CPSC - 8430
Homework 4 Report

Sravani Pati
GitHub Link: https://github.com/sravani919/Spati_HW4_Deep_Learning

Introduction

A discriminator or generator pair in a GAN must be trained to produce high-quality images identical to authentic images in a given dataset. The discriminator network gains the ability to distinguish between authentic and false images. In contrast, the generator network acquires knowledge to create counterfeit images replicating the distribution of natural images in the dataset. The discriminator network receives authentic images from the dataset during training and fake images created by the generator network. Each photo receives a probability score from the discriminator network, indicating how probable it is that it is accurate. In order to create false images that can deceive the discriminator network, the generator network is then updated based on the feedback it receives from the discriminator network. Together, the discriminator and generator networks are trained until the generator network can make fake images identical to the dataset's actual photos. At this stage, the GAN has mastered producing excellent ideas that may be applied to several tasks, including image synthesis, data augmentation, and more.

Models:

1) Convolutional neural networks are utilized for both the generator and discriminator networks in the Deep Convolutional GAN (DCGAN) architectural type of GAN. It has been effective at producing high-quality images, particularly in computer vision. Except for the generator output and discriminator input, the DCGAN skips ultimately linked layers and instead uses transposed convolutions for upsampling. After each layer, batch normalization is utilized to steady the training process. The discriminator tries to tell actual images from produced ones, while the generator creates images from noise as input. Up till the generator can produce realistic visuals, adversarial training is continued.

For downsampling in DCGAN, convolutional layers are recommended over max pooling layers. Except for the layers that comprise the generator output and discriminator input, completely connected layers are eliminated while transposed convolutions are employed for up-sampling. Except for the generator output and discriminator input layers, batch normalization is used. The discriminator's production is often run through a tahn function to produce a result in the -1 to 1 range. ReLU and Leaky ReLU activation functions are employed.

