# COMP 551 - Assignment 3 - Group 11

Yian Bian
yian.bian@mail.mcgill.ca

Yijia Jing
yijia.jing@mail.mcgill.ca

**Abstract**

There are multiple machine-learning solutions for image processing. In this project, we investigated the performance of the Multilayer Perceptron (MLP) model. We implemented the MLP class and trained multiple models with the "Fashion_MNIST" dataset in an attempt to make them "fashionable", i.e. correctly recognize the type of clothes. We found three optimal hyperparameters by evaluating the performance on the validation set. We found that models with one or two hidden layers do not have a significant difference and they both outperform one without any hidden layer. By conducting experiments, we observed that ELU is the empirically best activation function. We explored the effects of L2 Regularization and normalization, which are found to be important on the resulting accuracy. After optimizing our MLP model by adopting a different activation function on each layer, we obtained an accuracy of 87.9% and compared the performance with that of a Convolutional Neural Network (ConvNet) and discovered that ConvNet achieves a 3-4% higher accuracy than MLP.

## 1 Introduction

In this project, we used a multi-layer perceptron with different numbers of hidden layers and different types of activation functions to classify image data-Fashion MNIST. Fashion MINST contains 70,000 28*28 images of fashion products from 10 categories. It is commonly used to train and test various machine-learning algorithms. Alisson Stefens Henrique et al.[1] applied different Neutral Networks on this dataset and also compared the classification results with those using SVM. Also, Ahmed Anter and et al.[2] applied CNN LeNet-5 architecture to the classification of garments from this dataset. In this project, we also utilize the CNN model and compare its performance with our implemented MLP.

By conducting these experiments, we find that standardization will indeed improve the performance of the model, and it is not always true that the more layer the better: the model with 1 hidden layer performed the best when other settings remain the same. The activation function will also affect the classification accuracy. More specifically, ELU and Leaky-ReLU are better than the other functions we experimented with. Additionally, adding a convolutional hidden layer will also improve the accuracy.

## 2 Datasets

### 2.1 Description

In this project, we performed analysis on the Fashion_MNIST dataset, which is a dataset of article images consisting of a training set of 60,000 examples and a test set of 10,000 examples. Each example is a 28*28 grayscale image, associated with a label from 10 classes. We checked the class distribution of the training set in Figure 1 and found that it contains class-balanced balanced data with all the classes having 6,000 samples equally.

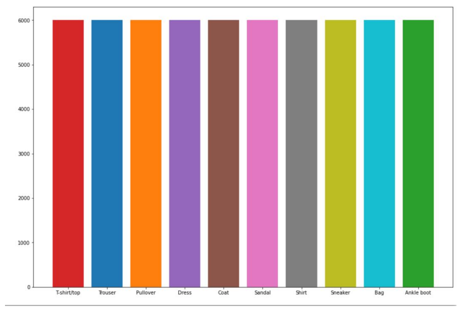


**Figure 1:** Class distribution

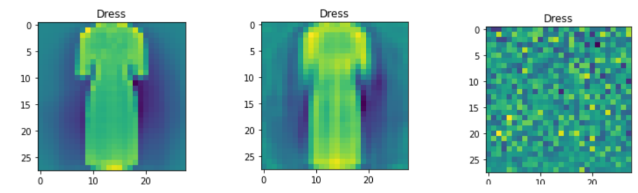
### 2.2 Data Cleaning

Since each sample in the data is a 28*28 image and we would like to apply MLP on the data, we first vectorized the data to the appropriate dimension - a two-dimensional matrix.

Then we started to normalize the data. We first subtracted its mean to make it zero-centered, then we divided each dimension by its standardization. After normalization, we visualized the data again, and we could observe that differences between pixels become smaller now, and the whole picture turned out to be more consistent.

Then we continued with another form of preprocessing - PCA, after using PCA, we reduced the number of features from 754 to 100, keeping the dimensions that contain the most variance which was most important to our classification. Then we visualized the data again. This time we could find that the images become more blur since we delete most of the features.

Finally, we tried to whiten the data, since adjacent pixel or feature values can be highly correlated, and whitening reduces this degree of correlation. We can see from the figure that, after whitening, we can not determine the category of these pictures as it is now deeply processed.



a) Normalization  b) PCA  c) Whitening

**Figure 2:** Data visualization after 3 forms of data prepossessing

## 3 Results

### 3.1 Hyperparameter Tuning

In this section we tuned 3 parameters - batch size, learning rate, and lambda to get a better model for our later prediction. We further split the training data into a training set (70%) and a validation set (30%).

