Extensive experimentation for Hyper-Parameters selection of Multi Layer Perceptron Neural Networks with Fashion MNIST data for image classification applications

Keerthiraj Nagaraj
Department of Electrical and Computer Engineering
University of Florida
k.nagaraj@ufl.edu

Abstract— **In this project, data classification problem is addressed using the concepts of supervised learning. Fashion MNIST dataset is used to train a Multi-Layer Perceptron Neural Network (MLPNN) model to classify the images of fashion products into their respective classes. We preprocess the Fashion MNIST data using the concepts of down sampling and Principal Component Analysis to reduce the amount of input for training neural networks. Hyper-parameters such as optimal number of units in the hidden layers, optimal number of hidden layers, learning rates, activation function, optimizer, number of epochs etc were experimentally decided based on mean square error for test data, classification accuracy for test data, and time taken for training the neural network model. The model with best hyper-parameters was chosen at the end from all these experiments and a confusion matrix is shown to the explain the performance of MLPNN model.**

I. Introduction

In this section, we will briefly introduce various technical concepts required to understand the analysis carried out in our project.

A. Supervised Learning

Supervised learning [1] deals with the class of problems in which we have a target dataset guiding the model on what decisions to make while it is being trained. In these problems, learning or the adaptation of model is supervised by the desired response. Supervised problems mainly deal with Polynomial Regression or also called as function finding and Logistic Regression or also called as Classification problems.

Polynomial regression tries to find a polynomial function for the output variable in terms of different combinations of input variables by assigning weights to them. The obtained polynomial equation could be used to predict the values of output for any values of the input. This problem can also be formulated as fitting a polynomial data curve through the data points which results in the least sum of normal distances from the curve to the points. In regression problems, the output value obtained is a continuous real valued number.

Logistic Regression or classification problems are the type of supervised learning methods where the outputs are non-continuous/discrete real values, where each value represents a different class. Each data point in the input is mapped into a class and a model is trained with this data. The objective of the trained model is to classify the test dataset into proper classes with least number of wrong classifications.

B. Multi-Layer Perceptron Neural Networks

Artificial Neural Networks (ANN) are supervised learning machines which contains many computational and processing units known as Processing Elements (PE) similar to Perceptrons in the brain neural network. ANN contains different kinds of layers namely input layer, hidden layer and output layer. The number of hidden layer defines the depth of neural networks. Multi-Layer Perceptron Neural Networks (MLPNN) [2] are the ANNs which has multiple number of hidden layers. Every layer in a MLPNN has a number of PEs which can have connections among themselves or connections with PEs of other layers. Each of these connections carry a weight on them which shows their significance during model training.

Each PE contains an activation function, one of most popular being sigmoid activation function whose gradient is close to zero if the input to sigmoid function is large and the gradient is around 0.25 when the input to sigmoid function is close to zero. The weights on the connections between PEs define a neural network. These weights are calculated in each iteration through 2 steps, Forward Feed calculations and Back propagation. Using the training data, the weights on each PE-to-PE connection is calculated moving from input layer to output layer through all the hidden layers. The final output obtained is compared with the target value, the error between the target value and the current model output is back propagated through methods similar to gradient descent and all the weights are modified accordingly. This procedure is repeated for all the input samples continuously for a fixed number of iteration or until convergence or until a model with satisfactory performance is obtained.

II. Dataset and Experimental Setup

In this section, we will briefly discuss about the dataset used and the experimental tools used to carry out this process.

A. Dataset

The dataset used in this project is called Fashion MNIST data [3] which is taken from a larger family of datasets from NIST. This dataset contains thousands of gray scale images of fashion products. There are images containing T-shirt/top, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, Ankle boot, totally images belonging to 10 different classes.

For this project, we have used 46,667 images for training of MLPNN models and 23,333 separate images for testing the trained models as mentioned in the project guidelines. Initially each image is converted into a 28x28 data matrix (784 input features).

B. Experimental setup and procedure

We have conducted the experiments in Python programming language in the Anaconda environment mainly using the libraries such as Keras, Scikit-learn, Pandas, Matplotlib, NumPy and SciPy. Fashion MNIST dataset was given to us in a .mat format, this file had boht inputs and labels for training and testing.

III. Experimental Results

In this section, we provide a detailed discussion about various experiments conducted in choosing the hyper parameters such as optimal number of units in the hidden layers, optimal number of hidden layers, learning rates, activation function, optimizer, number of epochs. We provide learning curves and performance metrics namely Mean Square Error percentage (MSE), Accuracy (in percentage) and training time for each of the experiments along with a note on the significant observations.

When varying a certain hyper-parameter all other are kept same to understand the impact of varying parameter on the performance of the model. The default parameters are

- Number of hidden layers: 1
- Number of PEs in 1st hidden layer: 50
- Learning rate or Step size: 0.005
- Activation function: Rectified Linear Unit (ReLU)
- Optimizer: Stochastic Gradient Descent
- Cost function: Mean Squared Error

A. Data Preprocessing

As discussed in the earlier sections, the MNIST dataset is huge and contains 70000 images and the information from the entire dataset results in a matrix of dimensionality 784x70000 which increases the computational time and complexity. For this reason, we experimented with different kinds of data preprocessing techniques mainly with an objective of reducing the number of input features.

1) Simple Down Sampling: In this method, we simply skip some features without giving any importance to the among of information lost. One simple technique used here is to use only the input features with numbers which are multiples of 1,2,3,4 and 5 out of overall 784 features. Intuitively speaking, this method just lowers the resolution of images. So, by conducting this experiment, we are assessing the performance of neural network model when trained and tested with images of different resolutions.

Fig. 1 shows the performance metrics of MLPNN model for different number of input features when simple down sampling method is used.

Fig. 2 shows the learning curves of MLPNN model for different number of input features when simple down sampling method is used. We can notice that as the number of input features reduces Accuracy decreases, MSE increases and time taken for training decreases. This shows that different levels of down sampling have direct impact of the performance of MLPNN model.

In Fig. 2, the legend bar shows the number of input features.

2) Principle Component Analysis: In this method, we initially apply PCA transformation to training and testing dataset and reduce the number of features based on different levels of information retention.

Fig. 3 shows the performance metrics of MLPNN model for different levels of information retention when PCA is used to reduce the number of input features.

Fig. 4 shows the learning curves of MLPNN model for different levels of information retention when PCA

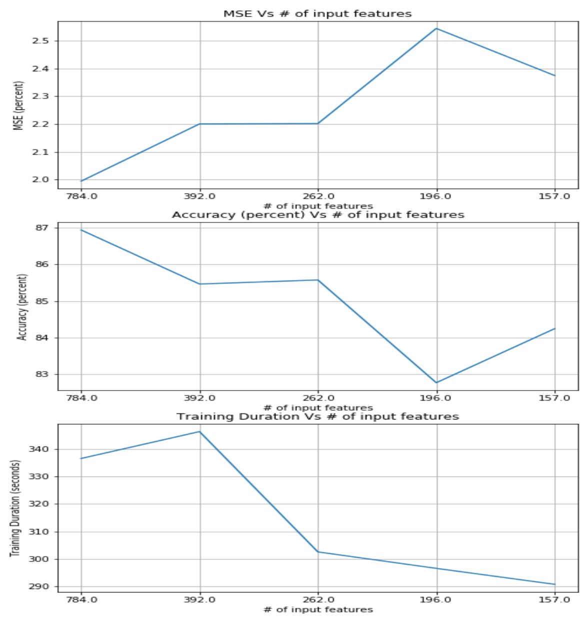


Fig. 1. Performance metrics for Down sampling method

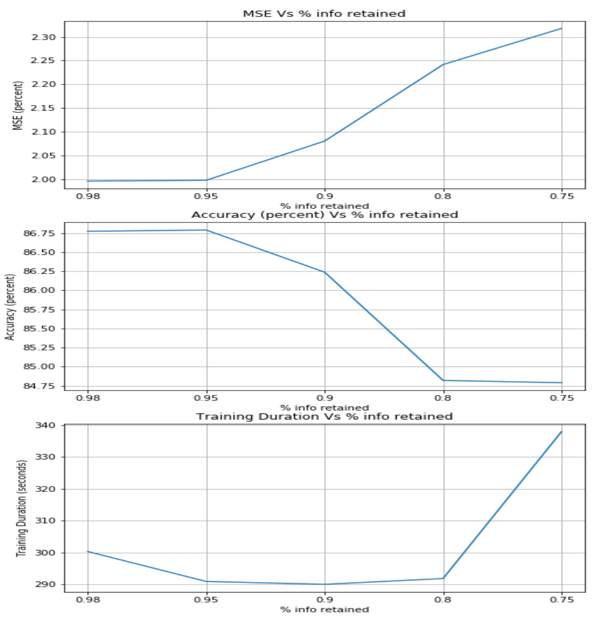


Fig. 3. Performance metrics for PCA method

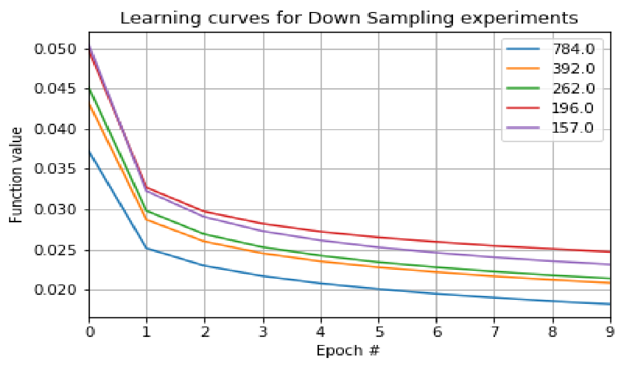


Fig. 2. Learning Curves for Down sampling method

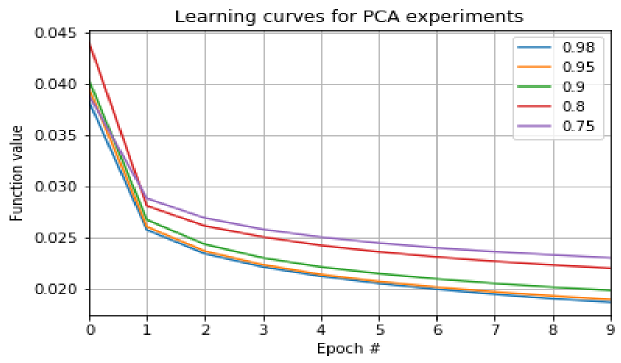


Fig. 4. Learning Curves for PCA method

is used to reduce the number of input features. We can notice that as the amount of information retained reduces the Accuracy decreases, MSE increases and time taken for training decreases.

In Fig. 4, the legend bar shows the percentage of information retained.

B. Topology - Hidden layers

Two of the most important hyper parameters that should be selected for MLPNN are the number of hidden layers and the number of inputs in each hidden layer. We experimented with the varried number of units in a hidden layer. We observed the values of training time, accuracy and MSE for test dataset for making the decisions. We initially varied the number of units in the first hidden layer from 10 to 100, based on the values performance values obtained for first hidden layer, we decided 50 as the optimal value and then varied number of PEs in second hidden layer. Although more number of PEs gives better results, this increase in accuracy is not worth the extra computation and time needed to train the model, so we decided on 50 PEs.

Fig. 5 shows the performance metrics of MLPNN model for different number of PEs in 1st hidden layer.

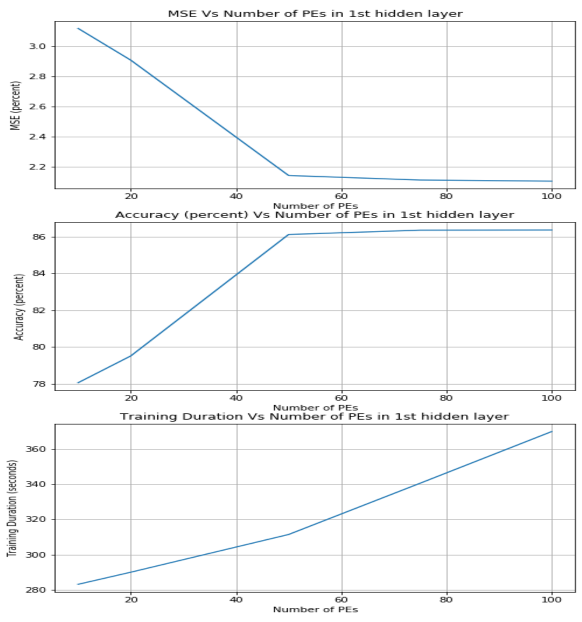


Fig. 5. Performance metrics for 1 hidden layer

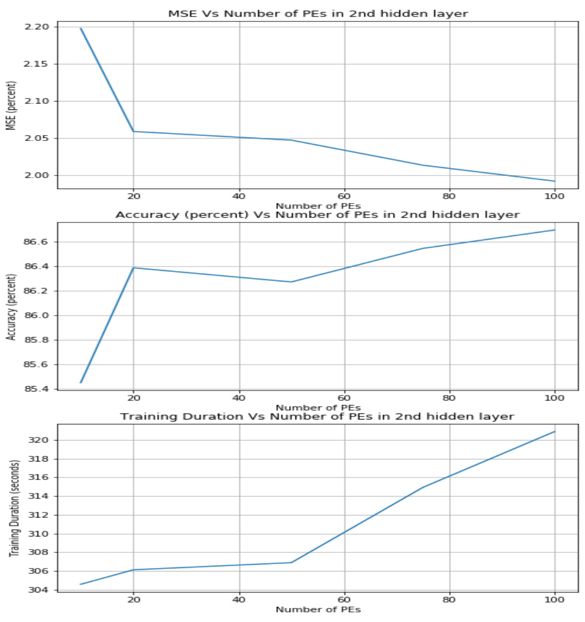


Fig. 7. Performance metrics for 2 hidden layers

Fig. 6 shows the learning curves of MLPNN model for different number of PEs in 1st hidden layer. We can notice that initially the accuracy improves significantly but remains at almost same level after 50 PEs. So, we decide 50 to be the optimal value.

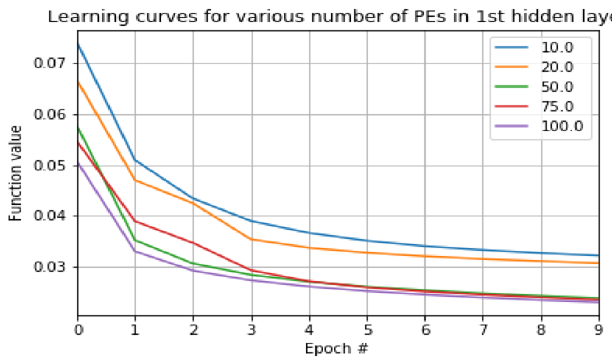


Fig. 6. Learning Curves for 1 hidden layer

In Fig. 6, the legend bar shows the number of processing elements in first hidden layer. Similar results are shown for 2 hidden layer MLPNN.

Fig. 8 shows the learning curves of MLPNN model for different number of PEs in 2nd hidden layer. We can notice that performance doesn't vary much when second layer is used compared to just 1 hidden layer but complexity of the model increases, Hence we have decided to use just 1 hidden layer with 50 processing elements for all the remaining experiments. In Fig. 8, the legend bar shows the number of processing elements in second hidden layer.

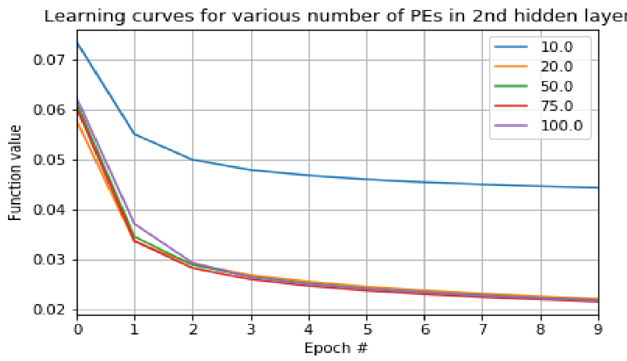


Fig. 8. Learning Curves for 2 hidden layers

C. Learning Rate

As we are using stochastic gradient descent, learning rate or step size is a very important hyper parameter. It can affect whether our model can miss in a global optima or not. It is important to vary step size in a large range and test its impact on the model performance. Here, we have experimented with different step sizes and show the performance of the MLPNN model. Fig. 9 shows the performance metrics of MLPNN model for different step sizes.

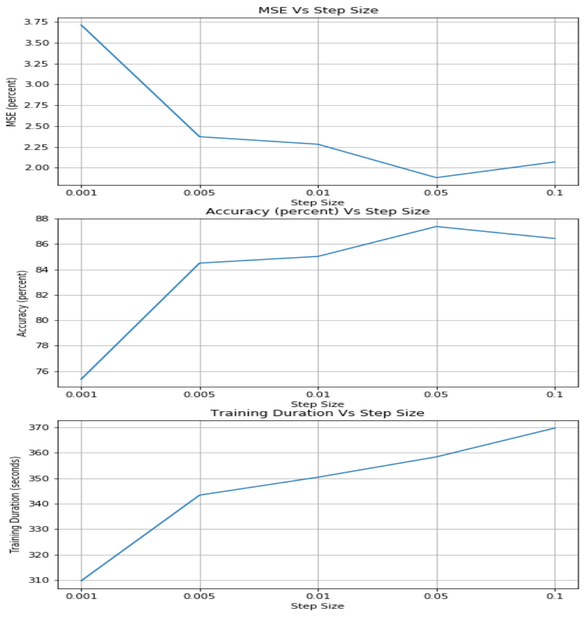


Fig. 9. Performance metrics for different step sizes

Fig. 10 shows the learning curves of MLPNN model for different step sizes. We can notice that initially the accuracy improves significantly but remains at almost same level after 0.005 step size. So, we decide 0.005 to be the optimal value. Larger step sizes are more probable in missing the global optima, hence when a lower step size is giving similar performance, we believe the lower step size should be considered. In Fig. 10, the legend bar shows the number of values of step sizes.

D. Activation Function

In this subsection, we have experimented with different activation functions such as ReLU (Rectified Linear Unit), ELU (Exponential Linear Unit), SELU (Scaled Exponential Unit), Sigmoid, Tanh and Hard Sigmoid and we show the performance of the MLPNN model for these different activation functions. Fig. 11 shows the performance metrics of MLPNN model for different activation functions.

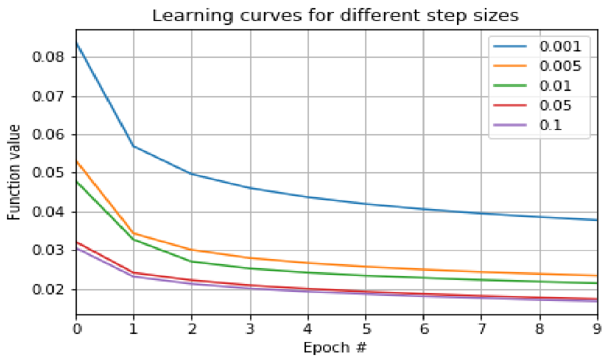


Fig. 10. Learning Curves for different step sizes

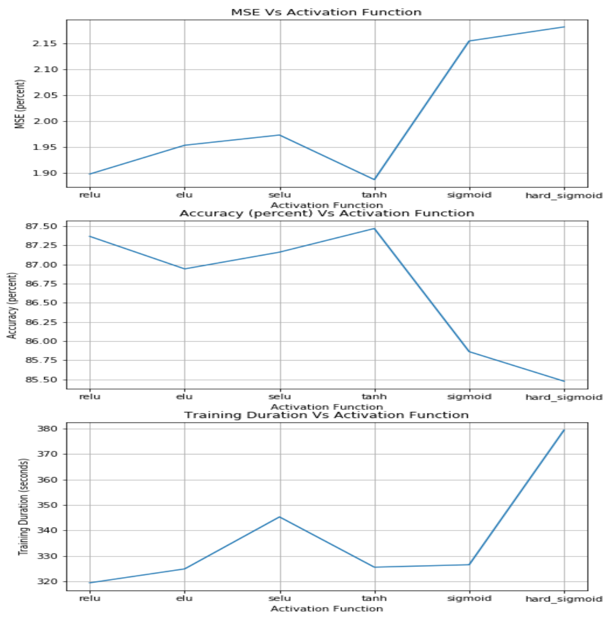


Fig. 11. Performance metrics for different activation functions

Fig. 12 shows the learning curves of MLPNN model for different activation functions. We can notice that ReLU gives best performance when you consider both training time and accuracy as the metrics. Tanh function also gives similar performance. In Fig. 12, the legend bar shows the name of activation function.

E. Optimizing Function

In this subsection, we have experimented with different optimizers such as Stochastic Gradient Descent (SGD), RMSprop, Adam, Adamax, Nadam and Adadelta, and we show the performance of the MLPNN

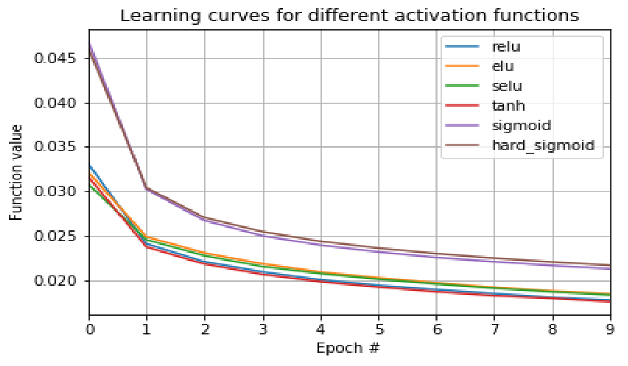


Fig. 12. Learning Curves for different activation functions

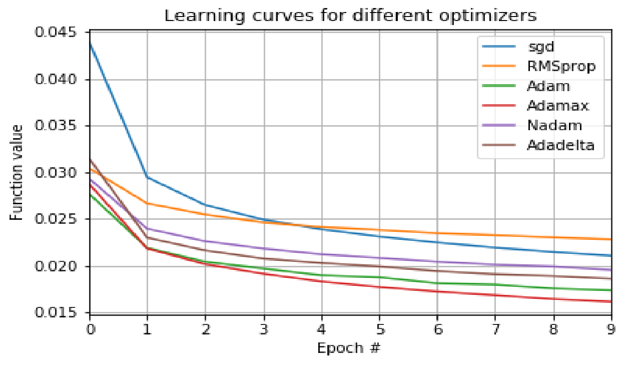


Fig. 14. Learning Curves for different optimizers

model for these different optimizers. Fig. 13 shows the performance metrics of MLPNN model for different optimizers.