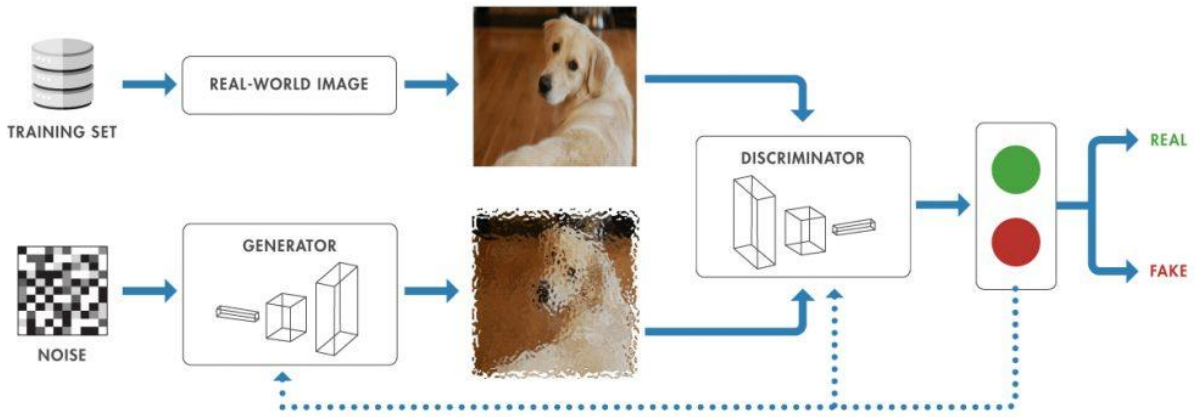


Figure 1: The process on images

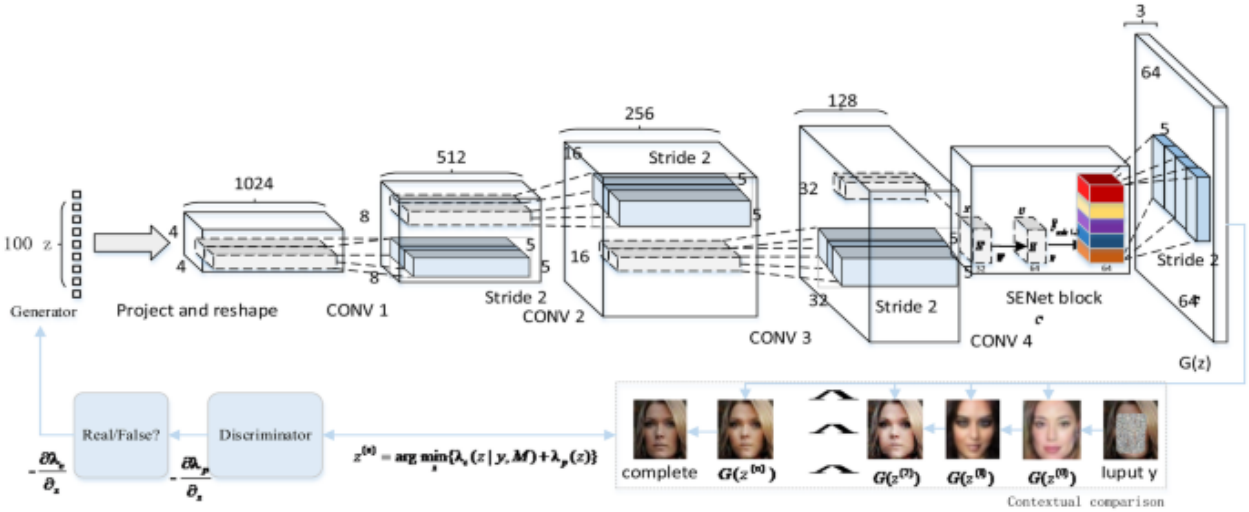


Figure 2:DCGAN generator working in a CNN

2)The training instability problem with GANs is addressed by the Wasserstein GAN (WGAN), a form of generative network that adds a loss function based on the Wasserstein distance. Since it measures the separation across probability distributions, this distance is a more trustworthy metric than the conventional cross-entropy loss. The DCGAN architecture has to be slightly altered for WGAN by applying linear activation and scaling actual and false pictures to -1 and 1, respectively. A low-learning, no-momentum form of RMSProp is employed to improve the network while the generator is examined less frequently during training than the essential model. The Wasserstein gradient penalty, which keeps the discriminator gradients from growing

out of control, is necessary for practical WGAN training. Minor weighted mini-batch updates are used to train the generator and critical models. A standard option for GAN architectures, especially where training stability is an issue, is WGAN, which has demonstrated remarkable effectiveness in producing high-quality images.Schematic for the Wasserstein GAN training process. A discriminator receives the real and fake images and computes a loss based on the Wasserstein distance after receiving the latent variable z from the generator and producing a false image in each iteration.

The CIFAR-10 dataset consists of 60,000 32x32 pixel color images in total. Ten classes with 6,000 photos each are created from these images. Ten thousand are set aside for testing, while the other 50,000 photos are used for training. Five training batches and one test batch, each with 10,000 photos, make up the dataset. The CIFAR-10 collection consists of ten images: an airplane, car, bird, cat, deer, dog, frog, horse, ship, and truck.

Execution

The Discriminator Model distinguishes between accurate and deceptive images and is a crucial part of the Generative Adversarial Network (GAN) architecture. A dataset of real photographs and a generator that creates fake images are used to train the model. Instead of the usual 2x2 pooling layers, the discriminator has three convolutional layers that downsample the input image. The discriminator model's last layer employs sigmoid activation to produce a probability value indicating whether the input image is authentic. The probability that the input image is actual is expressed as a single value in the output. The model employs a discrete entropy loss function, which is excellent for picture classification tasks. To prevent overfitting and ensure optimal performance, the discriminator model's design must be well thought out. The performance of the discriminator is affected by the activation functions, loss function, and the number of layers.

