Semi Supervised Learning using GANs

Akash Kulkarni
Arun Sethupat Radhakrishna

Final Project Report

November 2017

1 Abstract

We have seen that Generative Adversarial Networks have been of great use in solving varied set of problems. In this project we analyzed three types of GANs: vanilla GANs, CGAN and InfoGAN and realized their limitations and tried to come up with a new architecture to solve it. C-GANs requires the conditional codes to be sent to the model during the training. InfoGAN learns the inherent latent codes automatically in an unsupervised way but they are not interpretable. We combined the idea of CGANs and InfoGANs to come up with an architecture to learn interpretable latent codes.

2 Introduction

Deep learning has made immense progress in the recent years. Using these techniques, we have been able to perform various tasks that a human can do ranging from recognizing images and voices to enabling driving cars on their own.But the goal of automating human tasks is still a far stretched goal. There are many tasks like composing music or drawing paintings like an artist that a human can do which is tough to achieve using traditional deep learning techniques.

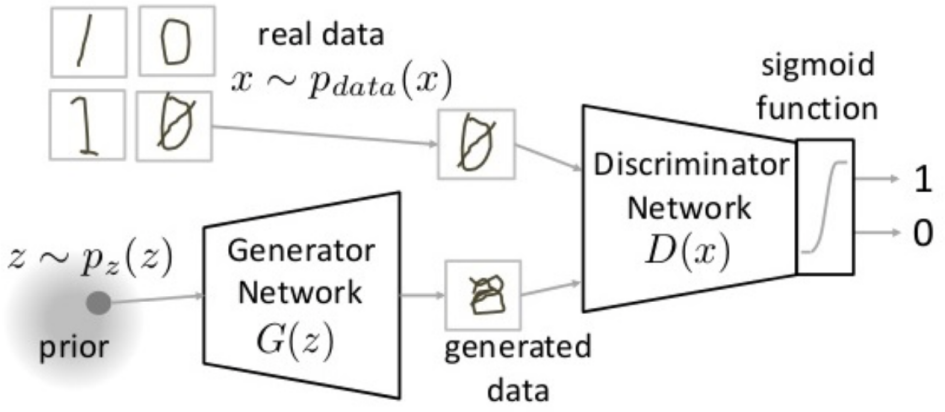
But now using Generative Adversarial Networks we can solve many such problems. We wish to provide a brief introduction of the different types of GANs we are using for this project.

2.1 Generative Adversarial Networks

Generative Adversarial Networks, first introduced in a NIPS 2014 paper by Ian Goodfellow, et al, is an interesting neural network for generating input data by learning its probability distribution.

2.1.1 How GAN works

GAN is a special case of Adversarial Process in which there are two component neural networks, namely Generator and Discriminator, who play the role of adversaries.



2.1.2 Generator Network

The Generator Network takes a random noise as input and generates a sample of data. In the above image, we can see that generator $G(z)$ takes a input z from $p(z)$, where z is a sample from probability distribution $p(z)$. It then generates a data which is then fed into a discriminator network $D(x)$.

2.1.3 Discriminator Network

The Discriminator Network takes input either x from the real data distribution $p_{data}(x)$ or from the generator $G(z)$ and try to predict whether the input is real or generated. $D(x)$ then solves a binary classification problem using sigmoid function giving output in the range 0 to 1 indicating the probability of x being real.

2.1.4 Objective Function

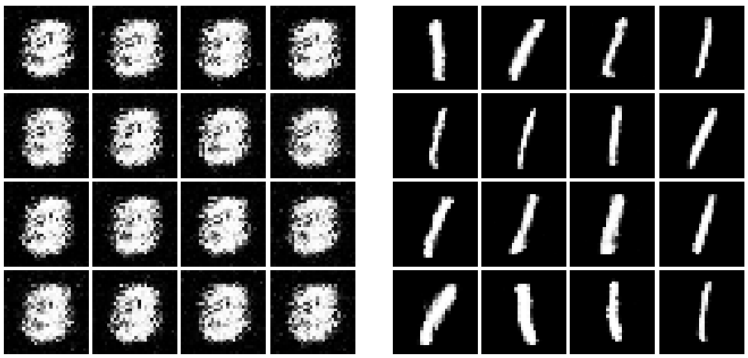
We try to minimize the following objective function.

$$min_G max_D V(D,G) = E_{x\ P_{data}(x)}[logD(x)] + E_{z\ P_z(x)}[log(1 - D(G(z)))]$$

2.1.5 Results

We implemented Vanilla GAN with the following parameters. We have performed the experiment with 2 different optimizers.

