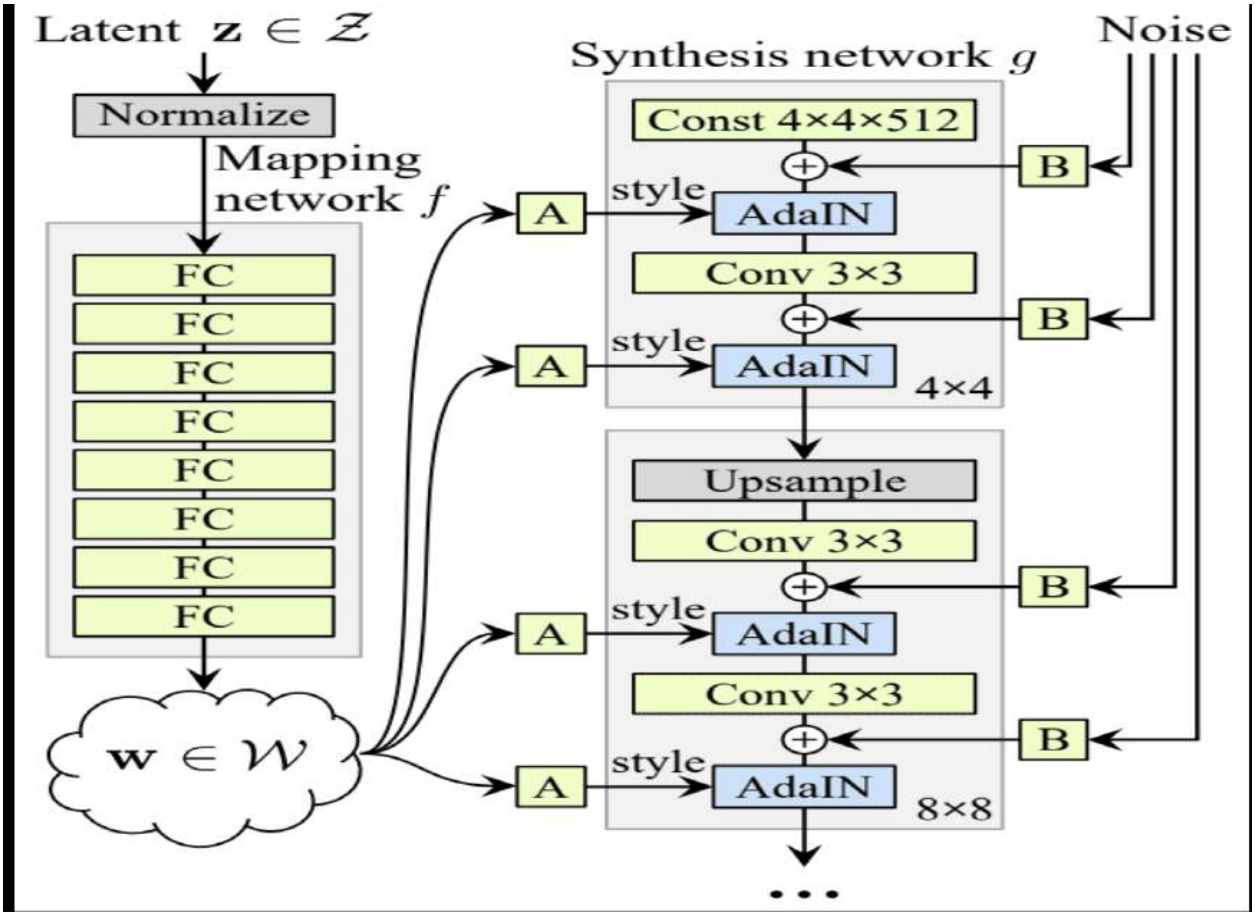
Indian Institute of Technology Delhi
ELL793 Assignment-3
Submitted by: *Sajal Tyagi [2019MT60761] , Ayush Goyal [2019MT10961]*

1 StyleGAN

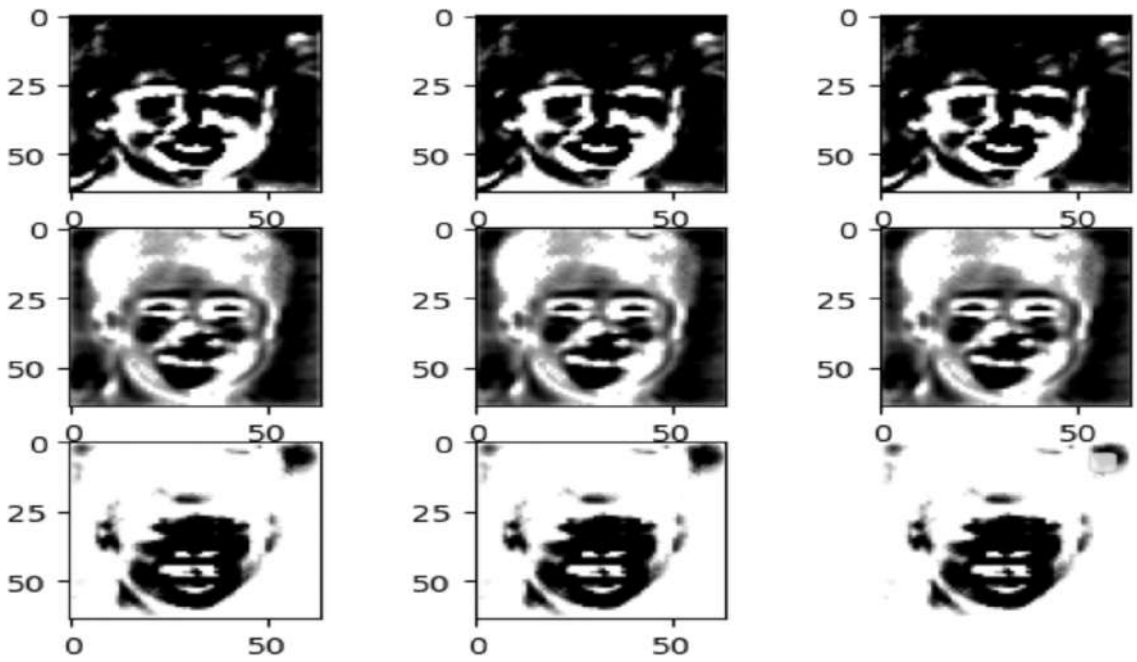
1.1 Model Architecture and Layer Weights

