Generative Adversarial Networks: Reproducibility Study

Nishant Mishra [260903177], Shubham Chopra [260903254] Group-40

Abstract

In this project, we study the 2014 published paper Generative Adversarial Networks. We have tried to reproduce a subset of the results obtained in the paper and performed ablation studies to understand the model's robustness and evaluate the importance of the various model hyper-parameters. We also extended the model to include newer features in order to improve the model's performance on the featured datasets, by making changes to the model's internal structure, inspired by more recent works in the field.

1. Introduction

Generative Adversarial Networks are the central topic of discussion, and we attempt to analyse them in detail in this report. Generative Adversarial Networks (GANs) were first described in (1) and are based on the zero-sum non-cooperative game, analysed thoroughly in the field of Game Theory. Initially, we provide some background and mathematical justification of the main ideas behind GANs.

1.1. Preliminaries

1.1.1. Non-Cooperative Game Theory

A Non-Cooperative Game, in Game Theory is a game which is modelled as a competition between individual players. This is opposed to Cooperative Games, where a coalition probability and joint actions are modelled. Non-cooperative games are generally analysed through a framework in Game Theory, which tries to predict players' individual strategies and payoffs and to find (2). Cooperative Game Theory does not analyze the strategic bargaining within each coalition.

1.1.2. Nash Equilibrium

In Game Theory, the Nash Equilibrium, named after the mathematician John Forbes Nash Jr., is the proposed solution of a non-cooperative game involving two or more players in which each player is assumed to know the optimal strategies of the other players, and neither player has anything to gain by changing only their own strategy. Informally, a strategy profile is a Nash equilibrium, if no player can do better simply by unilaterally changing their own strategy.
If each player has chosen a strategy, and no player can benefit by changing strategies while the other players keep theirs unchanged, then the current set of strategy choices and their corresponding payoffs constitutes a Nash equilibrium. In the context of Generative Adversarial Networks, the GAN model converges when the Discriminator and the Generator reach a Nash Equilibrium.

1.1.3. Zero-Sum Game

A Zero-Sum Game, in Game Theory, is a mathematical representation of a scenario modelled such that each participant's gain or loss of utility is exactly balanced by the losses/gains of the utility of the others. Zero-sum games are a specific example of constant sum games where the sum of each outcome is always zero. If the total gains of the participants are added up and the total losses are subtracted, they sum to zero. Hence, if one wins, the others must lose by an equal amount, collectively.
The Nash equilibrium for a two-player, zero-sum game can be found by solving a linear programming problem. Suppose a zero-sum game has a payoff matrix **M** where element Mi,j is

the payoff obtained when the minimizing player chooses pure strategy i and the maximizing player chooses pure strategy j. The game will have at least one Nash equilibrium. The Nash equilibrium can be found **(Raghavan 1994)** by solving the linear program to find a vector **u**:
Minimize:

$$\sum_i u_i$$

Subject to the Constraints:

$$u >= 0, Mu >= 1.$$

1.1.4. Generative Adversarial Networks

Generative Adversarial Networks (GANs) are a class of machine learning systems, first described in (1), usually modelled as a Zero-Sum Game between a Discriminator (D) and a Generator(G). The framework where both D and G networks are multilayer perceptrons, is referred to as Adversarial Networks. The Generative model is pitted against an adversary: a Discriminative model that learns to determine whether a sample is from the model/data distribution.
To learn the Generator's distribution *pg* over data **x**, a prior on input noise variables p**z**(**z**) is defined, and then a mapping to data space is represented as G(**z**; θg), where G is a differentiable function represented by a Multilayer Perceptron with parameters θg. Also, the Discriminator is defined as a second multilayer perceptron D(**x**; θd) that outputs a single scalar. D(x) represents the probability that **x** came from the data rather than *pg*. Thus, D and G are set up to play the following two-player Minimax Game with value function V (G, D):

$$\min_G \max_D V(G,D) = E_{\mathbf{x}\sim pdata(\mathbf{x})}[logD(\mathbf{x})]$$

$$+E_{\mathbf{z}\sim pz(\mathbf{z})}[log(1-D(G(\mathbf{z}))].$$

Adversarial nets have the advantages that Markov chains are never needed, only backpropagation is used to obtain gradients, no inference is required during learning, and a wide variety of factors and interactions can easily be incorporated into the model.

