Maniya Yash Rajeshbhai
B20CS033

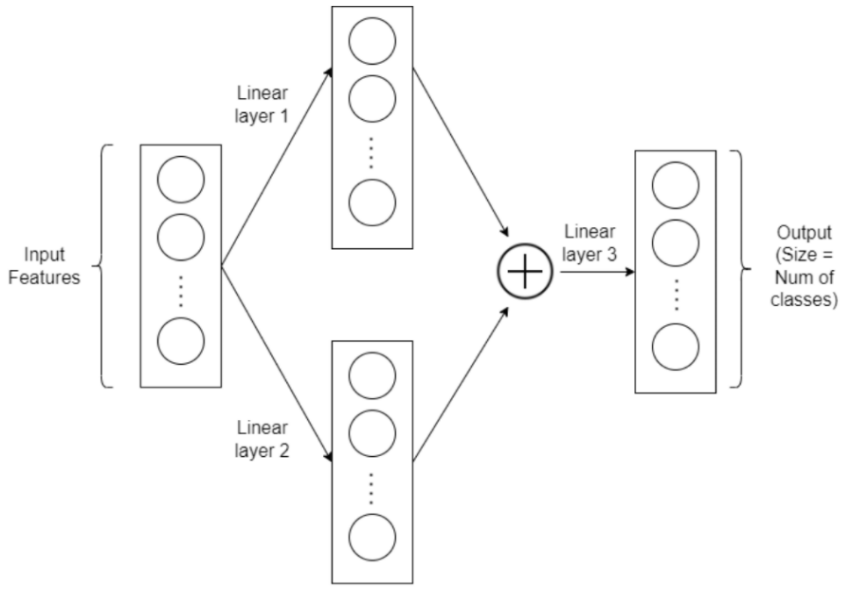
ML Lab 7 Report

Question 1 :

- **Preprocessing :**

  There were 8 features and class column given in the dataset. Did label encoding for sex feature and MinMaxScaled all the features.
  Also, I converted the Ring column into class wise data of 3 classes.
  0 class : 1-8
  1 class : 9-10
  2 class : 11-29

- **Creating a Neural Network :**



  Using pytorch, implemented a Neural Network with 1 input layer, 2 hidden layers and an output layer as mentioned in the question.
  Output is the sigmoid activation of the addition of the outputs of the two hidden layers set parallelly.

- **Training the Neural Network :**

  Now that the model was ready, wrote code for training it using Forward Propagation, Calculating Loss, Backward Propagation using Stochastic Gradient Descent and Updating the weights and biases.
  For this, inbuilt functions were used.

- **Hyperparameter Training :**

  The model had below mentioned parameters :

  1. batch_size
  2. num_epochs
  3. size_hidden
  4. learning_rate

  Initially I started with a large batch_size ~ 50 , num_epochs ~ 300, size_hidden ~ 50 and high learning_rate of 0.6.
  Then gradually started reducing the learning rate until the model gave a satisfactory increase in accuracy. (lr ~ 0.1)
  Also I tried reducing it furthermore along with increasing the num_epochs.
  Finally ended up with the following hyper-parameters :

  1. batch_size = 25
  2. num_epochs = 500
  3. size_hidden = 90
  4. learning_rate = 0.5 - 0.005 (dynamic)

  These hyperparameters gave an accuracy of **66.50 %** on the training dataset.

- **Testing the model :**

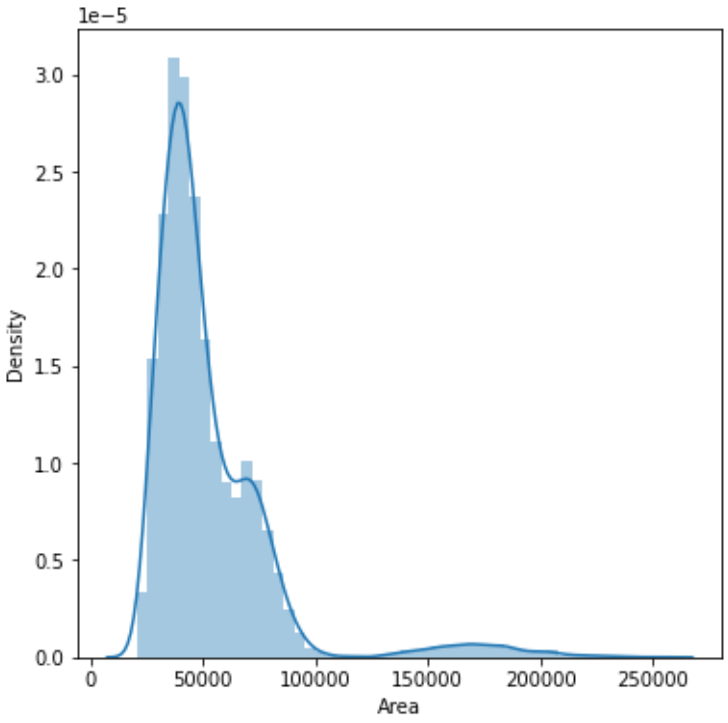
  Finally I tested it on the testing data and found the following accuracies :

