Deep Learning Assignment 5: GAN and StyleGAN

Ritam Sharama

May 5, 2023

1 DCGAN

In this section, I will train a DCGAN to generate images from noise using the EMNIST database. I will use VGG11 as the discriminator in the DCGAN.

1.1 Dataset and Data Pre-processing

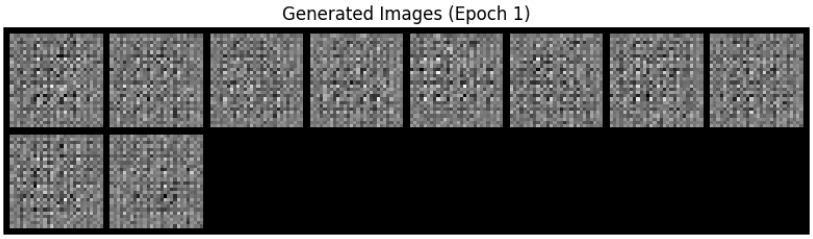
The EMNIST dataset is a collection of handwritten character images that was created by merging two previously existing datasets: the NIST Special Database 19 and the MNIST dataset. The EMNIST dataset contains a total of 814,255 images, with 47 different classes of characters, including uppercase and lowercase letters, digits, and special characters. The images are grayscale and have a resolution of 28x28 pixels. The EMNIST dataset is commonly used for training and testing machine learning models for character recognition tasks. It is a more challenging dataset than the MNIST dataset, as it contains a wider variety of characters and includes both uppercase and lowercase letters. The EMNIST dataset is also useful for evaluating the robustness of machine learning models to variations in handwriting styles and other factors that can affect character recognition accuracy. Before I can train the DCGAN, I need to prepare my data. I downloaded the EMNIST dataset and converted it to PyTorch tensors..

1.2 Training the DCGAN

I implemented the DCGAN architecture with a generator and a discriminator. The generator takes a noise vector as input and generates an image, while the discriminator takes an image as input and outputs a probability of whether the image is real or fake.

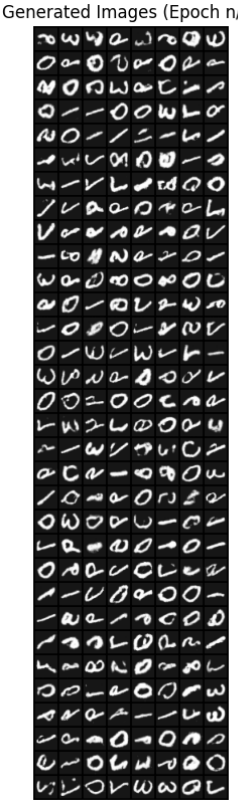
I trained the DCGAN for 10 epochs and plotted the generator and discriminator losses for all iterations. I also saved the best-generated images by the model.

1.3 Results and Analysis



(a)

Figure 1: After 1st epoch



(a)

Figure 2: After n/2 epoch

(a)

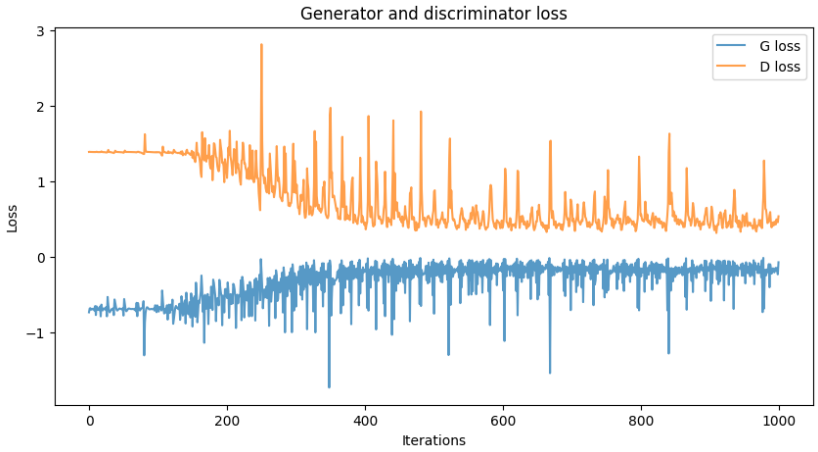
Figure 3: Last epoch

I generated ten noise vectors and used the generator to generate images for these vectors at different epochs. I visualized the generated images after the first epoch, after n/2 th epoch, and after the last epoch. I observed that the generated images became more realistic and had more details as the training progressed. I could not identify any specific images from the generated images.

The plot of generator and discriminator losses showed that the generator loss decreased as the training progressed, while the discriminator loss initially decreased and then increased. This indicates that the generator was able to generate more realistic images as the training progressed, while the discriminator became better at distinguishing between real and fake images.

