CS577-Assignment 3

Multi Class Classifier

Name : Sourav Yadav **Spring 2020**

AID: A20450418

1.Problem Statement :

Classification of iris flowers from sepal and petal dimensions. This is three class classification problem which we are going to solve by implementing various neural networks.

2.Proposed Solution :

We will design a neural network in Python with Keras and then train the network on train and validation data. Finally we will evaluate performance of the network on test data.

3.Implementation details :

- We first load the data the Irish Flower data from the below link. Attribute Information is also given.
  https://archive.ics.uci.edu/ml/datasets/Iris

Attribute Information:

1. sepal length in cm
2. sepal width in cm
3. petal length in cm
4. petal width in cm
5. class:
-- Iris Setosa
-- Iris Versicolour
-- Iris Virginica

- Then we are going to One Hot Encode the labels of the dataset.
- We split the Iris data in Train , Test and validation Set using train_test_split() method in 8:2 ratio.
- Then we normalized the features.
- We designed neural network with 4 nodes in the input layers , 16 nodes in two hidden layers followed by an output node with Softmax function which will return predicated probabilities of the input data.
- Activation used in the hidden layers is Relu and Activation used in the output node is Softmax.
- We used various Loss Function and Optimizer to evaluate the network that are discussed in the results section.
- Now we train our network with train data and observed its performance . Results are discussed in below section.

4. Results and discussion:

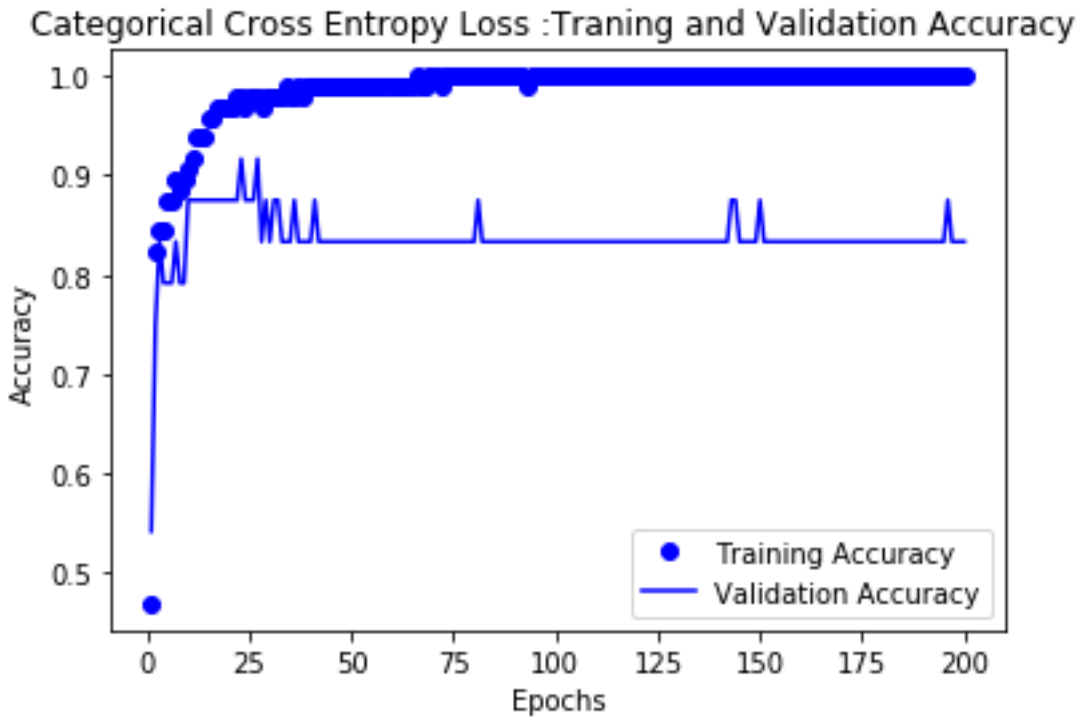
Evaluating Different Loss Functions:

- Initial hyperparameters used during training.
  Learning Rate : 0.001
  Epochs : 200
  Batch Size=1

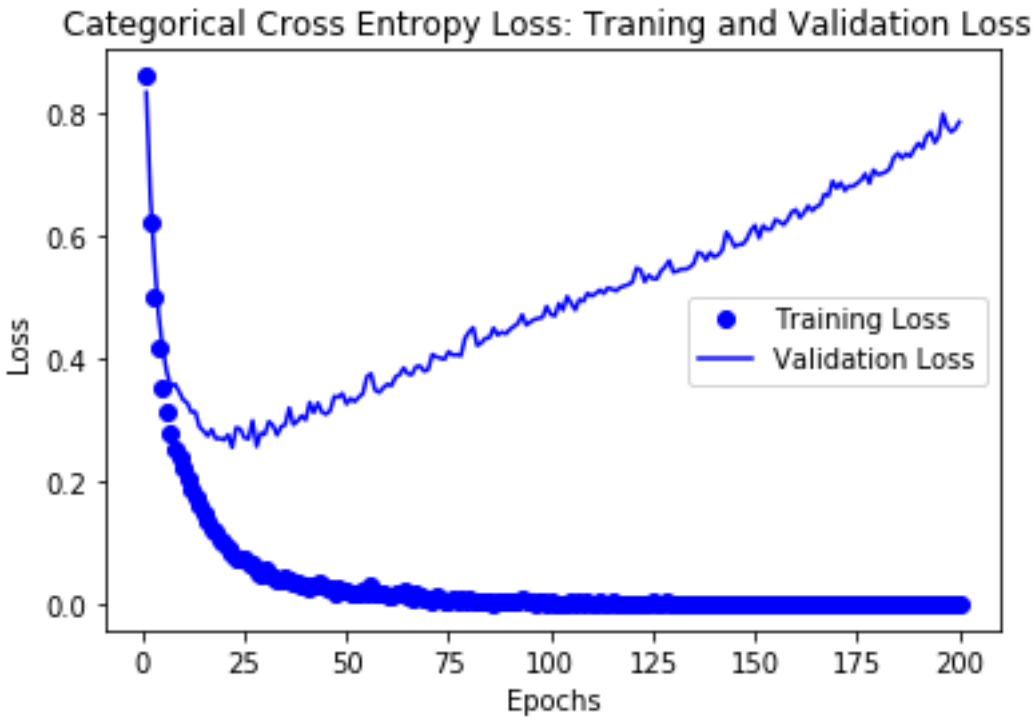
Categorical Cross Entropy:

- After training the network with above hyperparameters and tuning , we got below results on train and validation sets.

  Accuracy Graph : Training Accuracy and Validation Accuracy



  Loss Graph : Training Loss and Validation Loss.

