**Machine Learning Assignment-2**

**Documentation**

# Logistic Regression

Logistic Regression is a machine learning algorithm primarily used for binary classification algorithms. Here the model is required to predict whether the given point belongs to class A or to class B, where the classes are mutually exclusive.

## Implementation

The dataset was trained and tested on 10 random splits of 70:30 ratio.

For gradient descent, a descent function was used which was called <number of iteration> times.

This function used the derivative of the loss function and subtracted the dot product of the derivatives and entire Training Data from the weights.

All the equations above are vectorized.

For stochastic gradient descent, the descent function was called <number of iteration> times, updating weights for each data-point one at a time and instead of taking dot product of the entire training set, only the current sample is multiplied with the derivative of loss for updating the weights.

### Difference between SGD and GD:

In stochastic gradient descent the weights are updated very fast i.e. for each sample the weights are adjusted. As a result the optimum weights are arrived upon in a lesser number of iterations. This is less computationally intensive but might take longer as the weight update for each input can’t be vectorized like in the case of GD wherein the entire dataset is passed in one go.

In GD since the entire dataset is passed in one go, the weights are updated very less frequently and since it is vectorized it is more computationally intensive. However if vectorization is possible then GD is faster even if it takes a larger number of iterations, because computations are evaluated parallely.

So the choice between these two depends on the kind of hardware available. If you one has a larger memory and faster CPU then GD/MBGD is expected to perform better, however if resources are limited one is better off using SGD even though it might take longer to evaluate.

## Results

Final test and train metrics:

GD:

Training Accuracy (10 random splits) : 98.85416666666666

Training Precision (10 random splits) : 0.9835231691737334

Training Recall (10 random splits) : 0.990896422829312

Training F-Score (10 random splits) : 0.9871960286975288

Training loss (10 random splits) : 0.03704528357332083

Testing Accuracy (10 random splits) : 98.95631067961165

Testing Precision (10 random splits) : 0.9841458085479381

Testing Recall (10 random splits) : 0.9924925479416025

Testing F-Score (10 random splits) : 0.9883015553806088

Testing loss (10 random splits) : 0.03676691150719037

SGD:

Training Accuracy (10 random splits) : 98.77083333333334

Training Precision (10 random splits) : 0.9846436819024855

Training Recall (10 random splits) : 0.9878123478118713

Training F-Score (10 random splits) : 0.986225469694322

Training loss (10 random splits) : 0.050865832541637124

Testing Accuracy (10 random splits) : 98.6893203883495

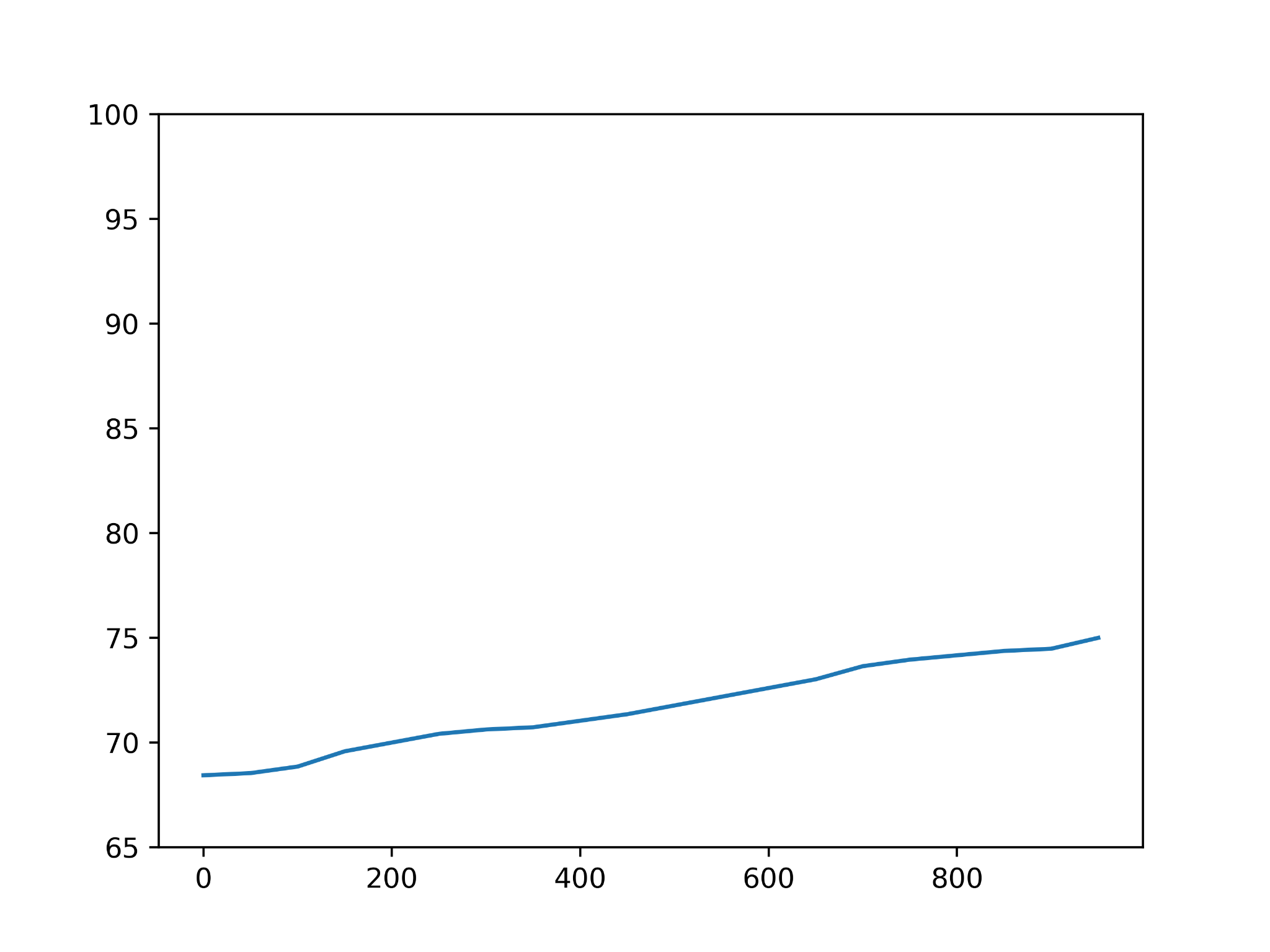
Testing Precision (10 random splits) : 0.985699589180606

