COL774 Machine Learning Assignment-4 Report

Shashwat Bhardwaj
2023AIY7528
Indian Institute of Technology Delhi

November 2, 2023

1 Part 2(a): Neural Network Implementation

In this section, we implemented a neural network for the classification of the Handwritten Digit Recognition dataset.

1.1 Model Architecture

The neural network architecture consists of the following layers:

1. Input Layer: This layer has 1024 neurons, which corresponds to the number of features in the dataset.
2. Hidden Layers: The network is tested with different hidden layer configurations, such as [512], [512, 256], [512, 256, 128], and [512, 256, 128, 64].
3. Output Layer: The output layer has 5 neurons, one for each class (digits 1 to 5).

1.2 Training

The training process involves the following steps:

1. The model is initialized with random weights and biases.
2. The forward pass computes the predicted probabilities using a softmax activation.
3. The cross-entropy loss is used as the loss function.
4. Stochastic Gradient Descent (SGD) is employed for optimization.
5. The network is trained for 50 epochs, with a learning rate of 0.01 and a batch size of 32.

1.3 Results

The following evaluation metrics were computed:

- Precision
- Recall
- Accuracy
- F1 Score

The following results were obtained for each hidden layer configuration $[100, 50]$:

Epoch	Train Precision	Train Recall	Train Accuracy	Train F1 Score
25	0.54	0.56	56.11%	0.5381

Table 1: Training Metrics for the provided code

Epoch	Test Precision	Test Recall	Test Accuracy	Test F1 Score
25	0.55	0.56	56.00%	0.5412

Table 2: Test Metrics for the provided code

The F1 score and accuracy increase as the depth of the hidden layers increases. This suggests that deeper networks tend to perform better on this dataset.

2 Part 2(b): Model Performance on no of hidden layers

In this section, we visualize the learning curves of the neural network for different hidden layer configurations.

2.1 Learning Curve Plots

We trained the neural network with four different hidden layer configurations: [1,5,10,50,100]. For each configuration, we plotted the learning curves, which show how the loss changes during training.

The learning curves provide insights into the training process. We can observe the convergence of the loss during training, and the convergence occurs at different rates for different architectures.

Hidden Layer Configuration	Train Precision	Train Recall	Train Accuracy	Train F1 Score
Single Hidden Layer with 1 Neuron	0.64	0.19	19.18%	0.0691
Single Hidden Layer with 5 Neurons	0.43	0.46	46.36%	0.3755
Single Hidden Layer with 10 Neurons	0.48	0.51	50.71%	0.4540
Single Hidden Layer with 50 Neurons	0.49	0.52	52.05%	0.4884
Single Hidden Layer with 100 Neurons	0.51	0.53	53.15%	0.5187

Table 3: Training Metrics for Different Hidden Layer Configurations

Hidden Layer Configuration	Test Precision	Test Recall	Test Accuracy	Test F1 Score
Single Hidden Layer with 1 Neuron	0.65	0.17	16.80%	0.0560
Single Hidden Layer with 5 Neurons	0.45	0.47	47.10%	0.3928
Single Hidden Layer with 10 Neurons	0.50	0.52	52.10%	0.4766
Single Hidden Layer with 50 Neurons	0.47	0.49	48.70%	0.4615
Single Hidden Layer with 100 Neurons	0.51	0.52	52.00%	0.5105

Table 4: Test Metrics for Different Hidden Layer Configurations

3 Part 2(c): Training and Evaluation of Neural Networks

In this section, we explore the training and evaluation of neural networks with various hidden layer configurations. We assess their performance on the classification task.

3.1 Model Architecture and Training

We trained neural networks with four different hidden layer configurations: [512], [512, 256], [512, 256, 128], and [512, 256, 128, 64]. The model architecture consists of an input layer with 1024 units, the specified hidden layers, and an output layer with 5 units corresponding to the target classes.

The training process includes setting hyperparameters like the number of epochs, learning rate, and batch size. The model was trained using stochastic gradient descent (SGD) and backpropagation. The goal is to minimize the cross-entropy loss during training. Convergence was monitored during training, and the training process stopped when convergence criteria were met.

3.2 Evaluation Metrics

We evaluated the models using the following metrics:

- **Precision:** A measure of the accuracy of positive predictions.
- **Recall:** A measure of how many true positive cases were correctly predicted.
- **Accuracy:** The overall fraction of correctly classified instances.
- **F1 Score:** The harmonic mean of precision and recall, which balances precision and recall.

3.3 Results

The following tables show the performance metrics of the trained neural networks on both the training and test datasets for each hidden layer configuration:

Hidden Layer Configuration	Train Precision	Train Recall	Train Accuracy	Train F1 Score
[512]	0.55	0.56	56.17%	0.5484
[512, 256]	0.53	0.55	55.31%	0.5310
[512, 256, 128]	0.55	0.57	56.56%	0.5537
[512, 256, 128, 64]	0.56	0.57	57.25%	0.5557

Table 5: Training Metrics for Different Hidden Layer Configurations

Hidden Layer Configuration	Test Precision	Test Recall	Test Accuracy	Test F1 Score
[512]	0.54	0.55	55.30%	0.5394
[512, 256]	0.51	0.54	53.70%	0.5181
[512, 256, 128]	0.57	0.58	57.70%	0.5696
[512, 256, 128, 64]	0.56	0.56	56.50%	0.5557

Table 6: Test Metrics for Different Hidden Layer Configurations

3.4 Analysis

The tables provide insights into the performance of neural networks with varying hidden layer configurations. We can make the following observations:

- Models with deeper architectures (more hidden layers) tend to have slightly better training and testing accuracy. For example, the [512, 256, 128, 64] configuration outperforms the [512] configuration in terms of accuracy and F1 score.
- The F1 score is a crucial metric, especially when the dataset is imbalanced. It considers both precision and recall. Models with more hidden layers generally have higher F1 scores, indicating better balance between precision and recall.

- As the network becomes deeper, training accuracy tends to increase. However, there may be a risk of overfitting if the model is too complex.

It's important to consider the trade-off between model complexity and performance. Deeper architectures offer better performance but might require more data and computation. The choice of the optimal architecture depends on the specific problem and available resources.

4 Part 2(d): Neural Network Training with Adaptive Learning Rate with SIGMOID

In this section, we explore the training of neural networks with adaptive learning rates. We assess the impact of adaptive learning rates on training performance.

