to D. The goal is to reach Nash equilibrium, in which case D is guessing completely randomly because p_{fake} is very close to p_{real}.

Vanilla GAN and DCGAN produce impressive, high quality results. However, controlled modification of the input space z does not lead to a meaningful change in output. In other words, we cannot ask a DCGAN generator to create an image of a dog, because we do not know what subset of the input space creates dogs rather than cats, planes, or cars. Instead we would have to make many random queries and keep track of the inputs that, when fed through the generator, become dog images. Chen *et al.*[3] refer to this issue as the input space being "entangled." We can use conditional

GANs to disentangle the input to varying degrees.

2.1.2 Conditional Generative Adversarial Networks

We can assign meaning to the input space by conditioning the networks on some sort of label, c. This label is commonly a class label or a latent code and can be introduced to the network in multiple ways. The cGAN model[15] approaches this task by feeding an image's class label directly into both the generator and the discriminator. This approach essentially makes the discriminator decide both whether or not the image is real and whether or not the label-image pairing is correct. The loss function slightly changes:

$$\underset{G}{\min}\underset{D}{\max}V(D,G) = \mathbf{E}_{x\backsim p_{real}}[log(D(x|y)] + \mathbf{E}_{z\backsim noise}[log(1-D(G(z|y))] \quad (2)$$

This updated loss function simply introduces labels as extra inputs into D and G. Now we can input a random noise vector along with the "dog" class label to a trained cGAN generator to produce a synthetic image of a dog.

Different models incorporate labels in different ways. Odena *et al.*[16] use a model in which class labels are fed into G but not D, and D predicts both whether the image is real or fake, and what class the image is from. Our 2NGAN model also feeds class labels into G but not D, and D uses a softmax layer with twice as many nodes as class labels, predicting real class versus fake class. For example, if the 2NGAN model thought an image was of a dog and from p_{real}, it would predict the dog class. If it classified an image as a dog but fake, it would predict the dog class + N, where N is the number of class labels. Chen *et al.*[3] use an information theoretic approach to maximize the mutual information between latent code inputs and generated outputs. Interestingly, this approach is completely unsupervised-it does not use class labels, but instead samples latent variables from categorical and continuous probability distributions that give structure to the input space. The mutual information maximization is accomplished via a neural network Q that predicts the statistics of the distributions that the latent codes were sampled from. This results in a highly disentangled representation of the input that makes controlled modifications of the latent codes produce predictable changes in output. Figure 1 shows a visual representation of the various ways in which the models use labels. A further discussion of the different implementation methods between these models can be found in Section 4.1

2.1.3 Inception Score

Evaluation metrics are an integral part of any machine learning algorithm. Unfortunately, the task of empirically measuring the success of a GAN is very difficult. This is because the GAN objective function does not directly tell us how realistic the photos are. We could calculate the percentage of correct predictions made by the discriminator for a given training batch, but the discriminator could just be a weak classifier, leading to an incorrect conclusion that the generated images are very realistic. Another intuitive method of solving the problem of GAN evaluation is to use humans to classify p_{real} versus p_{fake} samples, but this is clearly impractical and inefficient.

One approach that has shown promising results and is widely viewed as the best current method for GAN evaluation is Inception Score[19]. This technique uses a pretrained Inception net from Szegedy *et al.*[21] to classify generated images. The idea is that the Inception net is a great classifier of images similar to the ImageNet[5] database, so when classifying generated images it should exhibit *low entropy* for individual samples and *high entropy* over all generated samples[19]. Low entropy is desirable for individual samples because this means the network is confident in its prediction, likely due to the realism of the photo. High entropy across the sample set is desirable because it indicates a high diversity of images. Salimans *et al.*[19] found that this metric correlated well with human opinion of the quality of the generated samples. As of now Inception score is the most widely used GAN evaluation metric, though it has received some criticism[1].

2.1.4 Image-to-Image Translation

For image-to-image translation, images from a domain A (e.g. black-and-white photographs) are translated to a domain B (color photographs), and images from a domain B are translated to domain A. A successful image translation function has many potential applications: real photos can be translated to an artistic style, photos of a landscape at night could be translated to daytime, etc[24][10]

We investigated CycleGAN, a model proposed by Zhu *et al.*[24] that performs *unpaired* image-to-image translation. This means that training data in one domain does not have to be paired with data in the other domain. In other words, there does not have to be a one-to-one mapping between the domains. This makes CycleGAN a far more robust and flexible model than some other paired translation models.

All of the models discussed thus far have taken a random noise vector z and mapped it to p_{fake}. However, CycleGAN does not take noise as an input. Instead, a generator takes an image from one domain and translates it to the other. There are two generators G_{AB} and G_{BA}, taking images from A to B and B to A, respectively. It also uses two discriminators, D_A and D_B, which take real and translated images in their respective domains and predict whether or not they come from the training data. CycleGAN also uses a *cycle-consistent* loss to make sure an image translated away from its domain stays structurally similar to the original. This is incorporated by sending a translated image back to its original domain, and comparing the original and reconstructed images via a Least Absolute Errors loss function. The goal of this loss function is to make sure the original image and the reconstructed image are as close to identical as possible[24].

