# Generative Adversarial Networks

COMP 551 - Mini Project 4

**Track 2**

Ameya Bhope (260849407) Amanpreet Walia (260834477)
Surya Kumar Devarajan (260815492)

April 2019

**Abstract**

Generative adversarial networks (GANs) are deep neural net architectures comprised of two nets - Generator and Discriminator, pitting one against the other. The generator generates new data instances and Discriminator evaluates them for authenticity. The discriminator decides whether each instance of data that it reviews belong to the actual training dataset or not. The primary goal of the generator network is to maximize the likelihood that its counterpart misclassifies its output as real whereas discriminator optimizes towards a goal distinguishing between real and generated images, till both reach an equilibrium. In our framework, the generator will start to train alongside with discriminator. Both Generator and Discriminator can be trained alongside with backpropagation where the loss function provides the stopping criteria for the Generator and Discriminator training process. In our work, we have shown how GANs performed on various image datasets with convincing evidence that the adversarial model learned the latent representations of the scenes/objects in the training images. Additionally, we focussed towards improving the stability of GAN training by using WGANs.

***Keywords***— Generative Adversarial Network, Wasserstein GAN, Deep Convolutional GANs

## 1 Introduction

With machine learning becoming popular in various fields, the need for data is ever-increasing. There are many niche fields like medical imaging, drug testing, etc. where the availability of data in the public domain is very scarce. Hence, there arises a need for techniques which could generate more data, on the basis of existing data.

To address this issue of limited data availability, many schemes for generative modeling like Variational Auto-Encoders (VAEs), Restricted Boltzmann Machines, Pixel RNN, etc. have been proposed before the arrival Generative Adversarial Networks(GANs). However, images generated by these approaches tend to be more blurred, whereas GANs generate data in fine, granular detail. GAN's have shown great results in numerous generative tasks such as images, human language, and music. GAN architecture consists of two networks - Generator and Discriminator, pitted against each other playing the zero-sum non-cooperative game where both networks try to defeat each other till they reach Nash equilibrium, which is a stable state of a system involving the interaction of different participants, in which no participant can gain by a unilateral change of strategy if the strategies of the others remain unchanged . In a nutshell, GANs are a set of generative models, which can learn to generate data (ideally) identical to given data distribution. GANs can be used to learn reusable feature representations from large unlabeled datasets e.g. in computer vision, one can leverage the practically unlimited number of unlabeled images and videos to learn good intermediate representations, which can then be used on a variety of supervised learning tasks such as image classification.

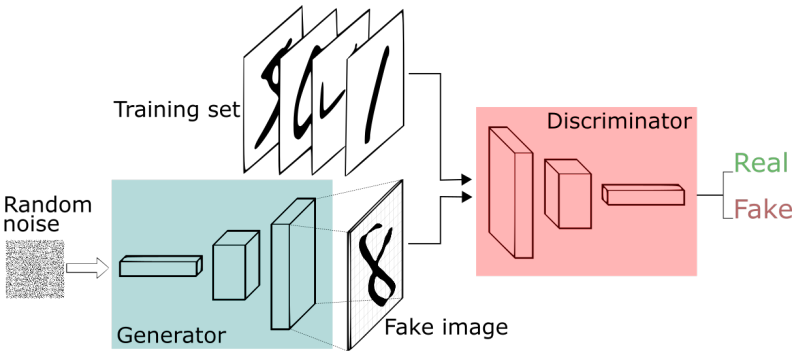



Figure 1: The objective of the generator is to model or generate data that is very similar to the training data and the objective of Discriminator is to identify if the data is real or fake. The Discriminator is trained on two sets of input. One input comes from the training dataset and the other input is the modeled dataset generated by Generator[Image]

For this project, we have applied GANs on various publicly available datasets to see how well the artificially generated images are close to the original dataset. Most of this work is identical to the work by original authors in the paper. In

addition, we also inculcated some important details from recent GAN papers to improve the training process. Lastly, we outline and discuss our key takeaways and propose some of the possible improvement techniques, in this report.

