CPSC 8430 Deep Learning
Homework 4

Shwetha Sivakumar (C20140199)

Introduction

With this assignment, we will be training a discriminator-generator pair using networks such as DCGAN, WGANs and ACGAN from scratch on CIFAR10 dataset. The common goal of these networks is to train a generator to produce realistic images from CIFAR-10 dataset while training a discriminator to accurately distinguish between real and generated images and draw meaningful insights from the data. Used libraries for this task are Numpy, Torch and Opencv.

Dataset

In this assignment, CIFAR-10 is used as a source for training and evaluating discriminator-generator pairs with the DCGAN, WGANs, and ACGAN architectures. It includes 60,000 color images in 10 classes, hence a very diverse dataset for image generation tasks. The generator networks are trained to generate realistic images similar to those in CIFAR-10 and minimize the difference between generated and real images perceived by the critic, the discriminator.

While at the same time the critic/discriminator is trained to classify between real CIFAR-10 images and the ones generated by the generator, trying to maximize its classification performance. In this training process, both generator and discriminator performances will be monitored for different validation or test sets from CIFAR-10 using accuracy and loss functions metrics. In this way, one is able to analyze several GAN architectures and their efficiency in the generation of high-quality images using the CIFAR-10 dataset.

GAN Models explanation:

In the deep learning field, the process of producing fake images, known as fake image generation, utilizes generative models, particularly Generative Adversarial Networks to produce images that approximately similar to real ones. GANs consist of a generator and discriminator pair where generator responsible for producing fake images and a discriminator is trained to differentiate real image from fake images. Through iterative training, both models refine their abilities until the generated images are very similar to real ones. After training, the generator is able to independently create new synthetic images with its own attributes. Applications of fake image generation using GANs are done in various domains for image synthesis, data augmentation, virtual reality, video game design, and artistic works.

Explanation of the generator model:

In GANs, a generator model is one of the two major components of the architecture, consisting of a deep neural network, which is especially developed to generate synthetic images just like authentic ones: based on the principle of turning random inputs, usually in forms of noise or latent vectors, the generator generates coherent visual output. The generator thus performs a very important task inside the adversarial training framework, training iteratively to predict the input patterns and mapping those patterns onto image space.The goal of this training is to obtain images which could not be distinguished by the human eye as being generated.

Thus, Generator plays crucial role in GAN, which has an ability in synthetic content generation that includes pictures. In this GAN framework, this generator and paired Discriminator works

adversarially, uniquely. In this discrimination between real and fake Images-what one tries to generate, then the other is generating such fake images looking more similar to originals. This adversarial training mechanism makes the generator to improve its generation of images iteratively, until it reaches a climax in the generation of synthesized images that are almost realistic as real-world images. The goal of the generator model is hence to generate high-fidelity fake images through iterative learning and generation processes.

Explanation of the discriminator model:
The discriminator model in GAN is a neural network that aims at distinguishing between the authentic and synthetic images produced by the generator model. The task discriminator does classification of input images as either real or fake. The discriminator thus represents an important element in this adversarial training process since the network receives input for an image and classifies that accordingly. This provides one critical source of feedback as part of the generated images evaluation to continuously refine its generation capability iteratively.

Thus, the Discriminator models constitute another major component of GANs, providing the adversarial interplay between generator and discriminator. While the discriminator is to classify the images into real or fake, the generator works side by side to create synthetic images that can actually deceive the discriminator. This is an adversarial dynamic that keeps both models improving one after another, with the discriminator feedback helping the generator in the improvement of generated images and that's how the discriminator model plays the key role of iterative refinements in the generation capabilities of the generator models within the GAN framework.

DCGAN Model Description:
The DCGAN, fully expressed as Deep Convolutional Generative Adversarial Network, is an enhanced version for use in GANs and thus intended to synthesize images. While it keeps the traditional GANs composition-a generator and discriminator-the model optimizes them for the generation of images. The generator takes a random noise vector in the framework of DCGAN and converts this into a realistic images via transposed convolutional layers, also known as deconvolutional layers. These layers provide the up-sampling of the input noise to generate images to enhance complexity and detail. The generator generates images capable of deceiving the discriminator, thereby resembling real images from the dataset.

On the other hand, the discriminator in DCGAN is a CNN-based classifier trained to identify the real and fake images. Employing convolutional layers with stride for down-sampling, batch normalization, and LeakyReLU activation functions, the discriminator scores images and returns a probability score representative of their realness. It is trained on both real and synthetic images, and it strives to correctly classify which ones are real and which ones are fake. During training, the generator and discriminator are put under certain model configurations and hyperparameters for optimal performance. More so, BCELoss function is used as the loss, and weight initialization is set to normal distribution where standard deviation is 0.02. DCGAN is a very strong structure for image generation; it offers a strong architecture and a training methodology apt for generating realistic synthetic images and has proved to be very effective for a wide variety of applications.

