Project work report: Binary sound classification

Introduction

The goal of this project was to implement binary sound classification for vehicle sounds. Binary classification is a form of supervised learning, which means that the labels for the samples are known beforehand when training the model. When testing the model, the accuracy of the classification can be evaluated by comparing the true and predicted labels. Binary classification also means, that there are two possible classes, which are assigned to the samples.

There were multiple options for the model to use in this project work, such as SVM, K-nearest neighbor, and logistic regression. The options for classes included car, tram, bus, truck, and motorcycle, and the samples were 5-6 seconds long. The classes, data samples, and model selection are discussed in the following sections.

Data description

For this project work, I decided to use car and tram classes. I recorded some samples of an idle car by myself, but the other samples I took from others, are listed at the end of this report. Most of the samples were recorded by phone and all of them originate from Tampere. Trams might sound different based on the city they are in, so the tram samples should be in line with each other. Also, the samples are recorded during the fall and winter, which can affect the audio. I collected 35 samples per class because I wanted enough training data and have the same number of samples for both classes.

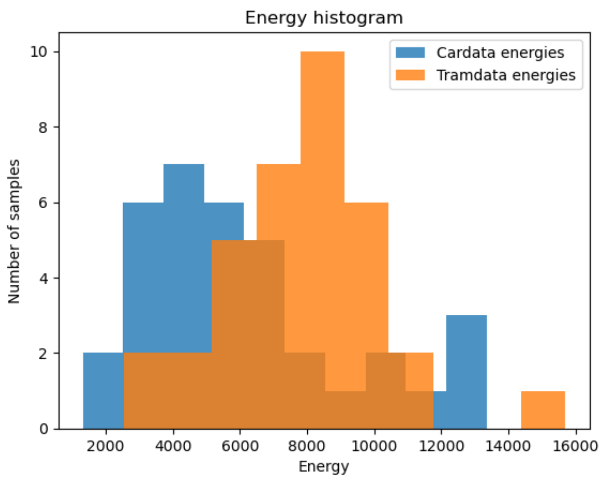
My initial guess was that a tram and a car would be easy to distinguish from each other because they have their distinct sounds. For humans, it’s easy to identify different vehicle sounds, so in the following feature extraction chapter, various features of audio signals are examined to train the model to classify the samples.

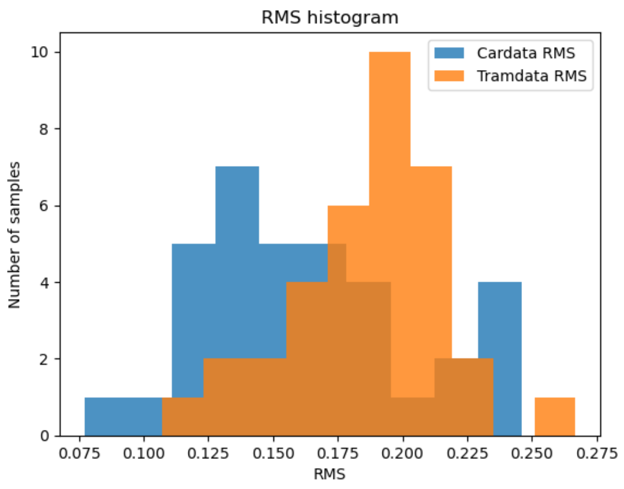
I also wondered if it is better to have samples of running and idle cars or only one of the former. On the other hand, a good model should have enough training data to make accurate classifications. A tram doesn’t make much noise when standing still, but it makes different noises depending on whether it’s slowing down, accelerating, or moving at a steady speed. I tried to collect samples from different situations, but they might not cover all possible cases.

Feature extraction

After collecting the data, the next part of the project was feature extraction. Before doing anything, the data needed some preprocessing, in this case, normalizing. Normalizing is helpful, for example, in cases where the sample values have different orders of magnitude, which could cause some samples to dominate the model and algorithms. In the literature, I came across various methods for audio normalization, such as peak and loudness normalization. I normalized the amplitudes of the signal because some samples were quieter and louder than others. The purpose of the normalization was to make the samples more comparable to each other.

