Deep Learning in Computer Vision - Generative Adversarial Networks

Javier Rico *(jvirico@gmail.com)*

I. INTRODUCTION

Generative Adversarial Networks (GANs) are a powerfull class of generative models. Despite their successes, the most appropriate choice of network architecture is still not well understood. GAN models for image synthesis have adopted a deep convolutional network architecture, which eliminates or minimizes the use of fully connected and pooling layers in favor of convolution layers in the generator and discriminator [6]. During this work we evaluate three different GAN architectures implemented with Keras framework in Python. The analysis is splitted in three parts, (1) using Multilayer Perceptron models (fully connected architecture) for Generator and Discriminator on MNIST data, (2) using a hybrid arcuitecture, with a small Fully-connected part and a Convolutional Neural Network (CNN) block for both, Generator and Discriminator on MNIST data, and (3) using a deeper Convolutional Neural Network on images of cats from CIFAR10 dataset.

II. METHOD

A. Generative Adversarial Networks

GANs consist of two main components, a generator and a discriminator, in this case both consisting of Convolutional Neural Networks (*CNNs*) [3]. The discriminator D is a binary classifier. Its objective is it to classify data samples from the real data set $x \backsim p_{data}$ as real, $D(x) = 1$, and generated samples $\hat{x} \backsim p_{model}$ as fake, $D(\hat{x}) = 0$. The generator G is tasked to create synthetic samples; it receives as input a noise vector z, also called latent space vector, sampled from a noise distribution p_z (*e.g. a normal distribution*) and maps it to the image space.

Both networks are pitted in a competition against each other. The generator attempts to produce samples which are indistinguishable from the real samples, $p_{model} \sim p_{data}$, thus fooling the discriminator. In the meantime, the discriminator's objective is to avoid being fooled through learning meaningful features which better distinguish between real and generated samples. This concept of opposing networks is well represented by the principles of game theory via the following min-max optimization task [2]:

$$min_G max_D V(D,G) = \\ _{x\sim p_{data}(x)}[logD(x)] +_{z\sim p_z(z)} [log(1 - D(G(z)))] \quad (1)$$

Where, in the generator training step, we want to make $D(G(z))$ to be closer to one, or $logD(G(z))$ to be large; and in the discriminator training step, we want $logD(x)$ to be large and $logD(G(z))$ to be small. We can see an example of GAN convergence in *Fig. 11*.

As a result, GANs have the ability to mimic data distributions and to synthesize realistic looking images. The generative modeling and distribution learning of GANs can maximize the probability density over the data, and discover its high dimensional latent distribution, which has lead to significant performance gains in the extraction of visual features [5].

III. IMPLEMENTATION

Keras library as an interface for TensorFlow library is used. The complexity abstraction that this framework provides makes the implementation quite straight forward, reducing the amount of code nedded to implement each of the models here presented.

IV. DATA

In this practical work, we have used two public datasets, MNIST and CIFAR10 (cat category).

A. MNIST

The first dataset we have used is **MNIST** handwritten digit dataset. Some examples of the dataset are shown in *Fig. 1*. The dataset contains a train set of 60.000 examples, and a test set of 10.000 examples. MNIST is a subset of a larger set available from NIST [1].

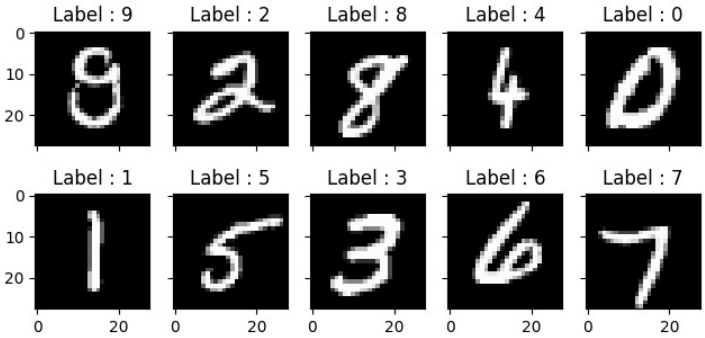


Fig. 1: Digits sample.
MNIST dataset.

The digits come size-normalized and centered in a fixed-size image, therefore no pre-processing and formatting is required.

B. CIFAR10

The second dataset we have experimented with is **CIFAR10** [7], which consists of 60.000 32x32 colour images in 10 classes, with 6000 images per class.

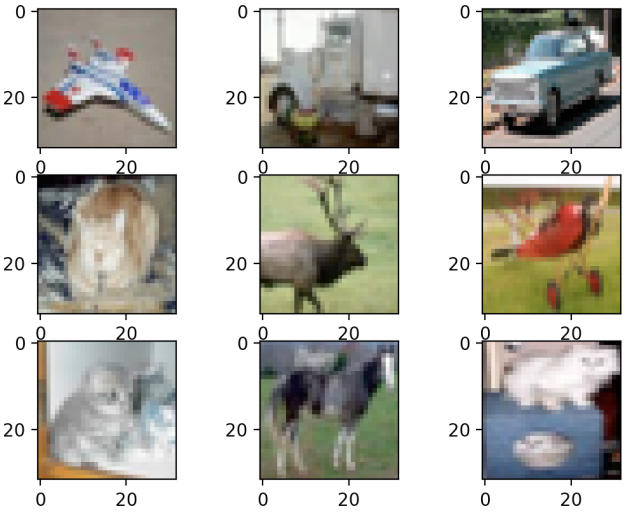


Fig. 2: Random samples.
CIFAR10 dataset.