1.2. Relevant Work

Deep Boltzmann Machines (3) are Deep Generative Models that provide a parametric specification of a probability distribution function. Generative machines are models that do not explicitly represent the likelihood, yet are able to generate samples from the desired distribution. Generative stochastic networks [4] are an example of a generative machine that can be trained with exact backpropagation. Kingma and Welling (4) developed more general stochastic backpropagation rules, allowing one to backpropagate through Gaussian distributions with finite variance, and to backpropagate to the covariance parameter as well as the mean. Also, more recent work on DC-GANs (5), cGANS (6), and LSGANs (7) extends the ideas of GANs by introducing more advanced concepts.

2. Datasets

2.1. About the Datasets

The MNIST database (8) (Modified National Institute of Standards and Technology database) is a large database of handwritten digits. The MNIST database, has a training set of 60,000 examples, and a test set of 10,000 examples. It is a subset of a larger set available from NIST. Each image has a size of 28 x 28. The digits have been size-normalized and centered in a fixed-size image.
The second dataset, The CIFAR-10 (9) dataset (Canadian Institute For Advanced Research) is a collection of images that are commonly used to train machine learning and computer vision algorithms. It contains 60,000 32 x 32 color images in 10 different classes. The 10 different classes represent airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks. There are 6,000 images of each class.

2.2. Ethical Considerations

Generative Models, by definition actually generate new content. With the wide-spread attention that GANs have received, State-of-the-Art models can already produce "fake" results, includ-

Figure 1: Random samples generated by GAN on MNIST data

ing, but not limited to realistic "media" (such images, videos, music, and speech), that are indistinguishable from "true" content to the naked eye. Combined with the ability to generate this "fake" data in large volumes with ease, it is important to consider who is tasked with deciding and maintaining the integrity of such online content.

3. Paper Reproduction

The provided code was implemented using the now obsolete Theano framework and using python2, hence it was really difficult to reconfigure and get it setup on our system. Nevertheless we managed to hack the code and get it to execute for the task of reproducing the results on MNIST dataset but proceeded to use the much more interpretable and relevant pytorch implementation for ablation studies and extension of the model. The original paper trains the presented GAN network on the MNIST, CIFAR-10 and TFD images. However, the Toronto Faces Database (TFD) is not accessible without permission, and the provided code does not include scripts for it. Hence, we do not reproduce their results on the TFD database.

3.1. MNIST Results

We trained the GAN network, as described in the paper with the same hyper-parameters, i.e., learning rates = 2e-5, momentum optimizers for both the Discriminator and the Generator and k value of 1, where k is the number of inner loops steps applied to the Discriminator model. Both the models are multi layer perceptrons. The Generator Nets used a mixture of rectified linear activations (ReLU) and sigmoid activations, while the Discriminator Net used Maxout activations. Dropout was applied in training the Discriminator Net. The model was trained for 200 epochs and results recorded. In general, our results were comparable to the ones showed in the paper, i.e., the Generator produced results that in some cases were in-distinguishable from the original images to the naked eye. *Figure 1* shows these generated Images. The images shown are randomly sampled, as described in the paper. Also, *Figure 2* shows the Discriminator and Generator Loss curves, with respect to the number of epochs.

3.2. CIFAR-10 Results

We tried the GAN network, as described in the paper with the same hyper-parameters, i.e., learning rates = 2e-5, momentum optimizers for both the Discriminator and the Generator, and k value of 1. The Model is the same as described in Section 3.1., the results obtained for CIFAR 10 were blurry and took a much longer time to train. Figure 4 shows the generated images when the model is reproduced on CIFAR-10 dataset.