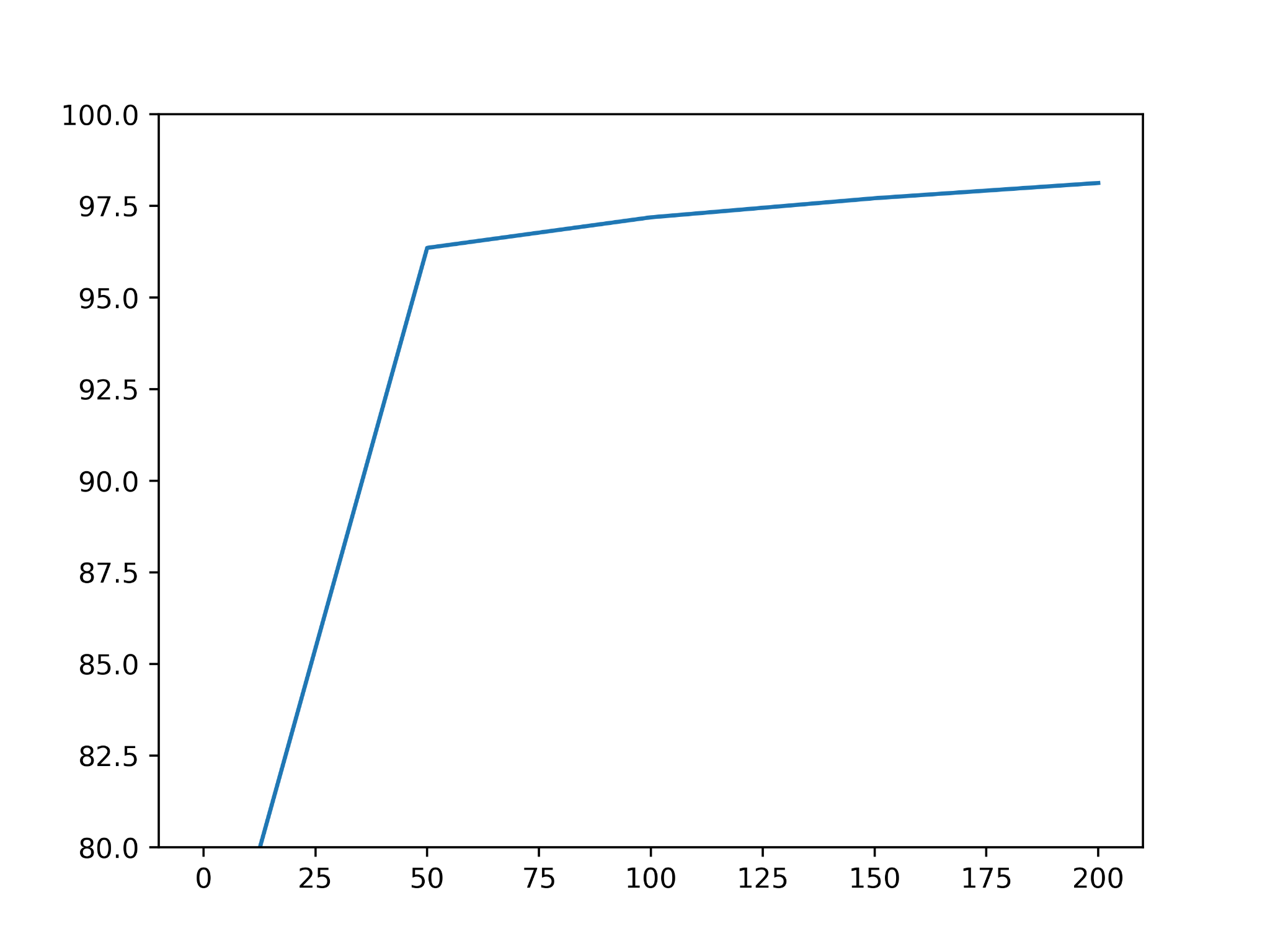
Testing Recall (10 random splits) : 0.9847897869474439

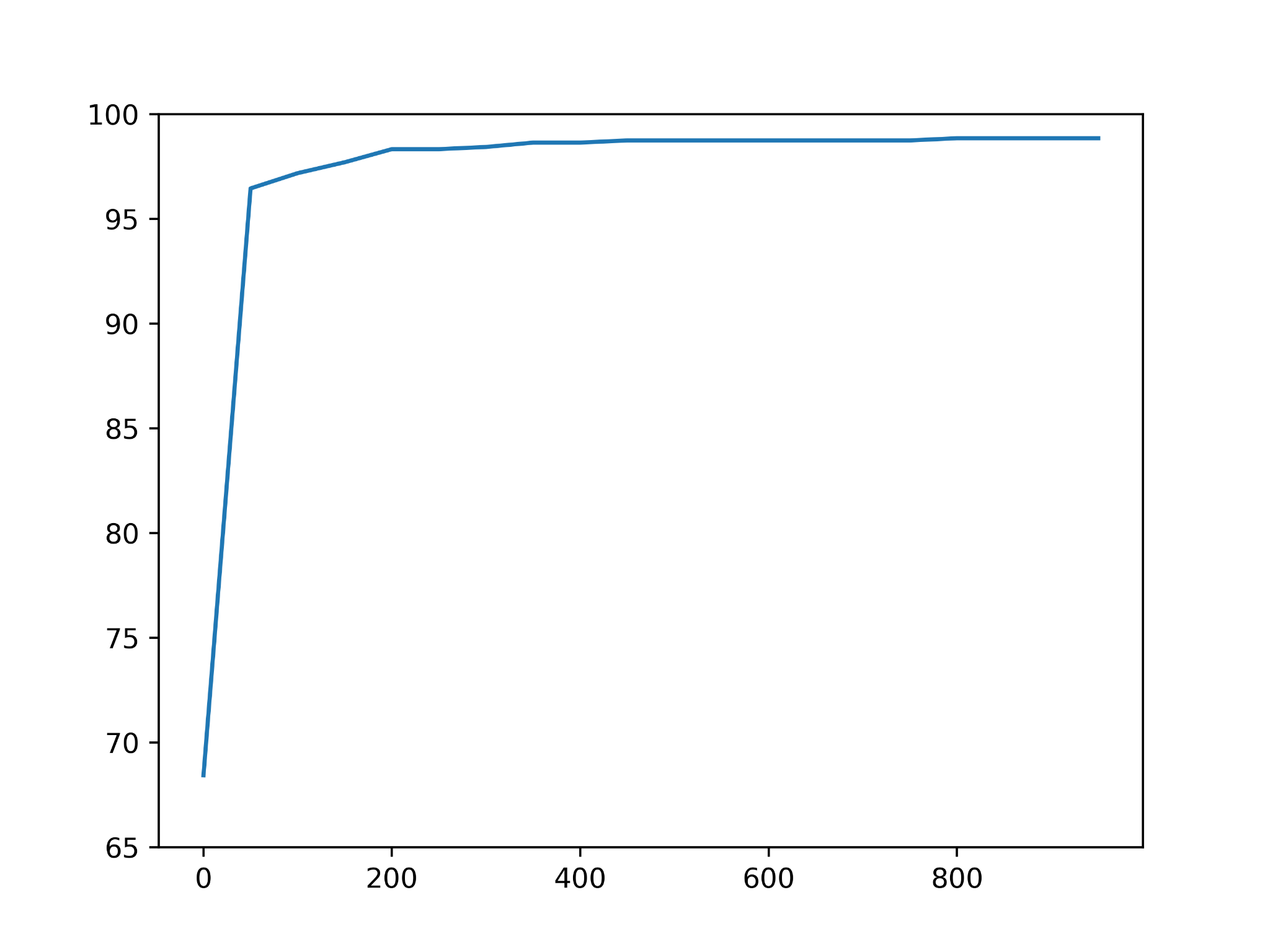
Testing F-Score (10 random splits) : 0.9852444780298798

