Timeseries Processing for Memory Optimization

1.1 Memory optimization for Temperature & Humidity Monitoring System

The timeseries for the collection of temperature and humidity data were created and initialized with a retention period of 30 days and 128 bytes for each data chunk. The same chunk size was adopted to generate additional timeseries with a different retention period, equal to 365 days, enabling the performance of aggregate operations. Such aggregate operations return the minimum, maximum and average value of both temperature and humidity within an hour. Ultimately, all the timeseries created so far have compression disabled by default. The sensor collects data according to a data acquisition interval of 2 seconds.

1.2 Calculations

In order to compute how much memory (in GB) is needed to provide the monitoring service for 1000 clients, we consider the average compression ratio.

Retention period of 30 days

Since the retention period is equal to one month and the number of clients is 1000, we have:

1 month=30 days*24 hours*6o minutes*60 seconds*1000=2592000 seconds

with a data acquisition interval equal to 2 seconds, the number of records becomes:

# of records=2592000 / 2=1296000

It is now possible to compute the uncompressed memory:

uncompressed memory=1296000*16 bytes=20736000 bytes

Let’s convert in MB:

20736000/1048576 =19.775 MB

Now it is possible to evaluate the compressed memory:

compressed memory= 19.775 - 90%=1.9775 MB

Retention period of 365 days

1 year=365 days*24 hours*6o minutes*60 seconds*1000=31536000000 seconds

# of records= 31536000000 / 2=15768000000

uncompressed memory=15768000000*16 bytes=252288000000 bytes

in MB: 252288000000/1048576 =240600.586 MB

compressed memory= 240600586 - 90%=24060.06 MB

Voice Activity Detection Optimization & Deployment

2.1 VAD Optimization

The goal is finding the optimal parameter values of the Voice Activity Detection (VAD) class, that allow an accuracy on the VAD dataset above 97% and a latency on the RPI under 25 ms. The algorithm employed works by iterating through all possible combinations of the following parameters: ***frame length in s, frame step, dBthres, duration threshold.*** The *sampling rate* is kept constant. The algorithm returns the configurations of parameters that enable achieving the top 5 highest accuracy values.

The parameters selected were inserted in the *VAD_latency.py* script, in order to determine the values that also respected the latency constraint.

2.2. VAD Deployment

The code used to realize the voice-controlled data collection is divided in three main parts:

1. The Voice User Interface part, containing the VUI class and the data collection methods
2. The Voice Activity Detection part, containing the VAD class and the Spectrogram class
3. The audio transformation part, containing the Normalization class and the AudioProcessor class

The VUI class continuously records chunks of audio of 1s duration. Before passing these chunks of audio to the method *is_silence* of the VAD class, the audio is processed and normalized by using the Normalization and AudioProcessor classes. Once this is done, the VAD class is employed to check whether the recording contains speech or not. In order to do so in the optimal way, the VAD class has been initialized with the parameters found in the previous section.

Since the data collection is initially disabled, if the audio does not contain speech, the VUI keeps recording audio, otherwise the VUI will enable the data collection. The change in state is controlled via a state variable that is updated every time the VAD class detects a sound. Before checking for any changes in the state, the system has to stay at least 5s in the current state. This check is realized by keeping track of the time of the last status change and subtracting it to the current time.

2.3 Considerations on the optimal parameters search

The hyperparameters were selected for the purpose of meeting the following constraints:

- Accuracy on the VAD dataset $>$ 97.6%
- Median Latency on the RPI $<$ 25 ms

Search space

Frame length: considering the nature of the dataset, which contains relatively short recordings, frame length values on the order of milliseconds have been chosen. The selected values are multiples of two, ranging from 0.008 ms to 0.64 ms. The choice of the range is motivated by the fact that, since the FFT

is particularly efficient with a computational complexity of NlogN when N is even, the data can be processed more efficiently if its size is a power of two.

Frame step: the values correspond to the overlap percentages. Conventionally, 25%, 50%, and 75% have been selected.

dBthreshold: the range is limited and goes from 10 to 15. This range provides an optimal trade-off between latency and accuracy of the VAD system. Reducing the value below 10 may lead to the incorrect detection of low-energy frames as words, while increasing it above 15 could cause a delay in word detection, thus increasing latency. Therefore, a *dBthreshold* between 10 and 15 is optimal for achieving balanced performance, considering the scenario of the VAD dataset where words are clearly pronounced and there is not much background noise.

Duration threshold: the range is from 0.15 to 0.20 seconds. Setting a value that is too low would lead to premature detection of short, negligible sounds; however, an excessive parameter could lead to delays in detecting short words.

Table 1 shows the hyperparameters and their corresponding optimal value, this configuration allows us to obtain an accuracy of 97.89% and a latency of 19.6 ms.

Hyperparameter	**Optimal values**
frame_length_in_s	0.032
frame_step	0.016
dBthres	10
duration_thres	0.150

Table 1: optimal values of the hyperparameters of the Vad class

Commentary on the impact of each parameter on latency and accuracy

Frame Length (s):

Shorter frame lengths capture rapid changes in audio, enhancing accuracy. In fact, it has been observed that by decreasing the parameter, accuracy tends to improve, while latency frequently decreases. Therefore, reducing frame length allows to achieve a good trade-off between accuracy and latency.

Frame step:

Increasing the frame step reduces both accuracy and latency. When the distance between two consecutive frames increases, there may be a loss of information, so a reduction in accuracy. At the same time latency decreases as well since the system needs to process fewer frames per time unit.

dB threshold:

When lowering the threshold accuracy increases, since a more permissive threshold ensures that no useful signal is ignored, allowing the VAD to better detect low-intensity sounds (such as the beginning

and end of a word). On the contrary, increasing the value reduces the accuracy, as parts of the vocal signal are ignored.

In reference to latency, no significant variations have been observed in the two cases considered; nevertheless, it remains below the constraint value.

Duration Threshold (s):

When reducing the duration threshold, latency decreases, while accuracy increases. This is due to the fact that less silent frames are necessary to classify a sound as speech, and since the dataset is made of short and noisy segments, it helps to detect them faster and better. If increased, it has been observed that latency remains almost unchanged, whereas the accuracy drops significantly.

Comparison with the original parameters

Comparing the original parameters of the *VAD_latency.py* script with the optimal configuration found, the greatest difference is observed for the *dB threshold* and *duration threshold* values. Table 2 shows how the value of these parameters is much higher in the original script, this can explain why the optimal parameters found lead to a lower latency.

A longer duration threshold requires more silent frames to verify that the signal is a valid word, thus increasing the time required for this process. A higher dB threshold leads to greater delay due to the time needed for a frame with sufficient energy to overtake this new threshold.

Hyperparameter	**Optimal values**	**Original values**
frame_length_in_s	0.032	0.04
frame_step	0.016	0.001
dBthres	10	20
duration_thres	0.150	0.4

Table 2: original and optimal values of the hyperparameters