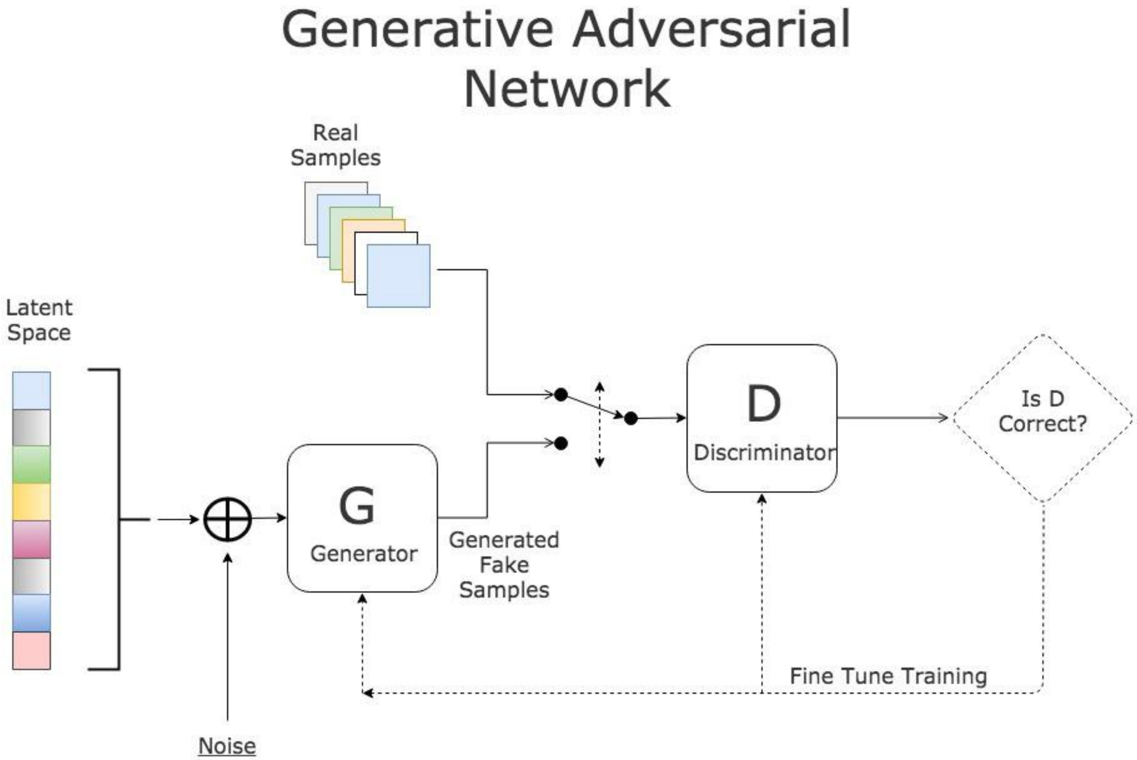
Generative Adversarial Networks (GANs)

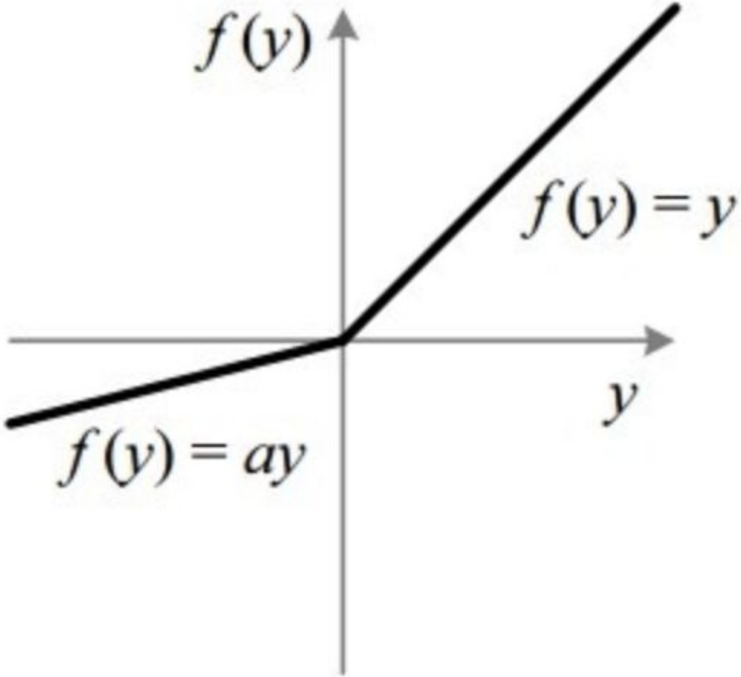
Group 3

What is a GAN?


