Florentin Degbo

ITAI 3377

Professor Patricia McManus

**L03 Deploying a Simple AI Model on a Simulated Edge Device using Visual Studio Code**

**Introduction**

This report documents deploying a simple AI model on a simulated edge device. It includes step-by-step instructions, code snippets, observations, and a reflective journal on the experience.

**Environment Setup**

**Required Installations**

To set up the environment, the following installations were completed:

* Installed Python from the official website.
* Installed Visual Studio Code (VS Code) from the official website.
* Installed TensorFlow using the command:
* pip install TensorFlow
* Installed Edge Impulse CLI by first installing Node.js and then running:
* npm install -g edge-impulse-cli

**Preparing the Dataset**

The MNIST dataset was loaded and preprocessed using TensorFlow.

**Code Snippet:**

Import tensorflow as tf

from tensorflow.keras.datasets import mnist

# Load the dataset

(x\_train, y\_train), (x\_test, y\_test) = mnist.load\_data()

# Normalize the data

x\_train, x\_test = x\_train / 255.0, x\_test / 255.0

# Reshape for CNN input

x\_train = x\_train.reshape((-1, 28, 28, 1))

x\_test = x\_test.reshape((-1, 28, 28, 1))

A computer code on a black background

AI-generated content may be incorrect.

**Training the AI Model**

The AI model was designed using a Convolutional Neural Network (CNN).

**Code Snippet:**

model = tf.keras.Sequential([

tf.keras.layers.Conv2D(32, (3, 3), activation='relu', input\_shape=(28, 28, 1)),

tf.keras.layers.MaxPooling2D((2, 2)),

tf.keras.layers.Flatten(),

tf.keras.layers.Dense(128, activation='relu'),

tf.keras.layers.Dense(10, activation='softmax')

])

# Compile the model

model.compile(optimizer='adam',

loss='sparse\_categorical\_crossentropy',

metrics=['accuracy'])

# Train the model

model.fit(x\_train, y\_train, epochs=5, validation\_data=(x\_test, y\_test))

A computer screen shot of a program code

AI-generated content may be incorrect.

**Model Conversion and Deployment**

**Convert Model to TFLite**

The trained model was converted to TensorFlow Lite for deployment on an edge device.

**Code Snippet:**

converter = tf.lite.TFLiteConverter.from\_keras\_model(model)

tflite\_model = converter.convert()

# Save the model

with open('model.tflite', 'wb') as f:

f.write(tflite\_model)

**Upload to Edge Impulse**

To upload the model to Edge Impulse, the following command was used:

edge-impulse-uploader --api-key <your-api-key> model.tflite

**Simulating and Testing the Model**

**Code Snippet:**

# Load the TFLite model

interpreter = tf.lite.Interpreter(model\_path="model.tflite")

interpreter.allocate\_tensors()

# Get input and output details

input\_details = interpreter.get\_input\_details()

output\_details = interpreter.get\_output\_details()

# Run inference on a test sample

import numpy as np

sample = np.expand\_dims(x\_test[0], axis=0).astype(np.float32)

interpreter.set\_tensor(input\_details[0]['index'], sample)

# Perform inference

interpreter.invoke()

# Get output

output\_data = interpreter.get\_tensor(output\_details[0]['index'])

print("Predicted Label:", np.argmax(output\_data))

**Observations and Performance Evaluation**

During testing, the model achieved the following results:

* **Accuracy:** (Insert Accuracy %)
* **Latency:** (Insert Latency time)
* **Observations:** The model performed well on simple digit recognition but struggled with noisy inputs.

**Add testing results from Edge Impulse**

import tensorflow as tf

from tensorflow.keras.datasets import mnist

# Load the dataset

(x\_train, y\_train), (x\_test, y\_test) = mnist.load\_data()

# Normalize the data

x\_train, x\_test = x\_train / 255.0, x\_test / 255.0

# Reshape for CNN input

x\_train = x\_train.reshape((-1, 28, 28, 1))

x\_test = x\_test.reshape((-1, 28, 28, 1))

**Reflective Journal**

**Observations on the Process**

The entire process was an insightful experience in deploying AI models on edge devices. The key takeaways include:

* The importance of **data preprocessing** in ensuring accurate predictions.
* The ease of **model conversion to TFLite** and its deployment on Edge Impulse.
* The significance of **inference speed and model optimization** for edge computing.

**Challenges Faced**

* **Installation Issues:** Configuring Edge Impulse CLI required troubleshooting due to Node.js dependencies.
* **Model Accuracy:** Fine-tuning the hyperparameters was necessary to achieve optimal accuracy.
* **Deployment Delays:** Uploading to Edge Impulse took longer than expected due to network issues.

**Improvements for Future Projects**

* Explore **quantization techniques** to reduce model size and improve latency.
* Utilize **more advanced CNN architectures** to boost accuracy.
* Optimize **data augmentation techniques** to enhance model robustness.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.