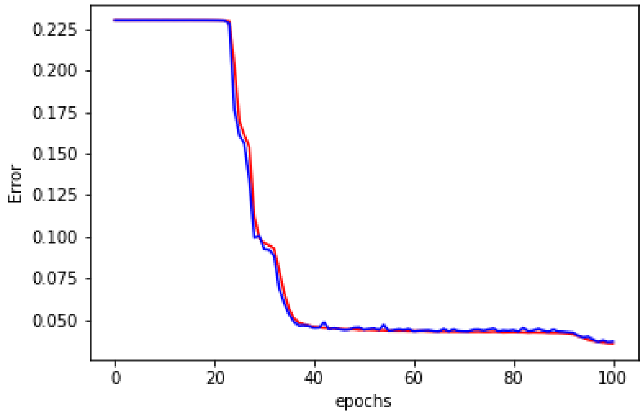
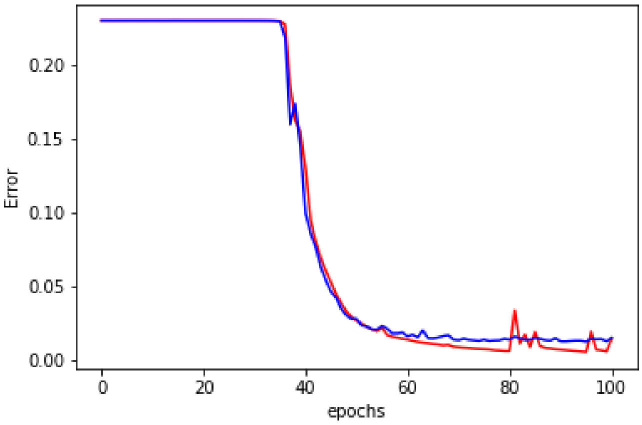
Q2) red→*Training Error, blue→Test Error*

