Machine Learning for IoT - Homework 1

Silva Bashllari
Student number: 299317

Davide di Mauro
Student number: 306089

Lal Akin
Student number: 300192

I. Exercise 1

For the VAD Optimization, the four hyper-parameters taken into consideration are the following: *frame_length_in_s*, *dbFSthresh*, *duration_time* and the *downsampling_rate*. We had to maintain the value of the *downsampling_rate* at 16000 Hz due to an increase in latency, up to 20 ms, if its value were any different. By utilizing a function similar to grid search, a greedy approach, we tried different values of the remaining three hyper-parameters, as can be observed in the Table 1:

TABLE I
Hyper-parameters Grid

Hyper-parameters	**Values**
downsampling_rate	{1600}
frame_length_in_s	{0.004, 0.008, 0.016, 0.032, 0.064}
dbFSthresh	{-80, -90, -100, -110, -120, -130, -140}
duration_time	{0.016, 0.032, 0.064, 0.128, 0.256, 0.512}

1) The parameters that impact **latency** are *downsampling_rate* and *frame_length_in_s*. This is justified by the fact that these parameters are used to define the variables *frame_length*, *frame_step* and *fft_length* needed for the *Short-Time Fourier Transform* in order to compute the spectrogram. We kept the *downsampling_rate* static, as mentioned above, thus, the only parameter tuned is the *frame_length_in_s*. When we increase and decrease its value, and consider the total number of computations *(nr-of-frequency-bins* multiplied by *nr-of-frames*) we observe that it is always in the range [8100-7695], the variance for each change is small and there is no pattern, thus, having an inconsequential impact on latency. This, combined also with the fact that **the biggest factor influencing latency is DeepNote's resources allocation system**, makes it difficult to draw conclusions and discern a clear relationship. Considering also the theoretical aspects of the computational complexity of STFT [1] but also taking into account the fact that our frame-step and frame-length are exactly the same value, it could justify our experimental results. Furthermore, we choose values in the range [10-50] ms as provıded by the theory and found out that using power of 2 values worked out best and the better performing ones were 0.016 and 0.032 seconds.
2) The parameters that impact **accuracy** are *downsampling_rate*, *dbFSthres*, *frame_length_in_s* and *duration_time*. The *dbFSthres* and the *duration_time* are not used in any computations but are just thresholds used to compare values, per consequence, not affecting the latency but affecting accuracy. To tune them we tried different values and we choose values of -140 dbFS and 0.128 seconds, as the best performing ones. As for the *frame_length_in_s*, considering that increasing this parameter, increases the number of frequency bins and reduces the number of frames, we have a higher frequency resolution but a lower time resolution. It's exactly the time resolution that is of interest to us for accuracy. As for the *downsampling_rate*, we kept it static but theoretically, decreasing it has an inverse impact on accuracy, as we would be losing information.

To create a better understanding, we visualized the hyper-parameter tuning process using the parallel coordinates graph, which can be observed in Figure 1.

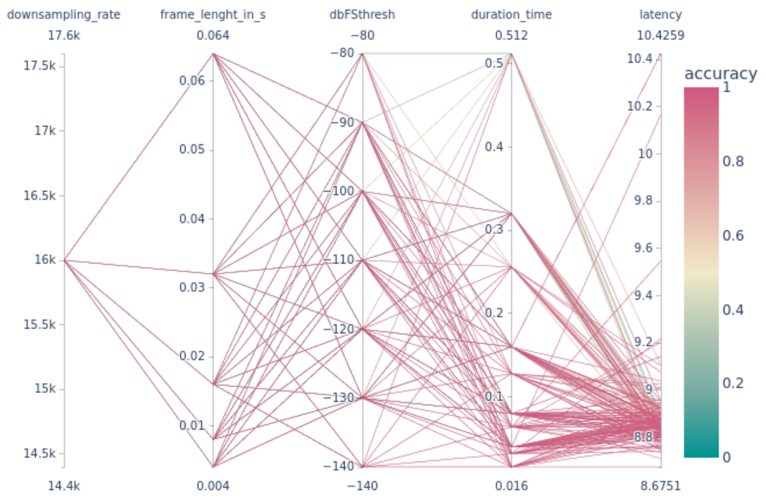

Fig. 1. Parallel Coordinates Graph

However, we have to consider only the values that are above the required threshold of accuracy = 0.98 and of latency = 9 ms. Thus, we used the Pareto frontier [2] in Figure 2 to highlight which are the "acceptable" ones from the points that satisfy the constraints. The Pareto frontier, used in multi-objective optimization, allows the restriction of attention to the set of efficient choices, and to make trade-offs within this set (in our case to consider trade-offs between latency and accuracy), rather than considering the full range of every parameter. In our case, multiple values are in the Pareto line

