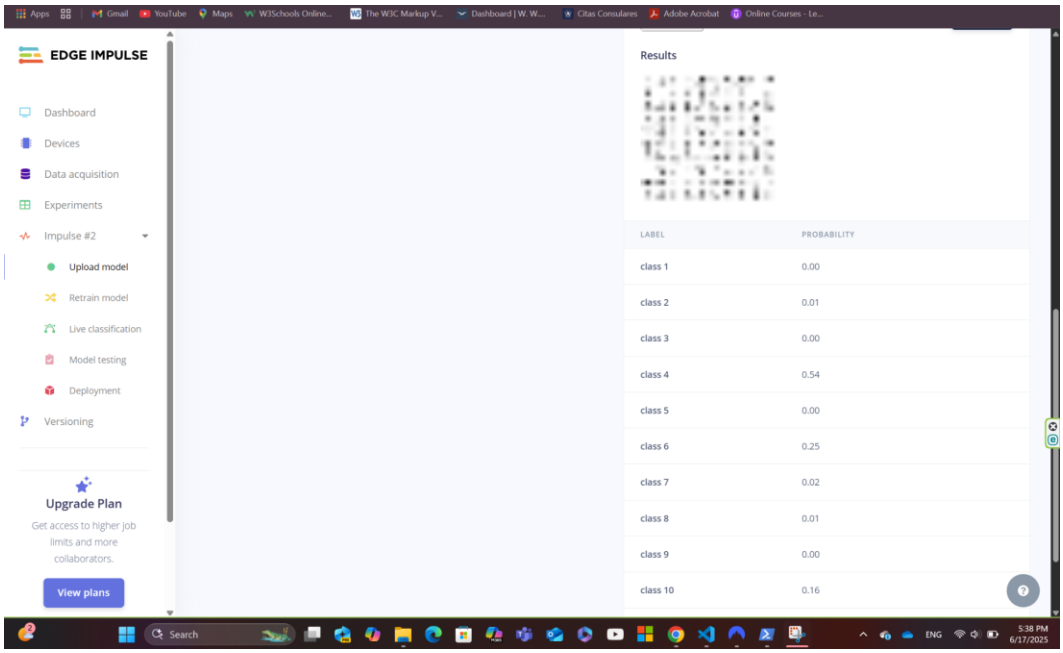
L03_Eduardo_Cabrera-Lopez_ITAI3377 - Reflective Journal

Doing this lab offered me a far better practical knowledge of what it actually means to apply an artificial intelligence model on a simulated edge device. Most of my work with artificial intelligence and machine learning up until recently has been on ordinary computers or cloud-based systems. Using TensorFlow, I trained models; investigated datasets; and assessed accuracy and loss among other performance measures. But this lab was the first time I had to think through how to use a functional model and really get it ready for use in a confined space, say an embedded system or microcontroller.

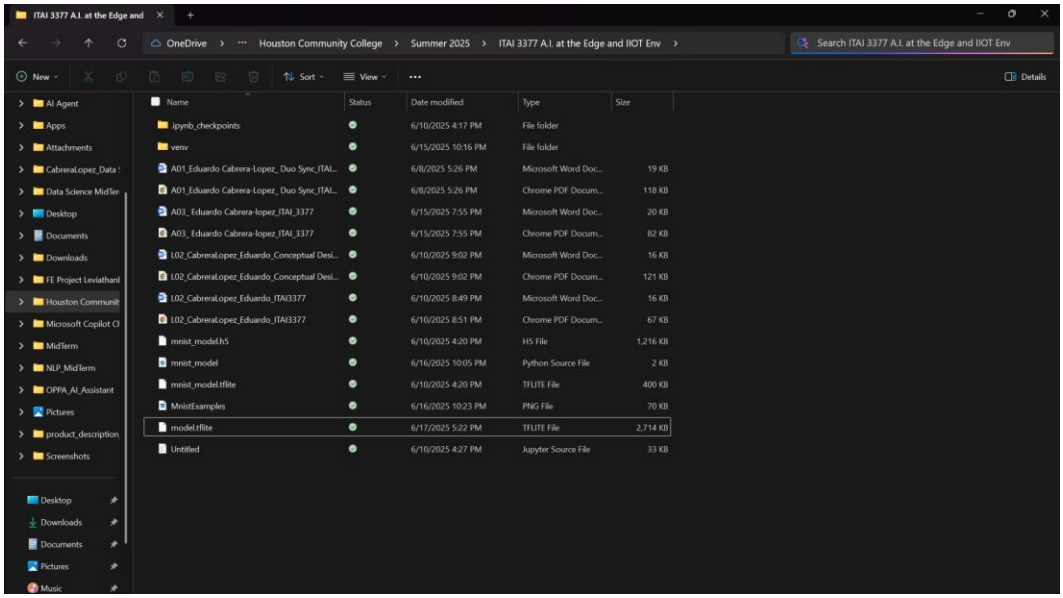


Though it was somewhat simple, organizing the surroundings helped me understand the number of instruments used in the artificial intelligence process. I wrote and ran the Python script using Visual Studio Code; Python itself handles data handling and model training; the Edge Impulse CLI and web dashboard simulate deployment. Installing TensorFlow and Edge Impulse PowerShell forced me to change my execution policies, which was a helpful lesson in managing system-level constraints and security on Windows.

Although training the CNN on the MNIST dataset was familiar ground, what caught my attention was the TFLite Lite converter approach used to translate the model. This was a step I had never taken before, and learning how crucial model optimization is for distributing to edge devices opened my eyes. While the original model performed well during training, it had to be compressed and streamlined to function well on devices with low RAM, CPU, and storage.

Once I moved from trying to utilize the CLI—which resulted in an error—to utilizing the

web dashboard, uploading the.tflite model to Edge Impulse was seamless. Edge Impulse's performance approximations for various hardware targets—including low-end MCUs and high-end CPUs—interestingly caught me. This kind of real-time profiling enabled me to see how, depending on where it runs, the same model could behave somewhat differently.



Using a test MNIST digit, running live classification on the uploaded model helped me to clearly visualize how deployment could appear in a practical use. Understanding the trade-offs between model complexity and inference speed came from the confidence scores and latency numbers. I particularly valued Edge Impulse's UI's simplicity in allowing one to replicate deployment without a real device.

Using Edge Impulse CLI and the API key presented one of my difficulties. Originally trying to post my model using VS Code, I found the CLI does not support.tflite file uploads. After some study and trial-and-error, I came to see the dashboard was the right approach. Though first annoying, this experience sharpened my troubleshooting abilities and gave me more confidence using tools particular to platforms.

Ultimately, this lab enabled me to close the distance between model construction and model implementation—a process sometimes disregarded in conventional machine learning instruction. I now see that creating a model mark only one stop on the road. A key ability is getting it ready to run effectively on the field; this simulation gave me first glimpse of what that looks like.