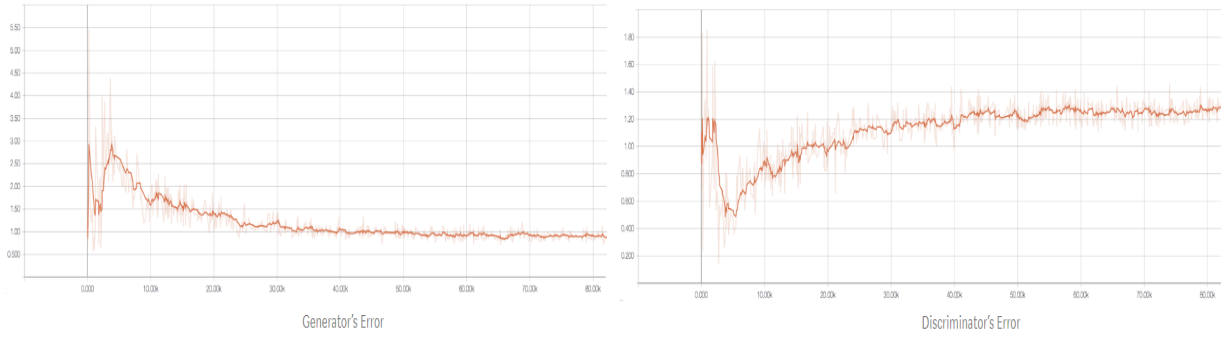


Figure 2: Discriminator and Generator loss plots of reproduced model for MNIST with given code

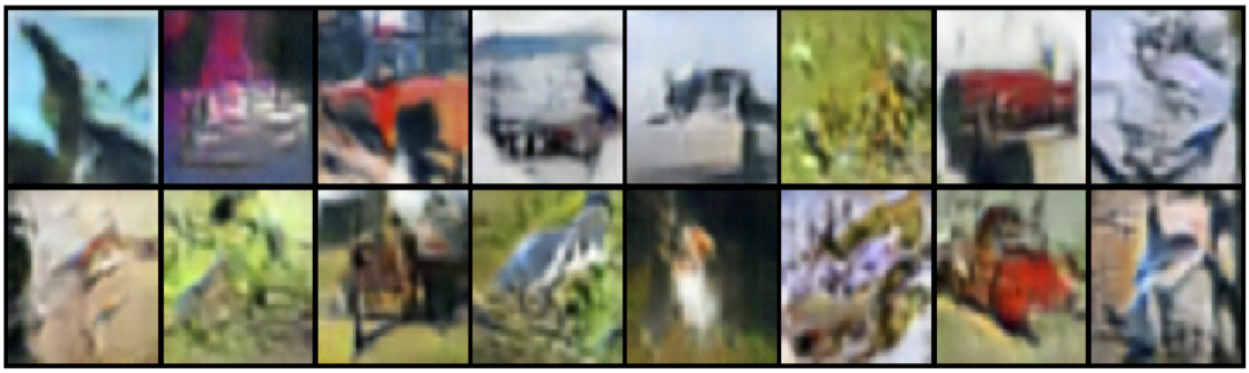


Figure 3: Generated samples by GAN reproduced using CIFAR-10 dataset

3.3. Analysis of the Reproduced Results with the Paper

The results obtained were very similar to the ones obtained in the paper. MNIST was faster to converge and gave decent quality images, but the images were blurry in case of CIFAR-10, but so were the results obtained in the original paper. A Gaussian Parzen window based log likelihood for the MNIST dataset was also estimated using the given code and was found out to be 262.2.

