CPSC-8430-001

Deep Learning: HW-3

Dineshchandar Ravichandran
C19657741
Email: dravich@g.clemson.edu

Contents

Introduction

- In this assignment, we implement a Generative Adversarial Network's (GAN), an unsupervised network, to train a discriminator/generator pair on the CIFAR10 dataset utilizing techniques from DCGAN Wasserstein GANs and ACGAN to generate fake images.
- GANs (generative adversarial networks) are computational structures that pit two neural networks against one other (thus the name "adversarial") to produce fresh, synthetic examples of data that can pass for real data. They're commonly utilized in the picture, video, and voice generation.
- The discriminative network evaluates the data while the generative network generates new data. In this case, the generative network adjusts the map from the latent space to the interest network's data distribution from the genuine data distribution. A generative network aims to generate a fresh set of data. A discriminative network cannot tell the difference between the data generated by the generative network and the data generated by the discriminative network.

Git Hub Location:

https://github.com/DineshchandarR/CPSC-8430-Deep-Learning-001/tree/main/HW3

1. Problem Statement

- Train a discriminator/generator pair on the CIFAR10 dataset utilizing techniques from DCGAN, Wasserstein GANs, and ACGAN.

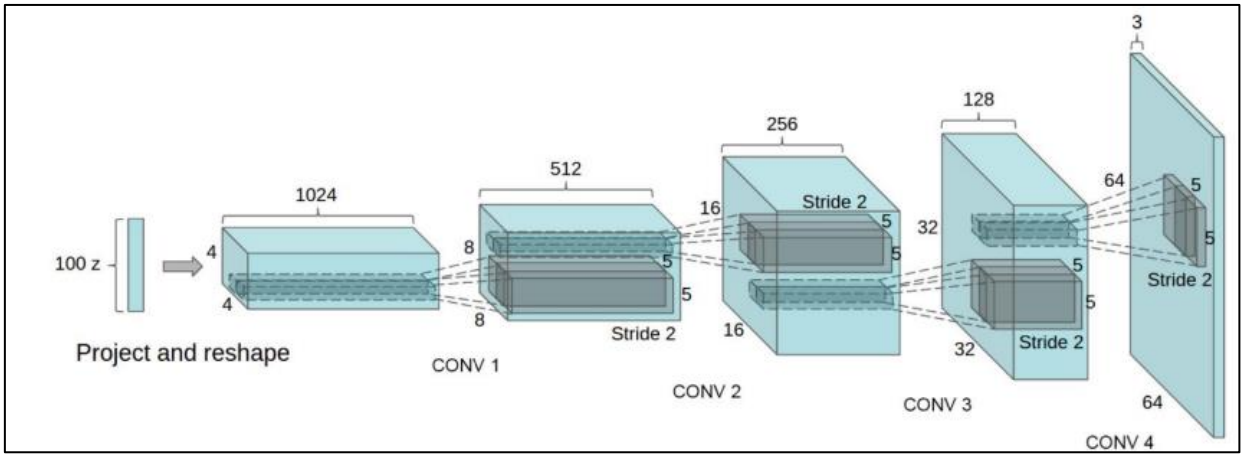
2. Implementation

2.1 Dataset:

- The CIFAR-10 dataset consists of 60000 32x32 color images in 10 classes, with 6000 images per class; the 10 classes are "Airplanes, Car, Birds, Cats, Deer, Dogs, Frogs, Horses, Ships and Trucks.
- Dataset is downloaded using the Pytorch's 'torchvision. Datasets' cifar-10 package.
- Since the images on the CIFAR-10 dataset are of inferior quality, the generated fake photos are of inferior quality.
- Furthermore, this dataset contains data from10 different classes, which, according to the studies I had referred to, makes it difficult for GAN networks to learn while training. So, I tried training with a single class, which yielded a somewhat better FID score. But, I've included the results from training the complete dataset in line with the clarification I received during the presentation.

