I. Introduction

In this project, we applied optimization techniques on DCGAN by modifying model architecture to perform better face generation on Anime Faces data. We analyze our model by looking at discriminator loss, generator loss, and the quality of the resulting image.

This paper is organized as follows, First, we describe our dataset and show the results of Exploratory Data Analysis. Next, we provide a description of the model, then we methodology in terms of model development and optimization, and discuss the comparison and analysis of results. Finally, we summarize our findings and discuss areas of growth. Our main goal is to gain a deep understanding of DCGAN, and our main contribution is to see how we can improve the quality of result images in the context of face generation.

As a brief result, our baseline model performed not very well; the faces it generated are fuzzy, smudgy, and not convincing with missing eyes, and mouths, and some of them can not be recognized as faces. And with the modification, the resulting image becomes more clear and more convincing.

II. Description of individual work

- Read papers and experiment optimization methods (model architecture, optimizer, Gaussian Noise, adjust label smoothing)
- Analyze the performance of baseline model and optimized model
- Data description

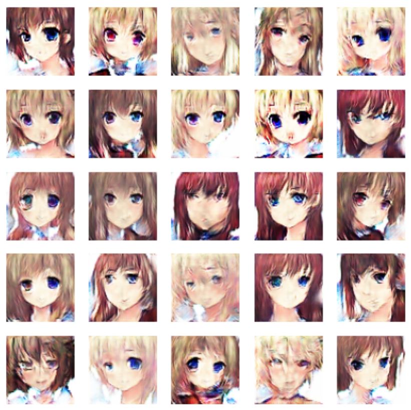
In this project, I am mainly working on research applications to improve DCGAN performance, and implement methods/thoughts from papers, also helping to analyze the performance between baseline model and optimized model. For the paper and presentation, my work is primarily related to the introduction, data description, model description, experiment set, methodology, and summary.

For the model architecture, I worked on research and implement methods from papers. I modified the architecture of the generator and discriminator based on the inspiration from papers. I added more layers, and a batch normalization layer after the input layer which can hasten the convergence of the model, also added a batch normalization layer after the Conv2DTranspose layer to sample, deconvolute and normalize the data, also used ReLU instead of LeakyReLU activation function for generator. For the discriminator, I added batch normalization layers after Conv2D layers because after the image is convoluted, normalization is needed, and activation remains as the LeakyReLU activation function. For the optimizer, I tried the Yogi optimizer with the modified model, but it didn't generate good results. The detail of the modification is shown below.

1. **BatchNormalization in discriminator:** Added batch normalization in the discriminator can help improve the stability of the training process by normalizing the activations of the previous layer, reducing the internal covariate shift, which can make it easier for the network to learn and converge to a good solution. Additionally, batch normalization can help prevent overfitting and improve the generalization performance of the discriminator. And the model with batch normalization in the discriminator has better performance on the result.

2. **Change activation function in generator:** In the baseline model, it has a LeakyReLU activation function for each layer of the generator. But the model with the ReLU activation function has achieved better image quality for the result. Because the generator is responsible for generating realistic images from random noise, it needs to be able to produce diverse and meaningful features. In order to do so, the ReLU activation functions can be more effective than LeakyReLU in producing sparse and diverse features, which can lead to more diverse and realistic images. Additionally, ReLU can help the generator produce sharper and more detailed images, which is important for generating high-quality images that are indistinguishable from real images.

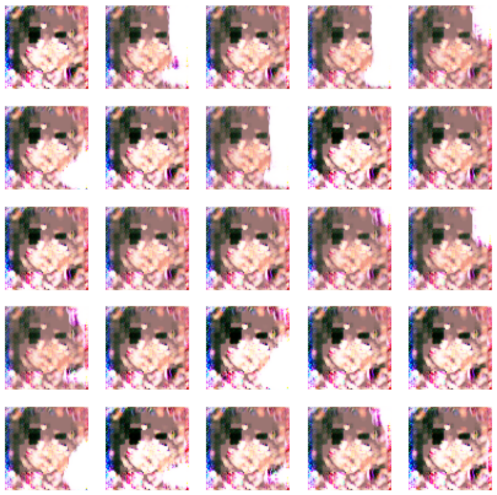
    - The result below shows the result of the modified model structure with more layers, batch normalization in the discriminator, and ReLU activation function in the generator. The model performance is clearly improved as shown below.



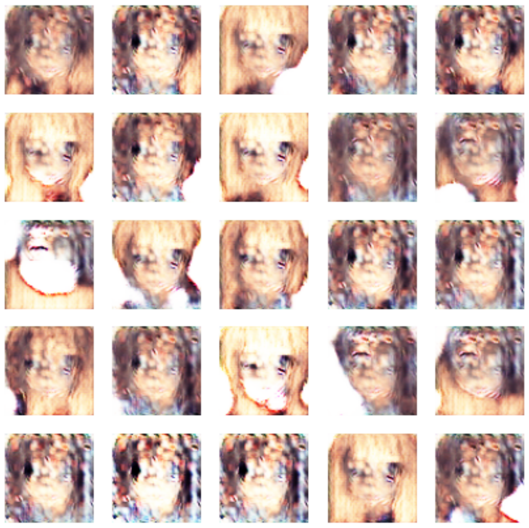
3. **Adding noise in discriminator:** While taking a look at the resulting image from the original code I notice the generated faces are smudgy, fuzzy, and not convincing, and easily collapsing. Therefore, I did a search on how to stable the generating process of DCGAN, and from the paper *Improved Techniques for Training GANs*, authors suggest adding Gaussian noise to the output of each layer of the discriminator, and I added it to the discriminator for the final project. According to the paper, adding noise to the discriminator can help stabilize the training process and improve the performance of the network by forcing the network to learn more robust features that are not affected by small variations in the input data,