4. Ablation Studies

4.1. Tuning Hyper-Parameters

GANs have been known to be unstable to train, often resulting in generators that produce nonsensical outputs. We decided to put this notion to test by tuning some of the hyperparameters involved in training the models. We experimented with different Hyper-Parameter settings for the model

4.1.1. Learning Rates

As part of our first experiment, we tuned the learning rates of both Generator and Discriminator models. We increased the Learning rate of the Generator to 3e-4 from 2e-4, while keeping the learning rate of the Discriminator fixed. Increasing the learning rate of the Generator, caused it to oscillate and it failed to converge. This led to the Discriminator loss reaching 0, while the Generator's loss function remained at a high constant. This led to images generated to be noisy and not feasible for acceptable fake synthetic images. This trend was strikingly consistent in the sense that, for the models to converge the learning rates of both the models have to be nearly similar and usually very small. This observation follows well from the paper where

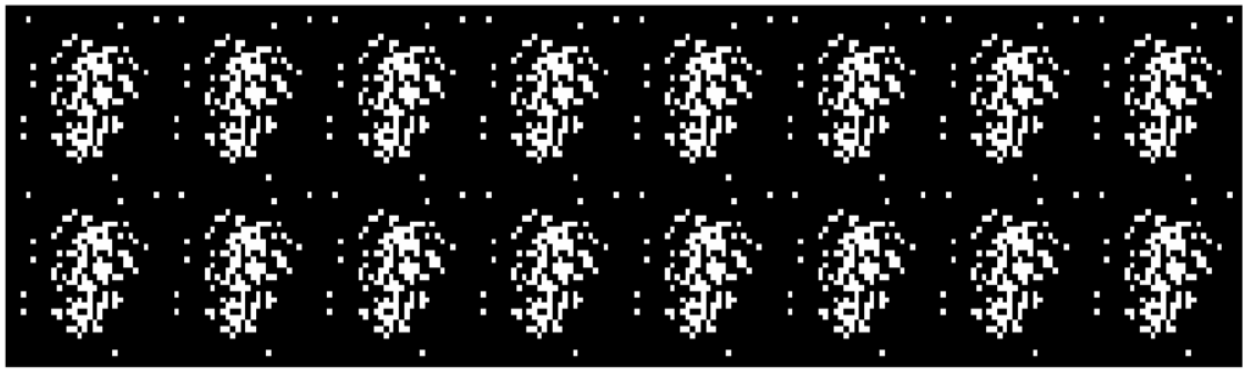


Figure 4: Noisy samples generated due to model not converging on changing learning rate

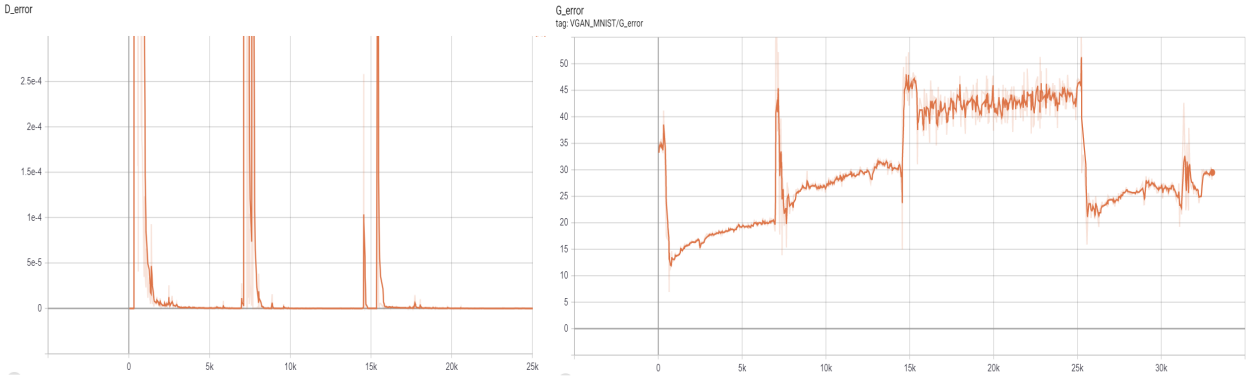


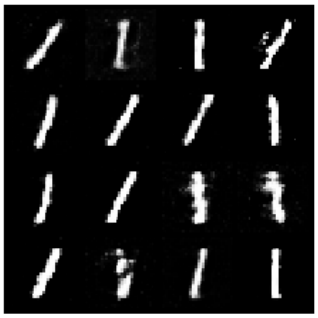
Figure 5: Discriminator and Generator loss plots when model diverges

it is mentioned that the two models need to be synchronized properly during training and that the generator network should not be allowed to update too much without updating D. Even increasing the learning rate for the discriminator lead to non convergence of the model. In Figure 4 we can see the noisy samples generated due to divergence of the models due to changing the learning rates. In Figure 5 we can see the plots of generator and discriminator loss for the diverged model, while the discriminator loss becomes zero, the generator loss keeps increasing.

