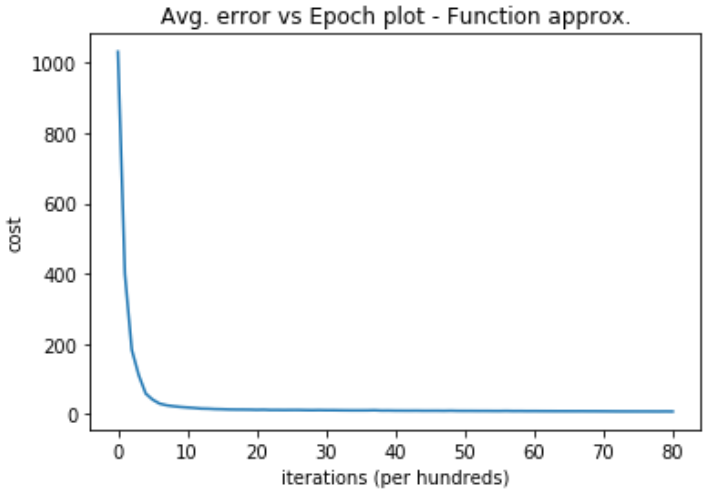
DEEP LEARNING (CS6910)

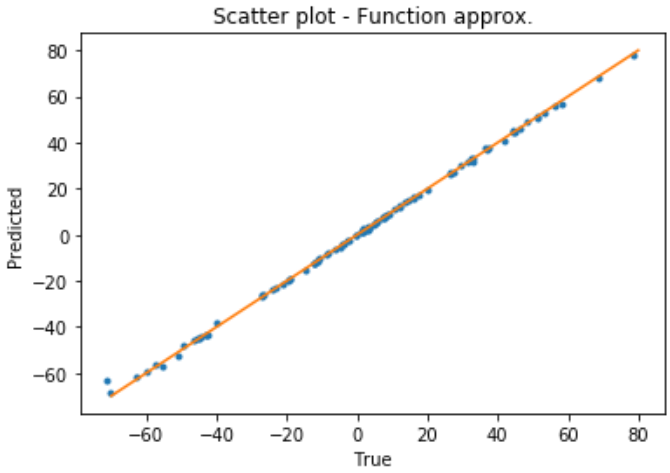
Programming Assignment 1

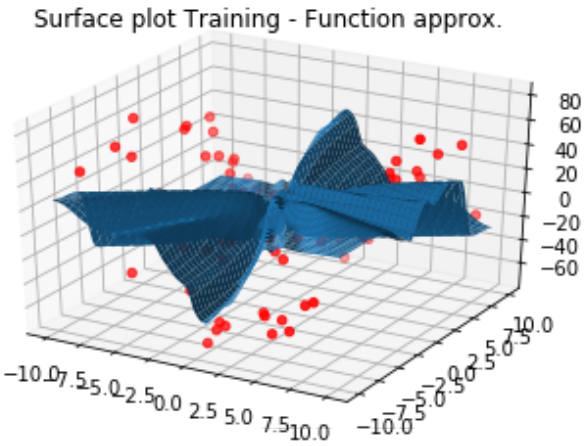
MLFNN with 2 hidden layers

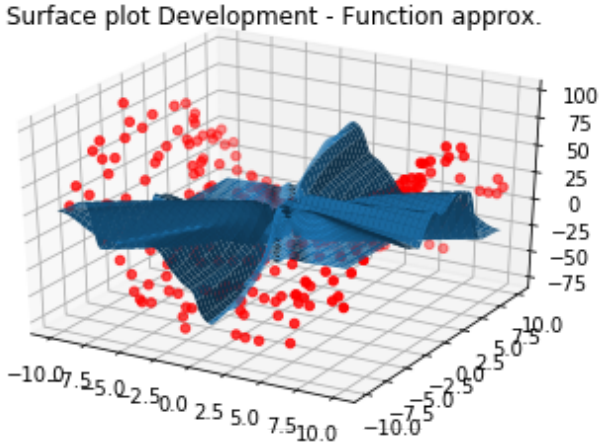
Function approximation task

Plots :









Inference :

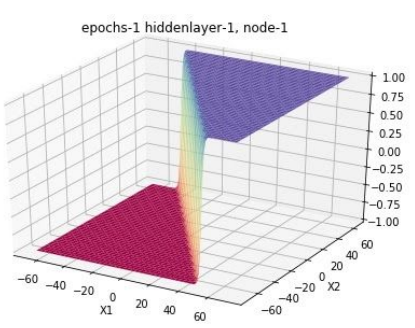
- Learning rate = 0.01, Delta cost = 10^(-6)
- Nodes in first hidden layer = 30, second hidden layer = 10

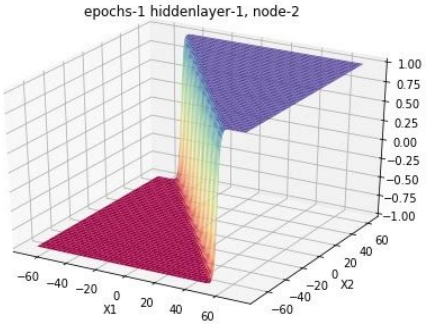
- RMS error for training is 1.01 , testing data is 11.82

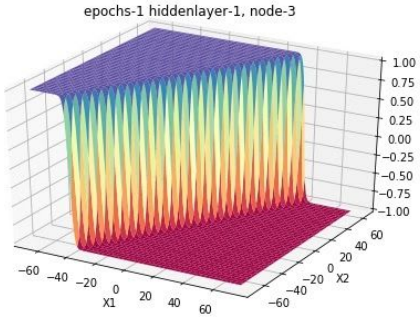
Classification task for 2-d data

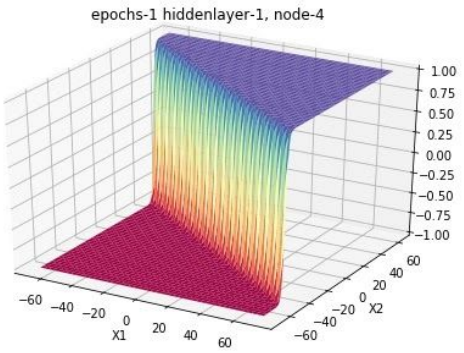
Plots :

