Project 2
Deep Learning E0 250 (CSA)
Classification of FashionMNIST Dataset
Multi - Layered NN and Convolutional NN

Rahul John Roy (M.Tech CDS) - 16703

1. The Problem.

This project is to implement neural network and convolutional neural network for the task of image classification. The classification task will be that of recognizing an image and identify it as one of ten classes. We are required to train the classifiers using Fashion-MNIST clothing images.

2. Structure.

Configuration used:

- Python 3.6.9
- Pytorch 1.4.0

The following files will be part of this assignment:

- main.py - The main file of this project. It downloads the test dataset from the inbuilt APIs in torch and tests it on pre-trained models that have been saved. It generates two output files - multi-layer-net.txt and convolution-neural-net.txt. These have the output classes predicted by the two networks and also the accuracy of each model in the testing phase. It creates the testing model by importing the corresponding class from the two python files MLNN.py and CNN.py.
- MLNN.py - This contains the implementation of the multi layered neural network using Pytorch and also the training code used to train on the dataset considered. The model is finally saved in the folder Models/MLNN. These can be loaded in the main.py during the testing phase.
- CNN.py - This contains the implementation of the convolutional neural network using Pytorch and also the training code used to train on the dataset considered. The model is finally saved in the folder Models/CNN.
- The model names have the parameters of the model and the parameter variables are used to load the same from the respective folders.

3. Methodology.

- **Multi-layered Neural Network.**

The input layer gets a vector of size 784 (28 * 28). This is the image which has a dimension of 28 x 28, resized into a one dimensional vector. This is a classification problem with the output fitting into ten different classes, so the output layer will have 10 nodes in it. The output is the probability vector of the image belonging to the corresponding 10 classes.

The optimizers were chosen as Adam and RMSProp for this project. Adagrad was tried out in the previous assignment. Adam and RMSProp were found to be far superior to Adagrad. So Adagrad has been tried out only in the trials of CNN. The learning rates have also been kept constant at 0.001 for the optimizers,as this was found to be optimum in the previous assignment for multi - layered NN.

- **Convolutional Neural Network.**

Two convolutional layers were set up. The filter size was kept at 5 and stride of the filters were kept at 1. Padding was used on both sides such that image size after the filter convolution is not compromised. Two max pooling layers after each of the convolutional layers with a step of 2 was initiated. The activation functions used were Relu. A single hidden linear layer was added after the final max pooling layer and then the output layer with number of nodes as 10 was added.

The optimizers chosen were Adam, RMSProp and Adagrad. The learning rates were varied and the results have been shown in the next section.

4. Results and Observations.

- **Multi layered Neural Network.**

In the multi - layered NN, the parameters that were varied were following

a. The number of epochs in training
b. Data Augmentation with varying the number of neurons in the hidden layer

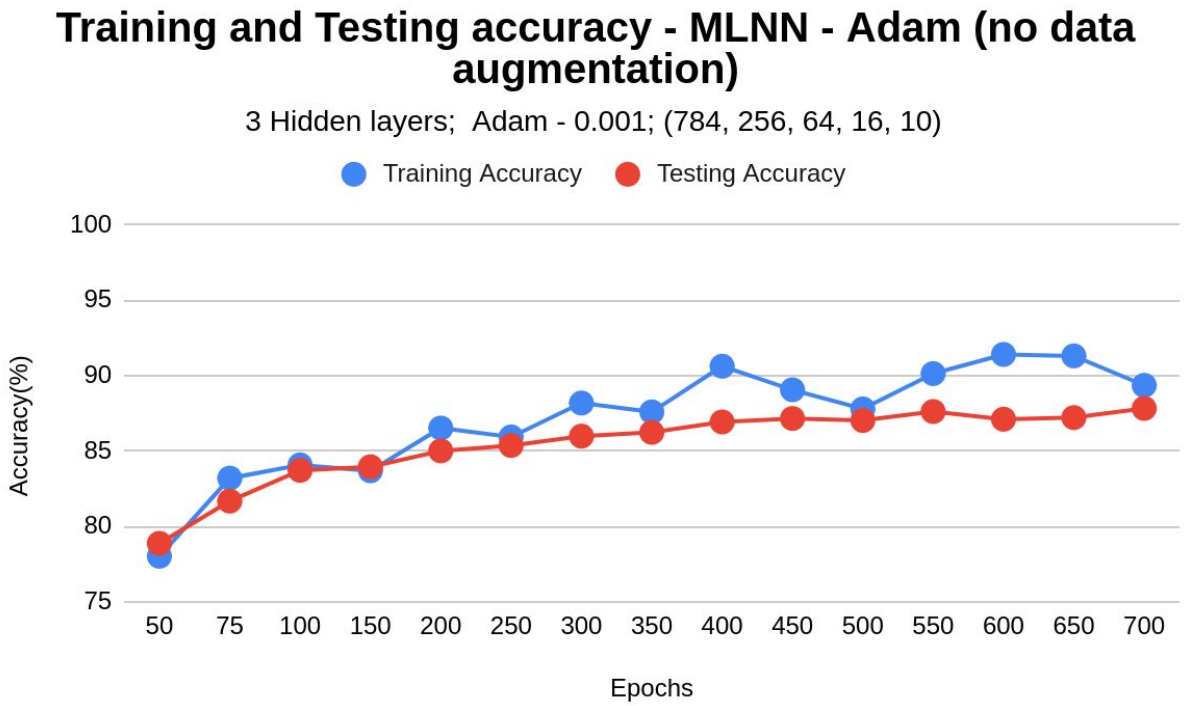
➢ **Trial 1.**

Parameter perturbed: No. of epochs with no data augmentation
No. of hidden layers: 3
Optimizer: Adam
Loss function: Categorical cross entropy
Learning Rate: 0.001
Number of epochs: 50 to 700
Data augmentation used: NO

Observation:

It can be seen that, with 50 epochs we get around 77% training accuracy and as the number of epochs increases, the accuracy also increases, to taper off around 88%. Successive increase in the number of epochs has very little effect on the training accuracy. The same observation can be derived from the testing accuracy in the initial stages. In the final trials, max testing accuracy was observed around 600 (91%), and then decreases below 90%.

This may be explained due to the overfitting that might happen due to the large number of epochs.

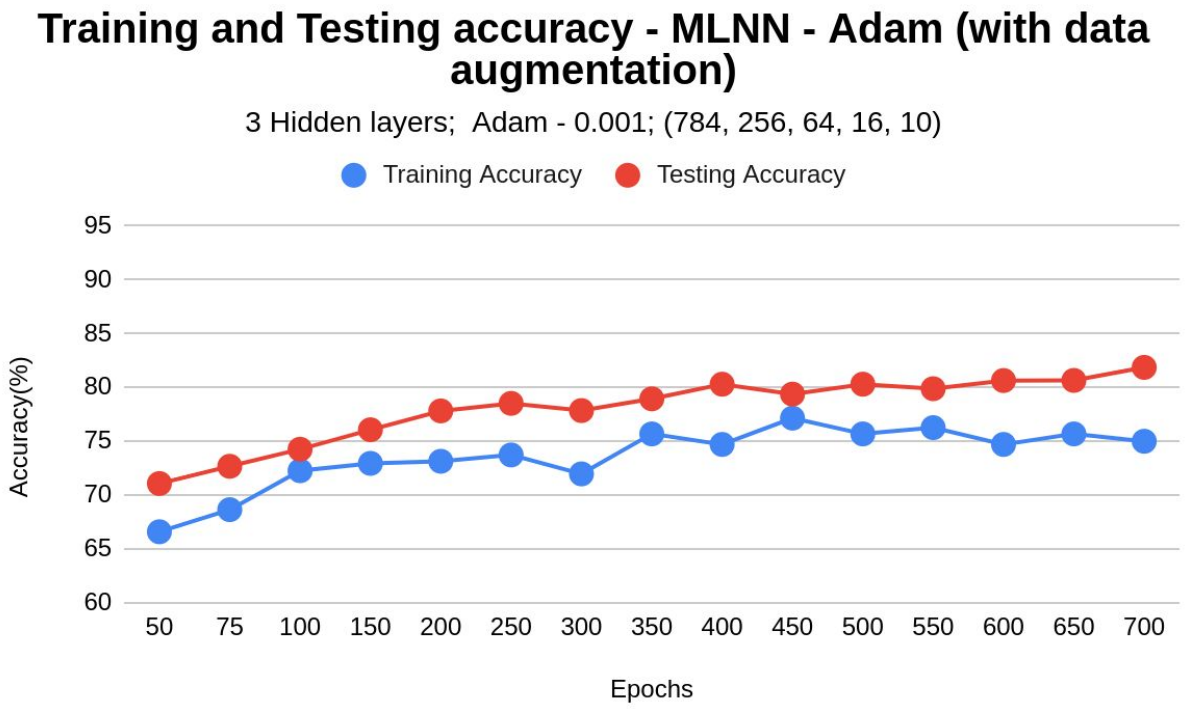


➢ **Trial 2.**

Parameter perturbed: No. of epochs with data augmentation
No. of hidden layers: 3
Optimizer: Adam
Loss function: Categorical cross entropy
Learning Rate: 0.001
Number of epochs: 50 to 700
Data augmentation used: YES

Observation:

We added a Gaussian noise of zero mean and variance 1 to the images in the dataset. As we did the trial, we could see that the model could not perform above a given mark in the training phase. This may be due to the fact that the added noise was restricting the model from over-fitting.
But contradicting what we saw in the last trial, the test accuracy saw a consistent increase over the epoch change, but it was fairly low compared to the trial where data augmentation was not used.



➢ **Trial 3.**

Parameter perturbed:

1. Number of epochs
2. Number of hidden layers
3. Number of neurons in the hidden layer

No. of hidden layers: 3&4
Optimizer: Adam
Loss function: Categorical cross entropy
Learning Rate: 0.001
Number of epochs: 50 to 700
Data augmentation used: YES

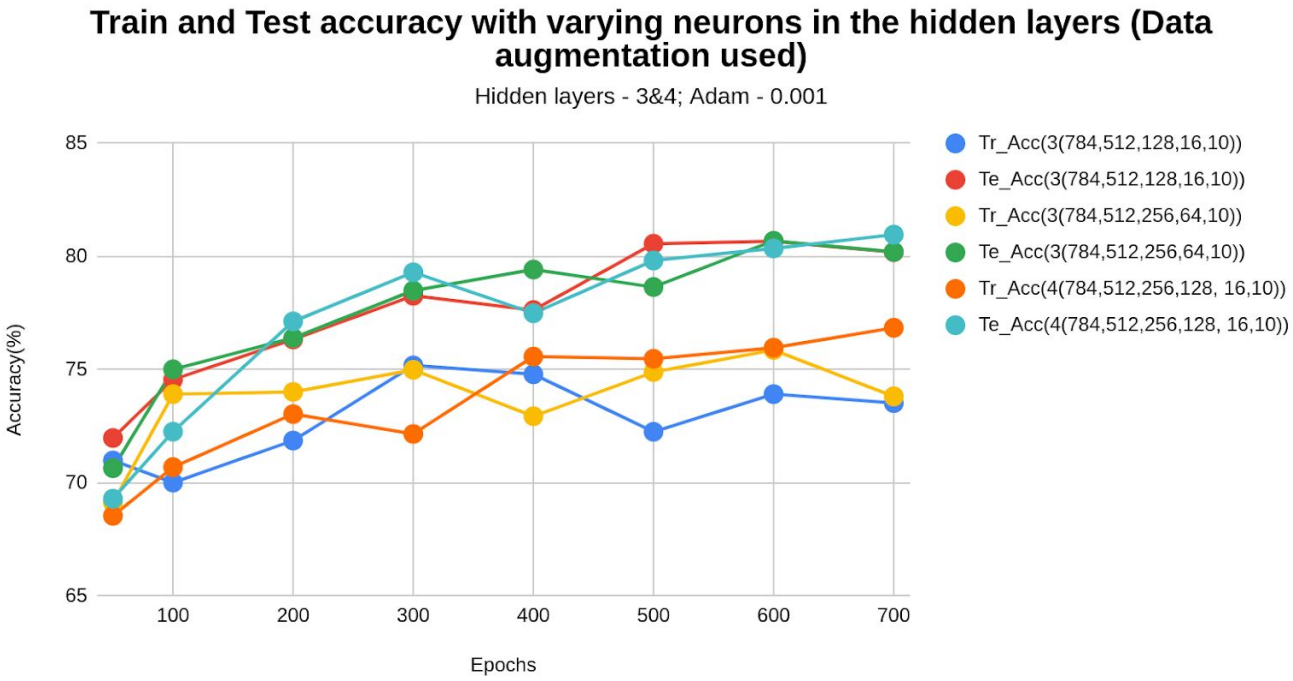
This trial has included data augmentation with Gaussian noise. The optimizer used was Adam. RMSProp was also tried out, but the results reported are for Adam. For the trial without Data augmentation, the results reported are with RMSProp as the optimizer.

Notation:

a. Tr_accuracy - training accuracy
b. Te_accuracy - testing accuracy
c. The first number in the parenthesis denoted the number of hidden layers and the subsequent numbers indicate the number of neurons in the model, the input and the output layer included.

Observation:

We can see that the data augmentation is helping the model from overfitting on the training set. Therefore the training accuracy is on the lower side, around 75%. The testing accuracy of the three models tried out are also reporting similar trends as before.
The green line (Testing accuracy for (3(784,512,256,64,10)) demonstrates consistency than others. In the other two models, we can see a drop in the accuracy in the 400th epoch.



➢ **Trial 4.**

Parameter perturbed:

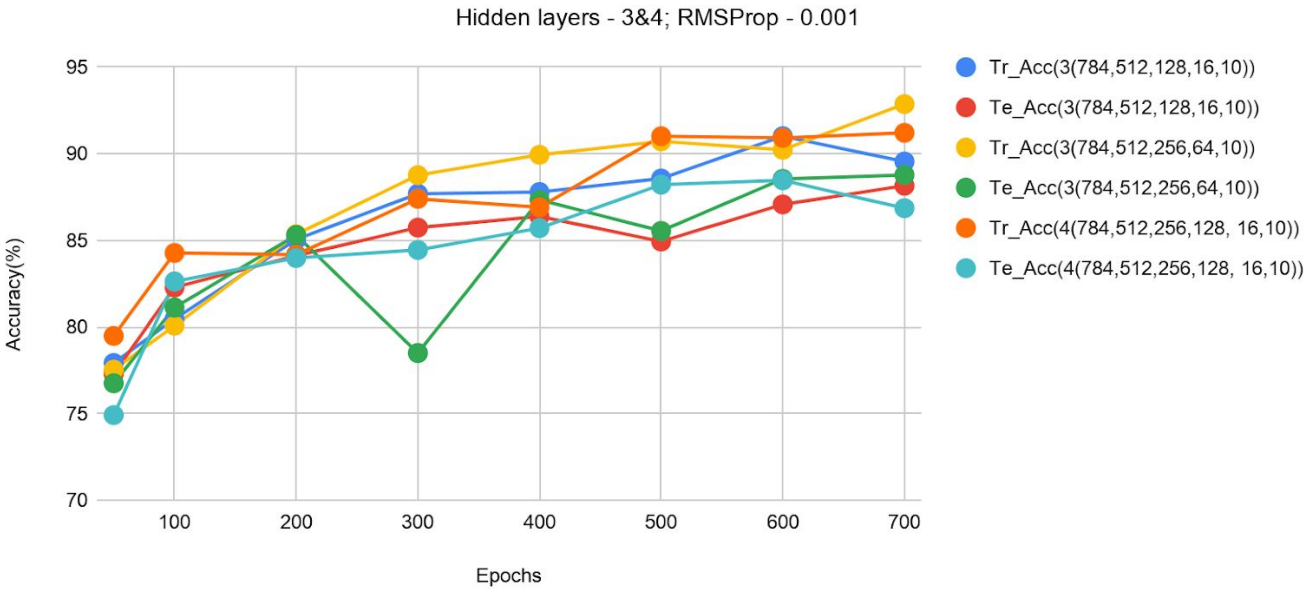
a. Number of epochs
b. Number of hidden layers
c. Number of neurons in the hidden layer

No. of hidden layers: 3&4
Optimizer: RMSProp
Loss function: Categorical cross entropy
Learning Rate: 0.001
Number of epochs: 50 to 700
Data augmentation used: NO

The following trial has been carried out without data augmentation. The optimizer used was RMSProp with a learning rate of 0.001.

Observation:
We can see good increase in both the training and the test accuracy. The maximum test accuracy was observed for the model with configuration (3(784,512,256,64,10)), around 92.8%. As the number of epochs are increased, we can see a taper off in both the train and the test accuracy.

Train and Test accuracy with varying neurons in the hidden layers (Without Data augmentation)

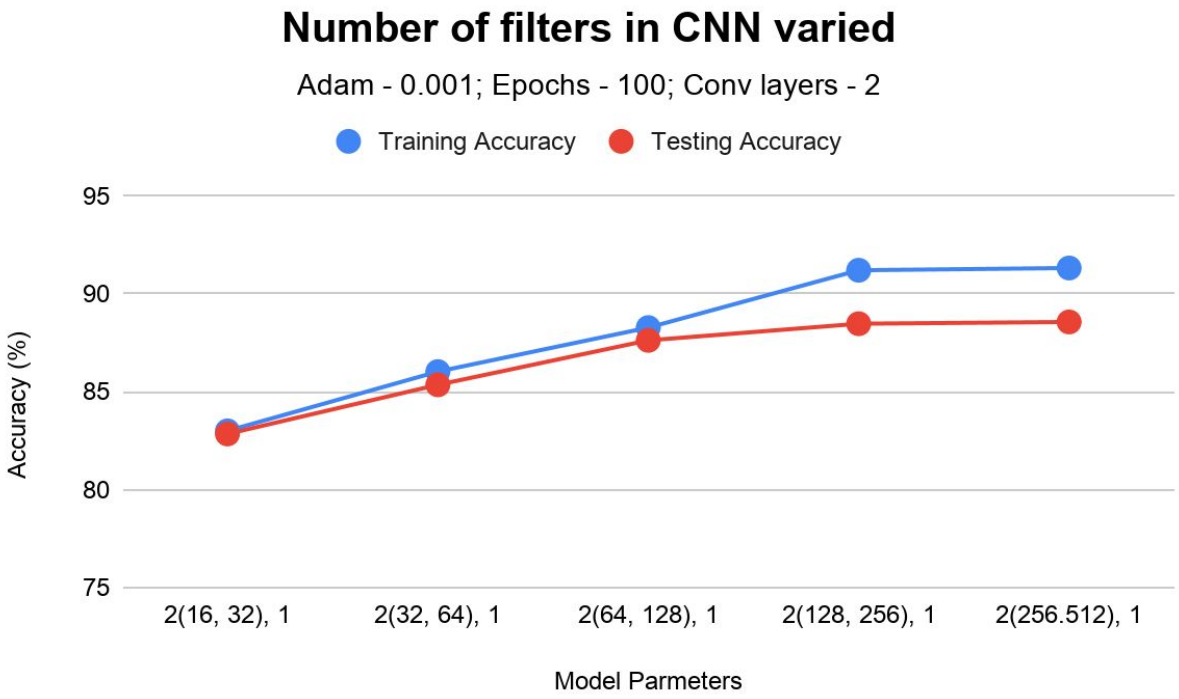


- **Convolutional Neural Network.**

➢ **Trial 1.**

Parameter perturbed: No. of filters in the convolutional layer
Filter size, stride, padding: 5, 1, 2
No. of hidden layers: 2 convolutional layers and 1 densely connected
Optimizer: Adam
Loss function: Categorical cross entropy
Learning Rate: 0.001
Number of epochs: 100
Data augmentation used: NO

Observation:
The number of filters in the convolutional layers were varied. We can see that there is an increase in both training and the testing accuracy. But after the number of filters get above 128 and 256 respectively in the two convolutional layers, we can see that the accuracy tapers off.
This may be due to the fact that there is no more information to learn from the images after the max pooling operation.

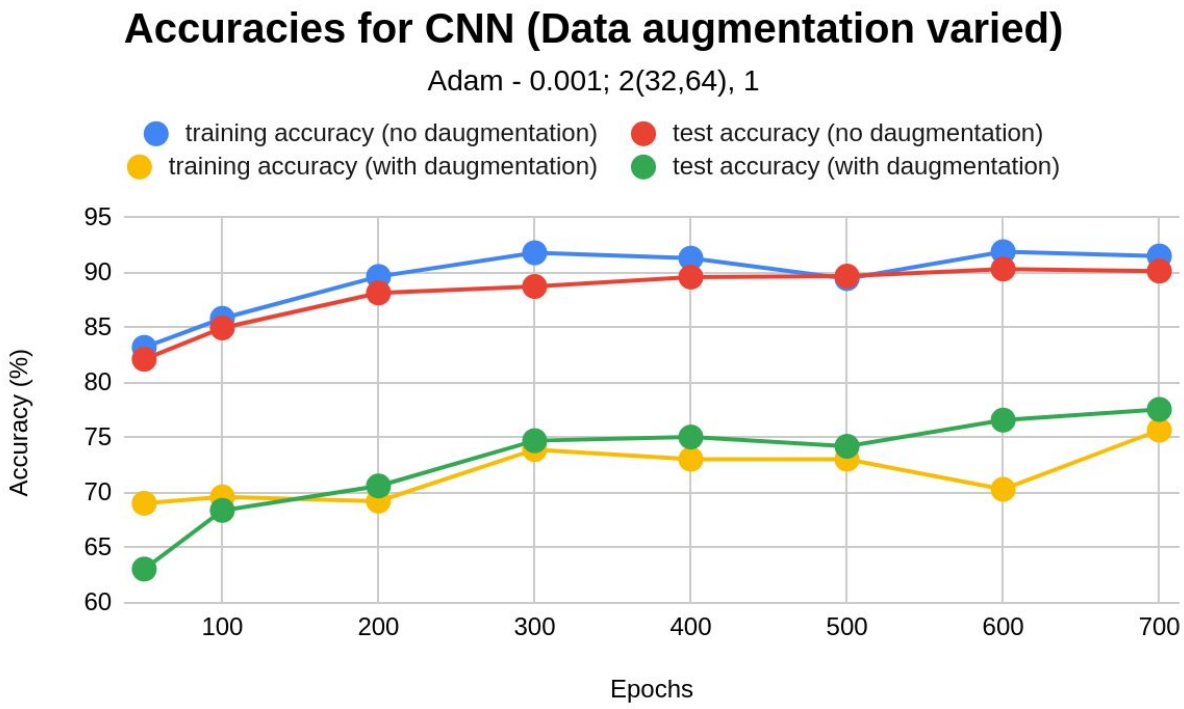


➢ **Trial 2.**

Parameter perturbed: Epochs and Data Augmentation
Filter size, stride, padding: 5, 1, 2
No. of hidden layers: 2 convolutional layers and 1 densely connected
Optimizer: Adam
Loss function: Categorical cross entropy
Learning Rate: 0.001
Number of epochs: 50 - 700
Data augmentation used: both

Observation:
We can see that, as observed in the multi - layered NN, the addition of noise to the training dataset, does not allow the model to raise beyond an accuracy level. THe training and the test accuracy of the data augmented model is well below the normal one. We can see the test accuracy taper off in the noise - added model, but the test accuracy of the augmented model is increasing with progress in the epochs.

The tapering off of the accuracy in testing and training can denote the saturation of the features that can be learned with just increasing the epochs. Increasing the number of filters in the convolutional layers might help in this regard.
Below we can see the above trial with RMSProp as the optimizer.

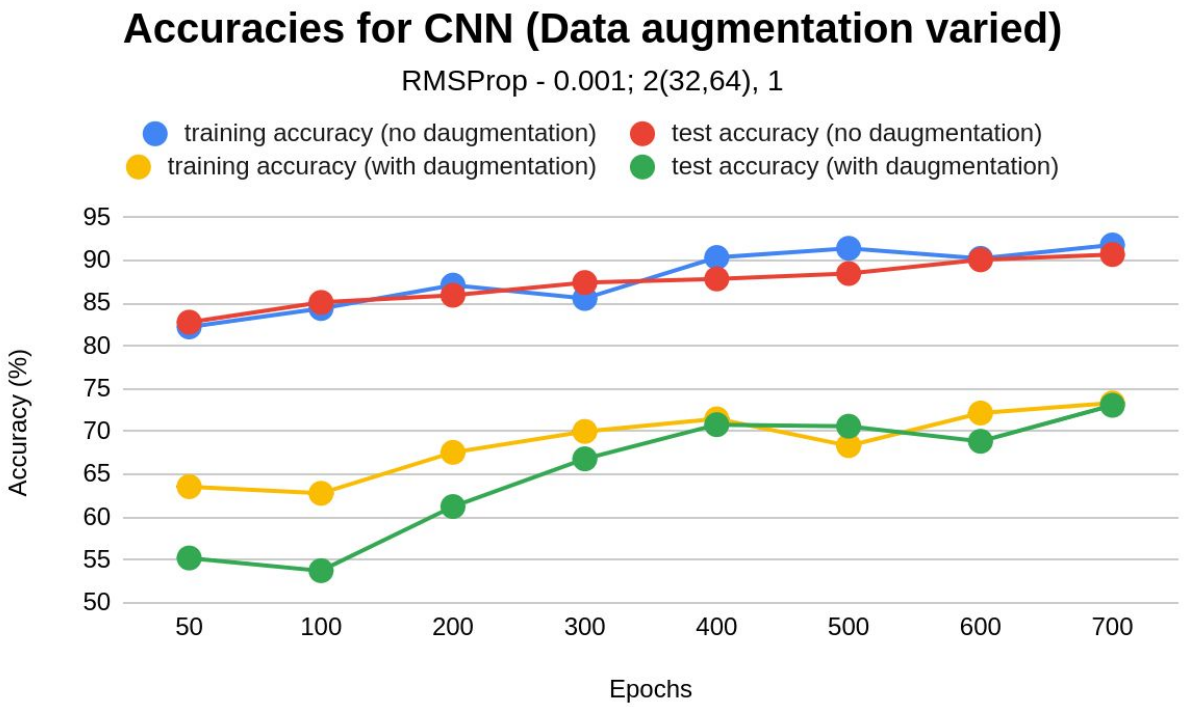


➢ **Trial 3.**

Parameter perturbed: Epochs and Data Augmentation
Filter size, stride, padding: 5, 1, 2
No. of hidden layers: 2 convolutional layers and 1 densely connected
Optimizer: RMSProp
Loss function: Categorical cross entropy
Learning Rate: 0.001
Number of epochs: 50 - 700
Data augmentation used: both

Observation:

We can observe similar results as the above trial by varying the epochs and the inclusion of data augmentation. The accuracy saturates after 500 epochs and the accuracy in both the training and testing phases are decreased in the models where the Gaussian noise is added to the model.



➢ **Trial 4.**

Parameter perturbed: Learning rate and optimizers (Adam, RMSProp and Adagrad)
Filter size, stride, padding: 5, 1, 2
No. of hidden layers: 2 convolutional layers and 1 densely connected
Loss function: Categorical cross entropy
Number of epochs: 50 - 700
Data augmentation used: NO

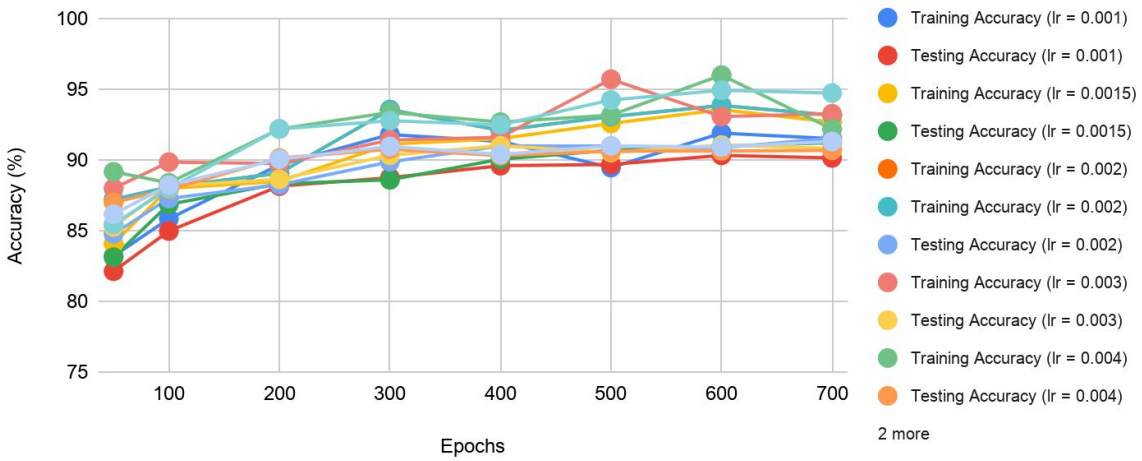
Observations:

Below are the graphs plotted with the testing and the training accuracy for Adam, RMSProp and Adagrad as the optimizers. The learning rate has been varied and has been increased from 0.001 to 0.005.

