Lab Report: Deploying TFLite & ONNX Models on Raspberry Pi

Enugu Harish Reddy
12503787
Embedded Systems
Instructor: Prof. Tobias Schaffer

24-06-2025

1 Introduction

The main aim of this lab is to evaluate the performance of two pre-trained machine learning models deployed in TensorFlow Lite(TFLite) and ONNX formats on a Raspberry pi by creating a virtual Python environment on a resource constrained edge device. The main relevance of this lab is in the fields of edge computing and Internet of Things(IOT) where deploying efficient AI models on small, low-power devices like Raspberry pi, Arduino boards, Drones and Smart sensors, etc is critical and this also help in understanding how different format models behave on those devices and choosing the right strategy for real-time applications.

2 Methodology

The pre-trained models namely 'model.tflite' and 'model.onnx' were deployed on a Raspberry Pi using Python and windows power shell using the **scp** command. We ran both the models in a Virtual Python environment on the Raspberry pi and compared their runtime for evaluating their efficiency . Dummy input data matching the model's expected shape was generated, and inference was performed using Python scripts. Execution time was measured to compare the performance of both model formats.

2.1 Hardware/Device Used

- Raspberry Pi 4 Model B, ARM Cortex-A72 CPU, 4GB RAM
- Used as the edge device to run pre-trained ML models in TFLite and ONNX format.

2.2 Programming Language

- Python 3 with TensorFlow and ONNX Runtime support

2.3 Development Environment

- · Raspberry Pi OS (Linux-based) terminal environment
- Scripts written using Nano or Vim text editors directly on the Pi
- SSH used for remote access
- Virtual environment (venv) used to manage Python packages

2.4 Libraries/Frameworks Used

- **TensorFlow**: For running .tflite models using tf.lite.Interpreter()
- **ONNXRuntime**: For executing .onnx models using InferenceSession()
- **NumPy**: For creating dummy input arrays matching the model's input shape
- **Time**: For measuring inference runtime in seconds

2.5 Algorithm Steps

1. Set up and activate a Python virtual environment on the Raspberry Pi
2. Install required libraries: tensorflow, onnxruntime, and numpy
3. Transfer model.tflite and model.onnx to Raspberry Pi using scp
4. Write and run run_tflite.py to:
    - Load the TFLite model
    - Allocate tensors and prepare dummy input
    - Run inference and print output + inference time
5. Load the TFLite model
    - Load the ONNX model
    - Prepare dummy input
    - inference and log output + inference time

6. Compare the performance of both models based on speed and output.

2.6 Data Structures Used

- numpy.ndarray: For holding dummy input data
- float: Used for storing and printing inference time
- dict and list structures: To access input/output tensor information from models

2.7 Tools Used

- Raspberry Pi 4 Model B
- SSH (for remote access and file transfer)
- **scp** (for secure file transfer from local machine to Pi)
- Python Virtual Environment (venv)
- Terminal for running scripts and viewing output

2.8 File and Code Organization

- model.tflite – Pre-trained TFLite model
- model.onnx – Pre-trained ONNX model
- run_tflite.py – Python script to run TFLite inference
- run_onnx.py – Python script to run ONNX inference
- All files stored in the home directory of the Raspberry Pi for simplicity

2.9 Software and Hardware Used

- **Programming Language**: Python 3
- **Libraries**: NumPy, TensorFlow, ONNX Runtime
- **Hardware**: Raspberry Pi (CPU execution only, no GPU)

2.10 Code Repository

Since I have not published this code on Github, I would like to present the code here itself
The code includes:
1. TFlite Inference Script
2. ONNX Inference Script

TFLite Inference Script

The code for generating a TFLite model ands train it using dummy values is depicted below:-

```python
import numpy as np
import tensorflow as tf
import time

interpreter = tf.lite.Interpreter(model_path="model.tflite")
interpreter.allocate_tensors()
input_details = interpreter.get_input_details()
output_details = interpreter.get_output_details()

input_shape = input_details[0]['shape']
dummy_input = np.random.rand(*input_shape).astype(np.float32)

start = time.time()
interpreter.set_tensor(input_details[0]['index'], dummy_input)
interpreter.invoke()
output = interpreter.get_tensor(output_details[0]['index'])
end = time.time()

print("TFLite predicted class:", np.argmax(output))
print(f"TFLite inference time: {end - start:.6f} seconds")
```

