Introduction

The aim of this project is to build a machine learning model that can somewhat accurately classify two sounds of a motor vehicle. The two vehicle classes chosen for this project are car and tram. The focus is on understanding what kind of audio features can be extracted from these signals, and which of them provide beneficial information in the classification. The signal processing pipeline is implemented in Python.

Data Description

The testing dataset for this project is collected from Tampere, Hervanta, using the microphone on iPhone 11. The recordings took place in late November and there was a lot of snow on the ground. This can be heard from the car audios, but not from the tram audios, since the snow does not really pack on the rails. The weather was windy at times, so some noise can be heard in the background. The training and validation datasets also consist of manually recorded car and tram sounds in Tampere, but these have been recorded by other people on the course, at different times of the year. As mentioned, there are two classes, car and tram. There are 169 samples in the car class, and 168 samples in the tram class.

All the audio files have been converted into .wav format and normalized using the librosa library's **normalize** method. All the audio files are between five to six seconds long, although the amount of quality car or tram sound varies between the files. Some files have constant car sound, while others may just have a couple seconds of the car passing by. Same with tram, some files have constants sound of a tram leaving a station, while others have a shorter clip of a tram passing by at high speed. There is no speech present in the files, and other sounds such as people walking on snow have been cut out.

Feature Extraction

Features analyzed for this project include root mean square (RMS) energy, zero-crossing rate (ZCR), spectral centroid and mel-frequency cepstral coefficients (MFCCs). The RMS energy is calculated by taking the square root of the average of the squared signal values, which reflects the amplitude variations.

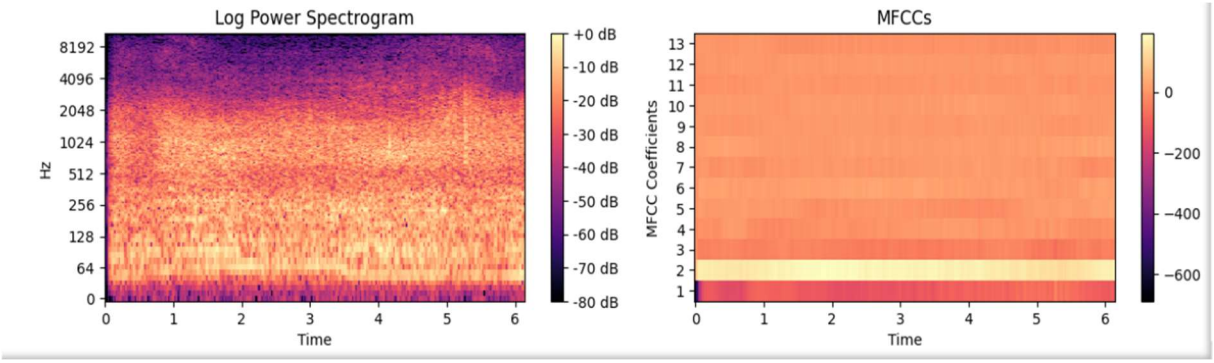
$$RMS = \sqrt{\frac{1}{N}\sum_{i=1}^{N} x_i^2}, \quad (1)$$

where N is the number of values. This feature was used and tested because the audios had different levels of energy, and even though the signals were normalized, there may have been subtle differences. For the actual calculations librosas **rms** method was used.

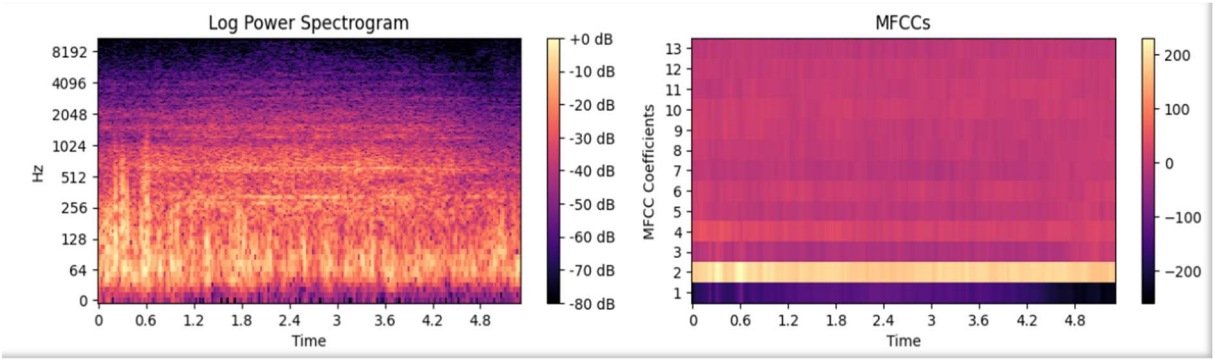
The ZCR is the rate at which the audio signal changes sign, crossing the zero value. It is computed by counting the number of times the signal crosses the zero in a given time frame. Even though ZCR is more used in speech recognition, it can also be used to differentiate between the more percussive sound of a tram and the smoother continuous sound of a car's engine. To calculate this librosas **zero_crossing_rate** method was used.

