# Experimental Series of GANs Models for Generating Handwritten Figures

Ashley M. Schuliger
Department of Computer Science, Worcester Polytechnic Institute, amschuliger@wpi.edu

**Additional Keywords and Phrases:** GANs, MNIST, generator, discriminator

## 1. INTRODUCTION

In the subsequent sections, we perform a series of experiments to find the optimal GANs model for generating handwritten figures from the MNIST dataset. These experiments include adjusting the entire structure of the GANs model, as well as adjusting the hyper parameters, such as the batch size and noise dimension. We provide the justification for each experiment as well as their performance in accurately generating numbers. We then discuss a set of special skills that we utilized in order to boost the overall performance of our model. Once we concluded our experiments, we provide our results and claim that the best GANs model that can be built for this task is a three-level linear neural network with a batch size of 25 and a noise dimension of 64.

## 2. Experiments

In this section, we provide a set of experiments that we performed in order to find the best structure and hyper parameters for our GANs model. We first discuss our experiments with neural networks as well as adjusting the number of neurons at each level and the hyper parameters, and then we discuss a network built with simple linear neurons. For each model in our experiments, we used the Adam optimizer and used BCE loss to measure the loss of our models. For our optimizer, we used a learning rate of 0.0002 as the Adam optimizer performs well with a lower learning rate.

### A. Convolutional Neural Network

Our first model consisted of a series of convolutional neural network layers. Convolutional networks are ideal for extracting information out of images, as they are able to analyze the spatial locality of individual points, unlike linear neuron layers. As a result, we performed a series of experiments with varying neuron layers as well as hyper parameters.

We initially created a control model from which to compare our subsequent experiments to. This model contained four convolutional layers, with output sizes 64, 128, 256, and 1. This structure also utilized batch normalization for optimization of the training process and Leaky Relu as an activation between each pair of layers. At the end of the discriminator model, we applied the Sigmoid activation function to our final output in order to produce our output in terms of 0 for fake images, and 1 for real images. Our generator applied the Tanh activation function to our final output in order to normalize our output to be in between -1 and 1. Figure 1 below shows the resulting image produced by this GANs model, and figure 2 shows the change in error of the generator and the discriminator for each epoch.

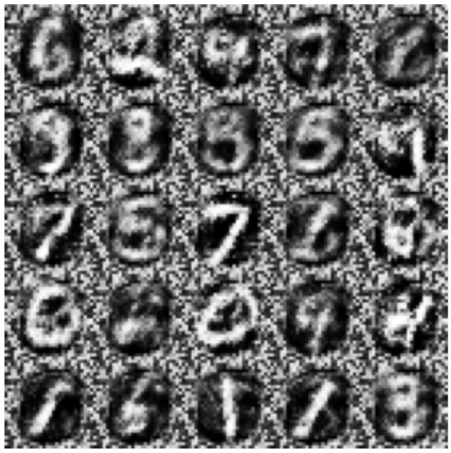

Figure 1: Resulting image from a 4-level CNN.

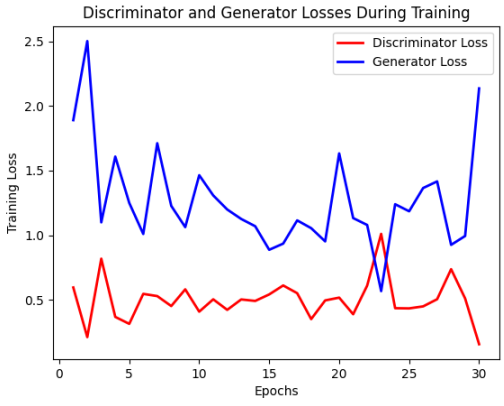


Figure 2: Discriminator loss versus the generator loss for a 4-level CNN.

As shown in the image, this architecture creates fuzzy images of the handwritten numbers, but cannot create distinct, clean images. We also notice that the loss plot shows that the generator was able to “trick” the discriminator around an epoch size of 23 and 28. These points in the plot are important as they produce the best image of handwritten numbers.

From here, we experimented with adjusting the number of layers within the neural network structure. We started by creating a convolutional neural network with 5 layers with input sizes 64, 128, 256, 512, and 1. This was the only change made to the control CNN architecture. Figure 3 below shows the resulting image produced by this 5-level CNN GANs model, and figure 4 shows the change in error of the generator and the discriminator for each epoch.