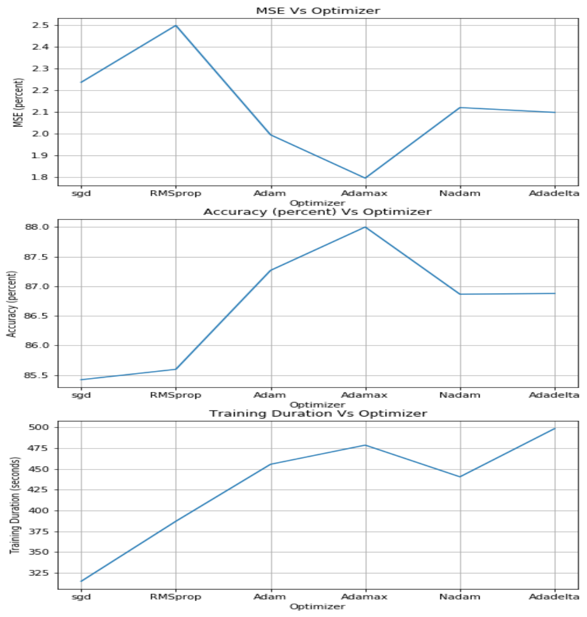


Fig. 13. Performance metrics for different optimizers

Fig. 14 shows the learning curves of MLPNN model for different optimizers. We can notice that Stochastic Gradient Descent gives best performance when you consider both training time and accuracy as the metrics. In Fig. 12, the legend bar shows the name of optimizing function.

F. Stopping Conditions

In this subsection, we have experimented with different number of epochs as the stopping condition and we show the performance of the MLPNN model for 10, 20, 30, 40, 50 epochs of model training. Fig. 15 shows the performance metrics of MLPNN model for different number of epochs.

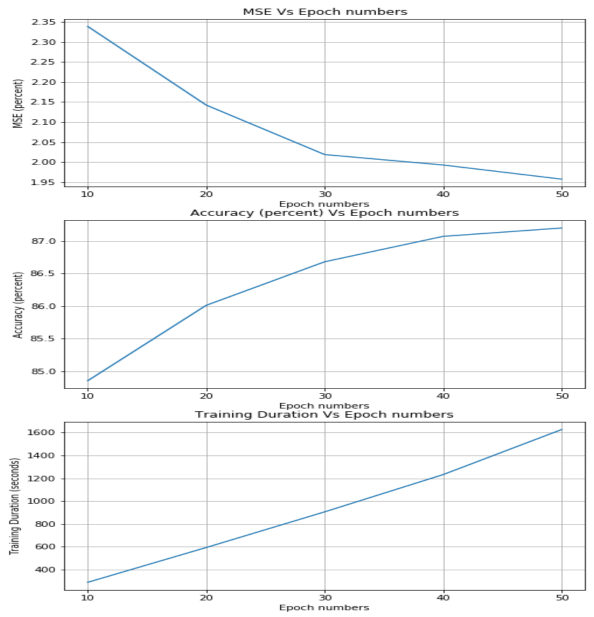


Fig. 15. Performance metrics for different number of training epochs

Fig. 16 shows the learning curves of MLPNN model for different number of training epochs. We can notice that performance doesn't vary much as the number of epochs increases and even with 10 epochs we are getting good accuracy. Hence, we decided to use 10 epochs for all the other experiments as it took less time for training the MLPNN model. In Fig. 16, the legend bar shows the number of training epochs.

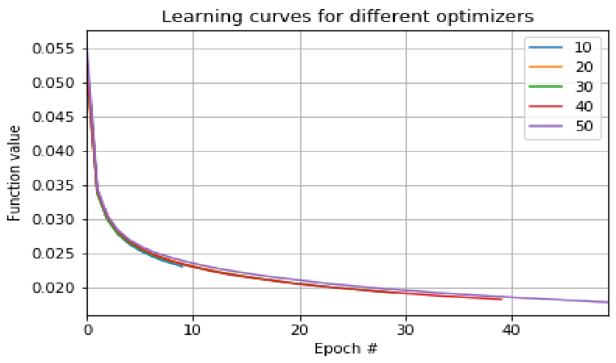


Fig. 16. Learning Curves for different number of training epochs

G. *MLPNN model with optimal hyper parameters and confusion matrix*

In this subsection, we show the results for a MLPNN model trained with optimal hyper-parameters chosen from all the previously conducted experiments. We have provided confusion matrix which shows the classification labels for each of the 10 classes in the Fashion MNIST dataset. We have also provided classification report in terms of precision, recall and F1-score for each of the classes. Fig. 17 shows the confusion matrix for the fashion MNIST dataset with the optimal hyper-parameters.

