Sound scene classifier for hearing aids

Audio Explorers challenge 2023

1 Intro/perspective

Modern hearing aids utilize many different algorithms for improving hearing comprehension and reducing ambient noise. In many situations, modern hearing aids are capable not only to attenuate unwanted noise but also sounds which could be of benefit for the hearing aid user. Often the amount of noise reduction, which is needed, will depend on the current sound environment. For example, babble background noise may prevent a hearing-impaired user from following a conversation. Another example is traffic noise. Even though traffic noise is annoying, it is important for the hearing-impaired listener to hear a vehicle approaching from behind in order navigate safely. It would thus be highly attractive to have the hearing aid automatically detect the nature of the sound environment and adjust the settings accordingly.

2 Task

In this challenge, we will ask you to design a sound environment classifier. A sound environment classifier can be implemented using supervised learning e.g. by using a neural network trained on labeled examples, i.e. short, labeled sound snippets.

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 computational 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 model 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 sound scene classifier using the machine learning scheme(s) of your choice.
Your classifier shall be able to correctly label unseen sound snippets. 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 classifier 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 classifier, it is also fine if you just write a good report on how you would do, if you were to implement actual code.

3 Dataset

The dataset is provided as labelled features from two-second clips of sound recordings. The labels are shown in the below table:

Integer label	Label	Description
0	Other	Sounds that do not belong to the below categories.
1	Music	Different types of music including recordings from different instruments.
2	Human voice	Sounds made from the human vocal system, such as speech, laughter and shouting
3	Engine sounds	Noises emitted from various engines, such as cars, power tools and jet
4	Alarm	Alarms sounds, such as sirens, alarm clocks, ringtones and doorbells

Besides the categories, which are regarded as important to hearing impaired, we also have a category called *other*, which contains everything that do not belong to the labelled categories. Notice that labelling is rarely perfect, and some of the sound clips sometimes could fit into more than one category. So do not expect that a sound classifier is able to label the data with 100% accuracy.

The provided audio is transformed 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. 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, the power spectrogram is converted to decibels. The feature extraction scheme is illustrated in the below figure. The data is stored in the file named *training.npy/training.mat* and the labels are found in the file called *training_labels.npy/training_labels.mat*. Both *.npy* and *.mat* versions are available with the same data. The dimensions of the dataset is (52890 x 32 x 96) corresponding to 52890 sound samples with 96 time frames each containing 32 frequency bands. Besides the data, we also provide some sound examples from each of the labelled categories.
Lastly, we supply a file called *test.npy/test.mat* containing the unlabeled test data of shape (5347, 96 x 101) which we will use to assess your implementation.
The data is available in the "Electrical Challenge" folder.

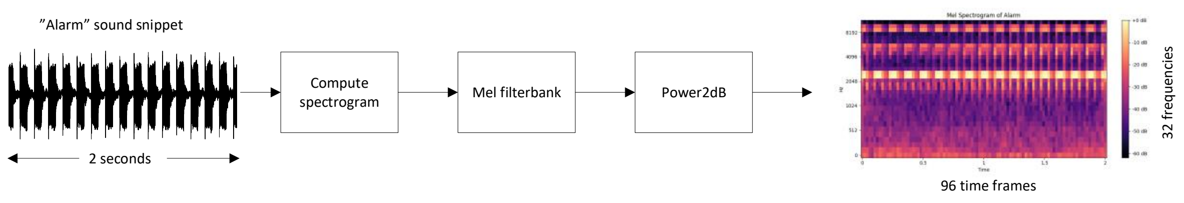

Figure 1: The labelled training dataset consists of 52890 two-second sound snippets stored as a mel-spectrogram of size 96 x 32. The unlabeled test dataset consists of 5347 mel-spectrograms.

4 Formalities

You must write a short report where you document your process and thoughts on the project. The report must be no longer than 5 pages excluding front page (and possible appendices), and it 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. You may discuss if you believe that some of the sound categories are more important to correctly detect than others, and you may e.g. assess your results (based on the training dataset) by plotting a confusion matrix.
Try also to reflect on the feasibility of implementing your algorithm in a hearing aid with realistic constraints on power consumption, memory and processing power. E.g. you may assume that the number of parameters in your model may not exceed 500000.

Once you are satisfied with your sound scene classifier, you should make predictions for each sample in the *test* 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 digit: "0", "1", "2", "3" or "4" where each digit corresponds to the label of the predicted sound scene being detected in the spectrogram. The file must not contain any header line. It must thus have exactly 5347 lines with a single integer corresponding to the predicted label 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 in a groups with 2 to 4 persons. You will mainly be assessed on our overall impression of your report but also taking into account how good your predictions were.