

Project3 – Section2

96222031 – Soroush Heidary

(don't mind the page count :\ its mostly pictures :/ and a BIG font)

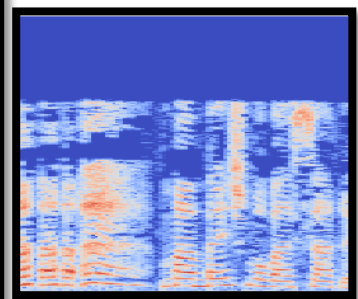
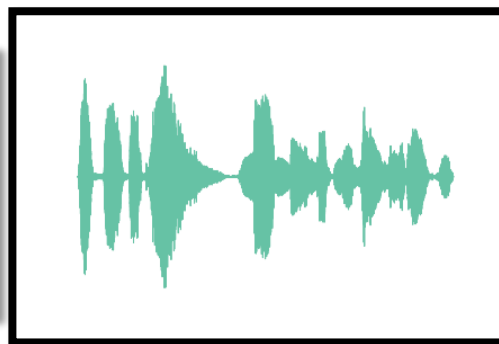
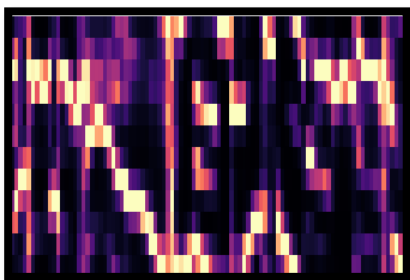
Main idea :

There were a couple of approaches tested to get the best results which are as follow :

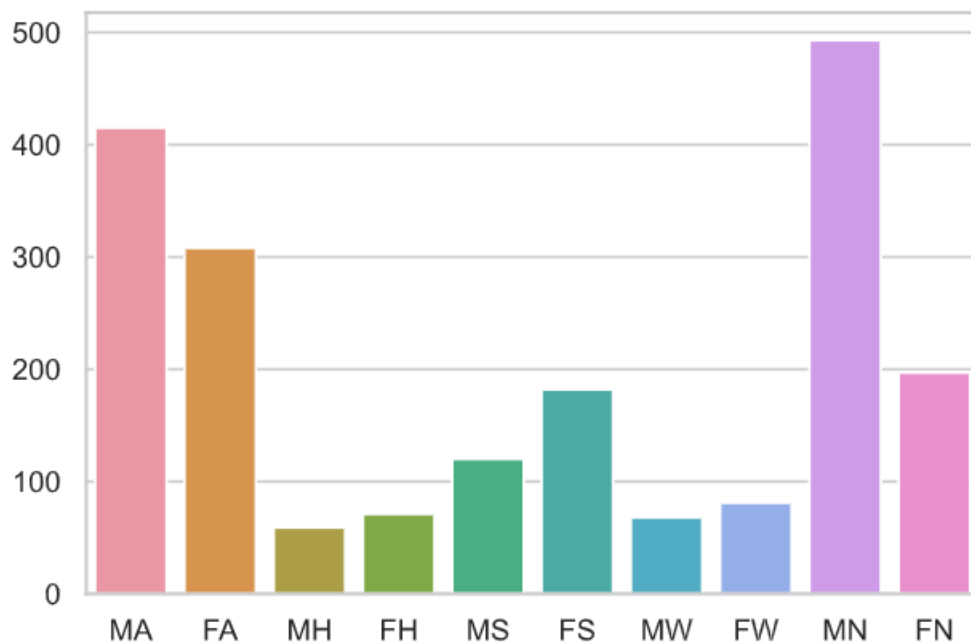
- 1- Plain LSTM (single input, single output)
- 2- Bidirectional LSTM (single input, single output)
- 3- Bidirectional LSTM (multiple input, single output)
- 4- Bidirectional GRU (multiple input, single output)
- 5- CNNs (3 different inputs)

From those above the most successful was the BiDiLSTM with 2 set of inputs (one the sequential features extracted by 36 time steps, the other some unary time steps which will be explained)

CNN also worked better than expected, these are the 3 kind of inputs used in it which will be explaine in details later on



Before hopping into the detailed explanations lets take a look at this class distribution chart :


















As we see the Happy class and Wondered class are almost dominated by the other classes, so we're gonna guess they'll get kind of ignored when trained, there were this idea to somehow augment to those classes by trimming the longer audios in H and W, but that didn't work out so good, so we'll just ignore that in the rest of the report

Detailed Explanation :

Feature extraction :

For extracting features the librosa library was used, here is a list of features this library can extract and which of which is used :

 chroma_stft ([y, sr, S, norm, n_fft, ...])	Compute a chromagram from a waveform or power spectrogram.
 chroma_cqt ([y, sr, C, hop_length, fmin, ...])	Constant-Q chromagram
 chroma_cens ([y, sr, C, hop_length, fmin, ...])	Computes the chroma variant "Chroma Energy Normalized" (CENS)
 melspectrogram ([y, sr, S, n_fft, ...])	Compute a mel-scaled spectrogram.
 mfcc ([y, sr, S, n_mfcc, dct_type, norm, lifter])	Mel-frequency cepstral coefficients (MFCCs)
 rms ([y, S, frame_length, hop_length, ...])	Compute root-mean-square (RMS) value for each frame, either from the audio samples 
 spectral_centroid ([y, sr, S, n_fft, ...])	Compute the spectral centroid.
 spectral_bandwidth ([y, sr, S, n_fft, ...])	Compute p'th-order spectral bandwidth.
 spectral_contrast ([y, sr, S, n_fft, ...])	Compute spectral contrast
 spectral_flatness ([y, S, n_fft, hop_length, ...])	Compute spectral flatness
 spectral_rolloff ([y, sr, S, n_fft, ...])	Compute roll-off frequency.
 poly_features ([y, sr, S, n_fft, hop_length, ...])	Get coefficients of fitting an nth-order polynomial to the columns of a spectrogram.
 tonnetz ([y, sr, chroma])	Computes the tonal centroid features (tonnetz)
 zero_crossing_rate ([y[, frame_length, ...]])	Compute the zero-crossing rate of an audio time series.

The blue one are those which were used in LSTMs

The orange ones are those which were tested but not used

The green ones are those which were used in CNNs

The rest just didn't fit our needs

LSTM :

The first version, plain LSTM didn't yield promising results, the accuracy was around 30 percent and it was predicting everything as A or sometime as N, probably because they were the dominant class

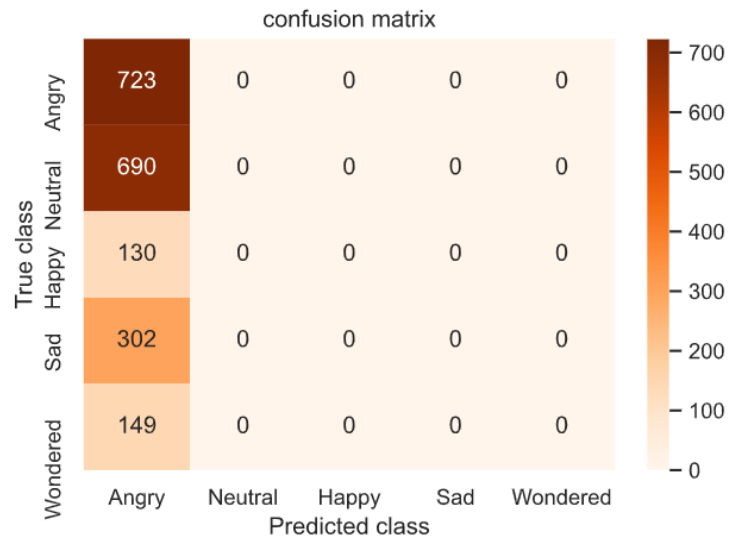
Also the features used for this version was both mfcc and chroma cens