## 2 Related Work

Since the introduction of GAN's by Ian Goodfellow [1] in 2014, there has been a lot of work related to it. This includes new variants of GAN's focusing on various applications as well as improving the stability of training and the resulting perceptual quality of GAN samples. In addition to replicating the work of authors in the original paper, we build on some of these techniques for our project. For instance, we implemented DCGAN architectural innovations proposed in Radford et al. [2]. This paper discusses several empirical techniques that worked well in practice, like batch normalization, normalizing the input image and choice of the activation function. Another important technique called minibatch discrimination is proposed by Salimans et.al.[3]. The idea is to introduce a layer that operates across samples to introduce coordination between gradients from different samples in a mini-batch.

An alternative training process named SGAN [4] uses several adversarial local pairs of networks trained independently so that a global supervising pair of networks can be trained against them. The goal is to train the global pair with the corresponding ensemble opponent for improved performances in terms of mode coverage. This approach aims at increasing the chances that learning will not stop for the global pair, preventing both to be trapped in an unsatisfactory local minimum, or to face oscillations often observed in practice. To guarantee the latter, the global pair never affects the local ones.

Probabilistic Generative Adversarial Network (PGAN) [5], a new GAN variant based on a new kind of objective function. The central idea is to integrate a probabilistic model (a Gaussian Mixture Model, in our case) into the GAN framework which supports a new kind of loss function (based on likelihood rather than classification loss), and at the same time gives a meaningful measure of the quality of the outputs generated by the network. Experiments with MNIST show that the model learns to generate realistic images, and at the same time computes likelihoods that are correlated with the quality of the generated images.

In order to improve the stability of the training process, Arjovsky et. al. used Wasserstein distance as the objective in Wasserstein GAN (WGAN) [6] work. This new objective has non-zero gradients everywhere. The implementation is as simple as removing the sigmoid function in the objective and adding weight clipping to the discriminator network. WGAN is shown to be free of the many problems in the original GAN, such as mode collapse and unstable training process. Related work to WGAN is Loss-Sensitive GAN [7], whose objective is to minimize the loss for real data and maximize it for the fake ones. We have tried to reproduce some of this work in our project as well.

SRGAN (Super-Resolution Generative Adversarial Network) [8] is another important application of GANs, which addresses the issue of recovering finer texture details in an image in an attempt to improve the image's resolution, which even with breakthroughs in Deep convolutional networks, is not able to combat.

## 3 Dataset and Setup

We used 4 datasets for this project.

1. **MNIST** - One of the most popular datasets used in Machine learning is MNIST dataset. It consists of handwritten digits of 60,000 examples. It is a subset of a larger set available from NIST dataset. The digits have been size-normalized and centered in a fixed-size image.
2. **CIFAR10** - The CIFAR-10 dataset contains 60,000, 32x32 color images in 10 different classes. The 10 different classes represent `airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks`, with 6,000 images of each class.
3. **Flower Dataset**[DownloadLink] - We have used the Visual Geometry Group's 102 category Flower Dataset[12] having 8,189 images spread over 102 categories from Oxford University. The images have large scale, pose and light variations. In addition, there are categories that have large variations within the category and several very similar categories.
4. **Fashion-MNIST** - It is a dataset of Zalando's article images consisting of a training set of 70,000 examples. Each example is a 28 x 28 gray-scale image, associated with a label from 10 classes. Fashion-MNIST is intended to serve as a direct drop-in replacement of the original MNIST dataset for benchmarking machine learning algorithms.

For implementing the models, we used PyTorch and Keras with Tensorflow backend. For computing resources, we have used Google Colab to do this project. Although the original paper's code is written in Theano, we have implemented our work in PyTorch and Keras, due to easier interface and familiarity.