The **10 categories** of images are ***airplane***, ***automobile***, ***bird***, ***cat***, ***deer***, ***dog***, ***frog***, ***horse***, ***ship*** and ***truck***, we can see some random examples in *Fig. 2*.

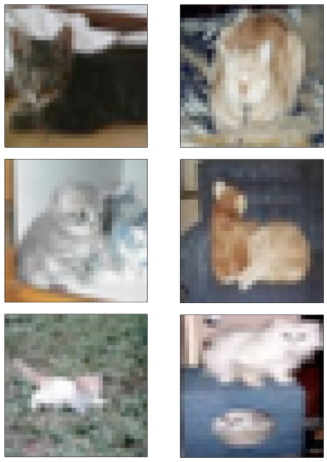


Fig. 3: Samples of Cat class.

We will be using the category Cat to train the second Convolutional Neural Network, see *Fig. 3*.

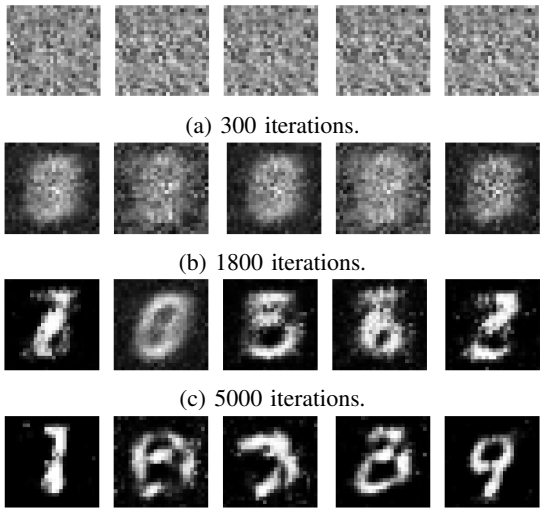
V. RESULTS AND ANALYSIS

In this section we present the experiments performed for each of the architectures and their analysis. Please be aware that all images that refer to losses, accuracies, and architecture diagrams are presented in Appendices I, II, III and IV, at the end of this report.

A. *Multilayer Perceptron GAN*

MNIST dataset is used to train a Generative Adversarial model with fully connected architecture for both, generator and discriminator models, see architecture in *Fig. 8*. To better understand the dynamics of an adversarial network we will analyze the results shown in *Fig. 4*.

To model the high dimensional latent distribution of the images, both models, generator and discriminator are pitted in a competition against each other as we already mentioned. The generator produces samples that are evaluated by the discriminator, while the discriminator is trained on MNIST images to slowly improve on differentiating real from fake MNIST images. These two actions are performed iteratively, with the promise of models convergence towards low losses and high discriminator accuracy. In the here example, we observe how this iterative process makes the generator loss to be reduced along iterations (*Fig. 9*), showing a clear improvement of the synthetic digits towards realistic ones. If we take a close look at the synthetic images, we see how at low phases of the process the synthetic images are pretty much noise, *Fig. 4, a)* after 300 iterations. This is due to the random initialization of the generator waits, and that the input vector is noise sampled from a Normal distribution. But also how the generator slowly captures the latent distribution of the digit images, starting to show visibly recognisable digits around 5000 iterations, *Fig. 4, c)*.



(a) 300 iterations.

(b) 1800 iterations.

(c) 5000 iterations.

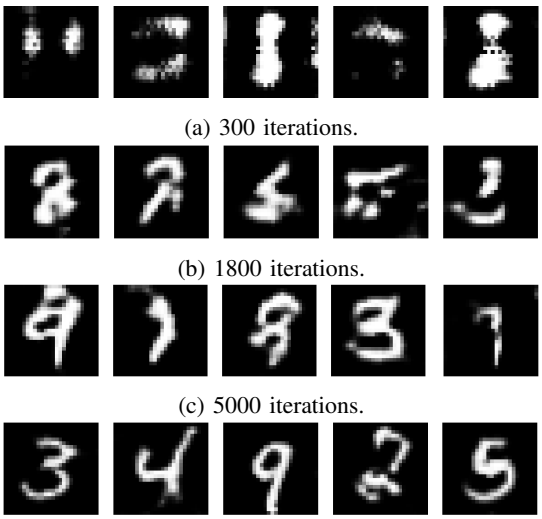
(d) 20000 iterations.

Fig. 4: Multilayer Perceptron GAN on MNIST data.
Synthetic digits of Generator along iterations from latent space vector of size 100.

Although the results after long training are far from fooling human eye, the results of this model in terms of stability are not bad, showing a stabilization of the losses (*Fig. 9*) and discriminator's accuracy (*Fig. 14, a)*), indicating that the models have converged, and no further training would probably allow the generator to better capture the latent distribution of the digits.

B. *Convolutional Neural Network GAN for MNIST*

Since existing research usually deploys CNNs for image synthesis tasks in GANs (CGAN), minimizing the use of fully connected and pooling layers, Convolutional Neural Network for generator and discriminator are used this time in the GAN's architecture, see architecture in *Fig. 10*. Targeting the same dataset, a visible improvement is immediately observed in the process when looking at synthetic generations in comparison to the former model, see *Fig. 4, a)* and *Fig. 5, a)*, both after only 300 iterations.



(a) 300 iterations.

(b) 1800 iterations.

(c) 5000 iterations.

(d) 20000 iterations.

