Homework 1

Fabrizio Battiloro (s317786) - Lorenzo Mossotto (s318971) - Arash Daneshvar (s314415)

Voice Activity Detection Optimization & Deployment

1. Methodology Adopted

Starting from the code provided by the professor, we began to analyze experimentally the general variations in performance. Initially, we manually tested different configurations to understand how each parameter influenced accuracy and latency. Once we grasped the training behavior, we selected a series of values that we considered potentially acceptable and inserted them into a grid search to thoroughly test all combinations. During this initial experimental phase, we applied theoretical concepts learned in class, especially concerning FFT analysis. Specifically, for analyzing audio tracks, the *preprocessing.py* script uses STFT (Short-Time Fourier Transform), which is essentially frequency domain analysis based on FFT. As we know, in this case, the number of input samples (N) must be a power of 2.

2. Result Table

In *TABLE I* are reported various results that meet the constraints specified in the assignment, namely an accuracy greater than 98.5% and a latency saving exceeding 20%. To properly interpret the provided data, it is important to underline that the reference latency, measured with the standard values, at the time of the test was 5.9 ms, while the sampling frequency always remained constant at 16000 Hz.

FL	MB	L	UF	dbFST	DT	A	L
0.032	10.0	0	6000	-35	0.1	98.89	3.92
0.032	10.0	0	8000	-35	0.1	98.89	4.03
0.032	10.0	100	6000	-35	0.1	98.56	4.00
0.032	10.0	100	8000	-35	0.1	98.56	3.86
0.064	10.0	0	6000	-35	0.1	98.67	3.90
0.064	10.0	0	6000	-35	0.2	98.67	3.88
0.064	10.0	0	8000	-35	0.2	98.89	3.91
0.064	10.0	100	6000	-35	0.1	98.67	3.87
0.064	10.0	100	6000	-35	0.2	98.67	3.89
0.064	10.0	100	8000	-35	0.1	98.89	3.88
0.064	10.0	100	8000	-35	0.2	98.78	3.95
0.064	20.0	0	6000	-35	0.1	98.89	4.05
0.064	20.0	0	8000	-35	0.1	98.89	4.06
0.064	20.0	100	6000	-35	0.1	98.78	4.05
0.064	20.0	100	8000	-35	0.1	98.89	4.05

3. Result explanation and conclusion

Regarding the frame length, four parameters were tested: 0.016, 0.032, 0.064, and 0.124. As demonstrated, positive results were not obtained with values of 0.016 or 0.124. In the first case, the size is too small to interpolate sufficient data, while in the second case, it is too large, resulting in significantly lower accuracy performance, suffering from overfitting and so unable to generalize well. Other relevant considerations can be made regarding the 'Duration Threshold.' In this context of audio analysis, it emerged that a threshold value of 0.1 benefits both accuracy and latency. This suggests that, for this dataset, an appropriate measure to determine the presence or absence of silence is 0.1 seconds. For what concerns lower and upper frequencies, they seem to have a low impact on the results, with minimal variations. However, it appears that an upper frequency of 6000 (with a slight latency saving) and a zero lower frequency (for a slight increase in accuracy) are preferred, but the variations are so minimal as to be almost negligible. Regarding the number of 'mel bins', we observed that an increase led to a moderate decrease in accuracy and lower latency. For this reason, we opted for a maximum of 20.

Memory-constrained Timeseries Processing

1. Time Series Explanation

Regarding the second issue, we used the codebase from the classroom lab where it was explained how to retrieve battery-related data (whether charging or not, and the percentage). Subsequently, we adapted it to our needs. Specifically, after logging into our Redis Database, we initialized the time series with its respective retention time. In particular, we created a new TimeSeries with the name *mac_address*:plugged_seconds and with a retention time of 30 days. Then, through an aggregation rule, we summed the values of the time series *mac_address*:power every hour, because it will have a value of 1 if the computer was connected to power or 0 otherwise. Utilizing this method has allowed us to quickly and efficiently calculate the total seconds the computer has been connected to power, without performing the calculation locally on the machine but directly on the server.

2. Number of client

To calculate the maximum number of clients based on the delivery specifications, we divided the available database space (100 MB) by the maximum space occupied by a single client.

The maximum size of a client is determined by the following formula:

$$\begin{aligned} \text{plugged\_seconds} &= 24 \times 30 \times 16 \\ \text{power} &= 60 \times 60 \times 24 \times 16 \\ \text{battery} &= 60 \times 60 \times 24 \times 16 \\ \text{client} &= \text{plugged\_seconds} + \text{power} + \text{battery} \end{aligned} \tag{1}$$

Lastly, we converted the unit from bytes to megabytes, by dividing by 2^{20}, and calculated the final number of megabytes after compression and taking into account the average compression ratio. Dividing 100 MB by the obtained value, 0.26477 MB, we derived a maximum of 377 clients.