1. Introduction

The project objective is to build a neuron network in python to implement audio binary classification. This document will discuss the process of construing the neuron network.

2. Data description

The specific data used in constructing the network is the audio of trams and buses with the final goal is to create a classifier that can distinguish between these 2 sounds. The data is from the shared data made by the students of the course. Because of its nature, audio quality might vary depending on the device.

Because the model that is going to be built is a binary classification, only 2 classes needed to be defined. In this case, the audio of trams is categorized as positive, and the buses are marked as negative.

3. Feature extraction:

There were servals of spectrogram can be considered for this project: conventional spectrogram, log-spectrogram, mel-spetrogram, log-mel-spectrogram, MFCCs spectrogram, and CQT spectrogram. Since MFCCs and CQT spectrogram main purpose are for speech and musical processing, it will not be considered. The four remaining spectrograms are different methods to visualize the audio, therefore, they will be evaluated. The spectrograms for each case can be shown as follows:

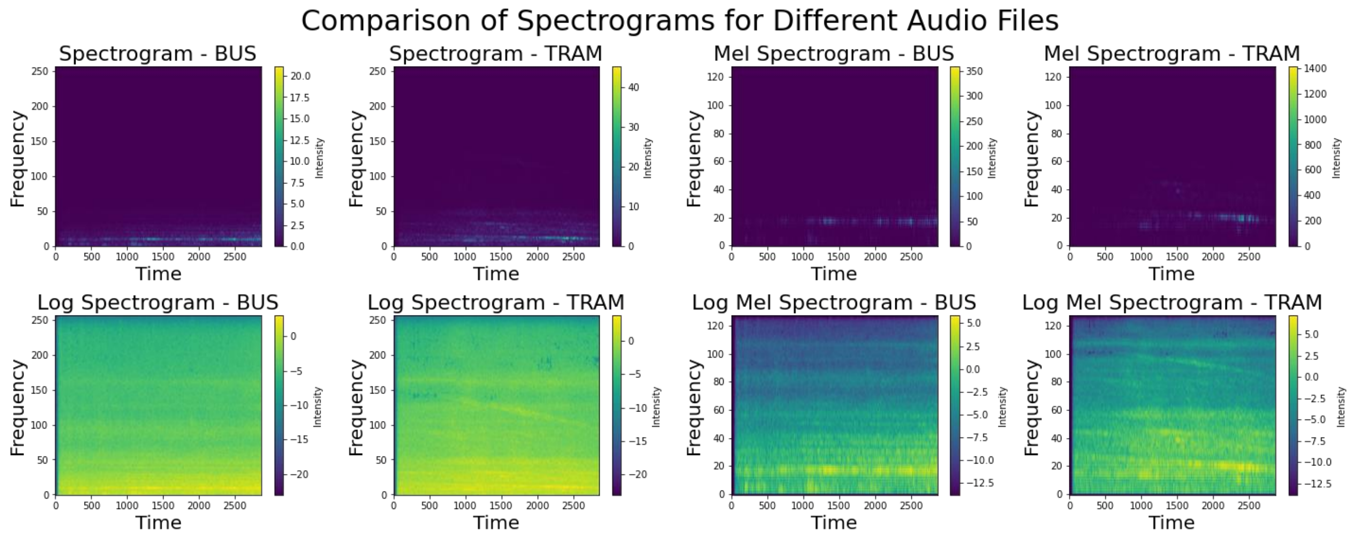
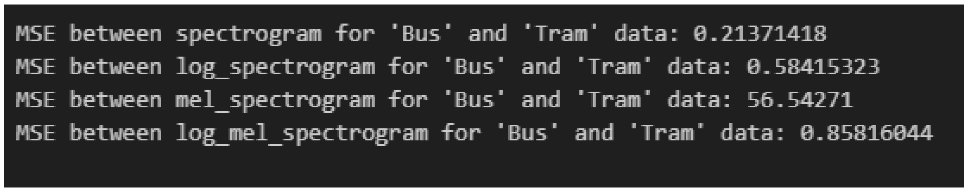