Mfcc had a little bit more accuracy and using both of them as a concatenated vector of both didn't gave us any better results than mfcc

alone so we just ignored the cens for the rest of the press, though we still used the chroma gram in CNN variants

This is the heatmap of this version :

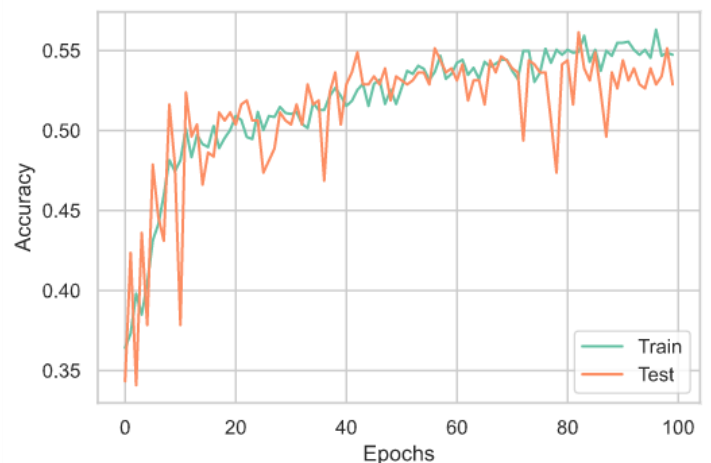
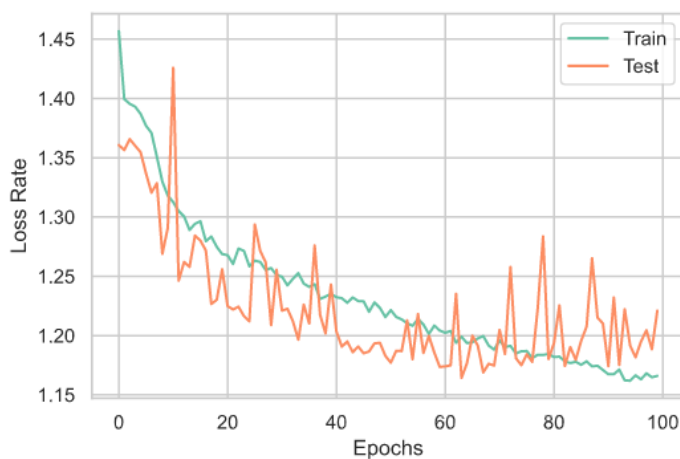
with 20 epochs, 1 LSTM layer of 36 time steps followed by 2 Dense layers of width 25 and 15 respectively and a output layer of 5 neurons, all the activations were than, Adam optimizer



As you saw the result on last try was completely %@&!

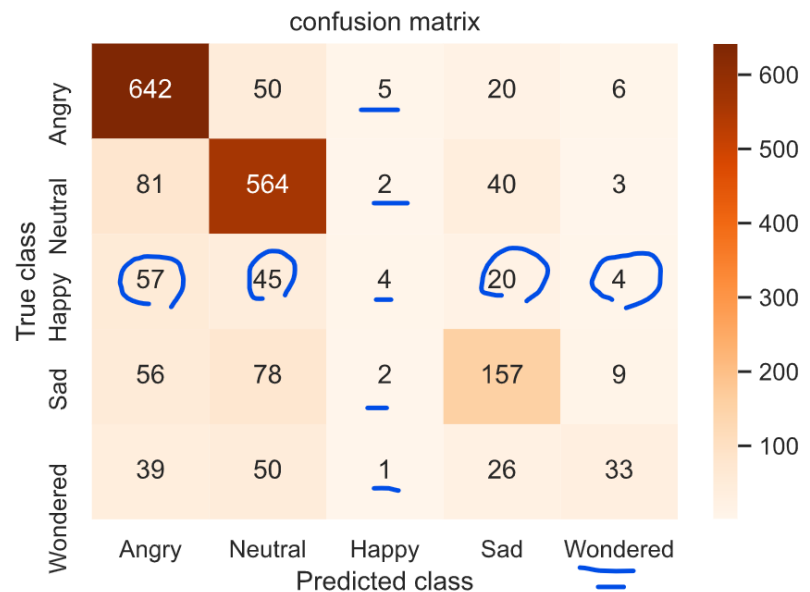
This time we'll go for a BiDirectional LSTM as suggested from a teacher assistant which improved the predictions quite nicely