WGAN Model Description:
One improved version of a traditional Generative Adversarial Network is the Wasserstein Generative Adversarial Network, or in short WGAN. GAN architecture where a loss function is applied based on the Wasserstein distance. This metric change training dynamics, which circumvents most of the challenges facing traditional methods, such as mode collapse and instability. In WGAN also, we have a generator of synthetic data and discriminator of differentiation between real-synthetic data. Compared to other vanilla GANs, in WGAN they seek stable training with gradient penalization and weight clippings. This encourages fast convergence and more reliable training results. The maximum difference which the discriminator will maximize in WGAN is between the distributions of real and synthetic data, rather than classifying them. This approach leads to more robust training and improvement in sample quality, and that's why WGANs are used in various generative modeling tasks.

WGAN-GP Model Description:
The Wasserstein Generative Adversarial Network - Gradient Penalty (WGAN-GP) leverages Wasserstein loss together with gradient norm penalty for getting Lipschitz continuity of the discriminator. In the WGAN-GP, an extra gradient penalty term is added in the loss function to impose the Lipschitz continuity. This penalty rewards smoother gradients, hence enhancing both training stability and sample quality. The architectures of generator and discriminator remain similar to WGAN; however, an additional gradient penalty term is used in its training process. The WGAN-GP is usually trained in an iterative updating of the generator and discriminator parameters with the minimization of the Wasserstein distance while penalizing gradients above some threshold. This approach gives better convergence and generation quality compared to the traditional WGANs.

ACGAN Model Description:
The Auxiliary Classifier Generative Adversarial Network is an improved version of the traditional GAN structure by incorporating class information into both the generator and discriminator. In addition to the discriminator's role in distinguishing between real and fake images, the auxiliary classifier predicts class labels for both real and synthetic images. This dual role helps the generator create class-consistent images. The training of ACGAN consists of iteratively optimizing both adversarial and auxiliary classification losses. This approach allows for the generation of diverse-looking images that align with specific class labels; this allows for ACGAN to perform applications such as controlled image synthesis and image style transfer. Discriminator and generator architecture, like DCGAN architecture with changes to include adding class labels alongside the input noise.

Code description:
Calculating FID Score:
The FID score is a sort of standard metric evaluator for generated images in GAN models. It tells us how close our generated images are to the real ones by comparing their features. A lower FID score means our generated images are more realistic and diverse, which is what we're aiming for in our work.

First of all, we initialize a pre-trained Inception-v3 model and extract features from Mixed_7c layer, which becomes the representation of image features. Second, we use this Inception-v3 model

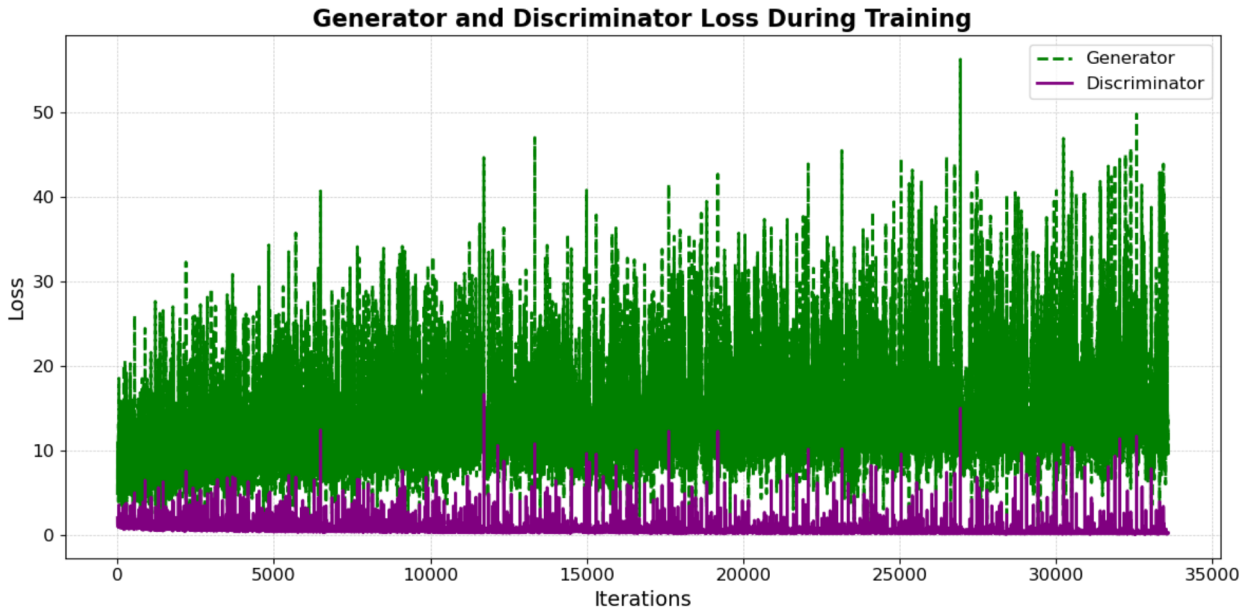
in extracting features from a batch of images and provides ways of getting feature representations for both real and generated images. Finally, the mean and covariance of the extracted features are calculated, which are needed in calculating the FID score.
FID score is compared with the statistics (mean and covariance) of feature representations between real and generated images, quantifying their similarity using the Fréchet distance formula. Additionally, utility functions for image preprocessing, loading images from directories, and multi-processing are used for efficient computation.
Code Link:
https://github.com/shwethasivakumar/Deep-Learning/blob/main/HW4/FID.py

