Machine Learning Assignment-2
Documentation

2. Artificial Neural Network

Neural networks are a set of algorithms that are modeled loosely after how the human brain works, and are designed to recognize patterns. Neural networks are useful when for clustering and classifying data.

Implementation

In the preprocessing stage, the data set was split into train and test sets in a 70:30 ratio, and a normalisation function was applied separately on the three non-binary attributes of the dataset for the train and test sets. The class column was expanded into a one-hot value matrix in order to facilitate the one-out-of-k classification that is required by the problem statement. Consequently, the output layer possesses 10 nodes, as k was 10 for the given dataset.

Activation Functions like ReLU, Leaky ReLU, Sigmoid and tanh were defined, along with their derivatives, to be used in forward and backward propagation respectively. Softmax, Cross Entropy and a function to find average log error of predictions, that is called on the final (output) layer to evaluate the predictions and start rectification through backward propagation.

A Neural Network was developed, with either one or two layers, a varying number of nodes and activation functions in each layer to uncover the best set of hyperparameters for the given dataset. *He initialisation* was used to initialise the set of weights, while the bias was initialised to zero.

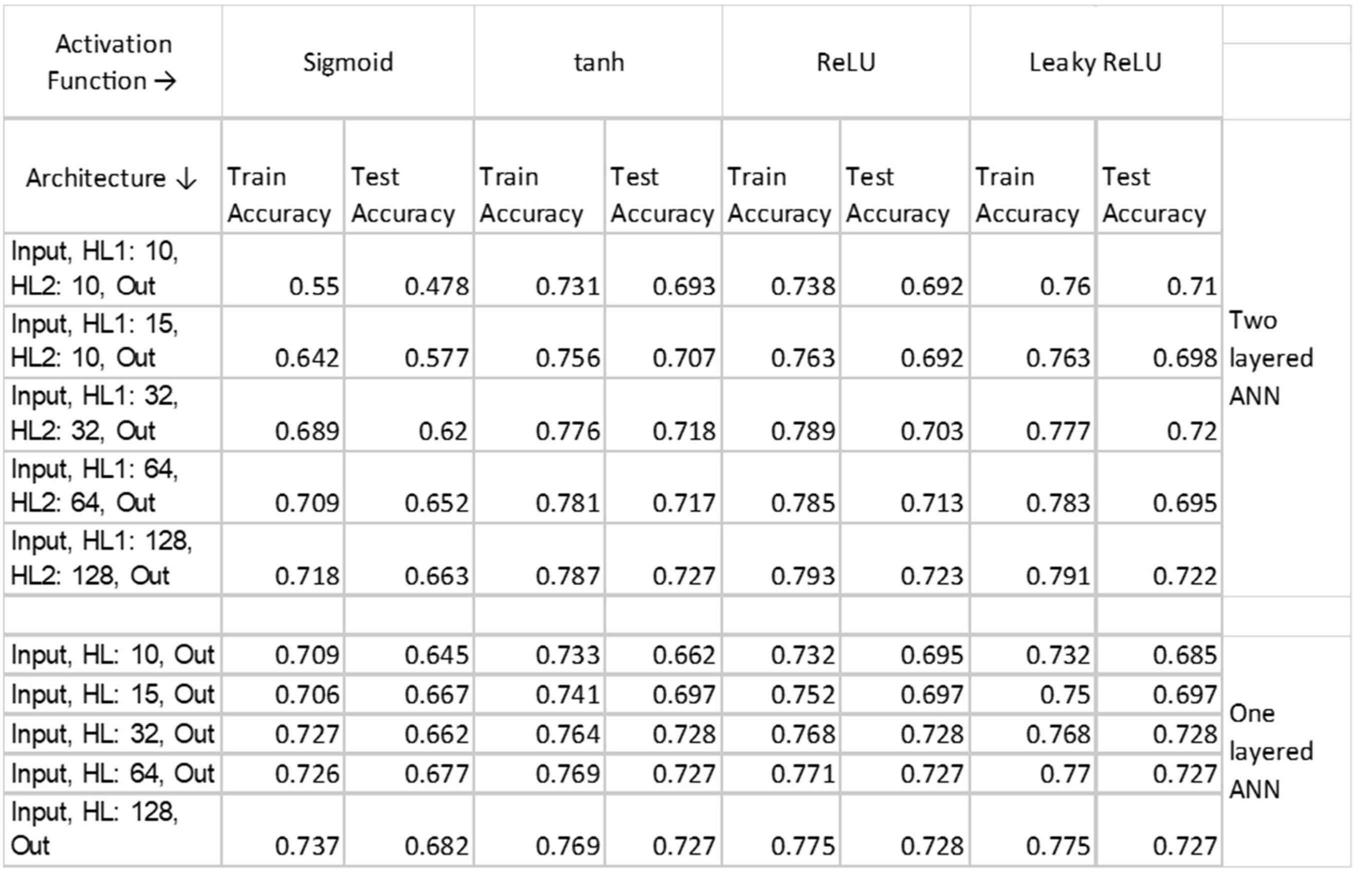
For forward propagation, a dot product was taken between the weights and the current set of values on the input layer and the first hidden layer. The activation function was applied to act as the input for the following layer to the input layer for one-layered ANN, and to the input layer and first hidden layer for two-layered ANN. For the last hidden layer(first in one-layered ANN, and second in two-layered, the softmax function was applied which would act as the input for the output layer.

