ML4IOT HomeWork1

Authors: Suyash Singh, Harsh Vasoya, Junaid Shah
Student IDs: s307798, s308347, s319928
Politecnico di Torino

Abstract- In this report, we optimize a **Voice Activity Detection (VAD)** system for IoT applications, focusing on accuracy and low latency. Our custom Python script records and efficiently detects speech using machine learning models. Additionally, we develop a **memory-efficient battery monitoring system for IoT devices,** addressing memory constraints. These exercises showcase the integration of machine learning with IoT constraints.

Exercise 1: Voice Activity Detection Optimization & Deployment

In Exercise 1, we optimized a Voice Activity Detection (VAD) system for IoT applications. By fine-tuning hyperparameters, including **frame length**, **mel frequency bins**, **lower frequency**, **upper frequency**, **dbFS threshold**, and **duration threshold**, we achieved over **98.5%** accuracy with a **20%** reduction in latency compared to the baseline model. This enhances VAD's performance in resource-constrained IoT environments.

Methodology and Hyper-Parameter Selection

Our methodology aimed at seamlessly integrating real-time audio processing with the VAD system. We employed a global buffer to manage ongoing audio recordings, with the script designed to terminate upon the user's command, ensuring operational flexibility.

The VAD hyper-parameters were meticulously chosen to meet constraints and enhance speech detection accuracy and responsiveness. The table below outlines the selected hyper-parameters:

Parameter	**Value**
frame_length_in_s	**0.035**
num_mel_bins	**12**
lower_frequency	**0 Hz**
upper_frequency	**7500 Hz**
dbFSthres	**-42**
duration_thres	**0.12 seconds**

Table 1: Selected VAD hyper-parameters.

Each parameter was chosen to strike a balance between accuracy and latency. For instance, the **dbFS threshold** value of **-42** ensures that only sufficiently loud audio is considered speech, enhancing accuracy. The **duration threshold** of **0.12** seconds minimizes the risk of cutting off brief speech moments, reducing latency. These parameters collectively contribute to a VAD system that surpasses the reference model in both responsiveness and reliability.

Exercise 2: Memory-constrained Timeseries Processing

2.1 Setup of mac_address:plugged_seconds Timeseries

The timeseries **mac_address:plugged_seconds** was created to track the duration (in seconds) a device remained plugged in every hour. The key steps in the setup are:

- Initialization of the timeseries with a retention period of **30 days**.
- Accumulation of plugged-in seconds, incremented every second when the device is plugged in.
- Recording the accumulated seconds in the timeseries *hourly*.

2.2 Client Capacity Calculation

To determine the client capacity under a **100 MB** memory limit, we consider the data retention and compression ratio. Key parameters are:

- Data retention: **1 day** for *battery* and *power*, **30 days** for *plugged_seconds*.
- Memory limit: **100 MB**.
- Compression ratio: **4:1**.

The formula for memory usage per client is:

$$\textbf{(MB)} = \frac{(DailyPoints \times 86400 \times 8) + (MonthlyPoints \times 24 \times 30 \times 8)}{4 \times 10^6} \tag{1}$$

Where,

- ***Daily Points*** $= 3$ (for battery and power).
- ***Monthly Points*** $= 1$ (for plugged_seconds).

Substituting values, the memory required per client is approximately **0.27 MB**.

Thus, the maximum client capacity is calculated as:

$$\textbf{Max Clients} = \frac{100}{0.27} \approx \mathbf{370}\, clients \tag{2}$$

This yields a maximum client support of about **370** with a **100** MB memory constraint.