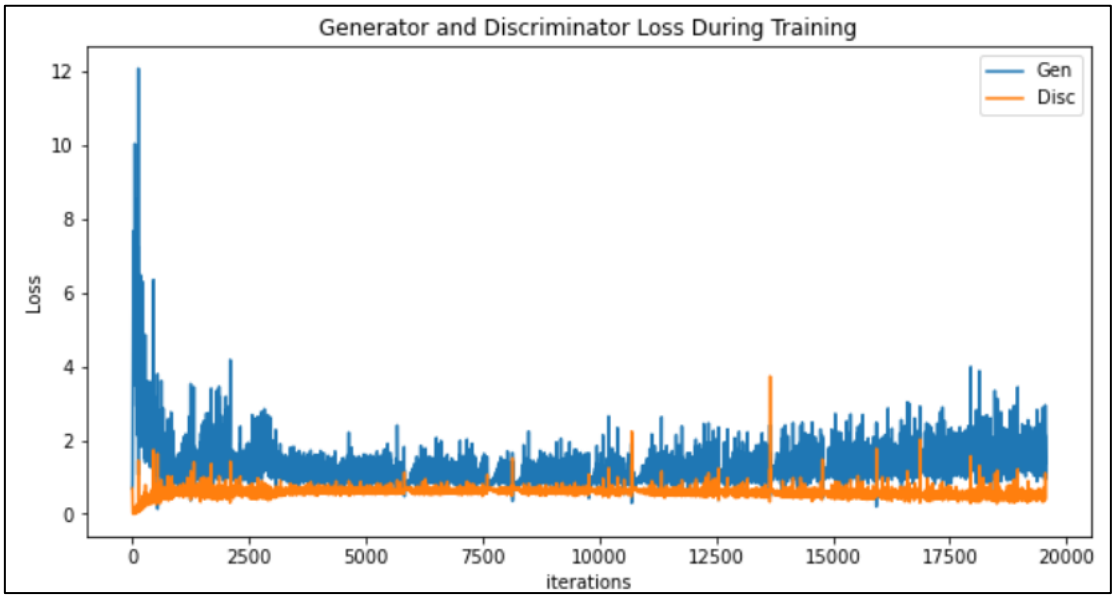
2.2 DCGAN:

- Deep convolutional generative adversarial networks (DCGANs) are types of CNN with specific architectural limitations and good options for unsupervised learning. DCGAN is one of the most well-known and successful GAN network developers; it's primarily composed of convolution layers that aren't entirely pooled or linked. It uses a coevolutionary stride and transposed convolutions for down and upsampling. The adaptability of DCGAN allows it to thrive.
- We've reached a point when increasing the generator complexity does not automatically improve picture quality.
- The images are up-sampled and down-sampled using transposed convolution and convolutional stride. Because DCGANs are challenging to train, there are architecture criteria for the network's training to avoid model collapse:
    - Replace max pooling with convolutional stride.
    - Using transposed convolution for upsampling.
    - Removing any layers that are entirely connected.
- Batch normalization is used for this assignment. I have used LeakyReLu for the generator and the tanh function for the discriminator's output. The following DCGAN structure is used implemented:



- As stated above, I have used the entire data set of CIFAR 10 for the training of GANS, and the following is the 128 sample of the CIFAR10:

- The network was trained for 50 epochs with the following, hyperparameter configuration:
    - LEARNING_RATE = 2e-4
    - BATCH_SIZE = 128
    - IMAGE_SIZE = 64, to resize the image.
    - CHANNELS_IMG = 3, since CIFAR 10 data set has colour images we pass the channel as 3 for RGB.
    - NOISE_DIM = 100
    - NUM_EPOCHS = 50
    - FEATURES_DISC = 64, Size of feature maps in discriminator
    - FEATURES_GEN = 64, Size of feature maps in generator
    - Optimizer = Adam
    - Momentum = 0.9, hyperparameter for Adam optimizer
    - beta = 0.5, hyperparameter for Adam optimizer
    - criterion = nn.BCELoss()
- After 50 epochs, the following was the generator and discriminator loss for 20000 iterations are as illustrated below:



- Here, we can see that near about 2500 iterations, the generator and discriminator loss settle and maintain a steady ratio, similar to the ideal condition.
- While trying it with various epochs and learning rates, in general, the generator loss started to increase after the 60th epoch, which lead to a drop in the quality of the generated images.