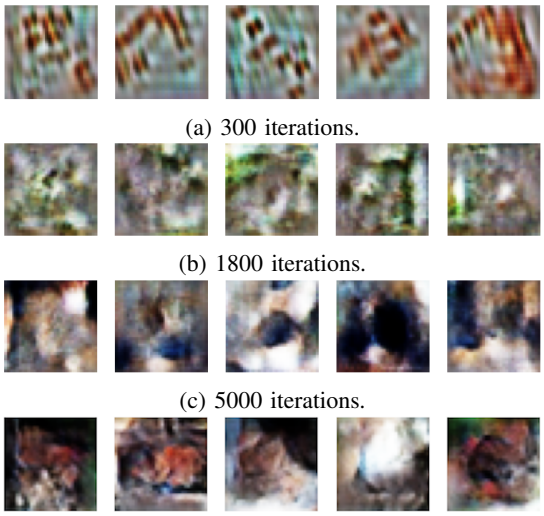
Fig. 5: CGAN on MNIST data.
Synthetic digits of Generator along iterations from latent space vector of size 100.

In this case, CGAN shows a better extraction of visual features than the multilayer perceptron model, for the same training configurations, both experiments performed for 20000 iterations. A sample of final results is shown in *Fig. 4, d)* and *Fig. 5, d)* for visual comparison.

C. *Convolutional Neural Network GAN for CIFAR10*

Cat images from CIFAR10 cat category are used to evaluate this model, *Fig. 3*. For this, a deeper Convolutional Neural Network with a Fully Connected layer at the beginning is used, see architecture in *Fig. 12*. The images are 32x32 pixels three color channeled. The discriminator is fed by a latent vector of size 100.

Despite the previous results, in practice can be difficult to train GANs due to instability issues such as mode collapse and vanishing gradient [2][4]. In this particular model and training configuration case for cat images, we see little improvement towards realistic cat images between iterations 5000 and 20000, *Fig. 6*, *c)* and *d)*, despite the long training period between synthetic samples. This is due to clear signs of **vanishing gradient effect** during the training of this model, see *Fig. 13*. As we can see, generator and discriminator losses separate one each other while the discriminator accuracy goes up fast, *Fig. 14, c)*, meaning that the discriminator is improving on differentiating between real cat images and fake ones in a fast rate, while the generator's rate of improvement is lower.



(a) 300 iterations.

(b) 1800 iterations.

(c) 5000 iterations.

(d) 20000 iterations.

Fig. 6: CGAN on CIFAR10 data.
Synthetic cats of Generator along iterations from latent space vector of size 100.

Furthermore, the generator is improving its loss by generating samples of low diversity that score well on the discriminator, and therefore poorer richness of generations are finally observed, see *Fig. 6*, *c)* and *d)*. This is a clear example of **generator collapse** to a reduced number of "safe" features so to speak.

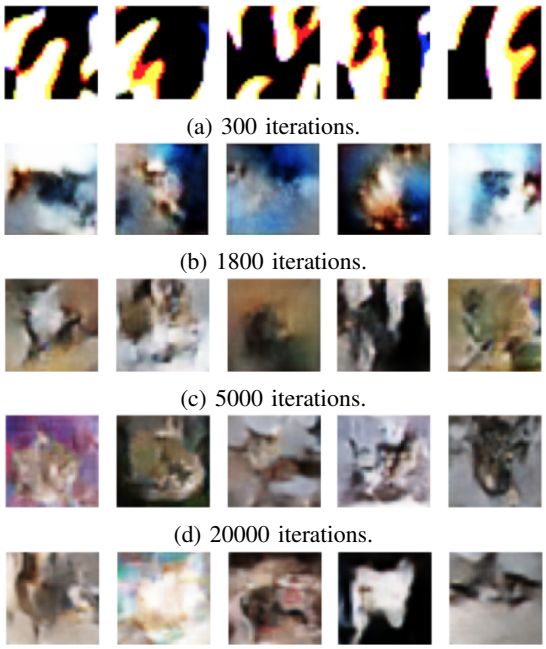
D. *Fixing Vanishing Gradient and Model Collapse*

Following, we tune the hyperparameters of the architecture, *Fig. 12*, aiming to mitigate vanishing gradient effect and model collapse we just observed. For this, we lower the learning rate of the discriminator to avoid it from improving much faster than the generator, while keeping the same configuration for the generator net.

As we observe in *Fig. 15, a) and b)*, longer number of iterations are now required for the model to converge, presenting a more stable evolution along iterations, and better visual results, *Fig. 7*, showing margin of improvement despite the added 10.000 iterations in comparison to the previous experiment (*Fig. 13*).

Comment that longer number of iterations, even when the model improvement is smooth and stable, does not always mean that the final results will be as expected, since the chances of the generator to fall into model collapse increases with iterations. Although the results are far from optimal, the latter effect is not observed after 30.000 iterations, as a rich

variety of generations is produced, as we present in *Fig. 15, c)*.



(a) 300 iterations.

(b) 1800 iterations.

(c) 5000 iterations.

(d) 20000 iterations.

(e) 30000 iterations.

Fig. 7: Second CNN GAN on CIFAR10 data.
Synthetic cats of Generator using smaller learning rate on Discriminator, from latent space vector of size 100.

Full results are presented in Appendix IV, at the end of this report.

VI. CONCLUSIONS

The choice of what architectures are most appropriate to use for GANs remains an open problem. Although we see that existing research usually deploys Deep Convolutional Networks (CNNs) for image synthesis tasks in GANs. Specifically in traditional architectures, convolutional layers are mostly used, and fully connected and pooling layers are eliminated or minimized [4]. We corroborate the efficiency of this tendency by comparing our fully connected model with a convolutional model on the same data, MNIST digits, with visually better results on the latter. Furthermore, we have experienced how in practice, it can be difficult to train GANs due to instability issues such vanishing gradient and model collapse, as we have seen for the more complex scenario evaluated. A possible extension of the work could be to modify the latter model using for example Deep Convolutional GAN (DCGAN) [3], a CNN adaptation of the original GAN proposed by Goodfellow et al. [2], well known for stability and quality of results on similar tasks.

