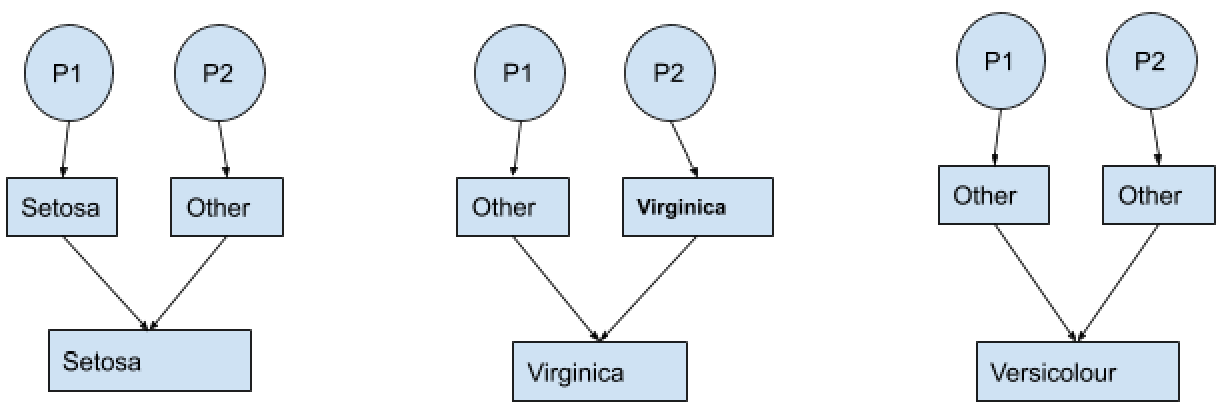
Report on Assignment - 1

CSL7050 Machine Learning 2
Srigowri M V
M20CS016
(MTech I Year,CSE)

Q1. Implement two perceptron models to perform classification on iris databases from scratch

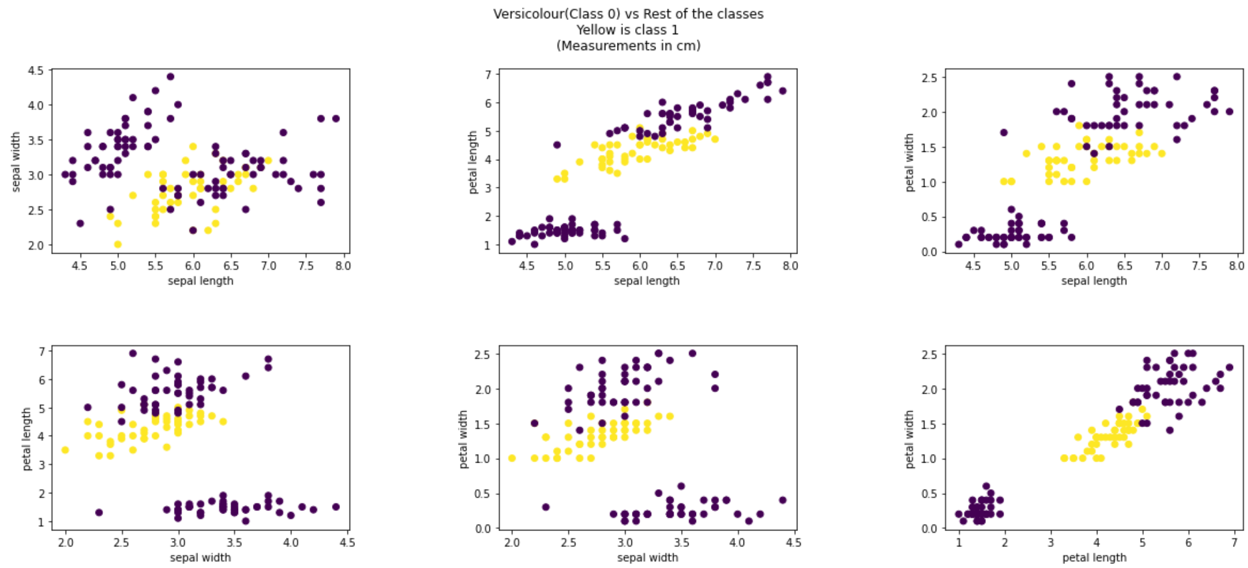
Dataset: IRIS DATA SET
Features: Sepal length in cm,Sepal width in cm, Petal length in cm,Petal width in cm
Classes: Iris Setosa, Iris Versicolour, Iris Virginica
Reference: https://archive.ics.uci.edu/ml/datasets/iris
Task: 3 class Classification using Perceptron training
Procedure:
Two perceptron models are used. Each perceptron predicts two classes as follows
The first model classifies as Class 0 (Setosa) vs Other
The second model classifies as Class 2 (Virginica) vs Other
P1,P2 are the perceptrons the 2nd level is their prediction and the 3rd level is the final prediction



Reasoning:
Diagram1 : If P1 predicts as Setosa and P2 predicts as not Virginica. It has to be Setosa (It can't be Versicolour otherwise P1 would have selected it as Others if that was the case)
Diagram2: If P1 predicts as others and P2 predicts as Virginica. It has to be Virginica (It can't be Versicolour otherwise P2 would have selected it as Others if that was the case)
Diagram3: If P1 predicts as others and P2 predicts as Others . It has to be Versicolour

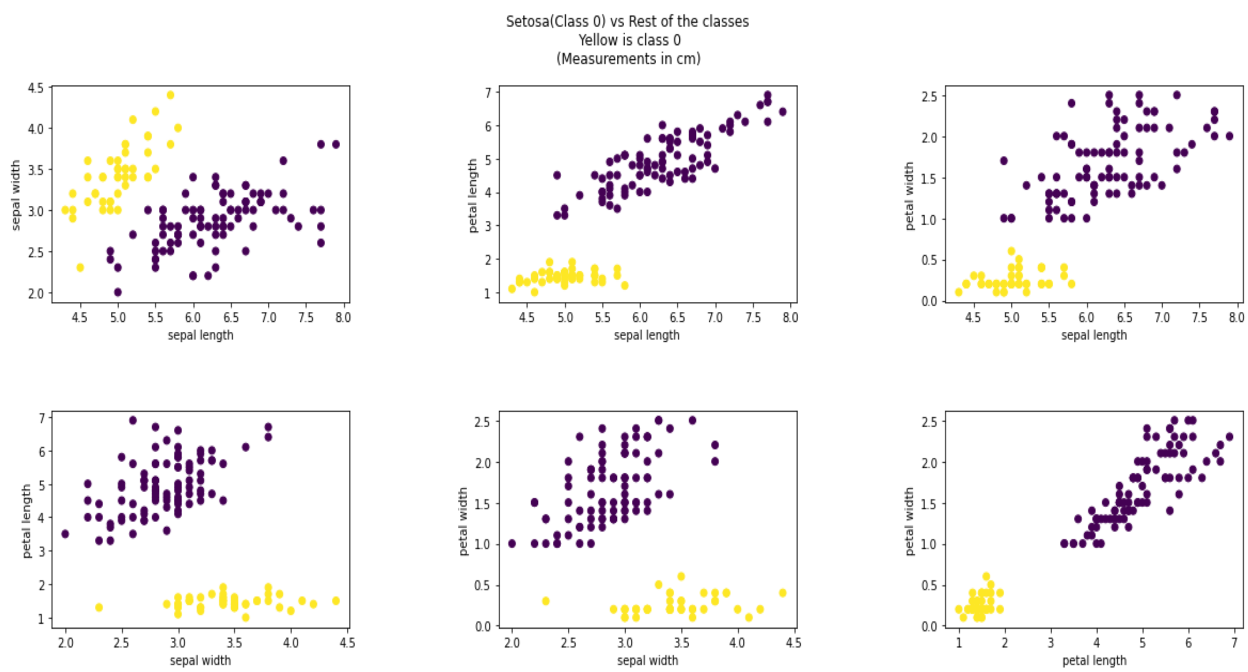
Why any Perceptron was <u>not selected</u> to classify Class 1(Versicolour) V/s Others

From the figure below, the class Versicolour (Yellow color) is not linearly separable from the rest of the class and it is not viable for a Perceptron training