Outcome:

```
Epoch [24/50] Batch 800/1563                    Loss Discriminator: 0.382, loss Generator: 1.621
Epoch [24/50] Batch 900/1563                    Loss Discriminator: 0.446, loss Generator: 1.745
Epoch [24/50] Batch 1000/1563                    Loss Discriminator: 0.469, loss Generator: 1.215
Epoch [24/50] Batch 1100/1563                    Loss Discriminator: 0.360, loss Generator: 2.194
Epoch [24/50] Batch 1200/1563                    Loss Discriminator: 0.431, loss Generator: 1.409
Epoch [24/50] Batch 1300/1563                    Loss Discriminator: 0.363, loss Generator: 1.896
Epoch [24/50] Batch 1400/1563                    Loss Discriminator: 0.456, loss Generator: 1.442
Epoch [24/50] Batch 1500/1563                    Loss Discriminator: 0.481, loss Generator: 2.053
Fretchet Distance:  255.24758450056015
Epoch [25/50] Batch 0/1563                  Loss Discriminator: 0.448, loss Generator: 1.431
Epoch [25/50] Batch 100/1563                  Loss Discriminator: 0.574, loss Generator: 1.291
Epoch [25/50] Batch 200/1563                  Loss Discriminator: 0.596, loss Generator: 1.582
Epoch [25/50] Batch 300/1563                  Loss Discriminator: 0.406, loss Generator: 1.315
Epoch [25/50] Batch 400/1563                  Loss Discriminator: 0.443, loss Generator: 1.105
Epoch [25/50] Batch 500/1563                  Loss Discriminator: 0.494, loss Generator: 1.338
Epoch [25/50] Batch 600/1563                  Loss Discriminator: 0.400, loss Generator: 1.617
Epoch [25/50] Batch 700/1563                  Loss Discriminator: 0.550, loss Generator: 1.636
Epoch [25/50] Batch 800/1563                  Loss Discriminator: 0.537, loss Generator: 1.189
Epoch [25/50] Batch 900/1563                  Loss Discriminator: 0.369, loss Generator: 1.668
Epoch [25/50] Batch 1000/1563                    Loss Discriminator: 0.506, loss Generator: 1.459
Epoch [25/50] Batch 1100/1563                    Loss Discriminator: 0.490, loss Generator: 1.192
Epoch [25/50] Batch 1200/1563                    Loss Discriminator: 0.472, loss Generator: 1.270
Epoch [25/50] Batch 1300/1563                    Loss Discriminator: 0.649, loss Generator: 2.864
Epoch [25/50] Batch 1400/1563                    Loss Discriminator: 0.606, loss Generator: 1.179
Epoch [25/50] Batch 1500/1563                    Loss Discriminator: 0.548, loss Generator: 1.081
```

