Team 18 - Homework 1

Andrea Parolin
Politecnico di Torino
Torino, Italia
s291462@studenti.polito.it

MohammadSadegh Radmehr
Politecnico di Torino
Torino, Italia
s301623@studenti.polito.it

Sara Ferrua
Politecnico di Torino
Torino, Italia
s292946@studenti.polito.it

I. PART1

A. Describe the methodology adopted to discover the VAD hyper-parameters compliant with the constraints.

To find the best hyper-parameters for our model, it is created a method that, given a possible combination of parameters as input, returns average processing time and accuracy for each configuration. The parameter search is optimized to have an average time of less than 9 ms and an accuracy greater than 98%.
In order to achieve these results, especially the imposed time limits, it is decided to optimize the search time of the hyper-parameters by not downsampling the audio files at all. In particular, if the *sampling_rate* of the audios of the dataset is 16 kHz by using the same value within the *downsampling_rate* the function *tfio.audio.resample* is never run and allows to mitigate the impact of getting the spectrogram. It is exploited the ParameterGrid method within the sklearn library, by creating all possible combinations of parameters as in Table I.

TABLE I
HYPERPARAMETER GRID

Hyperparameter	Values
downsampling _rate	16000
frame_length _in_s	10e-3, 20e-3, 30e-3, 8e-3, 16e-3, 32e-3
downsampling _rate	-120, -90, -130, -110
frame_length _in_s	0.05, 0.025, 0.01, 0.001

B. Add a table that reports the selected values of the VAD hyper-parameters (downsampling_rate, frame_length_in_s, dbFSthres, duration_thres).

The chosen hyper-parameters and the results obtained are shown in the first line of table II

TABLE II
GRIDSEARCH RESULT

down sampling _rate	frame _length _in_s	dbFSthres	duration _thres	Average Time	Accuracy
16000	0,016	-120	0,025	8,60ms	0,98

C. Comment the table and explain which hyper-parameters affect accuracy and latency, respectively. Motivate your answer.

- **downsampling_rate** = Downsampling values such as 8000 and 4000 are selected, but in these cases, the algorithm takes too long to exit the cycle *get_spectogram*, overcoming time constraints. According to this, the only solution adopted is to impose the *downsampling_rate* to 16000.
- **frame_length_in_s** = To get the spectrogram of each file audio the method *tf.signal.stft* is called. Short-time Fourier transform *stft* is fft applied many times and to optimize the runtime speed, the values to be assigned must be multiples of 2 because it reduces the complexity to $O(n \log n)$ instead of $O(n^2)$.
- **dbFSthres** = It represents a value that serves to discriminate single frames according to sound and silence. It is useful to determine which of the two cases (silence / not silence) has a stronger presence.
- **duration_thres** = It represents a threshold in seconds that must be exceeded in order to catalog the entire audio as silence rather than not silence. This is computed by summing the frames that exceed the *dbFSthres* threshold.

The accuracy is influenced by the last two hyperparameters: *duration_thres*, *dbFSthres* spectrogram to decibel in a log scale. The hyperparameters that influence the average computation time are the *downsampling_rate* and the *frame_length_in_s*.

II. PART2

A. Explain how you created and set up the mac_address: plugged_seconds timeseries.

The *try-except* condition is used everytime a time series is created in order to prevent errors.

The period linked to the plugged_seconds is equal to 24 hours, it is converted into milliseconds according to the following operation: $DAY = 24 * 60 * 60 * 1000$.

In order to calculate the plugged second in a day it is implemented the method of Redis-Client .*createrule*, in which are passed as argument the following parameters:

- *source_key* equals to *mac_power*;
- *dest_key* equals to *mac_plugged*;
- *aggregation_type* equals to *'sum'*. The power information could be equals to 0 or to 1. If this variable is treated as

a count, extending it in a range of 24 hours it is obtained the *plugged_second* in a day;
- *bucket_size_msec* is considered equals to *DAY* in order to calculate the plugged day once every day.

B. Report and comment the calculations made for setting the retention period of the three timeseries.

The retention period is setting based on the size of the acquisition period. The battery and the power have the same values both for memory size (*5MB*) and for acquisition period (*1 seconds*). Due to this, the computation performed are the same.

More in detail, the initial values of *5MB* is considered as a compressed memory that improves performance according to a lower number of memory accesses.

- By using an average compression ratio of 90%, the uncompressed memory is achieved with the following formula: $5 \quad MB + 90\% = 50 \quad MB$.
- Considering the following definition as a foundation $1 \quad Megabyte = 1048576 \quad byte$, the uncompressed memory is rewritten in the following way: $50 \quad Megabyte = 52428800 \quad byte$.
- Redis requires that each record be 16 bytes in size, hence the total amount of memory is splitted by this quantity. $52428800 \quad byte/16 = 3276800 \quad byte$.
- Each second, the information needs to be kept. $3276800 \quad byte * 1 \quad s = 3276800 \quad s$.
- The final outcome is converted into days and lead to a final result of 37,92 days $3276800/60/60/24 = 37,92 \quad days$.

To calculate the retention period for the plugged_second it is considered a memory size of *1MB* and an acquisition period of *24 hours*. The computations are similar to the previous one:

- $1 \quad MB + 90\% = 10 \quad MB$.
- $10 \quad Megabyte = 10485760 \quad byte$.
- $10485760 \quad byte/16 = 655360 \quad byte$.
- $655360 \quad byte * 24 * 60 * 60s = 56623104000 \quad s$.
- The final outcome has been converted into years and led to a final result of 1795,51 years $56623104000s/31536000(s/year) = 1795,51 \quad years$.