The architecture for the model is represented by the following figure -



We print the model class that we created and the printed output can be found in style gan print.txt

We visualize the outputs of the 4th block G-synthesis i.e. the collection of blocks. Here we present a grid of few of the filters:

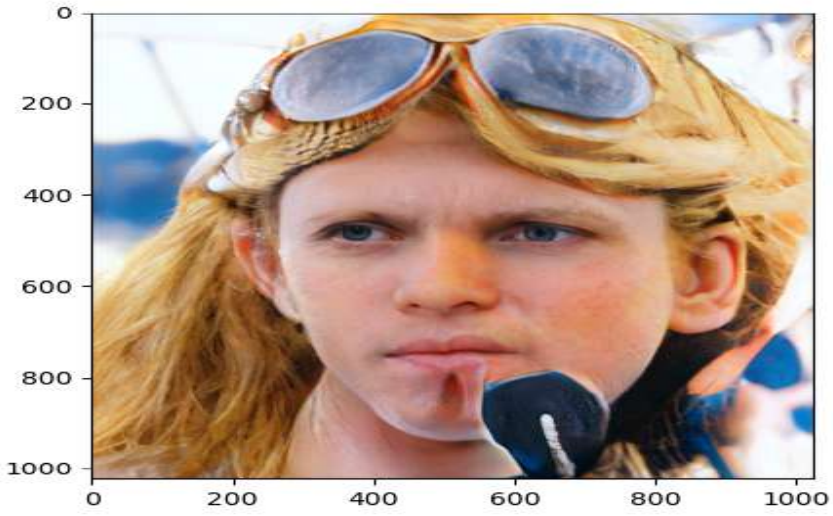


1.2 Style Mixing of 2 Latent Layer Vectors

The results for style mixing of the latent vectors obtained after applying the G-mapping layers have been provided below.

Output image generated from 1st Latent Vector

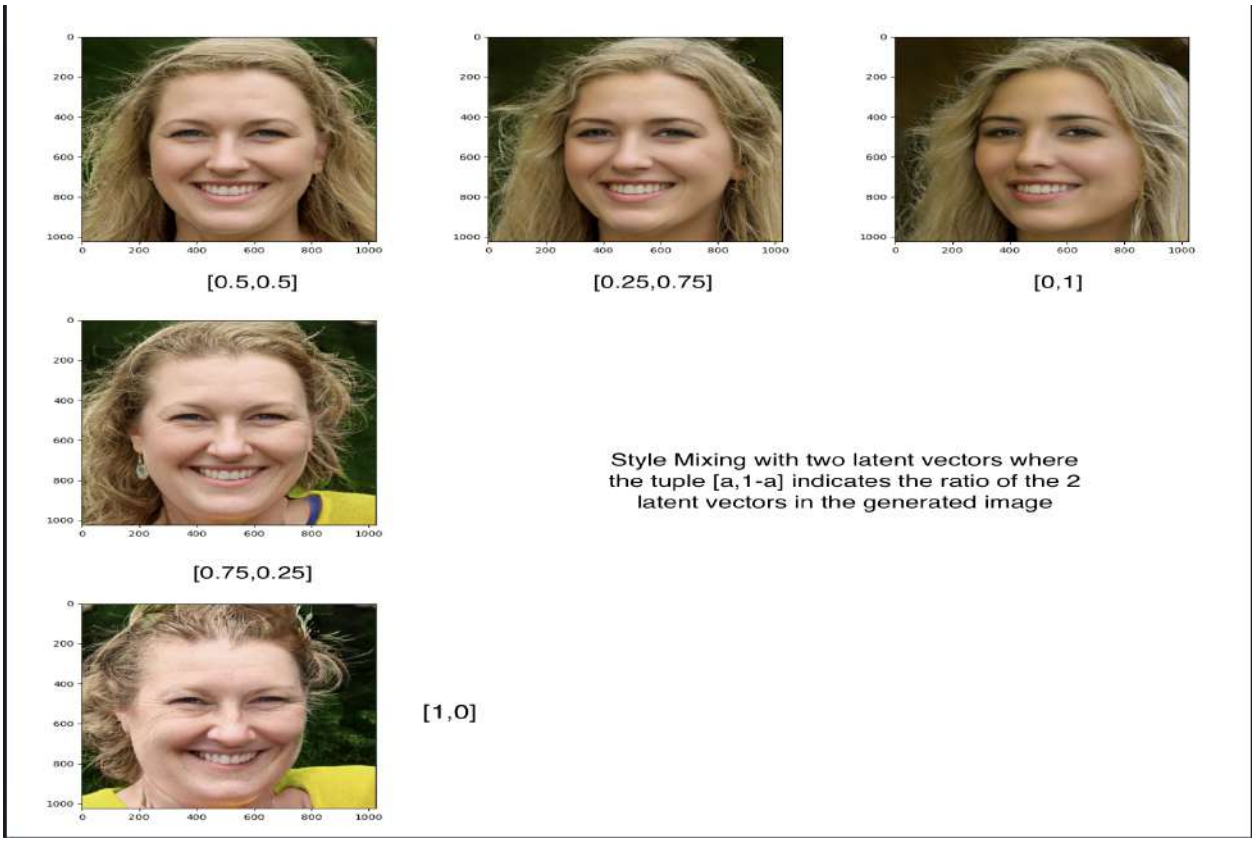
Output image generated from 2nd Latent Vector