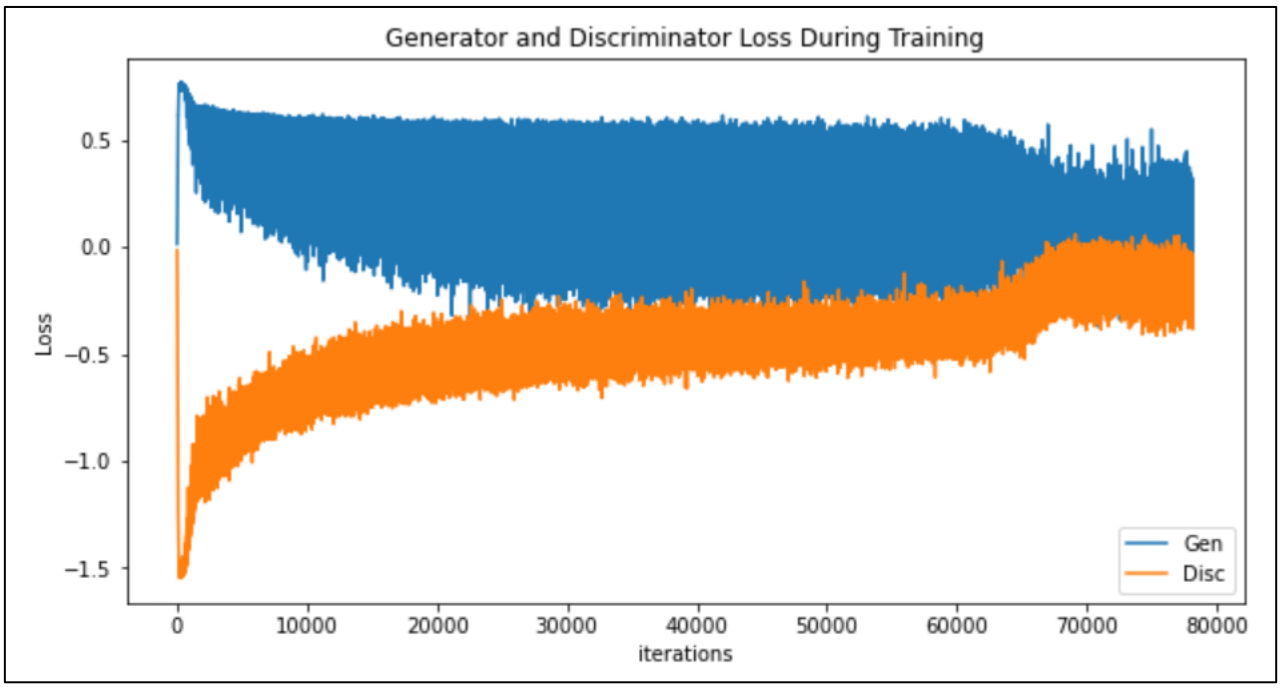
- Following are the images generated at epoch no 37, which had fetched the lowest FID score:

- As we can see, the images generated above are very pixelated and have lower definitions, and some look cluttered. This is due to the data set that the model was trained on.

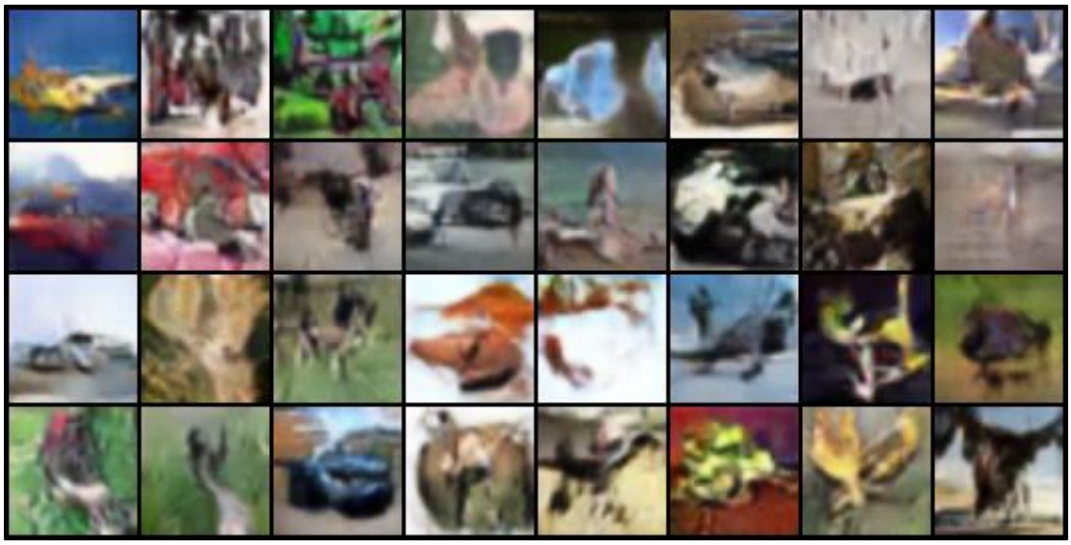
2.3 WGAN:

