CPSC-Deeplearning-8430
Homework4-report

Pragathi Pendem
CUID: C15110727
GitHub link: https://github.com/Pragathibruno/Deeplearning-8430-Homework4-DCGAN-WGAN-ACGAN-_PragathiPendem

Introduction:

A neural network called a GAN (Generative Adversarial Network) has two components: a generator and a discriminator. GAN training aims to improve the generator to produce superior images that can trick the discriminator, which has been trained to differentiate between genuine and fraudulent images.

Deep neural networks (DNNs) are used as the generator and discriminator in DCGANs (Deep Convolutionary GANs), a particular type of GAN. They are among the most fundamental GAN variants.

Another kind of GAN that is helpful for modelling multi-level long-range dependencies in image dataset areas are WGANs (Wasserstein GANs). The real and generated images are separated using a Wasserstein distance metric, which is optimized.

GAN training's overall objective is to create the highest-quality images in a specific set of data, like the CIFAR-10 dataset.

Specifications:

To create image generation models using the CIFAR-10 dataset, the project requires working with either,

- Python 3 and Pytorch or TensorFlow 1.15.
- The two types of models used in this project or DCGAN and WGAN.

Dataset: Below is the brief explanation about the dataset that have used in this project.

The CIFAR-10 dataset consists of 60,000 32x32 color images that have been separated into ten classes, each of which has 6,000 images. 10,000 photographs are included in the dataset for testing and 50,000 images are used for training. For each image five training batches and one test batch are created. Aerial vehicle, automobile, bird, cat, deer, dog, frog, horse, ship, and truck are among the classes included in the CIFAR-10 dataset. It is a frequently used benchmark dataset in computer vision research for picture classification tasks.

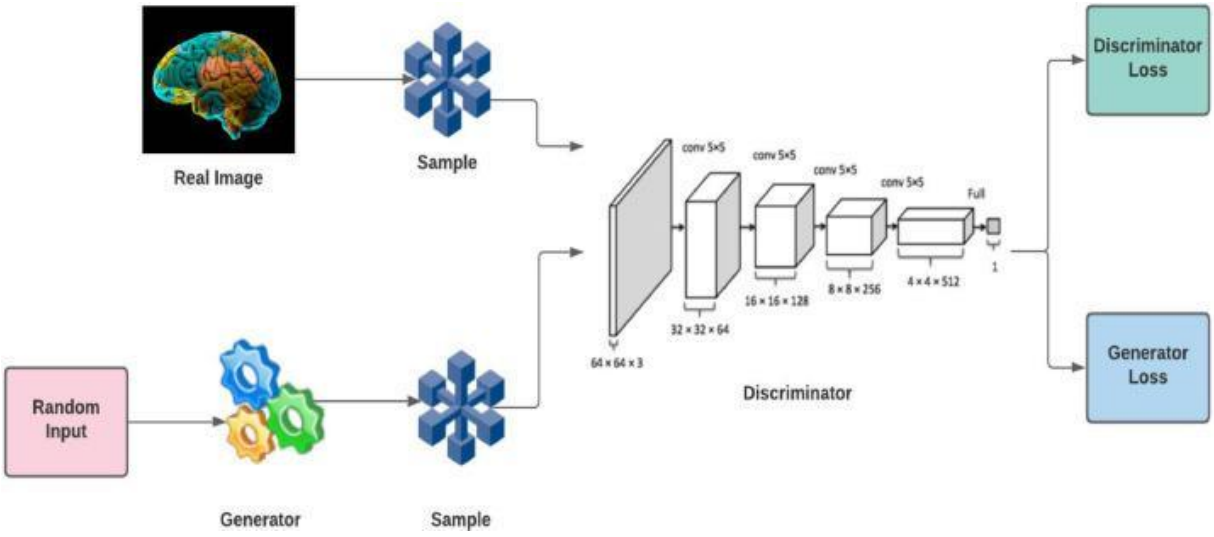
Models: The models used in this project are explained below.

