Hochschule Emden/Leer
Department of Electrical and Computer Engineering
Industrial Informatics

University of Applied Sciences
HOCHSCHULE EMDEN·LEER

Tensorflow Lite

Machine Learning framework

Manoj Selvaraju

6. Juni 2024

1. TensorFlow Lite
2. Benefits of TensorFlow Lite
3. Core Components of TensorFlow Lite
4. TensorFlow Lite Working
5. TF Lite Model Conversion
6. TFLite Interpreter
7. Getting started with TensorFlow Lite in Portenta H7 using Arduino IDE

8 Tensorflow Lite Installation

9 Quellen

TensorFlow Lite I

Overview

- What is TensorFlow Lite?
  - TensorFlow Lite is a lightweight version of TensorFlow, which is an open-source machine learning framework developed by Google.
  - TensorFlow Lite is specifically designed for mobile and edge devices, allowing machine learning models to run efficiently on smartphones, embedded systems, and other devices with limited computational resources.
  - TensorFlow Lite optimizes machine learning models for deployment on these devices by providing tools for model conversion, model optimization, and inference.

Benefits of TensorFlow Lite I

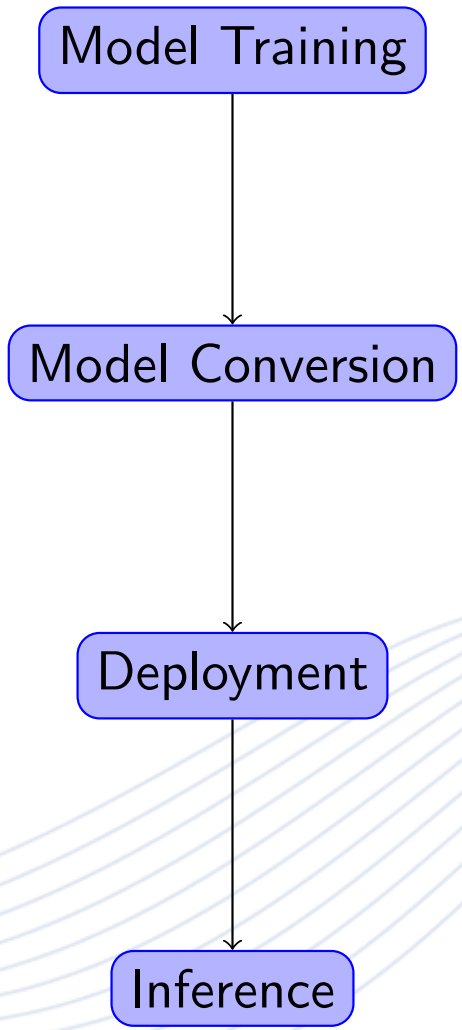
Key benefits

- Low Latency: Faster inference on-device.
- Reduced Size: Smaller model and binary sizes suitable for devices with limited resources.
- For common machine learning tasks such as image classification, object detection, pose estimation, question answering, text classification, etc. on multiple platforms.

Core Components of TensorFlow Lite I

- Interpreter: Executes the model.
- Converter: Converts TensorFlow models into a compressed flat buffer format.
- Supported Ops: Set of operations optimized for mobile and embedded devices.

TensorFlow Lite Working I



TensorFlow Lite Working I

Step-by-Step working

1. Model Training:
   - Train a model using TensorFlow on a desktop or cloud environment.
2. Model Conversion:
   - Convert the trained model to TFLite format (.tflite) using the TFLite Converter
3. Deployment:
   - Deploy the TFLite model on mobile, embedded, or IoT devices.
4. Inference:
   - Run the model using TFLite Interpreter to perform predictions on the device.

TensorFlow Lite Model Conversion I

- **TFLite Converter**:
  - The TensorFlow Lite converter takes a TensorFlow model and generates a TensorFlow Lite model (an optimized FlatBuffer format identified by the .tflite file extension)
  - You can convert your model using the Python API or the Command line tool
  - Command-line Tool: `tflite_convert`
  - Python API: `tf.lite.TFLiteConverter`
- **Optimization Techniques**:
  To make machine learning models more efficient for deployment on mobile and edge devices
  - **Quantization**: Reduces model size and latency.
  - **Pruning:** Removes unnecessary parts of the model.
  - **Clustering:** Groups similar weights to reduce model complexity.

TFLite Interpreter I

1. **Loading Model**
   - Load the TFLite model onto the device.
2. **Allocating Tensors**
   - Allocate memory for input and output tensors.
3. **Setting Input Tensors**
   - Set input tensor values with data for inference.
4. **Running Inference**
   - Execute the model with input data.
5. **Getting Output Tensors**
   - Retrieve output tensor values for results.