#### 3.1.1 Batch Size

We first explored the batch size, since we were using mini-batch gradient descent. A batch size that is too large will lead to poor generalization, but if the batch size is too small the model is not guaranteed to converge to the global optima. So here we selected several batch sizes and trained the model to find the appropriate size. From Table 3, we can observe that the model with a batch size of 200 has the highest accuracy. Therefore, we finally utilized a batch size of 200.

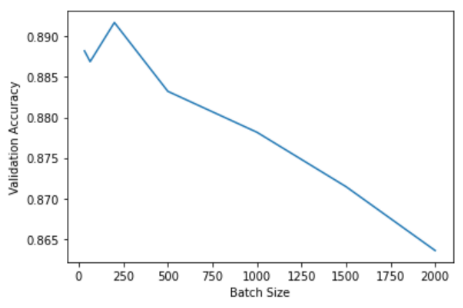
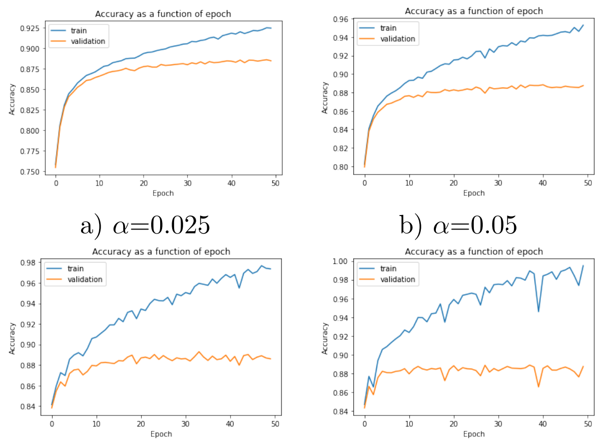


**Figure 3:** Accuracy for models with different batch size

#### 3.1.2 Learning Rate

We tuned the learning rate since a large learning rate will result in an oscillation in accuracy, while a small learning rate will take a long time to train. From Figure 4 we can find that models with a learning rate of 0.1 and 0.25 oscillate a lot and hence do not converge well to the optimum. Although the model with a learning rate of 0.05 has higher accuracy and is less oscillating, the difference is extremely small and with 0.05, it is still not stable. So we finally used a learning rate of 0.025.



a) $\alpha$=0.025  b) $\alpha$=0.05  c) $\alpha$=0.1  d) $\alpha$=0.25

**Figure 4:** Convergence plot with different learning rates

#### 3.1.3 Lambda for L2 regularization

We also varied lambda for L2 regulation as a tunable hyperparameter, which controls the strength of the penalty on weights. The larger the lambda, the more the coefficients are shrunk toward zero (and each other). From Table 1 we can find that the accuracy is the highest when lambda equals to 1.

### 3.2 Different Number of Hidden Layers

We investigated the effect of the number of hidden layers on the performance of our MLP model. We

**Table 1:** Accuracy with different $\lambda$'s for L2-norm

| $\lambda$ | 0.2 | 0.5 | 1 | 5 | 10 |
|---|---|---|---|---|---|
| Accuracy(%) | 88.51 | 88.44 | **88.52** | 85.01 | 82.49 |

fixed the activation function to ReLU and the number of units to 128 on each layer, if any, and recorded the test accuracies on models with no hidden layer, a single hidden layer, and two hidden layers. The results are presented in Table 2. We noticed that when there is no hidden layer, the model performed decently, contrary to what we believed. The highest accuracy, 87.47%, occurred when there is one hidden layer. As we increased this number to 2 and even 4, the accuracy barely changed and even decreased by a little. Therefore, it may not always be the case that "the deeper, the better".

**Table 2:** Accuracy with different numbers of hidden layers

| # hidden layer | 0 | 1 | 2 | 4 |
|---|---|---|---|---|
| Accuracy (%) | 84.08 | 87.47 | 87.31 | 86.17 |

By looking at the trend of convergence of accuracy in Figure 5, we found that at first (epoch 0), the higher the number of hidden layers, the lower the accuracy. This is because we randomly initialize weights for each hidden layer at first, so it is expected that the output we get does not make sense until further training is done. As we train the model for more epochs, the model with 2 hidden layers heads the most stably toward the optimum, which is desirable, while the one without the hidden layer oscillates a lot throughout the process and does not seem to converge well.