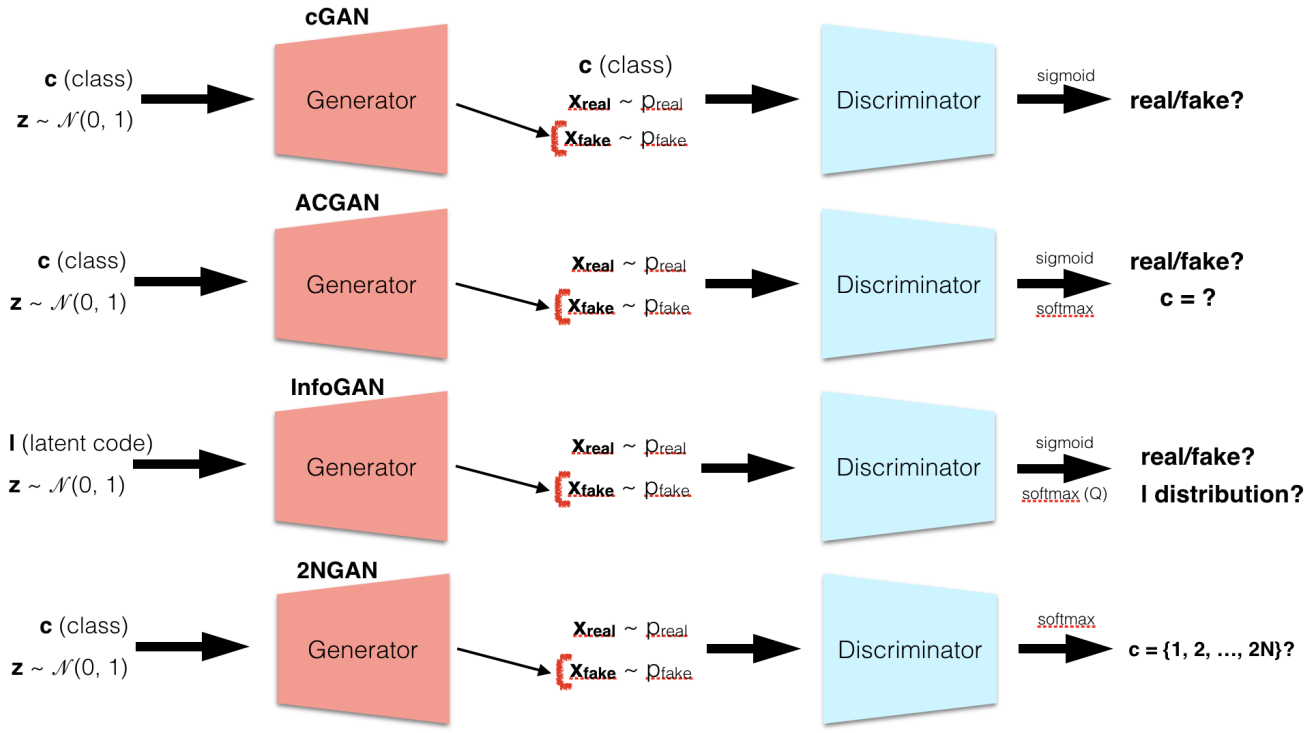


Figure 1: An illustration of some of the various ways conditional labels can be incorporated in GANs. Note: InfoGAN does not use class labels but instead uses a latent code made up of variables sampled from random distributions.

The full training process of CycleGAN is as follows:

- take domain A image, translate to domain B via G_{AB}
- train the D_B on real B and translated B image
- send translated B image back to domain A via G_{BA}
- compare reconstructed A with real A using cycle-consistent loss
- repeat above steps going in opposite direction (start with domain B)
- update generators using D loss and cycle loss

Generators learn to create fake images in the other domain that fool the discriminator, but also do not alter the original image so much that it cannot be reconstructed.

2.2 Problem and Motivation

The main motivation for the first part of our project stemmed from an observation that many GAN variants have been created and researched, some of which have much in common. We wanted to investigate the differences between some of these models and why it might be advantageous to use one approach over another. We decided to explore this problem by implementing all four conditional models (cGAN, ACGAN, InfoGAN, and 2NGAN) and comparing their results on benchmark datasets. While we attempted to use as much quantitative analysis as possible, we understood that solid empirical GAN metrics are not very common and sometimes ineffective.

For image-to-image translation, we were intrigued by the impressive and diverse results of Zhu et al.[24] and wanted to discover the limitations of the model. We implemented CycleGAN to explore the domains that it could translate reasonably well, as well as the domains that resulted in failure.