This time we made the network a little bit simpler using only one Dense layer of width 20, here are the results :



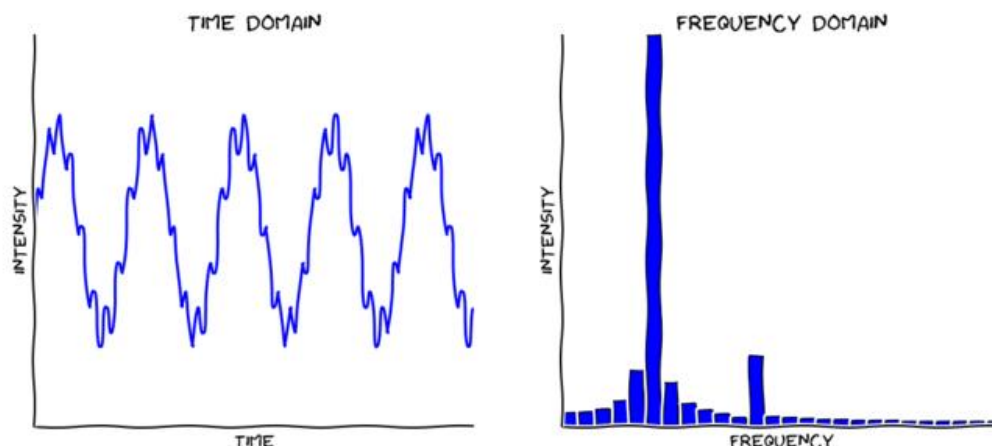
A looooot of fluctuations, we will make that better later on (actually we'll just make the network more efficient by using multi inputs and then move to 10 epochs, that will remove the fluctuation problem too

this is the result of the bi directional single input version, as expected we're having little to none correct predictions of classes H and W



