Table of Contents

❑ INTRODUCTION

➔ What is the Deep Neural Network?

A Deep Neural Network consists of one or more hidden layers it is also known as multi layers perceptron which is the most trivial form of neural network. Deep Neural Networks are preferred to work with unstructured data such as images, audio and video. They can be also used to handle structured (or tabular) data but most of the time this category of data is dealt with machine learning algorithms. The reason why Deep Neural Networks are so popular is that they can work with huge amounts of data

➔ MNIST Dataset

The dataset consists of 60000 handwritten images each of which is 28 * 28 pixels in size. All images are in grayscale format. We will use 10000 images to train and 1000 images to test your model.

❑ METHODOLOGY

1. Normalization

The original data set in a grayscale format in which each pixel values ranges between [0-255]. I normalized the dataset to bring all the feature values in the range [0-1].

2. Reshaping the data

The original dataset is in the form (60000, 28, 28). I converted all the (28, 28) images to a vector of 28*28=784 pixels to train a neural network.

3. Define structure of neural network

The network consists of an input layer, a hidden layer and an output layer. The input layers consist of 784 nodes for 784 features in each training example. For hidden layers, I will experiment with multiple numbers of nodes. And, finally the output layer consists of 10 nodes. Each output node represents a probability.

4. Initialize Parameters

Let us use n[0],n[1],n[2] to denote the number of nodes in each layer of the neural network. In my case n[0] = 784, and n[2] = 10. For the hidden layers I initialized the weights and bias matrices as $W^{[1]}$ and $b^{[1]}$. The size of $W^{[1]}$ is (n[1],n[0]) and size of $b^{[1]}$ is (n[1],1). For the output layer I will define parameters $W^{[2]}$ and $b^{[2]}$ of size (n[2],n[1]) and (n[2],1) respectively.

5. Training

In the forward propagation part I will first calculate output for the hidden layer. The equations will be

$$Z^{[1]} = W^{[1]} X + b^{[1]}$$
$$A^{[1]} = g(Z^{[1]})$$

For output layers I'll calculate the final output using result of the previous layers as

$$Z^{[2]} = W^{[2]} A^{[1]} + b^{[2]}$$
$$A^{[2]} = g(Z^{[2]})$$

Here g is the activation function, in this case it will be ReLU activation function.
In Backpropagation I will calculate the gradient of final output with respect to the initialized parameters in previous layers.
$A^{[2]}$ is the final output. The loss is defined as

$$\frac{dL}{dZ^{[2]}} = A^{[2]} - Y$$

Using the above formula we can calculate the gradients in the output and hidden layers. Gradient for the output layer

$$\frac{1}{[2]} \frac{dL}{dZ^{[2]}} [1]^T = \times$$

$$\frac{dL}{} = \frac{1}{} \times \sum \frac{dL}{} \quad db^{[2]}$$

Gradients for the hidden layer

$$_{j}\,dZ^{[1](j)}$$

$$= W^{[2]^T} \times \frac{dL}{dZ^{[2]}} \cdot \frac{[2]}{dZ}$$
[1]

$$\frac{dL}{dW^{[1]}} = \frac{1}{m} \times \frac{dL}{dZ^{[1]}} \times X^T$$

$$\frac{dL}{db^{[1]}} = \frac{1}{m} \times \sum_j \frac{dL}{dZ^{[1](j)}}$$

Lastly using the above equation you can update the parameters as follows

$$W^{[1]} = W^{[1]} - \alpha \times \frac{dL}{dW^{[1]}}$$

$$b^{[1]} = b^{[1]} - \alpha \times \frac{dL}{db^{[1]}}$$

$$W^{[2]} = W^{[2]} - \alpha \times \frac{dL}{dW^{[2]}}$$

$$b^{[2]} = b^{[2]} - \alpha \times \frac{dL}{db^{[2]}}$$

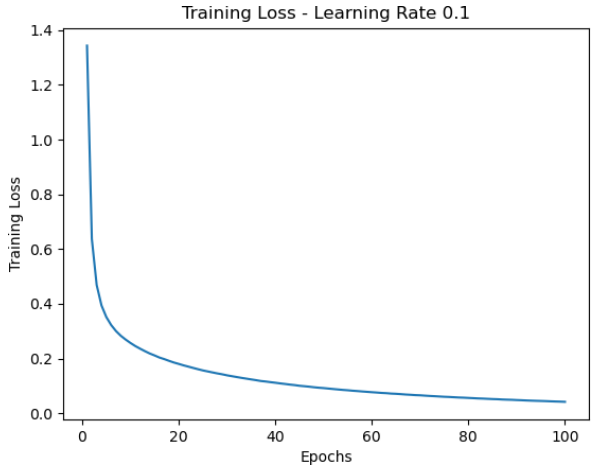
We will run the above code using multiple learning rates α and test with various nodes in the hidden layers.