REFERENCES

[1] Y. LeCun and C. Cortes, "MNIST handwritten digit database," 2010. [Online]. Available: `http://yann.lecun.com/exdb/mnist/`.

[2] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

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

[4] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," *arXiv preprint arXiv:1701.07875*, 2017.

[5] S. Kazeminia, C. Baur, A. Kuijper, B. van Ginneken, N. Navab, S. Albarqouni, and A. Mukhopadhyay, "Gans for medical image analysis," *arXiv preprint arXiv:1809.06222*, 2018.

[6] S. Barua, S. M. Erfani, and J. Bailey, "Fcc-gan: A fully connected and convolutional net architecture for gans," *arXiv preprint arXiv:1905.02417*, 2019.

[7] A. Krizhevsky, V. Nair, and G. Hinton, "Cifar-10 (canadian institute for advanced research)," [Online]. Available: `http://www.cs.toronto.edu/~kriz/cifar.html`.

APPENDIX I
MULTILAYER PERCEPTRON GAN

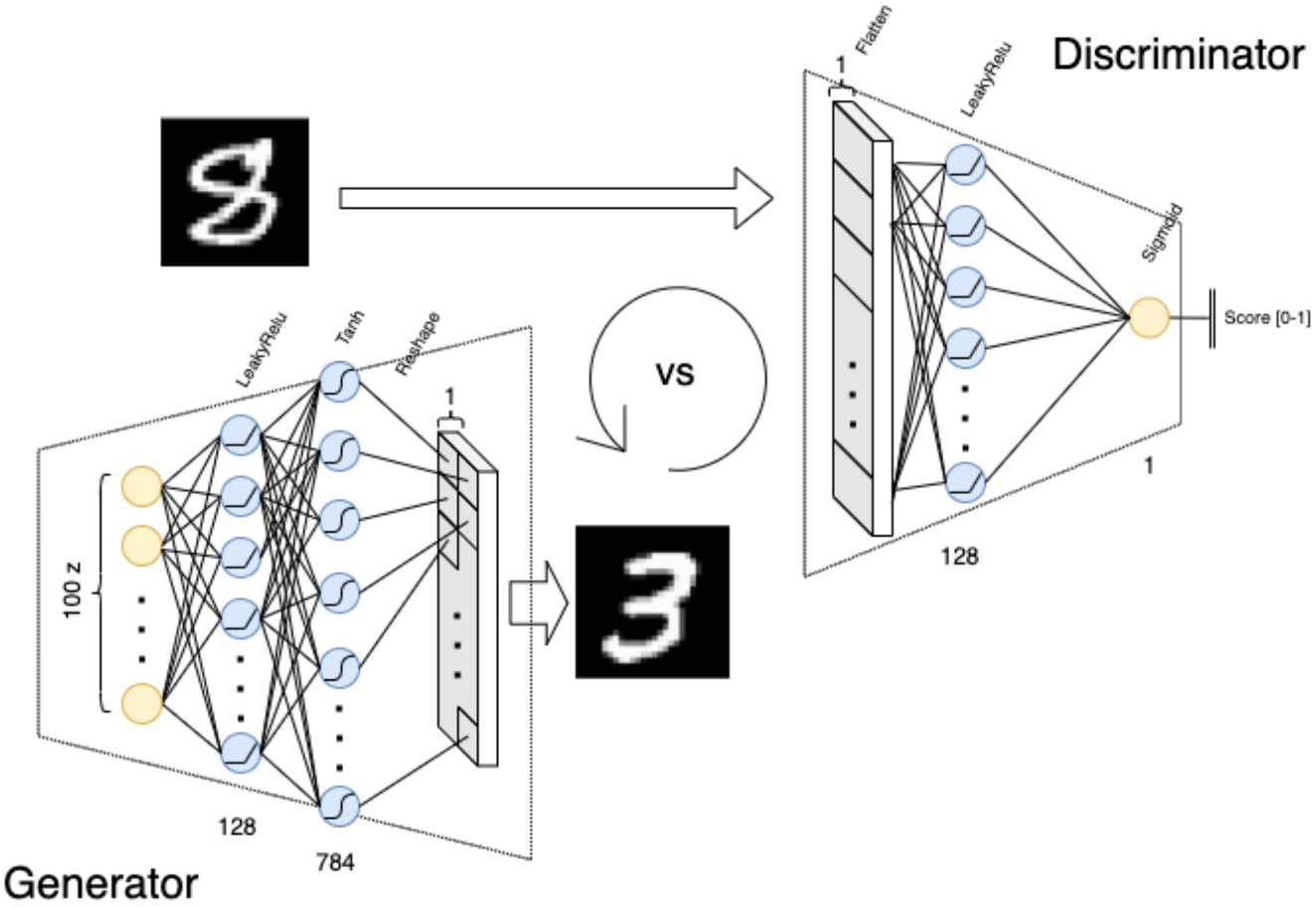


Fig. 8: CNN architecture for MNIST data.