- Number of Epochs : 200,000
- Optimizer: GradientDescentOptimizer
- Learning Rate: 0.01



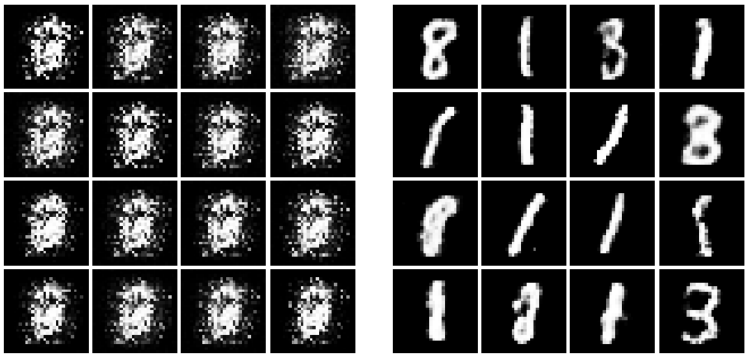
Images being generated at 1st iteration and 200,000th iteration.

The loss values after 200000 iterations are:

- Generator Loss: 2.805
- Discriminator Loss: 0.2714

We also tried the experiment with a different set of parameters. Following are the experimental details:

- Number of Epochs : 200,000
- Optimizer: Adam Optimizer
- Learning Rate: 0.01



Images being generated at 1st iteration and 20,000th iteration.

The loss values after 200000 iterations are:

- Generator Loss: 1.982
- Discriminator Loss: 0.5656

2.1.6 Observation

We make a couple of observations here.

- We notice that the images start getting crisper at a faster pace with AdamOptimizer over GradientDescentOptimizer. This is because in adamoptimizer some fraction of the previous update is added to the present update. Adam Optimizer also uses an adaptive learning.(Selects a learning rate each parameter)
- We also notice that the images that are being generated are completely random and are not directly related to the given specific input. i.e. For a same given input the network might generate images belonging to different classes.

Henceforth for all the experiments we would use the experimental setting with AdamOptimizer.

2.2 Conditional GAN

Conditional GANs are conditional version of generative adversarial nets where can generate samples of data based on a condition. In this, we simply feed both the data and the condition to both the generator and discriminator.

2.2.1 How CGAN works

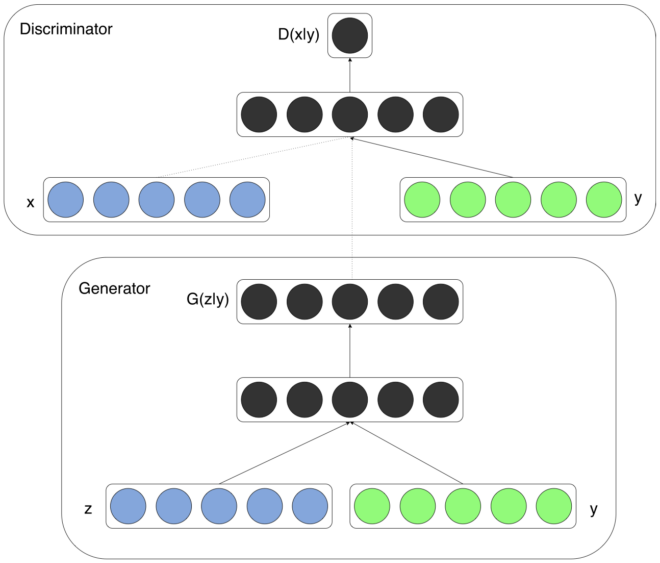
Given an extra information y (which could be any kind of information like class labels or data from other modalities) that we want to condition our model on, we feed y into both Generative and Discriminator networks.