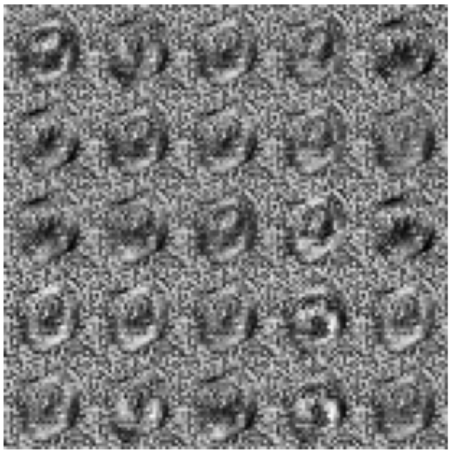

Figure 3: Resulting image from a 5-level CNN.

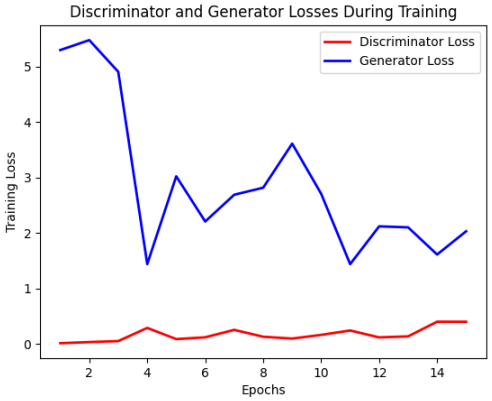


Figure 4: Discriminator loss versus the generator loss for a 5-level CNN.

We also experimented with a 3-level CNN model that used input sizes of 64, 128, and 1. This architecture also utilized Leaky Relu and batch normalization.

Below are the resulting image and loss graphs for this model.

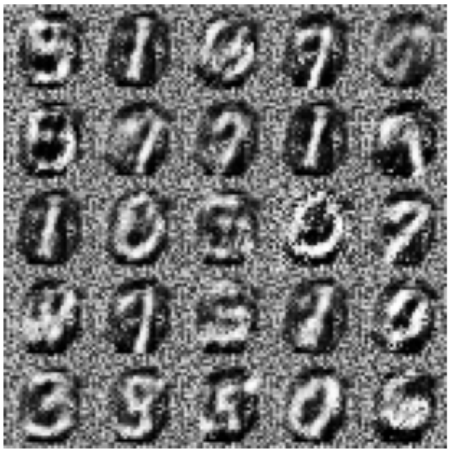

Figure 5: Resulting image from a 3-level CNN.

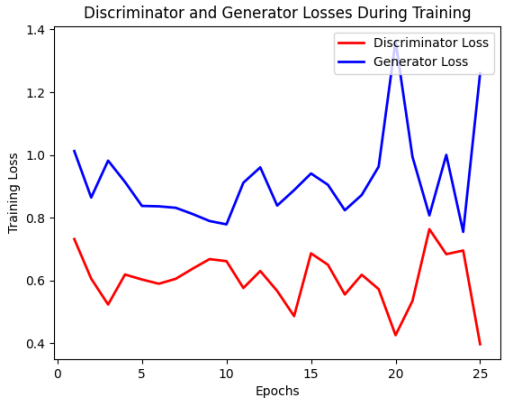


Figure 6: Discriminator loss versus the generator loss for a 3-level CNN.

As shown above, the 5-level CNN performs poorly and is unable to produce handwritten numbers. However, the 3-level CNN performs similarly to the 4-level CNN. Both CNN structures produce blurry images of the handwritten figures, rather than clear ones. As a result, we performed a series of experiments in adjusting the hyper parameters of both CNN architectures.

Before adjusting the hyper parameters, we did one final experiment with the values that we used as the output sizes for the four-level CNN. We were curious to see whether a higher output size or a lower input size would make a difference in our output image. Figures 7 and 8 below show the resulting image and loss graph for the CNN respectively with higher output

sizes of 128, 256, and 512. In contrast, figures 9 and 10 show the resulting image and loss graph for the CNN respectively with lower output sizes of 32, 64, and 128.