## 4 Background and Proposed Approach

The GAN architecture is inspired by Game Theory, where two neural nets, generator and discriminator compete against each other while making each other strong at the same time. Thus, the discriminator requires training for a few epochs before it actually starts giving meaningful results while classifying the images generated by the Generator network. Lastly, a loss function provides a stopping criteria for the generator and discriminator training process. For the discriminator, its objective is to maximize the probability of getting a correct output, that is, classifying correctly, whether the image is fake or real.

The generator will produce the images or output that we see after this entire training process is complete. When we talk about training GANs, it refers directly to training the generator. The Generator will take the random noise data $z$ and tries to reconstruct the input $x$.

The discriminator act as an adaptive loss function for the GAN. Discriminator classifies input as real or fake. Based on the classification, classification error is computed. Ultimately, the discriminator is going to evaluate the output of the real image and the generated image for authenticity. The real images will score high on the scale initially, while the generated images will score lower. Eventually, the discriminator will have trouble distinguishing between the generated and real images. The discriminator will rely on building a model and potentially an initial loss function.

GAN's need to optimize the minimax objective function.

$$\min_G \min_D V(D,G) = E_{xP_{data}}[logD(x)] + E_{zP_z(z)}[log(1 - D(G(x)))]$$

where, $[logD(x)]$ - Discriminator output for real data $x$ , $D(G(x))$ - Discriminator output for generated fake data G($x$)

The Generator G generates a probability $P_g$ as distribution of the samples $G(z)$ obtained from $z$. To learn the generator's distribution $P_g$ over data $x$, we define a prior on input noise variables $P_z(z)$. $D(x)$ represents the probability from the training data $x$ rather than $P_g$. The Discriminator wants to maximize objective such that $D(x)$ is close to 1 for real data and $D(G(z))$ is close to 0 for fake data. Generator wants to minimize objective such that $D(G(z))$ is close to 1 so that the discriminator is fooled into thinking generated $G(z)$ is real. We stop the training when the fake data generated by the Generator is recognized as the real data.

GAN models have had a tremendous success in the domain of image processing, e.g., for generating super-resolution photo-realistic images from text [8], face aging images in entertainment [9], blending of objects from one picture into the background of another picture, as well as in other applications, such as generating hand-written text, and music sequence generation [10].

## 4.1 Deep Convolutional GANs (DCGANs)

One of the most popular variants of GAN's is the Deep Convolutional GAN. It functioning is very similar to GANs, but specifically focuses on using Deep Convolutional Networks in place of Fully Connected Networks. Convolutional Nets, in general, find areas of correlation within an image, that is they look for spatial connections. Thus, for image or video data, DCGAN would be more fitting, whereas the general implementation of GAN (also referred to as Vanilla GAN), can be applied to wider domains.

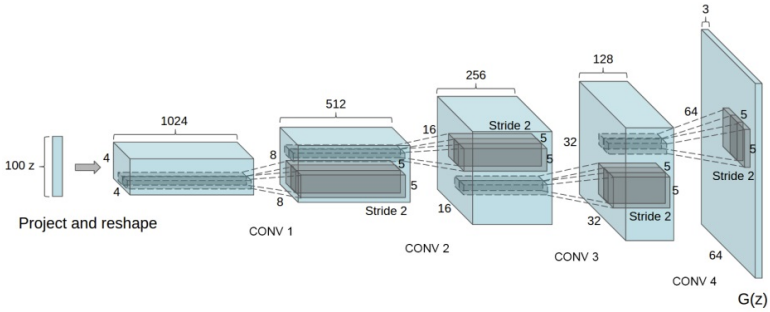



Figure 2: Design of DCGAN generator [2]

DCGAN introduces several constraints and modifications to the original GAN architecture for improved stability and performance. In DCGAN, the generator and discriminator are composed of multiple layers of convolutional computational units, as opposed to the multilayer perceptron networks proposed in the original GAN paper [1].