Our original deliverables for the project were substantially different from what we ended up pivoting to. We were hoping to venture into text-to-image synthesis GAN frameworks initially, but did not have a clear idea of exactly how we could make the project our own. Upon researching conditional GANs and observing the wide variety of similar models, we shifted our deliverables accordingly. Our updated deliverables were to perform a comparison between conditional GAN models and apply an image-to-image translation model to a novel data domain. The only differences between our updated deliverables and our final product are: we added a conditional model for comparison (our 2NGAN model), and we did not look at pix2pix. The reason we did not end up looking into pix2pix is detailed in Section 3.

3 Related Work

Variational Autoencoders (VAEs), proposed by Pu *et al.*[17], are the most popular generative modeling framework outside of GANs. VAEs use encoder and decoder networks to map a high-dimensional input to a low-dimensional latent representation Gaussian probability density. To learn anything meaningful about structural components of the data, these models use either spatial or sparsity driven constraints that force the decoder network to pick up on interesting properties that it can use for reconstruction. VAEs are easier

to train than GANs, however their generated samples have been observed to be blurry due to the use of mean-squared-error loss (when applied to images)[14]. VAEs have even been used in conjunction with GANs: Utilize the stable training of a VAE and create blurry images, then sharpen them and increase realism with a GAN.

There are other GAN evaluation metrics besides Inception Score that we have seen documented in the literature. Nearest-neighbors analysis can be conducted on generated images and training data to investigate whether or not the GAN is overfitting. GANs that use Wasserstein distance (WGANs) can use a Wasserstein critic to evaluate their models. Mode score and AM score are metrics that are very similar to Inception Score[2]. However, to our knowledge Inception Score has been the most effective evaluation method because it picks up on both realism and diversity of samples.

Another GAN model that performs successful image-to-image translation is proposed by Isola *et al.*[10]. This model (pix2pix) is a predecessor to CycleGAN and uses conditional labelling and a generator that maps input images from one domain to another. However, pix2pix requires paired training data, meaning there must be a one-to-one mapping between all images in each domain. This is clearly a severe limitation as it is rare that we would have training data of image pairs in two distinct domains. CycleGAN does not have this requirement, which is why we chose it over pix2pix as our model to investigate.

4 Methods

We implemented all models used in our paper as we wanted to become intimately familiar with the differences in architecture, training, and class label usage. In general, we tried to use model architectures specified in papers if provided. Though we did not perform any hyperparameter sweeps, this is something we hope to explore in the future. We implemented most models in the deep learning Python frameworks Keras and PyTorch and did not use GAN training techniques such as feature matching, label-smoothing, or minibatch discrimination to create a consistent baseline for all models[19].

4.1 Model Implementation

All of our models were implemented with a deep convolutional framework. Generator networks consisted of transposed convolutions for upsampling layers and used ReLU activation and Batch Normalization[9] layers. Discriminator networks used strided convolutions as well as Leaky ReLU activation layers (to prevent the gradient from vanishing), Batch Normalization, and Dropout[20], followed by fully connected layers for prediction. The only differences between the conditional models were network architecture and the label input methods.

4.1.1 cGAN

This framework has class labels fed directly into the generator and discriminator. To feed labels into the generator, we created a label embedding of the input class to convert the label to a vector. We used a label embedding rather than a one-hot encoding for a less sparse representation of the label. We then concatenated the embedded label vector with the noise input.

Labels could be fed to the discriminator in essentially any layer. However, concatenating upsampled label embeddings as an extra channel of the original image input resulted in the best output. We found that feeding labels into later discriminator layers resulted in more frequent mode collapse and other training failures.

Note: We refer to cGAN as cDCGAN for the remainder of the paper, as our implementation of the model uses a deep convolutional architecture.

4.1.2 ACGAN and 2NGAN

The ACGAN and 2NGAN models were highly similar in their implementations. Class labels were fed directly into the generator, but not to the discriminator. In ACGAN, the discriminator had two outputs: the first was a single sigmoid activated node for "real or fake" classification, and the second was a softmax layer with one node for each class prediction. The 2NGAN model uses only a softmax layer, but uses two nodes for each class. This allows the network to differentiate between a "real dog" and a "fake dog."

For 2NGAN, when we trained the discriminator, we fed in labels for generated images as $c + N$, where N is the number of classes. However, when we updated the generator we did not add N. This allowed the generator to step in the direction of p_{real}

4.1.3 InfoGAN

In order to maximize the mutual information between the latent code inputs and the generated data, we use a fully connected network Q to predict the statistics of the distributions that the latent codes were sampled from. We only used latent codes sampled from categorical and uniform distributions. The categorical variables could be approximated using categorical cross entropy, while the uniform distribution codes were approximated using a gaussian log-likelihood loss function that evaluates loss based on predicted mean and standard deviation.

