Machine Learning for IoT

Homework 2

*****DUE DATE: 10 Jan (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 <ML4IOT22 GroupN> (N is the group ID). Attached with the e-mail the following files:

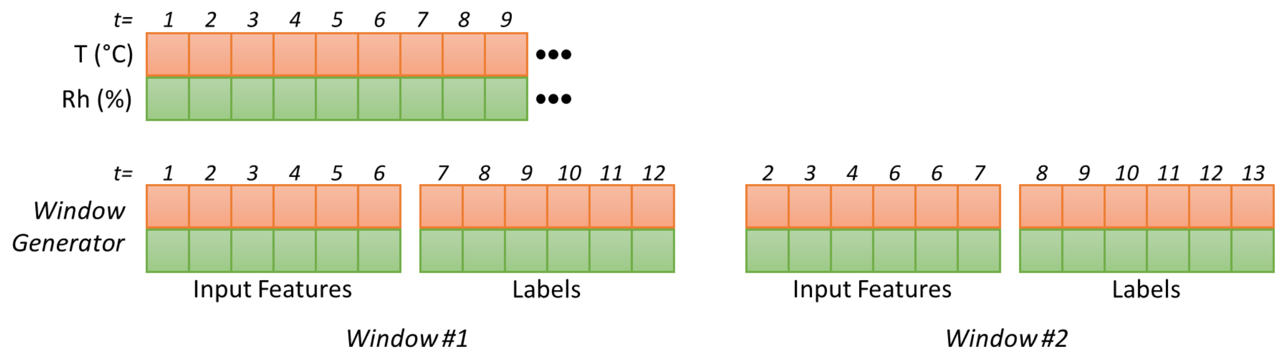
1. One single .py file for each exercise, titled <HW2_exM_GroupN.py>, where M is the exercise number and N is the group ID, containing the Python code. The code must use only the packages that get installed with *requirements.txt*.
2. One-page pdf report, titled <GroupN_Homework2.pdf>, organized in different sections (one for each exercise). Each section should motivate the main adopted design choices and discuss the outcome of the exercise.
3. The TFLite models generated in the exercises (more details provided later in the text).

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

Exercise 1: Multi-Step Temperature and Humidity Forecasting (3 points)

- Write a Python script to train multi-output models for temperature and humidity forecasting to infer multi-step predictions, i.e., a sequence of future values. The script must support different numbers of output steps. Use the Jena Climate Dataset with a 70%/20%/10% train/validation/test split (same as Lab 3).

- Implement a data-preparation pipeline compliant with multi-step predictions. Specifically, the labels shape should be [#Batches, #Output Steps, #Features], e.g., in the figure below the shape is [32, 6, 2]. Use the *WindowGenerator* class of Lab3 as starting point.



- Implement multi-output/multi-step models, i.e., with an output shape equal to [#Batches, #Output Steps, #Features]. Use the models developed in Lab3 as starting point.

- Implement a *Keras* metric that computes the mean absolute error of temperature and humidity on multi-step predictions (the error shape is [#Features]). Use the error metric developed in Lab3 as starting point.

- Train two different model versions, each one meeting the following constraints, respectively:
  Version a):
  - #Output Steps = 3
  - T MAE < 0.3 °C
  - Rh MAE < 1.2%
  - TFLite Size < 1.5 kB

  Version b):
  - #Output Steps = 9
  - T MAE < 0.7 °C
  - Rh MAE < 2.5%
  - TFLite Size < 1.8 kB

  Note #1: For each version, all the requirements must be met.
  Note #2: The models must be trained on the training set only and evaluated on the test set.

- Submit the TFLite models (named *GroupN_th_a.tflite* and *GroupN_th_b.tflite*), together with one single Python script to train and optimize them. If you have compressed the TFLite file with *zlib,* append .zlib to the filename (e.g., *GroupN_th_a.tflite.zlib*).
  The script should take as input argument the model version:

  ```
  python HW2_ex1_GroupN.py --version <VERSION>
  ```

  where `N` is the group ID and `<VERSION>` is “a” or “b”, and return as output the TFLite file.

- In the report, explain and motivate the methodology adopted to meet the constraints (discuss on model architecture, optimizations, hyper-parameters, etc.).

Exercise 2: Keyword Spotting (3 points)

- Write a Python script to train models for keyword spotting on the original mini speech command dataset. Use the train/validation/test splits and the label mapping provided in the *Portale*.

- Train three different model versions, each one meeting the following constraints, respectively:
  Version a): Accuracy > 92.0% and TFlite Size < 130 kB
  Version b): Accuracy > 91.5% and TFlite Size < 50 kB and Total Latency < 40 ms
  Version c): Accuracy > 91.0% and TFlite Size < 25 kB and Total Latency < 40 ms

  To measure Latency, run the script *kws_latency.py* provided in the *Portale*.

- Submit the TFLite models (named *GroupN_kws_a.tflite*, *GroupN_kws_b.tflite*, *GroupN_kws_c.tflite*), together with one single Python script to train and optimize them. If you have compressed the TFLite file with *zlib*, append .zlib to the filename (e.g., *GroupN_kws_a.tflite.zlib*).
  The script should take as input argument the model version:

  ```
  python HW2_ex2_GroupN.py --version <VERSION>
  ```

  where `N` is the group ID and `<VERSION>` is “a”, “b”, or “c”, and return the TFLite file.

- In the report, explain and motivate the methodology adopted to meet the constraints (discuss on pre-processing, model architecture, optimizations, hyper-parameters, etc.). Moreover, report the command used to measure the latency, specifying the values of the command-line input parameters (if different from the default values). E.g.:

  ```
  python kws_latency.py --rate 32000 --mfcc --coeff 15
  ```