Generative Adversarial Network
Real Samples
Latent Space
G
Generator
Generated Fake Samples
D
Discriminator
Is D Correct?
Fine Tune Training
Noise

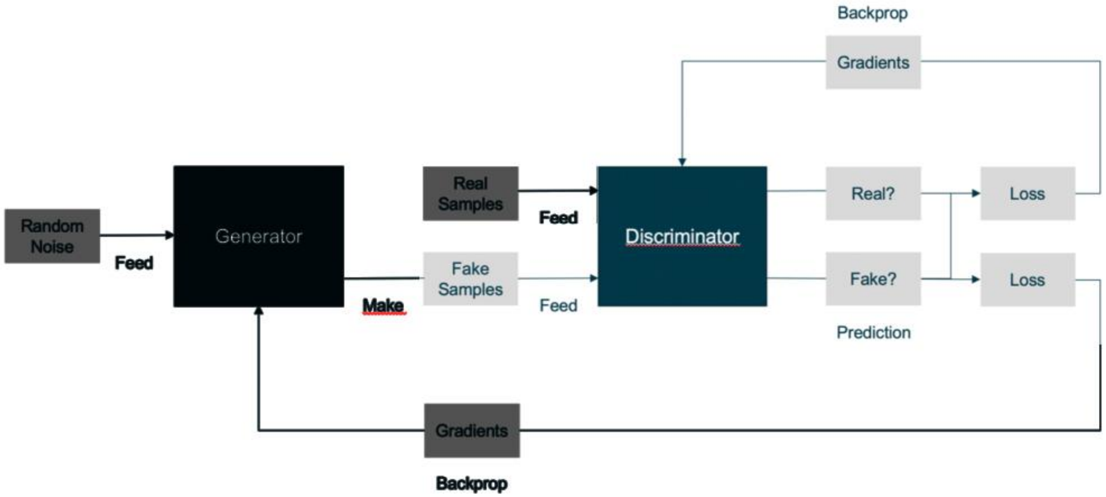
Activation Function

LeakyRelu.



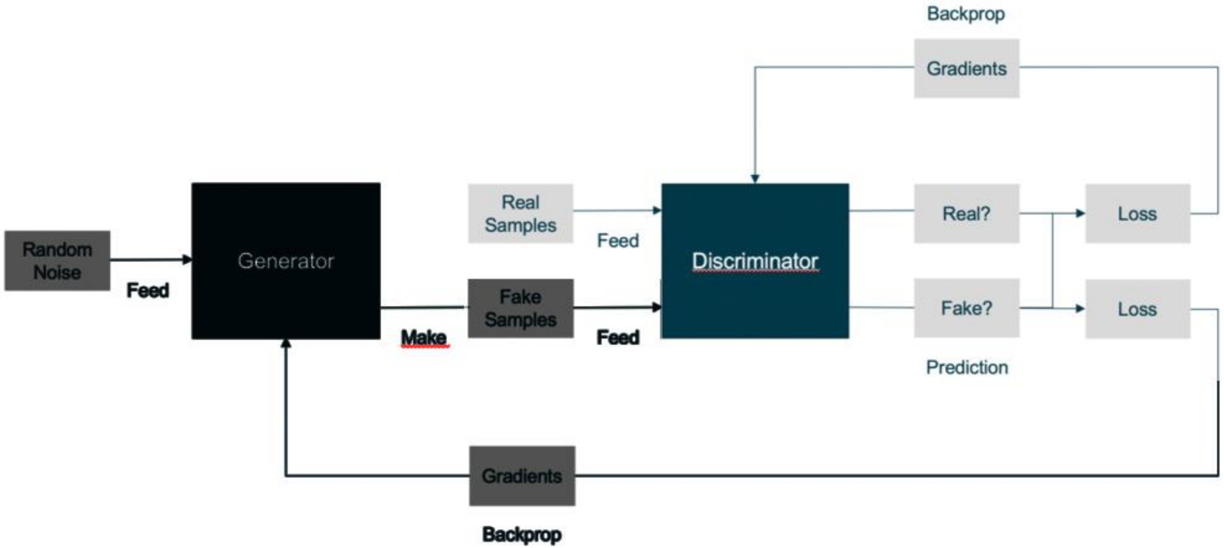
Training a GAN

Freeze the Generator and train the Discriminator once on generated samples. You get d1_loss