Deep Convolutionary Generative Adversarial Network (DCGAN):

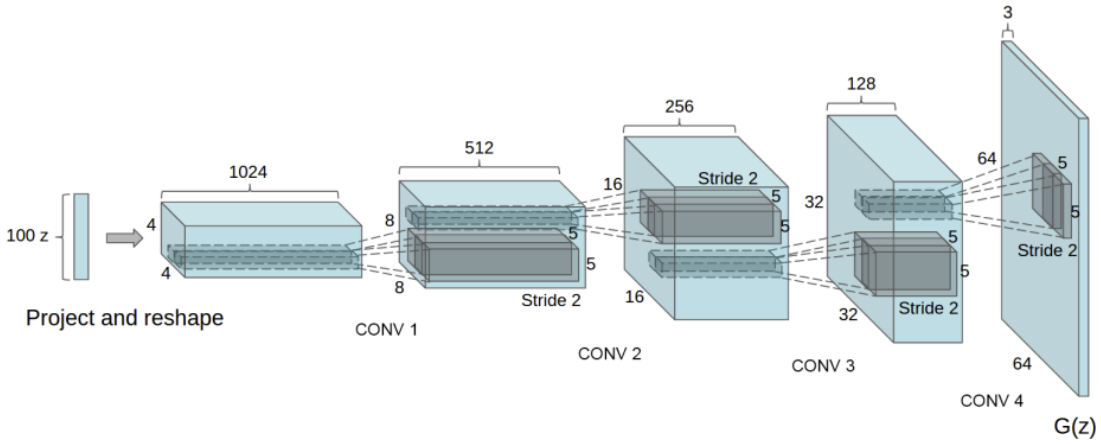
Convolutional layers are used in both the generator and discriminator networks of the DCGAN (Deep Convolutional GAN), a well-liked and effective version of GAN. Convolutional strides and transposed convolutions are utilized for down sampling and up sampling instead of the convolutional layers being fully coupled or pooled. Due to this method, DCGAN is very versatile and able to generate high-quality images without the need for intricate generator networks.

DCGAN advises employing convolutional layers rather than max pooling and transposed convolutional layers for up sampling to further enhance image quality. With the exception of the generator output and discriminator input layers, fully connected layers should be avoided. With the exception of the generator output and discriminator input layers, batch normalization is commonly utilized for all layers.

DCGAN employs ReLU and Leaky-ReLU for activation functions, but for the discriminator's The tanh function is frequently used in the output layer. In order to efficiently produce high-quality images, DCGAN's architecture stresses the usage of convolutional layers and the right activation functions.



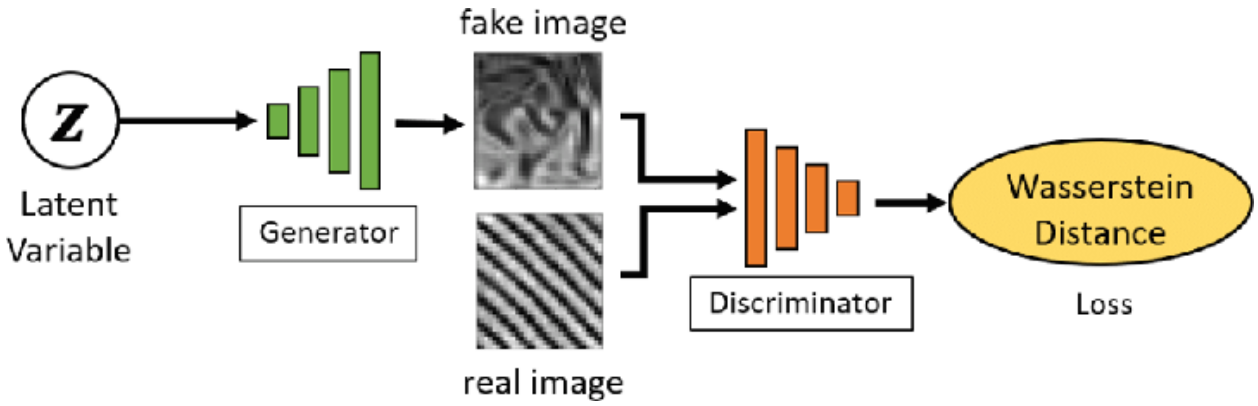
The above image is all about how actually the DCGAN works following is the brief explanation about the same Convolutional layers are used in both the generator and discriminator networks of a GAN called a DCGAN (Deep Convolutional GAN). For both down sampling and up sampling, it emphasizes the usage of convolutional layers rather than pooling layers and transposed convolutional layers. For effective picture generation, DCGAN's design also contains batch normalization and the relevant activation functions.



