**Naomi Berumen**

**ITAI-3377 Patricia McManus**

**L03 Simulation Documentation**

**Simulation Documentation: Deploying a Simple AI Model on a Simulated Edge Device**

**Introduction**

This report details the end-to-end process of setting up, deploying, testing, and validating a simple AI model on a simulated edge device using Visual Studio Code, Python, TensorFlow, and Edge Impulse. Logs and observations from each stage are also included.

**Setup Process**

*Software Installation*

Before proceeding with model training and deployment, the following installations were completed:

* Python 3.12 (reinstalled due to previous incorrect installation)
* Visual Studio Code (including Python and Jupyter extensions)
* Node.js and npm (for Edge Impulse CLI)
* TensorFlow (installed using pip install tensorflow)
* Edge Impulse CLI (npm install -g edge-impulse-cli)

*Environment Configuration*

* The working directory was set up in C:\Users\User\Naomi.
* Visual Studio Code was used for code execution.
* Edge Impulse account was created and configured.
* API keys were generated and used for uploading the trained model.

*Logs from Setup*

* Python 3.12 successfully installed.
* TensorFlow installed with pip.
* Edge Impulse CLI installed with npm.
* Project folder: C:\Users\User\Naomi

**Model Training and Deployment**

*Training the Model*

A Convolutional Neural Network (CNN) was built using TensorFlow and trained on the MNIST dataset, consisting of handwritten digits (0-9). The training process involved:

* Data normalization (scaling values between 0 and 1).
* Model architecture:
  + Conv2D, MaxPooling, Flatten, Dense layers
  + Softmax activation for classification.
* Training Parameters:
  + 5 epochs (later found to be insufficient for accuracy)
  + Sparse categorical crossentropy loss function
  + Adam optimizer

The model was successfully saved as mnist\_model.h5.

*Training Logs*

Epoch 1/5 - Accuracy: 91.3% - Loss: 0.29 - Val Accuracy: 98.0% - Val Loss: 0.06

Epoch 2/5 - Accuracy: 98.2% - Loss: 0.05 - Val Accuracy: 98.1% - Val Loss: 0.05

Epoch 3/5 - Accuracy: 98.9% - Loss: 0.03 - Val Accuracy: 98.6% - Val Loss: 0.04

Epoch 4/5 - Accuracy: 99.3% - Loss: 0.02 - Val Accuracy: 98.3% - Val Loss: 0.05

Epoch 5/5 - Accuracy: 99.6% - Loss: 0.01 - Val Accuracy: 98.5% - Val Loss: 0.05

*Model Conversion to TFLite*

After training, the model was converted into the TensorFlow Lite (TFLite) format, which is optimized for deployment on edge devices. The TFLiteConverter was used to transform the trained Keras model (.h5) into a more efficient .tflite format. This conversion allows the model to be executed with lower memory and computational requirements. The converted model was then saved as model.tflite and uploaded to Edge Impulse using the command-line uploader. Once uploaded, Edge Impulse processed the model, making it available for further testing and deployment.

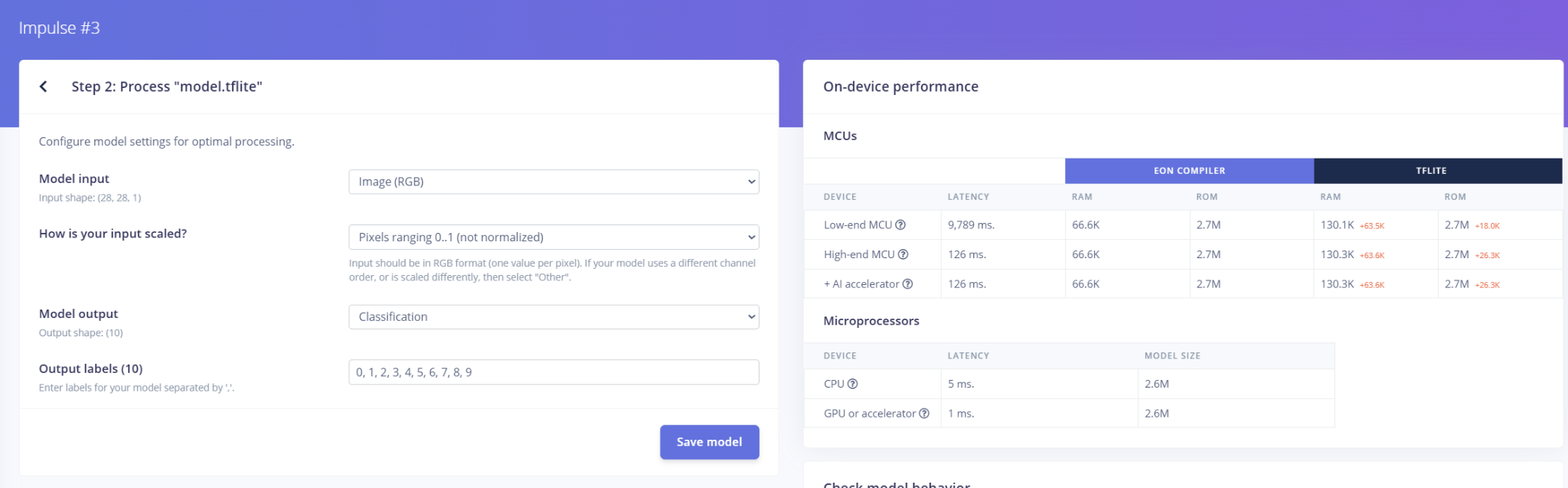
**Testing and Validation**

*Model Deployment on Edge Impulse*

The model was uploaded to Edge Impulse and configured under Image Classification.

Input was set to grayscale images (28x28, scaled 0...1).

Output labels were corrected to represent digits 0-9.



*Live Classification Tests*

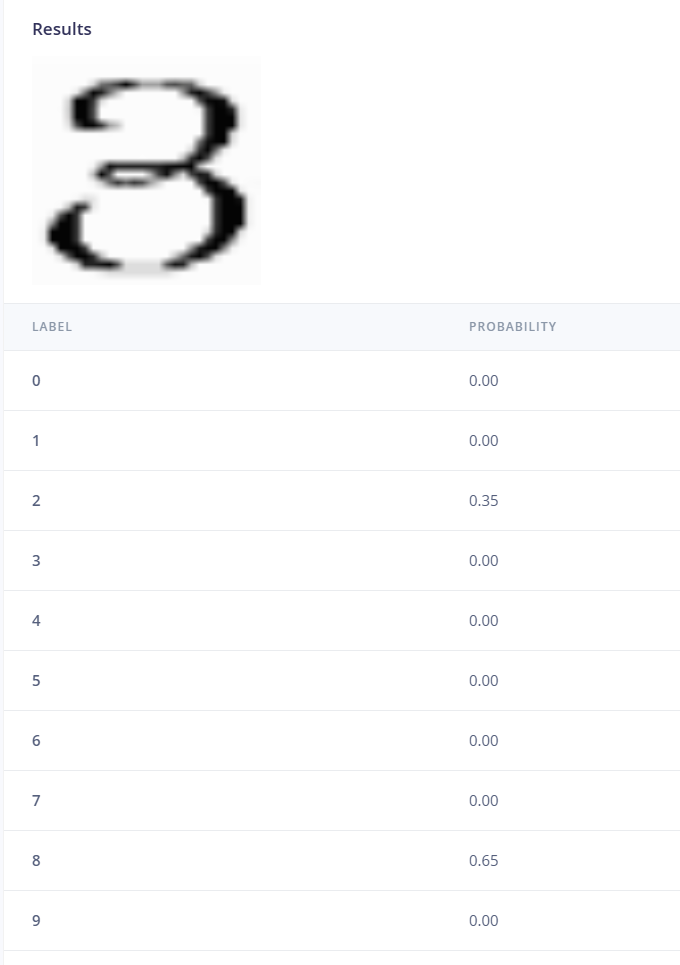
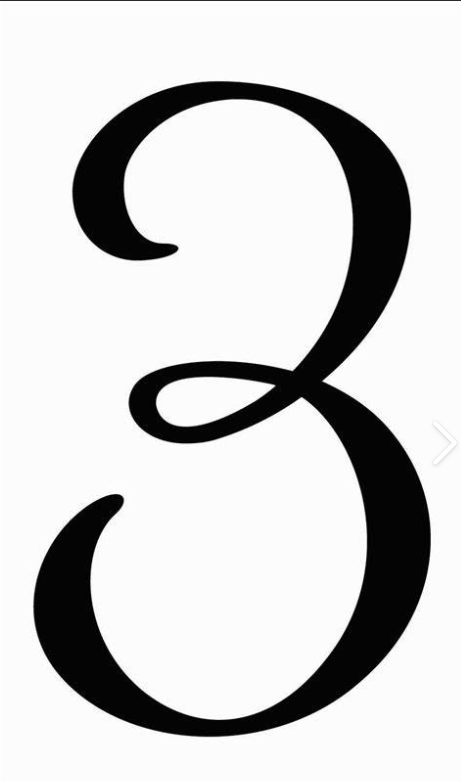
A series of classification tests were conducted using real digit images:

Test 1: Digit "3" → Model incorrectly classified as "2" (35%) and "8" (65%).

Test 2: Digit "0" → Model uncertain (36% for "0", 35% for "2").

Test 3: Digit "1" → Model performed poorly, with confidence spread randomly.

These results indicate that the model struggles to generalize and is essentially guessing, likely due to insufficient training epochs and missing preprocessing steps.



(These are the results for 3 as well as the numbers that were used during testing.)

**Conclusion and Future Improvements**

The deployment process was successful, but the model's performance on Edge Impulse was suboptimal. Future work will involve:

* Increasing training epochs to 15-20.
* Adding an extra convolutional layer for better feature extraction.
* Verifying Edge Impulse preprocessing matches training preprocessing.
* Running local predictions before deployment to catch issues earlier.

Overall, this exercise provided a great understanding of edge AI deployment, the limitations of low-epoch training, and real-world deployment considerations.