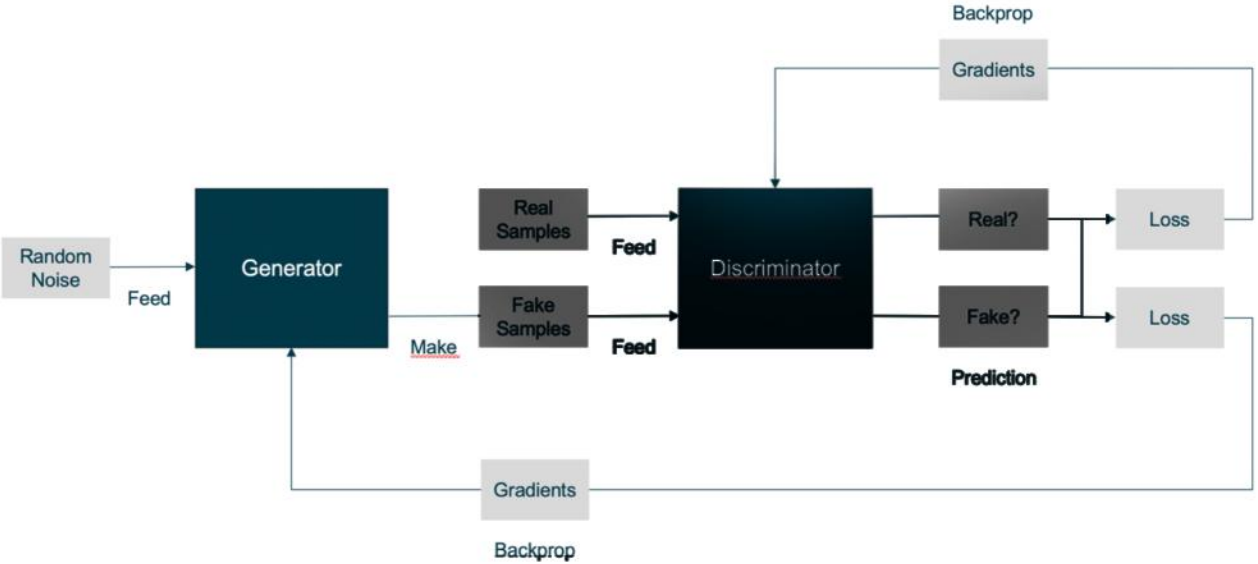
Training a GAN

Freeze the Generator and train the Discriminator once on ral samples. You get d2_loss



Training a GAN

Freeze the Discriminator and train the Generator with Latent vector. You get g_loss



Training a GAN

The goal is to minimize generator loss and maximize discriminator loss.

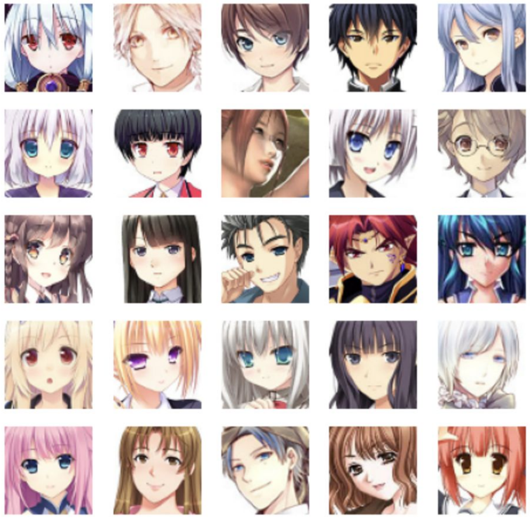
Meaning:

If g loss is small, it means the generator is able to produce very realistic images

If d loss is high, it means it is not able to classify images into real or fake, meaning the generator is doing a good job.

Data Description

- This is a dataset consisting of 21551 anime faces
- All images are resized to 64 * 64 for the sake of convenience.