Generating images using ACGAN:
ACGAN is a variant of the traditional GAN framework that adds an auxiliary classifier alongside the generator and discriminator, enabling class-conditional image generation. First, we had set up hyperparameters like learning rate as 0.0002, batch size as 64, image size as 64, epoch size 10, and model architecture specifications including the number of classes as 10, number of channels as 3, embedding size as 100 and dimensionality of the input noise as 100. It also defines data transformations for image preprocessing and loads the CIFAR-10 dataset for training.
Then, the generator and discriminator architectures are defined using PyTorch class nn.Module. The generator takes random noise coupled with the class labels as input and generates fake images, while the discriminator predicts the authenticity of the images along with their class labels. The models are initialized with appropriate weight initializations for training. The training loop alternates between training the discriminator and the generator. For each batch of real images, the discriminator is trained to classify them correctly, while also being trained to distinguish real and fake images produced by the generator. In parallel, the generator is trained to create realistic images with class labels.

During training, the losses of generator and discriminator will be tracked, and images created by the generator at an interval will be saved to track visualization and FID score for checking the similarity of the distribution of real and generated images. Finally, an animation of the training is displayed showing the images generated after each epoch.
Sample of generated Images by ACGAN:

Below are the 10 best fake images generated by ACGAN:

Below figure shows the Generator and Discriminator loss during training of ACGAN model:

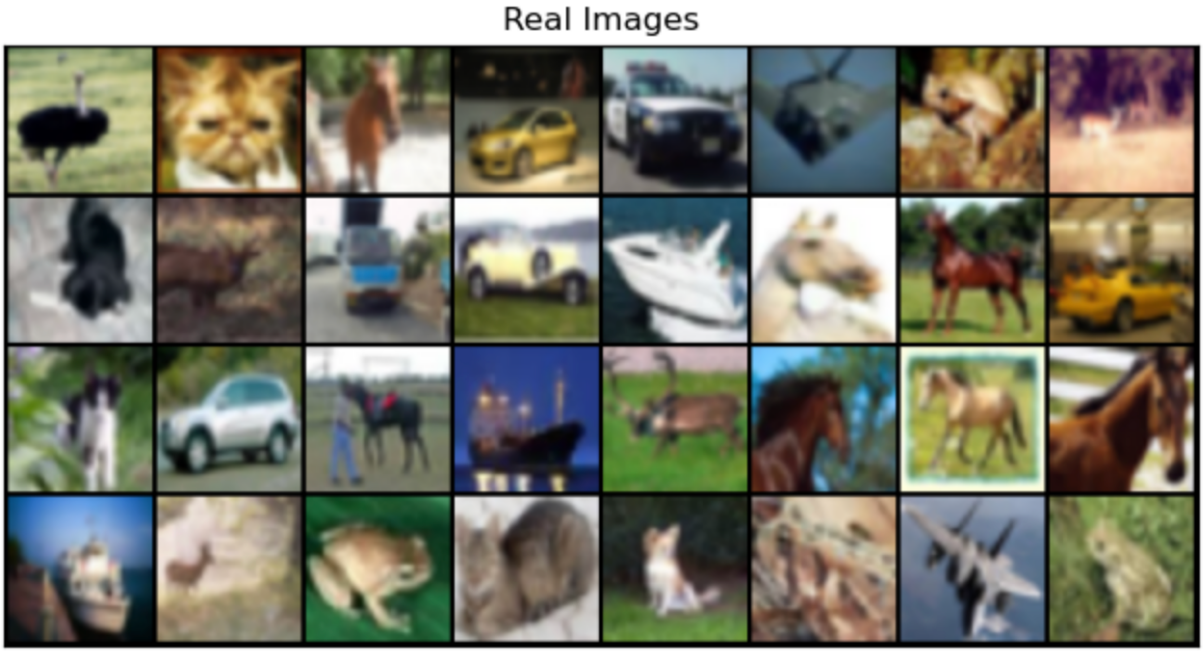


Code Link:
https://github.com/shwethasivakumar/Deep-Learning/blob/main/HW4/ACGAN_n.ipynb

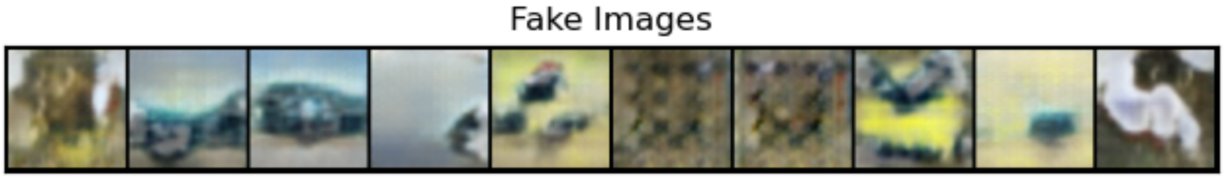
Generating images using DCGAN:
Here, we used Deep Convolutional Generative Adversarial Network (DCGAN) for image generation using CIFAR-10 dataset. We define hyperparameters- learning rate as 0.0002, Batch size as 128, Image size as 64, Image channels as 3 for RGB images and z_dim as 100 for the dimension of the latent noise vector. The DCGAN also comprises a generator and discriminator. The generator is defined with transpose convolutional layers, starting with a random noise vector of dimension z_dim and upsampling it to generate images of size 64 with 3 channels. The discriminator, on the other hand, comprises convolutional layers with LeakyReLU activation and batch normalization. Weight initialization is performed using normal distribution for convolutional and batch normalization layers to ensure stable training.

