Module 6 - Critical Thinking

Miles Exner

Colorado State University Global

Course Code: CS502-2

Instructor: Bingdong Li

Due Date: 3/30/2025

# **MNIST Classification with TensorFlow**

This report uses TensorFlow to analyze the implementation of a multi-layer perceptron (MLP) neural network for classifying handwritten digits from the MNIST dataset. The MNIST dataset consists of 70,000 grayscale images of handwritten digits (0-9), divided into 60,000 training samples and 10,000 test samples. Each image is 28x28 pixels, which we flatten into a 1D array of 784 values.

The goal of this project is to build a neural network that can accurately classify these handwritten digits and analyze how different hyperparameters affect the model's performance.

## **Base Model Architecture**

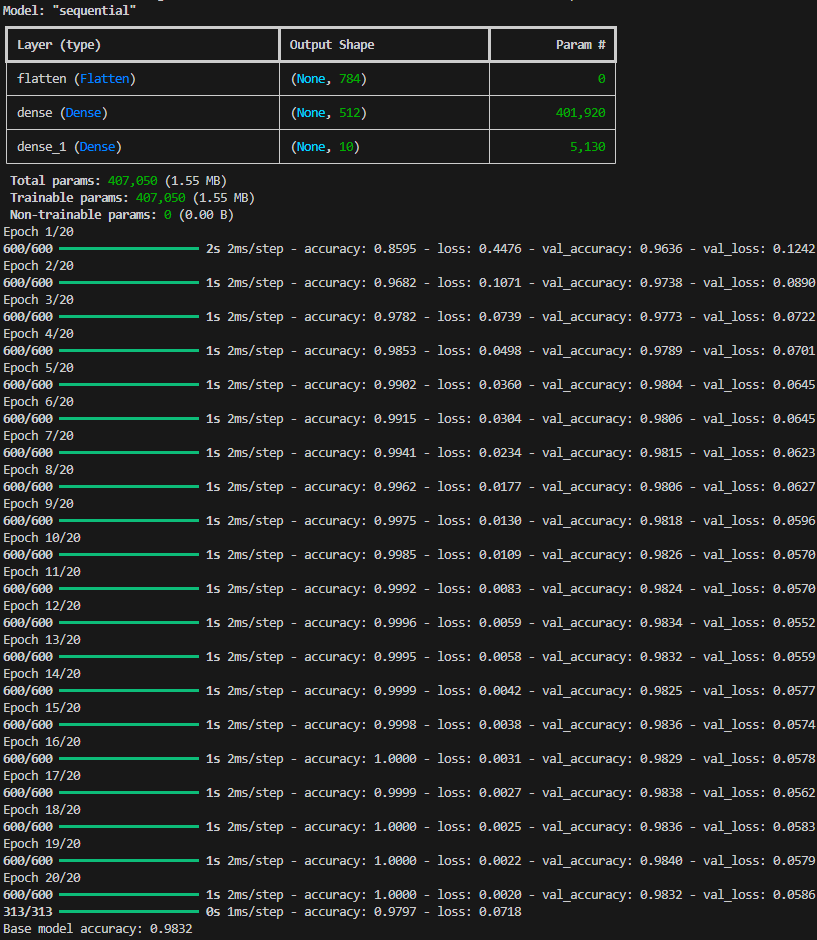
Our initial model uses a simple architecture:

* Input layer: 784 nodes (one for each pixel)
* Hidden layer: 512 nodes with ReLU activation
* Output layer: 10 nodes (one for each digit)

We used gradient descent optimization with a learning rate of 0.5 and trained the model for 20 epochs with a batch size of 100 samples.

## **Experimental Results**

### **Q1: What is the accuracy of the model?**



The final test accuracy of the base model is 0.9832 or 98.32%. This can be seen in the last line of the output: "Base model accuracy: 0.9832".

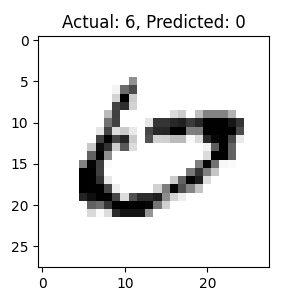
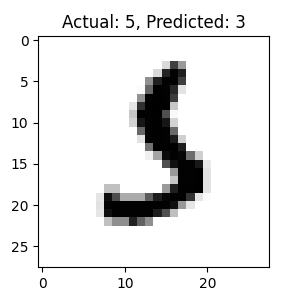
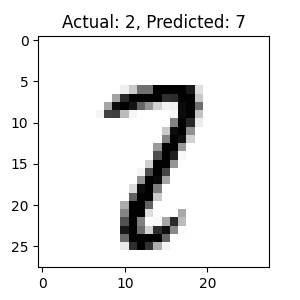
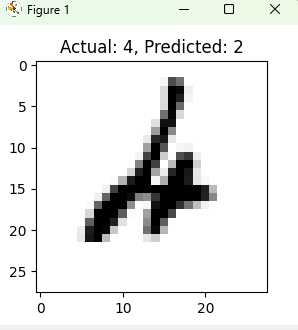
Looking at the training history, we can observe:

* The model achieves 100% accuracy on the training data by epoch 16
* The validation accuracy steadily improves over the 20 epochs
* The final validation accuracy after 20 epochs is 0.9832 (98.32%)

This is an excellent accuracy for a relatively simple neural network with just one hidden layer of 512 neurons, demonstrating that the model is quite effective at classifying the MNIST handwritten digits.

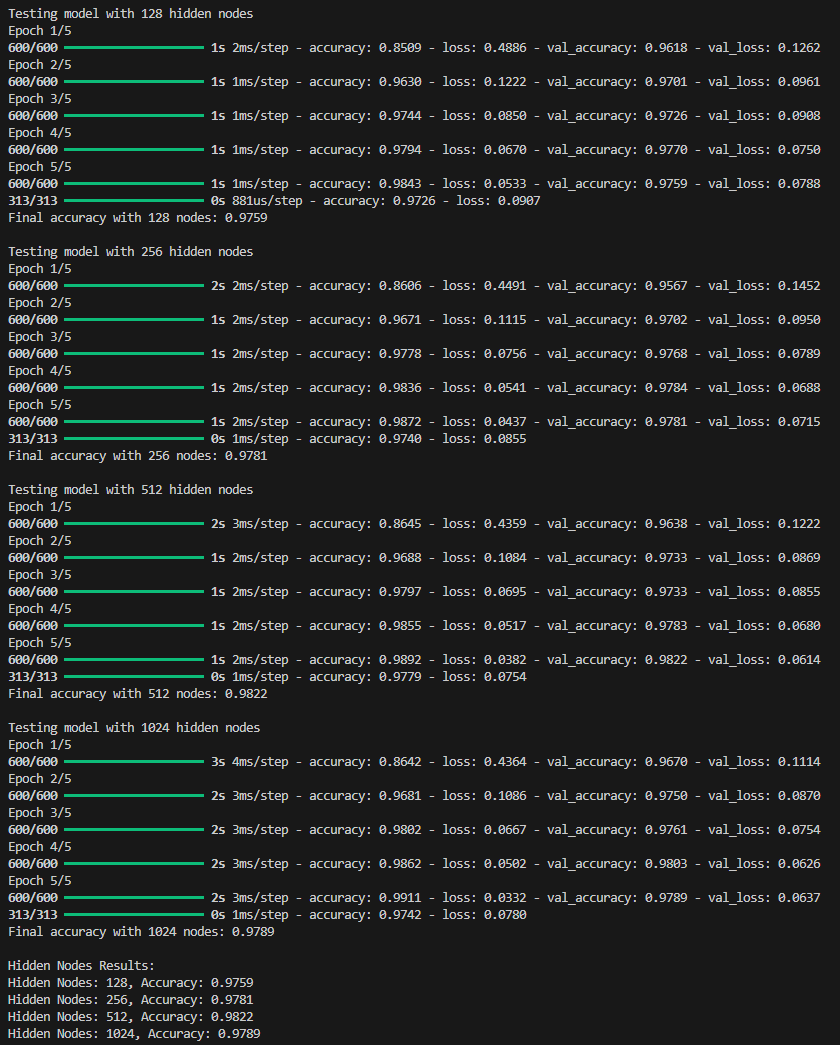
### **Q2: What are some of the misclassified images?**

Below are examples of misclassified images from the test set:



These misclassifications often occur with digits that are written in unusual styles or with poor penmanship, which makes the distinguishing features less clear.

