tflite_model_conversion

March 2, 2020

Copyright 2019 The TensorFlow Authors.

[1]:
```
#@title Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
# https://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
```

0.1 Do necessary imports

https://www.tensorflow.org/lite/performance/post_training_quantization

[2]:
```
import copy
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
import numpy as np
import pandas as pd

# import codebase
import thermalModel_main as tmm
import thermalModel_groupB as tm_gb

import importlib
importlib.reload(tmm)
importlib.reload(tm_gb)
```

```
Using TensorFlow backend.
```

[2]:
```
<module 'thermalModel_groupB' from
'C:\\Users\\user\\Anaconda3\\lib\\thermalModel_groupB.py'>
```

0.1.1 Load data

[3]:
```
df = tm_gb.load_csv(filename = 'LDPRF_2097.csv',
#                       data_list = ['Program␣
↪time','Current','Voltage','AhCha','AhDch','Temp'],
                      features_list =␣
↪['runtime_s','Current','Voltage','AhCha','AhDch','Amb','Temp'],
                      mode = 2)

df1 = tm_gb.load_csv(filename = 'LDPRF_2098.csv',
#                       data_list = ['Program␣
↪time','Current','Voltage','AhCha','AhDch','Temp'],
                       features_list =␣
↪['runtime_s','Current','Voltage','AhCha','AhDch','Amb','Temp'],
                      mode = 2)
```

```
C:\Users\user\Anaconda3\lib\thermalModel_groupB.py:47: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/indexing.html#indexing-view-versus-copy
  df['second'][set_index[index]:set_index[index+1]] =
df['second'][set_index[index]:set_index[index+1]] + second_increment[index]
C:\Users\user\Anaconda3\lib\thermalModel_groupB.py:49: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/indexing.html#indexing-view-versus-copy
  df['second'][set_index[index]:] = df['second'][set_index[index]:] +
second_increment[index]
C:\Users\user\Anaconda3\lib\thermalModel_groupB.py:56: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/indexing.html#indexing-view-versus-copy
  df['second'][set_index[index]:] = df['second'][set_index[index]:] +
seconds_summation[index]
```

[4]:
```
df.describe()
```

[4]:

	runtime_s	Current	Voltage	AhCha	\
count	435839.000000	435839.000000	435839.000000	435839.000000	
mean	24468.740483	-0.595961	3.775370	126.363856	
std	11440.765250	85.854861	0.091213	72.924632	
min	0.000000	-177.639340	3.536830	0.000000	
25%	14564.850000	0.009580	3.730960	64.452000	
50%	24470.300000	0.009580	3.766810	126.039000	
75%	34375.750000	0.019150	3.807290	187.997000	

```
max     44280.800000     223.268950       4.160100     252.040000
```

```
                AhDch            Amb           Temp
count  435839.000000   4.358390e+05  435839.000000
mean      144.644944   2.579465e+01      34.312581
std        74.703530   2.402277e-10       2.060416
min         0.000000   2.579465e+01      25.794650
25%        81.299000   2.579465e+01      33.008410
50%       145.061000   2.579465e+01      35.085100
75%       208.479000   2.579465e+01      35.850190
max       272.253000   2.579465e+01      36.724590
```

Do type conversion:

[5]:
```
df_float32 = copy.deepcopy(df).astype('float32')
print(df.dtypes)
# first optimisation, as required by tf lite
print(df_float32.dtypes)
```

```
runtime_s    float64
Current      float64
Voltage      float64
AhCha        float64
AhDch        float64
Amb          float64
Temp         float64
dtype: object
runtime_s    float32
Current      float32
Voltage      float32
AhCha        float32
AhDch        float32
Amb          float32
Temp         float32
dtype: object
```

[6]:
```
print(type(df_float32))
df_float32 = df_float32.drop(columns=['runtime_s'])
```

```
<class 'pandas.core.frame.DataFrame'>
```