Output image generated from the mixed Latent Vector

Output image generated from the mixed Latent Vector after mapping layers

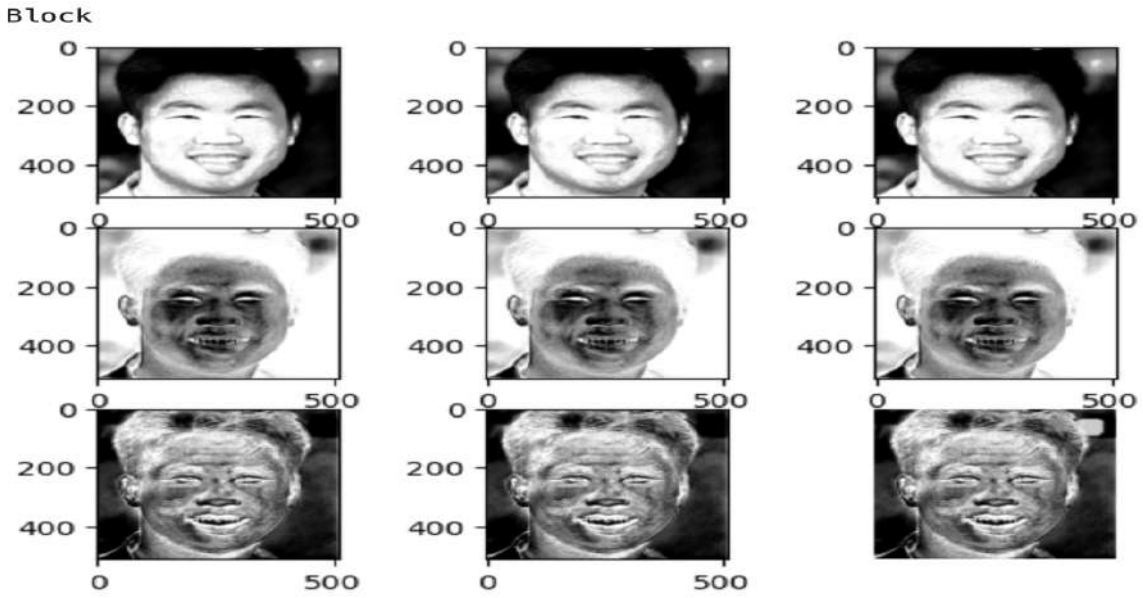
We can clearly see from the above images that style mixing works better than raw latent mixing, as in initial case we have a much better image as compared to the latter one.



Mixing latents

1.3 Layer 4 weights

We visualize the outputs of the 8th block G-synthesis i.e. the collection of blocks. Here we present a grid of few of the filters: :



Here, we can see that the activation's after the 8th block are much better than the outputs after the 4th Block. We can see clear human faces in later output, while we have distorted images in the initial output.

1.4 Concept of Style Mixing

Style mixing is a technique introduced in StyleGAN that allows for the mixing of different styles (i.e., the learned latent vectors that control the generated images) in different regions of an image. Specifically, style mixing refers to the process of sampling two sets of style vectors from the model, one for the lower resolution levels and one for the higher resolution levels, and then combining them to form a new set of style vectors that can be used to generate an image. The basic idea behind style mixing is that it enables the creation of more complex and varied images by allowing different styles to be used in different parts of the image. For example, one could use a style vector that corresponds to a particular hair style in the upper levels of the model and a style vector that corresponds to a particular eye color in the lower levels of the model. By combining these style vectors, one can create an image that combines the hair style and eye color in a natural and seamless way. Style mixing differs from traditional style transfer methods in several ways. First, style mixing is a generative approach, meaning that it generates images from scratch rather than modifying existing images. Second, style mixing allows for the mixing of different styles at different

levels of the model, which enables the creation of more complex and varied images. Finally, style mixing is a fully unsupervised method, meaning that it does not require any explicit alignment or correspondence between the style vectors and the image content. The effect of style mixing on the generated images can be quite striking. By mixing different styles at different levels of the model, one can create images that are highly detailed and complex, with a rich interplay of different visual features.The ability to mix styles also enables the creation of highly realistic and convincing images, since it allows for the natural combination of different visual features that are often found together in the real world. Overall, style mixing is a powerful technique that has opened up new avenues for image generation and creative expression in the field of deep learning.