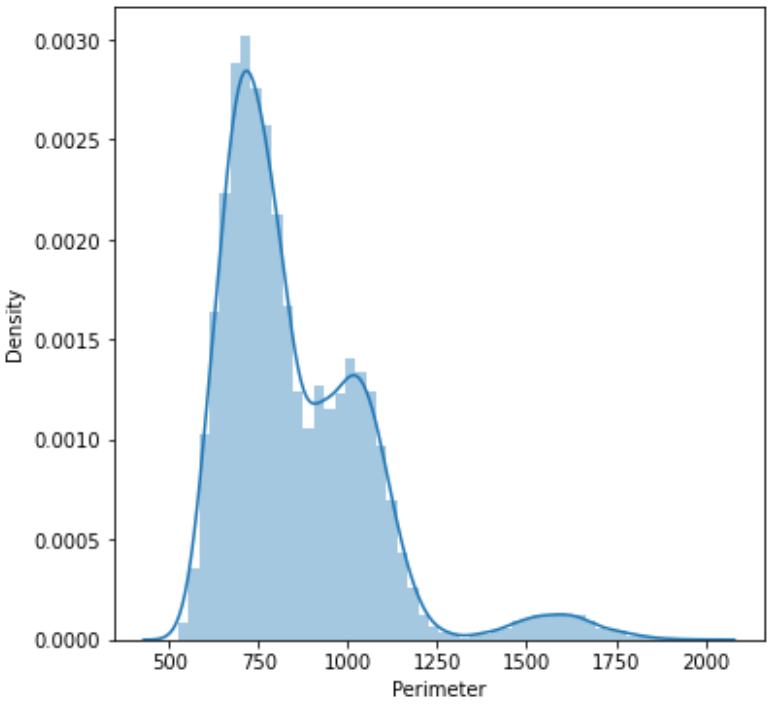
```
Accuracy on Training Dataset = 66.50 %
Accuracy on Testing Dataset = 63.64 %
```

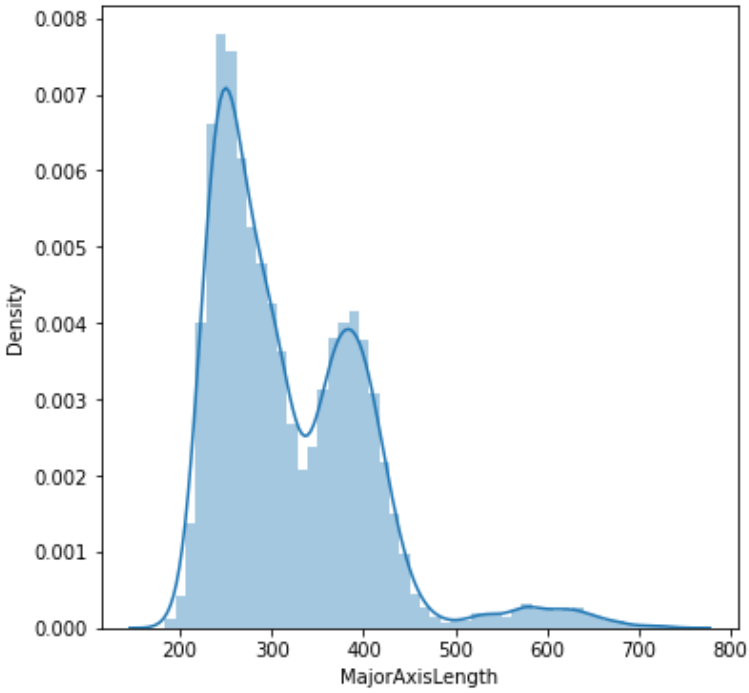
Question 2 :

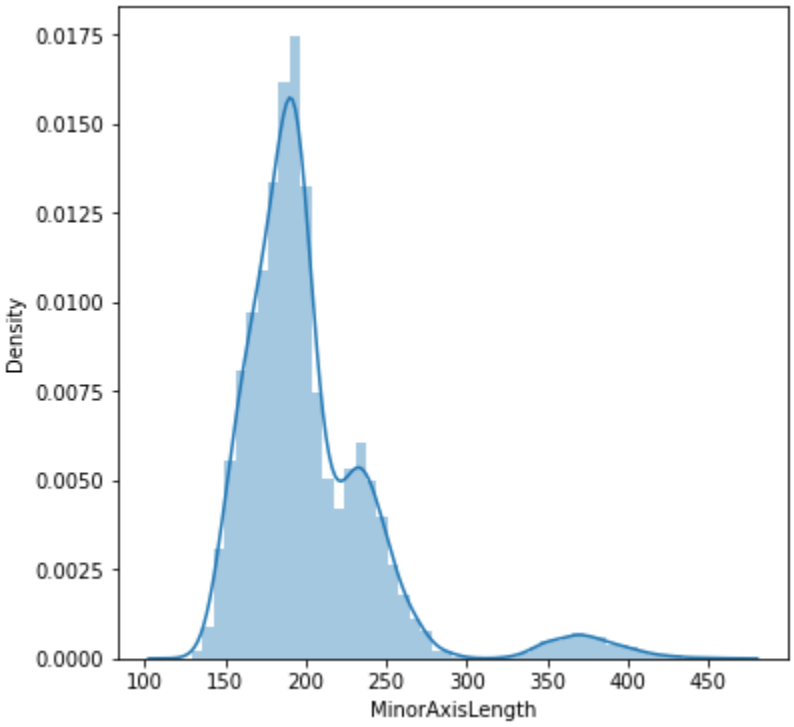
A. Preprocessing and Visualisation :

