Homework 1

GROUP 17

Juan Aragon (s291466) Hadi Nejabat (s246601) Rostislav Timuta (s280238)

I. Temperature and Humidity TFRecord

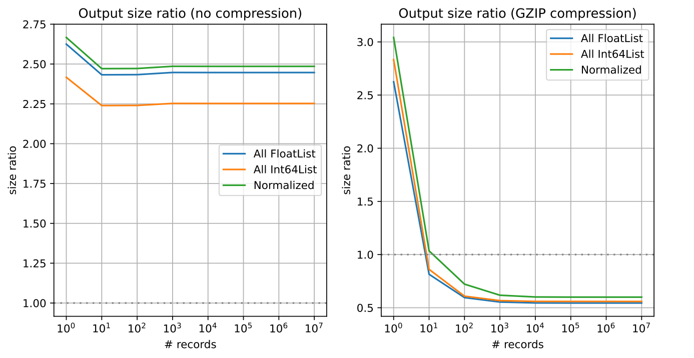

Fig. 1. File-size comparison expressed as the relative size to the original file.

The goal of this exercise consisted of encoding a dataset containing temperature and humidity records into a TFRecord-dataset, by choosing the correct data types for the different features available:

- A POSIX timestamp for which it was decided to use only the Int64List format.
- A temperature measurement for which it was decided to use Int64List when normalization was not required and FloatList otherwise.
- A humidity measurement which was encoded in the same format as in the previous field.

The motivation behind these choices are explained in Fig.1, where it is shown that when normalization is not required the storage requirements are minimized using Int64List, preserving the quality of each field since they are all integers. However, when the normalization is required the measurements values are encoded as FloatList in order to preserve the quality of the data stored. Regarding the timestamp field, although storing it as FloatList showed a lower memory size it was decided to keep using the Int64List, considering that the initial type is *integer* and it was desired to keep the original quality of the data.

Q: The best solution for handling the TFRecords when normalization is required is to exploit the Int64List format as much as possible for speeding up the transmission of data since we have seen it occupies less space in memory. For this reason we recommend to leave the normalization process at the very end. Additionaly we recommend to apply also compression techniques before sending data over the network since it reduces even more the size of the dataset.

II. Audio pre-processing optimization

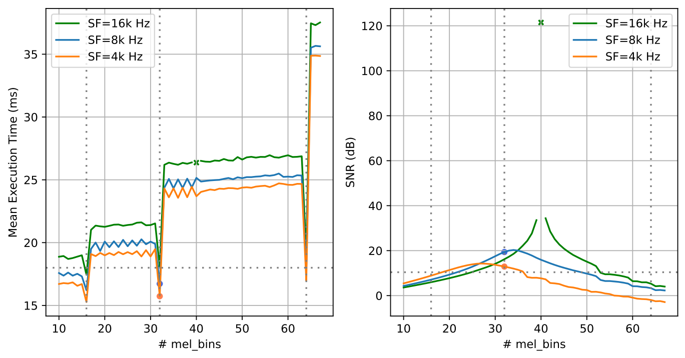

Fig. 2. Pipelines' performances for varying parameters.

The goal of this exercise was to accelerate the pre-processing pipeline for computing the *mfcc* of an audio file. Considering a standard implementation of it, using *Tensorflow*'s algorithms, the original pipeline was made with 4 stages: reading the audio file, computing the *stft*, calculating the mel's spectrum, and obtaining the mfcc. It was needed to choose the proper parameters for achieving on average 18ms of execution time for each transformation and an SNR above 10.4 dB with respect to its standard version.

The experiment consisted of optimizing 3 parameters: The sampling rate, which originally was 16kHz. The maximum frequency of the signal, which according to *Nyquist Shannon Theorem*, must be half the sampling frequency. And lastly, the number of mel_bins, being always greater than the number of desired coefficients (10), and initially set to 40.

For this purpose, all possible options were analyzed, obtaining the results shown in Fig.2. For the mean execution time, it was observed a reduction at the last stage of the pipeline by using values in the form 2^n, with n being an integer number. On the other hand, the SNR performed as a bell-curve, except for the case when the parameters were identical with the standard ones.

At the end of the experiments, two feasible configurations were obtained, satisfying both the constraints; as a result of the following parameter sets (SF=8kHz, MF=4kHz, mel_bins=32) and (SF=4kHz, MF=2kHz,mel_bins=32). Between these two solutions, it was considered the one with SF=8kHz as the optimal one, since it preserved a higher audio quality in terms of SNR. With this solution it was obtained an **ET** $\approx$ **16ms** (initially 29ms) and a **SNR = 15.06 dB**.