Module 2: Critical Thinking Option 1

Jocelyn Strmec

Colorado State University Global

CSC580 Applying Machine Learning and Neural Networks - Capstone

Pubali Banerjee

08/24/2025

"What is the accuracy of the model?"

The trained model achieved an accuracy of **0.98** on the MNIST handwritten digit dataset. This is a very strong result, considering the dataset contains many samples that are difficult even for humans to classify consistently. An accuracy near 98% demonstrates that the network architecture and training parameters were well suited to the task, capturing the relevant features in the digit images while avoiding significant overfitting. Although perfection is rarely possible with real-world data, this level of performance places the model in line with other strong baselines in the literature for MNIST.

"What are some of the misclassified images?"

Of the test samples that were misclassified many were ambiguous and took a moment to decipher even with the known prediction. For example, a poorly written "5" might closely resemble either a "6" or a "3" depending on the curvature, and digits like "1" and "7" can often be confused depending on how the author wrote them. In several cases, I found myself needing to double-check the label and prediction before deciding which interpretation made more sense. This illustrates an important point: model errors are not always clear-cut failures, but can arise from the subjective and noisy nature of handwriting itself.

"How is the accuracy affected by using more hidden neurons? Fewer hidden neurons?"

By using more hidden neurons the accuracy gradually increased, implying a correlation between more neurons and the higher accuracy. This trend supports the general principle in deep learning that larger capacity models perform better on sufficiently large datasets. As noted by Kaplan et al. (2020), scaling laws show a consistent positive correlation between model size, dataset size, and compute with improved performance. Of course, there are practical trade-offs: larger models are more computationally expensive and may risk overfitting in smaller datasets.

Num of Hidden Neurons	Accuracy
256	0.9821
512	0.9827
1024	0.9847

“How is the accuracy affected by using different learning rates? ”

In these variations it was interesting to see the difference between learning rates, in this dataset the largest learning rate performed the best. Interestingly, this goes against the common expectation that excessively high learning rates can destabilize training and reduce accuracy. In this case, however, the model seemed to benefit from more aggressive updates, converging more quickly and avoiding shallow minima. This result highlights how hyperparameter tuning is often dataset-specific, and assumptions about “good” learning rates should be empirically tested rather than taken for granted.

Learning Rate	Accuracy
0.005	0.9314
0.05	0.9757
0.5	0.9831
1.0	0.9841

“How is accuracy affected by adding another hidden layer?”

The number of hidden layers increased accuracy of the model, showing the more complexity the more nuanced the model can be and predict more accurately. This aligns with the results of more neurons also increasing accuracy. This suggests that deeper architectures enable the network to capture more complex hierarchical features of the handwritten digits, leading to slightly better generalization. The gains here were incremental rather than dramatic, but they reflect the general principle that depth allows neural networks to represent more abstract functions. That said, the improvements eventually plateau, and the benefit of adding additional layers diminishes without changes in other factors such as data size or regularization strategies.

Num of Hidden Layers	Accuracy
1	0.9844
2	0.9857
3	0.9860

“How is accuracy affected by using different batch sizes?”

When setting batch size with this model and dataset, the smaller the batch size the more accurate the model was. This shows that sample sizes for data are vital for model performance. The results imply smaller batches introduce more noise into the gradient updates, which can actually help the model escape local minima and generalize better. Larger batches, while computationally efficient, often result in more stable but less effective convergence, leading to slightly worse accuracy. These results underscore that while batch size may seem like a secondary parameter, it plays an important role in balancing computational efficiency and predictive performance.