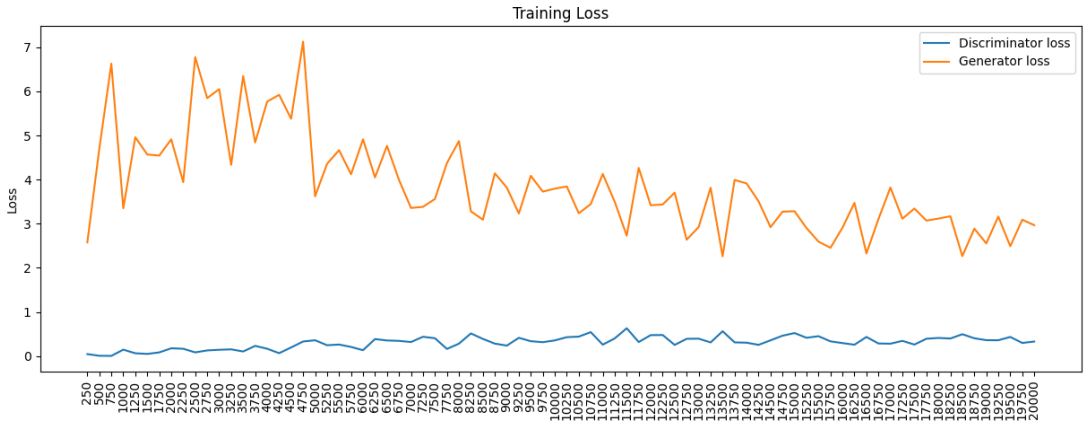


Fig. 9: Losses of Multilayer Perceptron GAN on MNIST data along 20.000 iterations.

APPENDIX II
CONVOLUTIONAL NEURAL NETWORK GAN

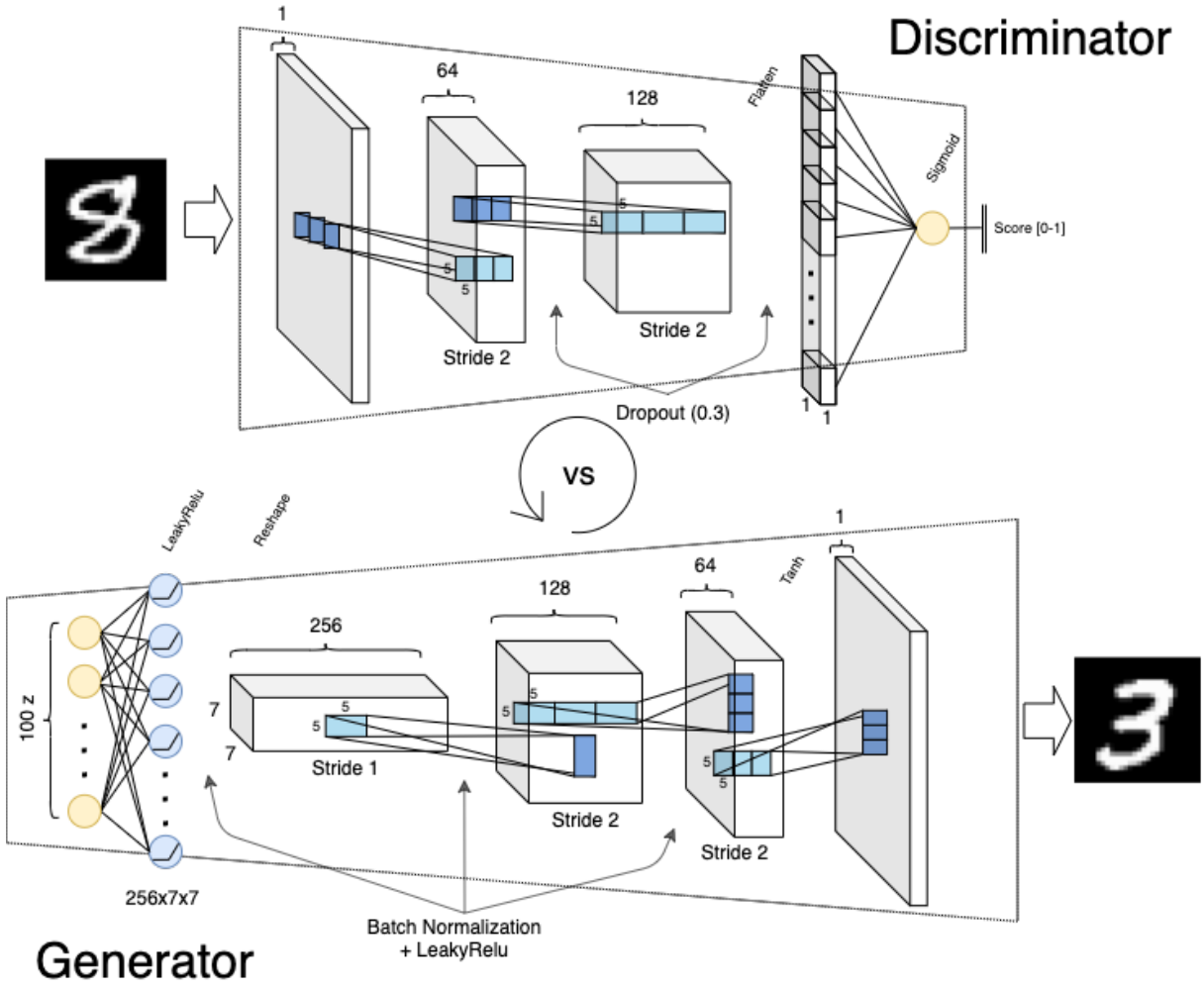


Fig. 10: Convolutional Neural Network GAN architecture for MNIST data.

