CS671: Deep Learning and Applications
Group - 26 Assignment#01

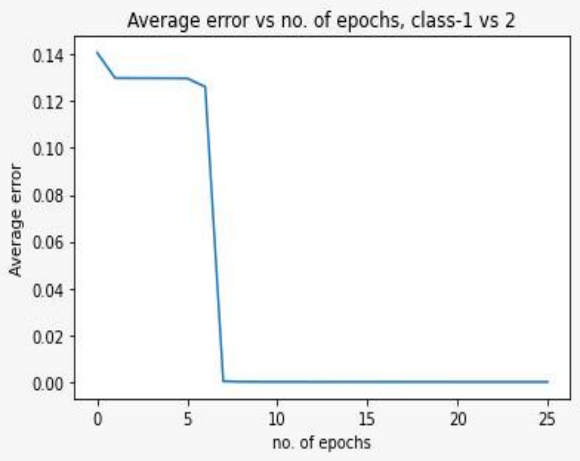
Task-1 classification:

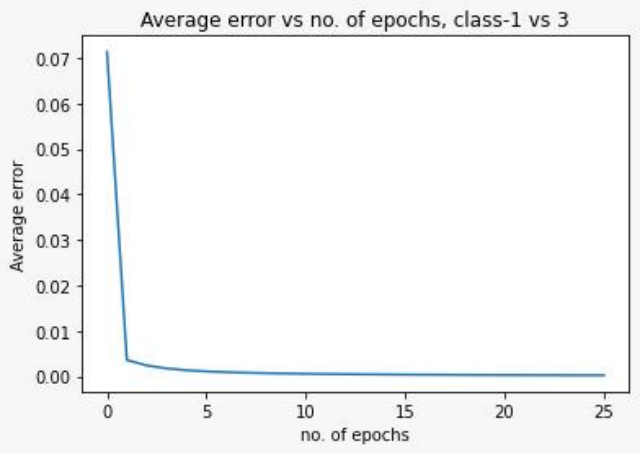
Model: perceptron using sigmoidal activation function

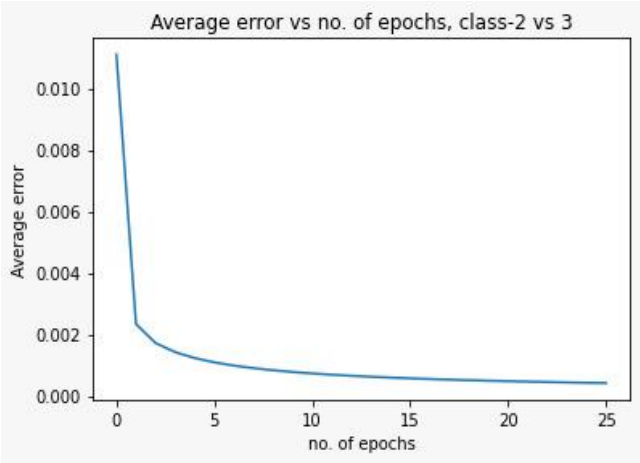
A. Linearly separable classes:

Firstly we split data into training and test sets in ratio 70:30. We are given a multiclass classification problem (3 classes), so we use the one-against-one approach for classification. We take two classes at once, train the model and obtain the parameters, thus getting 3 sets of parameters. For testing we pass test sample through each model and decide final label on the basis of majority of label it gets.

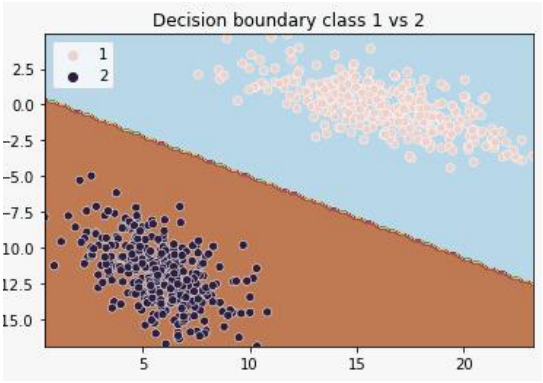
Plot of epochs vs average error: At low values of epoch the average error is higher than the average error at the larger values of epochs, as expected as the learning algorithm updates the internal model parameters in order to reduce the error.

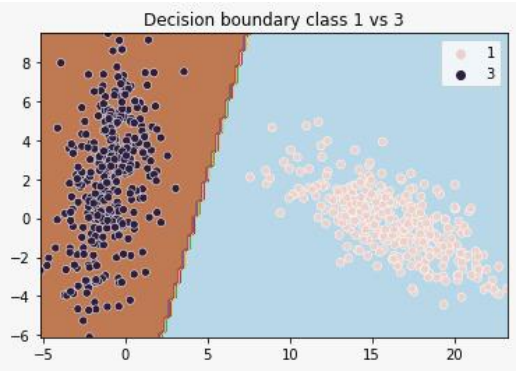


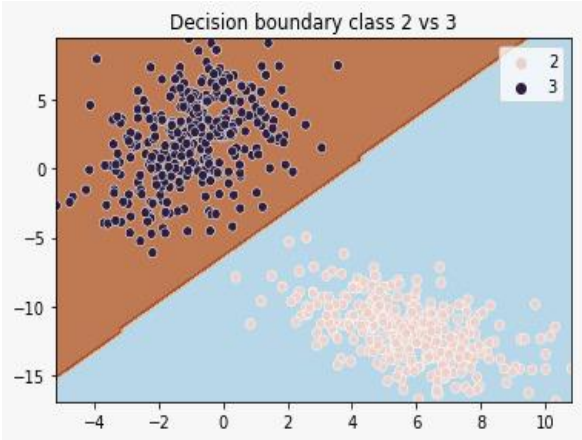


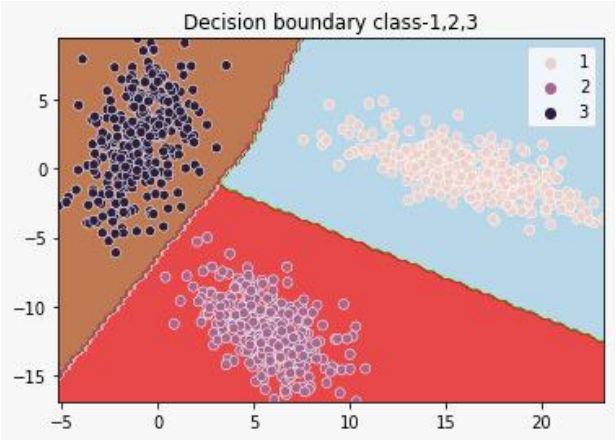


<u>Decision Region plot:</u> A decision boundary between each pair of classes as we do in one against one approach, then we plot decision boundary for all classes









Training dataset:

Accuracy : 1.0

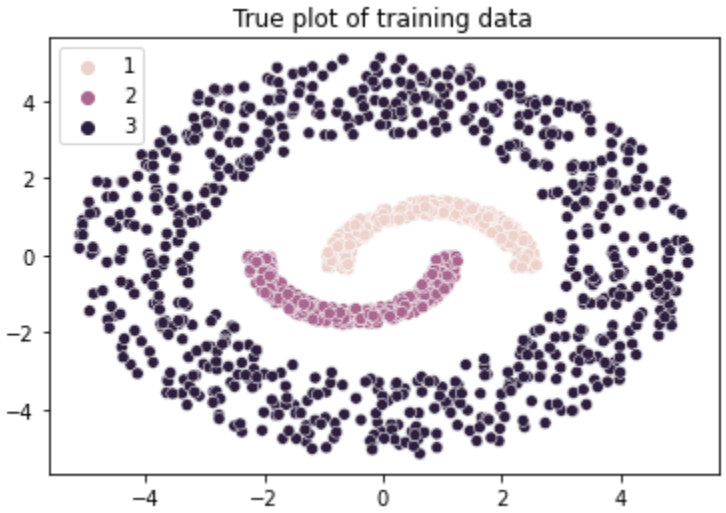
Confusion matrix : [[332 0 0]
[0 359 0]
[0 0 359]]

Testing dataset:

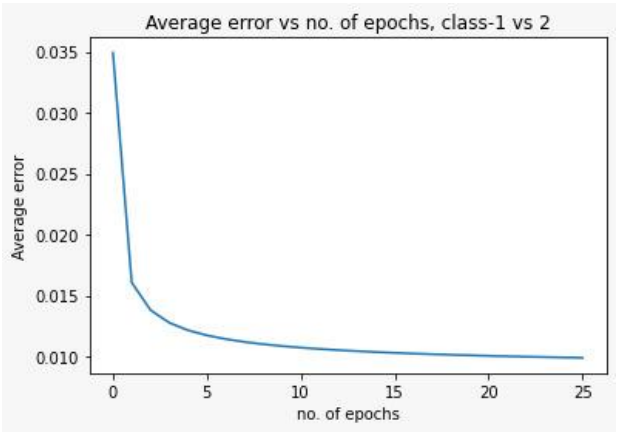
Accuracy: 1.0

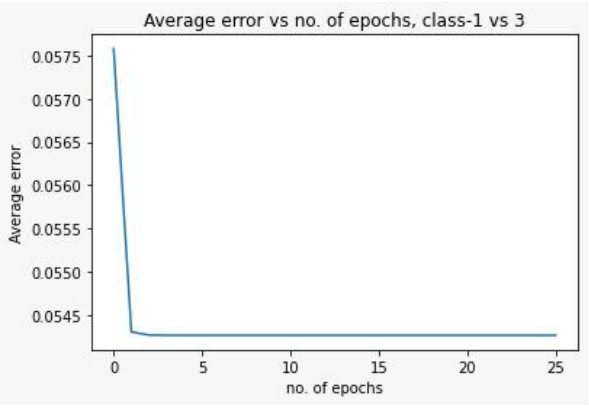
Confusion matrix: [[168 0 0]
[0 141 0]
[0 0 141]]

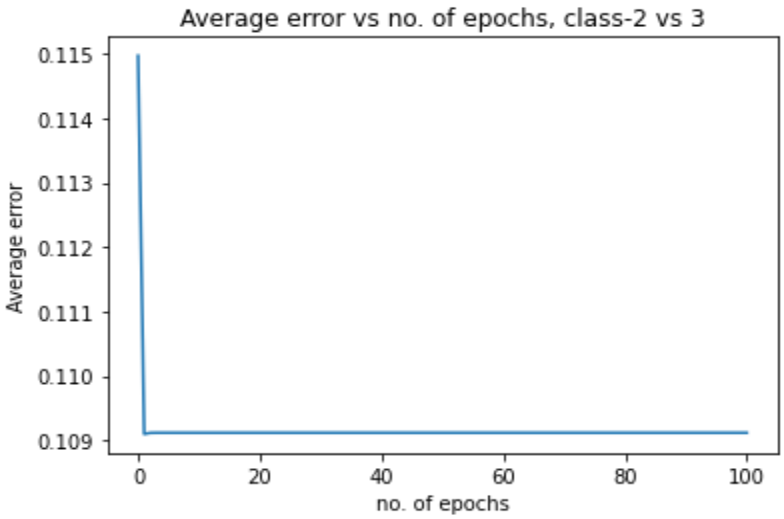
B. Non-Linearly separable classes:



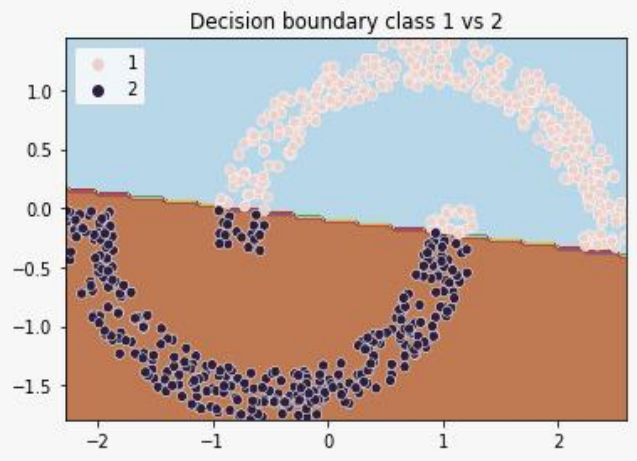
Plot of epochs vs average error:

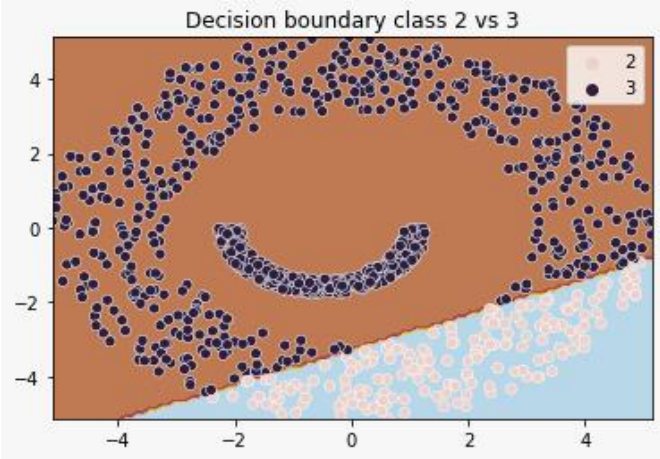


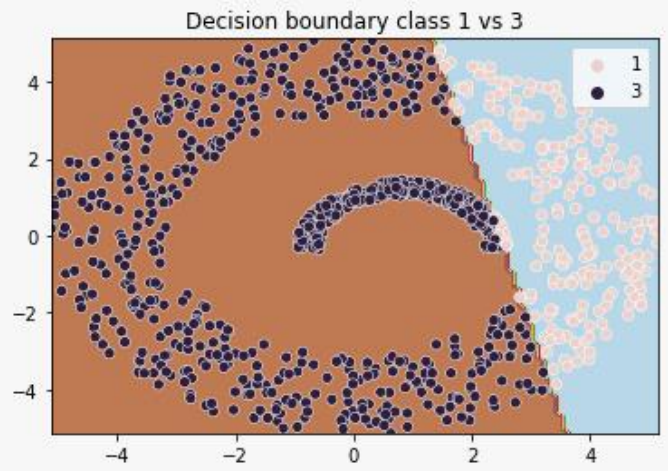


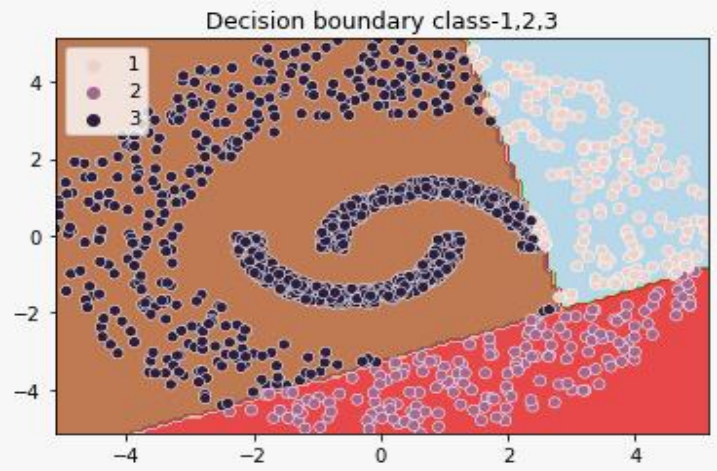


Decision Region plot:









For training dataset:

Accuracy: 0.2664285714285714

Confusion matrix: [[13 0 165]
[0 0 177]
[334 351 360]]

For testing dataset:

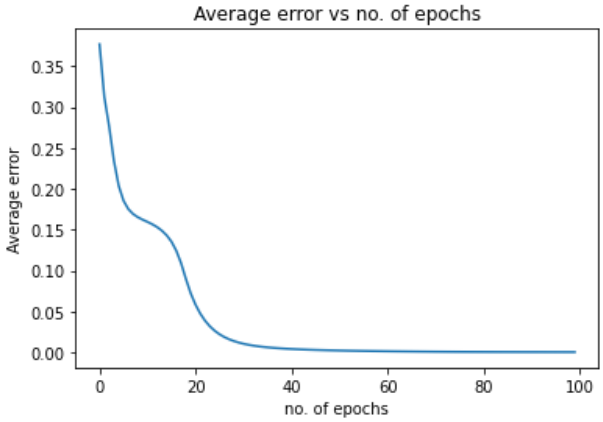
Accuracy: 0.2683333333333333

Confusion matrix: [[5 0 72]
[0 0 70]
[148 149 156]]

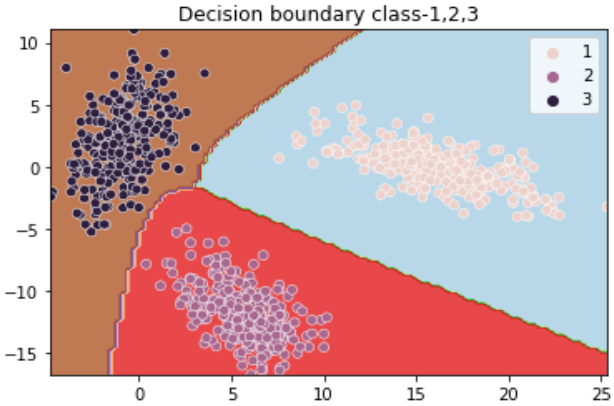
Model: FCNN using stochastic gradient descent

A. Linearly separable classes:

Plot of epochs vs average error(Best architecture):



Decision Region plot(Best architecture):



Results of different architecture:

No of neurons in each layer	Accuracy	Confusion matrix
[2,4,3,3]	1.0	[[103,0,0] [0,92,0] [0,0,105]]
[2,4,5,3]	1.0	[[103,0,0] [0,92,0] [0,0,105]]
[2,4,7,3]	1.0	[[103,0,0] [0,92,0] [0,0,105]]
[2,6,3,3]	1.0	[[103,0,0] [0,92,0] [0,0,105]]
[2,6,5,3]	1.0	[[103,0,0] [0,92,0] [0,0,105]]
[2,6,7,3]	1.0	[[103,0,0] [0,92,0] [0,0,105]]
[2,8,3,3]	1.0	[[103,0,0] [0,92,0] [0,0,105]]
[2,8,5,3]	1.0	[[103,0,0] [0,92,0] [0,0,105]]
[2,8,7,3]	1.0	[[103,0,0] [0,92,0] [0,0,105]]

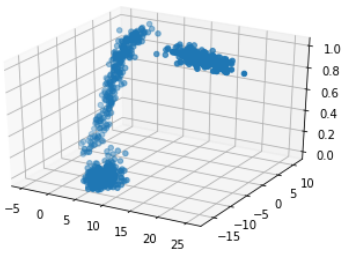
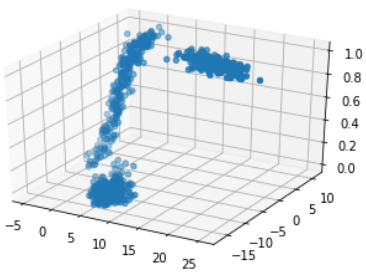
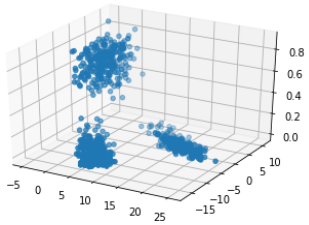
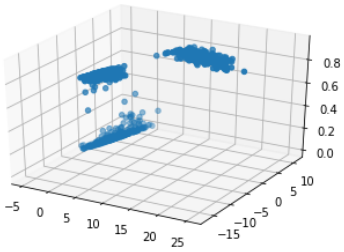
Best architecture in terms of parameters, we choose simplest as data is linearly separable:

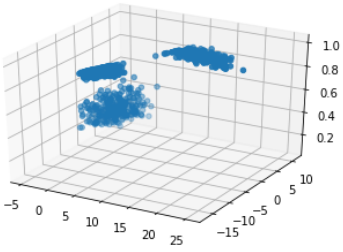
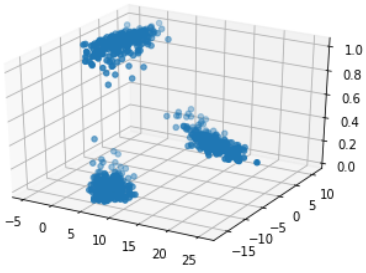
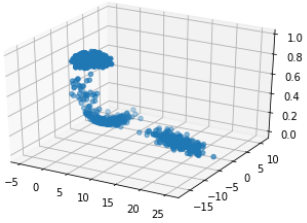
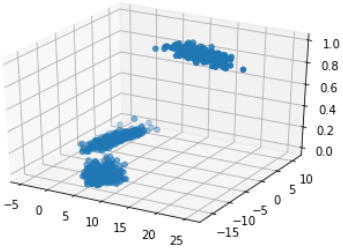
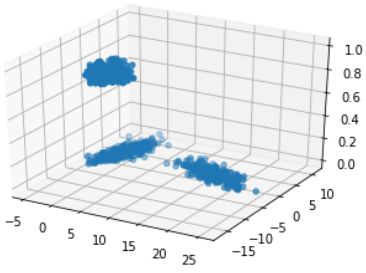
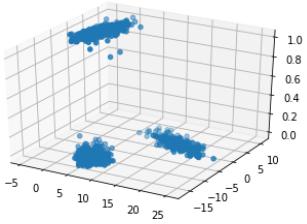
No of neurons in each layer-[2,4,3,3]

	Accuracy	Confusion matrix
Training	1.0	[[293,0,0] [0,322,0] [0,0,285]]
Validation	1.0	[[103,0,0] [0,92,0] [0,0,105]]
test	1.0	[[104,0,0] [0,86,0] [0,0,110]]

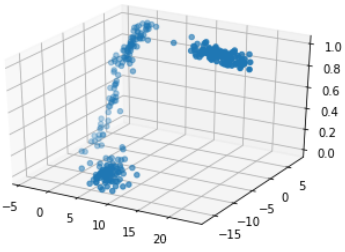
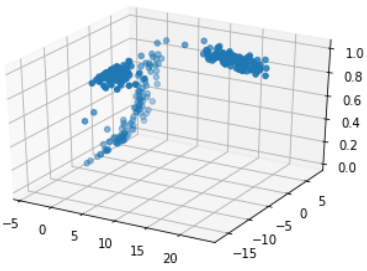
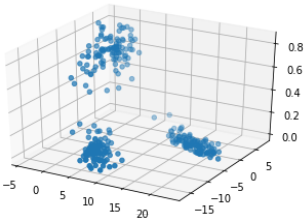
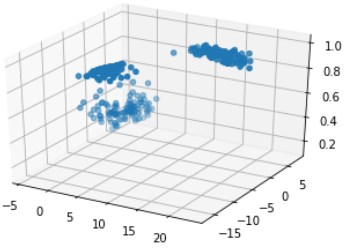
Output of each node in neural network:

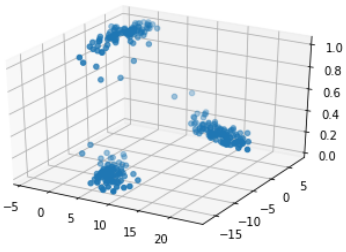
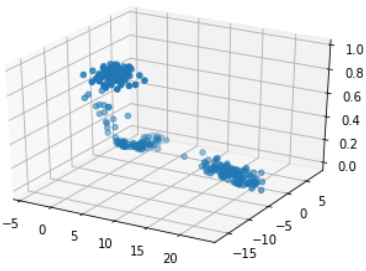
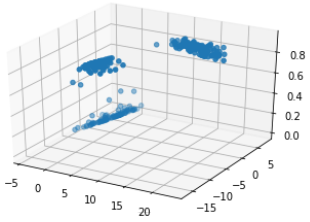
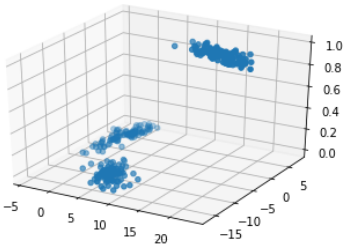
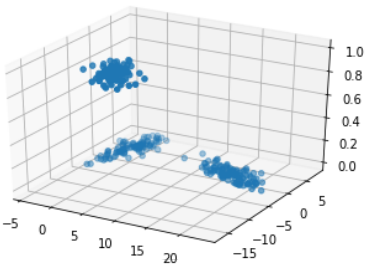
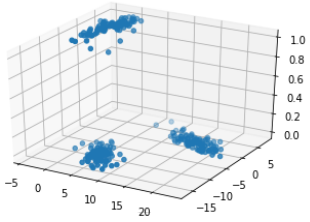
<u>On Training data</u>

	Nodes in sequence		
Hidden layer 1			
			

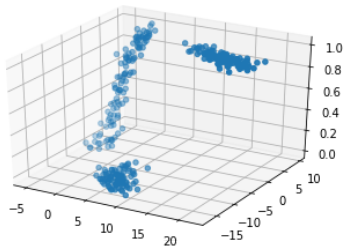
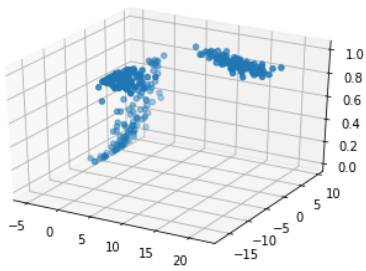
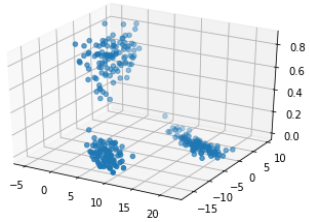
Hidden layer 2			
Output layer			

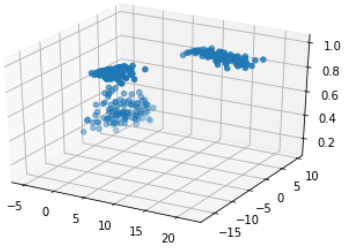
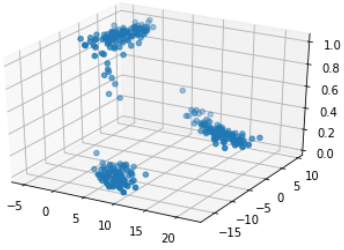
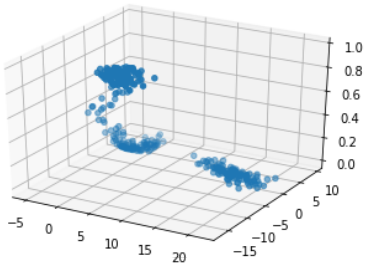
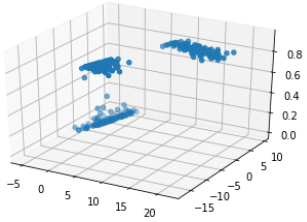
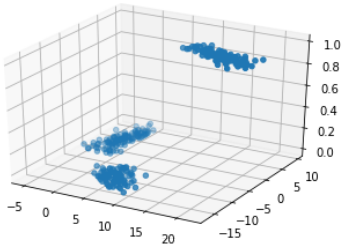
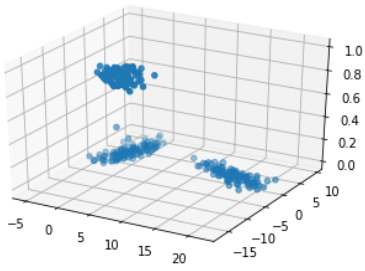
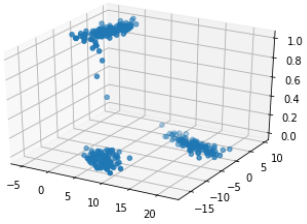
On Validation data

	Nodes in sequence		
Hidden layer 1			
			

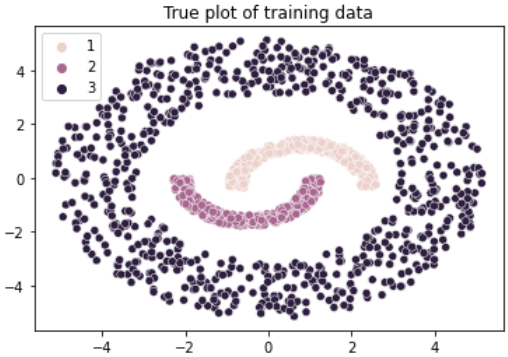
Hidden layer 2			
Output layer			

<u>On test data</u>

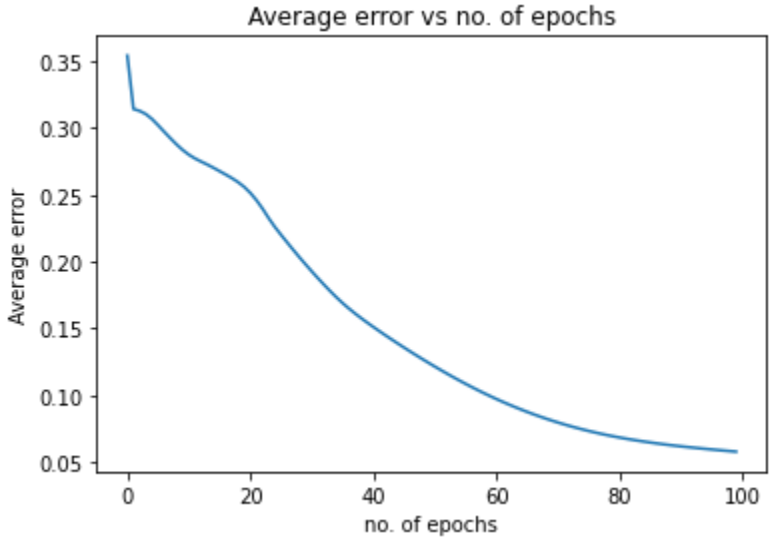
	Nodes in sequence		
Hidden layer 1			

			
Hidden layer 2			
Output layer			

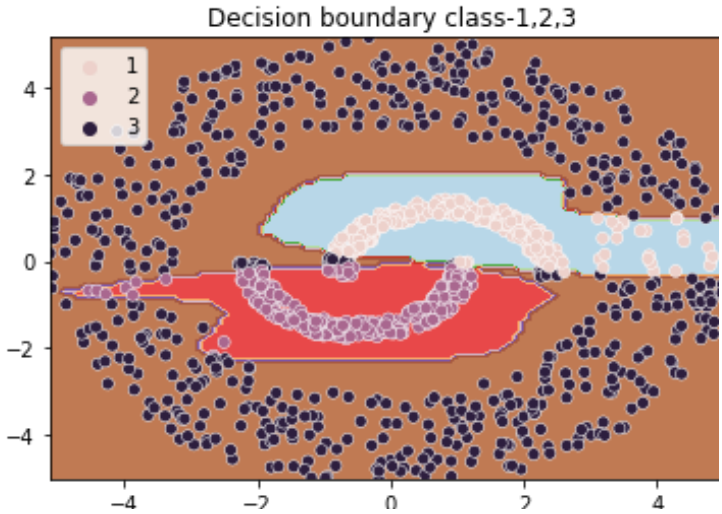
B. Non-Linearly separable classes:



Plot of epochs vs average error(Best architecture):



Decision Region plot(Best architecture):



Results of different architecture:

No of neurons in each layer	Accuracy	Confusion matrix
[2,4,3,3]	0.625	[[91,0,10] [3,0,107] [19,0,170]]
[2,4,5,3]	0.8875	[[93,6,2] [6,103,1] [20,10,159]]
[2,4,7,3]	.0.8875	[[87,6,8] [0,103,7]

		[14,10,165]]
[2,6,3,3]	0.9	[[85,6,10] [0,99,11] [8,5,176]]
[2,6,5,3]	0.855	[[92,7,2] [3,92,15] [17,14,158]]
[2,6,7,3]	0.93	[[91,6,4] [,102,7] [7,3,179]]
[2,8,3,3]	0.8975	[[90,6,5] [0,103,7] [7,16,166]]
[2,8,5,3]	0.93	[[86,6,9] [0,105,5] [6,2,181]]
[2,8,7,3]	0.9175	[[93,8,0] [1,109,0] [8,16,165]]

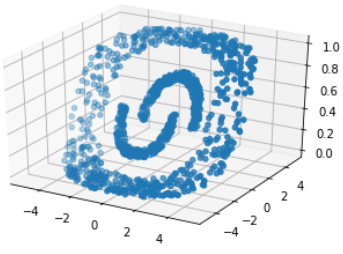
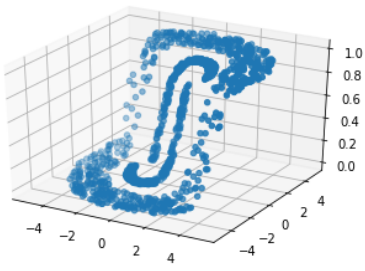
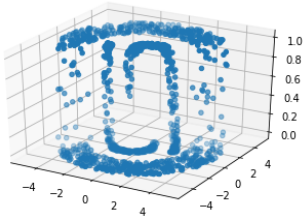
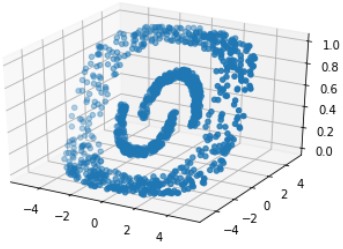
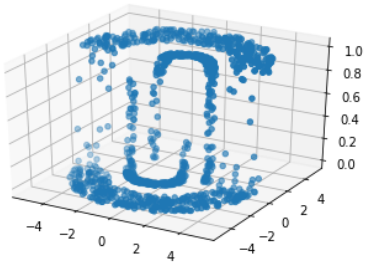
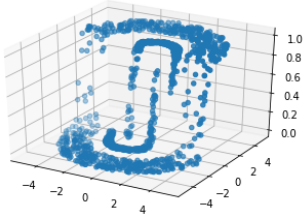
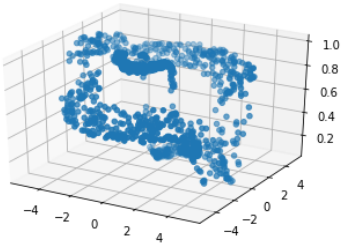
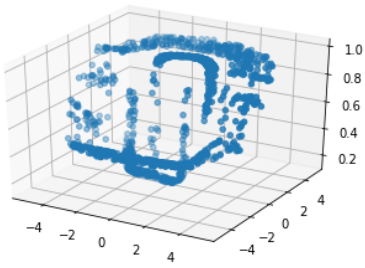
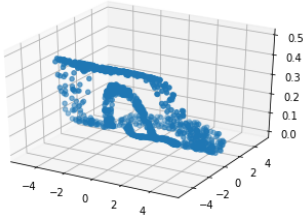
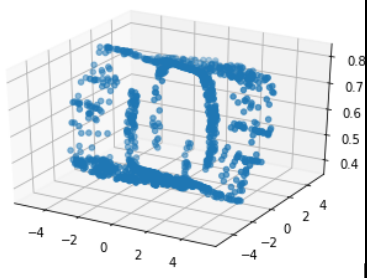
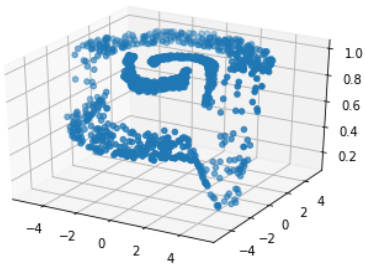
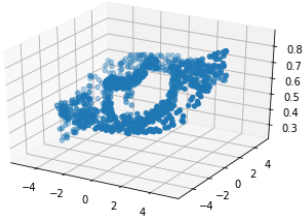
Best architecture in terms of parameters, we choose simplest as data is linearly separable:

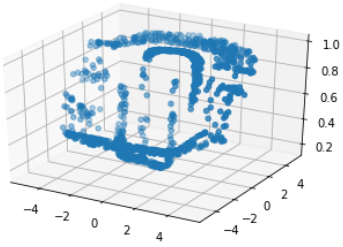
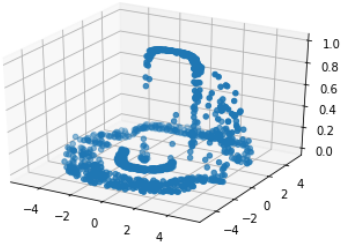
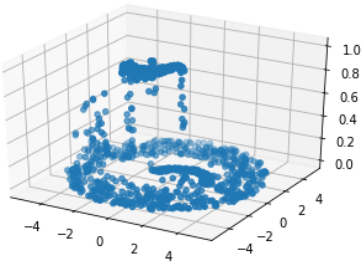
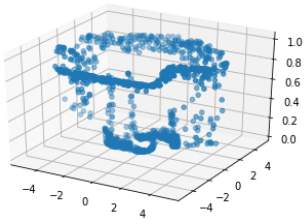
No of neurons in each layer-[2,6,7,3]

	Accuracy	Confusion matrix
Training	0.93	[[293,0,0] [0,322,0] [0,0,285]]
Validation	0.93	[[91,6,4] [1,102,7] [7,3,179]]
Test	0.9175	[[89,6,9] [3,85,7] [4,4,193]]

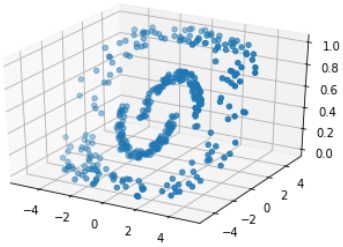
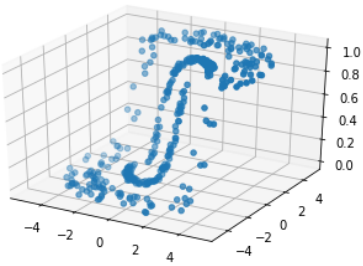
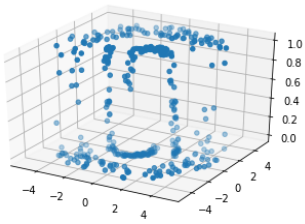
Output of each node in neural network:

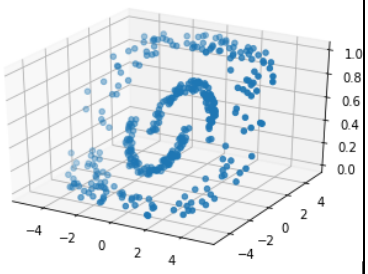
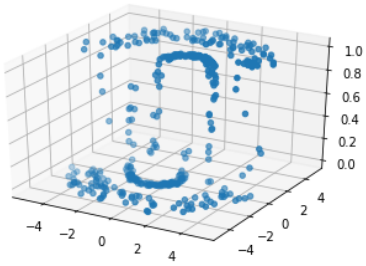
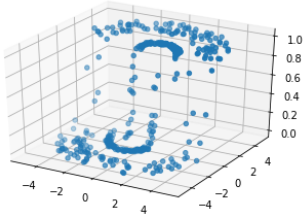
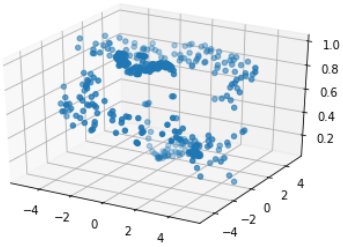
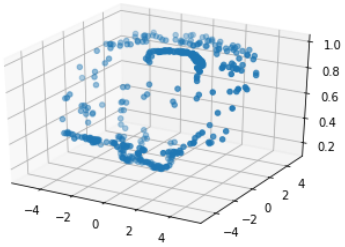
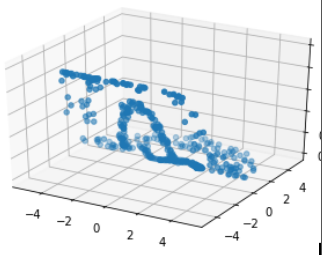
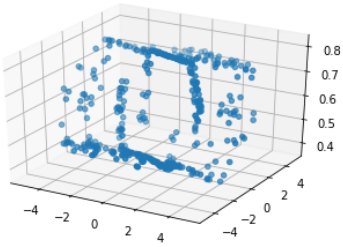
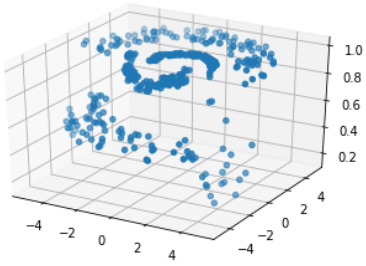
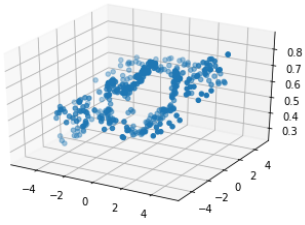
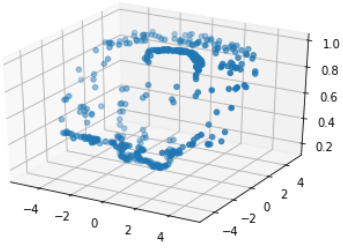
<u>On Training data</u>

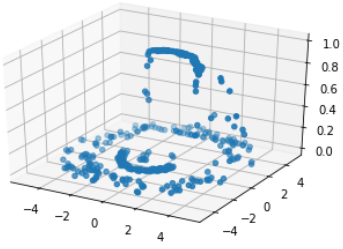
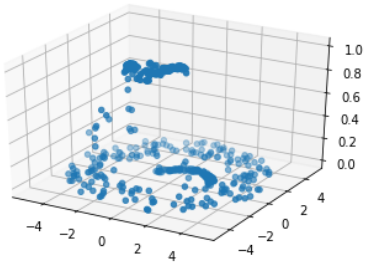
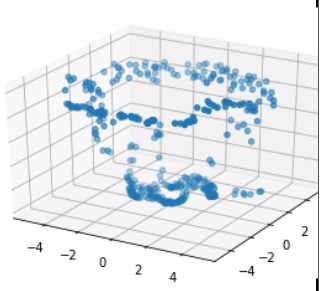
	Nodes in sequence		
Hidden layer 1			
			
Hidden layer 2			
			

			
Output layer			

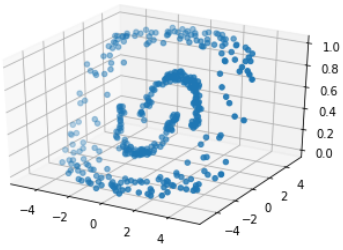
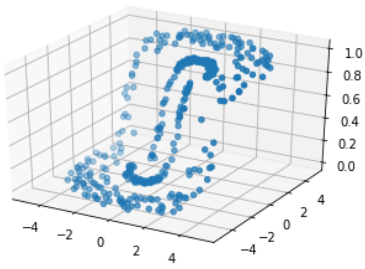
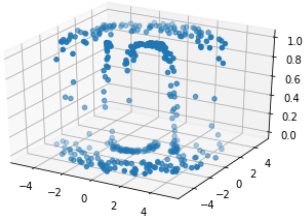
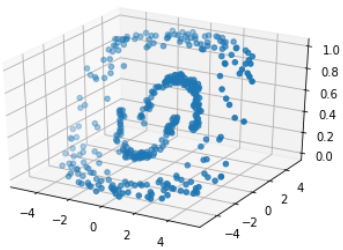
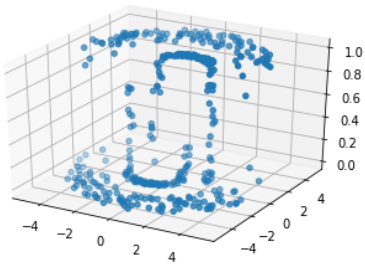
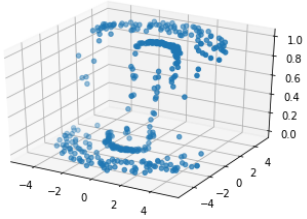
<u>On Validation data</u>

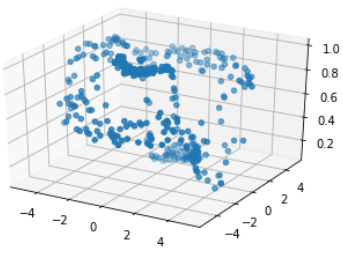
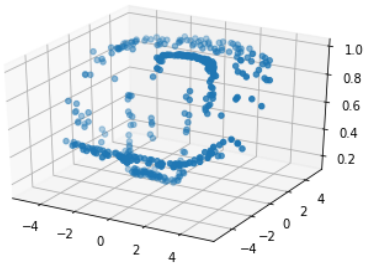
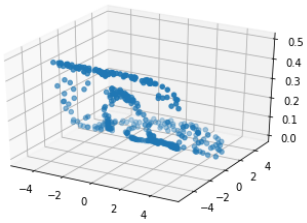
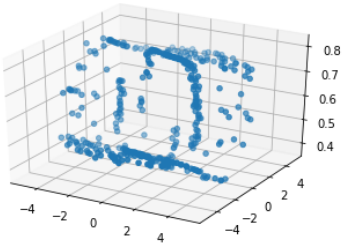
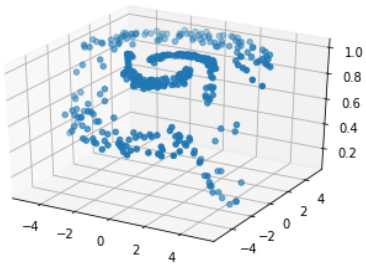
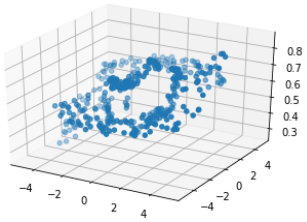
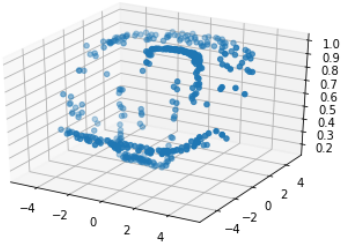
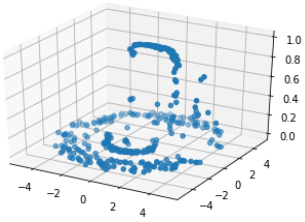
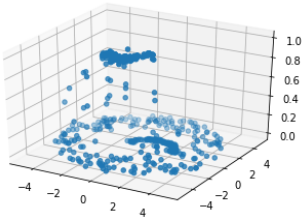
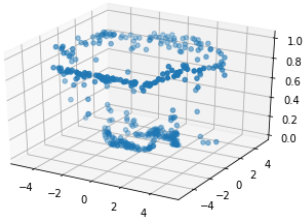
	Nodes in sequence		
Hidden layer 1			

			
Hidden layer 2			
			
			

Output layer			

<u>On test data</u>

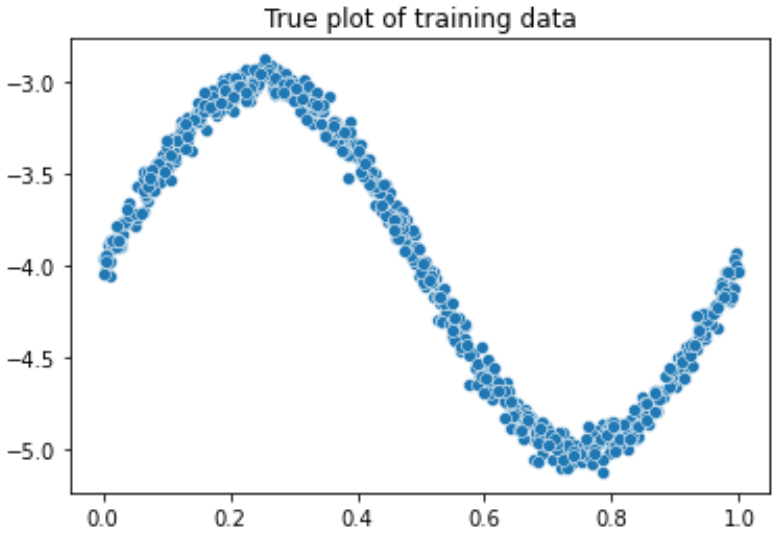
	Nodes in sequence		
Hidden layer 1			
			

Hidden layer 2			
			
			
Output layer			

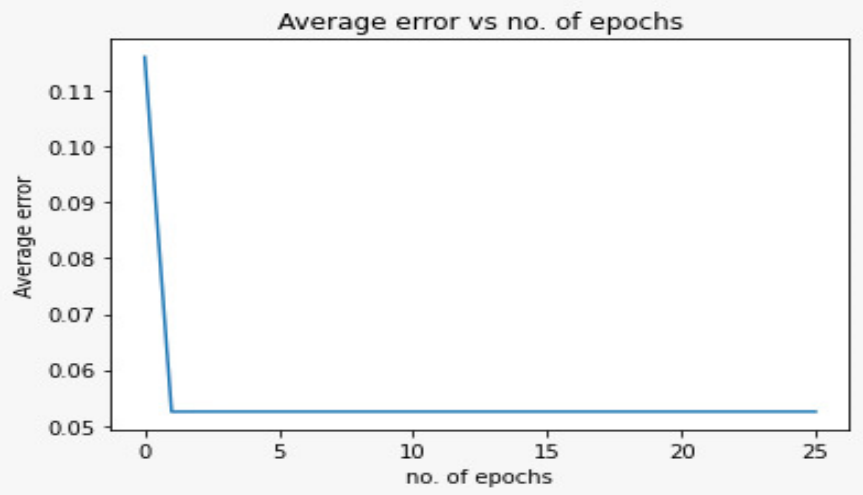
Task-2 Regression:

Model: perceptron using sigmoidal activation function

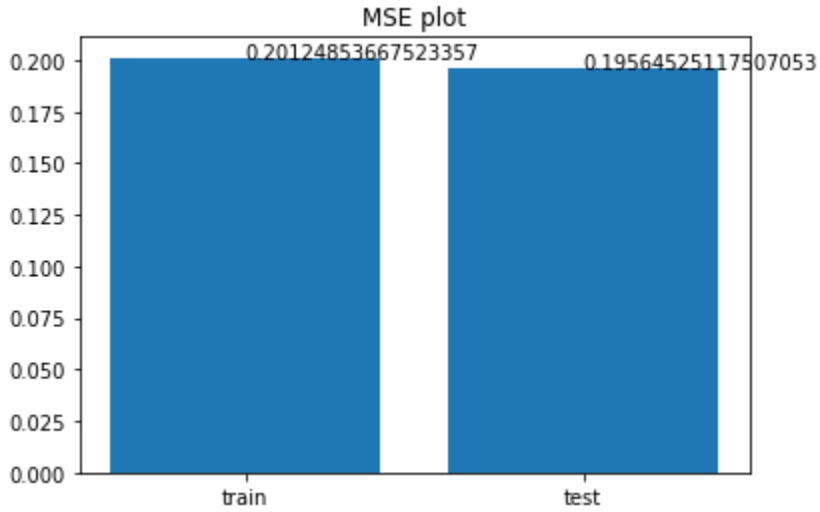
Dataset1: 1-dimensional (Univariate) input data:



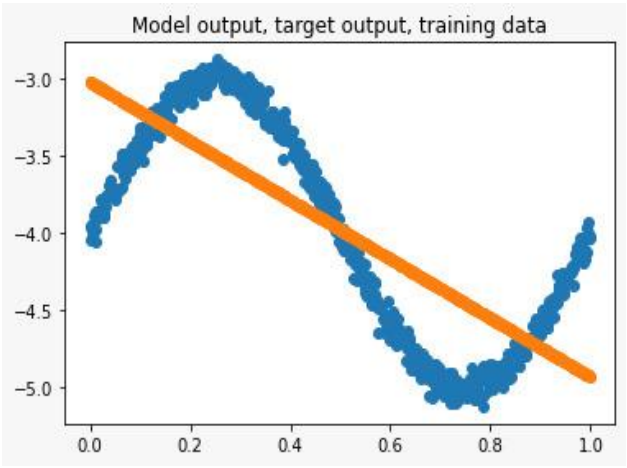
1. **Average error vs no. of epochs:**

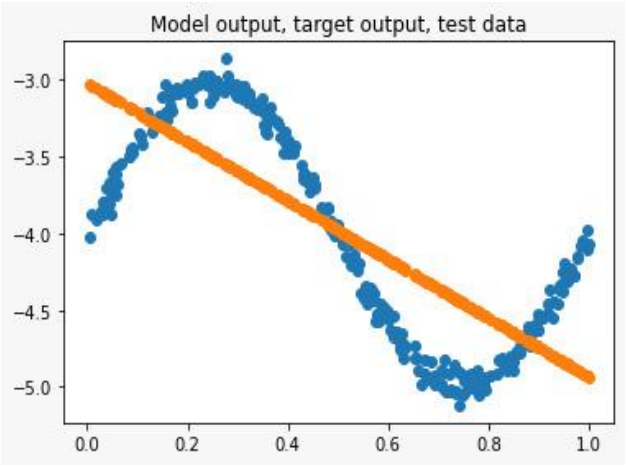


2. **Mean squared error:**

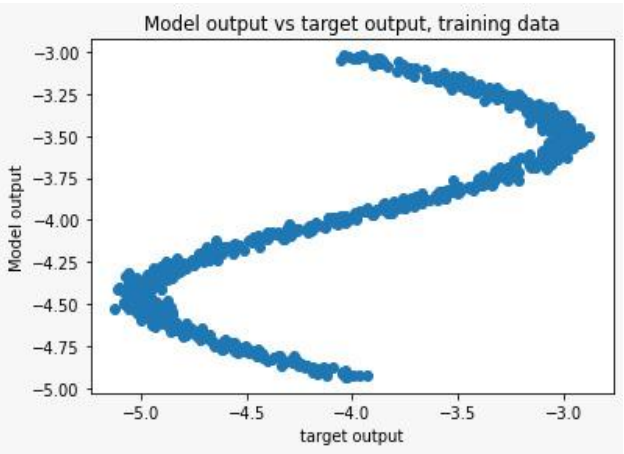


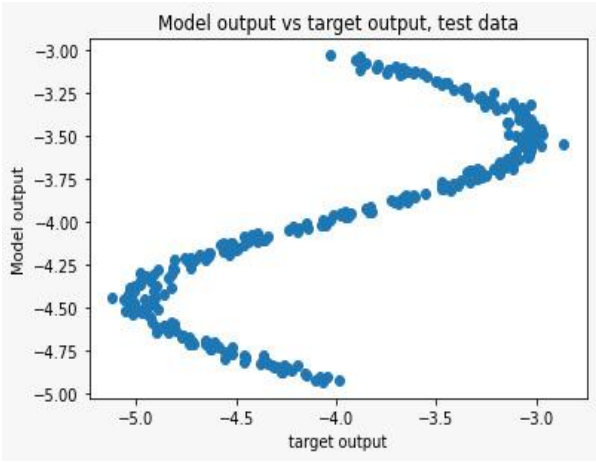
3. **Model output and target output for training and testing data:**



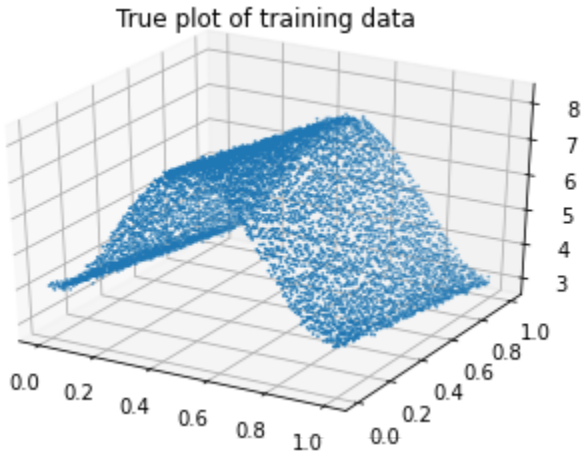


4. **Target output on x-axis and model output on y-axis:**

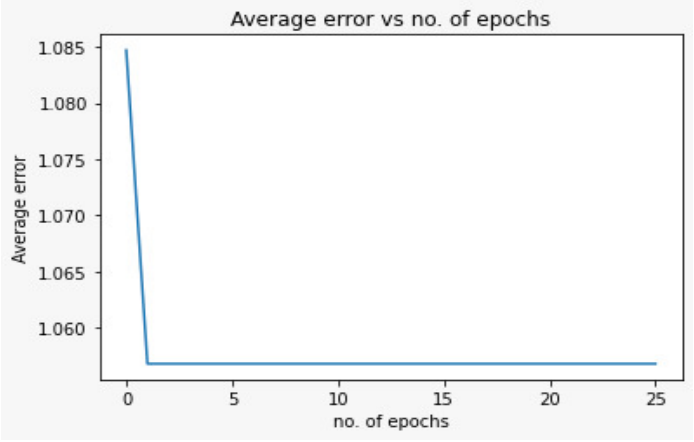




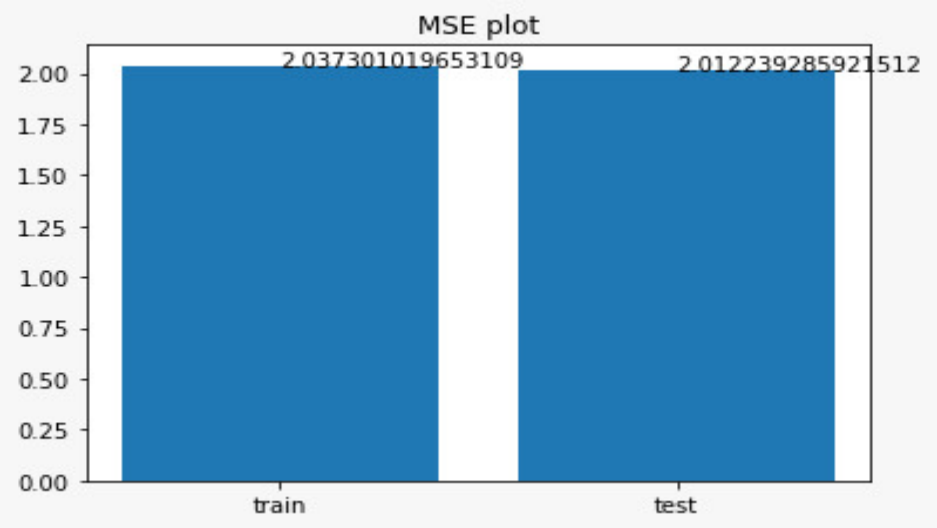
Dataset2: 2-dimensional (Bivariate) input data:



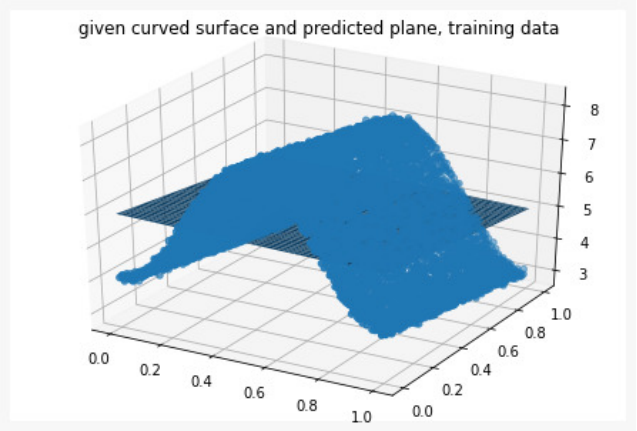
1. **Average error vs no. of epochs:**

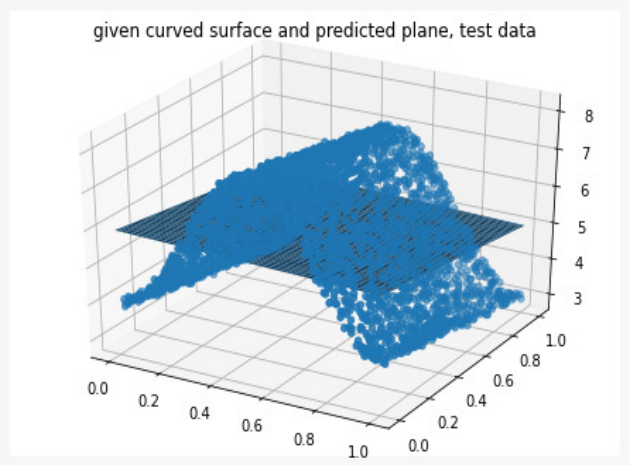


2. **Mean squared error:**

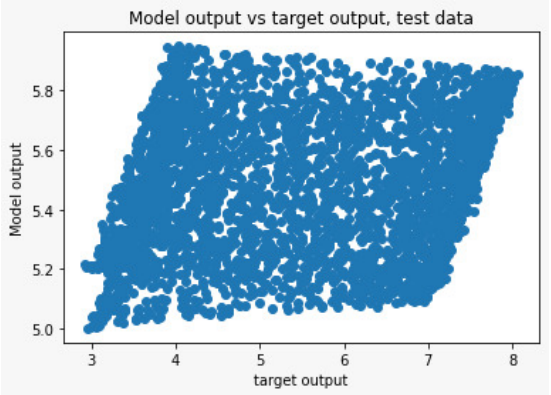


3. **Model output and target output for training and testing data:**



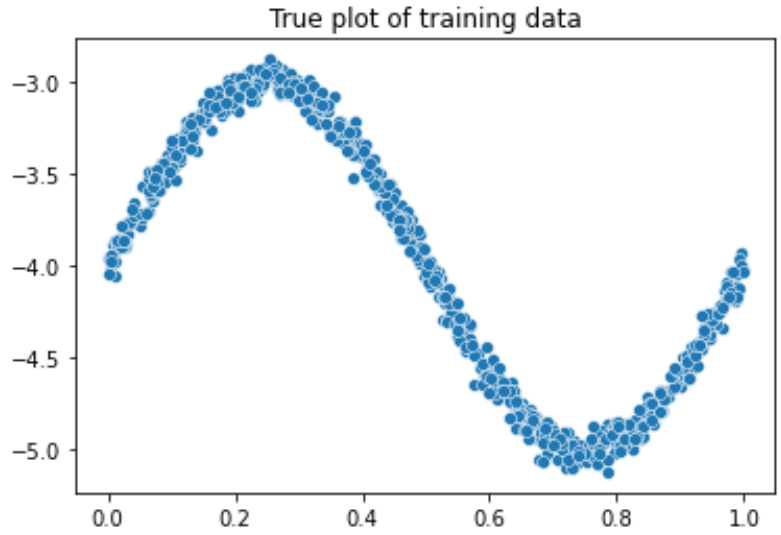


4. **Target output on x-axis and model output on y-axis:**

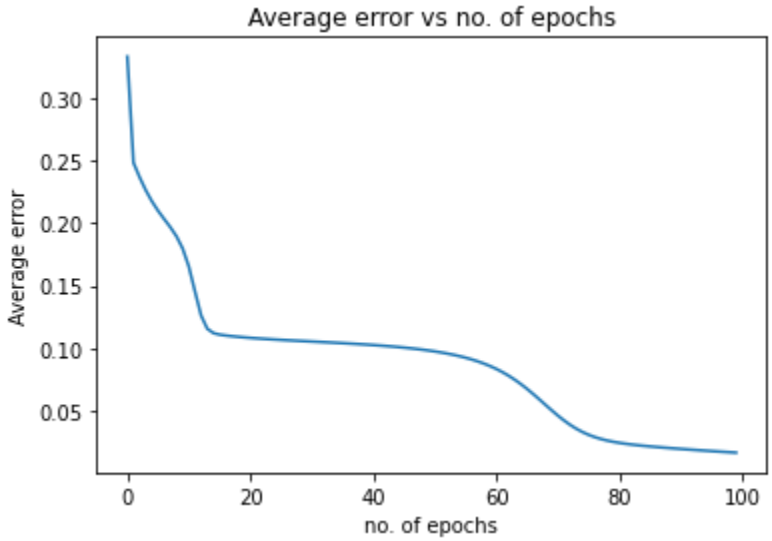


Model: FCNN using stochastic gradient descent

Dataset1: 1-dimensional (Univariate) input data:



Plot of epochs vs average error(Best architecture):



Results of different architecture:

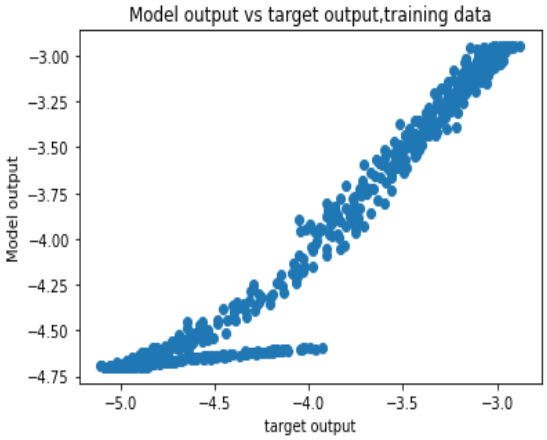
No of neurons in each layer	Root mean square Error
[1,4,3,1]	0.446
[1,4,5,1]	0.334
[1,4,7,1]	.0.448
[1,6,3,1]	0.273
[1,6,5,1]	0.238

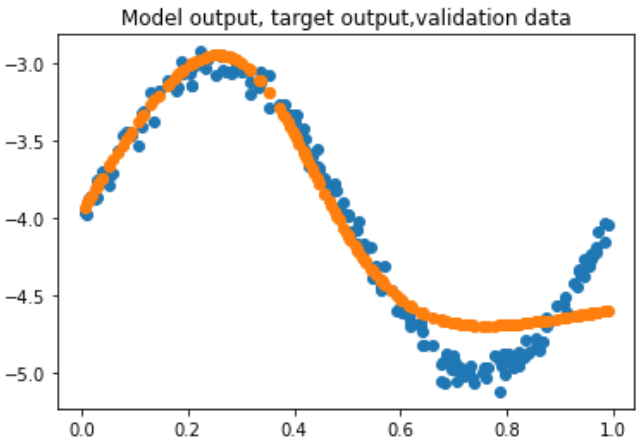
[1,6,7,1]	0.300
[1,8,3,1]	0.237
[1,8,5,1]	0.194
[1,8,7,1]	0.186

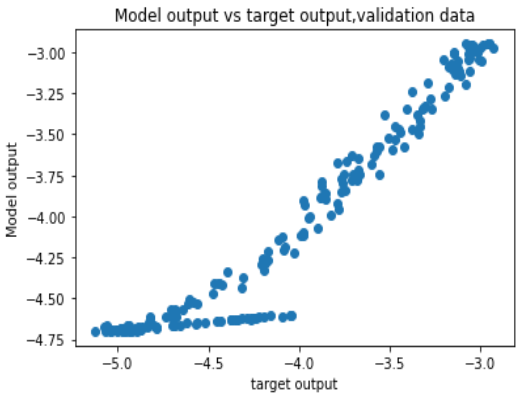
Best architecture in terms of parameters, we choose simplest as data is linearly separable:

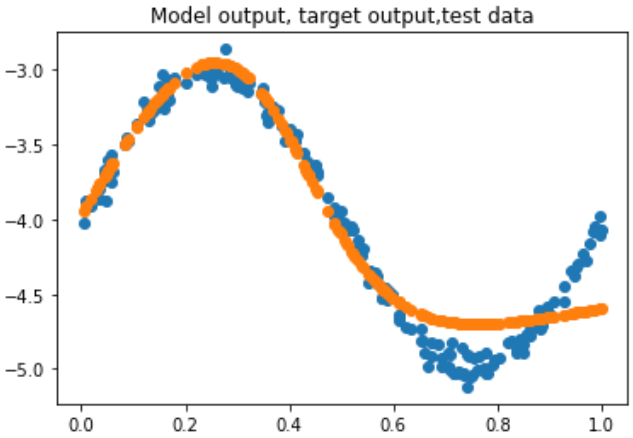
No of neurons in each layer-[1,8,7,1]

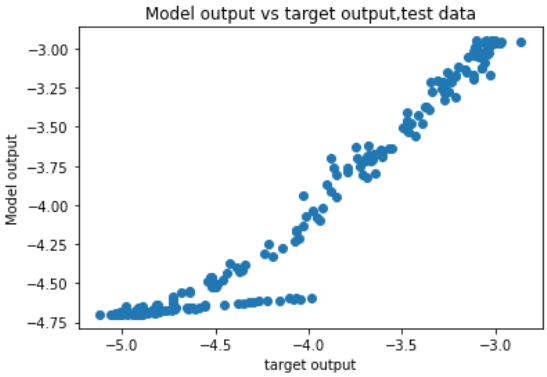
	RMSE
Training	0.176
Validation	0.186
Test	0.175





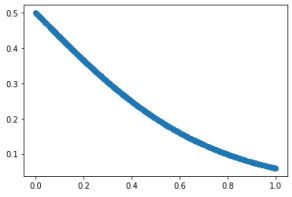
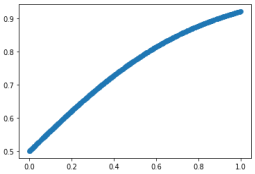
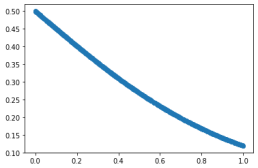
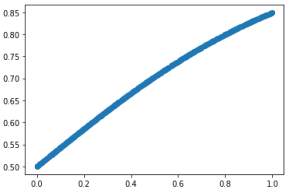


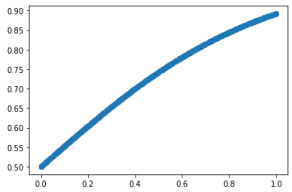
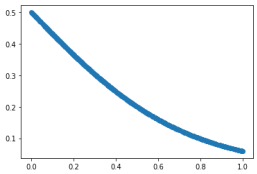
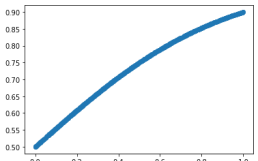
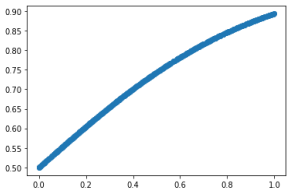
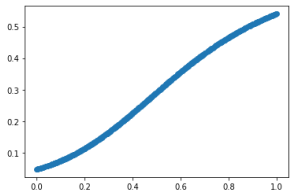
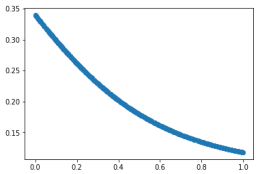
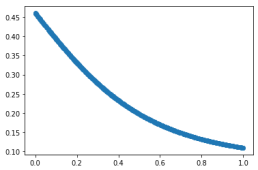
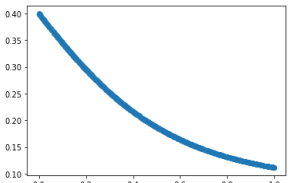
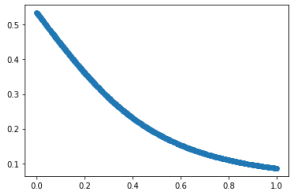
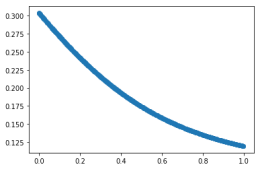
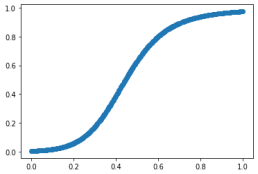
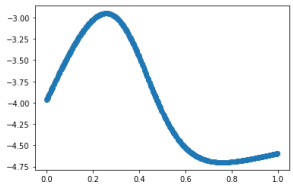