We also noticed that as we increase the number of hidden layers, there is less overfitting. This can be seen from the gap between the blue and orange lines in the figure. When there are 2 or 4 hidden layers, although the training accuracy is still higher than test accuracy, the level of overfitting is still within a valid range unlike with 0 or 1 layer. Therefore, a low or high number of hidden layers each have its own advantage - a lower one might sometimes lead to higher accuracy and a higher one reduces overfitting. We should choose the appropriate number of layers based on experiments and different metrics.

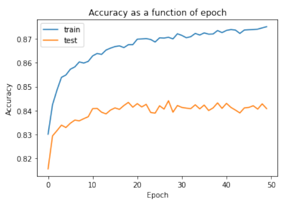
## 3.3 Different Activation Functions

An activation function decides whether a neuron should be activated or not. It plays an important role in transforming our weighted input into a value that we further feed to the next layer or the output. To find the ideal activation function, we measured the per-

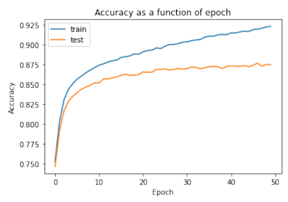
**Table 3:** Accuracy with different activation functions

| Activation function | Accuracy (%) |
|---|---|
| ReLU | 87.31 |
| Tanh | 86.58 |
| Leaky-ReLU | 87.48 |
| Logistic | 83.37 |
| ELU | **87.53** |

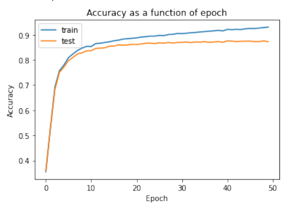
formance on five common ones - ReLU, Tanh, Leaky-ReLU, Logistic and ELU. In Table 3, we observed that ELU outperforms other functions, followed by Leakly-ReLU and ReLU. The resulting test accuracy is very similar among the three. For Tanh and Logistic, it is lower and hence they are less suitable in comparison with the others, although not by a lot. From our experiments, we could pick either one of the three to achieve a better performance. In fact, ReLU, Leaky-ReLU, and ELU, as suggested by their names, are somewhat similar in terms of computation, thus explaining their similarity in performance.



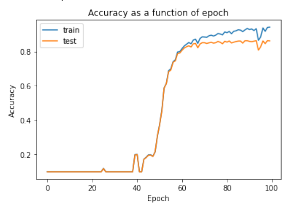
a) no hidden layer



b) 1 hidden layer



c) 2 hidden layers

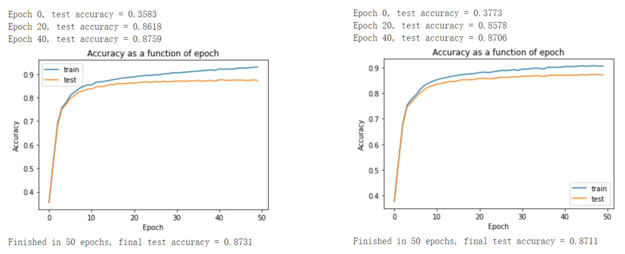


d) 4 hidden layers

**Figure 5:** Convergence plot with different # hidden layers

## 3.4 L2 Regularization

Previously, we found the most appropriate lambda (which is 1) for L2 regularization, and we also compared models with different layer settings. Here, we added L2 regularization to the network weights for all the layers and trained the MLP. From the results shown in Figure 6, on one hand, we can observe that model without regularization generally has a higher accuracy with different epochs. However, the differences between the accuracy of these two models are relatively small - around 0.2%, which might be explained by the randomness in data splitting and weight initialization. On the other hand, regularization may improve the model in the sense of decreasing loss.



```
Epoch 0, test accuracy = 0.3583
Epoch 20, test accuracy = 0.8618
Epoch 40, test accuracy = 0.8759
Finished in 50 epochs, final test accuracy = 0.8731
```

```
Epoch 0, test accuracy = 0.3773
Epoch 20, test accuracy = 0.8578
Epoch 40, test accuracy = 0.8706
Finished in 50 epochs, final test accuracy = 0.8711
```

a) without L2 regularization b) with L2 regularization

**Figure 6:** 2-layer MLP with & without L2-regularization

## 3.5 Training with Unnormalized Images

As mentioned before, we normalized our input images before conducting experiments in order to eliminate the negative effect of different scales. We would like to verify whether the normalization is as vital as what is commonly believed, so we trained our MLP model with two hidden layers with unnormalized images. Surprisingly, the accuracy is only 5.04%, significantly lower than our expectation. This is similar to the probability of a correct random guess, i.e. randomly picking one among ten categories. If we look at the original images, each pixel differs from the others a lot as it represents a color code. Therefore, we concluded that normalization helped greatly in this context and we should most likely consider doing that when training an MLP model.

