Machine Learning - Homework 2

Tintrim Dwi Ary Widhianingsih
20185116

December 4, 2018

Problem 8. Experiment of Convolutional Neural Network (CNN) for fashion MNIST dataset[1].

Fashion MNIST consists of 60,000 training and 10,000 validation images. Each example is a 28×28 grayscale image, associated with a label from 10 classes: T-shirt/top, trouser, pullover, dress, coat, sandal, shirt, sneaker, bag, and ankle boot.

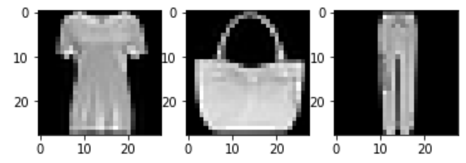
CNN is the one of the developed method that has powerful capability in object processing. In this report, CNN is implemented to fashion MNIST database. The experiment is conducted by using several configurations. By default, the number of convolutional and pooling layers respectively are 3 layers, which are using 32, 64, and 128 filters with the size of 3×3 in each convolutional layer and using 2×2 size of max-pooling window. The stride that is used in both layers is 1. The activation function, by default using ReLU, is activated during convolutional and classification process. In order to optimize the process in fully-connected layers, neural network with 2 hidden layer is used. The initializer and optimizer applied in this experiment are He (from normal distribution) and Adam. To decrease the number of the nodes when passing input layer, dropout penalization is implemented using 0.3 as the threshold. Finally, the activation function for getting probability of each category is calculated using Softmax.

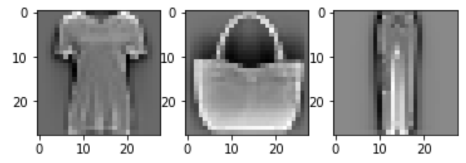
For the discussion, the default setting is change by some scenarios, that differing the initializer, filter, optimizer, activator, and regularizer. The model is trained using 50 epochs and using 60,000 training and 10,000 validation dataset. The following sub-section, each different scenario result is

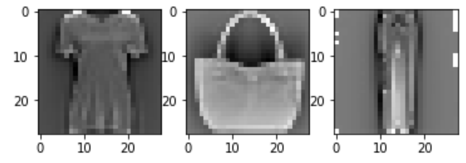
[1]https://arxiv.org/abs/1708.07747

shown and explained. In the graph that is shown below, for the epochs axis, 0 is equal to the first epoch.

a) Performing "Mean Subtraction" and "Normalization" on the input data.


(a) Original


(b) Mean Subtraction


(c) Z Normalization

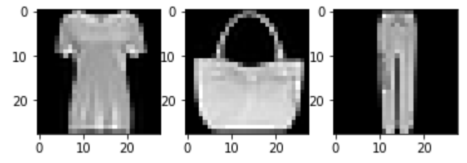

(d) Grayscale Normalization

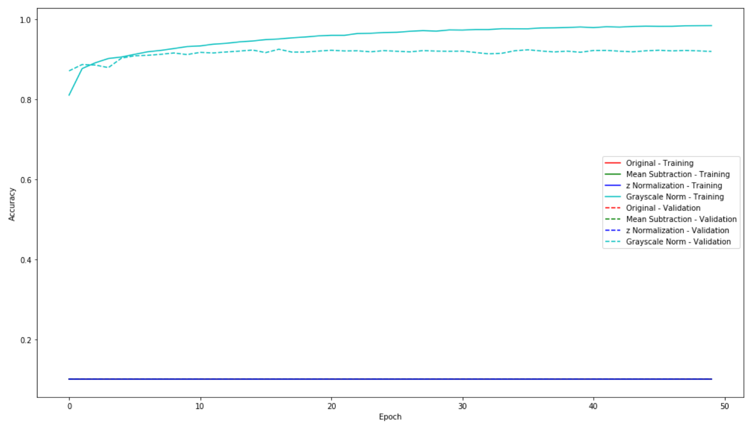
Figure 1: Visualization of 3 Categories: T-Shirt/Top, Bag, and Trouser

In the "Mean Subtraction" part, the mean value that is used is the mean of each category. The mean value of each pixel $i \times j$ for class c is calculated by the following equation:

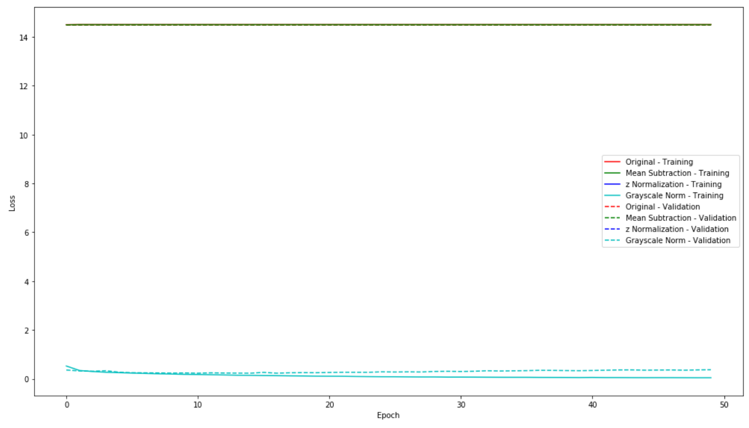
$$\mu_{i,j,c} = \frac{1}{n_{training}} \sum_{i=1}^{28} \sum_{j=1}^{28} x_{i,j,c}$$

where $x_{i,j,c}$ is the grayscale intensity value of the i-th and j-th pixel and class c, while $i = j = 1, 2, ..., 28$ and $c = 0, 1, ..., 9$. The number of training dataset is denoted as $n_{training}$.

Then, the normalization based on the mean subtraction that is applied in this part is z-normalization, so that for every pixel $i \times j$ has a mean ($\mu_{i,j,c}$) 0 and standard deviation ($\sigma_{i,j,c}$) 1. The mean and standard deviation value that is used is based on each class of the images respectively. For the validation dataset, the images are normalized by the mean and standard deviation of training dataset.



(a) Accuracy



(b) Loss

Figure 2: Experiment Result for Different Normalization Scenario

Another normalization method that is applied to the original images is the intensity normalization or called as ”grayscale normalization” in this report. The grayscale intensity value of each pixel is divided by 255, which is the maximum value of grayscale intensity. The normalized value using this method has range of $0 - 1$.

In Fig.1, There are shown 3 example results of each normalization. The shown images is the 10th, 100th, and 1000th instances, which are respectively from category: T-shirt, bag, and trouser. We can see in Fig.1b and 1c that the images from mean subtraction and z

Table 1: Time Consuming of Normalization Scenarios

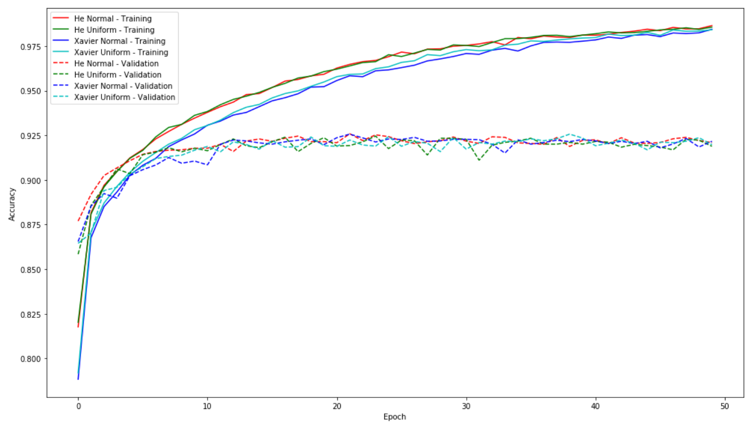
Normalization Method	**Time (s)**
Original	2107.37
Mean Subtraction	2217.47
z Normalization	2126.02
Grayscale Normalization	2169.97

normalization have softer images. The images are mostly gray but the edge is still detected. However, the grayscale normalization, see Fig.1d, obtains the closest images to the original images among the others.

