**Simulation Documentation**

To start the project, I already had Python and Visual Studio Code installed on my computer, which saved me a bit of time. From there, I installed TensorFlow using pip directly in the terminal. I also had to install Node.js and npm, which I hadn’t used before in an AI context, but I learned they were needed to run the Edge Impulse CLI. While I didn’t end up using the CLI to upload the model, it was useful to see how Node.js helps support web-based tooling in AI workflows.

I loaded the MNIST dataset using TensorFlow and reshaped and normalized the image data to prepare it for training. I created a simple convolutional neural network with a couple of basic layers and used the Adam optimizer and sparse categorical cross entropy as the loss function. After training the model for five epochs, I evaluated it and got a test accuracy of 98.36%, which was high enough to feel confident that the model was ready for deployment.

Once training was complete, I converted the model to TensorFlow Lite using the built-in TFLiteConverter and saved the output as a .tflite file. I then went to Edge Impulse Studio, created an account, and started a new project. Uploading the model was straightforward with the web interface. I set the input shape to (28, 28, 1) and selected grayscale input (Other, not RGB), then set the output shape to 10 and labeled the classes 0 through 9.

For testing, I created some sample image files from the MNIST test set and uploaded them to Edge Impulse using the "Live Classification" feature. The model predicted the correct digit each time I tried it. I was able to observe the output and class confidence live in the dashboard, which confirmed that the deployment and testing worked as expected.

**Reflective Journal**

I started this project with some experience in Python and TensorFlow, but I hadn’t worked with TensorFlow Lite before or uploaded a model to a service like Edge Impulse. I had used Roboflow in the past, mostly for data labeling and training image detection models, but Edge Impulse felt more focused on deployment, especially for edge devices. It was interesting to see how it handled model input and output configuration and how it allowed simulation through a web interface.

Setting up the environment gave me a chance to work with some new tools. I hadn't used Node.js in AI before, so that part was new to me. I realized it's more about enabling the command-line tools that interface with web services like Edge Impulse, which made more sense after seeing how the upload and device simulation tools worked.

Seeing the model work on uploaded MNIST digits inside the Edge Impulse platform was satisfying. It was also helpful to see how the platform handled model metadata, input scaling, and class labeling, things I usually have to manage manually. The testing worked well and the 98.36% accuracy was consistent with what I expected. Even though I used only a small set of sample inputs for testing, the classification was accurate and showed that the model deployed correctly.

Overall, I learned a lot about deploying models in a more production-like setting. Edge Impulse gave me a clearer idea of how models could be used in real edge environments, not just notebooks. Even though it’s a simulation, it feels like a step closer to real-world applications.