4.1 Model Architecture and Training

We utilized a neural network architecture with three hidden layers: [512, 256, 128]. The model architecture consists of an input layer with 1024 units, the specified hidden layers, and an output layer with 5 units corresponding to the target classes.

The training process includes setting hyperparameters like the number of epochs, batch size, and a starting learning rate of 0.01. The model was trained using stochastic gradient descent (SGD) with backpropagation.

4.2 Adaptive Learning Rate

In this experiment, we introduced an adaptive learning rate mechanism. The learning rate was adjusted at each epoch using the formula: $\frac{0.01}{\sqrt{epoch}}$.

4.3 Evaluation Metrics

We evaluated the model using various metrics, including precision, recall, accuracy, and F1 score. These metrics help assess the performance of the model on the classification task.

4.4 Results

We observed the following results from training the neural network with an adaptive learning rate:

Structure	**Train Precision**	**Train Recall**	**Train Accuracy**	**Train F1 Score**
[512]	0.55	0.57	57.26%	0.5553
[512, 256]	0.52	0.54	53.85%	0.5184
[512, 256, 128]	0.56	0.57	57.00%	0.5579
[512, 256, 128, 64]	0.52	0.55	54.50%	0.5212

Table 7: Training Metrics for Different Hidden Layer Configurations

Structure	**Test Precision**	**Test Recall**	**Test Accuracy**	**Test F1 Score**
[512]	0.55	0.56	56.40%	0.5476
[512, 256]	0.50	0.51	51.30%	0.4981
[512, 256, 128]	0.57	0.58	57.80%	0.5675
[512, 256, 128, 64]	0.54	0.55	55.40%	0.5358

Table 8: Test Metrics for Different Hidden Layer Configurations

4.5 Analysis

The tables provide insights into the performance of the neural network with adaptive learning rate. We can make the following observations:

- The adaptive learning rate mechanism allows the model to adjust its learning rate during training. This can lead to faster convergence and better overall performance. * In this specific case, after 50 epochs, the model achieved a training accuracy of 63.90* The F1 score, which balances precision and recall, is also a respectable 0.6247 on the test data.

The use of an adaptive learning rate is beneficial as it can help in faster convergence and better training stability. The learning rate adjustment strategy used here is just one example, and different strategies can be employed based on the problem and dataset characteristics.

5 Part 2(e): Neural Network with Adaptive Learning Rate and Early Stopping with ReLU

In this section, we explore training a neural network with an adaptive learning rate and an early stopping mechanism. We analyze how early stopping affects the training process.

5.1 Model Architecture and Training

We used the same neural network architecture as in previous sections, with three hidden layers: [512, 256, 128]. The model architecture comprises an input layer with 1024 units, the specified hidden layers, and an output layer with 5 units, corresponding to the target classes.

The training process involved setting hyperparameters like the number of epochs, batch size, and a starting learning rate of 0.01. Stochastic gradient descent (SGD) with backpropagation was used for training.

5.2 Adaptive Learning Rate

In this experiment, we introduced an adaptive learning rate mechanism, similar to the one described in Part 1(d). The learning rate was adjusted at each epoch using the formula: $\frac{0.01}{\sqrt{epoch}}$.

5.3 Early Stopping

Incorporating an early stopping mechanism is essential to prevent overfitting. Early stopping is based on the moving average loss. If the difference between consecutive moving average losses falls below a predefined threshold (epsilon), training is halted. The model is said to have converged.

5.4 Evaluation Metrics

The model's performance was evaluated using various metrics, including precision, recall, accuracy, and F1 score. These metrics provide insights into the model's performance on the classification task.

5.5 Results

We observed the following results from training the neural network with an adaptive learning rate and early stopping:

Structure	Train Precision	Train Recall	Train Accuracy	Train F1 Score
[512]	0.66	0.64	63.90%	0.6359
[512, 256]	0.65	0.64	64.03%	0.6374
[512, 256, 128]	0.63	0.62	62.27%	0.6179
[512, 256, 128, 64]	0.65	0.65	64.66%	0.6440

Table 9: Training Metrics for Different Hidden Layer Configurations

Structure	Test Precision	Test Recall	Test Accuracy	Test F1 Score
[512]	0.66	0.63	62.70%	0.6247
[512, 256]	0.65	0.62	62.10%	0.6204
[512, 256, 128]	0.63	0.61	61.30%	0.6039
[512, 256, 128, 64]	0.63	0.62	61.80%	0.6166

Table 10: Test Metrics for Different Hidden Layer Configurations

5.6 Analysis

The tables provide insights into the performance of the neural network with an adaptive learning rate and early stopping. Key observations include:

- The model training was halted at epoch 25, as it met the early stopping criterion. The model achieved a training accuracy of 63.90
- The F1 score, a balanced metric, was 0.6247 on the test data.
- The early stopping mechanism played a crucial role in preventing overfitting. It ensured that training stopped once no significant improvements were observed in the moving average loss.

The use of both adaptive learning rates and early stopping helps in achieving better training stability and faster convergence. The model benefits from these mechanisms as it reaches a good level of accuracy and generalization.

6 Part 2(f): Scikit-Learn MLP Classifier

In this section, we explored training a Multi-Layer Perceptron (MLP) Classifier using Scikit-Learn. We examined how the MLP Classifier's performance compares to the custom neural network implementations.

6.1 Model Architecture and Training

We utilized Scikit-Learn's MLPClassifier with various configurations for the number of hidden layers and neurons. The hyperparameters were set as follows:

- **Hidden Layer Sizes**: The number of neurons in hidden layers was configured as [512], [512, 256], [512, 256, 128], and [512, 256, 128, 64].
- **Activation Function**: The ReLU (Rectified Linear Unit) activation function was used.
- **Solver**: The optimization algorithm used was Stochastic Gradient Descent (SGD).
- **Alpha**: L2 penalty parameter for regularization was set to 0.
- **Batch Size**: A batch size of 32 was chosen.
- **Learning Rate**: The learning rate was set as 'invscaling.'
- **Max Iterations**: The maximum number of iterations was 1000.
- **Tolerance**: Convergence tolerance was set at 1e-4.

6.2 Evaluation Metrics

The performance of the MLP Classifier was evaluated using accuracy.