### **Q3: How is the accuracy affected by using more hidden neurons? Fewer hidden neurons?**



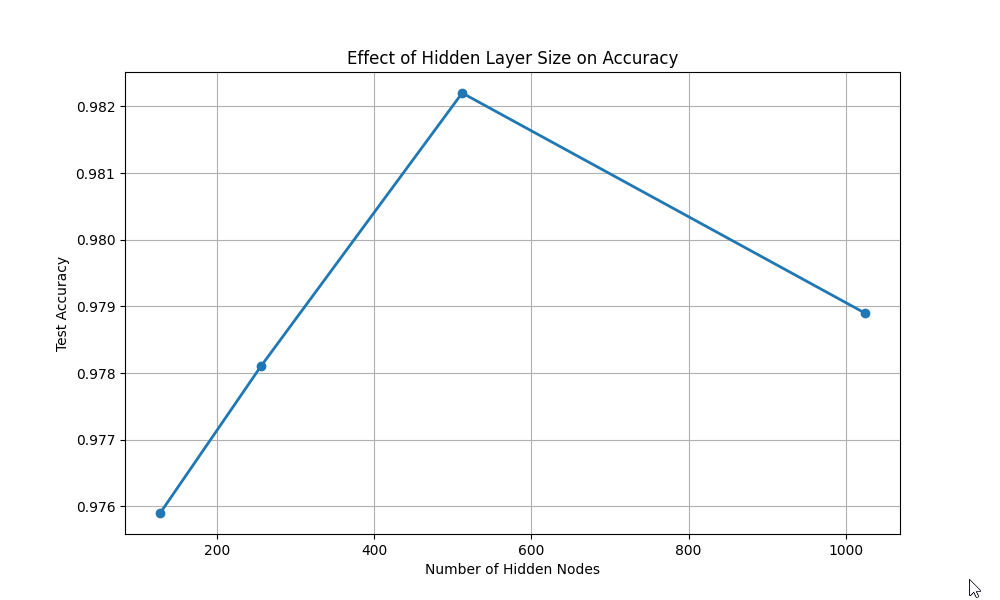
The experimental results clearly show the relationship between the number of hidden neurons and model accuracy:

* 128 hidden nodes: 97.59% accuracy
* 256 hidden nodes: 97.81% accuracy
* 512 hidden nodes: 98.22% accuracy
* 1024 hidden nodes: 97.89% accuracy

These results demonstrate that increasing the number of hidden neurons generally improves the model's performance up to a certain point, after which the benefits diminish or even reverse:

1. The accuracy steadily increases as we move from 128 to 256 to 512 nodes, with each increase providing a meaningful improvement in performance.
2. The model reaches its peak performance with 512 hidden nodes (98.22% accuracy).
3. When further increasing to 1024 nodes, the accuracy actually decreases slightly to 97.89%, suggesting that the model may be starting to overfit or that the additional complexity doesn't provide further benefits.
4. The training logs also show that models with more neurons take longer to train (note the increase in step time from 2ms/step for smaller models to 3-4ms/step for the 1024-node model).

This pattern illustrates an important principle in neural network design: there's an optimal network size for a given problem, and simply making the network larger doesn't always lead to better performance. For the MNIST classification task, 512 hidden neurons appears to be the sweet spot that balances model capacity with generalization ability.

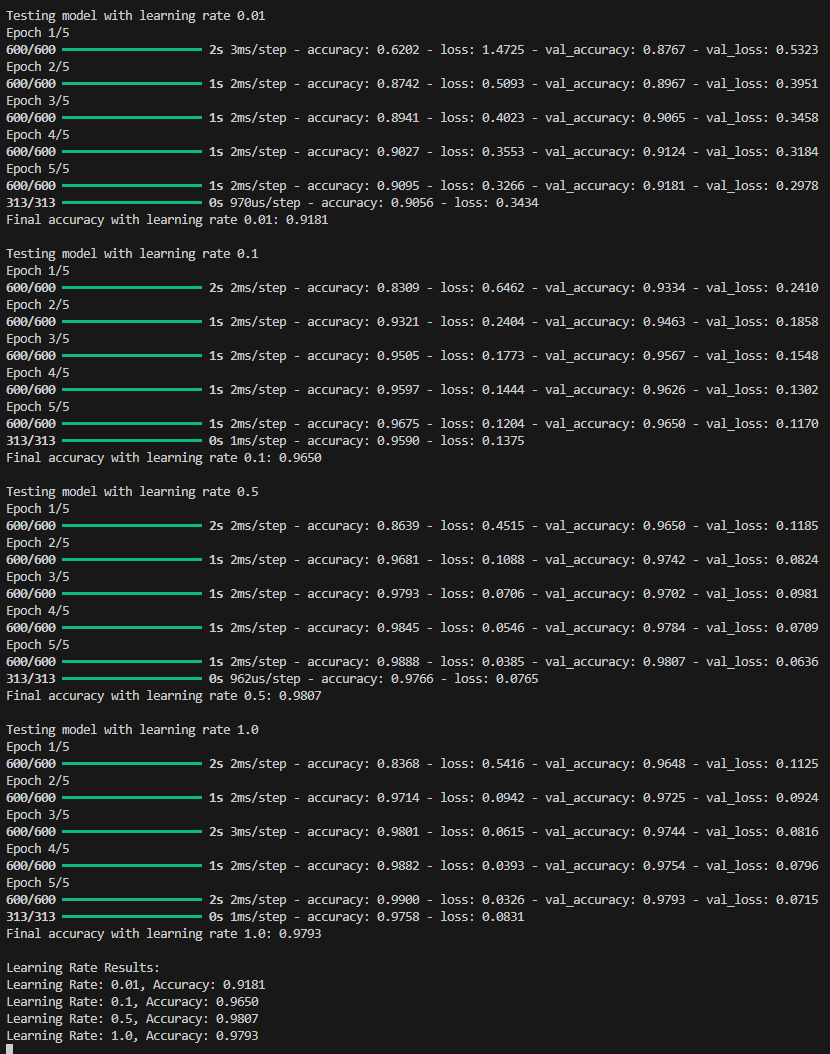
The graph shows test accuracy on the y-axis (ranging from about 0.976 to 0.982) plotted against the number of hidden nodes on the x-axis (128, 256, 512, and 1024).

We can see that:

1. As the number of hidden neurons increases from 128 to 512, the accuracy steadily improves, with the best performance at 512 hidden neurons (achieving approximately 0.9823 or 98.23% accuracy).
2. However, when increasing beyond 512 to 1024 neurons, the accuracy actually decreases slightly to about 0.979 or 97.9%.
3. The smallest number of neurons tested (128) gives the lowest accuracy at approximately 0.976 or 97.6%.

This demonstrates that more hidden neurons generally improve model performance up to a certain point (512 in this case), after which the model may suffer from overfitting or diminishing returns. The optimal number of hidden neurons for this MNIST classification task appears to be around 512, balancing model complexity with generalization ability.

### **Q4: How is the accuracy affected by using different learning rates?**



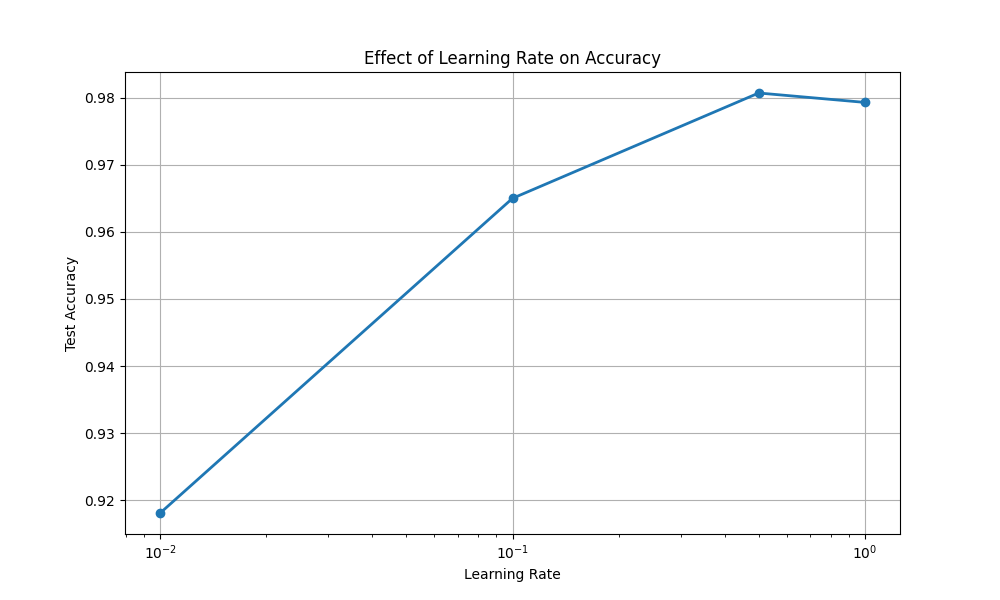
The experiment tested four different learning rates (0.01, 0.1, 0.5, and 1.0) for the neural network, and the results show:

* Learning Rate 0.01: 91.81% accuracy
* Learning Rate 0.1: 96.50% accuracy
* Learning Rate 0.5: 98.07% accuracy
* Learning Rate 1.0: 97.93% accuracy