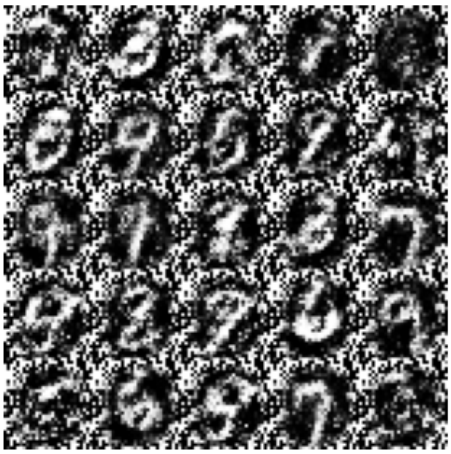

Figure 7: Resulting image from a 4-level CNN with higher output sizes.

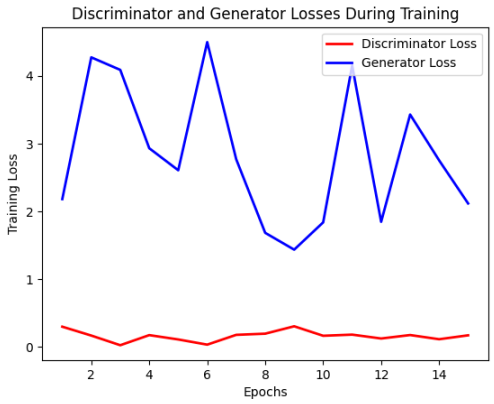


Figure 8: Discriminator loss versus the generator loss for a 4-level CNN with higher output sizes.

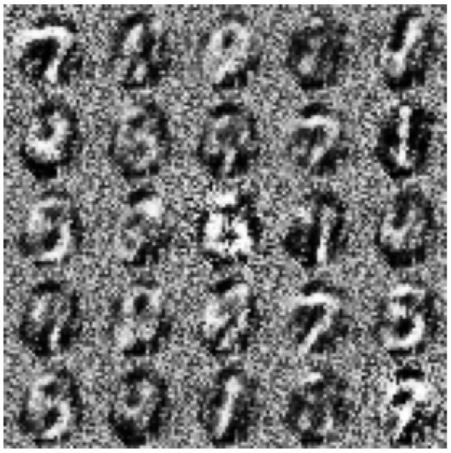

Figure 9: Resulting image from a 4-level CNN with lower output sizes.

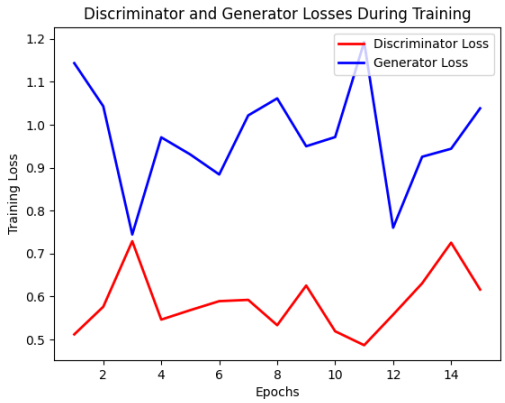


Figure 10: Discriminator loss versus the generator loss for a 4-level CNN with lower output sizes.

The resulting images show that adjusting the output sizes up will result in a model that is sharper; however, this sharper image loses the ability to accurately recreate the handwritten figures. On the other hand, decreasing the output sizes led to a blurrier image. Thus, moving forward, we will use the output sizes as specified by the control model.

### I. Batch Size

In terms of adjusting hyper parameters, we started by adjusting the batch size that we used to train our model. In our control model, we used a batch size of 25, as this was the smallest batch size we could use according to our expected output (i.e. 25 generated handwritten images). From here, we trained both our 4-level and 3-level CNN models with a batch size of 128. The resulting image and loss graph are shown in figures 11 and 12 below.

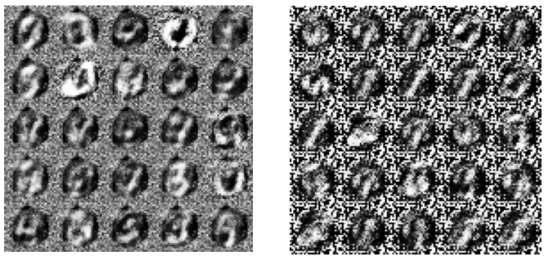

Figure 11: (left) Resulting image from a 3-level CNN with a batch size of 128. (right) Resulting image from a 4-level CNN with a batch size of 128.