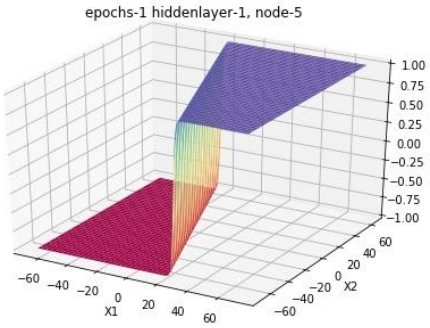
Epoch1

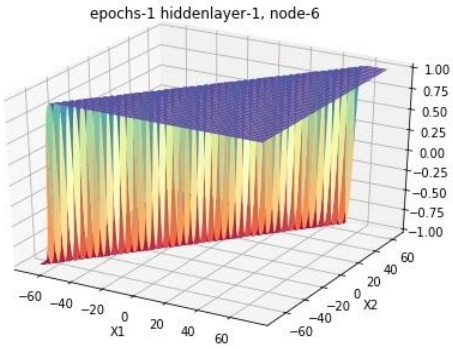


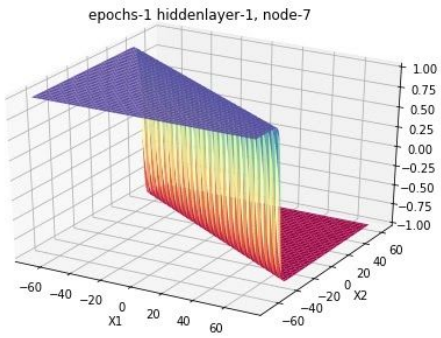


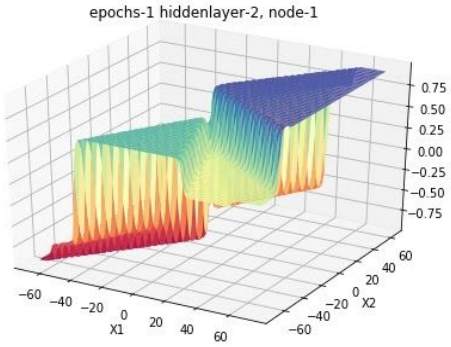


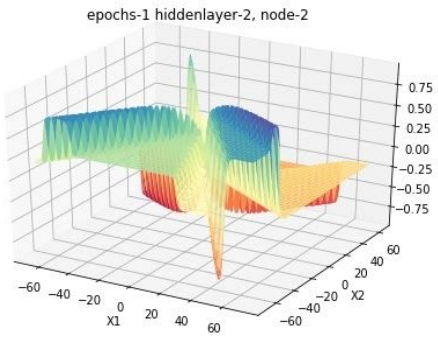


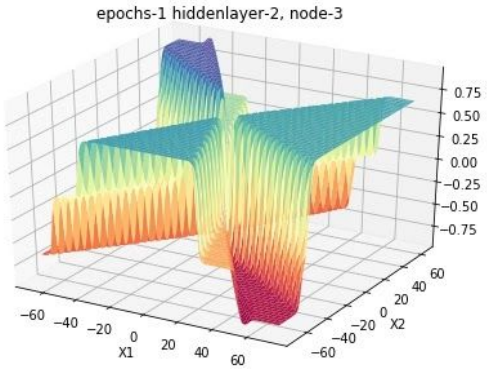


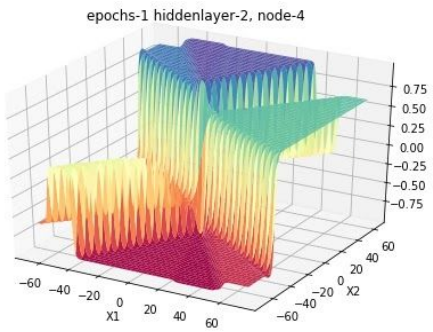


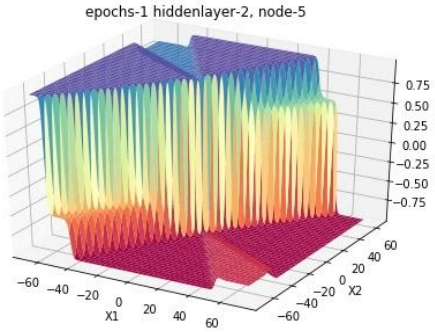


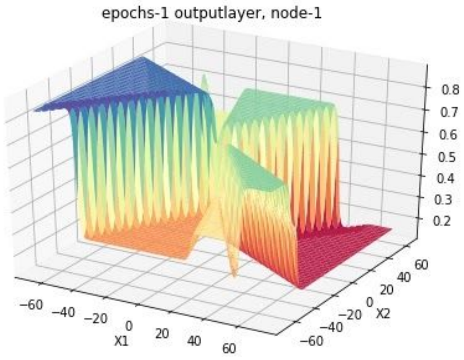


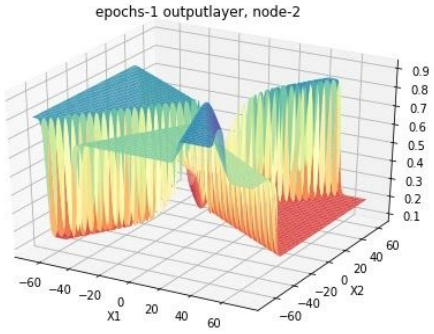


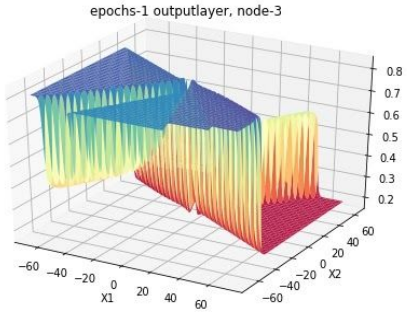




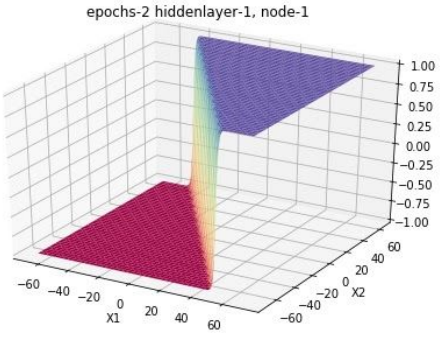


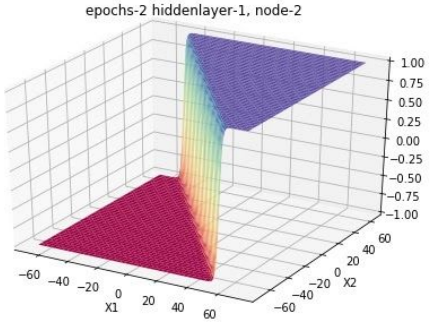

epochs-1 outputlayer, node-1
X1
X2

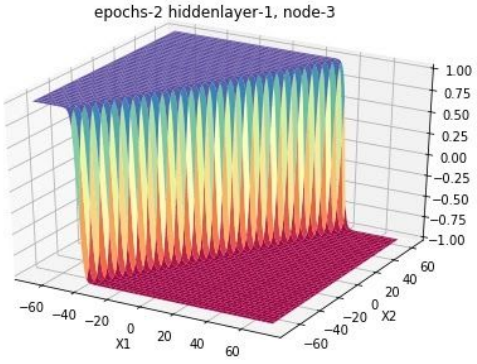

epochs-1 outputlayer, node-2
X1
X2

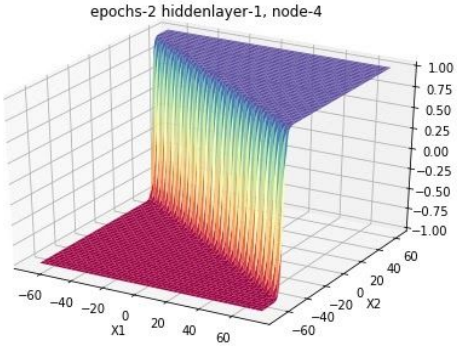

epochs-1 outputlayer, node-3
X1
X2

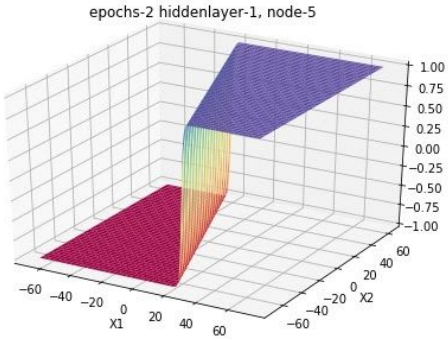
Epoch 2

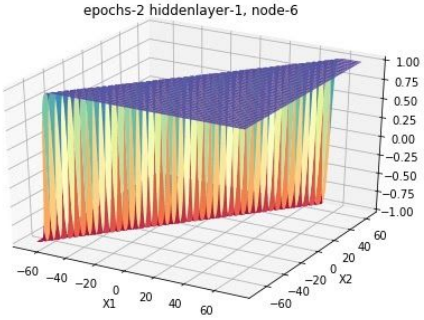

epochs-2 hiddenlayer-1, node-1
X1
X2

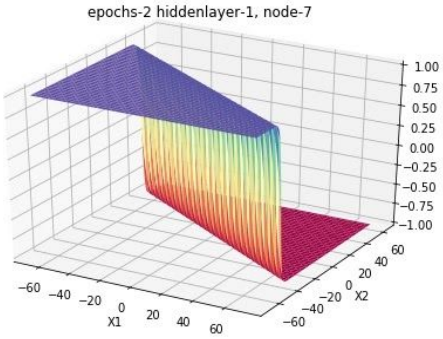

epochs-2 hiddenlayer-1, node-2
X1
X2