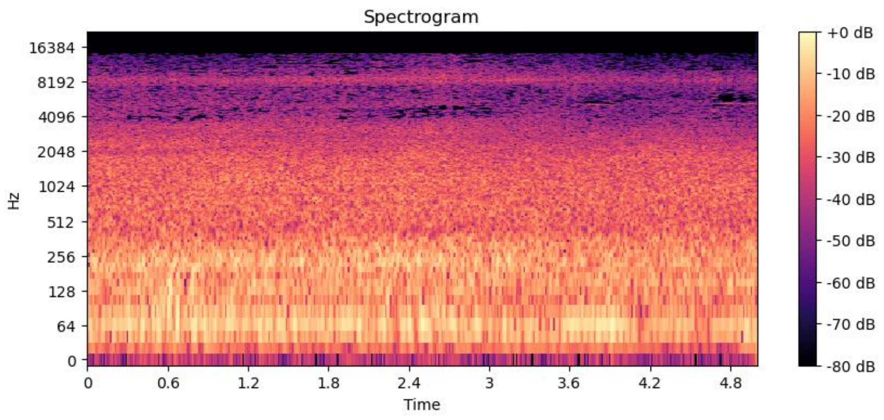
I started by calculating the energies and root mean square (RMS). The energy describes how the signal values distribute over time, and RMS computes the average value. I plotted histograms for both classes in the same plot, which shows the number of samples in each bin.

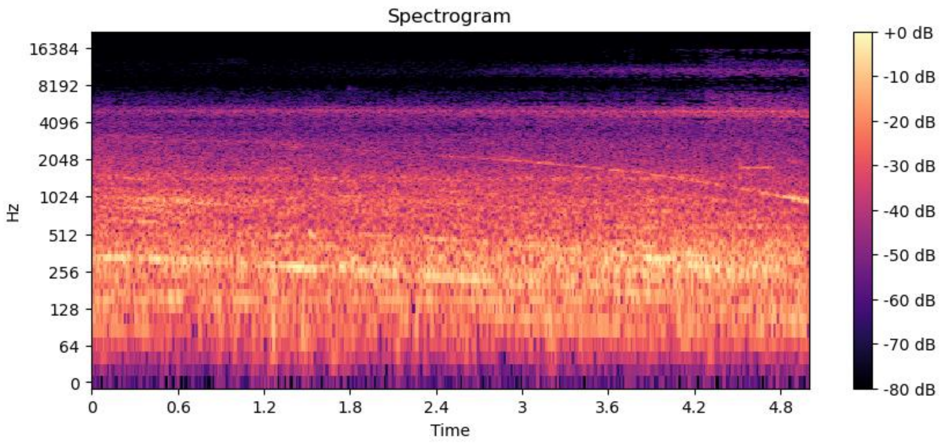




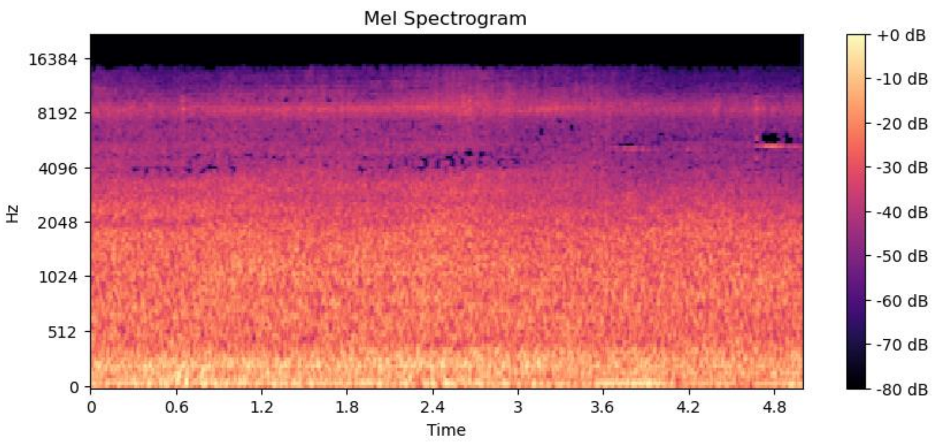
The classes overlap in the histograms. The RMS histogram seems a little better at separating the car and tram because trams have higher values on average than cars. The energy and RMS don’t tell about the content of the signal, so there might be better features to distinguish the classes from each other.

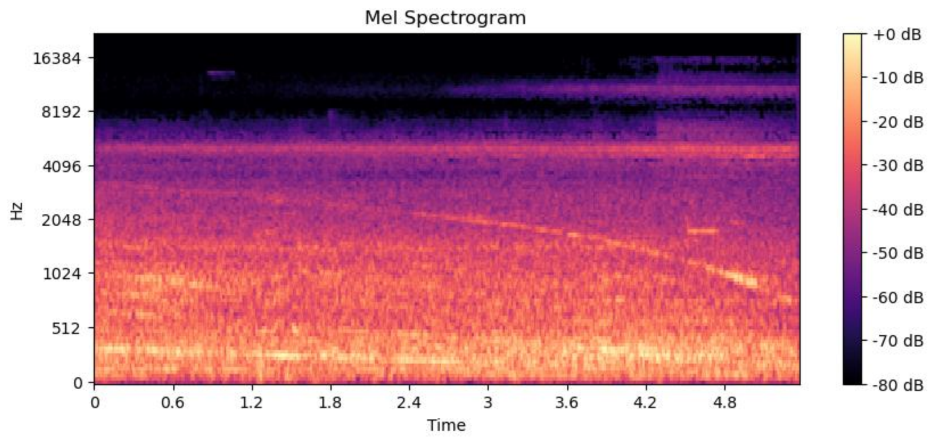
The spectrogram shows the amount of different frequencies in the signal over time. The first spectrogram below is from a car audio, and the second is from a tram. The following pictures are from the same samples, and in the same order.





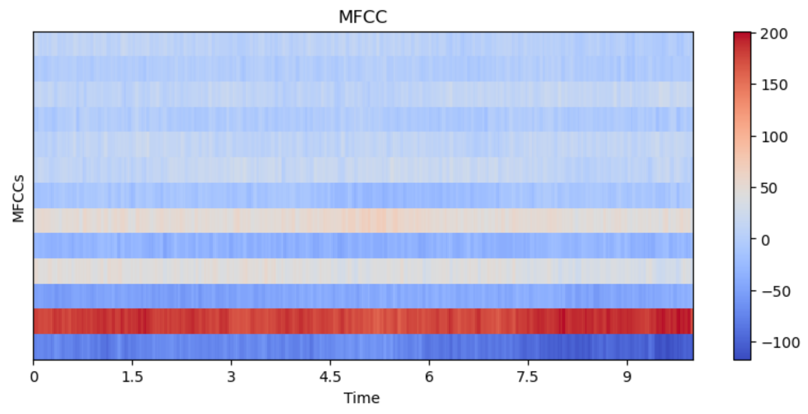
The car seems to have higher values on lower frequencies. The tram doesn't have large values on the lowest or the highest frequencies. Some differences can be seen in the spectrograms, but it is hard to say just from a few samples. I also calculated the Mel-spectrograms for the same samples, which are shown below.

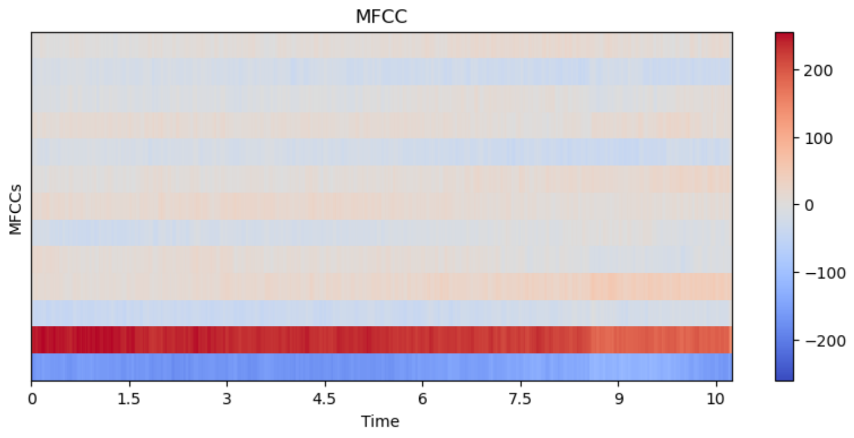




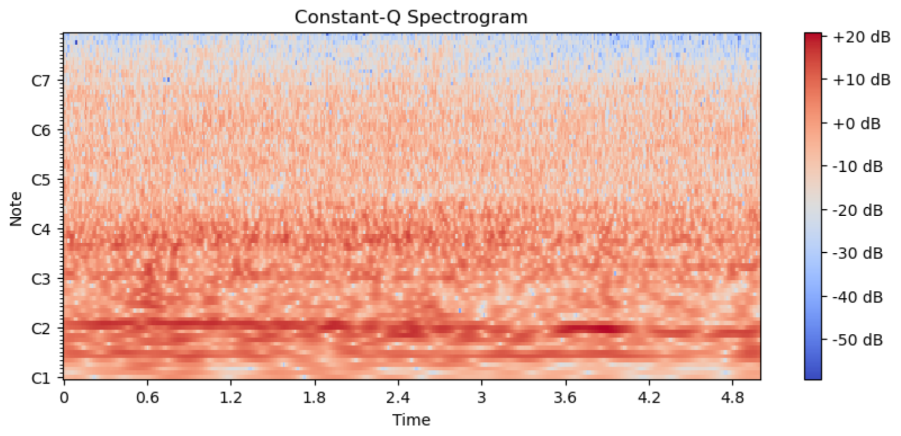
The Mel-spectrogram doesn't offer much additional information compared to the spectrograms, so I think only one of them should be used in the model. The Mel-spectrograms consider the human hearing characteristics so the Mel-spectrograms could be better for the classification.

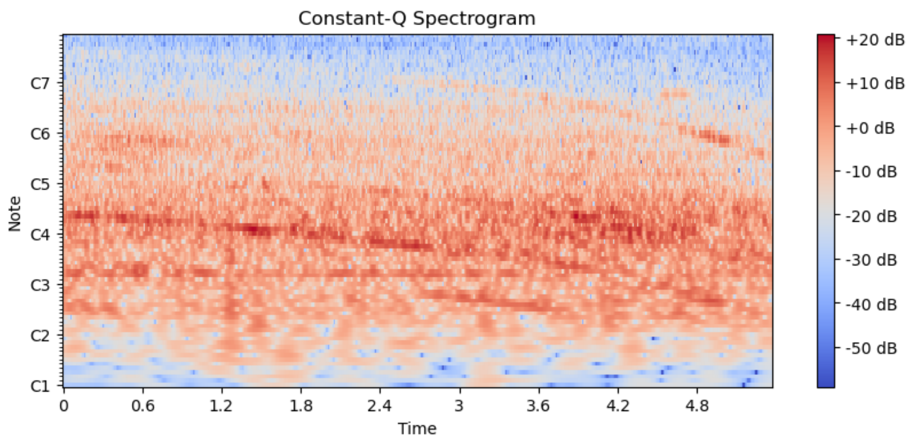
The Mel-frequency cepstral coefficients (MFCCs) are calculated by taking discrete cosine transform (DCT) from the log-mel spectrogram. MFCCs are a low-dimensional and uncorrelated way of representing features from the audio, and in the pictures below, there are 13 MFCCs. The first two MFCCs look similar in the pictures but the other values differ more. The MFCCs could be beneficial in the classification between cars and trams.





Finally, I calculated the constant-Q spectrograms for the audio signals. The CTQ-spectrogram shows the note values over time. From the samples below, the tram and car have their highest values in different notes. I think the CTQ-spectrogram could be useful in the classification, even though they are more used for music audio signals.





Based on the feature extraction results, I think Mel-spectrograms, MFCCs, and CQT-spectrograms are the most useful in the classification and could be used in training the model. I do not know if the features can be used in the model at once, so in the training part, I tried them separately.

Model selection

I chose KNN as the model that classifies the points based on the distances from other points. KNN can use different distance measurements, but Euclidean distance is a common choice. In KNN, it is usual to pick uneven k to avoid ties, and the class with more points close to the sample is assigned to it. Because the number of samples is not very large, I think k values 5 or 7 should be enough. The KNN model requires labels for the samples, so I used 0 for the car and 1 for the tram.

For the project, I had 35 samples of both classes, so I divided them as 25 for the training set, 5 for the validation set, and 5 for the test set. For the data division I used randomization, so that the set would contain an arbitrary mix of the samples. I decided to use MFCCs, Mel-spectrograms and Q-spectrograms to train the KNN model, but I ran into a problem because the arrays inside a list were different sizes. This was because the audio files were not the same length. The issue was solved by making a function that either adds padding or truncates the signal to make it the target length.

Results

The results with three different features are in the table. I repeated the process three times for each feature because I wanted to see if there was variation in the results. The training, validation, and test sets were divided randomly, so that could affect the results.

Feature	Validation accuracy	Validation precision	Validation recall	Test accuracy	Test precision	Test recall
MFCCs	90.00%	83.33%	100.00%	90.00%	83.33%	100.00%
MFCCs	90.00%	83.33%	100.00%	90.00%	83.33%	100.00%
MFCCs	90.00%	83.33%	100.00%	90.00%	83.33%	100.00%
Mel-spectrograms	90.00%	83.33%	100.00%	100.00%	100.00%	100.00%
Mel-spectrograms	90.00%	83.33%	100.00%	100.00%	100.00%	100.00%
Mel-spectrograms	90.00%	83.33%	100.00%	100.00%	100.00%	100.00%
CQT-spectrograms	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
CQT-spectrograms	100.00%	100.00%	100.00%	90.00%	83.33%	100.00%
CQT-spectrograms	100.00%	100.00%	100.00%	90.00%	83.33%	100.00%

The accuracy is the percentage of the correctly classified samples. The precision score describes the ratio between true positives over the total number of predicted positives. Lastly, the recall describes the number of true positives over all observations of the class.

The results didn’t vary with the other features than CQT-spectrograms. The results are very good with test accuracy 90-100%, test precision score 83.33-100%, and test recall score 100%. Surprising to me was that the model had the best accuracy with CQT-spectrograms and the lowest accuracy was with MFCCs. My guess initially was that MFCCs would be the best for the classification.

Conclusion

In this report, six features were examined for the car and tram samples. The conclusion for the feature extraction was that MFCCs, Mel-spectrograms, and CQT-spectrograms seemed most promising for the training of the model. All the previous features were applied separately to the KNN model, and the results were good.

In my preprocessing phase, I had to resample some of the samples, because most of them had a sampling rate of 44100, but a few had 48000. I also made all the samples 5 seconds long, so that the data would be easier to handle. After the previous phases, I normalized the audio signals.

Most of the data, around 70 percent, was used in the training, which plays a part in the results. Also, the quality of the audio samples and the preprocessing of the data probably had positive effect. The model worked well with the samples I had in this project, because I think they were quite similar with each other.

I think more data would be needed to make a more accurate binary classifier in different situations. Also, some other models might be better than KNN, for example neural network model. KNN might become computationally expensive if the amount of data is large, even though it worked well in this project.

List of downloaded sounds used in this project:

- mindyy1 (https://freesound.org/people/mindyy1/packs/39909/), licensed under CC0.
  - Auto1.m4a
  - Auto2.m4a
  - Auto3.m4a
- henmaar (https://freesound.org/people/henmaar/), licensed under CC0.
  - Uusi äänitys 8.wav
  - Uusi äänitys 12.wav
  - Uusi äänitys 15.wav
  - Uusi äänitys 24.wav
  - Uusi äänitys 22.wav
- inkaj (https://freesound.org/people/inkaj/packs/36886/), licensed under CC0.
  - Pack: Cars in Tampere
- publictransport (https://freesound.org/people/publictransport/packs/36726/), licensed under CC0.
  - Pack: trams in Tampere
- vegetals (https://freesound.org/people/vegetals/), licensed under CC0.
  - ratikka20.m4a
  - ratikka1.m4a
  - ratikka8.m4a
  - ratikka12.m4a
  - ratikka21.m4a
  - ratikka15.m4a
  - ratikka18.m4a

References

[1] https://haythamfayek.com/2016/04/21/speech-processing-for-machine-learning.html

[2] https://www.ibm.com/topics/supervised-learning