❑ RESULTS:

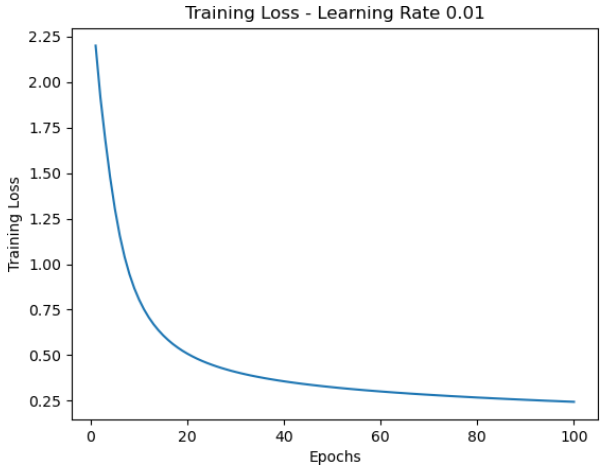
The overall accuracy of models run on binary images is lesser than the model's accuracy with grayscale images.

---------- **TASK 1** ----------

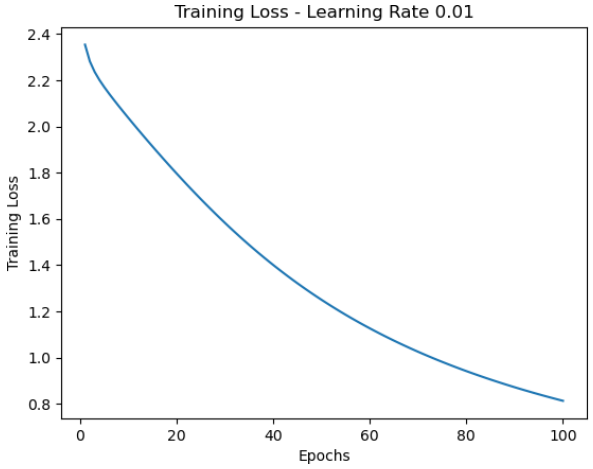
In task 1, the learning rate is 0.1, and the training is done for 100 epochs. The accuracy of this model is 93.3%.

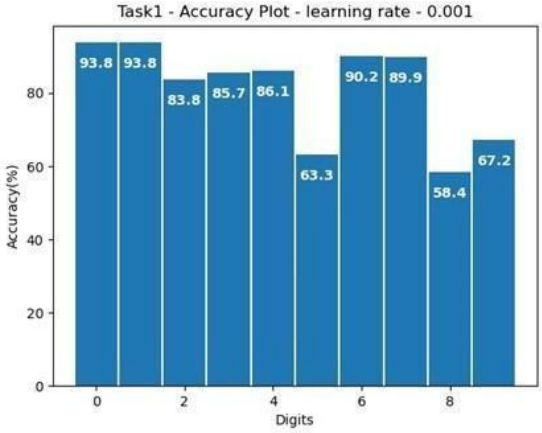


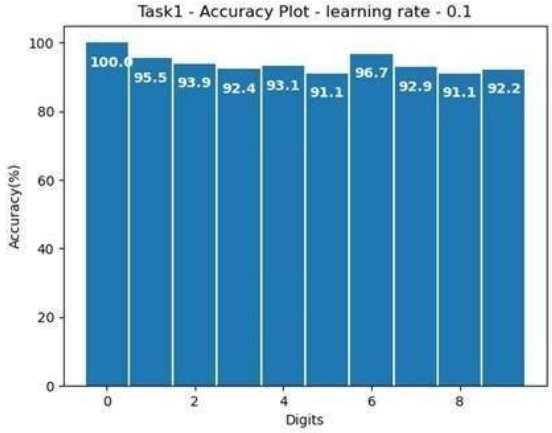
The second learning rate is 0.01. Decreasing the value of the learning rate decreased accuracy. The final accuracy is 90.1%.

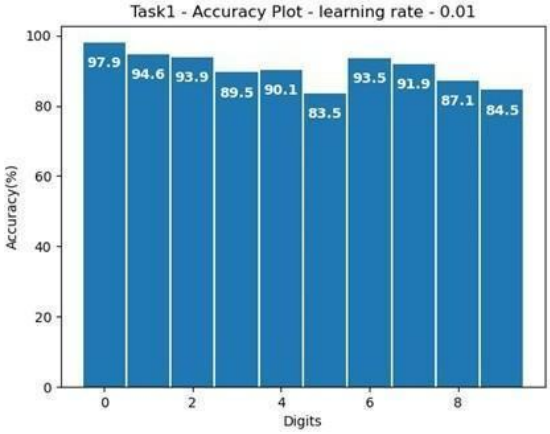


The third value of the learning parameter is 0.001. The model does not converge in 100 epochs, as shown in the graph below. The accuracy decreased to 80.8%.