The CIFAR-10 dataset is loaded and preprocessed using torchvision transforms, and the generator and discriminator classes are defined accordingly. The training loop toggles between updating the parameters of generator and discriminator. The discriminator is trained to differentiate real and fake images while the generator aims to generate images. The binary cross-entropy loss (BCELoss) is used to calculate the loss for both models. Adam optimizer is employed for optimizing the model parameters with a learning rate as 0.0002 and betas set as (0.5, 0.999).
During training, generated images are visualized and saved for monitoring the progress.

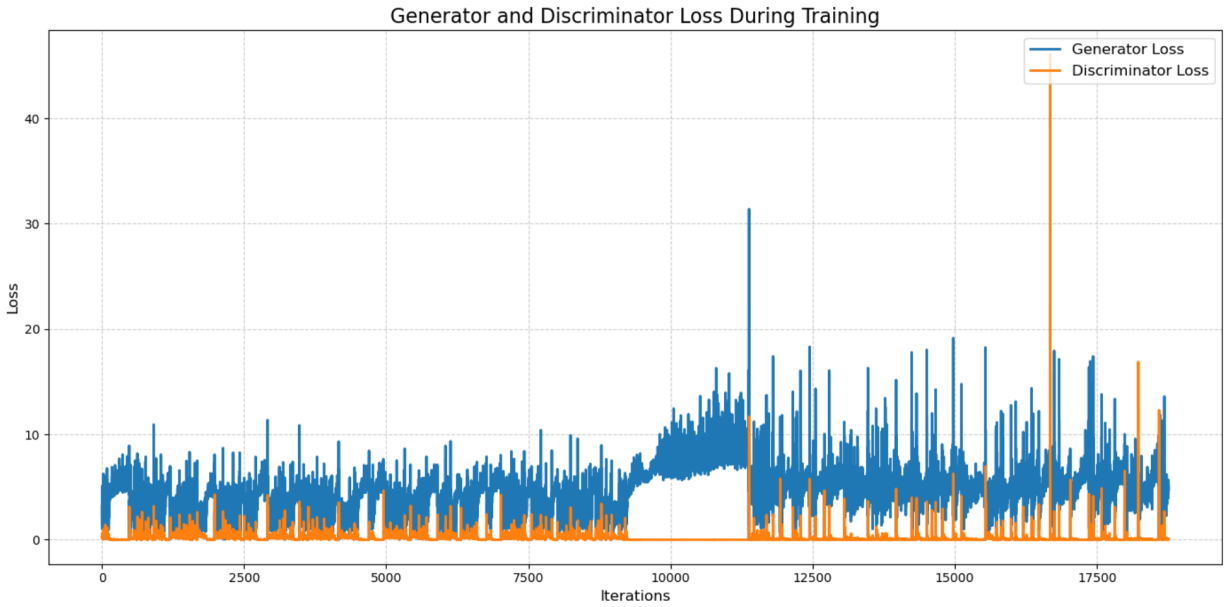
Besides, the losses incurred during training and calculated FID score at each epoch are saved for plotting. The generated images are visualized using an animation to observe the evolution of image generation over epochs, providing insights of the resulted images.
Below are the sample images generated by DCGAN:



Below are the 10 best fake images generated by DCGAN:



Below figure shows the Generator and Discriminator loss during training of DCGAN model:

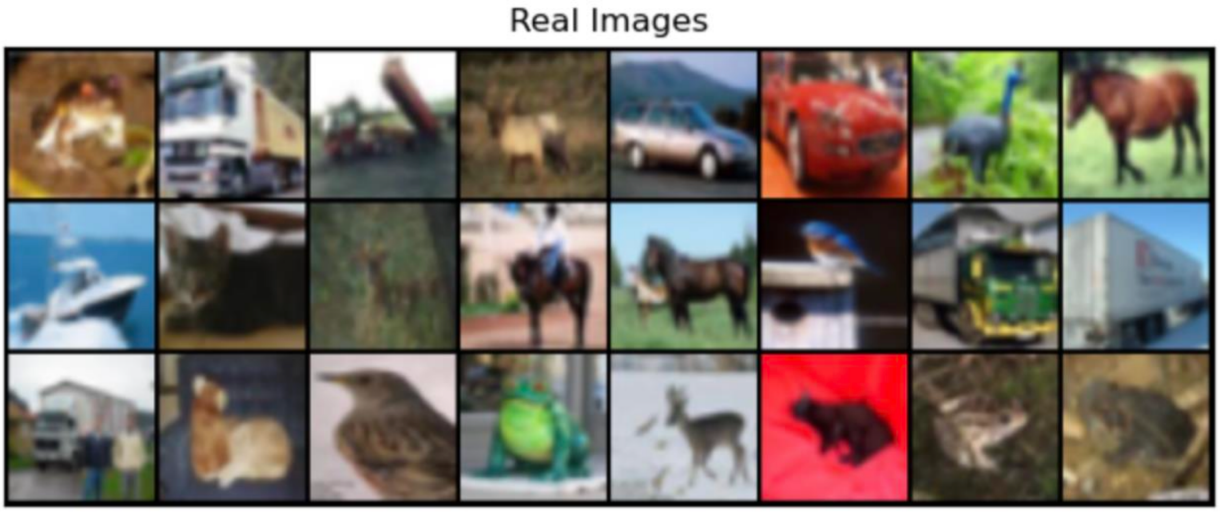


Code Link:
https://github.com/shwethasivakumar/Deep-Learning/blob/main/HW4/DCGAN_n.ipynb