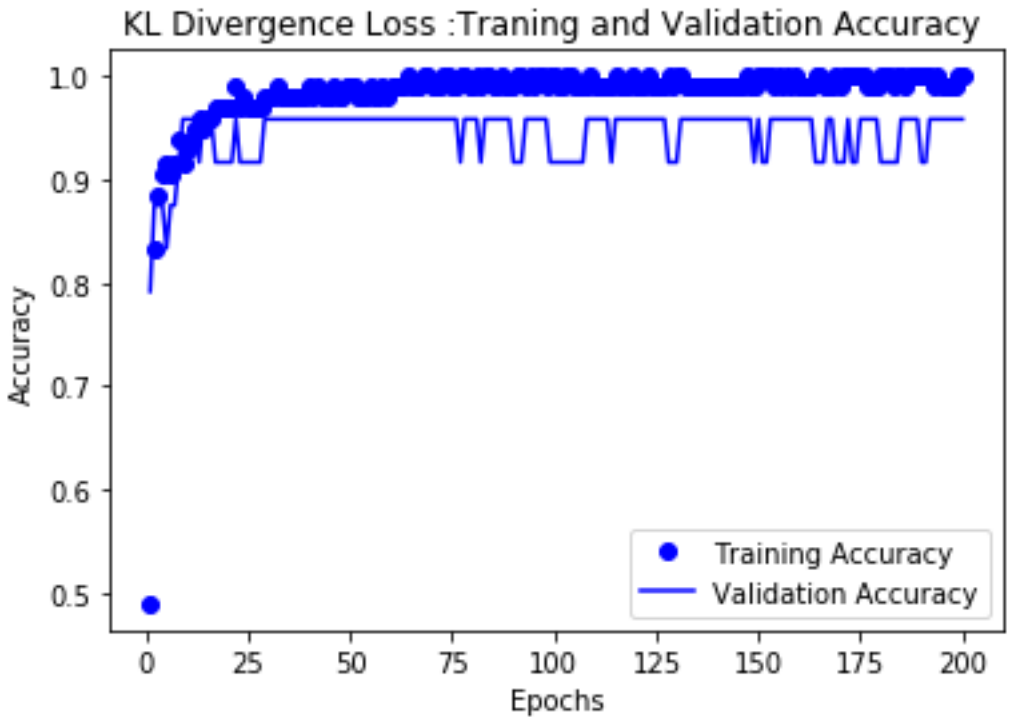


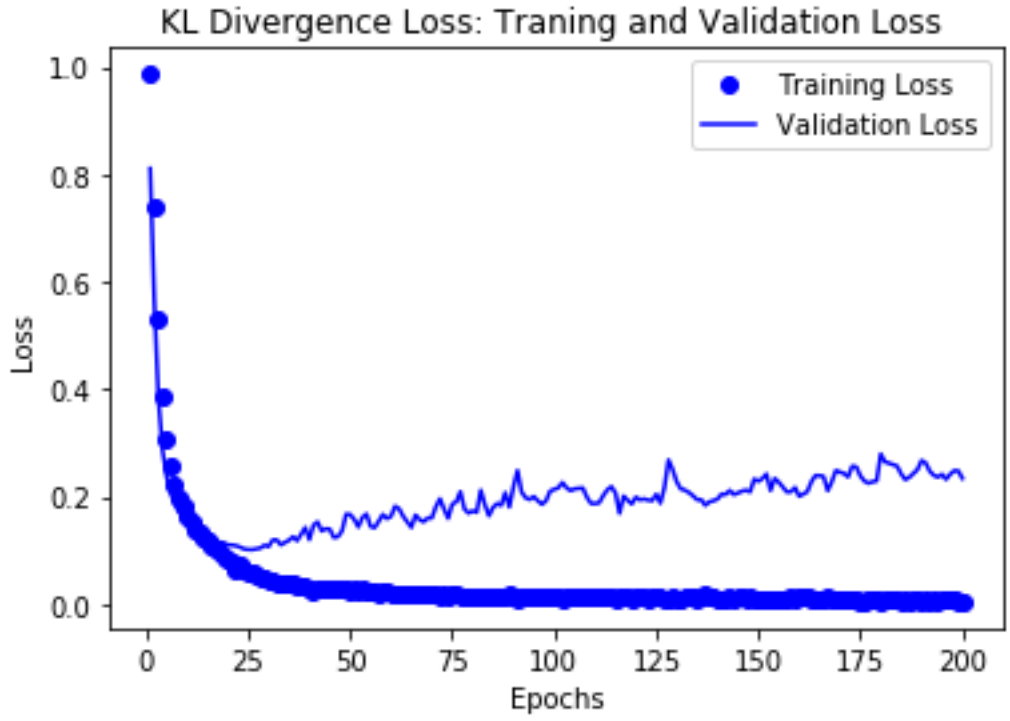
- From the above graph we see that our models validation loss starts to increase after 20 epochs and hence we choose our final number of epochs as 20
- Finally we evaluate our network on test data and got below results.

```
Loss: 0.38328394293785095
Accuracy: 0.8999999761581421
```

Kullback Leibler Divergence Loss:

- After training the network with above hyperparameters and tuning , we got below results on train and validation sets.



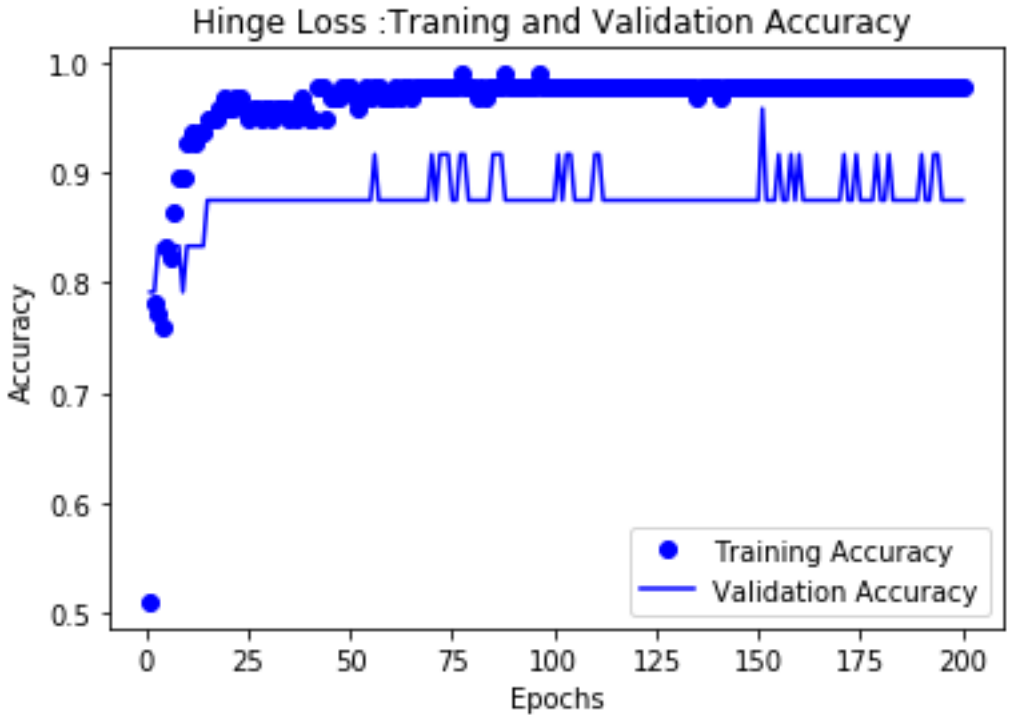


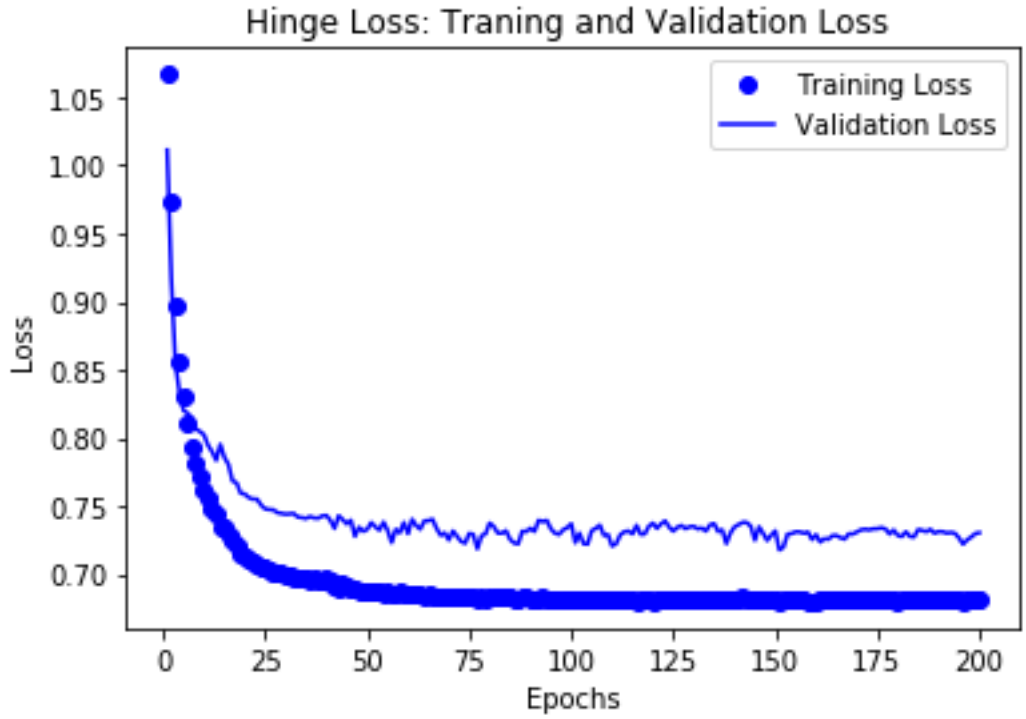
- From the above graph we see that our models validation loss starts to increase after 40 epochs and hence we choose our final number of epochs as 40.
- Finally we evaluate our network on test data and got below results.

```
Loss: 0.07447903603315353
Accuracy: 0.9666666388511658
```

Hinge Loss:

- After training the network with above hyperparameters and tuning , we got below results on train and validation sets.





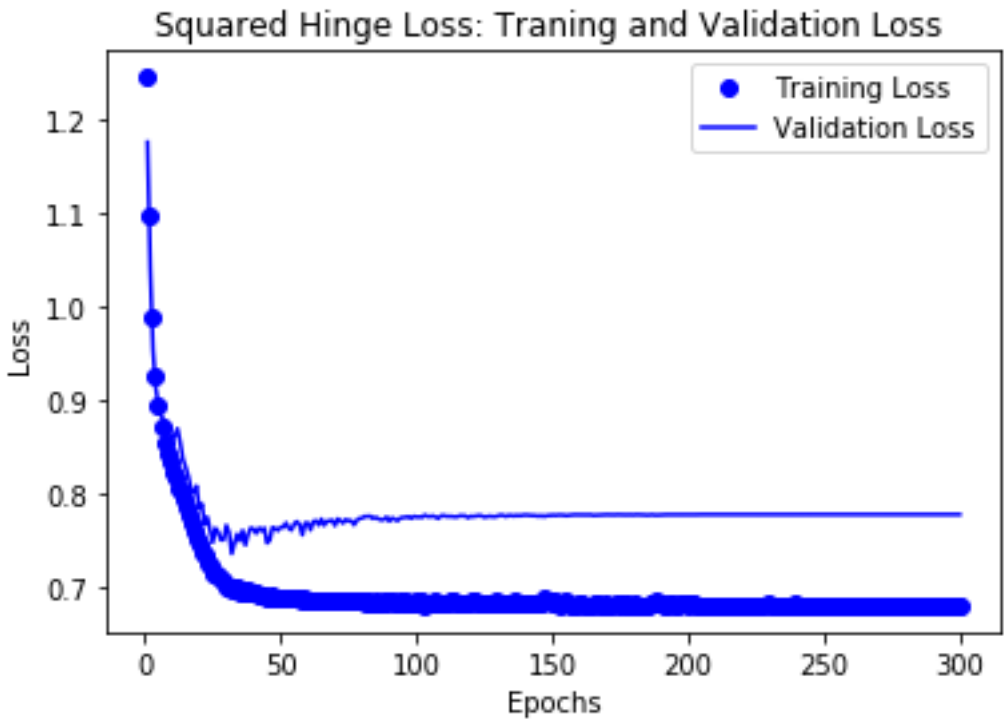
- From the above graph we see that our models validation loss starts to increase after 25 epochs and hence we choose our final number of epochs as 25.

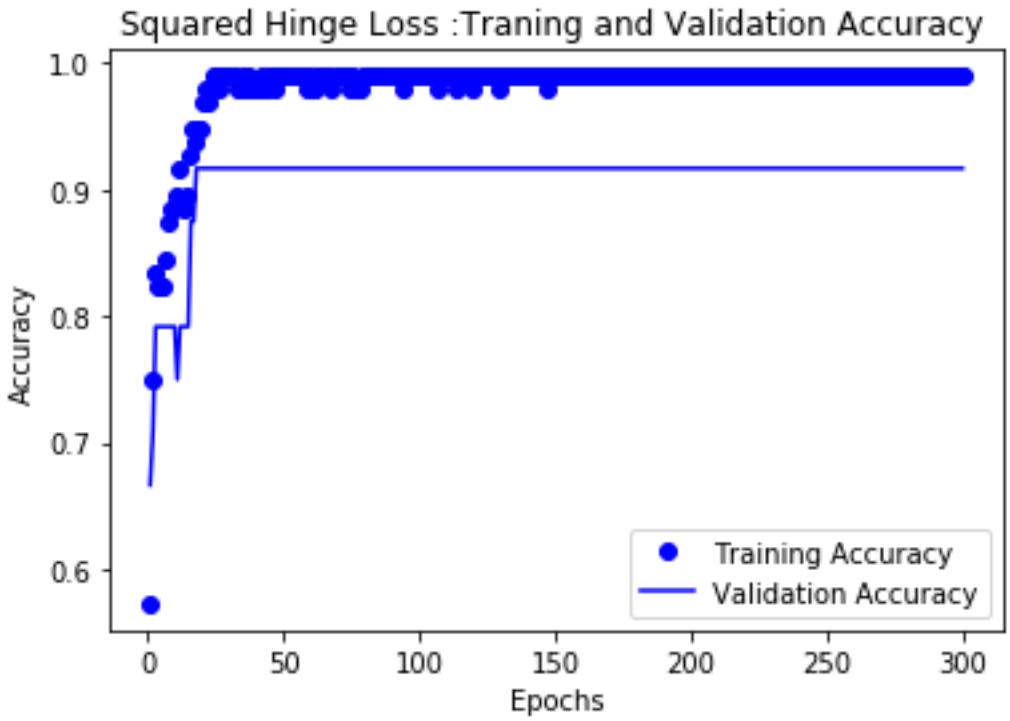
- Finally we evaluate our network on test data and got below results.

```
Loss: 0.775193989276886
Accuracy: 0.8999999761581421
```

Squared Hinge Loss:

- After training the network with above hyperparameters and tuning , we got below results on train and validation sets.





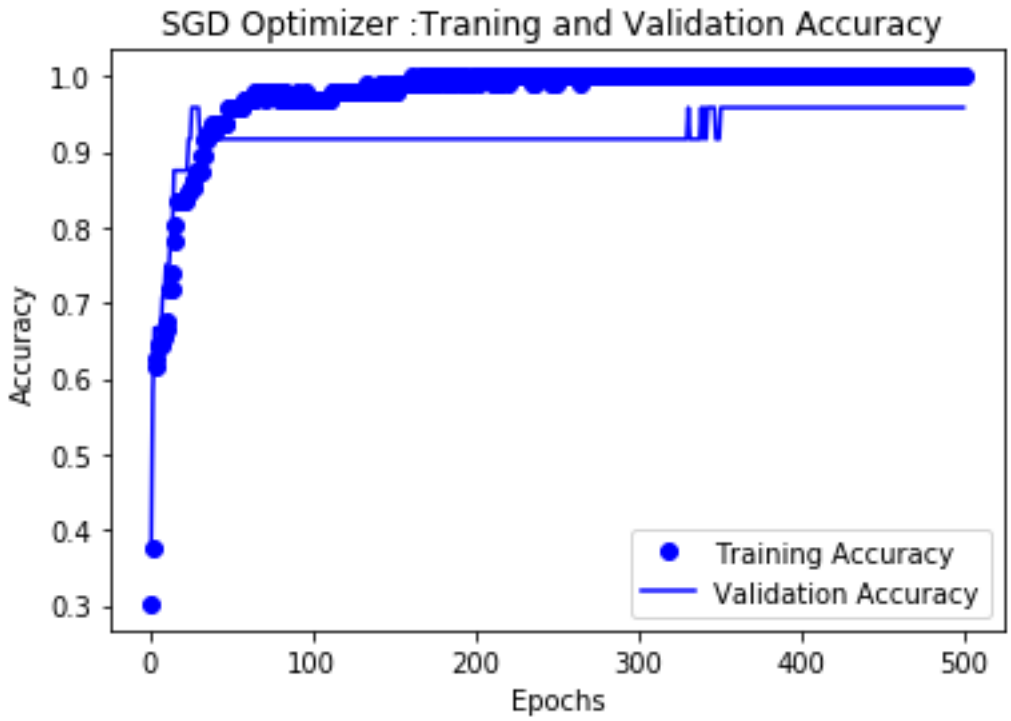
- From the above graph we see that our models validation loss starts to increase after 40 epochs and hence we choose our final number of epochs as 40.
- Finally we evaluate our network on test data and got below results.
- ```
  Loss: 0.7305534482002258
  Accuracy: 0.9333333373069763
  ```

Evaluating Different Optimizers:

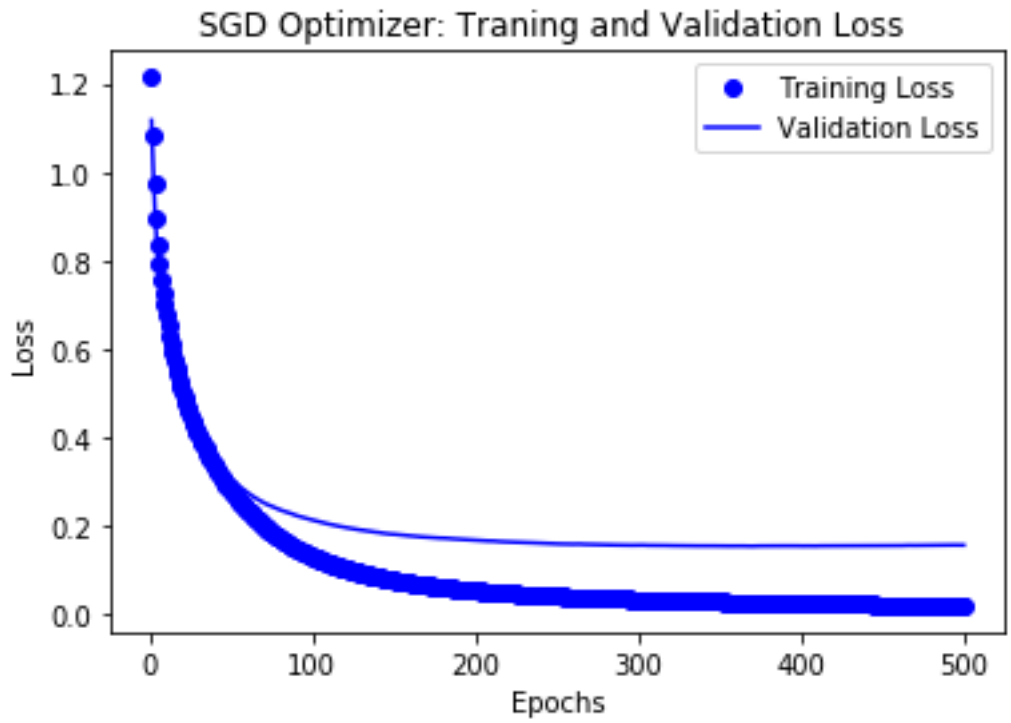
SGD Optimizer:

- After training the network with above hyperparameters and tuning , we got below results on train and validation sets.

  Accuracy Graph : Training Accuracy and Validation Accuracy
  ```



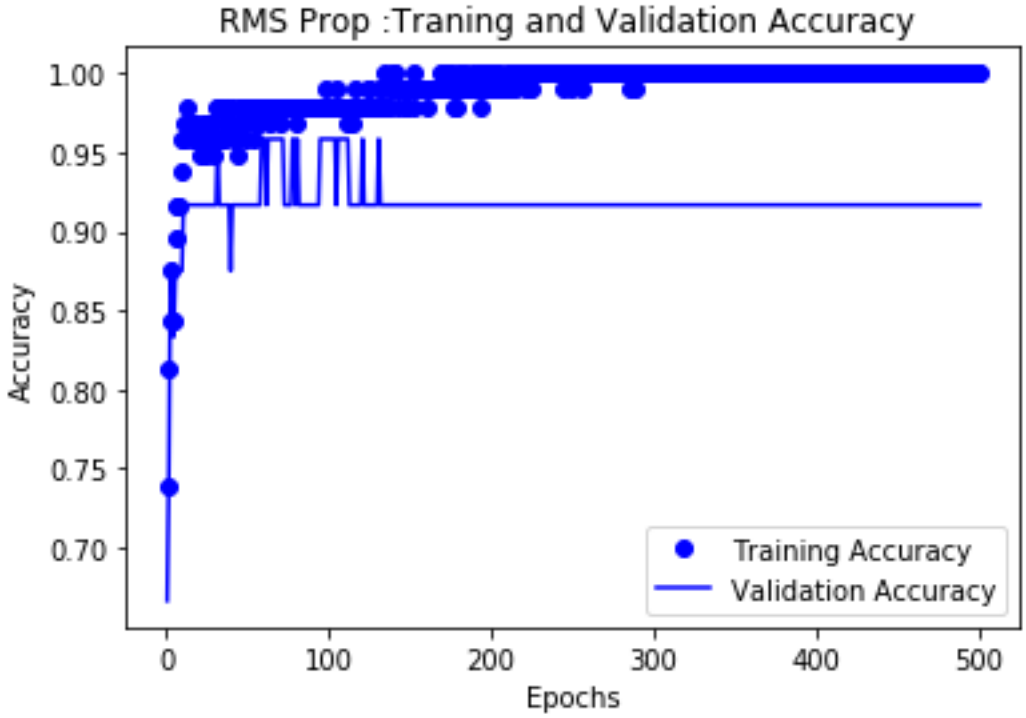
Loss Graph : Training Loss and Validation Loss.

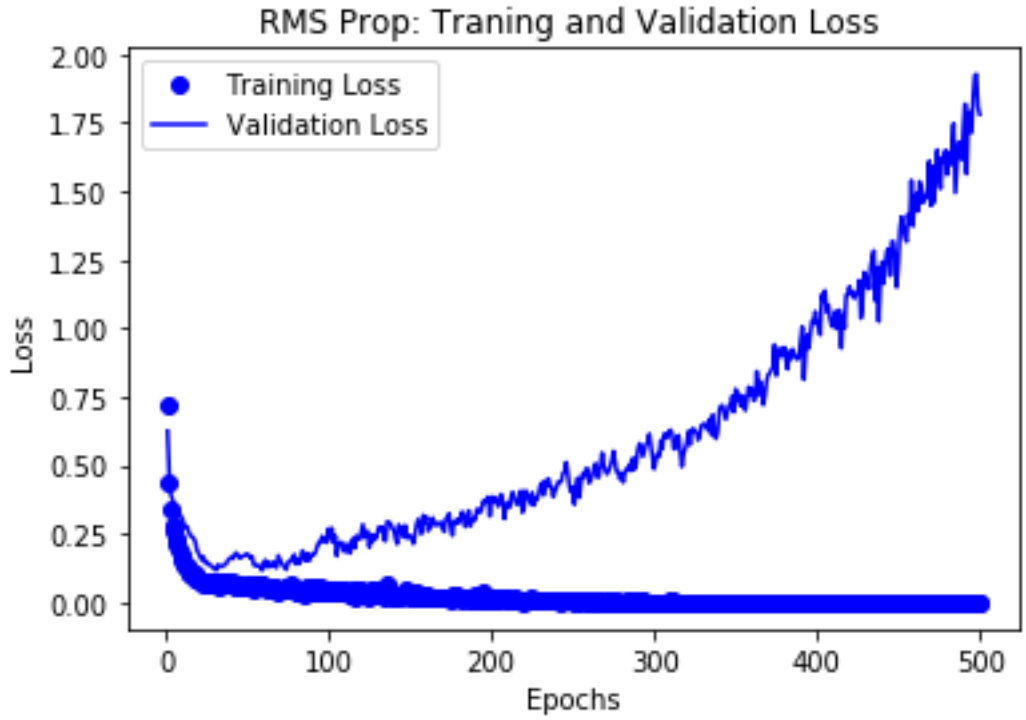


- From the above graph we see that our models validation loss starts to increase after 150 epochs and hence we choose our final number of epochs as 150
- Finally we evaluate our network on test data and got below results.

```
Loss: 0.11354430019855499
Accuracy: 0.9666666388511658
```

RMS Prop:

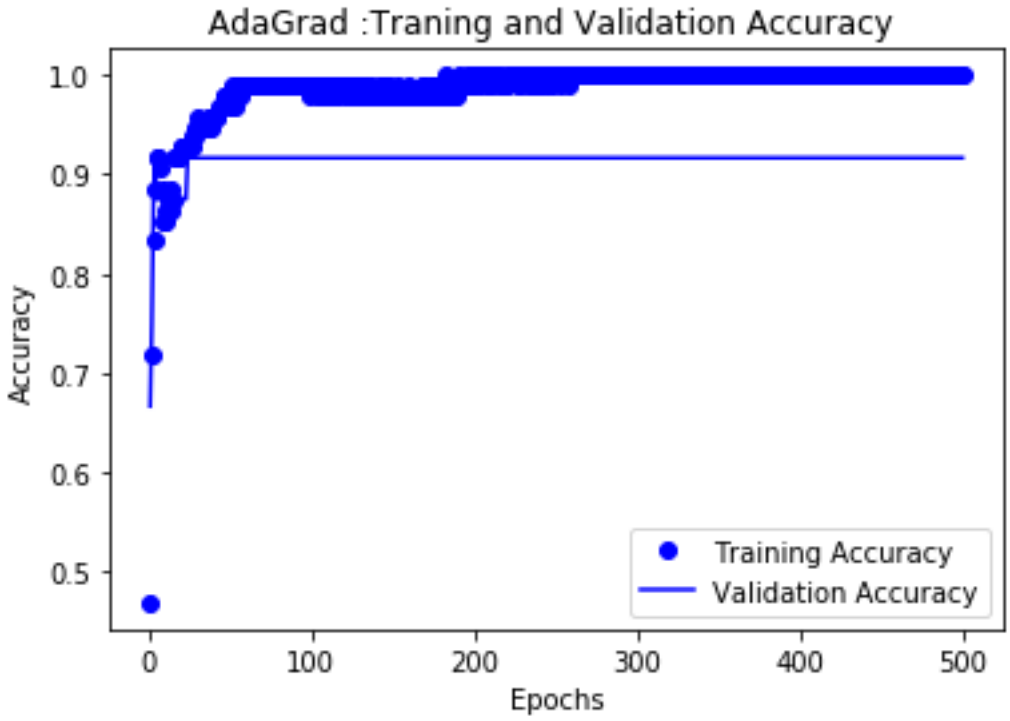


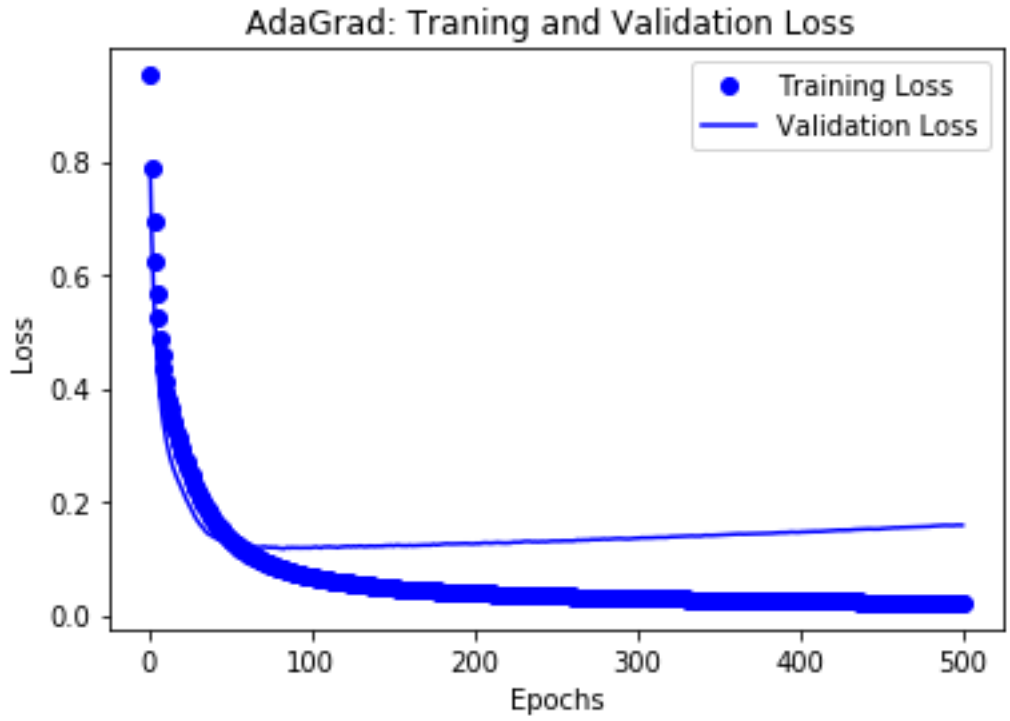


- From the above graph we see that our models validation loss starts to increase after 100 epochs and hence we choose our final number of epochs as 100
- Finally we evaluate our network on test data and got below results.

```
Loss: 0.006514747627079487
Accuracy: 1.0
```

AdaGrad Optimizer :



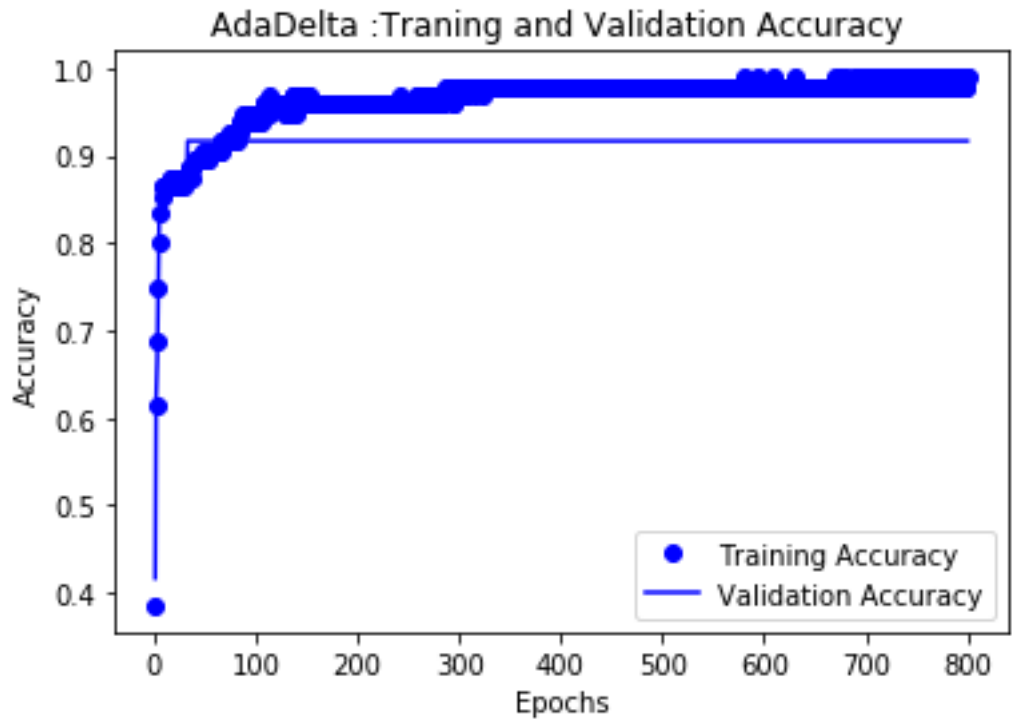


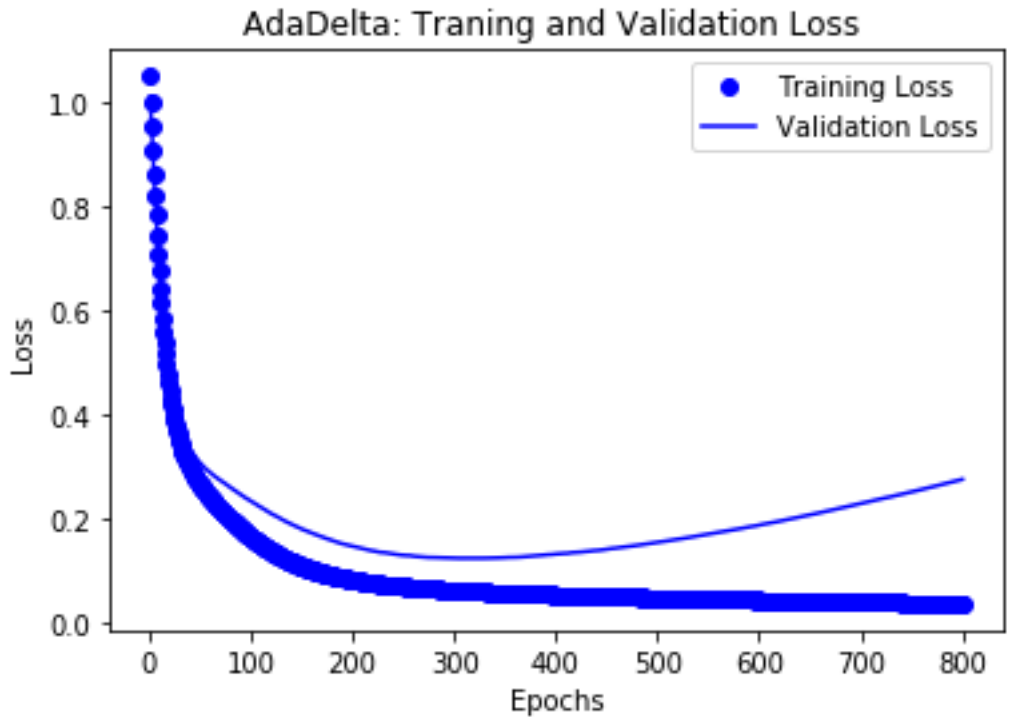
Epoch :100

- From the above graph we see that our models validation loss starts to increase after 100 epochs and hence we choose our final number of epochs as 100
- Finally we evaluate our network on test data and got below results.

```
Loss: 0.103841073811054 23
Accuracy: 0.9666666388511658
```

AdaDelta Optimizer :



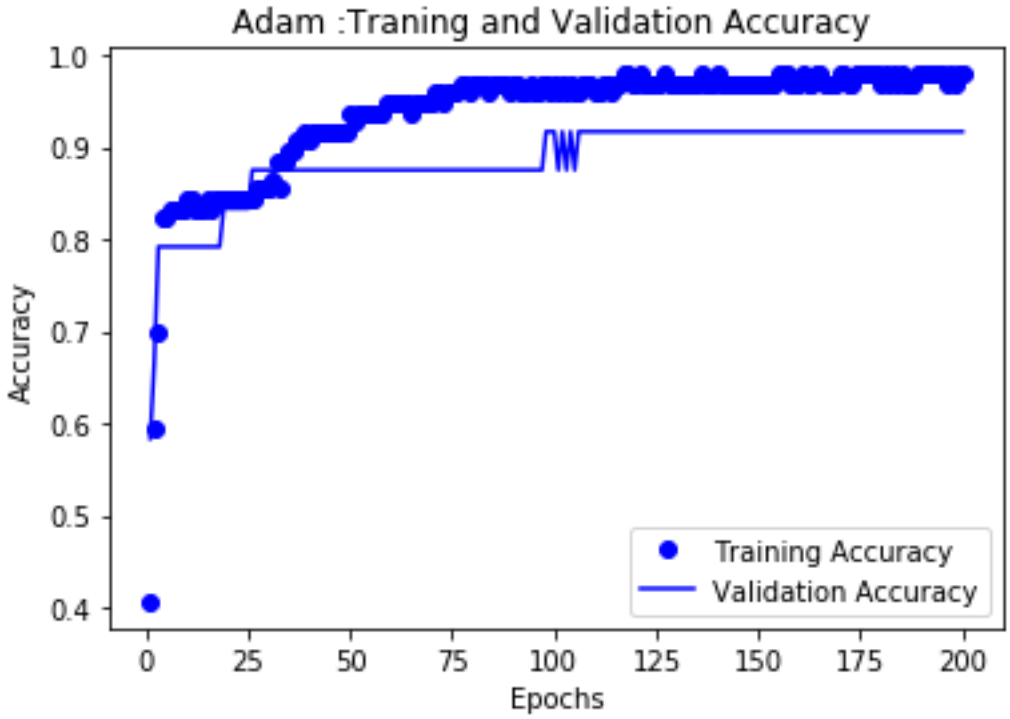


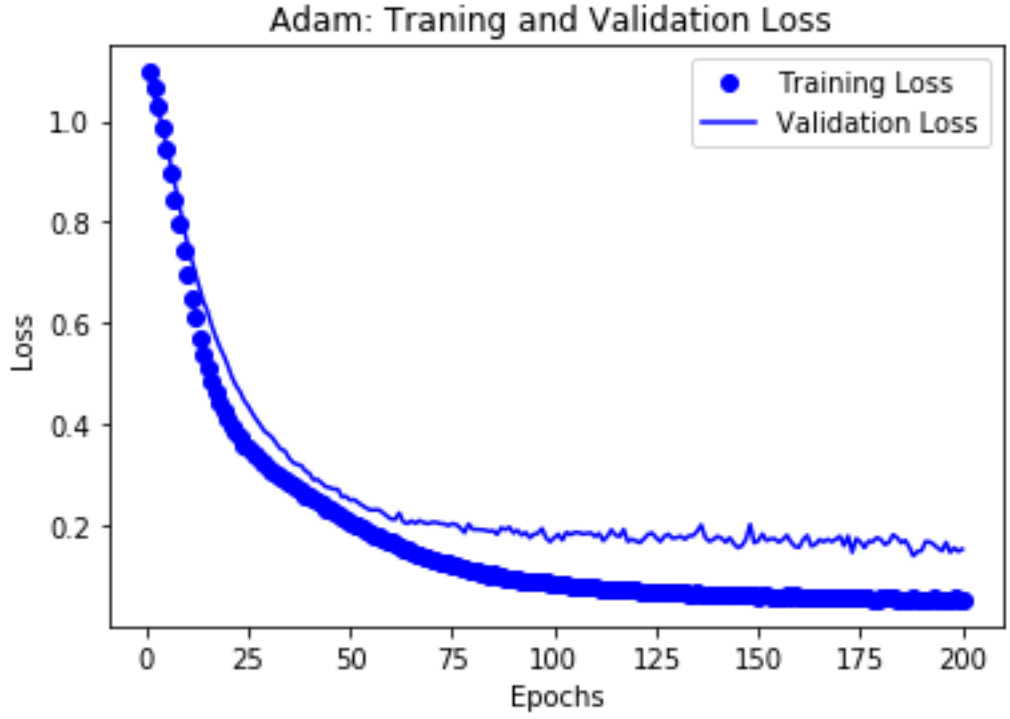
Epochs: 300

- From the above graph we see that our models validation loss starts to increase after 300 epochs and hence we choose our final number of epochs as 300
- Finally we evaluate our network on test data and got below results.

```
Loss: 0.034047365188598 63
Accuracy: 1.0
```

Adam Optimizer :



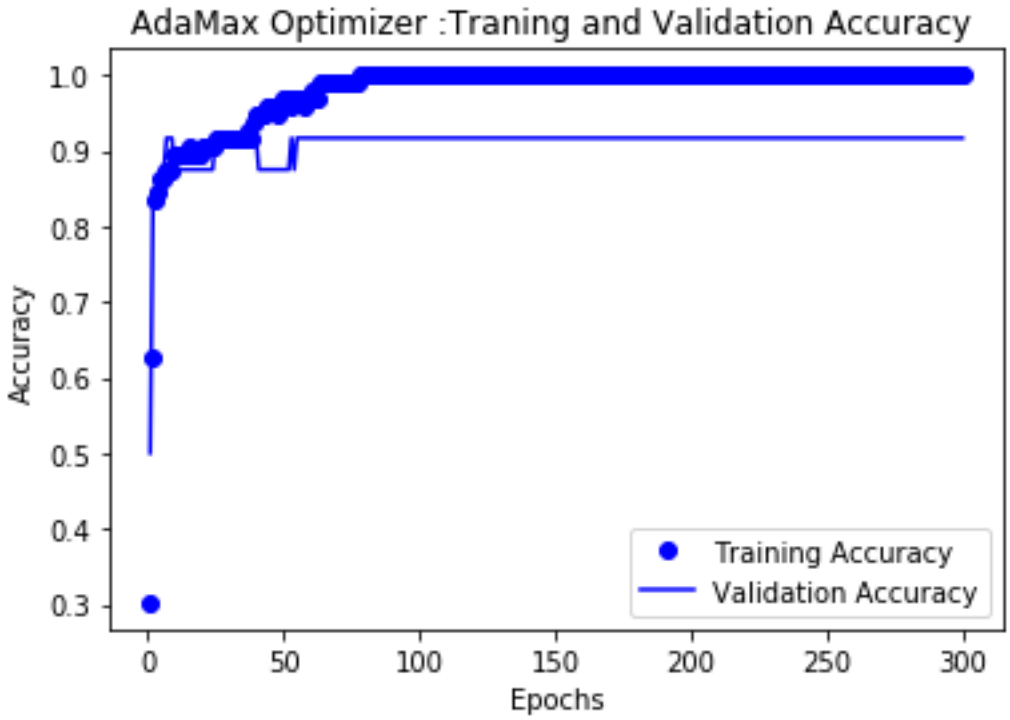


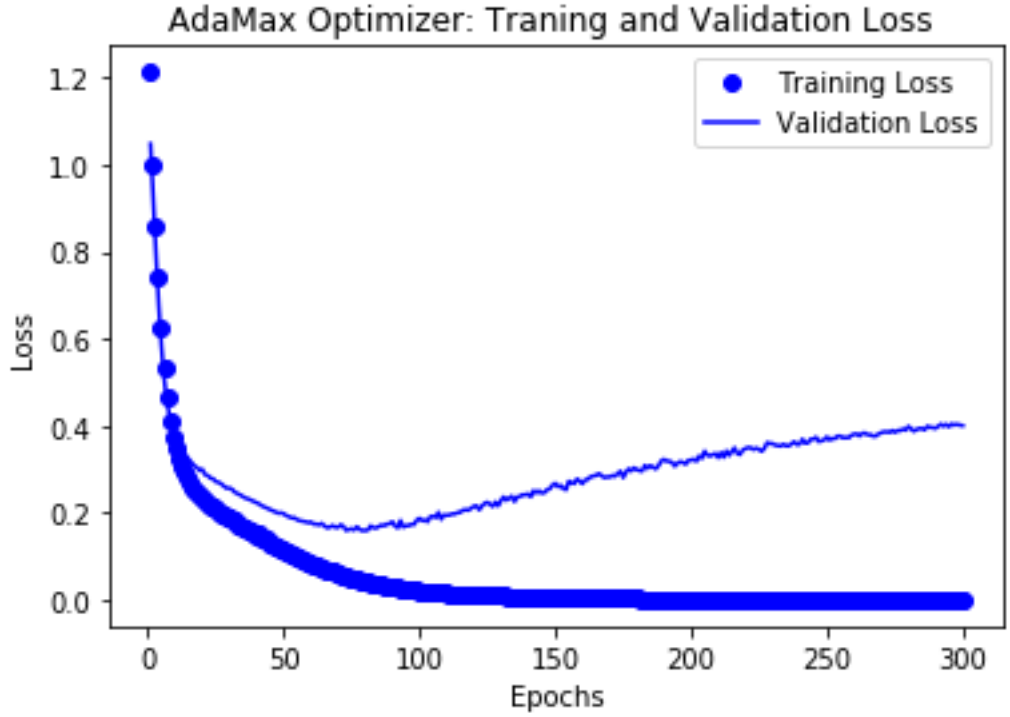
Epochs : 75

- From the above graph we see that our models validation loss starts to increase after 75 epochs and hence we choose our final number of epochs as 75
- Finally we evaluate our network on test data and got below results.

```
Loss: 0.08789864927530289
Accuracy: 0.9666666388511658
```

AdaMax Optimizer :



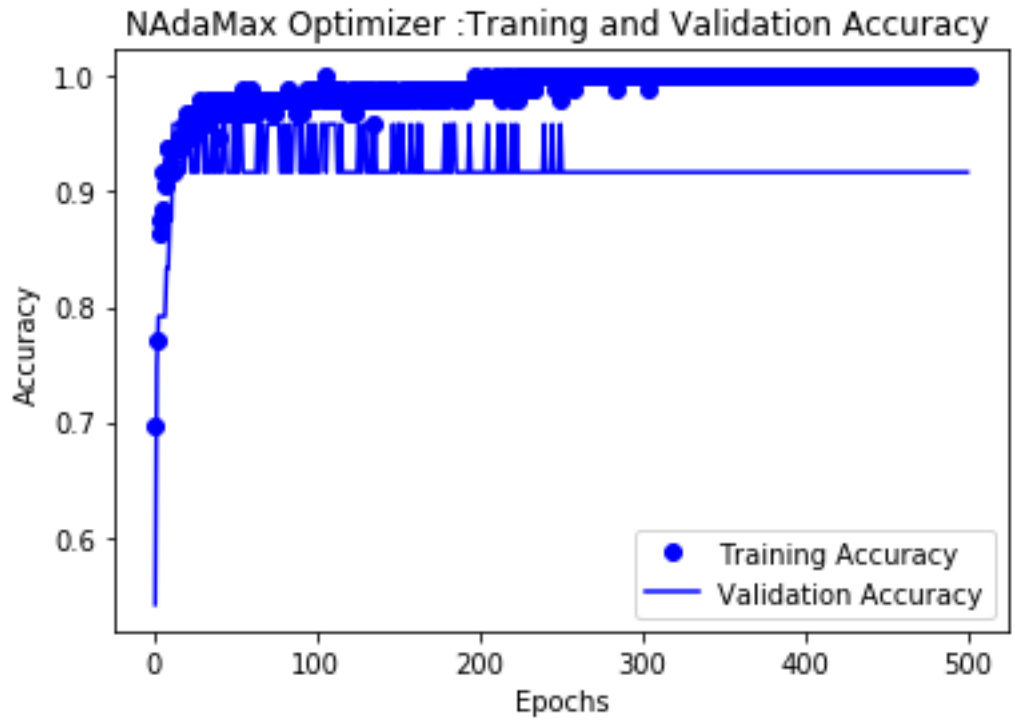


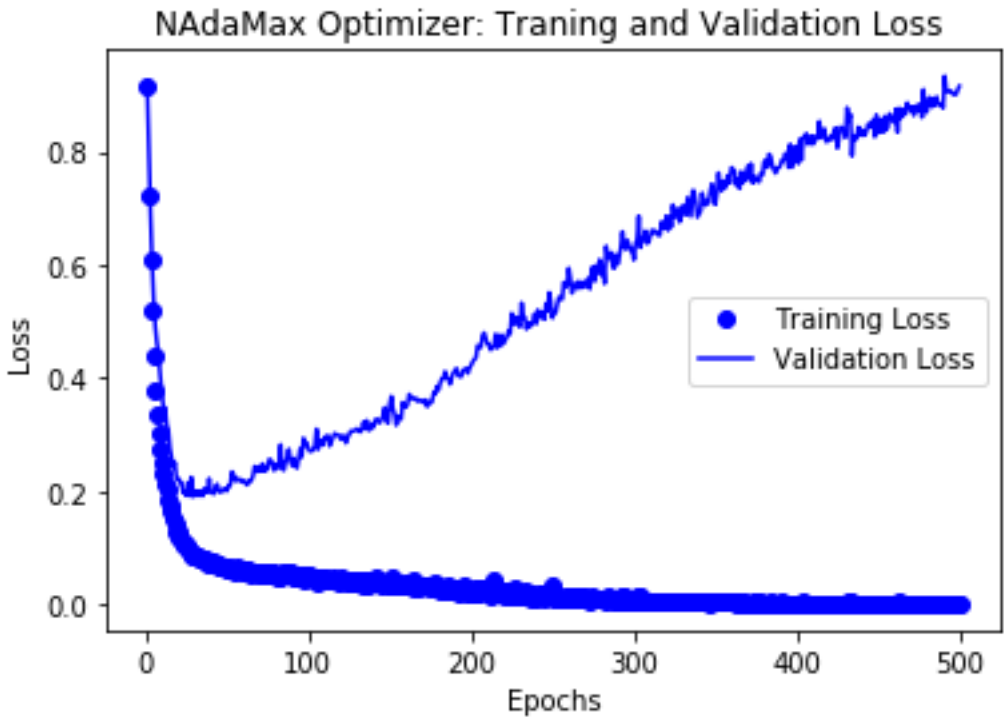
Epochs =100

- From the above graph we see that our models validation loss starts to increase after 100 epochs and hence we choose our final number of epochs as 100
- Finally we evaluate our network on test data and got below results.

```
Loss: 0.090604729950428O1
Accuracy: 0.9333333373069763
```

Nadamax Optimizer :





Epochs=50

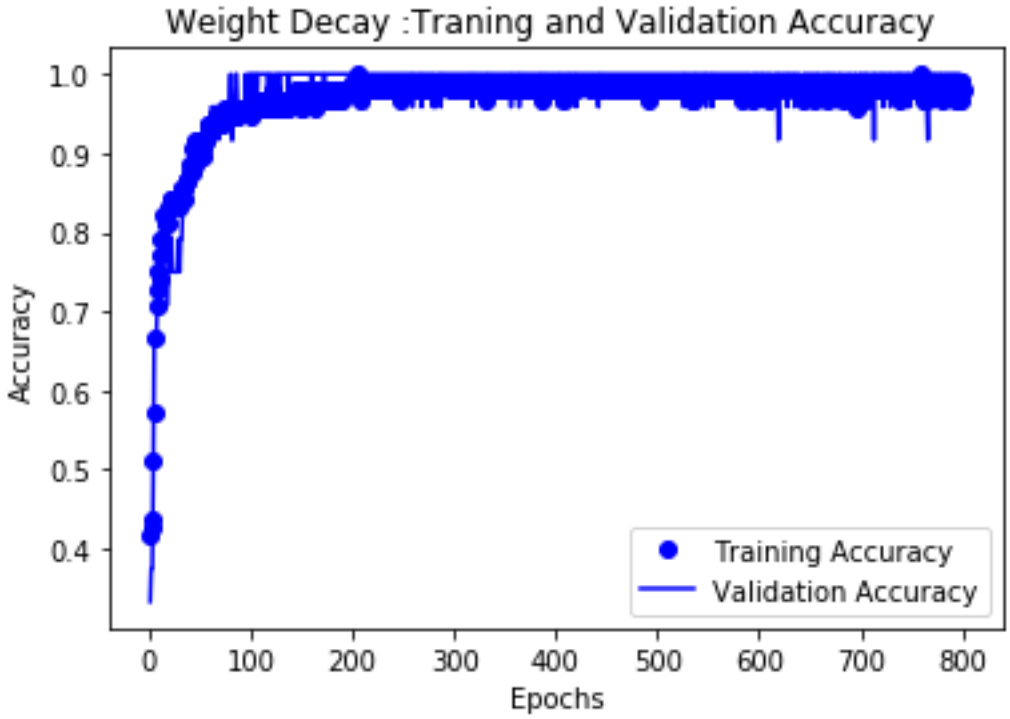
- From the above graph we see that our models validation loss starts to increase after 50 epochs and hence we choose our final number of epochs as 50
- Finally we evaluate our network on test data and got below results.

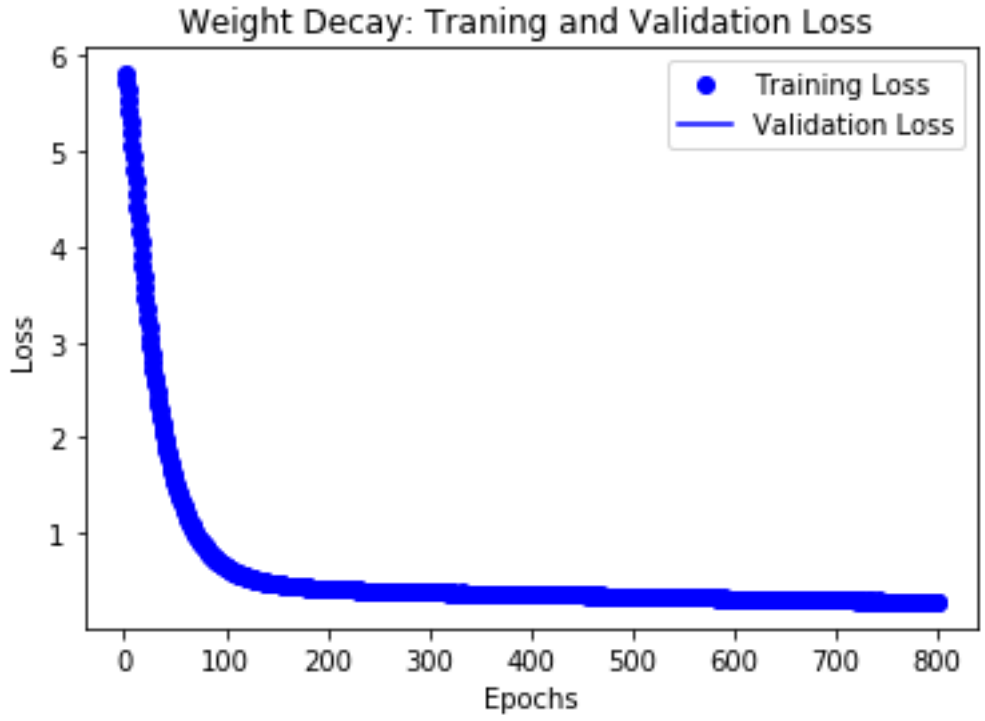
```
Loss: 0.23786404728889465
Accuracy: 0.8666666746139526
```

By comparing all of the above Optimizers , we can see that Adam Optimizers gives the best results by converging in around 75 epochs. AdaDelta takes the most time to converge , converging in around 300 epochs.

Evaluating Different Regularization Measures:

Weight Decay on Adam Optimizer:



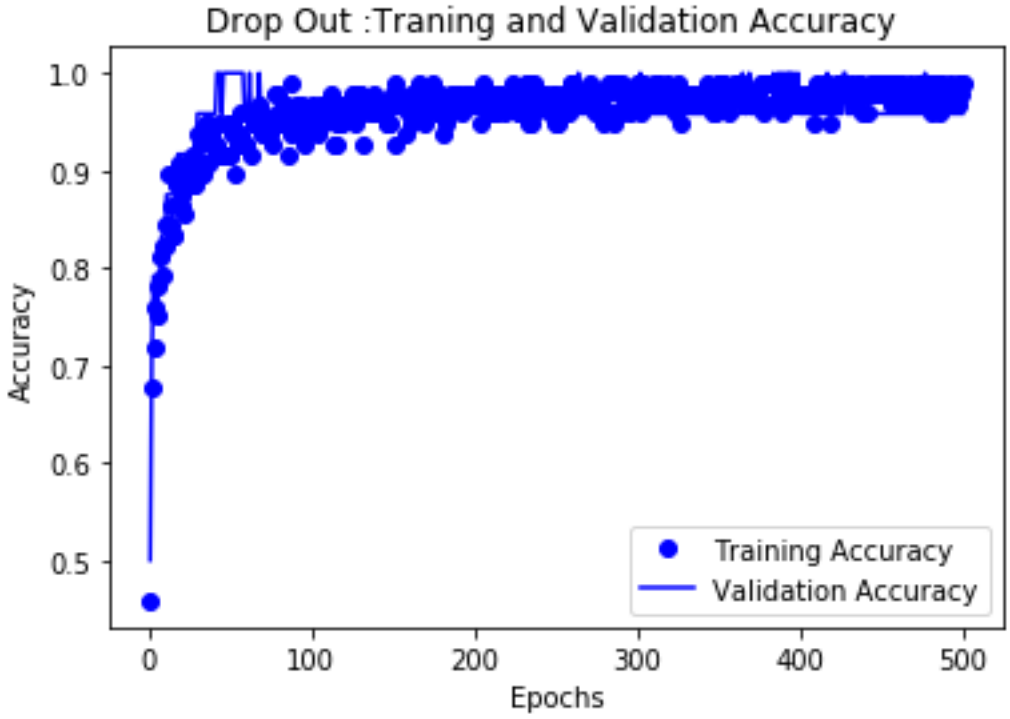


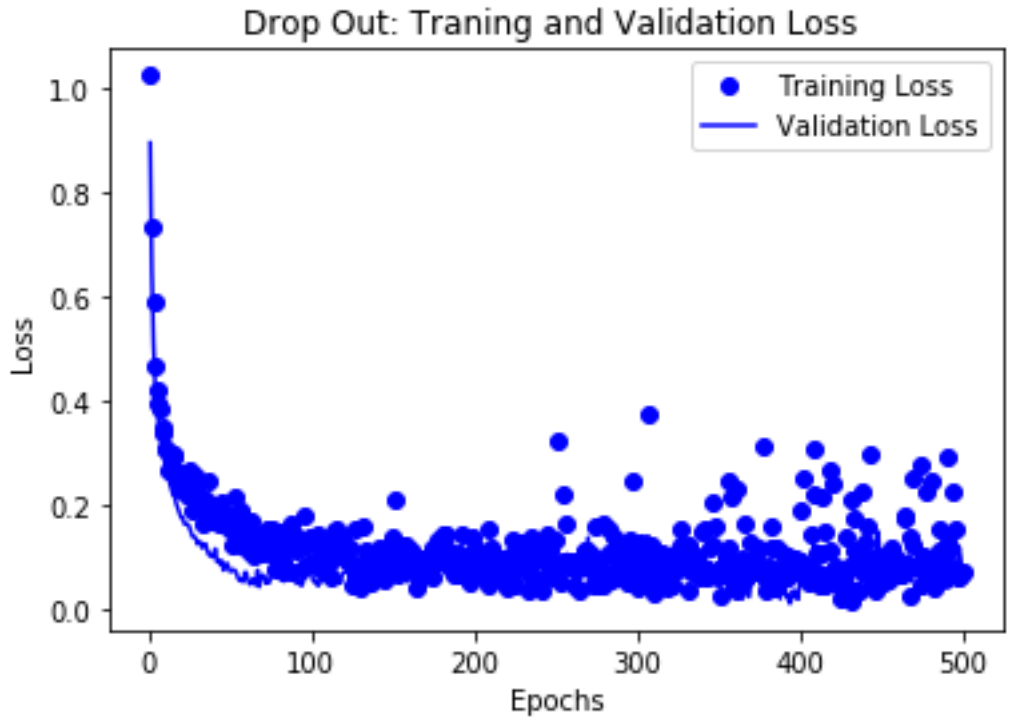
From the above graph we can see that weight decay regularization improves generalization on the train data and reduce overfitting.

Evaluating the above network on the test data with epoch around 200.

```
Loss: 0.37898150086402893
Accuracy: 0.9666666388511658
```

Dropout with RMS Prop :



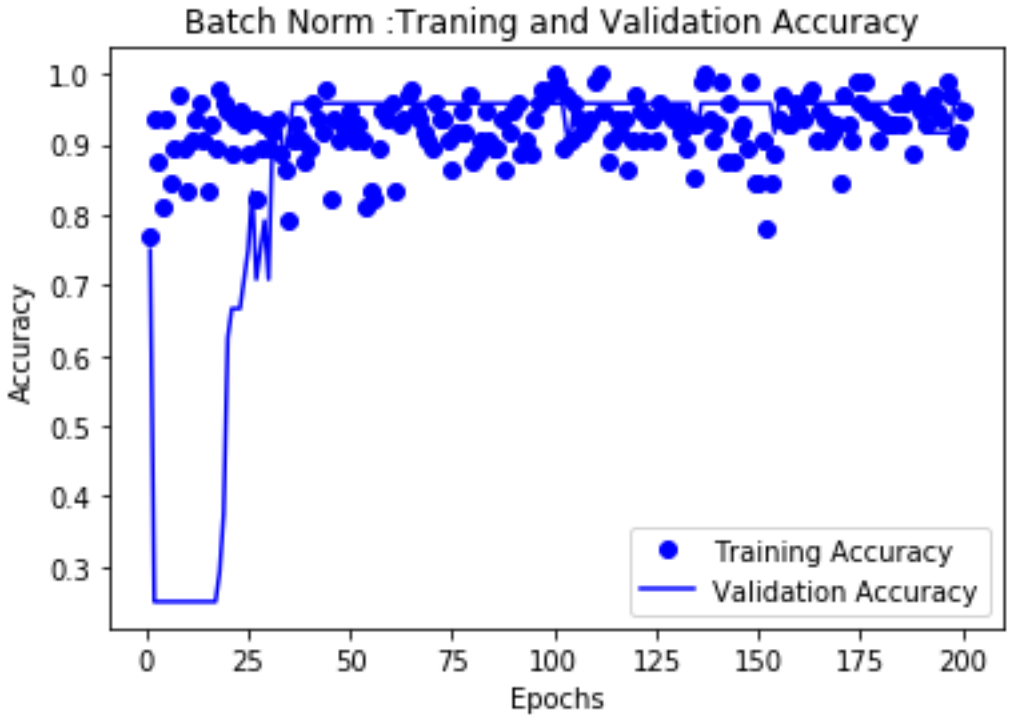


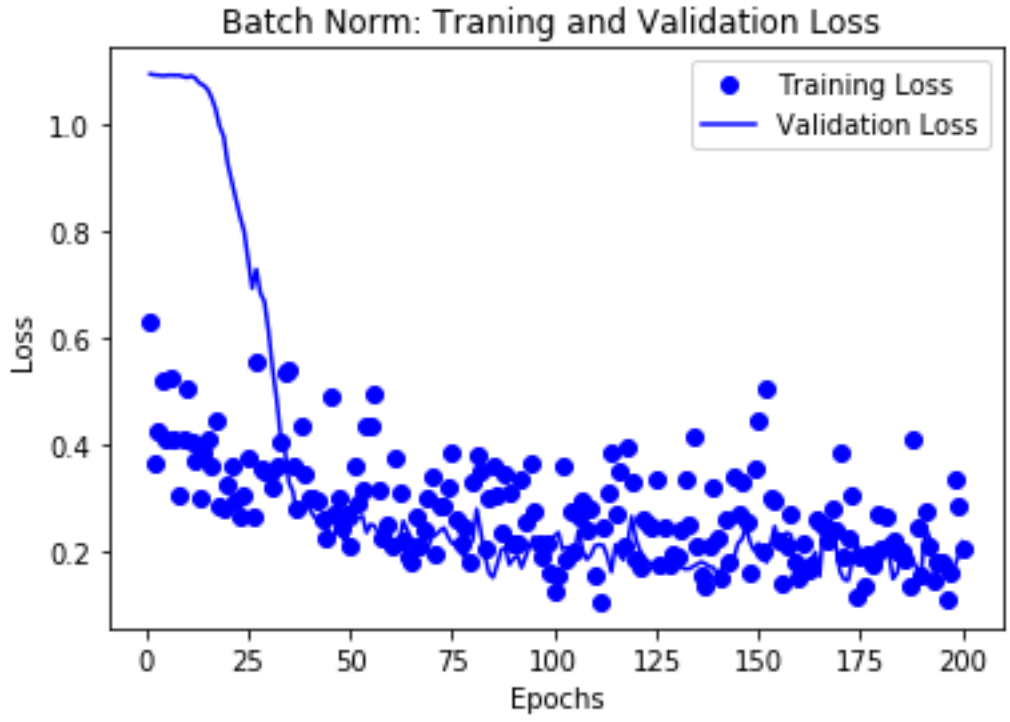
From the above graph we can see that Drop Out regularization improves generalization on the train data and reduce overfitting.

Evaluating the above network on the test data with epoch around 150.

```
Loss: 0.07055672258138657
Accuracy: 0.9666666388511658
```

Batch Noemalization with SGD:





From the above graph we can see that Batch Normalization regularization improves generalization on the train data and reduce overfitting.

Evaluating the above network on the test data with epoch around 75.

Evaluating on Test Data :

```
Loss: 0.2398199737071991
Accuracy: 0.9666666388511658
```

Ensemble Classifier Using Two Models(ADAM Classifier and RMS prop Classifier):

First we use Adam Classifier to predict values and then use RMSProp Classifier to predict values.

Then we can take average of two predicted values then compare it true label to get accuracy of the ensemble classifier.

```
Accuracy of Ensemmble Classifier 0.9666666666666667
```