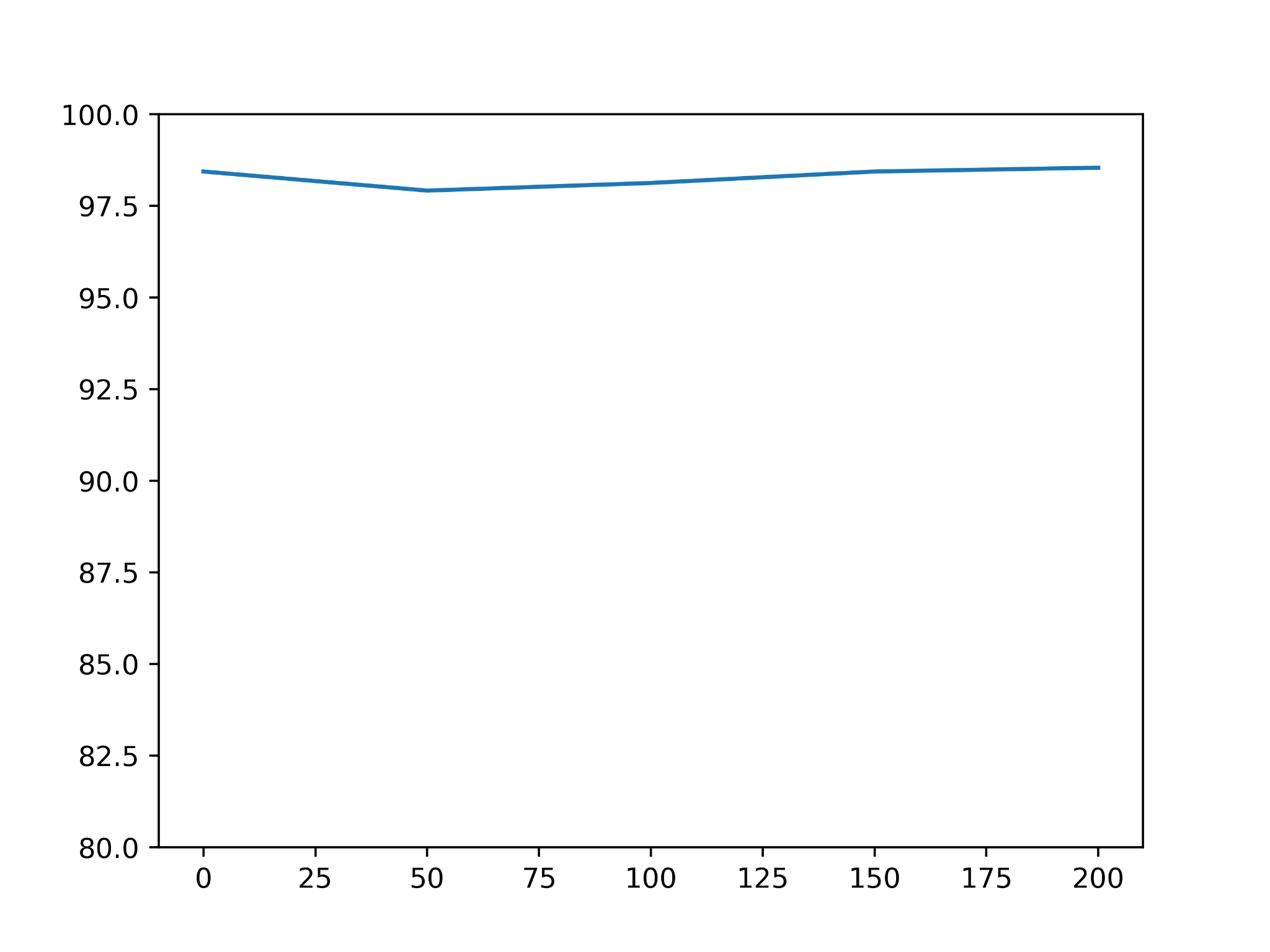
Testing loss (10 random splits) : 0.05514615170088646