Generating images with WGAN:
Here, we implemented the Wasserstein Generative Adversarial Network (WGAN) to generate images on the CIFAR-10 dataset. First of all, we define the hyperparameters like learning rate as 5e-5, Batch size as 64, Image size as 64, Image channels as 3 for RGB images, z_dim as 100 for the dimension of the latent noise vector, epoch as 50, discriminator features as 64, generator features as 64, critical iterations per generator iteration as 5, and clipping parameter for weights in the critic model as 0.015. The dataset is loaded and preprocessed using torchvision transforms, and the generator and discriminator classes are defined accordingly.

The training loop is alternated between updating critic (discriminator) and generator parameters. The critic is trained to maximize the difference between the mean scores of real and This means the fake images while being clipped inside a specified range by taking the advantage of the Lipschitz constraint. Minimize the mean score given by the critic for the fake images by training the generator. Optimizations are carried out in both models by RMSProp using the specified learning rate. The progress can be monitored by visualizing the generated images and saving them while training. The Losses on training, which includes generator loss, the critic's loss along with FID score is computed for every epoch. Above-generated images will be shown to provide visual information on how image generations change over the number of epochs, this informs much about the resulting generated images.

Following are a few generated sample images of the WGAN:



Here is shown 20 Best FAKE Images produced by the WGAN

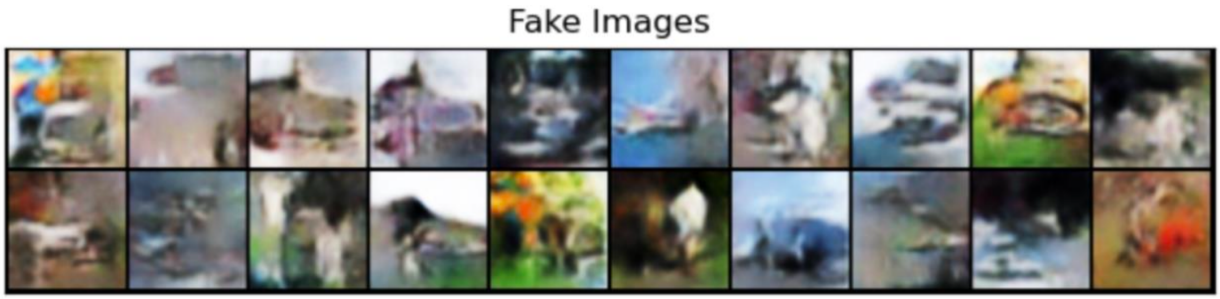
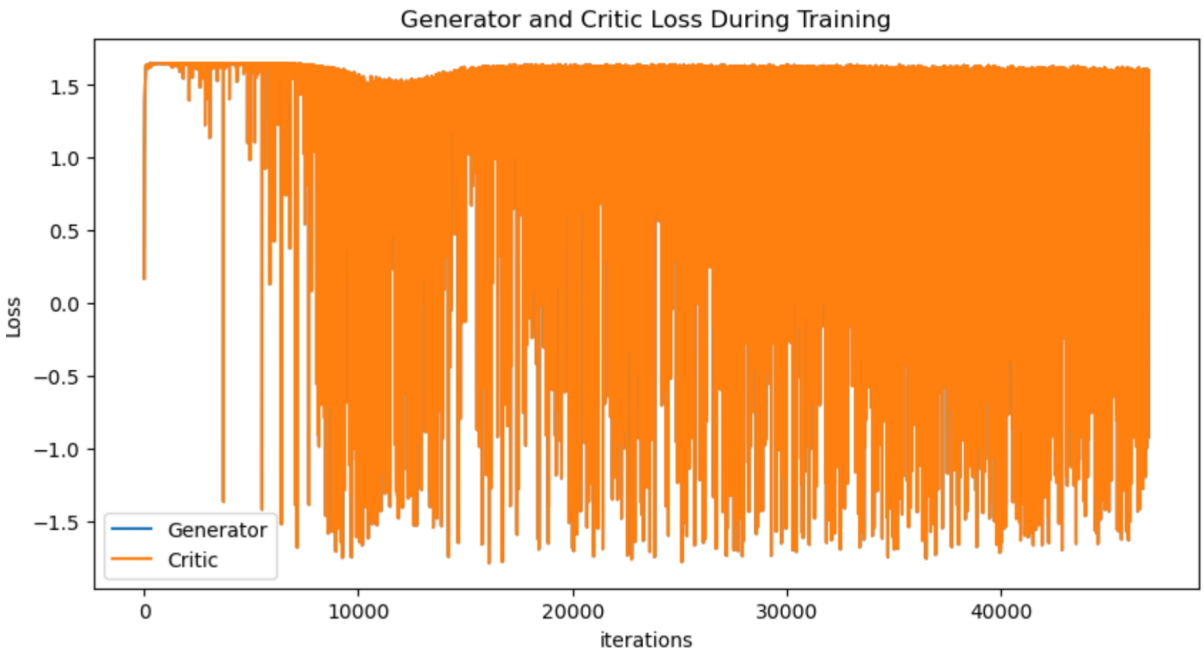