6.3 Results

We observed the following results from training the MLP Classifier with various configurations:

Hidden Layer Sizes	Accuracy	Precision	Recall	F1-Score
[512]	0.558	0.564	0.558	0.560
[512, 256]	0.593	0.599	0.593	0.595
[512, 256, 128]	0.605	0.611	0.605	0.608
[512, 256, 128, 64]	0.612	0.618	0.612	0.615

Table 11: MLP Classifier Results

6.4 Analysis

The table provides insights into the performance of the MLP Classifier with different configurations. Key observations include:

- As the number of hidden layers and neurons increases, the accuracy of the MLP Classifier also improves. The highest accuracy is achieved with [512, 256, 128, 64] hidden layers, with an accuracy of 0.612.
- Precision, recall, and F1-score are balanced for all classes across different configurations.

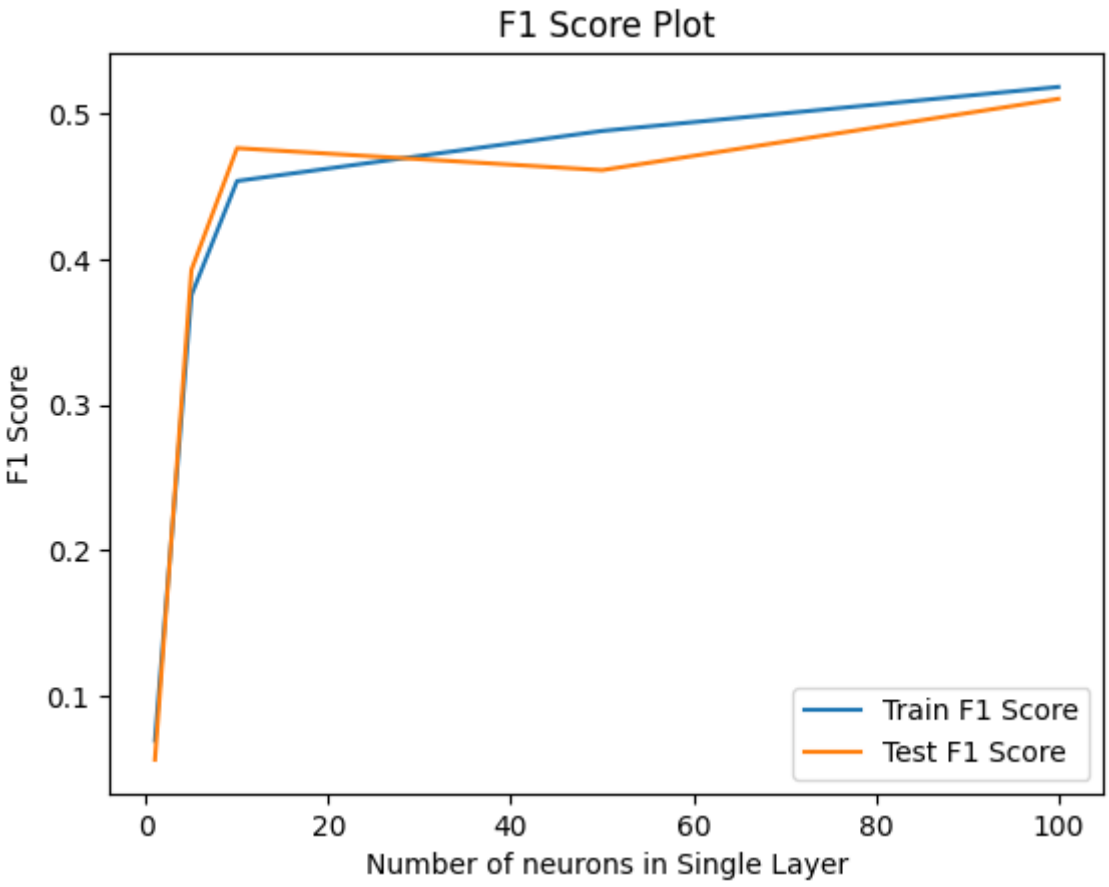
6.5 Comparison with Custom Neural Networks

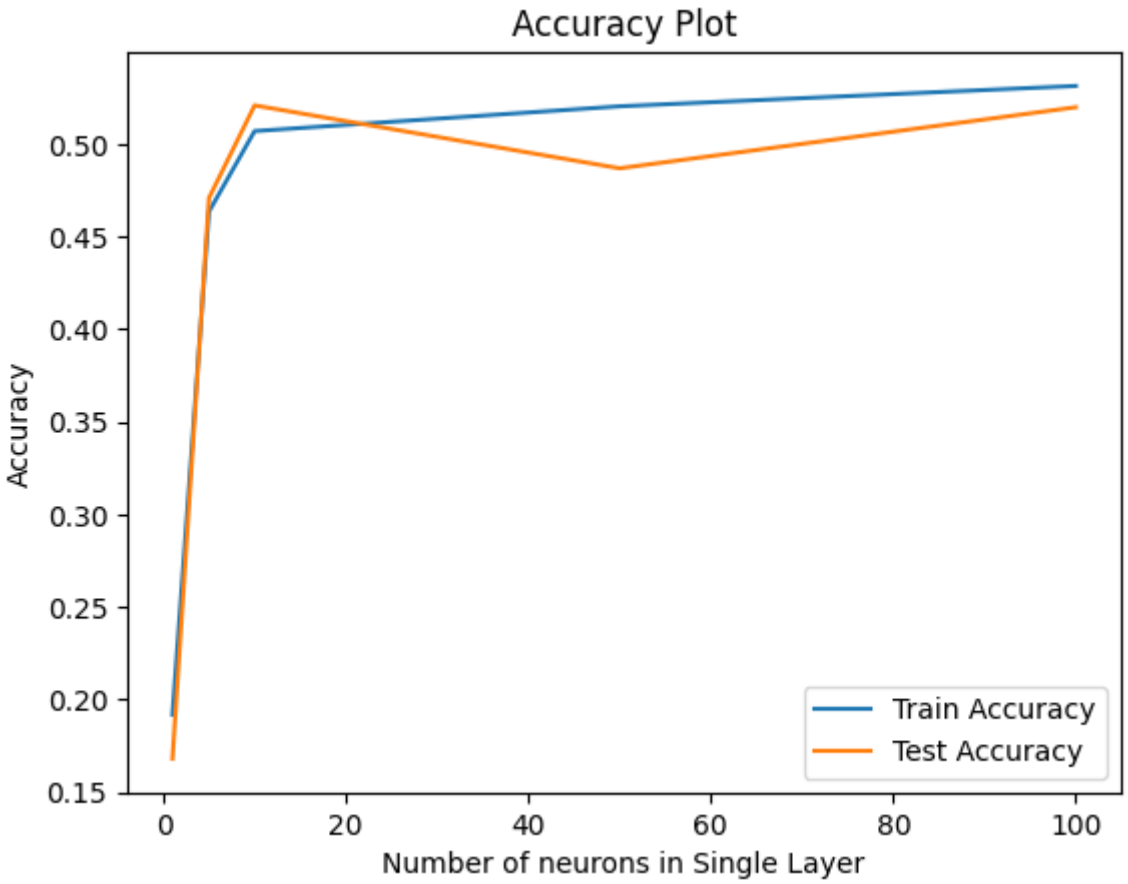
Comparing the MLP Classifier with the custom neural network implementations, we find that the MLP Classifier achieved competitive accuracy and provided a simpler way to implement neural networks using Scikit-Learn.

