Machine Learning for IOT: Homework I, Group 16

1st Francesco Capobianco
ID: s281307

2nd Pierluigi Compagnone
ID: s288301

3rd Carmine De Cristofaro
ID: s291129

I. EXERCISE I

In this exercise, given a csv file as input we are asked to build a TFRecord Dataset with the target of minimize its dimension according to the datatype we choose for each fields of the input file.
First of all we construct the datetime field from the original value of date and time and we store the values as POSIX timestamp. We decide to store the POSIX timestamp as ***tf.train.Int64List*** type because the ***tf.train.FloatList*** type would not be capable to represent, as different values, samples collecting within a few seconds. Regarding temperature and humidity values, we store them as different types depending on the application or not of a normalization phase. In the first case we store the values as ***FloatList*** type, since normalization returns them as float, whereas in the other case we handle them as ***Int64List*** type.
The dimension of the TFRecord Dataset, tested on the csv example input file given by the assignment, with and without normalization, are reported in the table I.

TABLE I
RESULTS: DIMENSIONS OF THE TFRECORD DATASET IN BYTES

With normalization	**Without normalization**
292	268

As we can deduce, applying normalization increases the dimension of the output file because, saving humidity and temperatures values as ***FloatList*** turned out to require more bytes with respect to ***Int64List***.
Assuming the normalization is required by training and inference pipelines, the best way to handle a TFRecord and the minimization of the storage requirements is to apply normalization after building the dataset. Indeed, considering the results of table I, the TFRecord Dataset is smaller without normalization and, by implication, is faster to be moved to and to be read in some other training and inference pipelines.

II. EXERCISE II

This exercise focuses on the implementation of a pre-processing pipeline applied on the audio files collected in *Yes_No* dataset. We are asked to compute the ***MFCC*** satisfying the latency constraints and trying to minimize the time in which the *raspberry* works in performance mode.

As first step we iteratively read the files from the dataset and for each one we obtain its Mel-frequency cepstral coefficients setting the parameters as requests by the assignment. The output of this step is denoted as ***MFCC_slow***. In order to get that result, we achieve these intermediate results:

1) spectrogram
2) mel-spectrogram
3) log-scaled mel-spectrogram

Afterwards, the objective is to compute the same features, optimizing the pre-processing pipeline in order to respect the constraints listed below.

- same shape between MFCC_fast and MFCC_slow
- execution time $< 18ms$
- SNR $> 10.40dB$,

where **MFCC_fast** is the output of the optimized pipeline and **SNR** is the signal-to-noise ratio.
Regarding the computation of the MFCCs_fast, the new pipelines is improved through of a resampling phase characterized by a downsample factor of 4. With respect to the previous pipeline, we use the same parameters for the STFT calculation, while for the MFCCs the paramereters are reported in table II.

TABLE II
PARAMETERS OF MFCC_SLOW AND MFCC_FAST

Parameters	**MFCC_slow**	**MFCC_fast**
# Mel bins	40	32
# MFCCs	10	10
lower frequency	20Hz	20Hz
upper frequency	4kHz	2kHz

The resampling is exploited because, although it requires additional computational time, thanks to it, the dimension of audio signal is decreasing and so we gain time in following calculations. In terms of number of mel bins, the reduction from 40 to 32 allows faster mel-spectrogram computation respecting a trade-off with the quality of the data. The upper frequency has been changed in order to be equal or less of the Nyquist frequency, moreover considering this frequency range the SNR is improved.
With the configuration of the MFCC_fast pipeline, as previously described, we obtain the following result:

execution time $\approx 17.7ms$ **SNR** $= 11.44$ dB