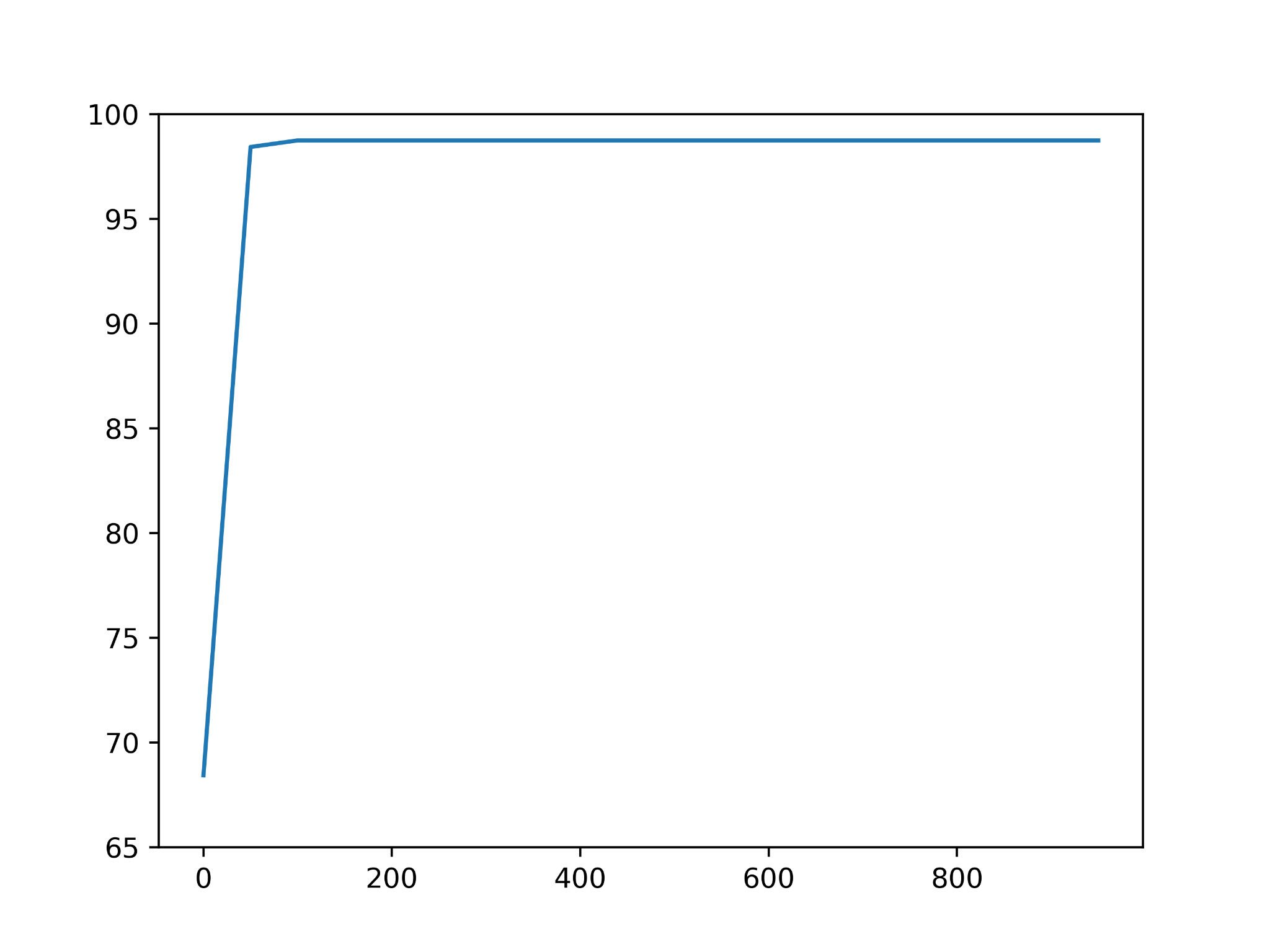
## Plots

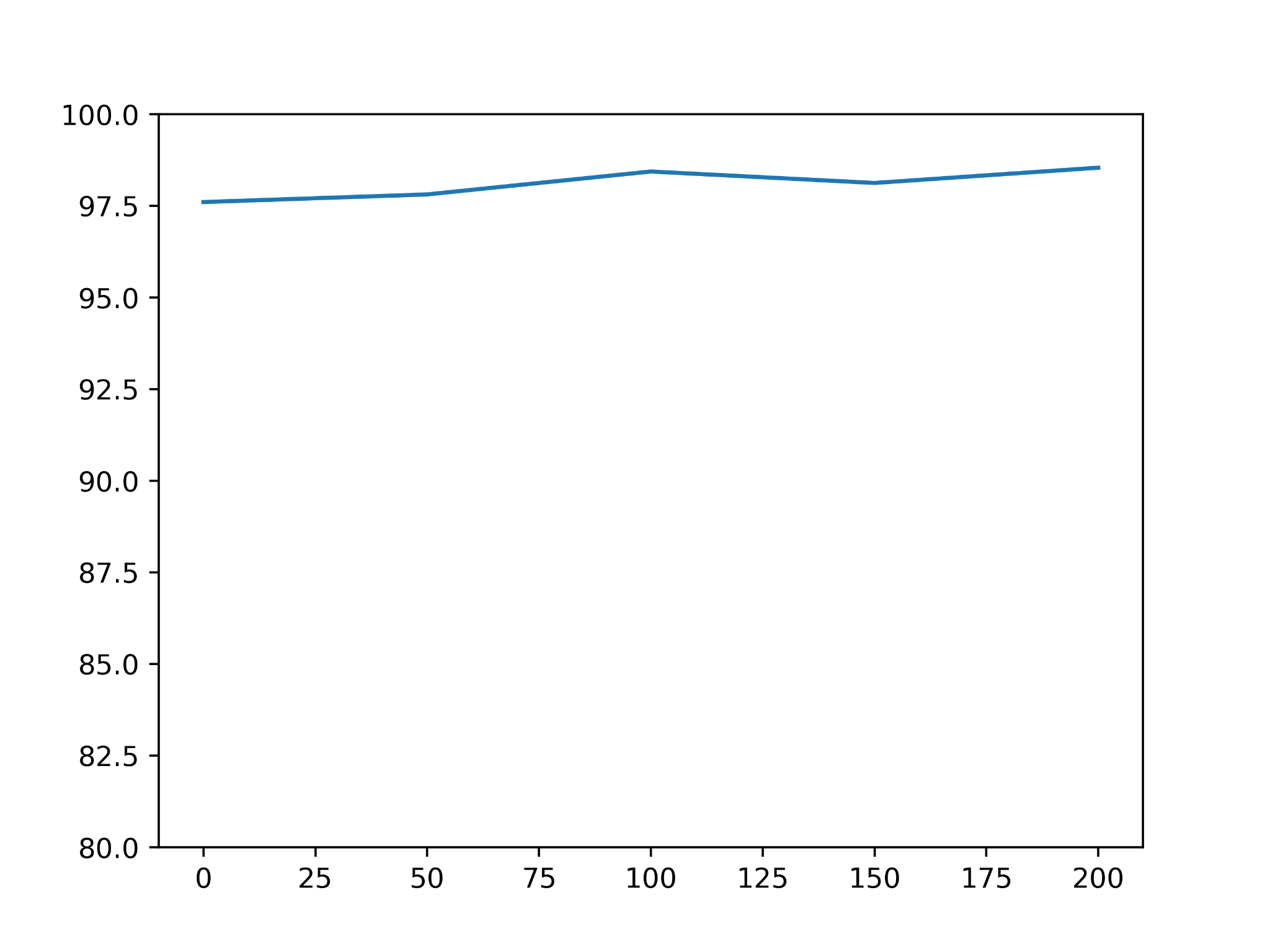
Plot of Accuracy for GD with learning rate = 0.0001

Plot of Accuracy for SGD with learning rate = 0.0001

Plot of Accuracy for GD with learning rate = 0.1

Plot of Accuracy for SGD with learning rate = 0.1

Plot of Accuracy for GD with learning rate = 5

Plot of Accuracy for SGD with learning rate = 5

# 

# 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.

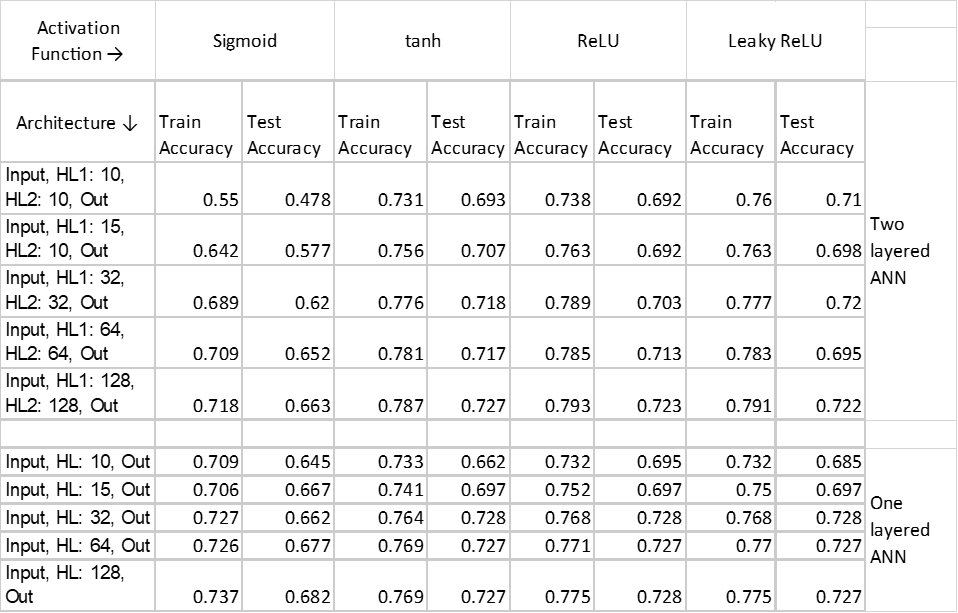
For forward propagation, a dot product was taken between the weights and the current set of values on the input layer and the first hidden layer. The activation function was applied to act as the input for the following layer to the input layer for one-layered ANN, and to the input layer and first hidden 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:



(Note: HL i -> Number of Nodes in ith 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

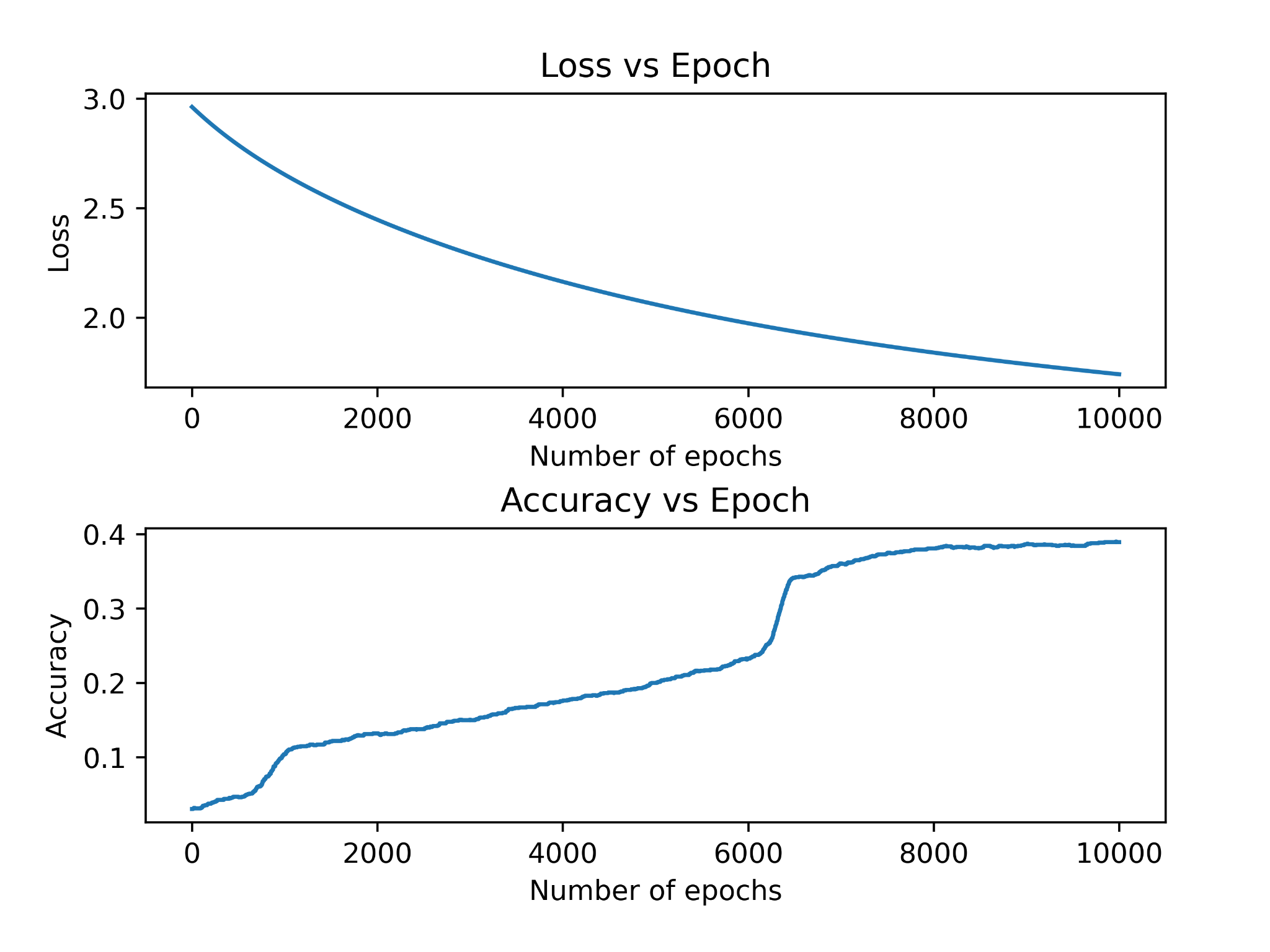
Loss for ANN with Two-layers: 0.5351930298262003

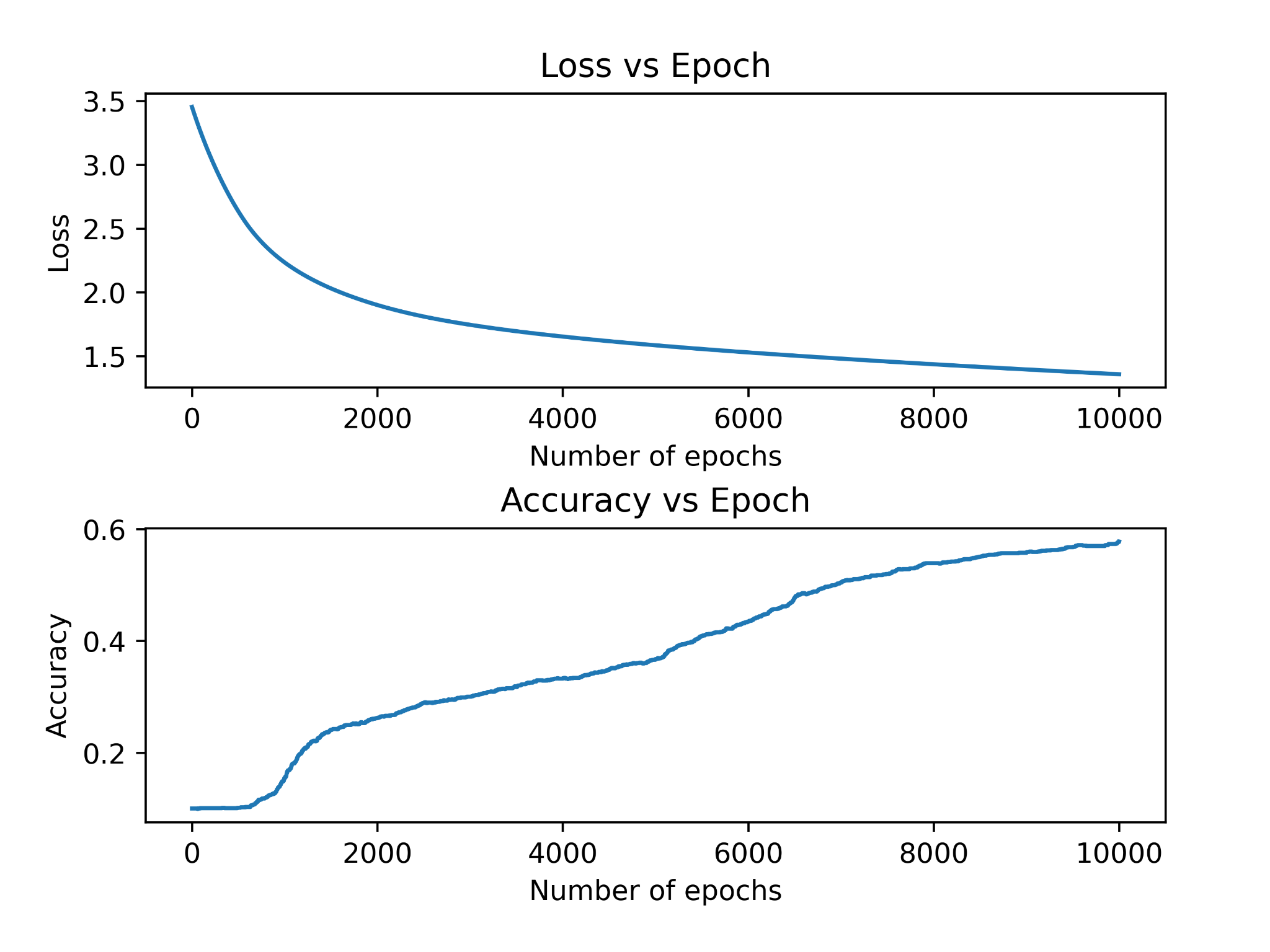
Training Accuracy for ANN with Two-layers: 0.777

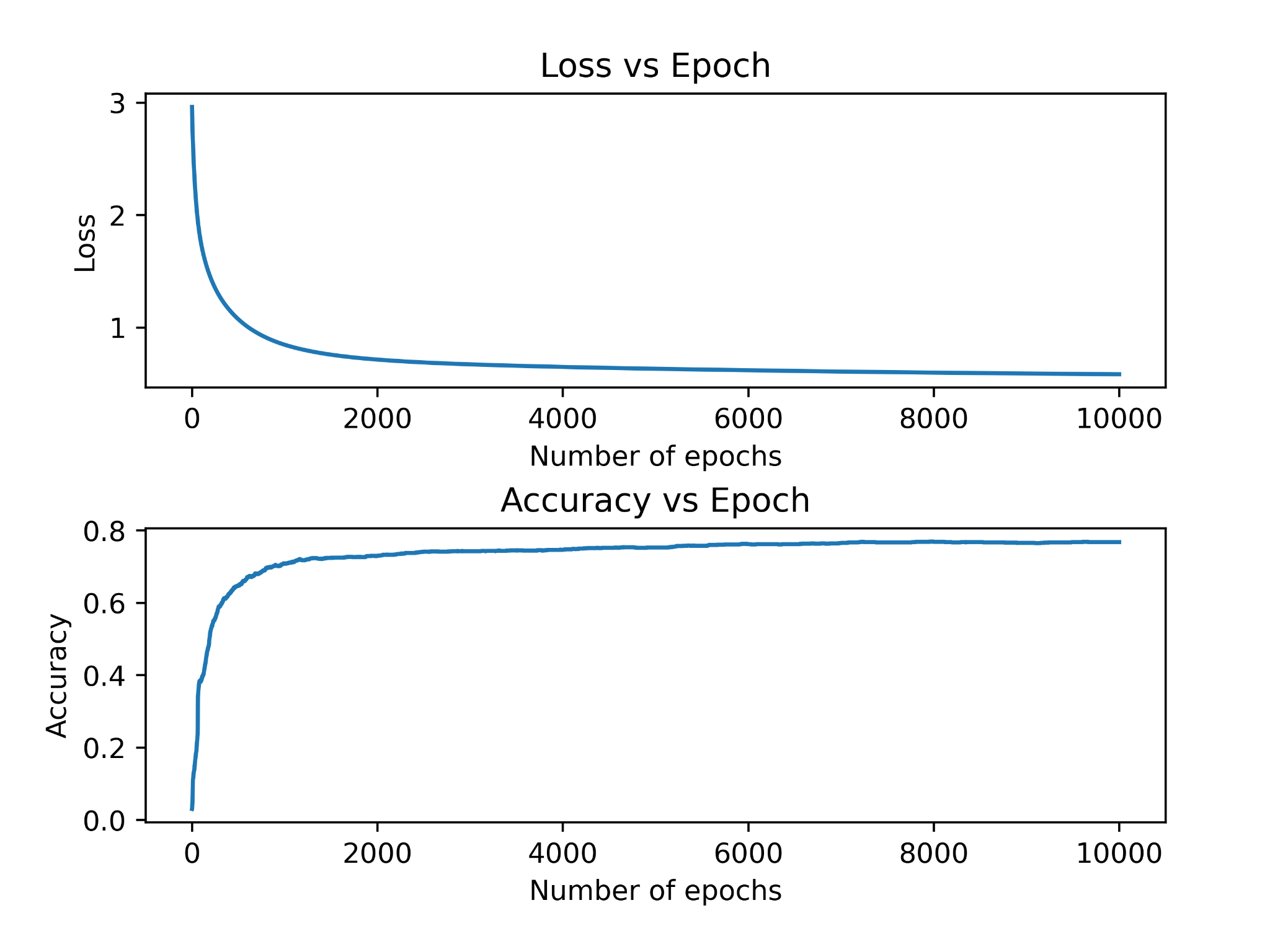
Testing Accuracy for ANN with Two-layers: 0.72

