GAN Implementation to generate plausible Fashion-MNIST images

Unit 7 Assignment: Implementation of GAN

Juliana Noronha

University of Maryland, Global Campus

DATA 655: Deep Learning and Neural Networks

Dr. Jeremy Bolton

March 4, 2025

Introduction

Objective

The objective of this analysis is to implement and evaluate a Generative Adversarial Network (GAN) that generates plausible clothing and accessories, using the Fashion-MNIST dataset. To achieve this, the architecture will be varied in myriad ways and inspected for impacts on the overall model accuracy.

Problem Domain

The Fashion-MNIST (Fashion - Modified National Institute of Standards and Technology) dataset was created by Han Xiao, an engineer at UK fashion outlet Zalando, as a more complex, drop-in replacement for the original MNIST digit dataset. A year after its release, it had already been referenced in over 7000 code snippets and quickly became a new machine learning benchmark dataset. Using Zalando's existing SKU catalog, Xiao transformed the images and data itself to have the same exact dimensions as MNIST in order to easily benchmark between the two. His impetus for creating the new dataset was that the MNIST dataset was too simple, resulting in excellent accuracy despite drastic changes to model architectures -- in other words, MNIST was not a complex enough dataset to return significant learnings (Xiao, 2018).

GANs offer an opportunity to train two models at once, each excelling at different tasks. The generator model should ultimately excel at generating realistic and novel classes based on the training data, and the discriminator model should ultimately excel at classifying real versus fake images. Once trained, each model can be used separately for different tasks. The discriminator can be used for anomaly detection, such as in medical imaging (Han, 2021) and feature extraction or feature learning, since the discriminator learns representations of the data (Radford, 2015). The generator is often the more used model between the two, with the ability to

perform image-to-image translations, text-to-image synthesis, super-resolution, face inpainting, and text-to-speech (GAN variations, n.d.).

Method Rationale

GANs are generative, which immediately makes this model architecture a candidate for the task of generating new, plausible samples based on training data. There are newer generative models that produce better generative results than GANs, such as DALL-E's diffusion-based model (Dhariwal, 2021), but this is outside the scope of this course.

Specifically, the architecture that will be employed to generate plausible Fashion-MNIST images is Deep Convolutional Generative Adversarial Network. DCGAN was introduced in 2015 as a way to combine the strengths of Convolutional Neural Networks (CNNs) with GANs, specifically for image generation. DCGANs differ from GANs in several key ways: all pooling layers are replaced with strided convolutions; batch normalization is used in both the discriminator and the generator; all fully connected layers are removed; ReLU is the only activation function used across generator layers, save for a tanh activated output layer; and finally, LeakyReLU is the only activation function used across discriminator layers. The convolutional layers allow for better spatial hierarchy representation, which is critical for image classification and generation, and the other changes promote model stability and prevent mode collapse (Radford, 2015).

Analysis

Data

The Fashion-MNIST dataset consists of 70,000 grayscale images of different clothing items fitting into ten classes: T-shirt/top, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, Ankle boot (Xiao, 2017). The images were originally taken from Zalando's SKU catalog

(Xiao, 2018). Each image is 28x28 pixels in size, size-normalized and centered. 60,000 images are designated as training data and the remaining 10,000 images are designated as testing data. When imported directly from the tensorflow data repository, the training feature data comes as a three-dimensional 60,000 length tensor of 28x28 arrays, containing values ranging from 0 to 255, where 0 is black and 255 is white. The label data is simply a 60,000 length tensor, containing values ranging from 0 - 9, which map to the class labels mentioned earlier.

Exploratory Analysis

First, the training labels are plotted as a barchart to evaluate whether there are any glaring class imbalances (Figure 1). There are exactly 6000 instances of each class present in the training data, so there is no risk of class imbalance.

