1.Problem Statement :

This is three class classification problem. Classification of iris flowers from sepal and petal dimensions.

2.Proposed Solution :

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

3.Implementation details:

- We first loaded the data the Iris Flower data from the below link. https://archive.ics.uci.edu/ml/datasets/Iris
- Then we are going to One Hot Encode the labels of the dataset.
- We split the Iris data into Train , Test and validation Set in 80:20 ratio and we normalized the features using below formula.

  Initial Hyperarameters :
  - Input nodes : 4
  - Output nodes : 3, activation for output layer : 'softmax'
  - first hidden layer Hidden nodes in: 16 , activation function: 'relu'
  - second hidden layer Hidden nodes in: 16 , activation function: 'relu'
  - Loss function: categorical cross entropy

3.Implementation details :

- 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.

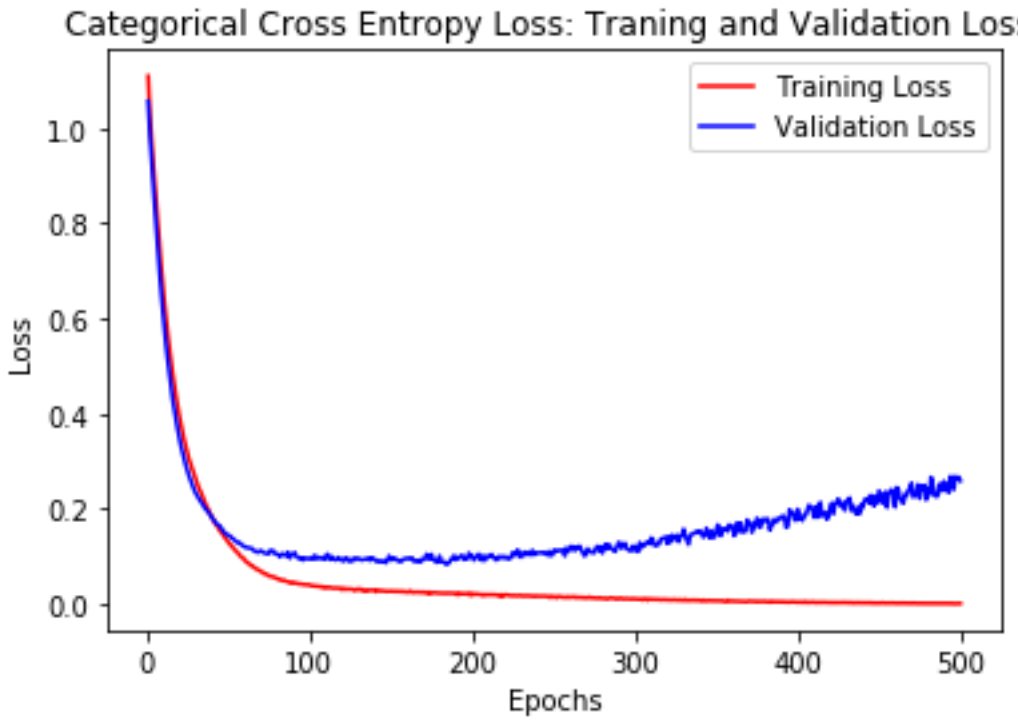
4. Results and discussion:

Evaluating Different Loss Functions:

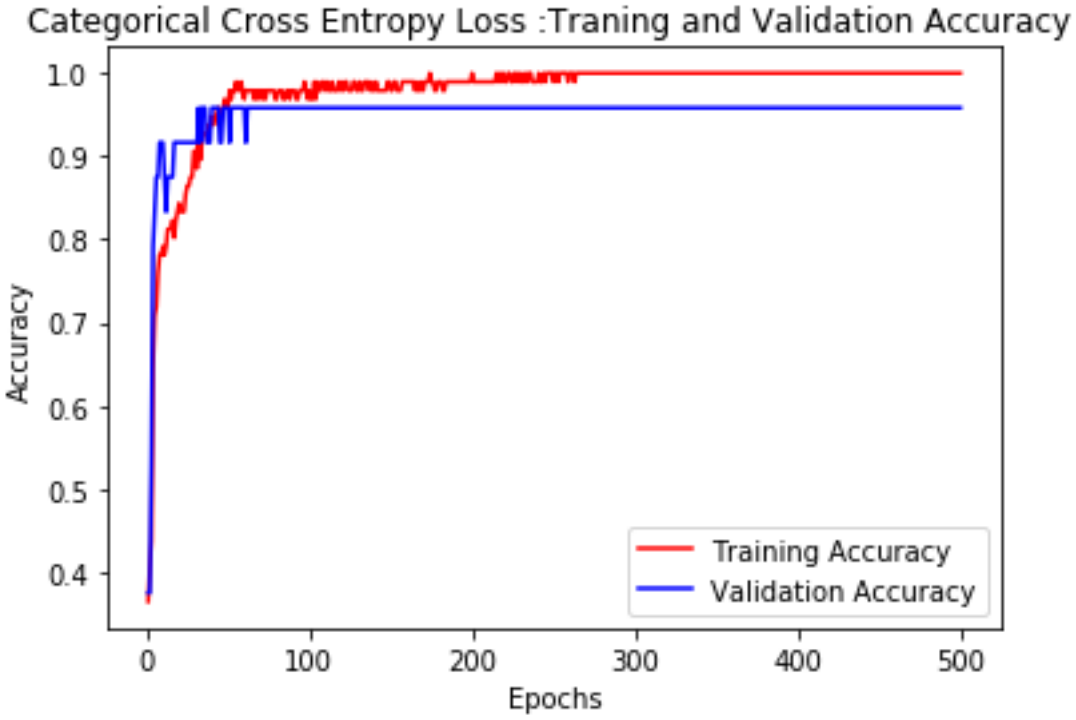
- Initial hyperparameters used during training.
  Learning Rate : 0.001
  Epochs : 200
  Batch Size=1

Categorical Cross Entropy:

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



Loss Graph : Training Loss and Validation Loss.

