SOC Week 2 Report

Mahathi Nakka, 23B0965

June 2025

1 Key Findings

I built a CNN model to recognize handwritten digits using the MNIST dataset. My model performed very well, reaching about 98% accuracy on the test data. This shows that it can correctly identify most digits, even ones it hasn't seen before. I used a simple architecture but added data augmentation techniques like rotation, scaling, and shearing to help the model learn better. These techniques made the model more flexible in handling different handwriting styles. During training, I saw both the accuracy and loss improve over time, which means the model was learning properly. Still, it made a few mistakes, especially with digits that look similar, such as 4 and 9 or 5 and 6. This shows that there's still room to improve the model further.

2 Model architecture and design choices

I used a basic CNN structure in my model, which has two convolutional layers followed by max pooling. This setup helps the model learn step by step—first it picks up simple things like edges, and later it learns more complex patterns like the shape of digits. One special thing I added was Adaptive Average Pooling. This makes sure the feature maps are always the same size (7x7), which makes the model simpler and more flexible, especially if the input size changes. I also added a dropout layer with a rate of 0.25 before the final layer to reduce overfitting. It does this by randomly turning off some neurons during training. At the end, the model uses a softmax layer to predict one of the 10 digit classes.

3 Training and Evaluation Results

During the 10 training epochs, I saw that both the training and test losses went down, and the accuracy kept getting better. The graphs of loss and accuracy showed that the model was learning well without overfitting. This means that the model size, the dropout layer, and the data augmentation all worked well together. After training, I tested the model and used the predictions to create a confusion matrix. This gave me a detailed look at how well the model did with

each digit. The model performed well overall, but it sometimes got confused between digits that look alike, like 3 and 5, or 4 and 9. Even people might find these hard to tell apart in messy handwriting.

4 Visuals and Output Samples

To see how well my model was learning, I created plots showing the loss and accuracy during training. These graphs helped me understand how the model improved over time. I also made a confusion matrix heatmap, which showed how well each digit was being predicted. Most of the correct predictions appeared along the diagonal, which means the model was mostly accurate. To find out where the model was struggling, I looked at some of the wrong predictions. For example, the model sometimes confused 5 with 3, or 9 with 4. These mistakes usually happened because of messy handwriting or unclear images. These visual results showed that the model works well, but also helped me see where it could be improved.

```
(virtual_env) mahathinakka@Mahathis-MacBook-Air ~/Desktop/dsaprac % python3 x.py
Epoch [1/10], Train Loss: 0.1909, Train Acc: 93.86%, Test Loss: 0.0422, Test Acc: 98.54%
Epoch [2/10], Train Loss: 0.0753, Train Acc: 97.72%, Test Loss: 0.0404, Test Acc: 98.70%
Epoch [3/10], Train Loss: 0.0606, Train Acc: 98.14%, Test Loss: 0.0232, Test Acc: 99.15%
Epoch [4/10], Train Loss: 0.0548, Train Acc: 98.33%, Test Loss: 0.0235, Test Acc: 99.21%
Epoch [5/10], Train Loss: 0.0485, Train Acc: 98.51%, Test Loss: 0.0220, Test Acc: 99.24%
Epoch [6/10], Train Loss: 0.0426, Train Acc: 98.63%, Test Loss: 0.0243, Test Acc: 99.16%
Epoch [7/10], Train Loss: 0.0399, Train Acc: 98.75%, Test Loss: 0.0233, Test Acc: 99.31%
Epoch [8/10], Train Loss: 0.0366, Train Acc: 98.89%, Test Loss: 0.0216, Test Acc: 99.33%
Epoch [9/10], Train Loss: 0.0368, Train Acc: 98.81%, Test Loss: 0.0174, Test Acc: 99.42%
Epoch [10/10], Train Loss: 0.0326, Train Acc: 98.96%, Test Loss: 0.0193, Test Acc: 99.36%
Model saved!

Common Misclassifications:
True: 3, Predicted: 5
True: 8, Predicted: 2
True: 2, Predicted: 7
True: 2, Predicted: 7
True: 7, Predicted: 9
```