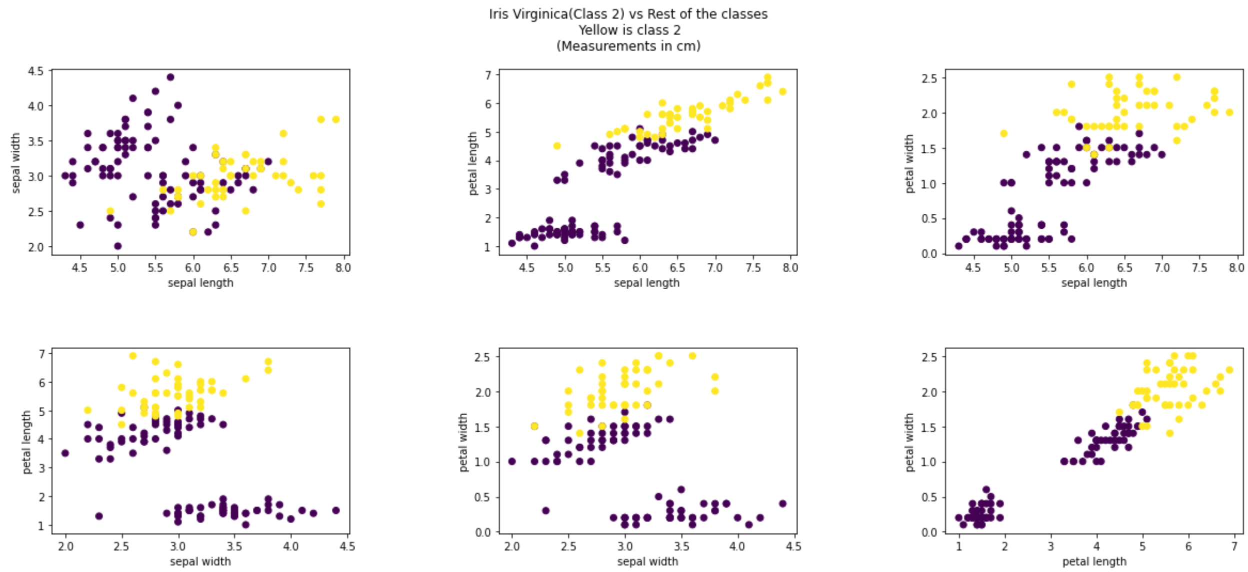
Why Perceptron 1 was chose to classify Class 0 (Setosa) V/s Others

From the figure below, it can be seen that Setosa is separable from the other two class with respect to any of the 4 features [Setosa is in Yellow color]



Why Perceptron 2 was chosen to classify Class 2 (Virginica) V/s Others

From the figure below, it can be seen that Virginica is almost separable from the other two class with respect to any of the 4 features [Virginica is in yellow color]



Training:

The data set has 150 feature vectors, 1-50 for class 0, 50-100 for class 1 and 100-150 class 2

The feature vectors from 1-45,55-95,100-145, a total of 135 are used for training and the rest is used for testing

Task 1:

(a) vary the learning rate and show the best learning rate value when you run it for 50 epochs.

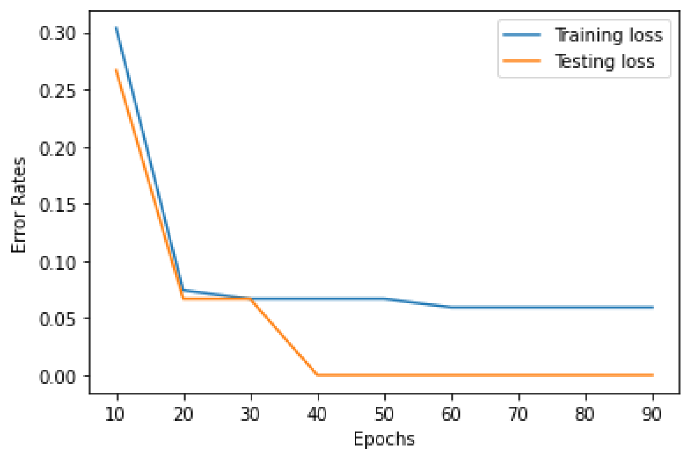
Learning Rate	Misclassification Rate
0.0001	42/135
0.001	9/135
0.01	15/135
0.1	44/134
1	19/135

Task 2:

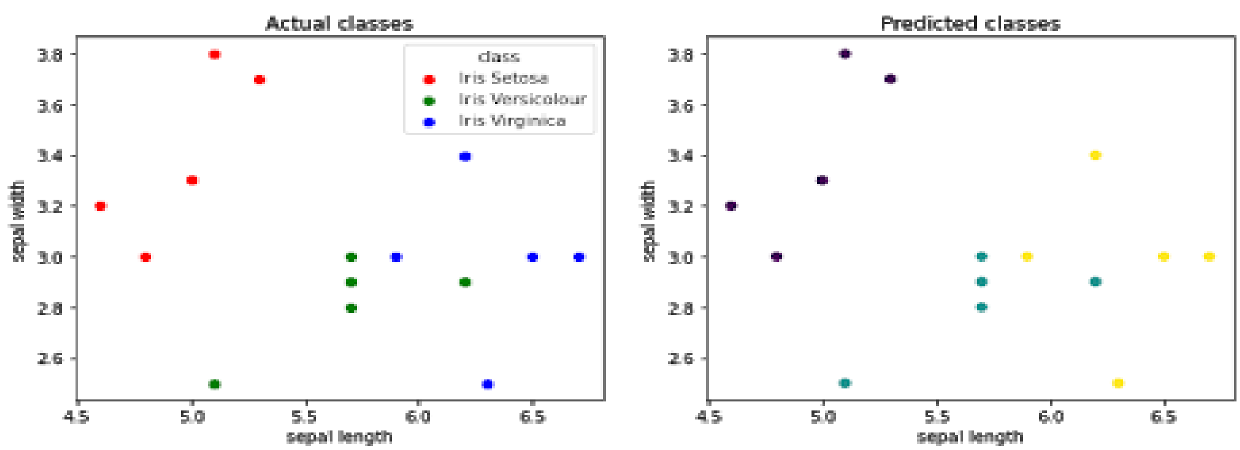
vary the number of epochs from 10 to 100 in a step of 10 and show the loss value curve (using the best learning rate obtained from (a))

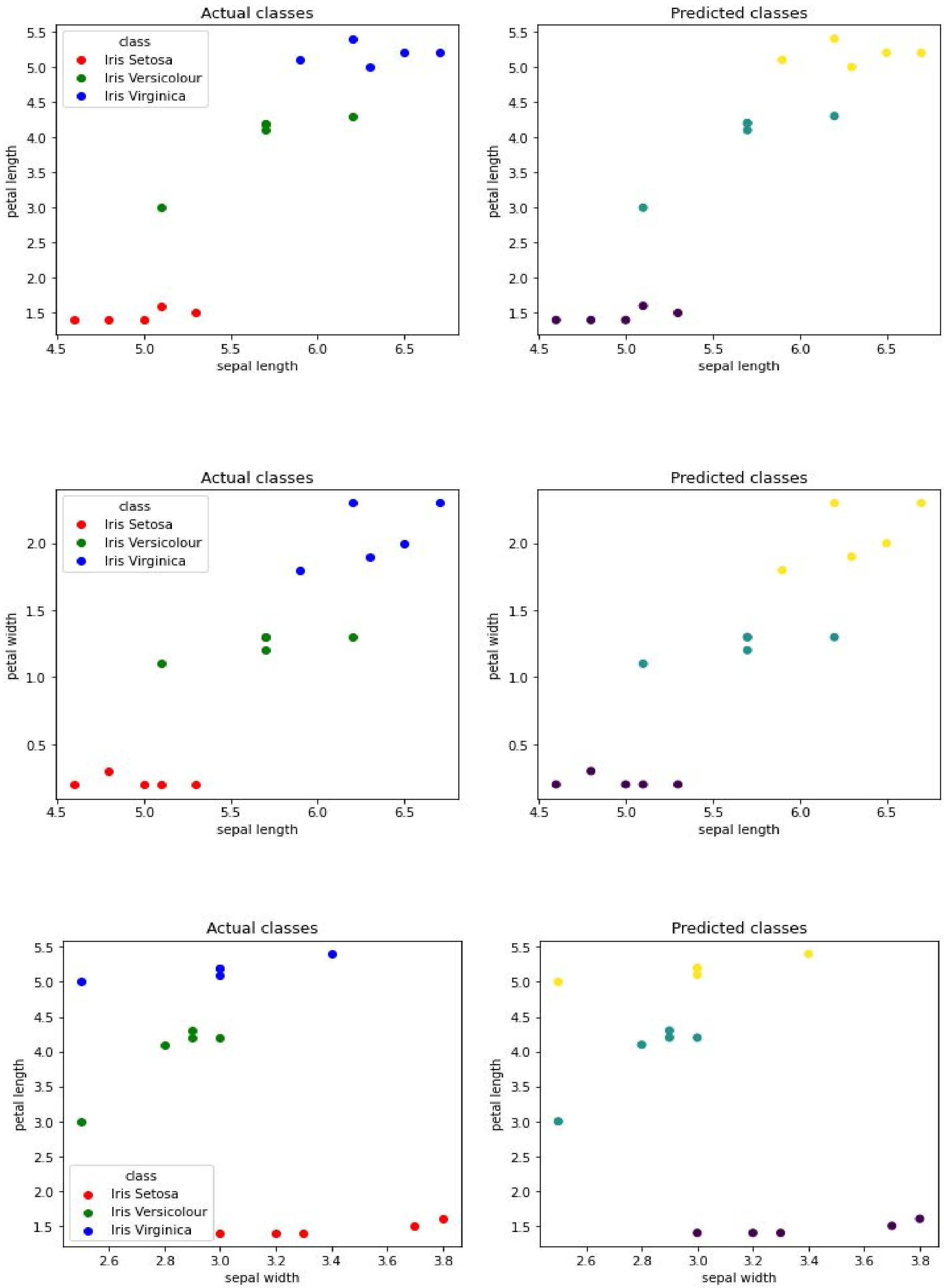
Taking the best value of learning rate from the above experiment and running for 10 to 100 epochs in increments of 10 gave the following results.
As the epochs increase the test loss is reducing.