## 3.6 Training a ConvNet

We used the existing library - TensorFlow to create a convolutional neural network with 2 convolutional and 2 fully connected layers where hidden units are both 128 and activation functions are both ReLu. We can find that accuracy for the training set is higher than 97.5% and that for the test set is also higher than 90%. The increase in accuracy indicates that adding 2 convolutional layers can improve the performance of MLP. However, we can also find that after the epoch equals 7.5, training accuracy continues to increase, but test accuracy mainly remains the same, and meanwhile, test error also starts to increase, which suggests the model is overfitting.

## 3.7 MLP vs. CNN

Since we want to compare the performance of our MLP with the CNN model, we made the number of the hidden layer equal to 4 which is the same as the CNN model. First, we set the assign all activation functions

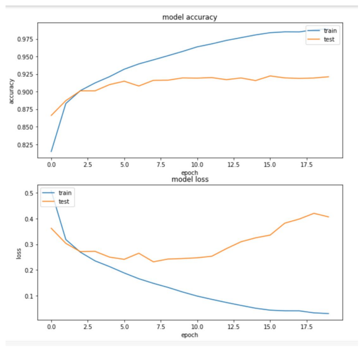
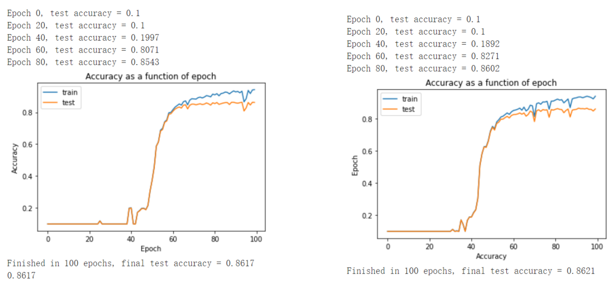


**Figure 7:** CNN's performance

to be ReLU. Then we changed the first two hidden layers to ELU since ELU had the best performance in our previous experiments. Also, it should be noticed that here we changed the epoch to 100 since when the epoch is 50, the accuracies were around 60%, and we could see a sharp increment as the epoch grew bigger than 40. The results are shown in Figure 8, we can see that these two models' performances are barely the same, but they are both worse than the CNN model. So we can conclude that for this dataset, the convolutional layers are more useful than fully connected layers in training to model to classify the image data.



```
Epoch 0, test accuracy = 0.1
Epoch 20, test accuracy = 0.1
Epoch 40, test accuracy = 0.1997
Epoch 60, test accuracy = 0.8071
Epoch 80, test accuracy = 0.8543
Finished in 100 epochs, final test accuracy = 0.8617
0.8617
```

```
Epoch 0, test accuracy = 0.1
Epoch 20, test accuracy = 0.1
Epoch 40, test accuracy = 0.1892
Epoch 60, test accuracy = 0.8271
Epoch 80, test accuracy = 0.8602
Finished in 100 epochs, final test accuracy = 0.8621
```

a) ReLU*4 b) ELU+ELU+ReLU+ReLU

**Figure 8:** 4 layer with different activation functions

## 3.8 Using a Different Activation Function on Each Layer

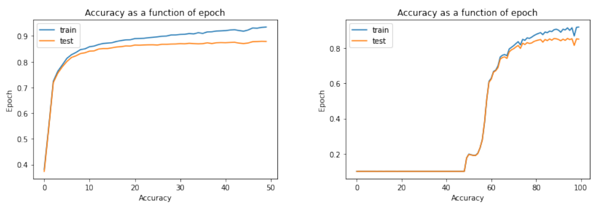
In an attempt to further optimize our MLP model, we modified the implementation such that we can adopt a different activation function on each layer. We experimented with various combinations and measured the accuracy in Table 4. We also noticed a few interesting points.