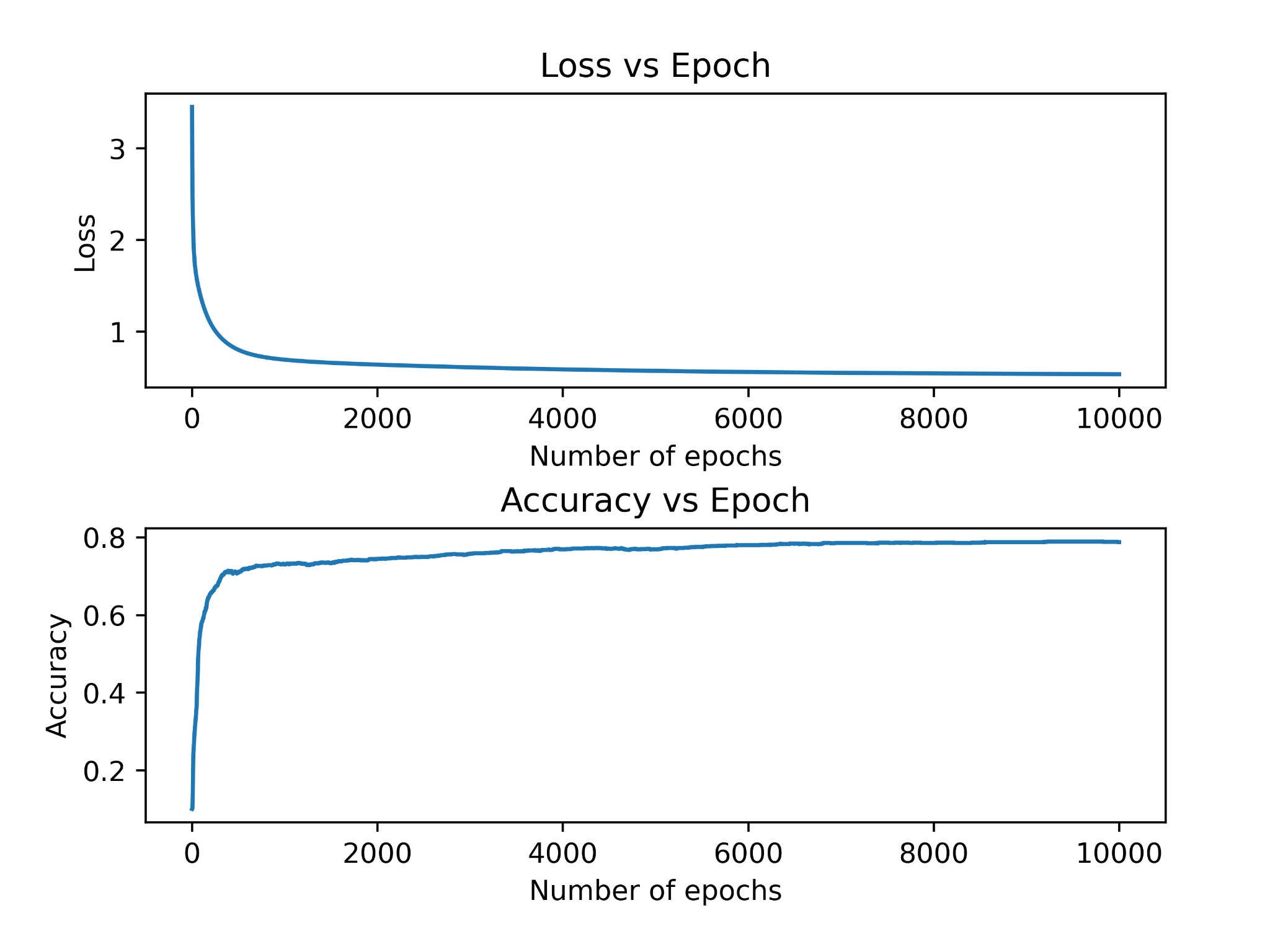
## Plots

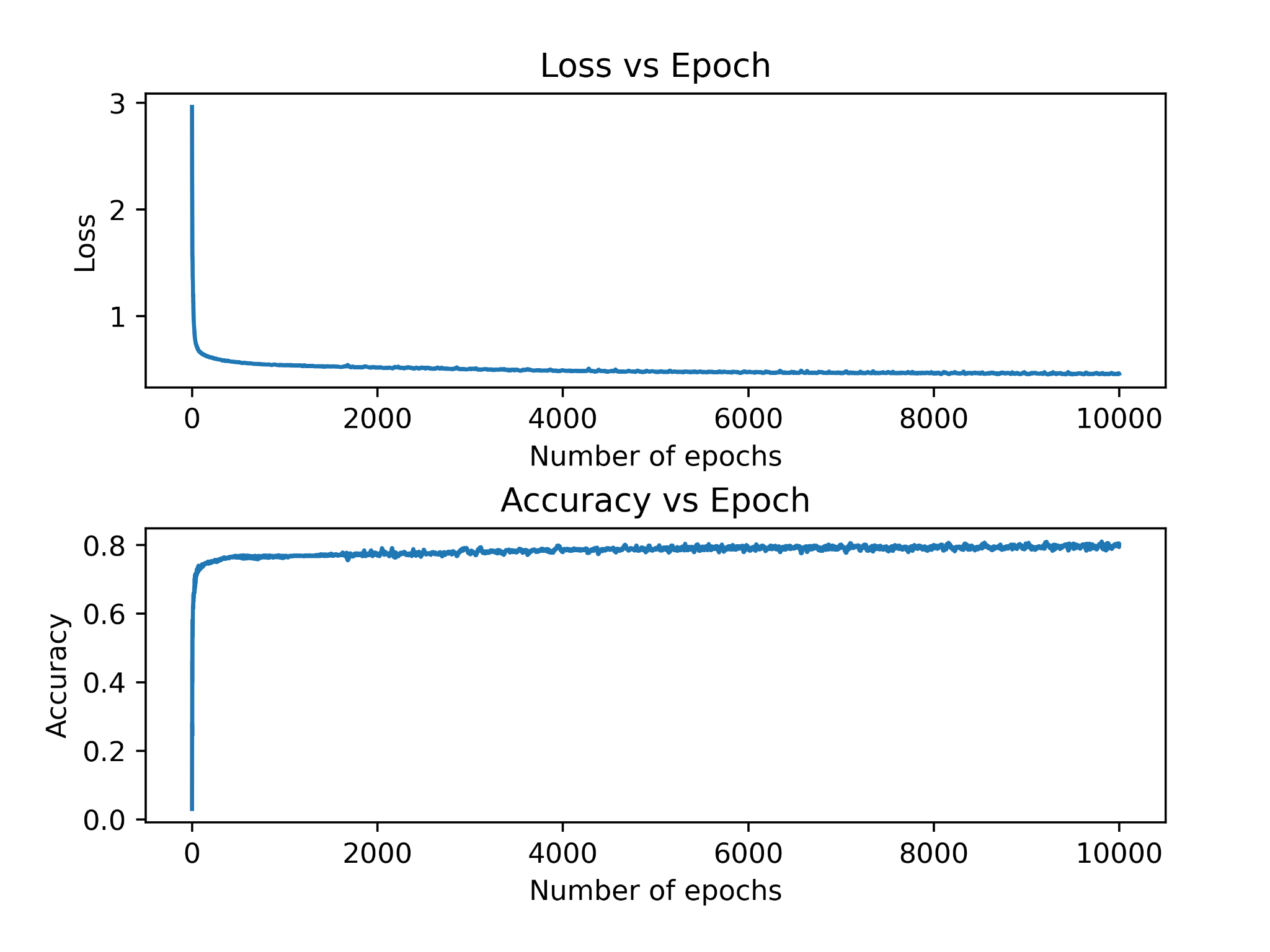
Plot of Accuracy for One-layer ANN with learning rate = 0.0001

