Trung Nguyen - Cars and Trams audio classification

Introduction

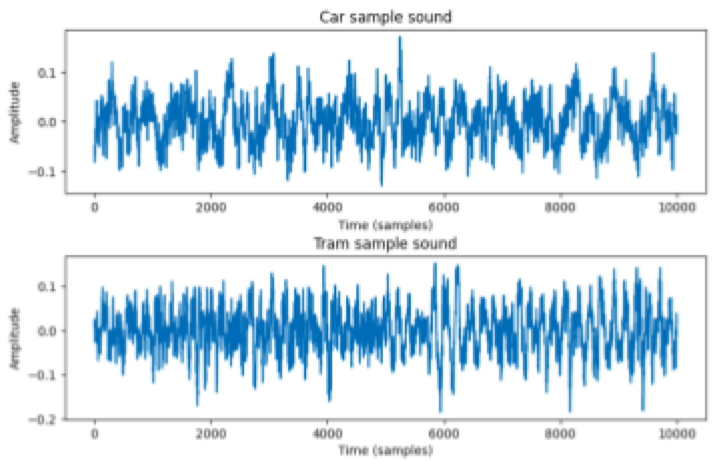
Sound is one of the most important parts in our daily life. We perceive the environment through what we see, smell, touch, taste and the main thing we are concerned about, to hear, which cannot be done without sound. Sound helps us to recognize and understand things without looking at it and this will come most essential for people that have low vision. It’s easy for humans to do this, but can machines do it? We call this task sound classification. Sound classification is one of the most widely used in many machine learning tasks. The potential is huge, from classifying man and woman voices to categorizing music into genres. In this project, I am going to perform a simple classification between sounds of two vehicle types, cars and electrical trams.

Data Description

The data contains two classes, cars and trams. For training, I collected the data from freesound.org website. Each class I will collect around 180 samples, in total I have 364 samples from both classes (195 for cars and 169 for trams). For testing, I collected the data by recording the sound using my own phone (Redmi 9C). Cars’ sounds are easy to collect because there are lots of cars in Hervanta. Everytime a car’s passing by, I start recording its sound when it’s 5 to 10 meters away and stop when it passes by 5 to 10 meters. Cars sound recorded in October so the road has lots of gravel, sometimes it's wet due to the rain, sometimes it has snow. So, cars' sounds can contain cracking sound and sound that comes from the wheel running through water puddles or snow. As for trams’ sounds, to conveniently record it, I go to Tampere's city center which has more passing trams. However, this leads to the fact that trams’ sounds are recorded in a closer environment compared to cars' sounds. Trams also run on tracks, which have a different condition from roads. For example, when the tram turns or brakes, it produces a cricket’s sound that has a high pitch. One more thing is that trams have different engines from cars, so the sound it produces is different. For testing, I recorded around 19 samples for cars and 22 samples for trams. For the baseline with randomly guesses, we know that we can predict with at least 50% of accuracy.