Using cross entropy, the error in the input passed by the final (second for two-layers, first for one-layer) hidden layer to the output layer was obtained, and the weights were updated using the derivative of the activation function and the learning rate (according to gradient descent). The bias was similarly updated. A similar method was subsequently applied for the further layers to complete one round of back-propagation.

The model was run for 10000 iterations, and the training and testing accuracies were evaluated, and graphs plotted.

Results

The following table showcases the training and testing accuracies for various configurations of the model:



Activation Function →	Sigmoid		tanh		ReLU		Leaky ReLU		
Architecture ↓	Train Accuracy	Test Accuracy	Train Accuracy	Test Accuracy	Train Accuracy	Test Accuracy	Train Accuracy	Test Accuracy	
Input, HL1: 10, HL2: 10, Out	0.55	0.478	0.731	0.693	0.738	0.692	0.76	0.71	Two layered ANN
Input, HL1: 15, HL2: 10, Out	0.642	0.577	0.756	0.707	0.763	0.692	0.763	0.698	
Input, HL1: 32, HL2: 32, Out	0.689	0.62	0.776	0.718	0.789	0.703	0.777	0.72	
Input, HL1: 64, HL2: 64, Out	0.709	0.652	0.781	0.717	0.785	0.713	0.783	0.695	
Input, HL1: 128, HL2: 128, Out	0.718	0.663	0.787	0.727	0.793	0.723	0.791	0.722	
Input, HL: 10, Out	0.709	0.645	0.733	0.662	0.732	0.695	0.732	0.685	One layered ANN
Input, HL: 15, Out	0.706	0.667	0.741	0.697	0.752	0.697	0.75	0.697	
Input, HL: 32, Out	0.727	0.662	0.764	0.728	0.768	0.728	0.768	0.728	
Input, HL: 64, Out	0.726	0.677	0.769	0.727	0.771	0.727	0.77	0.727	
Input, HL: 128, Out	0.737	0.682	0.769	0.727	0.775	0.728	0.775	0.727	

(Note: HL i -> Number of Nodes in i^{th} Hidden Layer, Activation functions were the same for both the hidden layers)

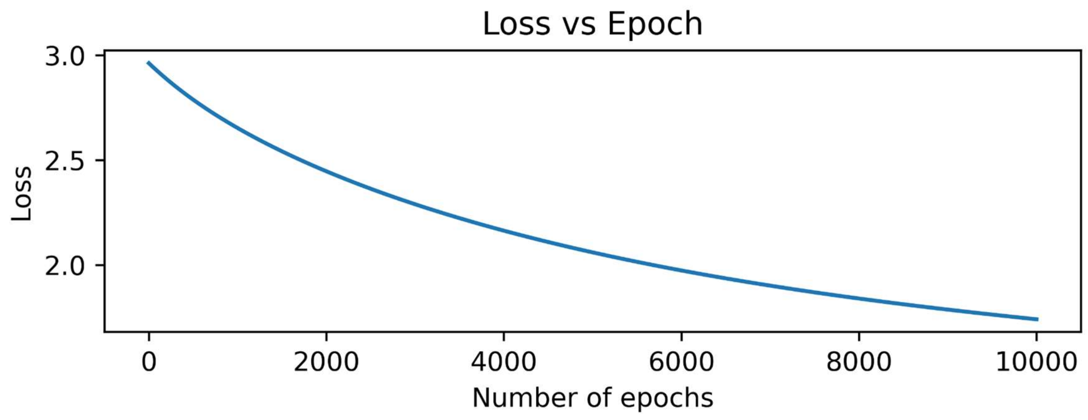
From the above table, it was decided that Leaky ReLU was the most appropriate Activation Function, while 32 nodes per hidden layer resulted in the best results for both single and two-layered ANNs. It was observed that using too many nodes resulted in overfitting and less generalization, as can be seen in the difference between training and testing accuracies. Moreover, a high rate of learning resulted in overfitting and oscillations, while if the rate of learning was too low, then the model failed to converge within 10000 iterations. The sweet spot, so to speak, was obtained with learning rate = 0.01

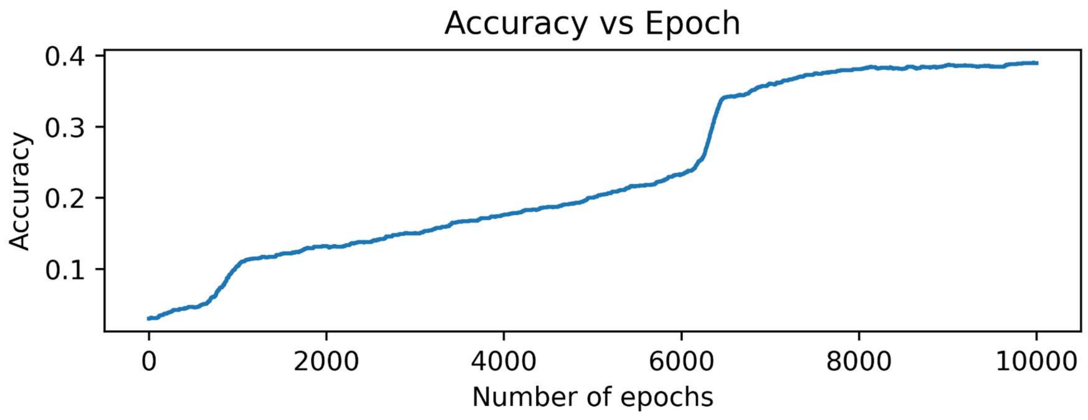
Loss for ANN with One-layer: 0.585808282482085
Training Accuracy for ANN with One-layer: 0.768
Testing Accuracy for ANN with One-layer: 0.728

Loss for ANN with Two-layers: 0.5351930298262003
Training Accuracy for ANN with Two-layers: 0.777
Testing Accuracy for ANN with Two-layers: 0.72

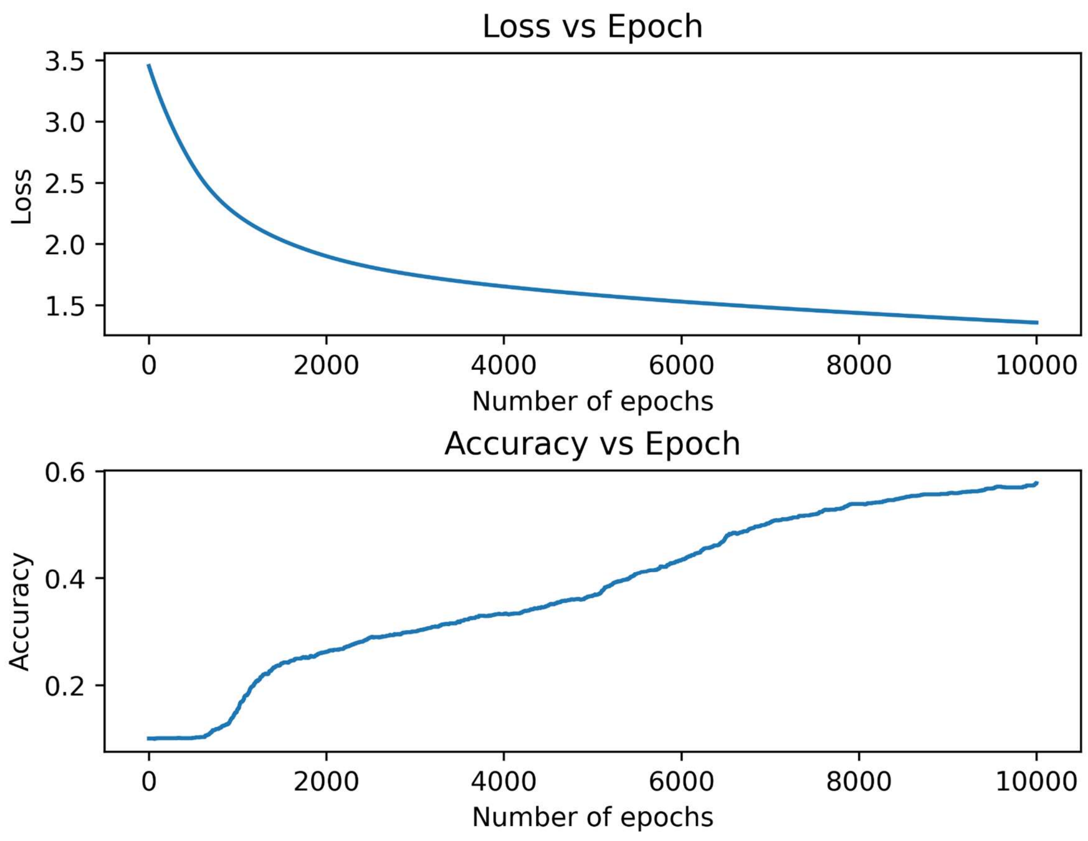
Plots

Plot of Accuracy for One-layer ANN with learning rate = 0.0001

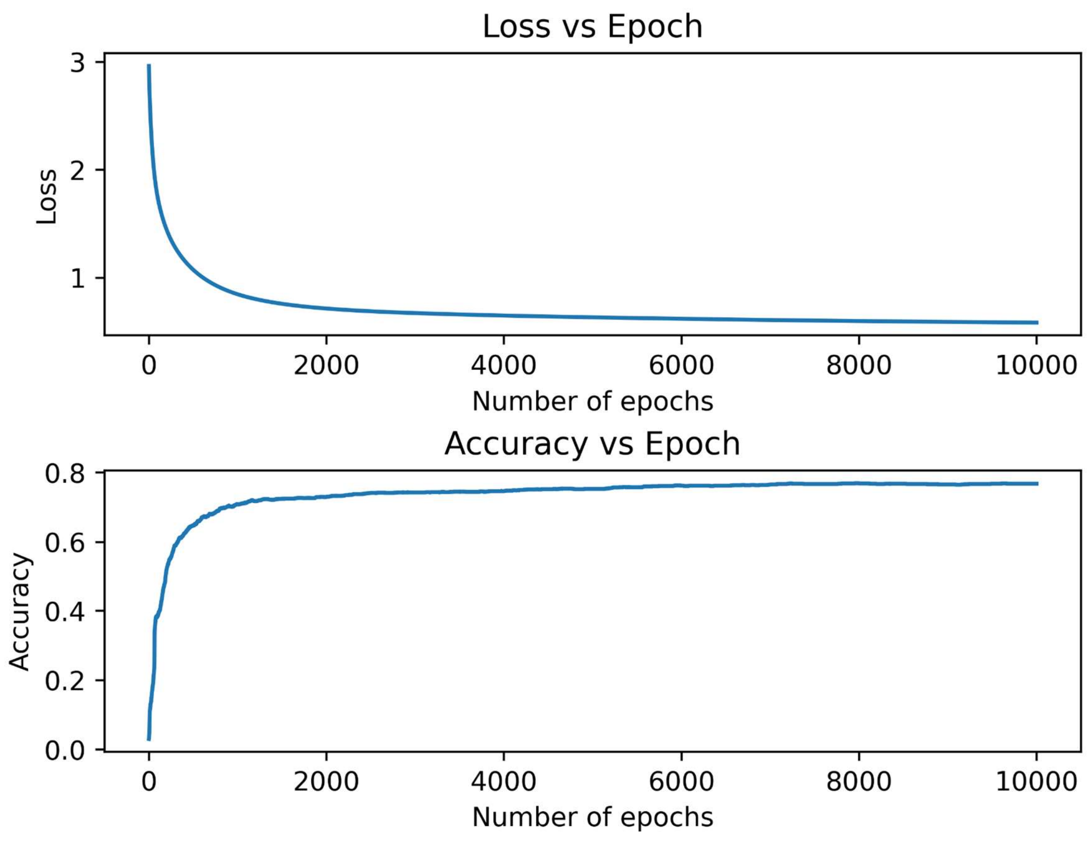

Loss vs Epoch
3.0
2.5
2.0
Loss
0
2000
4000
6000
8000
10000
Number of epochs


Accuracy vs Epoch
0.4
0.3
0.2
0.1
Accuracy
0
2000
4000
6000
8000
10000
Number of epochs

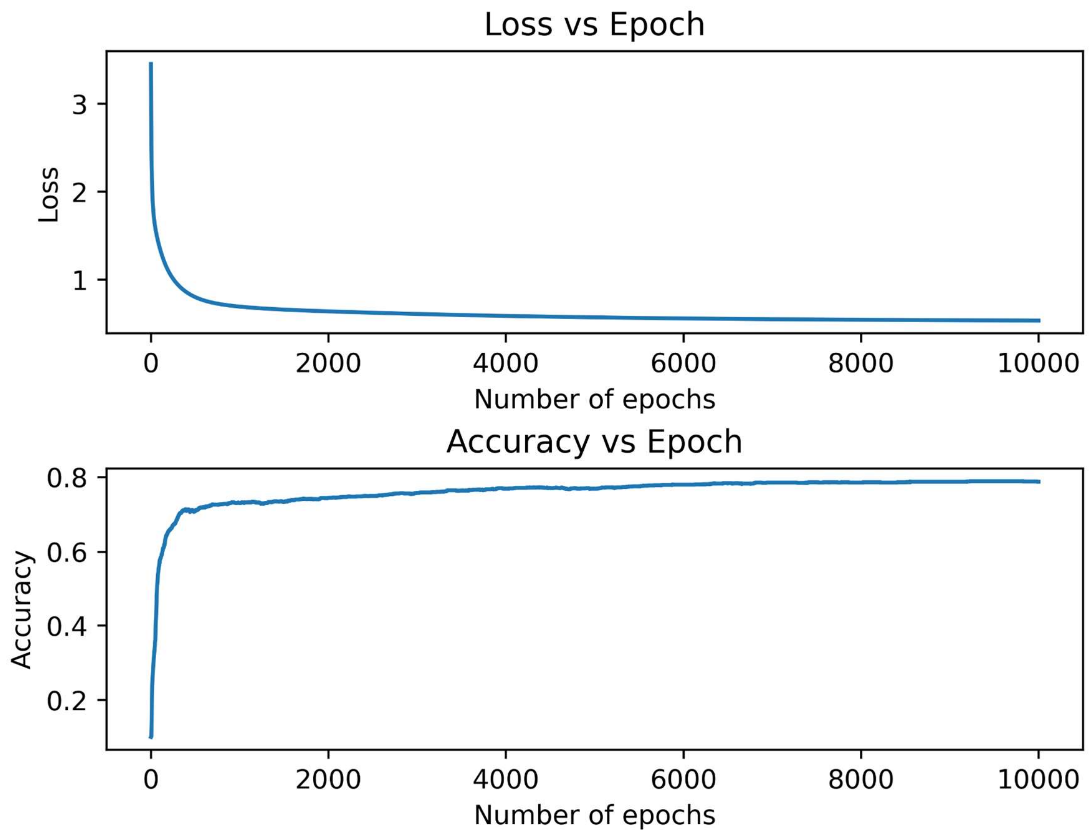
Plot of Accuracy for Two-layer ANN with learning rate = 0.0001