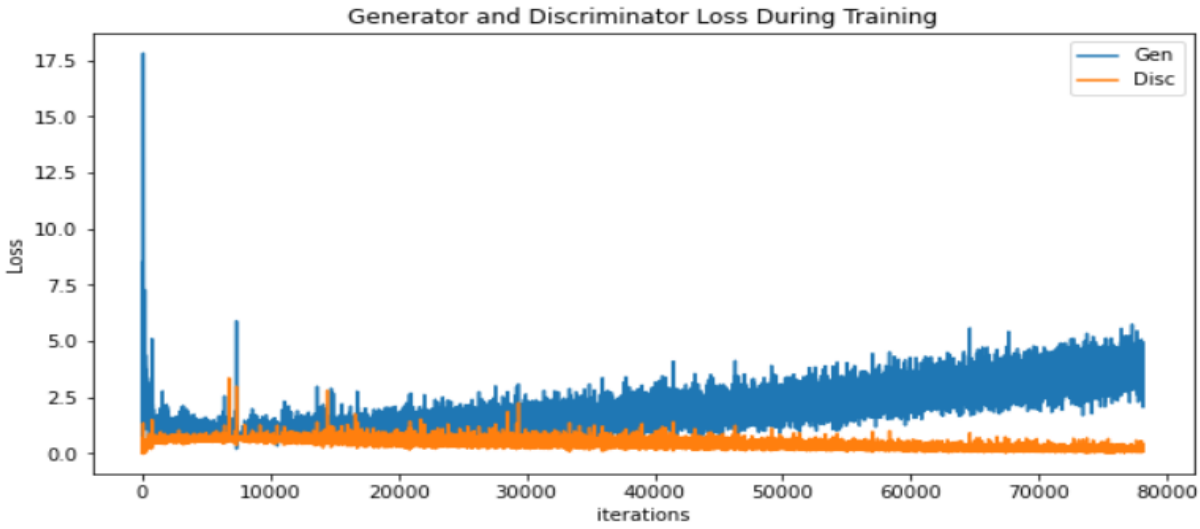
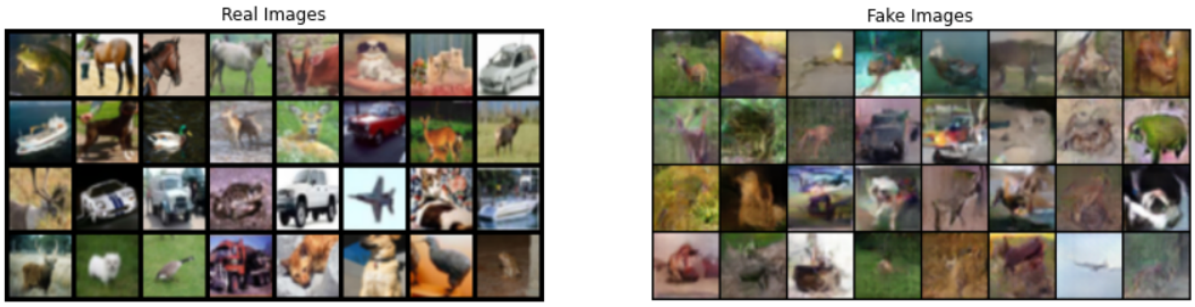
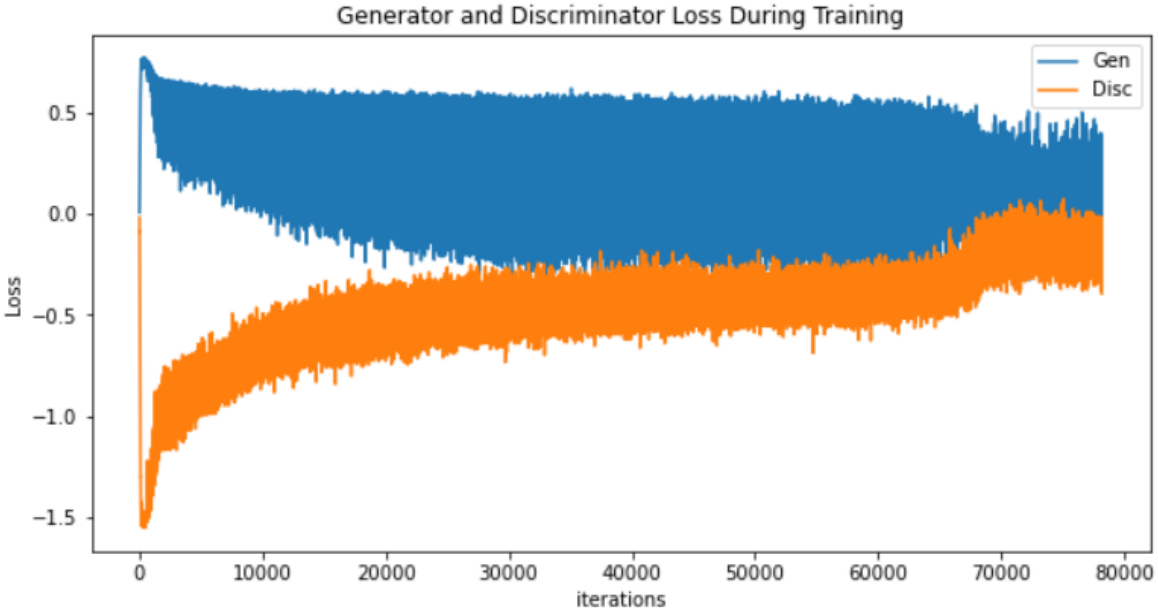