4.1.2. Loss Function

GANs usually use crossentropy loss function as the adversarial loss, which basically calculates the loss for both discriminator and the discriminator output of generator. Since cross entropy loss in general are classification losses, it is equivalent to classifying whether the output of the generator network is correctly classified by the discriminator. We decided to experiment with the L2 norm or Mean Squared error loss function. The motivation behind this experiment was that this L2 loss won't just care about whether data has been correctly labelled as in case of Binary cross entropy loss but will penalize the model based on how badly it is classified i.e how far the sample is from the correct label This in general gives more meaningful gradient information for backpropagation and should lead to better quality images. This line of reasoning turns out to be one of the significant motivation behind an advanced version of GANs called the Least Squared Error GAN (LSGAN) (7), although it also features other major new ideas such as weight decay etc. The results obtained are as shown in Figure 6

Figure 6: Random samples generated with MSE loss function



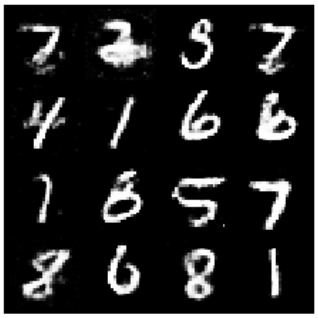


Figure 7: random samples, a) showing the model suffering from mode collapse b) after convergence, with d=2.

4.1.3. D_steps

The paper introduces another important hyperparameter d_steps, the number of steps to apply to the discriminator, which is basically the number of times the Discriminator is trained before updating the Generators. This is motivated by the proposition 2 mentioned in the paper that states Proposition 2 which states
"If G and D have enough capacity, and at each step of the training algorithm, the discriminator is allowed to reach its optimum given G, and p_g is updated so as to improve the criterion then p_g converges to p_{data}."
In the paper this parameter has been fixed to d=1, since it is the most computationally efficient. We decided to change this to 2 and observe the results. In this setting, initially the model suffered from *mode collapse*. A mode collapse refers to a generator model that is only capable of generating one or a small subset of different outcomes, or modes. For the first 25-30 epochs the model was only able to generate MNIST digit 1 as shown in the figure. But it later recovered and converged faster to an optimum than it did with d=1. The results obtained are as shown in Figure 7, as we can see from the loss plot (Figure 10 in Appendix) the Discriminator loss fluctuates heavily at the start denoting mode collapse before smoothing out towards the end.

4.2. Extensions of the Model

4.2.1. DCGAN

Deep Convolutional Generative Adversarial Networks (5) or DCGAN are a variation of GAN where the vanilla GAN is upscaled using CNNs. The DCGAN replaces the linear layers with convolutional layers and deterministic spatial pooling functions (such as maxpooling) with strided convolutions are replaced allowing the network to learn its own spatial downsampling. We use this approach in our generator, allowing it to learn its own spatial upsampling, as well as discriminator. After every convolutional layer, batch normalization was added to the generator model to stabilize learning by normalizing the input to each unit to have zero mean and unit variance. This helps deal with training problems that arise due to poor initialization and helps gradient flow in deeper models.// We tried this on the CIFAR-10 and converged to much better results in lesser

Figure 8: Random samples generated using DCGAN trained on CIFAR-10

epochs than with the MLP based implementation in the original code. We can see the generator output for the model trained on CIFAR-10 in Figure 8. The results are visibly better tahn the ones we observed in section 3.2 using vanilla GAN.

