Audio project:

Audio binary classification using 1D convolution neural network (1D-CNN)

Authors: Khoa Pham Dinh (050359620), Uyen Phan (151372454)

I. INTRODUCTION

Audio classification, which entails identifying and categorizing different audio signals based on their sound properties, is a key task in audio processing. In this report, we present the results and observations from our audio binary classification project, focusing on the impact of various audio features and neural network models.

Our group collected audio data from trams and cars as they approached. The goal of our work is to identify the combinations of features that yield the highest accuracy in audio classification using neural network models. Specifically, we investigate the ensembling of audio features such as the Log Frequency Spectrogram, Mel Spectrogram, Constant Q-transform, and Mel Frequency Cepstral Coefficients (MFCC). By constructing time-series inputs for our neural network model from these features, we assess their efficacy using the collected datasets.

II. DATA DESCRIPTION

In this report, we utilized data collected by fellow students in the course. Initially, we gathered our data, which consisted of metro sounds in Helsinki. However, the number of sample was insufficient, and since no other users were collecting metro sounds, we had to shift our focus to different vehicles. Table 1 presents the basic information. All sample sounds were recorded in Tampere. There are two classes, "tram" and "car," each with an equal number of samples. In total, we used a collection of 170 samples, with 85 samples in each class:

User	Device	Dataset	Samples/Class
aaroan	OnePlus11	Train	14
akahukas	OnePlus 7 Pro	Train	30
Polystyreeni	Sony Xperia 5-II	Validation	20
velho_sunny	Nokia 6.1 plus	Test	21

Table 1. Basic information of collected sounds

The samples underwent a two-step normalization process. Initially, we normalized the signal amplitude to a range of [-1, 1] by dividing it by its maximum value. Subsequently, we used Librosa to adjust the sound data to a consistent duration of 5 seconds each, with a sampling rate of 220,500 Hz.

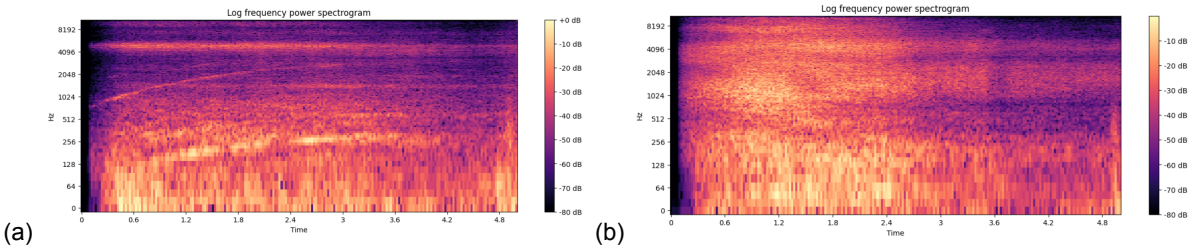
From what I have heard, the sound of cars does not follow any rhythm, while trams can produce rhythmic sounds due to their wheels passing over track sections. Consequently, the rhythm in tram sounds is more noticeable.

III. FEATURE EXTRACTION

The data given to a neural network should ideally be as clear and easy for the CNN to interpret as possible. To achieve this, some form of processing is necessary to enhance the relevant features in the raw audio data. Some features that have gained popularity for auditory recognition, and are used by our group, include:

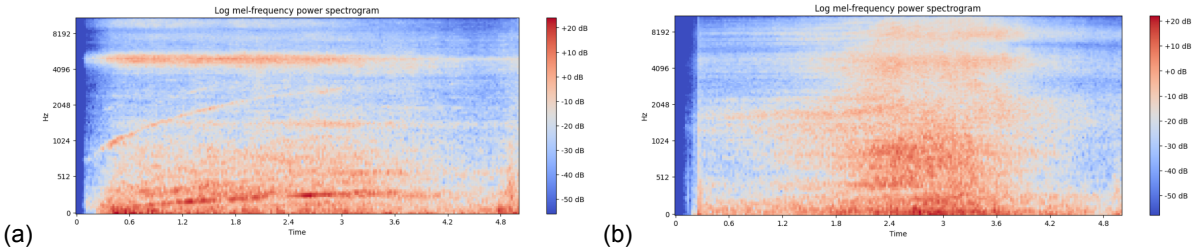
In our audio classification project, we use Log frequency, Mel spectrograms, and MFCCs extracted from Fast Fourier Transforms (FFT). Additionally, we incorporate insights from the Constant Q transform, energy, RMS, and zero-crossing rate for both tram and car classes. The Fourier transform provides a representation of sound in the frequency domain, offering insights into the distribution of frequencies within the audio signal.