Table 1: TFLite inference script inside a table

ONNX Inference Script

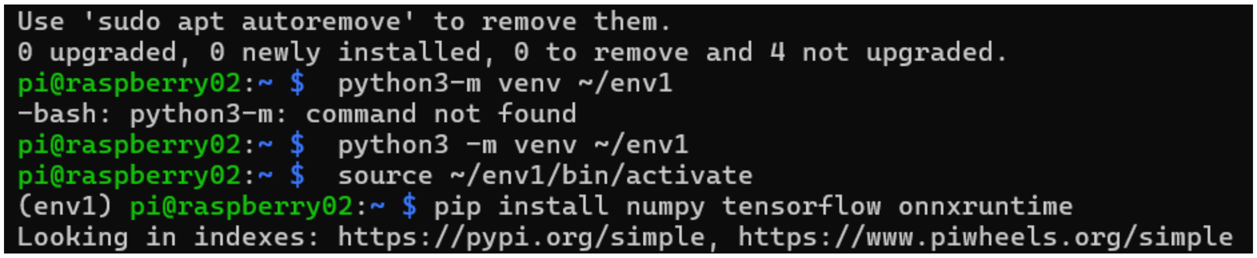
The code for generating a ONNX runtime format model that is to be deployed to raspberry pi is :-

```
import numpy as np
import onnxruntime as ort
import time

session = ort.InferenceSession
(”model.onnx”, providers=['CPUExecutionProvider'])
input_name = session.get_inputs()[0].name
input_shape = session.get_inputs()[0].shape

dummy_input = np.random.rand(*input_shape).astype(np.float32)
start = time.time()
output = session.run(None, {input_name: dummy_input})
end = time.time()

print(”ONNX predicted class:”, np.argmax(output[0]))
print(f”ONNX inference time: {end — start:.6f} seconds”)
```

Table 2: ONNX inference script inside a table

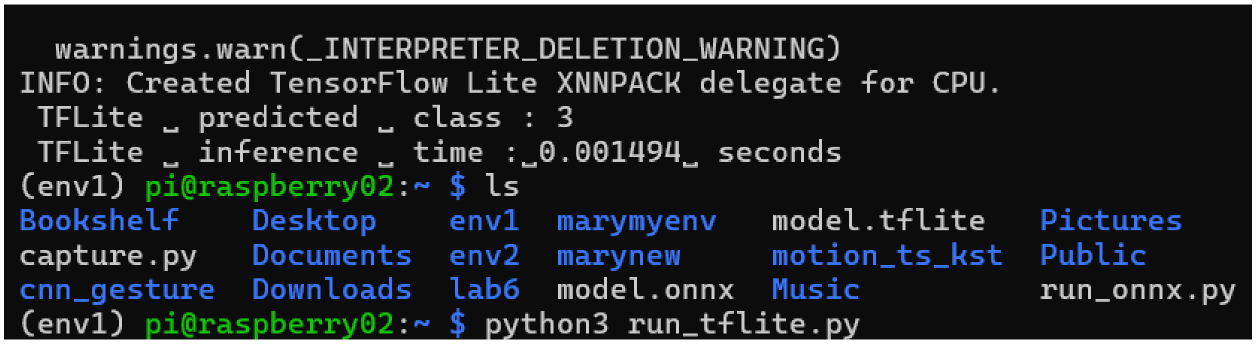
2.11 Code Implementation

The below pictures(screen shorts) depicts the Implementation of codes in a virtual Python environment(version 3) using Pre-trained Models. A virtual enivironment is created and the models were run using dummy inputs to get an output. Some random snapshots were taken to represent the implementation of the code and are presented below.



```
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 4 not upgraded.
pi@raspberry02:~ $  python3-m venv ~/env1
-bash: python3-m: command not found
pi@raspberry02:~ $  python3 -m venv ~/env1
pi@raspberry02:~ $  source ~/env1/bin/activate
(env1) pi@raspberry02:~ $ pip install numpy tensorflow onnxruntime
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
```

Figure 1: The picture shows the creation of a virtual python environment to run the models



```
  warnings.warn(_INTERPRETER_DELETION_WARNING)
INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
 TFLite ␣ predicted ␣ class : 3
 TFLite ␣ inference ␣ time :␣0.001494␣ seconds
(env1) pi@raspberry02:~ $ ls
Bookshelf    Desktop    env1  marymyenv   model.tflite   Pictures
capture.py   Documents  env2  marynew     motion_ts_kst  Public
cnn_gesture  Downloads  lab6  model.onnx  Music          run_onnx.py
(env1) pi@raspberry02:~ $ python3 run_tflite.py
```

