Machine Learning for IoT

Homework 2

*****DUE DATE: 7 Jan (h23:59)*****

Submission Instructions:

Each group will send an e-mail to daniele.jahier@polito.it and valentino.peluso@polito.it (in cc) with subject *ML4IOT25 GroupN* (replace *N* with the group ID). Attached to the e-mail a single ZIP archive (*.zip*) named *HW2_GroupN.zip* (replace *N* with the group ID) containing the following files:

1. The code deliverables specified in the text of each exercise.
2. One-page pdf report, titled *GroupN_Homework2.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: Training & Deployment of a "Up/Down" Keyword Spotter (6 points)

1.1 Training & Optimization (3pts)

In Deepnote, create a Python notebook to train and optimize a classification model for **"up/down"** keyword spotting on the Mini Speech Command (MSC) dataset.
The Python notebook must include the following steps:

- Develop a data pipeline that processes of **"down"** and **"up"** keywords from the MSC train, validation, and test splits, mapping **"down"** to **0** and **"up"** to **1**.
- Define the model architecture and the training, optimization, and testing flow.
- Generate and save the *TFLite* version of the trained model.

The model must meet all the constraints below:

- Accuracy measured on the test set > 99.4% (1pt)
- *TFLite* Size < 50 KB (1pt)
- Total Median Latency on the **RPI** < 40 ms (1pt)

The *TFLite* model can be provided in *TFLite* format or *ZIP* format. The *TFLite* Size must be measured on the selected format (1 KB = 1024 bytes).

- Use the Deepnote notebook *HW2: Evaluation of TFLite Accuracy and Size* to measure Accuracy and *TFLite* Size.
- Use the script *hw2_latency.py* (*Homework2* folder on *Portale della Didattica*) to measure Total Media Latency on the RPI.

1.2 Deployment & Integration in Smart Hygrometer (2pts)

On the RPI, develop a Python script (*ex1.py*) to measure temperature and humidity, controlled through a Voice User Interface (VUI) based on **VAD** and **KWS**.
The system should perform data collection every 2 seconds and **upload the collected data to Redis** (follow the specifications of *LAB1 – Exercise 2c* for the timeseries naming).
The VUI will enable or disable data collection based on voice commands. The script should follow the requirements below:

- Initially, the data collection is disabled.
- The VUI always runs in background and continuously records audio data with the USB microphone. Configure the audio recording with 1 channel, 16-bit depth (int16), and a sampling rate of 48 kHz.

- Every second, the VUI analyzes the last second of recorded audio to detect the presence of speech using the *is_silence* method from the VAD class introduced in *Homework 1* (you can use the default or optimized version).
- If the VAD returns *non-silence*, the recording is fed to the classification model for **"up/down" spotting** developed in 1.1 and one of the following actions is performed:
    - If the predicted keyword is **"up" with probability > 99%, enable data collection.**
    - If the predicted keyword is **"down" with probability > 99%, stop data collection.**
    - If the **top-1 probability (regardless of the predicted label) is ≤ 99%, remain in the current state.**
- If the VAD returns *silence*, maintain the current state.
- When enabled, the system should measure temperature and humidity every 2 seconds and **upload collected data to Redis.**

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.

1.3 Reporting (1pt)

In the PDF report:

- Describe the methodology adopted to discover the hyper-parameters compliant with the constraints of 1.1.
- Add a table reporting the pre-processing type (log-Mel Spectrogram or MFCCs) and hyper-parameters of the final solution.
- Add a table reporting the training hyper-parameters of the final solution.
- Describe the model architecture and the adopted optimizations.
- Add a table reporting Accuracy (%), *TFLite* Size (KB), and Total Median Latency (ms) of the final solution.
- Comment on the reported results.

Deliverables (3 files)

- Deepnote Training Notebook (1 file):
    - A Python notebook named *training.ipynb* for training and generating the *TFLite* model.

  The training notebook is intended to be run on Deepnote, must use only the packages that get installed with the *ML4IoT Template*, and the *reader.py* and *preprocessing.py* from the *Material* project in Deepnote. It must run without any additional dependency.
- Smart Hygrometer Script (1 file):
    - A single Python script named *ex1.py* that contains the code of 1.2. The code is intended to be run on the Raspberry PI without installing additional software or dependencies on the provided SD card. Moreover, the script must include all necessary classes and methods needed for its correct execution.
- *TFLite* Model (1 file):
    - The *TFLite* model of 1.1. The *TFLite* model must be named *modelN.tflite*, if provided in *TFLite* format, or *modelN.tflite.zip*, if provided in *ZIP* format (replace *N* with the group ID). During the evaluation, Accuracy, *TFLite* Size, and Total Median Latency will be measured on the submitted file.