MNIST Classifier on Portenta H7

Goal: Deploy a quantized MNIST model on Arduino Portenta H7 for fast digit recognition.

Steps:

1. Train a simple CNN on MNIST

1. Install Python 3 (with pip)
- Python version installed: 3.11.9 (fully supported, stable, and will work smoothly with TensorFlow)
- Needed for training and quantising the model.
2. Open Command Prompt on Windows to install packages needed for MNIST
- Install TensorFlow in command prompt via :

```
pip install tensorflow
```

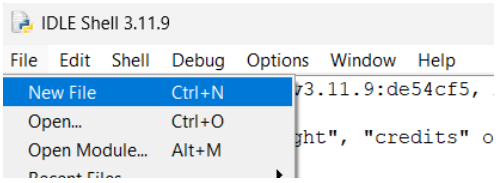
-Install NumPy in command prompt via

```
pip install numpy
```

```
->tensorflow) (2.19.2)
Requirement already satisfied: mdurl~=0.1 in c:\users\chuwe\appdata\local\programs\python\python311\lib\site-packages (
as>=3.10.0->tensorflow) (0.1.2)

[notice] A new release of pip is available: 24.0 -> 25.2
[notice] To update, run: python.exe -m pip install --upgrade pip

C:\Users\chuwe>pip install numpy
Requirement already satisfied: numpy in c:\users\chuwe\appdata\local\programs\python\python311\lib\site-packages (2.3.2

[notice] A new release of pip is available: 24.0 -> 25.2
[notice] To update, run: python.exe -m pip install --upgrade pip
```

2. Quantize to INT8 with TensorFlow Lite using a representative dataset

1. Open IDLE python (IDE for python)
2. Create a new Python file : File > New File



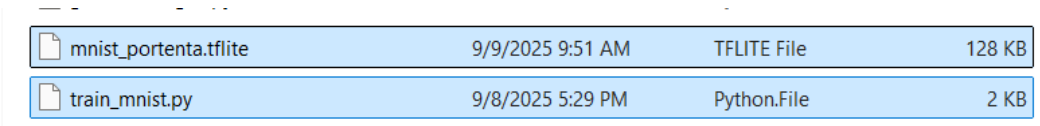
3. Paste code:

```
import numpy as np
from tensorflow.keras.datasets import mnist

# Load MNIST test data
(_, _), (x_test, y_test) = mnist.load_data()

# Quantization parameters (from previous step)
input_scale = 0.003921568859368563 # Replace with your actual values
input_zero_point = -128 # Replace with your actual values

with open("digits.h", "w") as f:
    f.write("// Sample MNIST digits 1-5 (quantized)\n")
    f.write("#ifndef DIGITS_H\n")
    f.write("#define DIGITS_H\n\n")
    f.write("// Quantization parameters\n")
    f.write(f"#define INPUT_SCALE {input_scale}f\n")
    f.write(f"#define INPUT_ZERO_POINT {input_zero_point}\n\n")

    for d in range(1, 6):
        idx = np.where(y_test == d)[0][0]
        arr = x_test[idx].astype(np.float32).flatten() / 255.0
        # Quantize the data
        quantized_arr = np.round((arr / input_scale) + input_zero_point)
        quantized_arr = np.clip(quantized_arr, -128, 127).astype(np.int8)
        f.write(f"const int8_t digit{d}[28*28] = {{\n")
        for i, val in enumerate(quantized_arr):
            f.write(f"0x{val & 0xFF:02X}, ")
            if (i+1) % 28 == 0:
                f.write("\n")
        f.write("};\n\n")
    f.write("#endif // DIGITS_H\n")
```

4. Save it as train_mnist.py
5. Run the script to start training the model from Command Prompt:

```
cd C:\Users\chuwe\AppData\Local\Programs\Python\Python311
python train_mnist.py
```

6. TFLITE file "mnist_portenta.tflite" is created in the same folder.



3. Generate properly quantized sample digits (1–5) as digits.h

In order to test on Portenta without a camera, export digits 1–5 into a header with proper quantisation.