4.1.4 CycleGAN

We modeled our CycleGAN implementation off of the official PyTorch CycleGAN GitHub repository[24]. The generators used in CycleGAN downsample the image in the first domain slightly using strided convolutions before sending the image through a series of Residual Network blocks[7], after which the image is upsampled back to the dimension of the second domain. CycleGAN discriminators use a "PatchGAN" for classification, which means that a number of patches of pixels are each evaluated as being "real or fake" rather than one sigmoid node making a single prediction for the entire image.

4.2 Data

For the conditional model comparison we used benchmark datasets MNIST[12] (black and white handwritten digits), Fashion-MNIST[22] (black and white images of clothing), and CIFAR10[11] (small color images of various animals and objects) for consistency. These data are frequently used as toy datasets in computer vision and machine learning. It is worth noting that the Inception Score metric only works on the CIFAR10 dataset, though we are interested in creating a similar method for evaluating GANs trained on MNIST. We trained models for 50 epochs on MNIST and Fashion-MNIST (we found they converged relatively quickly) and 100 epochs on CIFAR10.

We used three sets of unpaired image data for the CycleGAN analysis. One well documented application in the CycleGAN paper translated between Monet paintings and real photos using the monet2photo data[24], and we used this as a proof of concept for our implementation. We also used panda bear and brown bear synsets from ImageNet[5] in an attempt to stylistically translate between the two. Finally, we used Labelled Faces in the Wild[8] and Google's Cartoon Set compilation of thousands of cartoon avatars to try to translate real people to cartoon avatars. We expected this last application to be by far the most challenging for the network.

4.3 Comparison Metrics

All conditional models were compared in three categories: in-class variation (MNIST), training stability, and Inception Score. For in-class variation, we took large samples of the handwritten number one digit class and checked if the model picked up on the rare subset of "one digits" that were written with notches on the top and bottom. We also looked at other classes with more common variation such as "sevens" and "fours." To compare training stability we plotted the loss of each network after every training batch and viewed the amount that the network loss oscillated. We generally observed that networks with less variance in loss throughout the training procedure were less prone to training failures such as mode collapse. Finally, we used the code from Salimans *et al.*[19] to calculate Inception Score of generated CIFAR10 images for each model.

GitHub link: https://github.com/phbromley/seminar

5 Results

Our first goal was to investigate the comparative differences between four conditional GAN models. We then wanted to explore the limitations of the CycleGAN model.

5.1 Conditional Model Comparison

Figures 2 and 3 show images generated by three of the models (excluding InfoGAN) when trained on the benchmark training data (each row represents a different class). It is immediately noticeable that the generated images for MNIST and Fashion MNIST are almost completely indistinguishable from the real data distribution. Throughout our experimentation, we consistently observed that models generate realistic looking images with little trouble when trained on these datasets. The CIFAR10 results were less successful, but we thought that cDCGAN and 2NGAN did a good job producing realistic looking images. ACGAN, however, seemed to generate images lacking in clarity and diversity. The networks also seemed to generate certain classes better than others. For example, we thought the "car" class (second row) tended to look very realistic, while the cat and dog classes were poor in quality.

5.1.1 InfoGAN

This model differs from the rest in that it not only disentangles the input by separating out classes, but also by picking up on certain features of the data. Figure 4 shows the completely disentangled representation of MNIST, and the various attributes of the different styles of handwriting that the network picked up on. By conditioning the network on these latent codes we can not only query the generator for a realistic looking five, but we can also ask for the five to be heavily tilted backwards and thick.

InfoGAN had a harder time with the CIFAR10 dataset, but we thought this could have been due to an insufficient number of latent codes. For this experiment we only used one categorical code and two continuous codes, but might have had more success if we had increased those quantities. This is a potential downside to InfoGAN: it might take quite a few training runs to pinpoint the correct input composition necessary to pick up on the desired properties.

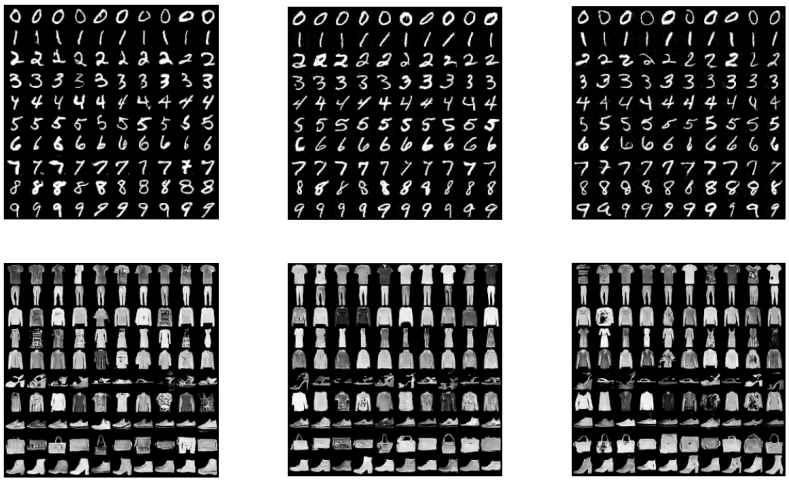
5.1.2 Training Stability

GANs are notoriously unstable models to train, and we wanted to investigate the extent to which class labels could mitigate this instability. To do this, we investigated the loss functions of each of the four models throughout their training processes. We focused on training runs on the CIFAR10 dataset for this experiment as it is much more difficult than MNIST for the models to handle.