First, the network structure in DCGANs replaces pooling layers with strided convolutions, which allows the sub-networks to adjust the spatial down-sampling and up-sampling based on the input data. Second, it eliminates fully connected layers that are commonly used after convolutional layers in deep Neural Networks, and it relies solely on convolutional layers. Third, batch normalization is applied to all layers, except to the output layer of the generator and the input layer of the discriminator. Fourth, ReLU activation function is used for all layers in the generator, except for the last layer where a Tanh activation function is applied. For the discriminator, leaky ReLU activation function is suggested for all layers.

By applying the above recommendations, improved classification performance has been demonstrated on various datasets of images, and capabilities of generating complex and visually realistic images.

## 4.2 Wasserstein GAN

Wasserstein GANs (WGANs) [6] introduce a new loss function for training the generator and discriminator subnetworks. The Wasserstein distance is the minimum cost of transporting mass in converting the data distribution q to the data distribution p. The loss function is based on this Wasserstein distance between the real data distribution $P_r$ and the model distribution $P_g$ learned by the generator.

$$W(P_r, P_g) = \inf_{\gamma \in \pi(P_r, P_g)} E_{(x,y)\sim\gamma}[x - y].$$

In the above equation, $\pi(P_r, P_g)$ denotes the set of joint distributions $\gamma(x, y)$ whose marginals are $P_r$ and $P_g$. In simpler terms, $\gamma(x, y)$ defines the amount of earth mass that needs to be moved from a point $x$ to a point $y$ in order $P_r$ and $P_g$ to be identical.

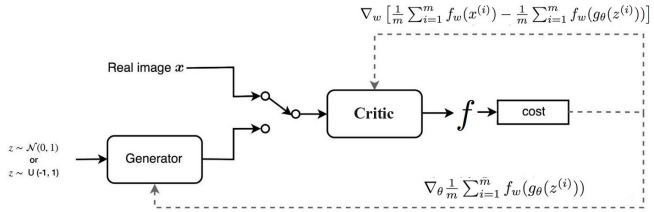



Figure 3: Network Design of WGAN [Image]

In the usual GAN, we want to maximize the score of classification. If the image is fake, the discriminator should give it as 0 scores; if the image is a real one, the 1 score should be gotten. In WGAN, it changes the task of discriminator as regression problem, and it is renamed as critics as shown in figure 3. The critics should measure the EM-distance that how many works should spend, and find the maximum case. WGAN can avoid the problem of gradient vanishment.

Based on the GAN paper, we first tested the MNIST dataset. We followed the default hyperparameter values, as specified in the original paper to create our first model. To understand how the latent variable dimensions affect the GAN's performance, we used different values of z which are 10, 100, 200. We also tested on the Fashion MNIST, with the value of z as 100. We tested the DCGAN architecture on CIFAR 10, Flower and MNIST dataset. For MNIST dataset, we used the default network as well as increase the depth of the network to 1024. We tested the performance of different optimizers like Adam, RMSprop, SGD, and AdaDelta. For WGAN, we tested on MNIST dataset with default parameters.

In GAN, the loss measures how well it fools the discriminator rather than a measure of the image quality. On the contrary, WGAN loss function reflects the image quality which is more desirable and it will be justified in results.

# 5 Results

In this section, we present the results of our experiments mentioned below: [1].

1. Replicating the work of Authors in Original Paper. (Experiment A)
2. Effect of changing dimension of latent variable on generated images. (Experiment B)
3. Using GANs(and variants) on new datasets.(Experiment C)
4. Performance of different optimzers with DC-GANs.(Experiment D)
5. Stabilizing the loss of networks using WGANs. (Experiment E)