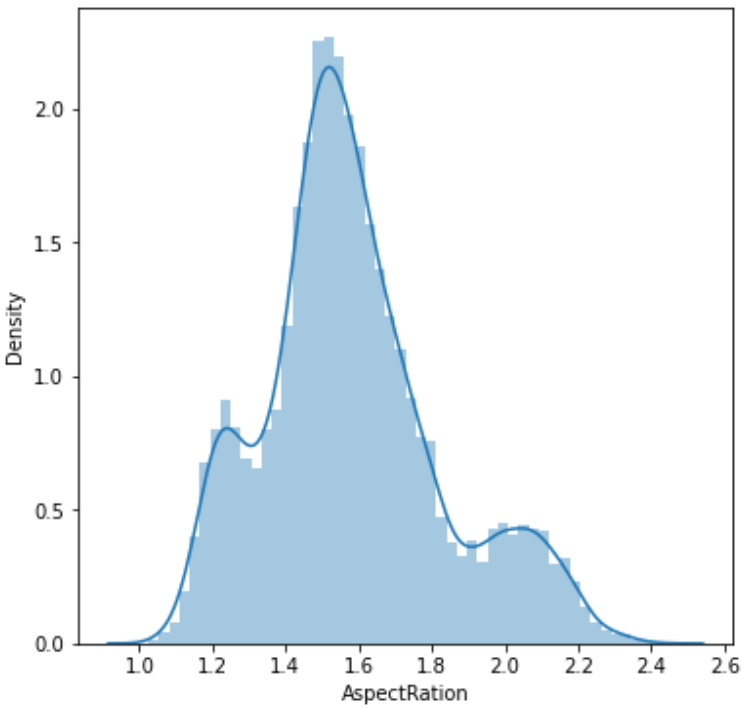
- Preprocessed the data and made distribution plots for all the features and class.