1.5 Layer Depth Effect

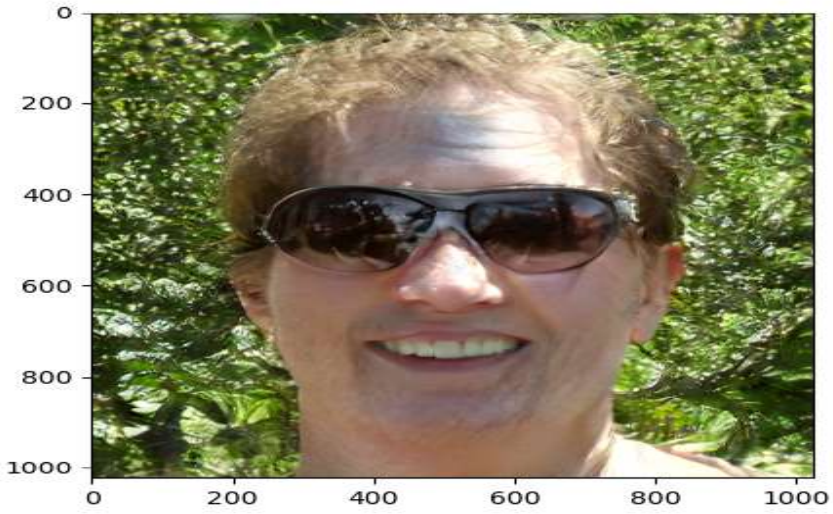
The depth of the chosen layer can have a significant impact on the quality of the generated images in style mixing. Specifically, choosing a layer that is too shallow or too deep can result in images that are of lower quality or less diverse.

If the chosen layer is too shallow (i.e., closer to the input layer), the mixed styles may not have enough resolution to fully capture the details of the image. This can result in images that are blurry or lack important features. On the other hand, if the chosen layer is too deep (i.e., closer to the output layer), the mixed styles may be too high-level and not capture enough of the fine-grained details of the image. This can result in images that are less diverse or less visually interesting.

In general, it's a good idea to choose a layer that is at an intermediate depth, where the styles have enough resolution to capture important details but are not too high-level. However, the ideal depth may depend on the specific dataset and task at hand, so it may require some experimentation to find the optimal depth for a particular application.

1.6 Style Mixing with more than 2 latent vectors

The results for style mixing of 5 latent vectors obtained after applying the G-mapping layers have been provided below.



Output image generated from the first latent Vector

Output image generated from the second Latent Vector

Output image generated from the third Latent Vector

Output image generated from the forth Latent Vector

Output image generated from the fixed Latent Vector

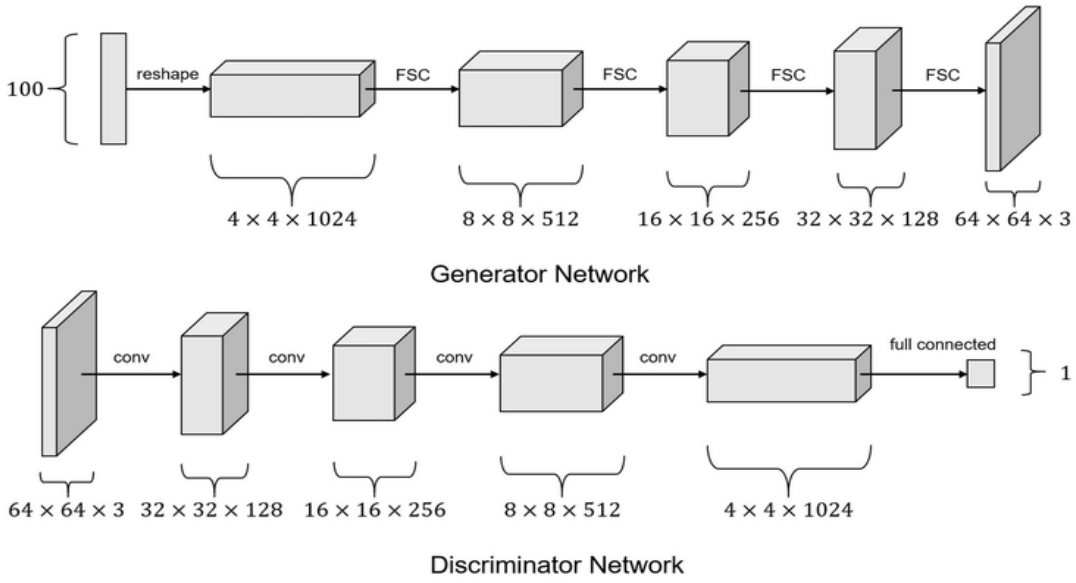
Output image generated from the mixed Latent Vector

2 GAN

2.1 DCGAN : Architecture

DCGAN, or Deep Convolutional GAN, is a generative adversarial network architecture. It uses a CNNs in both Generators and Discriminators. This architecture was proposed in the paper - "Un-

supervised Representation Learning with Deep Convolutional Generative Adversarial Networks" and we use that as a reference for attempting the question. The Neural Architecture for Generators and Discriminators can be found below -



2.2 DCGAN : Loss

The standard GAN loss function, also known as the min-max loss, was first described in a 2014 paper by Ian Goodfellow et al., titled "Generative Adversarial Networks". The generator tries to minimize this function while the discriminator tries to maximize it. Looking at it as a min-max game, this formulation of the loss seemed effective. In practice, it saturates for the generator, meaning that the generator quite frequently stops training if it doesn't catch up with the discriminator. The Standard GAN loss function can further be categorized into two parts: Discriminator loss and Generator loss.

2.2.1 Discriminator Loss

While the discriminator is trained, it classifies both the real data and the fake data from the generator. It penalizes itself for misclassifying a real instance as fake, or a fake instance (created by the generator) as real, by maximizing the below function.