so choosing between them depends on the specific application but we simply chose the one with the best accuracy.
Due to the large cardinality of the combinations, we are reporting only the best one we chose in Table 2, producing an accuracy of **0.991** and a latency of **8.835** ms.

TABLE II
BEST PERFORMING HYPERPARAMETERS

downsampling_rate	***frame_length_in_s***	***dbFSthres***	***duration_time***
16000 Hz	0.016 s	-140 dbFS	0.128 s

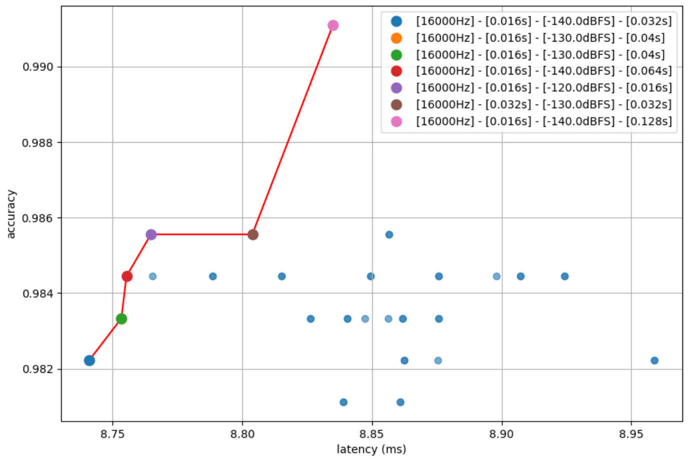

Fig. 2. Pareto Frontier

II. EXERCISE 2

1) **Power Plugged Seconds**
   We created a new time series to store how many seconds within the last 24 hours the computer was plugged in power.
   - First step is to define a new time series, *power_plugged_seconds*, which will store the total number of seconds the battery is plugged into power each 24 hours; the chunk size is set to 128 bytes in an arbitrary manner.
   - Next, to make the aggregation possible, we defined a new rule to give in the set-up configurations of *power_plugged_seconds*. This time series will contain the sum of the values of the *power* time series. The bucket duration the time series will use is set to a day in milliseconds.
   - For each second a new entry will be added to the two time series *power* and *battery*, and each 24 hours a new entry for the *power_plugged_seconds* time series will be automatically computed and stored.
2) **Memory Constrained Retention Periods** The retention periods are to be set so that they comply with the memory constraints. This is done by calculating how much time it takes to fill the memory as much as possible.
   - *battery* and *power* have 5 MB each as the constraint. The retention period for these two is calculated as dividing 2^{20} bytes (how many bytes there are in a MB) by 1.6 bytes (size of each compressed entry in Redis since the compression factor is taken as default 90%), then multiplying by 5 and 1000 to get the time for 5MB's, in milliseconds.
   $$5 \cdot \left(\frac{2^{20}}{1.6}\right) \cdot 1000 = 3.2768 \cdot 10^{9} \text{ ms} \qquad (1)$$
   - *power_plugged_seconds* retention period is calculated by the same steps but not multiplying by 5, because of the problem specifications. Also it is multiplied by the numbers below to convert to the right time since a new entry is added each day.
   $$\left(\frac{2^{20}}{1.6}\right) \cdot (24 \cdot 60^{2} \cdot 1000) = 56.623104 \cdot 10^{12} \text{ ms} \quad (2)$$

REFERENCES

[1] Wikipedia contributors, "Short-time fourier transform — Wikipedia, the free encyclopedia," https://en.wikipedia.org/w/index.php?title=Short-time_Fourier_transformoldid=1115436538, 2022, [Online; accessed 28-November-2022].
[2] ——, "Pareto principle — Wikipedia, the free encyclopedia," https://en.wikipedia.org/w/index.php?title=Pareto_principleoldid=1119913844, 2022, [Online; accessed 28-November-2022].