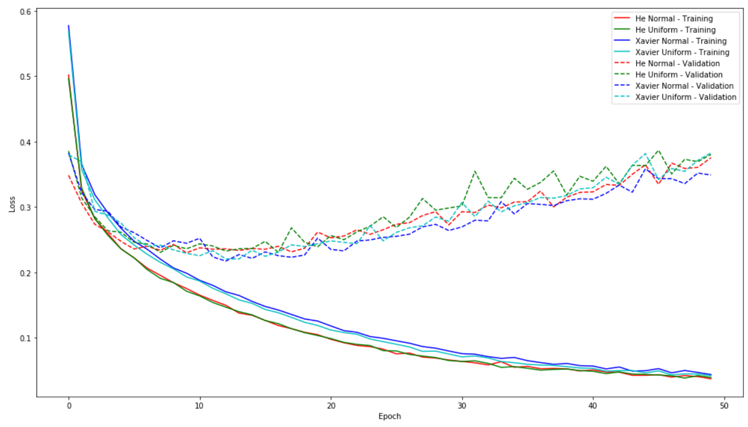
Now, the experiment of CNN is done by using original images and 3 types of normalization. The time consuming of this experiment is shown in Table 1. Relying on Fig.2a, we can see that there is a big different of the lines in the figure. All the images we use as the input obtain 0.1 of the model performance in all epoch in both training and validation dataset, except using grayscale normalization. For the original images, it is expected because of the range value of the intensity inside the images is too big, which is $0-255$, so it makes the model to be under-fit. For the mean subtraction and z normalization, it is presumed that the bad result is caused by the "soft image", which means that the there is no big difference between the intensity value inside the object and its background. We can see in Fig.1b and 1c, the grey color fills inside and outside the objects. In this case, it can make the edge of the object not rigorously detected. Hence, in the next analysis, images with gray-scale normalization are used as the default input for CNN.

b) Performing the result of using Xavier and He initialization.

In this scenario, the setting is to show and compare the performance of different initializers, which are He and Xavier. In Fig.3a for the training graph, we can see that using the default setting, He initialization outperforms Xavier for both normal and uniform distribution. In the other hand, using more epochs the performance of both initialization method is being closer and closer and always increasing until reaching accuracy level above 98%. However, the result in validation set of He and Xavier is not really different.



(a) Accuracy

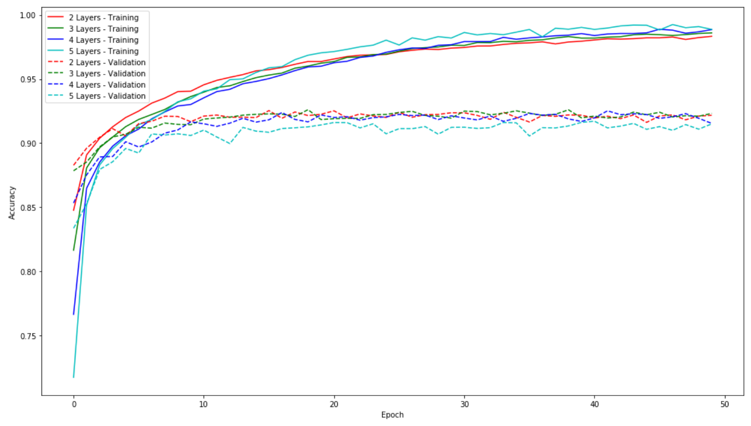


(b) Loss

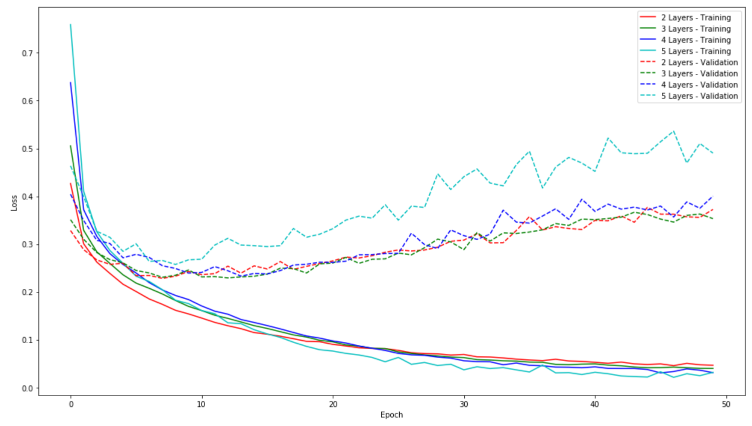
Figure 3: Experiment Result for Different Initialization Scenario

Comparing model performance in training and validation dataset based on Fig.3a, we can analyze the tendency of those method to overfit. Focusing on the results of He initialization, the green and red lines, we can see that the overfitting starts from epoch 5. In the other hand, the results of Xavier initialization, which are blue and cyan lines, start to overfit after epoch 8. It means that even if He initialization obtains better performance in training set, it has the faster overfitting model and the bigger overfitting value (in this case defined as the difference between training and validation performance). However, the performance of He initialization before epoch 5 has the better value in both training and validation dataset.

c) The result of the experiment using different setting of layers and hidden nodes.



