ML-A1: Artificial Neural Network in Python from Scratch

Group Members

- **Jhaveri Aditya Alok (2018A7PS0209H)**
- **Aryesh Harshal Koya (2018A4PS0637H)**
- **Vaishnavee Nautiyal (2018A7PS0286H)**

Brief Description of Model and Implementation:

Neural Networks are a series of algorithms that identify the underlying relationship in a set of data. These Algorithms are heavily based on the way a human brain operates.

Our Model takes the input of standardized training data and the training class labels dataset to train itself. We first introduce the function to calculate the mini batch gradient descent which returns the parameters, loss and accuracy for the training set. In that function our model iterates over a specific number of epochs in which in forward propagates the dataset and gets the gradient for the respecting parameters which are then changed over these epochs. Then error, predictions of our training data set and accuracies are also to be found out. The getBatch function is used to get the batch size. The forward propagation function returns the answers to the outlier layers which we have called them as nodes. It uses activation functions namely sigmoid function for the hidden layer activation and the softmax function for the output layer activation.

Brief Description of the parameters used

We have selected the learning rates to be 0.5, 0.05 and 0.1 and epochs to be 1000 to see how our model worked and we found out it seemed to be working fine on these parameters after a lot of testing on different ones.

We took 128 numbers of neurons for both hidden layers as we wanted it to be a power of 2 and after testing a lot of values and looking at the time complexity this was found to be appropriate and gave slightly better results.

We applied both single layered as well as doubled layered neural networks.

We used sigmoid function as an activation function for the hidden layers and the softmax function for the activation of the output layer. We used sigmoid because

The main reason why we use **sigmoid function** is because it exists between (0 to 1). Therefore, it is especially used for models where we have to predict the probability as an output. Since probability of anything exists only between the range of 0 and 1, **sigmoid** is the right choice. The **function** is differentiable.

We used softmax because it’s value ranges from 0 to 1 and the sum of all values is always 1.

The Loss and Accuracies obtained by our model with one and two hidden layers are as follows:

For learning rate = 0.5

One Layered Neural Network's results are as follow:

Training Accuracy for 76.07% for learning rate: 0.5
Testing Accuracy 74.50% for learning rate: 0.5
Final Loss 0.59

Two Layered Neural Network's results are as follow:

Training Accuracy 76.71% for learning rate: 0.5
Testing Accuracy 73.83% for learning rate: 0.5
Final Loss 0.61

For learning rate = 0.05

One Layered Neural Network's results are as follow:

Training Accuracy for 73.43% for learning rate: 0.05
Testing Accuracy 71.17% for learning rate: 0.05
Final Loss 0.65

Two Layered Neural Network's results are as follow:

Training Accuracy 76.14% for learning rate: 0.05
Testing Accuracy 72.67% for learning rate: 0.05
Final Loss 0.57

For learning rate = 0.01

One Layered Neural Network's results are as follow:

Training Accuracy for 69.93% for learning rate: 0.01
Testing Accuracy 67.50% for learning rate: 0.01
Final Loss 0.77

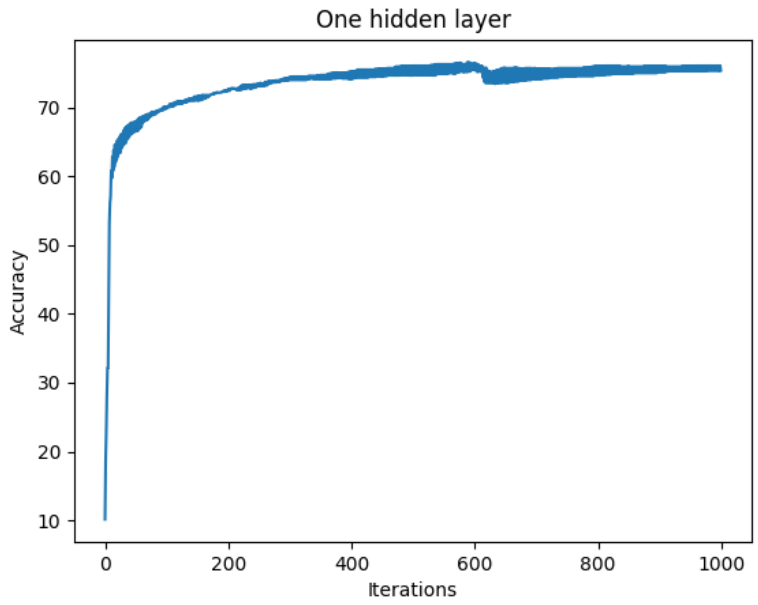
Two Layered Neural Network's results are as follow:

Training Accuracy 71.36% for learning rate: 0.01
Testing Accuracy 67.83% for learning rate: 0.01
Final Loss 0.71

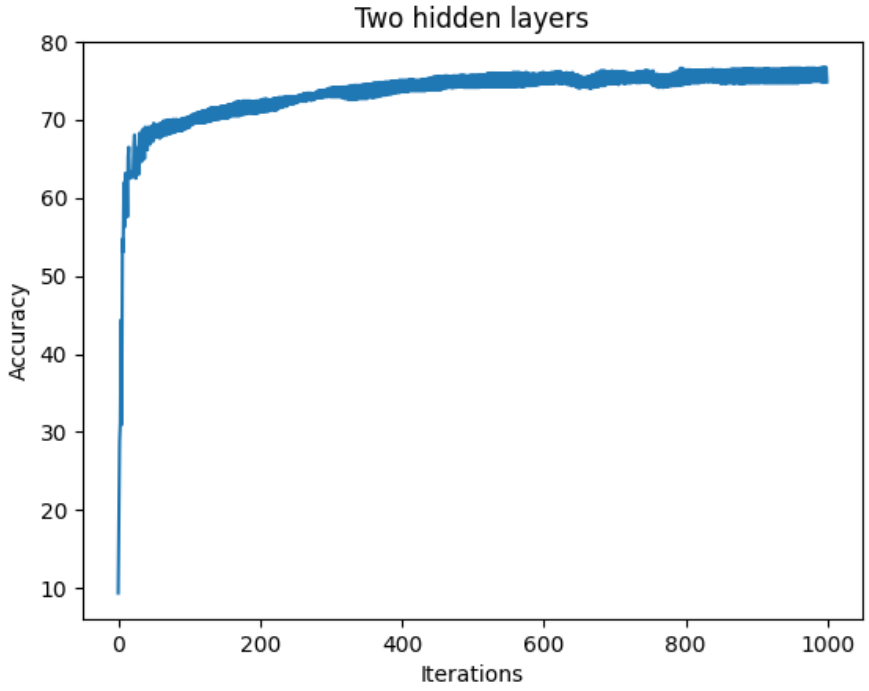
The Graphs Obtained of accuracies over 1000 epochs

For learning rate = 0.5

One - layered

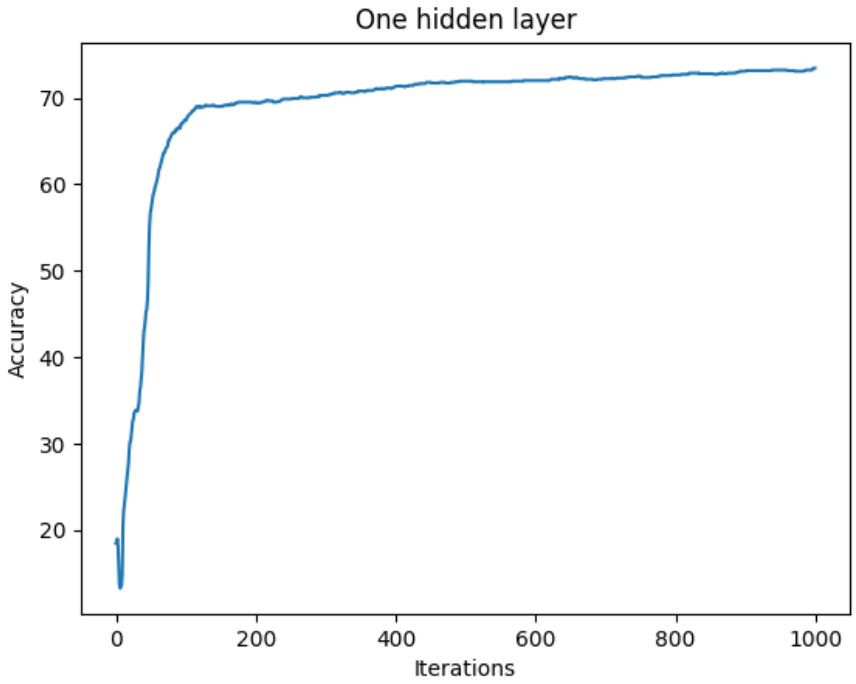


Two-Layered

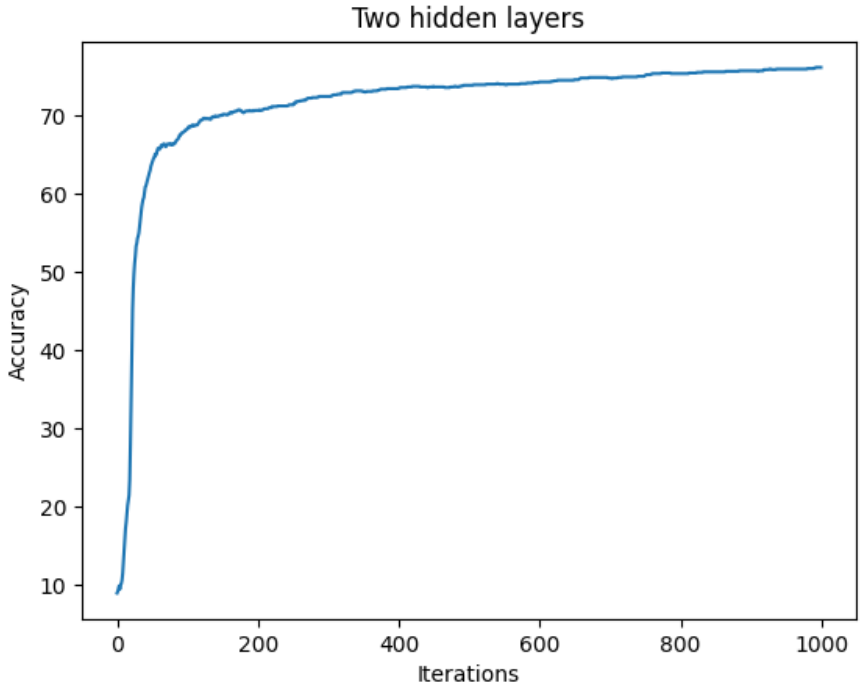


For Learning Rate = 0.05

One-Layered

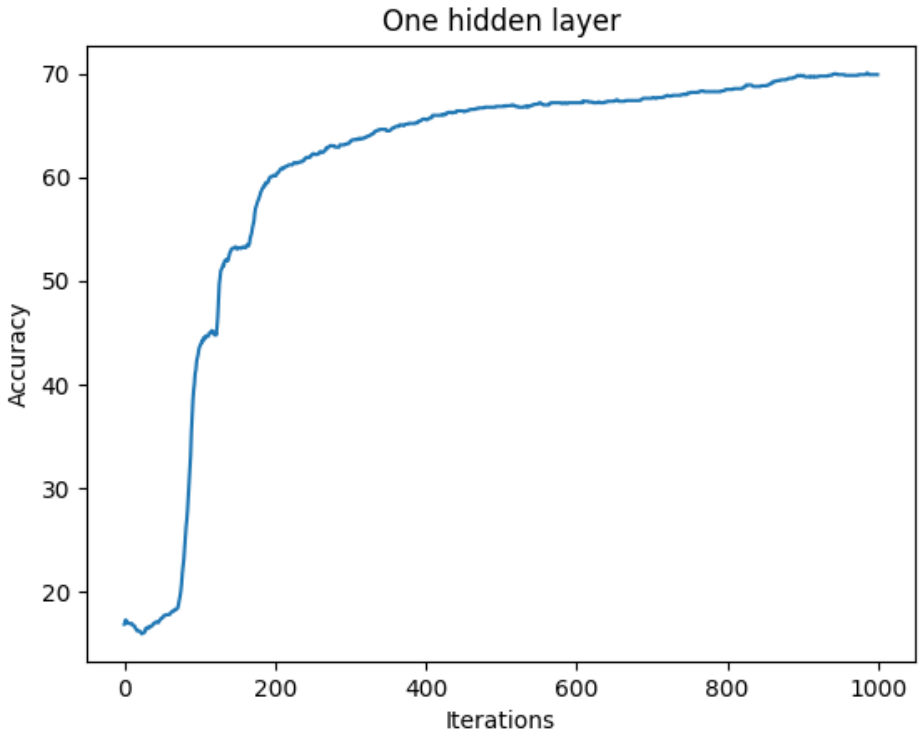


Two-Layered



For Learning rate = 0.01

One-layered



Two-Layered