Deep Learning- CSE641

Assignment - 2 [Part 1]

Name: Akanksha Shrimal **Roll No:** MT20055
Name: Vaibhav Goswami **Roll No:** MT20018
Name: Shivam Sharma **Roll No**: MT20121

FILES SUBMITTED : submitted .py file , .ipynb file and readme.pdf and output.pdf

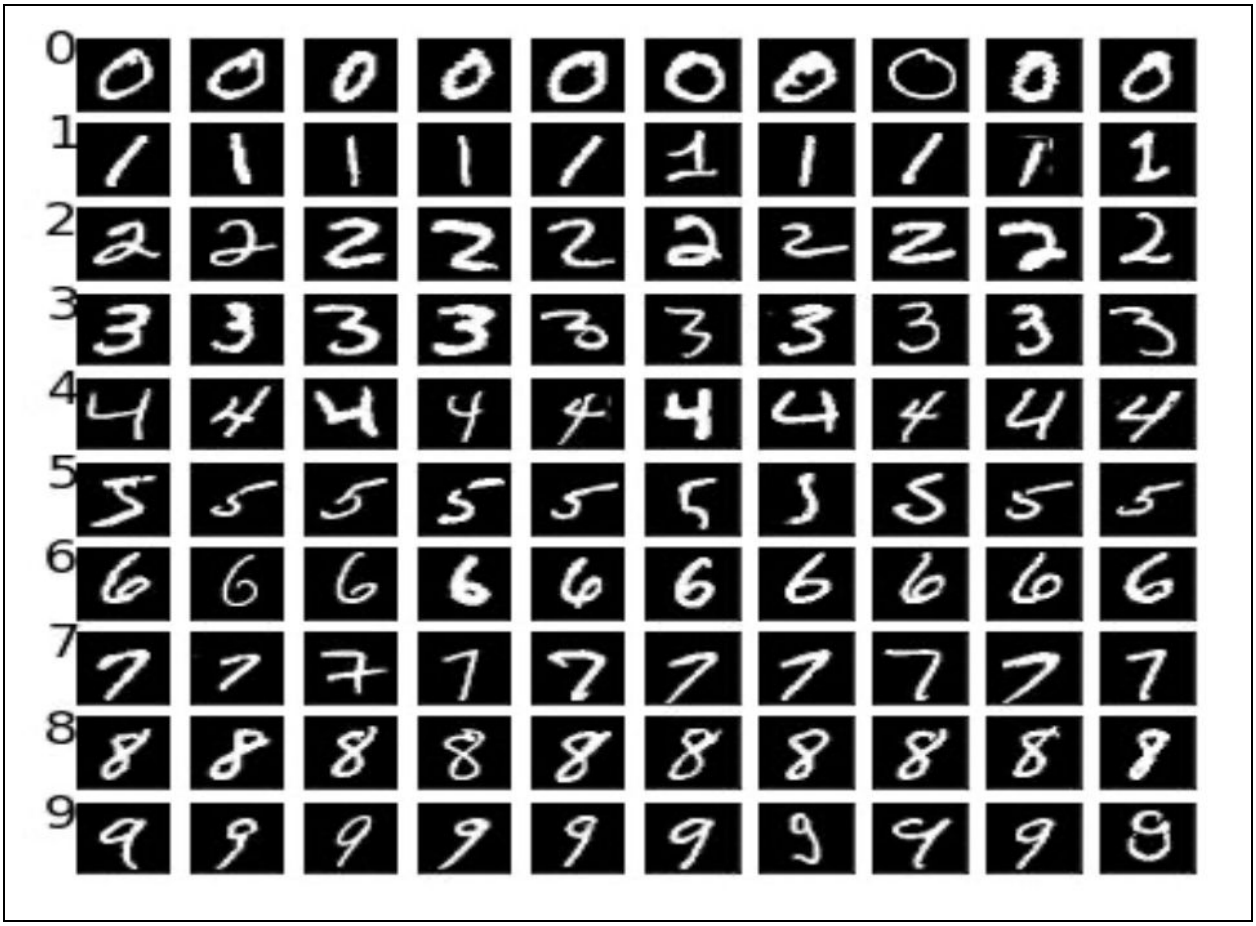
1. Neural Network

DL Toolkit - Part-1

1.1 About dataset

A brief description of the dataset : MNIST

No of samples	10,000 (Training) , 2000 (Testing)
dimensions of each sample	(28,28)
Unique Labels	0,1,2,3,4,5,6,7,8,9



The following observations are made for the data :

- The data is categorical
- Each sample is a handwritten digit which are gray scale images with 28*28 pixels
- Every MNIST data point, every image can be thought as an array of numbers describing
- how dark these pixels is so each value in 28*28 represent intensity value between 0 and 1
- All the images are given labels among [0,1,2,3,4,5,6,7,8,9]

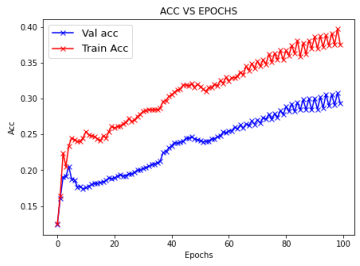
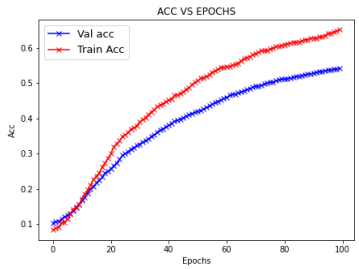
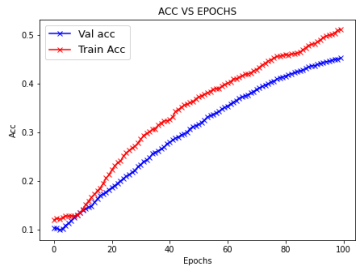
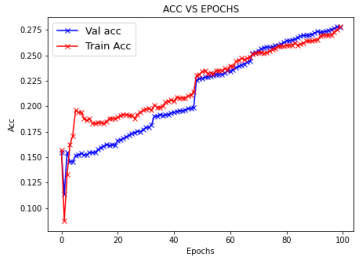
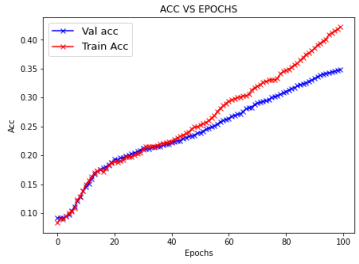
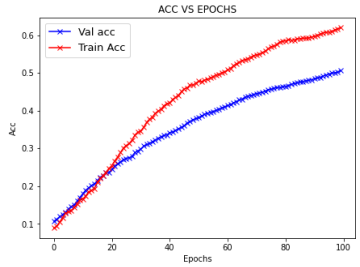
Training data is normalised using standard scaler from sklearn. And the same scaler is used to transform the Testing data.
One hot encoding is used for y-labels to do multi classification in MNIST.

1.3 Selection of Hyper Parameters

- ***Selection of Network***

  Two networks chosen :- **`[784,256,64,10] and [784,128,24,10]`**

  Keeping all other hyper parameters same the model is analysed for both these networks over sigmoid, tanh and relu.

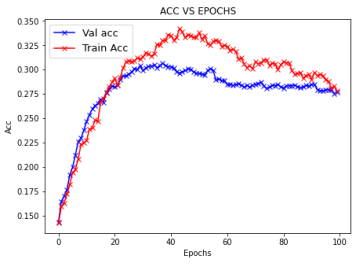
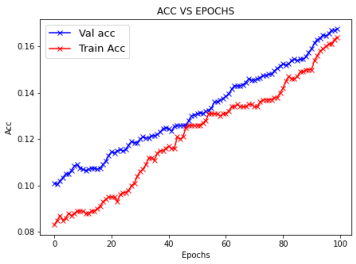
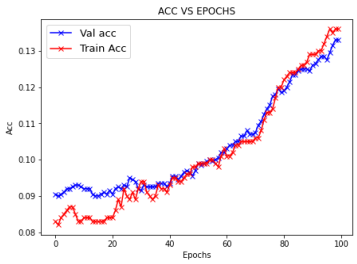
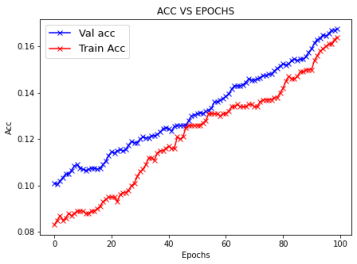
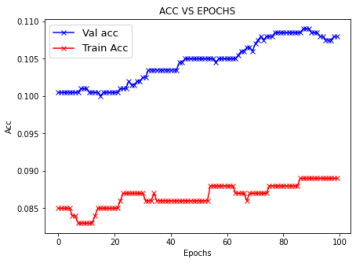
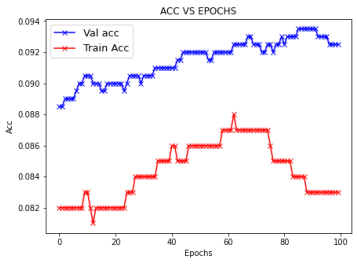
Relu	Tanh	Sigmoid
MODEL [784,256,64,10] , LR = 0.1 , EPOCH = 100		
		
MODEL [784,128,24,10] , LR = 0.1 , EPOCH = 100		
		

It can be observed that one is a complex model with more number of hidden nodes and one is a simpler model. Looking at the performance of both **(both perform equally similar across each activation)** and the resources available at hand we chose to go with a simpler model.

`[784,128,24,10] chosen`

- ***Selection of Learning Rate***
  Two Learning Rate chosen :- **`[0.01] and [0.001]`**

Keeping all other hyper parameters the same , the model is analysed for both these networks over sigmoid, tanh and relu.

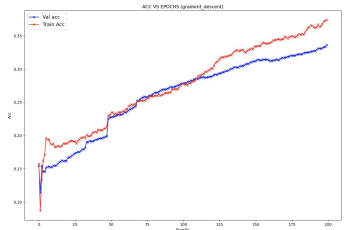
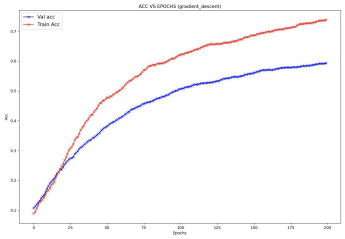
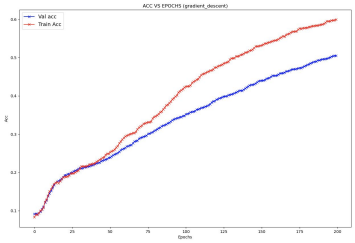
Relu	Tanh	Sigmoid
MODEL [784,128,24,10] , LR = 0.01 , EPOCH = 100		
MODEL [784,128,24,10] , LR = 0.001 , EPOCH = 100		
		
		

It is observed that Learning rate and number of epochs go hand in hand. If LR is chosen small then the model will take more number of epochs to converge and if taken small convergence can happen faster. Also if LR is kept too high then there might be too much deviation in losses across epochs, and if LR is kept too low then the model would not be able to converge.

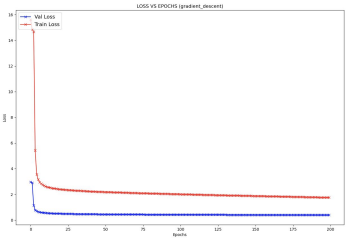
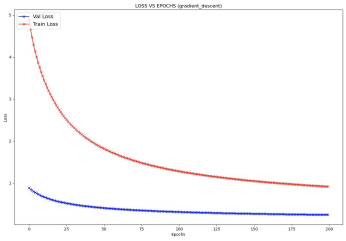
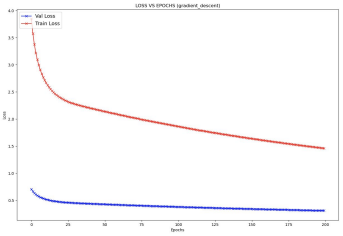
`0.01 chosen as LR`

- ***Comparison of Tanh , Relu , Sigmod***
  All the three activations are run for 100 epochs over the network [784,128,24,10] and LR=0.01. Following observations are made :-

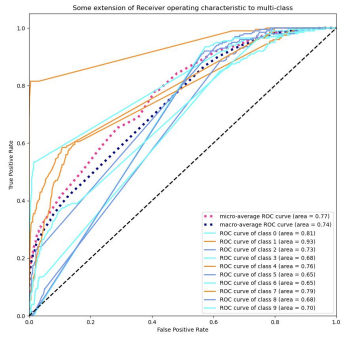
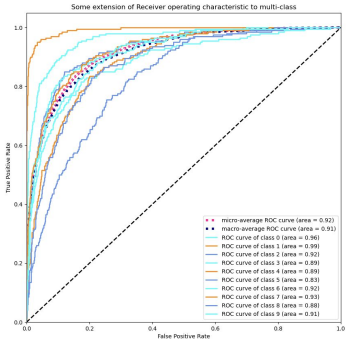
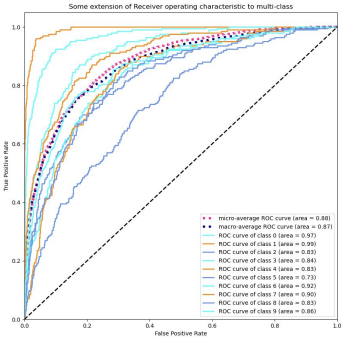
Accuracy vs Epoch

Relu	Tanh	Sigmoid
		

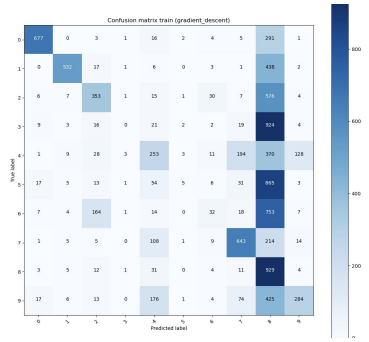
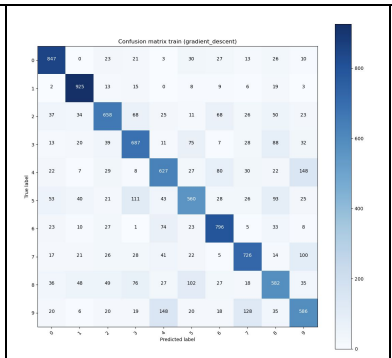
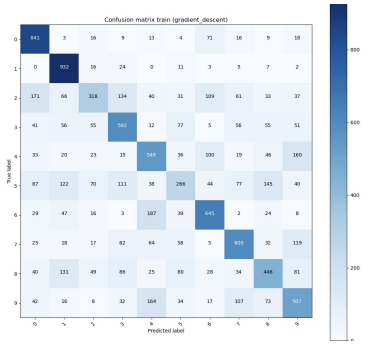
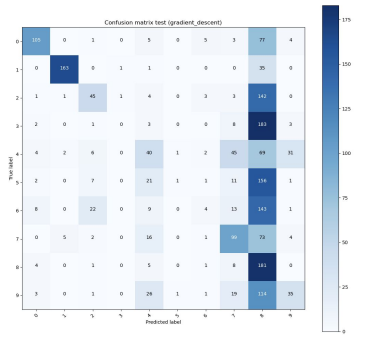
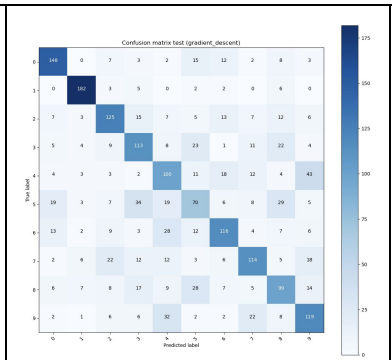
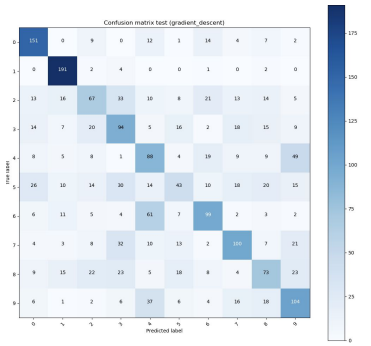
Loss vs Epoch

Relu	Tanh	Sigmoid
		

ROC curves

Relu	Tanh	Sigmoid
Test		
		

Confusion Matrix of Validation set

Relu	Tanh	Sigmoid
Train		
		
Test		
		

Observations :-

1. It can be observed that Relu is oscillating a lot at this learning rate and gets stuck on almost 35 percent accuracy only.
2. Sigmoid and Tanh comparatively improve consistently with each and every epoch , but in this case tanh performs better than sigmoid clearly. More over tanh is always considered as a better activation function compared to sigmoid.
3. So finally tanh is chosen as activation function.

```
Tanh chosen as the activation so final configuration becomes
Network = [784,128,24,10]
Learning Rate = 0.01
Epochs = Decided based on early stopping
Activation = Tanh
```

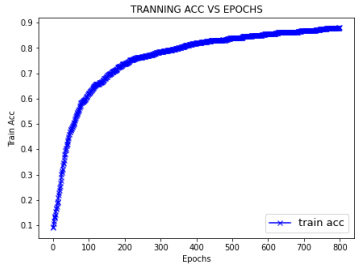
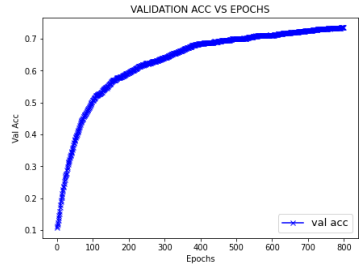
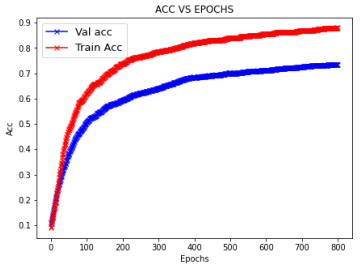
1.4 Finding Right number of epochs for Tanh activation

- ***Tanh without early stopping***

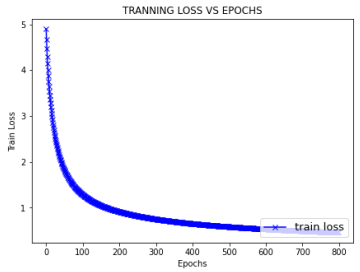
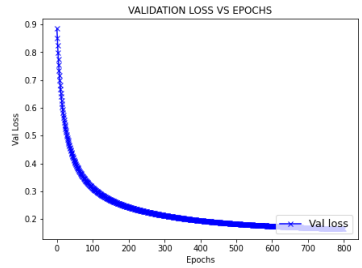
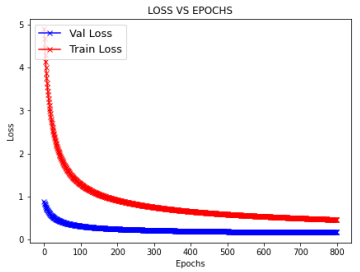
```
Network = [784,128,24,10]
Learning Rate = 0.01
Epochs = 800
Activation = Tanh
```

It can be easily observed from the graphs that the model is overfitting after 450 epochs as loss has become constant and there is no significant change in accuracy.

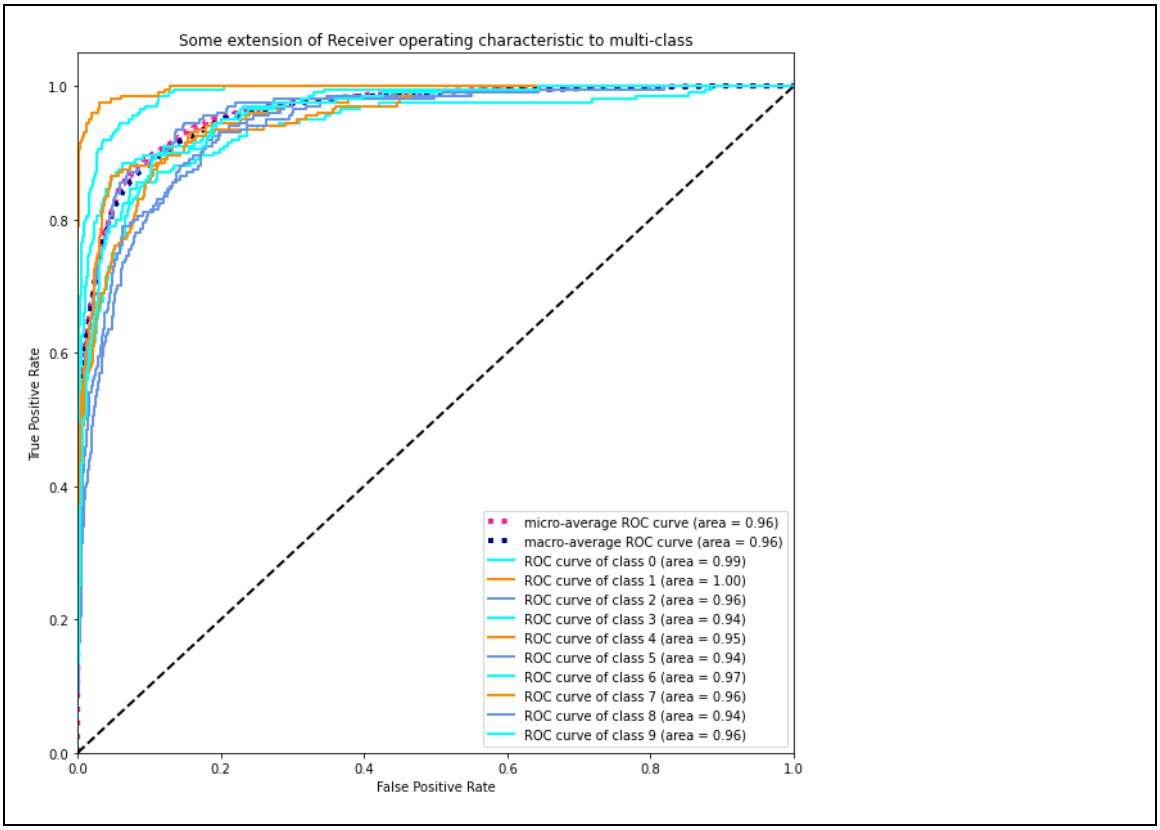
Accuracy vs Epoch

Train	Test	Combined
		

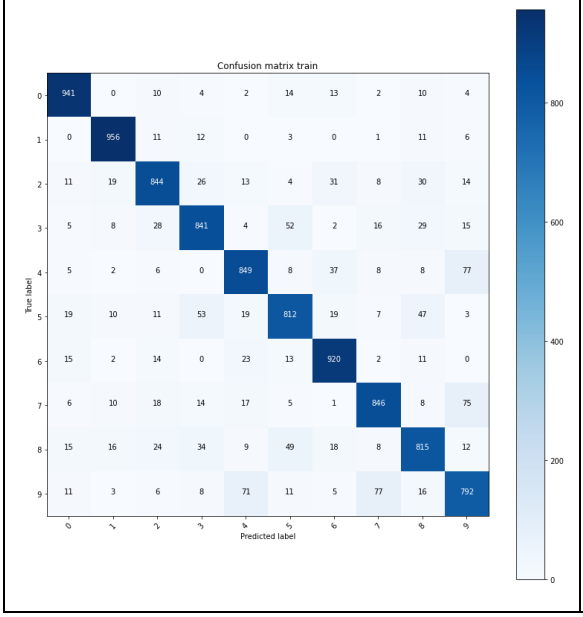
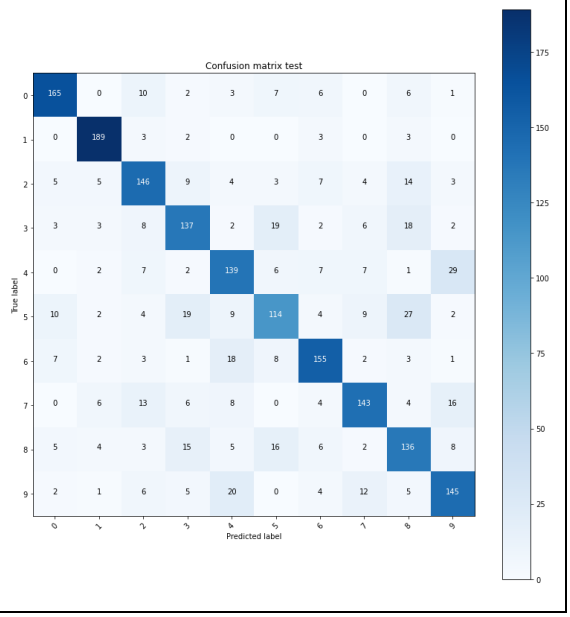
Loss vs Epoch

Train	Test	Combined
		

ROC curves



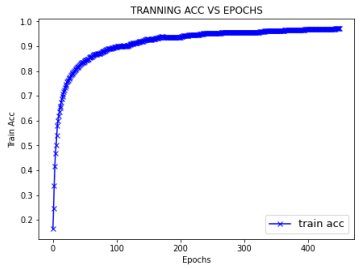
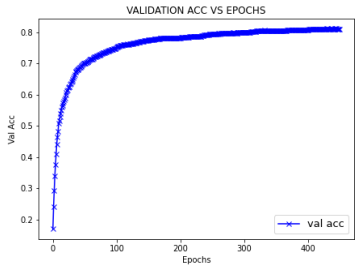
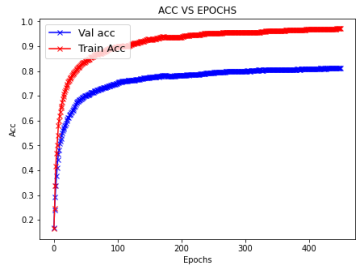
Confusion Matrix of Validation set

Train	Test
	

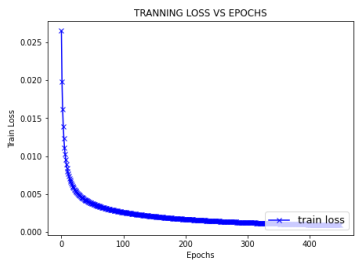
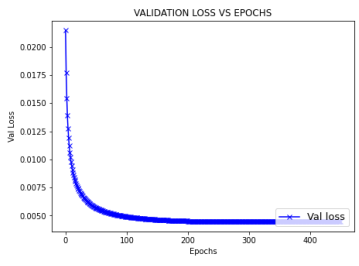
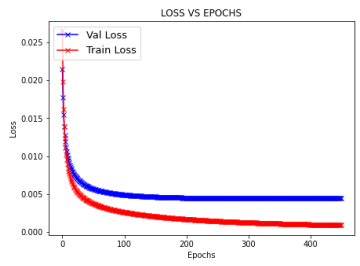
- ***Tanh with early stopping***

```
Network = [784,128,24,10]
Learning Rate = 0.01
Epochs = 450
Activation = Tanh

TESTING ACCURACY
81.10000000000001

TRAINING ACCURACY
96.61999999999999
```

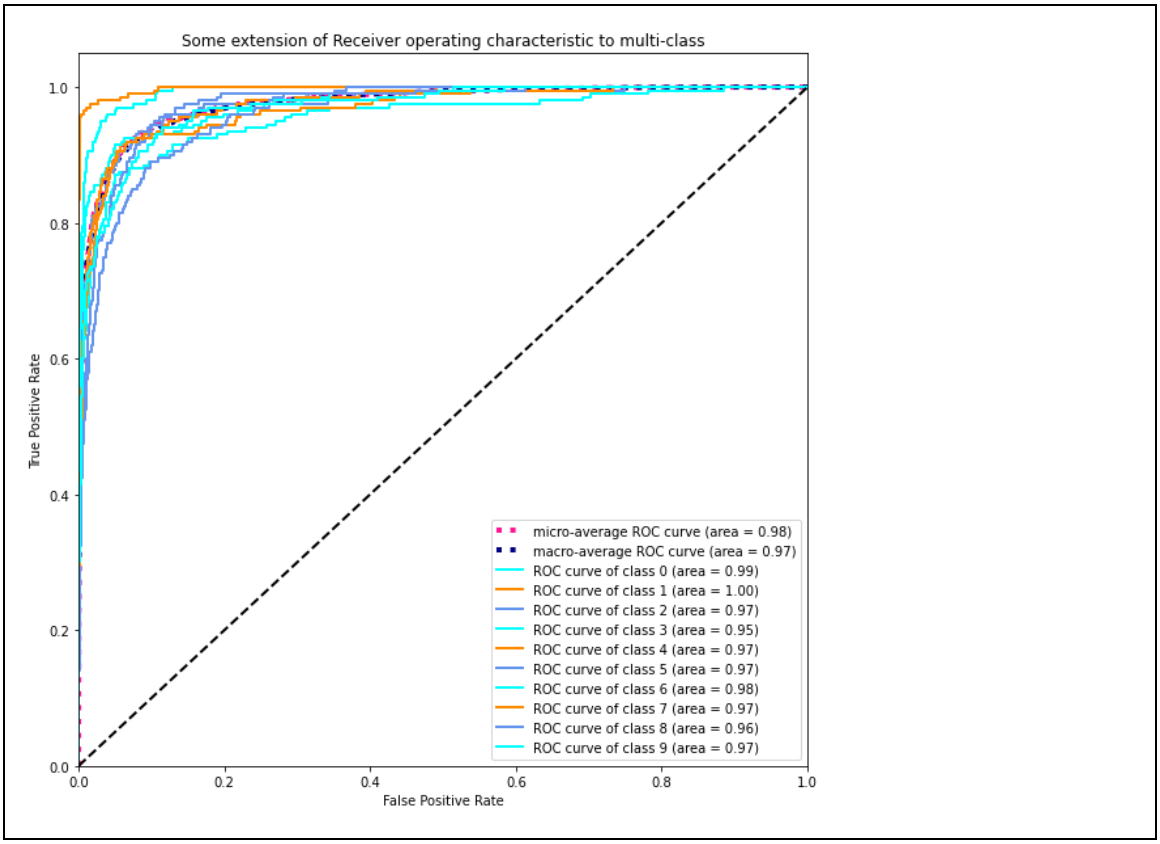
Accuracy vs Epoch

Train	Test	Combined
		

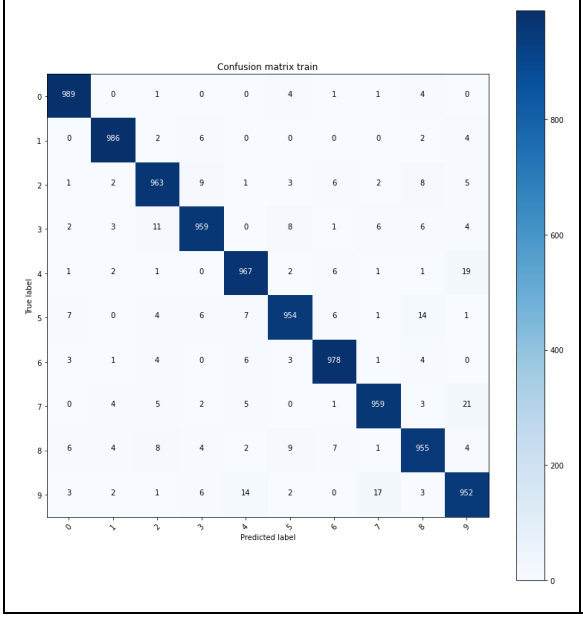
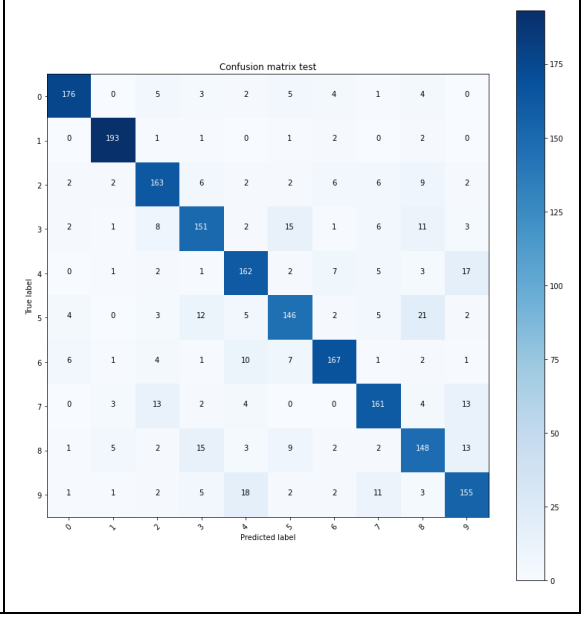
Loss vs Epoch

Train	Test	Combined
		

ROC curves (Val set)



Confusion Matrix of Validation set

Train	Test
	

1.5 Best Configuration form Part-1

To be used in later parts

Network = [784,128,24,10]
Learning Rate = 0.01
Epochs = 450
Activation = Tanh

```
Accuracy Stats for gradient_descent
Train Accuracy: 80.44
Train Accuracy: 69.1

================================================================================================
Optimizer: "gradient_descent"
------------------------------------------------------------------------------------------------
Epochs: 450
------------------------------------------------------------------------------------------------
Activation Fn(Hidden Layers): "tanh"
------------------------------------------------------------------------------------------------
Activation Fn(Output Layer): "softmax"
------------------------------------------------------------------------------------------------
Step size: 0.01
------------------------------------------------------------------------------------------------
Weight initialization strategy: "random"
------------------------------------------------------------------------------------------------
Regularization: "l2"
------------------------------------------------------------------------------------------------
Dropout: 0.2
------------------------------------------------------------------------------------------------
Batch size: 10000

------------------------------------------------------------------------------------------------
Layer 1: (128, 784)

------------------------------------------------------------------------------------------------
Layer 2: (24, 128)

------------------------------------------------------------------------------------------------
Layer 3: (10, 24)
```

3. Optimizers:

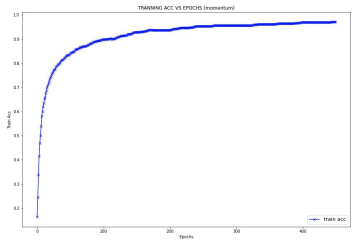
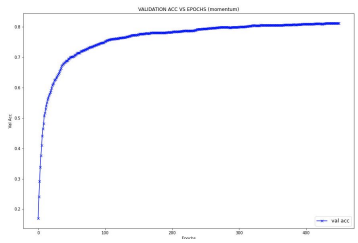
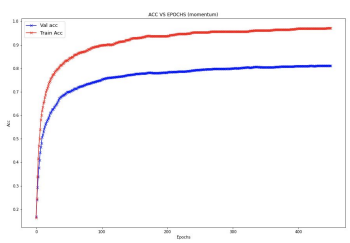
2.1 Plots and Observations

1. Gradient Descent with Momentum Optimizer:-
2. Nesterov's Accelerated Gradient Optimizer:-
3. AdaGrad Optimizer:-
4. RMSprop Optimizer:-
5. Adam Optimizer:-

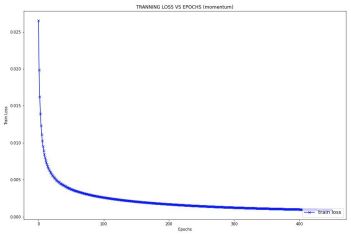
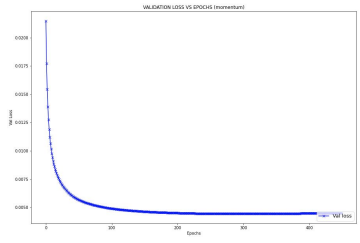
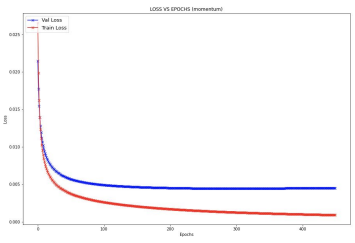
Gradient Descent with Momentum Optimizer:-

Plots

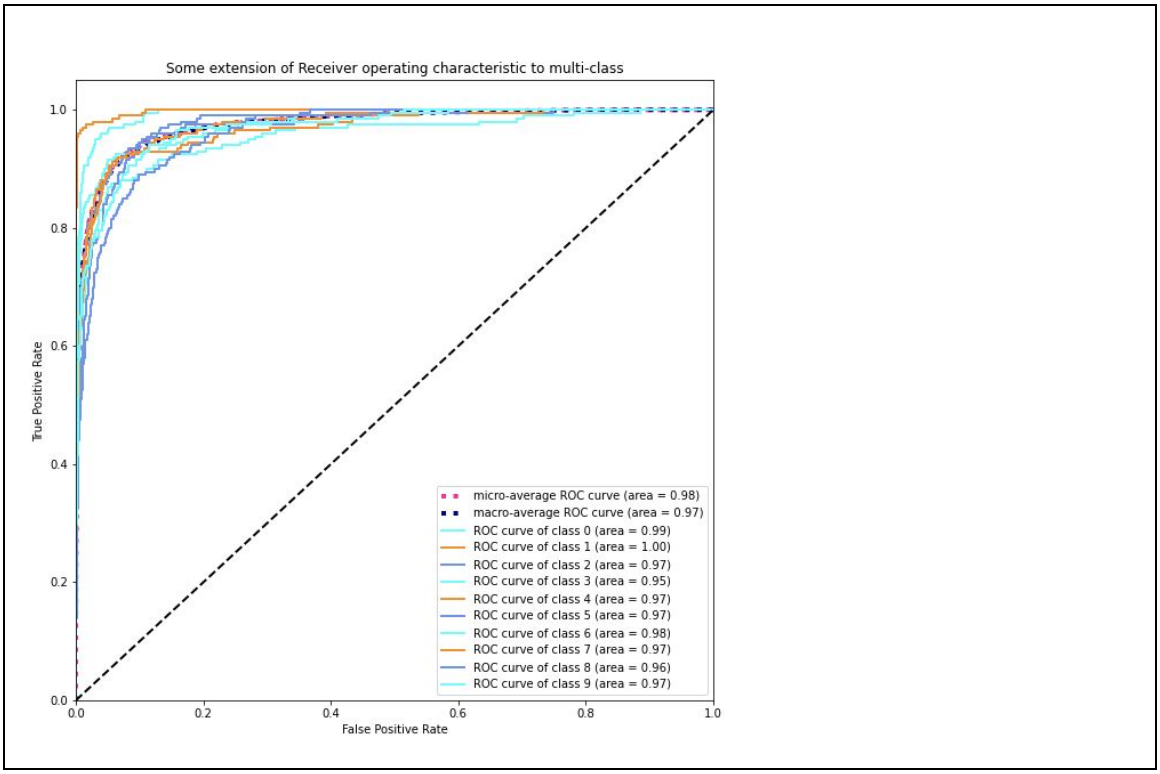
Accuracy vs Epoch

Train	Test	Combined
		

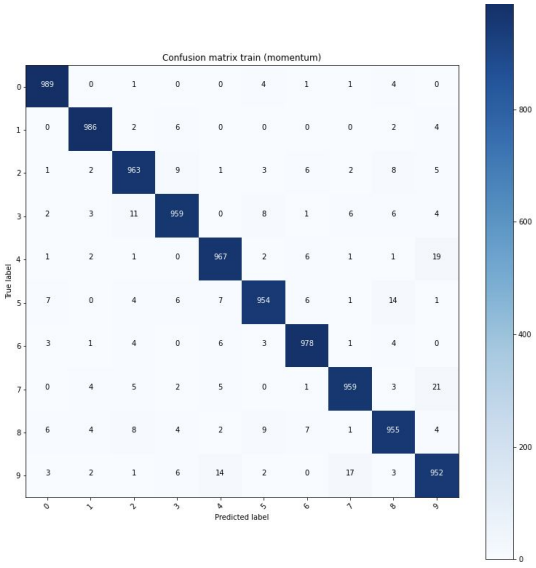
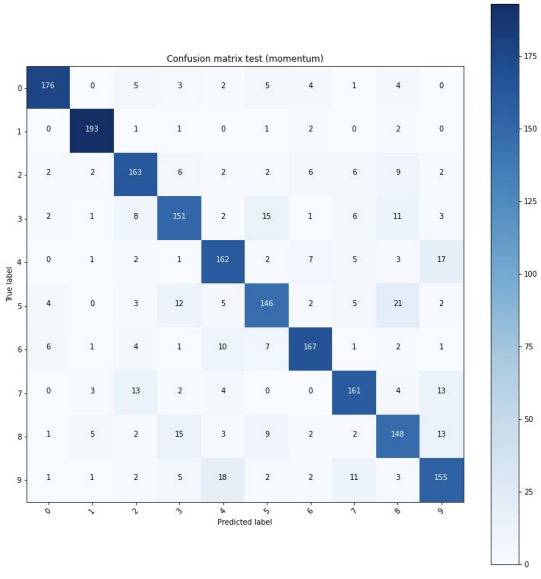
Loss vs Epoch

Train	Test	Combined
		

ROC curves (Val set)



Confusion Matrix of Validation set

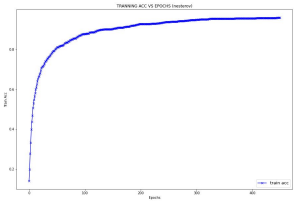
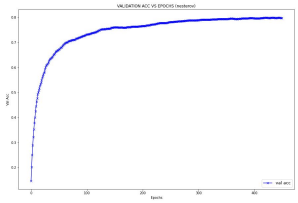
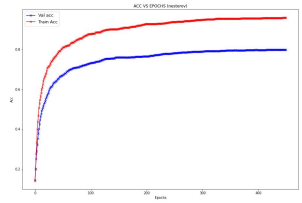
Train	Test
	

```
Accuracy Stats for momentum
Train Accuracy: 96.61999999999999
Train Accuracy: 81.10000000000001

================================================================================================
Optimizer: "momentum"
------------------------------------------------------------------------------------------------
Epochs: 450
------------------------------------------------------------------------------------------------
Activation Fn(Hidden Layers): "tanh"
------------------------------------------------------------------------------------------------
Activation Fn(Output Layer): "softmax"
------------------------------------------------------------------------------------------------
Step size: 0.1
------------------------------------------------------------------------------------------------
Weight initialization strategy: "random"
------------------------------------------------------------------------------------------------
Regularization: "l2"
------------------------------------------------------------------------------------------------
Dropout: 0.2
------------------------------------------------------------------------------------------------
Batch size: 64

------------------------------------------------------------------------------------------------
Layer 1: (128, 784)

------------------------------------------------------------------------------------------------
Layer 2: (24, 128)

------------------------------------------------------------------------------------------------
Layer 3: (10, 24)
```

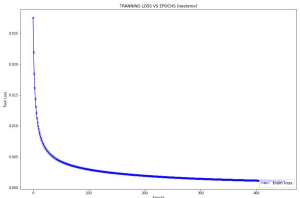
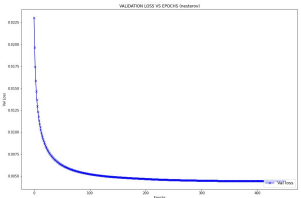
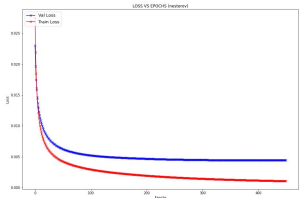
Nesterov's Accelerated Gradient Optimizer:-

Plots

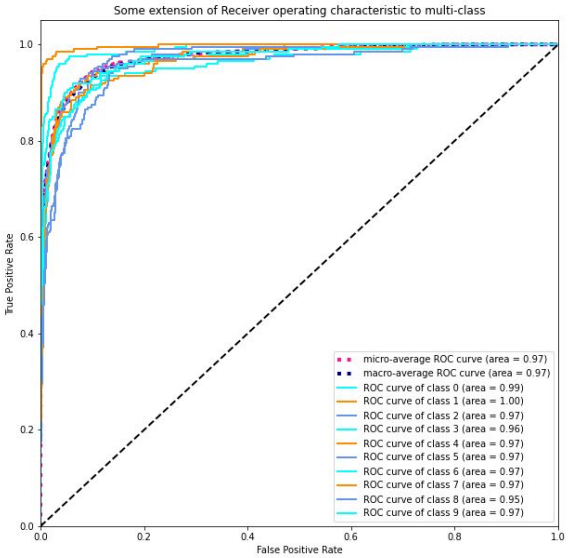
Accuracy vs Epoch

Train	Test	Combined
		

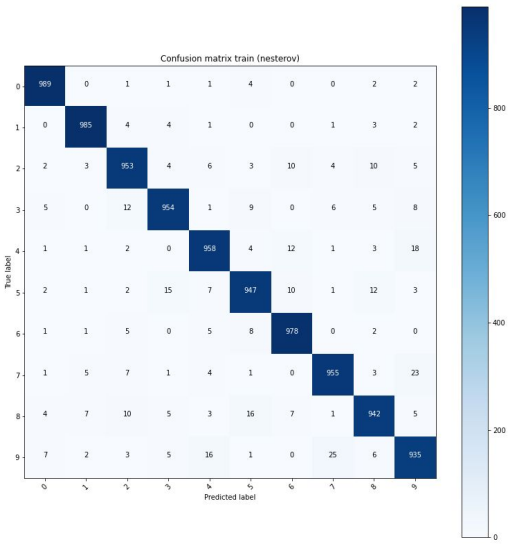
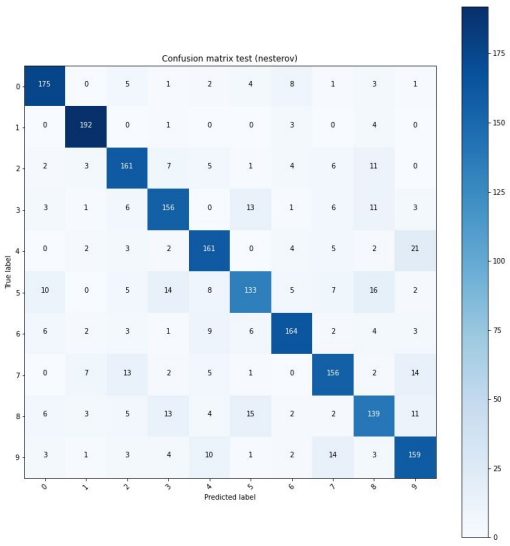
Loss vs Epoch

Train	Test	Combined
		

ROC curves (Val set)



Confusion Matrix of Validation set

Train	Test
	

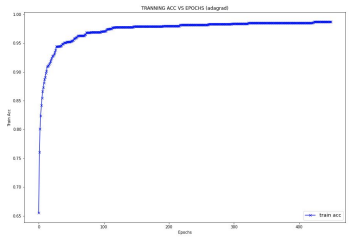
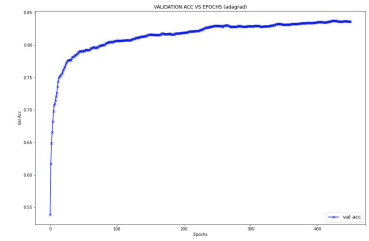
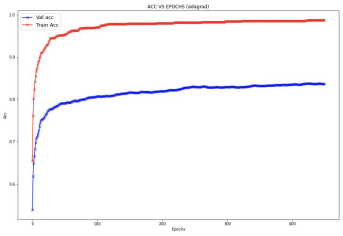
Observations

```
Accuracy Stats for nesterov
Train Accuracy: 95.96000000000001
Train Accuracy: 79.80000000000001

==================================================================================================
Optimizer: "nesterov"
--------------------------------------------------------------------------------------------------
Epochs: 450
--------------------------------------------------------------------------------------------------
Activation Fn(Hidden Layers): "tanh"
--------------------------------------------------------------------------------------------------
Activation Fn(Output Layer): "softmax"
--------------------------------------------------------------------------------------------------
Step size: 0.01
--------------------------------------------------------------------------------------------------
Weight initialization strategy: "random"
--------------------------------------------------------------------------------------------------
Regularization: "l2"
--------------------------------------------------------------------------------------------------
Dropout: 0.2
--------------------------------------------------------------------------------------------------
Batch size: 64

--------------------------------------------------------------------------------------------------
Layer 1: (128, 784)

--------------------------------------------------------------------------------------------------
Layer 2: (24, 128)

--------------------------------------------------------------------------------------------------
Layer 3: (10, 24)
```

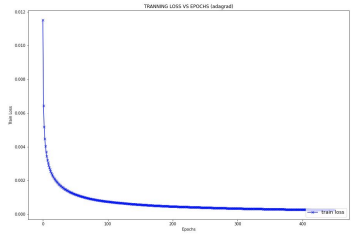
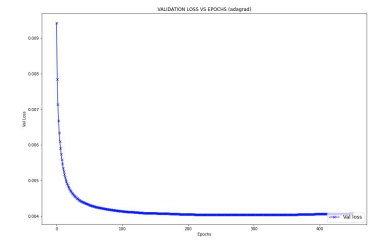
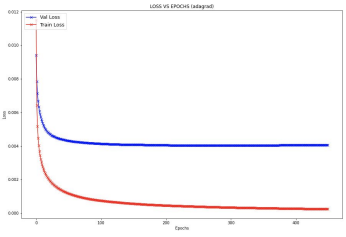
AdaGrad Optimizer:-

Plots

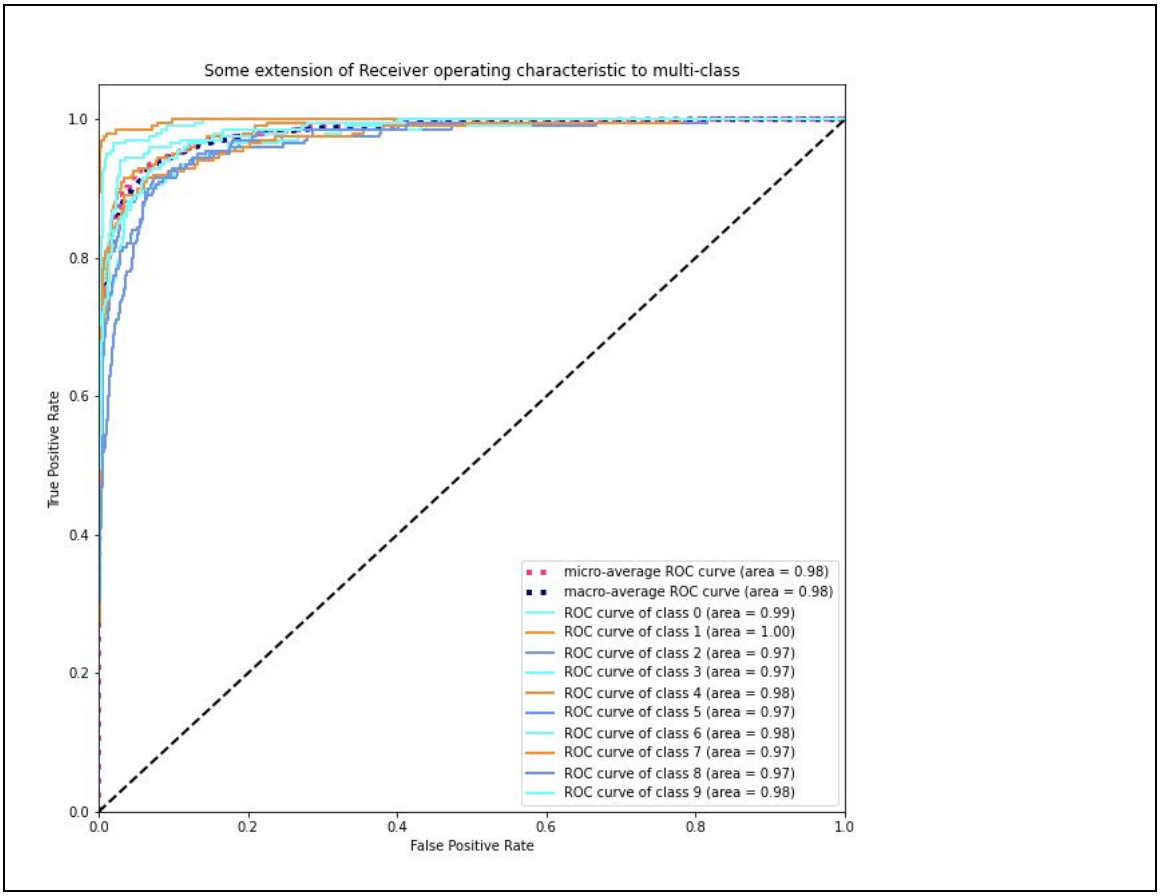
Accuracy vs Epoch

Train	Test	Combined
		

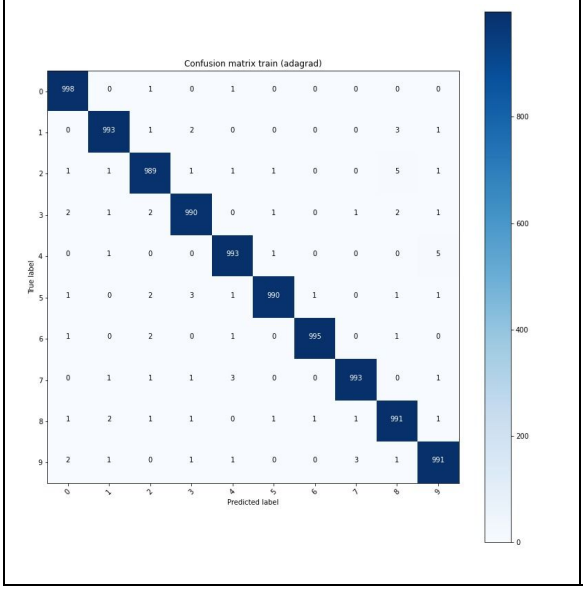
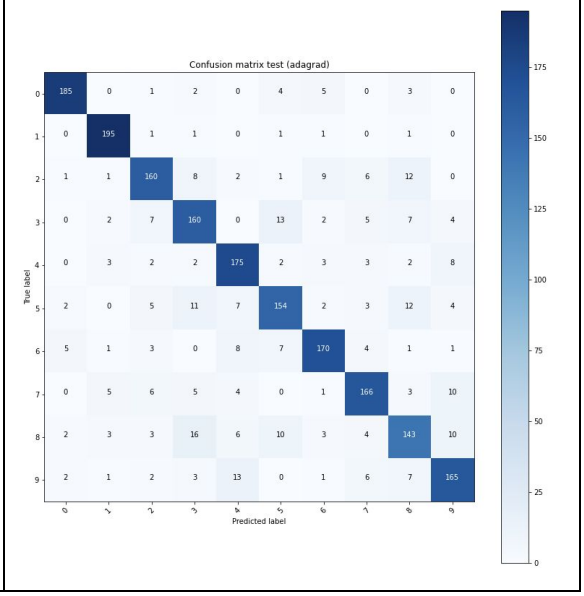
Loss vs Epoch

Train	Test	Combined
		

ROC curves (Val set)



Confusion Matrix of Validation set

Train	Test
	

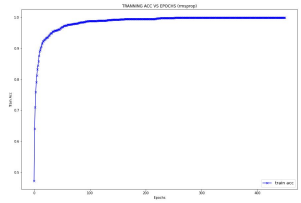
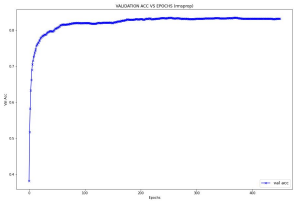
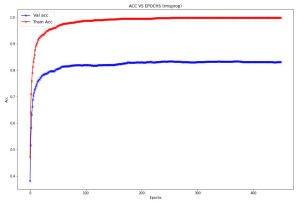
Observations

```
Accuracy Stats for adagrad
Train Accuracy: 99.22999999999999
Train Accuracy: 83.65

=================================================================================================
Optimizer: "adagrad"
-------------------------------------------------------------------------------------------------
Epochs: 450
-------------------------------------------------------------------------------------------------
Activation Fn(Hidden Layers): "tanh"
-------------------------------------------------------------------------------------------------
Activation Fn(Output Layer): "softmax"
-------------------------------------------------------------------------------------------------
Step size: 0.01
-------------------------------------------------------------------------------------------------
Weight initialization strategy: "random"
-------------------------------------------------------------------------------------------------
Regularization: "l2"
-------------------------------------------------------------------------------------------------
Dropout: 0.2
-------------------------------------------------------------------------------------------------
Batch size: 64

-------------------------------------------------------------------------------------------------
Layer 1: (128, 784)

-------------------------------------------------------------------------------------------------
Layer 2: (24, 128)

-------------------------------------------------------------------------------------------------
Layer 3: (10, 24)
```

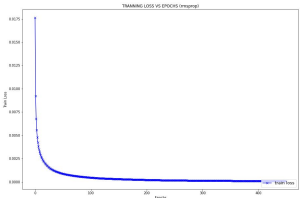
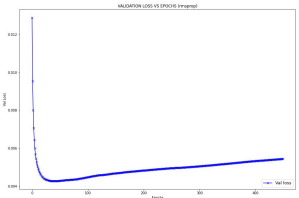
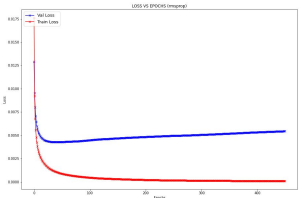
RMSprop Optimizer:-

Plots

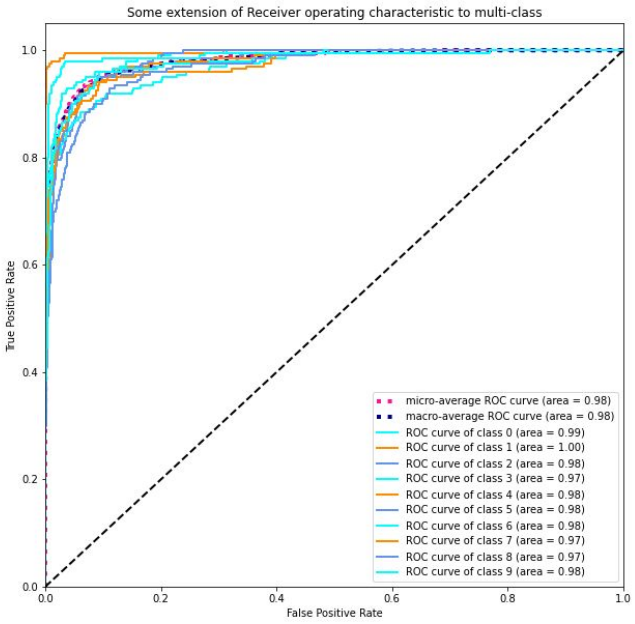
Accuracy vs Epoch

Train	Test	Combined
		

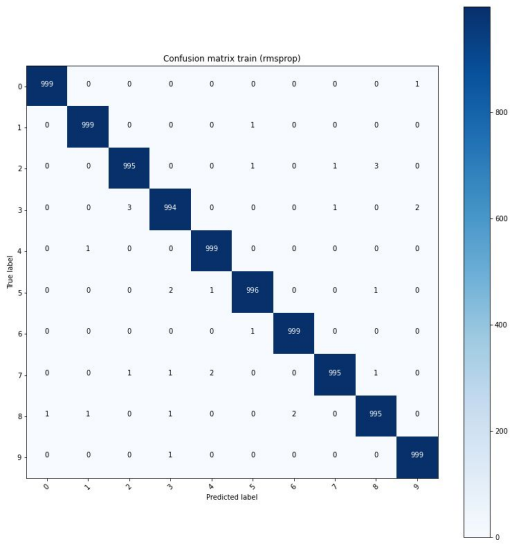
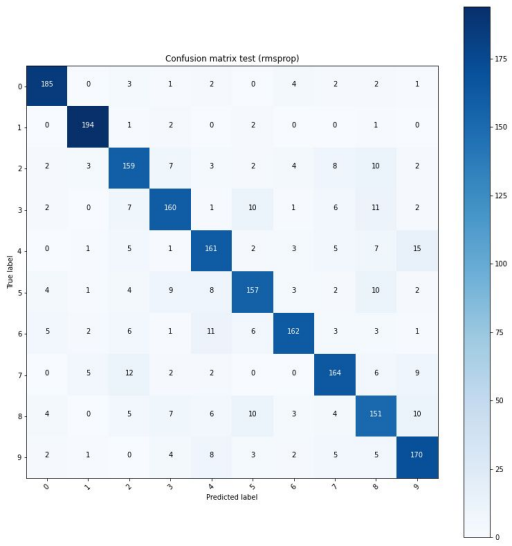
Loss vs Epoch

Train	Test	Combined
		

ROC curves (Val set)



Confusion Matrix of Validation set

Train	Test
	

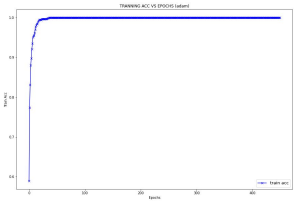
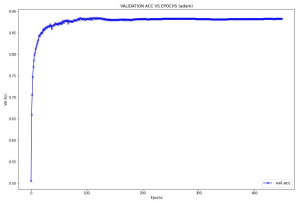
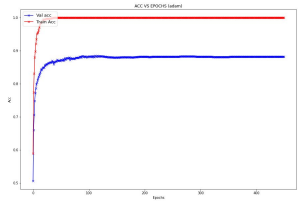
Observations

```
Accuracy Stats for rmsprop
Train Accuracy: 99.7
Train Accuracy: 83.15

=================================================================================================
Optimizer: "rmsprop"
-------------------------------------------------------------------------------------------------
Epochs: 450
-------------------------------------------------------------------------------------------------
Activation Fn(Hidden Layers): "tanh"
-------------------------------------------------------------------------------------------------
Activation Fn(Output Layer): "softmax"
-------------------------------------------------------------------------------------------------
Step size: 0.01
-------------------------------------------------------------------------------------------------
Weight initialization strategy: "random"
-------------------------------------------------------------------------------------------------
Regularization: "l2"
-------------------------------------------------------------------------------------------------
Dropout: 0.2
-------------------------------------------------------------------------------------------------
Batch size: 64

-------------------------------------------------------------------------------------------------
Layer 1: (128, 784)

-------------------------------------------------------------------------------------------------
Layer 2: (24, 128)

-------------------------------------------------------------------------------------------------
Layer 3: (10, 24)
```

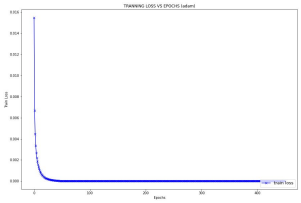
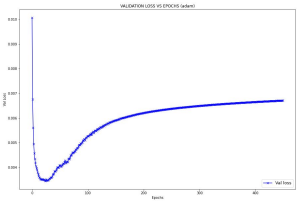
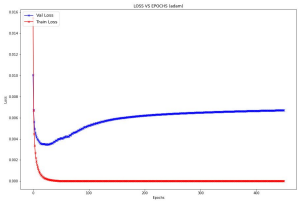
Adam Optimizer:-

Plots

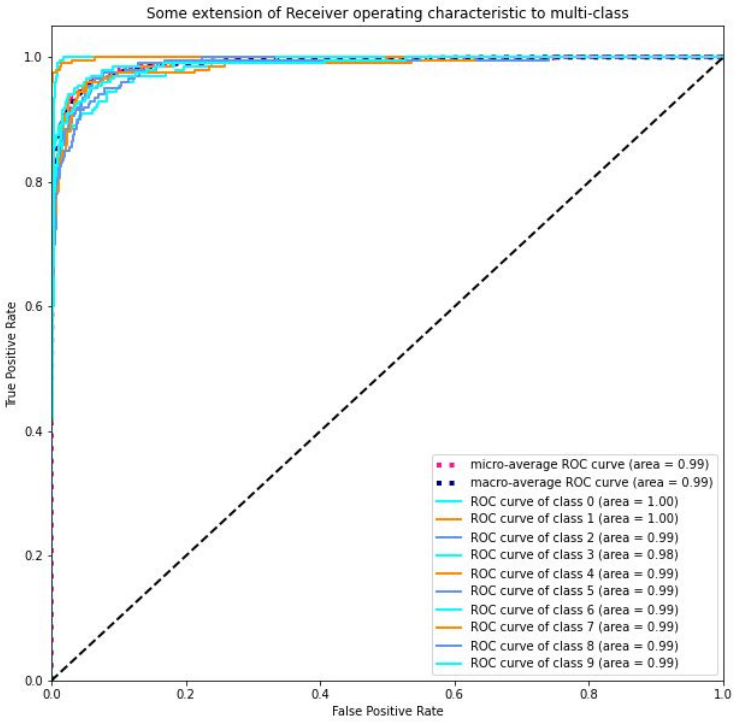
Accuracy vs Epoch

Train	Test	Combined
		

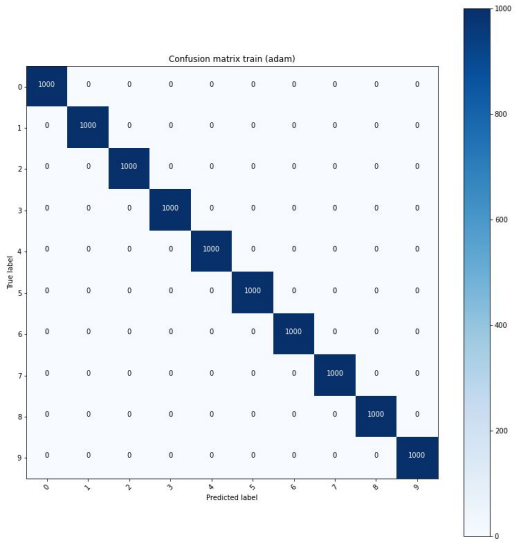
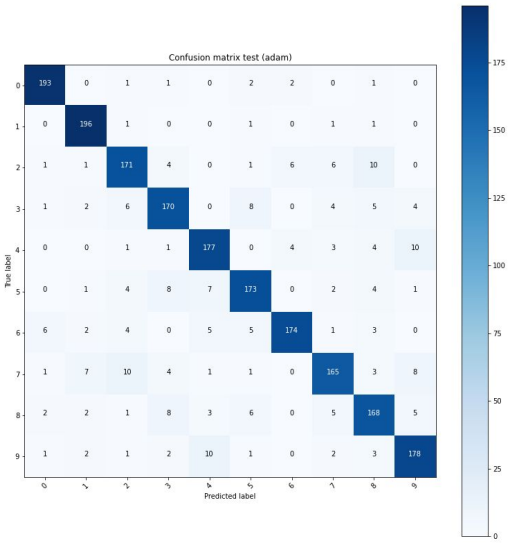
Loss vs Epoch

Train	Test	Combined
		

ROC curves (Val set)



Confusion Matrix of Validation set

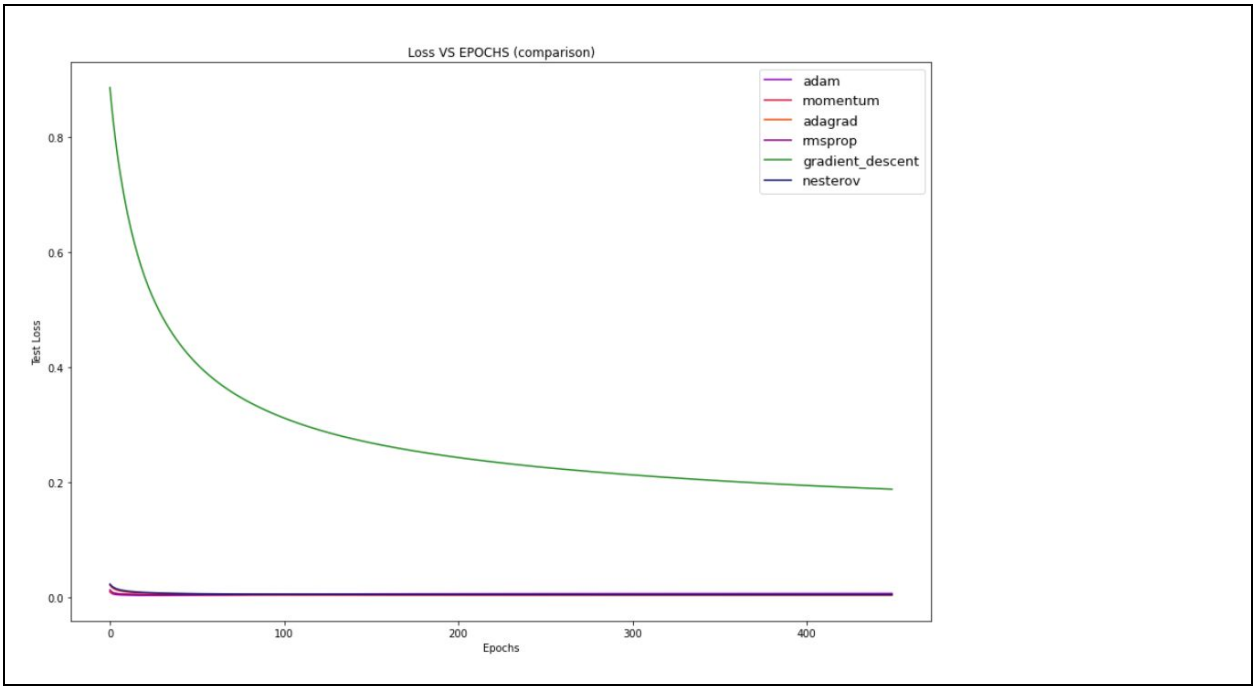
Train	Test
	

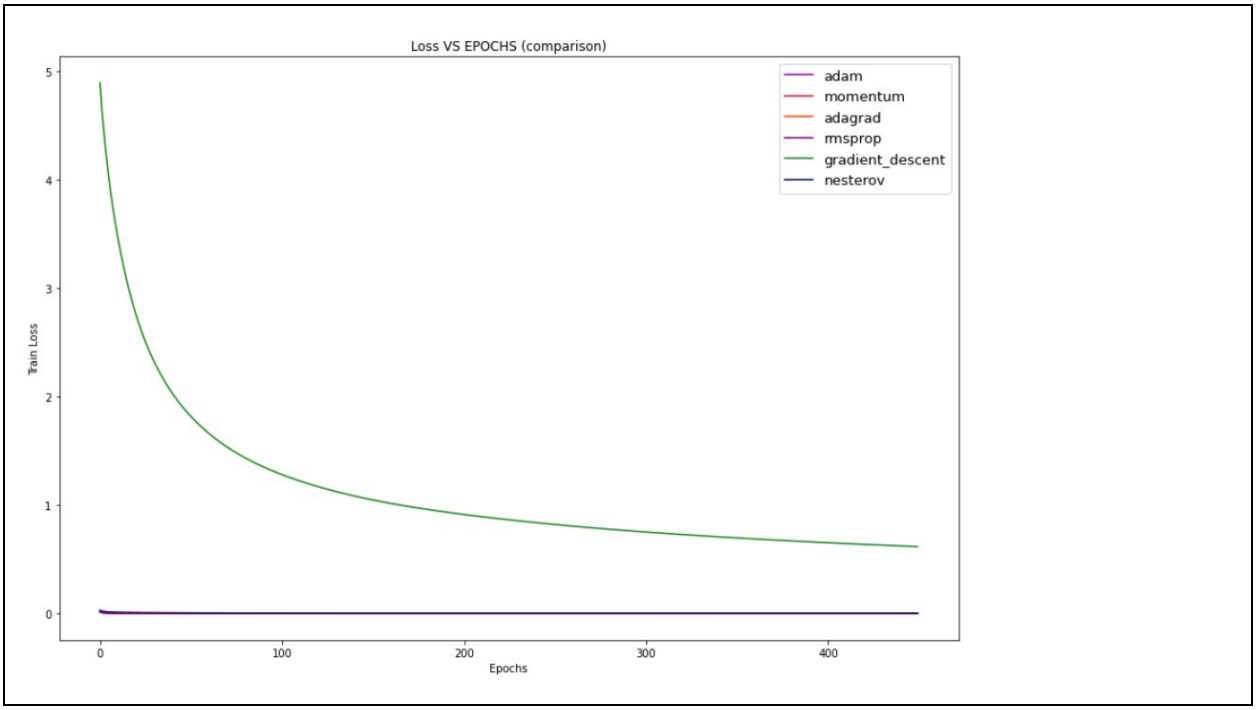
Observations

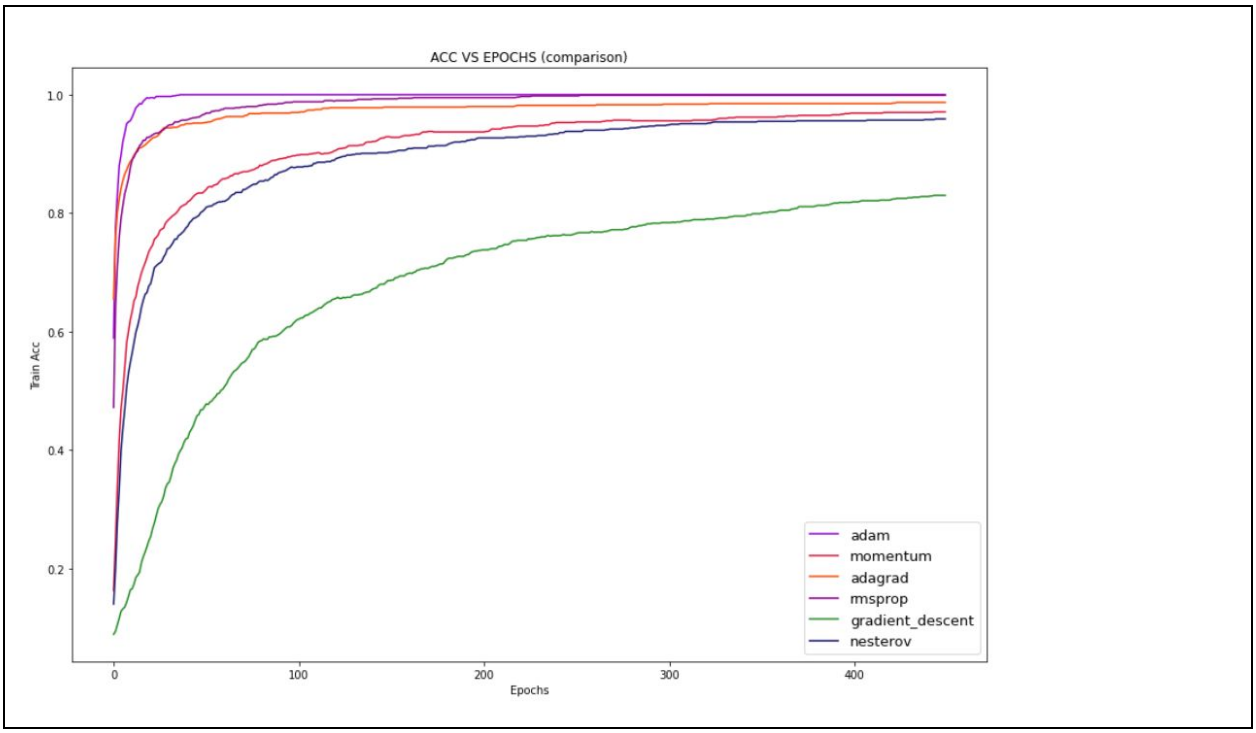
```
Accuracy Stats for adam
Train Accuracy: 100.0
Train Accuracy: 88.25

=============================================================================================
Optimizer: "adam"
---------------------------------------------------------------------------------------------
Epochs: 450
---------------------------------------------------------------------------------------------
Activation Fn(Hidden Layers): "tanh"
---------------------------------------------------------------------------------------------
Activation Fn(Output Layer): "softmax"
---------------------------------------------------------------------------------------------
Step size: 0.01
---------------------------------------------------------------------------------------------
Weight initialization strategy: "random"
---------------------------------------------------------------------------------------------
Regularization: "l2"
---------------------------------------------------------------------------------------------
Dropout: 0.2
---------------------------------------------------------------------------------------------
Batch size: 64

---------------------------------------------------------------------------------------------
Layer 1: (128, 784)

---------------------------------------------------------------------------------------------
Layer 2: (24, 128)

---------------------------------------------------------------------------------------------
Layer 3: (10, 24)
```

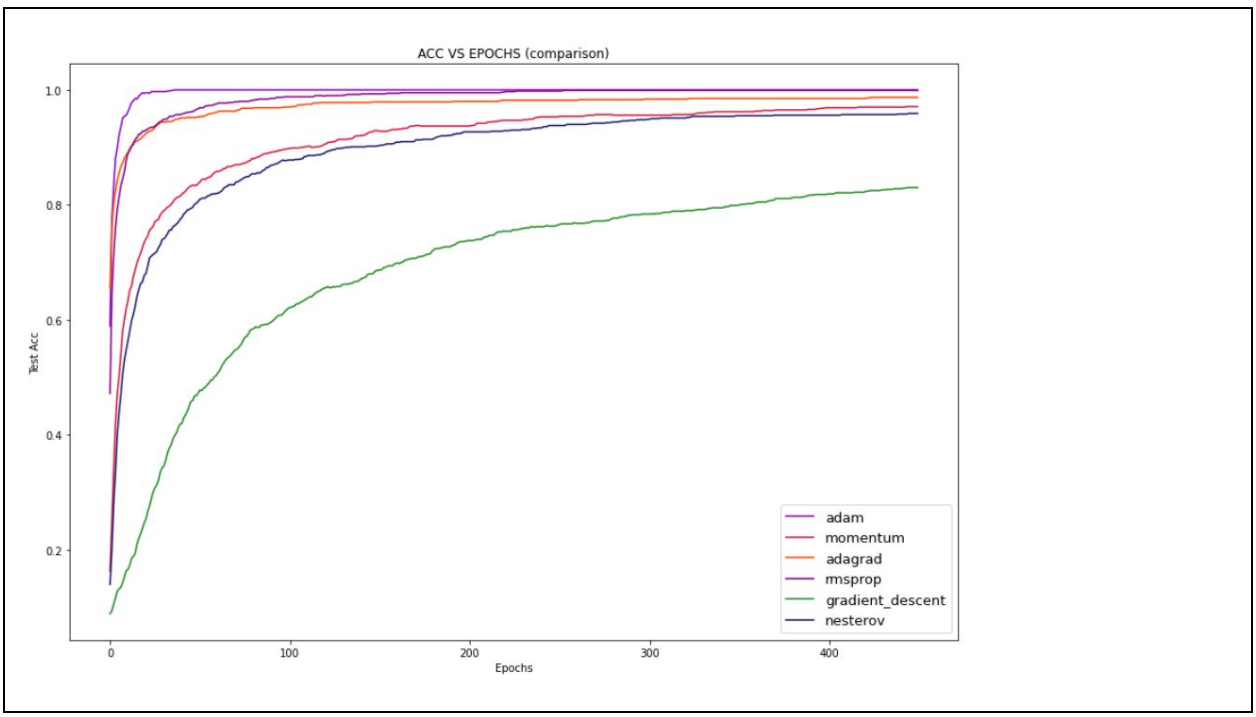
2.2 Best Optimizer and Model Comparisions

BEST OPTIMIZER AND MODEL :-






ACC VS EPOCHS (comparison)
Train Acc
Epochs
adam
momentum
adagrad
rmsprop
gradient_descent
nesterov


ACC VS EPOCHS (comparison)
Test Acc
Epochs
adam
momentum
adagrad
rmsprop
gradient_descent
nesterov

Our Best mode is Adam with the following architecture and accuracies :-

```
Accuracy Stats for adam
Train Accuracy: 100.0
Train Accuracy: 88.25

================================================================================================
Optimizer: "adam"
------------------------------------------------------------------------------------------------
Epochs: 450
------------------------------------------------------------------------------------------------
Activation Fn(Hidden Layers): "tanh"
------------------------------------------------------------------------------------------------
Activation Fn(Output Layer): "softmax"
------------------------------------------------------------------------------------------------
Step size: 0.01
------------------------------------------------------------------------------------------------
Weight initialization strategy: "random"
------------------------------------------------------------------------------------------------
Regularization: "l2"
------------------------------------------------------------------------------------------------
Dropout: 0.2
------------------------------------------------------------------------------------------------
Batch size: 64

------------------------------------------------------------------------------------------------
Layer 1: (128, 784)

------------------------------------------------------------------------------------------------
Layer 2: (24, 128)

------------------------------------------------------------------------------------------------
Layer 3: (10, 24)
```

Adam is basically a combination of RMS Prop and Momentum. Both Momentum and RMS Prop focus on different aspects to improve the convergence of models. Momentum accelerates the convergence towards the relevant direction and reduces the fluctuation to the irrelevant direction. And Rms prop helps in manually tuning the learning rate with the idea of slowing down as we reach close to the global minima.
Thus the combination of both Adam out performs by tuning learning rate and at the same time moving in the right direction.

OBSERVATIONS :-

1. Adam performs the best because it is a combination of Rms Prop and Momentum
2. Gradient Descent performs the worst among all the models because the learning rate is constant which is opposite to the fact that it is always good to keep a higher learning rate initially and then reduce it further as one reaches close to minimum. Also it may be the case that gradient descent gets stuck in local minima and is not able to get out of it and thus not able to achieve the highest accuracy.
3. In our case Momentum and Nesterov almost perform equally, though theoretically Nesterov is able to converge quickly because of the look ahead concept in it, But here in our cost function both performance is equally good.
   Compared to gradient descent they both are better because they use weighted averaging and so are able to avoid the local minima and move in proper direction with less damping.
4. After momentum and Neterov, AdaGad gives better performance as it eliminates the need to manually tune the learning rate.
5. RmsProp is better compared AdaGrad as it overcomes one flaw in adagrad. The accumulated sum keeps growing during training in adagrad. This in turn causes the learning rate to shrink and eventually become infinitesimally small, at which point the algorithm is no longer able to acquire additional knowledge. Rms Prop solves this problem.
6. In the end Adam performs the best.
   Adam optimizer is highly versatile, and works in a large number of scenarios. It takes the best of RMS-Prop and momentum, and combines them together.
   As it can be anticipated, Adam accelerates and decelerates thanks to the component associated with momentum. At the same time, Adam keeps its learning rate adaptive which can be attributed to the component associated to RMS-Prop.

Contribution of Each Member

1. **Shivam Sharma**
   Implemented Optimizers and Plots code and network class
2. **Akanksha Shrimal**
   Implemented Gradient Descent with back propagation.
3. **Vaibhav Goswami**
   Helped in report and Optimizers code