Adagrad provided the least accuracy of the three. Adam and RMSProp are consistent and give almost the same results in every trial. Also we can note that the accuracy is tapering off in all three cases.

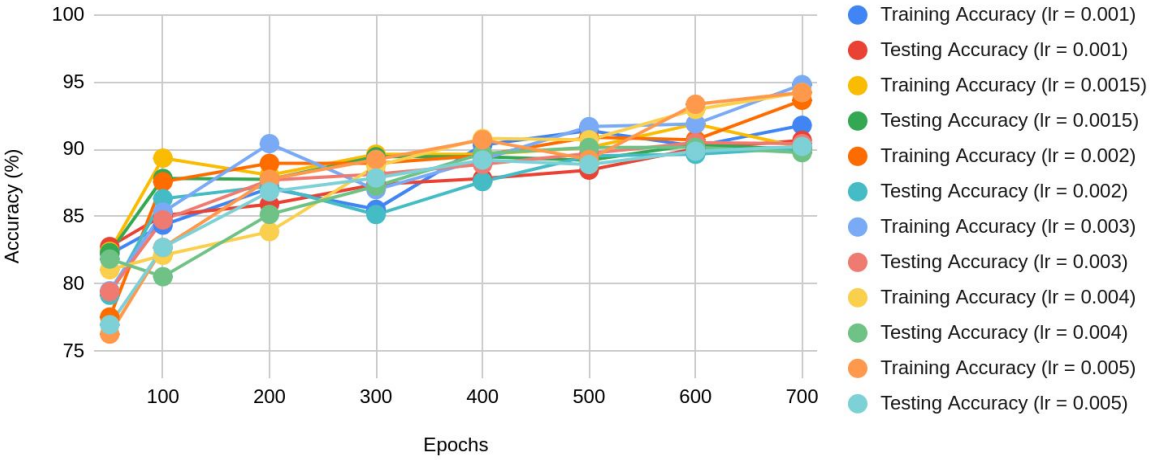
Accuracy with varied learning rates - Adam

Adam; 2 conv (32,64), 1 dense



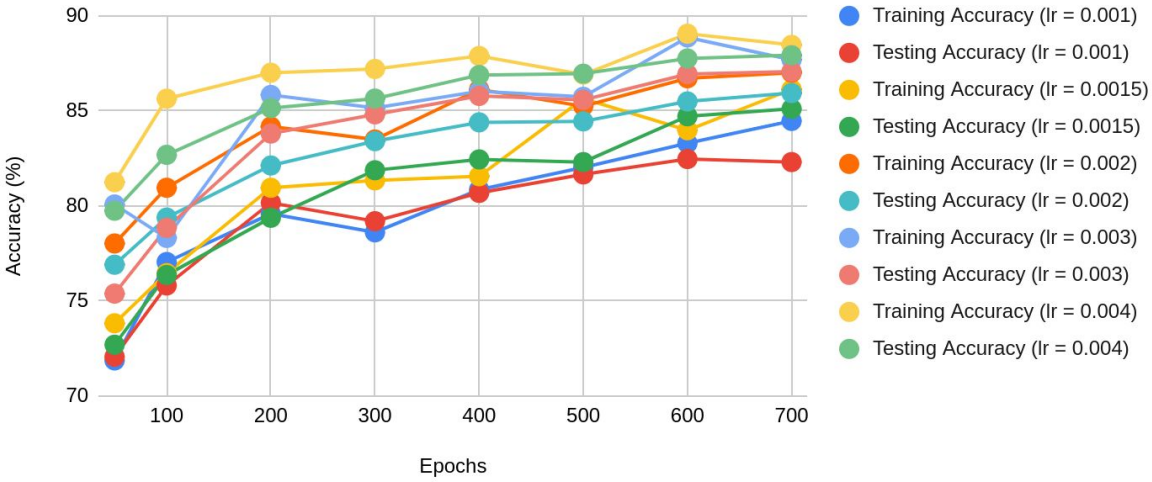
Accuracy with varied learning rates - RMSProp