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right] \tag{1}$$

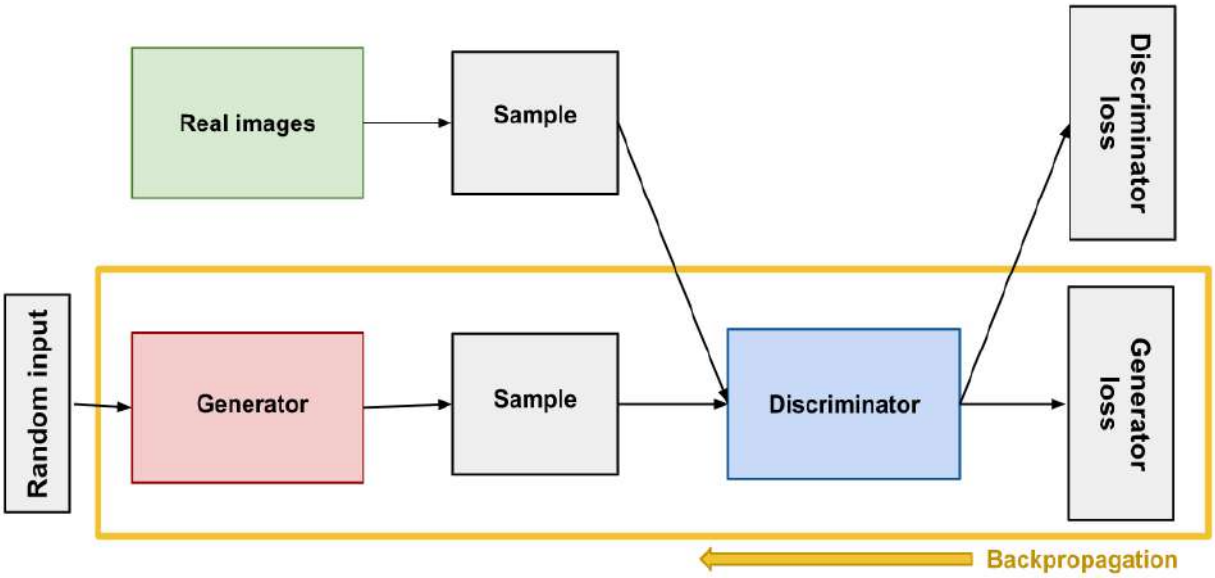
2.2.2 Generator Loss

While the generator is trained, it samples random noise and produces an output from that noise. The output then goes through the discriminator and gets classified as either "Real" or "Fake" based on the ability of the discriminator to tell one from the other.

The generator loss is then calculated from the discriminator's classification – it gets rewarded if it successfully fools the discriminator, and gets penalized otherwise.

The following equation is minimized to training the generator:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log \left(1 - D\left(G\left(z^{(i)}\right)\right)\right) \tag{2}$$

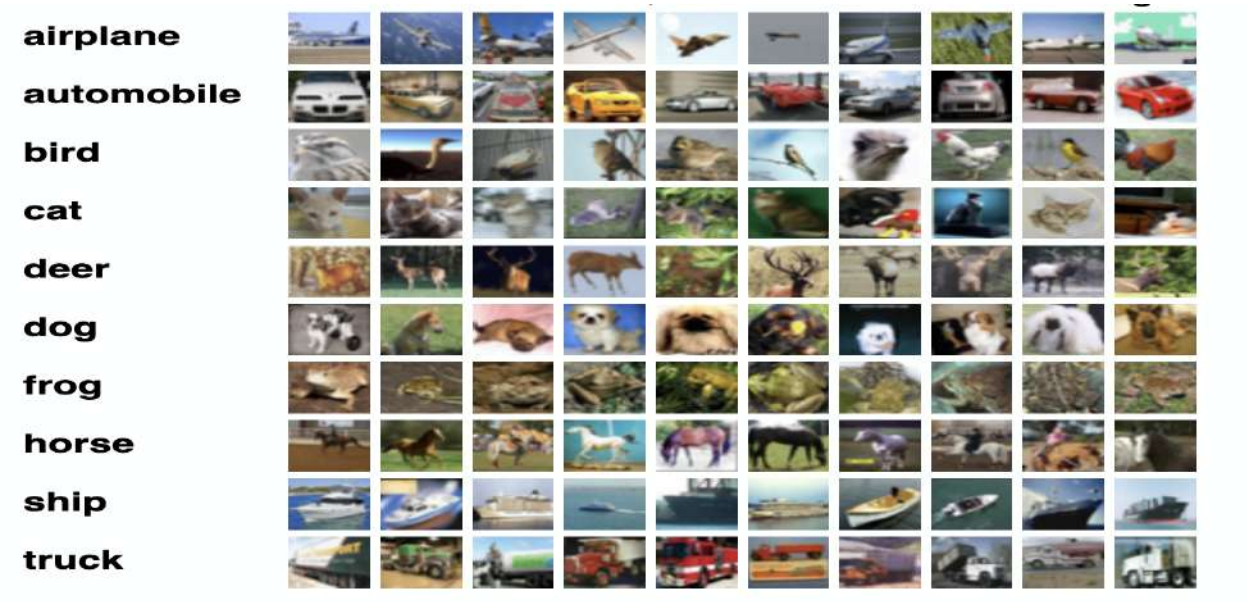


2.3 Dataset

The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images.