Furthermore, the decision to extract features such as Log frequency (Figure 1), Mel spectrogram (Figure 2), and even Constant-Q spectrogram (Figure 3) during the approach phase of tram and car sounds is justified by the distinct acoustic characteristics inherent to each vehicle type. The spectrograms capture the frequency content over time, revealing specific patterns associated with tram and car approaches. For trams, the dominant low-frequency components in the spectrogram correlate with the substantial mechanical components and characteristic movement of trams. Conversely, cars exhibit a mix of mid to high-frequency components, reflecting the sharper and more varied sounds typical of smaller, faster-moving vehicles. This difference, therefore, provides a robust foundation for effective audio classification between trams and cars.



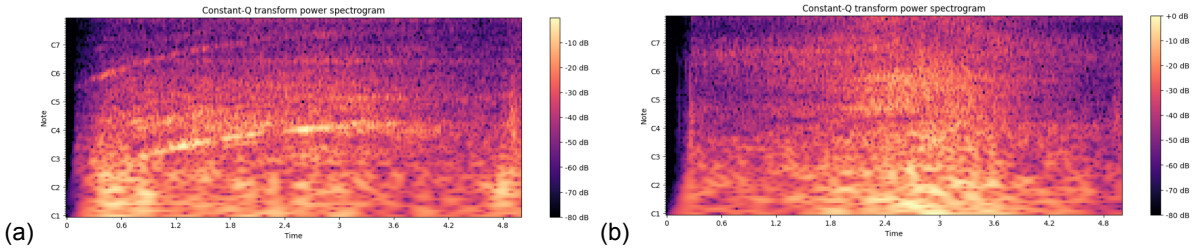
(a) (b)

Figure 1. Power Spectrogram extracted from sample tram (a) and car (b) audio files



(a) (b)

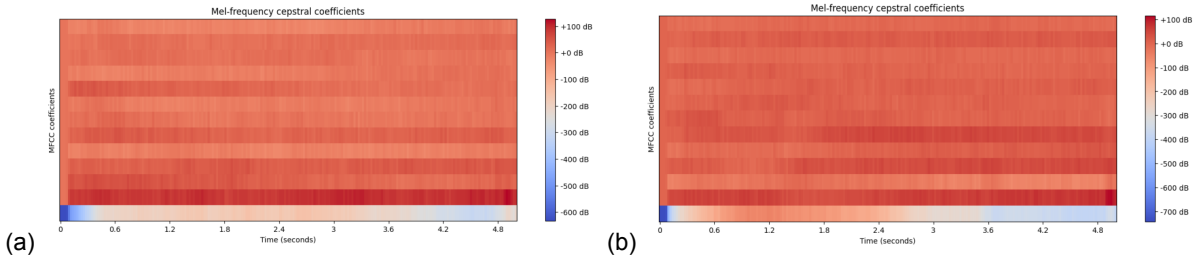
Figure 2. Mel Spectrogram extracted from sample tram (a) and car (b) audio files



(a) (b)

Figure 3. Constant-Q Transform Power spectrogram extracted from sample tram (a) and car (b) audio files

Another extracted characteristic is the Mel Frequency Cepstral Coefficients (MFCCs). Although the spectrograms effectively capture the distinctive features in our dataset, the MFCC representations (Figure 4) do not show noticeable differences. However, the strength of MFCCs often lies in their ability to capture subtle variations that might not be immediately apparent in the time or frequency domains. Therefore, by combining these diverse features, the neural network gains a robust capability to discriminate between tram and car sounds, leveraging both spectral and temporal information for accurate classification.



(a) (b)

Figure 4. MFCCs extracted from sample tram (a) and car (b) audio files

The next features that are taken into consideration are the Energy, Root Mean Square, and Zero Crossing Rate (ZRC). For Energy, car sounds concentrate around 1000 dB, while tram sounds span 100 dB to 6000 dB, showcasing a significant divergence in energy profiles.