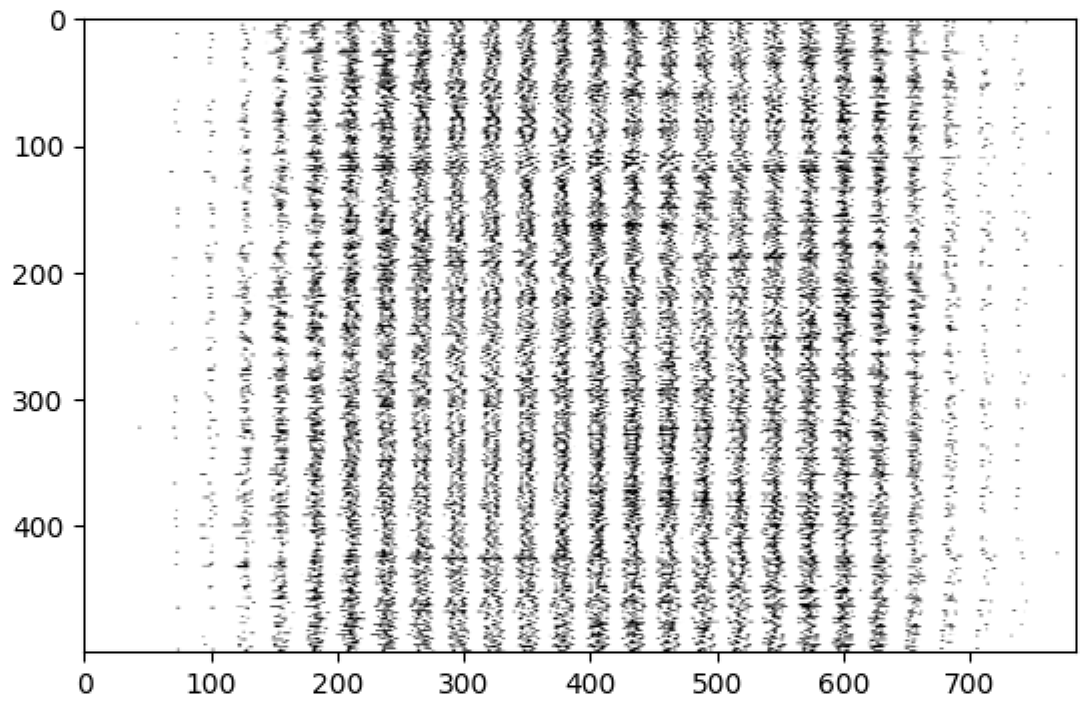
Batch Size	Accuracy
32	0.9853
64	0.9834
128	0.9825
256	0.9806