The above CNN image explains how DCGAN generator working in a CNN here is how it is A noise vector is converted into an image by the DCGAN generator using convolutional layers. The number of channels is decreased as the spatial resolution of the image continuously rises thanks to the convolutional layers. The pixel values are translated into a range between -1 and 1 by the tanh activation function at the generator's conclusion.

WGAN:

The Wasserstein GAN (WGAN) is a generative network that uses a loss function that is more closely connected with image quality and enhances the training stability of GANs. The implementation of WGAN, which is founded on sound mathematical concepts, requires just modest changes to DCGAN. The output layer of the discriminator now uses linear activation rather than sigmoid activation, and real images are scaled to -1 and fraudulent images to 1. It is advised to utilize a modest learning rate with no momentum during training and to examine the discriminator more frequently than the generator. Additionally, Wasserstein distance is used in WGAN to train the generator and discriminator, which necessitates limiting the weight updates.



Deployment:

Discriminative Model:
Inputting a picture sample and determining whether it is authentic, or fake is the initial stage in creating the discriminator model. The discriminator network is tasked with telling actual photos from fake ones in this situation, which is effectively a classification challenge. Starting with a standard convolution layer, the discriminator's architecture uses three convolutional layers rather than two 2x2 pooling layers to down sample the input picture. The discriminator's final layer, which makes no use of pooling, employs sigmoid activation to determine if an image is real or fraudulent. A discrete entropy loss function, which is ideal for classification problems with discrete categories, is the loss function used to train the discriminator.

Generative Model:
A technique for producing images that might not be completely correct in some dimensions is the generator model. A square picture from the latent space, a compressed form of the output space, is used

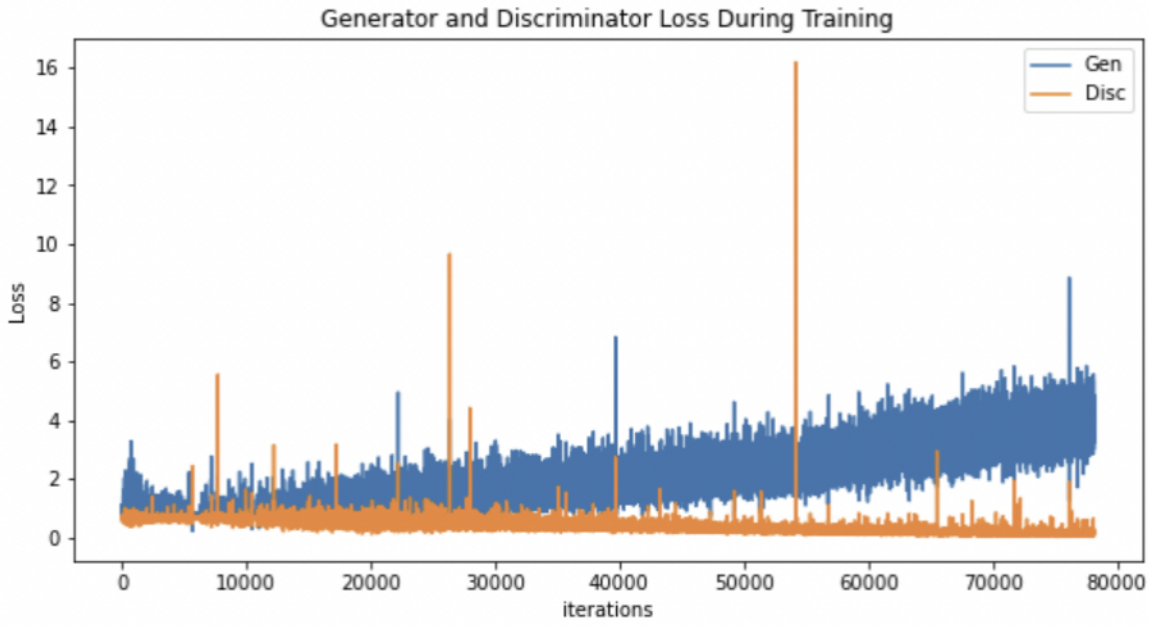
to do this. This method emphasizes the importance of the latent space and how it contributes to the production of images.

A thick layer with enough nodes is formed, identical to the previous secret layer, to produce a low-resolution image. Convolutional neural networks (CNNs) frequently employ this method to generate numerous parallel interpretations of the low-resolution image, which results in a variety of activation maps.