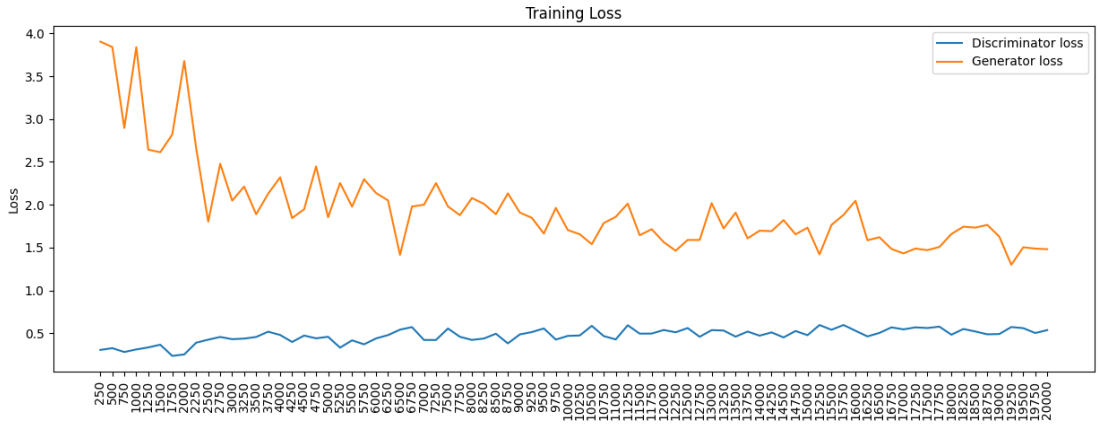


Fig. 11: Losses of CGAN on MNIST data along 20.000 iterations.

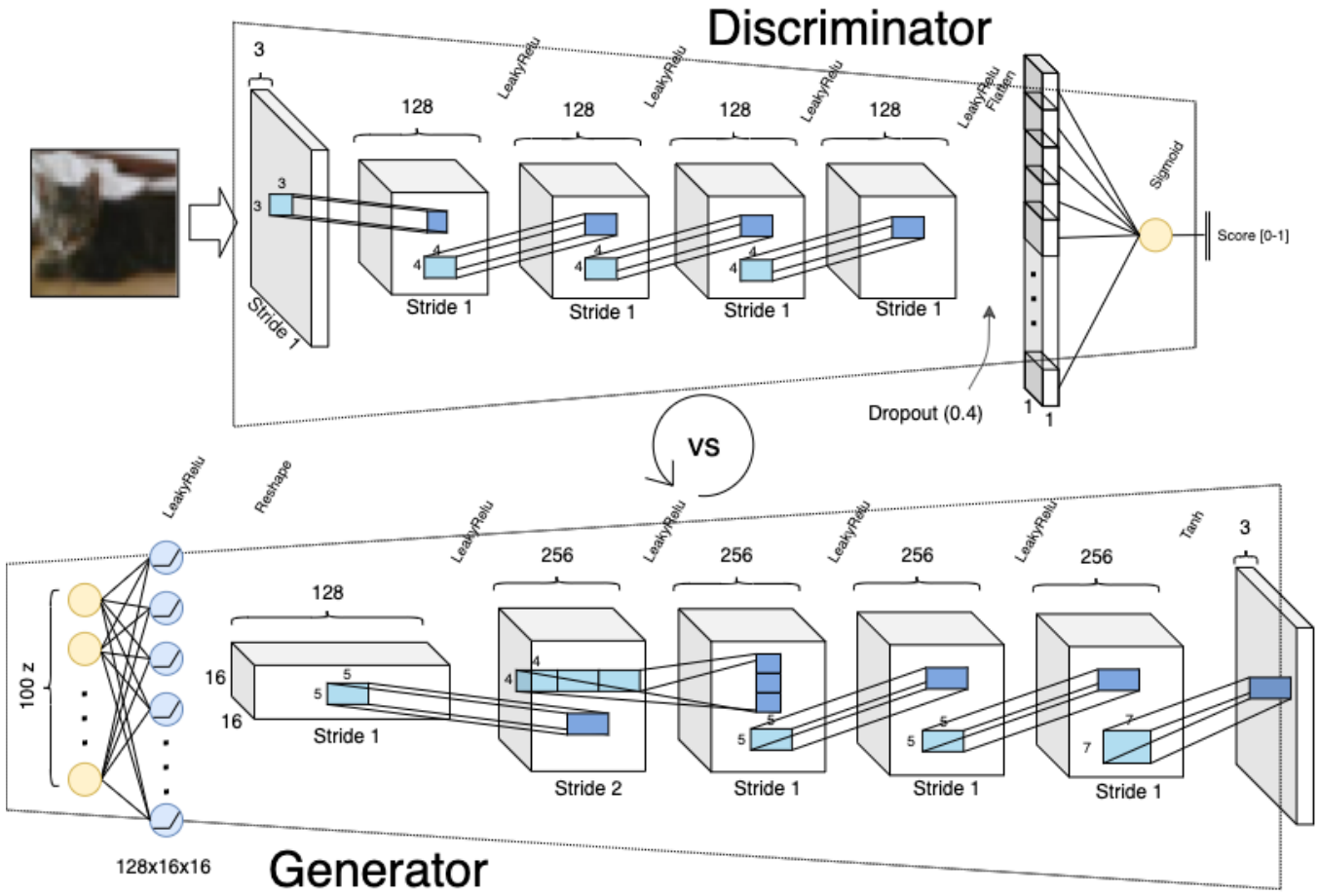


Fig. 12: Convolutional Neural Network GAN architecture for CIFAR10 data.

