Week #11

Wake-Word Detection – Model and Application Development and Testing

EECE-4710 IoT and Machine Learning

Cristinel Ababei
Electrical and Computer Engr., Marquette University

1. Objective

To go through the steps of **model and application development & testing** presented in Chapter 7 of the Textbook. The main steps in working on a project are presented again below. Usually, one would go through all these steps:

- **Step #1:** Model development, training, optimization (pruning, quantization), TFL-Micro model generation; Python; Done in Google Colab or locally on your computer.
- **Step #2:** Application development: Testing and Application Run on development machine; C/C++; Done on macOS (this is what textbook authors use) or Linux machine.
- **Step #3:** Application deployment to Microcontroller; C/C++; Arduino IDE; Done on local computer and executed on Arduino Nano 33 BLE Sense in our case.

2. Notes About TFL-Micro GitHub Changes

The github repository with the examples from the textbook used to be part of the main tensorflow repository as:
tensorflow/tensorflow/lite/*
All the examples in the printed textbook were from that initial/old repository. The links to code (of this type: https://oreil.ly/YiSbu) from the textbook are **deprecated** as the examples from the textbook have been relocated in github.

In June 2021, a dedicated new github repository was created to host "TensorFlow Lite (TFL)" whose port "TensorFlow Lite for Microcontrollers (TFLM)" is designed to run machine learning models on DSPs, microcontrollers and other devices with limited memory. The new repository is continuously updated and changes/updates are checked in all the time. The old repo is not updated anymore, but, its source files are reflected in the textbook, which has not been updated either.

For example, regarding the "hello-world" example, the old hello_world_test.cc that is discussed in detail in Chapter 5 of the textbook is now deprecated (last version as of 2019), i.e., it is not the latest version, even though generally the test does the same thing.
The old version of this file **hello_world_test.cc** for instance is this:
tensorflow/tensorflow/lite/experimental/micro/examples/hello_world/hello_world_test.cc
located in the old location:
https://github.com/tensorflow/tensorflow/blob/be4f6874533d78f662d9777b66abe3cdde98f901/tensorflow/lite/experimental/micro/examples/hello_world/hello_world_test.cc
(which is the link from the textbook itself from page 68 pointed to)
The new version of the same test example existed for a while with some small differences, but, it does not exist in exactly the same shape and form anymore – as things have been updated. The way testing is done may have been changed. The new location of the hello-world example is:

https://github.com/tensorflow/tflite-micro/tree/main/tensorflow/lite/micro/examples/hello_world

Regarding the "micro-speech" example:

The new version micro-speech example is located at:

https://github.com/tensorflow/tflite-micro/tree/main/tensorflow/lite/micro/examples/micro_speech <---- NEW location; this has updated examples, for the most part very similar to the initial ones; we will use this version, though one can use the initial OLD ones that match exactly the TextBook too.

The old version micro-speech example (as described in the TextBook) is located at:

https://github.com/tensorflow/tensorflow/tree/be4f6874533d78f662d9777b66abe3cdde98f901/tensorflow/lite/experimental/micro/examples/micro_speech <---- OLD deprecated location; this was the initial location where the code is exactly as in the TextBook!

As a final note, it's always a good idea to keep an eye on the website of "TensorFlow Lite for Microcontrollers": https://www.tensorflow.org/lite/microcontrollers

and keep up with any new changes and developments!

3. Step #1: Building and Training the Model - In Google Colab

Prerequisites:

1. **Read Chapters 7&8 from the textbook!**
2. Also, watch the youtube video of Pete Warden discussing Chapter 4, which presents basically very similar steps for the "hello_world" example: https://www.youtube.com/watch?v=8N6-WQsxwxA
3. **Also, read the "Micro Speech Training" at:** https://github.com/tensorflow/tflite-micro/tree/main/tensorflow/lite/micro/examples/micro_speech/train

While in theory you can do this step locally on your computer using the Anaconda framework, let's do it using Google Colab. Doing this Notebook on a local Windows laptop may be tricky because of some of the hard-coded paths to various .py or example files from the tensorflow installation, which may not exist in the particular tensorflow installation. This is an example where working in Google Colab has an advantage. So, log in into your google account, and start Colab by going to: https://colab.research.google.com/

We must bring in Colab the Jupyter Notebook for this example, created by the textbook authors and made publicly available at:

https://github.com/tensorflow/tflite-micro/tree/main/tensorflow/lite/micro/examples

In Colab, click on File->Upload Notebook.