NOTE: A tensor refers to a multi-dimensional array or a generalized vector. It represents the basic unit of data in TensorFlow Lite and is used to store input data, intermediate calculations, and output predictions

Code

```
import tensorflow as tf
# Load TFLite model and allocate tensors.
interpreter = tf.lite.Interpreter(model_path=
                                  "model.tflite")
interpreter.allocate_tensors()
# Get input and output tensors.
input_details = interpreter.get_input_details()
output_details = interpreter.get_output_details()
# Set the value of the input tensor.
interpreter.set_tensor(input_details[0]['index'],
                       input_data)

# Run the model.
interpreter.invoke()
# Get the value of the output tensor.
output_data = interpreter.get_tensor(output_details[0]
                                     ['index'])
```

Getting started with TensorFlow Lite in Portenta H7 using Arduino IDE

Tensorflow Lite Installation on PC I

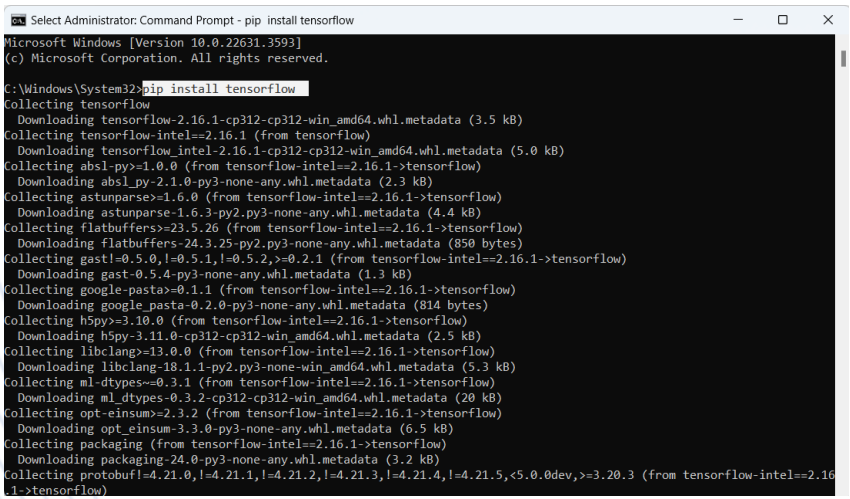


Figure1: TensorFlow Installation

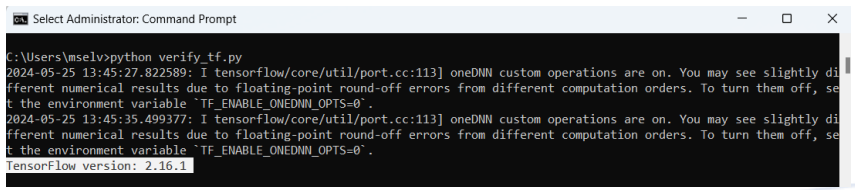


Figure2: TensorFlow Version

Tensorflow Lite Library Installation on Arduino IDE I

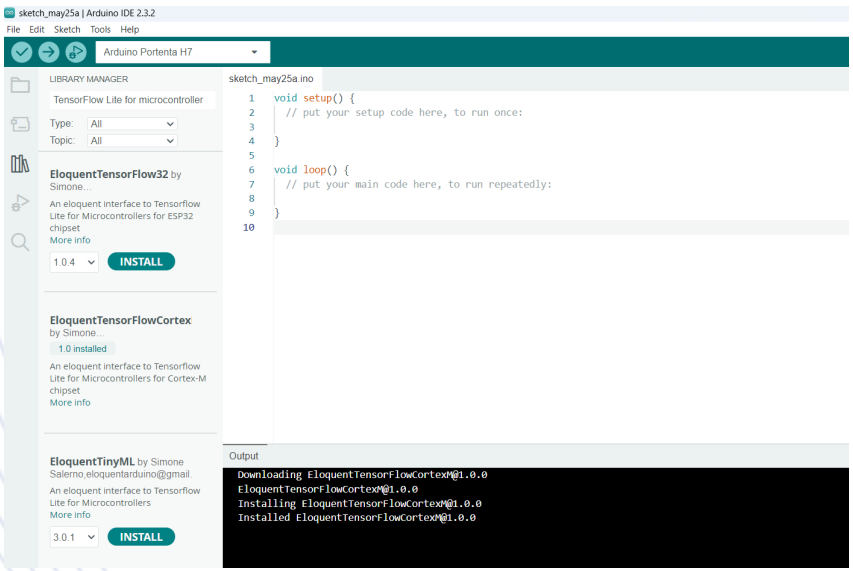


Figure1: TensorFlow Cortex Library

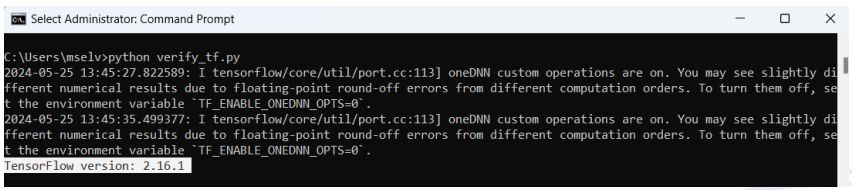


Figure2: TensorFlow Version

TensorFlow Model Training

```python
import tensorflow as tf
import numpy as np
input_shape = (2,) # Define the input shape
model = tf.keras.Sequential([ # Create a Seq.model
    tf.keras.layers.Input(shape=input_shape),
    tf.keras.layers.Dense(1)  # Single neuron for o/p
])
# Compile the model
model.compile(optimizer='sgd', loss='mse')
# Generate some training data
X_train = np.array([[1, 2], [3, 4], [5, 6]])
y_train = np.array([[3], [7], [11]])
# Train the model
model.fit(X_train, y_train, epochs=100, verbose=0)
# Save the model using recommended method for TF 2.x
tf.keras.models.save_model(model, 'addition_model.h5')
print("Model saved as 'addition_model.h5'")
```

Listing 1: Python code for training a TensorFlow model

STEPS I

1. Train a simple Keras model in TensorFlow and run the model in python to generate the file `addition_model.h5` in the same directory of tensorflow model.
2. Convert the Keras Model to TensorFlow Lite using the python script in the same directory and run the command `python convertModelTotflite.py` in command prompt
3. Using the python script convert the TensorFlow Lite Model to a C Array and run the python `ConvertTfliteToHeader.py` in the command prompt
4. Now include the model in the Arduino Sketch and run it on the Portenta H7.

TF Model to TF Lite

```python
import tensorflow as tf
from tensorflow.keras.losses import MeanSquaredError
# Load the Keras model with custom objects
model = tf.keras.models.load_model('addition_model.h5', custom_objects={'mse': MeanSquaredError()})
# Create a concrete function from the Keras model
@tf.function(input_signature=[tf.TensorSpec(shape=[None, 2], dtype=tf.float32)])
def model_concrete_func(x):
    return model(x)
# Convert the concrete fn to a TF Lite model
converter = tf.lite.TFLiteConverter.from_concrete_functions([model_concrete_func.get_concrete_function()])
tflite_model = converter.convert()
# Save the TensorFlow Lite model to a file
with open('addition_model.tflite', 'wb') as f:
    f.write(tflite_model)
```

Listing 2: Python code to convert keras model to tflite

TF Lite Model to C Array Header

```python
import numpy as np # convert_tflite_to_header.py
with open("addition_model.tflite", "rb") as f:
    tflite_model = f.read()
tflite_model_as_c_array = np.array(list(tflite_model),
    dtype=np.uint8)
with open("addition_model.h", "w") as f:
    f.write("#ifndef ADDITION_MODEL_H\n")
    f.write("#define ADDITION_MODEL_H\n\n")
    f.write("unsigned char addition_model_tflite[] = {
        ")
    for i, byte in enumerate(tflite_model_as_c_array):
        if i % 12 == 0:
            f.write("\n")
        f.write(f"0x{byte:02x}, ")
    f.write("\n};\n\n")
    f.write("unsigned int addition_model_tflite_len =
        {};\n".format(len(tflite_model_as_c_array)))
    f.write("#endif // ADDITION_MODEL_H\n")
```

Listing 3: Python code to convert Tensorflow Lite model to C Array

Arduino Sketch to run the model

```
#include <TensorFlowLite.h>
#include <Arduino_TensorFlowLite.h>

// Include the TensorFlow Lite model
#include "simple_addition_model.h" // Assuming this is y

void setup() {
        // Initialize serial communication
        Serial.begin(9600);

        // Initialize TensorFlow Lite interpreter
        if (!TfLite.begin(model_data, model_data_size))
                Serial.println("Failed to initialize Ten
                while (1);
        }
}
```

Arduino Sketch to run the model

```
void loop() {
        // Example inference
        float input1 = 5.0;
        float input2 = 3.0;

        // Prepare input tensor
        TfLiteTensor* input = TfLite.getInputTensor(0);
        input->data.f[0] = input1;
        input->data.f[1] = input2;

        // Run inference
        TfLite.run();

        // Get output tensor
        TfLiteTensor* output = TfLite.getOutputTensor(0)
        float result = output->data.f[0];
```

Arduino Sketch to run the model

```
    // Print result
    Serial.print(input1);
    Serial.print("+");
    Serial.print(input2);
    Serial.print("=");
    Serial.println(result);

    // Wait
    delay(1000);
}
```

Thank you

for your attention

Quellen I

Bibliography / References

[Ten24] TensorflowLite. *TensorflowLite 2024*. 2024. URL: https://www.tensorflow.org/lite.