Machine Learning for IoT
Homework I

Lorenzo Melchionna
Politecnico di Torino
Data Science and Engineering
s304651@studenti.polito.it

Edoardo Marchetti
Politecnico di Torino
Data Science and Engineering
s303873@studenti.polito.it

Edgar Gaytán
Politecnico di Torino
Data Science and Engineering
s300164@studenti.polito.it

***Abstract*—In this report, we describe how we have designed a voice activity detection system and a battery monitoring system. They were completed by combining the use of the Deepnote platform and local testing on the device.**

1. Voice Activity Detection Optimization & Deployment

This section describes the pipeline used for preprocessing and classification of audio files. Next, the process by which the hyper-parameters were selected to optimize the accuracy and latency of the voice-detection system is reported. During the hyperparameters selection, we did not consider changing the sampling rate (16 kHz) since down-sampling requires time and increases the latency.

1.1. VAD Pipeline

The pipeline used by the system is composed of the functions following indicated:

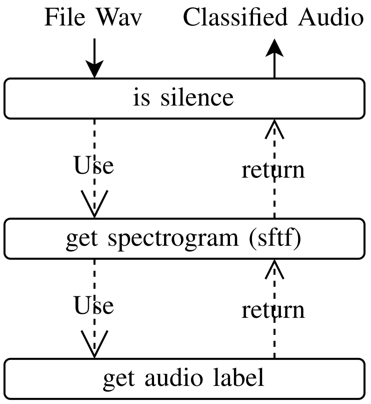

Figure 1. Voice Activity Detection pipeline

- **get_audio_label**: reads the audio file, applies padding if necessary and returns the audio as a tensor, the sample rate and the true audio label. In the local version, it has been modified to work with numpy arrays. The duration of the audio is set to 1s and the amplitude is scaled to [-1.0,1.0].
- **get_spectrogram**: its main goal is to take the contents of the file from the amplitude domain to the frequency domain. To achieve this the *Short-Time Fourier Transform* (*STFT*) is used, it divides the audio into frames with the same length (according to the received parameter *frame_length_in_s*) and applies a DFT to each frame increasing the time resolution of the spectrogram.
- **is_silence**: it is used to classify the audio based on its spectrogram. First, computes the energy of each frame, then counts the number of frames whose energy is greater than a given threshold (*dbFSthreshold*) and finally check if the duration of that frames is greater than a second threshold (*duration_threshold*). If the latter condition is satisfied the function classifies the audio as noisy, returning 0, otherwise it returns 1.

1.2. Hyperparameter selection

In order to optimize the VAD system, we had to find the best combination of the parameters in the table. To this end, we first applied a grid search, noting that frame length had an impact on latency, while thresholds had an impact on accuracy.

TABLE 1. VALUES USED FOR THE GRID SEARCH OF THE HYPERPARAMETERS

Param	Values
frame_length_in_s	[0.010,0.020,0.030,0.040,0.050] [0.008,0.016,0.032]
duration_thres	[0.10,0.20,0.30,0.40,0.50]
dbFSthreh	[-70,-80,-90,-100,-110,-120,-130]

Since Deepnote did not ensure the same result in terms of latency at each execution, we started analyzing the best value for the parameter step by step. First, we calculated the average energy of each frame by comparing noisy and silent audio. As shown in Figure 2 -120 seems to be a good threshold.

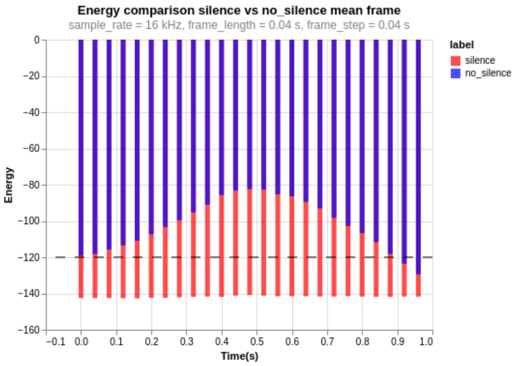

Figure 2. the blue bars represent the average energy in a frame for noisy audio, while the red ones the average energy for the silent audio.

