Project Four

Deep Convolutional Neural Networks and GANs

Introduction

This experiment focused on the use of Deep Convolutional Generative Adversarial Networks (DCGANs) to generate images of increasingly convincing handwritten characters from the EMNIST dataset.

GANs are networks underpinned by two connected sub-networks, a Generator (G) and a Discriminator (D). The role of the discriminator is to distinguish between real and synthetic images, while the role of the generator is to generate increasingly representative samples of the real distribution. The networks pass update information through the image (a high dimensionality vector) and use adversarial learning to create mutual updates, building stronger capability for generating and distinguishing detailed information over time. First, weights are randomly initialised. Then, we train discriminator D by fixing G. Next, we fix D and update G to train the generator, flipping real and synthetic labels to push the probability closer to one. In this way, we continue to alternate fixing and updating until the model reaches convergence. Mathematically, this is expressed as:

$$\boldsymbol{G^* = argmin_G Div(P_G, P_{data}) \leftarrow max_D V(G,D)} \textit{ and}$$

$$\boldsymbol{D^* = argmax_D V(D,G)} \textit{; i.e.,}$$

$$\boldsymbol{min_G max_D V(G,D) = E_{x\sim P_{data}(x)}[logD(x)] + E_{z\sim P_z(z)}[\log(1 - D(G(z)))]}$$

where the final expression is the maximum objective value related to the Jensen-Shannen (JS) divergence of two empirical distributions. [1]

A DCGAN is an extension of a GAN that uses convolutional-transpose and convolutional layers in the generator and discriminator, respectively. In a DCGAN setup, "the discriminator is made up of strided convolution layers, batch norm layers, and LeakyReLU activations. The input is a 3x64x64 input image and the output is a scalar probability that the input is from the real data distribution. The generator is comprised of convolutional-transpose layers, batch norm layers, and ReLU activations. The input is a latent vector, $\boldsymbol{z}$, that is drawn from a standard normal distribution and the output is a 3x64x64 RGB image. The strided conv-transpose layers allow the latent vector to be transformed into a volume with the same shape as an image." [1]

Experiment

The implementation began with the data loader, where EMNIST images were pulled into the environment. To avoid model overfitting, I applied data augmentation techniques such as random cropping and horizontal flipping to increase the diversity of input data and improve the generalisability of the discriminator. I also carefully controlled the upsampling and downsampling layers to ensure compatible shapes for convolution and image labelling. Ultimately, this strengthened the generator and helped it learn to produce more realistic images. The following output shows the DCGAN's learning progression over epochs (***Table 1: DCGAN Learning Progression***)

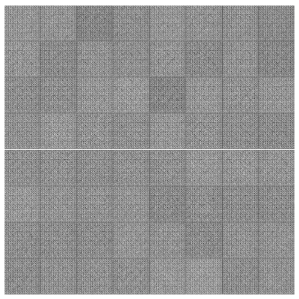
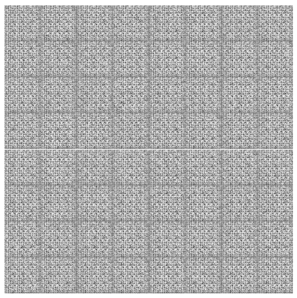
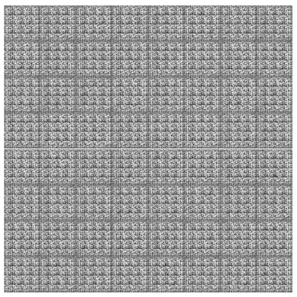
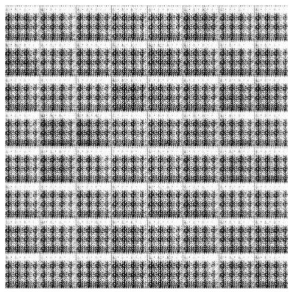
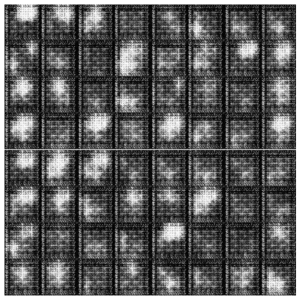
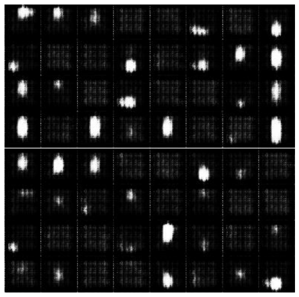
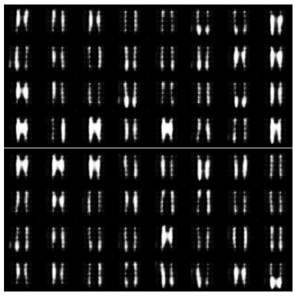
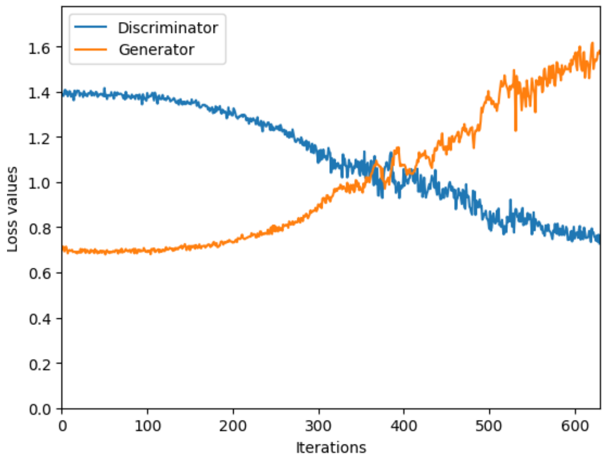
Epoch 1	Epoch 5	Epoch 10	Epoch 15
			