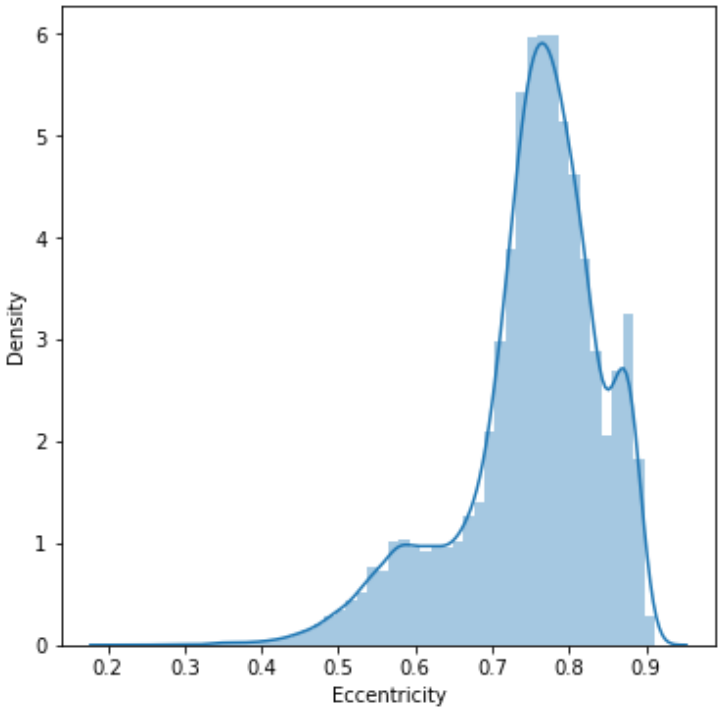


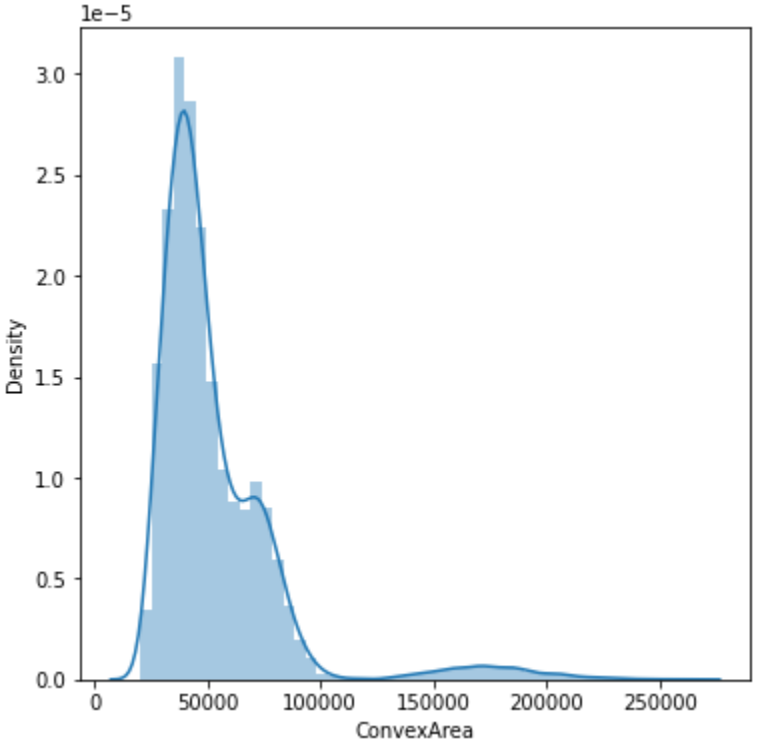


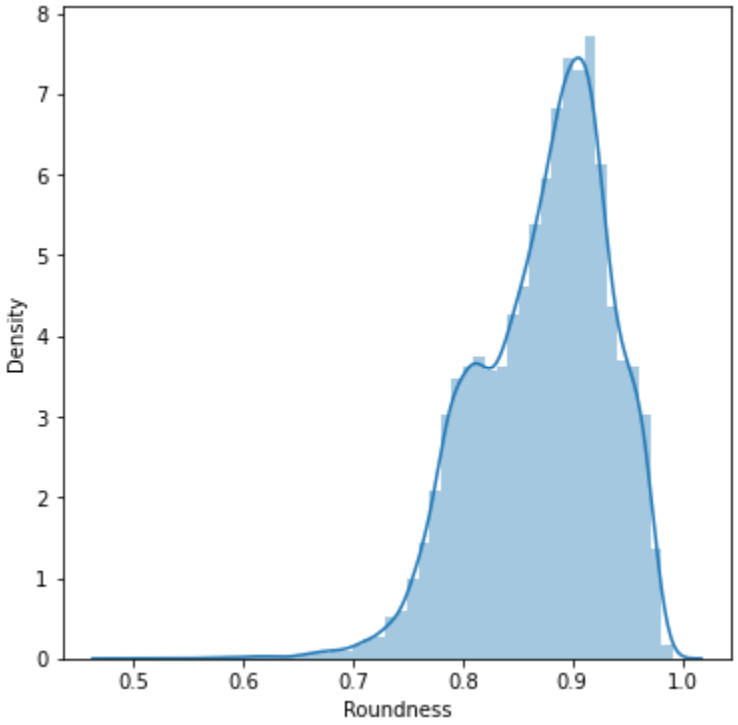


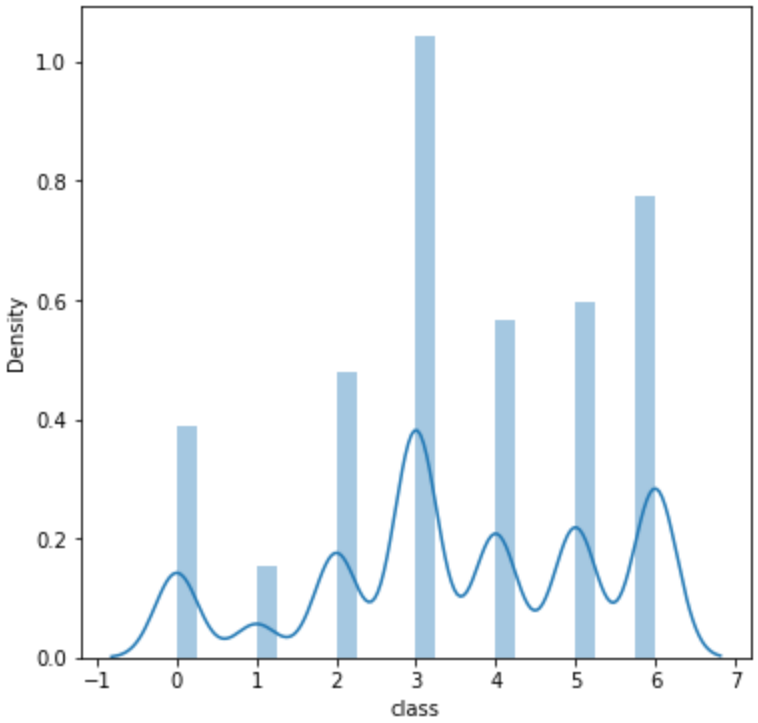



Density
AspectRation


Density
Eccentricity


1e−5
Density
ConvexArea

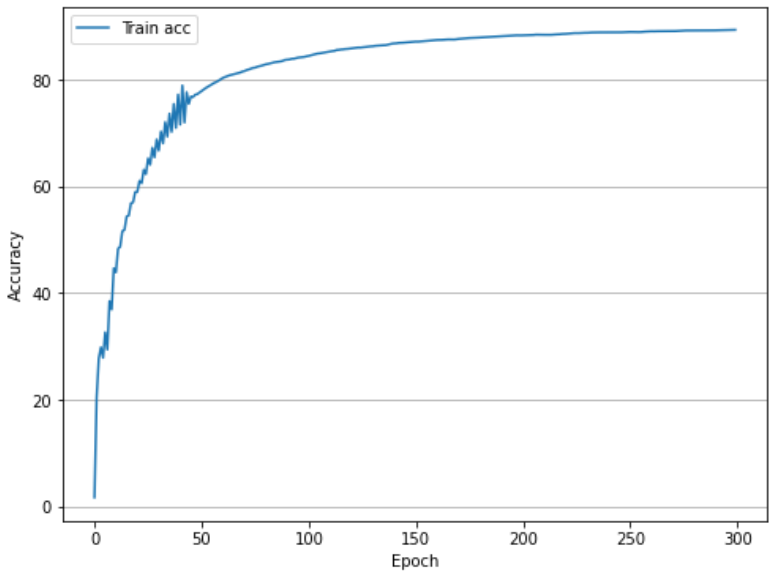

Density
Roundness



B. MLP Implementation from Scratch :

- Followed following steps for implementing a Multi-Layer Perceptron :
