Machine Learning for IoT

Homework 1

*****DUE DATE: 28 Nov (h23:59)*****

Submission Instructions:

Each group will send an e-mail to andrea.calimera@polito.it and valentino.peluso@polito.it (in cc) with subject *ML4IOT23 TeamN* (replace *N* with the team ID). Attached with the e-mail a single ZIP archive (*.zip*) named *HW1_TeamN.zip* (replace *N* with the team ID) containing the following files:

1. The code deliverables specified in the text of each exercise.
2. One-page pdf report, titled *TeamN_Homework1.pdf*, organized in different sections (one for each exercise). Each section should motivate the main adopted design choices and discuss the outcome of the exercise.

Late messages, or messages not compliant with the above specs, will be automatically discarded.

Exercise 1: Voice Activity Detection Optimization & Deployment (3 points)

1.1 VAD Optimization (1pt)

In Deepnote, optimize the hyper-parameters of the *is_silence* method (*LAB2 – Exercise* 2) such that all the constraints below are met:

- Accuracy on the *vad-dataset* > 98%
- Average Latency < 9 ms

Note: Measure latency in Deepnote using the *time* method from the *time* module.

1.2 VAD Deployment (1pt)

On your PC, develop a Python script that continuously record audio data and store it on disk if it contains speech. Follow the instructions reported below:

- In VS Code, write a Python script to record audio with your PC and the integrated/USB microphone. Set the # of channels to 1, the resolution to *int16*, and the sampling frequency to 16 kHz (if not supported by your microphone, apply resampling).
- In the same script, develop a new version of the *is_silence* method that takes as input a *numpy* array instead of a filename. Specifically, replace the *get_audio_and_label* method, with a new method called *get_audio_from_numpy* with the following code:

```
def get_audio_from_numpy(indata):
    indata = tf.convert_to_tensor(indata, dtype=tf.float32)
    indata = 2 * ((indata + 32768) / (32767 + 32768)) - 1
    indata = tf.squeeze(indata)

    return indata
```

- Every 1 second (in parallel with the recording), check if the recorded audio contains speech using the new version of the *is_silence* method with the hyper-parameters of 1.1. If *is_silence* returns 0, store the audio data on disk using the timestamp as filename, otherwise discard it.
- The script should be run from the command line interface and should take as input a single argument called --*device* (int) that specifies the ID of the microphone used for recording.

Example:

```
python ex1.py --device 0
```

1.3 Reporting (1pt)

In the PDF report:

- Describe the methodology adopted to discover the VAD hyper-parameters compliant with the constraints.
- Add a table that reports the selected values of the VAD hyper-parameters (*downsampling_rate*, *frame_length_in_s*, *dbFSthres*, *duration_thres*).
- Comment the table and explain which hyper-parameters affect accuracy and latency, respectively. Motivate your answer.

Code Deliverables

- A single Python script named *ex1.py* that contains the code of 1.2. The code is intended to be run on a laptop and must use only the packages that get installed with *requirements.txt* provided during the labs. Moreover, the script should contain all the methods needed for its correct execution.

<u>Exercise 2:</u> Memory-constrained Timeseries Processing (3 points)

2.1 Memory-constrained Battery Monitoring System (2pt)

Starting from the solution of *LAB1 – Exercise 2.1*, design and develop a memory-constrained battery monitoring system with the following specifications:

- Set the acquisition period of *mac_address:*battery and *mac_address:*power to 1 second.
- Create a new timeseries called *mac_address:*plugged_seconds that, every 24 hours, automatically stores how many seconds the power have been plugged in the last 24 hours.
- Set the largest retention period possible for the created timeseries such that the following constraints are met:
    - *mac_address:*battery memory size < 5 MB
    - *mac_address:*power memory size < 5 MB
    - *mac_address:*plugged_seconds < 1 MB

  Create all the timeseries with compression activated and consider the average compression ratio for computing the retention period.
- The script should be run from the command line interface and should take as input the following arguments:
    - --host (*str*): the Redis Cloud host.
    - --port (*int*): the Redis Cloud port.
    - --user (*str*): the Redis Cloud username.
    - --password (*str*): the Redis Cloud password.

2.2 Reporting (1pt)

In the PDF report:

- Explain how you created and set up the *mac_address:*plugged_seconds timeseries.
- Report and comment the calculations made for setting the retention period of the three timeseries.

Code Deliverables

a) A single Python script named *ex2.py* that contains the code of 2.1. The code is intended to be run on a laptop and must use only the packages that get installed with *requirements.txt* provided during the labs. Moreover, the script should contain all the methods needed for its correct execution.