1. Create a new Python file in the same folder: In IDLE → File → New File.
2. Paste the code:

```
import numpy as np
from tensorflow.keras.datasets import mnist

# Load MNIST test data
(_, _), (x_test, y_test) = mnist.load_data()

# Quantization parameters (replace with your actual values from training)
input_scale = 0.003921568859368563
input_zero_point = -128

with open("digits.h", "w") as f:
    f.write("// Sample MNIST digits 1-5 (quantized)\n")
    f.write("#ifndef DIGITS_H\n")
    f.write("#define DIGITS_H\n\n")
    f.write("// Quantization parameters\n")
    f.write(f"#define INPUT_SCALE {input_scale}f\n")
    f.write(f"#define INPUT_ZERO_POINT {input_zero_point}\n\n")

    for d in range(1, 6):
        idx = np.where(y_test == d)[0][0]  # find first sample of digit d
        arr = x_test[idx].astype(np.float32).flatten() / 255.0

        # Quantize the data
        quantized_arr = np.round((arr / input_scale) + input_zero_point)
        quantized_arr = np.clip(quantized_arr, -128, 127).astype(np.int8)

        f.write(f"const int8_t digit{d}[28*28] = {{\n")
        for i, val in enumerate(quantized_arr):
            f.write(f"0x{val & 0xFF:02X}, ")
            if (i + 1) % 28 == 0:
                f.write("\n")
        f.write("};\n\n")

    f.write("#endif // DIGITS_H\n")
```

3. Save it as generate_digits.py.
4. Open Command Prompt and go to your folder via:

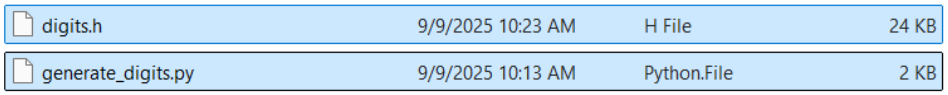
```
cd C:\Users\chuwe\AppData\Local\Programs\Python\Python311
```

can skip this step if the command prompt is already navigated to this folder.

5. Run generate_digits.py via:

```
python generate_digits.py
```

5. New file "digit.h" is created in same folder.



4. Convert to C array for Arduino

Convert your mnist_portenta.tflite into model_data.h so Arduino can use it

1. Create a new Python file in the same folder: In IDLE → File → New File.
2. Paste the code:

```
# convert_model.py
def convert_to_header(input_file, output_file, var_name):
    with open(input_file, "rb") as f:
        data = f.read()

    with open(output_file, "w") as f:
        f.write(f"unsigned char {var_name}[] = {{\n")
        for i, byte in enumerate(data):
            if i % 12 == 0:
                f.write("\n ")
            f.write(f"0x{byte:02x}, ")
        f.write("\n};\n")
        f.write(f"unsigned int {var_name}_len = {len(data)};\n")

convert_to_header("mnist_portenta.tflite", "model_data.h",
"g_mnist_portenta_tflite")
```

3. Save file as "convert_model.py" in the same folder.
4. Open Command Prompt and go to your folder via:

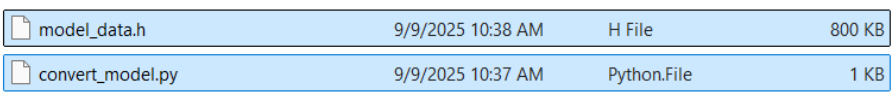
```
cd C:\Users\chuwe\AppData\Local\Programs\Python\Python311
```

can skip this step if the command prompt already navigated to this folder.