## 5.1 Experiment A

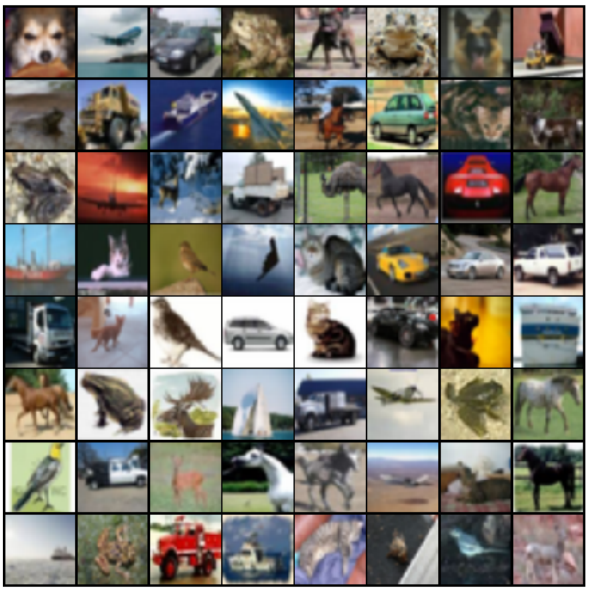
We are able to replicate the work of the authors in a qualitative manner by applying GANs on the datasets mentioned in the original paper.

### 5.1.1 MNIST Dataset

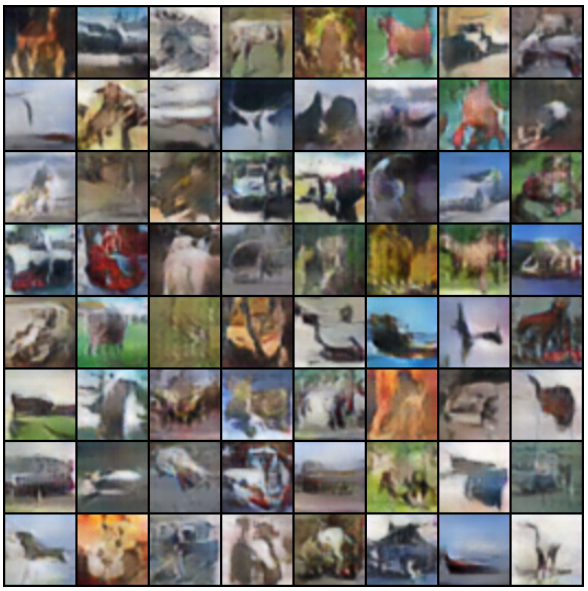
Figure 4: Fake MNIST images generated by GAN

### 5.1.2 CIFAR10 Dataset

[1]Some graphs and analysis are avoided because of the limitation of space, but they can be found in the Jupyter notebook provided



(a) Training Images from CIFAR10

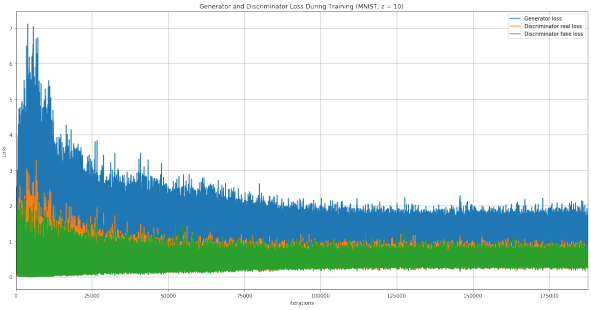


(b) Fake Images generated by GAN network

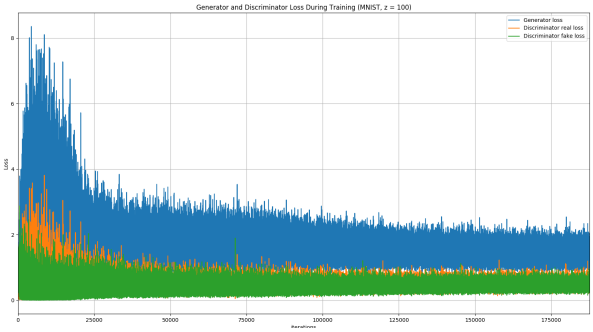
Figure 5: Comparison of images from original dataset with the one generated by GAN(`100 epochs`)