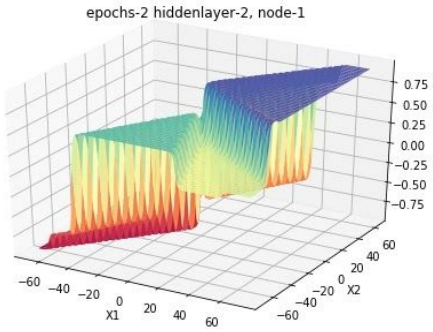

epochs-2 hiddenlayer-1, node-3
X1
X2

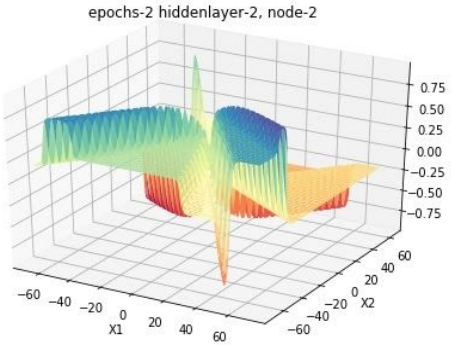

epochs-2 hiddenlayer-1, node-4
X1
X2

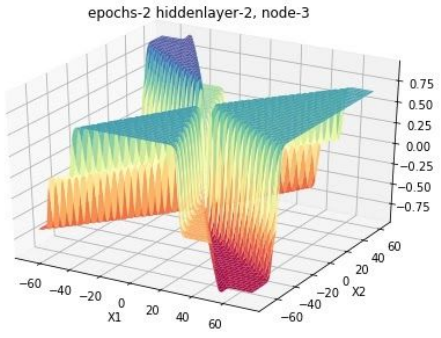

epochs-2 hiddenlayer-1, node-5
X1
X2

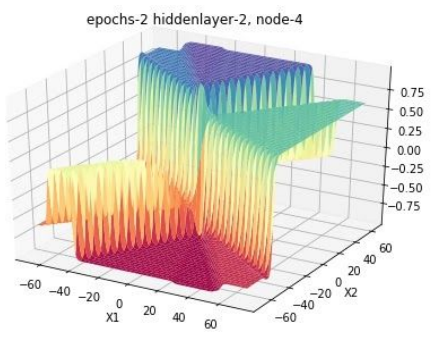

epochs-2 hiddenlayer-1, node-6
X1
X2

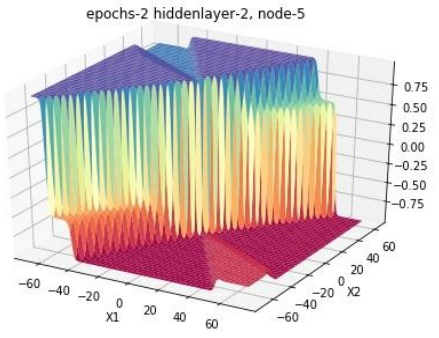

epochs-2 hiddenlayer-1, node-7
X1
X2

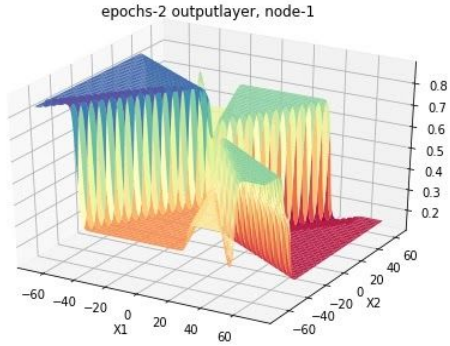

epochs-2 hiddenlayer-2, node-1
X1
X2

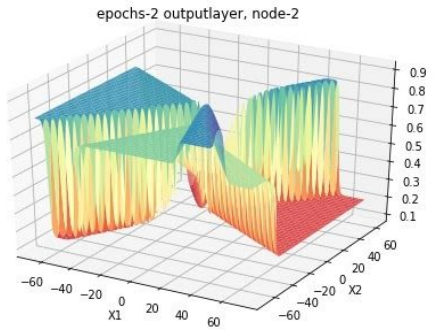

epochs-2 hiddenlayer-2, node-2
X1
X2

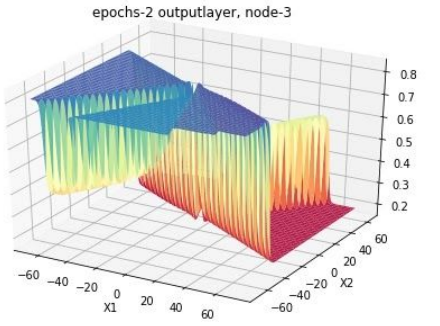

epochs-2 hiddenlayer-2, node-3
X1
X2


epochs-2 hiddenlayer-2, node-4
X1
X2

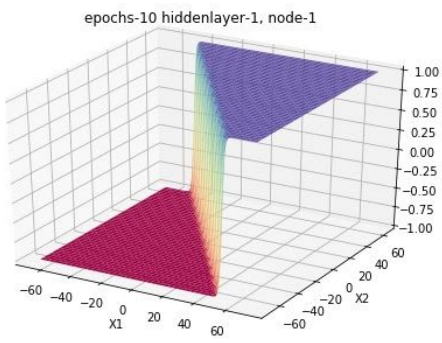

epochs-2 hiddenlayer-2, node-5
X1
X2

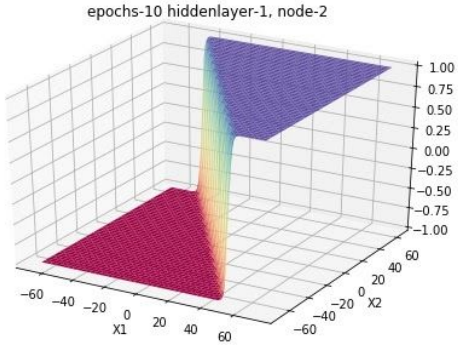

epochs-2 outputlayer, node-1
X1
X2

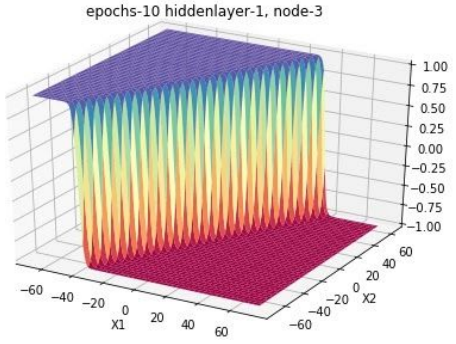

epochs-2 outputlayer, node-2
X1
X2

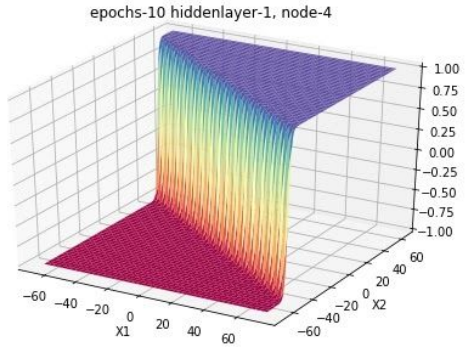

epochs-2 outputlayer, node-3