Next, the images themselves were inspected. Figure 2 displays the first 10 instances of each of the labels in the dataset. The first 10 instances of each class in the Fashion-MNIST dataset reveal significantly more visual complexity compared to the handwritten digit images from the classic MNIST dataset. While MNIST consists of simple, well-defined numerical shapes, the Fashion MNIST images depict clothing items with intricate details, such as textures, varying silhouettes, and overlapping features. For example, a T-shirt can have subtle shading that suggests folds, while a Sneaker displays a distinct sole and upper structure. All of this detail must also be captured in the same, small 28x28 pixel dimensions as the MNIST digits, which results in ambiguous or blurry looking features. In contrast, MNIST digits are generally uniform in shape, with minimal internal variation beyond differences in handwriting style.

There are also some clear examples of class instances that could easily be confused with one another. For example, many of the short-sleeved items in the “Shirt” category look remarkably similar to the items in the “T-shirt/top” category (except for the tank tops).

Presumably the only difference between these is a collar, which a model may or may not be able to pick up with the low resolution of these images. The short-sleeved Shirts & T-shirts also share visual similarities with the shorter short-sleeved dresses. Additionally, the seventh coat could easily be misconstrued as a dress, and in fact looks very similar to the third and sixth dresses. The low-resolution and inability to indicate relative proportions (e.g. if the images were scaled to a where they might fit on a body) might pose issues for the model downstream.

Preprocessing

The image data has already been centered, normalized and converted to gray-scale. There are no striking class imbalances to handle. The only pre-processing step required here is to scale the pixel values from 0 to 255 to -1 to 1, so that the larger values do not have undue influence on the model. For DCGANs, it is a common choice to scale the values between -1 and 1 instead of 0 and 1, because it is compatible with the tanh-activated output layer, which can produce values between -1 and 1 (Radford, 2015).

Algorithm Intuition

DCGANs leverage the adversarial training framework introduced by Ian Goodfellow and his team but improve it by using convolutional layers instead of fully connected ones, allowing the network to capture spatial hierarchies in images (Radford, 2015). Essentially, two neural networks -- a generator (G) and a discriminator (D) -- “compete” against each other. The generator is a deep convolutional neural network that takes in a random noise vector (latent vector) as input and outputs a series of normalized pixel values from which an image can be reconstructed. The hidden layers of the generator act to increase the spatial dimension of that latent vector to create a high resolution image. The discriminator is also a deep convolutional network, but works in reverse -- taking in an image as input and returning a vector of class

probabilities as output. In the case of the Fashion-MNIST data, the output would be a fully connected, single node layer with a sigmoid activation function to constrain the probability between 0 and 1. Outputs close to 0 indicate that the image is likely a fake, made by the generator, while outputs close to 1 indicate that the image is likely a real member of the dataset. The hidden layers are also convolutional layers, which downsample the data with increasing filters and strides.

When these models are first initialized and untrained, they each produce random outputs. The generator produces noisy, grainy images while the discriminator randomly classifies images as real or fake. Each epoch, the discriminator is first trained followed by the generator. There are several "games" with which one can train a GAN. For the purposes of this paper, the Minimax Game will be used, which formulates the cost functions of each model as follows (Goodfellow, 2014):

$$J^{(D)} = -\frac{1}{2}E_{x\sim p_{data}} logD(x) - \frac{1}{2}E_z log(1 - D(G(z))$$

$$J^{(G)} = -J^{(D)}$$

In the Minimax Game, the generator's loss function is just the mathematical opposite of the discriminator's loss function, effectively giving them opposite training goals -- the generator's goal is for the discriminator to always classify its output as 1, while the discriminator's goal is to always classify the generator's output as 0. The discriminator has the additional task of also classifying real data as 1, which is depicted in the first half of its cost function.

In the first epoch, the discriminator is trained on real images (labeled as 1) and updates its weights via backpropagation. Then, the discriminator is trained on fake images (labeled as 0), which are created with the untrained generator, and the weights are again updated. Finally, the

generator is trained, creating a new batch of images, passing them to the discriminator. Initially, the discriminator easily wins, but over time, the generator improves and produces increasingly plausible and realistic clothing items. The discriminator also improves, learning representations of both real and fake images. Given enough rounds of training, the generator will consistently "fool" the discriminator into classifying fake images as real.