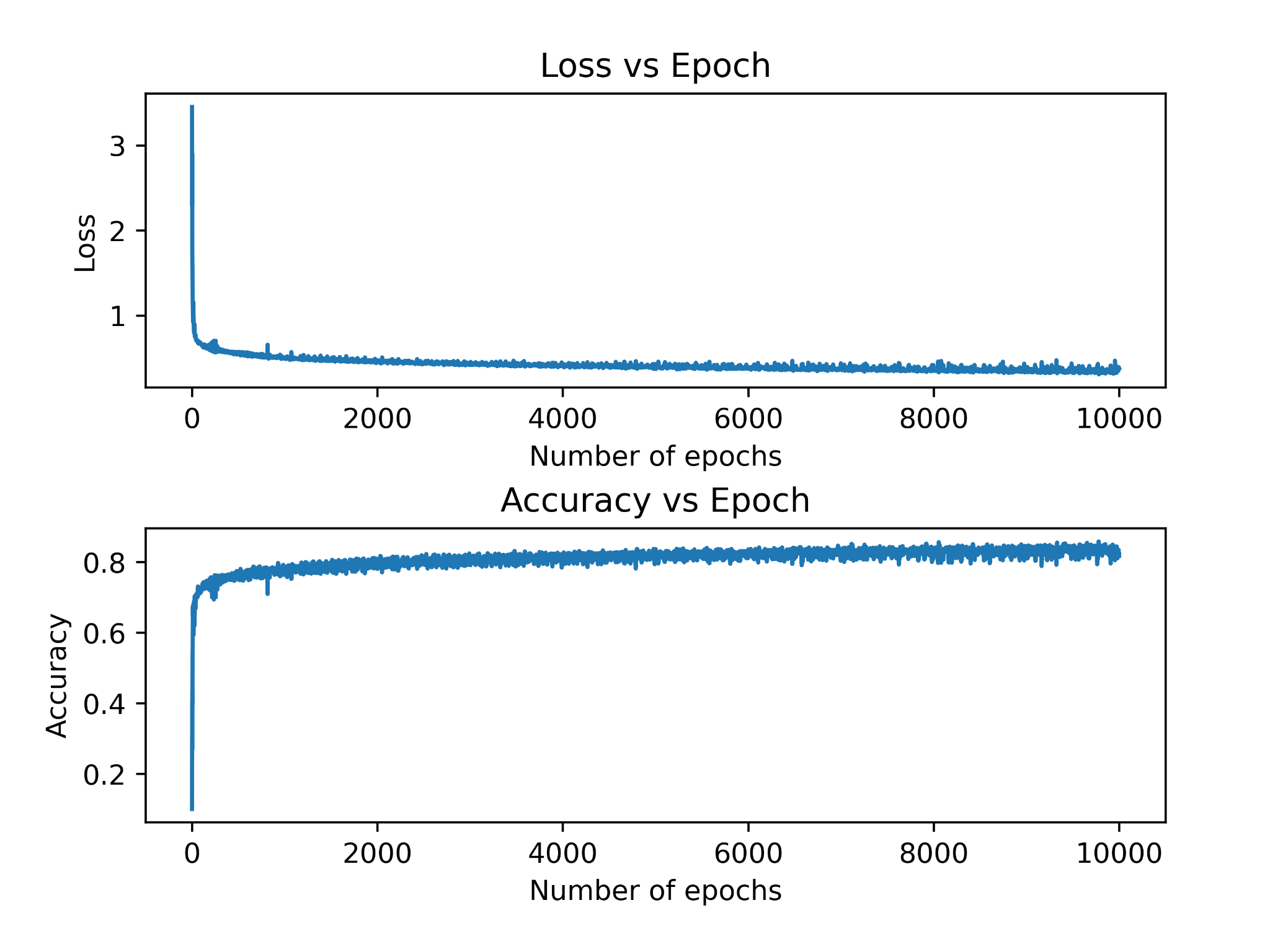


Plot of Accuracy for Two-layer ANN with learning rate = 0.0001  


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

# Comprehensive Comparison

The following were the results obtained after running 7-fold cross validation on each of the ML models:

### Fisher's Linear Discriminant

7-fold Cross Validation Results:

Test fold 1 : Accuracy = 0.9838709677419355  
Test fold 2 : Accuracy = 0.9879032258064516  
Test fold 3 : Accuracy = 0.9857038123167156  
Test fold 4 : Accuracy = 0.9866202346041055  
Test fold 5 : Accuracy = 0.9868035190615836  
Test fold 6 : Accuracy = 0.9857038123167156  
Test fold 7 : Accuracy = 0.9873533724340176

Mean of accuracies = 0.986279849183075

Std Dev. of accuracies = 0.001233128596493142

### Linear Perceptron

7-fold Cross Validation Results:

Test fold 1 : Accuracy = 0.9272360703812317  
Test fold 2 : Accuracy = 0.9627932551319648  
Test fold 3 : Accuracy = 0.9831378299120235  
Test fold 4 : Accuracy = 0.9767228739002932  
Test fold 5 : Accuracy = 0.9657258064516129  
Test fold 6 : Accuracy = 0.9525293255131965  
Test fold 7 : Accuracy = 0.8629032258064516

Mean of accuracies = 0.9472926267281105

Std Dev. of accuracies = 0.03832628544134992

### Naive Bayes

7-fold Cross Validation Results:

Test fold 1 : Accuracy = 0.9741568914956011  
Test fold 2 : Accuracy = 0.9772727272727273  
Test fold 3 : Accuracy = 0.9759897360703812  
Test fold 4 : Accuracy = 0.9772727272727273  
Test fold 5 : Accuracy = 0.9778225806451613  
Test fold 6 : Accuracy = 0.9761730205278593  
Test fold 7 : Accuracy = 0.9796554252199413

Mean of accuracies = 0.9769061583577712

Std Dev. of accuracies = 0.001585777537675608

### Logistic Regression

7-fold Cross Validation Results:

Test fold 1 : Accuracy = 0.9869868035190615  
Test fold 2 : Accuracy = 0.9886363636363636  
Test fold 3 : Accuracy = 0.9884530791788856  
Test fold 4 : Accuracy = 0.9897360703812317  
Test fold 5 : Accuracy = 0.9869868035190615  
Test fold 6 : Accuracy = 0.9869868035190615  
Test fold 7 : Accuracy = 0.9884530791788856

Mean of accuracies = 0.9880341432760787

Std Dev. of accuracies = 0.00099290349946894

### Artificial Neural Networks

7-fold Cross Validation Results:

Test fold 1 : Accuracy = 0.9803885630498533  
Test fold 2 : Accuracy = 0.9886363636363636

Test fold 3 : Accuracy = 0.9884530791788856  
Test fold 4 : Accuracy = 0.9897360703812317  
Test fold 5 : Accuracy = 0.9869868035190615  
Test fold 6 : Accuracy = 0.9869868035190615  
Test fold 7 : Accuracy = 0.9884530791788856

Mean of accuracies = 0.9810169669040638

Std Dev. of accuracies = 0.0020756096755976607

### Support Vector Machines

7-fold Cross Validation Results:

Test fold 1 : Accuracy = 0.9864369501466276  
Test fold 2 : Accuracy = 0.9888196480938416  
Test fold 3 : Accuracy = 0.9891862170087976  
Test fold 4 : Accuracy = 0.9902859237536656  
Test fold 5 : Accuracy = 0.9879032258064516  
Test fold 6 : Accuracy = 0.9875366568914956  
Test fold 7 : Accuracy = 0.9897360703812317

Mean of accuracies = 0.9885578131545874

Std Dev. of accuracies = 0.0012425440424910456

## Analysis

It is observed that while all the models display quite high levels of mean accuracy, the variance is the lowest in Logistic Regression, followed by SVM and then Naive Bayes.

SVM works well with unstructured and semi-structured data like text and images while logistic regression works with already identified independent variables.

Thus, Logistic Regression could be said to be the better performer for the given data set. Possible reasons for this can be an absence of multicollinearity, or a very low value of multicollinearity, in the dataset, as logistic regression works very well in such conditions.

## Box-Plots

