CPSC 8430:
Deep Learning

Homework 3

Reek Majumder

GitHub Repo:-
https://github.com/reek129/CPSC8430_HW2_V2/tree/main/hw3

Dataset: - CIFAR10

The CIFAR-10 dataset consists of 60000 color images of size 32 x 32. The dataset comprises ten classes airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks, with 6000 images for each class.

Generative Adversarial Networks (GANs) – Introduction

A generative adversarial network also referred to as GAN, is a class of machine learning frameworks designed by Ian Goodfellow and his colleagues in 2014. It involves automatically discovering and learning the regularities or patterns in input data so that the model can be used to generate or output new examples that plausibly could have been drawn from the original dataset.

GANs neural networks contain two networks: the generator network, which generates new examples, and the discriminator network, which classifies examples as either real from the training dataset or fake generated. The two models are trained together in a zero-sum game, adversarial until the discriminator model has been fooled about half the time; the generator model generates plausible examples.

The discriminator takes all the real and fake images and returns the odds of 0s and 1s with 1. Given a training set, this technique learns to generate new data with the same statistics as the training set. Results in images authentic to the human observer and having realistic characteristics. Although initially proposed for unsupervised learning. GANs have also proven helpful for supervised, semi-supervised learning, and Reinforcement learning.

DCGANs

DCGANs is a class of CNNs called deep convolutional generative adversarial networks (DCGANs), that have certain architectural constraints, and demonstrate that they are a strong candidate for unsupervised learning. DCGAN is one of the GAN network developers that is popular and successful.

DCGAN down and up sampling is employed using convolutional stride and transposed convolution. Except for the generator's output layer and the discriminator's input layer, DCGAN uses batch normalization and replaces all max-pooling layers with convolutional stride. It also uses transposed convolution for up sampling, and eliminates connected layers by using dropout while training. A non-activation layer such as RelU is used in the generator, and in the discriminator, Leaky ReLU is used.

