COMP.SGN.120-2022-2023-1

Introduction to Audio Processing

PROJECT REPORT

CLASSIFICATION OF AUDIO SAMPLES BASED ON THE FEATURE BEING EXTRACTED

Project Members –

Meghana Chegatagere Narasimhareddy - 151226461

Erfan Momeni Yazdi - 151265354

Table of contents

1. Collection of data

Data collection was done using our own mobile phones.

Description of dataset collection-

Collected Region - Hervanta and city center region of Hervanta, Tampere.

Collected classes – Tram (25 samples) and car (25 samples)

Car collection - Erfan

Tram collection – Meghana

2. Conversion of the data samples and uploading on freesound website

The tram data was directly recorded in .wav format and the car data were converted into .wav format and uploaded in the freesound.org website with each sample being approx. time duration of 5 seconds.

For the training of the model, we have collected the tram and car audio samples from freesound.org, which were uploaded by fellow peers, and it was around 446(total) in which 217 are tram data 229 are car data.

Freesound.org links for datasets –

Car data link – https://freesound.org/people/erfanmo/packs/37114/

Tram data link – https://freesound.org/people/pikachu_bear/packs/37085/

3. Feature extraction from the dataset

There were several factors to consider when deciding on the best feature extraction method for audio classification. We have considered few and tried to analyze what fits into the model to reach high accuracy level.

Among the features that we have tried to extract, we were inclined towards STFT and MFCC features as these will define the frequency levels more accurately at lower level as well.

STFT (Short-time Fourier Transform) allows for the analysis of the frequency content of a signal in the time domain, which can be useful for analyzing the spectral characteristics of an audio signal. Additionally, STFT is a time-frequency representation, which means that it can capture both time and frequency information in a single representation.

Mel-frequency cepstral coefficients (MFCCs) are common method for feature extraction in audio signal processing. One reason for their popularity is that they are designed to mimic the way that the human auditory system processes sound. These can

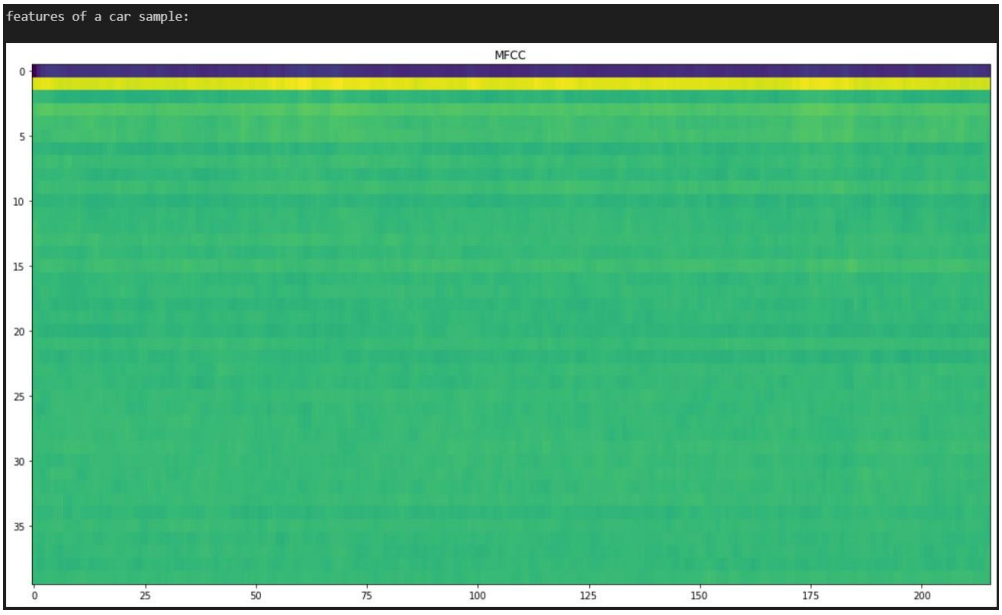
capture the details the lower range frequencies. MFCCs are derived from the log-magnitude spectrum of a signal, which has been filtered using a bank of Mel-frequency filters.

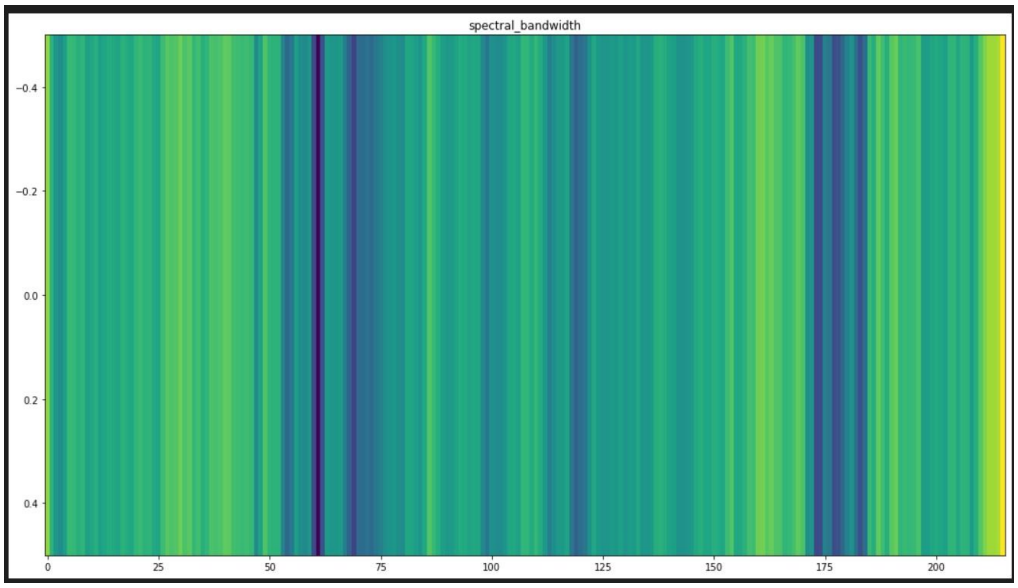
Additionally, the cepstral coefficients in MFCCs are derived from the log-magnitude spectrum, which allows for a more compact representation of the spectral information in a signal. In this regard, the tram and the car audio samples are always receptive with human hearing system and human being must be able to distinguish the sound of a car and tram in real life.

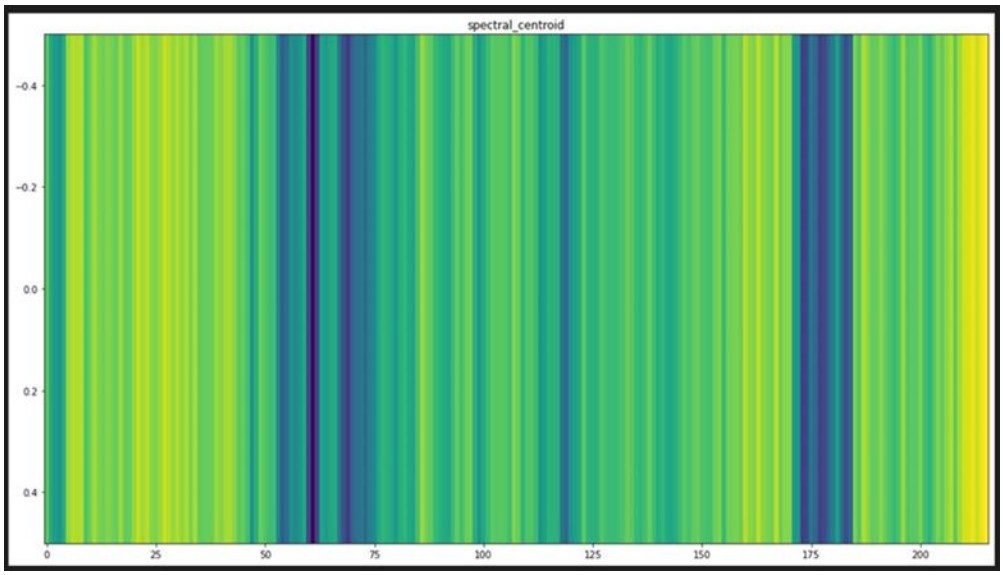
At the end, after model defining and fitting, we tried to run with STFT and MFCC feature individually and the MFCC feature gave better results to train and test the model. So, we decided to go with MFCC feature.

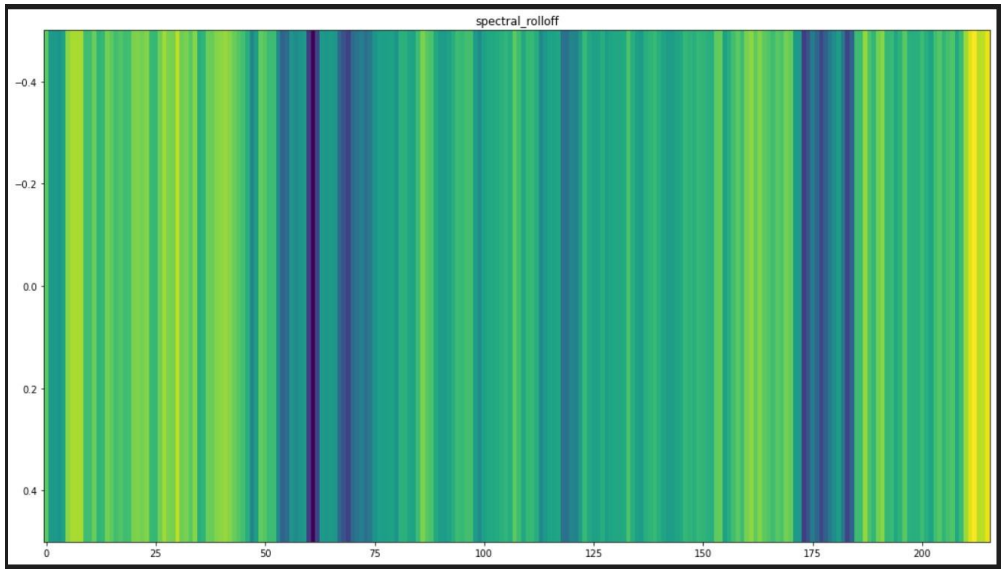
Below are the snapshots of the features being extracted and plotted-

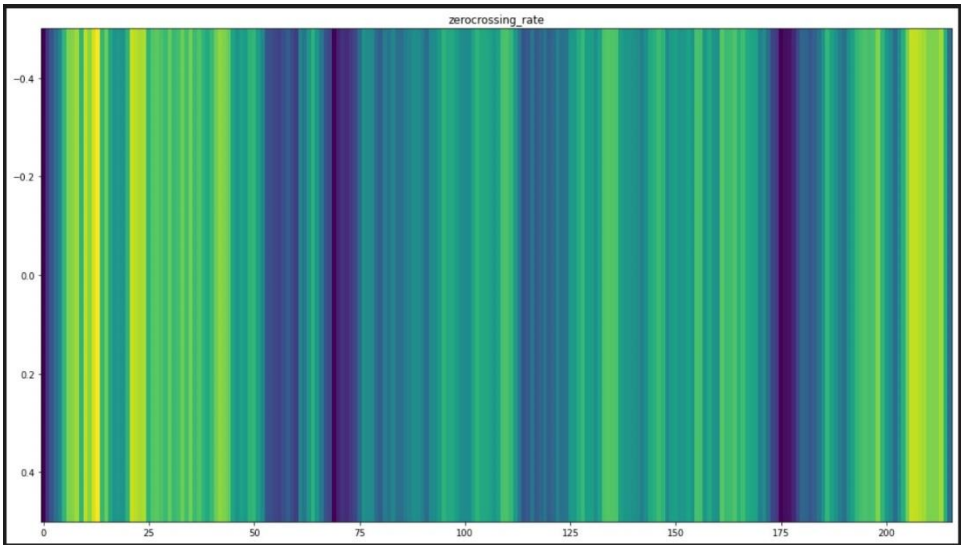
Features plots of CAR-

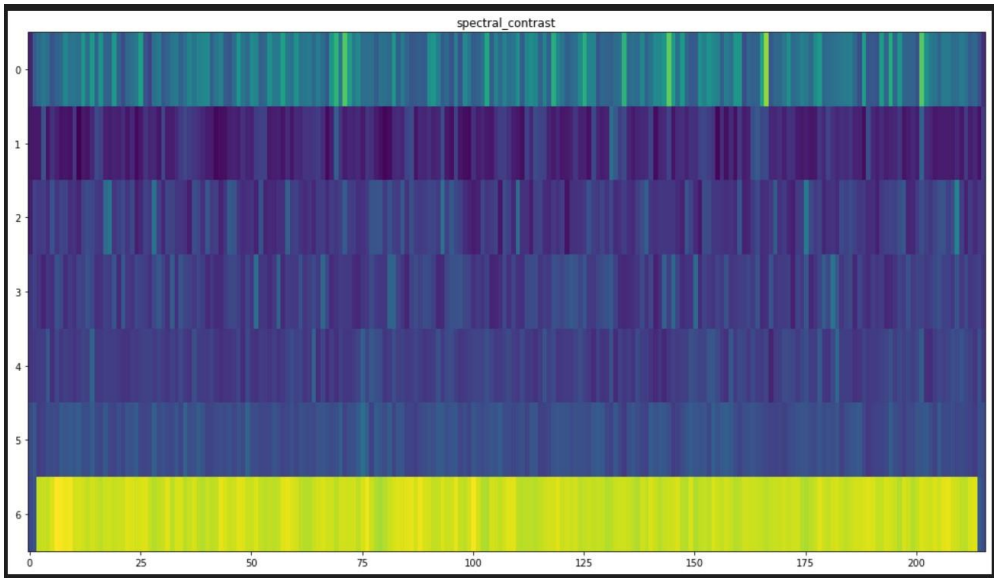


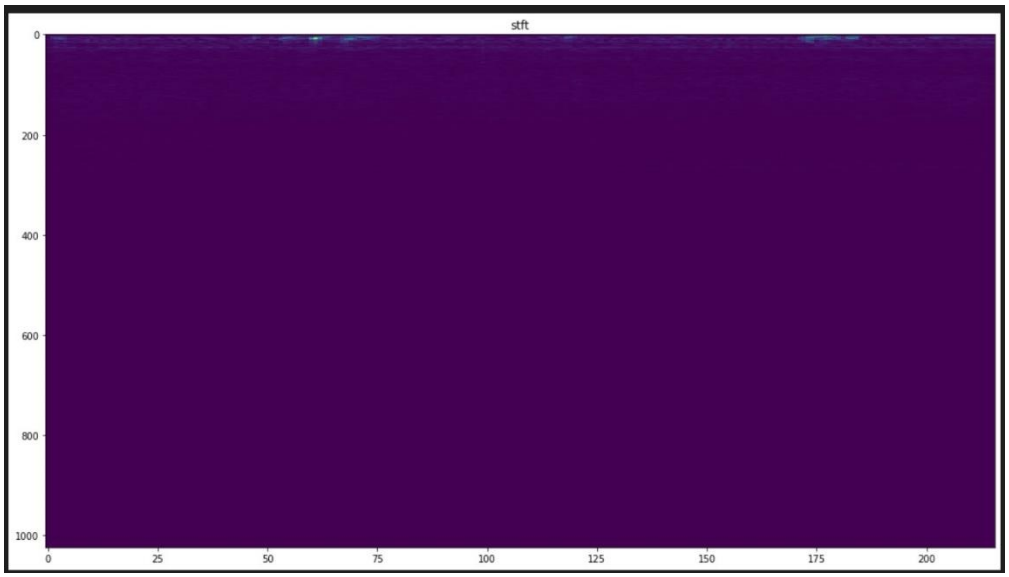

spectral_bandwidth
−0.4
−0.2
0.0
0.2
0.4
0
25
50
75
100
125
150
175
200


spectral_centroid
−0.4
−0.2
0.0
0.2
0.4
0
25
50
75
100
125
150
175
200

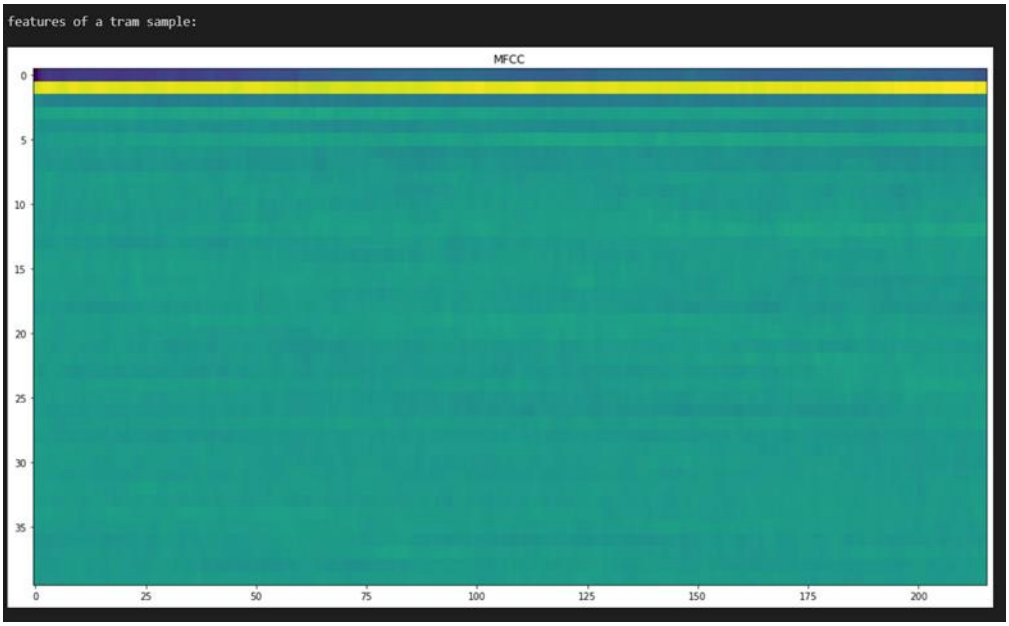

spectral_rolloff
−0.4
−0.2
0.0
0.2
0.4
0
25
50
75
100
125
150
175
200

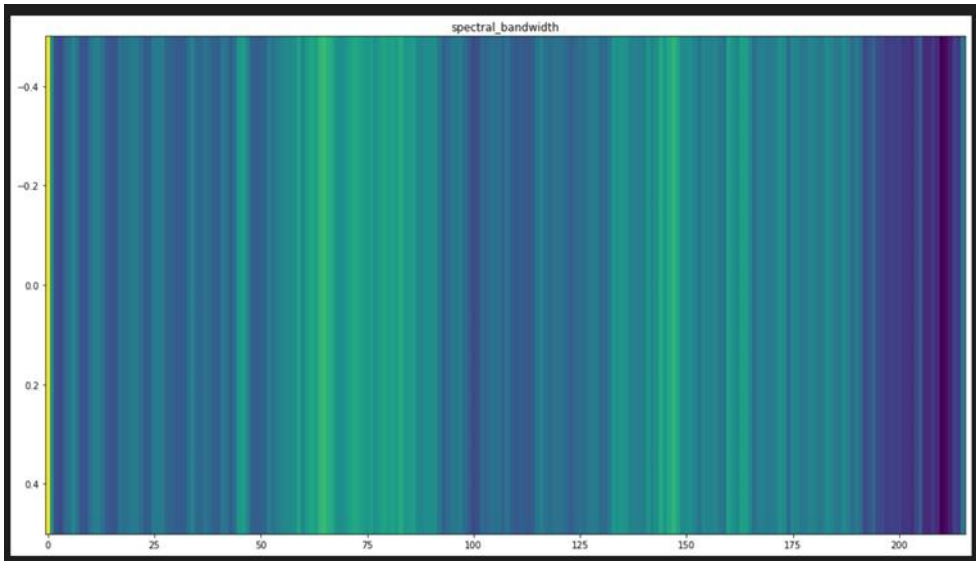

zerocrossing_rate
−0.4
−0.2
0.0
0.2
0.4
0
25
50
75
100
125
150
175
200

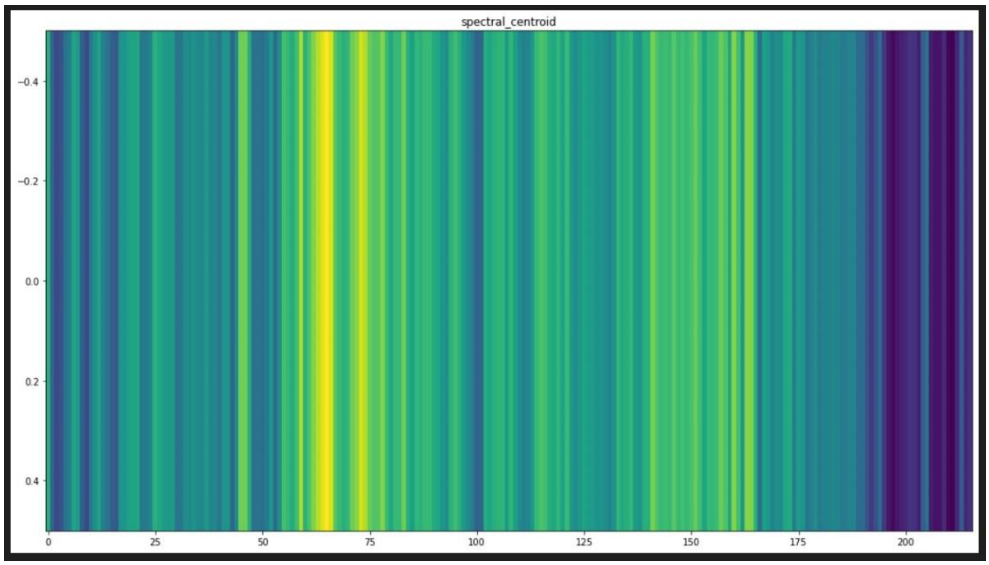

spectral_contrast
0
1
2
3
4
5
6
0
25
50
75
100
125
150
175
200

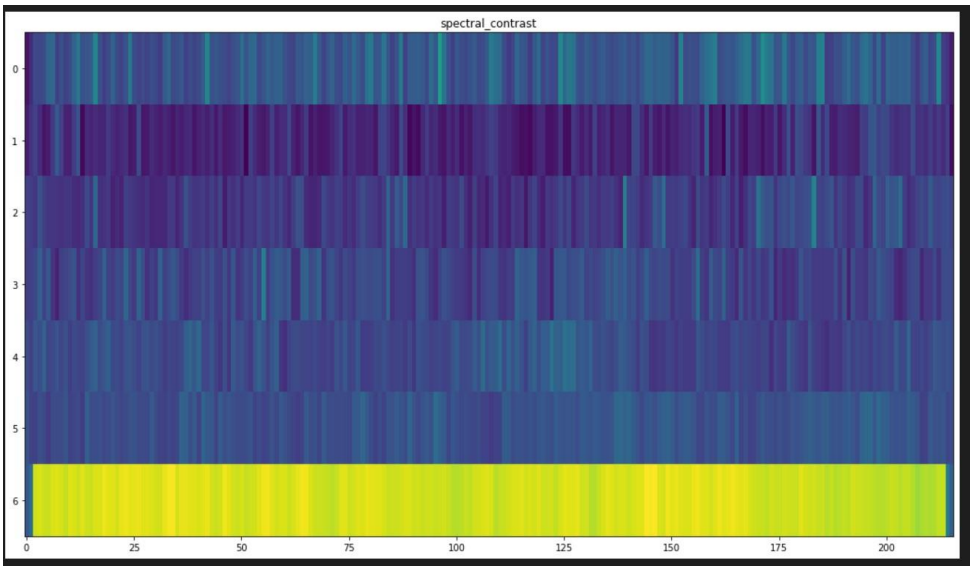

stft
0
200
400
600
800
1000
0
25
50
75
100
125
150
175
200

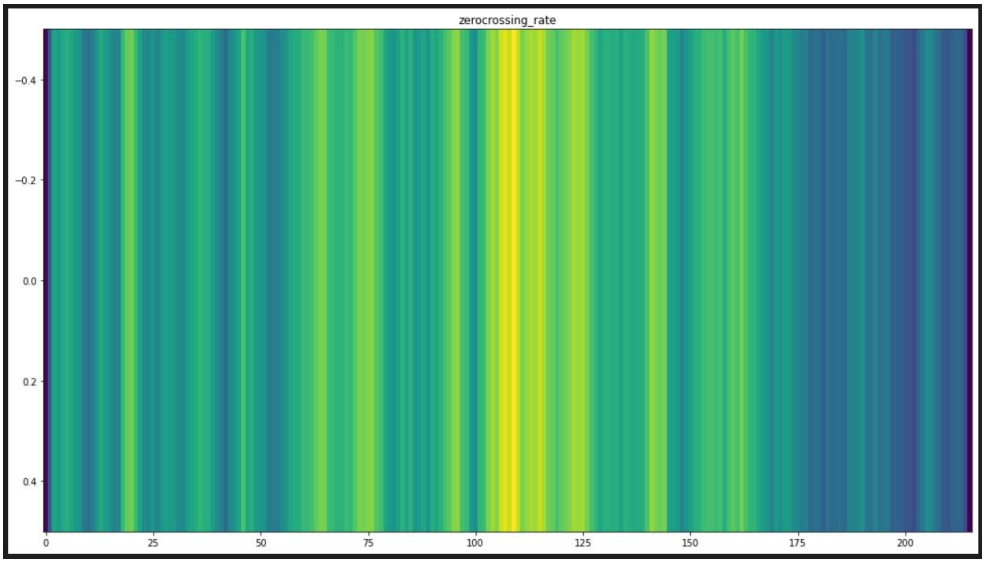
Features plots of TRAM-

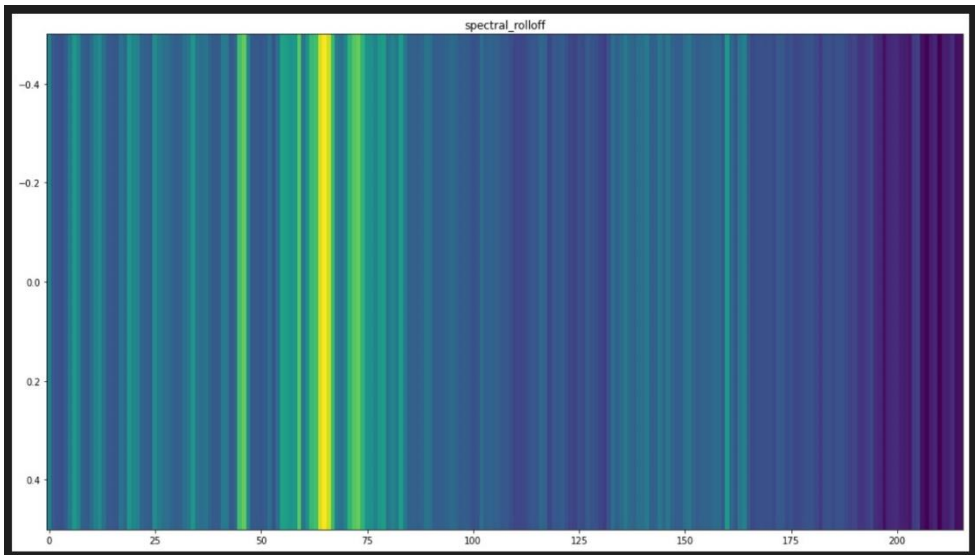


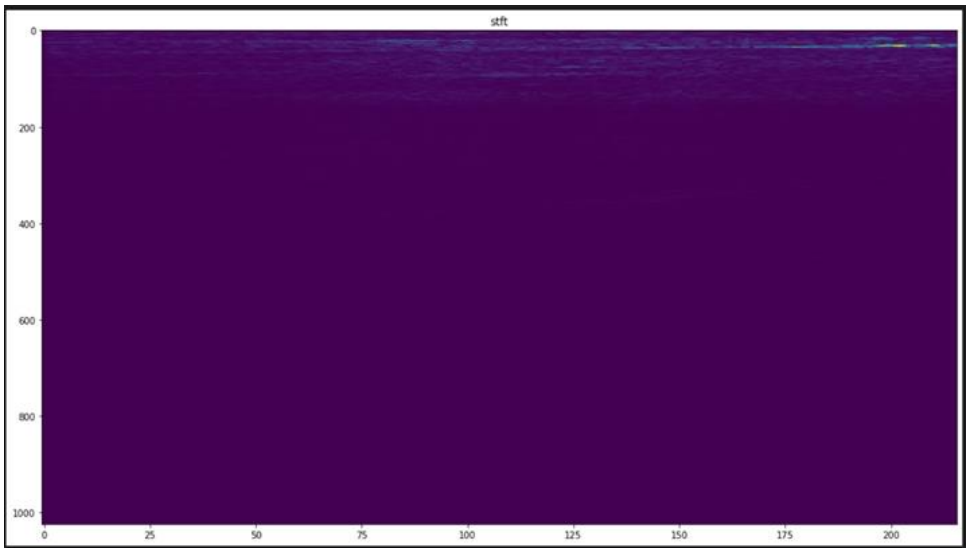



spectral_centroid
−0.4
−0.2
0.0
0.2
0.4
0
25
50
75
100
125
150
175
200


spectral_contrast
0
1
2
3
4
5
6
0
25
50
75
100
125
150
175
200


zerocrossing_rate
−0.4
−0.2
0.0
0.2
0.4
0
25
50
75
100
125
150
175
200


spectral_rolloff
−0.4
−0.2
0.0
0.2
0.4
0
25
50
75
100
125
150
175
200


stft
0
200
400
600
800
1000
0
25
50
75
100
125
150
175
200

4. Defining the model and training it

We have been given options to choose the model with SVM, Nearest Neighbor, Linear Regression and Neural Networks. Since the sample feature count is in a large scale as parameters, we decided to choose the Neural Networks since it can work better with handling the parameters and the computation of the classification will be high with neural network.

Although, it is known that CNN model is a black box implementation, we choose this because, they are well-suited for processing time-series data, such as audio signals. CNNs can learn time-invariant features from an input signal by applying filters that slide across the input in a predetermined pattern. This allows the model to learn to identify and extract relevant features from an input signal, which can be useful for tasks such as sound classification. Additionally, CNNs can learn hierarchical representations of the input data, which can capture both local and global patterns in the signal. This can be useful for tasks such as recognizing complex patterns or structures in an audio signal.

Below is the snap of the model summary -

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 20, 196, 4)        1768

 max_pooling2d (MaxPooling2D  (None, 4, 39, 4)         0
 )

 conv2d_1 (Conv2D)           (None, 2, 37, 16)         592

 flatten (Flatten)           (None, 1184)              0

 dense (Dense)               (None, 16)                18960

 dense_1 (Dense)             (None, 8)                 136

 dense_2 (Dense)             (None, 1)                 9

=================================================================
Total params: 21,465
Trainable params: 21,465
Non-trainable params: 0
_________________________________________________________________
None
```

We have trained the model with the data collected from fellow peers and split that data into training and testing batches.

5. Testing and evaluating the model

We have evaluated the model with the data that we both had collected from the 2 classes, Tram, and Car. We have achieved the accuracy score of 98% in this step.

Below are the snaps of the training and testing and evaluation scores –

```
Epoch 1/100
36/36 [==============================] - 3s 68ms/step - loss: 0.6106 - accuracy: 0.7753 - val_loss: 0.4608 - val_accuracy: 0.8111
Epoch 2/100
36/36 [==============================] - 2s 66ms/step - loss: 0.1283 - accuracy: 0.9635 - val_loss: 0.9013 - val_accuracy: 0.7333
Epoch 3/100
36/36 [==============================] - 2s 61ms/step - loss: 0.0539 - accuracy: 0.9860 - val_loss: 0.1054 - val_accuracy: 0.9778
Epoch 4/100
36/36 [==============================] - 2s 68ms/step - loss: 0.0313 - accuracy: 0.9944 - val_loss: 0.1859 - val_accuracy: 0.9333
Epoch 5/100
36/36 [==============================] - 3s 72ms/step - loss: 0.0306 - accuracy: 0.9916 - val_loss: 0.0435 - val_accuracy: 0.9889
Epoch 6/100
36/36 [==============================] - 2s 65ms/step - loss: 0.0242 - accuracy: 0.9944 - val_loss: 0.2711 - val_accuracy: 0.8778
Epoch 7/100
36/36 [==============================] - 2s 65ms/step - loss: 0.0495 - accuracy: 0.9860 - val_loss: 0.0548 - val_accuracy: 0.9889
Epoch 8/100
36/36 [==============================] - 2s 65ms/step - loss: 0.0559 - accuracy: 0.9775 - val_loss: 0.2068 - val_accuracy: 0.9333
Epoch 9/100
36/36 [==============================] - 2s 69ms/step - loss: 0.0517 - accuracy: 0.9803 - val_loss: 0.0667 - val_accuracy: 0.9889
Epoch 10/100
36/36 [==============================] - 2s 67ms/step - loss: 0.0093 - accuracy: 0.9972 - val_loss: 0.0729 - val_accuracy: 0.9889
Epoch 11/100
36/36 [==============================] - 3s 71ms/step - loss: 0.0021 - accuracy: 1.0000 - val_loss: 0.1939 - val_accuracy: 0.9111
Epoch 12/100
36/36 [==============================] - 2s 66ms/step - loss: 0.0013 - accuracy: 1.0000 - val_loss: 0.1230 - val_accuracy: 0.9667
...
Epoch 99/100
36/36 [==============================] - 2s 68ms/step - loss: 5.8670e-06 - accuracy: 1.0000 - val_loss: 0.3412 - val_accuracy: 0.9000
Epoch 100/100
36/36 [==============================] - 3s 71ms/step - loss: 5.7934e-06 - accuracy: 1.0000 - val_loss: 0.3382 - val_accuracy: 0.9000
```

Model Prediction –

```
14/14 [==============================] - 1s 35ms/step
[[1.00000000e+00]
 [9.99999285e-01]
 [1.00000000e+00]
 [1.00000000e+00]
 [1.00000000e+00]
 [1.00000000e+00]
 [1.00000000e+00]
 [1.00000000e+00]
 [1.00000000e+00]
 [1.00000000e+00]
 [1.00000000e+00]
 [1.00000000e+00]
 [1.00000000e+00]
 [9.99999881e-01]
 [1.00000000e+00]
 [1.00000000e+00]
 [9.96733725e-01]
 [9.98320162e-01]
 [1.00000000e+00]
 [1.00000000e+00]
 [1.00000000e+00]
 [9.99999940e-01]
 [1.00000000e+00]
 [1.00000000e+00]
 [1.00000000e+00]
...
 [1.04561775e-06]
 [1.88364752e-03]
 [5.96978469e-04]
 [8.48026538e-10]]
```

Model evaluation score –

```
    scores = model.evaluate(eval_features, eval_labels, verbose=0)
    print(scores)
2]  ✓ 0.3s

[0.043138906359672546, 0.9800000190734863]
```

DIVISION OF WORK –

1. The data collection of 2 different classes was done individually by each one of us per class.
2. While writing the code to extract feature, both of us connected and worked together to identify which features are being considered.
3. Defining the model and training and evaluating the model too was done together since we both connected with each other daily on teams.