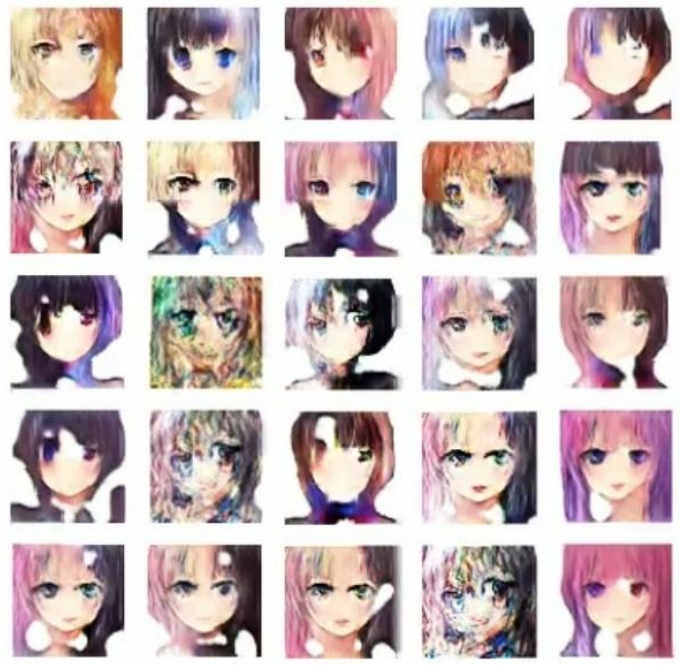


Baseline Model Description

- Generator
  - Input layer
  - Conv2DTranspose layer
  - Batchnormalization layer
  - LeakyReLU()
  - Conv2D layer
  - Tanh()

- Discriminator
  - Input layer
  - Conv2D layer
  - LeakyReLU()
  - Dropout(rate=0.3)
  - Flatten layer
  - Fully connected layer

Generated Samples of Baseline Model



Optimization Methods

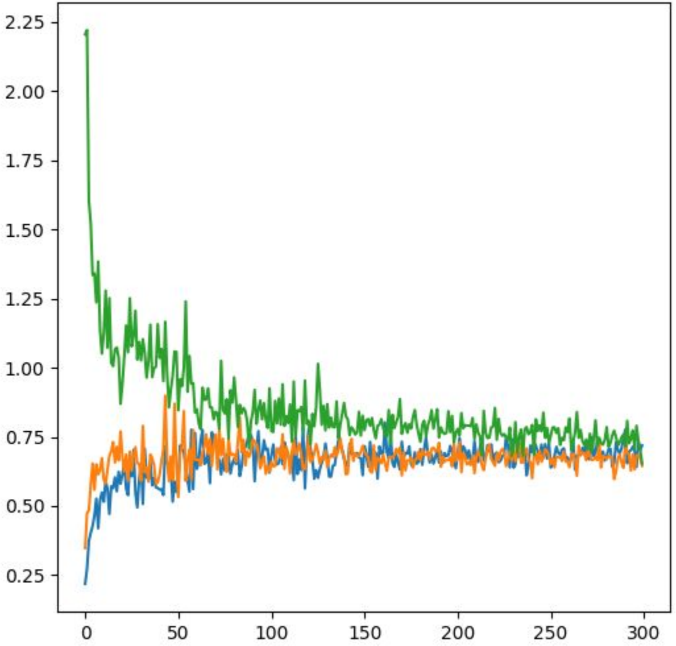
- **Adjust the learning rate (0.0001, 0.0002, 0.0005)**
- **Add more convolutional layers**
- **Use a different loss function (MeanSquaredError, Hinge Loss)**
- **Use a different optimizer (SGD,RMSprop)**
- **Changing Kernel Size**
- **LeakyRelu Value(0.05,0.1,0.5,0.2)**
- **Adding more Filters**
- **Increase the number of training epochs**
- **BatchNormalization in discriminator**
- **Change activation function in generator**
- **Adding noise in discriminator**

Losses

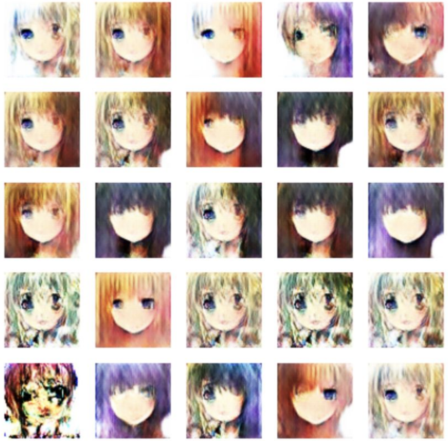
Generator

Discriminator(Generated Samples)

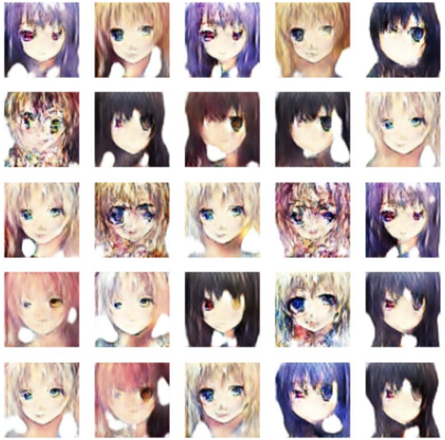
Discriminator(Real Samples)



