Machine Learning for IoT

Homework 2

*****DUE DATE: 23 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 <ML4IOT21 GroupN> (N is the group ID). Attached with the e-mail the following files:

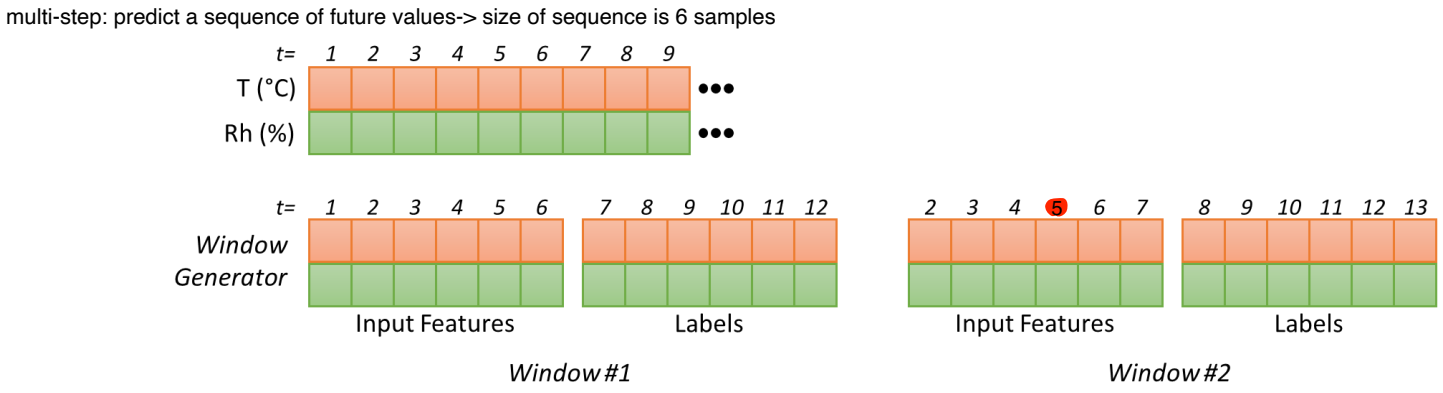
1. One single .py file for each exercise, titled <HW2_exM_GroupN.py>, where M is the exercise number and N is the group ID, containing the Python code. The code must use only the packages that get installed with *requirements.txt*.
2. One-page pdf report, titled <GroupN_Homework2.pdf>, organized in different 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. Set the number of output steps to 6. 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. [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, #Steps, #Features]. Use the models developed in Lab3 as starting point.

Results shouldn't be different by using TFLiteconverter.from_saved_model or TFLiteconverter.from_keras_model. The possible source of this error is the use of callbakcs to save to disk intermediate checkpoints. A common callback is the one that measures accuracy on validation set and stores on disks the model with best accuracy to the validation set, however the accuracy on the test set may be different. We must decide if we want to save the checkpoint with the best accuracy on the validation set or just the last version of the model. Maybe when I'm using frpm_keras_model, I'm using the model with weights obtained at the end of training, while if I use from_save_model, maybe I'm using the output of an intermediate checkpoint (with weights). Also file dimensions should be the same, follow the conversion process explained in ex5 Lab3.

to compute MAE we need to average acress batches and across different steps. Which is the order of averaging? See reduce_mean documentation, use this function. Which axis to do first. Axis argument can accept a list of dimension-> I need to average across batches and step. Therefore the dimensions are (0,1) (dimension of batches, dimension of steps). There isn't an order of operations because I can compute these averages with 1 single funciton, set the axis argument using a list containing 0 and 1. The input of the reduce_mean must be the difference betweent the prediction and the true label (compute absolute value)

- 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. average the error over 1 window the nubmer of features is still 2

- Train two different model versions, each one meeting the following constraints, respectively:
  Version a): T MAE < 0.5°C and Rh MAE < 1.8% and TFLite Size < 2 kB % is for humidity
  Version b): T MAE < 0.6 °C and Rh MAE < 1.9% and TFLite Size < 1.7 kB

  I can't feed more data, don't change the data

  N.B: 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.
  The script should take as input argument the model version:

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

  devo mettere tutte le prove nel codice o solo quella migliore?

  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.). I can change the models or start from lab 3

Exercise 2: Keyword Spotting (3 points) NO SILENCE CLASS

Testo
Testo
Testo

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

inference latency is only the model execution, no preprocessing
total latency is all

- Train three different model versions, each one meeting the following constraints, respectively:
  Version a): Accuracy > 90% and TFlite Size < 25 kB
  Version b): Accuracy > 90% and TFlite Size < 35 kB and Inference Latency < 1.5 ms
  Version c): Accuracy > 90% and TFlite Size < 45 kB and Total Latency < 40 ms

not equal to 90% but greater

  To measure Latency, run the script *kws_~~latency~~.py* provided in the *Portale*. inference

- 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.
  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.).

when I apply the rpuning, it starts aftern a numbe rof trining steps, therefore the pretraining is alerady embedded in the pruning procedeure of tensorflow. It may happen that applying the pruning after few steps, I could achieve a better result than applying the pruning after a lot of epochs. Try the application of pruning on models that have been initialized with random weights