Training Loss - Learning Rate 0.01
Training Loss
Epochs
2.4
2.2
2.0
1.8
1.6
1.4
1.2
1.0
0.8
0
20
40
60
80
100


Task1 - Accuracy Plot - learning rate - 0.001
Accuracy(%)
Digits
93.8
93.8
83.8
85.7
86.1
63.3
90.2
89.9
58.4
67.2


Task1 - Accuracy Plot - learning rate - 0.1
Accuracy(%)
Digits
100.0
95.5
93.9
92.4
93.1
91.1
96.7
92.9
91.1
92.2


Task1 - Accuracy Plot - learning rate - 0.01
Accuracy(%)
Digits
97.9
94.6
93.9
89.5
90.1
83.5
93.5
91.9
87.1
84.5

The percentage accuracy for each model with each learning rate are shown above. On the same test set each time the model gave different accuracy for different learning rates. For learning rate 0.1 each digit for correctly predicted more than 90%. For learning rates 0.01 the accuracy slightly decreased but seven out of ten digits are classified correctly more than 90% of the time

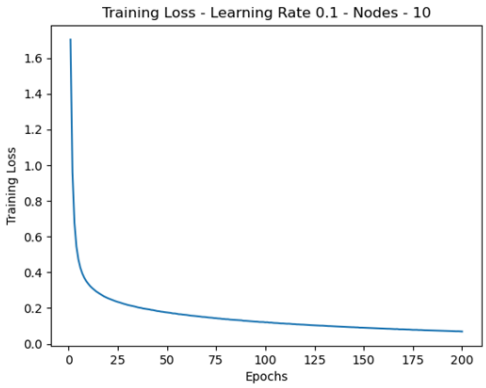
while remaining were classified correctly at least 80% of the time. When the learning rate is 0.001 the accuracy falls below 70% as shown in the graph in case of digits 5, 8 and 9. For the remaining digits the model accurately predicted more than 90% of the time.

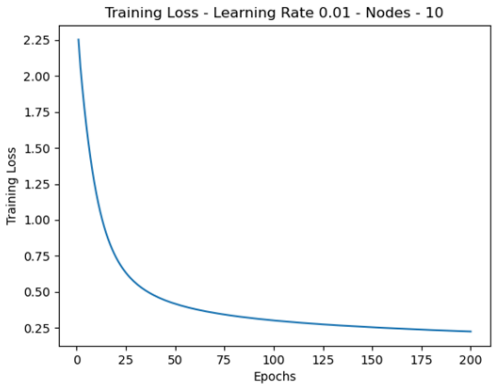
---------- TASK 2 ----------

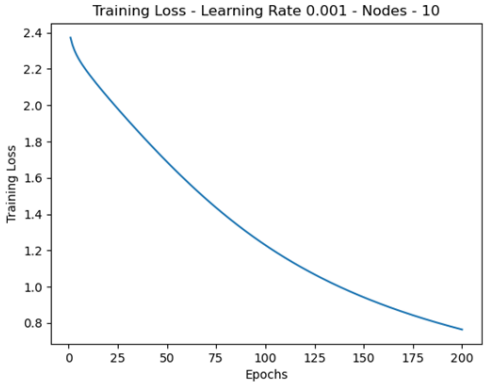
For the task we will use 0.1, 0.01 and 0.01 as learning parameters with different combinations of nodes in the hidden layer.

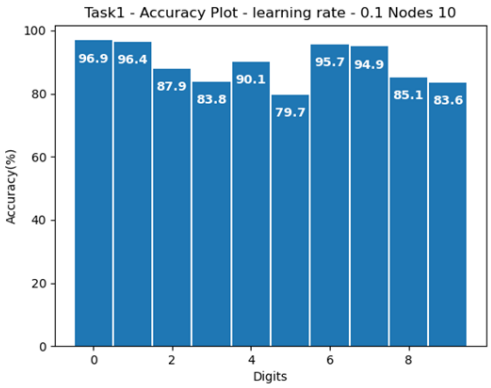
Nodes = 10

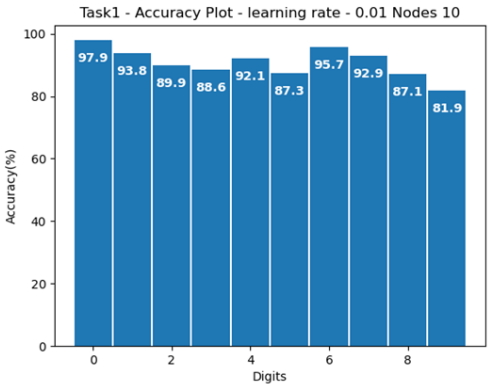
The accuracy plots are as follows:

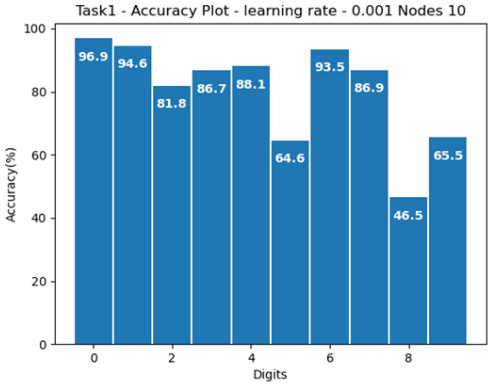

Training Loss - Learning Rate 0.1 - Nodes - 10
Training Loss
Epochs
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
0 25 50 75 100 125 150 175 200


Training Loss - Learning Rate 0.01 - Nodes - 10
Training Loss
Epochs
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
2.25
0 25 50 75 100 125 150 175 200

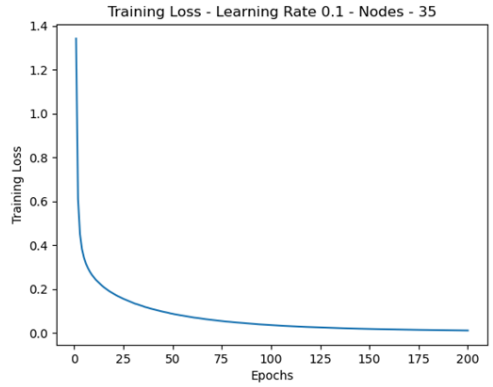

Training Loss - Learning Rate 0.001 - Nodes - 10
Training Loss
Epochs
0.8
1.0
1.2
1.4
1.6
1.8
2.0
2.2
2.4
0 25 50 75 100 125 150 175 200

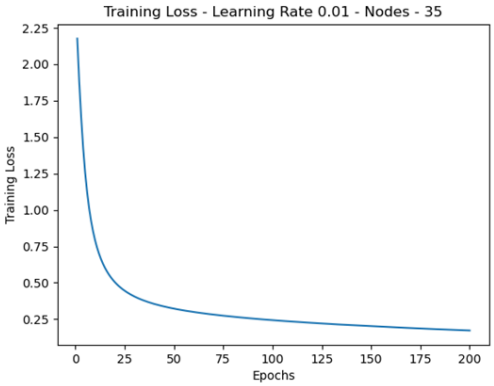

Task1 - Accuracy Plot - learning rate - 0.1 Nodes 10
Accuracy(%)
Digits
96.9
96.4
87.9
83.8
90.1
79.7
95.7
94.9
85.1
83.6

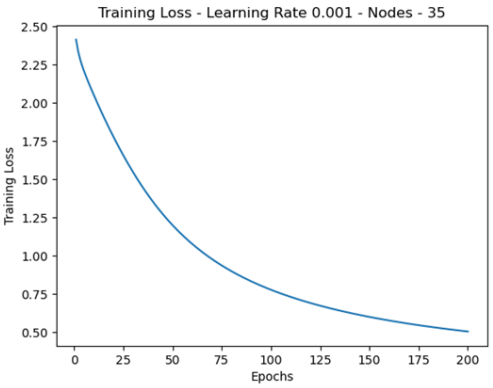

Task1 - Accuracy Plot - learning rate - 0.01 Nodes 10
Accuracy(%)
Digits
97.9
93.8
89.9
88.6
92.1
87.3
95.7
92.9
87.1
81.9

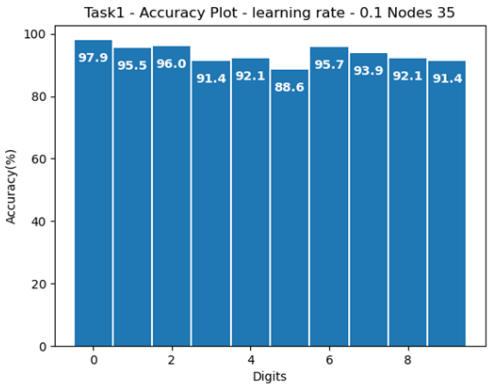

Task1 - Accuracy Plot - learning rate - 0.001 Nodes 10
Accuracy(%)
Digits
96.9
94.6
81.8
86.7
88.1
64.6
93.5
86.9
46.5
65.5

