Reflective Journal: Building a CNN for Digit Classification

During this lab, I discovered how Convolutional Neural Networks are structured and how they differ from traditional neural networks. I learned that CNNs use layers like *Conv2D* and *MaxPooling2D* to automatically detect features in images, such as edges or shapes, without needing manual feature extraction. This was new to me because I had only worked with fully connected neural networks before, which take flattened data instead of 2D images. Understanding the purpose of filters, kernels, and feature maps helped me realize how CNNs are designed for computer vision tasks like digit recognition. This lab connected to my previous knowledge of neural networks, especially concepts like activation functions, loss functions, and optimization. Before, I understood that neural networks learn by minimizing loss and adjusting weights, but I didn't fully know how they handle images. This lab helped me see how CNNs extend those same principles by applying them spatially—through convolution and pooling—to handle visual data efficiently.

I was surprised by how accurate the CNN became after only a few epochs. The model achieved over **99% accuracy** on the MNIST dataset, which showed me how powerful deep learning models can be when trained properly. Another surprise was how adding just a few layers, like another Conv2D or Dropout layer, could significantly change performance and help prevent overfitting. I also didn't expect how quickly the model could learn handwritten digits with relatively simple code.

One challenge I faced was understanding how to reshape the dataset correctly for CNN input. At first, I removed the channel dimension for experimentational reasons, and the model didn't run. Another challenge was figuring out which loss function and optimizer to use when experimenting. I had to research why *categorical_crossentropy* is best for multi-class classification problems and why *Adam* is a good optimizer for balancing speed and performance. I overcame these challenges by carefully reading the comments and documentation provided in the lab and by looking up short TensorFlow and Keras tutorials. I also experimented by changing the number of filters and epochs to see how it affected training accuracy and loss. The main resources that helped me were the TensorFlow documentation, class notes on neural networks, and YouTube tutorials that explained CNNs visually. Watching animations of how convolutions work made it easier to understand how filters slide across an image and extract features.

Before this lab, I saw deep learning as a "black box" where data goes in and predictions come out. Now I understand how layers interact and how CNNs extract and combine visual information step by step. It made me appreciate how much math and structure go into what seems like magic at first glance. I would like to explore how CNNs are used in more complex tasks beyond MNIST, like face recognition, self-driving cars, or medical image analysis. I'm also interested in learning about how CNNs can be combined with other models like Recurrent Neural Networks (RNNs) or Transformers for video or sequence data. Even though I had a basic idea of what CNNs were, this lab gave me a new perspective on how efficient they are at learning patterns directly from images. It also made me realize that deep learning is not just about coding—it's about understanding how data flows and transforms through each layer.

In conclusion, this lab helped me connect theory to practice. I not only learned how to build and train a CNN model but also gained a clearer understanding of how deep learning can recognize patterns that humans take years to learn. It was challenging at times, but the process of experimenting,

debugging, and finally seeing the high accuracy made it rewarding. This experience strengthened my interest in artificial intelligence and gave me more confidence to explore advanced topics in machine learning.