Reflection Journal – CNN Lab (MNIST Classification)

Working through this lab gave me a much more grounded understanding of how convolutional neural networks function in practice. Although I had learned about CNNs conceptually in earlier modules, implementing each component step-by-step made the architecture feel far more intuitive. One of my biggest learning moments came from seeing how image preprocessing directly affects model performance. Normalizing pixel values and adding the channel dimension were small changes in code, yet they made the data compatible with the structure that CNNs expect. This reminded me that many errors in deep learning pipelines come not from the model itself, but from mismatched shapes, preprocessing issues, or mislabeled data.

Building the CNN architecture was the most interesting part for me. I finally understood why convolutional layers use multiple filters and how deeper layers gradually shift from detecting simple edges to more abstract patterns. Experimenting with different numbers of filters or adding additional convolutional layers helped me see how model depth and capacity relate to accuracy and overfitting. Similarly, using MaxPooling layers showed how reducing spatial dimensions not only speeds up training but also makes the model more robust to small variations in the input images.

Training the model made the earlier theoretical ideas feel tangible. Changing the batch size and epoch count let me observe their impact on convergence speed and stability. A smaller batch size introduced more fluctuation in the loss curve but sometimes generalized slightly better, while a larger batch made the learning curve smoother but demanded more memory. Using the Adam optimizer also saved me from manually tuning the learning rate, and categorical cross-entropy felt appropriate once I saw how one-hot encoding aligned with the softmax output layer.

One challenge I encountered was understanding how different architecture choices influenced overfitting. Initially, the model performed extremely well on training data but lagged behind on validation accuracy. Adding a dropout layer immediately improved generalization, which helped me appreciate dropout not as a mysterious "trick" but as a statistical way of preventing the network from relying too heavily on any one set of activations. This lab also reinforced the importance of monitoring validation performance rather than relying solely on training accuracy.

Evaluating the model on the test set helped me reflect on what "good performance" really means. MNIST is a simple dataset, so achieving over 98 percent accuracy is expected. However, even with such strong numbers, I could see how small architectural changes—such as adding another convolutional layer or adjusting filter sizes—could push the model slightly higher or cause overfitting. This made me more aware of the trade-offs in designing neural networks and the need to balance model complexity with training stability.

Overall, this lab strengthened my conceptual and practical understanding of CNNs. It helped me appreciate how each preprocessing step, architectural decision, and hyperparameter affects the final performance. Moving forward, I want to explore more modern architectures such as residual networks and depthwise separable convolutions to see how these innovations improve efficiency. I also want to try training on more complex datasets where CNN limitations become more visible. This experience made deep learning feel less abstract and more like a system of coordinated design choices, each contributing to how well a model can learn from visual data.