IESTI01 - Lab 3

Project: KWS / Sound Classification

Introduction

In the class, we went through the steps of creating a speech recognition (or keyword spotting) system (in the case, using “UNIFEI and “IESTI”). The same basic steps can be used to create a device that recognizes and classifies other sounds, as broken glass, toss, etc.

In this lab, you should at least create a KWS project, using 3 classes:

- UNIFEI
- Silence (Try with some background noise)
- Second word of your choice.

Optionally you can try **a second and separated project**, starting with a simple, pre-made dataset, adding your own target sound, and build a classifier for those sounds. Try find sounds made with something other than your voice!

Required Hardware

For collecting sound data, you should have access to a recording device. This can be a the Arduino - 33 Sense (TinyML Kit), and/or smartphone, webcam, laptop, etc.

For deploying, you must use the Arduino Nano 33 BLE Sense.

Collect Data for optional Sound Project (not voice)

You can test a pre-made dataset from Edge Impulse that includes a generic noise category and the sound of a faucet running. You can use just that data, augment it with your own sounds, or collect your own, entirely new dataset.

To start, download the faucet dataset from this link. Unzip it somewhere on your computer.

If you are going to use your own sounds, collect at least 50 1-second audio samples of that sound class (or at least one 50 second recording). Make sure the sound source is in different environments and different distances away from the recording device. This will help create a more robust model that can differentiate that sound from other noises.

For example, recorded a fan, a blender or other machines running at different speeds. Move the microphone to several different angles to collect the sound in front of, behind, and above the machine, as the sound changes as the angles and distance varied.

The idea is to have the embedded system identifying among at least 3 different types of sounds: background noise, faucet running, or machine working (select only one: fan, blender, car engine, etc).

Curate Data

When you're done, transfer the sounds over to your computer. Remember that if you collect directly with your phone, you should have an app that sample sound using 16KHz/16bits. If not, you can use: Audacity. Open your audio file in Audacity. Select the drop-down menu, *Project Rate (Hz)* at the bottom of the screen. Change the sampling rate to **16000 Hz**. And click **File > Export > Export Selected Audio**. Set the bit depth to **Signed 16 bit PCM**. Save the audio selection as a WAV file. You can use Audacity to split the sound in 1 second clip, but using the "Split Feature" available at Edge Impulse Studio is easier (in my opinion).

Use the naming convention of <*label*>*.xxxx.wav*. The <label> before the first period (.) can be used by Edge Impulse to automatically determine the class. A number (xxxx) is used to uniquely identify the files. The ordering and exact number is not important (i.e. you could also use a hash).

NOTE: Of course, you can also collect audio data straight from your Edge Impulse project as we did in class! Go to *Data acquisition* in a new project and connect your smartphone or Arduino board.

Upload Data (when you have .wav data collected)

- Create a new project in Edge Impulse.
- Head to the **Data Acquisition** page.
- Click **Let's collect some data**.
- Select the **Go to the uploader** option.

On this new page, click **Choose files**. Select all of the files from your curated sample set.

- Click **Open**.

Leave *Automatically split between training and testing* selected. If you used the file naming scheme I outlined above, leave *Infer from filename* selected. If not, select *Enter label* and give your samples a label (e.g. “fan”).

- Click **Begin upload**.

Repeat this process for the *faucet* and *noise* sets that you downloaded in the *faucet_dataset* ZIP file you downloaded at the beginning. The WAV files in that set should follow the naming scheme outlined, letting you leave *Infer from filename* selected in order to add labels to the samples.

- Click on the **Data acquisition** link to go back to the Data Acquisition page. Here, make sure that all of your samples are present and that they are divided between the training and test sets (there should be about 20% of the samples in the test set).

Feature Extraction

Navigate to the **Impulse design** page of your project.

- Add an **Audio (MFCC)** processing block for KWS (human words) or **Audio (MFE)** (for non-voice audio)
- Add a **Neural Network (Keras)** learning block.
- Click **Save Impulse.**

Go to the **Spectrogram** page

- Click on the **Generate Features** tab.
- Click the **Generate Features** button, and wait a moment while your audio samples are converted into spectrograms. When it’s done, take a look at the *Feature explorer* to see if you can identify separation among your classes.

Model Training

Navigate to the **NN Classifier** page. Leave all of the hyperparameters at their defaults and click **Start training**. When it's done, scroll down to view the *Confusion matrix* of the validation data.

- Download the Jupyter Notebook (Expert mode) and run it at CoLab.
- See the history graphs and try changing some of the hyperparameters and re-training your model to see if it improves the per-class accuracy.
- Return to Edge Impulse Studio and re-train your model with eventual diferente hyperparameter.

Testing

Head to the **Model testing** page and classify all of the test data. Eventually you can try Live Test using the Kit. If you're happy with the test results, continue to the deployment step (I hope to see an accuracy better that 75%, if not, go back to collect more data and adjust hyperparameters).

Deployment

Head to the **Deployment** page in your Edge Impulse project.

- Select the **Arduino library** and click **Build**.
- When it's done downloading, open your Arduino IDE.
- Select **Sketch > Include Library > Add .ZIP Library…** Select the library file you just downloaded.
- Go to **File > Examples > <your-project-name> > nano_ble33_sense_microphone**
- Click **Upload** to compile and send the program to your Arduino board.

When it's done, open a Serial Monitor. Try holding the Kit (Arduino board) up to different noise sources.

Arduino LEDs

Modify the Arduino sketch **nano_ble33_sense_microphone,** in order to include the RGB LEDs (do it on a way that you can use the kit "stand alone" w/o looking at IDE Serial Monitor to see what class your project is selecting.

Report

For grade, delivery a report with relevant photos/print screens and the CoLab with your project.