The dataset is divided into five training batches and one test batch, each with 10000 images. The test batch contains exactly 1000 randomly-selected images from each class. The training batches contain the remaining images in random order, but some training batches may contain more images from one class than another. Between them, the training batches contain exactly 5000 images from each class.

Here are the classes in the dataset, as well as 10 random images from each:



The classes are completely mutually exclusive.

2.4 Finding Optimal Learning Rate

We mention the fixed configuration used in our model below -

1. Batch Size for Training : 128
2. Input latent vector size : 100
3. β_1 value for Adam optimizer : 0.5
4. number of epoch to train : 25

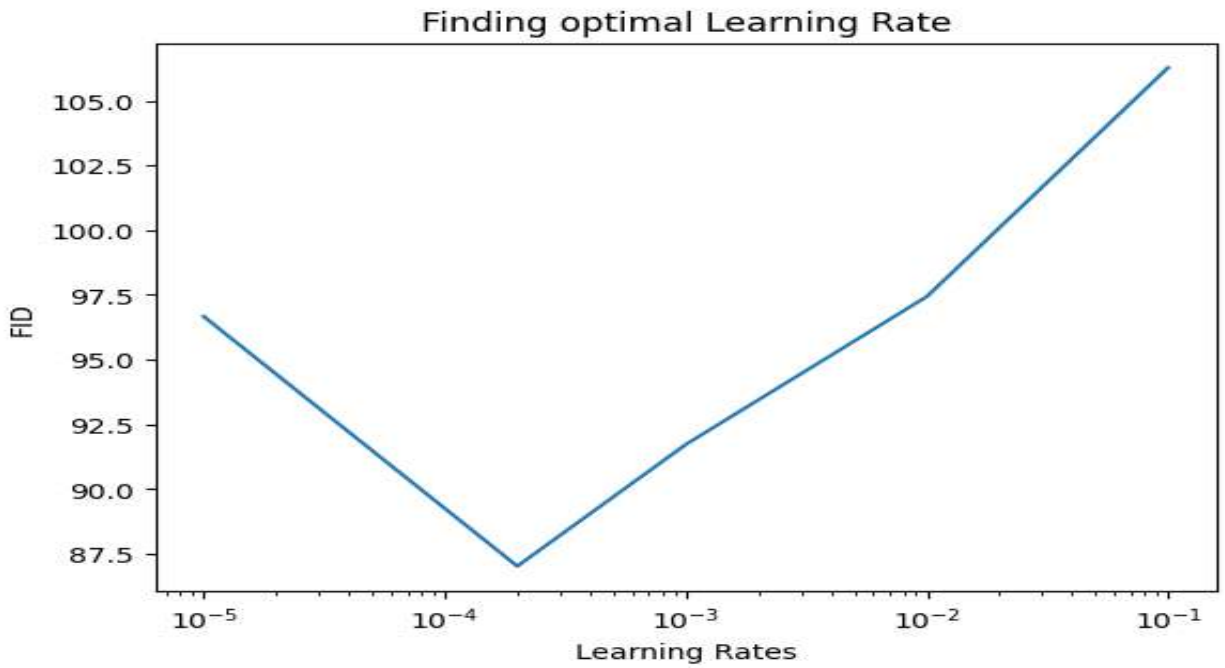
2.4.1 Metric for comparing GAN performance

Generator and Discriminator losses are commonly used to train Generative Adversarial Networks (GANs). However, comparing GAN performance using only these losses can be misleading. This is because the Generator and Discriminator losses do not always correlate with the quality of the generated images.

The Generator loss is a measure of how well the generated images match the real images, while the Discriminator loss is a measure of how well the Discriminator can distinguish between real and generated images. However, it is possible for the Generator loss to decrease while the Discriminator loss increases, and vice versa. This can happen when the Generator learns to generate images that are very similar to the real images but are still not quite convincing enough to fool the Discriminator.

On the other hand, the Fréchet Inception Distance (FID) is a metric that compares the statistics of the generated images to the statistics of the real images. The FID measures the distance between the distributions of features extracted from the generated and real images using an Inception

network, which has been pre-trained on a large dataset. The lower the FID, the better the quality of the generated images.
We generate 64 samples for each epoch and compare the FID of these with the CIFAR10 Dataset and use the best FID reached in first 25 iterators as the metric for comparing different learning rates. We plot our findings below :

