Greeshma Chandra Shekar **C20084146** **Email:** greeshc@g.clemson.edu

CPSC-8430
Deep Learning
Homework 1

GitHub Link: https://github.com/Greeshma-C20084146/hw4.git

1. Generative Adversarial Networks (GANs)

Generative Adversarial Networks (GANs) have been utilized in prior research to create new data points from various databases. In this assignment, I utilized three different types of GANs, namely DCGAN, WGAN, and ACGAN, to generate realistic images using the cifar-10 image database. The cifar-10 database comprises 50,000 images of 32x32 pixels with RGB color channels, which are categorized into ten distinct classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck.

The architecture of Generative Adversarial Networks (GANs) is straightforward, consisting of two key components: the generator (G) and discriminator (D). To generate synthetic images, the generator takes in input noises z and generates images that simulate real ones. The discriminator takes in both real and fake images, either x ~ pr for real or x ~ pg for fake and then outputs the likelihood of the image being real, that is, 1 for genuinely realistic or 0 for completely fake).

The loss functions used to train the generator and discriminator are as follows:

$$L_{\text{discrim}} = -E_{x\sim p_r}[\log(D(x))] - E_z[\log(1 - D(G(z)))] \quad (1)$$

$$L_{\text{gen}} = -E_z[\log(D(G(z)))] \quad (2)$$

2. Deep Convolutional Generative Adversarial Networks (DCGAN)

Deep Convolutional Generative Adversarial Networks (DCGANs) employ a convolutional neural network to construct the generator and discriminator, rather than using the feedforward neural network, which is typically utilized in conventional GANs. In this particular assignment, I used DCGAN to generate realistic photos from the cifar10 database. To construct the neural network, I utilized Tensorflow, and primarily relied on the functions tf.layers.conv2d and tf.layers.conv2d transpose. Table 1 displays the various training parameters that were used.

Optimizer	Adam	**Number of epochs**	100
Learning rate	5e-5	**Batch size**	64

The image samples produced are presented in the figure below.

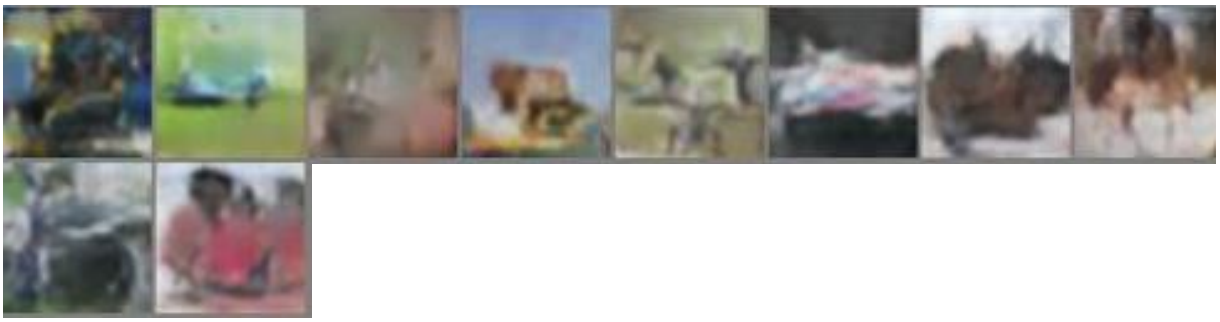

Figure 4: Top 10 DCGAN generated images

The DCGAN was evaluated, and the resulting FID score was determined to be 1.553204.

3. Wasserstein Generative Adversarial Networks (WGAN)

The loss function for conventional GANs (as described in Equations (1)-(2)) is designed to minimize the Jensen-Shannon (JS) divergence between the real and generated image distributions. However, when there is no overlap between these two distributions, the JS divergence is always constant. This is not ideal for gradient-based network training. To resolve this issue, the Wasserstein distance is employed to quantify the distribution difference. As a result of this, the loss functions for the generator and discriminator undergo some modifications and can be expressed as follows:

$$L_{\text{discrim}} = -E_{x\sim p_r}[D(x)] + E_z[D(G(z))] \quad (3)$$

$$L_{\text{gen}} = -E_z[D(G(z))] \quad (4)$$

To ensure that the k-Lipschitz property is met, the neural network weights are clipped. For this project, the implementation of WGAN is created using Tensorflow, with network structures identical to those of the previously described DCGAN in Section 2. The training configuration is detailed in Table 2.

Optimizer	RMSProp	**Number of epochs**	200
Learning rate	1e-3	**Batch size**	64

The figure below shows examples of the generated images.