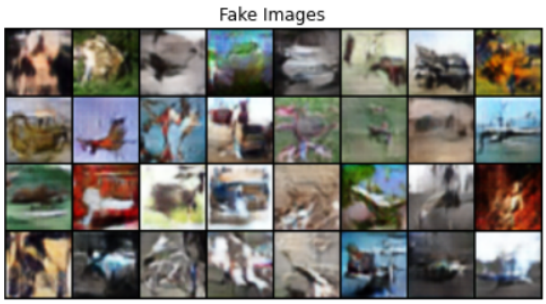
Fig: Generator and Discriminator loss



The GAN architecture's generator model tries to produce realistic images from random noise. These generated graphics could, however, have certain dimensional flaws and errors. A square image is extracted from the latent space and sent into the generator as input to solve this problem. The latent space, which serves as a condensed representation of the output space, is given significant weight by the generator. A dense layer is added to the model, identical to the first hidden layer, with the required nodes to reflect a low-resolution image to produce a low-resolution rendition of the image. As is common in CNN models, where several concurrent filters contribute to many low-resolution picture copies, several parallel interpretations are constructed to yield activation maps. The low-resolution image must be changed into a higher-resolution image in the following phase. This is accomplished by quadrupling the field in the Conv2DTranspose layer's input maps. To achieve the desired 32x32 performance image, two more cycles are needed. This is accomplished by creating numerous low-resolution image versions and then using the Conv2DTranspose layer to transform each into a high-resolution image. The discriminator model provides feedback to the generator model, which is continuously improved. The generator model is trained using a loss function comparing the generated images to the actual ones.

Outcome





Comparisons:

Training Stability: It has been demonstrated that WGAN, particularly for producing high-quality images, has superior training stability than DCGAN and ACGAN. This is because the WGAN's Wasserstein distance loss function, which is better at penalizing the generator for generating images of low quality, is utilized.

Scalability: DCGAN is well renowned for its scalability, making it possible to produce high-quality images across various image resolutions and dataset sizes. Instead of fully linked layers, deep convolutional neural networks are used to accomplish this, which might cause a bottleneck in more extensive networks.

Conditional Image Generation: ACGAN is a superior option for applications where the generated images must have specified properties because it is specifically developed to create ideas conditioned on specific qualities. ACGAN, for instance, can produce pictures of persons with particular hairstyles, attitudes, or attire.

Diversity of Generated Images: It has been demonstrated that WGAN and DCGAN produce more diverse images than ACGAN, which occasionally experiences mode collapse. Instead of

generating a wide variety of prints, the generator only creates a few pictures that meet a specific set of qualities.

Performance on Specific Tasks: The model selection is based on the particular image-generating task. For instance, DCGAN might be a better option for producing high-quality photographs of natural settings, whereas WGAN might be a better option for creating high-quality medical images. ACGAN is best suited for applications where the generated images must have particular qualities or properties.

BONUS:

Auxiliary Classifier Generative Adversarial Network is referred to as ACGAN. It is a deep learning model that blends generative adversarial networks (GANs) with additional classifiers to produce images that satisfy particular attribute requirements. The discriminator in ACGAN takes the image and the attribute label as inputs and outputs a probability score indicating whether the image is real or fake and a probability distribution over the attribute labels. The generator in ACGAN takes a random noise vector as input and produces an image.

The attribute labels for actual and fake images are predicted using the auxiliary classifier in ACGAN. This allows the generator to practice producing images that match precise attribute criteria, like a specific color or form. Compared to other GAN models, ACGAN provides several benefits. It can produce images with particular qualities, which makes it ideal for projects like data augmentation and image-to-image translation. Additionally, it enables more precise control over the process of creating images.

ACGAN, however, can experience mode collapse when the generator only generates a small number of images that match a particular set of attributes instead of a wide variety of snaps. Researchers have suggested several methods to overcome this problem, including spectral normalization and minibatch discriminating. The ability to generate images with particular qualities makes ACGAN a potent tool that may be applied to various tasks, such as image-to-image translation, data augmentation, and the creation of synthetic data for training machine learning models.

Images after first and last epoch: