EEL 6814 - NEURAL NETWORKS AND DEEP LEARNING

PROJECT 1

By-

Abhishek Jha

Responsible for MLP architecture and Parameter Grid-Search

Krishna Teja Nuthalapati

Responsible for CNN architecture and Performance Metrics

Evaluation of Multi-Layer Perceptron and Convolutional Neural Network models on F-MNIST Dataset

Abhishek Jha
Dept. of Electrical and Computer Engineering
University of Florida
Gainesville, FL United States
abhishek.jha@ufl.edu

Krishna Teja Nuthalapati
Dept. of Mechanical and Aerospace Engineering
University of Florida
Gainesville, FL United States
krishnateja@ufl.edu

***Abstract*—Artificial Neural Networks(ANNs) have risen in prominence due to their ability to map non-linear relationships and because ANNs achieve better performance on datasets when compared with other classification algorithms such as Random Forest Classifiers, Support Vector Machines, and Mixture Models. A Multi-Layer Perceptron(MLP) is an architecture that uses the backpropagation algorithm to optimize the values of the training parameters. An MLP consists of layers of neurons that are densely connected. Each Neuron is composed of an activation function and a bias. MLP used momentum learning to determine the appropriate weights. Convolutional Neural Networks(CNNs) have risen in popularity over recent years owing to its superior performance in classifying image data, computer vision tasks, time-series data, medical diagnosis, and natural language processing. It derives its name from Convolution which is a mathematical linear operation on matrices. CNN's are typically composed of one or more convolution layers, Max pooling layer, and then several layers of densely connected neurons. In this paper, we evaluate the performance of the MLP model and CNN Model for Fashion MNIST dataset, which consists of image data segmented into 10 different classes, one for each type of clothing.**

***Index Terms*—Multi-Layer Perceptron, Convolutional Neural Network, Deep Learning, Gradient Descent**

I. Introduction

A Multi-Layered Perceptron(MLP) also known as Artificial Neural Network (ANN) is capable of learning any non-linear function. A single neuron (or Perceptron/Processing Element) can be considered as a Logistic Regression. Generally, an ANN consists of multiple perceptrons in a given layer. An activation function is the heart of a neural network. They introduce the non-linear properties which enable the network to learn the complex relationship between the input and the output data. They are inspired by the biological neural network.

For image classification, a much better method is using Convolution Neural Network (CNN). In this type of network, a filter also known as a kernel spans across an image converted to gray-scale extracting the features to be learned. Following that the output from the filter(s) is input to the network. MLP and CNN both have their advantages and disadvantages which will be discussed later.

Any neural network must be trained before it can produce accurate outputs. The process of learning is done by feeding in inputs, obtaining the outputs, and adjusting the weights accordingly aiming to improve the accuracy. The learning rate dictates the size of the corrective step taken. A higher rate reduces the training time but reduces the final accuracy and may cause oscillations in the network. A lower rate increases the training time but results in higher final accuracy. There are adaptive learning rate algorithms where initially the rate is high and it keeps reducing with each training step.

II. Data and Feature Sets

In this section, we describe the dataset used through our models. Originally it comprises a total of 70,000 images which are distributed into 60,000 training samples and 10,000 test samples. In our training, we augment the 60,000 training samples with 10,000 test samples and split them into 2/3 as training samples (where 1/10 of it is for cross-validation) and the remaining 1/3 of the dataset are test samples.

A. Fashion-MNIST

As the name suggests, the data set pertains to classifying fashion objects. The data set belongs to Zalando Research[1]. The data set classifies 60,000 training images which are associated with 10 labels. Each image is a grey-scale image with a resolution of 28x28 pixels. The classes 0-9 consists of data described as a T-shirt/top, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, and Ankle boot respectively.

As we can see from the data set that it contains similar objects such as in class 0 and 6 which is T-shirt/top and Shirt and class 2 and 4 which are Pullover and Coat. This represents a challenging data set to train since they have objects which are similar in description.

III. Background Research

As we now know, an MLP can approximate any non-linear function with only one layer. Hence they are also known as Universal Function Approximators. But using MLP for image classification might not be the best approach. One of the two main reasons is that as the image size increases the number of trainable parameters increases drastically which takes a lot more computing power. And second, the input must be converted from 2-Dimensions to a 1-Dimensional vector where spatial features (arrangement of pixels relative to one another) are lost. To balance this one might use a more complex architecture. A very deep neural network may run into Vanishing and Exploding gradient issues associated with the backpropagation algorithm. The weights are updated by steepest descent but when there are many layers the gradient may be too small preventing the weight from getting updated. Similarly, the gradient may accumulate and become a very large value causing an unstable network.

A CNN is a neural network which is preceded by one or more layers of kernels which extract the relevant features. Here the spatial information is preserved as the kernel spans over the entire image and convolution is performed. Convolution reduces the size of spatial representation and hence requiring a less complex network compared to MLP. Drawbacks of a CNN include translation invariance where a change in an object's position or orientation does not make a difference in what's called 'firing up' of specific neurons. CNN is slower to train compared to MLP because of additional kernel layers. Nevertheless, it results in better classification accuracy comparatively.

A. Activation Functions

Rectified Linear Unit (ReLu) outputs linearly if the input is positive and zero if it is negative. Mathematically, it is given as follows:

$$f(x) = \max(0, x)$$

Sigmoid is a function which takes an input and outputs a value between 0 and 1. It is given as follows:

$$f(x) = \frac{1}{1 + e^{-x}}$$

Softmax is a function used for multi-class classification and can be called as 'Multi-Class Sigmoid'. It is given as:

$$f(\boldsymbol{x})_i = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_i}}$$

B. Optimizers

In this section, we discuss the different optimizers that were used in evaluating our model based on grid search. RMSProp is an improvement over Gradient Descent with momentum learning. RMSProp dampens the vertical oscillations which enable it to converge faster to minima.

Adam is a combination of Adagrad and RMSProp. This method uses the calculation of the value of gradient which is given by m and its square which is given by v. The computation step computes the value of alpha for each layer and then uses it to update the learning rate. This method of optimization performs better than RMSProp since it handles the problem of exploding gradients.

$$\alpha(n) = \alpha(\sqrt{1 - \beta 2(n)}/1 - \beta 1(n))$$

$$\eta(n+1) = \eta(n) - (\alpha(n)m(n)/\sqrt{v(n)} + \epsilon)$$

It was observed that Nesterov's Accelerated Gradient achieved better performance than Stochastic Gradient Descent with momentum. Thus we can use Nesterov's equation to get the value of gradient which is computed differently. It incorporates the use of look-ahead factor which becomes negative when the gradient reaches the optimum value. The velocity equation for the next step is slowed down by this look-ahead factor when the slope is negative and velocity is high. The velocity reduces and slowly the model reaches convergence.

$$v(t) = \gamma v(t-1) + \eta \Delta J(\Theta - \gamma v(t-1))$$

C. Categorical Crossentropy

Categorical Crossentropy, also known as softmax loss, is softmax outputs followed by cross-entropy loss. Cross entropy loss is a logarithmic loss where each predicted class is compared to the actual class. This loss penalizes more for larger differences close to 1 and penalizes less for small differences close to 0. The categorical cross-entropy loss is given as:

$$CE = -\sum_{i=1}^{n} T_i log(P_i)$$

where T is truth table vector and P is the output vector of softmax function.

IV. Proposed Architecture

In this section, we shall discuss the architectures of MLP and CNN used for our analysis. The block diagram for our architecture is shown in Figure 1. The first half of the figure represents the MLP architecture used to obtain the highest test accuracy, the second half of the figure represents the CNN architecture. First, we must understand what inputs are understood by the Neural Network. The first step is to convert the images to greyscale and then normalize the pixel values between 0 and 1.

Once this step has been accomplished, we must convert the class labels into binary class vectors. Then, the input sequence is converted into a 1-D array which refers to Flattening and is then used to train our Multi-Layer Perceptron. We have densely connected layers which consume the flattened input and use the method of error back-propagation to minimize the loss and update the weights of the network. We employ the technique of momentum learning to enforce speedy learning and better test accuracy. Further after every hidden layer, we apply batch normalization to reduce the covariate shift between two densely connected Layers. Also, we used dropout layers after every hidden layer. The value of the dropout chosen was equal to 0.3.

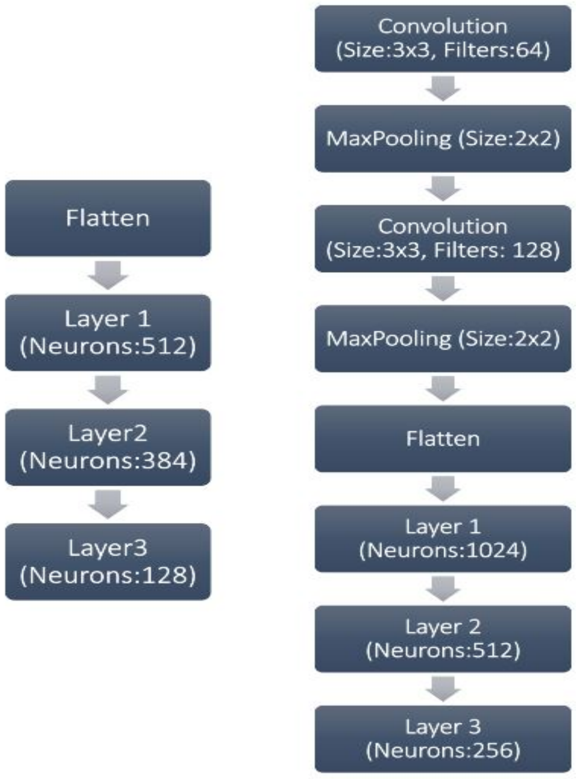


Fig. 1: Architecture for a) MLP b) CNN

Further, we discuss the architecture of our CNN model. The first step is to apply convolution with a kernel size of 3 for the processed image with 32 filters. The output from the CNN layer is fed into the Max pooling layer which reduces the dimensions of the input images by taking the maximum value for a given kernel size. This allows for faster computation. We used a kernel of 2x2 for max-pooling and the value of stride was chosen to be 2. This structure is repeated again this time with 64 filters and a similar and max-pooling layer of size 2x2. After this second pass, the inputs are flattened and converted to a 1D array, and then it acts as input for the hidden layers. We used batch normalization after every hidden layer and used dropout with values 0.1, 0.25, and 0.4 after each of the hidden layers for the model performing with the highest test accuracy. The hidden layers comprise 1024, 512, and 256 neurons which are then connected to the output layer.

V. Experiments and Results

In this section, we present the results that were achieved with our proposed architecture of Convolutional Neural Network and Multi-Layer Perceptron on the Fashion MNIST test data set. The test set is composed of 10,000 images distributed across 10 classes.

A. Multi-Layer Perceptron

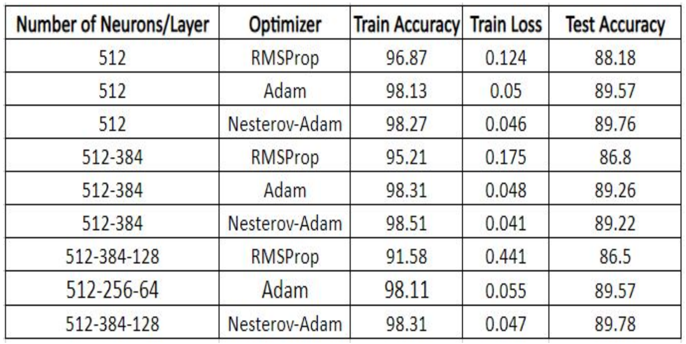
In this section, we evaluate the performance of MLP architecture for different number of hidden layer and different neuron sizes. When we consider a single hidden layer MLP, the best performance achieved for neuron sizes varying between 64 to 1024 neurons for that hidden layer was achieved by a 512-neuron layer. This choice gives our architecture around 407,050 parameters to train. We have chosen the Rectified Linear Unit as the activation function.

We achieved 89.76 percent test accuracy for this single hidden layer model with categorical cross-entropy loss function and Nesterov-Adam(N-adam) optimizer. The learning rate chosen for N-adam optimizer is equal to 0.001, the beta1 and beta2 values are equal to 0.9 and 0.999 respectively, and epsilon is equal to 1e-07.

The test accuracy achieved by RMSProp and Adam algorithms for a single 512-layer MLP was 88.18 percent and 89.57 percent respectively. the discounting factor for history gradient was chosen to be 0.9 for RMSProp. It was observed that RMSProp has a slightly weaker performance compared to Adam and N-adam.

Furthermore, we extend our architecture to a two hidden layer architecture with 512 neurons in the 1st layer and 384 neurons in the second layer. The highest test accuracy which was reported for this architecture was achieved by the Adam optimizer with the test accuracy breaching 89.26 percent. The test accuracy achieved was equal to 86.8 and 89.22 for RMSProp and Nadam respectively.

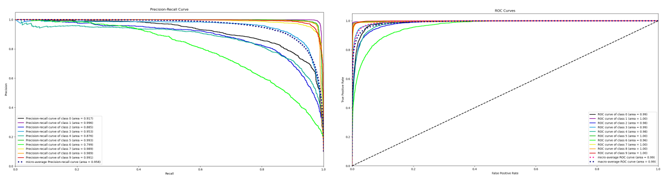
Next, we consider a three hidden layer architecture which consists of 512 neurons, 256 neurons and 128 neurons in the 1st, 2nd and 3rd layer respectively. The best test accuracy recorded was 89.78 for N-Adam.



Number of Neurons/Layer	Optimizer	Train Accuracy	Train Loss	Test Accuracy
512	RMSProp	96.87	0.124	88.18
512	Adam	98.13	0.05	89.57
512	Nesterov-Adam	98.27	0.046	89.76
512-384	RMSProp	95.21	0.175	86.8
512-384	Adam	98.31	0.048	89.26
512-384	Nesterov-Adam	98.51	0.041	89.22
512-384-128	RMSProp	91.58	0.441	86.5
512-256-64	Adam	98.11	0.055	89.57
512-384-128	Nesterov-Adam	98.31	0.047	89.78

Fig. 2: Test accuracy (Best) for MLP model

To select for the best hyper-parameters such as learning rate, batch size, activation functions and optimizers for our model, we followed the process of parameter grid search which provided us with the best training performance for different values of hyper-parameters. Our model evaluated batch sizes of 10 to 100 for different epochs. The performance

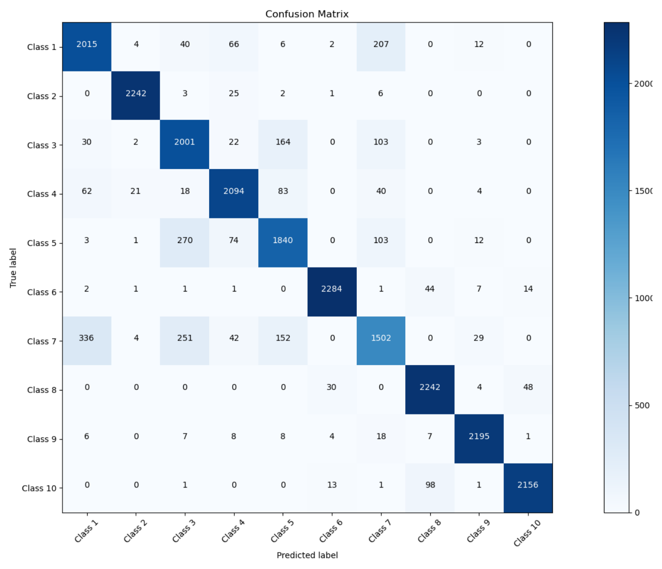


(a)

Fig. 3: a.) Precision-Recall Curve for MLP b.) ROC Curve for MLP

characteristics were acceptable for batch sizes of 40,50 and 60. Similarly, it was observed that N-adam optimizer worked with the highest test accuracy, for the best selected batch size.

Moreover, it was observed that the model reached training accuracy of the order of 95 percent within 65 epochs of training. Although, this could have been used as an early stopping criterion to halt the training, instead we ran the model for 100 epochs and then saved the model weights for each iteration and then generated test accuracy for each of the model weights. Figure 2 describes the highest accuracy recorded for different number of Hidden Layers. Figure 3a and 3b plot the precision recall curves, and Figure 4 provides us with the classification report for the highest test accuracy model.



	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Class 7	Class 8	Class 9	Class 10
Class 1	2015	4	40	66	6	2	207	0	12	0
Class 2	0	2242	3	25	2	1	6	0	0	0
Class 3	30	2	2001	22	164	0	103	0	3	0
Class 4	62	21	18	2094	83	0	40	0	4	0
Class 5	3	1	270	74	1840	0	103	0	12	0
Class 6	2	1	1	1	0	2284	1	44	7	14
Class 7	336	4	251	42	152	0	1502	0	29	0
Class 8	0	0	0	0	0	30	0	2242	4	48
Class 9	6	0	7	8	8	4	18	7	2195	1
Class 10	0	0	1	0	0	13	1	98	1	2156

Fig. 4: Confusion Matrix for MLP

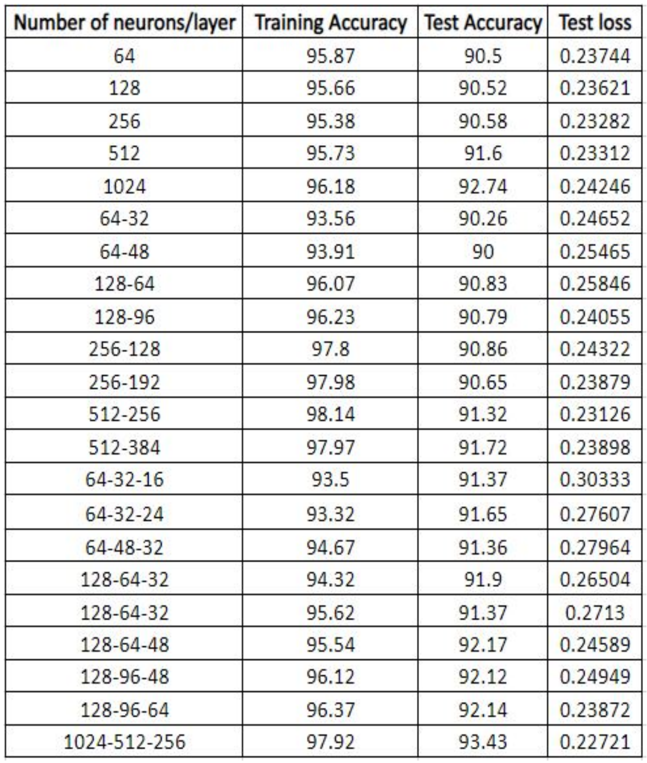
B. Convolutional Neural Network

In this section we analyze the performance of our Convolutional Neural Network architecture and it is observed that they have superior performance in comparison to MLP. The highest accuracy achieved for our model was 93.43 percent test accuracy, for a 3 hidden layer architecture. As part of model evaluation, we tested our model for upto 3 hidden layers with different neuron sizes with N-Adam optimizer.

For a single hidden layer network the model achieved the best training accuracy of 92.74 percent for 1024 neurons. The learning rate chosen is equal to 0.0001 based on our parameter grid search values. The hyper-parameters for our optimizer, beta1 and beta2 values are equal to 0.9 and 0.999 respectively, and epsilon is equal to 1e-07.

For two hidden layer network, the model achieved best training accuracy of 91.72 percent accuracy for a 512-layer densely connected with 384 neurons, which in turn is connected with the output layer. The model used the Rectified Linear Unit as the activation function for its hidden layer neurons.

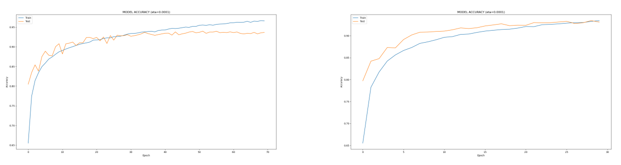
The best results were obtained for a 3-layer architecture with average test accuracy around 92 percent with the best accuracy of 93.43 percent. This model had a total of 1.8 Million parameters to train. We can observe best performing test accuracy values for different sizes and orientations of hidden layers from Figure 5.