5. Run "convert_model.py" via:

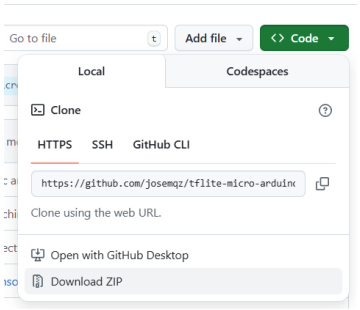
```
python convert_model.py
```

6. New H FILE "model_data.h" is created.

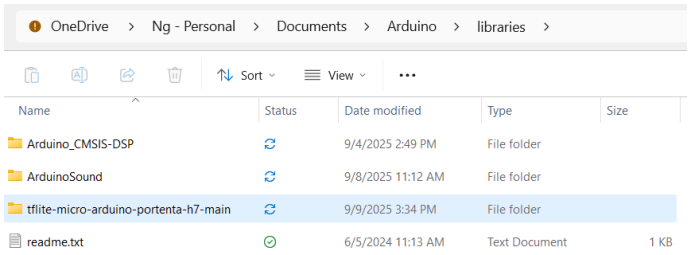


5. Flash to Portenta H7 via Arduino IDE (Connect Portenta H7 to PC via USB-C)

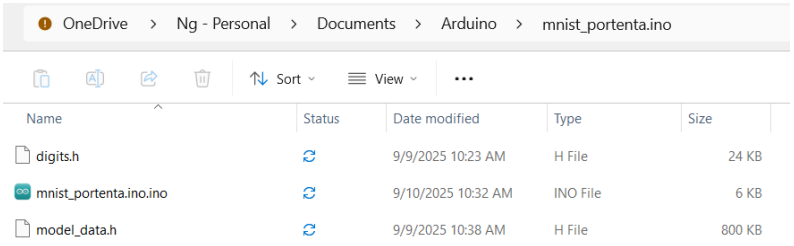
1. Install TensorFlow Lite for Microcontrollers
    - Navigate to [https://github.com/josemqz/tflite-micro-arduino-portenta-h7]: Click Code → Download ZIP



    - Go to the folder, extract ZIP file and copy the whole folder and paste to Arduino → libraries.



    - Restart Arduino, go to Sketch → Include Library, Arduino_TensorFlowLite is shown, proving the library is installed correctly
2. In Arduino IDE, make a new sketch: File → New.
3. Save it as mnist_portenta.ino.
4. Inside the sketch folder, paste your two header files:
    - model_data.h
    - digits.h



5. Paste the code:

```
#ifdef abs
#undef abs
#endif

// Include necessary libraries
#include "TensorFlowLite.h"
#include "tensorflow/lite/micro/all_ops_resolver.h"
#include "tensorflow/lite/micro/tflite_bridge/micro_error_reporter.h"
#include "tensorflow/lite/micro/micro_interpreter.h"
#include "tensorflow/lite/schema/schema_generated.h"

#include "model_data.h"   // Generated from .tflite
#include "digits.h"       // Quantized sample digits


// Globals
namespace {
  tflite::MicroErrorReporter micro_error_reporter;
  tflite::ErrorReporter* error_reporter = &micro_error_reporter;

  const tflite::Model* model = tflite::GetModel(g_mnist_portenta_tflite);
  tflite::MicroInterpreter* interpreter = nullptr;
  TfLiteTensor* input = nullptr;
  TfLiteTensor* output = nullptr;

  constexpr int kTensorArenaSize = 100 * 1024;
  uint8_t tensor_arena[kTensorArenaSize];
} // namespace

void setup() {
  Serial.begin(115200);
  while (!Serial) { ; }

  if (model->version() != TFLITE_SCHEMA_VERSION) {
    error_reporter->Report("Model version does not match Schema");
    return;
  }

  static tflite::AllOpsResolver resolver;
  static tflite::MicroInterpreter static_interpreter(
    model, resolver, tensor_arena, kTensorArenaSize);

  interpreter = &static_interpreter;

  if (interpreter->AllocateTensors() != kTfLiteOk) {
    error_reporter->Report("AllocateTensors() failed");
    return;
  }

  interpreter = &static_interpreter;

  if (interpreter->AllocateTensors() != kTfLiteOk) {
    error_reporter->Report("AllocateTensors() failed");
```

```
    return;
  }

  input = interpreter->input(0);
  output = interpreter->output(0);

  Serial.println("Portenta H7 MNIST ready!");
}

void loop() {
  // Select which test digit to classify
  const int8_t* test_image = digit1;  // Change to digit2, digit3, etc.

  // Fill input tensor with image data
  for (int i = 0; i < 784; i++) {
    input->data.int8[i] = test_image[i];
  }

  // Run inference
  if (interpreter->Invoke() != kTfLiteOk) {
    error_reporter->Report("Invoke failed");
    return;
  }

  // Find the index with the highest probability
  int8_t max_val = output->data.int8[0];
  int max_idx = 0;
  for (int i = 1; i < 10; i++) {
    if (output->data.int8[i] > max_val) {
      max_val = output->data.int8[i];
      max_idx = i;
    }
  }

  // Convert quantized output back to confidence
  float confidence = (max_val - output->params.zero_point) *
                     output->params.scale;

  // Print result
  Serial.print("Predicted digit: ");
  Serial.print(max_idx);
  Serial.print(", Confidence: ");
  Serial.println(confidence, 4);

  delay(2000);
}
```