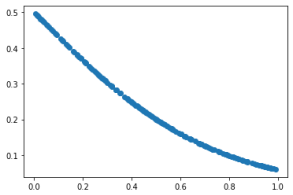
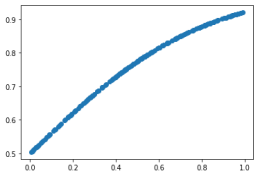
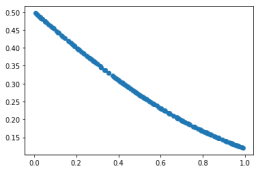
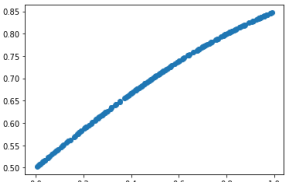
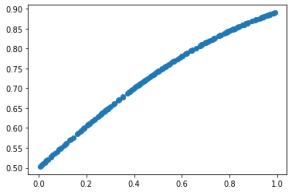
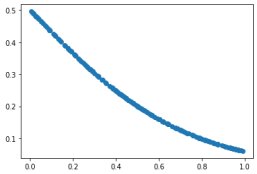
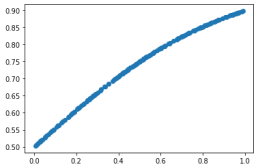
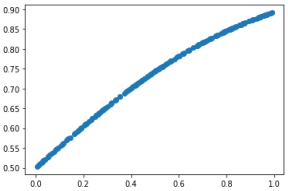
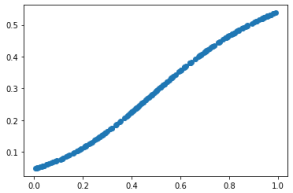
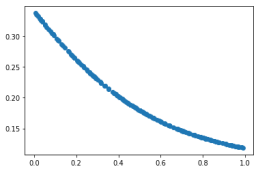
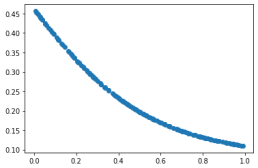
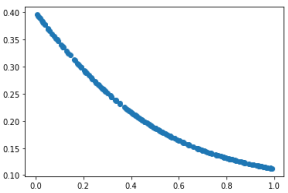
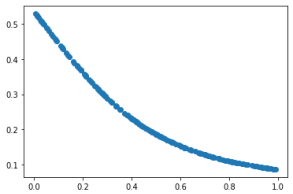
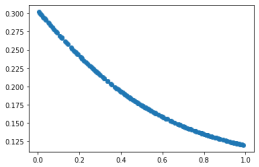
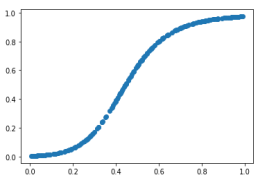
Output of each node in neural network:

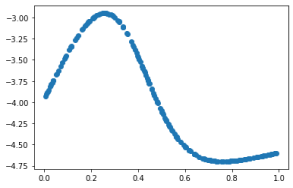
<u>On training data</u>

	Nodes in sequence			
Hidden layer 1				

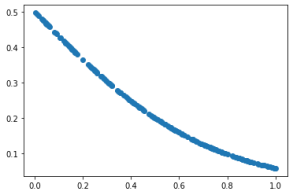
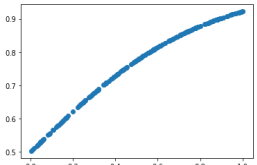
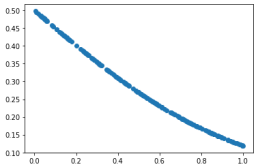
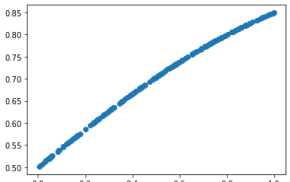
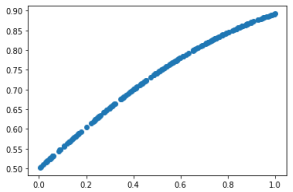
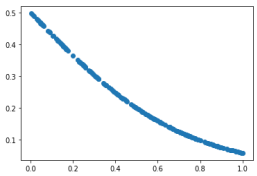
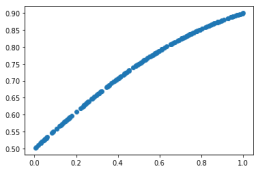
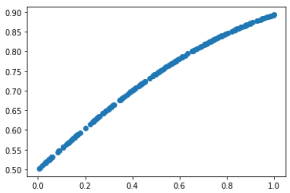
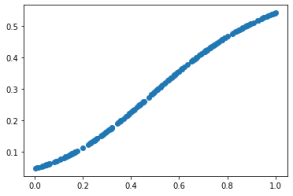
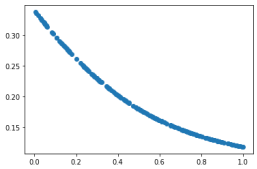
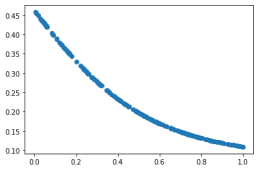
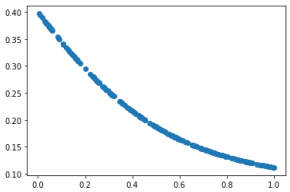
				
Hidden layer 2				
				
Output layer				

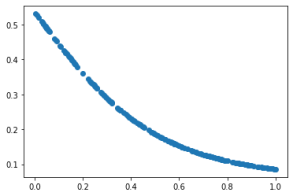
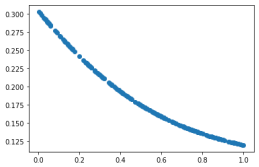
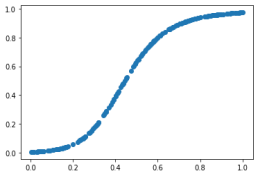
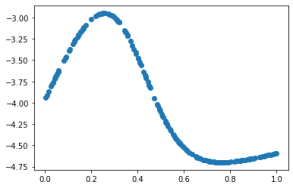
On validation data

	Nodes in sequence			
Hidden layer 1				
				
Hidden layer 2				
				

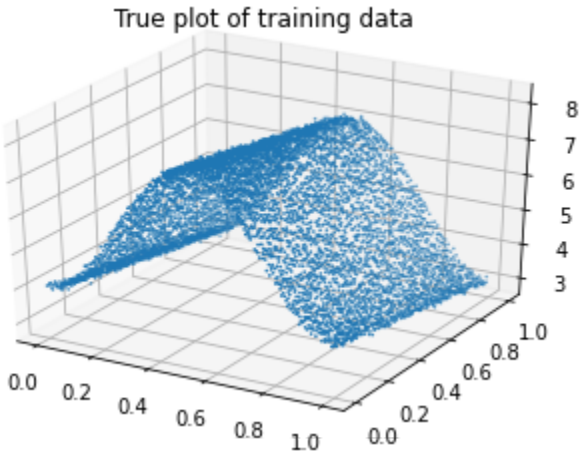
Output layer				

On test data

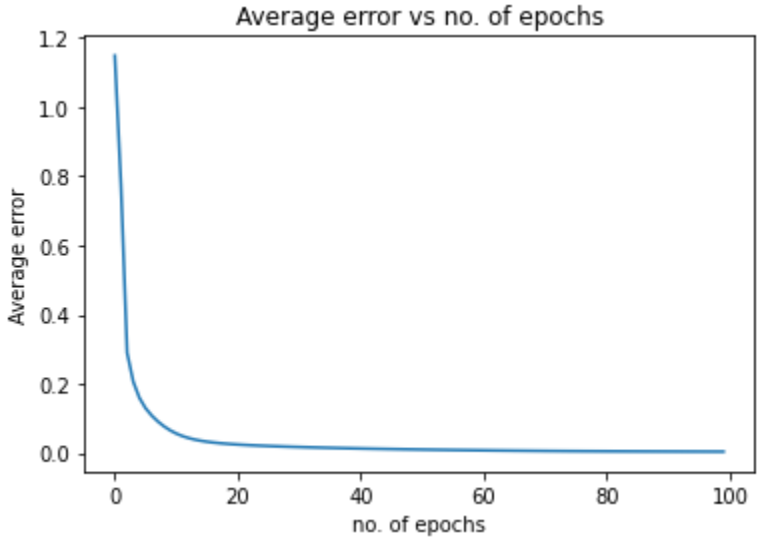
	Nodes in sequence			
Hidden layer 1				
				
Hidden layer 2				

				
Output layer				

Dataset2: 2-dimensional (Bivariate) input data:



Plot of epochs vs average error(Best architecture):



Results of different architecture:

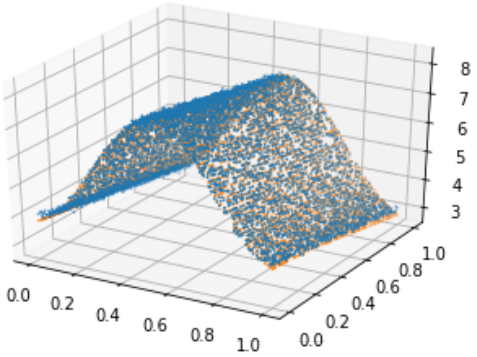
No of neurons in each layer	Root mean square Error
[2,4,3,1]	0.076
[2,4,5,1]	0108
[2,4,7,1]	.0.107
[2,6,3,1]	0.282
[2,6,5,1]	0.074
[2,6,7,1]	0.075
[2,8,3,1]	0.082
[2,8,5,1]	0.070
[2,8,7,1]	0.080

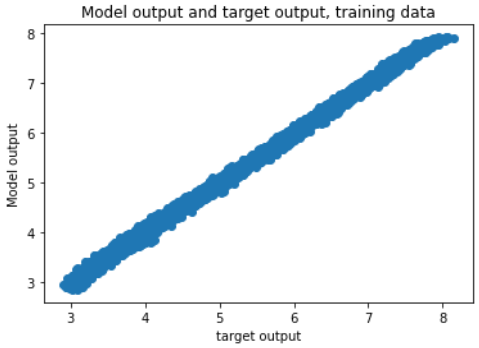
Best architecture in terms of parameters, we choose simplest as data is linearly separable:

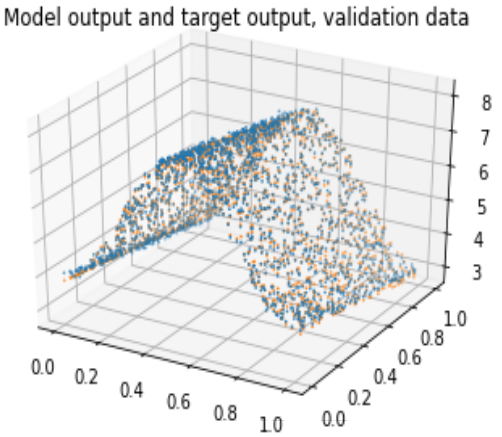
No of neurons in each layer-[2,8,5,1]

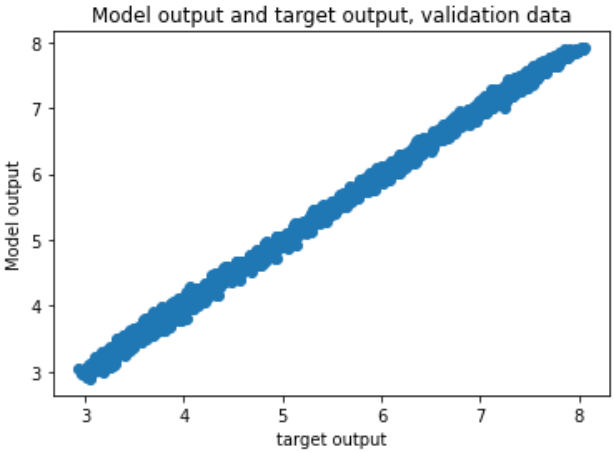
	RMSE
Training	0.070
Validation	0.070

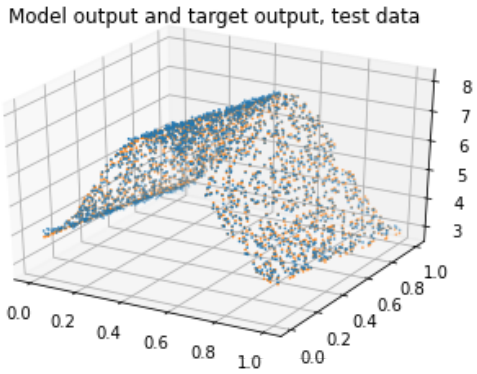
Test	0.0699




Model output and target output, training data
Model output
target output


Model output and target output, validation data

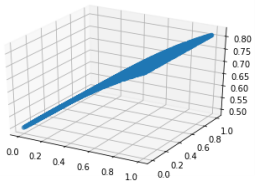
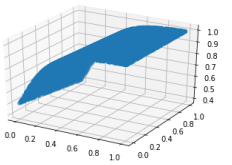
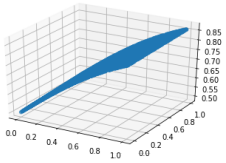
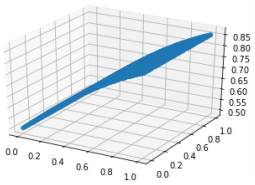
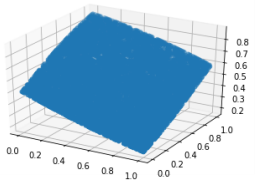
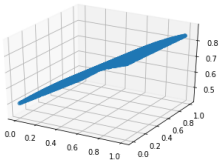
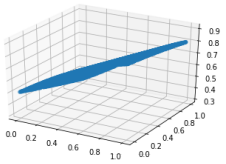
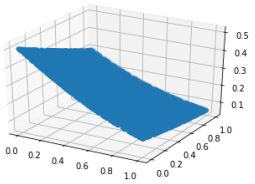
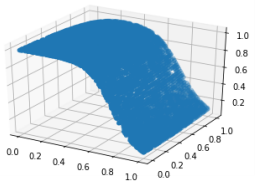
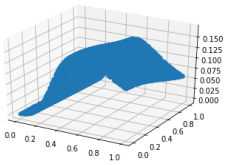
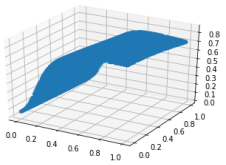
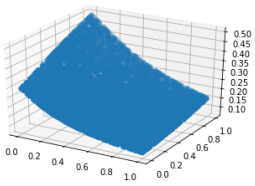
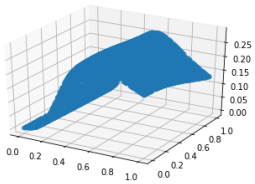