Figure shows the Generator and Discriminator loss during training of WGAN model:



Code Link:
https://github.com/shwethasivakumar/Deep-Learning/blob/main/HW4/WGAN.ipynb

Image generation with WGAN-GP:
We will also implement the WGAN-GP to generate the images on CIFAR-10 dataset. Firstly, we defined some hyperparameters including learning rate as 1e-4, Batch size as 64,Image size as 64x64 pixels, Image channels as 3 for RGB images, z_dim as 100 for the dimension of the latent noise vector, Epochs as 50, discriminator features as 64, generator features as 64, number of critic iterations per generator iteration as 7, and gradient penalty coefficient as 10. Dataset is loaded and preprocessed by using torchvision transforms.
Generator and discriminator architectures are defined in convolutional layers, and weight initialization is applied on both models. Moreover, a function to compute the gradient penalty is defined. The models are then trained in an alternatively, with the critic trained to maximize the Wasserstein distance between real and fake images while taking advantage of the Lipschitz constraint using the gradient penalty, and the generator trained to minimize the critic's score on generated images. Both models used Adam optimizer where learning rate is 0.0001 and momentum parameters.
During training, generated images are visualized and saved for monitoring the training progress. The losses for the training including generator and critic loss, are being saved for plotting. FID scores are calculated at each epoch as a way to measure the quality of the generated images in respect to real images. The FID scores are saved for analysis as well. At the end, an animation is used to show the generated images over epochs to see the resulted generated images.
Following are sample images generated by WGAN-GP:

Below are the best fake images generated by WGAN-GP:

Below figure shows the Generator and Discriminator loss during training of WGAN-GP model:

Code Link:
https://github.com/shwethasivakumar/Deep-Learning/blob/main/HW4/WGAN_GP.ipynb

Comparison of Performance of ACGAN, DCGAN, WGAN, and WGAN-GP:

It would be apt to compare ACGAN, DCGAN, WGAN, and WGAN-GP considering the major factors such as image quality, training stability, and mode collapse.

First of all, for the case of image quality, ACGANs produce highly superior, sharp featured images due to conditioning on the class label and noise vector. The ability of DCGAN in generating visually pleasing images with sharp features and high fidelity. WGANs tend to generate high-quality images similar to DCGANs because the Wasserstein distance metric reduces mode collapse. However, the incorporation of gradient penalty in WGAN-GP further improves the quality of images generated, leading to a more stable training process and better quality images.

As for training stability, ACGANs could be unstable, especially while finding a balance between the adversarial and auxiliary classification losses. On the other hand, DCGANs tend to be more stable during training but still suffer from issues such as mode collapse or vanishing gradients. WGANs are more stable compared with vanilla GANs; Wasserstein distance provides much smoother training. Most of the proposed models that included gradient penalty were rather stable: WGANGP provided quite consistent training processes.

Third, the mode collapse occurs variably depending on the model. Mode collapse may appear for from mode collapse, where the generator produces few varieties of images. WGANs address this issue better than traditional GANs, but mode collapse can still occur, especially in complex datasets. On the other hand, WGAN-GP effectively reduces mode collapse compared to WGAN by penalizing the norm of the gradient, favoring diverse samples from the generator.

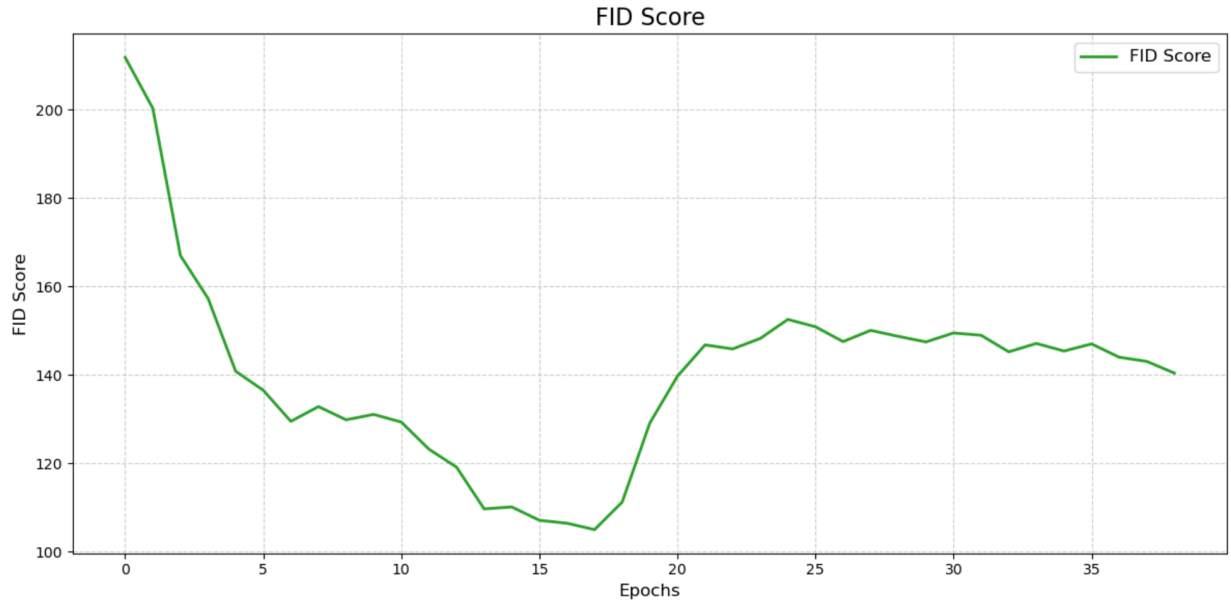
Finally, in terms of convergence speed, ACGANs might converge slower due to the extra auxiliary classification task, while DCGANs usually converge comparatively fast given proper hyperparameter tuning. WGANs might converge more slowly compared to DCGANs due to the computation of the Wasserstein distance involving several iterations of the discriminator. WGAN-GP might converge more slowly compared to WGAN and faster than traditional GANs due to the added computation of gradient penalty that adds computation but helps to stabilize the training process.