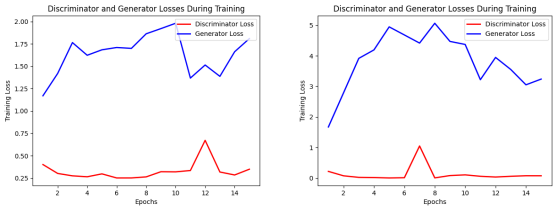


Figure 12: (left) Discriminator loss versus the generator loss for a 3-level CNN with a batch size of 128. (rightt) Discriminator loss versus the generator loss for a 4-level CNN with a batch size of 128.

Both resulting images produce blurry generated images of the handwritten figures. This is logical because as the batch size increases, the discriminator is given a large number of images to train one, which improves its performance tremendously. As a result, the discriminator is able to accurately label the generated images as fake, which does not give the generator a chance to train and learn to produce more real images. In this sense, a higher batch size will overpower the generator. Figure # above portrays this nicely, as it is clear that the discriminator has a steady excellent performance as compared to the poor performance of the generator. As a result, we chose to stick with a batch size of 25.

## II. Noise Dimension

We also experimented with the noise dimension of our random noise generated. Our control model utilized a noise dimension of 64 initially. We chose to experiment with a noise dimension of 256 for our three-level CNN structure. Figure 13 below shows the resulting image, and figure 14 shows the loss graph for both the generator and discriminator.

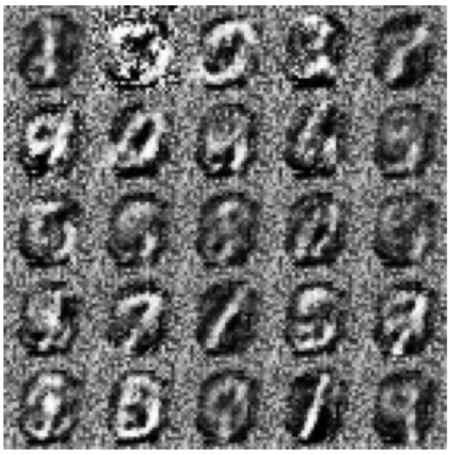

Figure 13: Resulting image from a 3-level CNN with a noise dimension of 256.

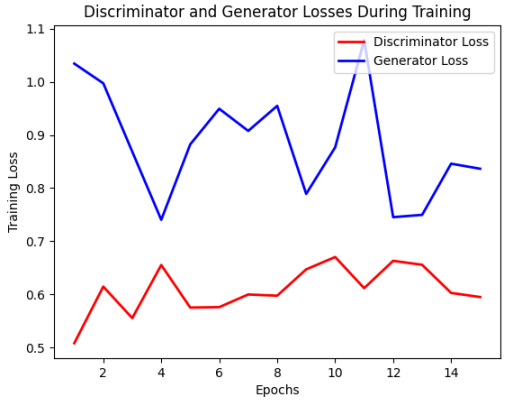


Figure 14: Discriminator loss versus the generator loss for a 3-level CNN with a noise dimension of 256.

The resulting image in this experiment looks similar to that of the three-level CNN with a noise dimension of 64. This led us to believe that the noise dimension would not affect the performance of our GANs model in this case.

In general, these CNNs struggled to create defined images of the handwritten numbers regardless of the hyper parameters being adjusted. As a result, we chose to abandon the CNN architecture in favor of a simple linear neural network.

## B. Simple Linear Neural Network

We also experimented with building a GANs model using a simple linear neural network as a result of our poorly performing CNN structure. We started by creating a control model from which to base the rest of our models off of. This control model contains four neuron levels with output sizes of 128, 64, 10, and 1. Between each linear layer, we perform the Leaky Relu

activation function. We also use the Sigmoid activation function as the last function for our discriminator and the Tanh activation function for our Generator, which is the same structure as our CNN in the previous section. Figure 15 below shows the image produced by our control model, and figure 16 shows the loss of both the generator and the discriminator models.

Figure 15: Resulting image from a 4-level linear neural network

.