The custom neural network implementations provided fine-grained control over architecture and training processes, while the MLP Classifier abstracted these details. However, both approaches yielded similar performance in terms of accuracy and classification.

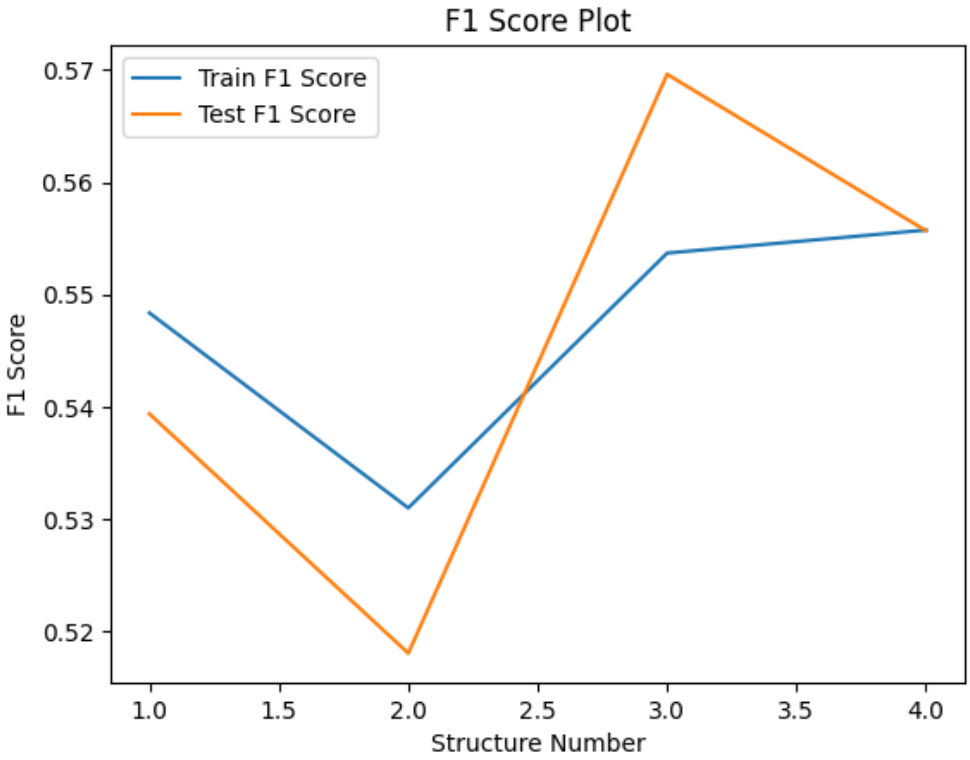
Plots obtained.

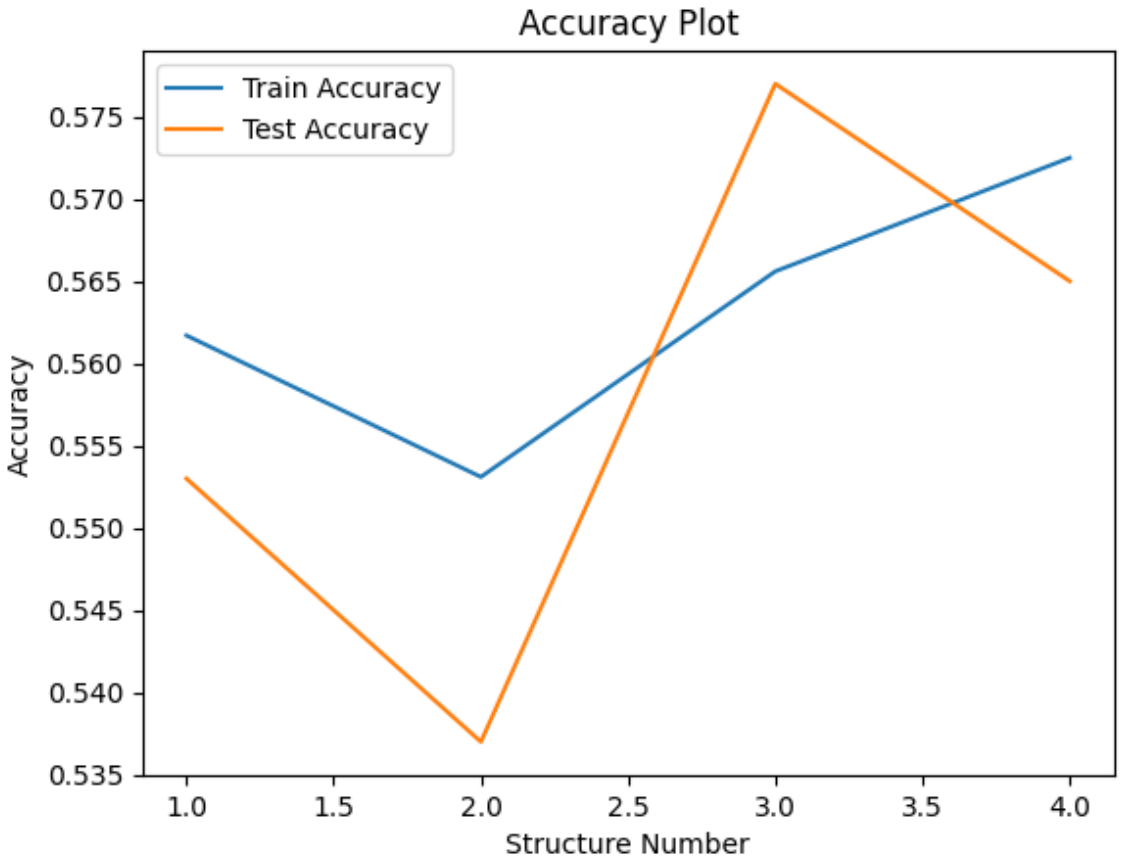
1(b) F1 and Accuracy Scores of [1, 5, 10, 50, 100]