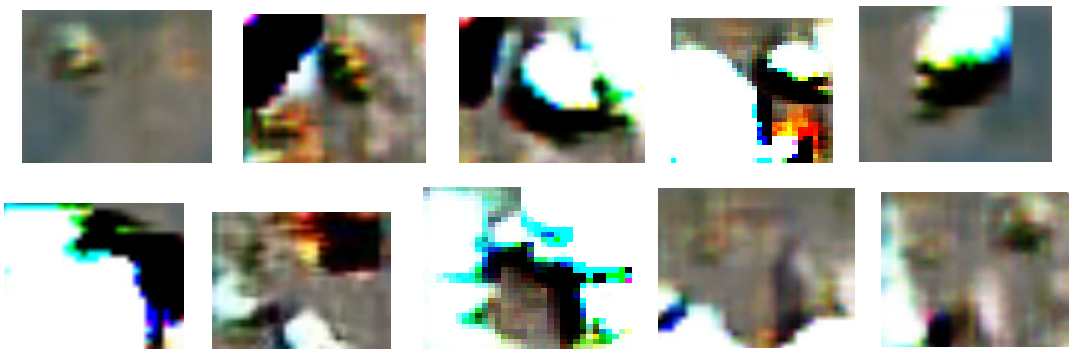

Figure 5: Top 10 WGAN generate images

The FID score achieved by the WGAN is 2.287725.

4. Auxiliary Classifier Generative Adversarial Networks (ACGAN)

To improve the performance, an additional classifier that can output the class of the pictures is added on top of the structure of the conventional GAN discriminator. The generator of ACGAN takes the fake label l as an extra input, while the discriminator still takes the images as the input. Due to the additional classifier, the discriminator is also able to predict the class of the input image. Fig. 6 shows the structure of ACGAN.

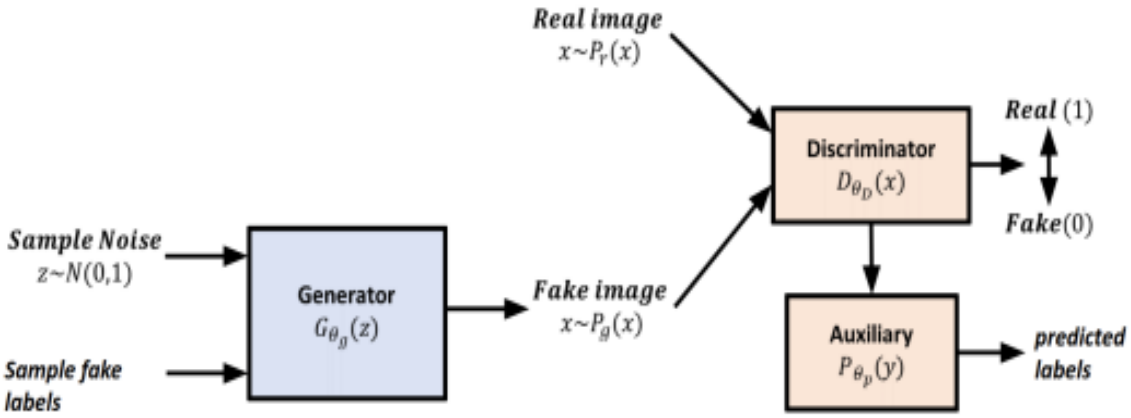

Figure 6: Diagram of the ACGAN

In this assignment, I constructed the ACGAN using Tensorflow, utilizing the same generator network structure as DCGAN.

As a new structure has been introduced, there have been changes in the loss functions of both the generator and discriminator, as shown below.

Greeshma Chandra Shekar C20084146 Email: greeshc@g.clemson.edu

$$L_{\text{discrim}} = -E_{x\sim p_r}[\log D(x)] - E_z[\log(1 - D(G(z)))] - E_{x\sim p_r}[\log P(c|x)] - E_{z,l}[\log P(c|G(z,l))] \quad (5)$$

$$L_{\text{gen}} = -E_z[\log D(G(z))] - E_{z,l}[\log P(c|G(z,l))] \quad (6)$$

To train the ACGAN model, the same training parameters as those used for the DCGAN in Section 2 are employed.

Figure 8: Top 10 ACGAN generated images

A score of 2.4819674 is obtained for the FID of ACGAN.

5. The comparison between DCGAN, WGAN, and ACGAN

This assignment involved building three different GANs (DCGAN, WGAN, and ACGAN) to generate images using the cifar-10 dataset. The FID scores for each GAN were 1.553204, 2.287725, and 2.4819674 for DCGAN, WGAN, and ACGAN respectively. It is worth noting that, based on the hyperparameters used in this project, ACGAN performed similarly to DCGAN and was also able to classify images. Conversely, WGAN's performance was worse than the other two GANs generally, however in this scenario DCGAN seems to be the worse, resulting in less realistic images. However, I ran DCGAN for 100 epochs only hence DCGAN FID score seems to be less due to less training. As the loss function is measured by the Wasserstein distance, it's possible that the hyperparameters and structure of WGAN could benefit from further optimization.

Further fine-tuning is required to improve performance. As a result, the ten most optimal images generated by DCGAN, illustrated in Figure 4, should be the focus of this assignment.