Figure 3.1: Spectrogram comparison

As illustrated in the figure, the log spectrogram and the log-mel-spectrogram give the most data for the machine learning model to analyze. Now, it is the problem of deciding which spectrogram type can show the most discrepancy between 2 classes. Mean square error (MSE) can be used to solve this problem. Calculating the mean square error between the spectrograms of each class can evaluate the efficiency of each spectrogram type. For example, the MSE of 'spectrogram' method can be calculated by taking the mean of all the spectrograms of buses noise and the mean of spectrograms of trams noise as parameters for spectrogram calculation. The MSE can then be used to decide if the spectrogram method should be used for the model. Here is the result of the spectrograms MSE calculation:



```
MSE between spectrogram for 'Bus' and 'Tram' data: 0.21371418
MSE between log_spectrogram for 'Bus' and 'Tram' data: 0.58415323
MSE between mel_spectrogram for 'Bus' and 'Tram' data: 56.54271
MSE between log_mel_spectrogram for 'Bus' and 'Tram' data: 0.85816044
```

Figure 3.2: MSE result

As seen in the result, the MSE of log-mel-spectrogram between Bus and Tram are the greatest, therefore, this type of spectrogram shows the most difference between 2 classes. Because of that, this log-mel-spectrogram will be used for the model.

4. Model selection, data split:

4.1 Data sample space:

The process starts by creating a loading function to load all the audio. The loading function (load_wav_16k_mono) loads the file into the correct format, which is in wave form. During this process, there was difficulty loading the file because it is not in the .wav format and some of the .wav files were not in 16bit. For this reason, some files of the samples had to be removed, while others were re-formatted.

The data set used for processing is created using tensor flow. The data set consists of two data as it said which is the positive and the negative, they are denoted as POS and NEG, these two subjects set our initialize as a path leading to the folder storing all the samples. It will then be converted into a TensorFlow data set by given the flag as mentioned on the introduction the bus is noise are marked as positive and tram armor marked as negative after this the two Subs the two sub-data will be added with a label one and 0 and then combined with the main data set

After the initialization of the datasets, it is crucial to calculate the mean cycle length of the audio. The reason for this is due to the sample's nature some audios are longer than others why the model needs uniform sample set. The mean cycle length can be achieved by taking the mean of the length of the of the tensor wave collected in the data set in the project the means cycle length is identified as 92770. this number will then be labor to use when converting audio to spectrogram.

The final crucial step of finishing the data set is to at the spectrogram to the data set, for this we need to build a preprocess function. To ensure we gain a uniform sample, the functions will only take the first 92,000 samples from an audio file if it has fewer than the mentioned sample, the function will perform zero padding to the short audios. After the function is finished, the spectrogram is mapped into the data set and the TensorFlow data pipeline is built.

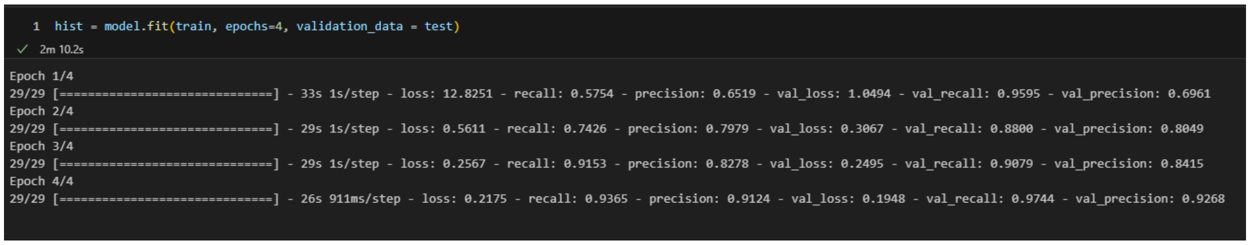
4.2 Building model

The model used in the project is a convolutional neural network (CNN). It comprises of two set of convolutional layers followed by a Max pooling layer to extract and condense features from the input images which are the spectrograms, the flattened output isn't fair into a dense layer for deeper representation and final by the reclassification using sigmoid activation.

To prevent our filtering the data set needs to be divided into different smaller said age had 29 members for training, why another 12 members are for validating.