2.2.2 Generator Network

The Generator Network takes a random noise as input and the auxiliary information y which are combined in joint hidden representation and generates a sample of data. In the above image, we can see that generator $G(z|y)$ takes a input z and y, where z is a sample from probability distribution $p(z)$ and y is the variable that we want to condition on. It then generates a data which is then fed into a discriminator network $D(x|y)$.

2.2.3 Discriminator Network

The Discriminator Network takes input either x from the real data distribution $p_{data}(x)$ or from the generator $G(z, y)$. Along with it also take y as an additional input and try to predict whether the input is real or generated. $D(x|y)$ then solves a binary classification problem using sigmoid function giving output in the range 0 to 1 indicating the probability of x being real.



2.2.4 Objective Function

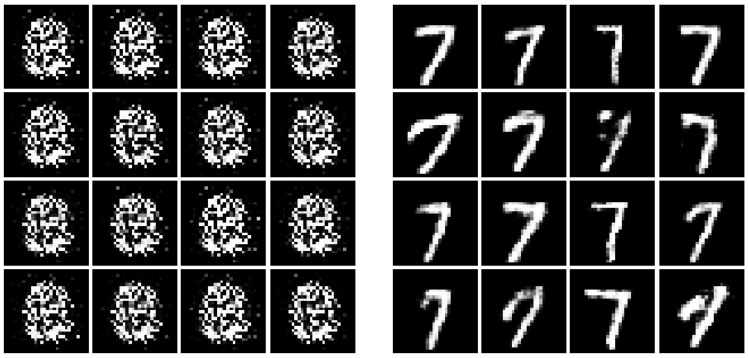
We try to minimize the following objective function.

$$min_G max_D V(D, G) = E_{x\ P_{data}(x,y)}[logD(x, y)] + E_{z\ P_z(x,y)}[log(1 - D(G(z, y)))]$$

2.2.5 Experimental Results

We implemented the above mentioned experiment with the following parameter settings.

- Number of Epochs: 200,000
- Optimizer: Adam Optimizer
- Learning Rate: 0.01
- Number trying to be generated: 7



Images being generated at 1st iteration and 200,000th iteration.

The loss values after 200000 iterations are:

- Generator Loss: 1.913
- Discriminator Loss: 0.591

2.2.6 Observation

We make a couple of observations here.

- We observe that the data points to be generated can be controlled by passing the conditional along with the distribution. This is an advancement over Vanilla GANs where there is no control over output.
- We also observe that we are able to generate the number we wish to generate.

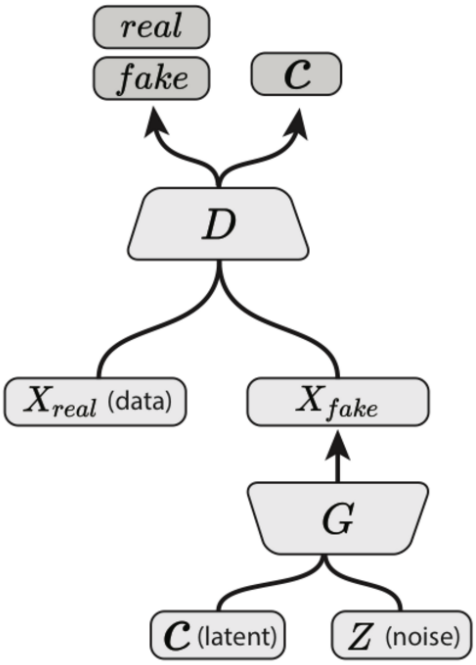
2.3 InfoGAN

The natural extension of GAN was to learn a conditional generative distribution, that can learn any class label or another images as context. However, we need to provide those conditionals manually, somewhat similar to supervised learning. InfoGAN, therefore, attempted to make this conditional learned automatically, instead of telling GAN what that is.

InfoGAN is a generative adversarial network that maximizes the mutual information between a small subset of the latent variables and the observation. This information-theoretic extension to the Generative Adversarial Network learns the disentangled representations in a completely unsupervised manner.

