Machine Learning for IoT
Homework 2
*****DUE DATE: 20 Dec (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 *HW2_TeamN.zip* (replace *N* with the team ID) containing the following files:
1. The deliverables specified in the text of the exercise.
2. One-page pdf report, titled *TeamN_Homework2.pdf*, that motivates the main adopted design choices and discusses 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 "Go/Stop" Classifier (6 points)

1.1 Training & Optimization (3pts)

In Deepnote, create a Python notebook to train and optimize a classification model for "go/stop" spotting. The Python notebook must include the following steps:
- Develop a data pipeline that processes only go/stop WAV files from the MSC dataset for training, validation, and testing.
- Define the model architecture and the training & optimization flow.
- Generate and save the *TFLite* version of the trained model.

The model must meet the following constraints:
- Accuracy > 97% (1pt) and *TFLite* Size < 25 KB (1pt) and Total Latency < 8 ms (1pt)

The *TFLite* model can be provided in *TFLite* format or *ZIP* format. The *TFLite* Size must be measured on the selected format.
The *Total Latency* must be measured on Deepnote computing the median over the test set.

1.2 Deployment & Integration in Smart Battery Monitoring (2pts)

Update the battery monitoring script of *LAB1 – Exercise 2* integrating a Voice User Interface (VUI) based on VAD and KWS.
The monitoring system must measure the battery status (battery level and power plugged) every 1 second and store the collected data on Redis (follow the specifications of *LAB1 – Exercise 2c* for the timeseries naming).
The VUI must provide the user the possibility to start/stop the battery monitoring using "go/stop" voice commands.

Specifically, the script must implement the following behavior:
- Initially, the monitoring is disabled.
- The VUI always runs in background and continuously records audio data with your PC and the integrated/USB microphone. Every 1 second, the VUI checks if the recording contains speech using the VAD function developed in *LAB2 – Exercise 2* (or its optimized version developed in *Homework 1*).

- If the VAD returns *no silence*, the recording is fed to the classification model for “go/stop” spotting developed in 1.1 and one of the following actions is performed:
    - If the predicted keyword is “go” with probability > 95%, start the monitoring.
    - If the predicted keyword is “stop” with probability > 95%, stop the monitoring.
    - If the top-1 probability (regardless of the predicted label) is ≤ 95%, remain in the current state.
- If the VAD returns *silence*, remain in the current state.

The script should be run from the command line interface and should take as input the following arguments:

- --device (*int*): the ID of the microphone used for recording.
- --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 that reports the pre-processing hyper-parameters of the final solution.
- Add a table that reports the training hyper-parameters of the final solution.
- Describe the model architecture and the adopted optimizations.
- Add a table that reports Accuracy, *TFLite* Size (in KB), and Total Latency (ms) of the final solution.
- Comment on the reported results.

Deliverables

- Deepnote files (2 files):
    - A Python notebook named *training.ipynb* for training and generating the *TFLite* model.
    - The pre-processing script named *preprocessing.py* containing the pre-processing methods used in *training.ipynb*.

  The training notebook is intended to be run on Deepnote, must use only the packages that get installed with *requirements.txt* uploaded on the workspace and must run without any additional dependency.

- Smart Battery Monitoring 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 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.

- *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 team ID). During the evaluation, Accuracy, *TFLite* Size, and Total Latency will be measured on the submitted file.