5. Result

The model is trained with epoch value set to 4. Bellows is the training process:



```
hist = model.fit(train, epochs=4, validation_data = test)
✓ 2m 10.2s

Epoch 1/4
29/29 [==============================] - 33s 1s/step - loss: 12.8251 - recall: 0.5754 - precision: 0.6519 - val_loss: 1.0494 - val_recall: 0.9595 - val_precision: 0.6961
Epoch 2/4
29/29 [==============================] - 29s 1s/step - loss: 0.5611 - recall: 0.7426 - precision: 0.7979 - val_loss: 0.3067 - val_recall: 0.8800 - val_precision: 0.8049
Epoch 3/4
29/29 [==============================] - 29s 1s/step - loss: 0.2567 - recall: 0.9153 - precision: 0.8278 - val_loss: 0.2495 - val_recall: 0.9079 - val_precision: 0.8415
Epoch 4/4
29/29 [==============================] - 26s 911ms/step - loss: 0.2175 - recall: 0.9365 - precision: 0.9124 - val_loss: 0.1948 - val_recall: 0.9744 - val_precision: 0.9268
```

Figure 5.1: Training result

The correlation of the training value and the validation value can be visualized as bellow:

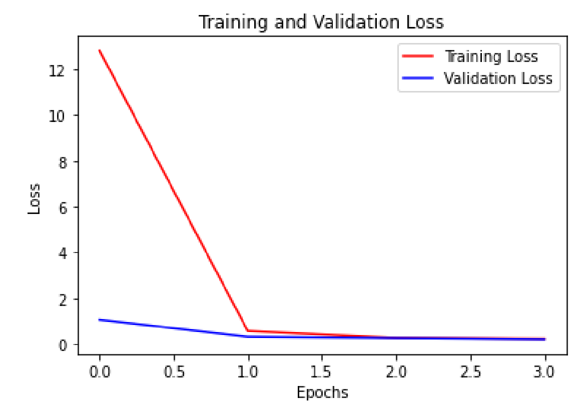


Figure 5.2: Training and Validation Loss

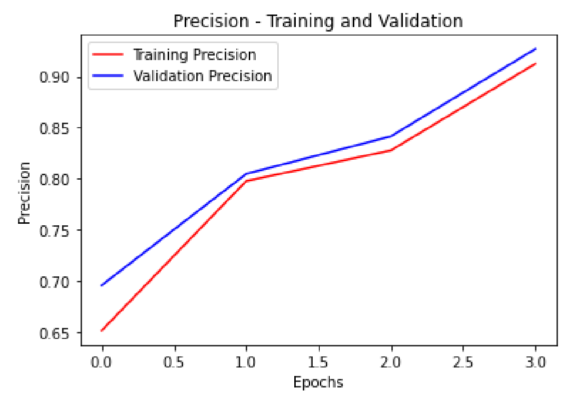


Figure 5.2: Precision – Training and Validation

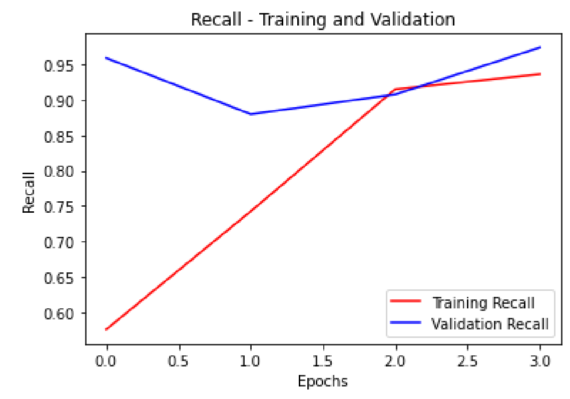


Figure 5.3: Recall – Training and Validation

As expected, the model performed poorly at first, and became more efficient at the end. The precision at the beginning was around 0.7292 and at the end, it reached 0.8985. The final model was efficient with sufficient accuracy.

6. Conclusion

During the project, I concluded the following. for data analysis sample collection process is especially important and to have a smooth operation it should be insist that all the sample are of the same format. Second, when doing audio classification, especially when related to spectrogram, it is worth noting that weak spectral graph suits best for the audios. Third, we're building that asset for machine learning model it is important to normalize the data for the best results.