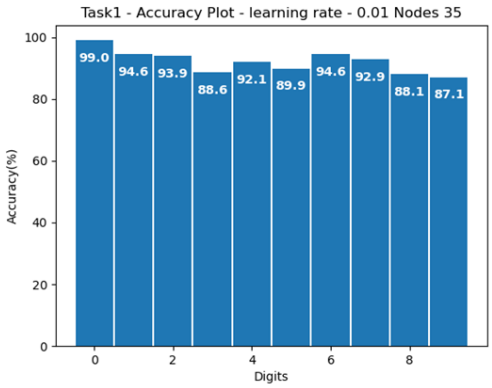
Nodes = 35

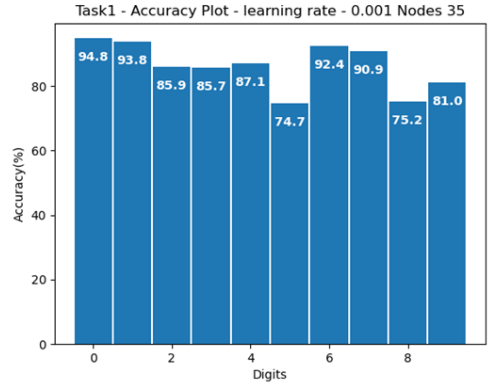

Training Loss - Learning Rate 0.1 - Nodes - 35
Training Loss
Epochs
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0
25
50
75
100
125
150
175
200


Training Loss - Learning Rate 0.01 - Nodes - 35
Training Loss
Epochs
2.25
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0
25
50
75
100
125
150
175
200

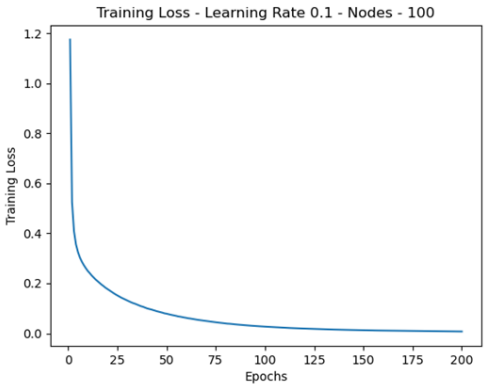

Training Loss - Learning Rate 0.001 - Nodes - 35
Training Loss
Epochs
2.50
2.25
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0
25
50
75
100
125
150
175
200

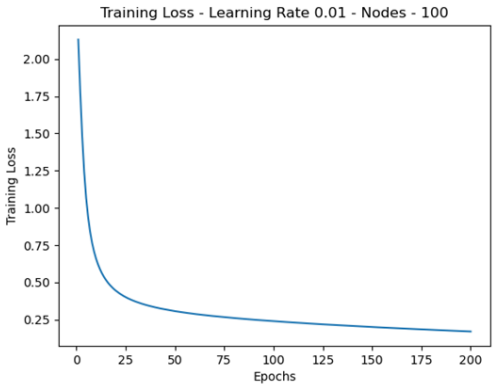

Task1 - Accuracy Plot - learning rate - 0.1 Nodes 35
Accuracy(%)
Digits
97.9
95.5
96.0
91.4
92.1
88.6
95.7
93.9
92.1
91.4

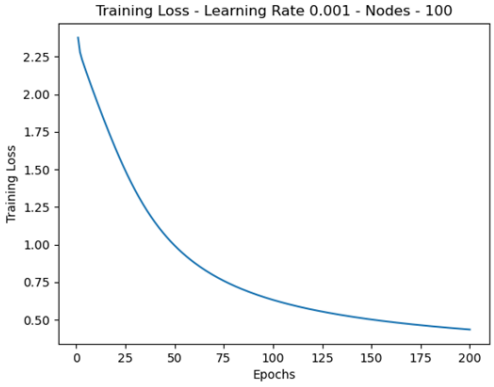

Task1 - Accuracy Plot - learning rate - 0.01 Nodes 35
Accuracy(%)
Digits
99.0
94.6
93.9
88.6
92.1
89.9
94.6
92.9
88.1
87.1

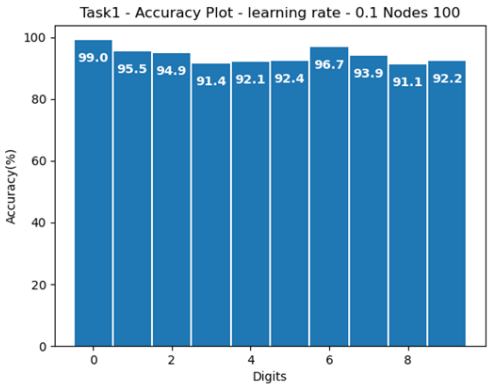

Task1 - Accuracy Plot - learning rate - 0.001 Nodes 35
Accuracy(%)
Digits
94.8
93.8
85.9
85.7
87.1
74.7
92.4
90.9
75.2
81.0

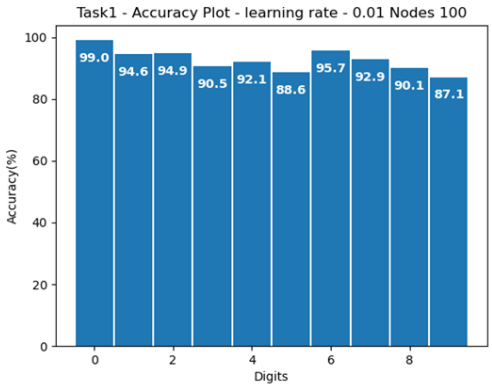
Nodes = 100

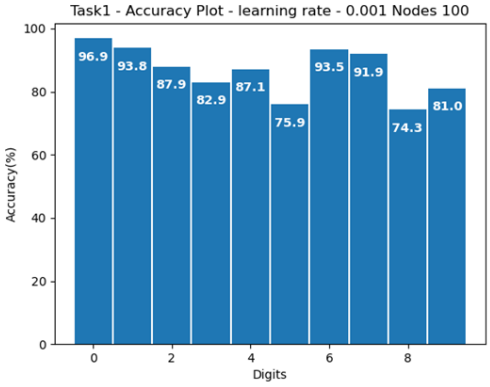

Training Loss - Learning Rate 0.1 - Nodes - 100
Training Loss
Epochs
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0
25
50
75
100
125
150
175
200


Training Loss - Learning Rate 0.01 - Nodes - 100
Training Loss
Epochs
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0
25
50
75
100
125
150
175
200

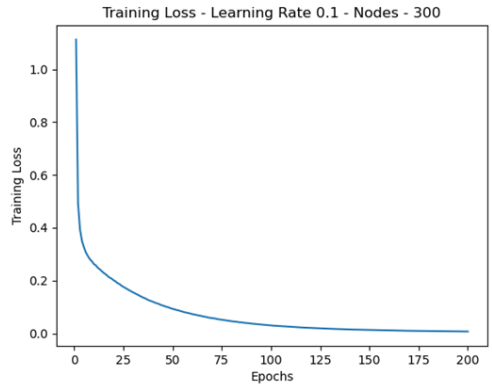

Training Loss - Learning Rate 0.001 - Nodes - 100
Training Loss
Epochs
2.25
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0
25
50
75
100
125
150
175
200

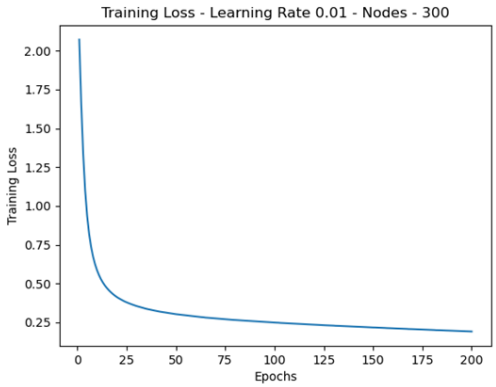

Task1 - Accuracy Plot - learning rate - 0.1 Nodes 100
Accuracy(%)
Digits
99.0
95.5
94.9
91.4
92.1
92.4
96.7
93.9
91.1
92.2

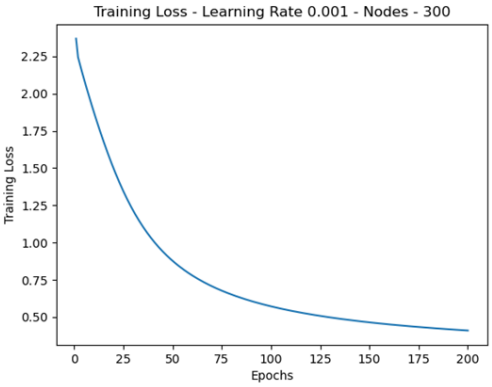

Task1 - Accuracy Plot - learning rate - 0.01 Nodes 100
Accuracy(%)
Digits
99.0
94.6
94.9
90.5
92.1
88.6
95.7
92.9
90.1
87.1