The loss functions appeared to oscillate in all cases, which makes intuitive sense. As the discriminator strengthens its loss goes down and the generator's loss goes up. Then the generator improves and decreases its loss, generating higher quality photos and "fooling" the discriminator more often. This oscillation pattern is expected, but the degree to which it occurs clearly varies (Figure 5).

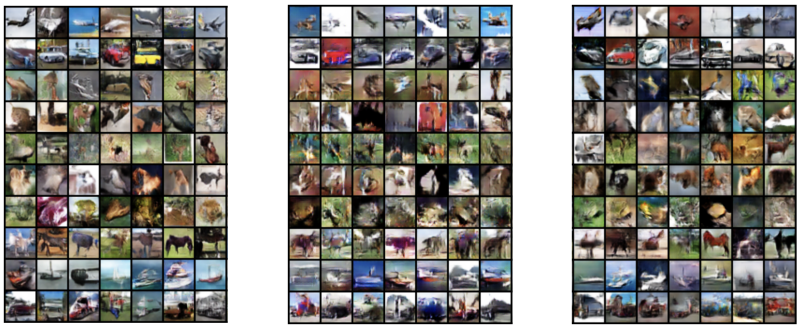
We found that our 2NGAN model has an extremely stable learning process when compared to the other frameworks. We think this could be due to the difficulty of the task for the discriminator. Since the discriminator network now only gets rewarded when it guesses the label correctly, it is prevented from getting too strong. The results from this experiment were very positive, but we remain skeptical that the model could be susceptible to mode collapse if the discriminator has a hard time learning.

The cDCGAN model exhibited heavy training oscillation and instability. This was somewhat unsurprising as we most frequently experienced training failures when experimenting with the cDCGAN model. We think this instability could stem from the directly fed label into the discriminator, as



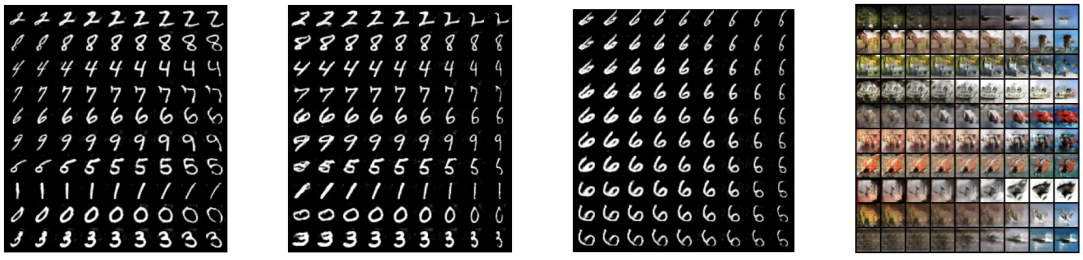
(a) cDCGAN (b) ACGAN (c) 2NGAN

Figure 2: Images generated by three of the conditional models trained on MNIST and Fashion MNIST. InfoGAN not included because it exhibits special qualities.



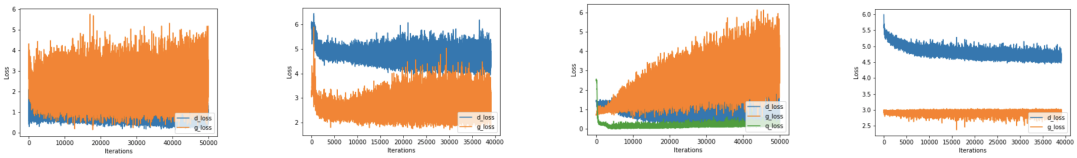
(a) cDCGAN (b) ACGAN (c) 2NGAN

Figure 3: Images generated by three of the conditional models trained on CIFAR10. Some classes are more prone to high quality generated images than others. InfoGAN not included because it exhibits special qualities.



(a) (b) (c) (d)

Figure 4: InfoGAN gives structure to the latent input using an unsupervised information theoretic approach. (a) and (b) show the effect of changing the categorical latent code (y-axis) and the first and second continuous codes (x-axis), respectively. The first code has picked up rotation while the second code has picked up width. Note that for each row in (a) and (b), the "noise vector" component of the input is held constant. (c) demonstrates the effect of holding all input constant except for the two continuous codes, which are modulated over each axis. Finally, (d) demonstrates a relatively unsuccessful application of InfoGAN to CIFAR10, though it is interesting to note that the codes seemed to pick up on color predominantly.