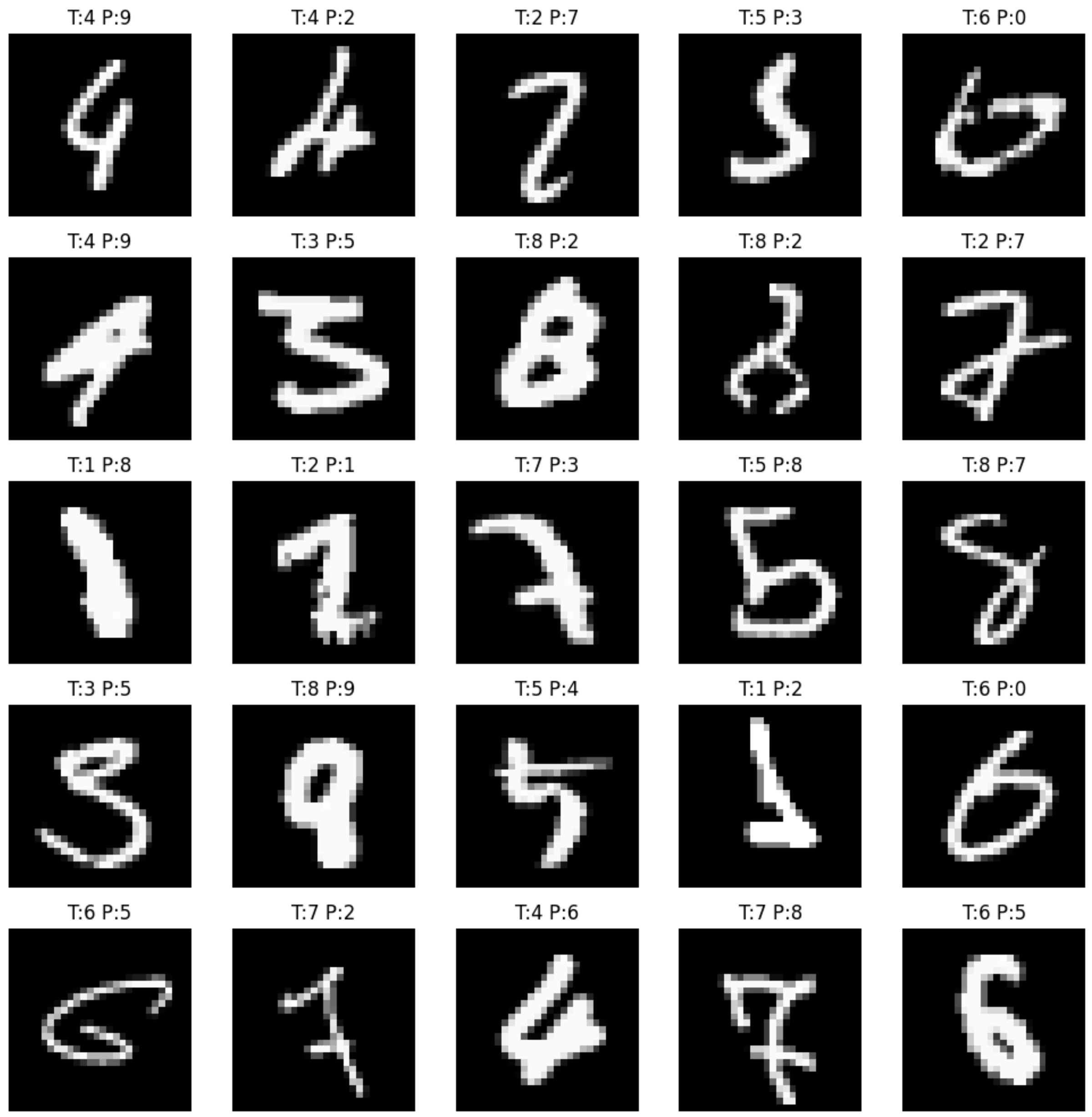
“What is the best accuracy you can get from this multi-layer perceptron?”

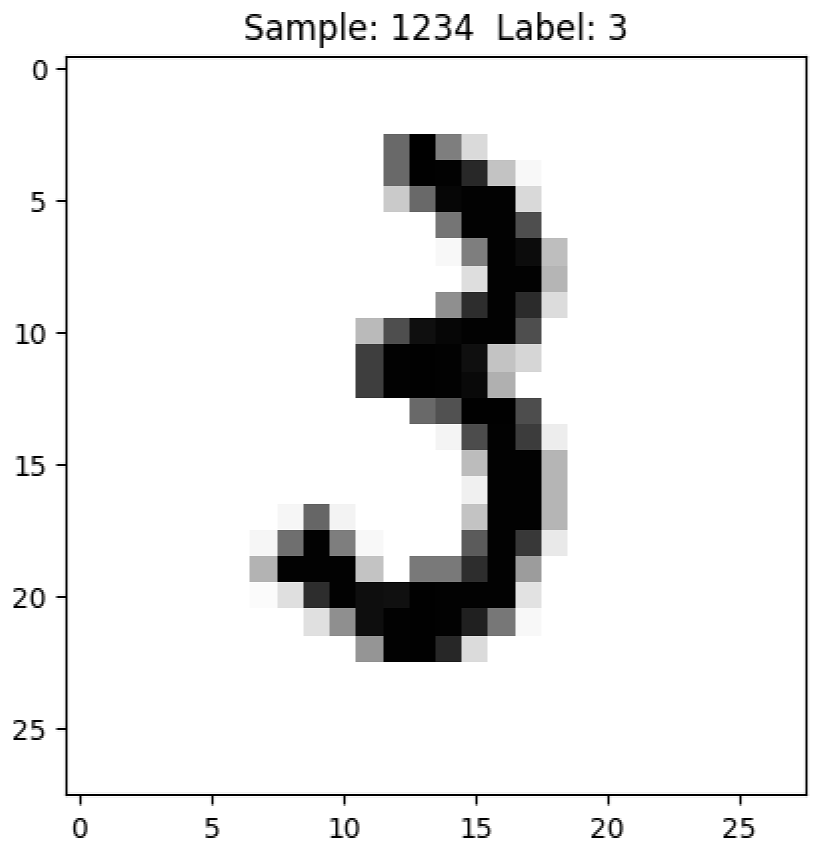
A run the best of every hyperparameter was carried out to see if the highest accuracy could be achieved, interestingly the accuracy was lower that the majority of previous runs at 0.0980. This supports that the parameters are interdependent and affect one another, so simply combining the best of all is not actually going to provide the best results. Over all the results the best accuracy was with three hidden layers, learning rate 0.5, neurons 512, and batch size 100.