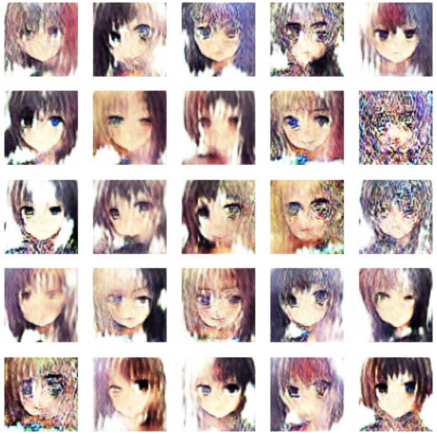
Results



LR:0.0001

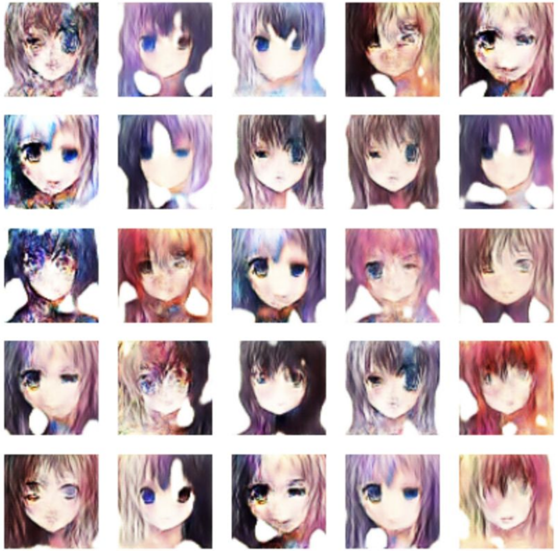


LR:0.0002

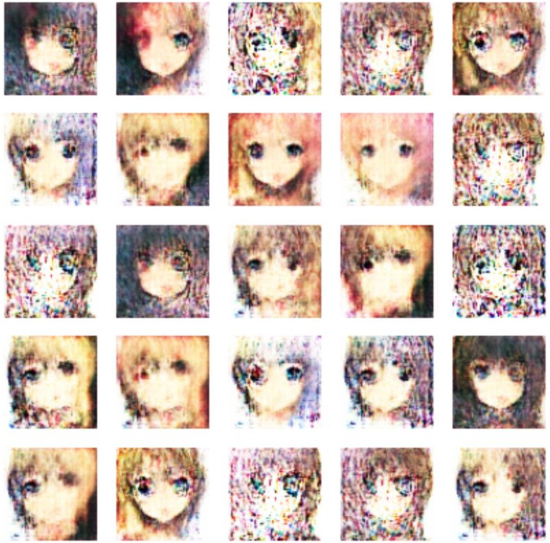


LR:0.0005

Results

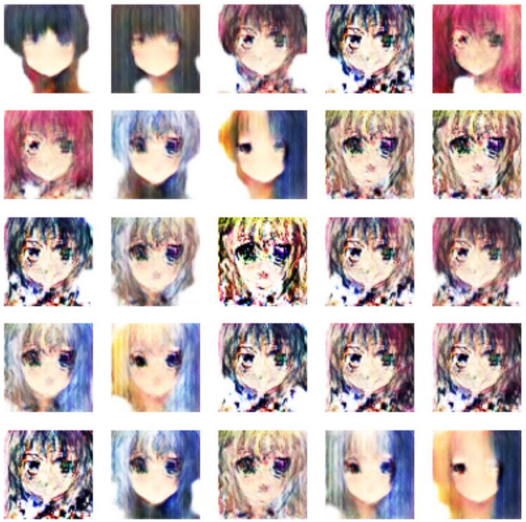


SGD

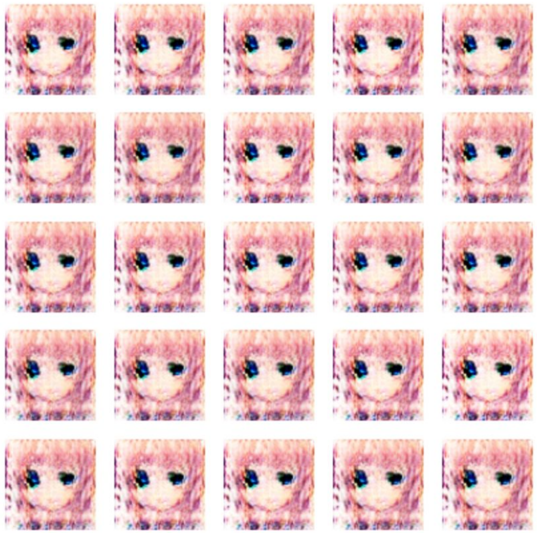


RMSprop

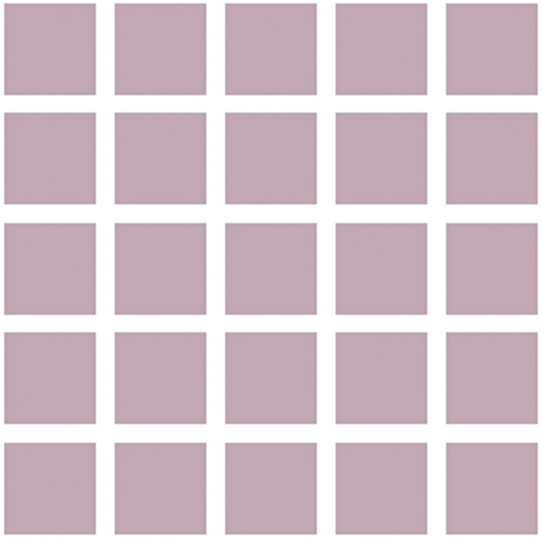
Results



Hinge loss



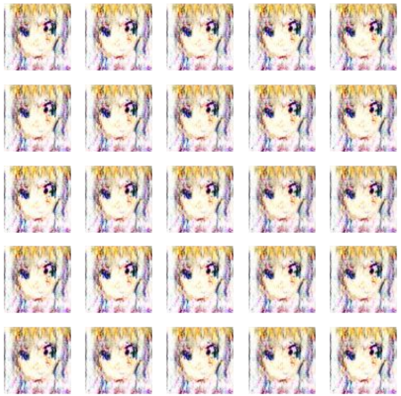
MeanSquaredError



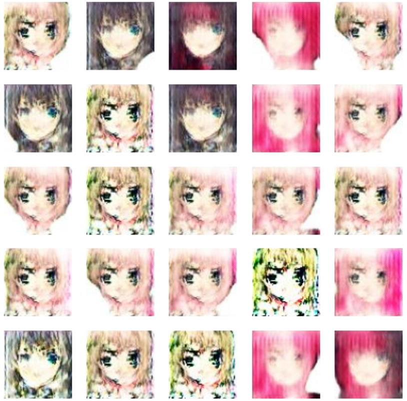
kl_divergence_loss

Results

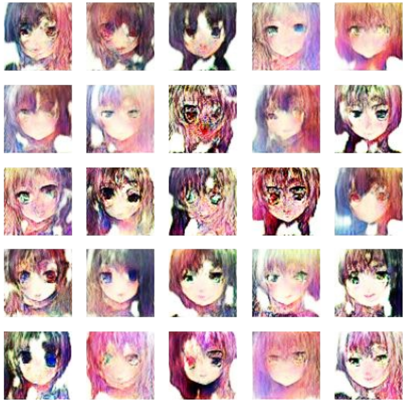
LeakyRelu Value



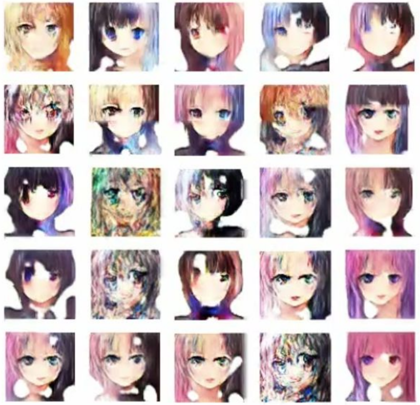
0.05



0.1