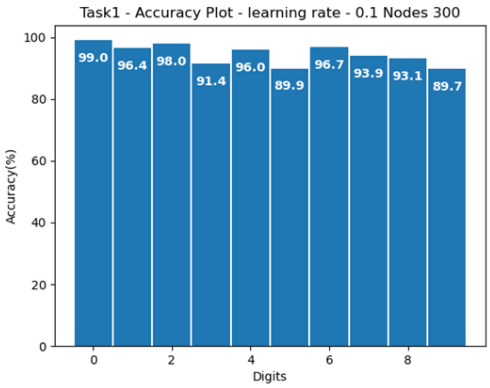

Task1 - Accuracy Plot - learning rate - 0.001 Nodes 100
Accuracy(%)
Digits
96.9
93.8
87.9
82.9
87.1
75.9
93.5
91.9
74.3
81.0

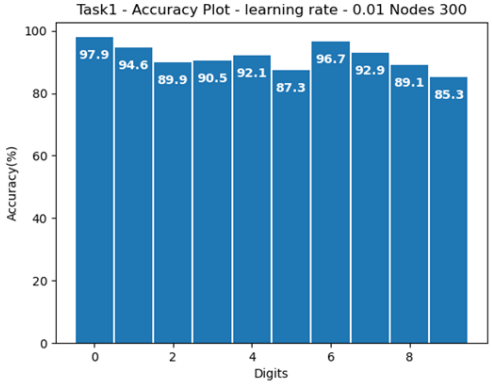
Nodes = 300

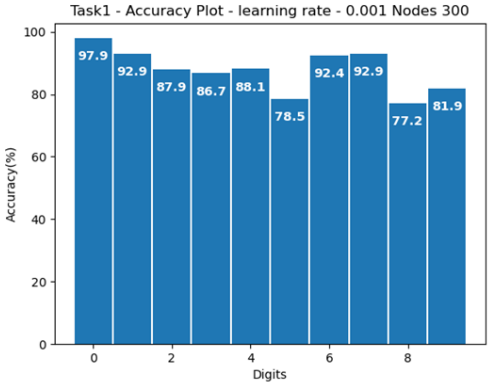

Training Loss - Learning Rate 0.1 - Nodes - 300
Training Loss
Epochs
0.0
0.2
0.4
0.6
0.8
1.0
0
25
50
75
100
125
150
175
200


Training Loss - Learning Rate 0.01 - Nodes - 300
Training Loss
Epochs
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
0
25
50
75
100
125
150
175
200

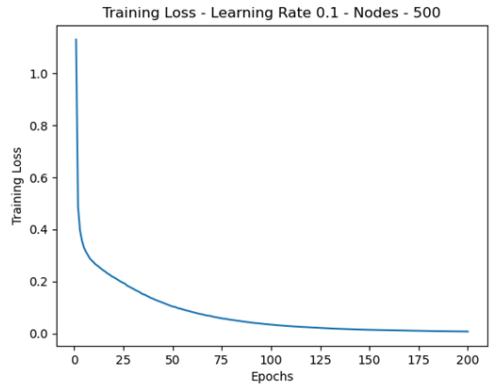

Training Loss - Learning Rate 0.001 - Nodes - 300
Training Loss
Epochs
0.50
0.75
1.00
1.25
1.50
1.75
2.00
2.25
0
25
50
75
100
125
150
175
200

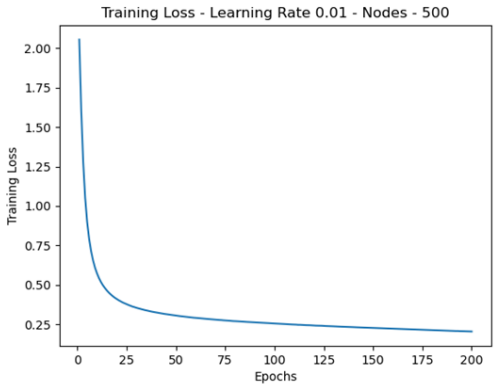

Task1 - Accuracy Plot - learning rate - 0.1 Nodes 300
Accuracy(%)
Digits
99.0
96.4
98.0
91.4
96.0
89.9
96.7
93.9
93.1
89.7

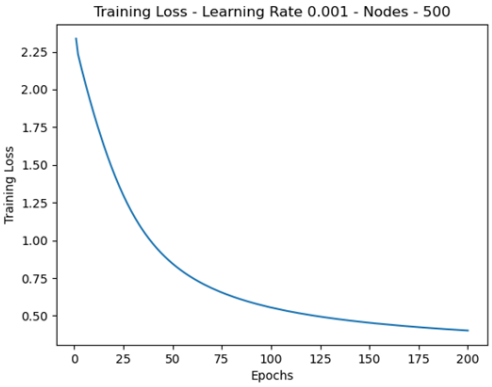

Task1 - Accuracy Plot - learning rate - 0.01 Nodes 300
Accuracy(%)
Digits
97.9
94.6
89.9
90.5
92.1
87.3
96.7
92.9
89.1
85.3