preventing mode collapse, and improving stability by adding noise to the discriminator’s inputs can make it more difficult for the discriminator to distinguish between real and fake images, which can improve the stability of the training process and prevent the discriminator from becoming too dominant and overpowering the generator. But the result of adding noise training on anime face data performs not very well.

- Result of adding noise after first input layer

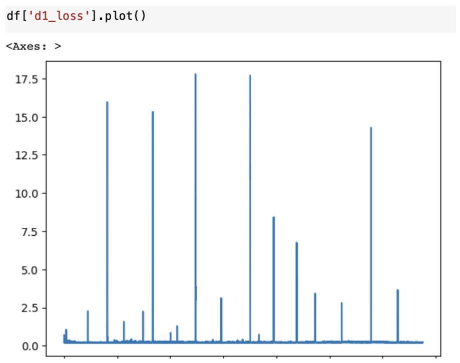


- Result of adding noise after each layer



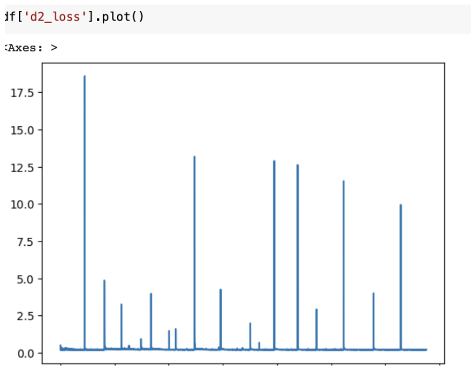
III. Results

The best result that I have achieved is using the modified model. Baseline model with ReLU in generator, and batch normalization in discriminator. As it is shown below, the loss of d1 and d2 is about 0.2, and g1_loss is about 3. Indicated that the discriminator is good at distinguishing fake and real images and the generator is perform not very well in generating high-quality fake images.



```
df['d1_loss'].plot()
```

```
<Axes: >
```

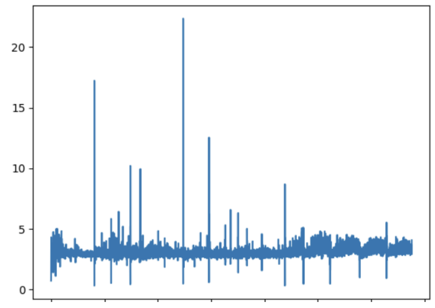


```
f['d2_loss'].plot()
```

```
Axes: >
```

```
df['g_loss'].plot()
```

```
<Axes: >
```



IV. Summary and Conclusions

In conclusion, this project is mainly on performing anime face generation tasks on anime face data using GANs based on replacing forward feed layers with deconv and conv layers. We fine-tuned the model by adjusting the model architecture, some hyper-parameters like batch size, kernel size, and learning rate.

In this project, while implementing all the work this project required, I feel I have gained a deeper understanding of GANs and am able to connect what I have learned from class, also it provides a great chance for me to do exploration and application. While searching for applications for improving DCGAN, I am getting to know more about model structures of GANs, loss function, and how it works to affect model generating high-quality images.

For the improvement of this project, If we have more time we would like to focus on three parts, the first is to do more research on loss function, and the second is to improve the data preprocessing, the third is hyperparameter turning. For the data, it might have some bad cropping results or some non-human faces so we should do more data exploration and also try to find out how to correctly add Gaussian noise to the discriminator. For the loss function and hyperparameters, we should do more research and implementation on it.

V. Code Calculation

Code_from_internet.py : 351lines
Code_edited.py :160 lines
Code_added.py: 30 Lines
Code from internet : 50%

VI. Reference

https://machinelearningmastery.com/how-to-develop-a-conditional-generative-adversarial-network-from-scratch/
https://github.com/hwalsuklee/tensorflow-generative-model-collections
https://machinelearningmastery.com/how-to-evaluate-generative-adversarial-networks/
https://www.hindawi.com/journals/misy/2022/9005552/
https://arxiv.org/pdf/1801.09195.pdf
https://github.com/nikhilroxtomar/DCGAN-on-Anime-Faces/blob/master/gan.py
https://arxiv.org/abs/1606.03498
https://pyimagesearch.com/2020/11/16/gans-with-keras-and-tensorflow/
https://proceedings.neurips.cc/paper_files/paper/2018/file/90365351ccc7437a1309dc64e4db32a3-Paper.pdf
https://paperswithcode.com/method/label-smoothing
https://arxiv.org/pdf/1511.06434v2.pdf
https://www.kaggle.com/datasets/soumikrakshit/anime-faces