CPSC 8430 DEEP LEARNING HOMEWORK 4

SATHWIKA ADIMULAM
Dr. Feng Luo

GitHub link:- https://github.com/sathwika-adimulam/Deep_Learning_HW4

Problem Statement

The objective of this homework assignment is to train a discriminator and generator on the CIFAR10 dataset using techniques from DCGAN (Deep Convolutional Generative Adversarial Networks), Wasserstein GANs (WGANs), and ACGAN (Auxiliary Classifier GANs).

Dataset

For image classification applications, the CIFAR-10 dataset is a popular one in the fields of machine learning and computer vision. It contains a total of 60,000 color images, each with a size of 32x32 pixels, and is divided into 10 different classes or categories. Each class contains 6,000 images. CIFAR-10 has emerged as a dataset for evaluating the performance of various machine learning models, particularly in the context of image generation tasks where the objective is to generate realistic images that closely resemble the images in the original dataset.

Overview

GAN, or Generative Adversarial Network, is a machine learning model that consists of two neural networks, a generator and a discriminator. While the discriminator is trained to distinguish between actual and fake data, the generator's main job is to produce fake data. The generator and discriminator are trained in an iterative process where they engage in a competitive interplay, resulting in improvements in their respective capabilities over time. In a GAN, the generator's main goal is to reduce the generator loss, which measures the difference between generated and actual data. On the other hand, the discriminator aims to minimize the discriminator loss, which measures the accuracy of its classification of real and fake data. To address the challenges associated with GAN training, various GAN architectures, such as DCGAN (Deep Convolutional GAN), WGAN (Wasserstein GAN), ACGAN (Auxiliary Classifier GAN), and others, have been proposed with different design choices and improvements to enhance their performance and stability.

Requirements

1.Python
2.Torch
3.Numpy
4.CUDA
5.CIFAR10 dataset

Model Execution

DCGAN: Deep Convolutional Generative Adversarial Network

DCGAN is a GAN architecture designed for image generation tasks. DCGAN utilizes deep convolutional neural networks (CNNs) as its underlying architecture. The generator in DCGAN typically employs transposed convolutional layers, also known as deconvolutional layers, to upsample the input noise and generate images with higher spatial resolution. On the other hand, the discriminator in DCGAN consists of convolutional layers that capture local image features and learn spatial hierarchies to accurately classify between real and fake images.

DCGAN incorporates several important techniques in addition to its base architecture. This includes the use of batch normalization in both the generator and discriminator, which aids in stabilizing the training process and enhancing the quality of generated images. DCGAN uses the Tanh function in the output layer instead of the ReLU function in the generator. In the discriminator, DCGAN employs the Leaky ReLU activation function to introduce non-linearity and mitigate the issue of gradient vanishing. These techniques collectively contribute to the overall effectiveness and performance of DCGAN in generating realistic images.

The hyperparameters listed below were utilized for both the generator and discriminator networks:

- Batch Size used for training: 128
- Learning Rate: 0.0002
- Dimension of Latent Vector: 100
- Adam is the optimizer with beta1=0.55 and beta2=0.999
- Loss Function: Binary Cross Entropy Loss
- The initial weight distribution has a standard deviation of 0.02 and is normal.

Evaluation Results

The best 10 fake images generated by DCGAN is given below:

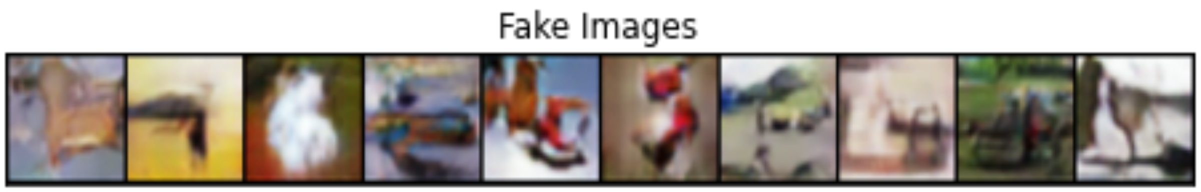

Figure 1: Images generated using DCGAN

WGAN: Wasserstein Generative Adversarial Network

WGAN is a variant of GAN architecture that introduces significant modifications to the traditional GAN framework. First, WGAN switches the critic model's output layer's sigmoid activation function for a linear activation function. This alteration facilitates better gradient flow during the training process, mitigating the issue of vanishing gradients and promoting more stable updates to the model parameters. Secondly, WGAN employs weight clipping technique on the parameters of the critic model, which constrains their values to a predefined range.

In addition to the modifications to the activation function and weight clipping, WGAN also employs a small learning rate during the optimization process. This choice of learning rate helps to prevent large parameter updates that could disrupt the training process and destabilize the models. Furthermore, WGAN utilizes the RMSProp optimizer for updating the model parameters, but without momentum. This optimizer choice helps to stabilize the training process by adaptively adjusting the learning rates for different model parameters based on their recent update history. It's important to note that when using larger weight clipping values, WGAN may require longer training times.

