Week #8

TensorFlow Lite (TFL) and TFL Micro. Hello World Example

EECE-4710 IoT and Machine Learning

Cristinel Ababei
Electrical and Computer Engr., Marquette University

1. Objective

Learn about TensorFlow Lite (TFL) and TFL Micro through several examples. We see how to convert a TF model into a TFLite model and how to use it via an Interpreter. Also, we see how to generate a TFL Micro model and save it into a .cc file. We develop a simple neural network and use it for regression y=sine(x).

2. Assignment

Example 1:

The first example we look at is inside the Jupyter Notebook below. Start Colab, then, click on File->Upload Notebook and select from your computer the said Notebook. Once you uploaded the Notebook, do first Edit->Clear All Outputs. Then, work through all its code cells and follow the assignment tasks:
cifar_10/CNN_Cifar_10_TFLite.ipynb

In this example, we again work with the CIFAR10 dataset (we have seen this in Week #4) and build a **CNN model** to recognize the 10 classes of CIFAR ('airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship' and 'truck'). So, **this is a classification problem**.
The example shows how to save the model:

```
model.save('cifar_10_model')
```

After that, the model is converted from Tensorflow to **TensorFlow Lite** directly or after reading it from a previously saved model:

```
converter = tf.lite.TFLiteConverter.from_keras_model(model)
tflite_model = converter.convert()
```

The example looks at post-training quantization; for example, it shows how to statically quantize only the weights from floating point to integer with 8-bit precision. Also, it shows how to test a **TFLite** model using an **Interpreter**. An Interpreter is an interface for running TensorFlow Lite models.

Furthermore, the example shows how to generate a TensorFlow Lite for Microcontrollers Model - **TFL Micro** Model. To convert the TensorFlow Lite quantized model into a C source file that can be loaded by TensorFlow Lite for Microcontrollers on MCUs, we need to use the **xxd** tool to convert the .tflite file into a .cc file.

```
MODEL_TFLITE = 'cifar10_quant_model.tflite'
MODEL_TFLITE_MICRO = 'cifar10_quant_model.cc'
!xxd -i {MODEL_TFLITE} > {MODEL_TFLITE_MICRO}
REPLACE_TEXT = MODEL_TFLITE.replace('/', '_').replace('.', '_')
!sed -i 's/'{REPLACE_TEXT}'/g_model/g' {MODEL_TFLITE_MICRO}
```

This example ends with saving on your computer the .cc file **cifar10_quant_model.cc**.

The assignment – in the context of this example - is to do it entirely, but, for a number of `epoch=100`. Describe clearly in your own words in one paragraph what this example does. Include your accuracy vs. epoch plot and report also the accuracy you get:

Train: 85% - 90%;

Validation: 68%-70%
Test: 66%-68%

Download onto your computer your own **cifar10_quant_model.cc** generated for your own trained model for 100 epochs.

Example 2:
This example is inside this Jupyter Notebook among the files you downloaded for this week:
TFLite-Micro-Hello-World/train_TFL_Micro_hello_world_model.ipynb

In this example, we develop and train a simple **neural network** to model data generated by a `sine` function. This trained model can then take a value `x`, and predict its numerical sine value `y=sine(x)`. So, **this is a regression problem**. The example demonstrates the process of training a 2.5 kB model using TensorFlow and converting it for use with TensorFlow Lite for Microcontrollers. The model created in this notebook is used in the `hello_world` example:
https://github.com/tensorflow/tflite-micro/tree/main/tensorflow/lite/micro/examples/hello_world
from TensorFlow Lite for MicroControllers:
https://www.tensorflow.org/lite/microcontrollers

The example looks at the training and evaluation loss (i.e., mean squared error, MSE); it also looks at mean absolute error (MAE), which is another way of measuring how far the network's predictions are from the actual numbers. The example shows how too small of a NN model can result in poor/unacceptable performance and that increasing the size of the NN model can improve enough its capacity to learn the full complexity of the sine wave function.

Then, the example converts the NN model using a TensorFlow Lite Converter into two TFLite models: one with quantization, one without. Quantized models take less memory to store and also run faster, since the calculations required are simpler. These models are then compared in terms of their predictions and loss on the test dataset. For that purpose, again an **Interpreter** is used:

```
# Initialize the TFLite interpreter
interpreter = tf.lite.Interpreter(model_content=tflite_model)
interpreter.allocate_tensors()
```

The sizes of the models are compared as well:

```
Model                      Size
TensorFlow                 4096 bytes
TensorFlow Lite            3192 bytes (reduced by 904 bytes)
TensorFlow Lite Quantized 2680 bytes (reduced by 512 bytes)
```

It is observed that the original TF model, the TFLite model, and the quantized TFLite model are all close enough to be indistinguishable - even though they differ in size. This implies that the quantized (smallest) model is good enough and ready to use!
Next, we convert the TensorFlow Lite quantized model into a C source file that can be loaded by TensorFlow Lite for Microcontrollers. That is done using the **`xxd`** command. The C file created is **model.cc**. Download this file to your computer. Open it with a text editor; you should see its contents look like this:

```
unsigned char g_model[] = {
  0x1c, 0x00, 0x00, 0x00, 0x54, 0x46, 0x4c, 0x33, 0x14, 0x00, 0x20, 0x00,
  0x1c, 0x00, 0x18, 0x00, 0x14, 0x00, 0x10, 0x00, 0x0c, 0x00, 0x00, 0x00,
…
  0x0c, 0x00, 0x10, 0x00, 0x0f, 0x00, 0x00, 0x00, 0x08, 0x00, 0x04, 0x00,
  0x0c, 0x00, 0x00, 0x00, 0x09, 0x00, 0x00, 0x00, 0x04, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x09
};
unsigned int g_model_len = 2680;
```

The assignment – in the context of this example - is to do modify the second NN model from this example and apply at least one technique to mitigate overfitting. Recall that we looked at such techniques in previous lectures. Then, go through all the remaining steps of this example to generate a new **model.cc** file at the end of it that should represent the better model, created using the overfitting mitigation technique. Download the new **model.cc** to your computer.

3. Deliverables

You must write (typed) a report and upload it as a PDF file on D2L. The report should be named "**LastName_hw8.pdf**". The report should include the following sections:

1) Title + course info + your name
2) **Summary.** Describe in one paragraph what the objective of the assignment is.
3) **From TF to TFL to TFL Micro.** Describe briefly your experiments from Example 1 and the results you got. Include plots and accuracy values.
4) **Overfitting Technique for NN Model.** Describe briefly what technique you used to prevent overfitting. List the relevant Python code in your report, nicely formatted! Include also the plot titled "Comparison of various models against actual values" created using your new model.
5) **Conclusion.** Present your conclusions and describe what issues you encountered and how you solved them.
6) **References.** Include all references that you used, as a numbered list. Cite them in the report itself; do not just list them here.

You should also upload on D2L your modified notebooks and the two .cc files that you created at the end of each notebook. Include all these files and the report into a .zip named "**LastName_hw8.zip**".