4.2.2. cGAN

We also implemented one of the extensions mentioned in the paper itself, that of a conditional generative model by feeding in the output along with the input to both generator and discriminator. This conditional Generative Adversarial Network(cGAN) allows us to direct the generation process of the model by conditioning it on certain features, here, the class labels. In the generator the prior input noise $p_z(z)$, and y are combined in joint hidden representation, and the adversarial training framework allows for considerable flexibility in how this hidden representation is composed. In the discriminator x and y are presented as inputs and to a discriminative function (embodied again by a MLP in this case). The results are as shown in Figure 9

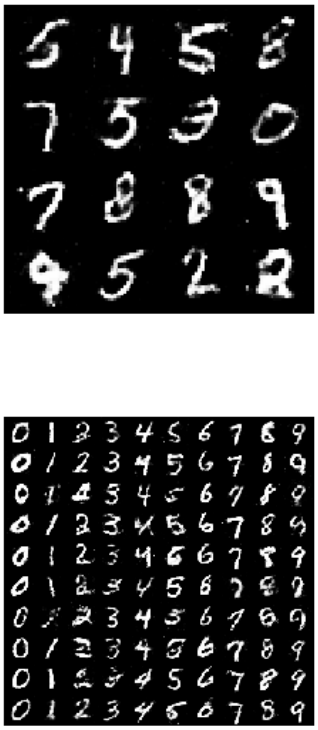


Figure 9: Generated Samples a) conditioned on labels y= **[5,4,5,8,7,5,3,0,7,8,8,9,9,5,2,2]** b) for y=0 to 9.

5. Conclusion and Discussion

From our experiments we conclude that Generative Adversarial Networks are indeed a very powerful class of generative models and can be used for a number of down stream tasks apart from data generation. The generative and discriminative models trained competitively boost each other to be used as stand alone models for various tasks. GANs, however, still suffer from a number of shortcomings. MOst notably they are highly unstable to train, and therefore require careful fine tuning of parameters. Not only this, high quality GANs are also very computationally expensive to . Newer developments in the field of generative learning and further extensions to the GAN model have been made to overcome these disadvantages and add newer capabilities and are consistently growing making it a hot topic of research.

6. Statement of Contributions

This project is a collaborative effort between three team members. The tasks were evenly distributed and every member contributed equally.

References

[1] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *arXiv preprint arXiv:1406.2661*, 2014. 1, 2

[2] J. F. Nash, "Equilibrium points in n-person games," *Proceedings of the National Academy of Sciences*, vol. 36, no. 1, pp. 48–49, 1950. 1

[3] R. Salakhutdinov and G. Hinton, "Deep boltzmann machines," pp. 448–455, 2009. 2

[4] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013. 2

[5] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015. 2, 6

[6] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014. 2

[7] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. Paul Smolley, "Least squares generative adversarial networks," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2794–2802, 2017. 2, 5

[8] Y. LeCun, "The mnist database of handwritten digits," *http://yann. lecun. com/exdb/mnist/*. 2

[9] A. Krizhevsky, V. Nair, and G. Hinton, "The cifar-10 dataset," *online: http://www. cs. toronto. edu/kriz/cifar. html*, vol. 55, 2014. 2