## 5.2 Experiment B

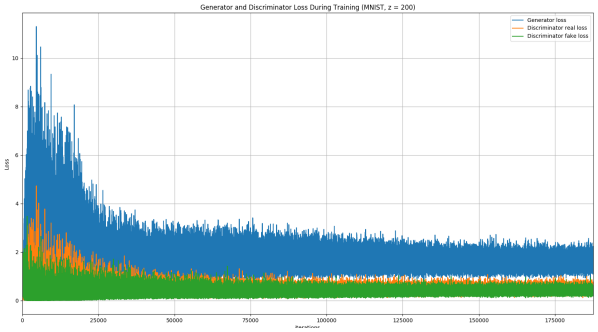
In this section, we experimented with three different values of the latent variable to see how does it affect the loss functions for G/D network.




(a)




(b)




(c)

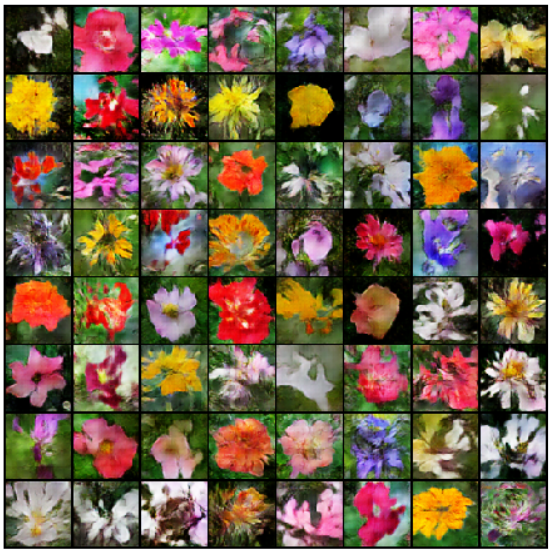
Figure 6: Generator and Discriminator loss with (a) $\mathbf{z = 10}$ (b) $\mathbf{z = 100}$ (a) $\mathbf{z = 200}$

As we can see from the figures above, the change of dimension of the latent variable doesn't affect the output noticeably. Even the images generated with different latent variable sizes look identical.

## 5.3 Experiment C

We have applied original vanilla GAN network on Fashion-MNIST dataset. As we can see in Figure 7b, the results generated are quite accurate considering that we have only trained the networks for 200 epochs. We have implemented DCGAN on Flower Dataset. This implementation was done with the default hyperparameters as used by their original authors. The results obtained(Figure 7a) looks very close to the flower images in the original dataset.

(a) Best generated image generated for the Fashion MNIST Dataset



(b) Best generated image for the Flower Dataset

Figure 7: Comparison of images from original dataset with the one generated by GAN(`100 epochs`)

## 5.4 Experiment D

Figure 8 shows the performance of the Discriminator real Loss function with `Adadelta, Adagrad, Adam, RMSprop, SGD` optimizers. Although the original authors of the WGAN paper used RMSprop and Adam optimizer in their work, we find that SGD optimizer provides us with the most clearest images and also has the most stable Loss function. We ran all the models here for 10000 epochs.