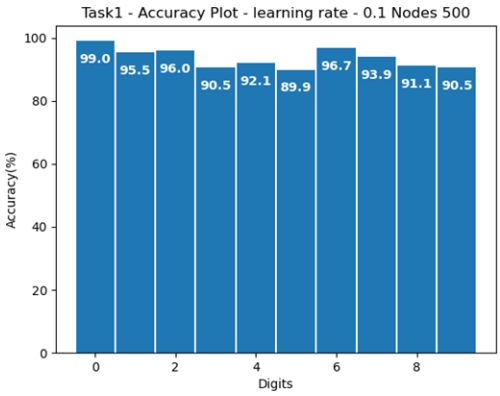

Task1 - Accuracy Plot - learning rate - 0.001 Nodes 300
Accuracy(%)
Digits
97.9
92.9
87.9
86.7
88.1
78.5
92.4
92.9
77.2
81.9

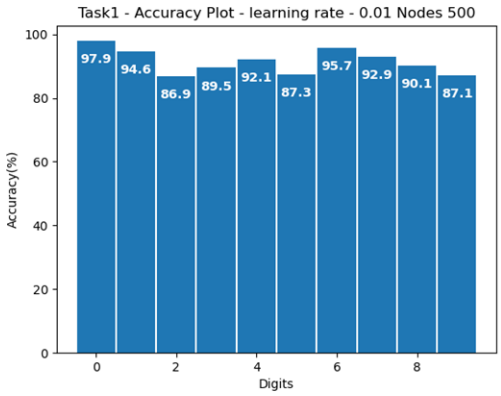
Nodes = 500

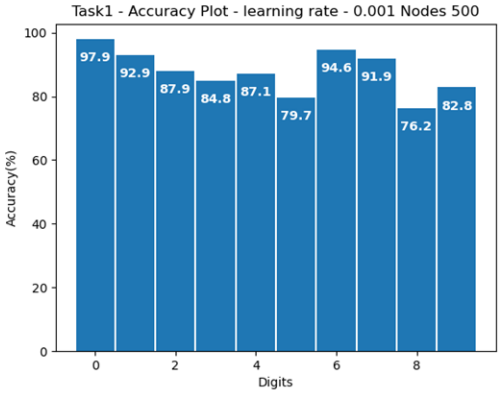

Training Loss - Learning Rate 0.1 - Nodes - 500
Training Loss
Epochs
0.0
0.2
0.4
0.6
0.8
1.0
0
25
50
75
100
125
150
175
200


Training Loss - Learning Rate 0.01 - Nodes - 500
Training Loss
Epochs
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
0
25
50
75
100
125
150
175
200


Training Loss - Learning Rate 0.001 - Nodes - 500
Training Loss
Epochs
0.50
0.75
1.00
1.25
1.50
1.75
2.00
2.25
0
25
50
75
100
125
150
175
200







The accuracy line plots confirm the fact that lowering the value of the learning parameter also slows learning. From the plots you can find that the learning rate 0.01 is most suited for modelling as the training loss curve is smooth.

Accuracy Table

Accuracy(%)	Learning Rate		
Nodes	**0.1**	**0.01**	**0.001**
10	89.8	89.7	83.9
35	93.9	91.1	86
100	94	91.8	86.9
300	94.1	91.2	87.1
500	94.3	91.2	87.4

---------- **TASK 4** ----------

Comparing SVM and Single Layer Perceptron Results

In our SVM model we achieved maximum accuracy using RBF kernel taking C equals to 100 and gamma equals to 0.01 on grayscale images. The maximum accuracy in case of Single Layer Perceptron is 89.2 percent on grayscale images with learning rate as 0.5.

In case of binary images using threshold value as 150 the maximum accuracy we will get in SVM model is 88.6 percent whereas in case of perceptron the maximum accuracy is 86.4 percent. For binary images both models performed equally well.

Perceptron models irrespective of learning rate and on the same dataset take more time than SVM. This is due to the fact that perceptron requires more iterations to converge to minimum as compared to SVMs.

Your best MLP model is the one with 500 nodes and learning rates set to 0.1 which is better than both SVM and Perceptron models.

❑ CONCLUSION:

The multi-layer perceptron is a strong classifier in comparison with both Support Vector Machines and Perceptron models. The learning in case of MLP is highly affected by the learning rate and the number of nodes in the hidden units. In the Multilayer perceptron we have found that the accuracy increases with the number of nodes and it is directly proportional to the learning rate. Lower the learning rate lower is the model's accuracy.

❑ REFERENCES:

https://scikit-learn.org/stable/user_guide.html
http://yann.lecun.com/exdb/mnist/
https://seaborn.pydata.org/
https://matplotlib.org/stable/contents.html
https://www.tensorflow.org/guide