Error Rate V/S Epoch

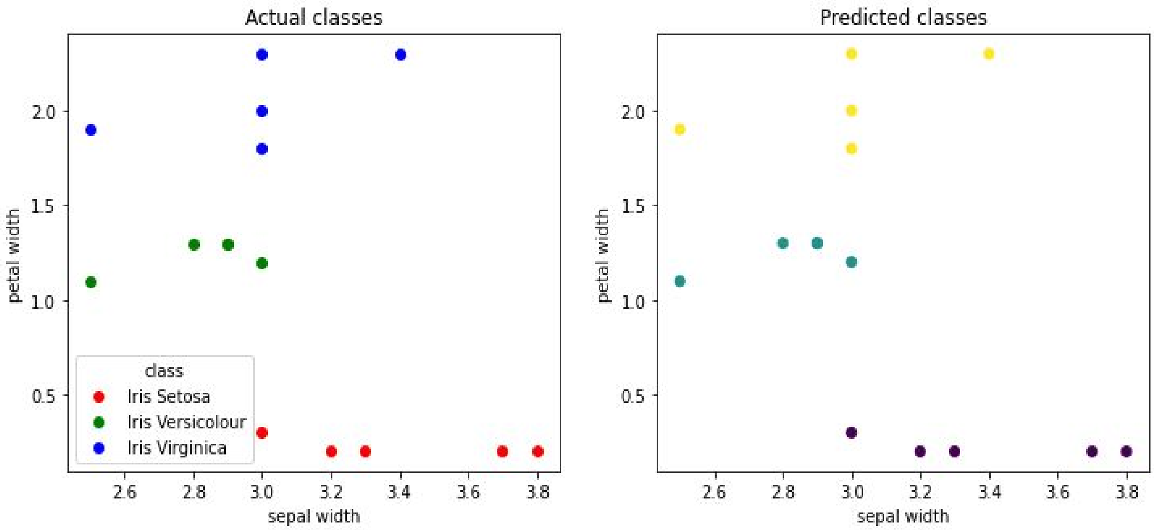


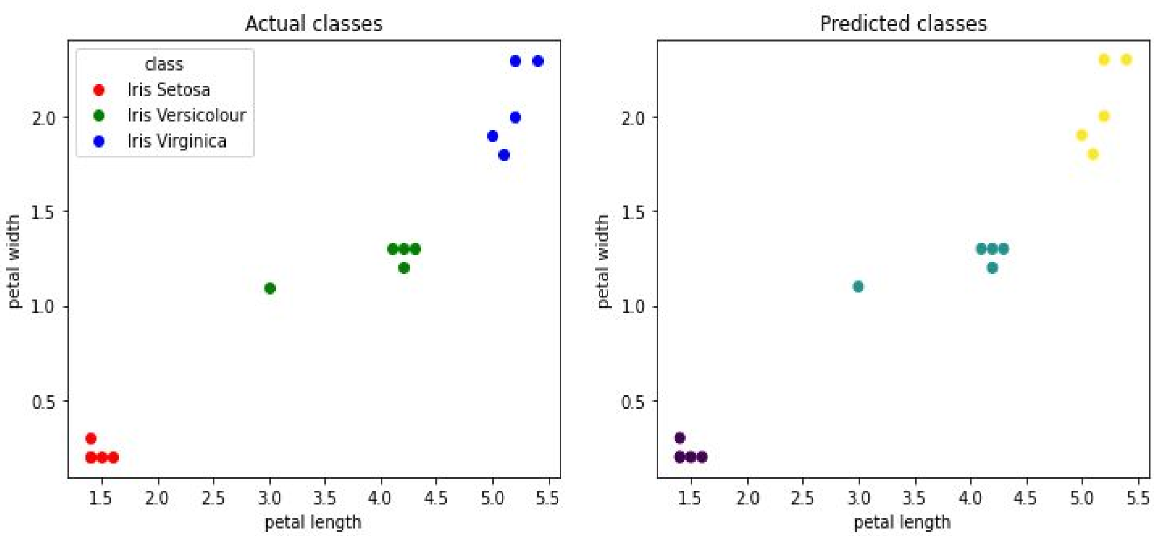
Result of the Testing:
In the results below, the test data gave the results on the Right part of the diagram. The right parts are the true classes/actual classes of each test sample. The diagrams are drawn for each combination of the features Totall 3*2=6 combinations of the features]





Actual classes
Predicted classes
petal length
petal width
sepal length
sepal width
class
Iris Setosa
Iris Versicolour
Iris Virginica





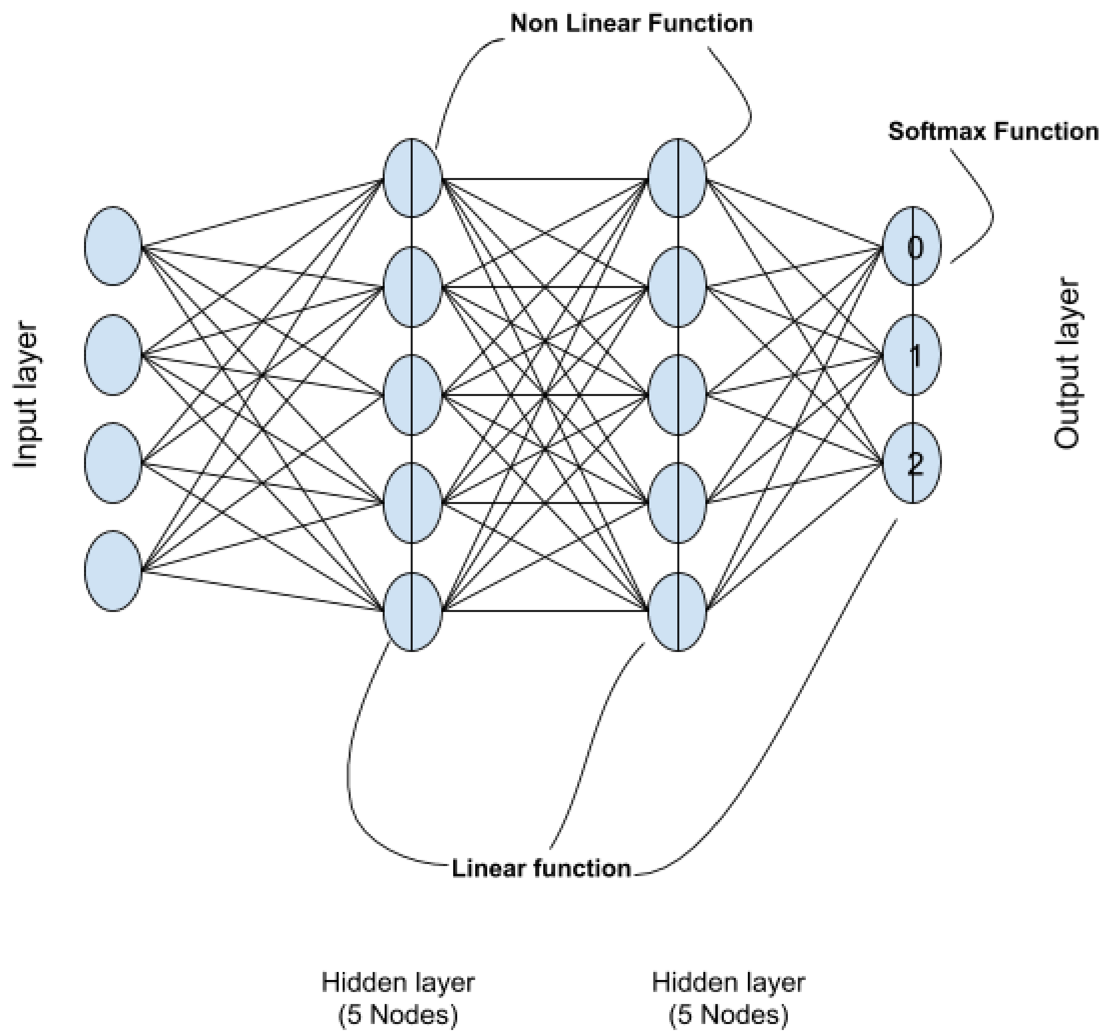
Conclusion:

From the above two experiments done, the best learning rate and the best epoch gave an accurate test result. The experiments can be done on a larger dataset to see the efficiency.

Q2. Implement a 3-class backpropagation NNet on your own to classify iris data, i.e. from scratch. You should not be using any inbuilt function for this implementation (except reading the data).

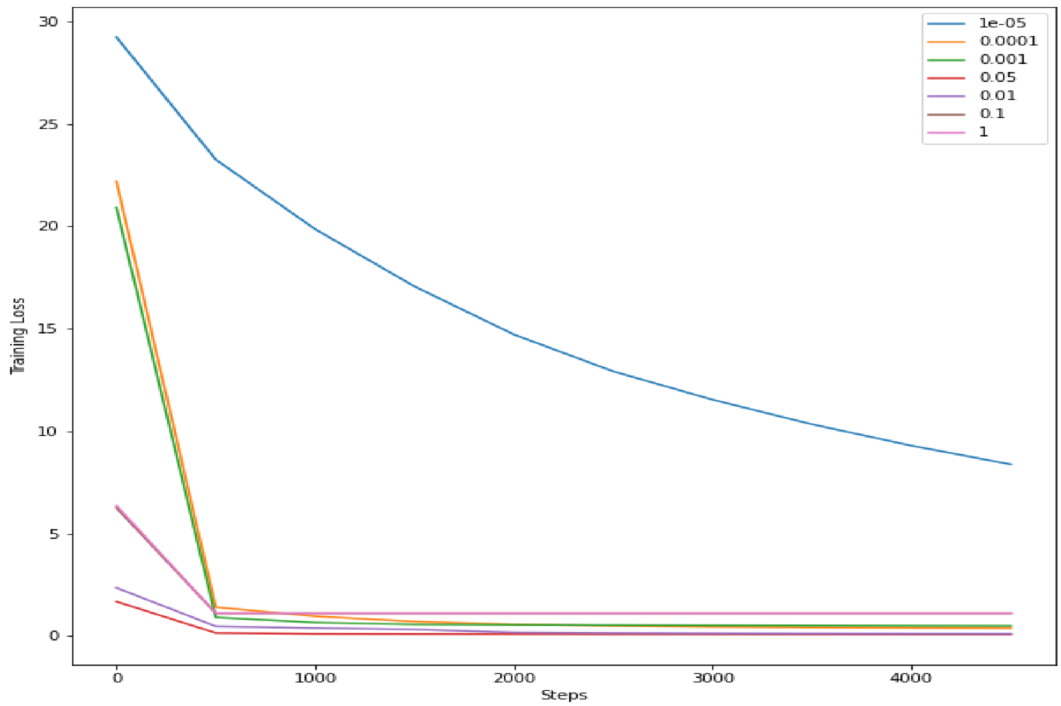
Dataset: IRIS DATA SET
Features: Sepal length in cm,Sepal width in cm, Petal length in cm,Petal width in cm
Classes: Iris Setosa, Iris Versicolour, Iris Virginica
Reference: https://archive.ics.uci.edu/ml/datasets/iris

Model: 3 Layer Neural Network 3 class classification
Output layer: is the softmax probabilities
Loss function: Cross Entropy
Objective is the minimize the Cross Entropy loss + L2 norm(w)
Learning Algorithm: Gradient Descent with backpropagation

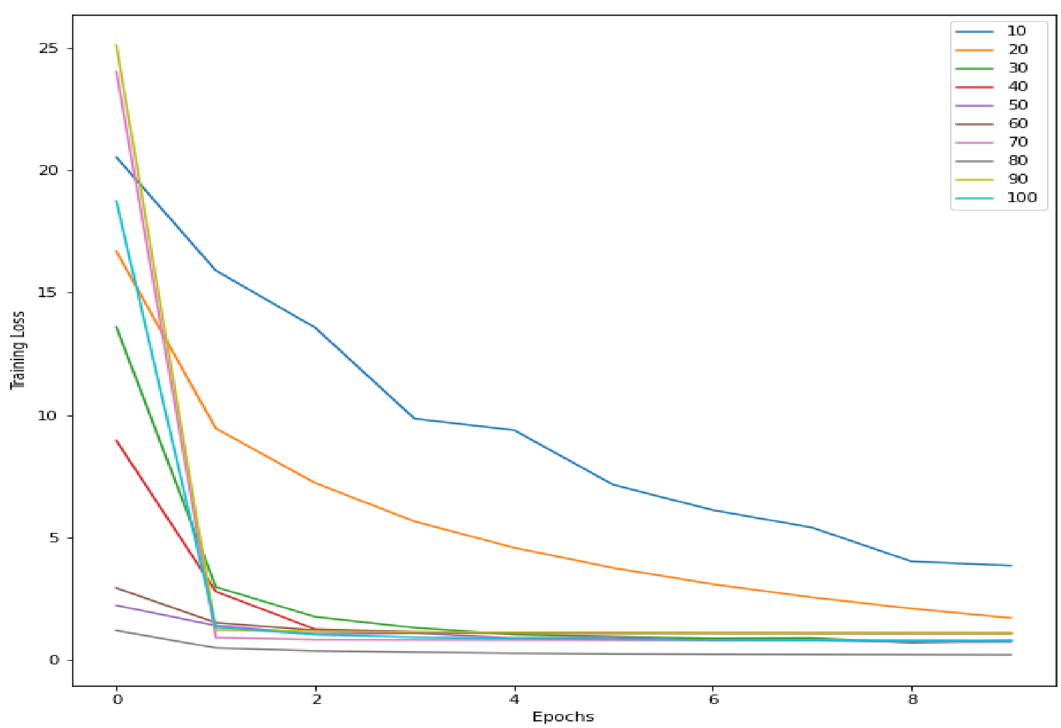


Task 1: Running the experiment for fixed eta=5000 and varying learning rates

Learning Rate chosen `[0.00001,0.0001,0.001,0.05,0.01,0.1,1]`

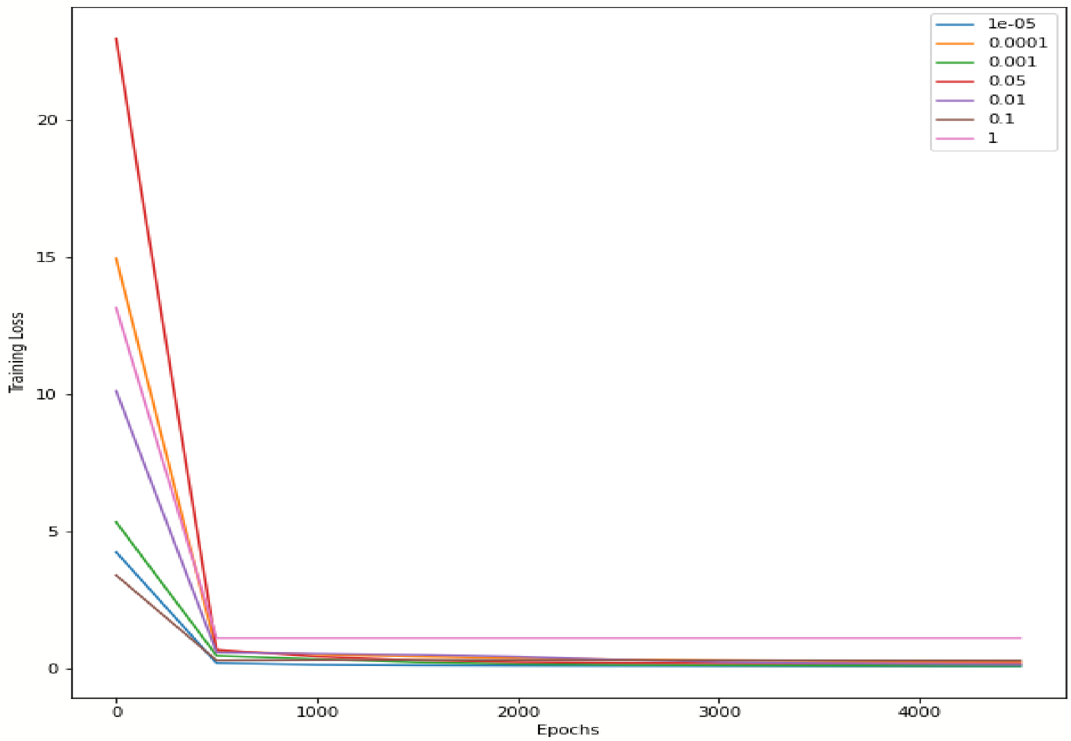


Task 2: Running the experiment for fixed learning rate and for different Epochs
Epochs `[10,20,30,40,50,60,70,80,90,100]`



Task 3:Introducing L2 Regularization to the gradient updates of weights

```
Regularization Parameters chosen [0.00001,0.0001,0.001,0.05,0.01,0.1,1]
```



Test Results and conclusion: Classification of test data gave the following results

The results may vary depending on the weight initialization

Loss at step   0: 1.4091392382680314

Loss at step 500: 0.21624354805485005

Loss at step 1000: 0.08886141914873352

Loss at step 1500: 0.7633213467753158

Loss at step 2000: 0.07946941873468218

Loss at step 2500: 0.07704998414721871

Loss at step 3000: 0.07623777706102967

Loss at step 3500: 0.07612791330640656

Loss at step 4000: 0.09608262838384918

Loss at step 4500: 0.07129482137392218

The results may vary depending on the weight initialization

Training accuracy for eta 0.01 epochs 5000 = 98.33%

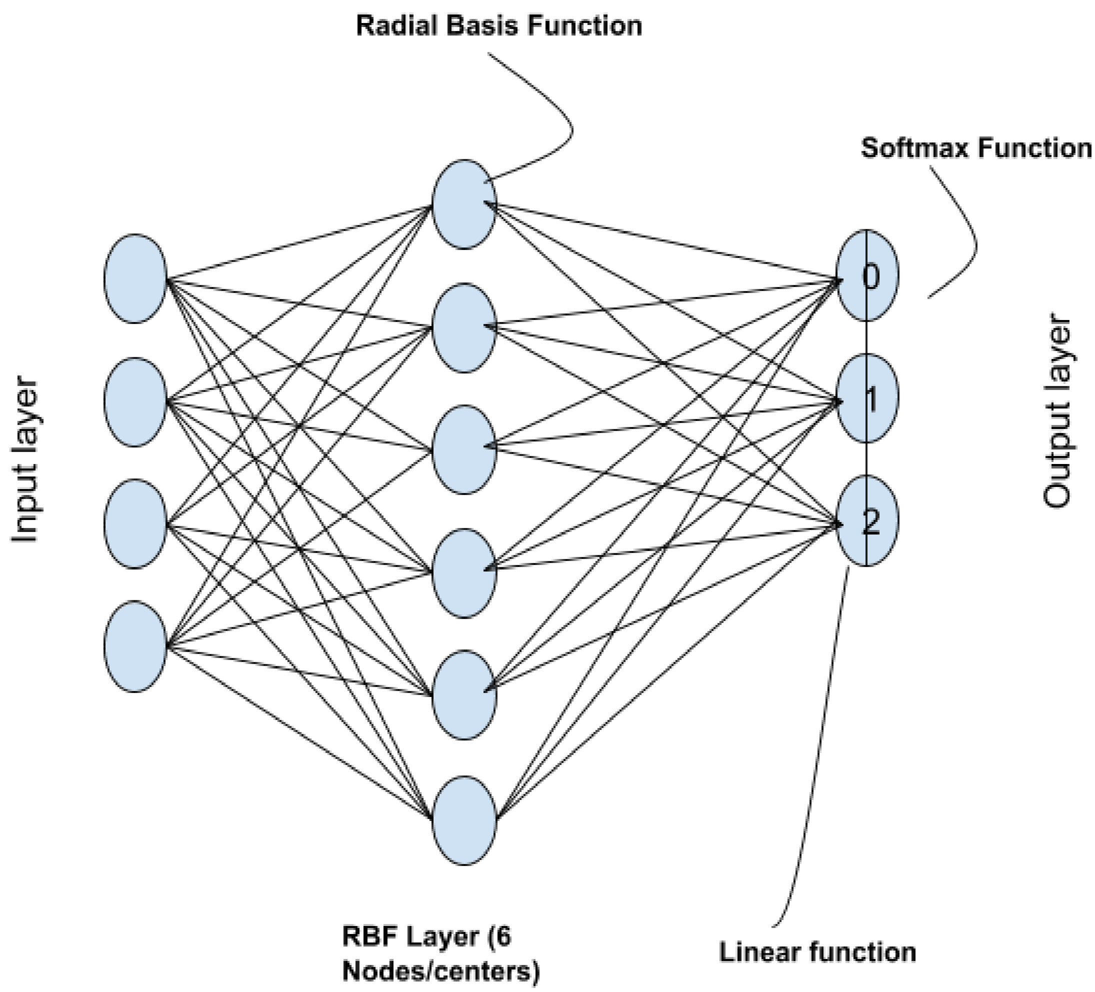
Testing accuracy for eta 0.01 epochs 5000 = 100.00%

Q3. Use any toolbox in python and implement RBF NNet to solve one of the problems/databases (of your choice from the UCI ML database Repo). Analyze your results with respect to varying learning rate and epochs. You are not allowed to use someone's code available online. UCI databases: https://archive.ics.uci.edu/ml/datasets.php

Dataset: IRIS DATA SET
Features: Sepal length in cm,Sepal width in cm, Petal length in cm,Petal width in cm
Classes: Iris Setosa, Iris Versicolour, Iris Virginica
Reference: https://archive.ics.uci.edu/ml/datasets/iris

Learning Algorithms: KMeans, KNN, Gradient descent with backpropagation
Loss: Cross Entropy
Output: Softmax Probabilities

RBF Neural Network Architecture



Training:

1. **Number of neurons is equal to the number of centers/representatives in the given dataset. For the IRIS dataset, the centers chosen =6. Hidden layer has 6 neurons**
2. **Gaussian function is used as the representation of Radial Basis Neuron**
3. **The final layer is trained using backpropagation and gradient descent algorithm**

Layers

a. The centers corresponding to each hidden layer neuron is found using KMeans concepts implemented here in the code

   KMeans Algorithm:

   Randomly initialize the centers from the feature vector,

   Do

   for each point in the feature vector

   Assign it to the closest center using euclidean distance,

   After all points are assigned to a center/cluster, find the mean of points with each cluster to get the new centers

   Repeat this procedure until max iterations or until the centers do not change

b. The spread/radius of each gaussian is found using K Nearest Neighbour function implemented here in the code

   Procedure:

   For each center find the distances to every other point in the feature space, pick the k nearest neighbours

   Radius_i = Sqrt((Sum(center - ith_neighbour)^2)/k)

c. The output layer is the softmax probabilities, the target is converted to a one hot vector encoding, to find the cross entropy loss to two distribution

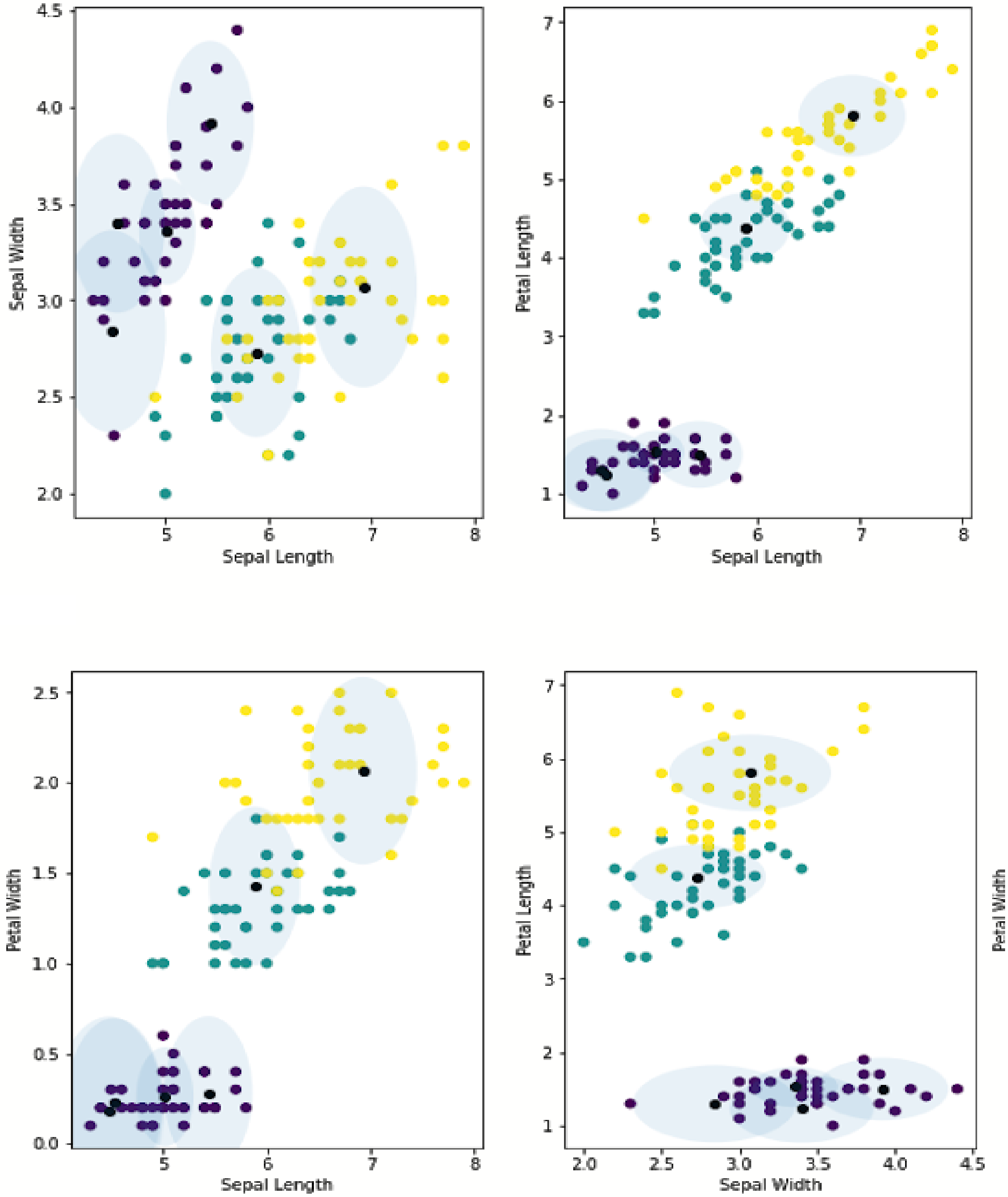
.

For the IRIS Dataset:

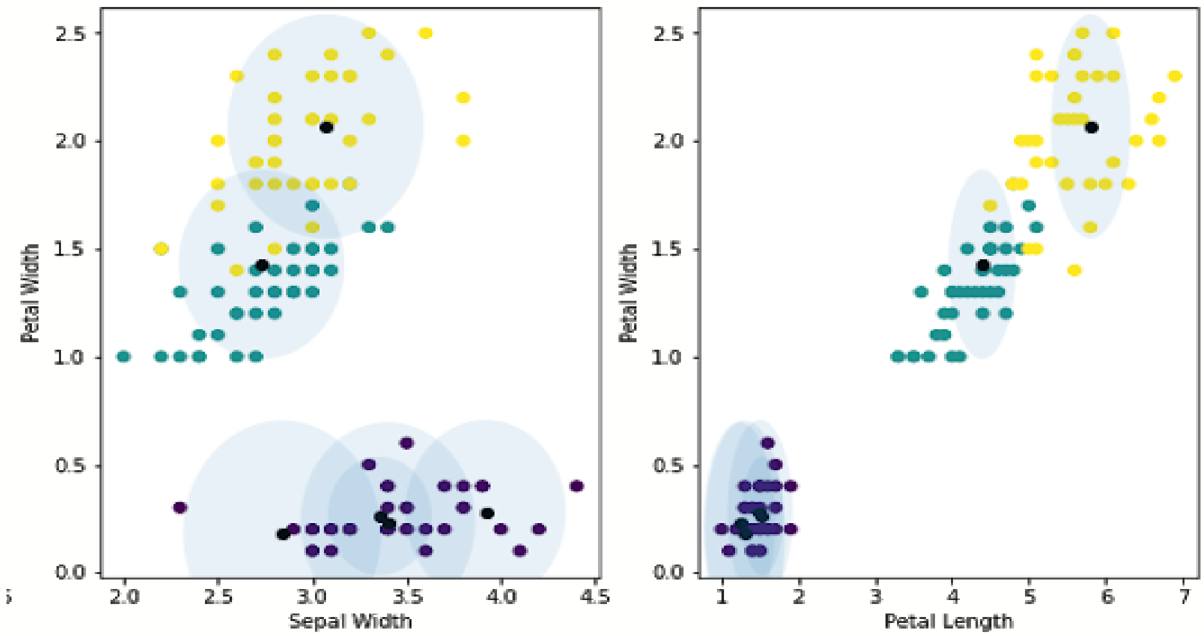
Centers chosen = 6

Number of nearest neighbors = 15

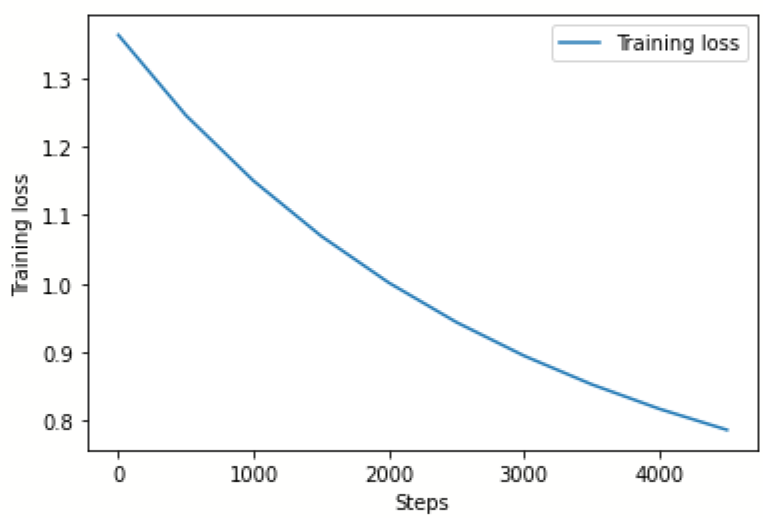
Below diagram shows the centers and the spread of the radial basis.
Each diagram is plotted for a combination pair of every feature. The color encodings is that of the classes representing that feature



Below diagram shows the centers and the spread of the radial basis.
Each diagram is plotted for a combination pair of every feature. The color encodings is that of the classes representing that feature



Below figure depicts the decrease in training loss with epochs (5000)



The results may slightly differ based on the weights initialization

Loss at step 0: 1.363112556960716
Loss at step 500: 1.2456909497813247
Loss at step 1000: 1.150052154213281
Loss at step 1500: 1.0694052255866888
Loss at step 2000: 1.0012612582940676
Loss at step 2500: 0.9435614288639236
Loss at step 3000: 0.8945610154100447
Loss at step 3500: 0.8528013127213779
Loss at step 4000: 0.8170737385077516
Loss at step 4500: 0.7863823611147417

Number of misclassification = 13
Training accuracy: 89.17%
Test accuracy: 86.67%

Q4. Using MNIST database, code Autoencoder model with three encoding and three decoding layers. Show the visualization of the feature maps. On the features, add a classifier to perform 10-class classification and show the training loss curve and test accuracy.

Dataset:

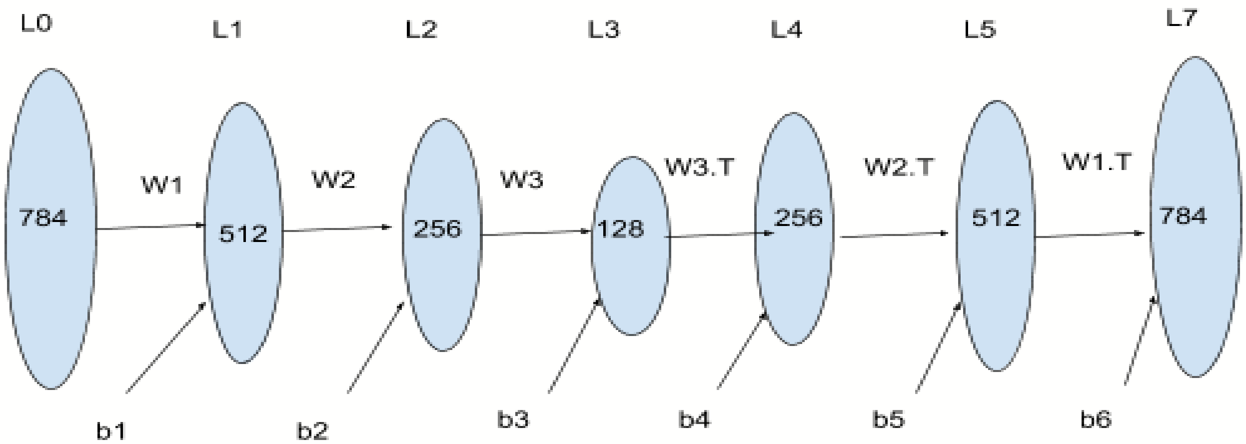
The MNIST is a database of handwritten digits as images. It has a training set of 60000 images, and a test set of 10000 images. The image data have been size-normalized and centered in a fixed-size. Each image is a 28*28 pixel grayscale images of handwritten single digits between 0 and 9.

[References: http://yann.lecun.com/exdb/mnist/]

Autoencoder Model

The autoencoder has 3 encoding layers + 3 decoding layers

The weights are share between the first 3 and the last three to make the learning greedy by layer

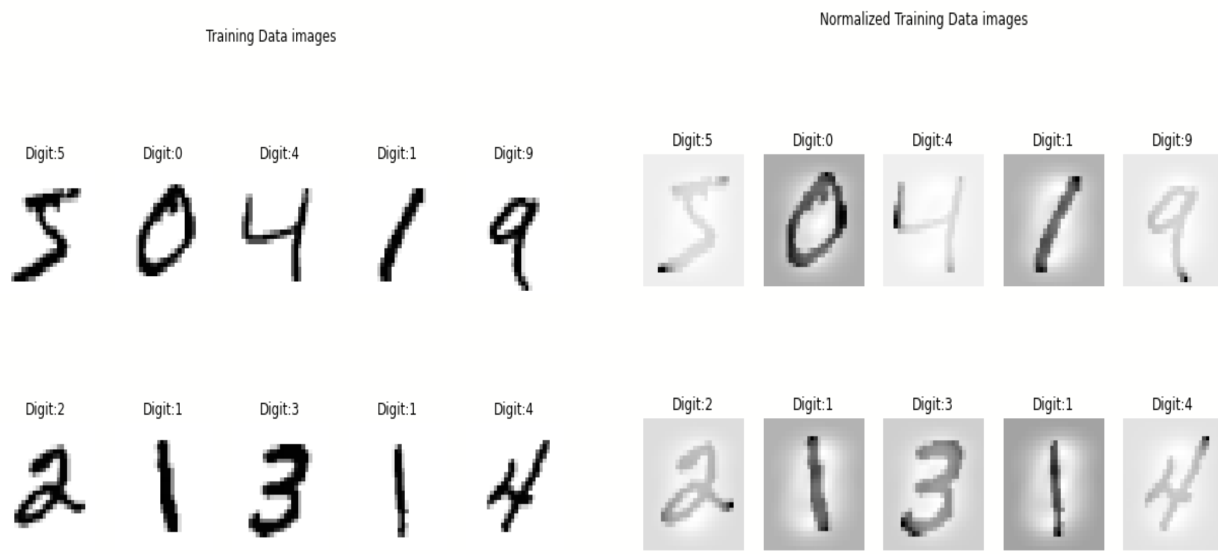


Layer0	Layer1	Layer2	Layer3	Layer4	Layer5	Layer6
W1	W2	W3	W3*	W2*	W1*	
x	a->h	a->h	a->h	a->h	a->h	a = x'

Here, a is the linear function and h is the logistic activation function

Wi* are the transposes of W, Only 3 weight vector are used in the autoencoder

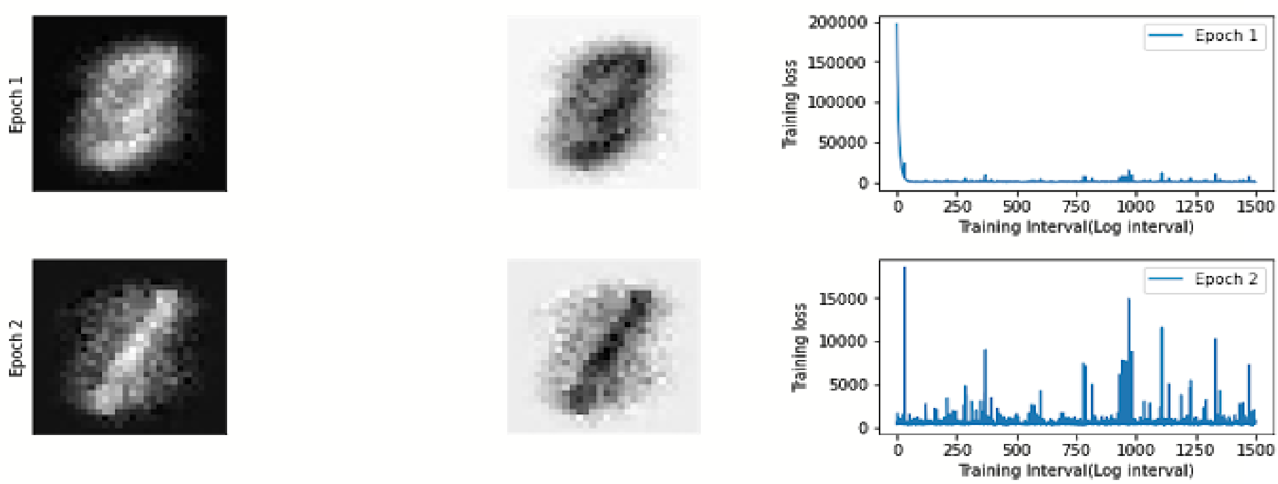
Training: Input to the autoencoder is normalized. Original data unnormalized data gave very large errors. Since the training data is very large(60000), gradient descent is run on a **minibatch** of the training data for epoch runs of the entire data

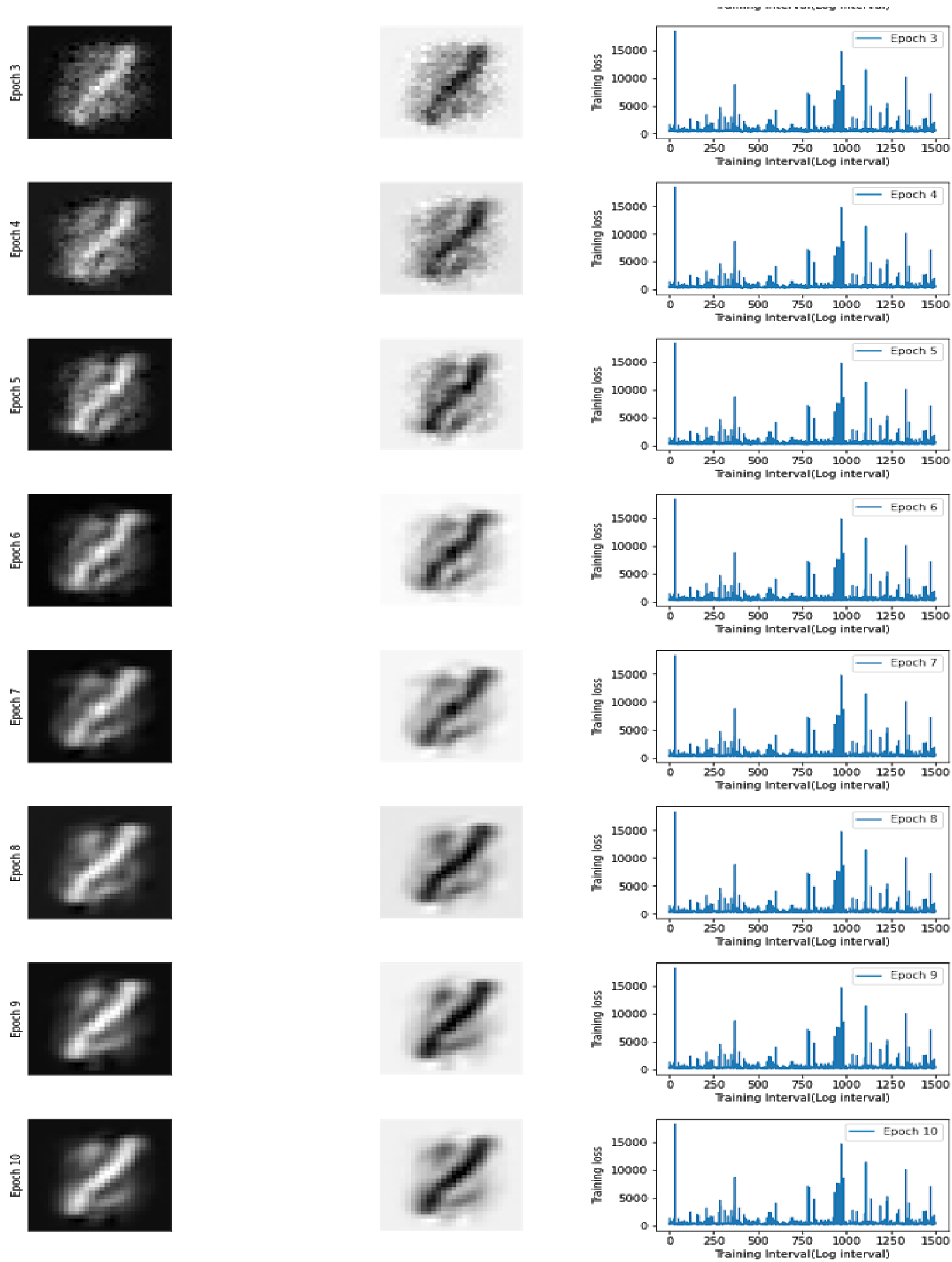


Feature maps/Outputs of the auto encoder:

Plotting the features learn in 10 epochs

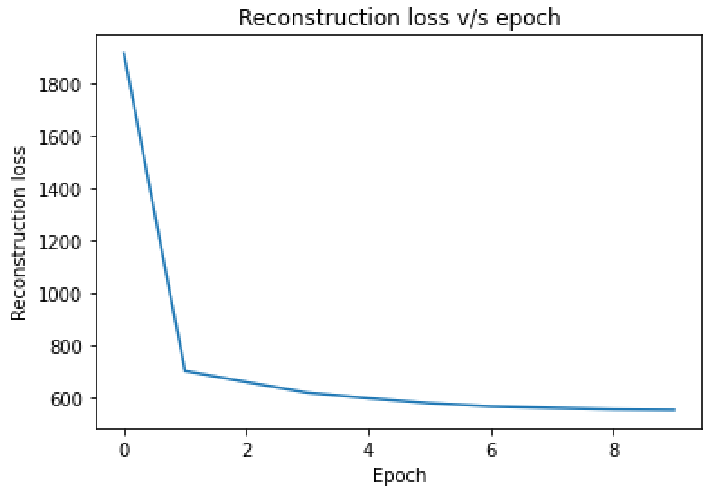
Diagram 1: Gray scale, Diagram 2: Binary Image, Diagram 3: Training Loss v/s Training interval