But for all models, WGAN-GP shows the highest qualitative results, stability during the process of training, mode collapses reduction, and convergence speed, outperforming ACGAN, DCGAN, and vanilla WGAN on these points. Finally, we have to choose models depending on the particular demand, such as dataset and resource of processing, or important requirements ranging from stability to quality image and vice versa. FID score computation of ACGAN, DCGAN, WGAN, and WGAN-GP models:

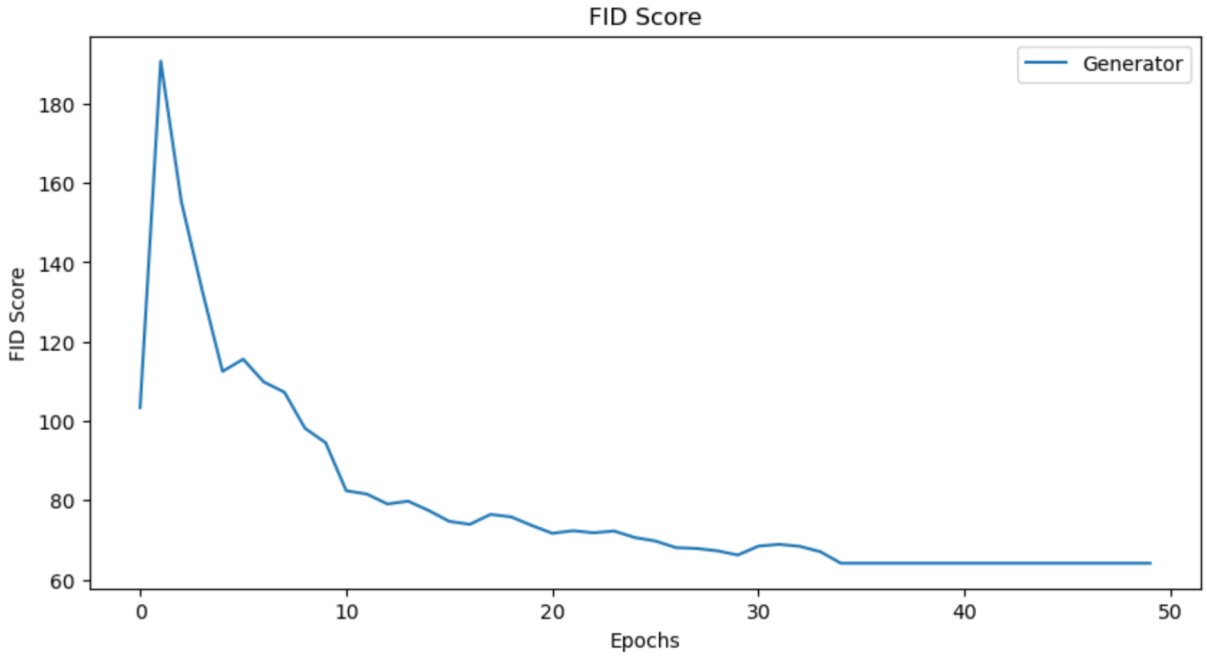
To assess the performance of these models further, the FID score of each model is computed. FID essentially captures the similarity between feature representations of synthetic and authentic images using the Inceptionv3 model. Higher the FID score indicates lesser similarity between images, and vice-versa. The code for calculating the FID score was adapted from and included in the routine for evaluation.