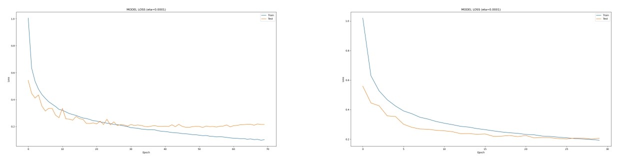
Number of neurons/layer	Training Accuracy	Test Accuracy	Test loss
64	95.87	90.5	0.23744
128	95.66	90.52	0.23621
256	95.38	90.58	0.23282
512	95.73	91.6	0.23312
1024	96.18	92.74	0.24246
64-32	93.56	90.26	0.24652
64-48	93.91	90	0.25465
128-64	96.07	90.83	0.25846
128-96	96.23	90.79	0.24055
256-128	97.8	90.86	0.24322
256-192	97.98	90.65	0.23879
512-256	98.14	91.32	0.23126
512-384	97.97	91.72	0.23898
64-32-16	93.5	91.37	0.30333
64-32-24	93.32	91.65	0.27607
64-48-32	94.67	91.36	0.27964
128-64-32	94.32	91.9	0.26504
128-64-32	95.62	91.37	0.2713
128-64-48	95.54	92.17	0.24589
128-96-48	96.12	92.12	0.24949
128-96-64	96.37	92.14	0.23872
1024-512-256	97.92	93.43	0.22721

Fig. 5: Model accuracy for different CNN architectures

To estimate the best values of the hyper-parameters we perform Grid-Search, we performed comparison between the learning rate varying between 1e-02 and 1e-05. The learning rate of 1e-04 produced the highest accuracy for the CNN model. Next, we observed the effect of batch size on model accuracy. It was observed that batch size of 50 provided us with the highest accuracy. Fig. 6a and 6b describe the Model Accuracy and Model Loss for train and test samples. It is used to establish the number of epoch it takes just before it starts to over-fit. It was observed that after 29 epochs the model started to over-fit. This provided us with an estimate to run our model for around 30 epochs, that is when the training loss is the least and training and test accuracy converge.



(a)



(b)

Fig. 6: a.) Model Accuracy vs Epoch - Over-Fit model vs Best-Fit b.) Model Loss vs Epoch - Over-Fit model vs Best-Fit

When we evaluated the model for epoch size 30, we achieved a test accuracy of 92.87 percent upon the completion of 30th epoch. Highest test accuracy of 93.43 percent was recorded for the weights generated after the 70th epoch. We can observe the confusion matrix which is presented for the most accurate model in Figure 7.

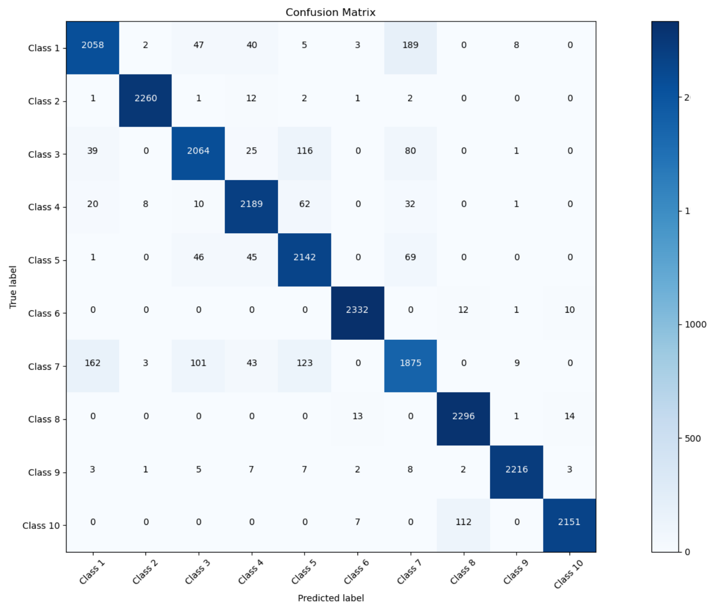


Fig. 7: Confusion Matrix

Next, to evaluate the performance of different classes we observe the Precision Recall Curves and Reciever Operating Characteristic curve for the different classes. As we can observe from Fig. 8 and Fig. 9 classes 0,2,5, and 6 fall below the micro-average PR curve. This indicates a higher presence of False Positives for these classes. The other classes performed better than the micro-average PR curve which indicates a higher true positive rate and diminished false positive rate for other classes.