The following hyperparameters were used for both the generator and discriminator networks:

- Batch Size used for training: 64
- Learning Rate: 5e-5
- Dimension of Latent Vector: 100
- Optimizer: RMSProp with momentum set to 0
- Weight clipping set to 0.015
- Critic Iteration set to 5
- Wasserstein loss computed as the mean of predicted and true values.
- Initial weight distribution is normal, with a standard deviation of 0.02

Evaluation Results
The best 10 fake images generated by WGAN is given below:

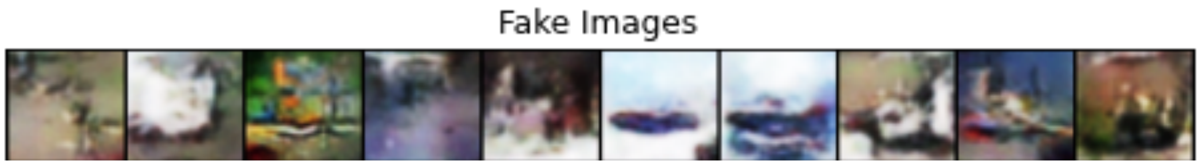

Figure 2: Images generated using WGAN

ACGAN: Auxiliary Classifier Generative Adversarial Network

ACGAN is a GAN variant that introduces auxiliary classifiers in both the generator and discriminator networks. In ACGAN, the generator takes both latent noise and a class label as input and generates a synthetic image that corresponds to the given class label. The discriminator, on the other hand, accepts a image as input and produces two outputs the likelihood that the image is real and the anticipated class label for the image. The class label serves as a guiding factor for the generator, allowing it to synthesize images that align with the input class label during the training process.

Prior to input, embedding is used to represent the latent vector for better model performance. During training, ACGAN computes two losses. The binary cross-entropy loss is first utilized to quantify the difference between the generated image's true label and expected probability. This loss ensures that the generated images align with the desired class label. Second, the log-likelihood of the predicted class label given the generated image is calculated, which measures the likelihood of the predicted class label being accurate. These losses are combined and optimized during the training process to enable the generator to generate realistic images that not only resemble the input class label but also exhibit high-quality visual characteristics. This dual loss approach allows ACGAN to produce images that are not only visually coherent with the desired class but also demonstrate high-quality visual attributes.

The hyperparameters employed for both the generator and discriminator networks, which include:

- Batch Size used for training: 64
- Learning Rate: 0.0002
- Dimension of Latent Vector: 100
- Adam is the optimizer with beta1=0.5 and beta2=0.999.
- Loss Function: Binary Cross Entropy Loss and NLL_Loss
- With a standard deviation of 0.02, the initial weight distribution is normal.

Evaluation Results
The best 10 fake images generated by ACGAN is given below:

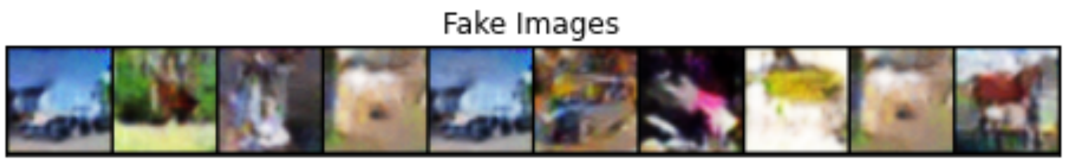

Figure 3: Images generated using ACGAN

Performance Evaluation

Upon the conclusion of the training process, the FID a widely used metric for measuring the accuracy of generated images, was used to evaluate the performance of the models. FID score is determined by comparing the statistical characteristics of the generated images with those of a reference set of real images. Subsequently, the FID scores were calculated for each model, and the findings are presented in a tabular format, showing the performance of each model based on their respective FID scores.

Models	FID Score
DCGAN	64.281
WGAN	127.524
ACGAN	72.933

The plot displayed below illustrates the FID scores for each model during the training process, with the FID score plotted against the number of epochs.

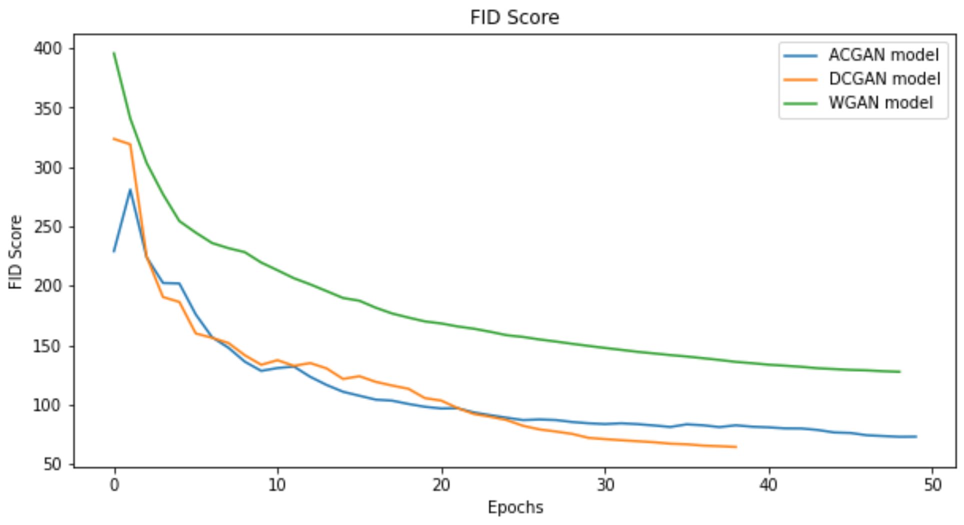

Figure 4: FID Score vs Epochs for each model during the training

Conclusion

Based on the results presented in the table, it can be observed that DCGAN achieved the lowest Frechet Inception Distance (FID) score of 64.281, indicating best performance compared to other models.