The FID Score of DCGAN model is 140.383

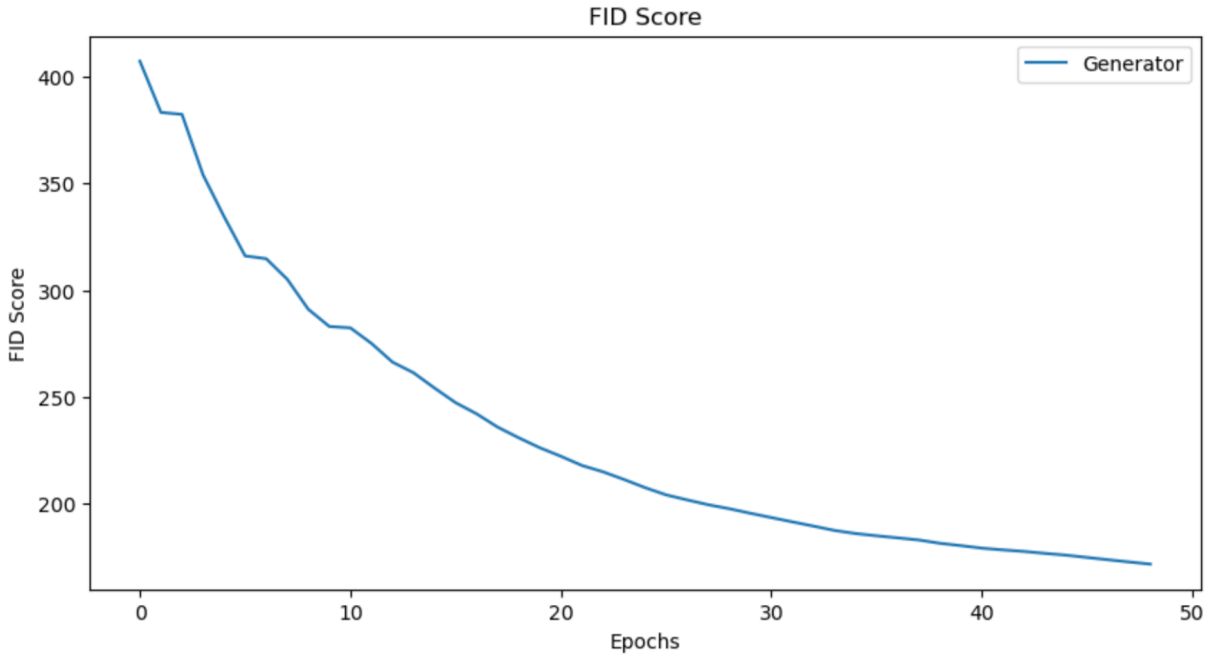
Here is the plot for DCGAN model FID Score :



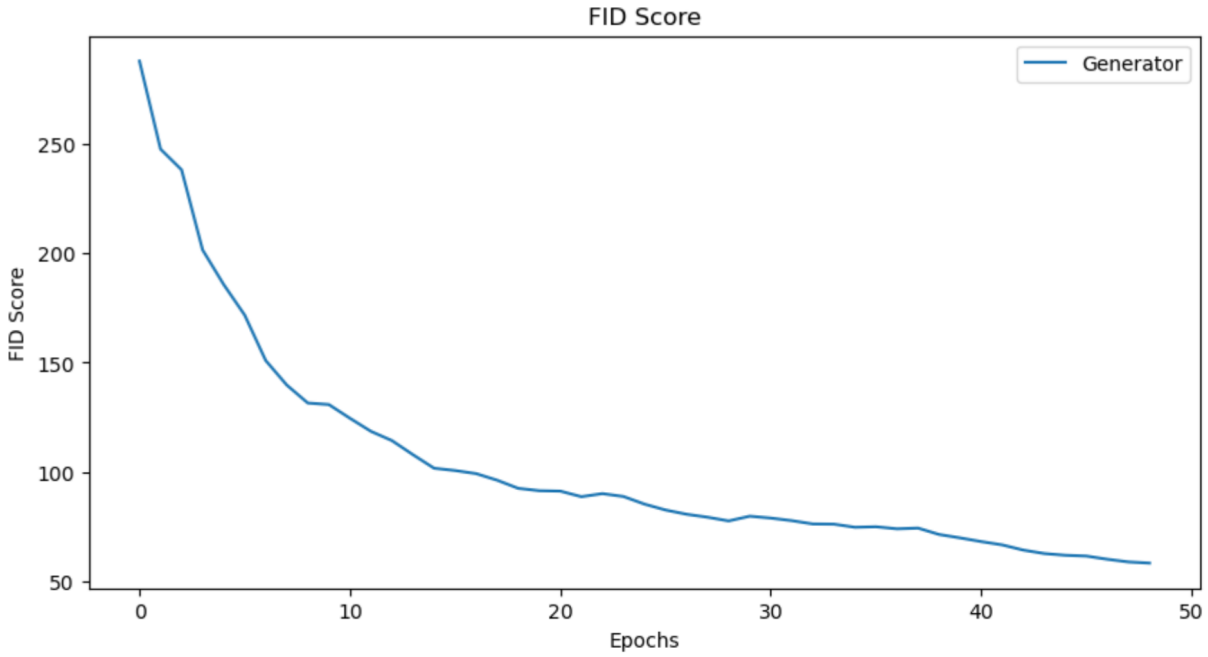
The score for FID Score ACGAN model is 64.0929
Below is a plot regarding the ACGAN model for FID Score:



Now, the FID score for the WGAN model is 171.369
Here comes a plot related to WGAN model FID Score



The FID score for the WGAN-GP model is 58.539
Following is the plot for the WGAN-GP model regarding its FID Score:



Conclusion: We can conclude here that WGAN-GP performed well compared to other GAN the FID Score is low in those models, 58.539, and we need to choose the GAN Models depending on our problem statement and requirements of image generation.