  - Wrote three activation functions - sigmoid, tanh, ReLU
  - Initialised the weights and biases.
  - Forward propagated the input.
  - Backward propagated the error.
  - Trained the network using stochastic gradient descent.
  - Predicted the output for the given validation sample.

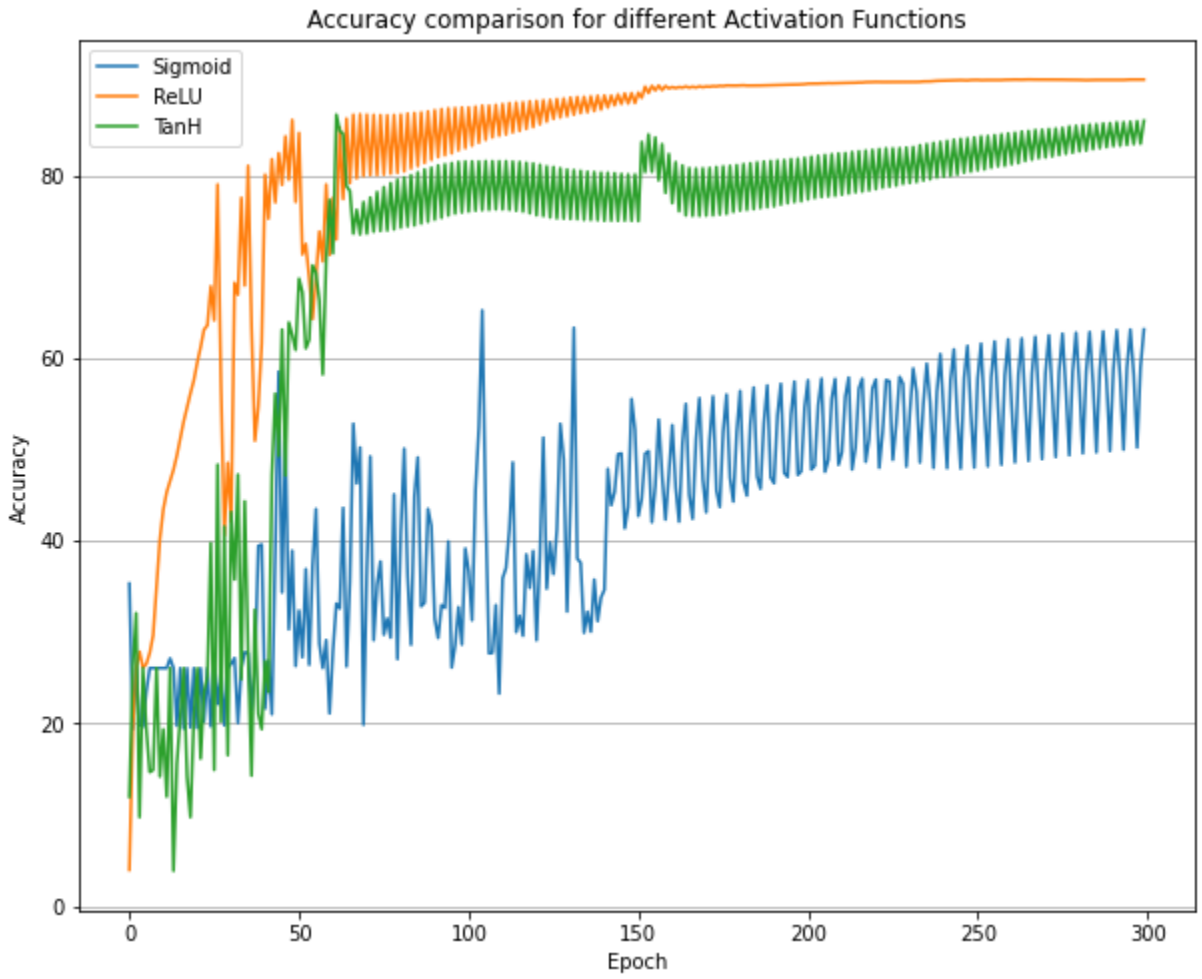
Following were the results while training the MLP on the training set.