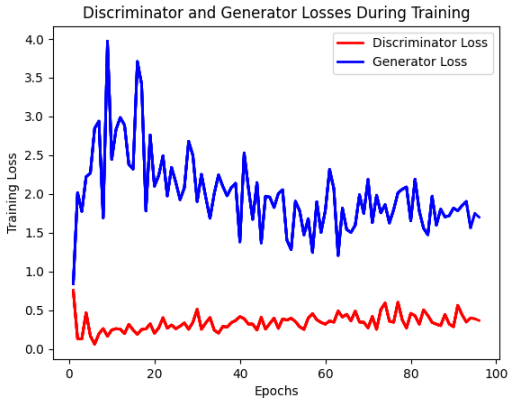


Figure 16: Discriminator loss versus the generator loss for a 4-level linear neural network.

Immediately, we notice that the linear neural network performs incredibly better than the CNN's in the previous section. Although some of the images are still illegible, some of the figures are sharp and represent real numbers. This model shows promise, so we ran multiple experiments on this structure.

Before experimenting with hyper parameters, we chose to experiment with removing a linear layer from our control model. This three-level simple linear neural network utilized output sizes of 128, 64, and 10. The other hyper parameters stayed constant between the control model and this new model. Figure 17 below shows the resulting image from using this model, and figure 18 shows the resulting loss graph of both the generator and discriminator.

Figure 17: Resulting image from a 3-level linear neural network.

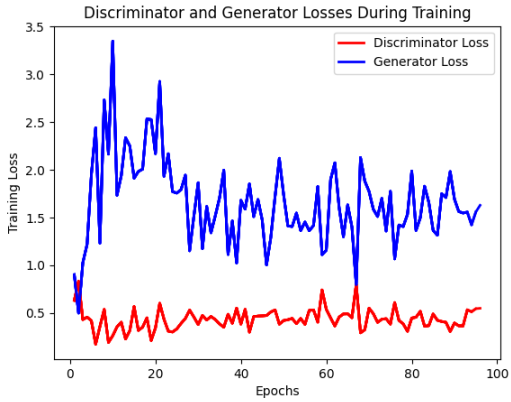


Figure 18: Discriminator loss versus the generator loss for a 3-level linear neural network.

At first glance, this model appears to perform similarly to the control model. However, this new model not only contains fewer blurry figures, but it also contains at least one image of each handwritten figure. Since the goal of these experiments was to find a model that is able to accurately generate all figures clearly, this three-level linear model outperforms our CNN models and our control model. Moving forward, this model will be the one that the other experiments will compete with.

I. **Batch Size**

We started adjusting our hyper parameters by experimenting with the batch size that we used to

train our model. However, we did not expect to see outstanding results from an increased batch size due to our results with the CNN models. In our control model, we used a batch size of 25, similarly to the CNN models in the previous section. From here, we trained our four-level linear neural network with batch sizes of 128 and 256. The resulting images are shown in figure 19 below.

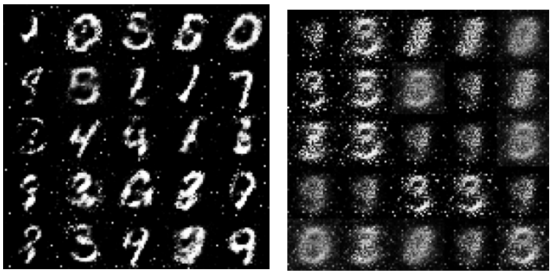

Figure 19: (left) Resulting image from a 4-level linear neural network with a batch size of 128. (right) Resulting image from a 3-level linear neural network with a batch size of 256.

As we expected, an increase in batch size decreased the performance of our GANs model dramatically. This is even clearer by looking at the significant performance decrease between a batch size of 128 and a batch size of 256. As a result, our subsequent experiments worked with a linear neural network with a batch size of 25.

### II. Noise Dimension

We also experimented with the noise dimension of our random noise generated. Based on the performance of our CNN models, we expected the noise dimension to not have a large impact on the performance of our model. Our control model utilized a noise dimension of 64 initially. We chose to experiment with a noise dimension of 256 for both our four-level and three-level CNN structure. Figure 20 below shows the resulting image, and figure 21 shows the loss graph for both the generator and discriminator.

Figure 20: (left) Resulting image from a 3-level linear neural network with a noise dimension of 256. (right) Resulting image from a 4-level linear neural network with a noise dimension of 256.