2.3.1 How InfoGANs work?

In addition to generator and discriminator network, we train an additional network, denoted by Q, to compute the mutual information. The architecture is given as follows:



2.3.2 Objective Function

We try to minimize the following objective function.

$$I(c, G(Z, c)) = E_{c\ P(c),x\ G(z,c)}[logQ(c|X)] + H(c)$$

$$min_G max_D V(D, G) - \lambda I(c, G(Z, c))$$

2.3.3 Generator Network

The Generator Network takes a random noise as input and the latent code c, sampled from some distribution $P(c)$ and generates a sample of data. In the above image, we can see that generator $G(z|y)$ takes a input z and y, where z is a sample from probability distribution $p(z)$ and y is sampled from $P(c)$. It then generates a data which is then fed into a discriminator network $D(x)$.

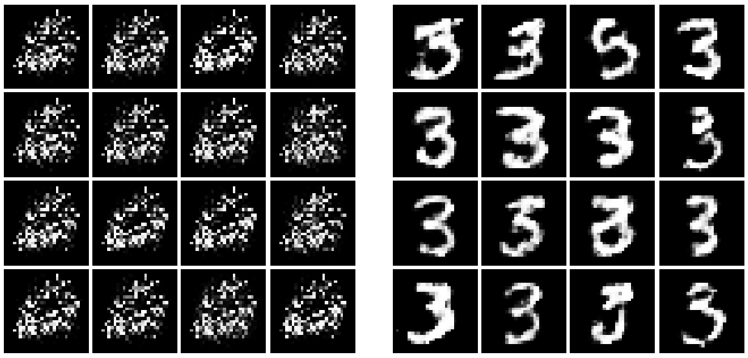
2.3.4 Discriminator Network

The Discriminator Network takes input x either from the real data distribution $p_{data}(x)$ or from the generator $G(z, y)$. $D(x)$ then solves a binary classification problem using sigmoid function giving output in the range 0 to 1 indicating the probability of x being real.

2.3.5 Experimental Results

We implemented the above mentioned experiment with the following parameter settings.

- Number of Epochs : 200,000
- Optimizer: Adam Optimizer
- Learning Rate: 0.01
- Number trying to be generated: 7



Images being generated at 1st iteration and 200,000th iteration.

The loss values after 200000 iterations are:

- Generator Loss: 1.964
- Discriminator Loss: 0.5386

2.3.6 Observation

We make the following observations here.

- InfoGANs can successfully learn the class labels even without mentioning the classes effectively
- InfoGANs have the problem of interpretability as the classes generated can not be interpreted by humans.
- We observed that the generated number was 3 when we passed 7 as the latent variable y

3 Semi Supervised Learning with GANs

InfoGAN provide no means of exerting control on what kind of features are found. Disentangled data representations that are extracted are not always directly interpretable by humans and lack direct measures of control due to the unsupervised training scheme. Conditional GAN provides control over the data points being generated. However, it requires us to send the label explicitly with every input.

To provide control over what factors are learned/identified , we take a semi-supevised learning approach. We exploit the already existing labels from the dataset to force the disentangled representations to be learned in such a way that it corresponds to the class labels.

3.1 Hypothesis 1

In order to accomplish the task of performing semi-supervised learning with GANs, we hypothesize that if we use the infoGAN architecture and pass on labels we have in place of random noise for the supervised data and random noise otherwise , the labels would be interpretable.

3.1.1 Objective Function

$$SupervisedLoss = E_{c\ c_{sup},x\ G(z,y)}[log(Q(c|x))] + H(c)$$

$$UnsupervisedLoss = E_{c\ c_{us},x\ G(z,c)}[log(Q(c|x))] + H(c)$$

$$Loss = UnsupervisedLoss + SupervisedLoss$$

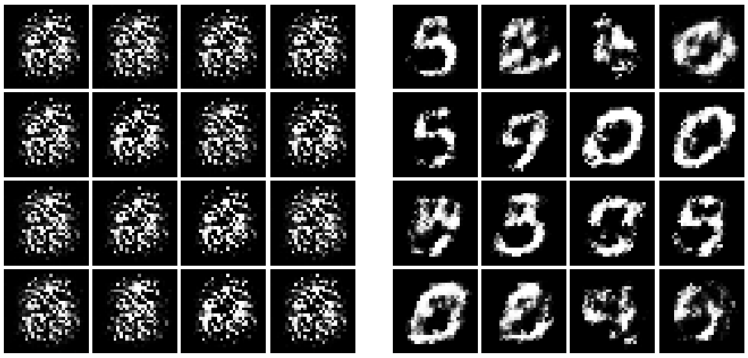
The final Objective Function:

$$min_{G,Q} max_D V_{infoGAN}(D,G) - \lambda Loss$$

3.1.2 Experimental Results

We implemented the mentioned experiment with the following parameter settings. We used 10% of the data as supervised and remaining data as unsupervised.

- Number of Epochs : 20,000
- Optimizer: Adam Optimizer
- Learning Rate: 0.01



Images being generated at 1st iteration and 20,000th iteration.

The loss values after 20000 iterations are:

- Generator Loss: 1.882
- Discriminator Loss: 0.6656

3.1.3 Observation

We observed that the results were not as expected. We understood that the reason for the failure of this experiment could one of the following reasons

- The amount of supervised data that is being passed is low. We passed 5% data as supervised data.
- The other reason could be that after passing the supervised data with labels, the unsupervised data is diverging away from the original class values. This could explain the reason why the images initially start getting crisper and then the images got blurrier.

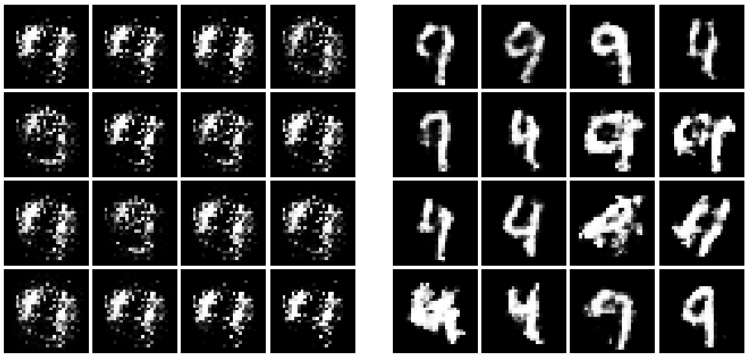
3.2 Hypothesis 2

We perform the following experiment inorder to overcome the problem of lack of sufficient data for supervised learning. We hence increase the percentage of supervised learning data to 25%.

3.2.1 Experimental Results

We implemented the mentioned experiment with the following parameter settings. We used 10% of the data as supervised and remaining data as unsupervised.

- Number of Epochs : 200,000
- Optimizer: Adam Optimizer
- Learning Rate: 0.01



Images being generated at 1st iteration and 20,000th iteration.

The loss values after 20000 iterations are:

- Generator Loss: 1.882
- Discriminator Loss: 0.956

3.2.2 Observation

We observed that even after increasing the supervised data percentage to 10%, the results are still not as expected. We do not get crisper images.
Since for supervised data, we were trying to maximize the mutual information between synthesized x and label y (considered as latent code) and for unsupervised data, maximizing the mutual information between synthesized x and random noise , the efficiency was not enough. Thats why images generated are neither crisper nor it corresponds the label that we pass to generate

3.3 Hypothesis 3

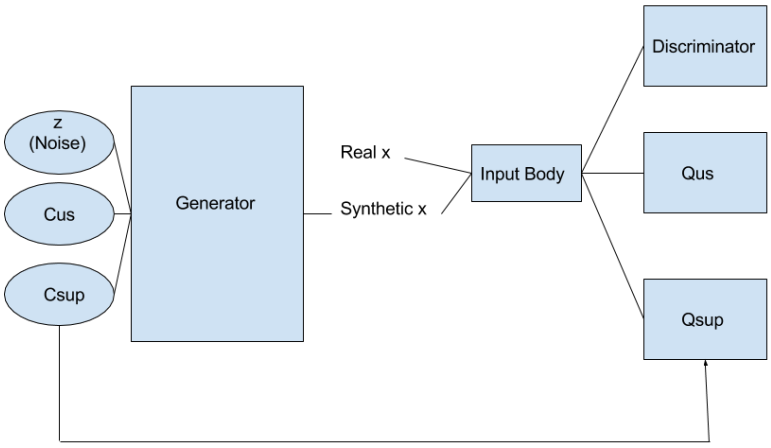
We perform the following experiment to overcome the issues we faced in the Hypothesis 1. In this approach, apart from maximizing the mutual information between a code and synthetic sample we do two more things. For supervised data set:

- We maximize the mutual information between the class label (acting as latent code) and real data sample X. This increases the interpretability of the learned latent code representations.
- We maximize the mutual information between the class label and synthetic data sample X. Since the generator and discriminator tries to map the generated data sample X to real data sample X, maximizing the mutual information between the sample class label and X (which is supposed to be closer to X) increases the interpretability of the learned latent code even more.

For unsupervised data set:

- We follow the vanilla InfoGAN architecture to maximize the mutual information between the latent code and synthetic samples and also tries to close the gap between the distribution that Generator learns and real distribution.

3.3.1 Architecture



3.3.2 Objective Function

We try to minimize the following objective function.

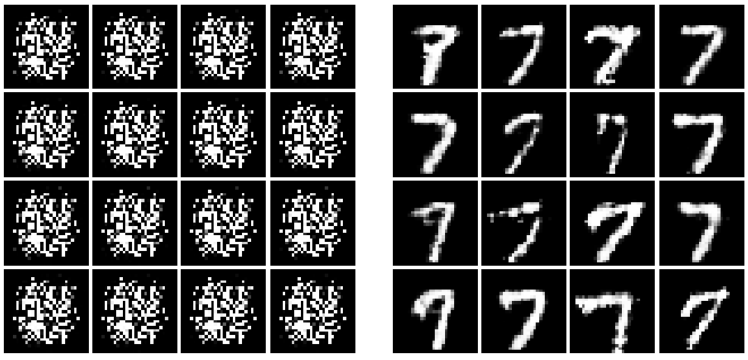
$$SupervisedLoss = E_{c\ c_{sup},x\ P_{data}}[log(Q_{sup}(c|x))] + H(c)$$
$$UnsupervisedLoss = E_{c\ c_{us},x\ G(z,c)}[log(Q_{us}(c|x))] + H(c)$$
$$Loss = UnsupervisedLoss + SupervisedLoss$$

The final Objective Function:

$$min_{G,Q_{us},Q_{sup}} max_D V_{infoGAN}(D,G) - \lambda Loss$$

3.3.3 Experimental Results

- Number of Epochs : 20,000
- Optimizer: Adam Optimizer
- Learning Rate: 0.01
- Number to be generated: 7



Images being generated at 1st iteration and 200,000th iteration.

The loss values after 200,000 iterations are:

- Generator Loss: 1.245
- Discriminator Loss: 0.8356

3.3.4 Observation

The following are the observations we could make from this experiment.

- When we passed digit 7, we were able to generate the corresponding label's image.
- Images are clearer compared to InfoGAN.
- Compared to previous hypothesis, since now we try to maximize the mutual information between real x and real y also, the intrepretability of the latent code generated got better

We believe that

- Passing the class labels instead of random noise in the case of supervised data enables the model to enhance the intrepretability of the generated latent code
- The unsupervised dataset is helping the model to learn the real class distribution efficiently
- Combining both, we were able to generate a model which can generate better images corresponding to the class label.

4 Conclusion

In our experiment, we tried to develop an architecture which can be used to perform semi-supervised learning using GANs which can generate crisper images quickly. We hypothesised a method which allowed us to use InfoGAN architecture to pass on the label information for the supervised dataset and random noise for the unsupervised dataset. However, this did not work. But on perusal of the results, we could underpin two important drawbacks of our architecture. On making changes which can overcome the drawbacks, we observed that we were able to leverage semi-supervised learning to generate the images based on the condition (eg. class label) that we passed . We could also generate crisper images in comparison to InfoGAN.