Figure 2: Model files on Raspberry Pi

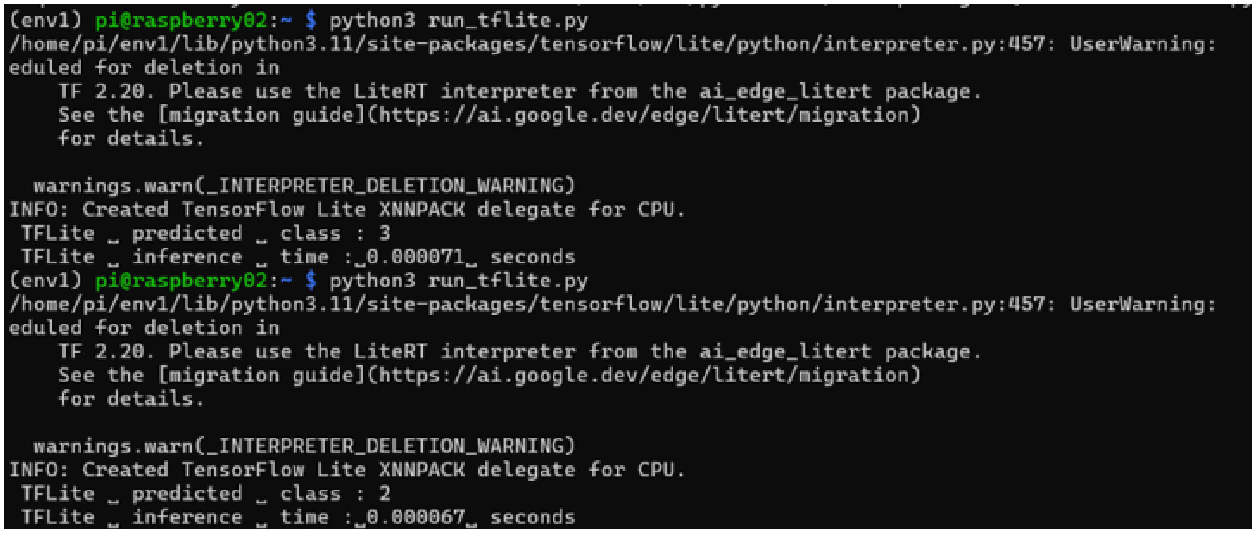
3 Results

After deploying the TFLite and ONNX models on the Raspberry Pi, each model was run twice using randomly generated dummy input data. The predicted classes and inference times for each run were recorded and are summarized in the table below.

The variations in predicted class are expected due to the use of random input on each run. All inferences were executed on CPU-only, without any hardware acceleration.

3.1 TFLite Inference Script

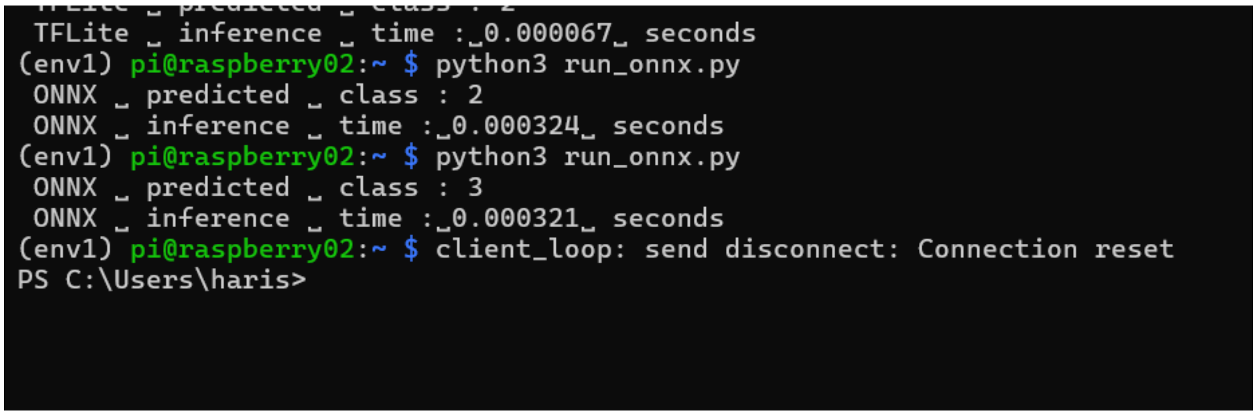
The results obtained while running the tensor flow model are:-



```
(env1) pi@raspberry02:~ $ python3 run_tflite.py
/home/pi/env1/lib/python3.11/site-packages/tensorflow/lite/python/interpreter.py:457: UserWarning:
eduled for deletion in
    TF 2.20. Please use the LiteRT interpreter from the ai_edge_litert package.
    See the [migration guide](https://ai.google.dev/edge/litert/migration)
    for details.

  warnings.warn(_INTERPRETER_DELETION_WARNING)
INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
 TFLite ␣ predicted ␣ class : 3
 TFLite ␣ inference ␣ time :␣0.000071␣ seconds
(env1) pi@raspberry02:~ $ python3 run_tflite.py
/home/pi/env1/lib/python3.11/site-packages/tensorflow/lite/python/interpreter.py:457: UserWarning:
eduled for deletion in
    TF 2.20. Please use the LiteRT interpreter from the ai_edge_litert package.
    See the [migration guide](https://ai.google.dev/edge/litert/migration)
    for details.

  warnings.warn(_INTERPRETER_DELETION_WARNING)
INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
 TFLite ␣ predicted ␣ class : 2
 TFLite ␣ inference ␣ time :␣0.000067␣ seconds
```

