Team 25 - Homework 1

Machine Learning for IoT
Politecnico di Torino

I. VAD Optimization and Deployment

The approach adopted to identify the VAD hyper-parameters compliant with the constraints has been the following one:

1) Focusing on one hyper-parameter at a time and analyzing in broad terms how it affects both accuracy and latency, exploiting the theoretical knowledge.
2) Conducting a more fine-grained analysis within the range of values previously identified as the most promising, still considering one hyper-parameter at time.
3) Once having a sub-optimal value for each of the hyper-parameters in the scenario in which all the others are kept fixed, trying to consider the possible interactions and fine-tune them.

The following table shows the parameters' values compliant with the constraints. In particular, using the following values has been possible achieving an accuracy equals to **98.78%** and a median latency saving around to **26%**.

Parameter	Selected Value
Frame Length (s)	0.032
Number of Mel bins	20
Lower Frequency (Hz)	50
Upper Frequency (Hz)	450
dbFS Threshold (dB)	-95
Duration Threshold (s)	0.10

In order to be compliant with the constraints the following analysis on the different hyper-parameters has been conducted.

- **Frame Length:** it represents the hyper-parameter with the greatest impact on the ***latency*** saving.
  Its value affects the shape of the matrices employed in the MatMul operation used to create the Mel-spectrogram. Therefore, decreasing their dimensionality allows to speed-up the process.
  In addition, using power-of-two values significantly increase the performance in terms of latency due to the way FFT algorithms are designed. Its value influences also the level of ***accuracy*** due to its direct impact on the trade-off between time and frequency resolutions. .
  It is reasonable to be more interested in having an higher time resolution with respect to the frequency one, since in **Voice Activity Detection(VAD)** problems distinguishing the different phonemes is not relevant.
- **# of Mel Bins:** it has a significant impact on the ***accuracy*** since its value affects the resolution in the frequency domain, larger value should lead to higher precision.
  It should be noticed that this parameter also affects the time complexity of the MatMul operations that convert the frequency values from the linear to the Mel scale.
- **Lower/Upper Frequency:** their values have an impact on the ***accuracy***, since they represent the starting and the ending point of the Mel scale. In particular, their values have been set taking into account that the human voice fundamental frequency is between 60 Hz (male) and 350 Hz (female or children).
- **dbFS Threshold:** it strongly affects the ***accuracy*** level, since based on its value a frame can be classified as silent or non-silent.
  Since the number of Mel bins affects the Mel Spectrogram values and consequently the energy values computed. It has been noticed that there is a connection between the optimal value of dbFS Threshold and the number of Mel Bins.
- **Duration Threshold:** it has a great impact on the ***accuracy***. In the dataset provided, the words are quite short in duration (go, down, left,..) for this reason a duration threshold excessively high (e.g. 0.2 seconds in the reference setting) could lead to an elevated number of non-silent audio classified as silent ones.

II. Memory-constrained Timeseries Processing

Before creating the ***mac_address*:plugged_seconds** timeseries has been noticed that in order to compute how many seconds the power has been plugged in the last hour was necessary to consider the *mac_address*:power timeseries, take the values in the last hour (0 or 1) and sum them. It was possible since the acquisition time of the *mac_address*:power timeseries was set to 1 second.
It has been done by using the *createrule* command. The retention time instead has been set through the *alter* command.

In order to compute how many clients can be monitored considering a database with a maximum of 100 MB of memory, it has been necessary calculating the number of bytes used for each client. The first step consisted of computing the number of records stored for each client. It has been obtained dividing the retention period (in seconds) by the acquisition period (in seconds).
Then, the number of bytes for each client has been obtained multiplying the previous result (# of records) by 16 that is the number of bytes for each record stored in a Redis timeseries. The previous steps have been repeated for the three timeseries.

TimeSeries Name	Acquisition Period	Retention Period	Total n. of record	Bytes stored
battery	1s	1 day	86400	1382400
power	1s	1 day	86400	1382400
plugged_seconds	3600s	30 days	720	11520

The total number of bytes for each client has been computed summing the number of bytes of the three timeseries, obtaining a result equals to 2776320 Bytes.
Before calculating the number of clients that can be monitored it has been necessary taking into account that all the timeseries are stored with compression activated. It has resulted in decreasing the total number of bytes by 90%, (average saving), obtaining 277632 Bytes. Finally, the number of clients has been obtained using the following formula:

$$\text{Number of Clients} = \frac{100 \cdot 2^{20}}{277632} = 377$$

It is necessary to underline that:

1) The header size has been neglected.
2) The fact that the memory usage is always a multiple of the chunk size has been not considered.