Linear: 89.56

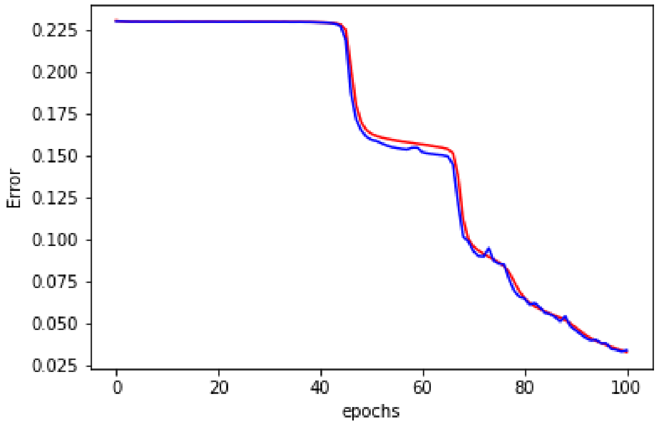
ReLU: 95.57

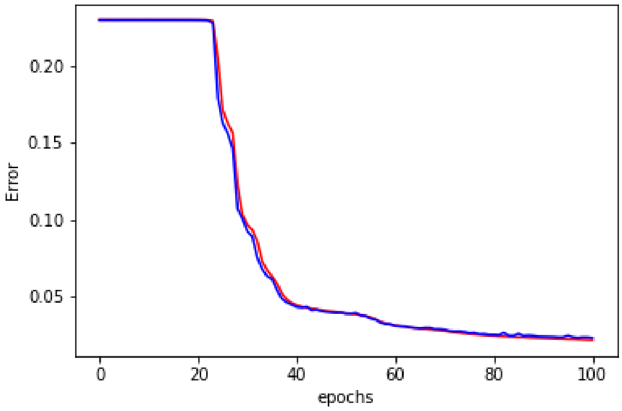




Sigmoid: 89.57

TanH: 93.24





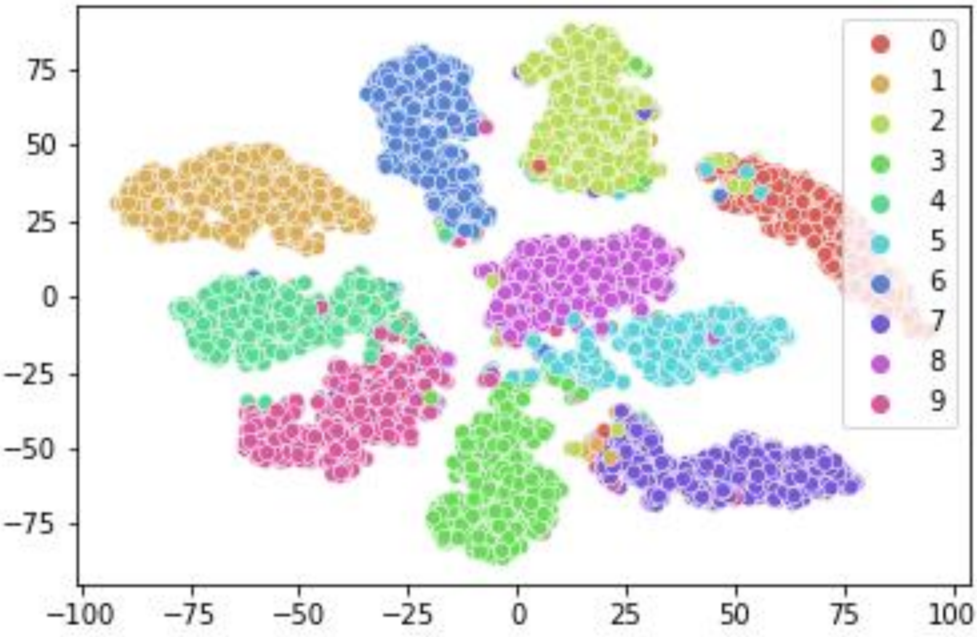
2018186
Saksham Dhull

2.1) Link to the drive folder: -
https://drive.google.com/drive/folders/1nwKZBCeif3us9u2iR-BH-LLfByGVYtoW?usp=sharing

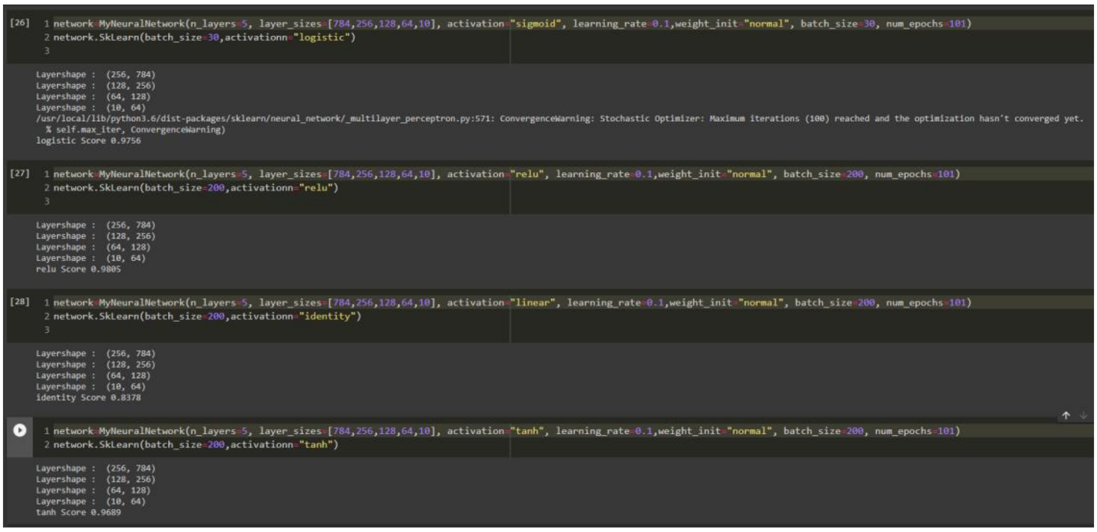
2.3) Softmax function is used at the last layer. It is used to get normalized probability distribution of the scores. The softmax transforms the values obtained in range of 0 to 1, so that they can be interpreted as probabilities. These probabilities help us label our obtained values in (0-9).