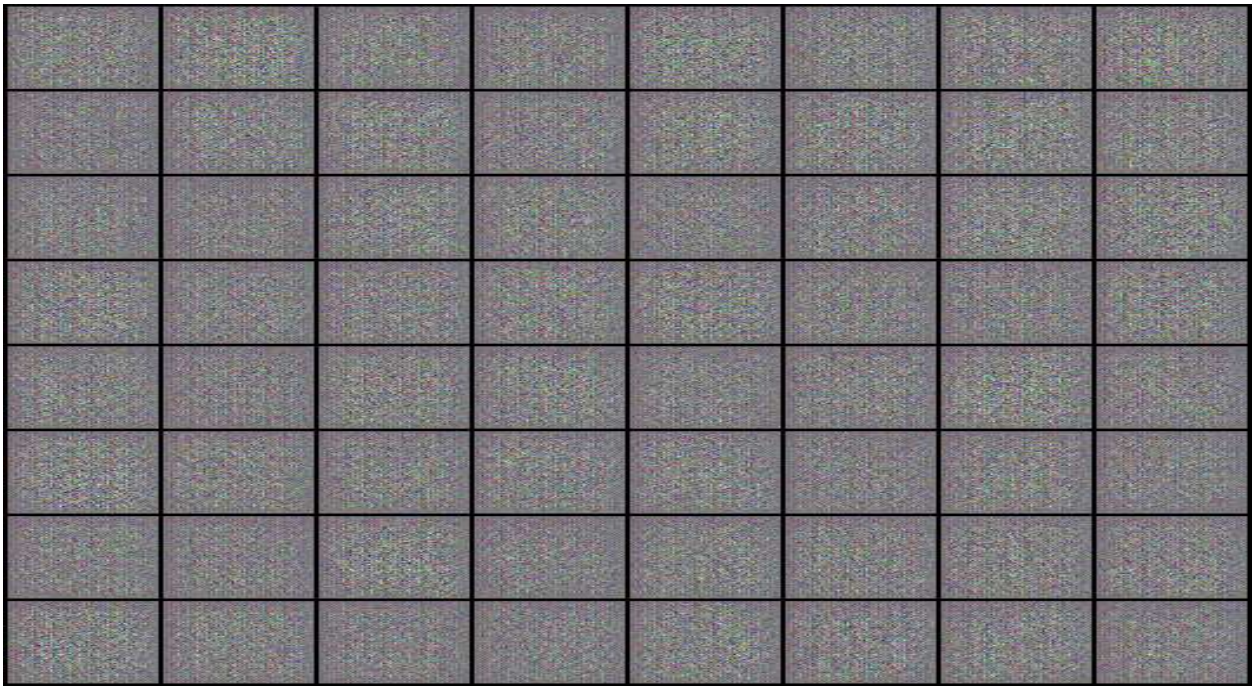


As we can see the author proposed LR of 0.0002 works best w.r.t FID.

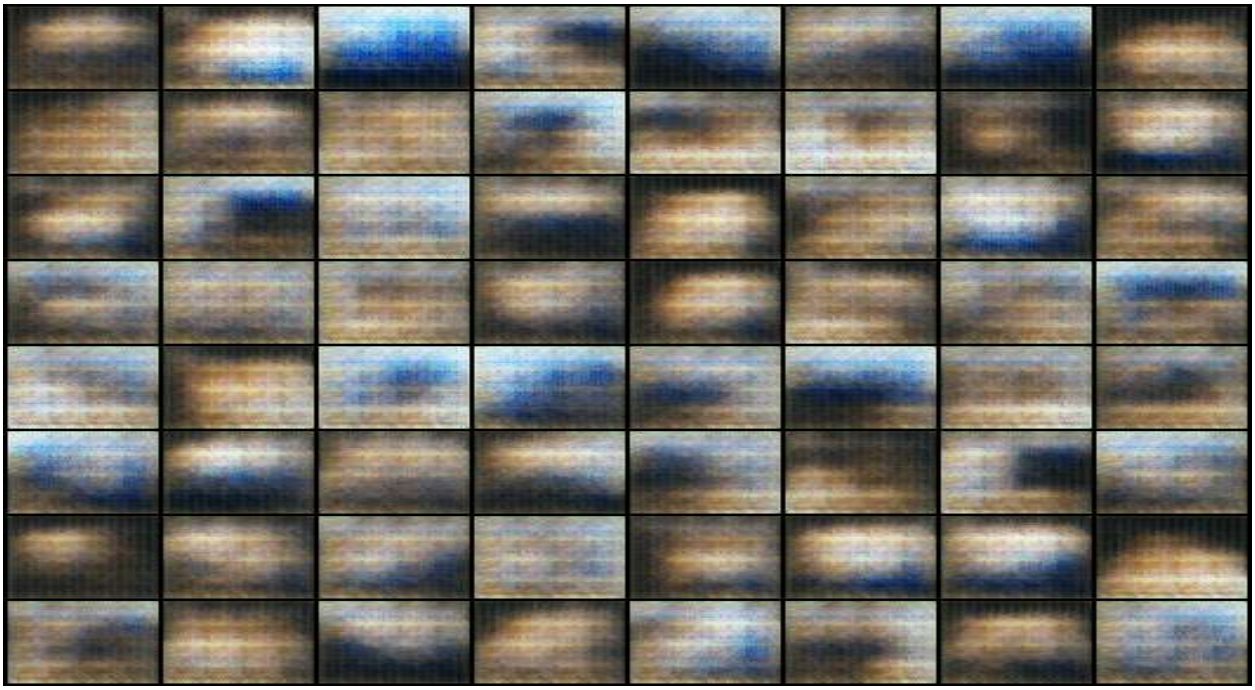
2.5 Generator Outputs

We take a look at the generator outputs before, during and after training for 25 epochs. We provide all the examples in the outputs folder inside the DCGAN folder and also provide a GIF which summarizes the whole training process through the outputs generated at each epoch.