Then we have analyzed the effect of the frame length on the latency and the *duration_threshold* on the accuracy.

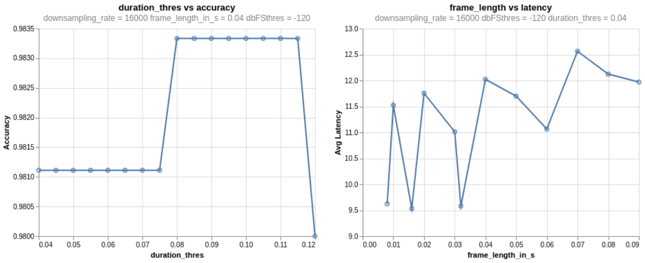

Figure 3. The left plot shows how the accuracy varies with respect to the duration_threshold parameter, while the right one represents the variation of the latency with respect to the frame_length_in_s

Figure 3 shows how increasing the duration threshold also the accuracy increases up to 0.12s. Instead the optimal frame length seems to be a power of 2. Indeed as indicated in [1], when the frame length is a power of 2 the STFT is computed by exploiting the radix-2 FFT [2] speeding up the computation. In order to choose the best combination of *frame_length_in_ms* and *duration_threshold* we plotted figure 4.

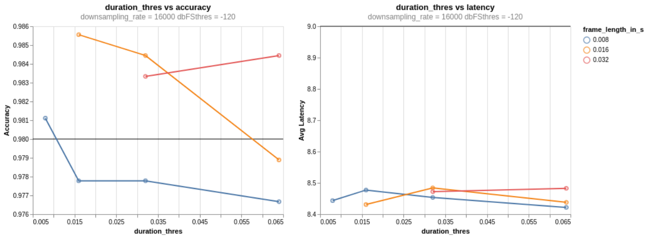

Figure 4. The graph shows how the accuracy and the latency vary with respect to the value of the *duration_thres* and the *frame_length_in_s* parameters. It can be observed that when the two parameters are equal the latency is lower. For this reason we set both the frame and the threshold duration equal to *0.016 seconds*.

Based on the results described above the final set of selected parameters are:

downsampling_rate	frame_length_in_s	dbFSthreshold	duration_threshold
16000	0.016	-120	0.016

2. Memory-constrained Timeseries Processing

In this second part of the task, we had to design a memory-constrained battery monitoring system that stores on a Redis database the battery charge level of the device on which it is being run and whether power is plugged in every second. In addition, every 24 hours the system must keep track of how many seconds the power has been plugged in.

To do this, three timeseries were created:

- *mac_address:battery_memory*: stores the battery level of the device, each value is within (0,100] interval;
- *mac_address:power*: stores 1 if the power was plugged otherwise 0;
- *mac_address:plugged_seconds*: stores how many seconds the power have been plugged in the last 24h.

2.1. Retention times calculation

In order to calculate the maximum retention time to meet the given constraints, first of all, the number of records that can be saved in one MB was calculated considering the average compression ratio $(\approx 90\%)$:

$$\frac{1024^2}{16*0.1}\ \text{bytes} = 655360\ \text{records} \tag{1}$$

In the denominator we round the compressed record size since the number of bytes occupied by a (*timestamp*, *value*) pair must be an integer. In the first two timeseries the number of records within a single MB corresponds also to the number of seconds that we can store in 1 MB since the system registers one value each second. For this reason, we set the **retention time** of *mac_address:battery_memory* and *mac_address:power* equal to $655360 * 5 * 1e3$ ms, which corresponds to about 38 days. e If we consider the *mac_address:plugged_seconds* timeseries, the number of records corresponds to the number of days, so we have to multiply it by the number of seconds in a day and then by one thousand to get the ms. The selected value is $5,6 * 1e13$ ms.

References

[1] Brian McFee *Digital Signals Theory* page: Defining the STFT

[2] Brian McFee *Digital Signals Theory* page: Radix-2 Cooley-Tukey