X1
X2

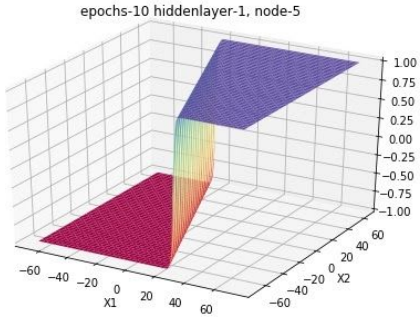
Epoch 10

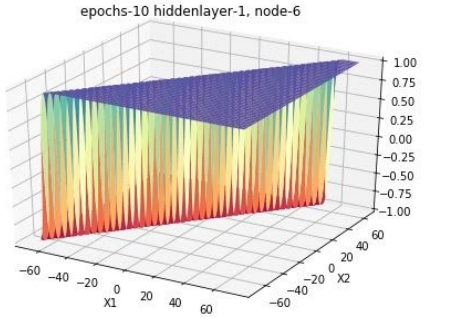

epochs-10 hiddenlayer-1, node-1
X1
X2

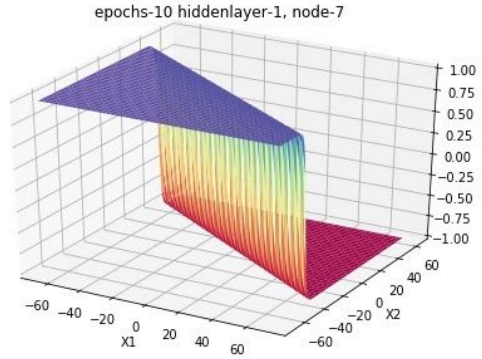

epochs-10 hiddenlayer-1, node-2
X1
X2

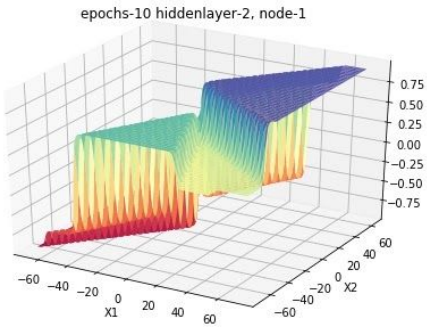

epochs-10 hiddenlayer-1, node-3
X1
X2

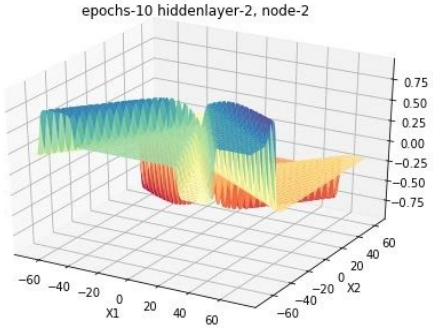

epochs-10 hiddenlayer-1, node-4
X1
X2

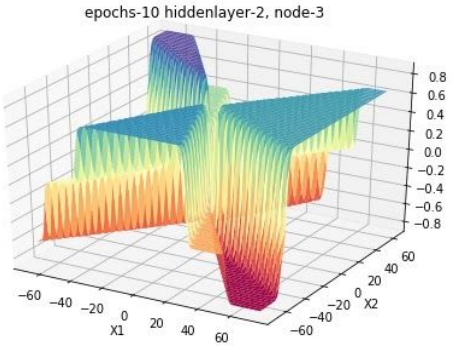

epochs-10 hiddenlayer-1, node-5
X1
X2

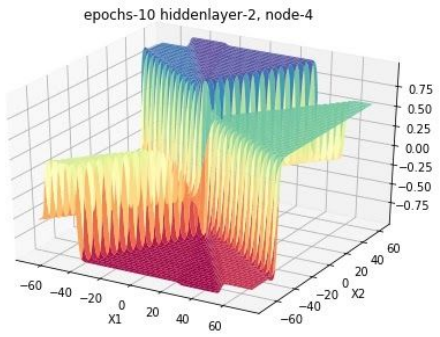

epochs-10 hiddenlayer-1, node-6
X1
X2

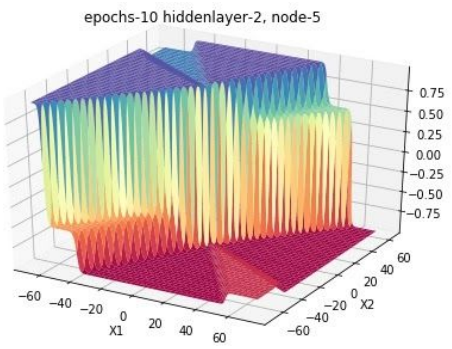

epochs-10 hiddenlayer-1, node-7
X1
X2

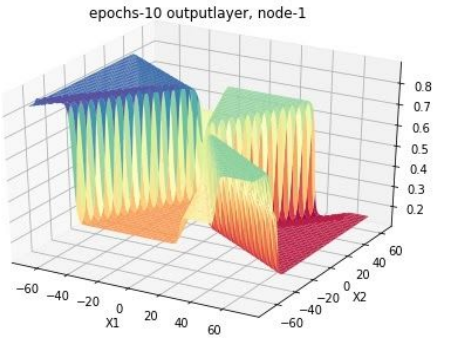

epochs-10 hiddenlayer-2, node-1
X1
X2

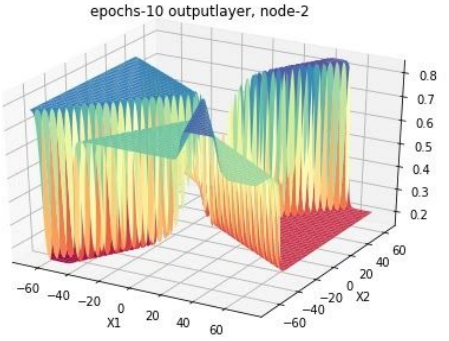

epochs-10 hiddenlayer-2, node-2
X1
X2

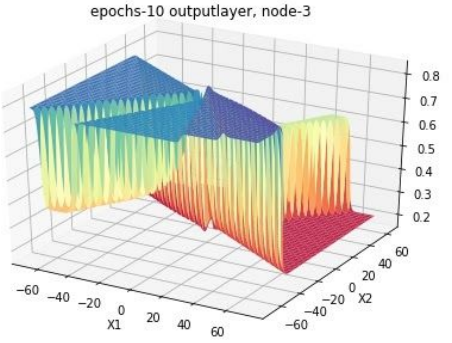

epochs-10 hiddenlayer-2, node-3
X1
X2


epochs-10 hiddenlayer-2, node-4
X1
X2

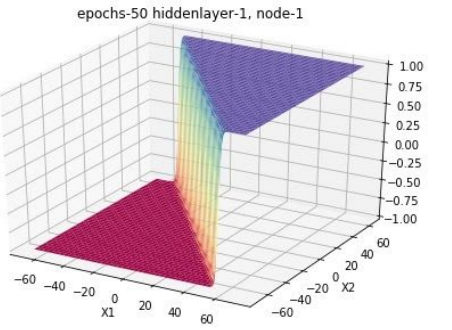