- The Wasserstein GAN grows into the generative network, enhancing model training reliability and including a loss feature relevant to image quality. Although only a few small alterations to the standard, deep coevolutionary, generative opponent network or DCGAN are required in reality, WGAN has a solid mathematical motivation.
- It uses the linear activation feature in the essential model's output sheet instead of the sigmoid function. Instead of 1 and 0, it utilizes -1 for real photographs and 1 for fake pictures. Wasserstein depletion is used to train the vital and generator models. And the small-batch update should be limited to a small range of key product weight (e.g. [-0.01,0.01]).
- Furthermore, for each epoch, the discriminator is trained 5 times, whereas trained the generator only once
- Following are the hyperparameters used to tune this network:
    - LEARNING_RATE = 5e-5
    - BATCH_SIZE = 32
    - IMAGE_SIZE = 64
    - CHANNELS_IMG = 3
    - NOISE_DIM = 100
    - NUM_EPOCHS = 50
    - FEATURES_DISC = 64
    - FEATURES_GEN = 64
    - weight_clip = 0.01
    - critical_iteration = 5

- Following is the Generator and Discriminator loss generated for 50 epochs:



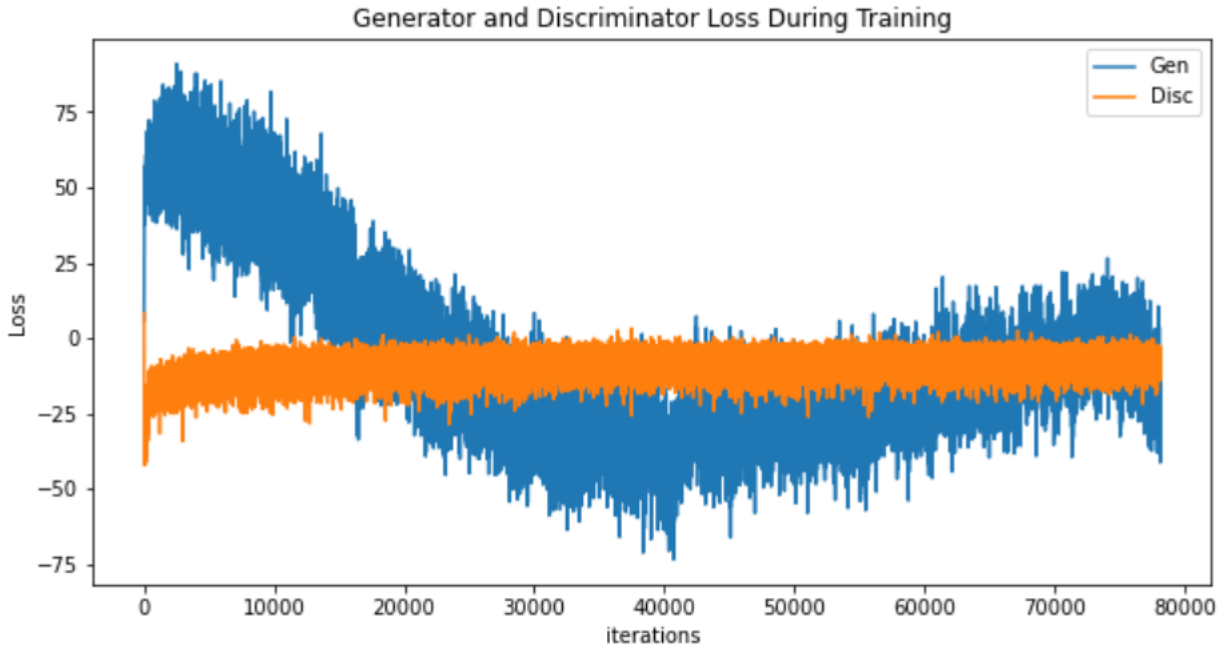
- Following are the images generated by the WGAN model:



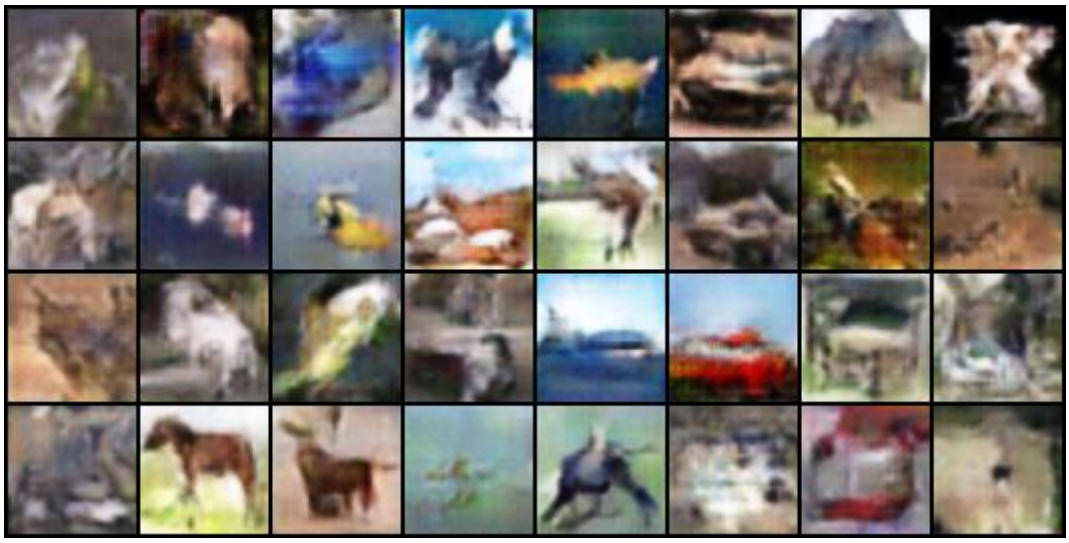
- As illustrated, the images generated by the WGAN are more pixelated when compared to the above DCGAN; this may be because the WGAN has not yet trained enough. WGAN, as it trains discriminator 5 times per epoch, takes a good deal of time despite training it on a system with GPU.

2.4 WGAN with GP

- The Wasserstein GAN + Gradient Penalty, or WGAN-GP, is a generative adversarial network that achieves Lipschitz continuity by combining the Wasserstein loss formulation with a gradient norm penalty.
- The original WGAN uses weight clipping to achieve 1-Lipschitz functions. Still, unwanted behavior such as pathological value surfaces, underused capacity, and gradient explosion/vanishing can result in undesirable behavior without careful calibration of the weight clipping parameter.
- A Gradient Penalty is a softer form of the Lipschitz requirement, according to which functions are 1-Lipschitz if the gradients are all of the same norm. As a gradient penalty, the squared difference from norm 1 is employed.The following are the hyperparameter used for the network:
    - LEARNING_RATE = 2e-4
    - BATCH_SIZE = 32
    - IMAGE_SIZE = 64
    - CHANNELS_IMG = 3
    - NOISE_DIM = 100
    - NUM_EPOCHS = 50
    - FEATURES_DISC = 64
    - FEATURES_GEN = 64
    - criticItr = 5
    - lambda_GP = 10
- Following is the Generator and Discriminator loss generated for 50 epochs:



- Following are the images generated by the WGAN with the Gradient Penalty model:

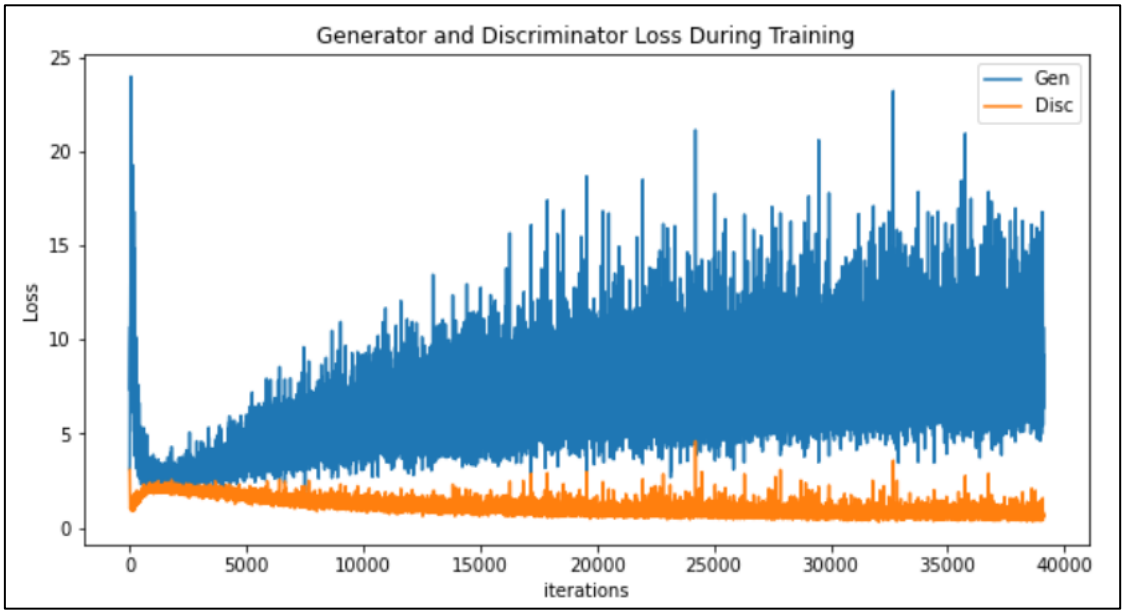


- Here, the model generated a bit better than normal WGAN, where few of them are recognizable, but their performance is still not as good as DCGAN.

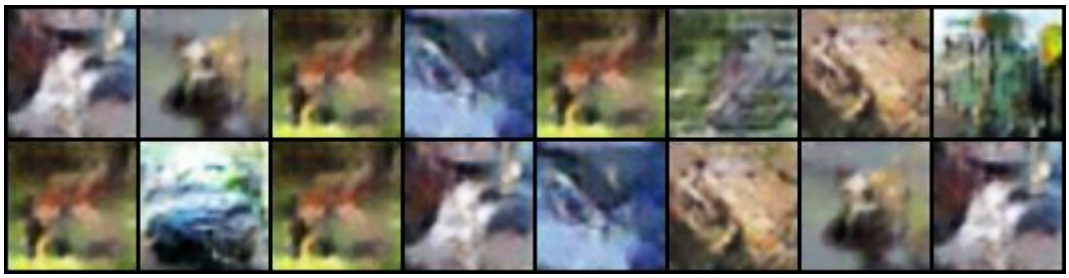
2.5 ACGAN

- The Auxiliary Classifier GAN, or AC-GAN, is a GAN architecture expansion based on the CGAN extension. In the 2016 paper "Conditional Image Synthesis using Auxiliary Classifier GANs," Augustus Odena et al. from Google Brain introduced the technique.
- Like the conditional GAN, the AC-generator GAN's model is given a point in the latent space and the class label as input, so the picture creation process is conditional. The discriminator model takes the image as input, whereas the conditional GAN takes the image and the class label. The discriminator model must next determine whether the image is genuine or not.
- The key difference is in the discriminator model, which accepts the image as input instead of the conditional GAN, which takes both the image and the class label. The discriminator model must then determine whether the given image is real or phony and the image's class label.
- The following are the hyperparameter used for the network:
    - LEARNING_RATE = 1.5e-4
    - BATCH_SIZE = 64
    - IMAGE_SIZE = 64
    - CHANNELS_IMG = 3
    - NOISE_DIM = 100
    - NUM_EPOCHS = 50
    - FEATURES_DISC = 64
    - FEATURES_GEN = 64
    - NUM_CLASSES = 10
    - EMBED_SIZE = 100