3.2 ONNX Inference Script

The results obtained while running the ONNX model in the virtual python environment are :-



```
 TFLite ␣ inference ␣ time :␣0.000067␣ seconds
(env1) pi@raspberry02:~ $ python3 run_onnx.py
 ONNX ␣ predicted ␣ class : 2
 ONNX ␣ inference ␣ time :␣0.000324␣ seconds
(env1) pi@raspberry02:~ $ python3 run_onnx.py
 ONNX ␣ predicted ␣ class : 3
 ONNX ␣ inference ␣ time :␣0.000321␣ seconds
(env1) pi@raspberry02:~ $ client_loop: send disconnect: Connection reset
PS C:\Users\haris>
```

4 FINAL OUTPUT

The following table classifies the outputs and identifies the efficient model in the experiment

From the table, We can clearly identify that TFLite model was more efficient than ONNX model according the results of the run values

Run	Format	Predicted Class	Inference Time (s)	Library Used
1	TFLite	2	0.000067	TensorFlow Lite
2	TFLite	2	0.000070	TensorFlow Lite
3	ONNX	2	0.000324	ONNX Runtime
4	ONNX	3	0.000321	ONNX Runtime

4 Challenges, Limitations, and Error Analysis

During the implementation and testing of the deployment of TFLite and ONNX models on the Raspberry Pi, several challenges were encountered, along with some minor errors and limitations. These aspects are important to reflect on, as they affect the repeatability, scalability, and overall usability of machine learning on edge devices.

4.1 Challenges Faced

· Deploying two different format pre-trained models into the raspberry pi
· Ensuring compatibility between model input shapes and dummy data
· Installing ONNXRuntime properly on Raspberry Pi with ARM architecture

4.2 Error Analysis

- Minor syntax mismatches while indexing tensors (['index'])
- Connection dropped unexpectedly during SSH session due (client_loop: send disconnect)

- The variation in predicted classes across runs is due to the use of randomized dummy input data, which naturally produces different outputs for each inference.

4.3 Limitations

- Only CPU inference tested; no GPU acceleration
- Models tested with dummy inputs, not real-world data, so the results only reflect performance, not actual model accuracy or reliability in practical applications.
- Power consumption, heat generation, and thermal throttling were not measured, though they are important in edge deployment scenarios.

5 Discussion

The lab results from the experiment support that the common observation that TFLite is better suited for lightweight, real-time applications on edge devices, while ONNX serves well in more flexible, cross-platform environments. The use of dummy inputs ensured safe testing, although it limits accuracy evaluation as the real-time values may affect the performance

This lab highlighted practical issues such as dependency compatibility, network transfer errors, and the importance of understanding input/output tensor handling, all of which are crucial in real-world embedded ML deployment

6 Conclusion

This lab successfully demonstrated how to deploy and run both TFLite and ONNX models on Raspberry Pi. TFLite showed better performance, reinforcing its suitability for mobile and embedded systems. Future work could explore real-world input data and GPU/accelerator compatibility.

Future improvements could include testing with real input data, adding more model types for comparison, enabling hardware acceleration, and monitoring power and thermal metrics for a more complete performance analysis.