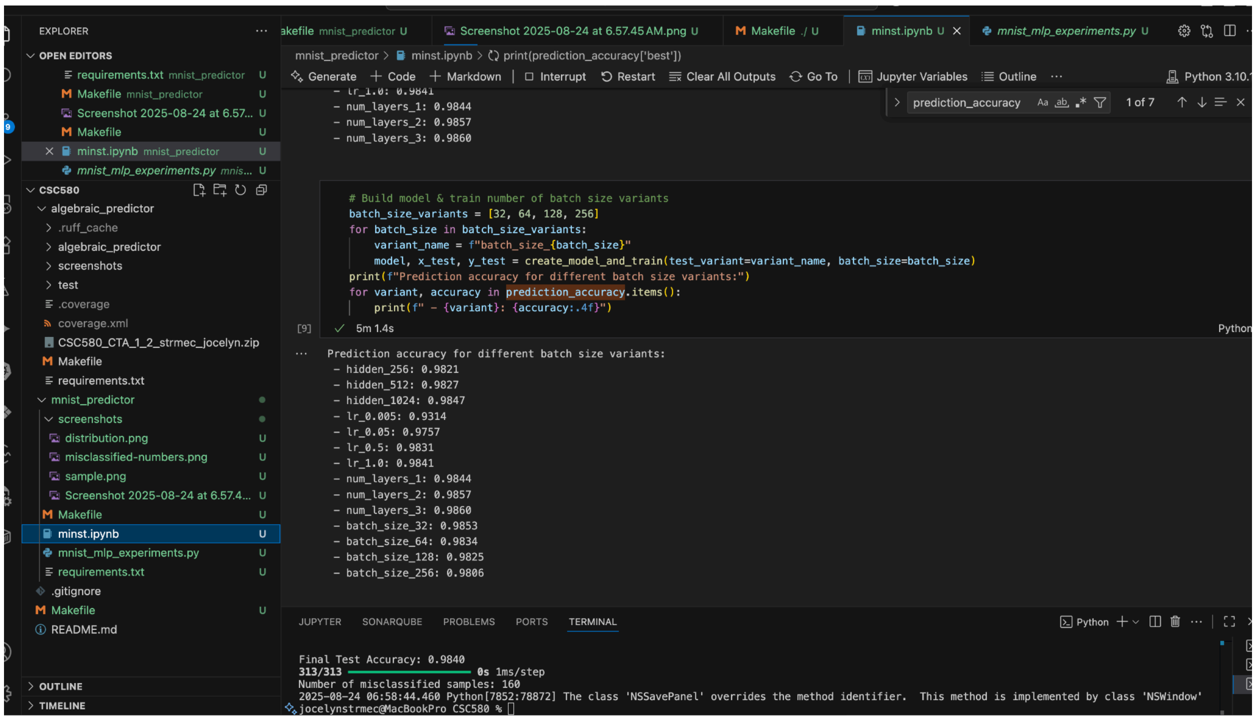
References

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., & Amodei, D. (2020). *Scaling Laws for Neural Language Models*. ArXiv.org. https://arxiv.org/abs/2001.08361




T:4 P:9
T:4 P:2
T:2 P:7
T:5 P:3
T:6 P:0
T:4 P:9
T:3 P:5
T:8 P:2
T:8 P:2
T:2 P:7
T:1 P:8
T:2 P:1
T:7 P:3
T:5 P:8
T:8 P:7
T:3 P:5
T:8 P:9
T:5 P:4
T:1 P:2
T:6 P:0
T:6 P:5
T:7 P:2
T:4 P:6
T:7 P:8
T:6 P:5





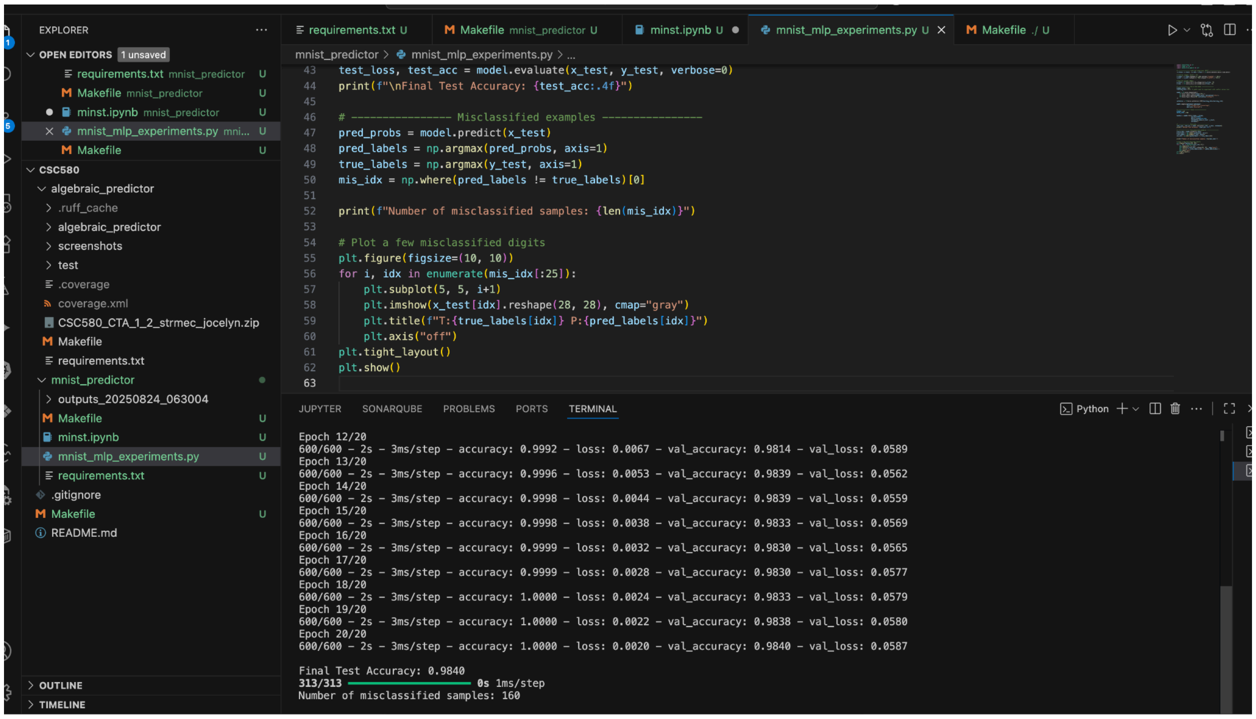
```
 – lr_1.0: 0.9841
 – num_layers_1: 0.9844
 – num_layers_2: 0.9857
 – num_layers_3: 0.9860
```

```python
# Build model & train number of batch size variants
batch_size_variants = [32, 64, 128, 256]
for batch_size in batch_size_variants:
    variant_name = f"batch_size_{batch_size}"
    model, x_test, y_test = create_model_and_train(test_variant=variant_name, batch_size=batch_size)
print(f"Prediction accuracy for different batch size variants:")
for variant, accuracy in prediction_accuracy.items():
    print(f" – {variant}: {accuracy:.4f}")
```