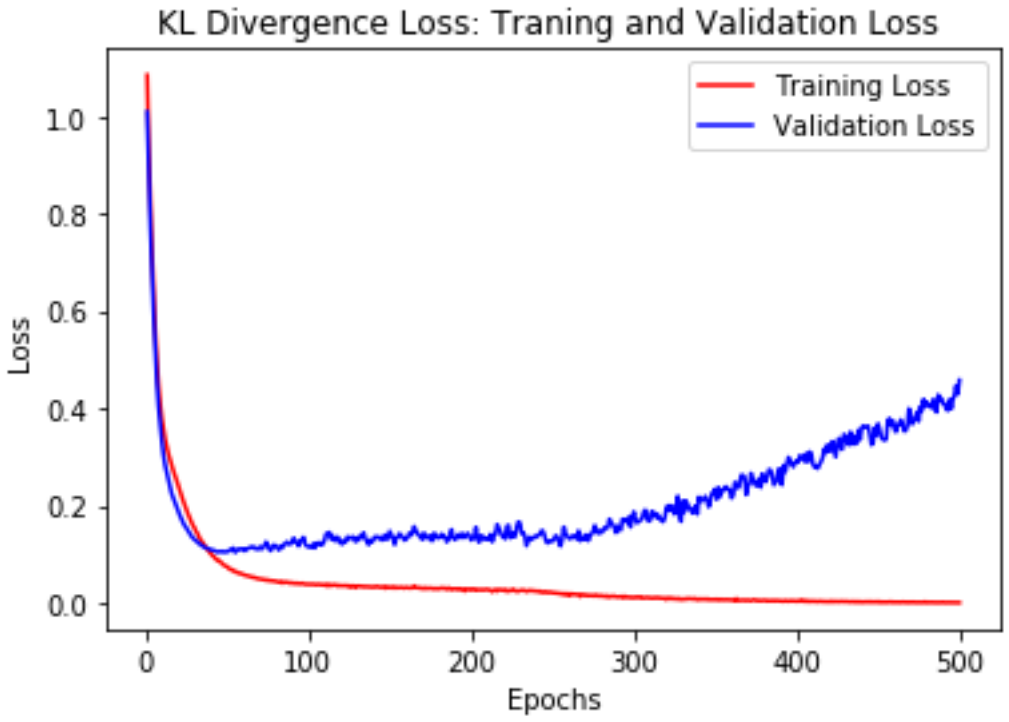


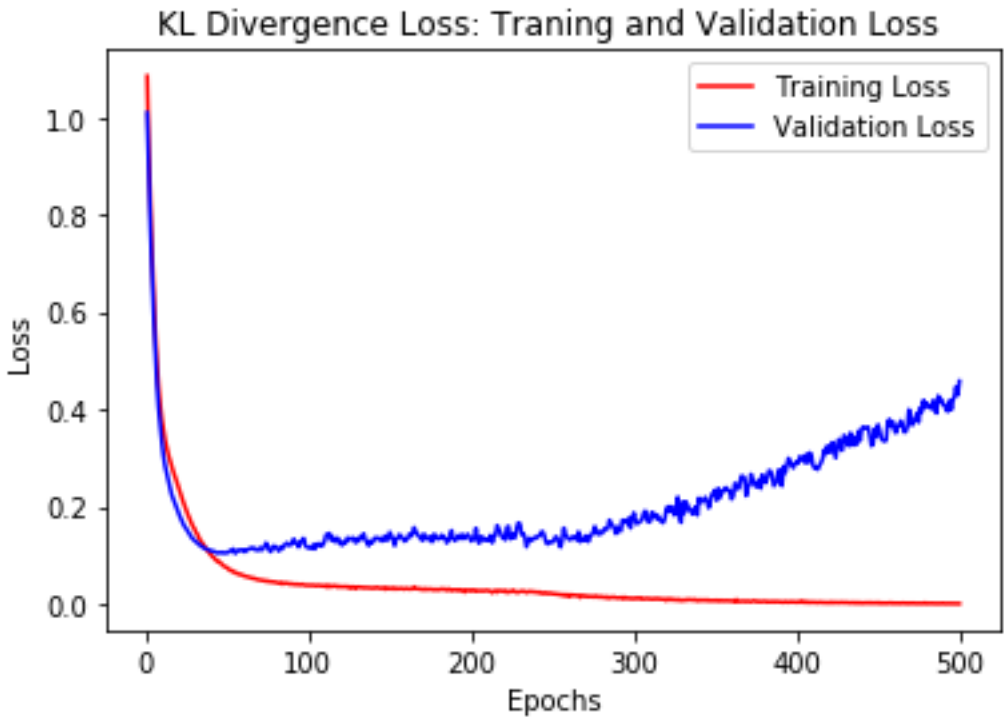
Accuracy Graph : Training Accuracy and Validation Accuracy

- From the above graph we see that our model's validation loss starts to increase after 60 epochs and hence we choose our final number of epochs as 60
- Finally we evaluate our network on test data and got below results.
    - Loss : 0.10407
    - Accuracy : 0.96666

Kullback Leibler Divergence Loss:

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

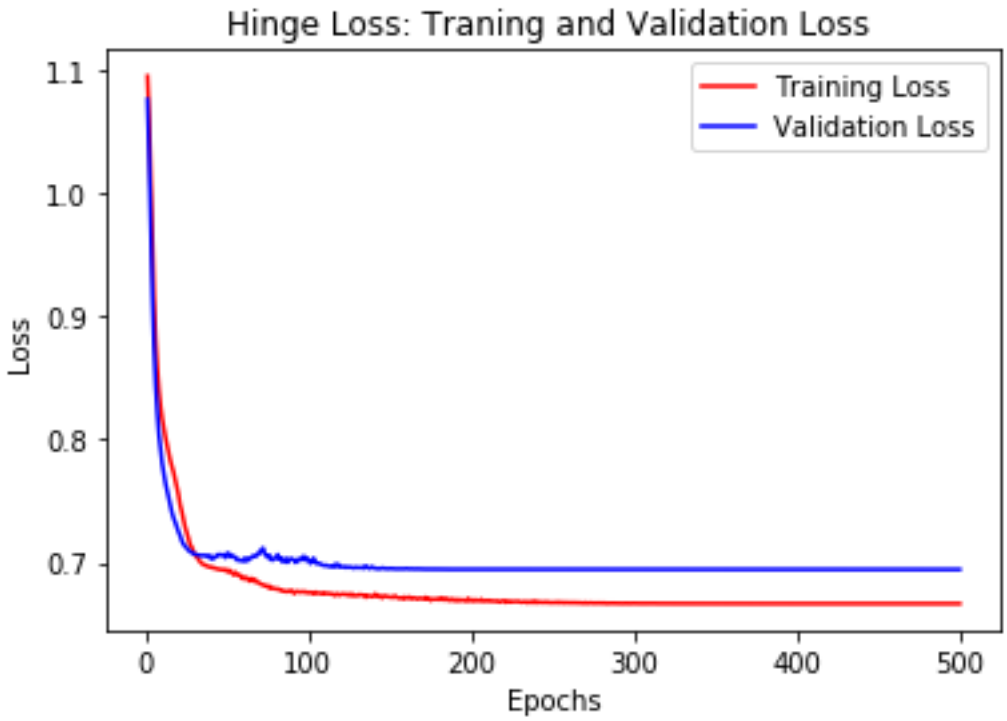


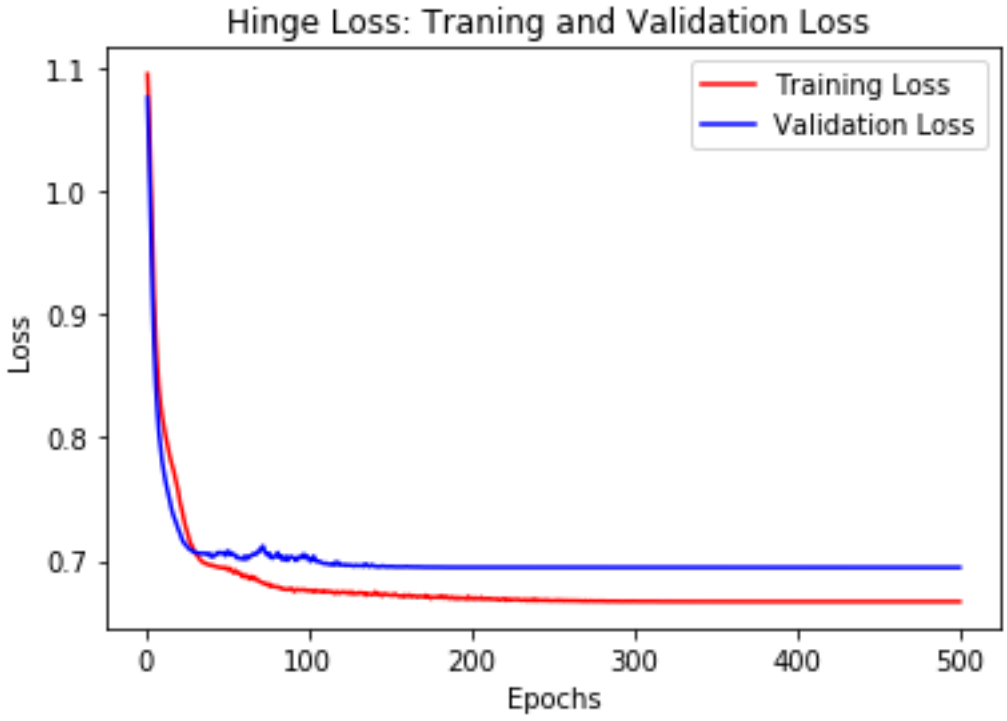


- From the above graph we see that our models validation loss starts to increase after 50 epochs and hence we choose our final number of epochs as 50.
- Finally we evaluate our network on test data and got below results.
    - Loss: 0.06634516268968582
        - Accuracy: 0.9666666388511658

Hinge Loss:

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

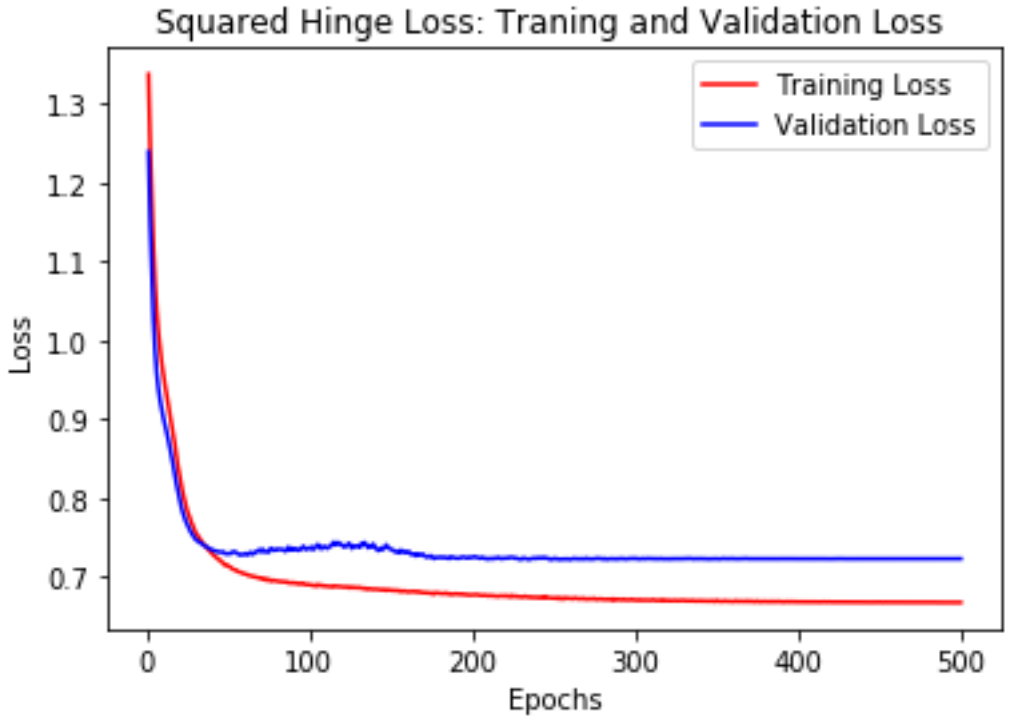


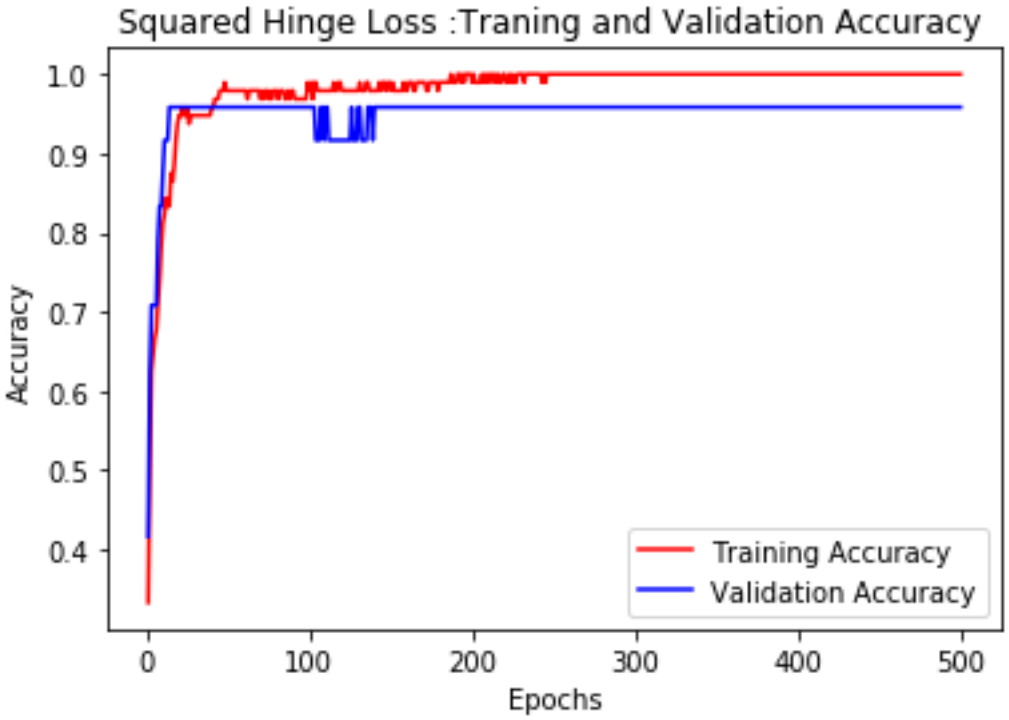


- From the above graph we see that our models validation loss starts to increase after 50 epochs and hence we choose our final number of epochs as 50.
- Finally we evaluate our network on test data and got below results.
    - Loss: 0.6972643733024597
    - Accuracy: 0.9666666388511658

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 60 epochs and hence we choose our final number of epochs as 60.
- Finally we evaluate our network on test data and got below results.
    - Loss: 0.7161979675292969
    - Accuracy: 0.9666666388511658

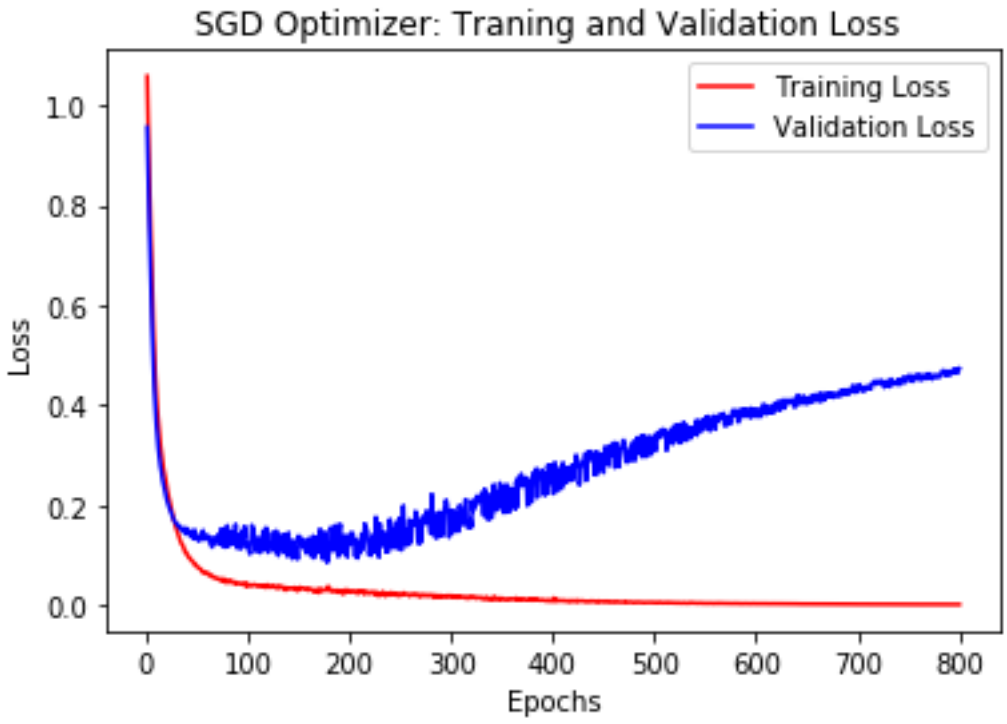
Evaluating Different Optimizers:

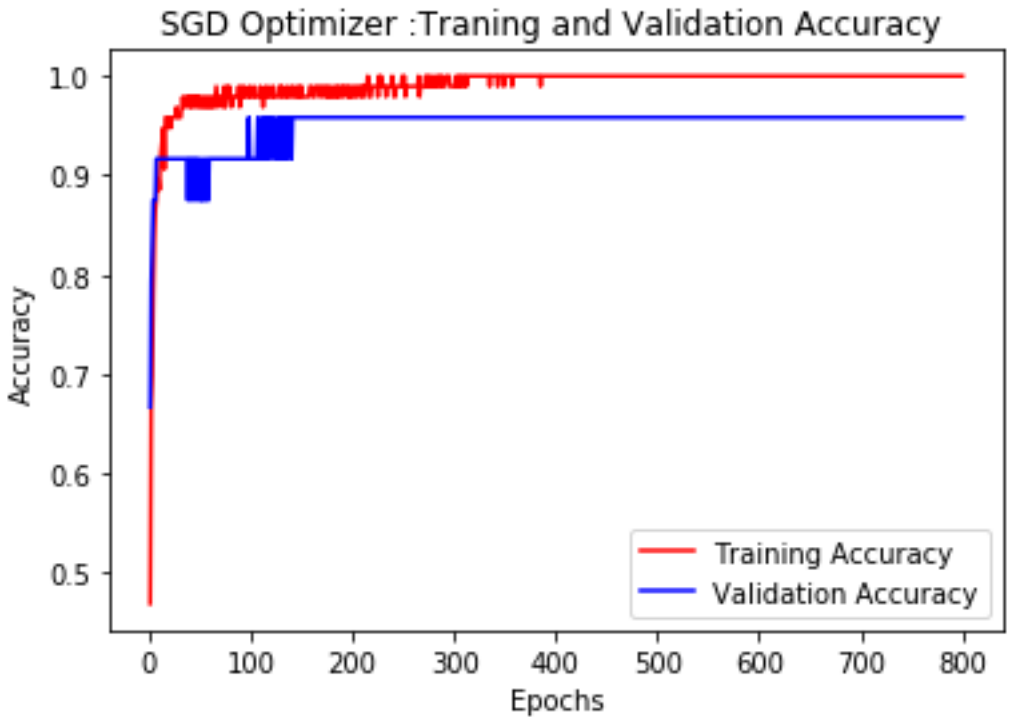
SGD Optimizer:

- 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

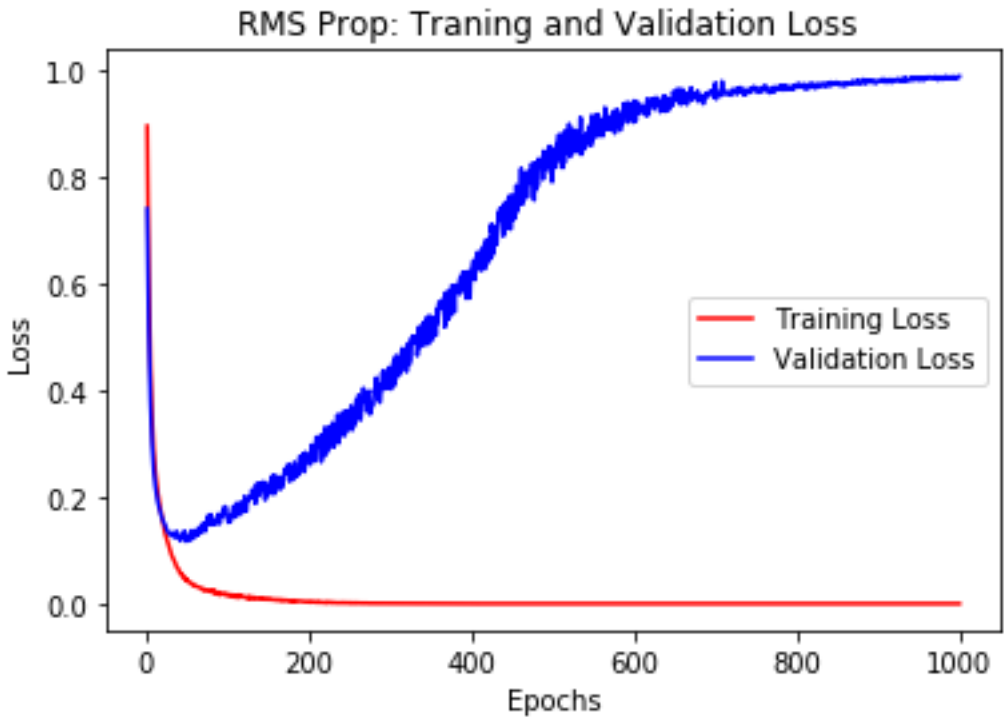
  Loss Graph : Training Loss and Validation Loss.

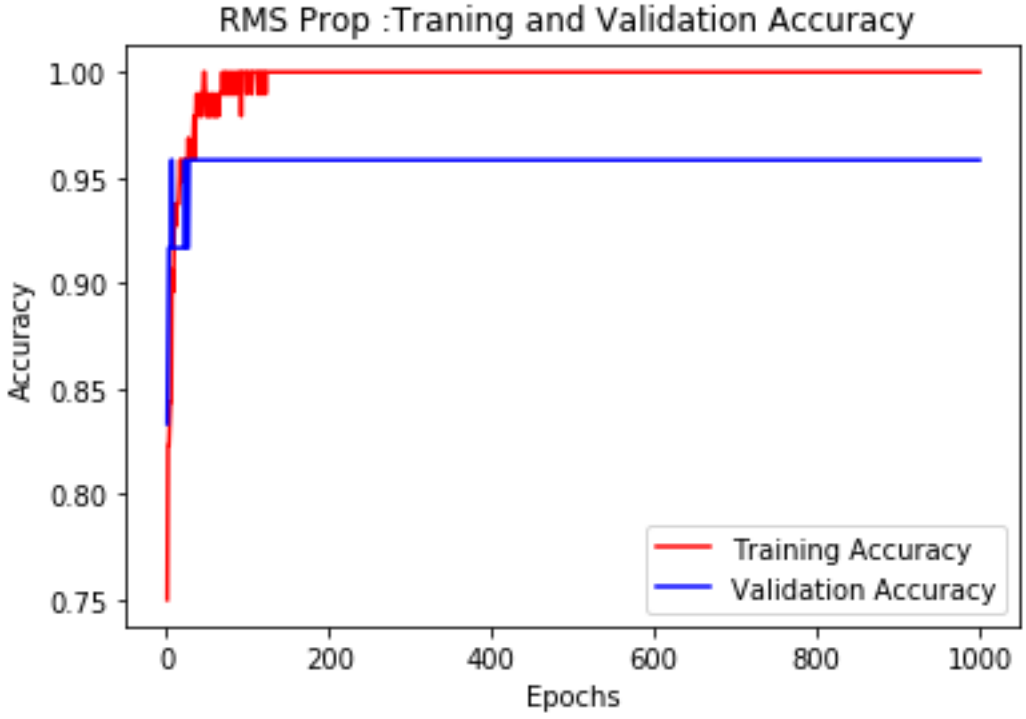




- From the above graph we see that our models validation loss starts to increase after 150 epochs and hence we choose our final number of epochs as 150
- Finally we evaluate our network on test data and got below results.
    - Loss: 0.0532163567841053
    - Accuracy: 0.9666666388511658

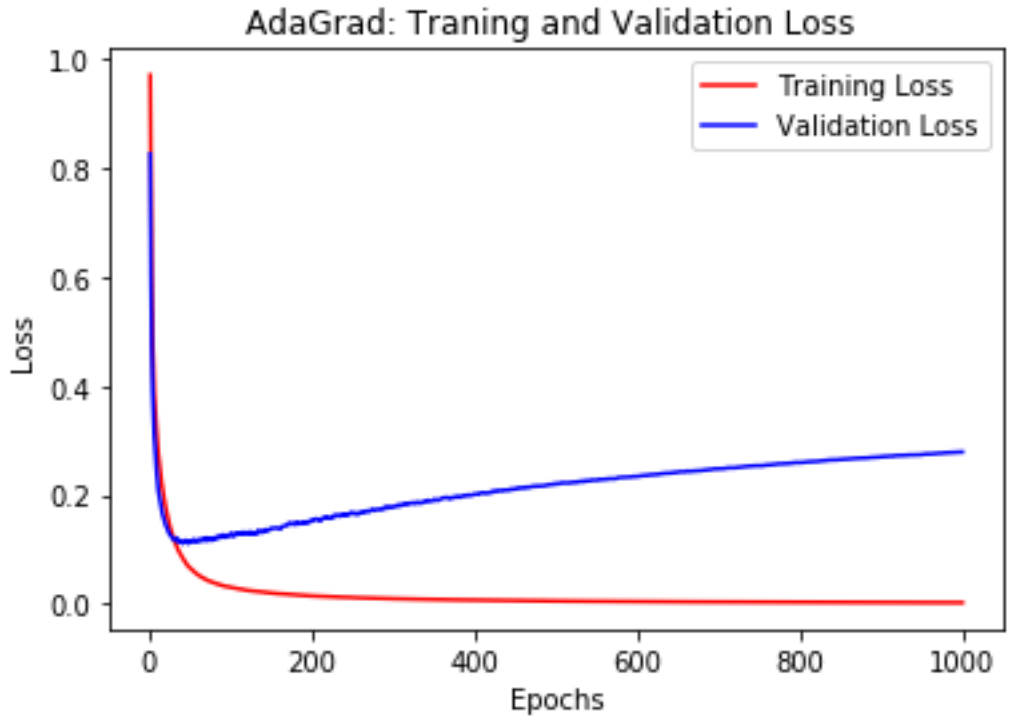
RMS Prop:

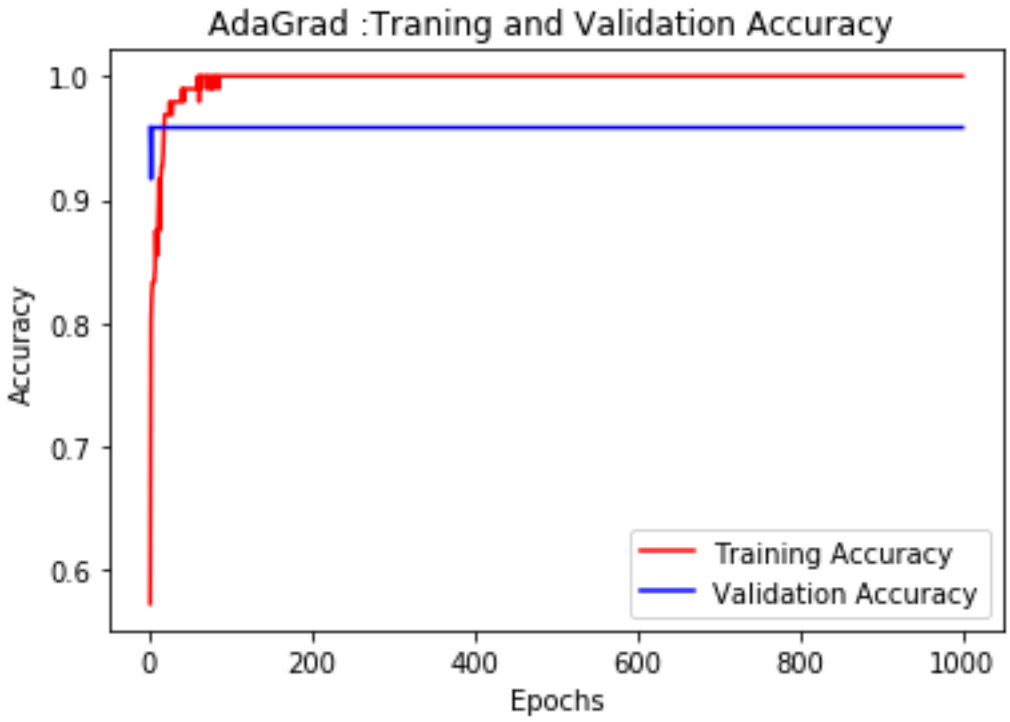




- From the above graph we see that our models validation loss starts to increase after 20 epochs and hence we choose our final number of epochs as 20
- Finally we evaluate our network on test data and got below results.
    - Loss: 0.2021666169166565
    - Accuracy: 0.8999999761581421

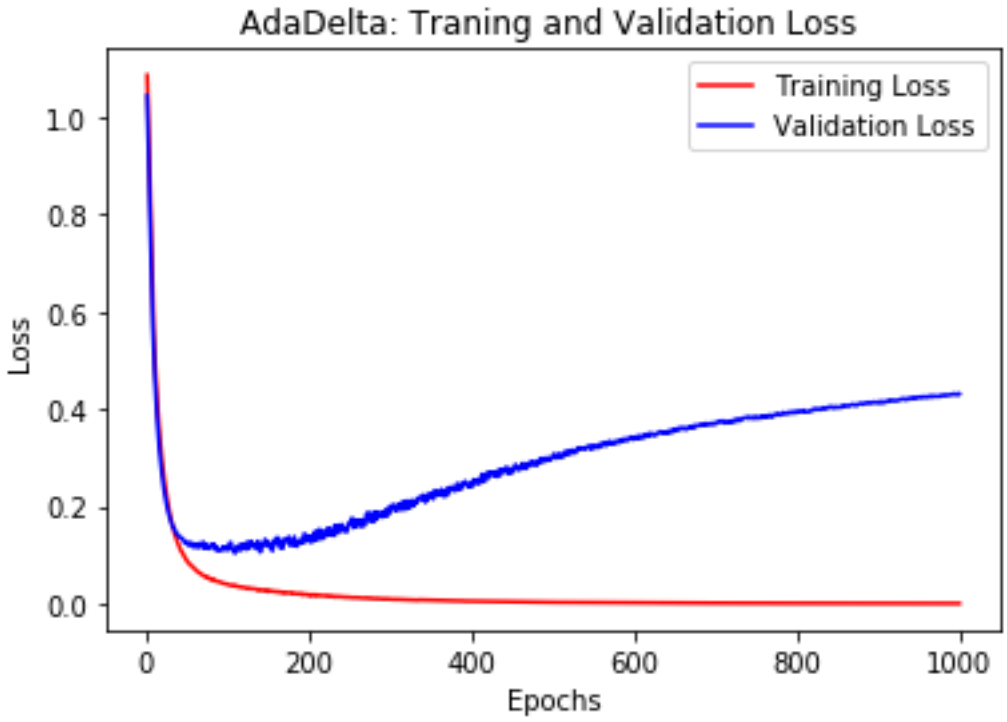
AdaGrad Optimizer :

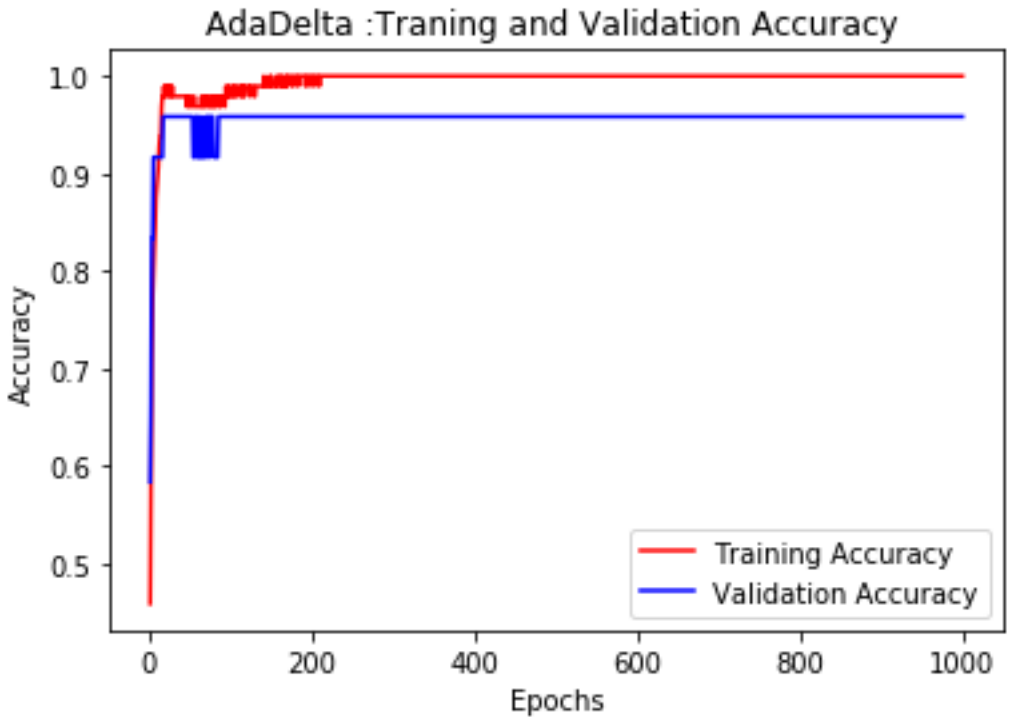




- From the above graph we see that our models validation loss starts to increase after 47 epochs and hence we choose our final number of epochs as 47
- Finally we evaluate our network on test data and got below results.
    - Loss: 0.11705661565065384
    - Accuracy: 0.9333333373069763

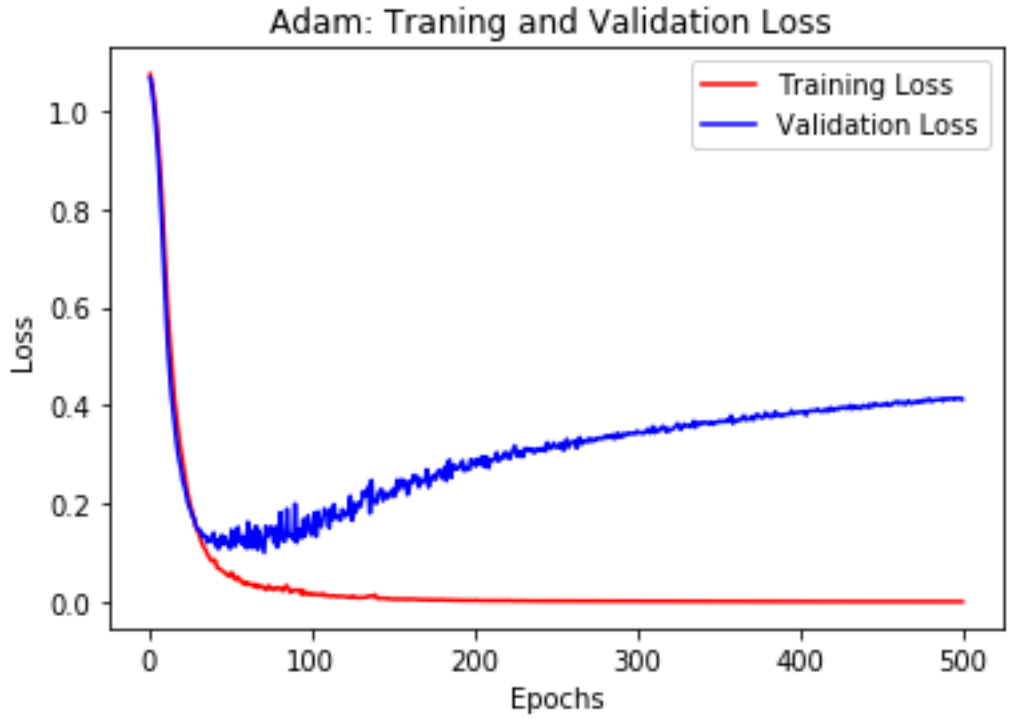
AdaDelta Optimizer :