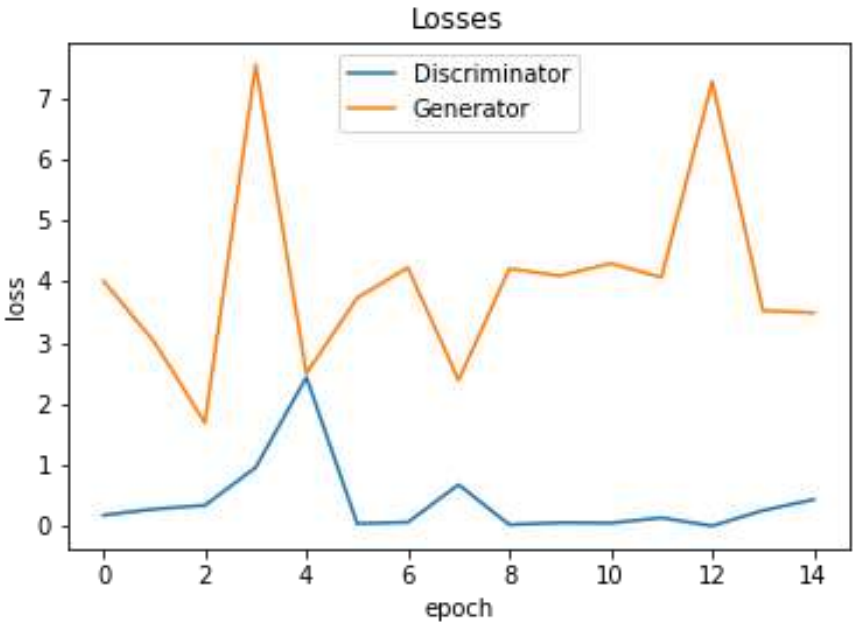

Figure 1: Loss Generated by Discriminator and Generator of DCGAN

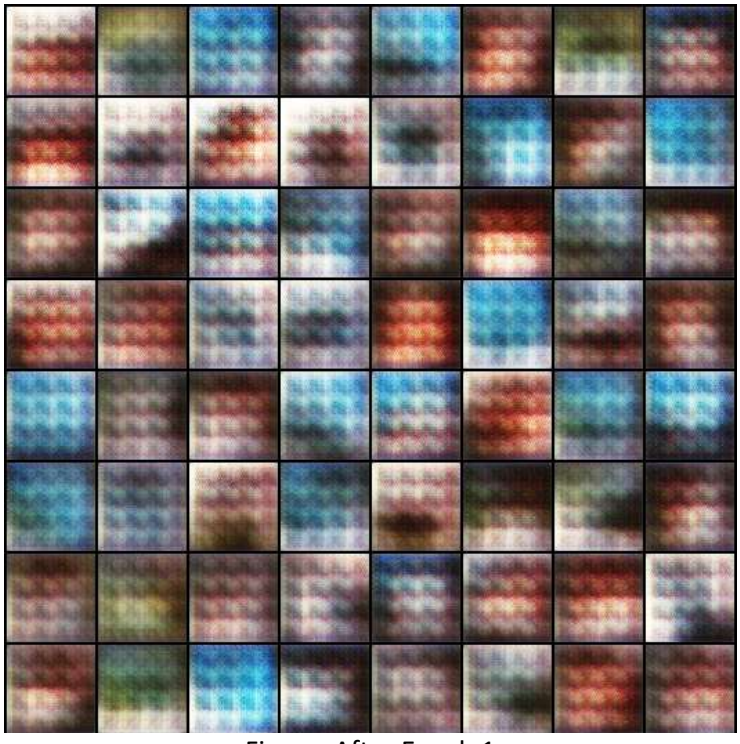
Figure: After Epoch 1

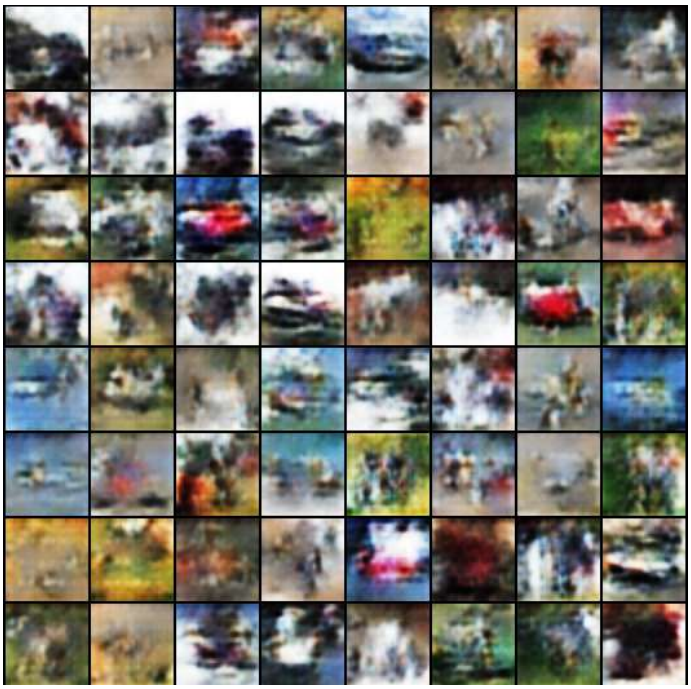

Figure : Results after 5th Epoch

Figure: Image after Epoch 10

WGAN

The Wasserstein GAN expands into the generative network, improving reliability during model training and having a loss feature related to image quality. WGAN offers higher stability to the training model than simple GAN architectures. The loss function utilized in WGAN also gives us a termination

criterion for evaluating the model. While it may sometimes take slightly longer to train, it is one of the better options to achieve more efficient results.

$$\max_{w \in \mathcal{W}} \mathbb{E}_{x \sim \mathbb{P}_r}[f_w(x)] - \mathbb{E}_{z \sim p(z)}[f_w(g_\theta(z)]$$

Figure : WGAN Equation for optimization

The max value represents the constraint on the discriminator in the above equation. The discriminator is referred to as the critic in the WGAN architecture. The absence of a sigmoid activation function to limit the values to 0 or 1, which means real or fake, is one of the reasons for this convention. Instead of returning a single value, the WGAN discriminator networks return a range of values, allowing it to act less strictly as a critic.