First, the functions that previously achieved the highest accuracies by themselves seem to have the best performance when combined together. This is verified by the fact that the model with two hidden layers each with ReLU and ELU as the activation function achieved the highest accuracy.

Second, the number of layers generally does not play an important role, although one that is too high does hinder the performance. As the table is ordered by accuracy, we can see that models with 2 and 3 layers perform similarly, while those with four layers seem to be too much and are at the bottom of the table.

Third, the order of activation functions does matter. In a model with four layers, where two have ReLU as their activation functions and the other two have ELU, ELU+ELU+ReLU+ReLU has a higher accuracy than ELU+ReLU+ELU+ReLU. This is an empirical observation and the best order is certainly not deterministic.

Finally, the higher the number of hidden layers, the more epochs we need in order to reach convergence. Because the weights are randomly initialized, when there are more hidden layers, the output is more randomized at first. Therefore, we need to train for more epochs for the model to be tailored to the training data. To this end, we increase the number of epochs from 50 to 100 when it comes to a 4-layer model. Figure 9 shows such a trend. A 2-layer model converges well at epoch 25 while for a 4-layer model, the increase in accuracy would not stop until epoch 90.



a) two layers    b) four layers

**Figure 9:** Convergence plot for different models trained for different number of epochs

**Table 4:** Accuracy with different activation functions

| Type | Accuracy (%) |
|---|---|
| ReLU+ELU | 87.9 |
| ELU+Tanh | 87.57 |
| ReLU+Tanh | 87.5 |
| ReLU+ELU+LeakyReLU | 87.23 |
| ReLU+Logistic | 86.57 |
| ELU+ELU+ReLU+ReLU | 86.21 |
| ELU+ReLU+ELU+ReLU | 85.11 |

## 4 Discussion and Conclusion

### 4.1 Key Takeaways

For the Fashion-MNIST data set, which involves image classification, the CNN model clearly did a better job than the simple MLP. The number of hidden layers will affect the accuracy. For this dataset, the model with 1 layer generally performs the best while models with 2 or 4 layers have less overfitting - they both have advantages and disadvantages. Also, different activation functions affect performance as well. From our experiments, ELU produces the highest accuracy of 87.53% among the 5 common activation functions.

Techniques like data normalization and L2 regularization also make a difference to the model performance. By comparing the accuracy of models trained by raw data and normalized data, we found the accuracy increased dramatically after we standardize the input. By just evaluating based on the accuracy, L2-regularization actually worsens the performance. However, accuracy is not the only metric. If we take loss and AUROC into consideration, the situation might change.

Diversifying the activation functions also help with the performance. ELU+ReLU results in the highest accuracy of 87.9% compared to all others - either with all ReLU or with other combinations of activation.

### 4.2 Future Investigation

To further improve our model, we can experiment with even more hidden layers. Although we observed that 4 layers were worse than 3 or 2 in our experiments, it might be that when we reach the appropriate, yet a high number of hidden layers, the accuracy could drastically increase. These kinds of models are computationally expensive to train but interesting to try out with enough resources.

We can also try different neural network models, such as LeNet, AlexNet, and some state-of-the-art models. They leveraged more sophisticated neural network architectures and can therefore achieve a better performance than a simple MLP.

What's more, in this project, we only standardized the data in the input layer, but we can also apply batch normalization to each hidden layer since it can reduce sensitivity to the learning rate and reduces gradient explosion.

### 4.3 Statement of Contribution

Yian was responsible for acquiring the data, implementing the MLP and L2 regularization, creating CNN, and writing the corresponding part of the report and the Introduction, Key takeaways, and Future Investigation. Yijia was responsible for implementing the MLP, running experiments on hyperparameters, activation functions, and normalization, and writing the corresponding Results, Abstract and Discussions.

## References

[1] Alisson Steffens Henrique, Anita Maria de Rocha Fernandes, Rodrigo Lyra, Valderi Reis Quietinho Leithardt, Ségio D.Correia, Pail Croker, and Tudimar Luis Scaranto Dazzi, Classifying Garments from Fashion-MNIST Dataset Through CNNs, February 16, 2021

[2] Mohammed Kayed, Ahmed Anter, and Hadeer Mohamed, Classification of Garments from Fashion MNIST Dataset Using CNN LeNet-5 Architecture, March 26, 2020