The spectral centroid of a signal's spectrum indicates where the center of mass of the spectrum is located. In this case, where most of the energy in the signal is concentrated across frequencies. The spectral centroid is a good predictor of the brightness of a sound. In this classification problem the car sounds sounded darker and heavier in most cases, so this feature was worth testing.

The MFCCs provide a feature representation that captures the most perceptually important aspects of sound. This makes them a valuable tool for distinguishing between different audio sources. There are several steps for extracting the MFCCs: taking the Fourier transform of a short, windowed part of a signal, then collecting the power spectrum and multiplying that with mel filterbank. The filters are spaced according to Mel scale, which is designed to match the frequency sensitivity of the human ear. The output is then compressed logarithmically, and finally a discrete cosine transform is taken of the signal. This final part produces the coefficients. The number of coefficients calculated is 13. Changing the number of coefficients didn't really make a difference in the end results, and more coefficients means more calculations. Again, to calculate this feature, librosas **mfcc** method was used.



Figures 1 and 2: The log power spectrogram and the MFCCs of a randomly chosen car audio clip



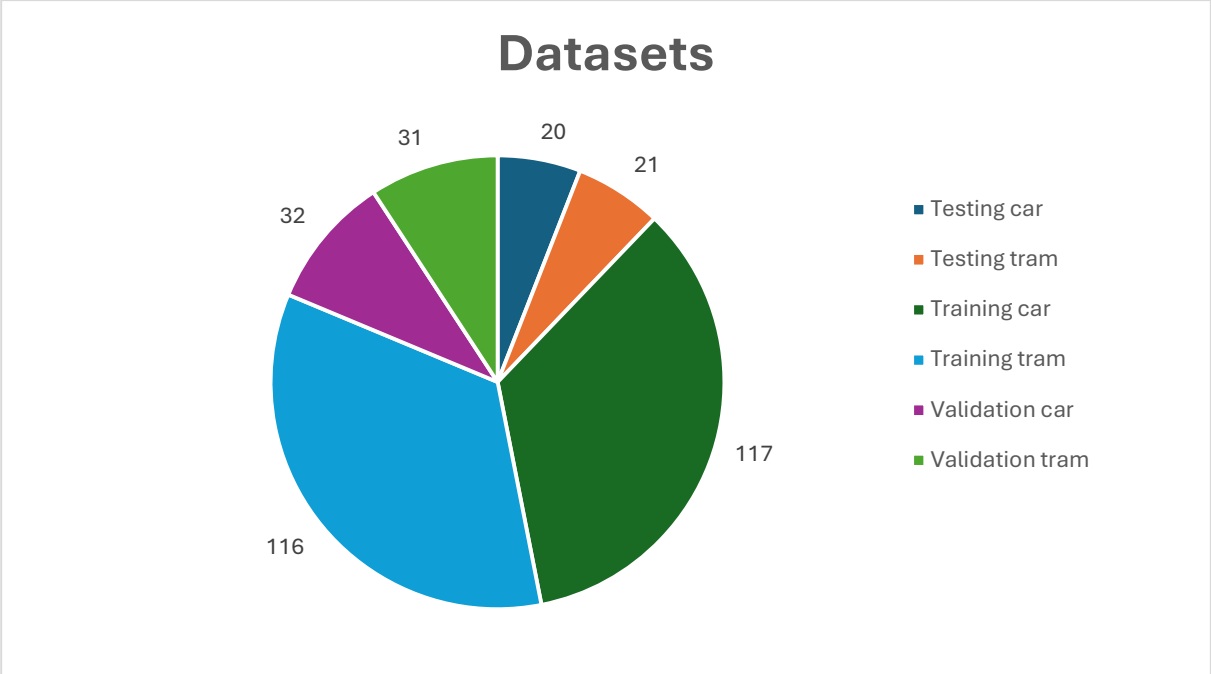
Figures 3 and 4: The log power spectrogram and the MFCCs of a randomly chosen tram audio clip

In figures 1 through 4 the difference between car and tram logarithmic power spectrum and MFCCs is noticeable. The car power spectrum has more powerful frequencies, especially above the 1024Hz mark. The trams power spectrum looks a bit tidier. In both MFCC figures the second coefficient stands out. Coefficients 3 through 13 have almost the same content, and the first coefficient does not differentiate that much either. This trend continued with other randomly chosen audio files. The files were chosen from the testing data, since the files were

of my own recording. This way the difference in microphone quality did not alter the results in figures. The feature chosen for the model is MFCCs, since it captures the subtle differences better than the other features. In testing the even if combined, the other features could not produce better results than the MFCCs.

Model Selection

The machine learning model that was chosen for this project is a support vector machine (SVM), with a linear kernel. This model was chosen because it is ideal for binary classification and works well with small datasets. The implementation of the model is from Scikit Learns sklearn.svm package. The model uses default parameters, except for the *kernel* which is set to linear and the *random_state* which is set to three. There was some experimenting with different values for the parameters like C, which is the regularization parameter, but there was no effect in the end results.



Graph 1: Pie chart of the datasets

As can be seen in graph 1, the data has been split quite evenly between the two classes. In the validation and training datasets the car class has one more sample than tram, but there are a good number of samples in whole so that should not have any meaningful effect in the results. The tram class on the other hand has one more sample in the testing dataset, but that also should not have effect in the results. The testing dataset consists completely of my own recordings. The training dataset has samples from twelve different people, so the quality does vary a lot. The validation dataset is completely from one person in the class. The quality is good and consistent over all the audio clips.

Results

Metric	Validation Set	Testing Set
Accuracy	98.41%	58.54%
Car Precision	97.00%	54.00%
Car Recall	100.00%	100.00%
Car F1-Score	98.00%	70.00%
Tram Precision	100.00%	100.00%
Tram Recall	97.00%	19.00%
Tram F1-Score	98.00%	32.00%

Table 1: Results of the model on validation data and testing data

On the overall accuracy results in table 1, the validation set performs extremely well, achieving 98% accuracy, which the F1-scores for car and tram classes reflect. On the test set, the accuracy drops to 58%, which reveals significant difficulties in classification. For the car class, the recall is 100%, so all car samples are correctly identified, however the precision is 54%, meaning the model misclassifies many tram samples as car. The precision for the tram class is 100%, meaning that when the model predicts tram, it is always correct, however the recall drops drastically to 19%, showing that the model misses most tram samples, misclassifying them as car. The F1-scores reflect these results, as cars F1-score is quite good at 70%, due to frequently predicting trams as cars. And the trams F1-score is terrible at 32%, because of the same reasoning.

Conclusion

The results for the validation dataset were a pleasant surprise, but the results for the testing dataset were disappointing. This can be the result of different audio recording devices used for the datasets, and the difference in other recording conditions. Because the classification is clearly more difficult for the tram class, the tram dataset may not represent the full diversity of tram sounds.

The issue may also lie in the data pipeline. There could be audio features that better capture the differences between car and tram sounds other than the ones used and tested in this project. The classification model could also be chosen differently, for example nearest neighbor and logistic regression are good binary classifiers. Neural networks can also be tested.

Choosing the right features and trying to truly understand what they represent was the hardest part of this project work. The recording part was really time costly, due to the amount of time it took trams to pass by, which was two every eight minutes. Also, some of the audio clips had to be discarded because of low quality, which added to the time it took to get the needed samples. Other than that, no significant difficulties were encountered for example when writing the code, due to the Python libraries doing most of the harder parts, like feature extraction.