Conv2DTranspose, a feature that boosts the low-resolution image's resolution, is used in the following phase. When this procedure is repeated twice, it produces an image with a 32x32 resolution. In order to produce the final image output, the generator model primarily uses the latent space and low-quality images, which are then increased in resolution using Conv2DTranspose.

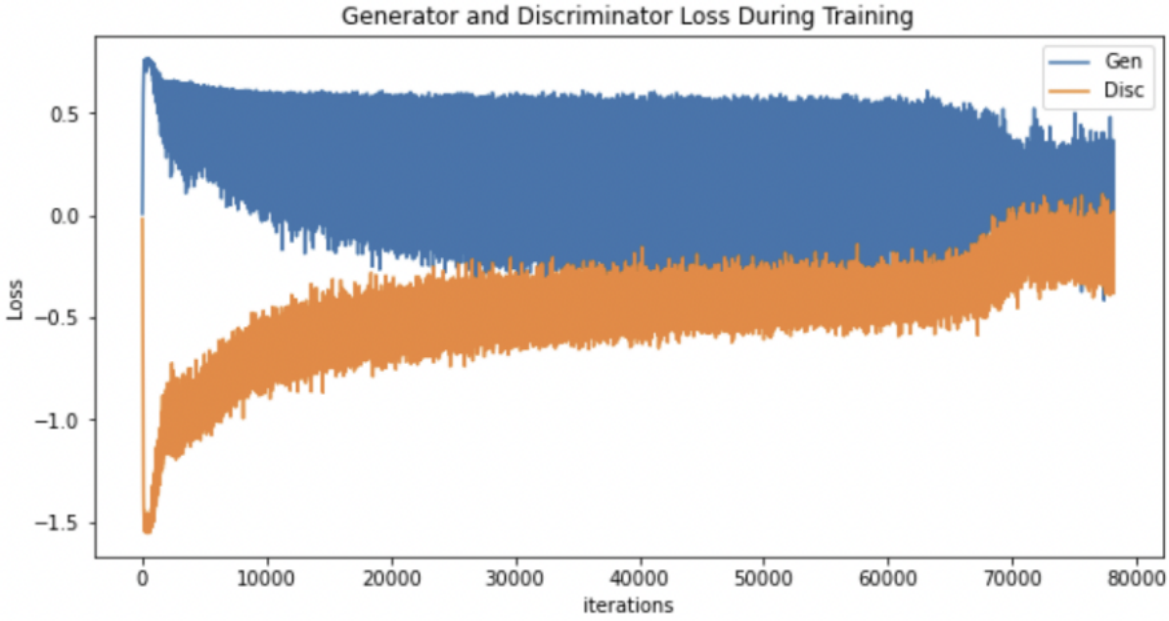
Results:
1) DCGAN
Number of epochs = 50
Learning rate = 0.0002
Latent space dimension = 128
Input image size = 3 X 64 X 64

Below is the discriminator and generative loss during training for DCGAN.



Comparison between real and fake images for DCGAN:

Below is the discriminator and generative loss during training for WGAN:
2) WGAN
Number of epochs = 50
Learning rate = 0.00005
Latent space dimension = 128
Input image size = 3 X 64 X 64



Comparison between real and fake images for WGAN:

```
[14]: plt.figure(figsize=(15,15))
      plt.subplot(1,2,1)
      plt.axis("off")
      plt.title("Real Images")
      plt.imshow(np.transpose(utils.make_grid(real_batch[0].to(device)[:64], padding=5, normalize=True).cpu(),(1,2,0)))
```

Performance Comparison among DCGAN,WGAN,ACGAN :
The primary focus of DCGAN is on network design enhancements, convolutional layers are used in both the generator and discriminator networks of the well-known GAN architecture known as the DCGAN. Although it produces realistic visuals with high levels of detail and texture, it can be unstable during training, making it difficult to teach.

In WGAN, the distance between the real and generated image distributions is calculated using the Wasserstein distance rather than the Jensen-Shannon divergence. Compared to DCGAN, it generates results that are more stable and trustworthy with less artifacts and mode collapse, although it may take longer to train.