2.4) Hidden layers=3, Total layers=5

2.5) The model with highest test accuracy is with Relu activation function.

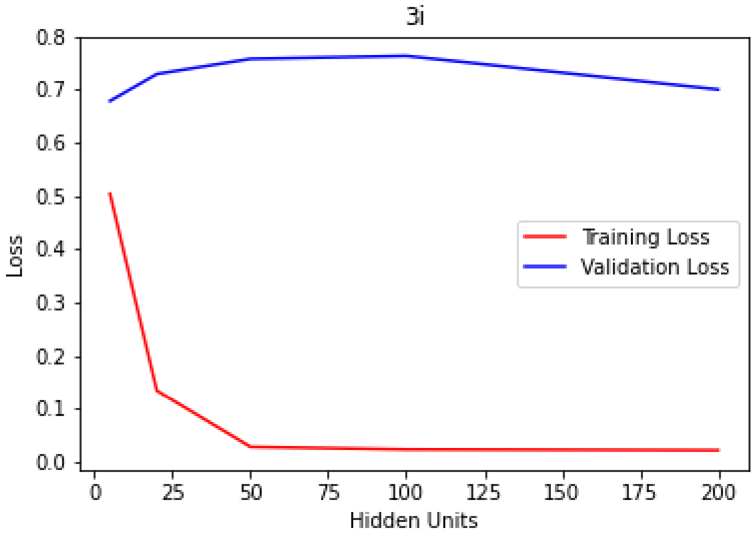


2.6) SkLearn produces almost similar accuracies as of our model

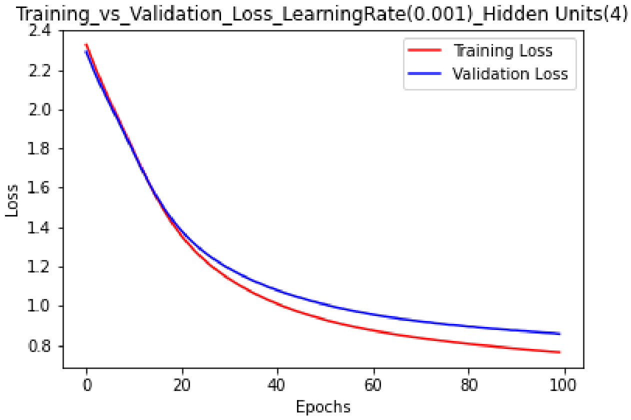


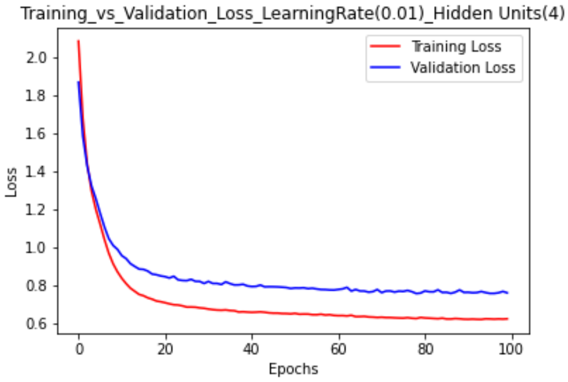
```
[26] network=MyNeuralNetwork(n_layers=5, layer_sizes=[784,256,128,64,10], activation="sigmoid", learning_rate=0.1,weight_init="normal", batch_size=30, num_epochs=101)
     network.SkLearn(batch_size=30,activationn="logistic")

Layershape :  (256, 784)
Layershape :  (128, 256)
Layershape :  (64, 128)
Layershape :  (10, 64)
/usr/local/lib/python3.6/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:571: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (100) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
logistic Score 0.9756

[27] network=MyNeuralNetwork(n_layers=5, layer_sizes=[784,256,128,64,10], activation="relu", learning_rate=0.1,weight_init="normal", batch_size=200, num_epochs=101)
     network.SkLearn(batch_size=200,activationn="relu")

Layershape :  (256, 784)
Layershape :  (128, 256)
Layershape :  (64, 128)
Layershape :  (10, 64)
relu Score 0.9805

[28] network=MyNeuralNetwork(n_layers=5, layer_sizes=[784,256,128,64,10], activation="linear", learning_rate=0.1,weight_init="normal", batch_size=200, num_epochs=101)
     network.SkLearn(batch_size=200,activationn="identity")

Layershape :  (256, 784)
Layershape :  (128, 256)
Layershape :  (64, 128)
Layershape :  (10, 64)
identity Score 0.8378

     network=MyNeuralNetwork(n_layers=5, layer_sizes=[784,256,128,64,10], activation="tanh", learning_rate=0.1,weight_init="normal", batch_size=200, num_epochs=101)
     network.SkLearn(batch_size=200,activationn="tanh")

Layershape :  (256, 784)
Layershape :  (128, 256)
Layershape :  (64, 128)
Layershape :  (10, 64)
tanh Score 0.9689
```

