

Machine Learning for IoT

Homework 1 - Group 8

Rosi Gabriele
s291082

Fabio Tecco
s285569

Merlo Matteo
s287576

1 Exercise 1

The first Python script aims to create a *TFRecord* containing datetime, temperature and humidity values. The values are read from a csv file created previously with another script. In order to minimize the storage, the following data types have been used:

- **Timestamp**: has been converted to POSIX format and saved as an *Int64List*. Although the space occupied was slightly less when using float, this led to data loss and incorrect result;
- **Temperature** and **Humidity**: have been read as int32 and saved as *Int64List*, so the original data quality is maintained. We also experimented with *FloatList* type, but in the end we did not see an improvement in the dimensions of the output file.

If requested from the command line, the temperature (T) and humidity (H) are normalized using min-max normalization. The min and max value are referred to the sensor min-max operating range [1]. For the DHT-11 the operating temperature is 0-50°C, while the operating humidity is 20-90%RH.

2 Exercise 2

The second exercise required us to write a Python script in order to iteratively read audio samples and process the MFCCs of each of them. The audio dataset contains 1-second recordings of yes/no keywords sampled at 16kHz and Int16 resolution. In order to speed up and guarantee a stable execution on the Raspberry Pi, we had to manually switch the board to *performance* mode.

We used two different pipelines to obtain the MFCC of each audio file. After that, we compare them in terms of average execution time and the average SNR¹.

The first pipeline tested is the slowest one, and it is denoted as *MFCC_{slow}*. For STFT it uses *frame_length* = 16ms and *frame_step* = 8ms. Instead, for the MFCC uses *num_mel_bins* = 40, *lower_edge_hertz* = 20Hz and *upper_edge_hertz* = 4kHz.

In order to speed up the execution, we proposed an alternative pipeline (denoted as *MFCC_{fast}*) in which we decided to downsample all the audio files. Since the creation of the *mel weight matrix* takes more than 18ms alone, we instantiate it only once, thanks to the fact that all files have the same size. Also, we conducted some test to study the execution time of each function in the pipeline and the STFT (Short-time Fourier transform) takes almost 60% of the total time.

We also tested the speed of reading and resampling methods with TensorFlow or SciPy, and we prefer the first one because it is slightly faster than the second one.

We have observed that using a downsampling factor greater than 2, the execution time decreased but the average SNR worsened a lot. We also noticed that modifying the frequency interval do not affect significantly the execution time. Instead, the SNR is more correlated to the frequency interval: the larger it is, the higher the SNR will be. So we choose to not modify the lower frequency and the upper frequency.

Finally, we consider a downsampling factor of 2, a sampling rate of 8kHz and the range of frequency from 20Hz to 4kHz. After numerous trial, we found that the *num_mel_bins* should be reduced to 32 in order to comply to the given constraints.

¹The SNR compares the level of a desired signal to the level of background noise

In conclusion, our pipeline achieves an average execution time of $17.07ms$ and an average SNR of $19.38dB$, satisfying the given constraints.

References

- [1] <https://www.mouser.com/datasheet/2/758/DHT11-Technical-Data-Sheet-Translated-Version-1143054.pdf>.