ITAI 3377

Professor Patricia McManus

Hoang Dinh

Lab Report: Deploying an AI Model on a Simulated Edge Device

**Introduction**

This lab aimed to deploy a simple AI model on a simulated edge device to gain practical experience with edge computing and AI integration. The MNIST dataset was used for image classification.

**Procedure**

Step 1: Set Up the Environment

Download and install **Python** from Python’s official website. A quick note is that TensorFlow will be installed later, and it only supports Python version 3.9 to 3.12, so the latest version 3.12.9 was chosen.

A screenshot of a computer program

AI-generated content may be incorrect.

Download and install **Visual Studio Code** (VS Code) from Visual Studio Code’s official website.

Install **TensorFlow** in VS Code by entering the following command in VS Code’s Terminal:

pip install tensorflow

Use command “pip show tensorflow” in VS Code’s Terminal to show the installed version of TensorFlow



Download and install **Node.js** and **npm** from Node.js’s official website. Check the installed versions of them by using commands “node -v” and “npm -v” in VS Code’s Terminal.

A black text on a white background

AI-generated content may be incorrect.

Install Edge Impulse CLI by entering the command “npm install -g edge-impulse-cli” in VS Code’s Terminal.

Step 2: Prepare the Dataset

A computer code with text

AI-generated content may be incorrect.

Step 3: Train a simple AI Model

A screenshot of a computer program

AI-generated content may be incorrect.

Step 4: Convert the Model to TFLite

A screen shot of a computer

AI-generated content may be incorrect.

Step 5: Upload TFLite model to Edge Impulse

The instruction says using the command “edge-impulse-uploader --api-key <your-api-key> model.tflite” to upload the TFLite model to Edge Impulse. After getting many errors, I found out that the edge impulse uploader does not support “.tflite” file. Therefore, I had to go to Edge Impulse website and upload my model directly.

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

Step 6: Test and Validate the Model

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

The result was not accurate, and probabilities are not high for any classification. An attempt to improve the model’s performance was to increase the number of layers.

A computer code with many text

AI-generated content may be incorrect.

Results:

A screenshot of a white box

AI-generated content may be incorrect.

A screenshot of a cellphone

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**Conclusions**

The process of training and deploying the model was successful but its performance was not valid. Although the number of layers was increased, it still provide wrong classification outputs.

Reflective Journal

The purpose of this lab was to train a Convolutional Neural Network (CNN) using TensorFlow and the MNIST dataset, then convert the trained model into a TensorFlow Lite (TFLite) model for deployment on lightweight devices.

I found this lab both interesting and challenging. One of the first obstacles I encountered was during the Python installation process. Initially, I installed the latest version, Python 3.13.2, but soon realized that TensorFlow was not compatible with it. When attempting to install TensorFlow, I repeatedly encountered the following error:

*“Could not find a version that satisfies the requirement tensorflow (from versions: none).”*

To troubleshoot, I tried upgrading pip, uninstalling and reinstalling Python, but nothing worked. Eventually, I discovered that TensorFlow only supports Python versions 3.9 to 3.12. After switching to Python 3.12.9, I was finally able to install TensorFlow successfully.

The most difficult challenge came when trying to upload the TFLite model to Edge Impulse. Initially, I used the command:

edge-impulse-uploader --api-key YOUR\_API\_KEY\_HERE model.tflite

However, this resulted in an error stating that the uploader.js file was missing. I attempted several solutions, including running different commands with npm and npx, as well as uninstalling and reinstalling the Edge Impulse CLI, but the issue persisted. After further troubleshooting, I found that I needed to install **Build Tools for Visual Studio**. Once installed, the uploader.js file was finally available.

Unfortunately, another issue arose, the uploader did not support .tflite files. After additional research, I discovered that I could manually upload the model directly onto the Edge Impulse platform, which ultimately solved the problem.

Despite successfully uploading the model, it did not recognize the digits correctly. As shown in the screenshots above, none of the predictions were accurate, and the probabilities were quite low (less than 0.5). To improve performance, I increased the number of layers in my CNN, which resulted in a slightly better probability of 0.76. However, the prediction was still incorrect.

At this point, I am continuing my research to explore different ways to enhance the model’s accuracy. Some potential next steps include experimenting with different architectures, fine-tuning hyperparameters, and increasing training data augmentation.