The performance of a DCGAN depends heavily on a few key hyperparameters: latent vector size, feature map depth, batch size and other traditional architectural considerations such as number and types of layers, etc.

Model Fitting

First, model hyperparameters were determined using Keras Tuner with a Random Search strategy. In order to speed up tuning, only a subset of the data (5000 of the training images) was used, with three max trials and one execution per trial.

For this experiment, the number of filters or nodes in each layer were tested as follows:

Layer	**Feature tested**	**Values tested**	**Optimal Hyperparameter**
Dense Layer (7 x 7 x *value)*	# of nodes	128, 256, 512	128
Reshape Layer	Third dimension	128, 256, 512	128
Generator Convolutional Layer 1	# of filters	128, 256, 512	128
Generator Convolutional Layer 2	# of filters	64, 128	128
Generator Convolutional Layer 3	# of filters	32, 64	32
Discriminator Convolutional Layer 1	# of filters	32, 64, 128	64
Discriminator Convolutional Layer 2	# of filters	64, 128, 256	256

Results

Output

Using the hyperparameters achieved during tuning, the generator and discriminator models were trained. Figure 3 displays the generator and discriminator losses over 50 epochs. This loss graph is volatile and only displays slight improvement of the discriminator over time. The volatility may indicate that a smaller learning rate may need to be applied. The generator appears to worsen with each epoch, increasing steadily from 1.25 to about 2.00 between epochs 4 and 15, then continuing to hover between 2.00 and 2.25 until epoch 40, where it comes down very slightly until the end of training. This may mean that the discriminator overpowered the generator during training, preventing the generator from learning. The small loss decrease for the generator and loss increase for the discriminator nearing the end of training may also indicate that longer training times are required, or perhaps that the networks are too shallow to model the complexity in the data.

Figures 4 and 5 display generator outputs at epoch 1 and epoch 50, respectively. The outputs at epoch 1 are mostly noise, but the model has already seemed to learn that there are white masses in the center of the image, surrounded by dark masses. By epoch 50, the generator was able to generate some pretty convincing shoes, shirts and pants to the human eye, producing unintelligible masses some percentage of the time.

Model Properties

The generator and discriminator model summaries can be found in Figures 6 and 7, respectively. All generator models tested had the following basic structure: Dense > Batch Normalization > Leaky ReLU > Reshape > [Convolutional Layer > Batch Normalization > Leaky ReLU] x 2 > Convolutional Layer. The first dense layer projects the latent vector into a

higher dimensional representation, and the reshape layer reshapes it into a three-dimensional volume to prepare it for the convolutional layers. Each convolutional layer progressively upsamples the data, increasing the resolution. The batch normalization layers increase model stability, while the leaky reLU activation functions introduce non-linearity. The output convolutional layer effectively produces a 28 x 28 pixel image, with 1 channel (grayscale).

The discriminator models had this structure: [Convolutional Layer > Leaky ReLU > Dropout (30%)] x 2 > Flatten > Dense (1 node, sigmoid activation). This model takes a 28 x 28 x 1 image and progressively downsamples it to a single number: the probability. The downsampling is specifically achieved by the dropout and convolutional layers.

The final generator model had 11 layers (not including the input layer) and the discriminator model had 8 layers.

Evaluation

Visual inspection is covered in the output section of this paper, but it is worth noting that the increasing losses of the generator function bely some of the impressive image outputs. There were plausible shoes, shirts and pants generated per the human eye.

Moving to more objective scores, Figure 8 shows that the inception score for the test data was 1.057, which is poor. Higher inception scores indicate that the generated images are diverse and high in quality (Ahuja, 2023). This score essentially tests how well the Inception v3 image classification model can classify the generated images. Since the original images themselves are grainy and grayscale, the Frechet Inception Distance (FID) may be a better metric (Figure 9). FID takes the target or training distribution into account, measuring how similar the statistics of generated images are to ground truth images. The FID for this data was also very poor, coming in at about 125.