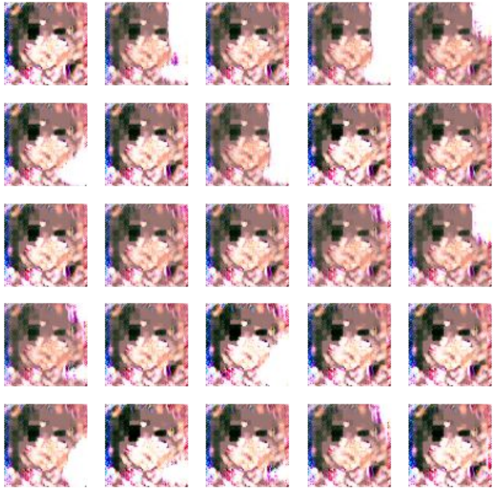
0.5



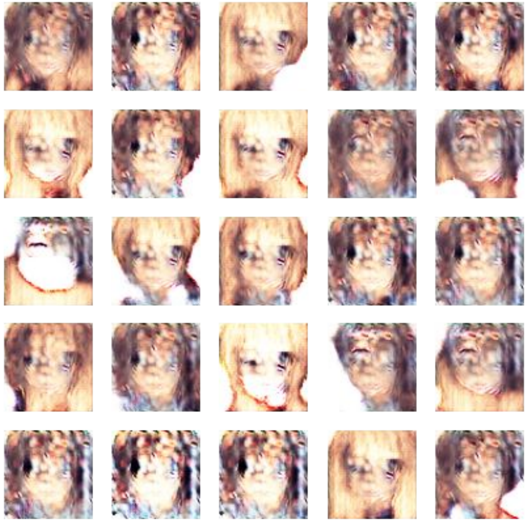
0.2

Results

Adding noise in discriminator



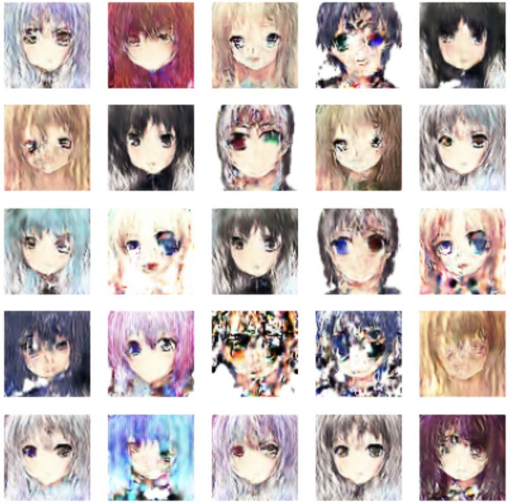
adding noise after first input layer



adding noise after each layer

Results

Filter, epochs

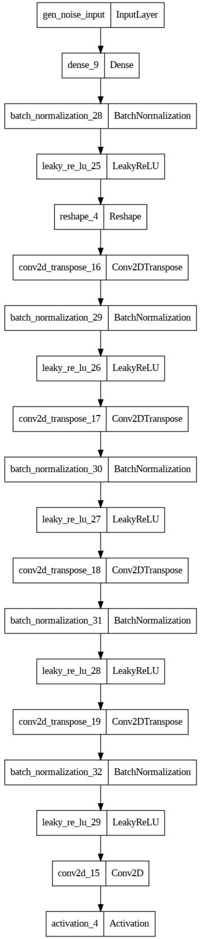


Optimized Model Description

- Generator
  - Input layer
  - Conv2DTranspose layer
  - Batchnormalization layer
  - ReLU()
  - Conv2D layer
  - Tanh()

- Discriminator
  - Input layer
  - Conv2D layer
  - Batchnormalization layer
  - LeakyReLU()
  - Dropout(rate=0.3)
  - Flatten layer
  - Fully connected layer

Architecture



gen_noise_input | InputLayer

dense_9 | Dense

batch_normalization_28 | BatchNormalization

leaky_re_lu_25 | LeakyReLU

reshape_4 | Reshape

conv2d_transpose_16 | Conv2DTranspose

batch_normalization_29 | BatchNormalization

leaky_re_lu_26 | LeakyReLU

conv2d_transpose_17 | Conv2DTranspose

batch_normalization_30 | BatchNormalization

leaky_re_lu_27 | LeakyReLU

conv2d_transpose_18 | Conv2DTranspose

batch_normalization_31 | BatchNormalization

leaky_re_lu_28 | LeakyReLU

conv2d_transpose_19 | Conv2DTranspose

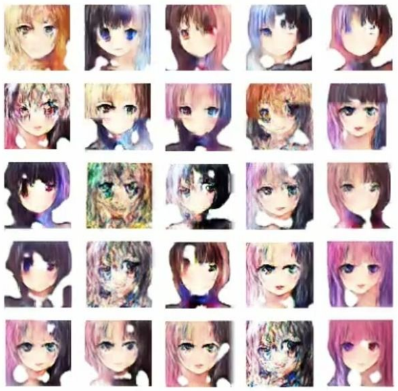
batch_normalization_32 | BatchNormalization

