Key Word Spotting – Model Development and Arduino Application Development and Testing

EECE-4710 IoT and Machine Learning

Cristinel Ababei
Electrical and Computer Engr., Marquette University

1. Objective

To develop a tinyML model to spot keywords "marquette" and "engineering". Then, to develop an Arduino application that will continuously listen and infer with the help of the developed model when either one of the two words is spotted. React to that by turning the RGB LED Green or Blue.

2. Basics

This is an example that is similar to the "micro speech" example from the textbook, described in Chapters 7 and 8. Normally, we would follow the typical workflow with these steps:

- Step #1: Model development, training, optimization (pruning, quantization), TFL-Micro model generation.
- Step #2: Application development: Testing and Application Run on development machine or cloud.
- Step #3: Application deployment to Microcontroller board, Arduino Nano 33 BLE Sense.

However, in this tutorial, we will focus on Steps 1 and 3 only.

Before going into the model development, we first revisit spectrograms. For that, load up into Google Colab the notebook (provided in the .zip file for this week):

audio_raw_data_analysis/audio_raw_data_analysis.ipynb

and follow the instructions and steps therein!

This exercise requires you to record two speech signals for "marquette" and "engineering" and to create the corresponding spectrograms.

3. Putting Together a Dataset – On Your Computer

Before developing a model, we need a dataset.

In our case, we want to create a KWS project for three classes, which are three key words:

1. **engineering**
2. **marquette**
3. **silence** (with some background noise, such as a faucet running)

For collecting sound data, we must use a recording device, which can be basically anything with a microphone, including smartphone, webcam, laptop, Arduino Nano 33 BLE Sense, etc. The Arduino application at the end will use the Arduino Nano 33 BLE Sense. You are free to use any approach for this assignment, including any of the following options:

Option 1: I used the *Voice Recorder app* on my Samsung Galaxy smartphone to record 50 spoken "marquette" signals and 50 "engineering" signals, each for 1000 ms long. Make sure you configure settings of the app to save in **.wav** format and do sampling at **16KHz**. Do your recordings with some background noise, from different distances from the microphone, with different volume of your voice, etc. to create a more robust dataset.

Option 2: As a better alternative to the smartphone app as a recording means, you could use on your laptop any popular and free application, such as *Audacity* (can be used for editing of the recordings done with the smartphone app). Audacity is an easy-to-use, multi-track audio editor and recorder for Windows, macOS, GNU/Linux and other operating systems. You can download it from here:

https://www.audacityteam.org/download/

For "silence" class, let's use a pre-existing available dataset that includes a generic noise category and the sound of a faucet running. We will use that data, and augment it with our recorded sounds.

So, to begin with, download the faucet dataset from this link:

https://github.com/ShawnHymel/ei-faucet-dataset/raw/master/faucet_dataset_v01.zip

Unzip it somewhere on your computer.

Then, create your own dataset into a new folder, which let's call "dataset_v1" like this:

Copy 50 files from faucet_dataset/faucet into our dataset as:

dataset1/silence/silenceXX.wav (with XX = 00, 01, ..., 49)

Place your own 50 "marquette" recordings into:

dataset1/marquette/marquetteXX.wav (with XX = 00, 01, ..., 49)

Place your own 50 "engineering" recordings into:

Dataset1/engineering/engineeringXX.wav (with XX = 00, 01, ..., 49)

Note that Audacity can be used to curate your own audio files for "marquette" and "engineering" if needed to do so. For example, you can trim signals or export them into 16-bit PCM, etc. **You should use it to trip all your recordings to exactly 1 second long each.**

4. Build and Train Model & Download Arduino Application - With EdgeImpulse (EI)

Follow the procedure learned in previous lectures to create an EI project, which will implement a classifier to detect the detection of any of the three classes "marquette", "engineering", and "silence".

To train the classifier we will use the dataset created in the previous section, which you will need to upload to your EI project.

The main steps of doing this are illustrated in the PPT slides:
W12_1_KeyWord_Spotting_Appl_Devel_EI.pptx

At the end of this exercise, you must download from EI the Arduino library and install it inside your Arduino IDE. Then, launch and test the application on your Arduino Nano 33 BLE Sense board, as illustrated in the above slides.

5. "Under the Hood" of Build and Train Model - In Google Colab

Start Google Colab and upload the provided Jupyter Notebook:

keyword_spotting_example/keyword_spotting_project_nn_classifier.ipynb

Read the Jupyter Notebook carefully as you go through all the code cells. Observe everything that happens as you run cell by cell.

The last code cells in the Notebook creates a "model for microcontrollers" that is saved as: **model.cc**

Download it to your local computer. Use it to replace the model information in the Arduino application, which you then should recompile and test again.