Finally, the trained discriminator was applied to a set of real and fake images to test its classification accuracy. Figure 10 shows that the discriminator was able to accurately classify real images 30.2% of the time and accurately classified fake images 91.8% of the time. Ideally, both of these values would be close to 50% -- the discriminator should not be so good at identifying fake images. This is another indicator that the generator may need more training to fool the discriminator better.

Conclusion

Summary

This analysis explored the implementation of a DCGAN to generate plausible fashion items using the Fashion-MNIST dataset. The objective was to evaluate the impact of different architectural choices on the model's ability to produce realistic images. Through hyperparameter tuning and model evaluation, the generator improved over time, producing recognizable shoes, shirts, and pants. However, the generator loss steadily increased throughout training, and quantitative evaluations such as Inception Score (1.057) and Frechet Inception Distance (125) indicate that the generated images lack diversity and realism. The discriminator’s high accuracy on fake images (91.8%) and low accuracy on real images (30.2%) further suggest that the discriminator overpowered the generator, preventing effective learning.

Limitations

The loss curves indicate instability, possibly due to an overly strong discriminator or suboptimal hyperparameters. The low resolution (28×28) of Fashion-MNIST also poses a challenge, as finer details that distinguish clothing types may not be adequately represented. Additionally, the limited dataset diversity means that generated images may not generalize well beyond the dataset’s specific styles and silhouettes.

Improvement Areas

Future improvements could focus on enhancing training stability by reducing the discriminator's learning rate, implementing label smoothing, or using techniques like Wasserstein GAN (WGAN-GP) to prevent mode collapse. Lastly, longer training times or architectural refinements (such as increasing generator depth or feature maps) could further improve results.

References

Ahuja, N. (2023, January 11). Inception score(IS) and Fréchet inception distance(FID) explained. Medium. https://ahujaniharika95.medium.com/inception-score-is-and-fr%C3%A9chet-inception-distance-fid-explained-2bc28a4faea7

Dhariwal, P., & Nichol, A. (2021). Diffusion models beat GANs on image synthesis. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2105.05233

GAN variations. (n.d.). Google for Developers. https://developers.google.com/machine-learning/gan/applications

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative adversarial networks. arXiv (Cornell University). https://doi.org/10.48550/arxiv.1406.2661

Han, C., Rundo, L., Murao, K., Noguchi, T., Shimahara, Y., Milacski, Z. Á., Koshino, S., Sala, E., Nakayama, H., & Satoh, S. (2021). MADGAN: unsupervised medical anomaly detection GAN using multiple adjacent brain MRI slice reconstruction. BMC Bioinformatics, 22(S2). https://doi.org/10.1186/s12859-020-03936-1

Radford, A., Metz, L., & Chintala, S. (2015). Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. arXiv (Cornell University). https://doi.org/10.48550/arxiv.1511.06434

Xiao, H., Rasul, K., & Vollgraf, R. (2017). Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. arXiv (Cornell University). https://doi.org/10.48550/arxiv.1708.07747

Xiao, H. (2018, September 28). Fashion-MNIST: Year in Review · Han Xiao Tech Blog - Neural Search & AI Engineering. https://hanxiao.io/2018/09/28/Fashion-MNIST-Year-In-Review/

