A music detector for hearing aids
Audio explorers problem

April 2022

1 Intro/perspective

Modern hearing aids utilize many different algorithms for improving hearing comprehension and reducing ambient noise. Many of these algorithms are tailored for the individual person and have different settings for many different sound environments. Many of these settings require the user to manually activate them using buttons on the device, voice commands, or the connected smartphone app. Constantly changing settings can be cumbersome and difficult if it requires the user to go through too many pages in the app, especially since hearing aid users are often elderly and not proficient in navigating the digital world.
It would thus be highly attractive to have the hearing aid automatically detect the nature of the sound environment and adjust the settings accordingly. One such sound environment could for example be listening to music.

2 Task

In this challenge we will ask you to design a music detector. One way of building a music detector is to implement a sound classifier able to classify sound samples as either music or other sounds. A sound classifier can e.g. be implemented as a neural network trained on labeled examples, i.e. small sound snippets either labeled as music or non-music. The design should take into consideration that a hearing aid due to its small size and small battery has limited computational power. We thus have a tradeoff between performance and computatonal load. One way of keeping the computational load at an acceptable level is to base the classifier on good input features. For example, instead of basing the classification directly on the hearing aid microphone signals, we base our classifier on features derived from the microphone signal. Hereby we save computational power as we do not have to implement a large neural network in order to learn similar features directly from the microphone signals. An example of such a feature is a time-varying magnitude spectrum of the sound (spectrogram). This feature may as well be reused for other purposes in a hearing aid.
In addition to a limited computational load, a hearing aid also has limited memory. It is thus desirable to do frame-by-frame classification rather than assuming that several seconds of data is available simultaneously.
Your task is to design a music detector using the machine learning scheme(s) of your choice. Your detector should output one of two labels: whether the spectrogram of a sound clip contains music or not. In order to solve the challenge we have provided a labeled dataset. You are free to split the data set into any kind of training and validation sets according to your chosen validation/testing scheme. Ideally, you should implement the detector in actual code, such that

you can make predictions on the unlabeled set. However, if you are not able to write the code for the detector, it is also fine if you just write a good report on how you would do, if you were to implement actual code.

3 Data

The data is based on two-second clips of sound recordings which either contain music or some other sounds which could be e.g., ambient sounds or speech. The audio file is converted into the frequency domain using a Short-time Fourier transform (STFT). Hereby we obtain a complete spectrogram with different frequency intensities at different times. The human perception of sound is not linear across frequency. We have much higher frequency resolution at low frequencies compared to higher frequencies. A commonly used way to present audio features across frequency is to use the mel-frequency scale. We therefore convert our spectrogram to the mel-scale by combining frequency components according to the triangular weights shown in the figure. The mel-scale has much higher frequency resolution at low frequencies compared to high frequencies, hereby resembling the human perception of sounds across frequency. Lastly, power spectrogram is converted to decibels. The data is found in the files `music_data` and `other_data` depending on whether the samples contain music or other sounds. Both `.npy` and `.mat` versions are available with the same data. The dimensions of each file is (10500, 30, 79) corresponding to 10500 samples with 79 time windows containing 30 frequency bands. We also provide ten examples of the types of sound recordings corresponding to music and other sounds. Lastly, we also supplied a file called `test_data` containing the unlabeled test data of shape (4000, 30, 79) which we will use to assess your implementation. The data is available via the wetransfer link.

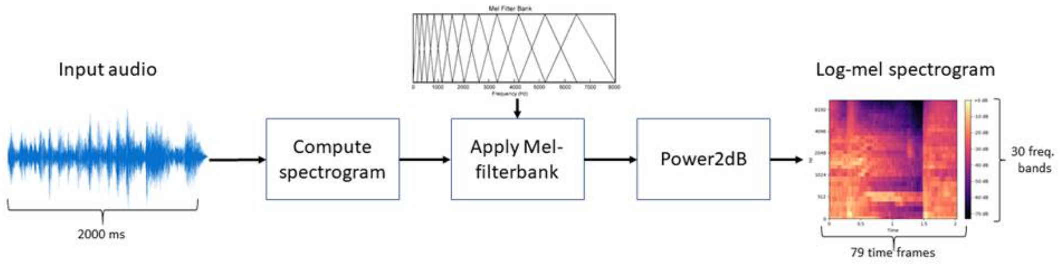

Figure 1: Data preparation overview

4 Formalities

You must write a short report where you document your process and thoughts on the project. The report must no longer than 5 pages excluding front page (and possible appendices), must contain each group member's name, and should describe e.g., how you approached the project, what worked and what did not, how you validated and tested your implementations, and how you selected the final model and estimated its accuracy. Try also to reflect on the feasibility of implementing your algorithm in a hearing aid with realistic constraints on power comsumption, memory and processing power.
Once you are satisfied with your music detector, you should make predictions for each sample in the `test_data` file and send them back to us for judging. The predictions should be in the form

of a text file with each row containing one number: “1” or “0”, where 1 corresponds to music being detected in the spectrogram, and 0 corresponds to no music being detected. The file must not contain any header line. It must thus have exactly 4000 lines with a single integer (1 or 0) on each line. If you solved the problem theoretically without writing actual code (i.e., by making the report outlining your proposed solution), you of course should not make these predictions.
You can work alone or in small groups of your own choosing. You will mainly be assessed on our overall impression of your report but also taking into account how good your predictions were.