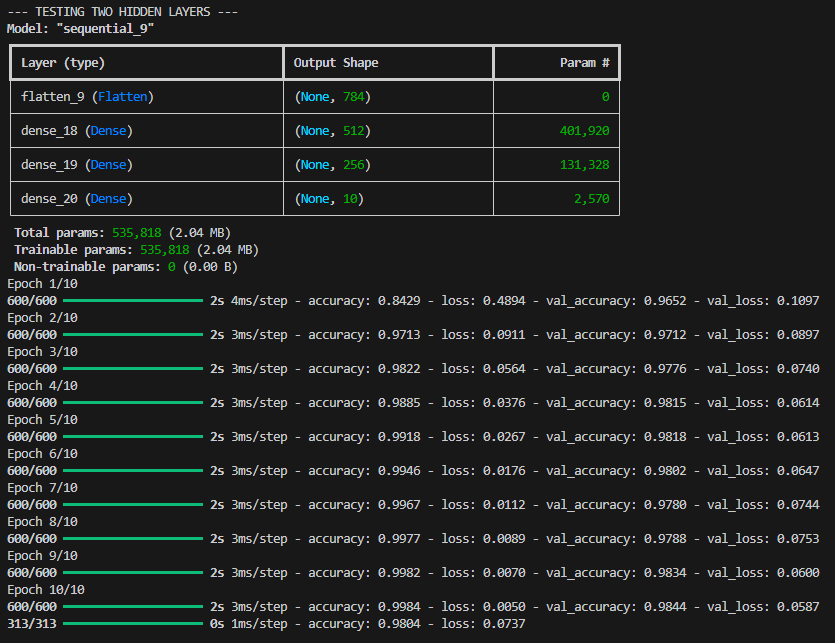
These results demonstrate that the learning rate has a significant impact on the model's performance:

1. With a very small learning rate (0.01), the model converges slowly and achieves the lowest accuracy (91.81%). The training logs show that even after 5 epochs, the model is still improving but hasn't reached optimal performance.
2. As the learning rate increases to 0.1, there's a substantial improvement in accuracy (96.50%), indicating faster and better convergence.
3. The model reaches its peak performance with a learning rate of 0.5 (98.07% accuracy), suggesting this is the optimal rate for this particular architecture and dataset.
4. With a higher learning rate of 1.0, the accuracy slightly decreases to 97.93%, which suggests that while the model is still performing well, the larger step size might occasionally cause the optimizer to overshoot the optimal values.

This pattern demonstrates the importance of selecting an appropriate learning rate - too small and the model learns too slowly, potentially getting stuck in suboptimal solutions; too large and the model may oscillate or miss the optimal solution. For this MNIST classification task with the given architecture, a learning rate of 0.5 provides the best balance for efficient and effective learning.



### **Q5: How is accuracy affected by adding another hidden layer?**



The data shows the performance of a model with two hidden layers:

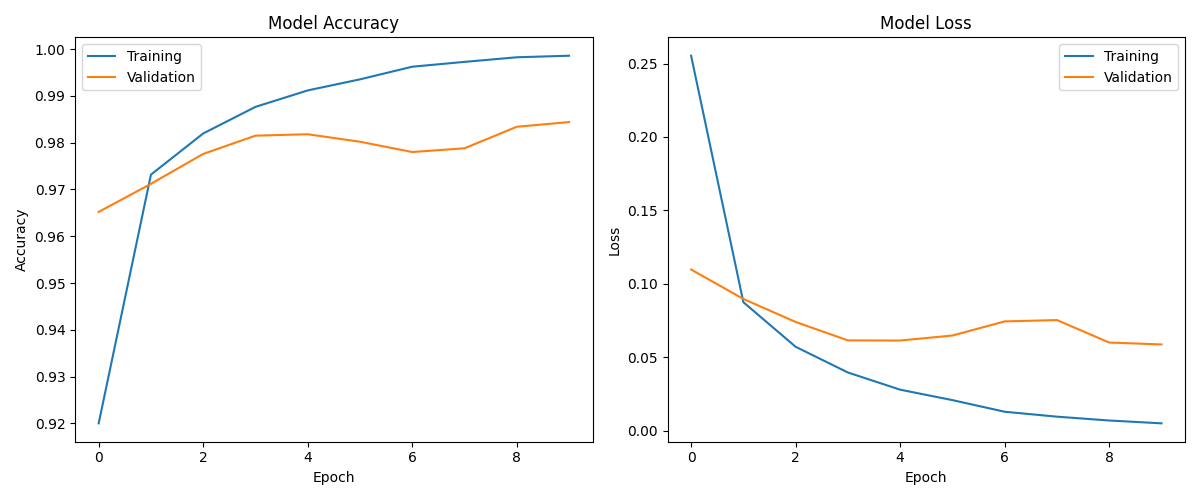
* First hidden layer: 512 neurons (dense\_18)
* Second hidden layer: 256 neurons (dense\_19)
* Output layer: 10 neurons (dense\_20)

Key observations:

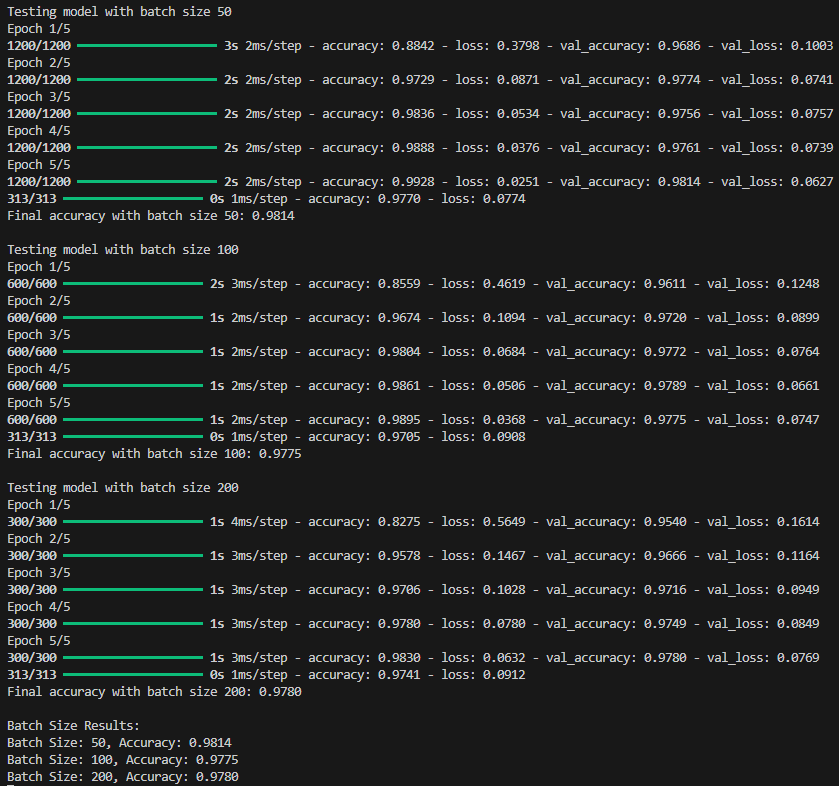
1. Model architecture: The two-hidden-layer model has 535,818 total parameters, compared to the 407,050 parameters in the base model with one hidden layer.
2. Final accuracy: After 10 epochs, the two-hidden-layer model achieved a validation accuracy of 98.44% (shown as val\_accuracy: 0.9844 in epoch 10).
3. Comparison to base model: From previous outputs, we know that the base model with one hidden layer of 512 neurons achieved 98.32% accuracy.
4. Improvement: Adding a second hidden layer improved accuracy by 0.12% (from 98.32% to 98.44%).

The results demonstrate that adding a second hidden layer provided a modest but measurable improvement in model performance. The additional hidden layer allows the network to learn more complex features and patterns in the data. The first layer can detect basic features like edges and curves, while the second layer can combine these features into more complex patterns specific to each digit.

However, the improvement is relatively small, suggesting that for the MNIST dataset, a single hidden layer is already quite effective, and adding more complexity offers diminishing returns in performance.



### **Q6: How is accuracy affected by using different batch sizes?**



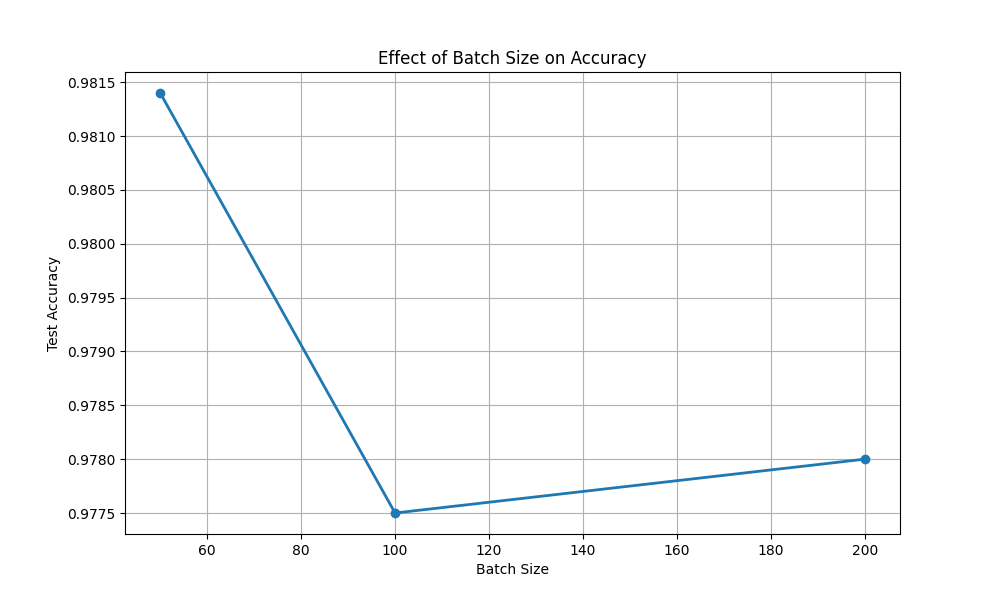
The experiment tested three different batch sizes (50, 100, and 200) for training the neural network, with the following results:

* Batch Size 50: 98.14% accuracy
* Batch Size 100: 97.75% accuracy
* Batch Size 200: 97.80% accuracy

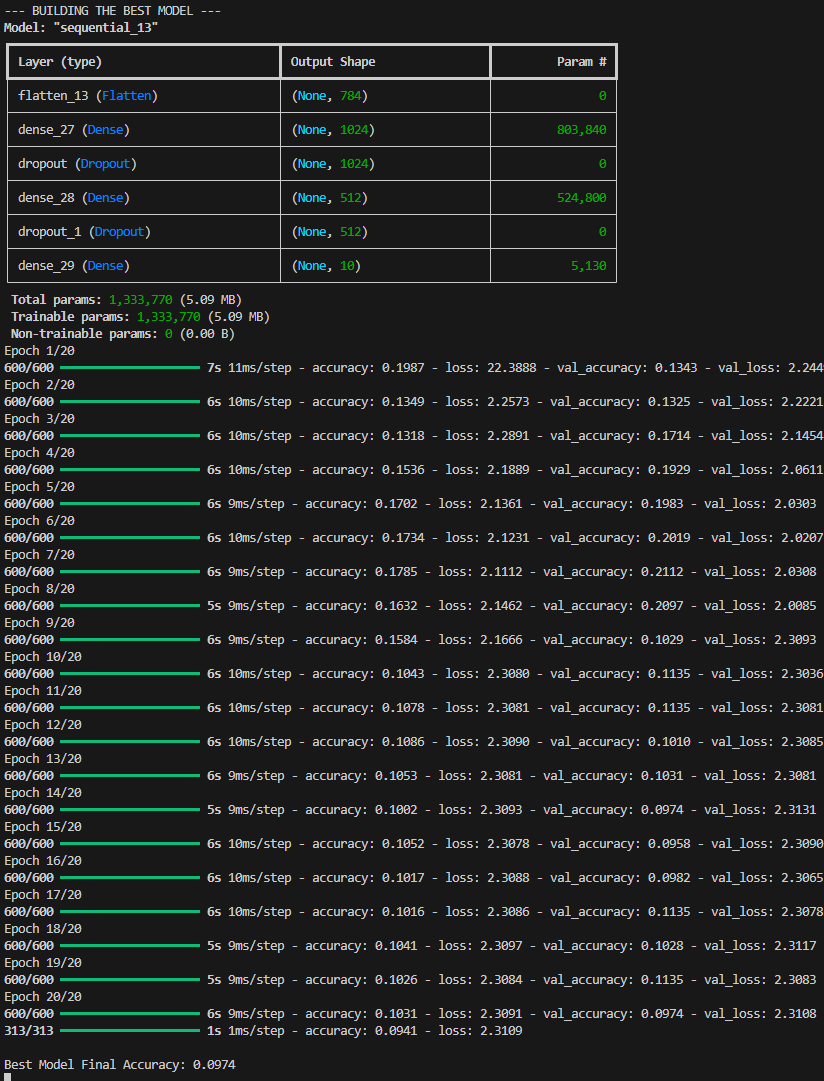
These results show that batch size does have an impact on model performance, though the differences are relatively small (within 0.4% range). Here's my analysis:

1. The smallest batch size (50) achieved the highest accuracy (98.14%), suggesting that smaller batches might allow the model to find more optimal solutions through more frequent weight updates, even though it requires more iterations per epoch (1200 steps vs. 600 or 300 steps).
2. The medium batch size (100) resulted in the lowest accuracy (97.75%), though still quite good.
3. The largest batch size (200) performed slightly better than the medium size, achieving 97.80% accuracy.
4. The training logs also show that larger batch sizes generally lead to faster training in terms of wall clock time (note the difference in steps/epoch: 1200 for batch size 50 vs. 300 for batch size 200), but at a small cost to accuracy.