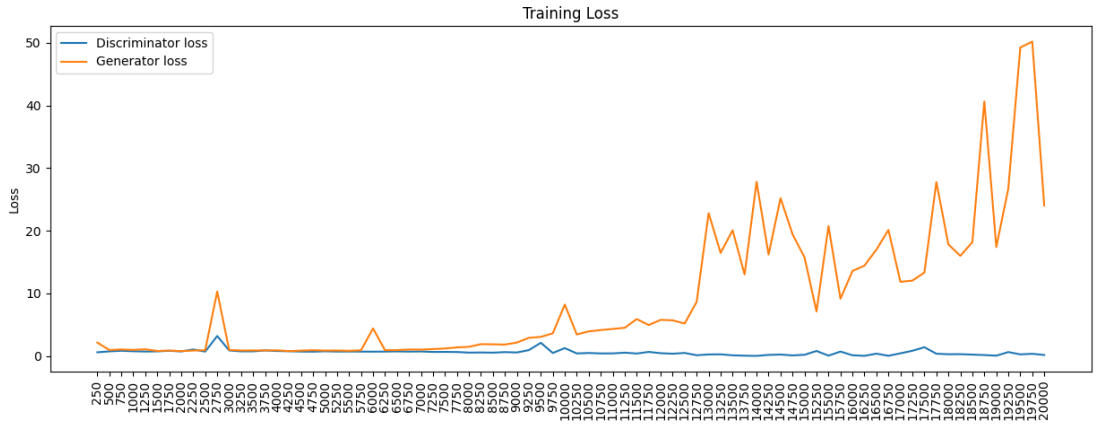
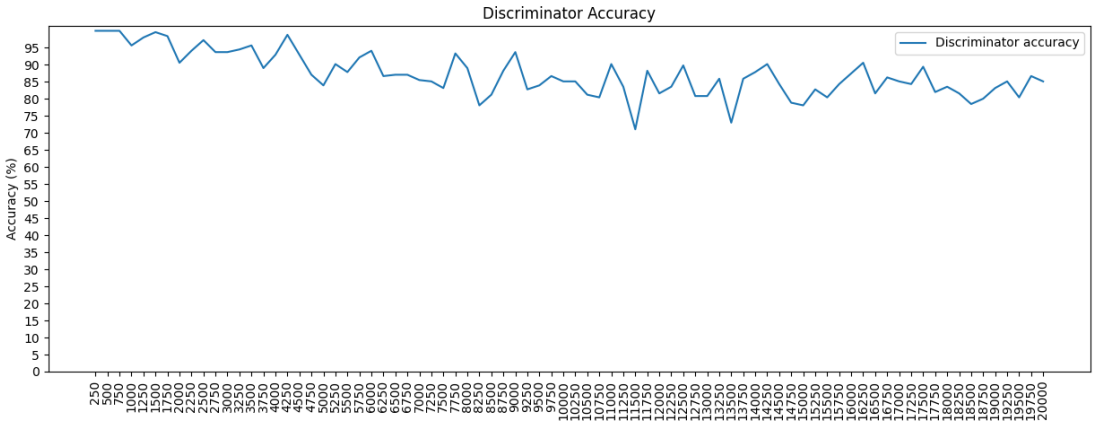
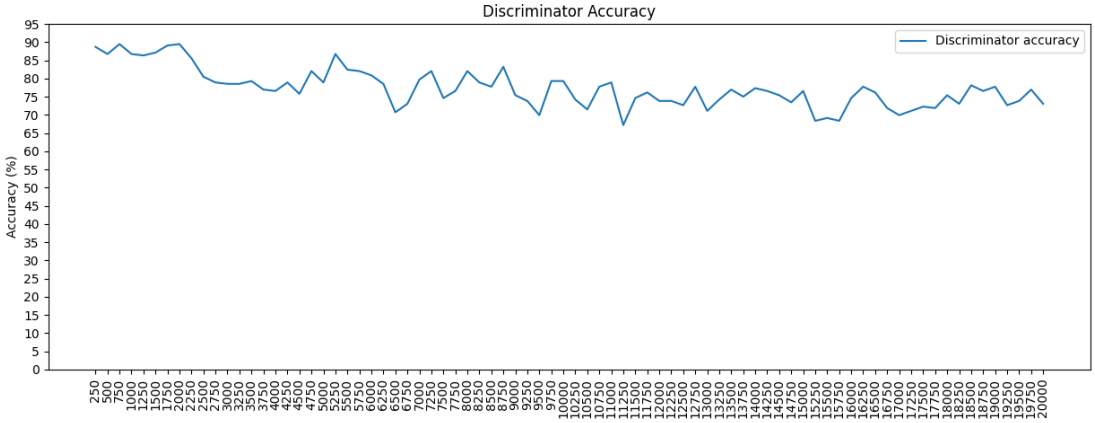


Fig. 13: Losses of CGAN on CIFAR10 data along 20.000 iterations.

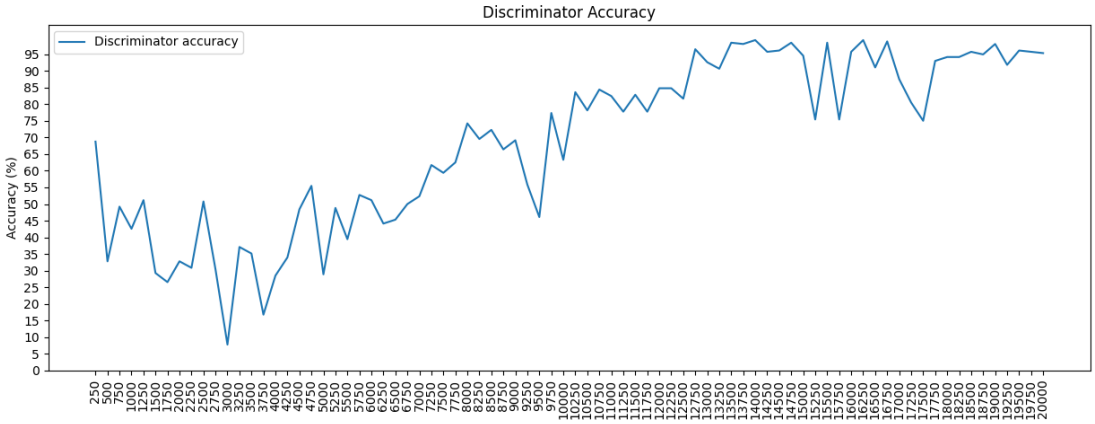
APPENDIX III
ACCURACIES



(a) Multilayer Perceptron GAN.