Epoch 20	**Epoch 25**	**Epoch 30**	**MODEL RESULTS**
			Loss_D: 0.7280 Loss_G: 1.5803 D(x): 0.8270 D(G(z)): 0.2155 / 0.2071

Table 1: DCGAN Learning Progression



At first, the discriminator overpowered the generator too quickly. This made it difficult for the generator to learn to produce convincing outputs. In response, I adjusted the learning rates and betas for both networks, which helped balance the competition between G and D, leading to more stable training.

Figure 1: DCGAN G and D Losses over Epochs

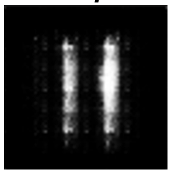
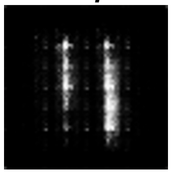
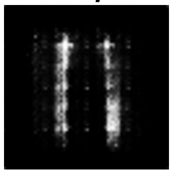
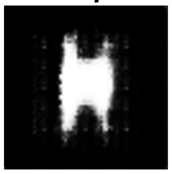
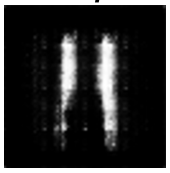
Sample	Sample	Sample	Sample	Sample	Sample
					

Table 2: Generated Image Samples

As is typical of GANs, the model exhibited some instability during training, such as mode collapse. The caused the generator to produce nearly identical outputs. By fine-tuning the learning rate and introducing dropout to counterbalance batch normalisation, I was able to restabilise the model and maintain training progress. I also found that I needed to perturb the real and fake images with noise to increase the model's robustness. Finally, I applied label smoothing to make the discriminator initially less confident in its decisions.

After ~30 epochs, the model stabilised within a desirable range (0.3 > D > 0.7; 1.8 > G > 2.8). The final success of the model is reflected in the it loss plot, as well as the 50 images generated at the end of execution (***Figure 1***: ***DCGAN G and D Losses over Epochs; Table 2: Generated Image Samples***).

Ideally, the model would have included a less arbitrary early-stopping condition, such as model patience dependent on an Frechet Inception Distance (FID) score. The FID score is one of the most significant performance evaluation metrics to image generation. Hence, I chose to include an FID calculation, but needed to limit the evaluation to only the first and final generated images because of the computational cost. Unfortunately, the results of the FID aren't intuitively aligned to the qualitative performance of the model, as the FID score reportedly increases between the first and final image, where the final image displays significantly less noise than the first.

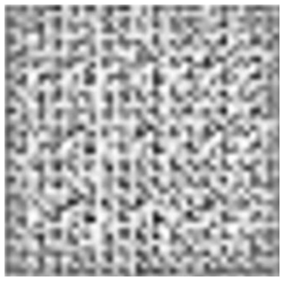
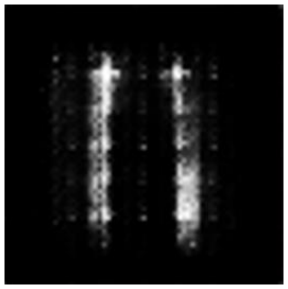
First Image 	**FID for the first generated image:** **369.72**
Final Image 	**FID for the final generated image:** **445.84**

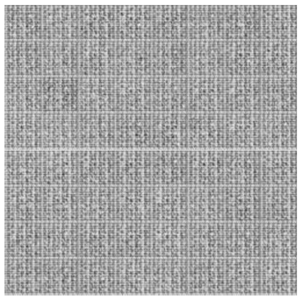
Table 3: DCGAN FID Results Table

In the future, I would expand the investigation to further refine the FID calculation and aim to further improve the model's generative performance.

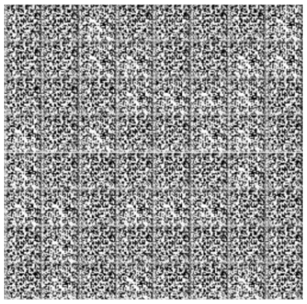
In the Conditional DCGAN implementation I continued building on the existing model, modifying the data loader, generator, and discriminator to handle class labels. In doing so, I embedded the class labels and concatenated them with the noise vector in the generator, ensuring the discriminator processed both the image and label together. This should have resulted in a conditional DCGAN capable of generating images for specific characters (e.g., 'A' and 'B') based on the provided labels.

In practice, the model successfully ran and generated outputs, but a visual assessment of these outputs indicates possible mode collapse, despite the second model having a very similar architecture to the first. The learning progression of the Conditional-DCGAN is displayed below (***Table 4: Conditional-DCGAN Learning Progression***).