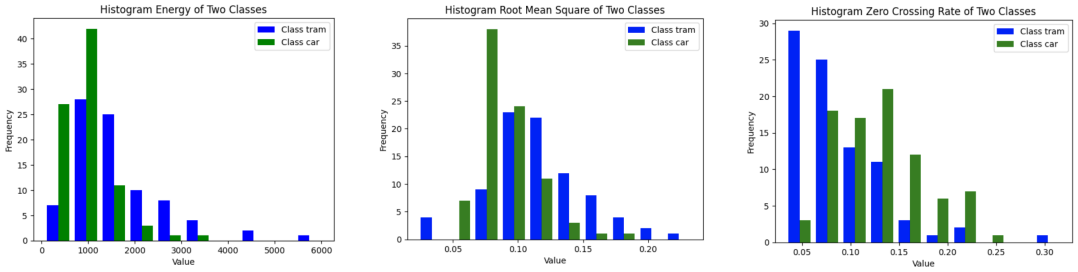


Figure 5. (from left to right) Energy, RMS, ZRC histograms of tram and car

In terms of Root Mean Square (RMS), car sounds maintain a consistent RMS of 0.07, whereas tram sounds exhibit variable values. Lastly, the Zero Crossing Rate (ZCR), which measures signal changes, contributes to distinguishing between correlated and uncorrelated features, offering substantial performance gains. Figure 5 displays distinctly separate patterns between the histograms of tram and car classes. While these three features exhibit notable differences, we have opted not to include them in our neural network, a decision that is further explained in subsequent sections.

IV. MODEL SELECTION

Among the seven features we extracted, four were presented as multivariable time series: spectrogram, mel-spectrogram, constant Q transform (CQT), and Mel-frequency cepstral coefficients (MFCCs). As mentioned in Part III, these four features provide a thorough and distinctive analysis of sound samples. Therefore, we aimed to utilize a model adept at handling multivariable time series data. Although recommended methods such as SVM and kNN have garnered significant interest, they are not optimized for time series data. These methods are more suitable for tabular data, which typically involves structured data in rows (vectors) of independent values. Consequently, we utilized 1D convolution as the basis of our approach and developed a simple 1D Convolutional Neural Network for training on the dataset. Figure 5 illustrates our model, which consists of two 1D Convolution layers followed by one Fully Connected layer (Gemm) at the end. This structure is interspersed with activation layers and includes Max Pooling 1D operations between the layers. The output is a single probability value.

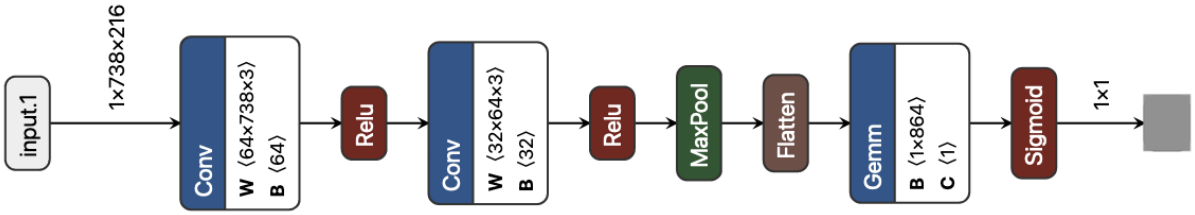


Figure 6. Model diagram

The data is divided into training, validation, and test datasets, with a disjoint set of users (sources) for each. The distribution is detailed in Table 1. To prepare an input for the model from a sound sample, we initially extract various features from the sound. These include spectrogram, mel-spectrogram, CQT, and MFCCs, which have dimensions of (513, 216), (128, 216), (84, 216), and (13, 216), respectively. The second dimension represents the temporal aspect of the data, while the first dimension corresponds to the feature-specific spectral or cepstral characteristics. Subsequently, we concatenate these features along the first dimension, resulting in a multivariable time series with dimensions (Spatial, Temporal) = (738, 216). This effectively merges the spectral or cepstral features with the temporal evolution.

Given that energy, root mean square (RMS), and zero-crossing rate (ZCR) have distinct data types, incorporating them would require a sophisticated implementation of a multi-head neural network. However, this complexity is deemed unnecessary, as demonstrated in Part V, where it is shown that the four aforementioned features alone yield satisfactory results in the test set.

V. RESULTS

The evaluation metrics used to assess the performance of our model include: firstly, accuracy, defined as the ratio of correctly predicted observations to the total observations; secondly, precision, representing the ratio of correctly predicted positive observations to the total predicted positive observations; thirdly, recall, which is the ratio of correctly predicted positive observations to all observations in that class; and finally, the F1 Score, which is the harmonic mean of precision and recall, providing a balance between them.
The performance of our 1D CNN model was evaluated on three datasets: the training, validation, and test sets with accuracy as main metric. The training protocol is as follows:

1) Train model on train dataset.
2) Make inferences on the train and validation dataset. We did not touch the test dataset at this step to avoid bias in fine-tuning.
3) Based on the evaluation of the validation set, fine-tune the model.
4) Repeat steps 1 to 3 until achieve a satisfying validation result.
5) Make inferences on the test dataset.

We used the number of epochs = 100, learning rate = 0.0001, Adam optimizer, and Binary Cross Entropy loss for binary classification.
To enhance the model generalization, two key modifications were made during our fine-tuning process. First, we adjust the convolutional layers, whose initial configurations were:
self.conv1 = nn.Conv1d(in_channels=738, out_channels=128, kernel_size=3, stride=2)
self.conv2 = nn.Conv1d(in_channels=128, out_channels=32, kernel_size=3, stride=2)
After fine-tuning, these values were modified to:
self.conv1 = nn.Conv1d(in_channels=738, out_channels=64, kernel_size=3, stride=2)
self.conv2 = nn.Conv1d(in_channels=64, out_channels=32, kernel_size=3, stride=2)
This change was designed to minimize overfitting in the model, since smaller models generally perform better in terms of generalization, especially when dealing with the small dataset utilized in this project.
The second adjustment is the learning rate, which varied between 0.001 and 0.00001.
The optimal learning rate was identified as 0.0001, as it produced the best results in terms of accuracy, precision, recall, and F1 score.
Eventually, after hyperparameter tuning, on the test dataset, the model showed strong generalization, achieving an accuracy of 97.6%, a precision of 95.5%, a recall of 100%, and an F1 score of 97.7%. These results underscore the importance of hyperparameter tuning and architecture adjustments to improve the neural network's overall performance.

	Accuracy (%)	Precision (%)	Recall (%)	F1 Score (%)
Training set	100	100	100	100
Validation set	55	52.6	100	69
Test set	97.6	95.5	100	97.7

Table 2.Inference on the train, valid, test dataset after 100 epochs.

Overall, the model demonstrated exceptional performance on the training dataset, achieving perfect accuracy, precision, recall, and F1 score, all indicating a flawless fit to the training data. However, the model's performance on the validation dataset exhibited a noticeable drop, with an accuracy of 55%, a precision of 52.6%, a recall of 100%, and an F1 score of 69%. This suggests that the model might be overfitting to the training data, as it does not generalize as well to unseen samples. Therefore, further investigations were made to fine-tune the model architecture as analyzed above.

Originally, at step (1) of our training protocol, we save the parameters of the best epochs based on the accuracy of the validation set during training and reload it to the model during inference. We ran the training multiple times (with the same settings) and always achieved accuracy higher than 50% on every set, indicating the predicting power of our model. In most cases, we observed high accuracy on the train set, 55%-67% on the validation set, and over 85% on the test set. In some rare cases, we have achieved as high as 85% accuracy, 100% precision, 70% recall, and 82.4% on the validation set. However, as the performance on the validation set increased, the results on the train and test set were hindered, as shown in Table 3. Because of this contrary, we removed the implementation to save parameters and load the final trained model at epoch 100, which results in Table 2 mentioned earlier.

	Accuracy (%)	Precision (%)	Recall (%)	F1 Score (%)
Training set	78.4	90.3	63.6	74.7
Validation set	85	100	70	82.4
Test set	61.9	100	23.8	38.5

Table 3. Inference on train, valid, test dataset after 100 epochs when best weights are saved.

The contradiction between validation and test sets suggests some fundamental differences between the data in the validation set compared to test and train sets. Further investigation to see whether the problem comes from recording devices or methods of sound collection is needed.

VI. CONCLUSION

In this report, with our 1D CNN model, we successfully tackled the task of audio classification, specifically differentiating between tram and car sounds based on their approaching and departing audio signals. The outcomes of our audio classification project offer valuable insights into the model's behavior and areas for improvement. Our primary goal was to achieve optimal generalization and enhance performance on the dataset.

The chosen feature set, including the Log Frequency Spectrogram, Mel Spectrogram, CQT, and MFCC, provides a solid foundation. However, we believe that experimenting with various feature ensemble combinations would allow for a comparative analysis. By addressing the identified improvements, the audio classification system can become more accurate and reliable in recognizing different sound categories.

We encountered two primary issues. The first is the unverified quality of the data. Since the training, validation, and test sets originated from different user groups, their sound recording techniques were not uniform. This discrepancy could lead to challenges in part V, where the recording methods of users in the training set might not be suitable for effective classification in the other sets. Secondly, we observed numerical inconsistencies when the training was executed under identical settings, necessitating multiple runs to verify the results. Both issues could be addressed by adding more training data and conducting training over more epochs.