RMSProp; 2 conv (32,64), 1 dense



Accuracy with varied learning rates - Adagrad

Adagrad, 2 conv (32, 64), 2 dense



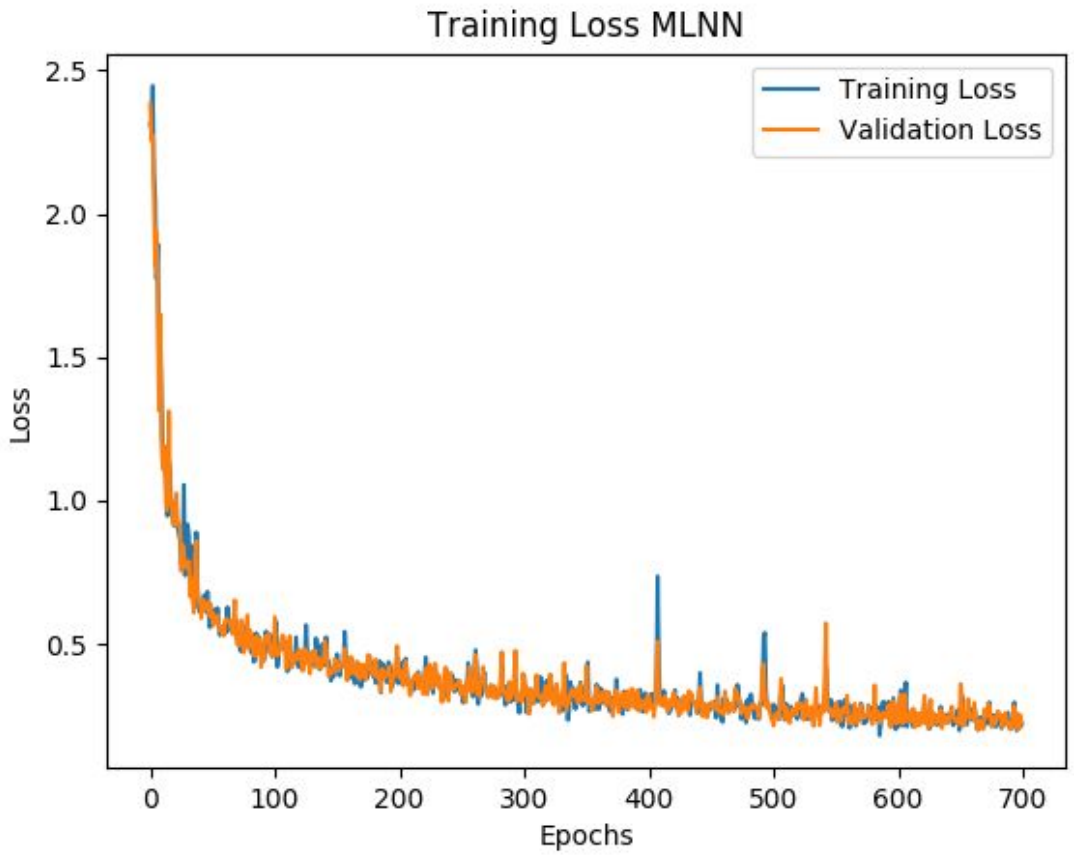
5. **Conclusions.**

- **Multi - Layered NN:**
  Parameters.
  - ★ No. of hidden layers: 3 (512, 256, 64)
  - ★ Optimizer: RMSProp
  - ★ Loss function: Categorical cross entropy
  - ★ Learning Rate: 0.001
  - ★ Number of epochs: 700
  - ★ Data augmentation used: NO

  Reason for choosing the model:

As we can see from the learning curve below, the training and the validation curves are similar to each other. This indicates that there is no overfitting in the model. Based on the trials, the model trained with 700 epochs, performed the best both in the training and the testing phase. Increasing the number of hidden layers to 4, did not have much effect. So for the final model, 3 hidden layers were selected. Regarding the optimizer, RMSProp was selected, since this was found to be similar to Adam. The default learning rate of 0.001 was chosen.

Training Accuracy: 92.8711
Testing Accuracy: 88.5%
Loss on Test Data : 0.32660067081451416



- **Convolutional NN:**
  Parameters.
  - ★ Number of filters in the convolutional layer - 32, 64
  - ★ Filter size, stride, padding: 5, 1, 2
  - ★ No. of hidden layers: 2 convolutional layers and 1 densely connected
  - ★ Optimizer: Adam
  - ★ Loss function: Categorical cross entropy
  - ★ Learning Rate: 0.002
  - ★ Number of epochs: 50 - 700
  - ★ Data augmentation used: NO

  Reason for choosing the model:

In CNN too, we can see that both training and validation curves follow the same trend. Validation curve is found to have lower losses than the test curve as the epochs progress. So the highest epoch 700 was chosen for the final model. This in turn indicates that there is no overfitting in the model. The number of filters in the two convolutional layers were fixed at 32 and 64 respectively. Optimizer was chosen as Adam with a learning rate of 0.002. This was chosen after a number of trials as mentioned in the observations. Data augmentation was not used in the final model reported.

Training Accuracy: 93.9453
Testing Accuracy: 91.18%
Loss on Test Data : 0.2725245952606201