(a) Accuracy



(b) Loss

Figure 4: Experiment Result for Different Configuration Scenario

In this scenario, we want to know how the different configuration influence the performance of the CNN model. The configuration setting that is implemented in this experiment are using different number of convolutional layers, in this case using 2, 3, 4, and 5 layers. In the first configuration, we use 32 and 64 filters. The second configuration uses

Table 2: Time Consuming of Initialization Scenarios

Optimization Method	**Time (s)**
He (normal)	2067.04
He (uniform)	2014.57
Xavier (normal)	2098.95
Xavier (uniform)	2212.42

32, 64, and 128 filters. The third one uses 32, 64, 128, and 256 filters. And the last configuration uses 32, 64, 128, 256, and 512 filters.

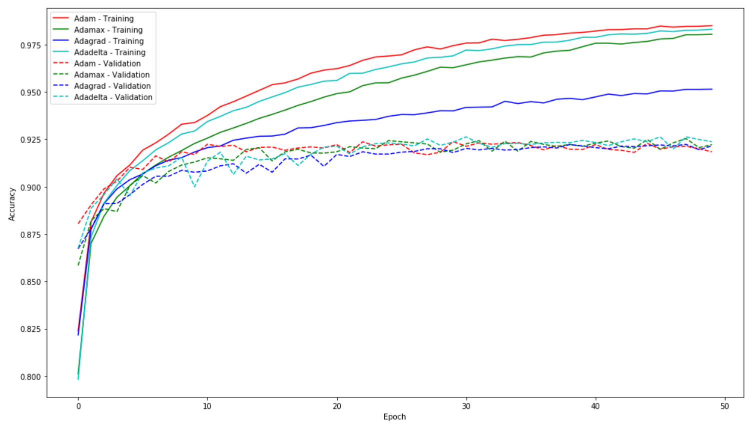
We can see in Fig.4a that in the beginning, using the less layers, the performance of the model is better. In this case, 2-layers is better than 3, 4, and 5-layers, 3-layers is better than 4 and 5-layers, and 4-layers is better that 5-layers. However, together with the increasing number of the epoch, the performance of the model with 5-layers is being higher and higher. In the middle, its performance pass the performance of other configurations. Since that, in the last epoch of the experiments, the model with 5-convolutional layers obtains the best performance, followed by 4-layers, 3-layers, and 2-layers.

Furthermore, when we compare the performance in training and validation dataset, in Fig.4a, we can see that all the configuration scenarios obtain the over-fitting model if we using the big number of epochs. However, among the models that are implemented, 5-layers model obtains the biggest different performance between training and the validation dataset, which is can be seen from the biggest performance in training while obtaining the worst performance in the validation dataset. In this analysis, we can say that instead of having the simpler architecture, using $2-4$ layers in the model is better.

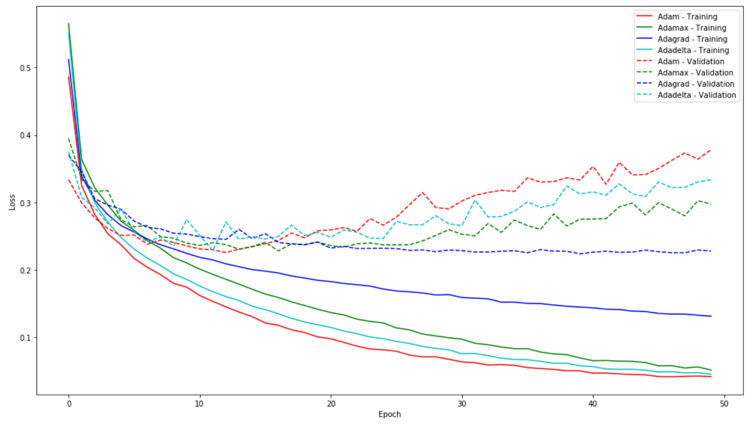
The time consuming of the initialization scenarios is shown in Table 2.

d) The result of the experiment using different gradient optimization techniques.

In this scenario, we compare different type of the optimizer that is implemented to CNN model. The optimizers that is used in the experiment are Adam, Adamax, Adagrad, and Adadelta. Using those different type of optimizers, we can see in Fig.5a that Adagrad obtains



(a) Accuracy



(b) Loss

Figure 5: Experiment Result for Different Optimization Scenario

the worse performance in training dataset compared to the others. The best performance is consistently obtained by Adam optimizer.