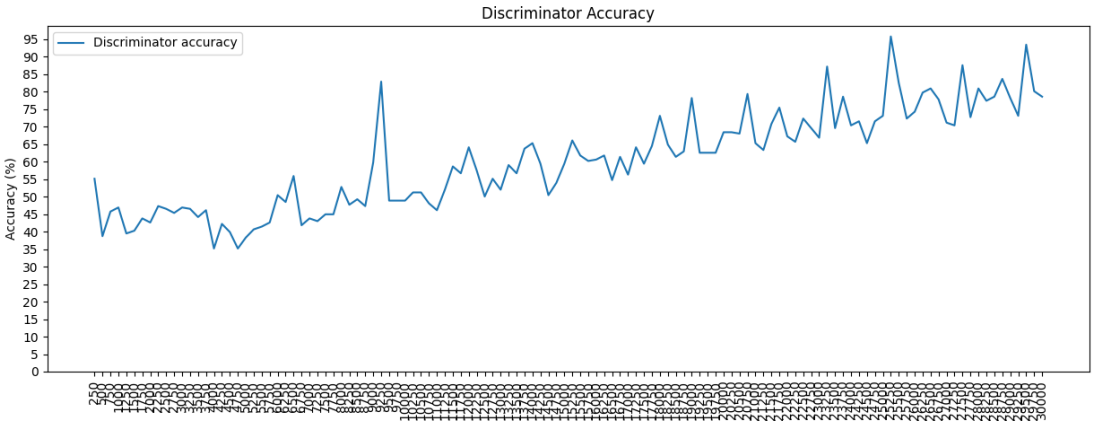
(b) CGAN for MNIST.



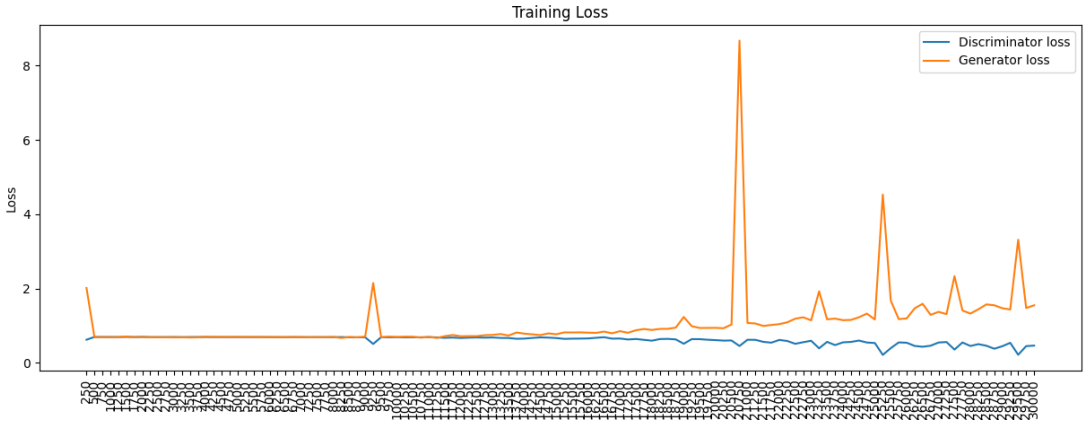
(c) CGAN for CIFAR10.

Fig. 14: Discriminator accuracies over time.
All models trained for 20.000 iterations.

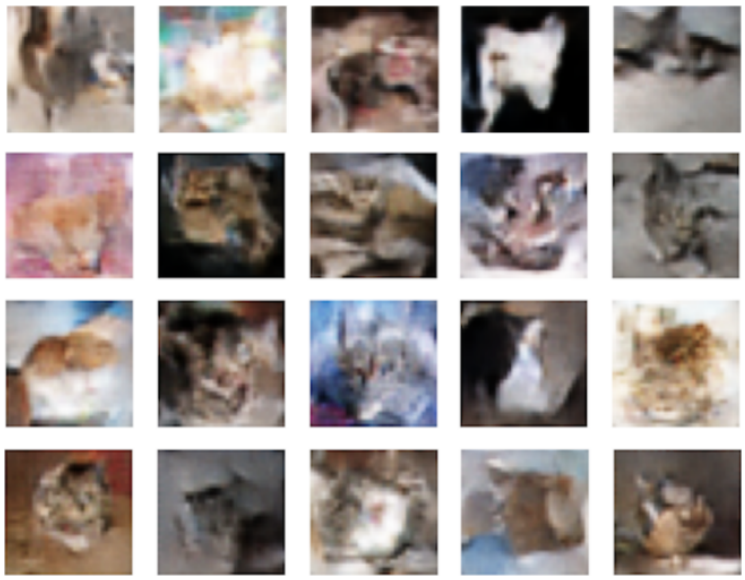
APPENDIX IV
FIXING VANISHING GRADIENT AND MODEL COLLAPSE



(a) Discriminator's accuracy.



(b) Losses.



(c) Random generations after 30.000 iterations.

Fig. 15: CGAN trained on class cat of CIFAR10.
Trained with smaller learning rate for Discriminator to avoid Vanishing Gradient and Generator's collapse.