NNDL Lab 1 Report

Sharvita Paithankar

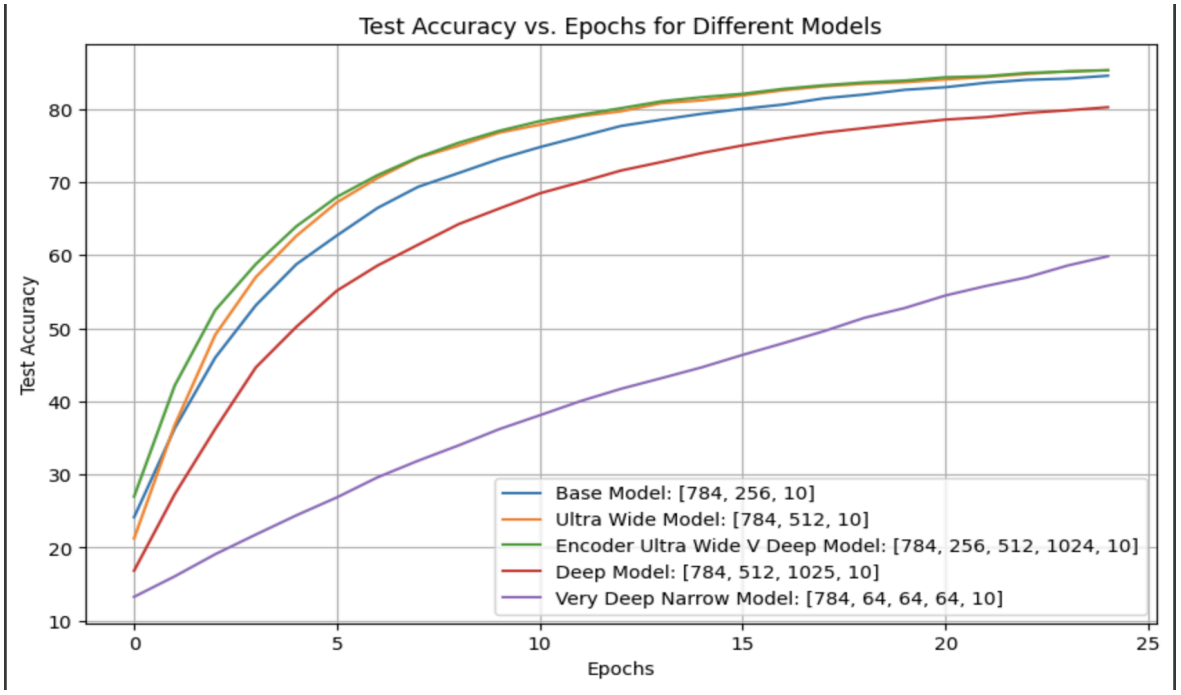
5.2.1 Experimental Design

The following are the experimental results of the model with different nodes at the hidden layer, different number of layers or both, different learning rates and LeakyReLU vs ReLUs:

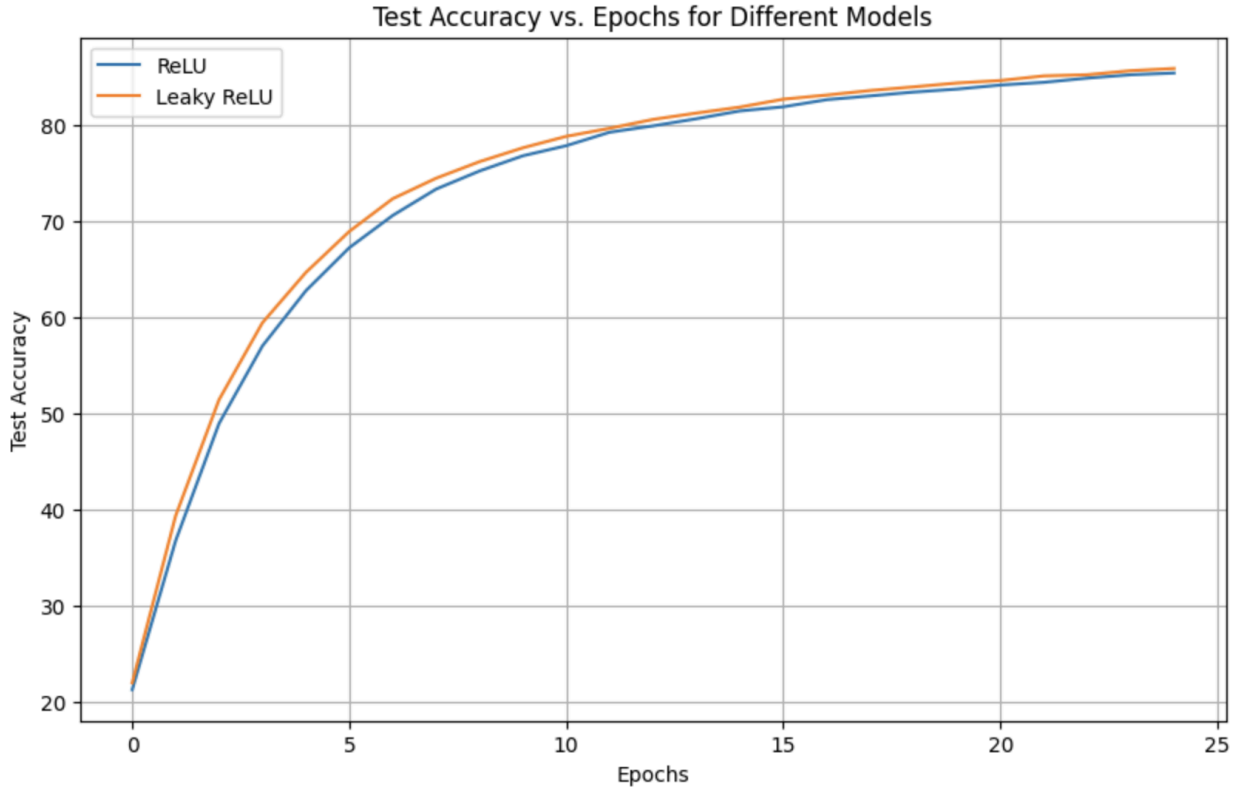
Models	Accuracy	
MLP([784, 256, 10])	84.5600357055664	
MLP([784, 512, 10])	85.33100128173828	More nodes in hidden layer
MLP([784, 512, 1025, 10])	85.3104476928711	more number of layers
MLP([784, 256, 512, 1024, 10])	80.25287628173828	More nodes and layers
MLP([784, 64, 64, 64, 10])	59.8684196472168	Added layers with less nodes
Relu	85.3412857055664	
LeakyRelu	87.275683744	
Learning rate 1e-6	85.47491455078125	
Learning rate 1e-4	96.3301773071289.	