For the validation dataset, in the beginning Adam obtain the best performance. The difference of the performance among all optimizers is being smaller and smaller together with the increasing number of epoch. It can be seen when the model reaching 30 epochs, the model performance in validation dataset is convergence to a particular value of accuracy. It means that using different optimizer, the result is same as long as we use enough number of epochs.

Table 3: Time Consuming of Optimization Scenarios

Optimization Method	**Time (s)**
Adam	2109.27
Adamax	2179.43
Adagrad	2172.76
Adadelta	2049.26

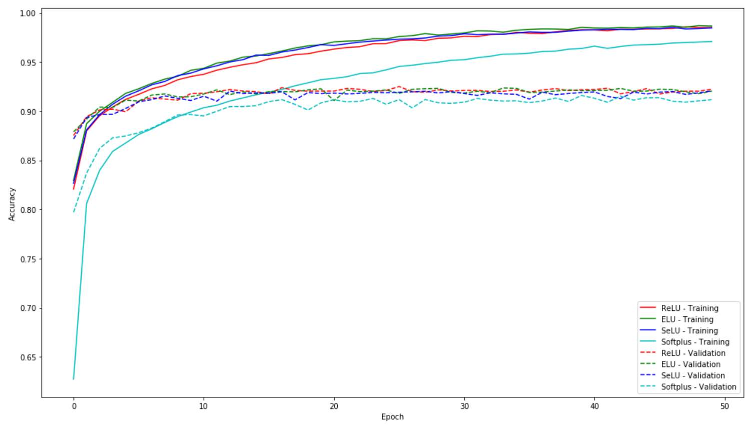
Furthermore, if we compare the result in training dataset to the result in validation dataset, we can see that all of the optimizers obtain the over-fitting model. If we see in the graph, the optimizer that has big different value in between training and validation dataset is Adam. In contrast, Adagrad has the small difference of it. Hence, relying on how big the difference of the model performance in between training and validation dataset in this experiment, we can conclude that Adagrad is the best optimization method.

The time consuming of the four optimization method is shown in Table 3.

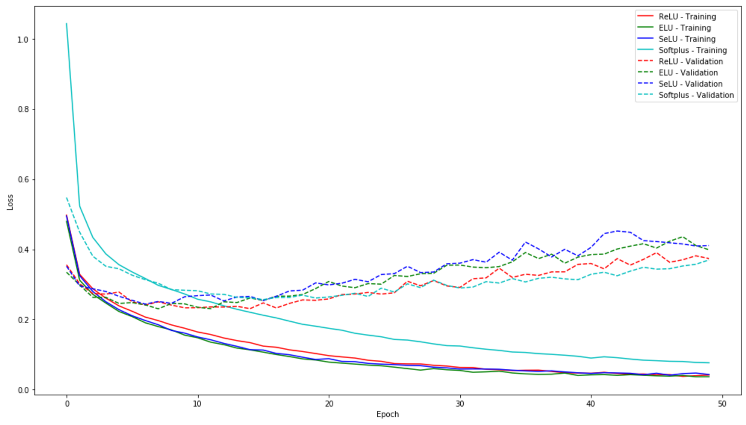
e) The result of the experiment using different activation function

In this section, we want to see the effect of different activation function. Three activation functions that we use in this experiment is ReLU, ELU, SeLU and Softplus. From Fig.6a, we can see that Softplus obtains the worst performance among the others in both training and validation datasets. In the beginning of the epoch, the difference of the performance between Softplus and the others is so big, but together with the increasing of the epochs, its performance is getting higher and higher. However, it is still below others performance.

Among ReLU, ELU, and SeLU in Fig.6a, we can see that in the training dataset ReLU has the smallest performance until 30 epochs. After that, it becomes closer to the others. For the performance in the validation dataset, the three activation functions obtain the same convergence value of accuracy. Then if we see at Table 4, we know that ReLU has the fastest time to run. Although this method has the lowest performance in training dataset, it has the same performance in the validation set. So in this experiment, we can say that ReLU is the best alternative to use among the other activation functions.



(a) Accuracy



(b) Loss

Figure 6: Experiment Result for Different Activation Scenario

f) The result of the experiment using different type of regularization method

   In this implementation, we want to know how the different type of regularization influence the performance of CNN. We use L1, L2, and Dropout regularization. For L1 and L2, we use 0.0001 as the regularization parameter and for dropout, we use 0.3 as the treshold.