Another GAN architecture called ACGAN integrates a support classifier into the discriminator network. This makes it handy for activities like picture synthesis, style transfer, and image editing since it helps generate images with certain attributes or classes. Even though it can produce high-quality images with particular features, it might need a more complicated network design and longer training cycles than DCGAN and WGAN.

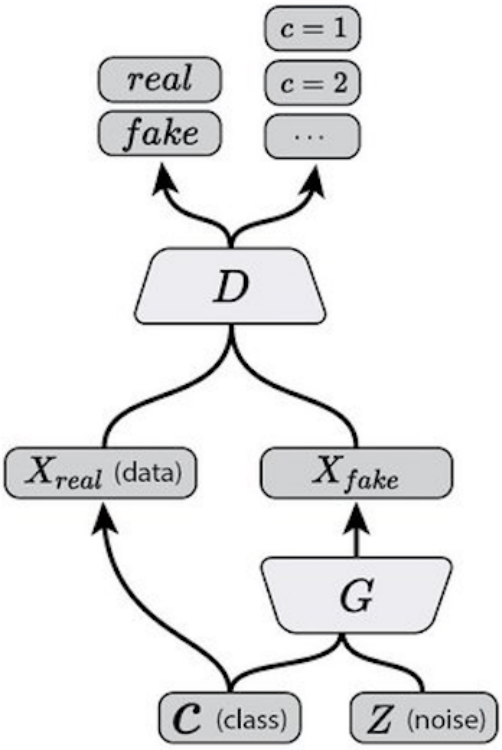
Bonus Part- ACGAN:

A form of conditional GAN called the AC-GAN model anticipates the class label of an input image through its discriminator, rather than receiving it as input. This approach allows for generating large, high-quality images while simultaneously learning a latent space representation that does not rely on the class label.

It also contributes to improving the stability of the training process. The AC-GAN generates images conditionally, similar to the conditional GAN, by providing the generator model with a point in the latent space and a class label.

Looking at it intuitively, the Generator and Discriminator in a GAN are essentially two different neural networks. The Generator should attempt to produce images that are as similar to the input photos as feasible.

In this method, the generator model receives two inputs: a class label and random points from the latent space, which are utilized to generate images for that specific class. The image generator may now produce images that are dependent on the class label by using the class label as input, making the image production and classification procedure class conditioned. This change enhances the training process' stability and enables the generator to generate images of a particular type based on the provided class label.



Since the discriminator model in the AC-GAN requires both as input, a picture, and the class label, it differs from the C-GAN. The discriminator predicts if a picture is real or false, just like the conditional GAN, but it also predicts the image's class label. Due to the shared model weights between the discriminator and auxiliary classifier, they can be viewed as independent models.

In actuality, Combining the discriminator and auxiliary classifier in the AC-GAN results in a single neural network model with two outputs. The likelihood that the input image is real is represented by the first output, which is calculated using a sigmoid activation function and optimized with binary cross-entropy. This result is comparable to the result of a typical GAN discriminator. The chance that an image belongs to each class is represented by the second output, which is produced by a SoftMax activation function

that has been enhanced with categorical cross-entropy. This result is comparable to the result of a standard neural network for multi-class classification.

Outcome of AC-GAN:
3) AC-GAN
Number of epochs = 100
Batch Size = 100
Learning rate = 0.0002
Latent space dimension = 64
Input image size = 3 X 64 X 64
Iterations = 100
Image after first epoch:

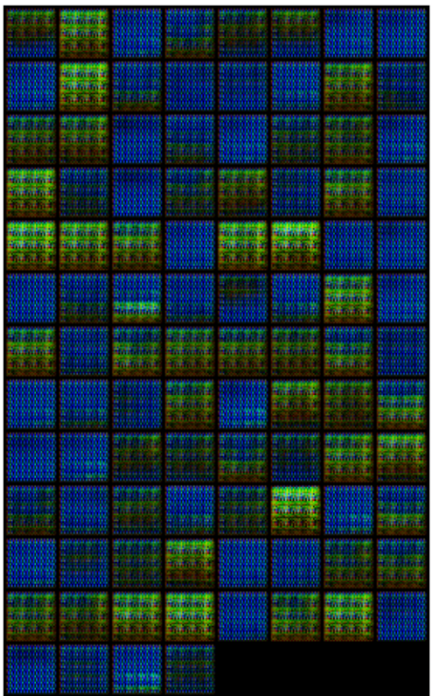


Image after last epoch: