# COMP 551: Assignment 3 Report

Authors: Yiran Shi, Alina Tan, Ge Gao

Due date: December 5th, 2022

## Contents

## 1 Abstract

In this assignment, our group coded the multi-layer perceptron learning models for image classification tasks. We used this MLP model and the built-in convolutional neural network model to classify the fashion-MNIST image dataset. We intensively tested for hyperparameter settings, activation functions and architectures of these neural network models to achieve the highest testing accuracies. We also compared the performance of differently configured MLP models and CNN models and took measures to prevent overfitting issues. Eventually, we found out that 0-layer, 1-layer and 2-layer MLPs all can achieve a classification accuracy of just over 80% with different parameter settings respectively, but the CNN model can perform a lot better (over 90%) in general.

## 2 Introduction

Basic tasks of this assignment is to acquire and preprocess the fashion-MNIST image dataset, to implement MLP learning models, and to tune hyperparameters and architectures of MLP/CNN models to achieve a great-performing model on the dataset in order to perform classification task on this dataset. Apart from the MLP model that we implemented ourselves, we used keras, a high-level API of the TensorFlow Python library for the easy construction of CNN models. Some imports of datasets were also accomplished with TensorFlow built-in fashion-MNIST dataset.

The fashion-MNIST dataset was intended to replace MNIST, a dataset formed by images of handwritten digits. The machine learning community has called for stopping the usage of MNIST for various reasons like being overused and its failure to represent most realistic modern tasks. (Xiao, H. et al) The fashion-MNIST dataset was then used in many image-oriented CNN training and testing tasks. For example, Mohammed Kayed et al. has developed a CNN model named LeNet-5, which achieved a whopping 98.80% test accuracy compared to other state-of-art models like Alex Net, Google Net and VGGNet that mostly achieved 91%-92% testing accuracy. Another fact that can be used as a reference in our task is that they achieved 78.33% testing accuracy when using MLP models to classify this dataset.

Doing classification on fashion-MNIST with MLP and CNN models with different architectures gave multiple findings. Firstly, increasing MLP layer amount does not improve classification accuracies on this dataset, however, increasing layer width had significant improving effects. Secondly, the greater the number of layers in the MLP, the smaller we need to set the learning rate, and it would be more ideal to add a decent L2 regularization factor with MLP of two layers as this has a positive effect in reducing fluctuation of testing loss. Lastly, CNN models from TensorFlow greatly outperformed MLP models on this dataset regardless of architecture or hyperparameters.

## 3 Datasets

The fashion-MNIST dataset comprises 28x28 grayscale images of 70,000 fashion products in 10 categories, each category representing a kind of product. This means that each data point will have 784 parameters. Samples in the dataset were carefully selected to have a highly balanced class distribution: among the 60000 data points in the training set, each class had exactly 6000 of all data points; among the 10000 data points in the training set, each class had exactly 1000 data points.

To improve the performance of MLP models on the dataset, we normalized the dataset by subtracting the mean value of each feature from every feature column, then dividing each column by 255 as the original feature had values 0-255 to represent grayscale brightness. We also split the training set of size 60000 into a training set of size 40000 and a validation set of size 20000 to perform hyperparameter tuning.

## 4 Results

Please note that for all experiments of MLP below, we used train set and validation set to select the model with highest validation accuracy to predict the test set.

### 4.1 Three MLP models with different hidden layers

After implementing the multilayer perceptron (MLP), we performed some experiments to test how well the function is and what is the best hyperparameters. We first normalized the data and shuffled and split them into training data and validation data. Then, we created three different models: the MLP with no hidden layer, a single hidden layer, and two hidden layers with ReLU activation. We used the train and validation set to choose the best learning rate among [0.01, 0.02, 0.05, 0.1, 0.5, 1] (see Appendix Fig 1-3). We reduced the number of epochs when the number of hidden layers increased to maintain efficiency. The result showed the best hyperparameter we chose to run on test data and the test accuracy. MLP with 0 hidden layer had the highest test accuracy of 79.03% (Table 1). The small perturbation was 6.357e-13 $<$ 1e-8 which confirms our gradient is calculated correctly.

|  | MLP with 0 hidden layer | MLP with 1 hidden layer | MLP with 2 hidden layers |
|---|---|---|---|
| Activation Function | NA | ReLU | ReLU |
| number of hidden units | [ ] | [128] | [128,128] |
| learning rate | 0.5 | 0.5 | 0.02 |
| Epoch | 2000 | 500 | 300 |
| Batch size | 100 | 100 | 100 |
| Test Accuracy | 79.03% | 78.29% | 74.90% |

Table 1: The test accuracy of MLP with no hidden layer, a single hidden layer with 128 units with ReLU activations, and two hidden layers with 128 units with ReLU activations.

### 4.2 Three MLP models with different activation functions

In addition, we applied other activation functions such as tanh and LeakyReLu to the MLP model with 2 hidden layers with 128 hidden units to observe their test accuracy. We used the validation set to determine what is the best learning rate for each model (see Appendix Fig 4-5). **The model with LeakyReLU as activation function and with 0.01 learning rate gave the highest accuracy**, which is 75.89% (Table 2). The model with tanh gave the lowest accuracy of 68.96% among 2 hidden layers MLP models.

| MLP with 2 hidden layers | model with ReLU | model with tanh | model with LeakyReLU |
|---|---|---|---|
| number of hidden units | [128,128 ] | [128,128] | [128,128] |
| learning rate | 0.02 | 0.1 | 0.01 |
| Epoch | 500 | 500 | 500 |
| Batch size | 100 | 100 | 100 |
| Test Accuracy | 74.90% | 68.96% | 75.89% |

Table 2: The test accuracy of MLP with two hidden layers with ReLU activation function; with tanh activation function; with LeakyReLU activation function.

### 4.3 L2 regularization on MLP models

The third experiment we performed is adding L2 regularization (lambda) to MLP with 2 hidden layers each having 128 units with ReLU activation. The result is summarized in table 3. We tuned the lambdas within the range of 0.2, 0.4, 0.6, 0.8 and the learning rate within the range of 0.01, 0.02, 0.05, 0.1. It can be seen that the best combination of learning rate and lambda for this model is 0.02 and 0.2 (Table 3). A lambda value that is too high will severely impact accuracies. The test accuracy for lambda = 0.02 and learning rate = 0.2 was 77.74%. **The test accuracy improved compared to the same model without L2 regularization.**

| learning rate | lambda | training accuracy | validation accuracy |
|---|---|---|---|
| 0.01 | 0.2 | 0.763775 | 0.7557 |
|  | 0.4 | 0.7698 | 0.76865 |
|  | 0.6 | 0.681825 | 0.68625 |
|  | 0.8 | 0.54035 | 0.5391 |
| 0.02 | 0.2 | 0.7878 | 0.7873 |
|  | 0.4 | 0.4679 | 0.4747 |
|  | 0.6 | 0.19025 | 0.19035 |
|  | 0.8 | 0.099675 | 0.10065 |
| 0.05 | 0.2 | 0.468625 | 0.4741 |
|  | 0.4 | 0.099675 | 0.10065 |
|  | 0.6 | 0.099725 | 0.10055 |
|  | 0.8 | 0.099725 | 0.10055 |
| 0.1 | 0.2 | 0.370925 | 0.3757 |
|  | 0.4 | 0.099725 | 0.10055 |
|  | 0.6 | 0.099725 | 0.10055 |
|  | 0.8 | 0.099725 | 0.10055 |

Table 3: This table showed the training and validation accuracies after adding L2 regularization (lambda) to MLP with 2 hidden layers each having 128 units with ReLU activations with varied learning rates.

### 4.4 MLP on unnormalized images

Next, we used the same MLP model (2 layers with ReLU activation and 128 hidden units) as above, but this time we trained the algorithm with unnormalized images. Since the images were not normalized, we need to use lower learning rates such as 1e-6, 5e-6, 1e-5, 5e-5, and 1e-4. We run 500 epochs and the highest accuracy was obtained with a learning rate of 1e-5, which was **67.35%** (see Appendix Fig 6 for choosing the learning rate). This result was lower than models trained with the normalized images. **Therefore, it can be seen that unnormalized images reduced the performance of the MLP.**

### 4.5 Convolutional Neural Network

The Convolutional Neural Network (CNN) was built with TensorFlow functions such as rescaling, Conv2D, MaxPooling2D, flatten, and dense. We constructed the model with **ReLU activation, 10 epochs, same padding, no stride, 2 convolutional (3*3*32 for the first filter and 3*3*64 for the second filter) and 2 fully connected layers with 128 hidden units for each**. Within 10 epochs, the best accuracy for test images was **92.07%** at the 6th epoch. We observed that overfitting starts to occur at the 6th epoch, so we should stop the training there (see Appendix Fig 7). Note that here the validation set is the same as the test set. **We can conclude that the test accuracy predicted by CNN is higher than MLP models.**

### 4.6 Best MLP architecture

We tried to find the best MLP architecture that could classify the images in the most precise way. **First, we tried to find the best combination of MLP layer count and learning rates.** This time, we used relatively smaller learning rates to test MLP models with 0-4 hidden layers, otherwise, it could cause severe accumulated numerical problems for MLP with 3 or 4 hidden layers. Much as expected, a combination of minimal learning rate and 0-2 layers did not perform well. For 3 hidden layers, we used the set of learning rates [0.001, 0.002, 0.005] and for 4 hidden layers, [0.0001, 0.0002, 0.0005]. Through the experiments, we found that 0.002 is the best learning rate for 3 layers and 0.0002 is the best learning rate for 4 layers (Table 4). The test accuracy of 3 hidden layers was 71.97%, and the test accuracy of 4 hidden layers was 71.38%. Overall, a combination of more layers and smaller learning rates could achieve decent accuracy, but as they are still not as competitive compared to previous models, plus the fact that their loss function fluctuates a lot upon testing, and it consumes more time, so we decided to not use these models.

**Then, we investigated the effect of batch size on accuracies**, which were set to be 16, 32, 64, and 128, with the 2 hidden layers model. In this round, the best result showed a test accuracy of 77.34%, a tie between batch size 64 and 128, both with a learning rate of 0.02. However, this accuracy still did not overcome the former champion among 2 layer MLPs that had a batch size of 100, so we concluded that batch size did not have a major effect on accuracies as long as they were not unacceptably small.

Up until now, we have identified some models that give the best performance: the best testing accuracy is still maintained at about 78% by the 1 layer ReLU with learning rate of 0.5 without regularization and batch size of 100, but the loss during testing fluctuates a lot for that. Another model that is quite promising and with much stabler loss is 2 layers ReLU, L2 regularization lambda of 0.2, batch size of 100, learning rate 0.02 with testing accuracy about 77%. We did some tests with combinations of these conditions combined with different batch sizes, but nothing exceeded the accuracies of these prototypes (Table 5).

**Next, we tried to increase the number of epochs to see its effect on training and testing accuracies.** Initially, we used the model of hidden layer 1 with 128 hidden units, 100 batch size, no regularization, and another model of hidden layer 2 with $\lambda$ of 0.2, 128 hidden units, 100 batch size. Results showed that although the 1 layer MLP outperforms the 2 layer MLP, its training/testing accuracies kept fluctuating (See Appendix Figure 8). To fix this issue, we tried to introduce some minimal regularization into the 1 layer MLP model. After further experiments, combined with findings uncovered later

| MLP | 0 layer | 1 layer | 2 layers | 3 layers | 4 layers |
|---|---|---|---|---|---|
| learning rate | 0.005 | 0.005 | 0.005 | 0.002 | 0.00002 |
| activation function | NA | ReLU | ReLU | ReLU | ReLU |
| hidden units | NA | 128 | 128 | 128 | 128 |
| Epoch | 500 | 500 | 500 | 500 | 500 |
| Training Accuracy | 23.37% | 63.74% | 75.14% | 73.75% | 72.96% |
| Validation Accuracy | 23.46% | 64.30% | 74.31% | 72.967% | 71.44% |

Table 4: The table summarized the best case of training and validation accuracies of various learning rates for 0 hidden layer, 1 hidden layer, 2 hidden layers, 3 hidden layers, and 4 hidden layers

| hidden layer(s) | learning rate | batch size | hidden units | lambda | epoch | Test Accuracy |
|---|---|---|---|---|---|---|
| 1 | 0.01 | 64 | 128 | 0.2 | 500 | 68.80% |
| 1 | 0.02 | 64 | 128 | 0.2 | 500 | 75.73% |
| 2 | 0.01 | 64 | 128 | 0.2 | 500 | 74.88 % |
| 2 | 0.02 | 64 | 128 | 0.2 | 500 | 77.34% |
| 2 | 0.02 | 32 | 128 | 0.2 | 500 | 77.16% |
| 2 | 0.02 | 128 | 128 | 0.2 | 500 | 77.34% |

Table 5: The table summarized test accuracies of MLP model with 2 hidden layers and different hyperparameters.

from section 4.7.1, we were able to tune a model that had **1 hidden layer with 0.5 learning rate, hidden units of 256, batch size of 128, and $\lambda$ of 0.001** that achieved testing accuracy of **85.72%** and would not show signs of overfitting at even 2500 epochs (See Appendix Fig 10-11). On the other hand, it is quite obvious for the 2-layer model with regularization that the best time to stop training, so that overfitting can be prevented, is at about 400-450 epochs (see Appendix Fig 9), and the best testing accuracy was fixed at about 78% at this moment (could be further improved at section 4.7.1, but still not compatible with the 1-layer MLP). **Eventually, we arrived at the conclusion that the best MLP architecture is this single-layered MLP that had a large learning rate and small L2 regularization.**

## 4.7 More Investigations

### 4.7.1 Effect of the width

As the MLP model with 2 hidden layers performed best earlier, we continued to investigate the effect of the width (number of units in the hidden layers) on the test accuracy. We increased the width in the range of 32, 64, 128, 256, and 512. The model was constructed with 2 hidden layers, with 0.02 learning rate, 0.2 lambda, 100 batch size, and 400 epochs. **The plot indicated that increasing hidden layer width has a significant effect on improving accuracies** (see Appendix Fig 12). The test accuracy finally reached 80.3%. The only problem is that doubling layer width means almost tripled time, but the accuracy can only be improved by around 2%. We finally chose 512 hidden units for the 2 hidden layers model and 256 hidden units for the 1 hidden layer model.

### 4.7.2 Effect of the ConvNets' hyperparameters

We also explored the effect of the ConvNets' hyperparameters related to their convolutional layers such as filter size, number of filters, and strides on its test accuracy. The filter size set to 5 showed the highest test accuracy (see Appendix Table 6). The table revealed that the number of filters set to (32, 64) for the two layers seems to be optimal for the dataset, with greater or smaller sizes giving equal or slightly worse performance (see Appendix Table 7). In addition, **the greater stride size improves efficiency but impairs accuracies compared to the smallest, (1, 1) stride size** (see Appendix Table 8).

### 4.7.3 Effect of the dropout node proportions on ConvNets

We also examined the effect of different dropout node proportions on the final performance (see Appendix Table 9). **The proportion of 0.1 gave the best result**, test accuracy of 90.31%, in the range of 0.1, 0.3, and 0.5.

### 4.7.4 Training the MLP and ConvNet with $10^k$, $k \in \{0, 1, 2, 3, 4\}$ images and plotting the test accuracy

We split the training set size into 1, 10, 100, 1000, and 10000 images and then test the accuracy with 0 hidden layers, 1 hidden layer of the MLP model, and CNN. All three plots showed increased test accuracy as the training set increased; however, the test accuracy increased slower after 2000 images of training set size for both MLP and CNN (see Appendix Fig 13-15). **This might show that we could reduce the train set size to 2000 images to train our models.**

## 5 Discussion and Conclusions

The task of this assignment is to classify the fashion-MNIST dataset using MLP and CNN respectively. By tuning the hyperparameters and architectures of both two models, we compared their classification ability to the image dataset. We found that CNN performs better than MLP since MLP does not detect the image pattern whereas CNN does.

For each experiment, we used the shuffled dataset and split the original train set into the train set and validation set for model selection, and we predicted the test set using the model with the highest validation accuracy. Below we will discuss the experiments and the results we did for tuning the hyperparameters of MLP and CNN.

In **4.1**, we developed three different models, an MLP with no hidden layer, an MLP with 1 hidden layer having 128 hidden units and ReLU activation, and an MLP with 2 hidden layers each having 128 units with ReLU activation. After hyperparameter tuning, we found that the MLP with no hidden layer and the MLP with 1 hidden layer demonstrated similar performance, the MLP with 2 hidden layers did not predict as well as these two models. This shows that neither non-linearity nor network depth has a significant effect on the predictions' accuracy. However, deeper networks work better than shallow ones. The non-linear activation function is used to decide whether the neuron's input to the network is important or not in the process of prediction. And so, both non-linearity and network depth would contribute to better prediction. **We think our result is not expected because we did not add a regularization term when performing gradient descent. Deeper network might be overfitting.**

In **4.2**, we took the MLP model with 2 hidden layers from 4.1 and make predictions using tanh and Leaky-ReLU as activation function respectively. After hyperparameter tuning and comparing their test accuracies, we observed that the models with activation functions of ReLU and Leaky-ReLU demonstrated similar predictive ability, whereas the model with tanh activation function had a significant drop in the test accuracy. **This is an expected result since tanh function saturates at 1 for large positive and -1 for large negative inputs.** Once saturated, it becomes challenging for the algorithm to continue to adapt the weights to improve the performance of the model. Another potential risk of using tanh function is that it could cause **vanishing gradient** problem when backpropagating the partial derivative. Furthermore, **the test accuracy of Leaky-ReLU is slightly better than ReLU's** because in some cases, ReLU would omit the importance of negative neurons when the learning rate is high and there is a large negative bias whereas Leaky-ReLU would not.

We investigated the impact of L2 regularization in **4.3** and discovered training the model using unnormalized images in **4.4**. **The test accuracy of using unnormalized images drop significantly. The importance of using normalized data is to keep all the features on the same scale.** If data are not on the same scale, we might be overcompensating for a correction in one weight dimension while under-compensating in another. Additionally, we added L2 regularization to the weights of the MLP model with 2 hidden layers from 4.1. After tuning the hyperparameters, **we observed that the test accuracy of adding the L2 regularization to the model is higher than the model without L2 regularization.** This improvement is expected since the complex model is easier to overfit, we need L2 penalty to adjust the weights and **prevent overfitting** and improve the prediction accuracy.

Then in **4.5**, we built a CNN model to compare its performance with the MLP model we developed in the previous sections. We observed that the test accuracy of CNN is 92.07% which is a lot higher than all the MLP models. This is because we **detect patterns by image template matching in CNN, and the algorithm learns the template.** Comparatively, MLP treats each pixel independently, it does not consider where the location of the template is but only cares whether there is a pattern or not. **Therefore, CNN outperformed MLP on the image dataset.**

Lastly, we found the best architecture of MLP by tuning the parameters of the learning rate, L2 penalty, number of hidden layers, number of hidden units, number of epochs, and batch size in **4.6** and **4.7**. In 4.7, We also further discussed how would hidden unit width, different hyperparameters (filter size, number of filters, stride, dropout node) of CNN, and train set size affect the accuracy. We found that the best MLP model is the **1 hidden layer model with the hyperparameters of ReLU activation function, 0.5 learning rate, 256 hidden units, batch size of 128, 2500 epochs, and lambda of L2 penalty 0.001**. The test accuracy achieved **85.72%**. In 4.7, we found that widening network (number of hidden units) consistently improves the performance of the models. This observation holds for both MLP and CNN. In addition, smaller stride and larger filter size could also improve the accuracy of CNN. Changing the number of filters does not seem to improve the performance significantly. Smaller dropout nodes returned the highest test accuracy, however, the CNN model might not be overfitting even without dropout to classify fashion-MNIST. And so, the best CNN model we found is the same as the one in 4.5, the model with **ReLU activation, 6 epochs, same padding, no stride, 2 convolutional (3*3*32 of the first filter and 3*3*64 of the second filter) and 2 fully connected layers with 128 hidden units for each**. The test accuracy achieved **92.07%**. In the last experiment, we observed that although increasing train set size helps to improve the model performance for both MLP and CNN, the test accuracy grows slowly after 2000 images of train set. We could actually reduce our train set size to train our models.

In conclusion, CNN outperformed MLP for this fashion-MNIST dataset. CNN actually classifies image dataset better than MLP in most cases since its algorithm learns the image pattern. For further investigation, we could also try out AlexNet to classify image dataset. In addition, since we found that the test accuracy is still increasing even after running 2500 epochs for the MLP model with 1 hidden layer, we could train the model with larger number of epochs. Although it will take a long time to do this, it is still worth to discover how much accuracy it could improve.

## 6 Statement of Contributions

All three of us implemented the MLP and did the experiments together. For the report, Yiran Shi is responsible for the abstract, introduction and datasets; Ge Gao is responsible for the discussion and conclusion, and Alina Tan is responsible for the result and statement of contributions.

## 7 Appendix

| | Filter size = 1 | Filter size = 3 | Filter size = 5 |
|---|---|---|---|
| Train Accuracy | 86.47% | 91.96% | 91.84% |
| Test Accuracy | 85.71% | 90.39% | 90.82% |

Table 6: The table that summarized the test and training accuracies of CNN when changing the filter size.

| | number of filter=16,32 | number of filter=32,64 | number of filter=64,128 |
|---|---|---|---|
| Train Accuracy | 91.30% | 92.06% | 92.18% |
| Test Accuracy | 90.08% | 91.29% | 91.12% |

Table 7: The table that summarized the test and training accuracies of CNN while altering the number of filters.

| | stride value (1,1) | stride value (3,3) |
|---|---|---|
| Train Accuracy | 89.12% | 87.08% |
| Test Accuracy | 88.08% | 86.84% |

Table 8: The table that summarized the test and training accuracies of CNN while altering the strides value.

| | proportion of 0.1 | proportion of 0.3 | proportion of 0.5 |
|---|---|---|---|
| Train Accuracy | 91.21% | 89.94% | 87.98% |
| Test Accuracy | 90.31% | 89.88% | 89.34% |

Table 9: The table that summarized the test and training accuracies of CNN while applying different dropout node proportions.

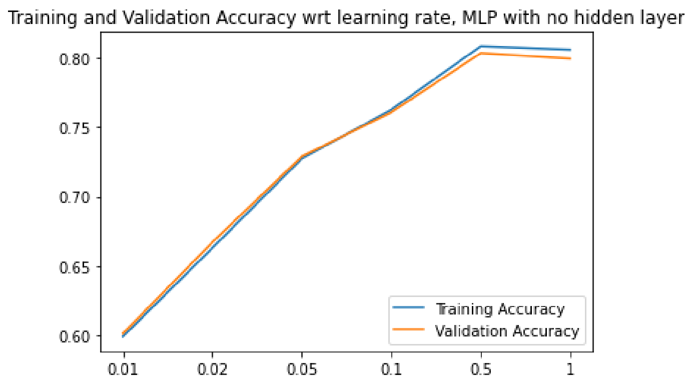

Figure 1: The plot showed the training accuracy and validation accuracy of 0 hidden layer MLP w.r.t learning rates.

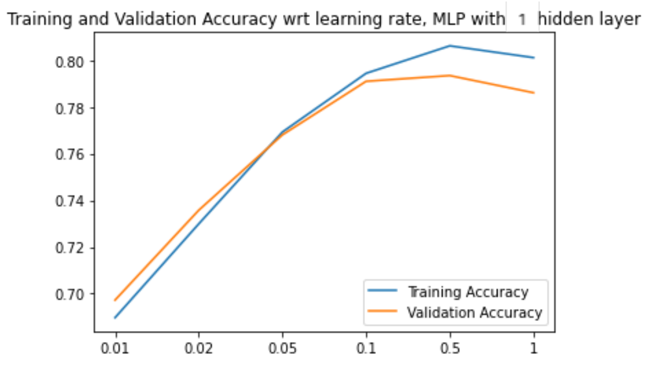

Figure 2: The plot showed the training accuracy and validation accuracy of 1 hidden layer MLP w.r.t learning rates.

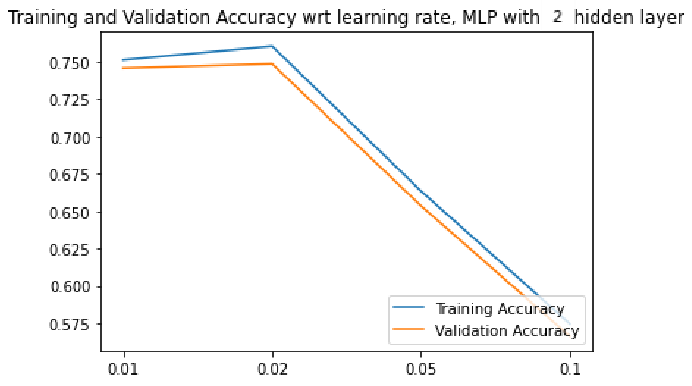

Figure 3: The plot showed the training accuracy and validation accuracy of 2 hidden layer MLP w.r.t learning rates.

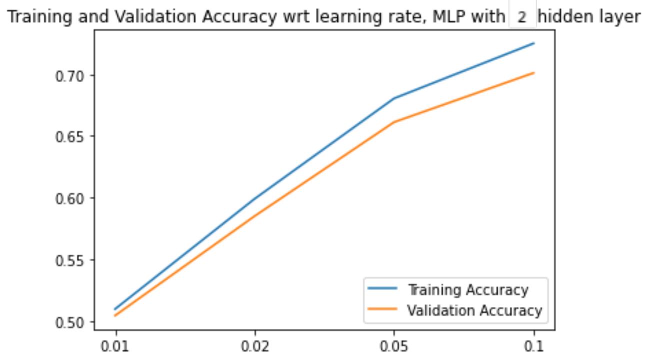

Figure 4: The plot showed the training accuracy and validation accuracy of 2 hidden layer MLP with tanh activation function w.r.t learning rates.

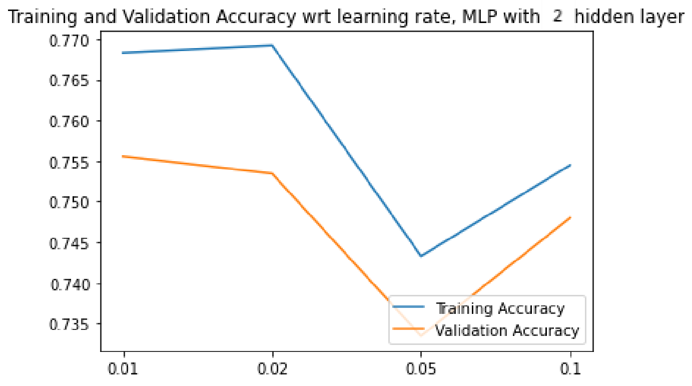

Figure 5: The plot showed the training accuracy and validation accuracy of 2 hidden layer MLP with Leaky-ReLU activation function w.r.t learning rates.

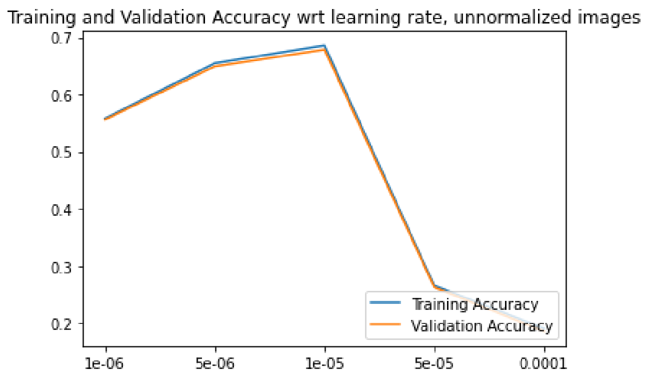

Figure 6: The plot showed the training accuracy and validation accuracy of 2 hidden layer MLP trained by unnormalized image w.r.t learning rates.

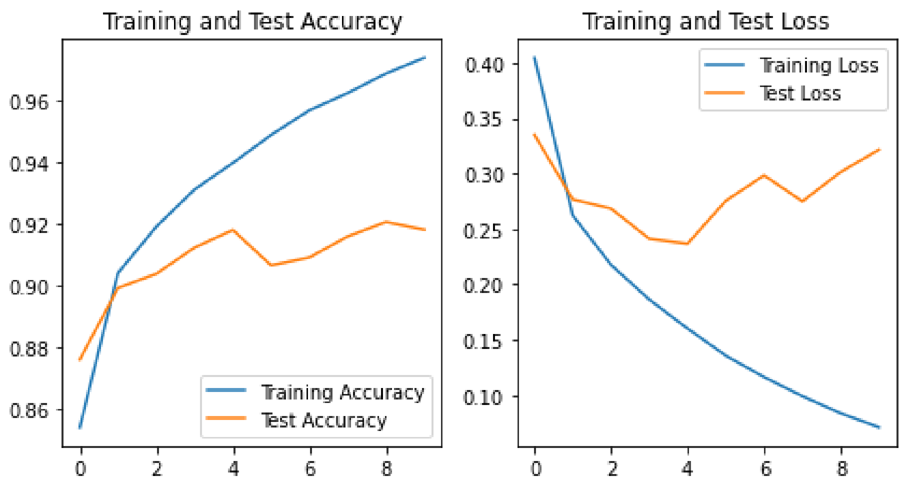

Figure 7: The plot included training and test accuracies and loss of CNN as a function of epoch

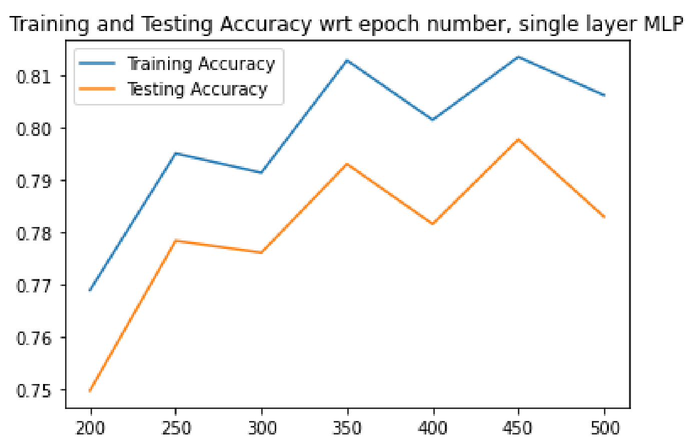

Figure 8: This plot showed the training and test accuracies of 1 hidden layer with fixed other hyperparameters as a function of epochs from 200 to 500

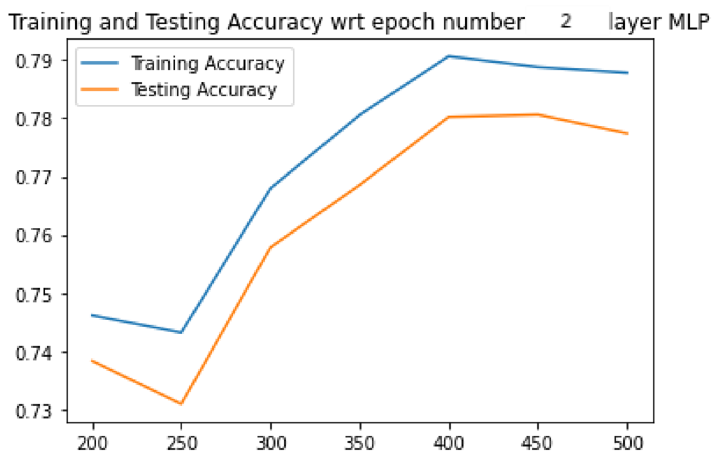

Figure 9: This plot showed the training and test accuracies of 2 hidden layers with fixed other hyperparameters as a function of epochs from 200 to 500

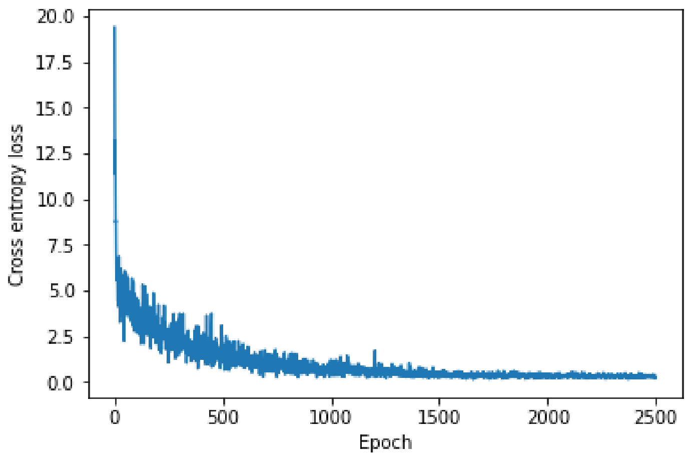

Figure 10: The plot showed that the cross entropy loss of our best model decreases to 0 as we increase the number of epochs.

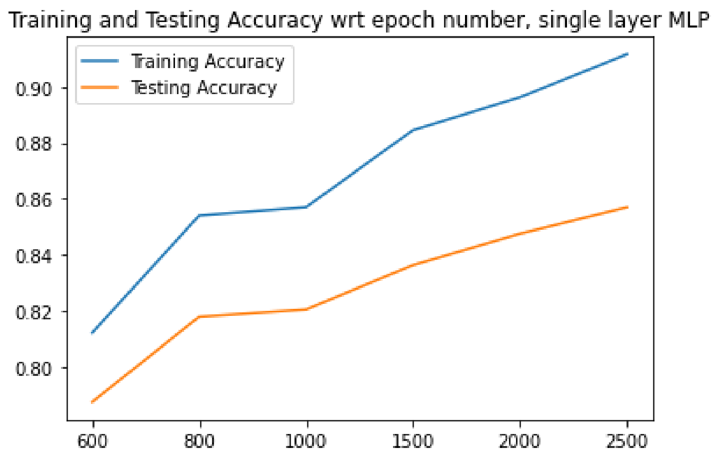

Figure 11: This plot showed the training and test accuracies of 1 hidden layer with fixed other hyperparameters as a function of epochs from 600 to 2500

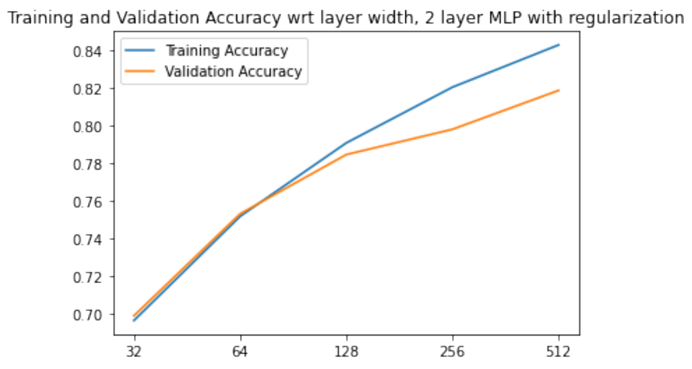

Figure 12: The plot included training and validation accuracies of 2 hidden layers MLP as a function of the number of hidden units.

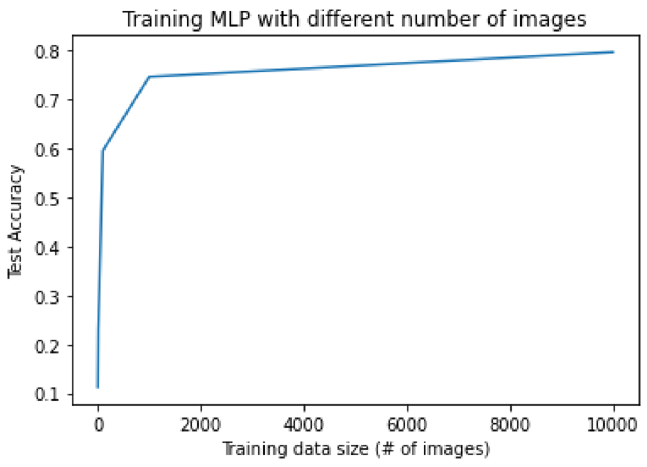

Figure 13: This plot showed the test accuracy of 0 hidden layer MLP as a function of training dataset size.

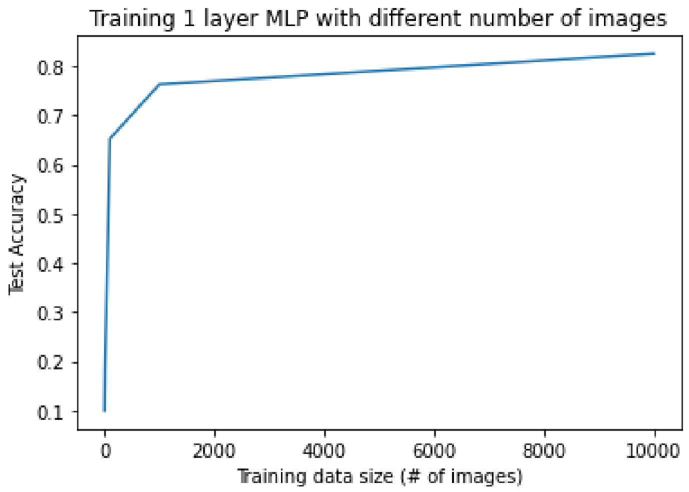

Figure 14: This plot showed the test accuracy of 1 hidden layer MLP as a function of training dataset size.

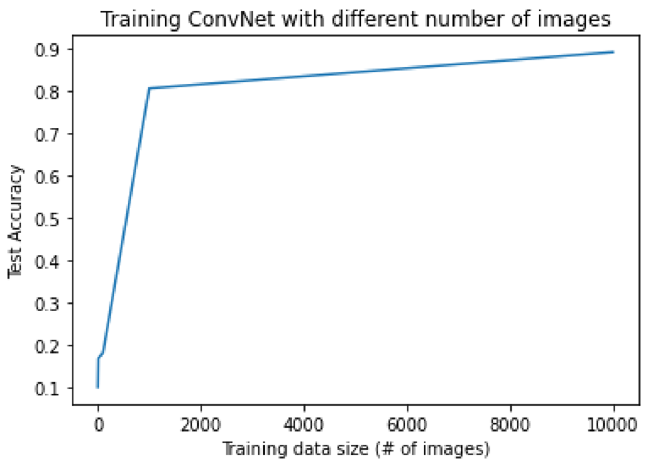

Figure 15: This plot showed the test accuracy of CNN as a function of training dataset size.

## References

[1] Kayed, M., Anter, A., Mohamed, H. (2020). Classification of garments from fashion mnist dataset using CNN lenet-5 architecture. 2020 International Conference on Innovative Trends in Communication and Computer Engineering (ITCE).

[2] Module: Tf.keras TensorFlow V2.11.0. TensorFlow. (n.d.). Retrieved December 2, 2022, from https://www.tensorflow.org/api$_docs/python/tf/keras$

[3] Xiao, H., Rasul, K., Vollgraf, R. (2017, September 15). Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms. paperswithcode.com. Retrieved December 2, 2022, from https://arxiv.org/pdf/1708.07747v2.pdf

[4] Yu, H., Samuels, D. C., Zhao, Y.-yong, amp; Guo, Y. (2019, March 4). Architectures and accuracy of artificial neural network for disease classification from OMICS data - BMC Genomics. BioMed Central. Retrieved December 3, 2022, from https://bmcgenomics.biomedcentral.com/articles/10.1186/s12864-019-5546-z