The real data is represented by the first half of the equation, while the second half represents the generator data. The discriminator (or the critic) in the above equation aims to maximize the distance between the real data and the generated data because it wants to distinguish the data successfully. Because it wants the generated data to be as real as possible, the generator network aims to reduce the distance between real and generated data.

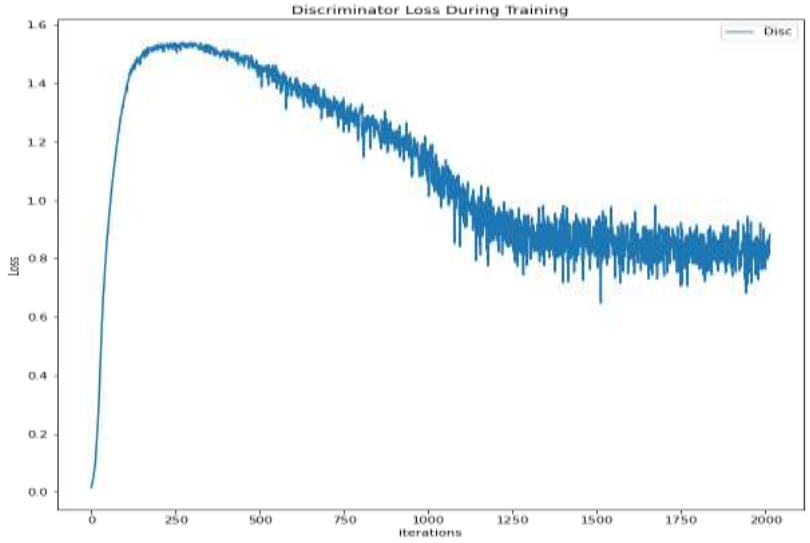

Figure: - Discriminator Loss during Training

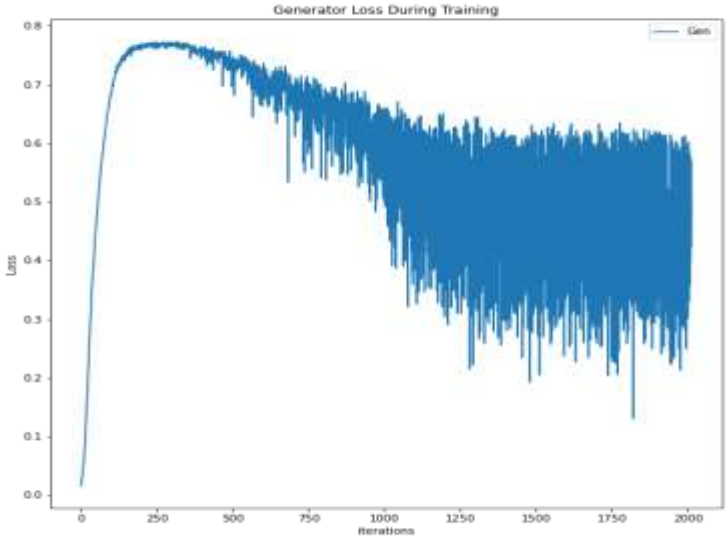

Figure: Generator Loss during training

Figure: WGAN images After 5 epochs

ACGAN

Auxiliary Classifier Generative Adversarial Network (ACGAN) is a highly specialised GAN that excels at image synthesis. Another important feature of ACGAN is that, when compared to previous approaches, it produces images with a relatively high resolution. However, any image can be resized and its size increased using bi-linear interpolation. ACGAN was the first to propose using a pre-trained Inception network to check image discriminability.

As with any GAN network, the ACGAN has a generator and a discriminator. In the ACGAN, however, in addition to the noises z, each generated sample has a corresponding class label c to C (Available classes). The model uses this class label to synthesize the image based on the label passed in. The image is generated by the generator G, which uses both the class label and the noise z.

The generator's architecture is straightforward. It is made up of a series of deconvolutional layers (also known as transposed convolutional layers) that are stacked on top of each other.