- 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: 0.04625893011689186
    - Accuracy: 1.0

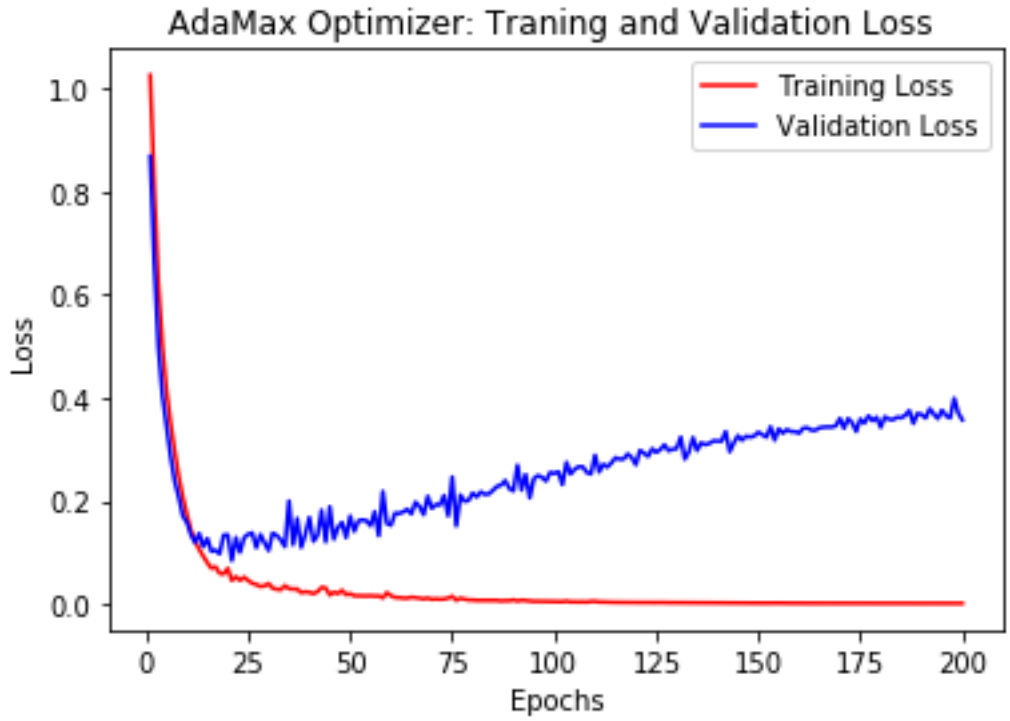
Adam Optimizer :

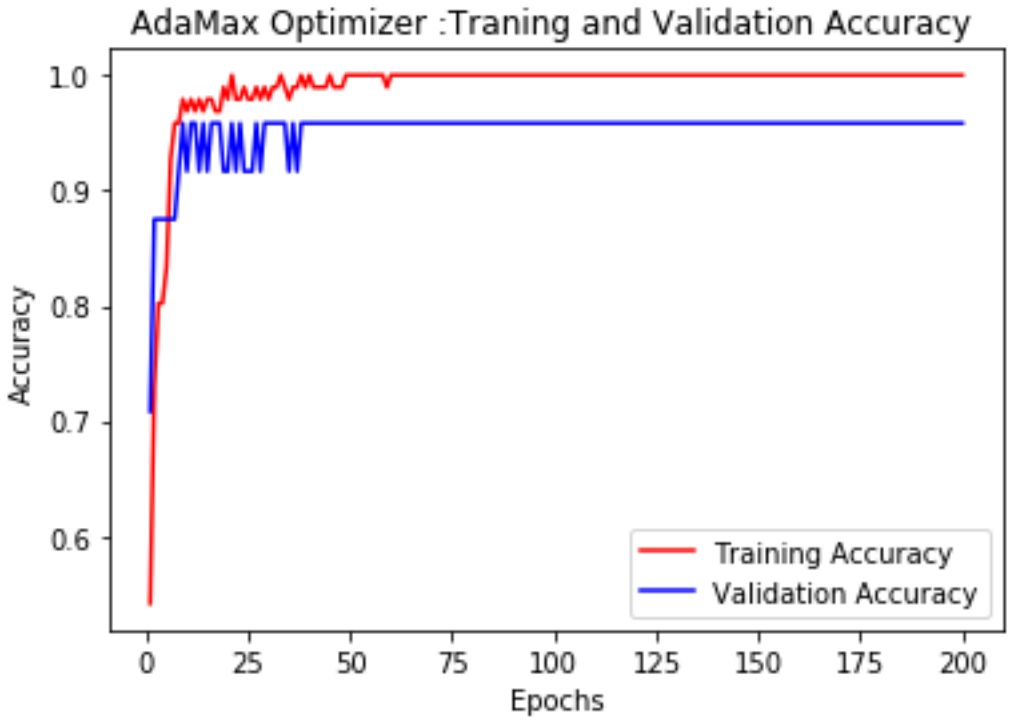


- From the above graph we see that our models validation loss starts to increase after 40 epochs and hence we choose our final number of epochs as 40
- Finally we evaluate our network on test data and got below results.

- Loss: 0.10385104268789291
- Accuracy: 0.9666666388511658

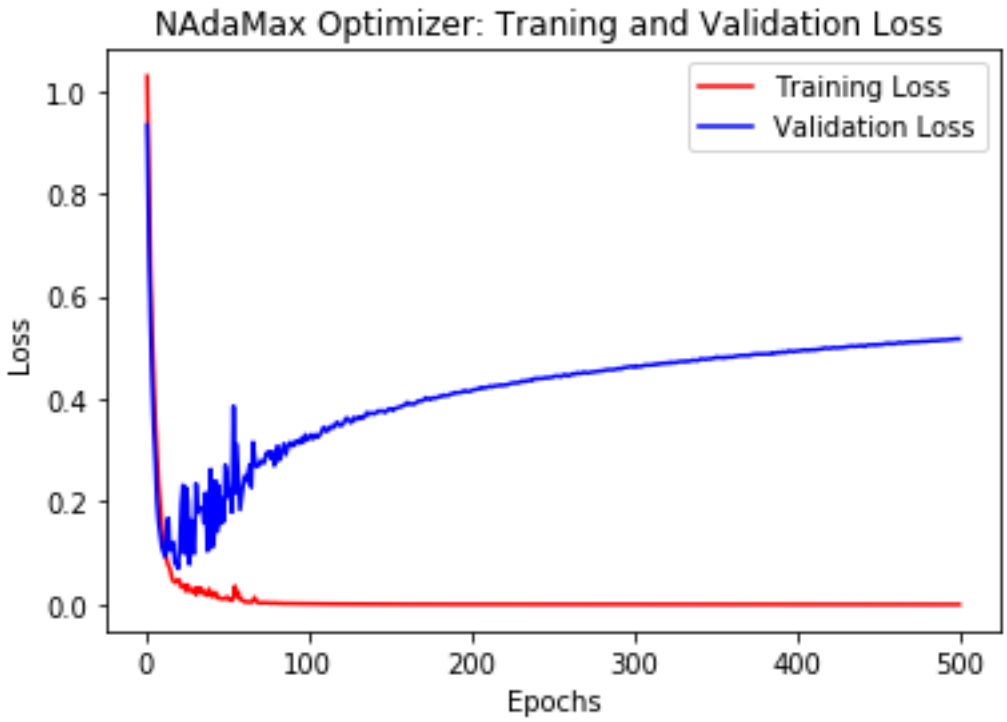
AdaMax Optimizer :

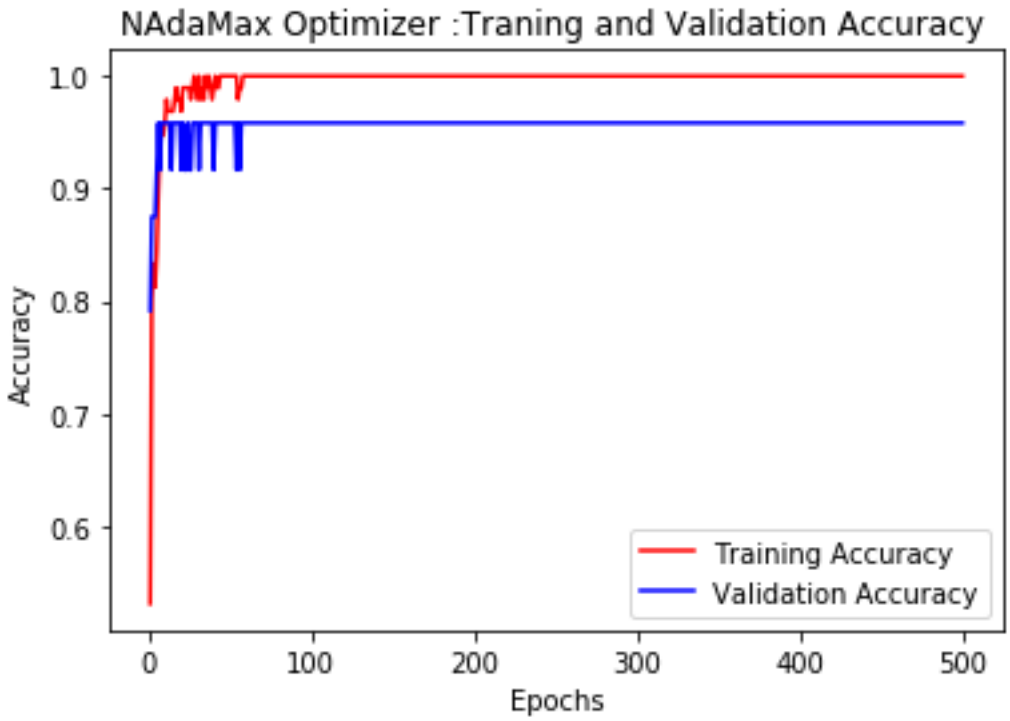




- From the above graph we see that our models validation loss starts to increase after 15 epochs and hence we choose our final number of epochs as 15
- Finally we evaluate our network on test data and got below results.
    - Loss: 0.15642783045768738
    - Accuracy: 0.9666666388511658

Nadamax Optimizer :





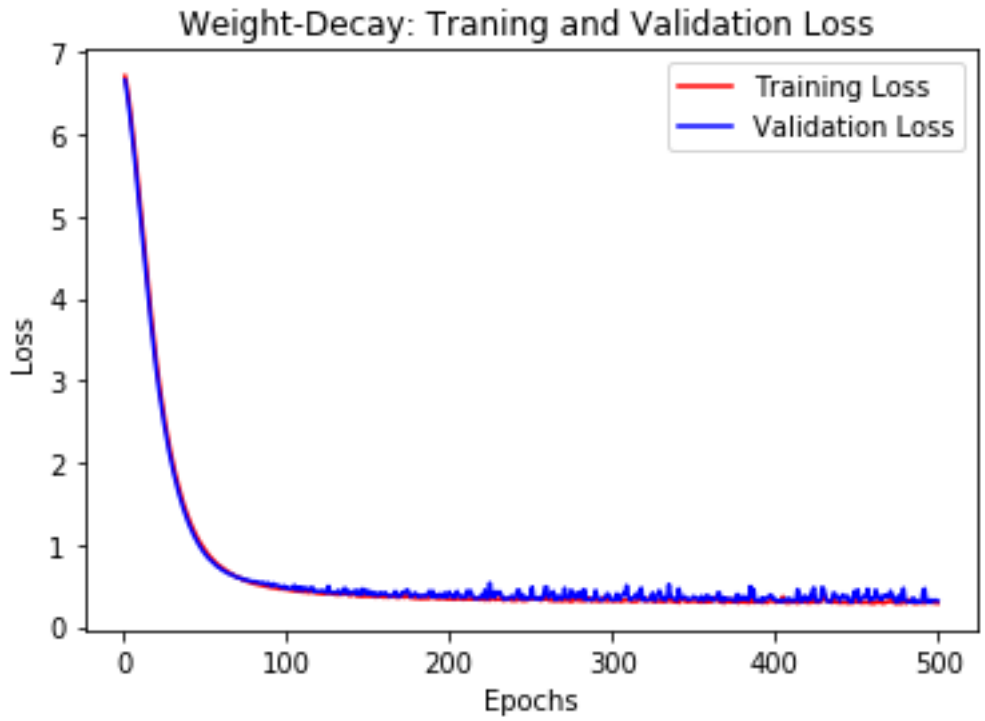
~~Results on~~

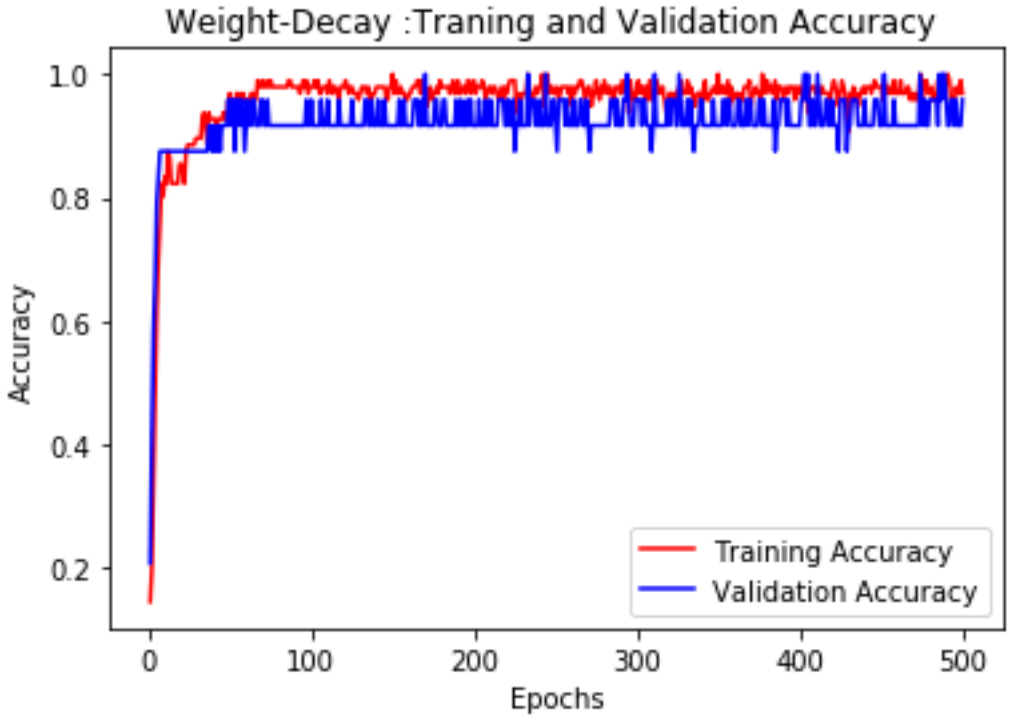
- From the above graph we see that our models validation loss starts to increase after 9 epochs and hence we choose our final number of epochs as 9
- Finally we evaluate our network on test data and got below results.
    - Loss: 0.1524132788181305
    - Accuracy: 0.9666666388511658

We compare all of the above Optimizers , we can make a statement that Adam Optimizers gives the best results (lowest loss).

Evaluating Different Regularization Measures:

Weight Decay on Adam Optimizer:

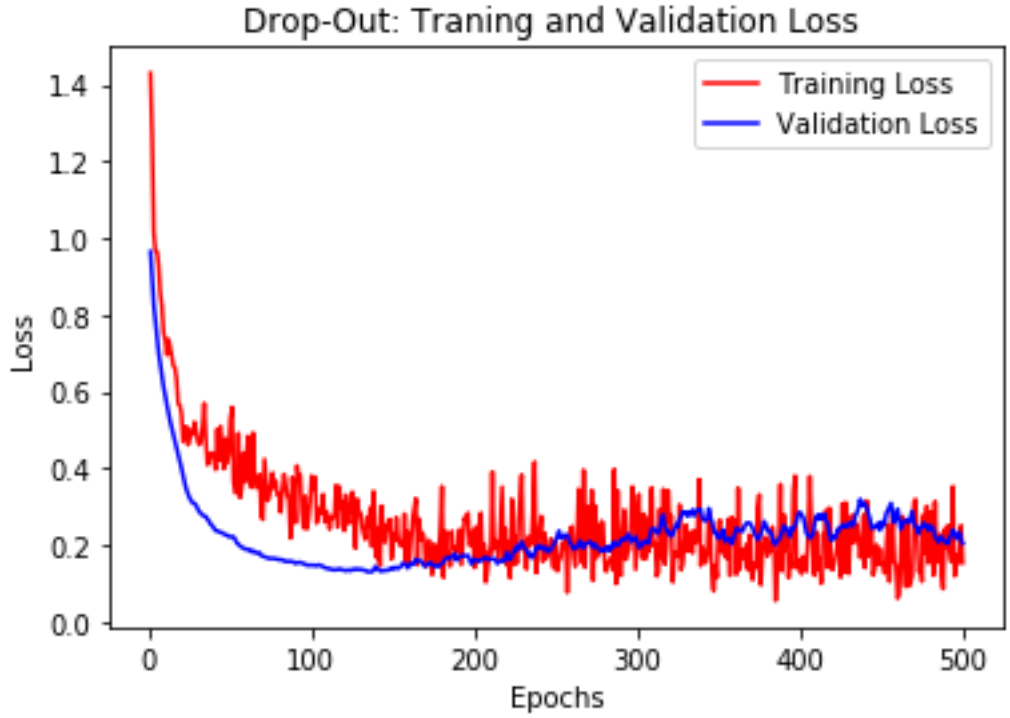


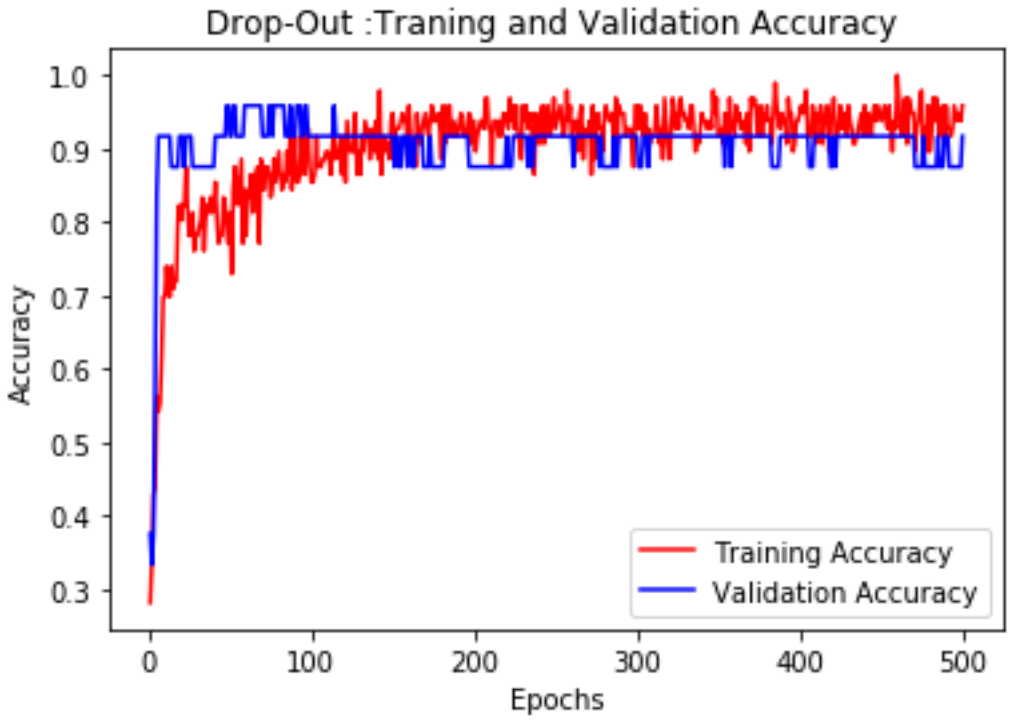


From the above graph we can see that weight decay regularization improves generalization on the train data and reduce overfitting. Evaluating the above network on the test data with epoch around 500.

- Test se performance:
  - Loss: 0.28592681884765625
  - Accuracy: 1.0

Dropout with RMS Prop :



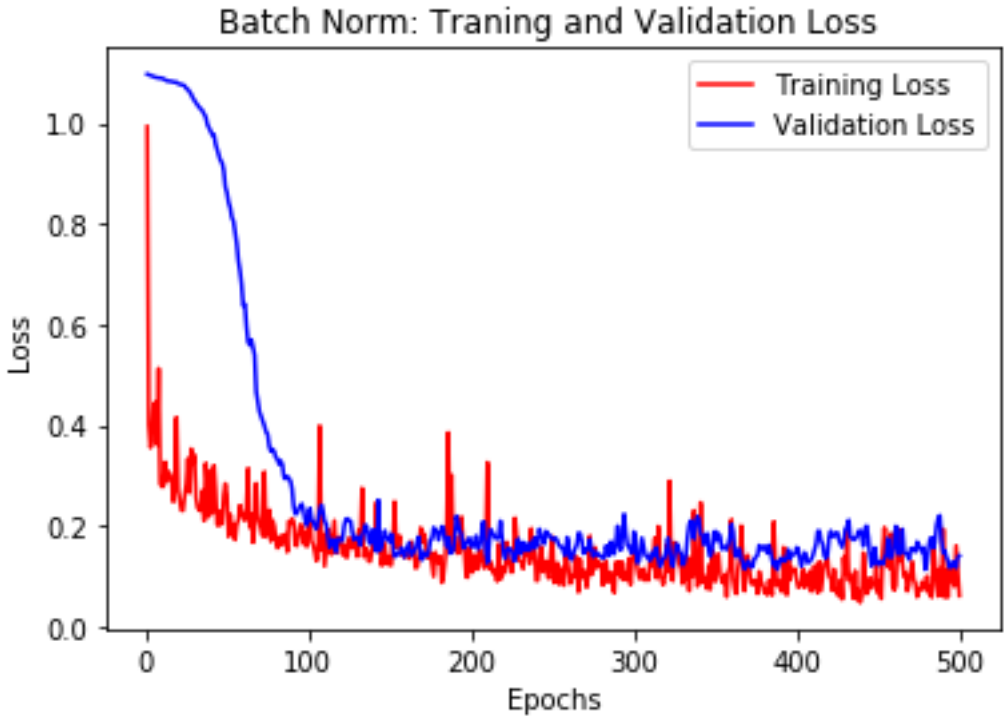


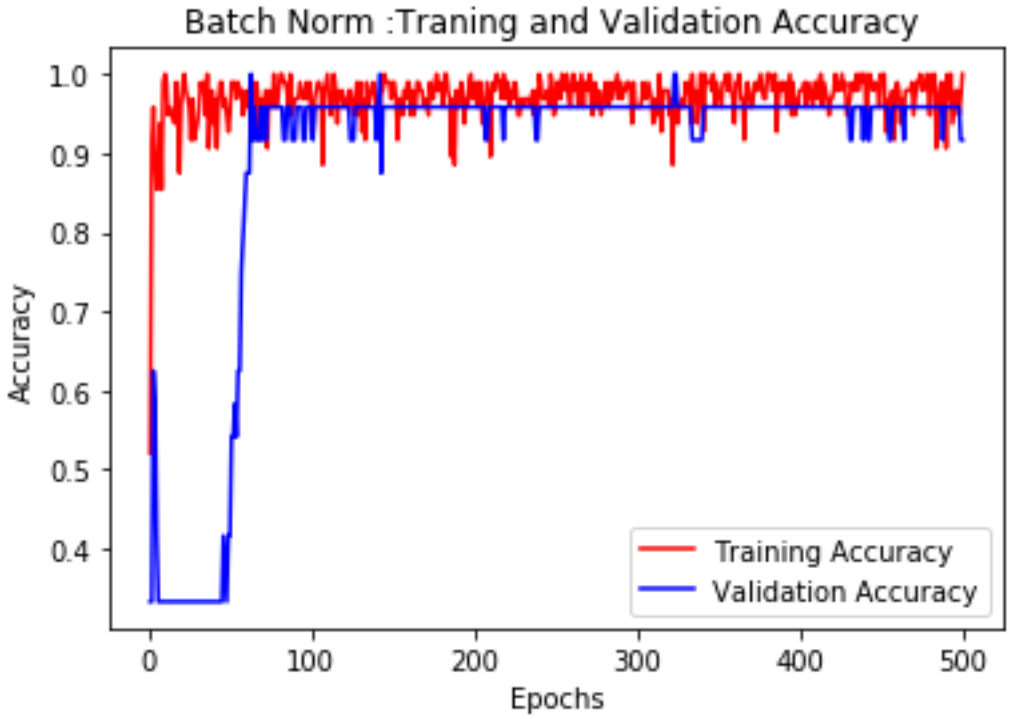
From the above graph we can see that Drop Out regularization improves generalization on the train data and reduce overfitting.

Evaluating the above network on the test data with epoch around 130.

- Test data performance:
  - Loss: 0.10729759186506271
  - Accuracy: 0.9666666388511658

Batch Normalization 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: 0.2646564245223999
  - Accuracy: 0.9666666388511658

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

- we use two Classifier namely : 1. Adam 2. RMSProp to predict values
- taking the average of the above two predicted values and comparing it to true label we calculate accuracy
- Accuracy of an Ensemble Classifier 0.9666666666666667