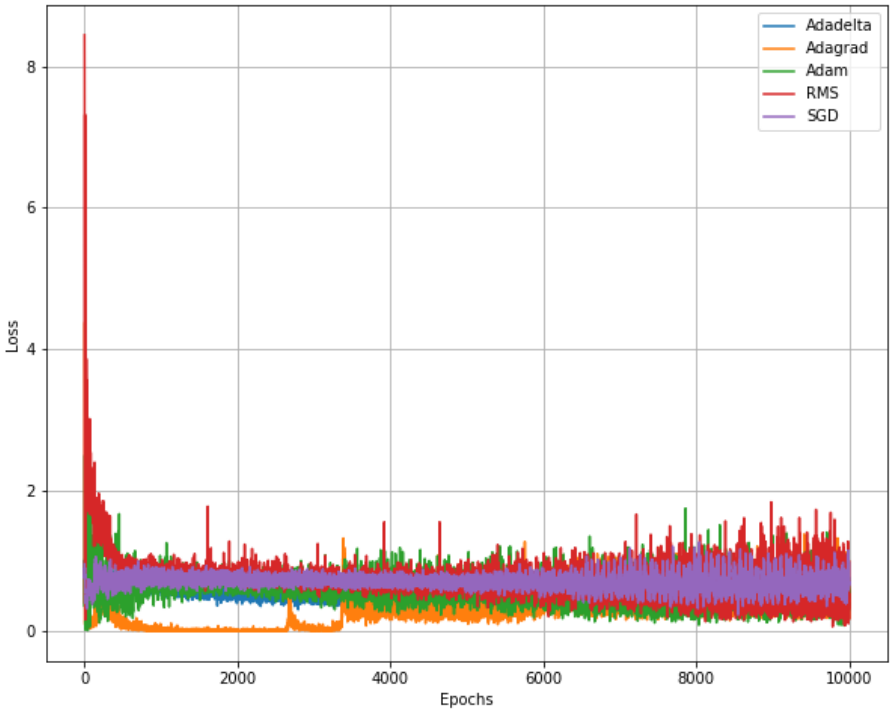



Figure 8: Plot for the real discriminator loss function for various optmizers

## 5.5 Experiment E

Wasserstein GAN was implemented and we generated the MNIST dataset images. On plotting the loss function as shown in figure 9, we find that the loss axis is scaled quite differently than the one for DCGAN and GAN. The reason being, WGAN loss function reflects the image quality which is more desirable.

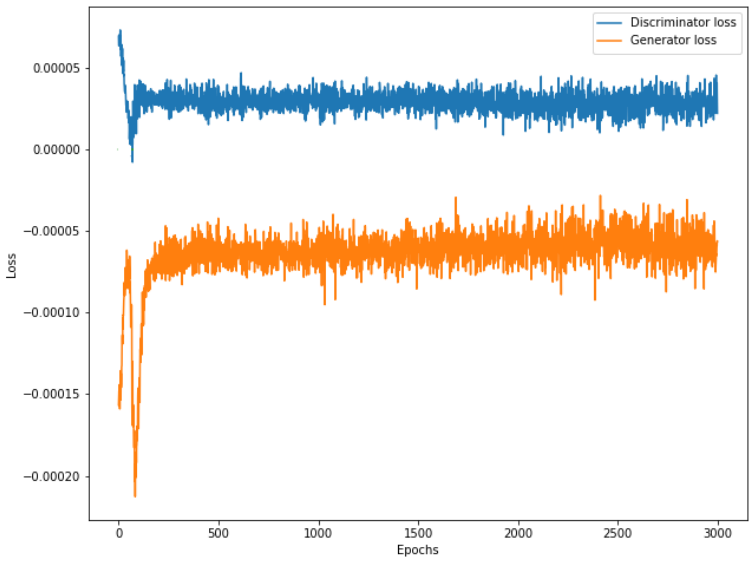



Figure 9: Plot for the loss of discriminator and generator for WGAN model on MNIST dataset

# 6 Discussion and Conclusion

In this project, we have first implemented the GAN architecture on MNIST and Fashion-MNIST dataset. Three experiments were carried out by varying the latent variable size $z = \{10, 100, 200\}$. We have noticed that this does not affect the performance of GANs. For DCGANs, we have used discriminator network very similar to the usual CNN classification network whereas the generator looks very similar to the reverse of the discriminator except the convolution layer is replaced with the transpose convolution layer. We tested DCGANs with three datasets- `CIFAR 10`, `Flower Dataset` ,and `MNIST` dataset. We experimented by increasing the depth of the network which turns out to generate better results than the default network. We also tested the network by using different optimizers like Adam, RMSprop, SGD, and Adadelta in which SGD yielded the best results. Finally, we use WGAN which doesn't use the JS Divergence to measure divergence, instead relies upon Wasserstein distance with MNIST dataset with default parameters. As indicated by Figure 9, we obtain loss function which converges faster as compared to original GAN network. One of the limitations we faced while replicating the results from WGAN paper is the limited computation power on Google Colab, which repeatedly failed with LSUN dataset. In the future, we would like to attempt to experiment with various GAN architectures and hope to improve our model. Some advancements to GAN architectures that can be further implemented and studied are :