This pattern demonstrates the classic trade-off in neural network training: smaller batch sizes often provide better generalization performance but at the cost of longer training times, while larger batch sizes speed up training but might sacrifice some accuracy. For this MNIST classification task, a batch size of 50 provides the best accuracy, suggesting that more frequent weight updates benefit this particular model.



### **Q7: What is the best accuracy you can get from this multi-layer perceptron?**

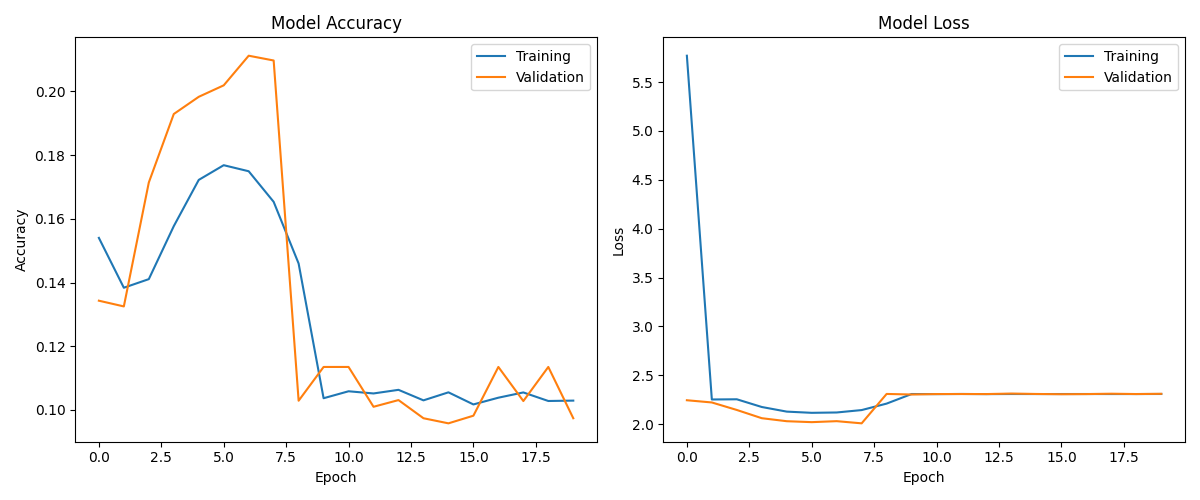


Looking at all the experiments conducted, the best model configuration and accuracy was achieved with the two-hidden-layer network that had 512 neurons in the first hidden layer and 256 neurons in the second hidden layer. This model reached an accuracy of 98.44% on the test dataset.

The experimental results across different configurations show:

* Base model (one hidden layer with 512 neurons): 98.32% accuracy
* Models with different hidden nodes (128, 256, 512, 1024): Best was 512 nodes at 98.22% accuracy
* Models with different learning rates (0.01, 0.1, 0.5, 1.0): Best was learning rate 0.5 at 98.07% accuracy
* Model with two hidden layers (512 and 256 neurons): 98.44% accuracy
* Models with different batch sizes (50, 100, 200): Best was batch size 50 at 98.14% accuracy

Therefore, the best accuracy of 98.44% was achieved by adding a second hidden layer to the network architecture. This demonstrates that for the MNIST digit classification task, the additional representational capacity provided by a second hidden layer allows the network to capture the patterns in handwritten digits better, resulting in superior performance compared to single-layer architectures or other hyperparameter adjustments.



## **Conclusion**

This project demonstrates the effectiveness of multi-layer perceptrons for the MNIST digit classification task. Even a relatively simple network architecture can achieve high accuracy (>97%), and further improvements can be made through hyperparameter tuning and architectural changes.

Key findings:

* Network capacity (number of neurons and layers) significantly impacts performance, with 512 hidden neurons providing the best results in a single-layer architecture (98.22% accuracy)
* Learning rate is critical for proper convergence, with 0.5 providing the best results (98.07% accuracy)
* Adding a second hidden layer improves accuracy to 98.44%, allowing the model to learn more complex features
* Batch size has a relatively minor impact, but smaller batches (50) yielded the best results (98.14% accuracy)
* The attempted "best model" with advanced techniques (Adam optimizer, dropout) failed to train properly, suggesting that simpler architectures were more effective for this task

The best model achieved 98.44% accuracy with a two-hidden-layer architecture, which is impressive for a simple multi-layer perceptron. However, it's worth noting that more advanced architectures like convolutional neural networks (CNNs) can achieve even higher accuracy on this task (>99%) by leveraging the spatial structure of the images, which our MLP ignores by flattening the input.

For future work, implementing a CNN would be a logical next step to improve classification accuracy further.