C. Experimenting various Activation Functions :

Used the same MLP class but with different Activation Functions to see the changes in Accuracies with the used activation functions.

We see that ReLU performs best among all the three.

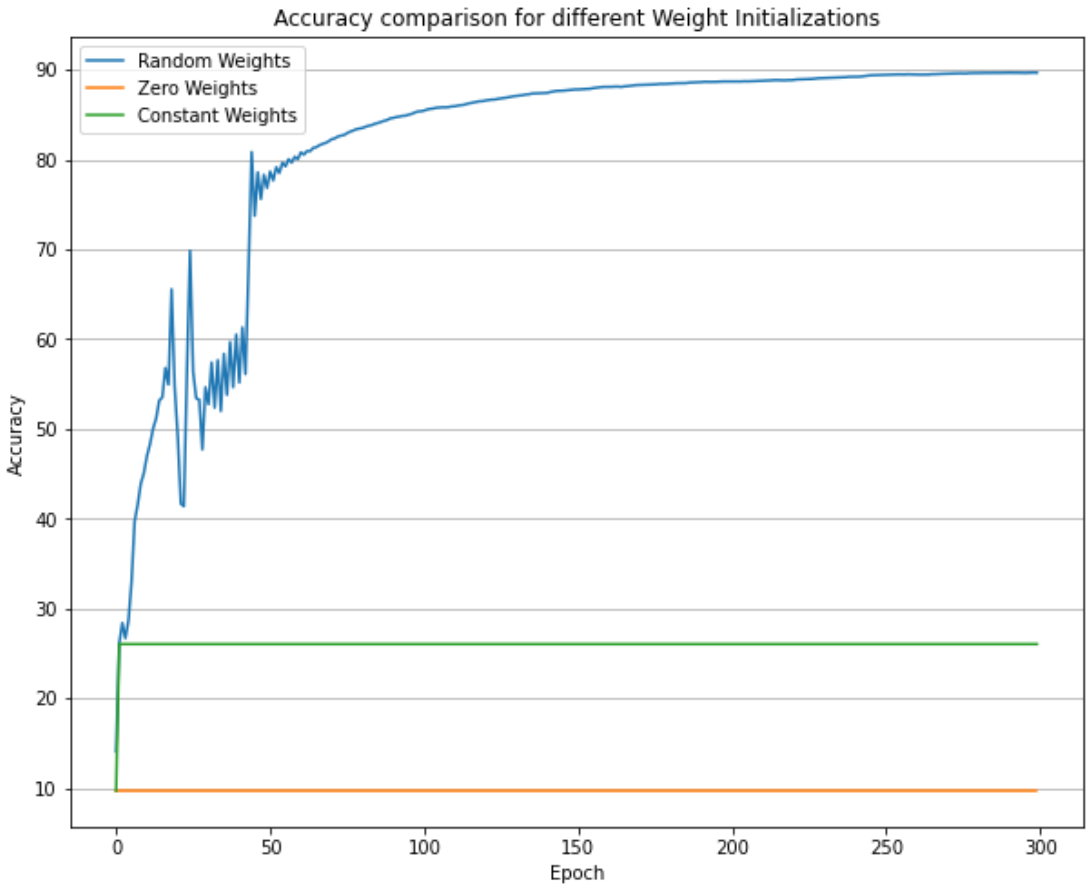


D. Experimenting various Weight Initialisations :

Used different weight initialization methods to check the variations in the validation accuracies.