5.2.2 Results

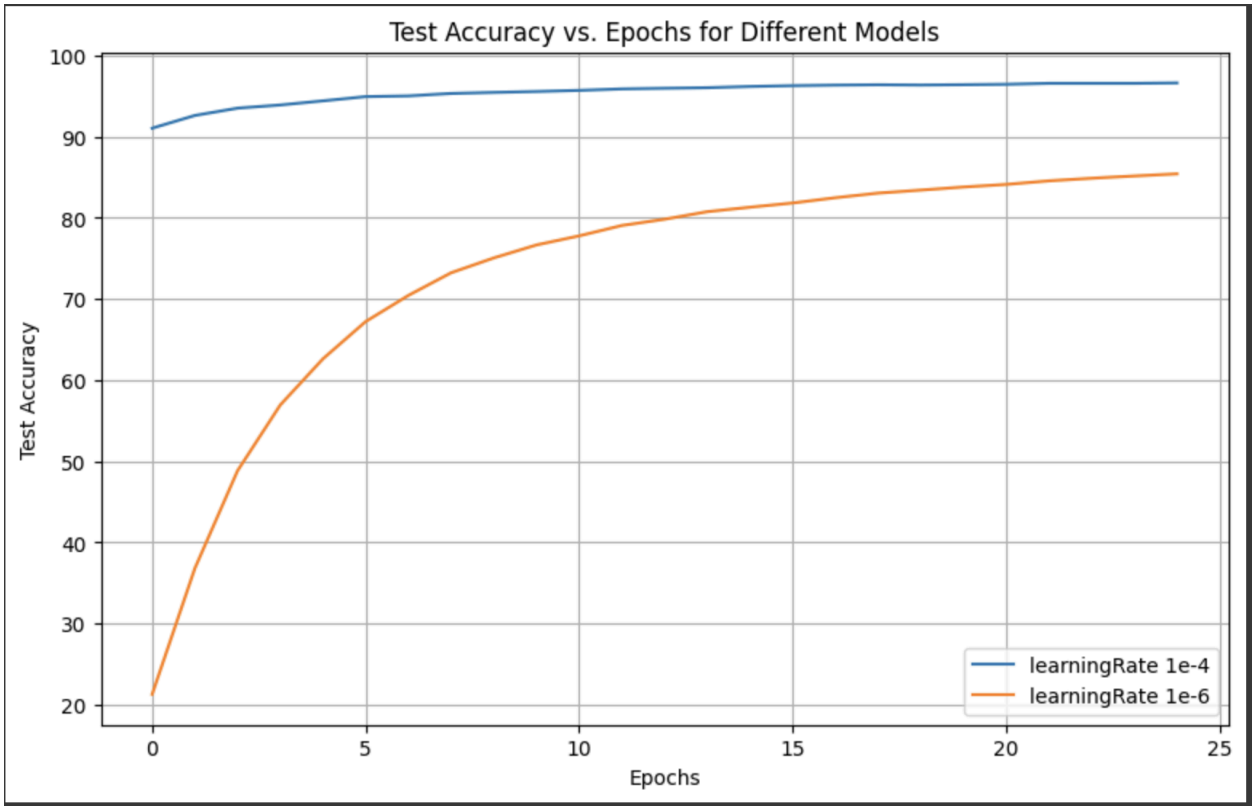
Here are the results for the first 5 elements in the table:



Here are the results for the Relu and LeakyRelu table:



Here are the results for the learning rate of 1e-4 and 1e-6:



5.2.3 Analysis

For an MLP([784, 256, 10]), the test accuracy was 84.5600357055664
This was the very first model that I tried and can be considered as a base to compare our other models.

For the MLP([784, 512, 10]), the test accuracy was 85.33100128173828
I increased the number of nodes in the hidden layer and can be considered as the base model and I expect a high accuracy for this. As expected, increasing the number of nodes in the hidden layer helped to improve the performance. More nodes means a better representation of the input and makes sense why this model performs the best.

For MLP([784, 512, 1025, 10]),the test accuracy was 85.3104476928711
I increased the number of layers and I expected a higher accuracy than the MLP([784, 256, 10]) model. Adding a layer should have improved the performance but with 25 epochs, the layer did not contribute as much and probably because it might need more training. I think it slowed down how much the earlier layers can update.

For MLP([784, 256, 512, 1024, 10]), the test accuracy was 80.25287628173828
I increased both nodes and layers, and I expected a higher accuracy for this model and for it to be the best performance out of all. The performance actually dropped to 80.25, and adding

more layers decreased the impacts of updates on the layers, where the input features are learned.

For MLP([784, 64, 64, 64, 10]), the test accuracy was 59.8684196472168.
I added more layers with less nodes and I expected a decreased accuracy for this model. Adding more layers did not help since the layers are too small. With 64 nodes, the model did not perform well and shows that deeper does always mean better results.

For the Learning rate change and activation function:

I picked MLP([784, 512, 10]) with the test accuracy accuracy of 85.33100128173828 since it had the best performance, I decided to change:

Learning Rate: I increased learning rate from 1e-6 to 1e-4 and expected to get a test accuracy that would increase exponentially.
The actual test accuracy I got was 96.3301773071289.
The learning rate improved the test accuracy exponentially. The higher the learning rate helps the weights to be updated faster and causes more effective learning.

Activation function: I used Leaky ReLU instead of ReLU and expected to get a slightly increased test accuracy.
The actual test accuracy I got was 87.275683744
ReLU can sometimes cause neurons to become inactive but switching ReLU to LeakyReLU helped fix the issue and increased the accuracy.