Appendix

Figure 1: Barchart of Training Labels

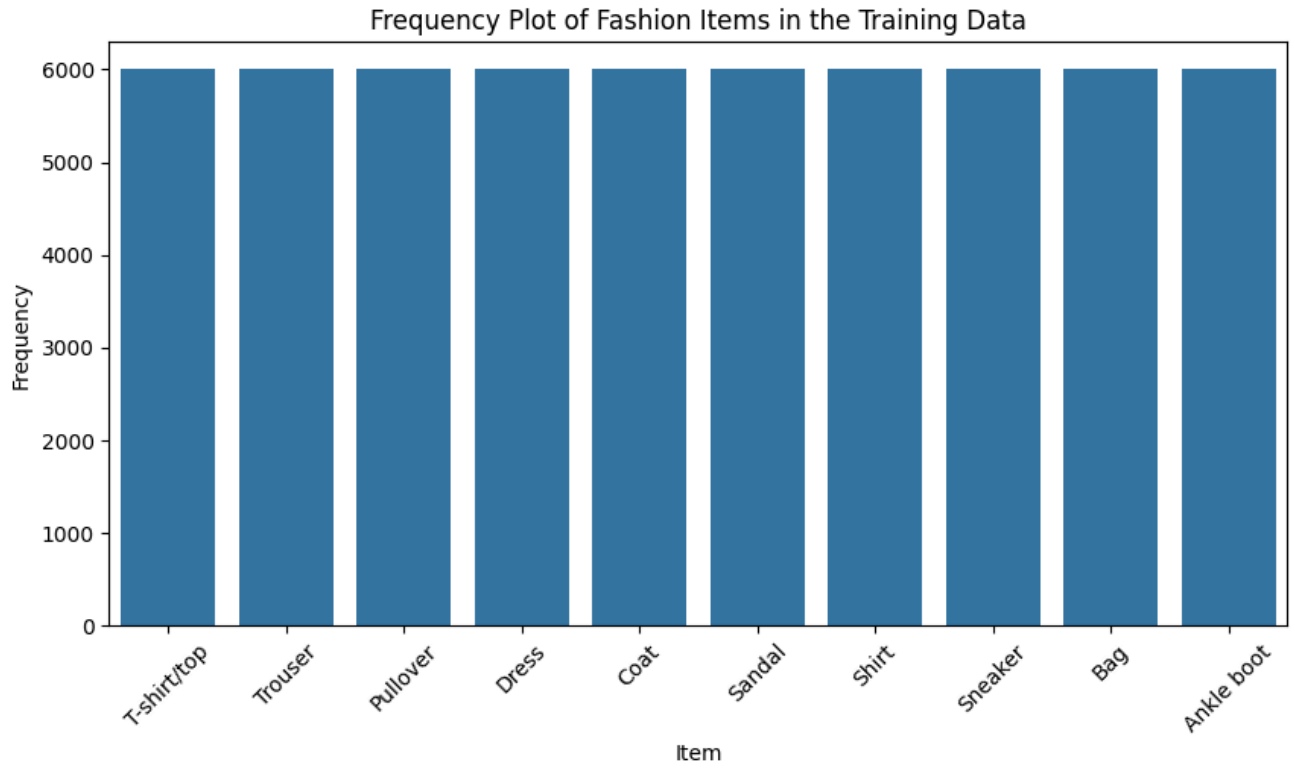

Figure 2: The first 10 instances of each label

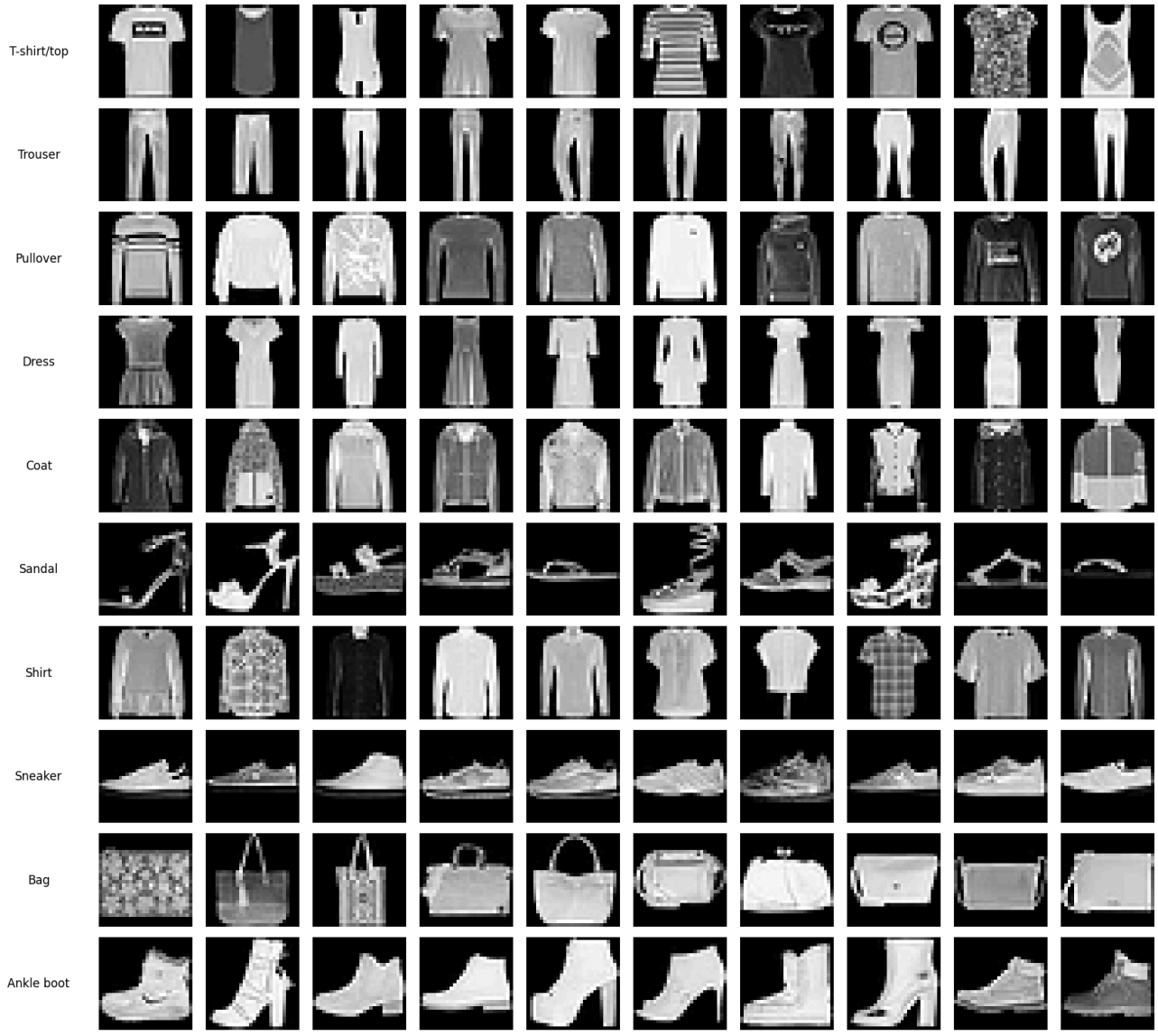

Figure 3: Training Losses by Epoch

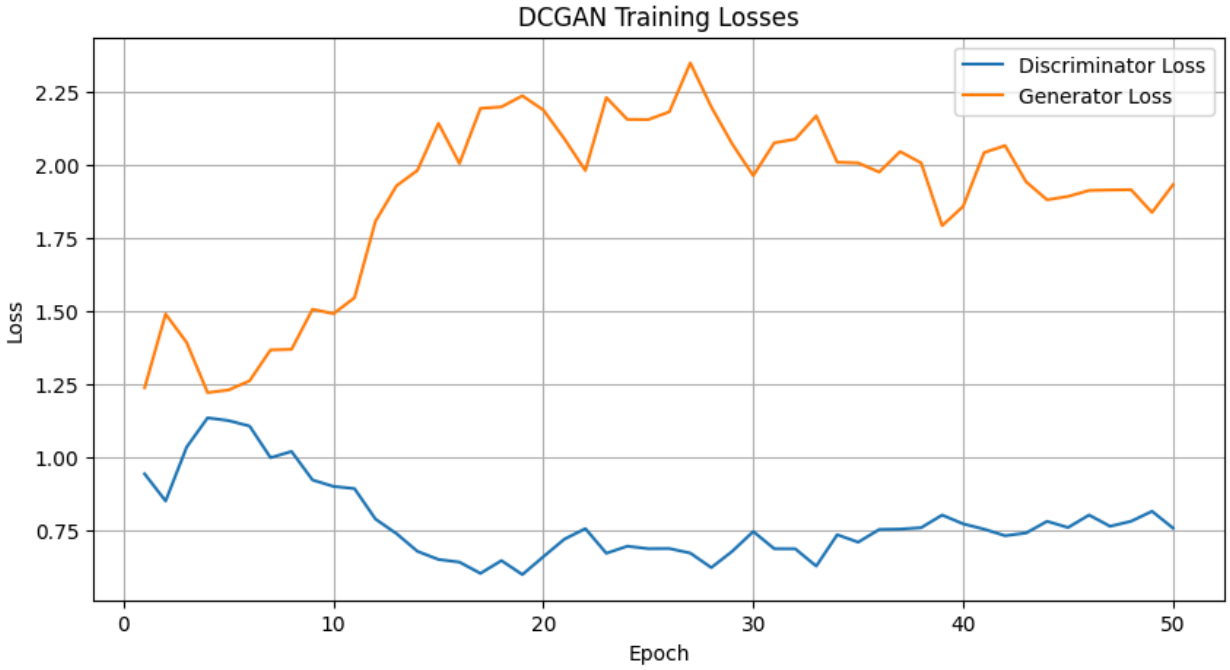

Figure 4: Generator outputs at epoch 1

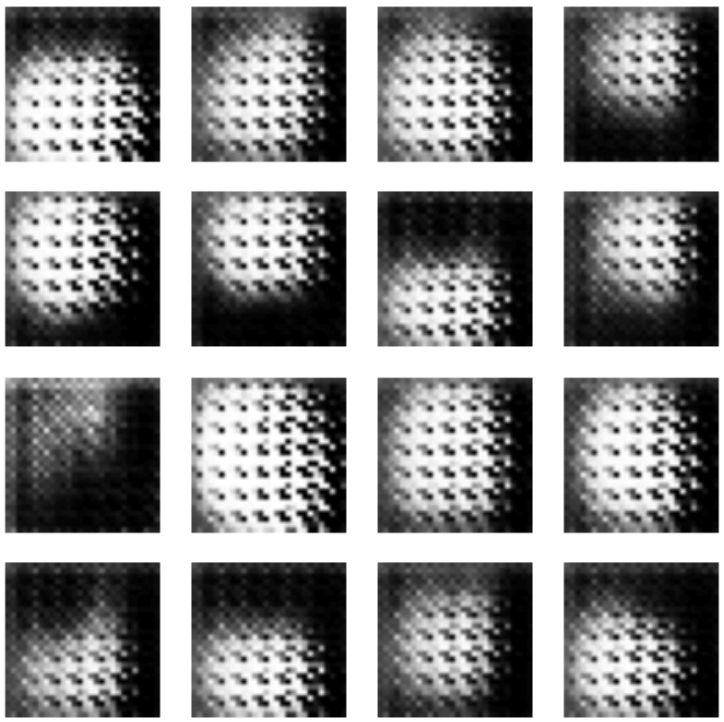

Figure 5: Generator outputs at epoch 50

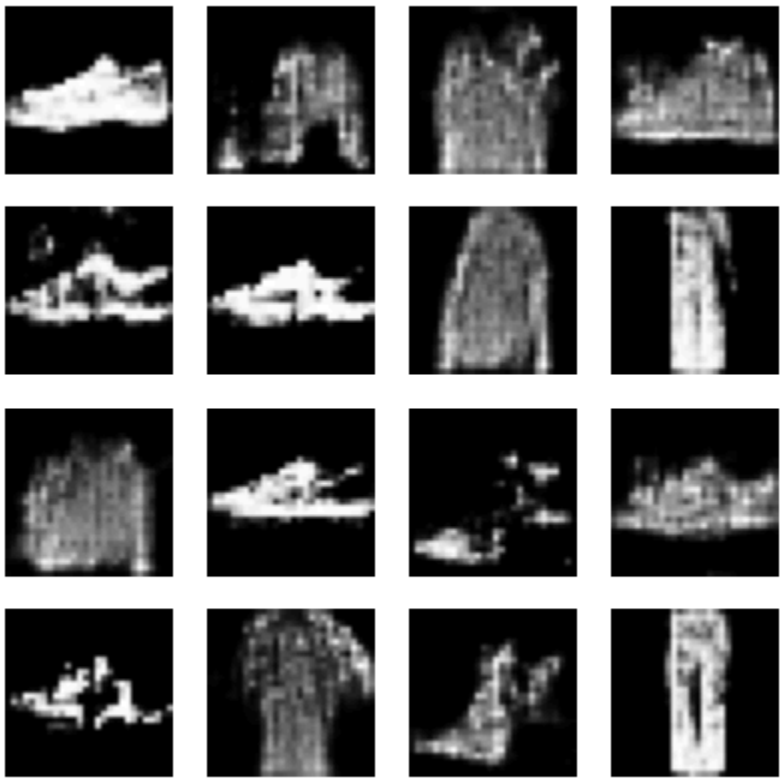

Figure 6: Generator Model Summary

Model: "Generator"

Layer (type)	Output Shape	Param #
dense_2 (Dense)	(None, 6272)	627,200