(a) cDCGAN (b) ACGAN (c) InfoGAN (d) 2NGAN

Figure 5: Loss functions for the four models training on CIFAR10. cDCGAN appears to heavily oscillate, while our 2NGAN model stays the most stable

this seems to be hugely advantageous for the classification network. It only has to classify real or fake photos, and now it gets extra information about what class the photo is supposed to be from. This could result in large, quick improvements by the discriminator.

5.1.3 In-Class Variation (MNIST)

We noticed that the "ones" class in the MNIST dataset was made up of mostly simple "straight-line" ones, but a few multi-stroke ones were also included. These "fancy" ones were rare so we were curious whether or not our models would pick up on this feature. We generated a large sample of ones for each model and checked for multi-stroke digits.

As shown in Figure 6, cDCGAN picks up on this variation while ACGAN and 2NGAN underfit the distribution. We also found that InfoGAN can occasionally pick up on the variation when we set the continuous latent input responsible for width to an extreme. We were unsure why cDCGAN had this advantage over the others. It is also important to note that cDCGAN tends to generate incorrect digits far more often than the other models (i.e. generating a nine when we ask for a seven). This speaks to the general instability of the model.

We also looked at sevens and fours (a subset of sevens had a notch in the middle, and a subset of fours had closed tops), but we found that all models could pick up on these variations. However, the four and seven variations were seen far more frequently in the training data than the multi-stroke "one" was.

5.1.4 Inception Score Evaluation

To quantitatively evaluate our models' performances, we used the popular Inception Score metric. For each model we generated ten-thousand images and calculated the score using an Inception net and a batch size of 10. Our 2NGAN model scored highest with a mean of 6.36 and a standard deviation of 0.17. Next was cDCGAN with a mean of 5.34 and a standard deviation of 0.12, followed by InfoGAN (mean 4.46, std. 0.14) and ACGAN (mean 3.35, std. 0.07).

We should note that this does not mean that our model is better than the others. Inception score is a disputed metric, and we did not perform any hyperparameter sweeps to optimize image quality. Additionally, ACGAN claims an Inception Score of greater than eight in their paper and we are not sure why our test was so different (we used the exact model architecture specified in their paper). That being said, at the very least this proved that our 2NGAN model showed promise and should be explored further.

Though we are unsure exactly why our model scored highly on the Inception score, we think it is because of the emphasis on class labels. The error in our model is based entirely on class prediction, while others have more of an emphasis on the general realistic nature of the image. As described in Section 2.1.3, Inception Score is a highly class-based metric that could have an easier time predicting the "class-emphasized" images generated by our model. This is not based on observational evidence, but is simply a theory that we hope to explore soon.

5.2 Unpaired Stylistic Image Transfer

We implemented and used the CycleGAN framework on three sets of training data (as described in Section 4.2). Because the two novel applications (BrownBear2Panda and Face2Cartoon) involved image domains that were slightly structurally different, we hypothesized that CycleGAN would have trouble performing the translations (especially Face2Cartoon).

Our implementation of CycleGAN did a decent job translating between Monet paintings and photos (Figure 7), though our results were not as impressive as those shown in the original paper. BrownBear2Panda (Figure 8) did not work quite as we had hoped, but the model did pick up some key components of each image domain. For example, the panda-to-brown-bear generator attempted to make the white fur of the pandas brown and also more coarse. The brown-bear-to-panda generator covered the brown bears in a smooth black and white coat. However, brown bears and pandas have somewhat significant structural differences (mainly in face shape), and the network was not able to translate these features at all. We tried decreasing the cycle-consistent loss to get around this issue but did not see great improvement.

Unfortunately, CycleGAN completely failed to perform Face2Cartoon translation. A large part of this could be due to the presence of a background in the Face domain and the lack of a background in the Cartoon domain. The networks fell into mode collapse and used the same cartoon faces for every image.

CycleGAN seems to be relatively prone to a particularly nasty failure case where information is stored in strange ways to make reconstruction or general translation easier. From a viewer's perspective this manifests itself as strange look-

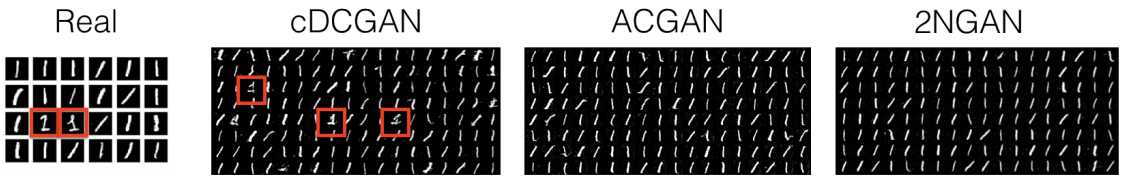


Figure 6: The "ones" class in MNIST includes rare but noticeable cases in which the top and bottom of the figure has a "notch." cDCGAN seems to pick up on this variation, while ACGAN and 2NGAN do not. However, cDCGAN is also more prone to generating digits in the incorrect class