0.1.2 Target row and inputs/ outputs:

[7]:
```
test_pdSeries_row = copy.deepcopy(df_float32.iloc[100])
print(test_pdSeries_row)
```

```
Current   -46.004620
Voltage     3.784130
```

```
AhCha       0.000000
AhDch      12.521000
Amb        25.794649
Temp       27.215540
Name: 100, dtype: float32
```

[8]:
```
input_npArray = test_pdSeries_row[:-1].to_numpy()
input_npArray = input_npArray.reshape(5,1).T
output_npArray = test_pdSeries_row[-1]
print(type(input_npArray))
print(type(output_npArray))
print(input_npArray)
print(output_npArray)
```

```
<class 'numpy.ndarray'>
<class 'numpy.float32'>
[[-46.00462   3.78413   0.       12.521    25.79465]]
27.21554
```

[9]:
```
tf_model_32 = tf.keras.models.load_model('DNN_0.1_hybrid_model_1.h5')
h5_output = tf_model_32.predict(input_npArray)
print(h5_output)
```

```
[[26.764807]]
```

0.2 Convert to TensorFlow Lite

We now have an acceptably accurate model in-memory. However, to use this with TensorFlow Lite for Microcontrollers, we'll need to convert it into the correct format and download it as a file. To do this, we'll use the TensorFlow Lite Converter. The converter outputs a file in a special, space-efficient format for use on memory-constrained devices.

Since this model is going to be deployed on a microcontroller, we want it to be as tiny as possible! One technique for reducing the size of models is called quantization. It reduces the precision of the model's weights, which saves memory, often without much impact on accuracy. Quantized models also run faster, since the calculations required are simpler.

The TensorFlow Lite Converter can apply quantization while it converts the model. In the following cell, we'll convert the model twice: once with quantization, once without:

[10]:
```
# Convert the model to the TensorFlow Lite format with float16 quantization
converter = tf.lite.TFLiteConverter.from_keras_model(tf_model_32)
converter.optimizations = [tf.lite.Optimize.DEFAULT]
converter.target_spec.supported_types = [tf.float16]
model = converter.convert()
# Save the model to disk
open("model.tflite", "wb").write(model)
```

[10]: 2392

[11]:
```
# Instantiate an interpreter for each model
model = tf.lite.Interpreter('model.tflite')

# Allocate memory for each model
model.allocate_tensors()

# Get input and output tensors
tflite_model_float16_input_details = model.get_input_details()
tflite_model_float16_output_details = model.get_output_details()
```

[12]:
```
# Create arrays to store the results
tflite_model_float16_predictions = np.empty((1, 1))

# Test the TensorFlow Lite model
input_shape = tflite_model_float16_input_details[0]['shape'] # same for all
output_shape = tflite_model_float16_output_details[0]['shape'] # same for all

# preprocess:
input_data = input_npArray

# The function `get_tensor()` returns a copy of the tensor data.
# Use `tensor()` in order to get a pointer to the tensor.

model.set_tensor(tflite_model_float16_input_details[0]['index'], input_data)
model.invoke()
tflite_results = model.
 ↪get_tensor(tflite_model_float16_output_details[0]['index'])
print(tflite_results)
```

```
[[26.776821]]
```

0.3 Write to a C file

The final step in preparing our model for use with TensorFlow Lite for Microcontrollers is to convert it into a C source file. You can see an example of this format in hello_world/sine_model_data.cc.

To do so, we can use a command line utility named xxd. The following cell runs xxd on our quantized model and prints the output:

[13]:
```
import os
cwd = os.getcwd()
print(cwd)
```

```
C:\Users\user\Desktop\FYP final analysis\profiling\tests
```

[14]:
```
# # start linux shell
# run: xxd -i model.tflite > model.cc
```

[15]:
```
# # instead of these official instructions:
# !cd $cwd
# !bash
# !xxd -i model.tflite > model.cc
# !cat model.cc
```