6. Compile and Upload the code.
7. In Serial Monitor is shown:

```
Portenta H7 MNIST ready!
Predicted digit: 1, Confidence: 0.9961
```

8. TensorFlow Lite model is now running successfully on the Portenta H7 and making predictions. The output shows:

- Predicted digit: 1 - The model is classifying the input as digit "1"
- Confidence: 0.9961 - Very high confidence (~99.61%), which indicates the model is quite certain about its prediction

9. Test Different Digits:

- “digit2” is tried. Confidence is 0.9961. Predicted digit is changed from 1 to 2.

```
void loop() {
  // Select which test digit to classify
  const int8_t* test_image = digit2;  // Change to digit2, digit3, etc.

  // Fill input tensor with image data
  for (int i = 0; i < 784; i++) {
    input->data.int8[i] = test_image[i];
  }
```

```
Output   Serial Monitor ×
Message (Enter to send message to 'Arduino Portenta H7' on 'COM7')
Predicted digit: 1, Confidence: 0.9961
Predicted digit: 1, Confidence: 0.9961
Portenta H7 MNIST ready!
Predicted digit: 2, Confidence: 0.9961
Predicted digit: 2, Confidence: 0.9961
```

- “digit3” is tried. Confidence is 0.9961. Predicted digit is changed from 2 to 3.

```
63  void loop() {
64    // Select which test digit to classify
65    const int8_t* test_image = digit3;  // Change to digit2, digit3, etc.
66
67    // Fill input tensor with image data
68    for (int i = 0; i < 784; i++) {
69      input->data.int8[i] = test_image[i];
70    }
```

Output Serial Monitor ×

Message (Enter to send message to 'Arduino Portenta H7' on 'COM7')

```
Predicted digit: 2, Confidence: 0.9961
Predicted digit: 2, Confidence: 0.9961
Portenta H7 MNIST ready!
Predicted digit: 3, Confidence: 0.9961
Predicted digit: 3, Confidence: 0.9961
```

- The confidence remains the same (around 0.9961) due to how quantisation and dequantization work in TensorFlow Lite. When the model outputs its predictions, the values are stored in int8 format. To make them human-readable, they are converted back to floating-point using the formula (value-zero_point) * scale. Since the model is very confident about the correct digit in MNIST, the winning class almost always takes the maximum possible output value after quantisation. As a result, when you dequantize, the top score repeatedly maps to nearly the same number (≈0.9961). This does not mean the model yields the same result for different digits—it still selects different indices (digit 1, digit 2)—but the reported confidence saturates at a near-constant maximum because the quantised output range is limited.