CPSC-8430
Deep Learning Homework-4
Avinash Komatineni

Github Link: https://github.com/avinash-komatineni/HW4
I uploaded DCGAN pickle file in
https://drive.google.com/file/d/1PiVm-wVVgIJk1cTUQa69MYx4iwvQVic0/view?usp=sharing

Abstract:
The main goal of the project is train a GAN type model on CIFAR-10 dataset by implementing DCGAN, WGAN, ACGAN models.

Introduction:
Generative Adversarial Networks are a class of deep learning neural networks that will generate realistic samples from a given dataset. GAN have a generator network that generate new images and a discriminator that classifies whether the image is real or generated. GANs have seen widespread success in image generation tasks, such as generating realistic images of faces, animals, and other objects.

Dataset:
The CIFAR-10 dataset is a collection of 60000 color images in ten classes. The dataset is widely used in computer vision research, and it provides a challenging benchmark for image classification and generation tasks.

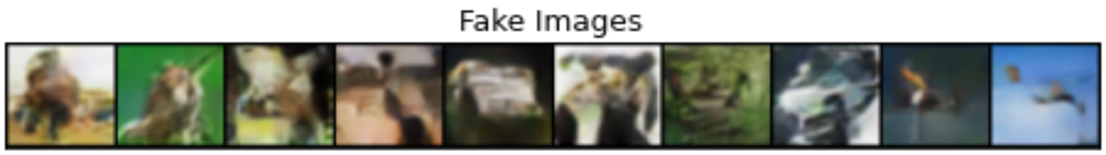
Requirements:
- CIFAR-10 dataset
- DCGAN Model
- WGAN Model
- ACGAN Model
- Python3
- Pytorch

Deep Convolutional Generative Adversarial Networks:
As our main model for creating images, we used DCGAN. The DCGAN architecture substitutes convolutional stride for the discriminator network and convolutional transpose for the generating network in place of fully connected layers. The generator consists of batch normalization layers, a ReLU activation function, a convolutional transpose which strides for

upsampling the latent vector and gives 3x64x64 image with a Tanh activation function. The discriminator network starts with an input of a 3x64x64 image and continues with batch normalization layers, a LeakyReLU activation function, and a convolutional stride for downsampling images, with the exception of the output layer which uses the sigmoid activation function.
We trained the DCGAN with the following hyper parameters:

- Size of the training batch is 128
- 100 for the latent vector dimension
- Adam with beta1 is 0.55 and beta2 is 0.999 is the optimizer
- learning rate: 0.0002
- Initialization of weights using a normal distribution with a 0.02 standard deviation

Images Generated:



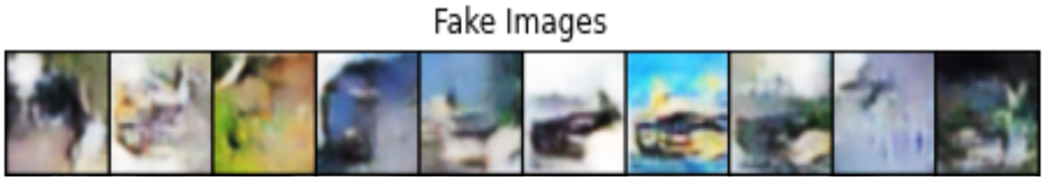
Wasserstein Generative Adversarial Networks:

Wasserstein GAN (WGAN) is a stable version of GAN that uses Wasserstein distance instead of Jensen-Shannon divergence for measuring the distance or similarity in the probability distributions of generative and distributive models. The discriminator in WGAN is also known as Critic, which goal is to maximize the distance between generated and real image data by Wasserstein distance. To stabilize training process I replaced linear activation function in the last layer of critic model to sigmoid function. The model uses RMSProp for optimization with no momentum and employs weight clipping on the Critic model's parameters. The network structure of WGAN is like DCGAN, but the Critic model uses linear activation at the output layer. During training, the Critic model is trained seven times for each epoch, while the generator is trained once.

We trained the WGAN with the following hyper parameters:

- Size of the training batch is 64
- 100 for the latent vector dimension
- RMSProp for the optimizer is set to zero
- Weight clipping is 0.015
- learning rate is 5e-5
- Critic Iteration is 7
- Initialization of weights using a normal distribution with a 0.02 standard deviation

Images Generated:



Auxiliary Classifier Generative Adversarial Networks:

The Auxiliary Classifier Generative Adversarial Network (ACGAN) utilizes both a generator and discriminator network. The generator produces a synthesized image that belongs to the input class label after receiving latent noise and a label as inputs. The discriminator receives an image as input and forecasts the probability that the image is real as well as its classification. To help the model produce the right image, the class label is used. The discriminator network architecture of ACGAN is like that of DCGAN, except it has two outputs: the class label and the probability of the image. The model architecture of the generator is very similar to DCGAN, it also accepts labels and latent noise as inputs.