epochs-10 hiddenlayer-2, node-5
X1
X2

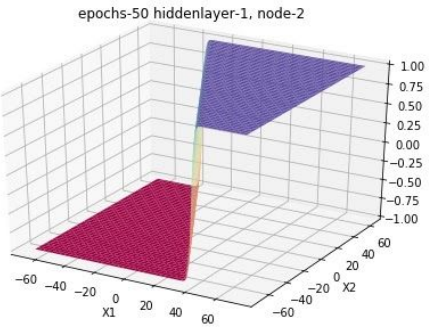

epochs-10 outputlayer, node-1
X1
X2

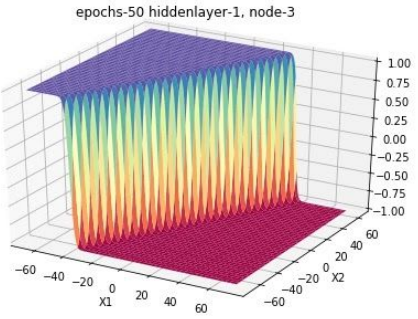

epochs-10 outputlayer, node-2
X1
X2

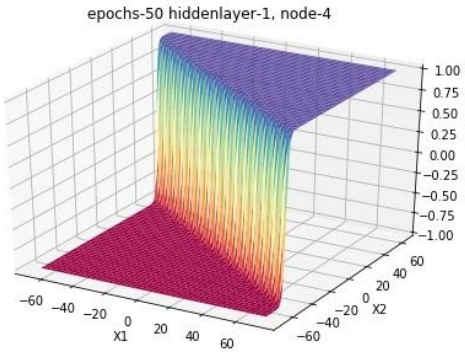

epochs-10 outputlayer, node-3
X1
X2

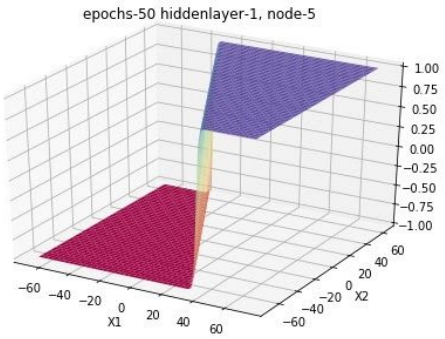
Epoch 50

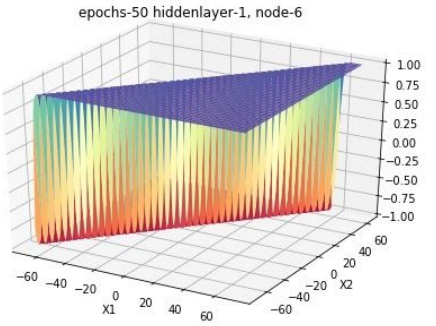

epochs-50 hiddenlayer-1, node-1
X1
X2

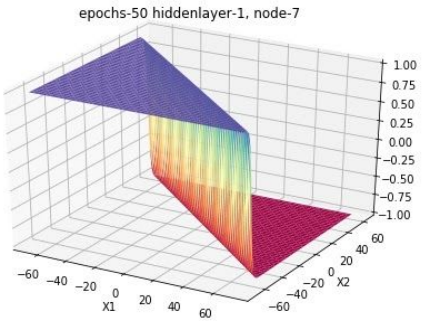

epochs-50 hiddenlayer-1, node-2
X1
X2

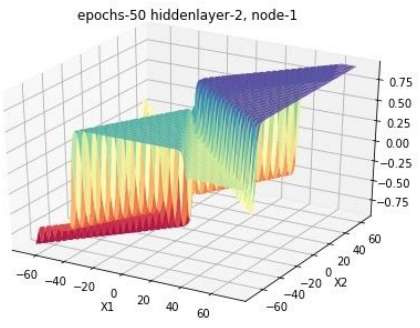

epochs-50 hiddenlayer-1, node-3
X1
X2

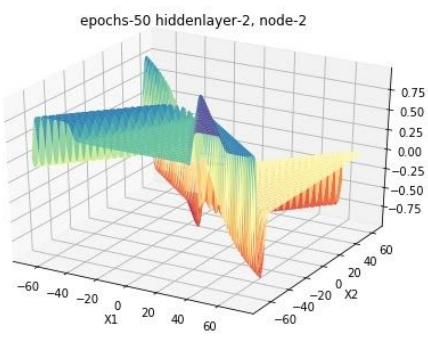

epochs-50 hiddenlayer-1, node-4
X1
X2

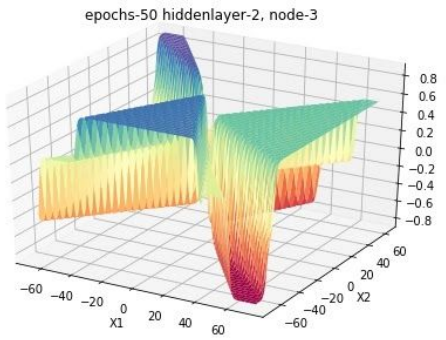

epochs-50 hiddenlayer-1, node-5
X1
X2

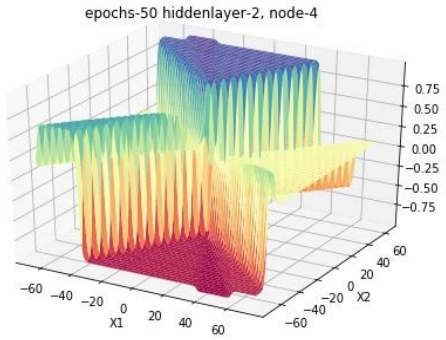

epochs-50 hiddenlayer-1, node-6
X1
X2

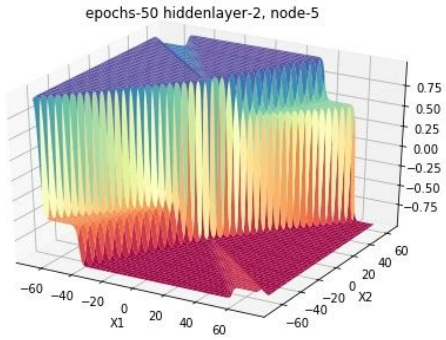

epochs-50 hiddenlayer-1, node-7
X1
X2

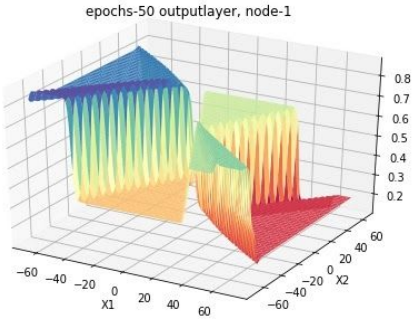

epochs-50 hiddenlayer-2, node-1
X1
X2

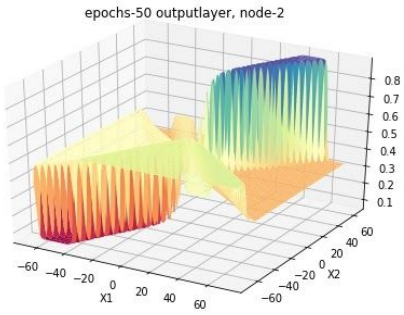

epochs-50 hiddenlayer-2, node-2
X1
X2

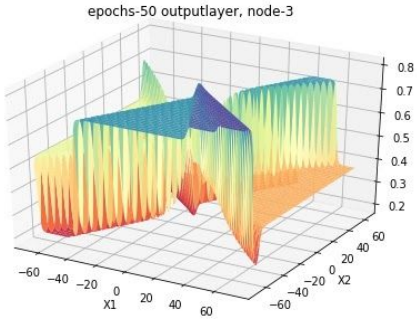




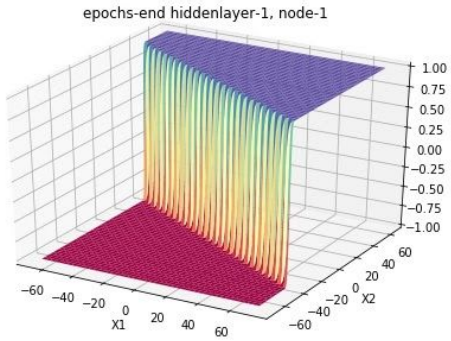


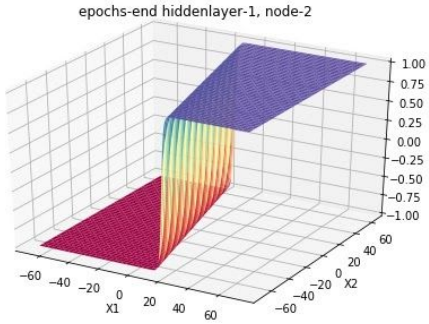


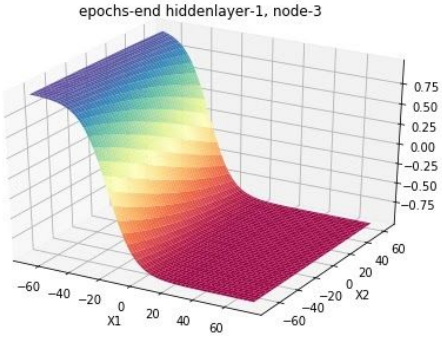


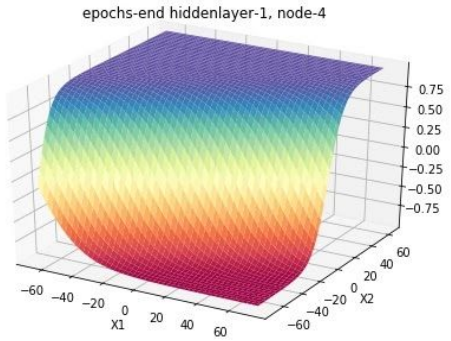


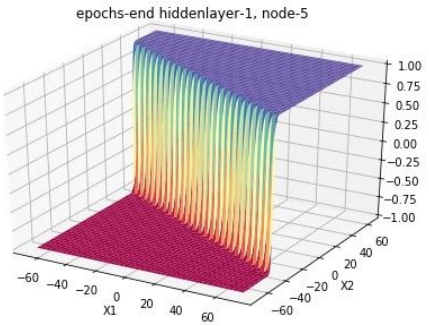
Termination

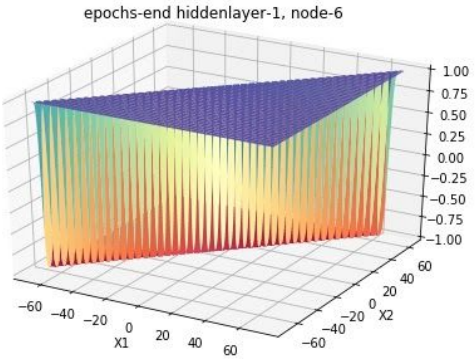


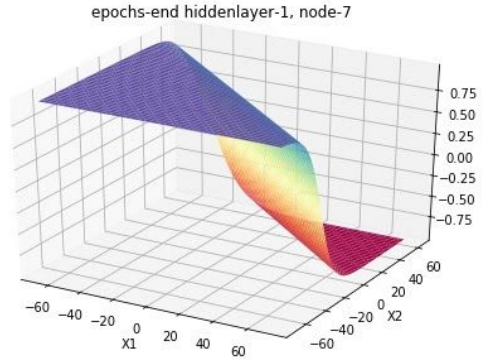


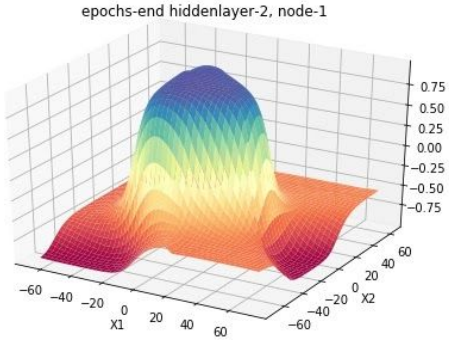


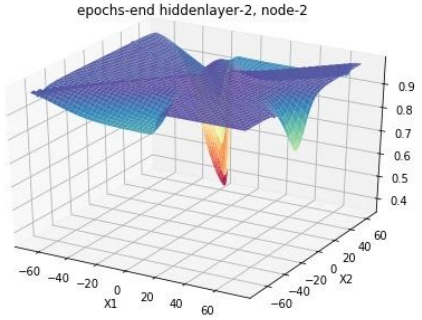


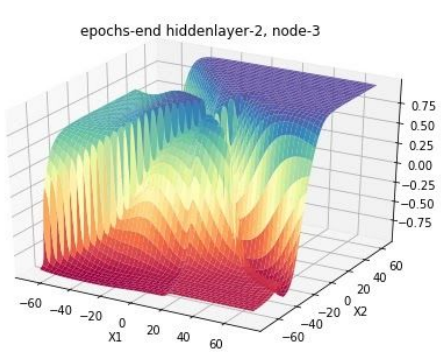


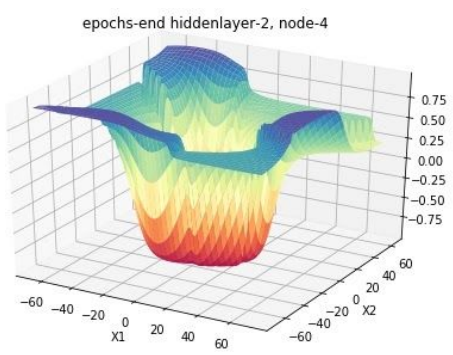


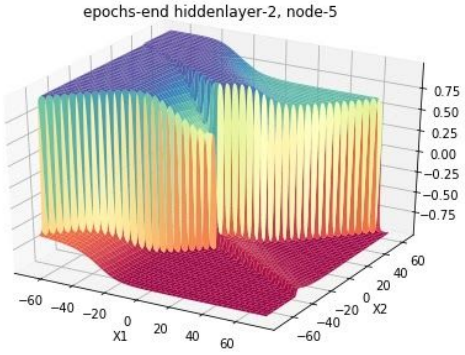


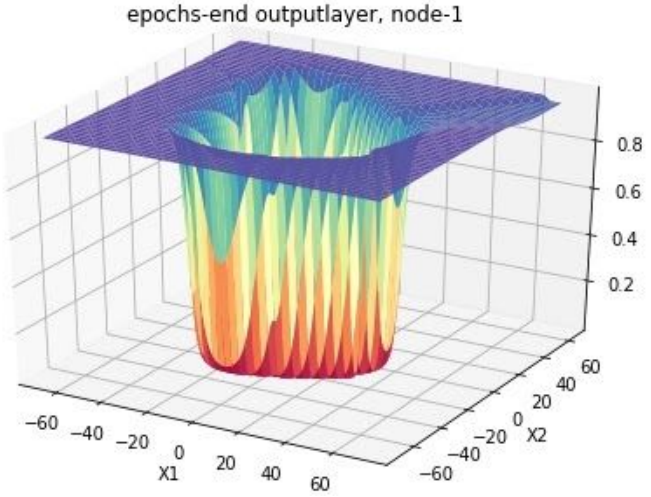


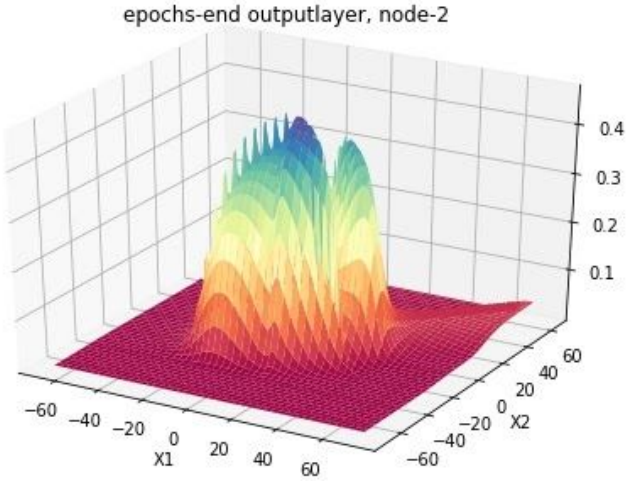


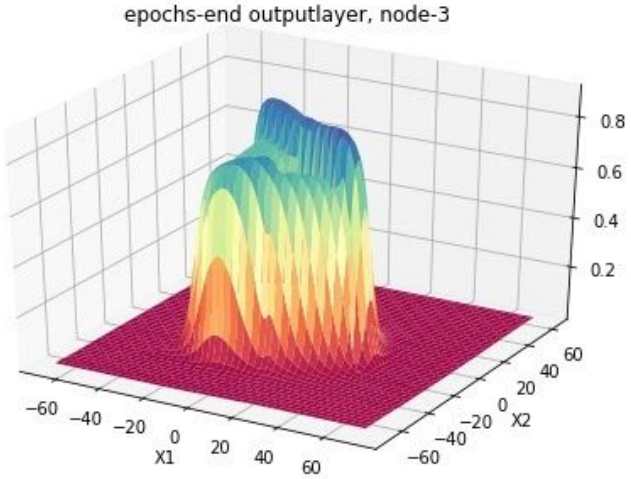

epochs-end hiddenlayer-2, node-3
X1
X2

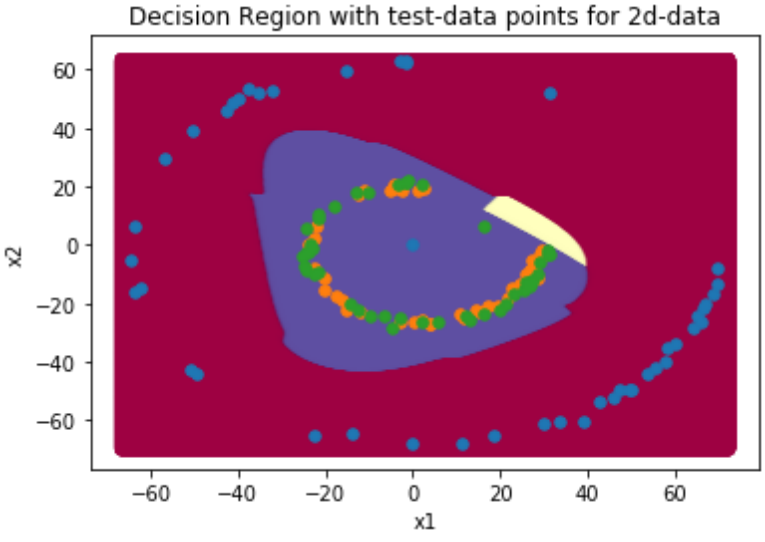

epochs-end hiddenlayer-2, node-4
X1
X2

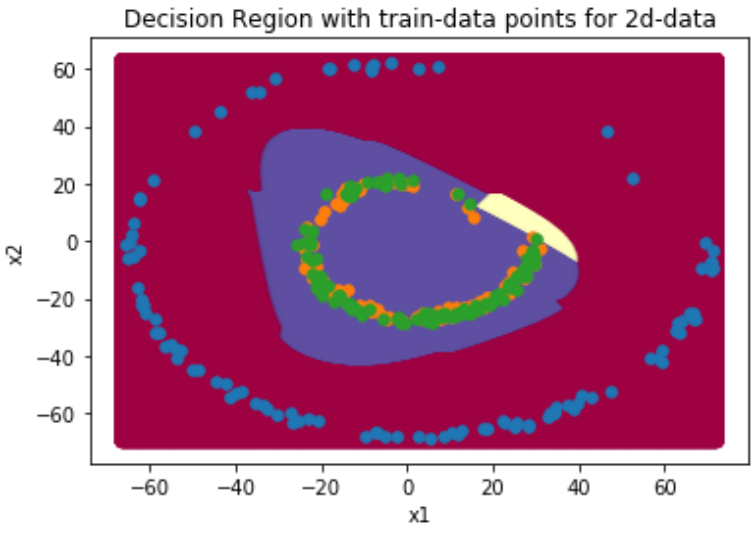

epochs-end hiddenlayer-2, node-5
X1
X2

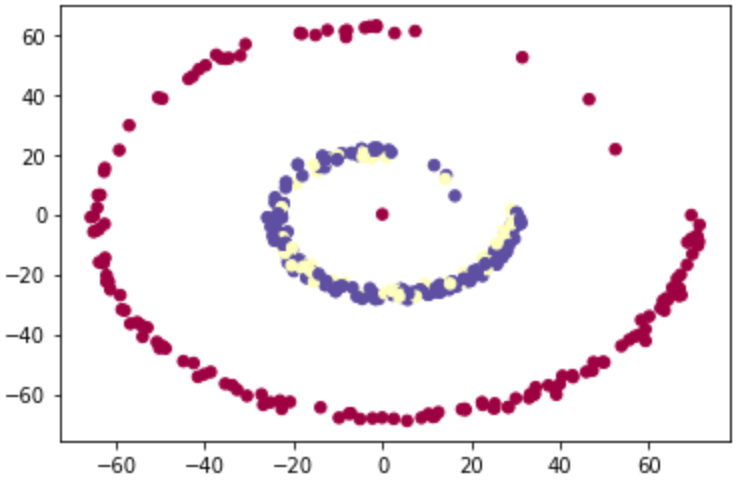

epochs-end outputlayer, node-1
X1
X2


epochs-end outputlayer, node-2
X1
X2


epochs-end outputlayer, node-3
X1
X2


Decision Region with test-data points for 2d-data
x1
x2


Decision Region with train-data points for 2d-data
x1
x2



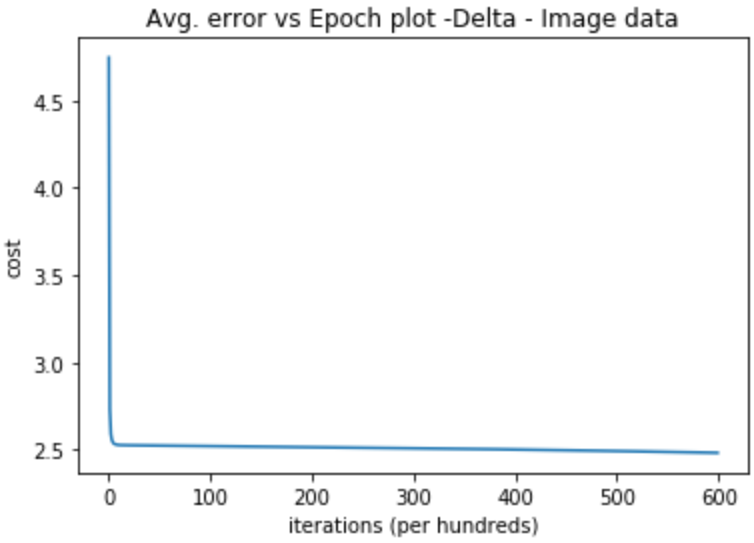
Inference :

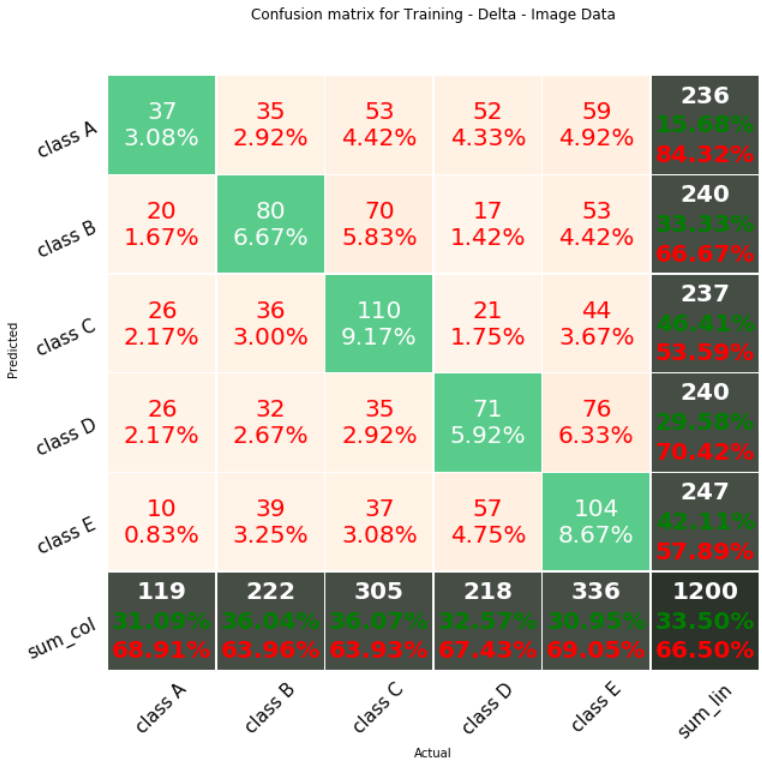
- Learning parameter = 0.01 and delta cost = 10^(-4)
- Nodes in first hidden layer = 7 and second hidden layer = 5
- In given data 2 classes blue and yellow require more complex boundaries to separate them. A deep network will classify them more accurately.
- The accuracy is 83% on training data and 80% on test data.
- On increasing the number of nodes, the curve is getting over-fitted.

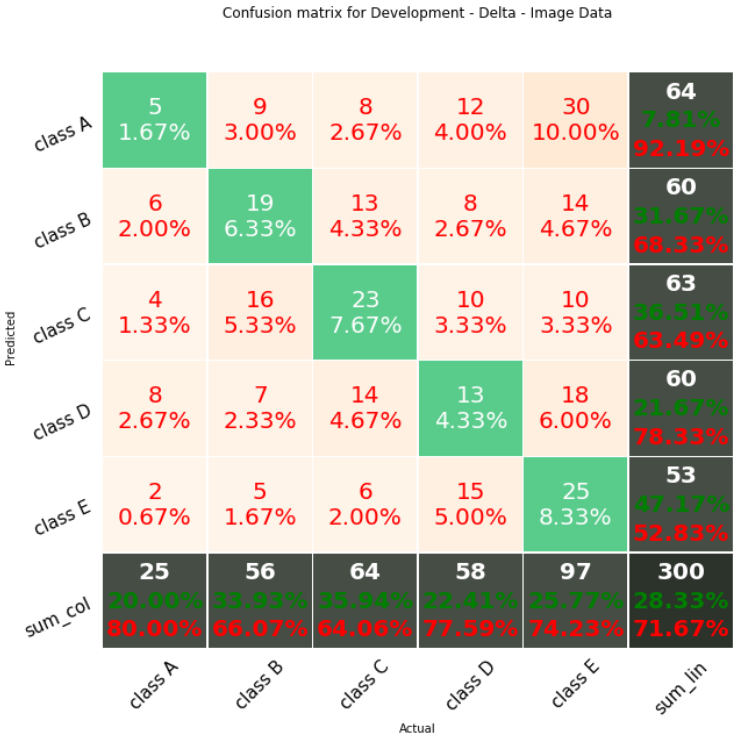
Classification task for image data

Delta :

Plots :



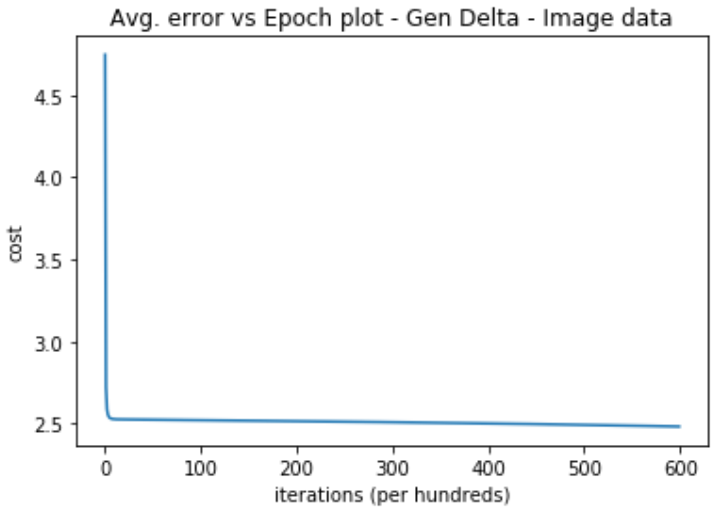




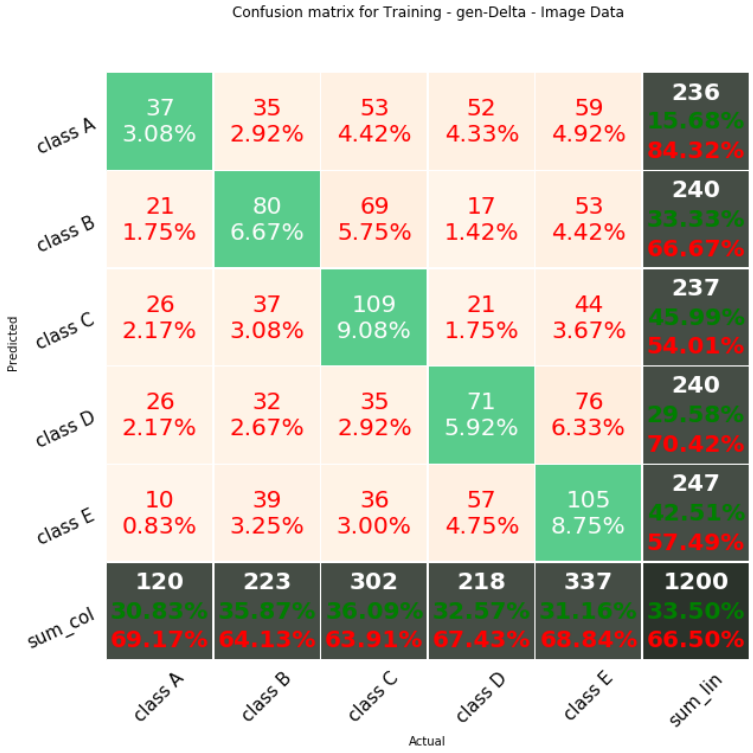
Inference :

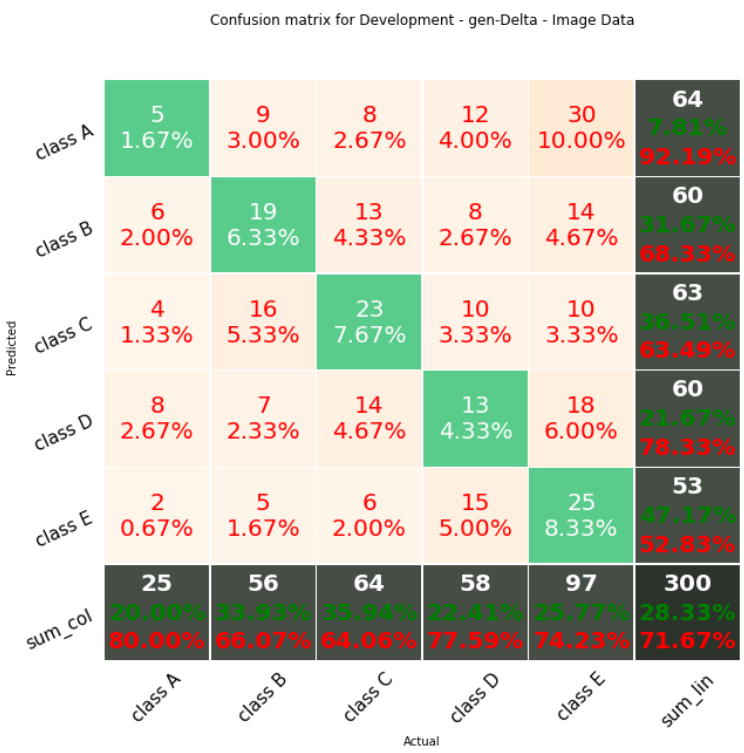
- Nodes in first hidden layer = 40, second hidden layer = 28
- Learning rate is 0.0005 and delta_cost = 10^(-8)
- Beta = 0.9 and gamma = 0.99
- On decreasing learning rate, we are ending up at minima which has greater avg cost.
- On increasing the learning rate, we are going up and down around the second minima.
- Here it's reaching local minima we can infer this from much lower cost obtained from adam.

Generalised Delta or classical momentum method :



Plots :



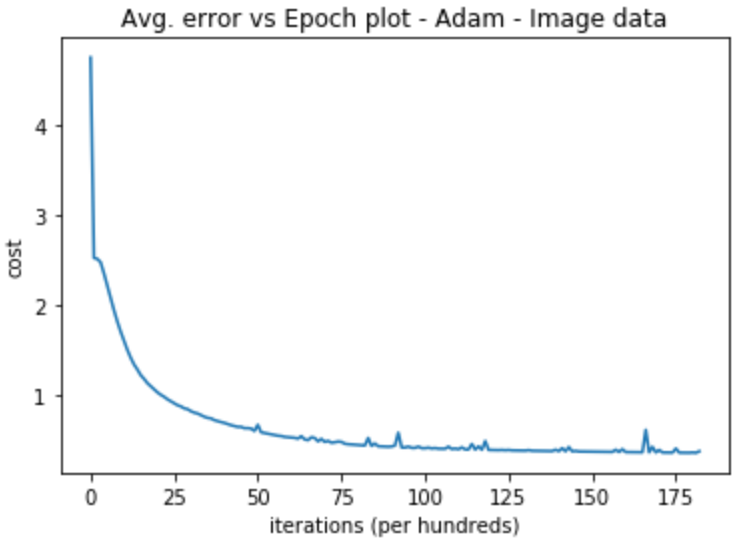


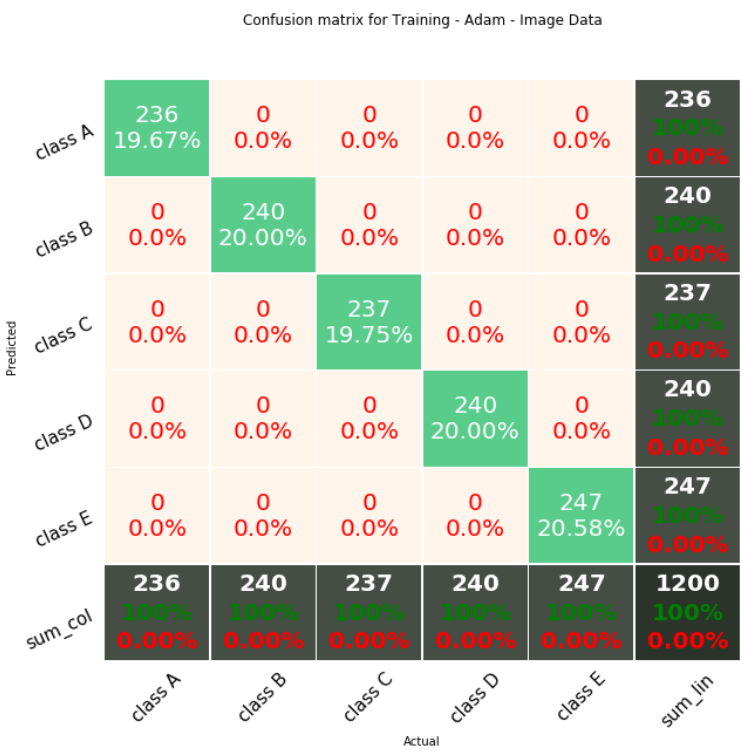
Inference :

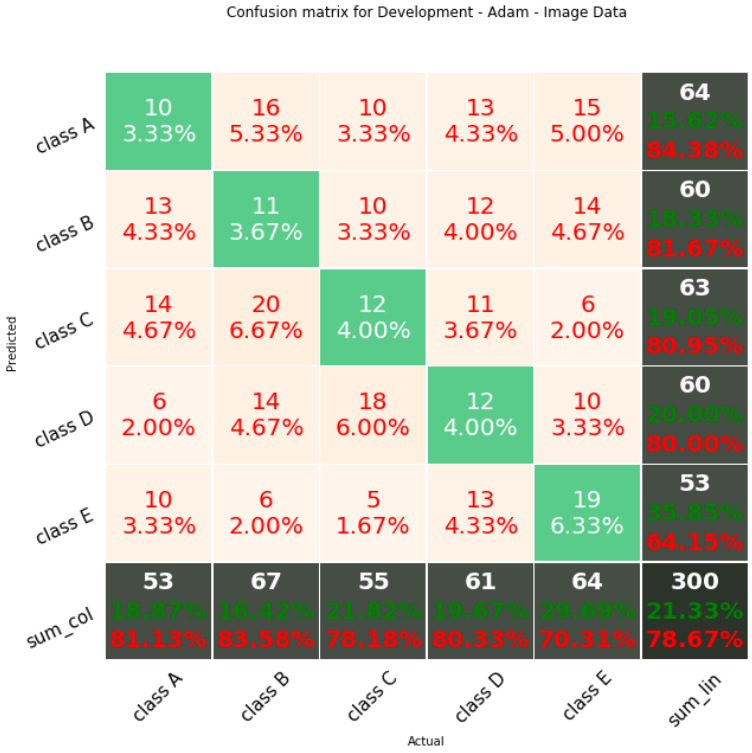
- Nodes in first hidden layer = 40, second hidden layer = 28
- Learning rate is 0.0005 and delta_cost = 10^(-8)
- Similar to above we are reaching a local minimum.

Adam :

Plots :







Inference :

- Nodes in first hidden layer = 40, second hidden layer = 28
- Learning rate is 0.0005 and delta_cost = 10^(-8)
- This algorithm achieves good results faster than the above ones.
- We can observe 100 percent accuracy on training data. This is the case of overfitting and that is why we are observing lower development results.
- We know that above algorithms converged at a local minimum of 2.5 referring to that we can observe a dip in the graph.