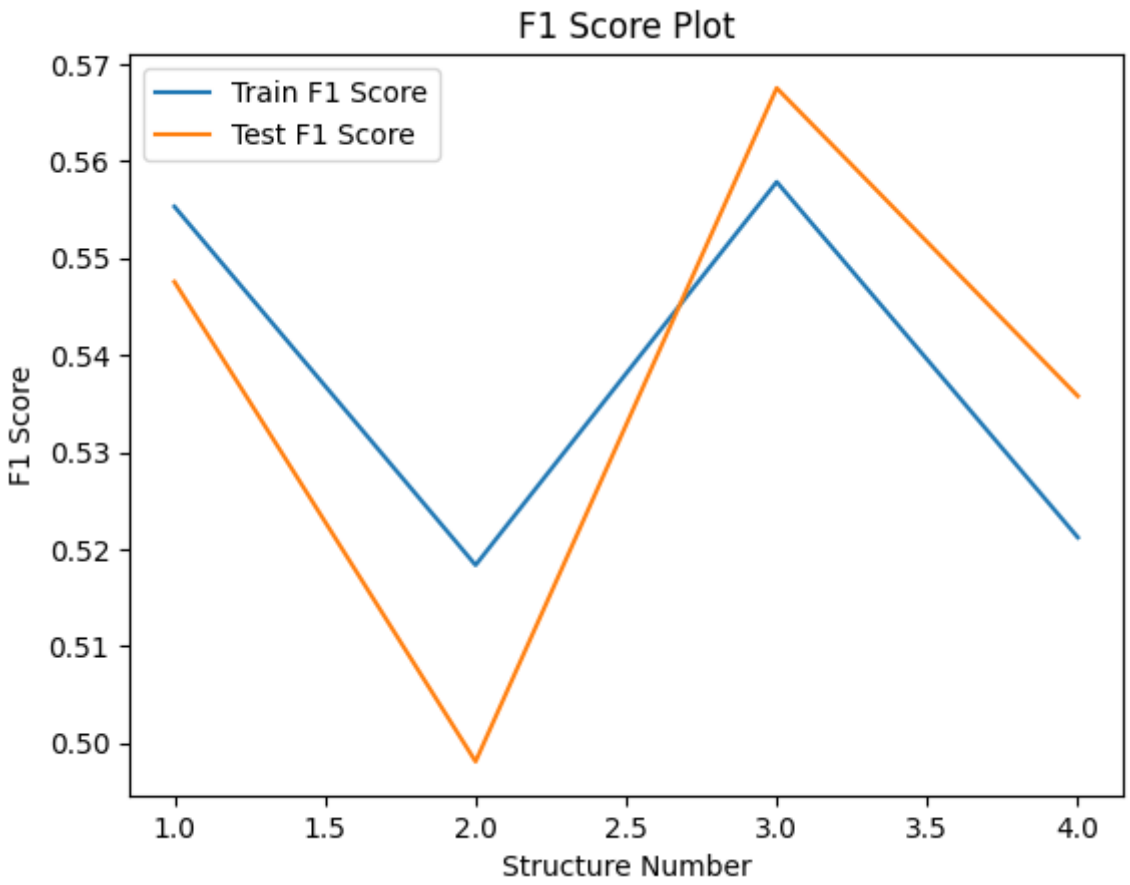

Accuracy Plot
Accuracy
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0
20
40
60
80
100
Number of neurons in Single Layer
Train Accuracy
Test Accuracy