We trained the WGAN with the following hyper parameters:

- Size of the training batch is 64
- 100 for the latent vector dimension
- Adam with beta1 is 0.5 and beta2 is 0.999 is the optimizer.
- learning rate is 0.0002
- Initialization of weights using a normal distribution with a 0.02 standard deviation

Images Generated:

Fake Images

Performance comparison among DCGAN, WGAN and ACGAN:

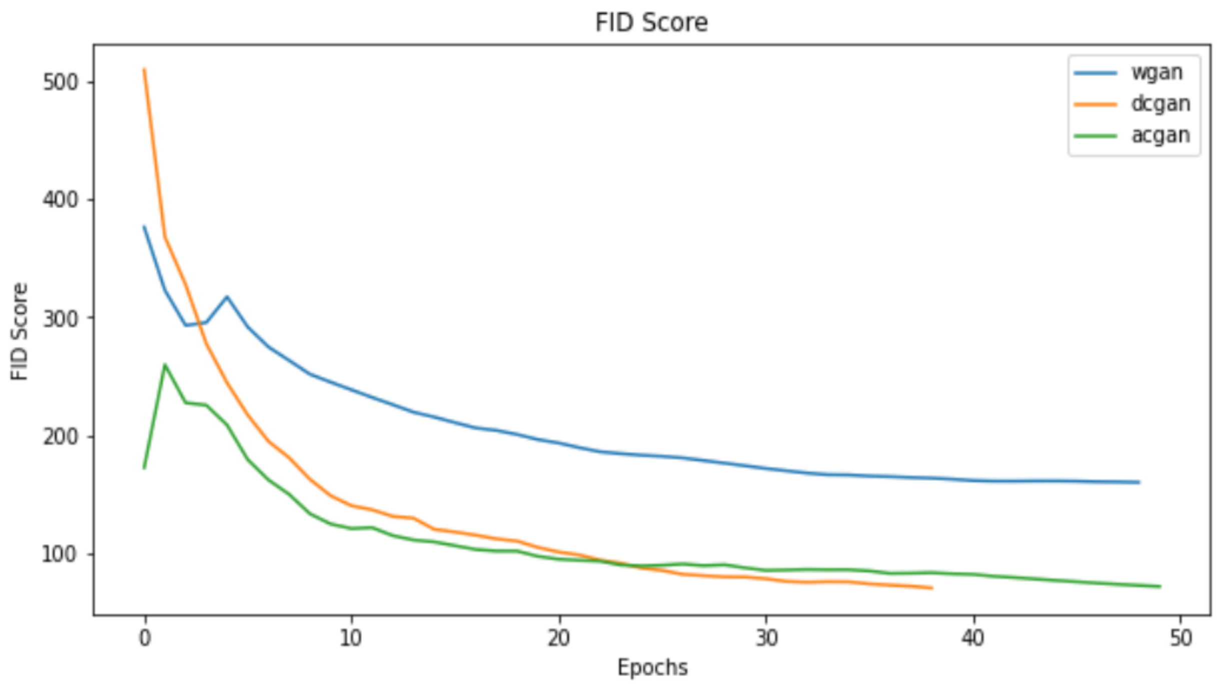
A popular metric for comparing the similarity of generated and real images based on their feature vectors is the Fréchet Inception Distance (FID). The Inceptionv3 model was used in this study to compute the FID by extracting feature vectors from both real and generated images. A higher degree of similarity between the two sets of images is indicated by lower FID scores. After each epoch, images were generated by the FID calculation using the most recent batch of training data and saved to a file. The FID scores for each model were recorded and presented in a table.

Conclusion:

Model	FID Score
DCGAN	50.20
WGAN	166.05
ACGAN	70.98

From the above table, compared to WCGAN, ACGAN we can see DCGAN has the lowest FID score of 50.20. That means DCGAN is generating images that are almost resembling the real-world images as we can see the performance in the images above.

Below shown is the plot of comparing generator performance of all models:


FID Score
wgan
dcgan
acgan
FID Score
Epochs