

Machine Learning Mini-Project Report
MTCS - 204(P) Fashion MNIST Problem

Group:

Saurav Rai (17558)

Saichand A V R P (17552)

Akhilesh Pandey (17551)

AIM :

Variations of Neural Network Models for image classification: fashion-MNIST.

Experimental Procedure :

Platforms Used :

1. (Saurav Rai) : **Python3**(backend tensorflow).
2. (Saichand) : **IPython**(anaconda3), **Keras, Python** (backend Tensorflow).
3. (Akhilesh Pandey) : **Python3, Pytorch**

Dataset Description :

Fashion-MNIST is a dataset of Zalando's article images - consisting of training set of 60,000 examples and a test set of 10,000 examples. Each example is a 28x28 gray scale image, associated with a label from 10 classes. An example of how the data looks:

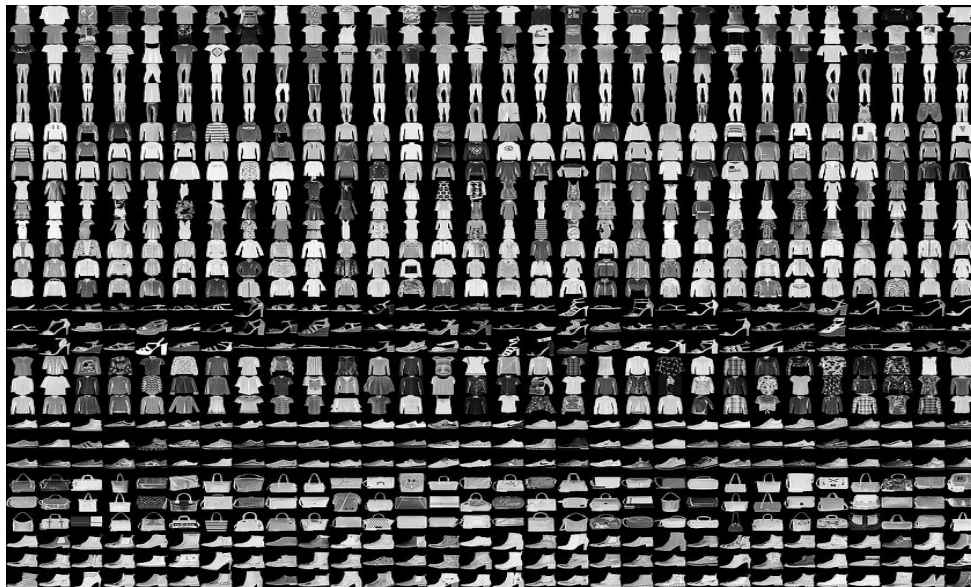


Figure reproduced from zalando's github page¹.

Each class takes three-rows in the data visualized above. 7-step conversion process is used to generate the Fashion-MNIST dataset². Fashion-MNIST poses a more challenging classification task than the simple MNIST digits data.

¹ <https://github.com/zalando-research/fashion-mnist>; ² [1708.07747v2\[cs.LG\], 15 Sep, 2017 arxiv.](https://arxiv.org/abs/1708.07747v2)

Models Used :

S.No.	Classifier	Activation	Optimizer
1	3 ConvLayers with maxpooling and 2 FC, 241546 parameters (keras)	Relu,softmax	Adam
	3 ConvLayers with maxpooling and 2 FC, 241546 parameters (keras)	Relu, tanh	Adagrad
	3 ConvLayers with maxpooling and 2 FC, 241546 parameters (keras)	Tanh, softmax	Adamax
	3 ConvLayers with maxpooling and 2 FC, 241546 parameters (keras)	Sigmoid, relu	Adam
2	MLP with one hidden layer(256), (tensorflow API)	Relu, sigmoid	Adagrad
	MLP with one hidden layer(256), (tensorflow API)	Sigmoid, Relu	Adam
	MLP with one hidden layer(256), (tensorflow API)	Sigmoid, Relu	Gradient Descent
3	Logistic Regression (python)	Softmax	LBFGS
	K-Class Logistic Regression	Softmax	liblinear

We have used 3 distinct models, with varied Activation functions and Optimization techniques. 3 Models used are:

√ **Convolutional Neural Network (CNN):** Algorithm has 3 convolution layers with different activation functions and each followed by a maxpooling phase with pool_size (2, 2), which gives translation invariance. A dropout phase is used after every maxpool phase for regularization. The algorithm uses around 2.5 lakh parameters. 1.5 lakh parameters are given as input for the first Fully Connected layer and 1290 parameters to the second Fully Connected layer.

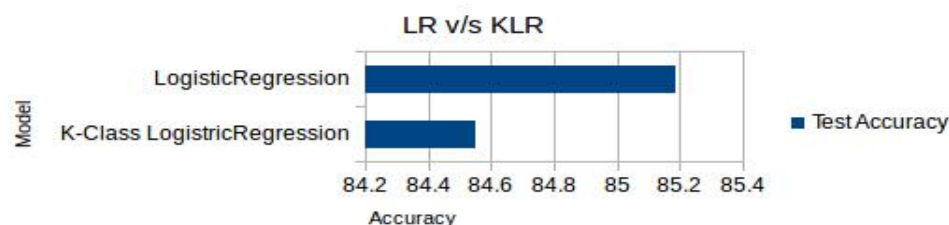
√ **Multi Layer Perceptron (MLP):** Algorithm used is a simple multi layer perceptron model with one hidden layer. Initial parameters to input node are 784. Hidden layer size is 256. The output layer has nodes equal to the number of labels(10). Weights are initialized using *random_normal* function of tensorflow package. *Softmax_cross_entropy_with_logits* is the cost function from the neural network package of tensorflow.

√ **Logistic Regression (LR):** Algorithm is implemented using the linear_model library of python-sklearn, that has LogisticRegression model. We use 'lbfgs' solver as the optimizer for the model, and softmax function for predicting probabilities. We also

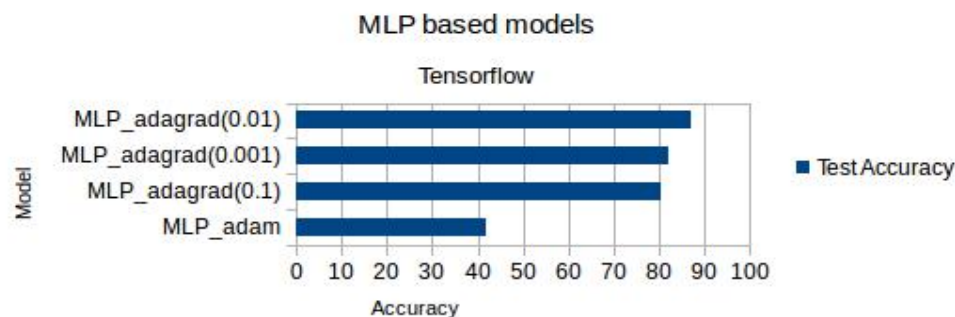
implemented the k-class LR model with *multinomial = 'ovr'*, which fits a binary problem for each label. The default optimizer used for this model is 'liblinear' which is an open source library for large-scale linear classification.

Experiments Conducted:

- ❖ At first, we dealt with the fashion-mnist image classification problem using Simple **Logistic Regression(LR)** to classify images, which are saved as a features in a .csv file for train and test data. A fixed train set (60,000) and a fixed test set (10,000) are used for the model as given by the creators of the fashion-MNIST (zalando). The accuracies observed in this model are not satisfactory. We observe that LR works better than k-class LR, which uses ovr(one versus rest) policy. But LR gives only 85.19 and KLR(K-Class LR) gives 84.55.

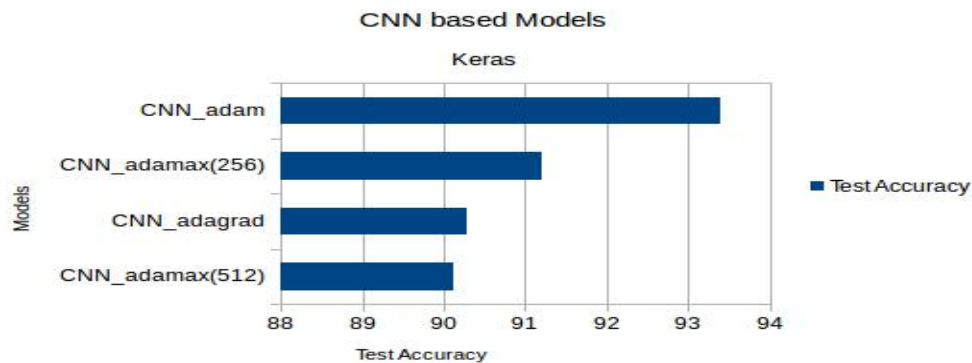


- ❖ Next, we tried to approach the problem using Simple Multi-layer Perceptron Model. We varied the optimizers and also activation functions at different layers. We noted down the accuracies for all the models and projecting the best few models that stood out in the analysis. The test accuracy has gone up till 87.01 for Adagrad optimizer with a learning rate of 0.01. But, this doesn't seem to impress because it takes enough time about half an hour, but improves by 3 percent. In the figure, by accuracy we mean the test accuracy. We even tried the Gradient Descent Technique, due to its inability of oscillation, it converged to local optima and the test accuracy was only 35.

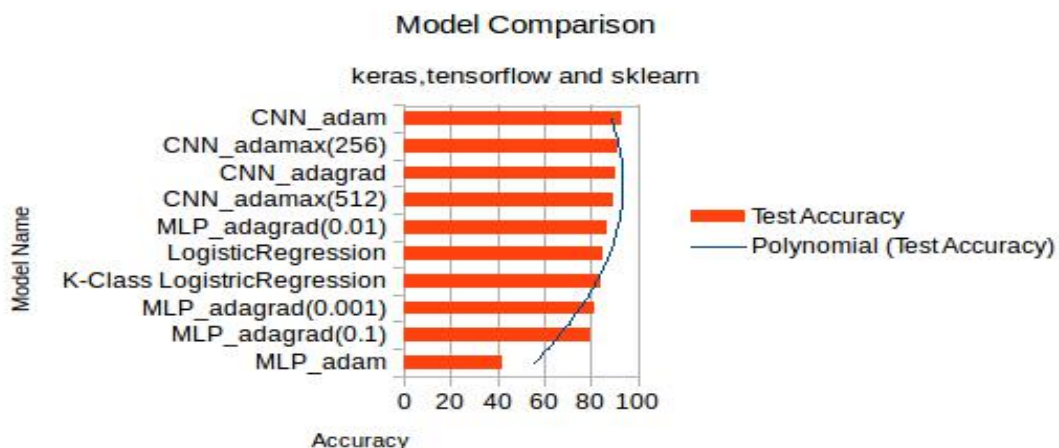


- ❖ Then we used CNN using keras with tensorflow backend, which showed good improvement in the accuracy. The maximum test accuracy we touched was 93.4, with adamax optimizer. We say that CNN based model performed better, based on

the model and also the time taken to execute was only 15 minutes compared to double the time taken for MLP and even more for LR models.



The Comparison for all the three different models shows that CNN based models outperform other models, due to the ability of multi-layer processing. The hyper parameters of the models also have been tuned to see the accuracy improve in case of both MLP and CNN models.



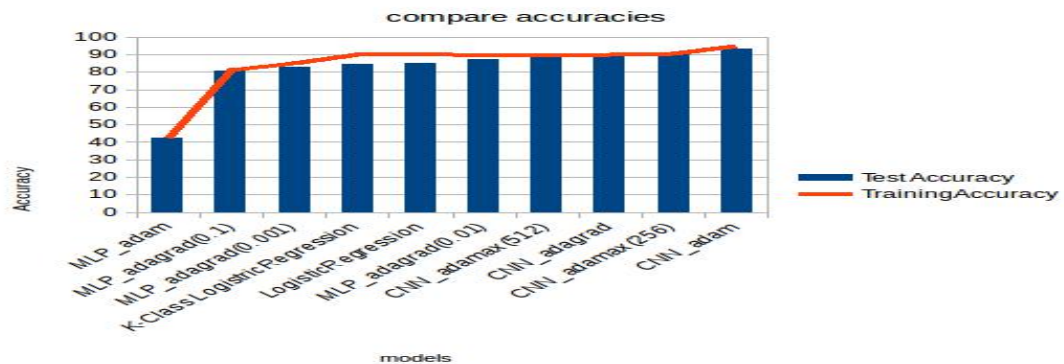
The figures 256 and 512 in CNN_adamax model, correspond to the batch size. ROC curves for the models have been plotted and the training and test accuracies are noted down. In the above figure, the polynomial trend line shows the shift in the test accuracy.

Observations:

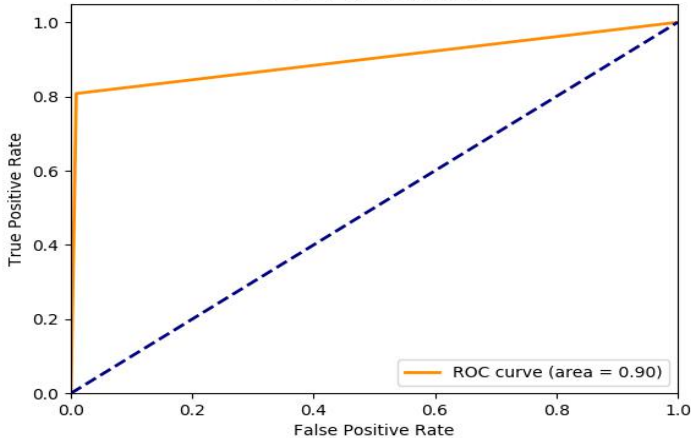
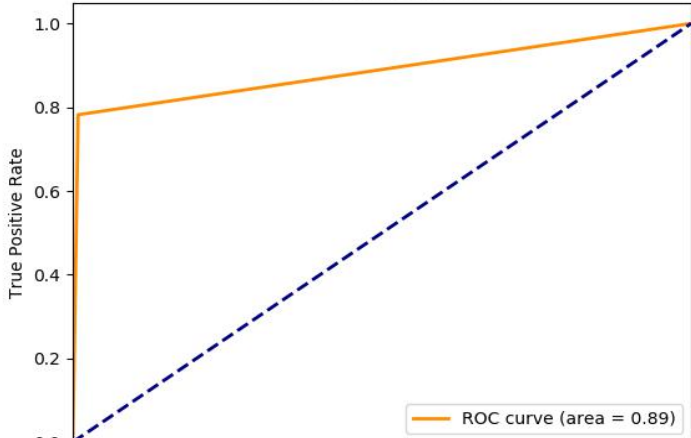
The observations on the experimental models are as follows:

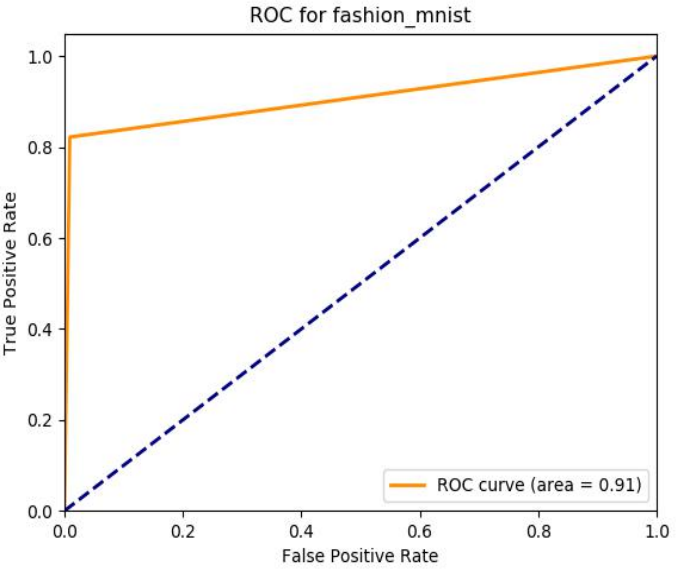
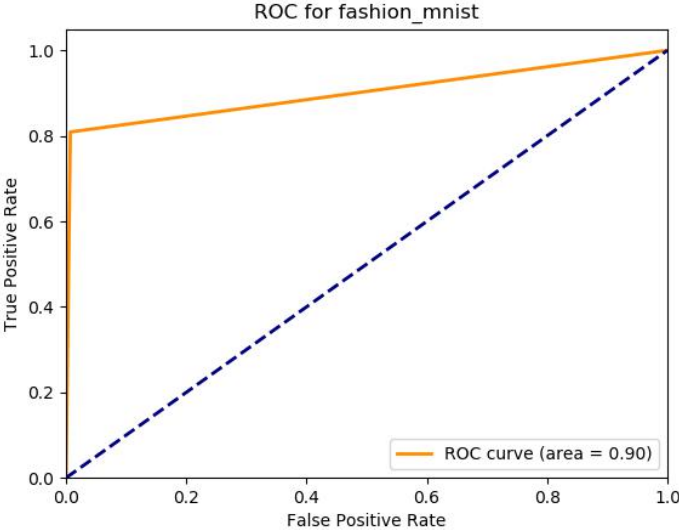
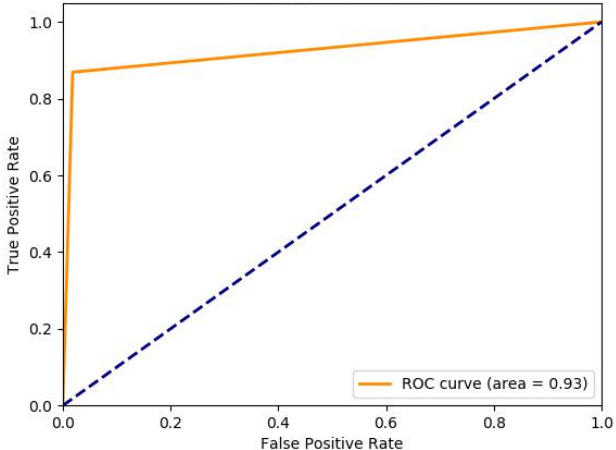
The training and test accuracies for most of the models seem to be close, and in fact for MLP and LR models, the test accuracy is less than the training accuracy. This indicates that the models when trained on themselves(training set) are better than when ran on the test data. Hence, the generalization error is more. Also, the training accuracies

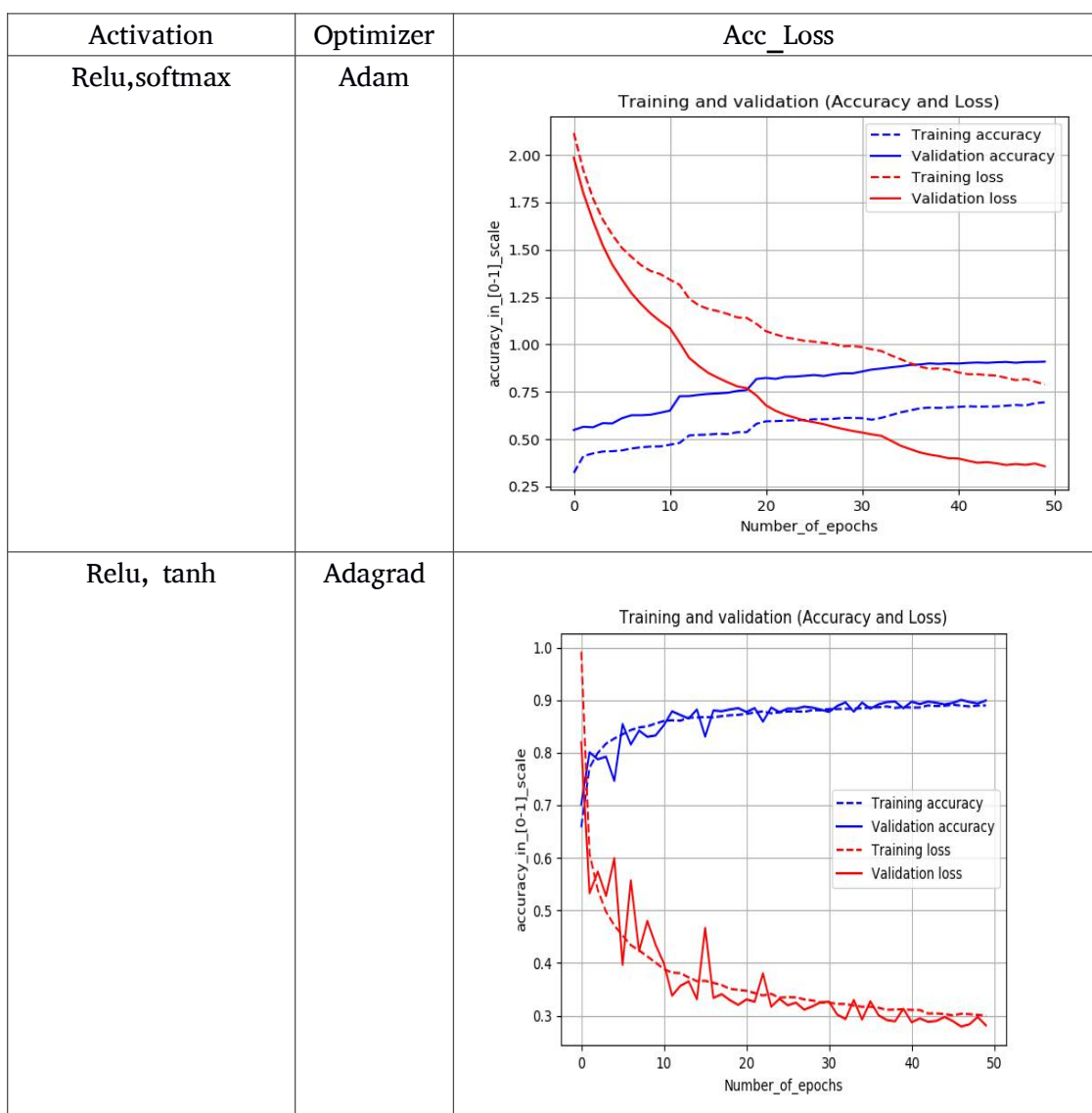
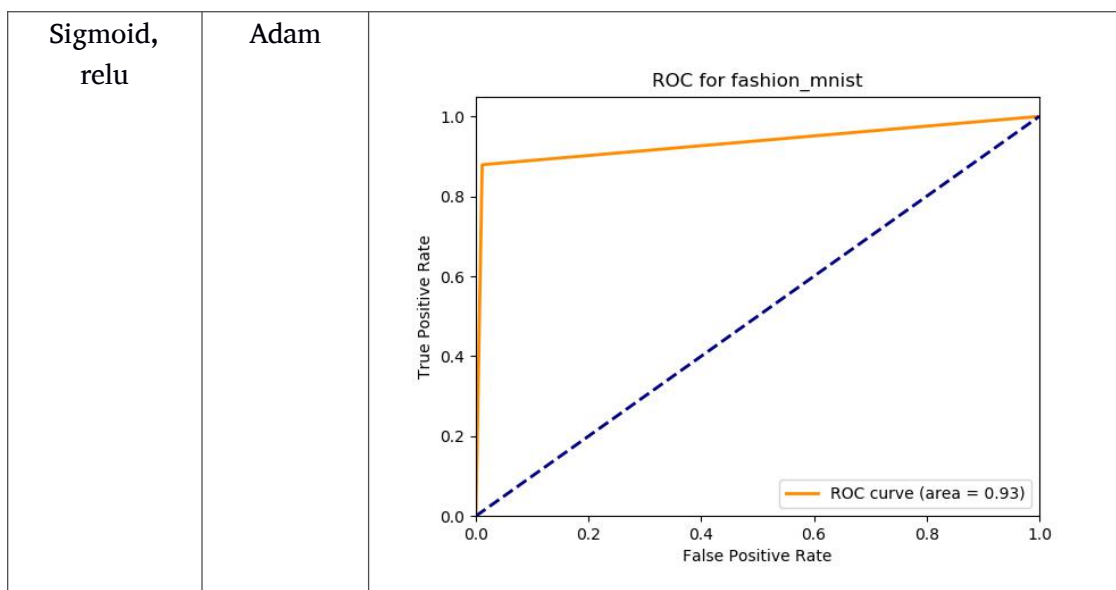
are all in the range of 85-95. Hence the models are not overfitting the training data. For different models we have plotted the ROC curves, given after the comparison of accuracies. The ROC curves also show that the area under the curve is maximum for CNN based model, with adam optimizer.



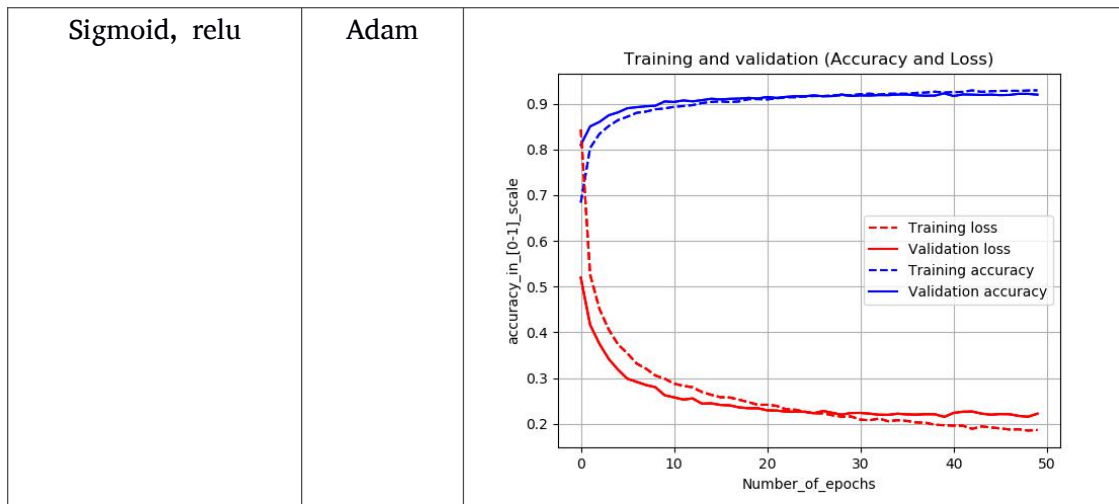
ROC and Accuracy, Losses for CNN Models :

Activation	Optimizer	ROC
Relu,softmax	Adam	 <p>ROC for fashion_mnist</p> <p>True Positive Rate</p> <p>False Positive Rate</p> <p>ROC curve (area = 0.90)</p>
Relu, tanh	Adagrad	 <p>ROC for fashion_mnist</p> <p>True Positive Rate</p> <p>False Positive Rate</p> <p>ROC curve (area = 0.89)</p>

Relu,tanh	Adagrad (0.01)	 <p>ROC for fashion_mnist</p> <p>True Positive Rate</p> <p>False Positive Rate</p> <p>ROC curve (area = 0.91)</p>
Tanh, softmax	Adamax (256)	 <p>ROC for fashion_mnist</p> <p>True Positive Rate</p> <p>False Positive Rate</p> <p>ROC curve (area = 0.90)</p>
Tanh, softmax	Adamax (512)	 <p>ROC for fashion_mnist</p> <p>True Positive Rate</p> <p>False Positive Rate</p> <p>ROC curve (area = 0.93)</p>



Relu,tanh	Adagrad (lr = 0.01)	<p>Training and validation (Accuracy and Loss)</p>
Tanh, softmax	Adamax (256)	<p>Training and validation (Accuracy and Loss)</p>
Tanh, softmax	Adamax (512)	<p>Training and validation (Accuracy and Loss)</p>



Inferences:

From our analysis, we infer that CNN boosts the accuracy, with less computation time, compared to other classifiers because of the factor of multi-layer processing. Also the validation phase of the CNN models help in boosting the accuracy. Because we split the training and validation data as 80 ad 20 percent, from the training data set. The area under the curve for CNN with adam and adamax optimizers outstands at 93. We haven't experimented thoroughly with variations of Batch_size for the convolutional neural networks. Also, there is no much work done in MLP also. May be we can increase the hidden layers and improve the accuracies. The changes in the learning rates for CNN, MLP have showed some intuitive understandings as to, if lr is low (0.001), it takes more time and also learns very slow. If lr is too high (0.1, 0.2..), it halts much early, but drastic changes lead to more bias and reduction in accuracy can be observed. Hence, we found that a good learning rate for CNN with adagrad and adam is 0.01. But, strangely, for CNN with adamax optimizer, we see that learning rate of 0.2 gave 93 percent test accuracy.

Conclusion:

In this mini-project, We have done an image classification for the fashion-MNIST dataset. We have used various classifiers such as CNN, MLP and LR. We have further studied the accuracies and concluded that with our minimal experiments, we observe that CNN performs better. We can extend the work to improve the performance by tuning the batch_size for CNN, number of hidden layers for MLP and better novel solver for Logistic Regression.

