TensorFlow Lite inference

The term *inference* refers to the process of executing a TensorFlow Lite model on-device in order to make predictions based on input data. To perform an inference with a TensorFlow Lite model, you must run it through an *interpreter*. The TensorFlow Lite interpreter is designed to be lean and fast. The interpreter uses a static graph ordering and a custom (less-dynamic) memory allocator to ensure minimal load, initialization, and execution latency.

This page describes how to access to the TensorFlow Lite interpreter and perform an inference using C++, Java, and Python, plus links to other resources for each supported platform.

TensorFlow Lite inference typically follows the following steps:

1. **Loading a model**
   You must load the .tflite model into memory, which contains the model's execution graph.
2. **Transforming data**
   Raw input data for the model generally does not match the input data format expected by the model. For example, you might need to resize an image or change the image format to be compatible with the model.
3. **Running inference**
   This step involves using the TensorFlow Lite API to execute the model. It involves a few steps such as building the interpreter, and allocating tensors, as described in the following sections.
4. **Interpreting output**
   When you receive results from the model inference, you must interpret the tensors in a meaningful way that's useful in your application.
   For example, a model might return only a list of probabilities. It's up to you to map the probabilities to relevant categories and present it to your end-user.

Linux Platform

On Linux platforms (including Raspberry Pi), you can run inferences using TensorFlow Lite APIs available in C++ and Python, as shown in the following sections.

Running a model

Running a TensorFlow Lite model involves a few simple steps:

1. Load the model into memory.
2. Build an Interpreter based on an existing model.
3. Set input tensor values. (Optionally resize input tensors if the predefined sizes are not desired.)
4. Invoke inference.
5. Read output tensor values.

Following sections describe how these steps can be done in each language.

Load and run a model in Python

The Python API for running an inference is provided in the tf.lite module. From which, you mostly need only tf.lite.Interpreter to load a model and run an inference.

The following example shows how to use the Python interpreter to load a .tflite file and run inference with random input data:

This example is recommended if you're converting from SavedModel with a defined SignatureDef. Available starting from TensorFlow 2.5

```
class TestModel(tf.Module):
  def __init__(self):
    super(TestModel, self).__init__()

  @tf.function(input_signature=[tf.TensorSpec(shape=[1, 10], dtype=tf.float32)])
  def add(self, x):
    '''
    Simple method that accepts single input 'x' and returns 'x' + 4.
    '''
    # Name the output 'result' for convenience.
    return {'result' : x + 4}


SAVED_MODEL_PATH = 'content/saved_models/test_variable'
TFLITE_FILE_PATH = 'content/test_variable.tflite'
```

```
# Save the model
module = TestModel()
# You can omit the signatures argument and a default signature name will be
# created with name 'serving_default'.
tf.saved_model.save(
    module, SAVED_MODEL_PATH,
    signatures={'my_signature':module.add.get_concrete_function()})

# Convert the model using TFLiteConverter
converter = tf.lite.TFLiteConverter.from_saved_model(SAVED_MODEL_PATH)
tflite_model = converter.convert()
with open(TFLITE_FILE_PATH, 'wb') as f:
  f.write(tflite_model)

# Load the TFLite model in TFLite Interpreter
interpreter = tf.lite.Interpreter(TFLITE_FILE_PATH)
# There is only 1 signature defined in the model,
# so it will return it by default.
# If there are multiple signatures then we can pass the name.
my_signature = interpreter.get_signature_runner()

# my_signature is callable with input as arguments.
output = my_signature(x=tf.constant([1.0], shape=(1,10), dtype=tf.float32))
# 'output' is dictionary with all outputs from the inference.
# In this case we have single output 'result'.
print(output['result'])
```

Another example if the model doesn't have SignatureDefs defined.

```
import numpy as np
import tensorflow as tf

# Load the TFLite model and allocate tensors.
interpreter = tf.lite.Interpreter(model_path="converted_model.tflite")
interpreter.allocate_tensors()

# Get input and output tensors.
input_details = interpreter.get_input_details()
output_details = interpreter.get_output_details()

# Test the model on random input data.
input_shape = input_details[0]['shape']
```

```
input_data = np.array(np.random.random_sample(input_shape), dtype=np.float32)
interpreter.set_tensor(input_details[0]['index'], input_data)

interpreter.invoke()

# The function `get_tensor()` returns a copy of the tensor data.
# Use `tensor()` in order to get a pointer to the tensor.
output_data = interpreter.get_tensor(output_details[0]['index'])
print(output_data)
```