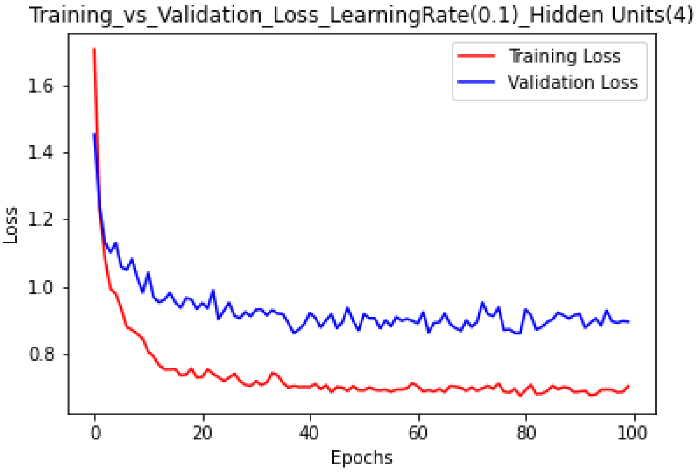
3i) On increasing the number of hidden units,the training loss is decreasing and saturating after a point, but the validation loss remains high always, which means that our model is overfitting on the data.



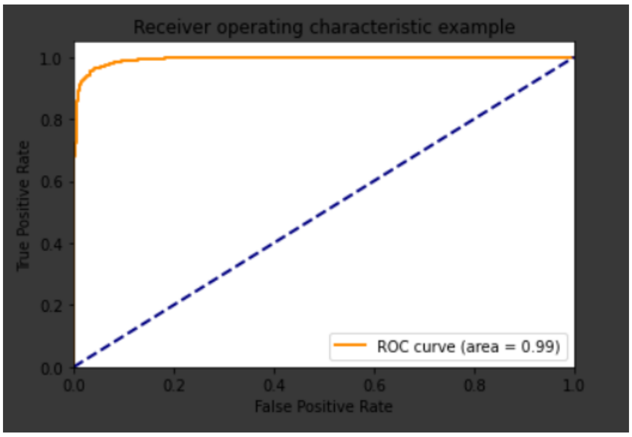
3ii) On increasing the learning rate here the models do not exactly converge, like when the lr=0.1 the step size is too big so it gives irregularities. Lr=0.001 does not converge in 100 epochs, lr=0.01 converges properly.