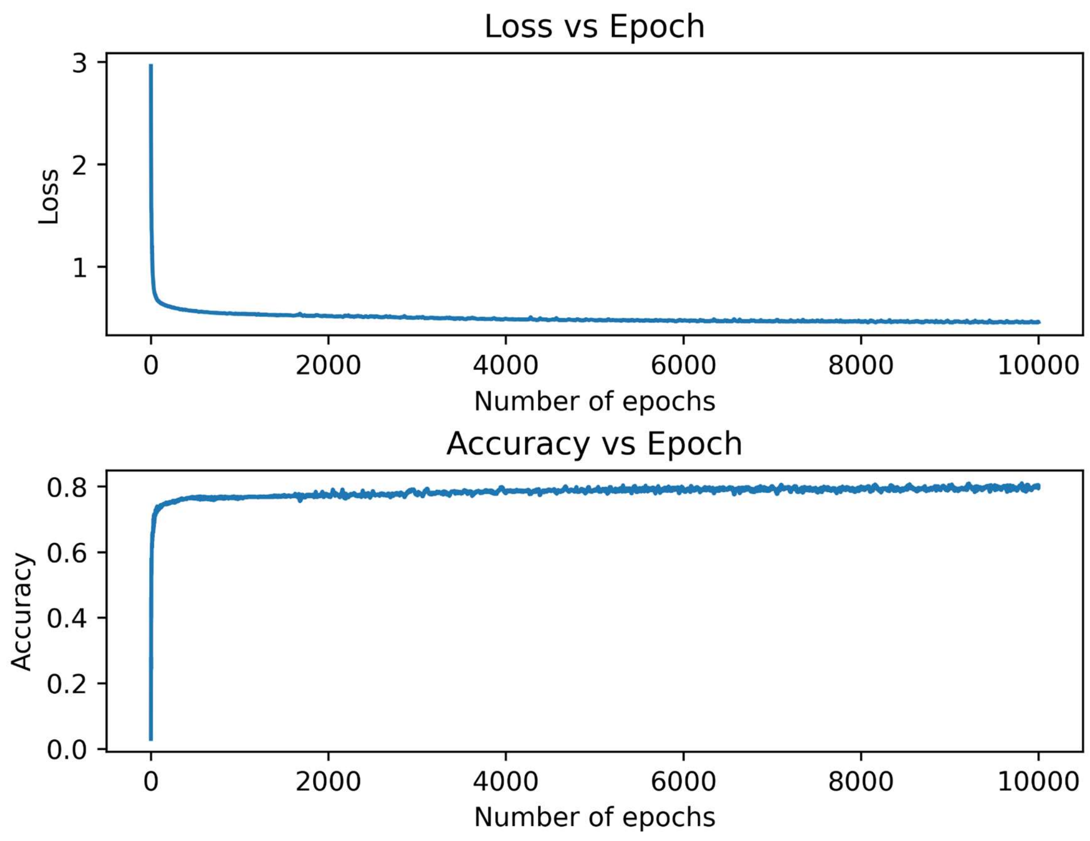
Plot of Accuracy for One-layer ANN with learning rate = 0.01