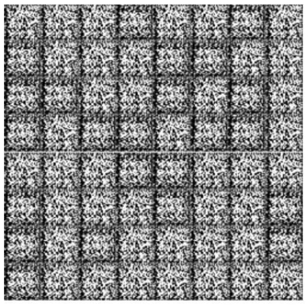
Epoch 1



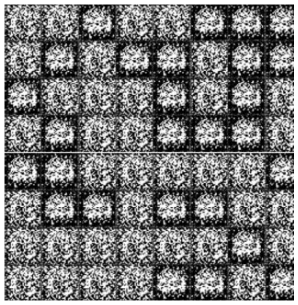
Epoch 5



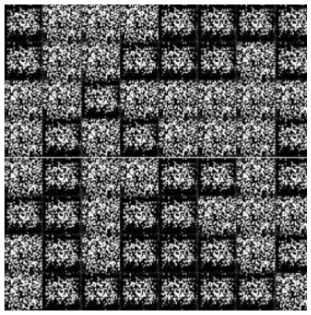
Epoch 10



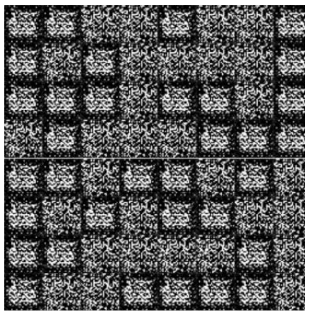
Epoch 15



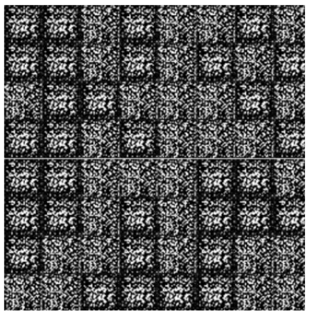
Epoch 20



Epoch 25



Epoch 30



MODEL RESULTS

Loss_D: 0.3347

Loss_G: 1.8130

D(x): 0.1245

D(G(z)): 0.1245 / 0.1684

Table 4: Conditional-DCGAN Learning Progression

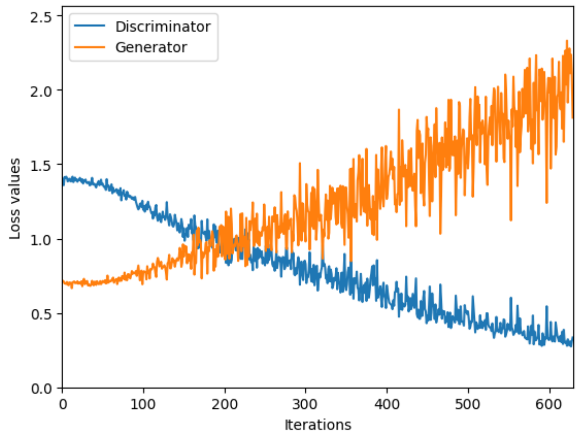


Figure 5: Conditional-DCGAN G and D Losses over Epochs

Further, the generator still demonstrated significant oscillation leading up to the end of execution; this means that it is unlikely that G and D are approaching stabilization/convergence (***Figure 2: Conditional-DCGAN G and D Losses over Epochs***).

It is possible that if the model were allowed to run for longer that the result might be more positive than in the current final epoch, 30. However, without addressing the visible issue of mode collapse through model fine-tuning, it wouldn't be reasonable to expect a significant change in generative quality. Samples of generated outputs can be seen in ***Table 5***, below. As the samples show, the output is noisy, unrefined, and not representative of the desired conditions.

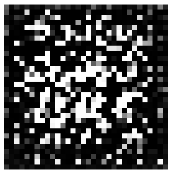
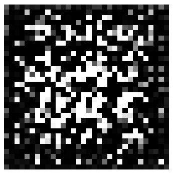
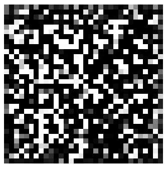
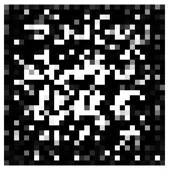
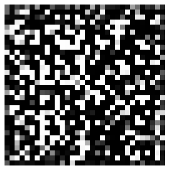
Sample	***Sample***	***Sample***	***Sample***	***Sample***	***Sample***
					

Table 5: Generated Image Samples

As expected, the initial and final FID scores for the Conditional-DCGAN's generated images are very high and close in value (***Table 6: Conditional-DCGAN FID Results Table***). This indicates a large Frechet inception distance (significant discrepancy) between the distributions of the real and synthetic images.

First Image	**FID for the first generated image:** **580.41**
Final Image	**FID for the final generated image:** **582.92**

Table 6: Conditional-DCGAN FID Results Table

In the future, I would expand the investigation by continuing model fine-tuning and perhaps investigate the use of spectral normalisation in place of dropout.

References

[1] PyTorch, “DCGAN Tutorial - PyTorch,” 2024.

Source Code

This is my own work. Supporting code can be found on GitHub and Google Colab.