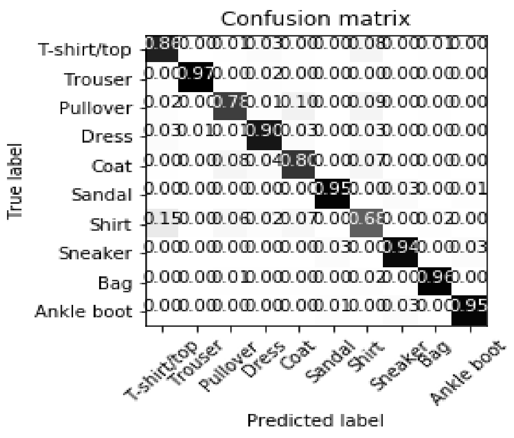
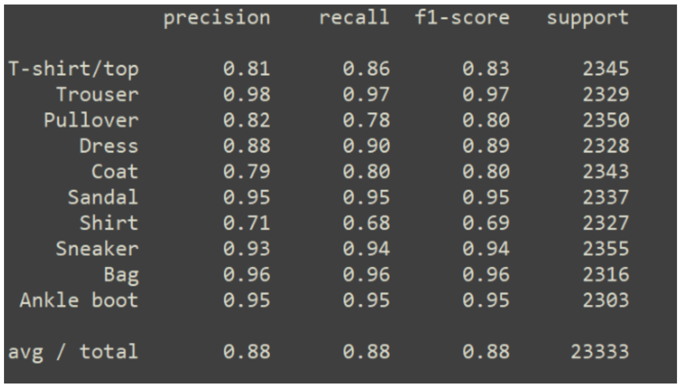


Fig. 17. Confusion Matrix

Fig. 18 shows the classification report for the fashion MNIST dataset with the optimal hyper-parameters. From Fig. 17, we can say that for majority of classes, the true prediction rate is more than 90% which is a good performance for simple MLPNN model.



```
             precision    recall  f1-score   support

T-shirt/top       0.81      0.86      0.83      2345
    Trouser       0.98      0.97      0.97      2329
   Pullover       0.82      0.78      0.80      2350
      Dress       0.88      0.90      0.89      2328
       Coat       0.79      0.80      0.80      2343
     Sandal       0.95      0.95      0.95      2337
      Shirt       0.71      0.68      0.69      2327
    Sneaker       0.93      0.94      0.94      2355
        Bag       0.96      0.96      0.96      2316
 Ankle boot       0.95      0.95      0.95      2303

avg / total       0.88      0.88      0.88     23333
```

Fig. 18. Confusion Matrix

IV. CONCLUSIONS

In this project, we trained a MLPNN model with Fashion MNIST dataset to solve a classification problem. We chose the hyper parameters such as optimal number of units in the hidden layers, optimal number of hidden layers, learning rates, activation function, optimizer, number of epoch by considering performance metrics such as training time, accuracy and MSE for test dataset. We provided plots of training time, accuracy and MSE for each of the experiment conducted. This project project report serves as a guide to understand how a classification problem can be solved on popular dataset such as Fashion MNIST dataset and how to select various hyper parameters and what kind of plots are important and how they are helpful to understand the performance of MLPNN model. 2 options were given to us: Either to conduct extensive experiments with one model or to compare performance with Conjugate gradient or Levenberg-Marquadt techniques. So, we have selected extensive experiments options for one model for this project

REFERENCES

[1] A. K. Jain, R. P. W. Duin and Jianchang Mao, ”Statistical pattern recognition: a review,” in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22, no. 1, pp. 4-37, Jan 2000.

[2] S. K. Pal and S. Mitra, ”Multilayer perceptron, fuzzy sets, and classification,” in IEEE Transactions on Neural Networks, vol. 3, no. 5, pp. 683-697, Sep 1992. doi: 10.1109/72.159058

[3] Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. Han Xiao, Kashif Rasul, Roland Vollgraf.