A transposed convolutional layer is a convolutional layer that has padding added to the original input. As a result, when convolution is applied with stride 1 and no padding, the output's height and width are greater than the input's. We can perform up-sampling with a transposed convolutional layer with a stride of 2, just as we can perform down-sampling with a stride 2 convolutional layer.

The loss function of ACGAN is divided into two parts
a) The log likelihood for the source being checked
b) The log likelihood of the class being checked

$$Souce\ Loss(\ L_s) = E[\log \quad P\ (S = real|X_{real})] + \ E[\log \quad P\ (S = fake|X_{fake})]$$

$$Class\ Loss(L_c) = \ \ E[\log \quad P\ (C = c|X_{real})] + \ E[\log \quad P\ (C =|X_{fake})]$$

As is evident from the above loss functions, the generator and the discriminator 'fight' over this loss function. The generator and the discriminator both try to maximize the class-loss. The source-loss is however a min-max problem. The generator tries to *minimize* the source-loss and fool the discriminator. The discriminator on the other hand tries to *maximize* the source-loss and tries to prevent the generator from gaining an upper hand.

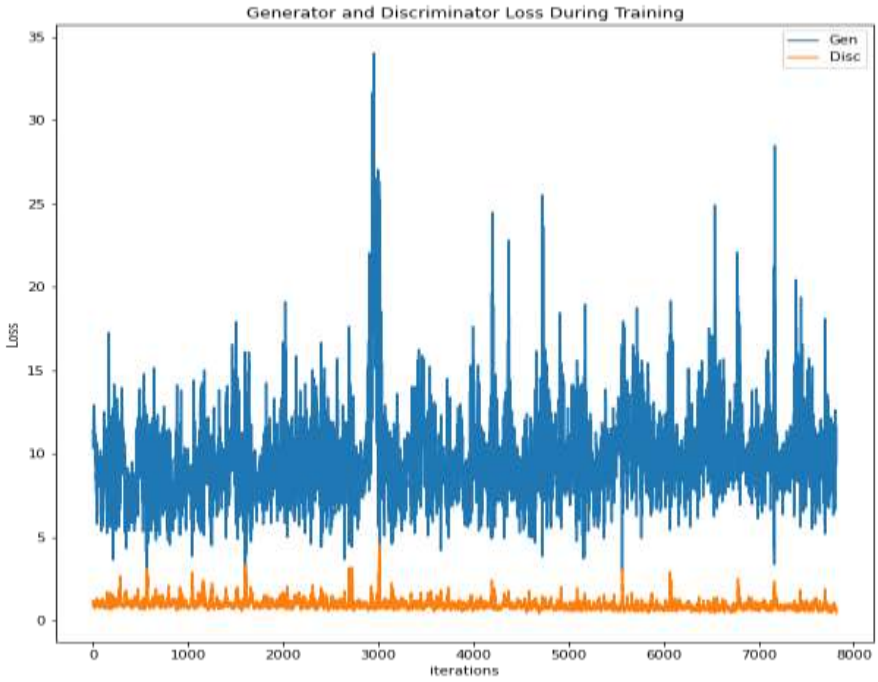

Figure:- Discriminator and Generator loss after 10 epochs

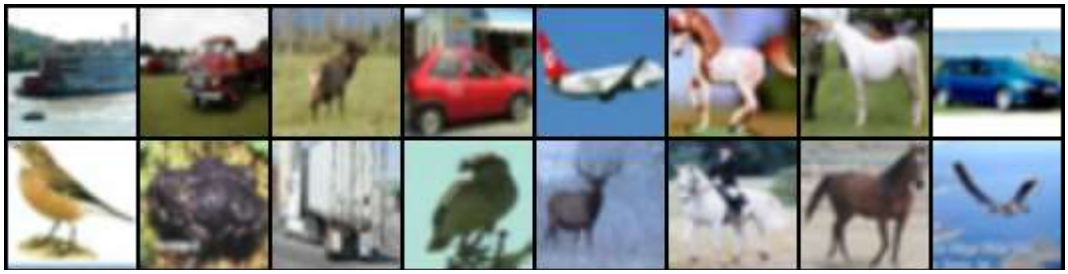

Figure: ACGAN Results after 5 Epochs

Figure: ACGAN Results after 10 Epochs