A new window pops up; then, click on the GitHub tab of that window. Paste into it the URL of the Jupyter Notebook of our hello world example (**use this one, as it should be up-to-date!**):

https://github.com/tensorflow/tflite-micro/blob/main/tensorflow/lite/micro/examples/micro_speech/train/train_micro_speech_model.ipynb

NOTE: This notebook is also be included in the .zip file for this week as: **train_micro_speech_model.ipynb (), but, it may not be the latest one (but, it should still work).**

Read the Jupyter Notebook carefully as you go through all the cells. Observe everything that happens as you run cell by cell. Click on Run (located inside "[ ]" at the top-left of each cell) for each cell to have the cell's Python code executed.

As you execute code boxes, please make sure you first make the following change. Replace:

```
WANTED_WORDS = "yes,no"
```

With:

```
WANTED_WORDS = "go,stop"
```

That is because we want to re-train the model to be able to classify the following four categories:

1. Go
2. Stop
3. Unknown
4. Silence

At some point, you may get an error like this:

```
AttributeError: module 'tensorflow' has no attribute 'Session'
```

When trying to run a code box that uses TF1 code:

```
with tf.Session() as sess:
```

One way to fix that is to use before that:

```
import tensorflow.compat.v1 as tf
tf.disable_v2_behavior()
```

or, replace:

```
tf.Session()
```

with:

```
tf.compat.v1.Session()
```

anywhere in the notebook.

The last code cells in the Notebook creates a "model for microcontrollers" that we save on our local computer as a text file, say called:

Chapter7_g_model_STOP_GO.txt

This "model for microcontrollers" is done with the following code:

```
# Install xxd if it is not available
!apt-get update && apt-get -qq install xxd
# Convert to a C source file
!xxd -i {MODEL_TFLITE} > {MODEL_TFLITE_MICRO}
# Update variable names
REPLACE_TEXT = MODEL_TFLITE.replace('/', '_').replace('.', '_')
!sed -i 's/'{REPLACE_TEXT}'/g_model/g' {MODEL_TFLITE_MICRO}
```

And then print it like this:

```
# Print the C source file
!cat {MODEL_TFLITE_MICRO}
```

Which prints the following model with hexadecimal values:

```
unsigned char g_model[] = {
  0x20, 0x00, 0x00, 0x00, 0x54, 0x46, 0x4c, 0x33, 0x00, 0x00, 0x00, 0x00,
  0x14, 0x00, 0x20, 0x00, 0x1c, 0x00, 0x18, 0x00, 0x14, 0x00, 0x10, 0x00,
  0x0c, 0x00, 0x00, 0x00, 0x08, 0x00, 0x04, 0x00, 0x14, 0x00, 0x00, 0x00,

  …
  0x03, 0x00, 0x00, 0x00, 0x03, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03,
  0x0c, 0x00, 0x0c, 0x00, 0x0b, 0x00, 0x00, 0x00, 0x00, 0x00, 0x04, 0x00,
  0x0c, 0x00, 0x00, 0x00, 0x16, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x16
};
unsigned int g_model_len = 18960;
```

Take the above with Copy and Paste and save it into your own file, locally on your computer:

Chapter7_g_model_STOP_GO.txt

We will use that later to update the Arduino program.

4. Install VirtualBox and Linux – Needed for Development/Testing (Optional)

Before deploying and running a program on a microcontroller board, we will confirm that the program works on a Development machine, that is my laptop. At the time of writing this, I do not have an environment on Windows that could make that possible headache free. This would be easy if you had a Mac laptop – which is the environment that that examples from the Textbook were developed in.

To get around that, I installed VirtualBox (VB) on my Windows laptop, and then, installed Linux Ubuntu on that. I will do all programs development inside the Linux machine. It is more painful this way – but, it is the price I must pay for sticking still with Windows laptops… ☹ I could have used for this step my Linux workstation, but, that is hard to carry around compared to my laptop 😊.

So, next, we will install VirtualBox and then Linux Ubuntu on it. If you already have a Linux machine or you have already installed VirtualBox, you do not need to do this.

The term virtualization means that you can have another OS over an existing OS. For instance, you can run Windows on a Mac or you may install Linux on a Windows 10 machine using virtualization software.

Installation Step:
Create an Ubuntu VM following the steps described in this tutorial:
(please download and install a latest stable edition, LTS, of Ubuntu!)
https://henricasanova.github.io/files/vbox/VirtualBoxUbuntuHowTo.html
In my case, I installed VB in:
M:\VirtualBox

At this time, you should have Ubuntu installed inside the VirtualBox.
In my case I installed Ubuntu in:
M:\VirtualBox_VMs

NOTE: On Installing "Guest Additions" – Before doing the actions for that, first open a Terminal and install a couple of things like this:

```
> su root
> nano /etc/sudoers
```

And uncomment the line:

```
#sudo ALL=(ALL)  ALL
```

And also add this new line (change "cristinel" to your own user name you created for your Linux user account):

```
cristinel ALL=(ALL)  ALL
```

Then save the change by doing CTRL-X and then Yes.

```
> sudo apt update
> sudo apt install -y build-essential
```

You should now see on the left side of your Ubuntu screen the Disc icon and if you hover above, it should say "VBox_GAs_7.0.2"
Click on it.
Place mouse inside the new window and right-click, select Open in Terminal.
A new Terminal should open. Do in terminal to see all files:

```
>ls
>./autorun.sh
```

A new window will pop-up. Type your password in.
Then wait for the VB Guest Additions to install.
Shut-off (not restart, though you could try that first) the Ubuntu machine.
Start the machine again and login.
That will finish the installation of Guest Additions.
After that select View->Auto-Resize Guest Display
At this time you should be able to maximize the window of your Linux Ubuntu!

At this time, go on with the "Creating a Shared Folder" action.

I created mine as:

M:\VirtualBox_SharedFolder

Once your VM instance has restarted and you're logged in, open a Terminal and do:

```
>cd /media
>ls
```

And you should see the shared folder sf_ VirtualBox_SharedFolder (with an "sf_" in front)

One last issue to do: add your username to the "group" called vboxsf. That's is to avoid to do sudo all the time when dealing with the sf_ folder. In a Terminal do:

```
>sudo usermod -a -G vboxsf cristinel
```

(replace "Cristinel" with your user name)

Shutdown/restart your VM instance one last time, and you are set. From now on, you can always use the /media/sf_VirtualBox_SharedFolder directory to allow files to exist both on your own machine and within your Ubuntu VM.

5. Step #2: Application Development and Testing – on Linux inside VB (Optional)

Before anything, **read again Chapters 7&8 from the textbook!**

5.1. Just Study the Application and Testing Code – On Windows machine (no Linux or Mac needed)

Unfortunately, the application examples from this textbook were not developed on Windows or with Windows support. So, I have not managed yet to compile them successfully on a Windows 10 laptop. However, you could still download the github repository to study the source code on a pure Windows machine. To grab the code from github on Windows, you could use git:

Start->Git-> Git Bash

Then navigate to a folder where you would like to bring the github repository; in my case I did it in:

M:\MARQUETTE\EECE4710_IoT_and_ML\BOOK_TinyML

Then in the git terminal type:

>git clone https://github.com/tensorflow/tflite-micro.git

It takes a few good minutes to download.

Then, get inside it:

>cd tflite-micro

And investigate using a text editor files of interest. For example, the makefile that can be used for the hello-world test (refer to Chapter 5 from textbook) is located in:

tensorflow/lite/micro/tools/make/Makefile

And, it could be used like this:
>make -f tensorflow/lite/micro/tools/make/Makefile test_hello_world_test **<---- but, it does not work on Windows currently**

5.2. Running Application and Tests – On Development Machine: Linux inside VirtualBox, Installed on Windows 10 Laptop

In this task because there is no support in Windows for the examples from the textbook, what we'll do instead is we will work this application on Linux Ubuntu inside VirtualBox, which you should have already installed. If you are a Mac user, then, you should be all set and work directly on your Mac laptop, no need for VirtualBox and Linux Ubuntu.

So, start VB, and then launch the Ubuntu Linux machine. Open a Terminal.

First, before anything, let's install a couple of utilities that we'll need (these need to be done only once of course):

> sudo apt update

> sudo apt -y upgrade

> sudo apt-get install build-essential

> sudo apt-get install git

> sudo apt-get install curl

> sudo apt install aptitude

> sudo aptitude install g++

> sudo apt install -y python3-pip

> sudo apt install -y libssl-dev libffi-dev python3-dev

> sudo apt install emacs

With a text editor (e.g., emacs just installed) open and edit ~/.bashrc file. Place this new line in this file:

alias python=python3

> source .bashrc

> sudo apt install python-is-python3

> sudo apt install python3-numpy

> sudo apt install bazel-bootstrap

Installing Bazel on Ubuntu (https://bazel.build/install/ubuntu):

> sudo apt install apt-transport-https curl gnupg -y

> curl -fsSL https://bazel.build/bazel-release.pub.gpg | gpg --dearmor >bazel-archive-keyring.gpg