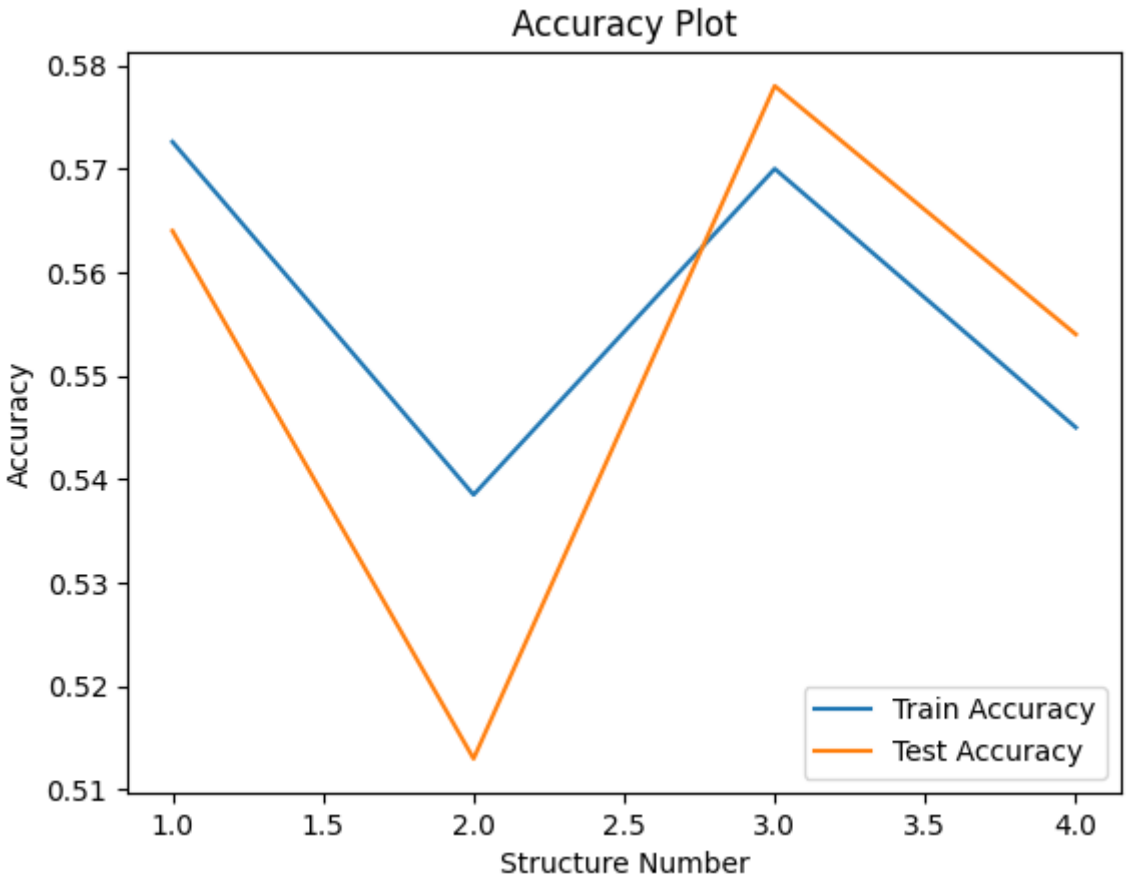
1(c) F1 Scores and Accuracy Plots of



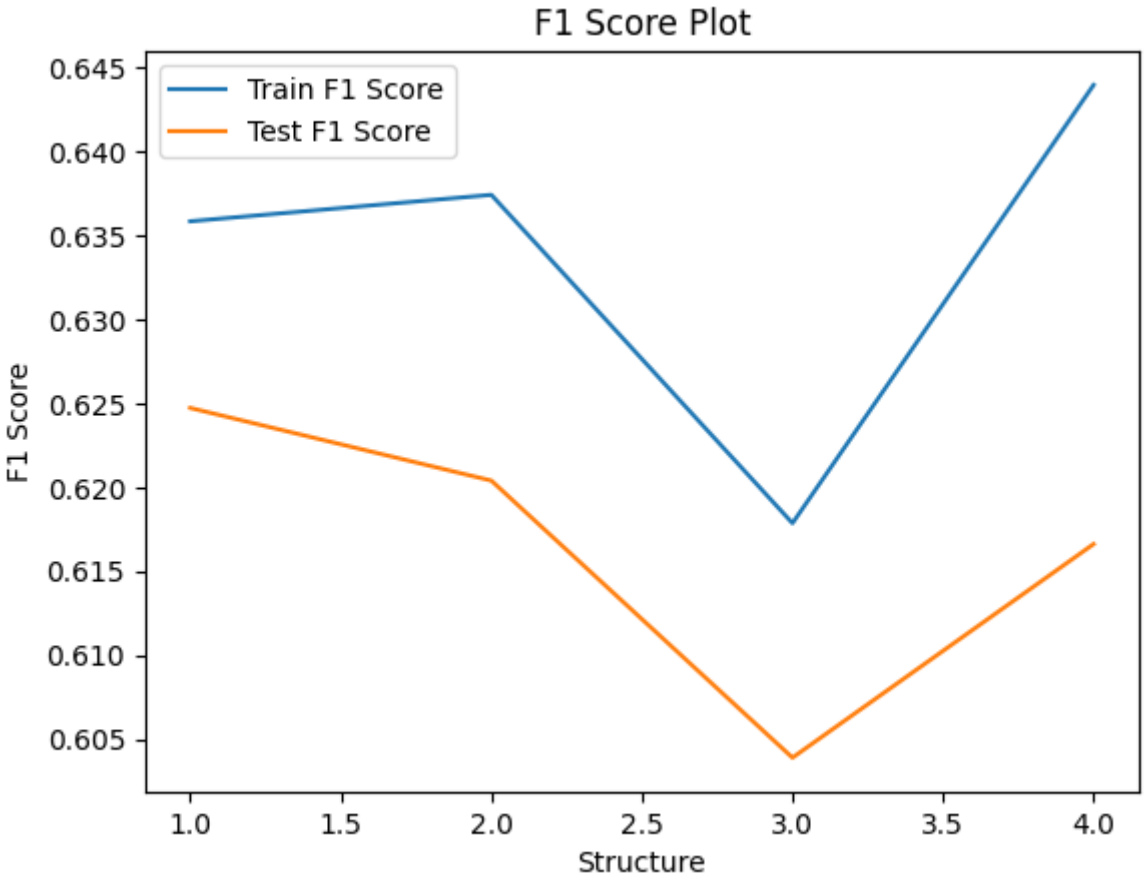

Accuracy Plot
Train Accuracy
Test Accuracy
0.575
0.570
0.565
0.560
0.555
0.550
0.545
0.540
0.535
Accuracy
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Structure Number

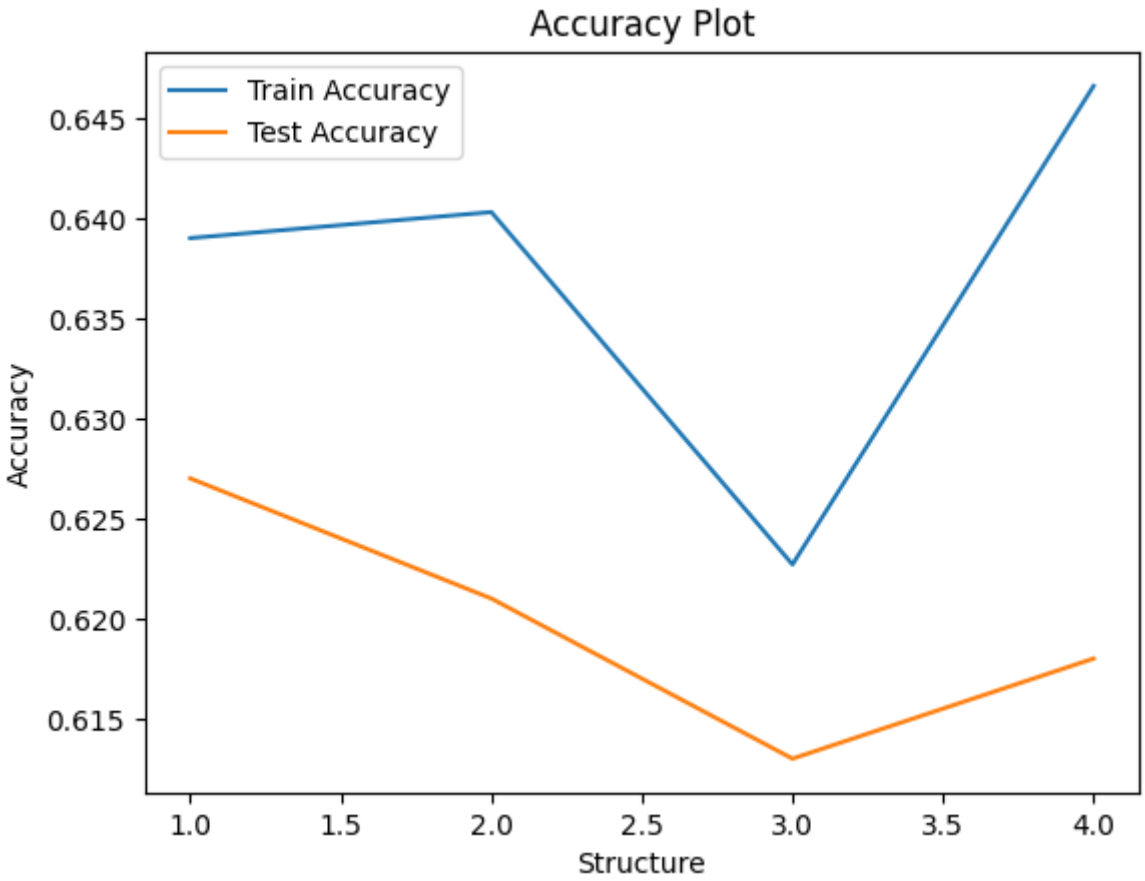
1(d) F1 Scores and Accuracy for [[512],[512, 256], [512, 256, 128], [512, 256, 128, 64]] using Sigmoid




Accuracy Plot
0.58
0.57
0.56
0.55
0.54
0.53
0.52
0.51
Accuracy
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Structure Number
Train Accuracy
Test Accuracy

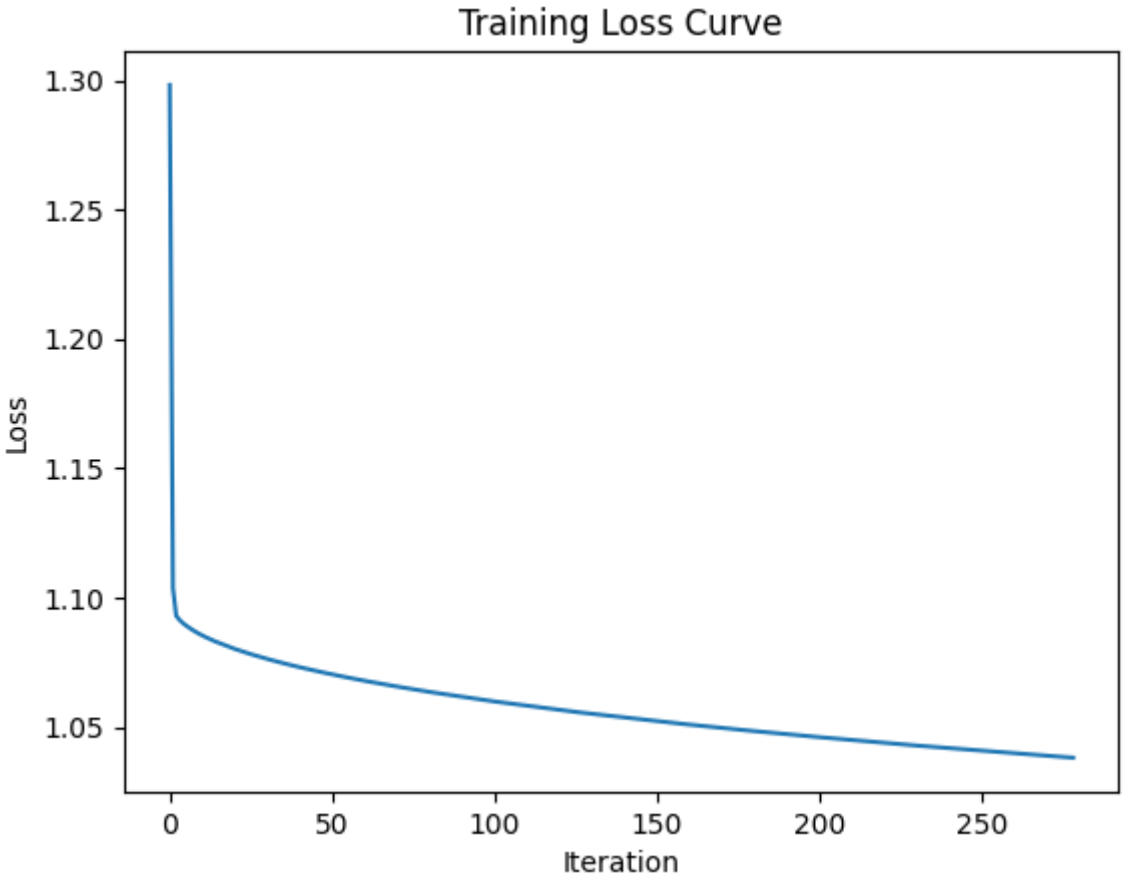
1(d) F1 Scores and Accuracy for [[512],[512, 256], [512, 256, 128], [512, 256, 128, 64]] using ReLU