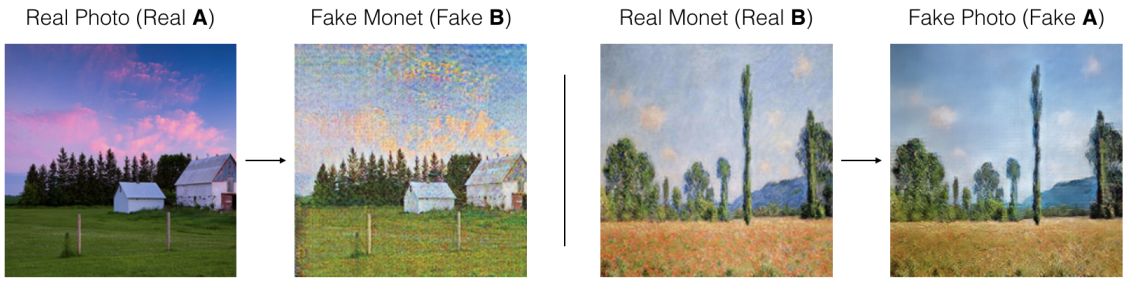


Figure 7: Our implemented CycleGAN model translating real photos to Monet paintings (left) and Monet paintings to real photos (right)

Figure 8: Our implemented CycleGAN model attempting to translate pandas to brown bears (left) and brown bears to pandas (right). The model picks up defining features such as fur quality and color, but fails to make complete compelling translations

ing noise. It was especially noticeable in the Face2Cartoon training process, as fake images could be reconstructed almost perfectly despite having lost almost all structural similarity in the initial translation. This could be a similar phenomenon to "adversarial examples," where slight perturbations to neural network inputs drastically alter the output[23].

6 Discussion

In our comparative investigation, we found that the ways in which labels are utilized in conditional GANs had a profound impact on the training process and the quality of output. The cDCGAN model impressively picked up on nuances in the training data, but also was an unstable model prone to training failures and incorrect class output. ACGAN and 2NGAN did not pick up on the same nuances that cDCGAN did, but were much more stable and reliable in their output (with 2NGAN exhibiting by far the most stability). ACGAN produced low quality CIFAR10 output, while 2NGAN boasted the best Inception Score. InfoGAN was also a relatively stable model, but most notably allowed for unsupervised representation learning and a complete "disentangling" of the latent input space.

Our 2NGAN model performed well in almost all categories of our comparison. As mentioned in Section 5.1.4, we did not conclude from these results that our 2NGAN model is better than the other models, however we do believe that it has enough potential to warrant further examination. We are slightly concerned that this label-emphasizing approach could have an unfair advantage with regards to Inception Score, though we have no empirical evidence of this. We are also worried that our approach could suffer when applied to more challenging training data due to a weak discriminator. Similarly, we are unsure how our model would perform if we used training data with many class labels. That being said, our initial results are promising and we are excited to explore the model further.

We were disappointed that we could not replicate the results (namely the Inception Score) reported in the ACGAN paper. We are not sure if this is an error on our part, but we feel as though we could be doing the ACGAN model a disservice.

If we had more time, we would've liked to optimize all models by performing a hyperparameter sweep. We are worried that some of the models are not performing up to their full capabilities due to sub-optimal architecture (we noticed slight changes to model architecture had an astonishing impact on the training process).

As for image-to-image translation, though CycleGAN allows for unpaired training data it is still heavily limited in which domains it can successfully translate between. Images must structurally be almost identical (i.e. horses and zebras), and even a slight difference can result in sub-optimal translation. CycleGAN networks also seem to search for "shortcuts" that they can use to store information in meaningless ways.

We are excited to continue working with GANs in the future. We look forward to exploring our 2NGAN model more thoroughly, as well as diving deeper into the image-to-image translation. Now that we have a solid understanding of the framework, we also hope to implement more complicated models used for text-to-image synthesis such as StackGAN. This is an exciting time to be interested in GANs, and we are excited to try to help contribute to the rapid progress.

7 Acknowledgments

I would like to thank Professor Grant as well as my CS701B classmates for guidance and suggestions. I would also like to thank the authors of papers that provided their source code in an open source format online.

8 References

[1] S. Barratt and R. Sharma. A Note on the Inception Score. *ArXiv e-prints*, Jan. 2018.

[2] A. Borji. Pros and cons of GAN evaluation measures. *CoRR*, abs/1802.03446, 2018.

[3] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. *CoRR*, abs/1606.03657, 2016.

[4] U. Demir and G. B. Ünal. Patch-based image inpainting with generative adversarial networks. *CoRR*, abs/1803.07422, 2018.