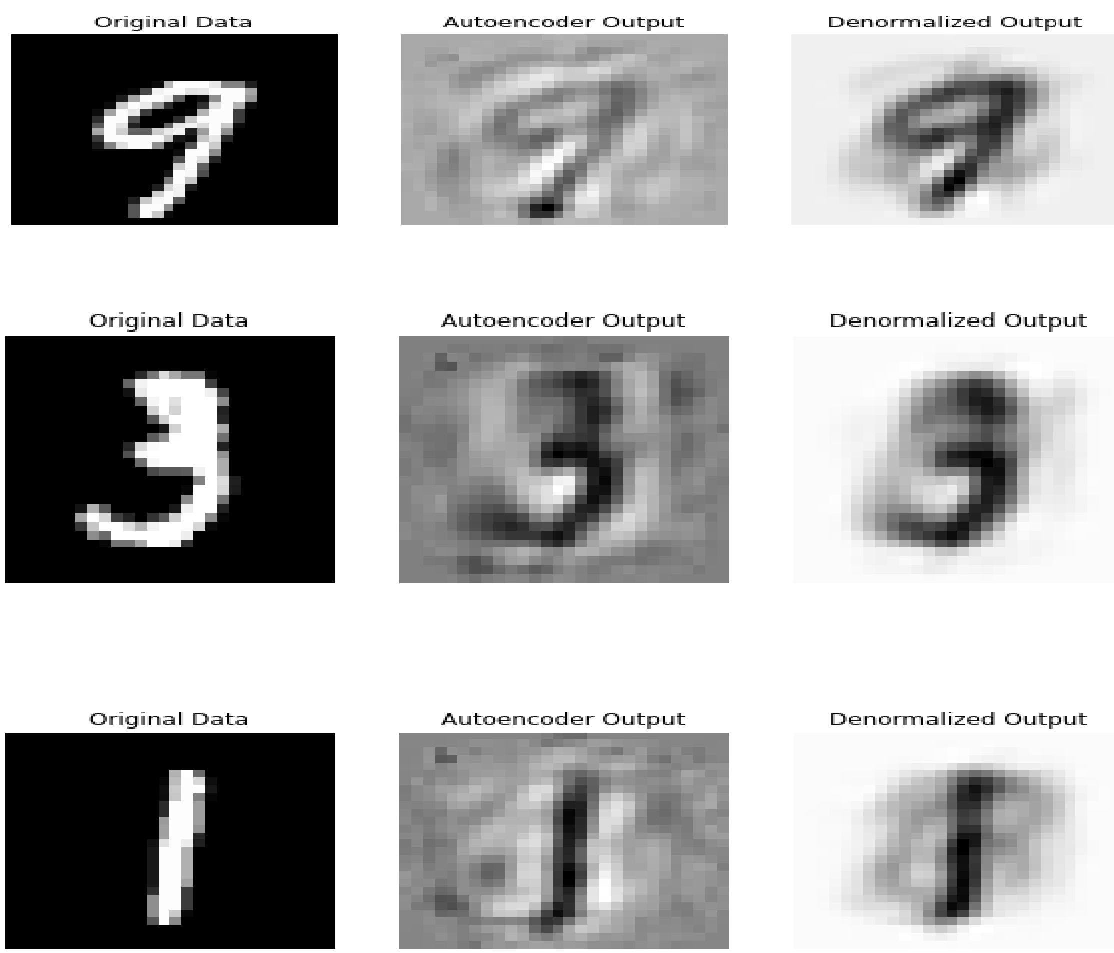


Epoch 3
Epoch 4
Epoch 5
Epoch 6
Epoch 7
Epoch 8
Epoch 9
Epoch 10
Training loss
0
5000
10000
15000
0
250
500
750
1000
1250
1500
Training Interval(Log interval)

The experiment was run for 8 epoch of the training data,



Column 2 is the new representation of the autoencoder output, Column 3 is the de normalized new representation of the autoencoder output

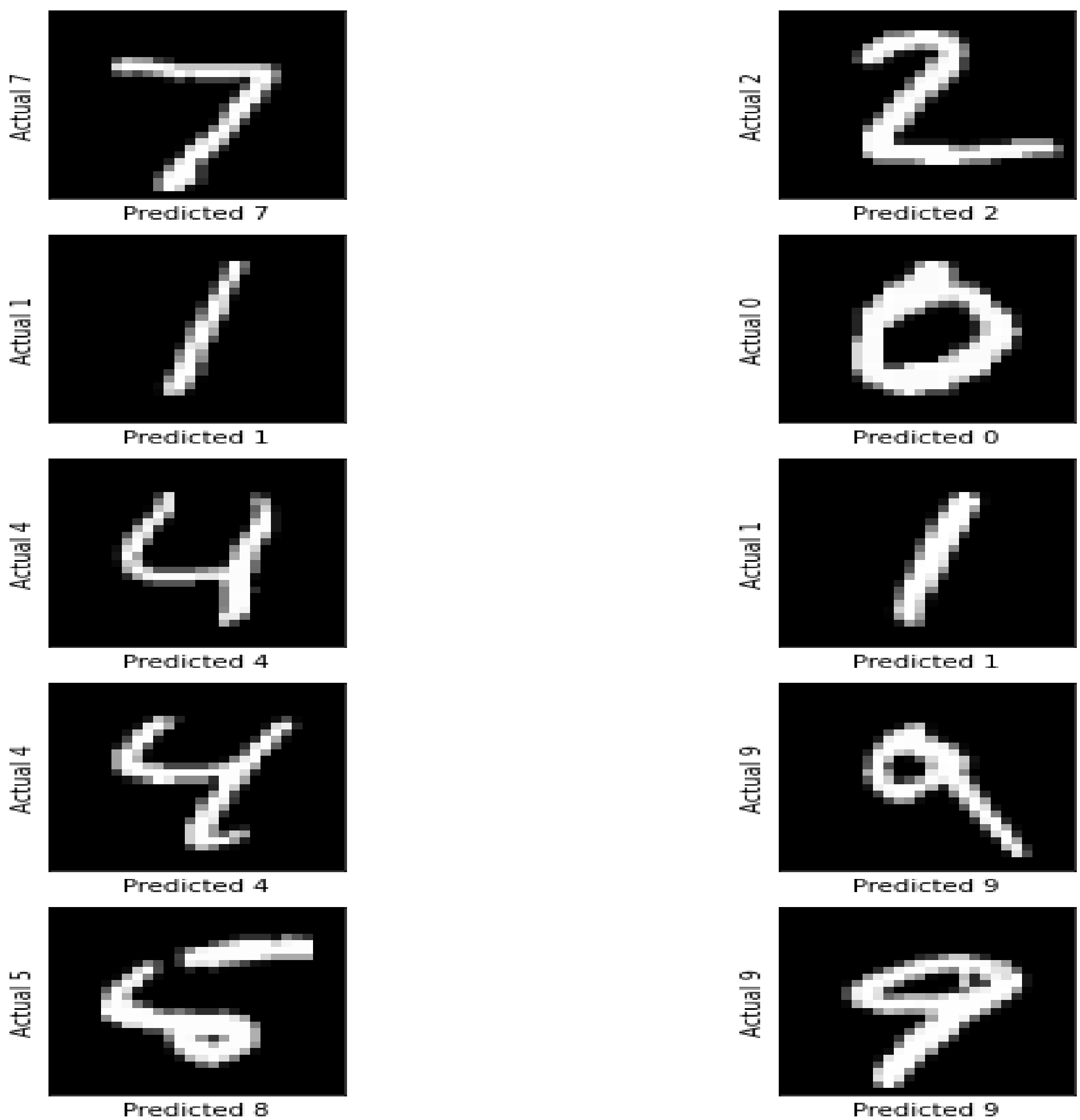


Classification of the new representation of the data

The output of the autoencoder is used for classification. A CNN model provided as the keras function is used for this task.
Classification is done using a 3 layer CNN - input layer, dense layer with relu and output layer
Loss function used: Cross entropy

`Test results:`

Classification of test data gives an accuracy :96.800%



References:

Dataset Reference:

1. Q1-Q3: https://archive.ics.uci.edu/ml/datasets/iris
2. Q4: http://yann.lecun.com/exdb/mnist/

Algorithm Reference:

1. https://www.ics.uci.edu/~pjsadows/notes.pdf [Backpropagation]
2. https://web.stanford.edu/class/cs294a/sparseAutoencoder_2011new.pdf [Autoencoder]

Tools used:
Language: Python
Library Functions: Numpy, Matplotlib, Sklearn dataset, Math, Keras for MNIST data and classifier

Other References:
Dr. Mayank Vatsa's class PPTs
Dr. P K Biswas conceptual video lecture on RBF Neural Network[NPTEL]