leaky_re_lu_29 | LeakyReLU

conv2d_15 | Conv2D

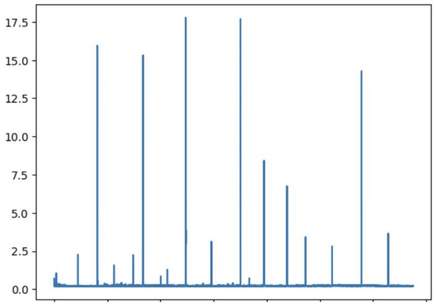
activation_4 | Activation

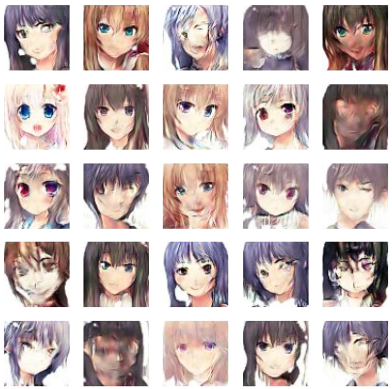
Optimized Model Result



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

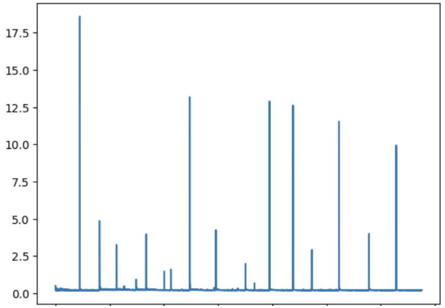
```
<Axes: >
```

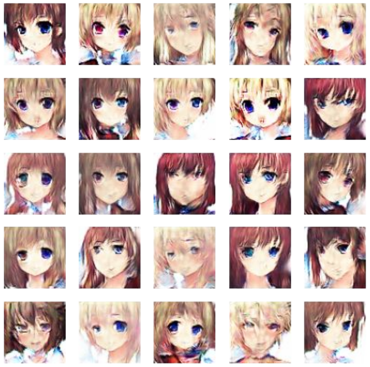




```
if['d2_loss'].plot()
```

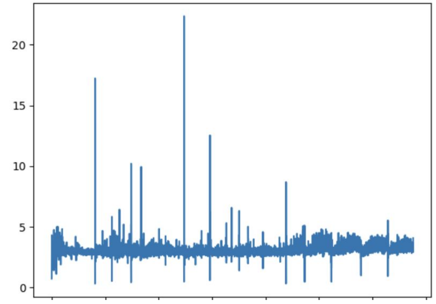
```
<Axes: >
```



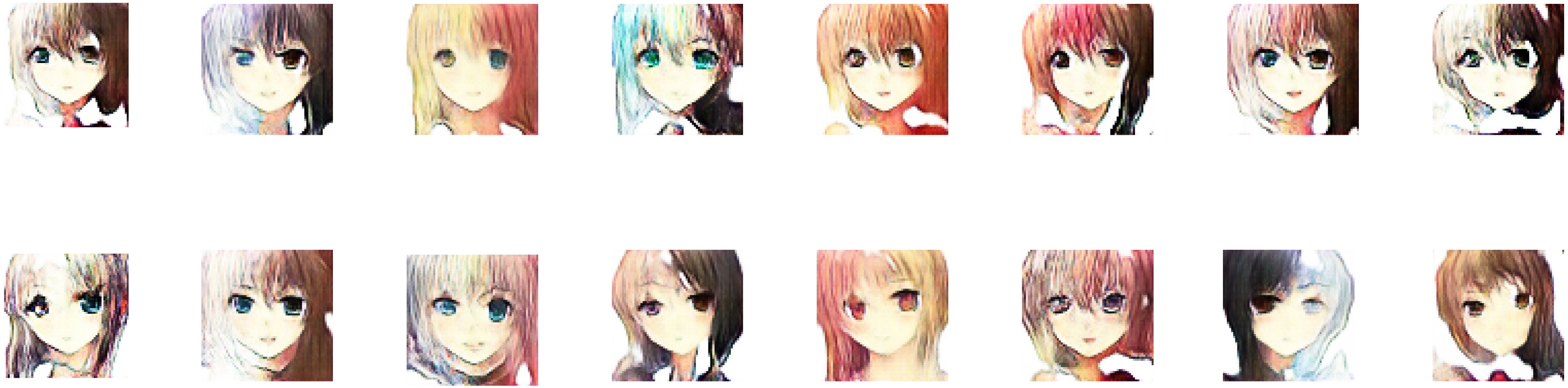


```
df['g_loss'].plot()
```

```
<Axes: >
```



Best Result



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