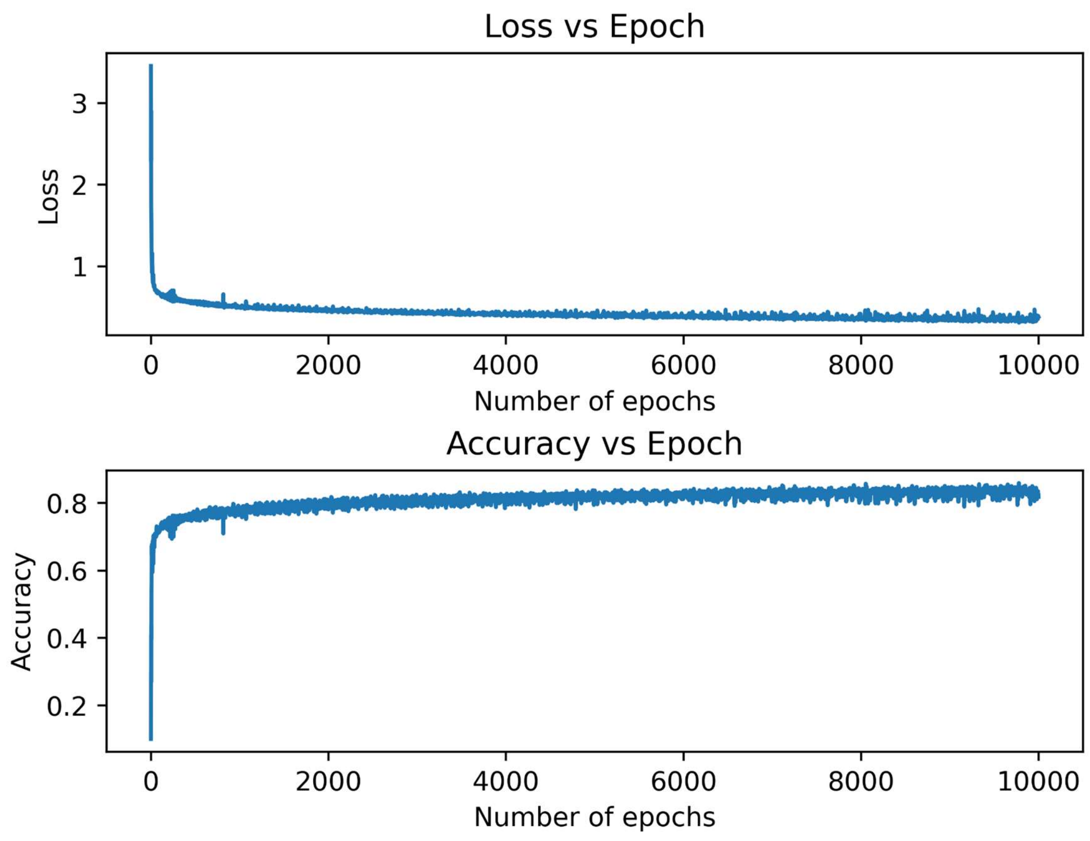
Plot of Accuracy for Two-layer ANN with learning rate = 0.01



Plot of Accuracy for One-layer ANN with learning rate = 0.5



Plot of Accuracy for Two-layer ANN with learning rate = 0.5


Loss vs Epoch
Loss
Number of epochs
Accuracy vs Epoch
Accuracy
Number of epochs