> sudo mv bazel-archive-keyring.gpg /usr/share/keyrings

> echo "deb [arch=amd64 signed-by=/usr/share/keyrings/bazel-archive-keyring.gpg] https://storage.googleapis.com/bazel-apt stable jdk1.8" | sudo tee /etc/apt/sources.list.d/bazel.list

> sudo apt update && sudo apt install bazel-7.0.0

> sudo apt-get install python3-matplotlib

Second, let's study and then compile applications from the textbook, which we create a special folder for:

> mkdir tinyml_book

> cd tinyml_book

> git clone https://github.com/tensorflow/tflite-micro.git

> ls

You should see a new folder tflite-micro, get inside it:

> cd tflite-micro

Great! Now, let's do the actual examples discussed in the textbook.

Example 1: "hello_world"

Open in a text editor the readme file of the hello_world example:

> emacs tensorflow/lite/micro/examples/hello_world/README.md

To run the tests for this example follow the steps indicated in there.

For example, to evaluate the models of this example, we need to do:

> bazel build tensorflow/lite/micro/examples/hello_world:evaluate

> bazel run tensorflow/lite/micro/examples/hello_world:evaluate

> bazel run tensorflow/lite/micro/examples/hello_world:evaluate -- --use_tflite

If everything is ok, then, you should see the real and predicted sine-waves created by matplotlib, as shown in Figure 1.

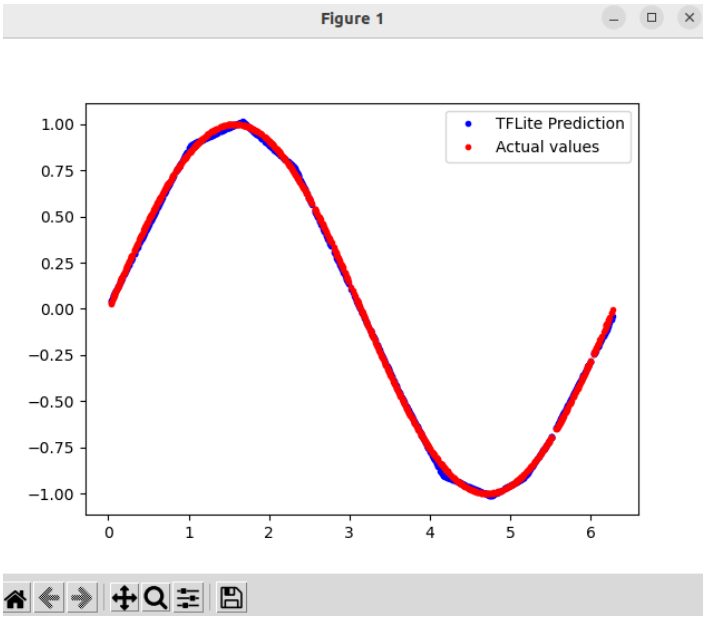

Figure 1: Running of evaluate portion of the hello_world example on development machine (Linux Ubuntu inside VirtualBox).

Now, do all the remaining steps described in the above README.md.

Example 2: "micro_speech"

Open in a text editor the readme file of the micro_speech example:

> emacs tensorflow/lite/micro/examples/micro_speech/README.md

Read thorough it and do all the steps described therein.

We will discuss and do in-class these steps as well.

If all Ok, then, you should be able to build and run all the tests.

The main takeaways from this exercise are:

- In developing applications like those from the textbook, one should create tests for testing various portions of the application.
- These tests and in fact these whole applications could be used as "templates" that one can modify to create new applications without the need to do everything from scratch.

6. Step #3: Application Deployment – on Arduino Nano 33 BLE Sense

Here, we use the TFL-Micro model saved as **Chapter7_model_MCU.txt** to update the example "micro_speech" in Arduino, which we recompile, and test again on the Arduino Nano 33 BLE Sense.

NOTE: Last week, we already went through a code-walk-through process of this "micro-speech" example in Arduino.

To replace the default model (which recognizes "yes" and "no") in the "micro_speech" example, located here:

C:\Users\Cristinel Ababei\Documents\Arduino\libraries\Harvard_TinyMLx\examples\micro_speech

We need to edit:

micro_features_model.cpp

and replace the hexadecimal values of:

```
const unsigned char g_model[] DATA_ALIGN_ATTRIBUTE = {…}
```

with those from your **Chapter7_g_model_STOP_GO.txt** that you created earlier.

Then, re-compile the Arduino sketch and test.