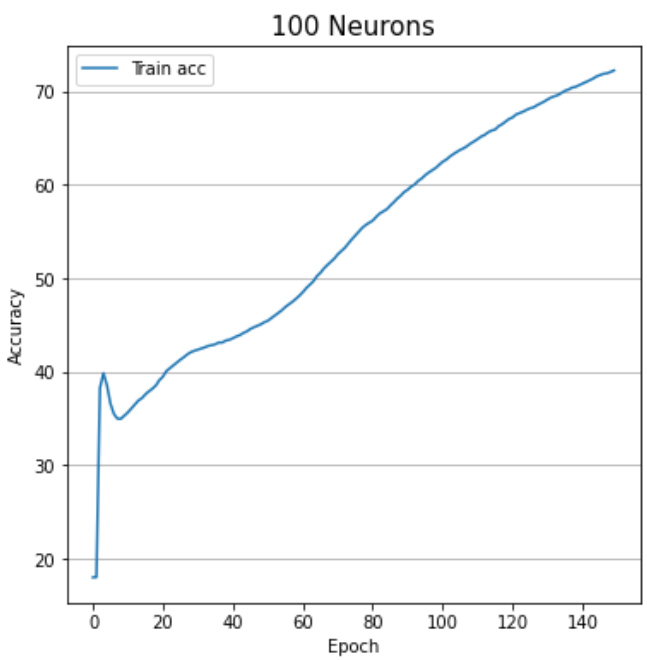
We see that the randomly generated weights and biases perform better than the other two methods.

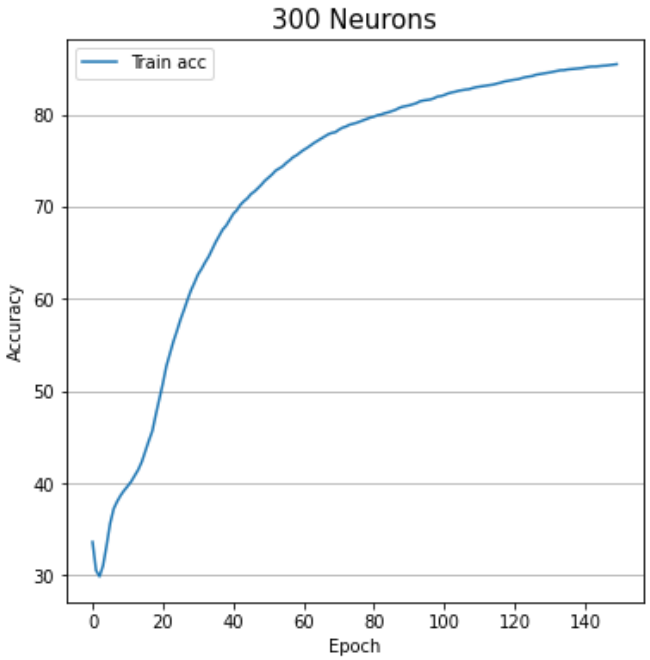
```
Accuracy on Validation Data : 89.42 %   [ Random Weights ]
Accuracy on Validation Data : 9.70 %    [ Zero Weights ]
Accuracy on Validation Data : 26.05 %   [ Constant Weights ]
```

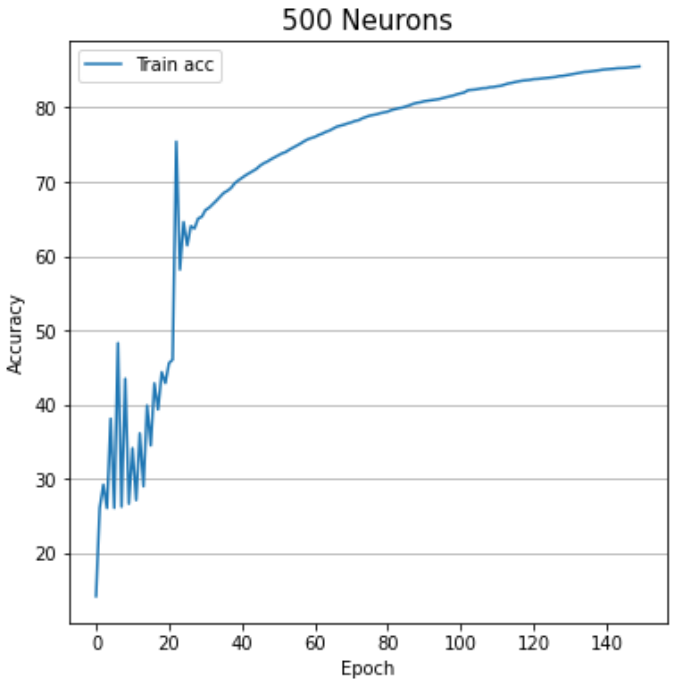


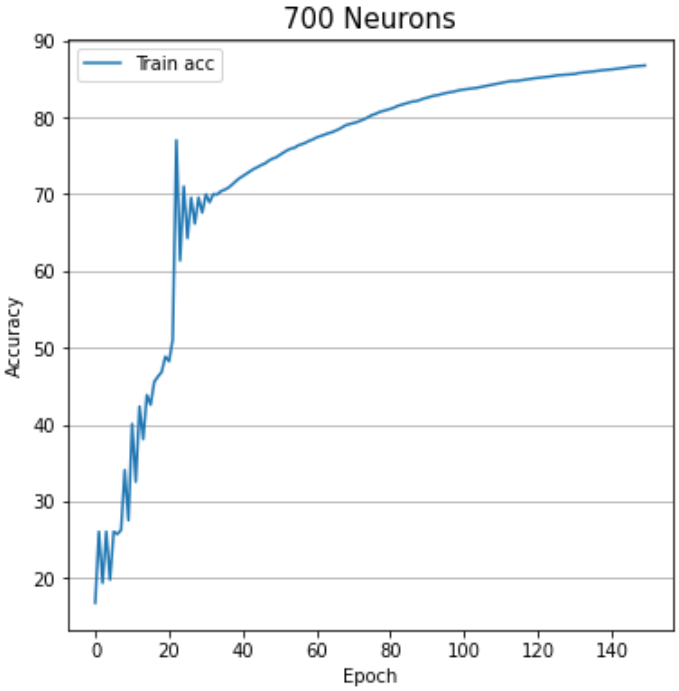
E. Changing the Neurons in the hidden layers :