Appendices

Loss Plots

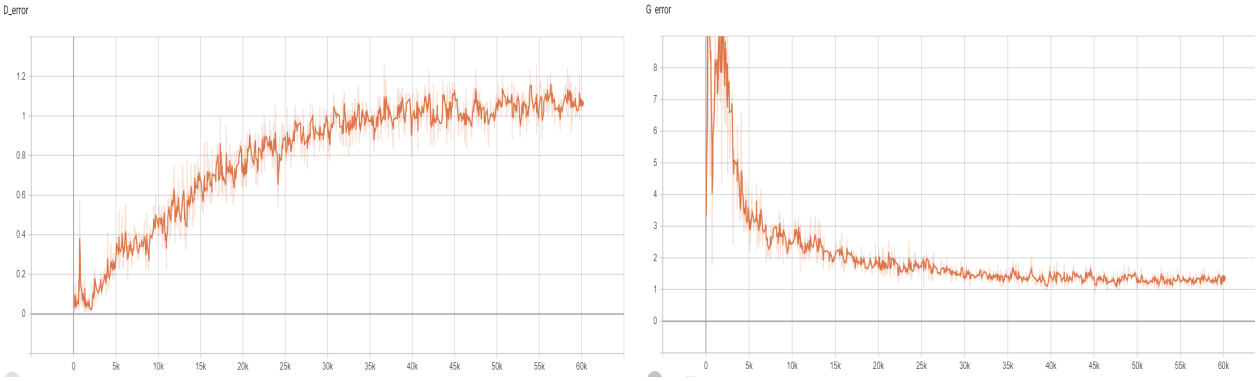


Figure 10: Discriminator and Generator loss plots with d=2

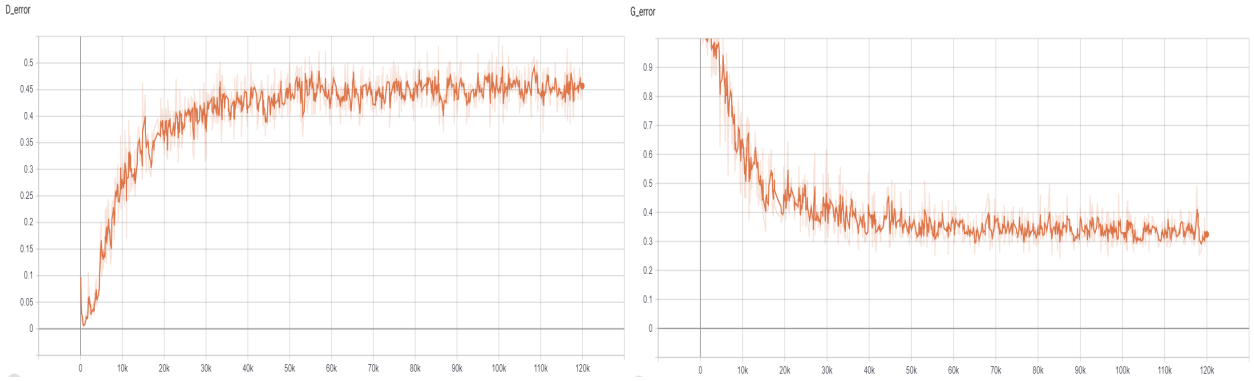


Figure 11: Discriminator and Generator loss plots with MSE loss function

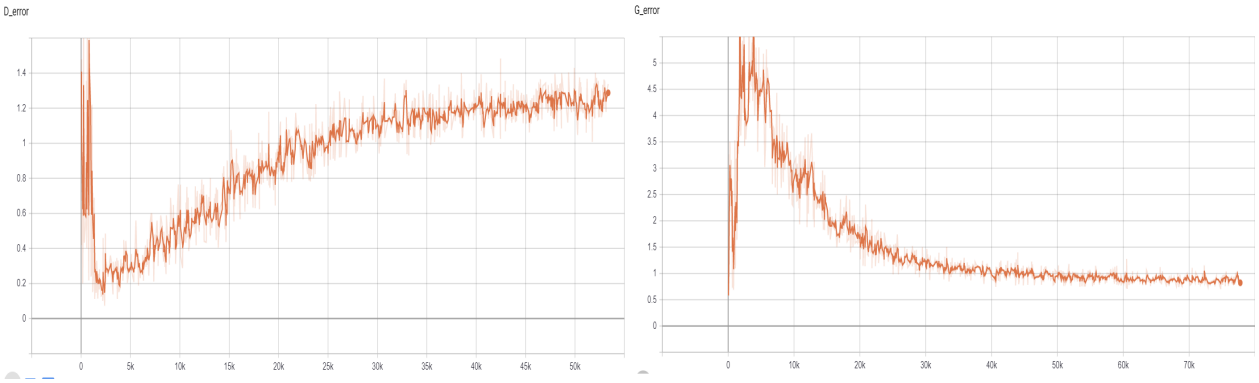


Figure 12: Discriminator and Generator loss plots for cGAN