batch_normalization_3 (BatchNormalization)	(None, 6272)	25,088
leaky_re_lu_5 (LeakyReLU)	(None, 6272)	0
reshape_1 (Reshape)	(None, 7, 7, 128)	0
conv2d_transpose_3 (Conv2DTranspose)	(None, 7, 7, 128)	409,600
batch_normalization_4 (BatchNormalization)	(None, 7, 7, 128)	512
leaky_re_lu_6 (LeakyReLU)	(None, 7, 7, 128)	0
conv2d_transpose_4 (Conv2DTranspose)	(None, 14, 14, 32)	102,400
batch_normalization_5 (BatchNormalization)	(None, 14, 14, 32)	128
leaky_re_lu_7 (LeakyReLU)	(None, 14, 14, 32)	0
conv2d_transpose_5 (Conv2DTranspose)	(None, 28, 28, 1)	800

Total params: 1,165,728 (4.45 MB)
Trainable params: 1,152,864 (4.40 MB)
Non-trainable params: 12,864 (50.25 KB)

Figure 7: Discriminator Model Summary

Model: "Discriminator"

Layer (type)	Output Shape	Param #
conv2d_2 (Conv2D)	(None, 14, 14, 64)	1,664
leaky_re_lu_8 (LeakyReLU)	(None, 14, 14, 64)	0
dropout_2 (Dropout)	(None, 14, 14, 64)	0
conv2d_3 (Conv2D)	(None, 7, 7, 256)	409,856
leaky_re_lu_9 (LeakyReLU)	(None, 7, 7, 256)	0
dropout_3 (Dropout)	(None, 7, 7, 256)	0
flatten_1 (Flatten)	(None, 12544)	0
dense_3 (Dense)	(None, 1)	12,545

Total params: 424,065 (1.62 MB)
Trainable params: 424,065 (1.62 MB)
Non-trainable params: 0 (0.00 B)

Figure 8: Inception Score

```
16/16 ━━━━━━━━━━━━━━━━━━━━ 25s 769ms/step
Inception Score: 1.0573
```

Figure 9: FID

```
Generated images shape: (500, 75, 75, 3)
Test images shape: (500, 75, 75, 3)
16/16 ━━━━━━━━━━━━━━━━━━━━ 154s 490ms/step
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
FID Score: 124.5130
```

Figure 10: Discriminator Accuracy

```
Discriminator Accuracy on Real Images: 30.20%
Discriminator Accuracy on Fake Images: 91.80%
```