Used different number of neurons in the hidden layers to check the variations in the validation accuracies.

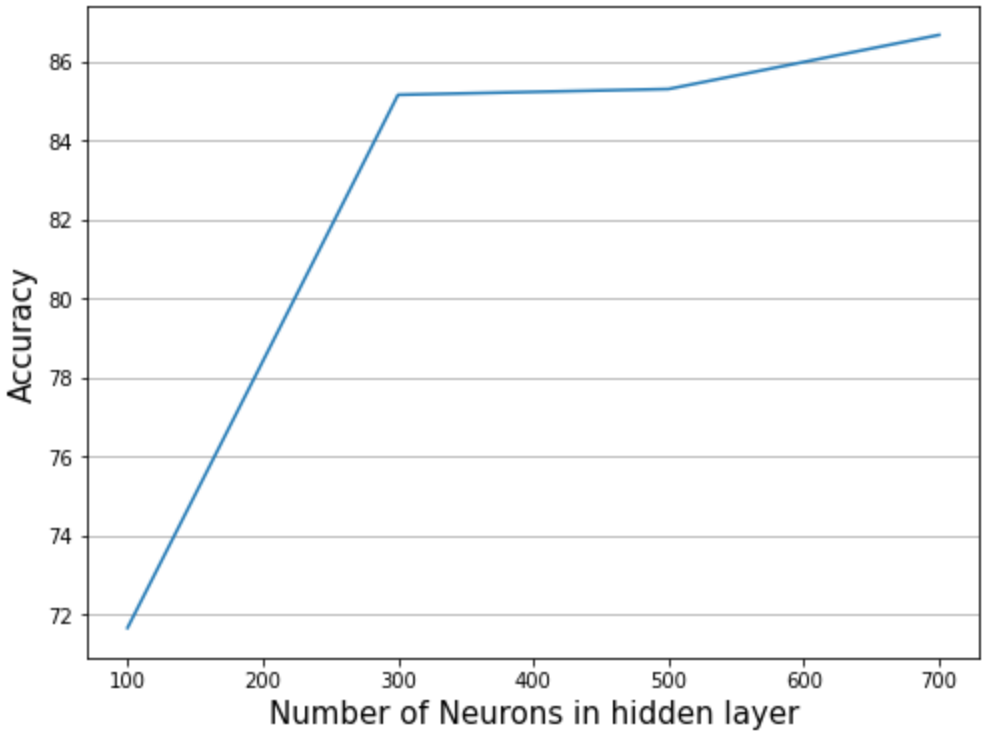








It is pretty sure that with increase in the number of neurons in the layer, the accuracy increases. However the training time does increase too.



F. Making the model adaptable to update Weights :

The model was already built in a way that we can update the weights manually.

```
Weights of hidden layer 2
[[-0.18536647 -0.33167907  0.46944935 -0.24150343  0.05579619 -0.28885031
   0.2001942 ]
 [-0.21588364  0.25559813  0.18838739 -0.08841169 -0.13804327 -0.28995321
  -0.49821909]
 [ 0.05964031  0.26295755 -0.30239944  0.37116612  0.11241491  0.1527992
   0.21269083]
 [ 0.09167242 -0.04358007 -0.099779   -0.22670499 -0.13342277  0.42952798
  -0.38257738]
 [ 0.22244195 -0.3535402   0.44665309 -0.47134674 -0.22559781 -0.32067536
   0.20399576]
 [ 0.45319987 -0.26265419  0.19184158 -0.08419543 -0.31407421  0.3912685
  -0.15660982]
 [ 0.32485873  0.39779447  0.0968952  -0.31340244  0.45197661  0.09527822
   0.28630941]
 [ 0.44440301  0.07443723 -0.04408363 -0.38160673 -0.00124207 -0.04024009
  -0.42766254]
 [-0.05977325 -0.09352857 -0.02695963  0.20781907 -0.400415    0.22395851
  -0.16864685]
 [-0.14434423 -0.25206595  0.00153924  0.15905723  0.21601971  0.02880958
   0.30496988]]

Biases of hidden layer 2
[[0. 0. 0. 0. 0. 0. 0.]]
```

End of the Report !