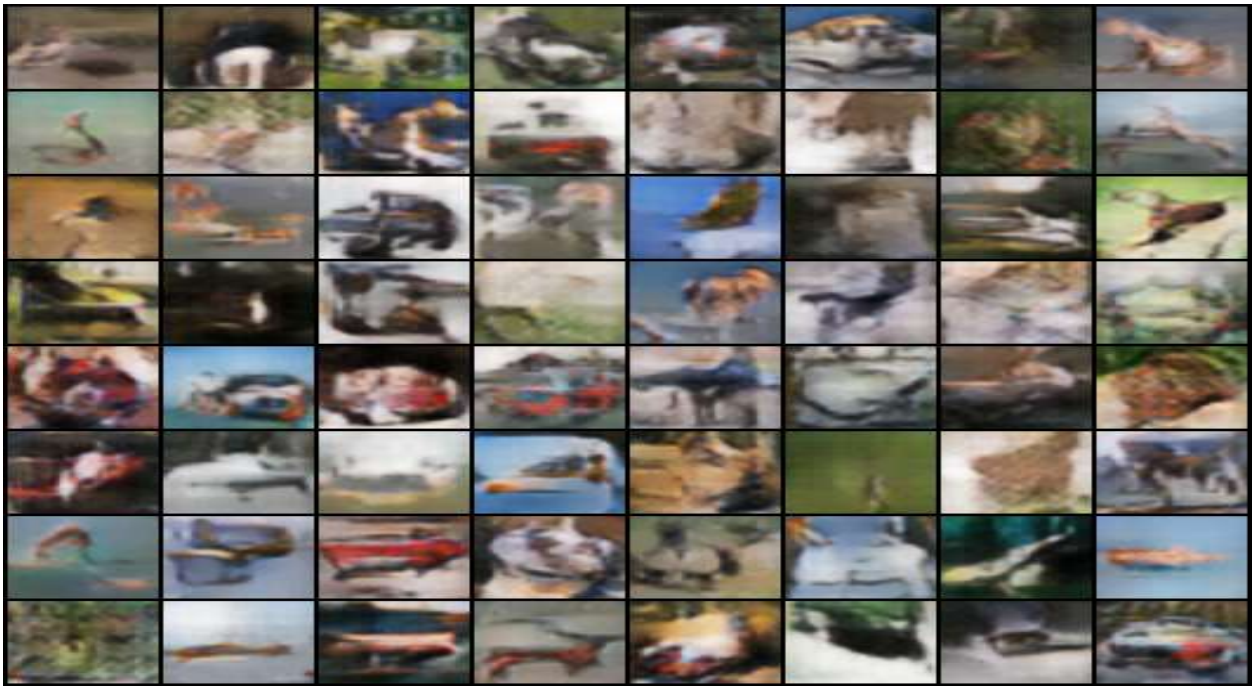
- Before Training -



- During Training -



- After Training -

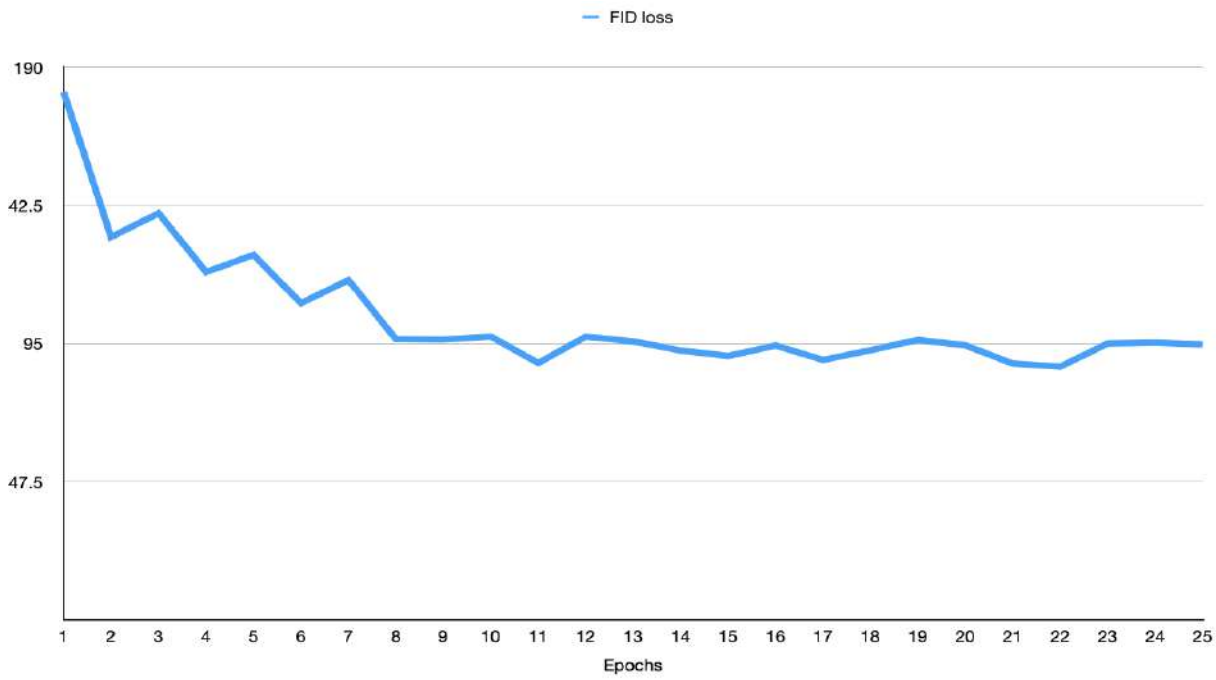


As we can see, the before training image just includes white noise with no discern-able structure in it. During training as epochs increase, outputs of better quality, which start to look similar to

images in the dataset get produced. This trend seems to continue and GANs benefit from repeated training; due to contraints in computing power and time, we weren't able to experiment till the author's prescribed epochs i.e. 3000+, but still these output examples are enough to validate the performance of GANS for image generation.

2.6 Training Plots

We provide the curve for Generator/Discriminator loss as well as the FID loss for each epoch of training. The final model was chosen to be the epoch with highest FID loss.



From this FID plot we choose the model obtained from the 22nd epoch to be the final model.