- Following is the Generator and Discriminator loss generated for 50 epochs:



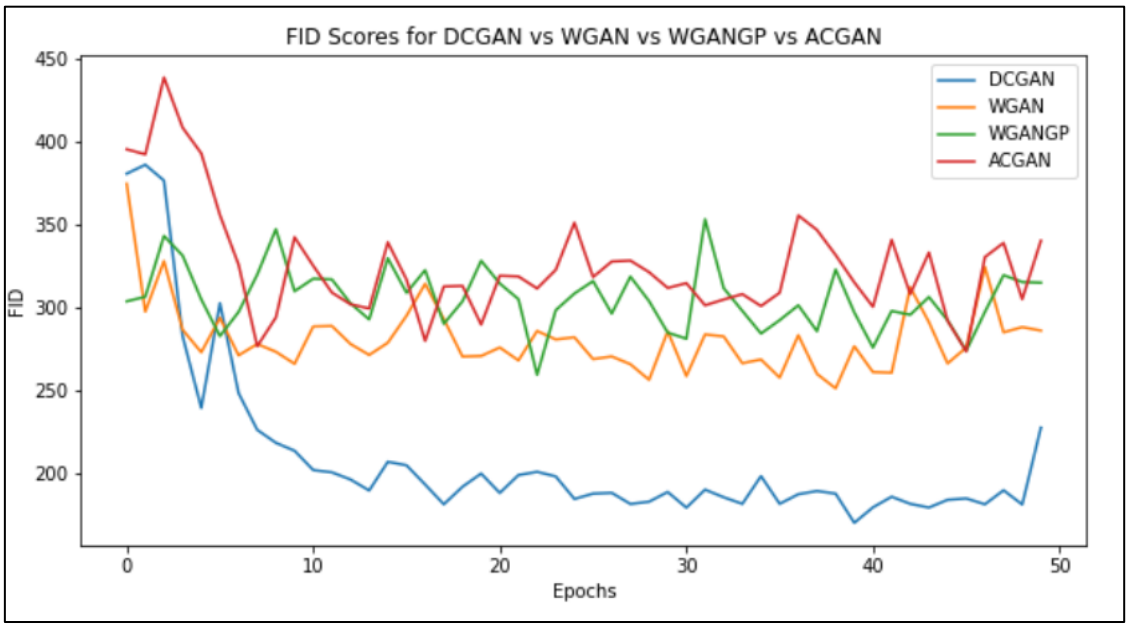
- Following are the images generated by the ACGAN model:



3. Evaluation

- The Frechet Inception Distance score (FID), which calculates the distance between feature vectors derived for real and produced images, is used to evaluate the performance of the above implemented GANs and their output.
- The score reflects how comparable the two groups' statistics on computer vision aspects of raw images calculated with the inception v3 image classification model. Lower scores imply that the two groups of photos are more comparable or have similar statistics, whereas a perfect score of 0.0 indicates identical.

- Following is the graph of the FID score for DCGAN, WGAN, WGANGP and ACGAN generated for 50 epochs:



Model	FID MIN	FID MAX	FID @ 50 epoch
DCGAN	170.420	209.563	227.700
WGAN	251.371	281.515	286.187
WGAN-GP	259.590	305.695	315.004
ACGAN	273.598	325.848	340.265

- As observed in the above statistics, DCGAN has the best performance of all the models.

4. Reference

1 https://arxiv.org/abs/1511.06434

2 https://machinelearningmastery.com/how-to-code-a-wasserstein-generative-adversarial-network-wgan-from-scratch/#:~:text=The%20Wasserstein%20Generative%20Adversarial%20Network%2C%20or%20Wasserstein%20GAN%2C,that%20correlates%20with%20the%20quality%20of%20generated%20images.

3 https://machinelearningmastery.com/how-to-implement-the-frechet-inception-distance-fid-from-scratch/