[9] ✓ 5m 1.4s

```
Prediction accuracy for different batch size variants:
 – hidden_256: 0.9821
 – hidden_512: 0.9827
 – hidden_1024: 0.9847
 – lr_0.005: 0.9314
 – lr_0.05: 0.9757
 – lr_0.5: 0.9831
 – lr_1.0: 0.9841
 – num_layers_1: 0.9844
 – num_layers_2: 0.9857
 – num_layers_3: 0.9860
 – batch_size_32: 0.9853
 – batch_size_64: 0.9834
 – batch_size_128: 0.9825
 – batch_size_256: 0.9806
```

```
Final Test Accuracy: 0.9840
313/313 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step
Number of misclassified samples: 160
2025-08-24 06:58:44.460 Python[7852:78872] The class 'NSSavePanel' overrides the method identifier.  This method is implemented by class 'NSWindow'
jocelynstrmec@MacBookPro CSC580 % 
```



```python
test_loss, test_acc = model.evaluate(x_test, y_test, verbose=0)
print(f"\nFinal Test Accuracy: {test_acc:.4f}")

# ---------------- Misclassified examples ----------------
pred_probs = model.predict(x_test)
pred_labels = np.argmax(pred_probs, axis=1)
true_labels = np.argmax(y_test, axis=1)
mis_idx = np.where(pred_labels != true_labels)[0]

print(f"Number of misclassified samples: {len(mis_idx)}")

# Plot a few misclassified digits
plt.figure(figsize=(10, 10))
for i, idx in enumerate(mis_idx[:25]):
    plt.subplot(5, 5, i+1)
    plt.imshow(x_test[idx].reshape(28, 28), cmap="gray")
    plt.title(f"T:{true_labels[idx]} P:{pred_labels[idx]}")
    plt.axis("off")
plt.tight_layout()
plt.show()

```

```
Epoch 12/20
600/600 – 2s – 3ms/step – accuracy: 0.9992 – loss: 0.0067 – val_accuracy: 0.9814 – val_loss: 0.0589
Epoch 13/20
600/600 – 2s – 3ms/step – accuracy: 0.9996 – loss: 0.0053 – val_accuracy: 0.9839 – val_loss: 0.0562
Epoch 14/20
600/600 – 2s – 3ms/step – accuracy: 0.9998 – loss: 0.0044 – val_accuracy: 0.9839 – val_loss: 0.0559
Epoch 15/20
600/600 – 2s – 3ms/step – accuracy: 0.9998 – loss: 0.0038 – val_accuracy: 0.9833 – val_loss: 0.0569
Epoch 16/20
600/600 – 2s – 3ms/step – accuracy: 0.9999 – loss: 0.0032 – val_accuracy: 0.9830 – val_loss: 0.0565
Epoch 17/20
600/600 – 2s – 3ms/step – accuracy: 0.9999 – loss: 0.0028 – val_accuracy: 0.9830 – val_loss: 0.0577
Epoch 18/20
600/600 – 2s – 3ms/step – accuracy: 1.0000 – loss: 0.0024 – val_accuracy: 0.9833 – val_loss: 0.0579
Epoch 19/20
600/600 – 2s – 3ms/step – accuracy: 1.0000 – loss: 0.0022 – val_accuracy: 0.9838 – val_loss: 0.0580
Epoch 20/20
600/600 – 2s – 3ms/step – accuracy: 1.0000 – loss: 0.0020 – val_accuracy: 0.9840 – val_loss: 0.0587

Final Test Accuracy: 0.9840
313/313 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step
Number of misclassified samples: 160
```