Model output and target output, validation data
Model output
target output

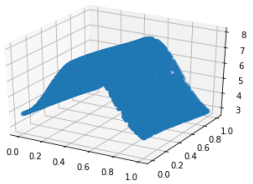

Model output and target output, test data


Model output and target output, test data
Model output
target output

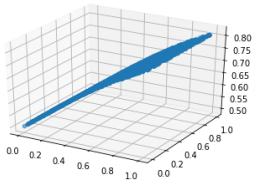
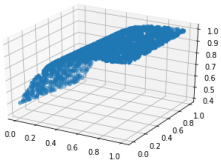
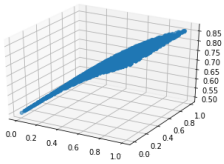
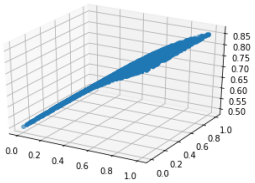
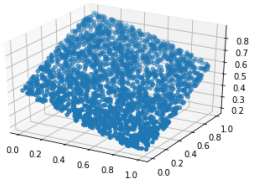
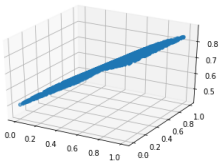
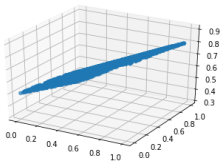
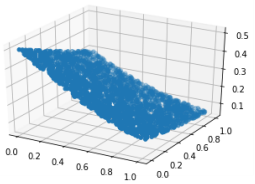
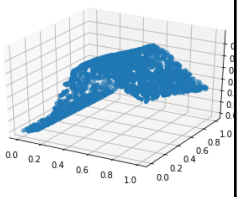
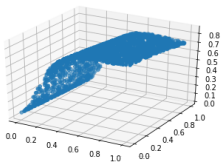
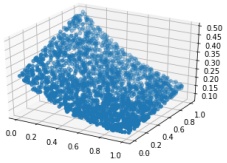
Output of each node in neural network:

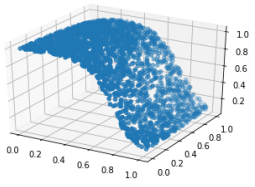
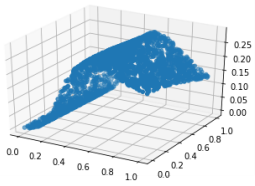
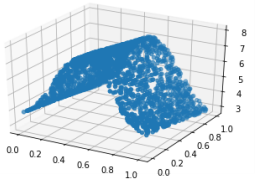
<u>On training data</u>

	Nodes in sequence			
Hidden layer 1				
				
Hidden layer 2				
				

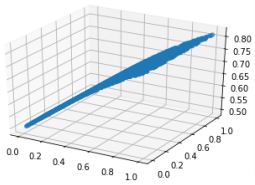
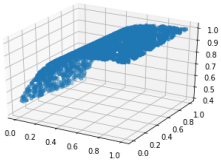
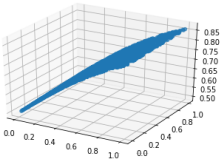
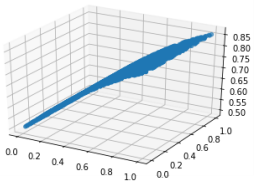
Output layer				

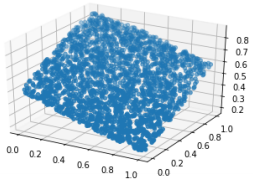
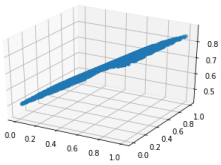
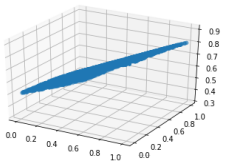
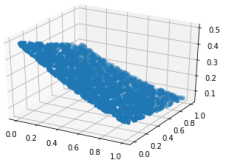
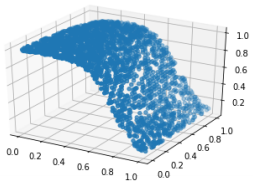
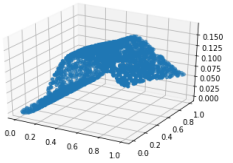
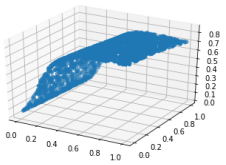
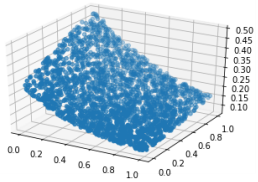
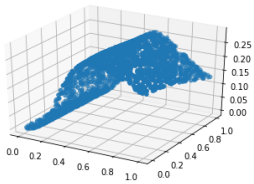
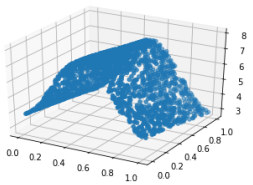
<u>On validation data</u>

	Nodes in sequence			
Hidden layer 1				
				
Hidden layer 2				

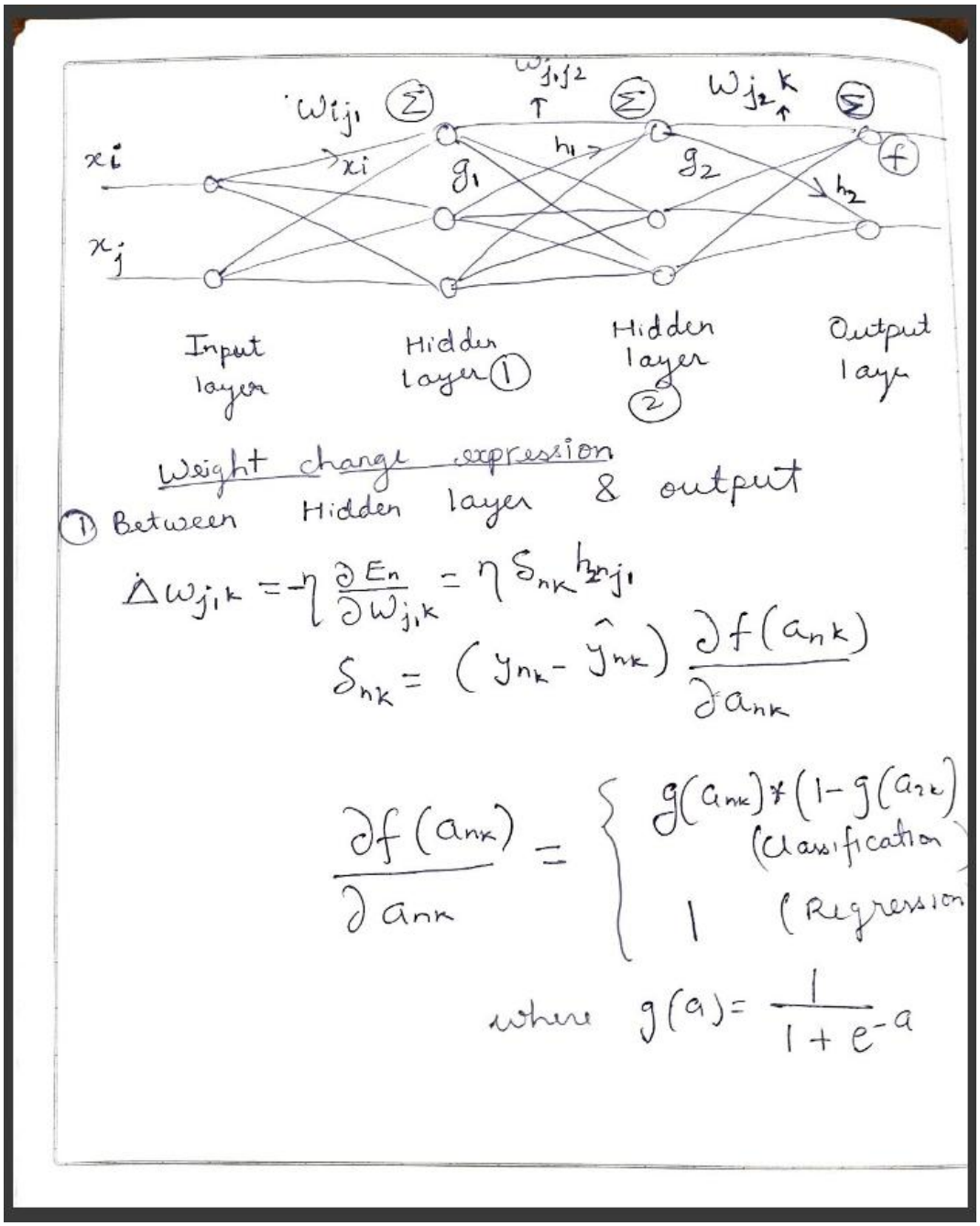
				
				
Output layer				

<u>On test data</u>

	Nodes in sequence			
Hidden layer 1				

				
Hidden layer 2				
				
Output layer				

Derivation of weight change in back propagation-



Weight change expression

(1) Between Hidden layer & output

$$\Delta w_{j1k} = -\eta \frac{\partial E_n}{\partial w_{j1k}} = \eta \delta_{nk} h_{2nj1}$$

$$\delta_{nk} = (y_{nk} - \hat{y}_{nk}) \frac{\partial f(a_{nk})}{\partial a_{nk}}$$

$$\frac{\partial f(a_{nk})}{\partial a_{nk}} = \begin{cases} g(a_{nk}) * (1 - g(a_{nk})) & \text{(Classification} \\ 1 & \text{(Regression} \end{cases}$$

where $g(a) = \frac{1}{1 + e^{-a}}$

For rest layers, expression is same for Regression & classification

(2) Between hidden layers

$$\Delta w_{j_1 j_2} = -\eta \frac{\partial E_n}{\partial w_{j_1 j_2}} = \eta \, \delta_{n j_2} \, h_{j_1}$$

$$\delta_{n j_2} = \left[\sum_{k=1}^{K} \delta_{n k} \, w_{j_2 k} \right] \frac{\partial g_2(a_{n j_2})}{\partial a_{n j_2}}$$

(3) Between input & hidden layer

$$\Delta w_{i j_1} = -\eta \frac{E_n}{\partial w_{i j_1}} = \eta \, \delta_{n j_1} \, x_i$$

$$\delta_{n j_1} = \left[\sum_{j_2=1}^{J_2} \delta_{n j_2} \, w_{j_1 j_2} \right] \frac{\partial g_1(a_{n j_1})}{\partial a_{n j_1}}$$

Conclusion/Comparison-

Classification-

When linearly separable data is taken, perceptron model is enough to classify the test points with full accuracy, as it learns a separating line/plane between different classes, but when non-separable data is given perceptron gives very poor performance. When we use FCNN model for classification of similar data, accuracy reaches ~95%, increasing number of nodes further increases the accuracy as discriminating surface tends to be more curved.

Classification-

The data given is in form of a curved line in 2D, and curved surface in 3D. When using perceptron for prediction, it fits straight line in plane and plane in space, witg large mean square error error. When we use FCNN for prediction, it tries to learn a similar curve/plane of data. RMSE also decreases significantly as compared to the perceptron model. Increasing number of nodes decreases the RMSE as fit curve tends to be similar to data.