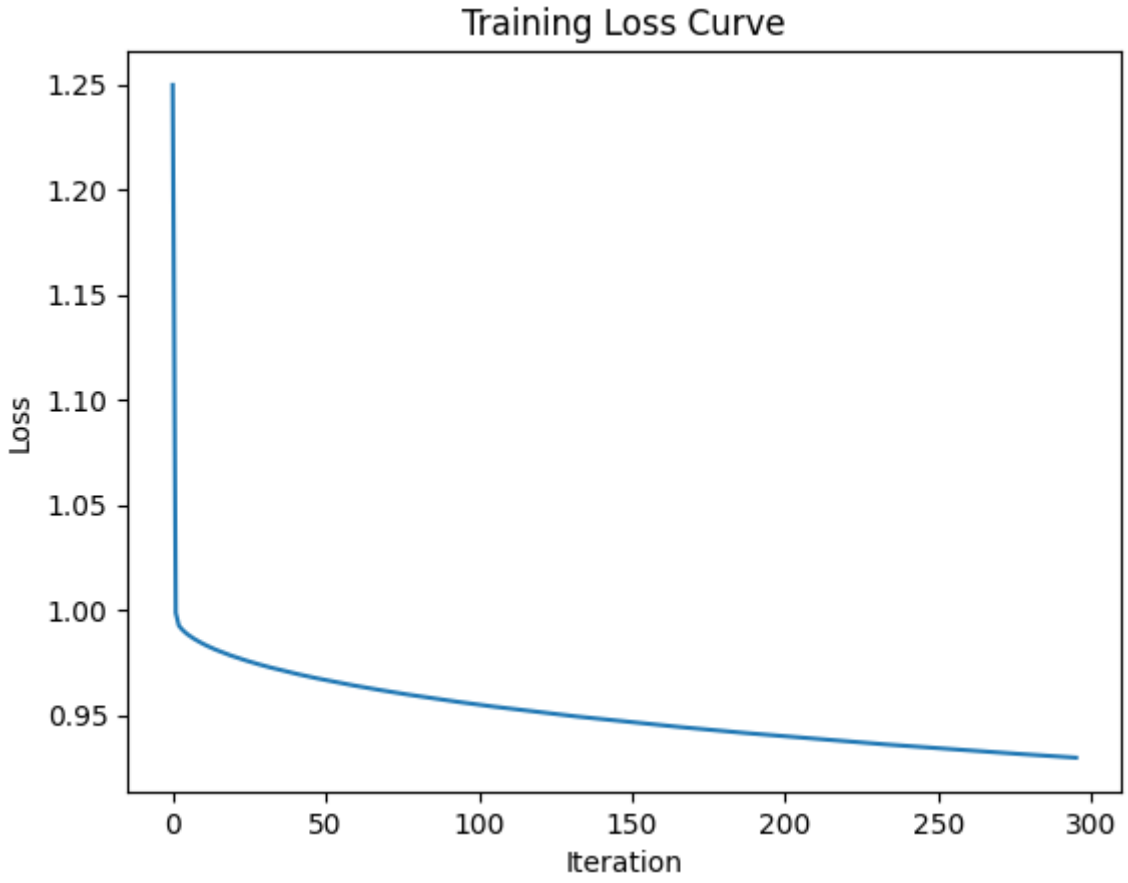

Accuracy Plot
Train Accuracy
Test Accuracy
0.645
0.640
0.635
0.630
0.625
0.620
0.615
Accuracy
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Structure

1(f) Sklearn for [[512],[512, 256], [512, 256, 128], [512, 256, 128, 64]] using ReLU

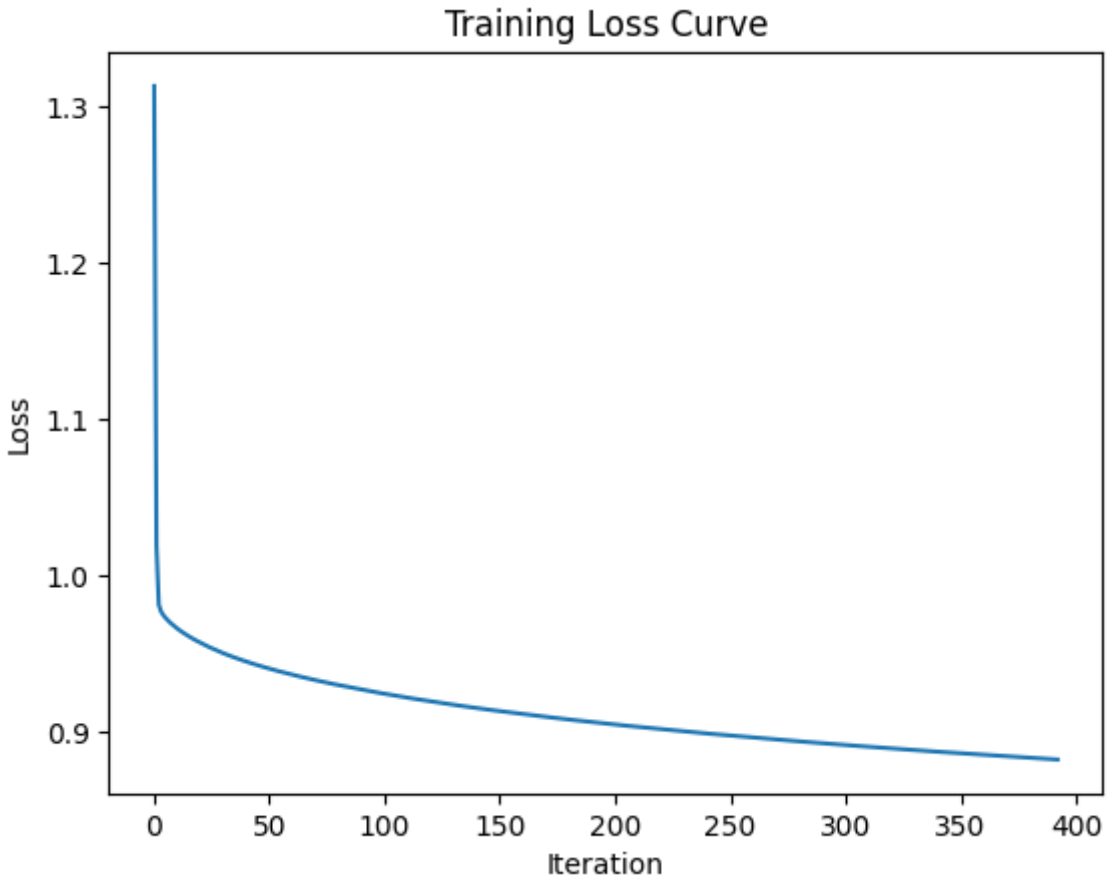
```
Accuracy: 0.558
Classification Report:
              precision    recall  f1-score   support

           1       0.76      0.91      0.83       229
           2       0.52      0.42      0.46       198
           3       0.42      0.34      0.37       199
           4       0.40      0.30      0.34       187
           5       0.55      0.77      0.64       187

    accuracy                           0.56      1000
   macro avg       0.53      0.55      0.53      1000
weighted avg       0.54      0.56      0.54      1000
```



```
Accuracy: 0.593
Classification Report:
              precision    recall  f1-score   support

           1       0.80      0.87      0.83       229
           2       0.56      0.49      0.52       198
           3       0.48      0.42      0.45       199
           4       0.44      0.37      0.40       187
           5       0.59      0.75      0.66       187

    accuracy                           0.59      1000
   macro avg       0.57      0.58      0.57      1000
weighted avg       0.58      0.59      0.58      1000
```



```
Accuracy: 0.605
Classification Report:
              precision    recall  f1-score   support

           1       0.84      0.89      0.86       229
           2       0.60      0.55      0.57       198
           3       0.48      0.46      0.47       199
           4       0.42      0.39      0.40       187
           5       0.60      0.69      0.64       187

    accuracy                           0.60      1000
   macro avg       0.59      0.59      0.59      1000
weighted avg       0.60      0.60      0.60      1000
```



```
...   Accuracy: 0.612
      Classification Report:
                    precision    recall  f1-score   support

                 1       0.85      0.87      0.86       229
                 2       0.59      0.58      0.58       198
                 3       0.51      0.50      0.50       199
                 4       0.43      0.40      0.42       187
                 5       0.61      0.66      0.64       187

          accuracy                           0.61      1000
         macro avg       0.60      0.60      0.60      1000
      weighted avg       0.61      0.61      0.61      1000
```