1. Improved WGAN[11], where clipping of weights is eliminated by adding a penalization term to the norm of the gradient of the critic function. This improvement would also address the issue of Mode-Collapse, which is often encountered in the images generated by vanilla GAN, where the generator generates the same image in every iteration.
2. Least Squares GAN[12], where the Loss function is used instead of a critic and Weight decay regularization is used to bound the loss function
3. StyleGAN[13] model is arguably the state-of-the-art in its way, especially in Latent Space control. This model borrows a mechanism from Neural Style Transfer known as Adaptive Instance Normalization,(AdaIN), to control the latent space vector z and improves the state-of-the-art in terms of traditional distribution quality metrics, leads to demonstrably better interpolation properties variation.

Overall, we are able to reproduce results which are decent considering the constraints of time and resources. Despite the limitations, GANs prove to an excellent generative modelling technique as confirmed by our results, which might provide several breakthroughs in many important fields.

# 7 Statement of Contributions

A.W., A.B and S.K.D have contributed equally to the project. A.W. and A.B. mostly focused towards experimenting with different models whereas S.K.D contributed towards documentation.[2]

# References

[1] Mehdi Mirza Bing Xu David Warde-Farley Sherjil Ozair Aaron Courville Yoshua Bengio Ian J. Goodfellow, Jean Pouget-Abadie. Generative adversarial networks. *arXiv preprint arXiv:1406.2661*, 2014.

[2] Chintala S Radford A, Metz L. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv:1511.06434v2 [cs.LG]*, 2016.

[2]Github link - https://github.com/amanwalia92/GANs

[3] Wojciech Zaremba Vicki Cheung Alec Radford Tim Salimans, Ian Goodfellow and Xi Chen. Improved techniques for training gans. in advances in neural information processing systems. In *In Advances in Neural Information Processing Systems*, pages 2226—-2234, 2016.

[4] François Fleuret Tatjana Chavdarova. Sgan: An alternative training of generative adversarial networks. 2017.

[5] Gerhard Widmer Hamid Eghbal-zadeh. Probabilistic generative adversarial networks. *arXiv preprint arXiv:1708.01886v1*, 2017.

[6] Soumith Chintala Martin Arjovsky and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

[7] Guo-Jun Qi. Loss-sensitive generative adversarial networks on lipschitz densities. *arXiv preprint arXiv:1701.06264*, 2017.

[8] Huszar F Caballero J Cunningham A Acosta A-et al Ledig C, Theis L. Photo-realistic single image super-resolution using a generative adversarial network. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 105—-114, 2017.

[9] Dugelay JL Antipov G, Baccouche M. Face aging with conditional generative adversarial networks. In *IEEE International Conference on Image Processing (ICIP)*, pages 2089—2093, 2017.

[10] Yoo J Hong Y, Hwang U. How generative adversarial nets and its variants work: An overview of gan. *arXiv:1711.05914v6 [cs.LG]*, 2018.

[11] 1349723655104342. An intuitive introduction to generative adversarial networks (gans), Jan 2018.

[12] Haoran Xie Raymond Y.K. Lau Zhen Wang Stephen Paul Smolley Xudong Mao, Qing Li. Least squares generative adversarial networks. *arXiv:1611.04076*, 2016.

[13] Othman Sbai, Mohamed Elhoseiny, Antoine Bordes, Yann LeCun, and Camille Couprie. Design: Design inspiration from generative networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 0–0, 2018.