Figure 1: Output

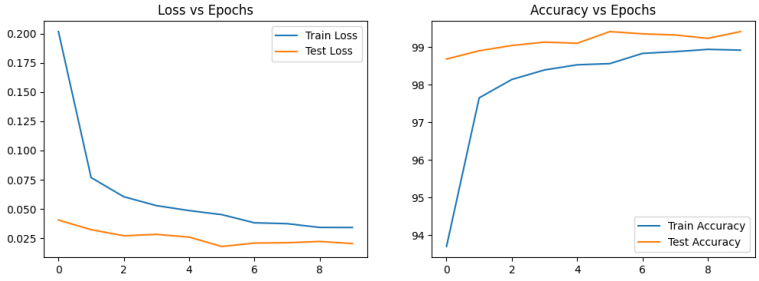

Figure 2: Loss vs epoch and Accuracy vs epoch

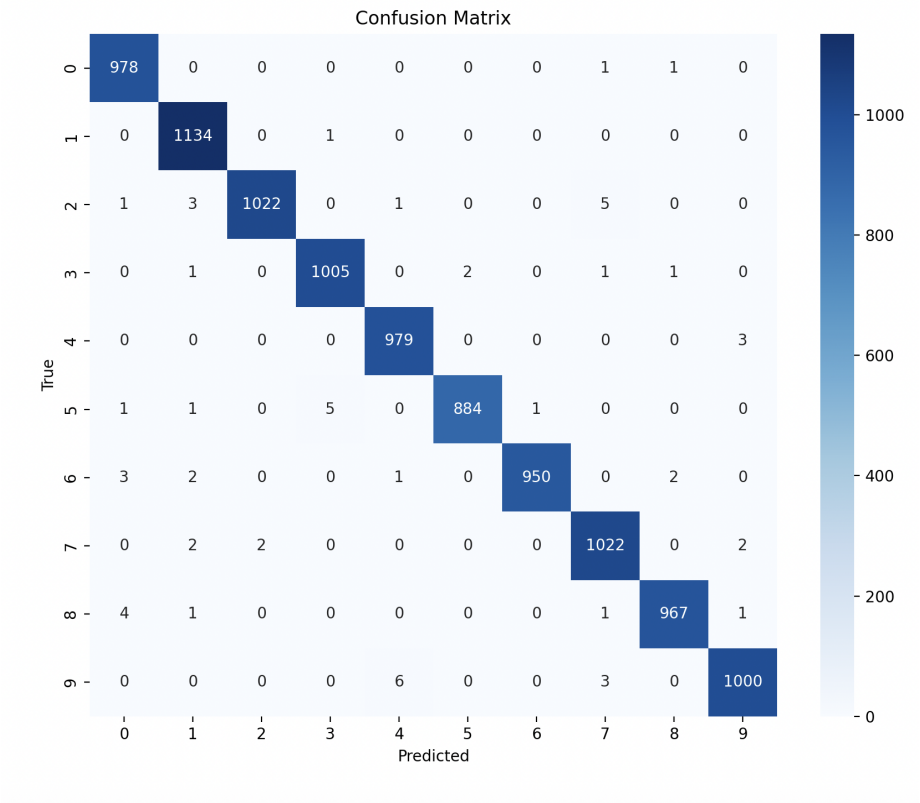

Figure 3: Confusion matrix

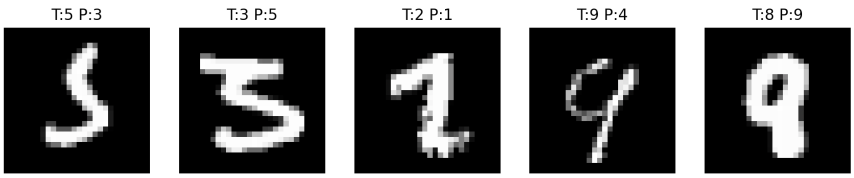

Figure 4: Misclassification