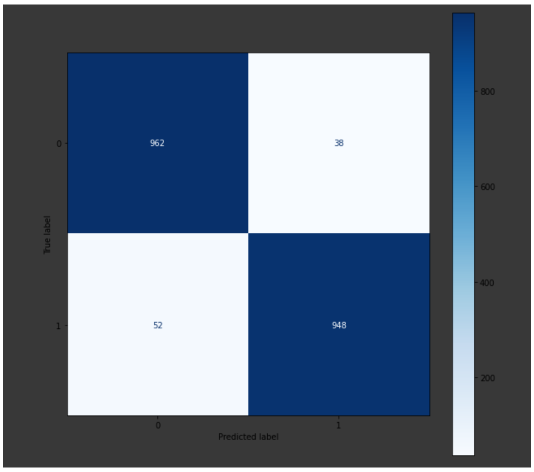


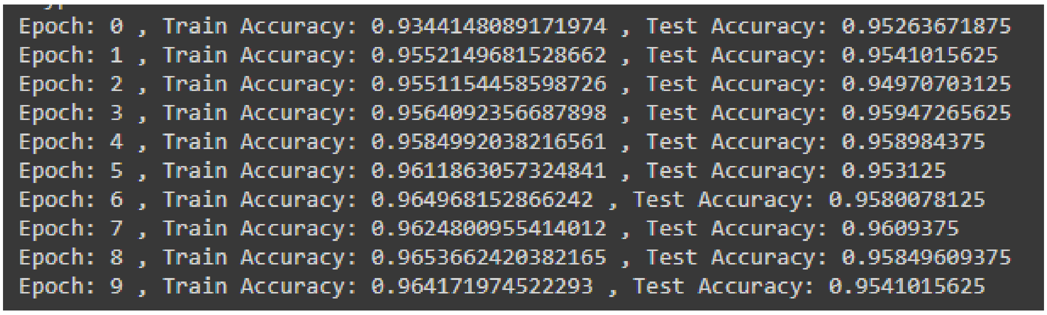




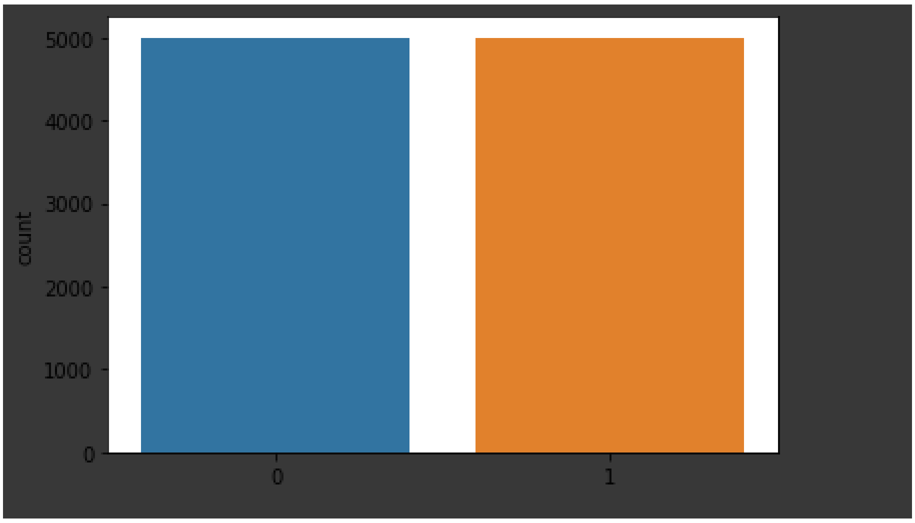
4)







```
Epoch: 0 , Train Accuracy: 0.9344148089171974 , Test Accuracy: 0.95263671875
Epoch: 1 , Train Accuracy: 0.9552149681528662 , Test Accuracy: 0.9541015625
Epoch: 2 , Train Accuracy: 0.9551154458598726 , Test Accuracy: 0.94970703125
Epoch: 3 , Train Accuracy: 0.9564092356687898 , Test Accuracy: 0.95947265625
Epoch: 4 , Train Accuracy: 0.9584992038216561 , Test Accuracy: 0.958984375
Epoch: 5 , Train Accuracy: 0.9611863057324841 , Test Accuracy: 0.953125
Epoch: 6 , Train Accuracy: 0.964968152866242 , Test Accuracy: 0.9580078125
Epoch: 7 , Train Accuracy: 0.9624800955414012 , Test Accuracy: 0.9609375
Epoch: 8 , Train Accuracy: 0.9653662420382165 , Test Accuracy: 0.95849609375
Epoch: 9 , Train Accuracy: 0.964171974522293 , Test Accuracy: 0.9541015625
```


count