   From the experiment result in Fig.7a, we can see that in the validation dataset Dropout has the best performance. This can be inferred that using this configuration (particular parameter value of L1 and L2),

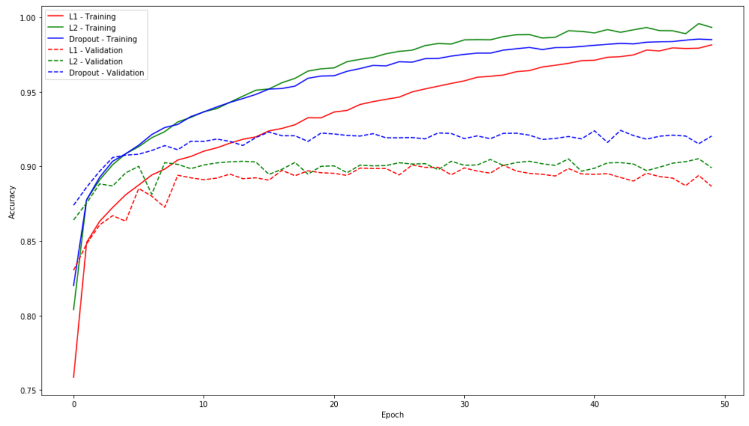
Table 4: Time Consuming of Different Activation Function Scenarios

Activation Function	**Time (s)**
ReLU	2152.01
ELU	2361.55
SeLU	3339.49
Softplus	14339.59

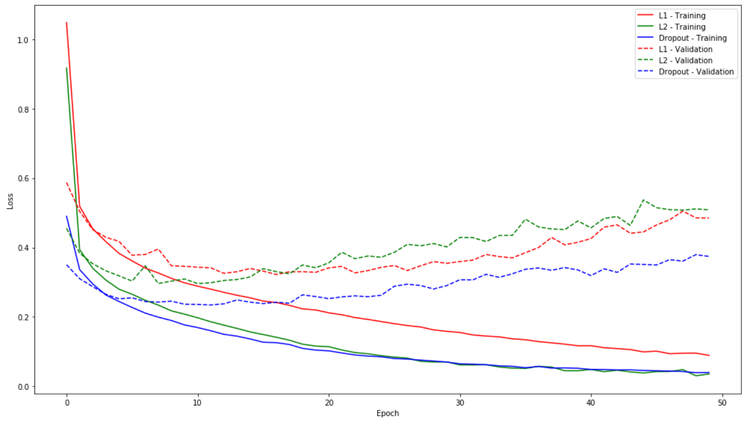
Table 5: Time Consuming of Regularization Scenarios

Regularization Method	**Time (s)**
L1	2177.63
L2	2173.93
Dropout	2177.95

Dropout is the best alternative among the regularization scenarios.



(a) Accuracy



(b) Loss

Figure 7: Experiment Result for Different Regularization Scenario

Problem 9. Experiment of Generative Adversarial Nets (GAN) on fashion MNIST dataset.

In this experiment, I applied Conditional GAN. The loss function from the generator in Fig.8 is always greater than discriminator. Both network reaching the convergence value after 800 epoch.