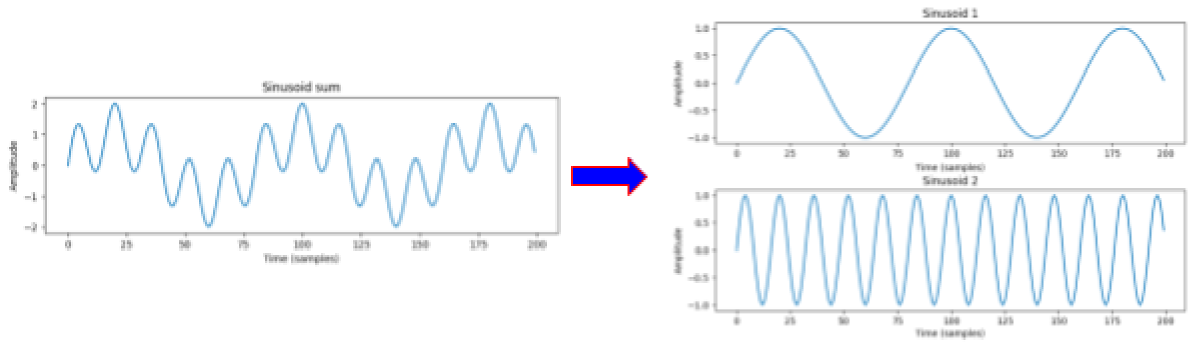
Feature Extraction

Now, we have quite enough data for each class, let's try to extract features out of those. Let's start with a simple thought. Sound is a continuous wave of signals that are transmitted through the environment in various quantities in a specific duration. For audio the quantity factor here is the air pressure strength. An audio sample has its own duration and a piece of sound or air pressure quantity which we call an amplitude in each timestamp, bigger amplitudes louder the sounds. Firstly, let's try to visualize this time series phenomenon for both cars and trams to see if we can catch anything to help classify them. In (1), we can see how audio is represented in time domain.



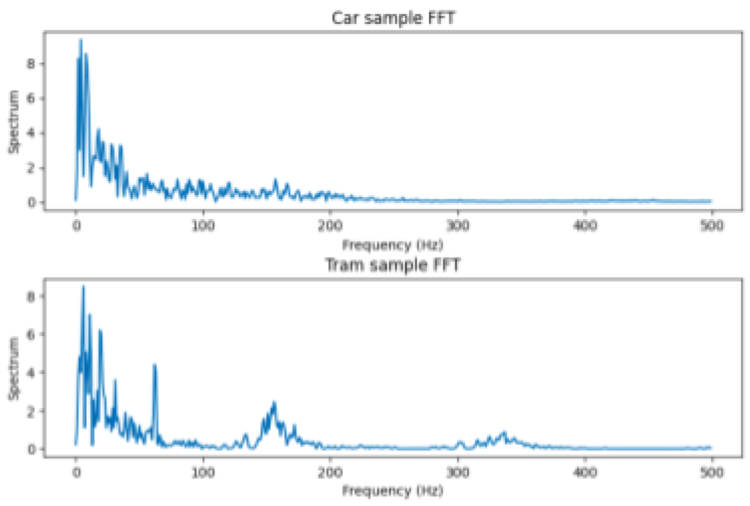
(1). Cars and Trams sound visualization in time domain

Now we have a digital representation of audio that we can somehow work with. However, this seems very frustrating for both cars and trams. They somehow look like a drawing piece that comes from an earth wake detector and nothing is related. In fact, each car or tram audio sample has its own pattern in time domain due to the environment and many other aspects. Actually, the amplitude we see in each timestamp is a combination or sum of many different sound pieces which we call sound frequencies, so we basically cannot extract much feature using time domain representation for audio. Luckily, we have a tool to disassemble that combination into components in order to analyze it, we call it Fourier Transform. Fourier transform is a formula that helps us decompose the sound piece which is the summation back into components. On the left side of (2) we can see sinusoid which is a summation of two sinusoids on the right side.



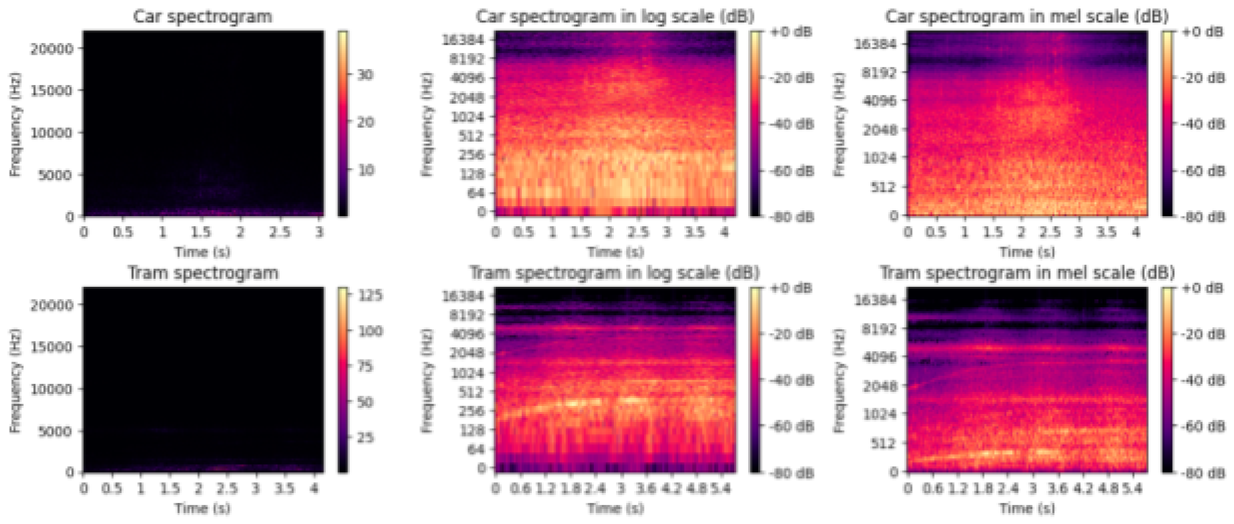
(2). Visualization of the Fourier transform effect on sound

In the real world, sound is a continuous wave of signals, but in the digital world we cannot have a continuous representation of it, we have to discretize it to be able to store and process it. This procedure is called the sampling process, in which, we take a piece out of the signal after a specific duration, the smaller duration leads to more accurate signal but more things to process. For this, we are going to use the frequency of 44100 Hz which is one of the most commonly used frequencies in nowadays recording devices. Therefore, the Fourier transform we use on these sparse samples is also called Discrete Fourier Transform (DFT). In practice, we use Fast Fourier Transform (FFT) which utilizes the divide and conquer strategy along with the recursion method to effectively compute the DFT of a sample. Furthermore, we cannot perform FFT on a whole audio at the same time because the audio's duration is arbitrary. For this, we use a time window with the side of 32 ms to process the audio partially and slide this window throughout the audio file. Last but not least, to equalize every audio file, we will take exactly 192 samples with 32 ms window size and 50% window size sliding. In the end, each sample will cover around 3 seconds in the original file. In (3), we can see frequencies that appear in one segment and its spectrum.



(3). Visualization of cars and trams sample FFT

This looks better, somehow we can acknowledge that trams contain some high frequency components compared to cars as we assumed. However, this is only the FFT for one window which can also be called a spectrum, it's time for us to use the sliding window. Each sliding step, in order not to cut-off the connection between parts, we only slide for 50% of the window size. However, if we keep the amplitude scale to visualize the spectrogram, it will be really hard to see, so changing it to log scale (dB) seems to be a good idea. Furthermore, log scale helps us to minimize the difference between large and small amplitudes of different frequencies so that more frequencies can be processed. One more thing to consider is how humans perceive the sound. In the beginning, we assume that trams will produce sound in higher pitch compared to cars and we want to use that assumption to classify cars and trams. In practice, the way we perceive sound pitch is not on a linear scale, for example, we can easily know the difference between 500 Hz and 1000 Hz but hard to tell between 10000 Hz and 10500 Hz. We will apply this phenomenon to our samples with a tool called Mel scale. In practice, Mel scale is a bit like log scale but it will stretch out the difference in lower frequencies and shrink in higher frequencies. For this Mel scale, we are going to use 192 bins as 192 features in each timestamp. The image (4) is the visualization of the audio spectrograms in amplitude, log scale (dB) and Mel scale (dB).



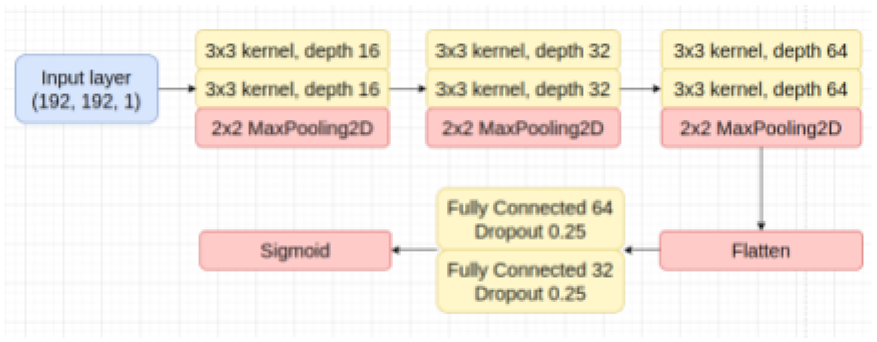
(4). Visualization of car and tram spectrogram in amplitude scale (left) log scale dB (mid) and mel scale dB (right)

In the end of this procedure, for each audio sample, we will have 192 feature vectors (time), each vector is 192 dimensions (Mel bins), which form a 2D matrix.

Classification Model

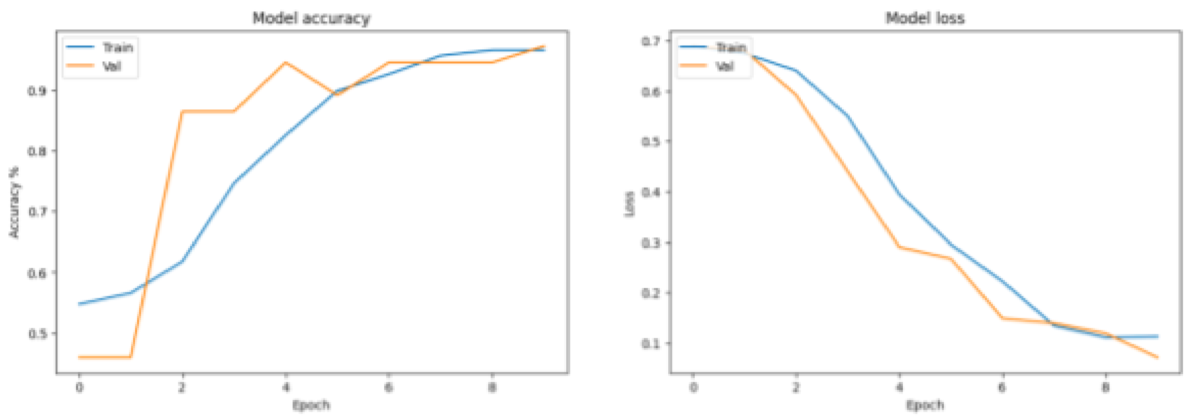
We know that for each audio we can get a 2D matrix with size of (192, 192) so after the feature extraction process, we will have a total 364 matrices for training and 41 matrices for testing. In training, we will also use 10% of training data as validation, data that are chosen to be validation are random for every training iteration. As for models, neural networks nowadays have been proving its reliability in classification tasks. We are now having a bunch of 2D matrices so it’s maybe a good idea to use the convolutional neural network type here. Let's have a check. We have 2D matrices, in row-wise we have Mel values for frequencies from low to high, in column-wise we have Mel values for a specific frequency in time series. With the usage of convolutional layers we can extract the feature both in temporal (time series) and spatial (neighborhood frequencies).

In this project, we are going to use a simpler version of VGG-net to be our classifier. The original VGG can be a bit overdo for our purpose and because we don't have that much computing power. In image (5), we can see the summary of our currently used neural network to classify our data.



(5). VGG-Net inspired CNN model

Before feeding our data into the network for training, let's standardize it first. We will scale our data into range (0, 1) and train with 10 epoches (iterations) and batch size of 16. After training we reach 96% accuracy for the training set. Let's try our model with the testing set. In (6) is the training process.



(7). Accuracy and Loss of training and validation set

The below table shows loss score and prediction accuracy of our model

	Loss	Accuracy (%)
Training	0.1126	96.64
Testing	0.3962	85.37

Conclusion

In our project, we know how sound is represented in digital, what its shape is in multiple domains and Mel is a good feature to extract from sound to classify it. We also learn that neural networks can be a good classifier for sound if we conduct it correctly. Also collecting data is a time consuming process. In conclusion, this is only a small set of data, if we want the model to work for a bigger set of data in real life, more training data will be required.