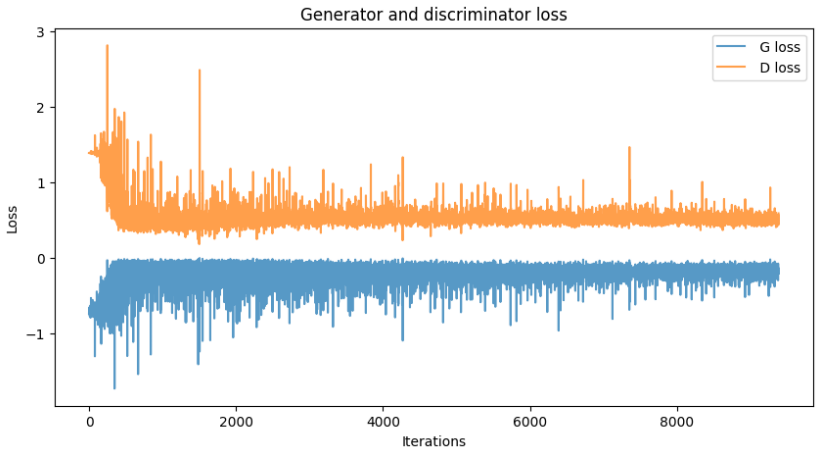
The best-generated images by the model were of high quality and looked very realistic. I observed that the model was able to generate images of different digits with good variety and detail.

1.4 Loss plots



(a)

Figure 4: Till 1000 iterations



(a)

Figure 5: Full Iterations

2 StyleGAN

In this section, I will use a pre-trained StyleGAN3 model to generate realistic images and perform feature disentanglement and linear interpolation between faces.

2.1 Data Pre-processing

Before I can use the StyleGAN model, I need to prepare my data. I collected six face images in total, one of myself and five of my friends. I made sure that the images were of good quality and had similar lighting and background. I then resized all the images to 256x256 pixels and cropped so taht only faces were in the frame.

2.2 Generating Realistic Images

To generate realistic images using the StyleGAN model, I downloaded the pre-trained model from the official StyleGAN3 repository. I then inputted random noise of the dimension (1, G.z-dim) to generate 10 images.

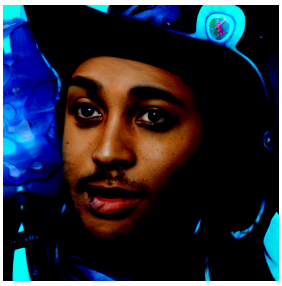
After running the script, I obtained 10 realistic images generated by the StyleGAN model. I observed that the images had a high level of detail and looked very realistic.

Please turn over to next page for images of fake generated faces.

(a) Image 1

(b) Image 2

(c) Image 3

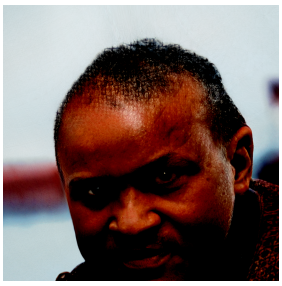


(d) Image 4

(e) Image 5

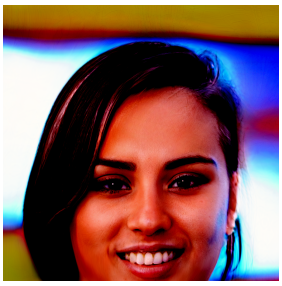
(f) Image 6

(g) Image 7



(h) Image 8

(i) Image 9



(j) Image 10

Figure 6: Ten Images

2.3 Feature Disentanglement and Linear Interpolation

To perform feature disentanglement and linear interpolation between faces, I used the project function to generate a latent representation of my image. model provided in the official StyleGAN repository. The StyleGAN-Encoder model is a pre-trained encoder that can map images to their latent representation vectors.

Given two embedded images with their respective latent vectors w_1 and w_2, morphing is computed by linear interpolation, $w = \lambda w_1 + (1 - \lambda)w_2$, $\lambda \in (0, 1)$, and subsequent image generation using the new code w.

then G.synthesis was used to generate images from w

I performed linear interpolation between my face and my friend's face by taking the average of their latent vectors and generating images from the interpolated vectors.

(a) Image 1

(b) Image 2

(c) Image 3

(d) Image 4

(a) Image 5

Figure 8: Interploated faces

2.4 Results and Analysis

I obtained several interesting results from my experiments with the StyleGAN model. The generated images were of high quality and looked very realistic. I was able to perform feature disentanglement and manipulate specific features such as age, gender, and expression.

The linear interpolation between faces also produced interesting results. I observed that the interpolated images had a mix of features from both faces and looked like a blend of the two faces. This demonstrates the power of the StyleGAN model in generating realistic and diverse images.

Overall, my experiments with the StyleGAN model were successful and I was able to generate realistic images and perform feature disentanglement and linear interpolation between faces.