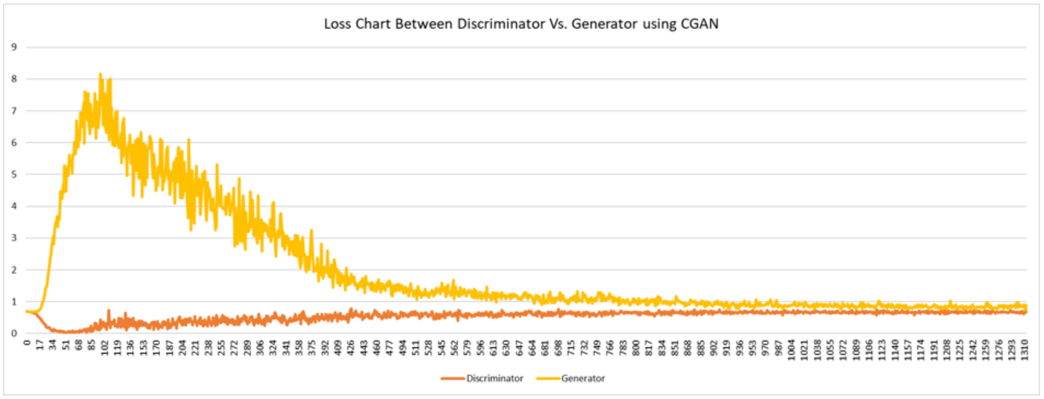


Figure 8: CGAN Experiment Loss Result on Fashion MNIST

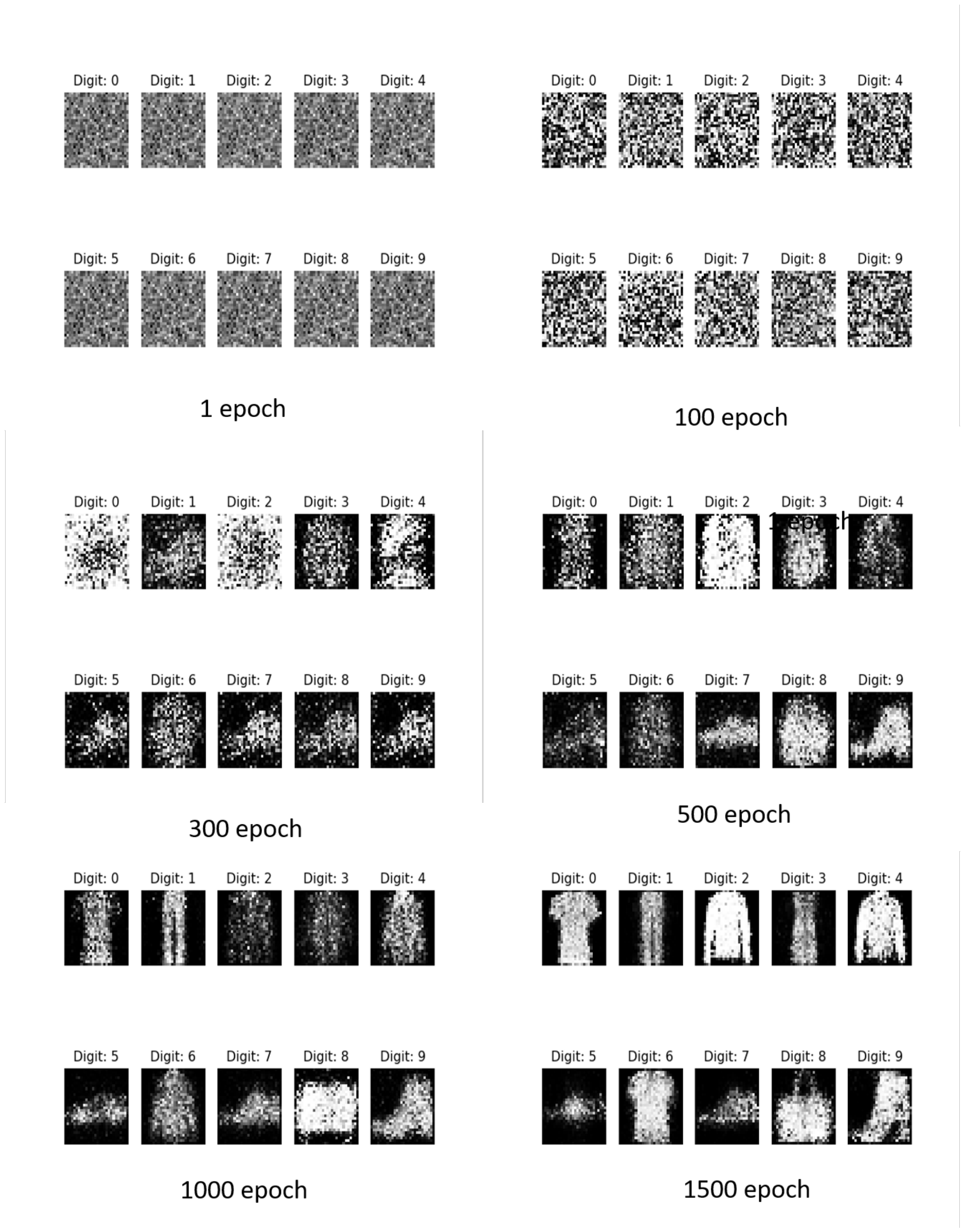


Figure 9: CGAN Experiment Image Result on Fashion MNIST