Similarly, we observe for the ROC curve to establish the performance of different classes. Class 6 gives an indication of presence of False Positives. We can observe from Fig. 6 that Other than this class, the other classes have a higher true positive rate and the ROC curves for other classes appear almost ideal.

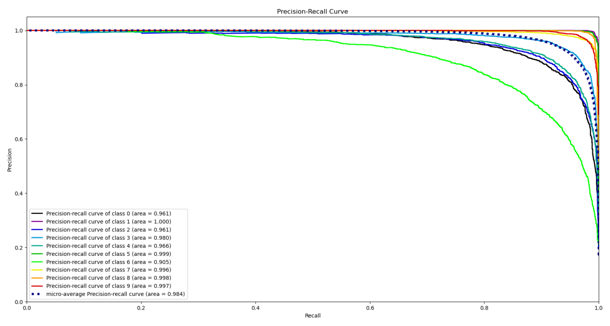


Fig. 8: Precision-Recall Curve

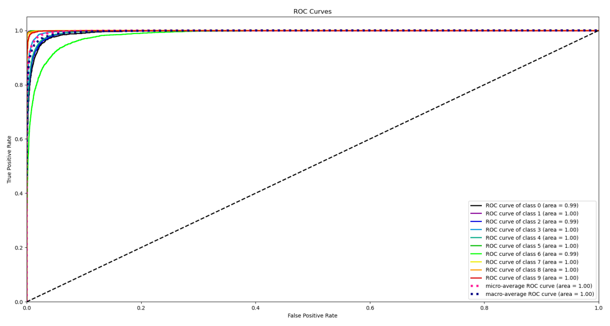


Fig. 9: Receiver Operating Characteristic

VI. Conclusion

Comparing the MLP and CNN classifiers we could conclude that CNN classifier had better classification accuracy compared to that of MLP. This is majorly due to the fact that CNNs preserve spatial correlation. CNNs reduce the dimensions of images due to convolution and max pooling operations.

Another important insight from the data is the fact that the data contains similar objects classed as different classes. The main highlight are classes 0 and 6 which have poorer accuracy performance than other classes. Similarly classes 2 and 5 have a higher number of false positives among them. One of the major reasons for it is that the resolution for images is quite low to decipher the differences between similar class labels. If the data set has higher resolution it is possible that there will more details associated with the samples and we can extract more information from it. Another way to increase accuracy is to perform Data Augmentation techniques. There are several techniques available to augment data into similar class labels which range from rotation, flipping, etc. to using Generative networks to generate similar images.

Based on our evaluation of network parameters using grid search, the best accuracy was achieved for CNN with learning rate of 0.0001. The best optimizer function compared between RMSProp, Adam and Nesterov-Adam was N-adam with the hyper parameters beta1 = 0.9 and beta2 = 0.999 and epsilon = 1e-07.

References

[1] https://github.com/zalandoresearch/fashion-mnist

[2] Xiao, Han, Kashif Rasul, and Roland Vollgraf. "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms." arXiv preprint arXiv:1708.07747 (2017).

[3] Hu, W., Huang, Y., Wei, L., Zhang, F., Li, H. (2015). Deep convolutional neural networks for hyperspectral image classification. Journal of Sensors, 2015.

[4] Neural and adaptive systems: fundamentals through simulations, JC Principe, NR Euliano, WC Lefebvre Wiley.

[5] Abdel-Hamid, O., Mohamed, A. R., Jiang, H., Deng, L., Penn, G., Yu, D. (2014). Convolutional neural networks for speech recognition. IEEE/ACM Transactions on audio, speech, and language processing, 22(10), 1533-1545.

[6] D. Ciregan, U. Meier, and J. Schmidhuber. Multi-column deep neural networks for image classification. In Computer Vision and Pattern Recognition (CVPR), IEEE, 2012.