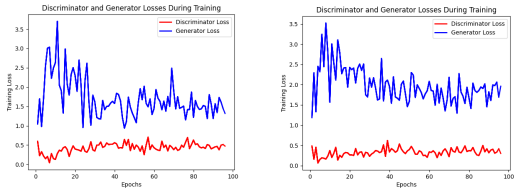


Figure 21: (left) Discriminator loss versus the generator loss for a 3-level linear neural network with a noise dimension of 256. (rightt) Discriminator loss versus the generator loss for a 4-level linear neural network with a noise dimension of 256.

The resulting images closely match the performance or potentially, perform slightly worse than the control models. As a result, we chose to stick with a noise dimension of 64 for our final model.

### C. Number of Epochs

One important observation we made across all of our experiments in both architectures is that the performance of the GANs model increased as the number of epochs used increased. This is logical as the GANs model is given more chances to perfect itself and train on the given images. While our images became clearer as the number of epochs increased, we never surpassed 100 epochs, so we cannot testify to the performance of the models for a larger number of epochs. We did, however, notice that in some of our models, the GANs model would do an excellent job of producing one number multiple times. Figure 22 below shows the resulting image of a four-level linear structure at epoch 96 that may have been overfit.

Figure 22: Resulting image from an overfit 4-level linear neural network at epoch 96.

The resulting image contains 12 figures that represent the number 1, indicating that the model may be overfitting on the number one. As a result, we

conclude that as the number of epochs increases, there may be a higher chance of overfitting our GANs model. However, more experiments need to be performed in order to firmly conclude this.

## 3. SPECIAL SKILLS

In terms of special skills that we used to improve performance, we started by choosing our optimizer to be Adam. This optimizer performs particularly well for GANs models, so we were able to increase our performance by using Adam. Along with this, we performed batch normalization on our CNN models in order to construct mini-batches of the same label. This additional normalization step improved our CNN and allowed it to somewhat generate figures. We were unable to use it for our linear neural network, so in the future, batch normalization may help to make our final model generate clear accurate images consistently. Along with this, we also avoided the use of sparse gradients in favor of Leaky Relu. Sparse gradients have been known to poorly affect the performance of GANs models, so we used Leaky Relu in order to boost performance.

## 4. EVALUATION & RESULTS

As a result of the previously mentioned series of experiments, we concluded that the best model to our knowledge could be built using a three-level simple neural network with a batch size of 25 and a noise dimension of 64. Figure 23 below shows the resulting image, and figure 24 loss graph of our final model.

Figure 23: Resulting image from our final 3-level linear neural network.

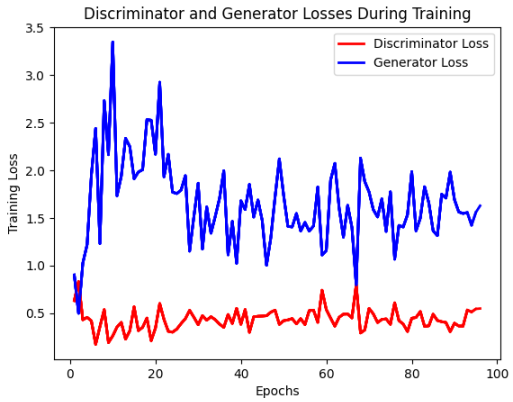


Figure 24: Discriminator loss versus the generator loss for our final 3-level linear neural network.

As shown above our model is able to accurately generate each of the given numbers 0-9. Our model is also well fit since it does not overly produce one specific figure. However, some of the given images still do not resemble digits. Due to this, it is clear that more adjustment can be made to improve this model, such as trying a completely different structure, adjusting the parameters for our optimizer, and adjusting the activation functions used.

## 5. CONCLUSION

In this paper, we proposed a GANs using simple linear neural network layers to generate handwritten figures from a training set of figures. Our structure consisted of three linear neural network levels with a batch size of 25 and a noise dimension of 64. Training our model using BCE loss and the Adam optimizer allowed us to find the optimal weights for our model to increase our accuracy. We also completed a series of experiments to justify our model and analyze the effect of certain hyper parameters on the performance of our model. We also discussed a set of special skills that we used in order to increase the performance of our model. The proposed GANs model produces accurate images of the handwritten figures. In the future, we plan to perform more experiments to increase the consistency at which our model produces legible and clear figures.