[5] J. Deng, W. Dong, R. Socher, L. jia Li, K. Li, and L. Fei-fei. Imagenet: A large-scale hierarchical image database. In *In CVPR*, 2009.

[6] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014.

[7] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[8] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst, October 2007.

[9] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.

[10] P. Isola, J. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. *CoRR*, abs/1611.07004, 2016.

[11] A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[12] Y. Lecunn. The mnist database of handwritten digits. http://yann.lecun.com/exdb/mnist/.

[13] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. P. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi. Photo-realistic single image super-resolution using a

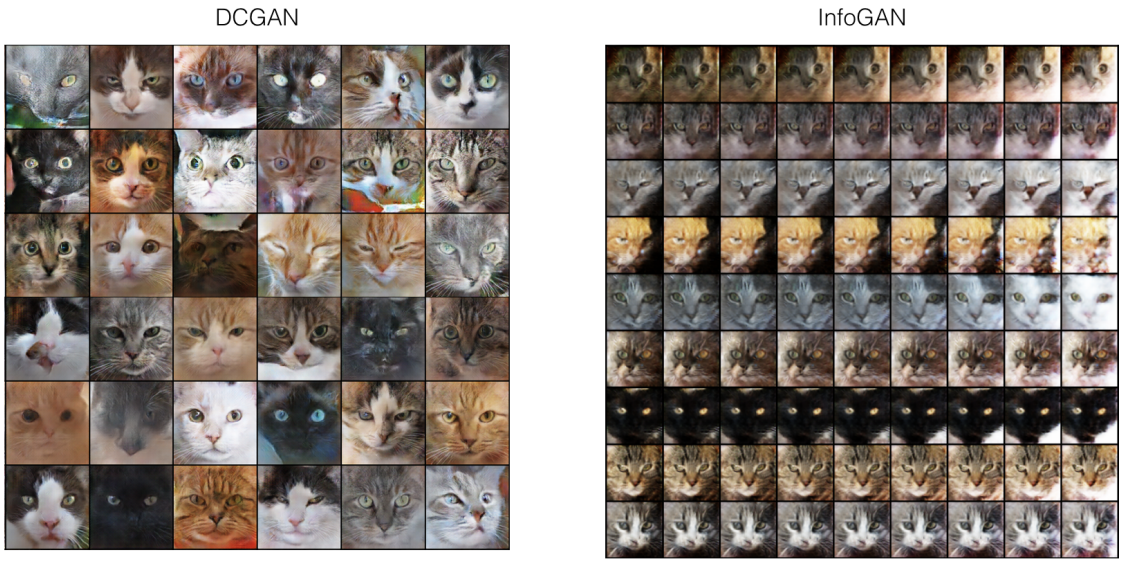


Figure 9: (Note: Not part of the official project goals) An experiment with high-resolution, unlabeled cat faces. Left: DCGAN after 200 epochs, Right: InfoGAN with one categorical variable (y-axis) and two continuous variables (the first code is modulated on the x-axis) after 100 epochs. Like the InfoGAN CIFAR example, the model seemed to pick up predominantly on color. However we probably should have used a larger latent code input sequence.

Figure 10: (Note: Not part of the official project goals) Linear interpolation through the latent space of the DCGAN input. Some of the interpolations (third and fourth row) do not look very smooth. We wonder if this is because we are overfitting, or because we are using linear interpolation rather than "great circle" interpolation.

generative adversarial network. *CoRR*, abs/1609.04802, 2016.

[14] L. M. Mescheder, S. Nowozin, and A. Geiger. Adversarial variational bayes: Unifying variational autoencoders and generative adversarial networks. *CoRR*, abs/1701.04722, 2017.

[15] M. Mirza and S. Osindero. Conditional generative adversarial nets. *CoRR*, abs/1411.1784, 2014.

[16] A. Odena, C. Olah, and J. Shlens. Conditional Image Synthesis With Auxiliary Classifier GANs. *ArXiv e-prints*, Oct. 2016.

[17] Y. Pu, Z. Gan, R. Henao, X. Yuan, C. Li, A. Stevens, and L. Carin. Variational autoencoder for deep learning of images, labels and captions. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 2352–2360. Curran Associates, Inc., 2016.

[18] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *CoRR*, abs/1511.06434, 2015.

[19] T. Salimans, I. J. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. *CoRR*, abs/1606.03498, 2016.

[20] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958, Jan. 2014.

[21] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Computer Vision and Pattern Recognition (CVPR)*, 2015.

[22] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *CoRR*, abs/1708.07747, 2017.

[23] X. Yuan, P. He, Q. Zhu, R. R. Bhat, and X. Li. Adversarial examples: Attacks and defenses for deep learning. *CoRR*, abs/1712.07107, 2017.

[24] J. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *CoRR*, abs/1703.10593, 2017.