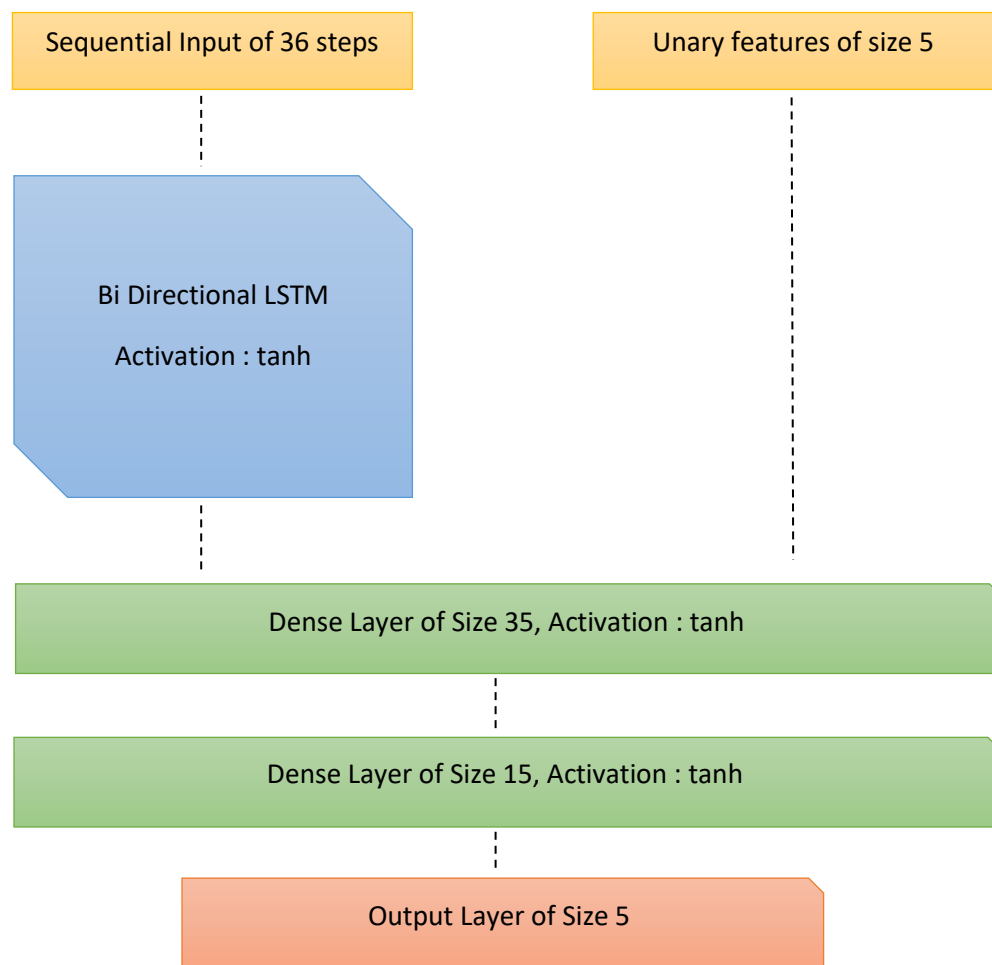
The next step is to try to give our network a sense of when to actually know if it's of class H and W and not always one of S, A or N

For this purpose we're going to add another input layer after the LSTM layer which will contain information like how many times the waveform plot crosses the zero line or how many time steps has the frequency of



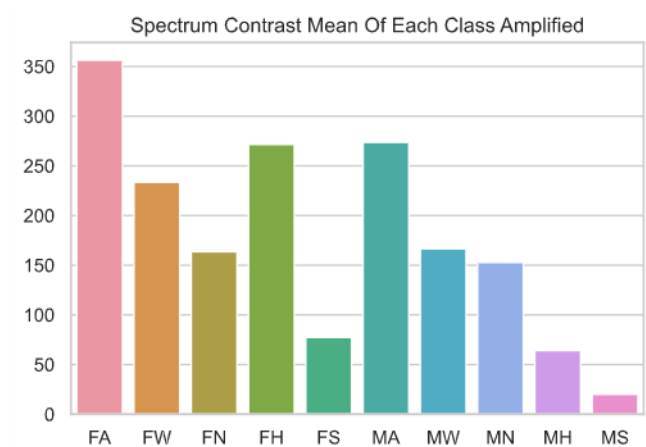
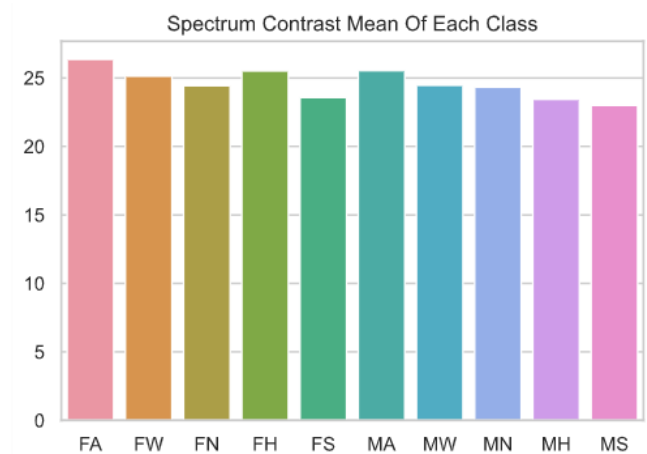
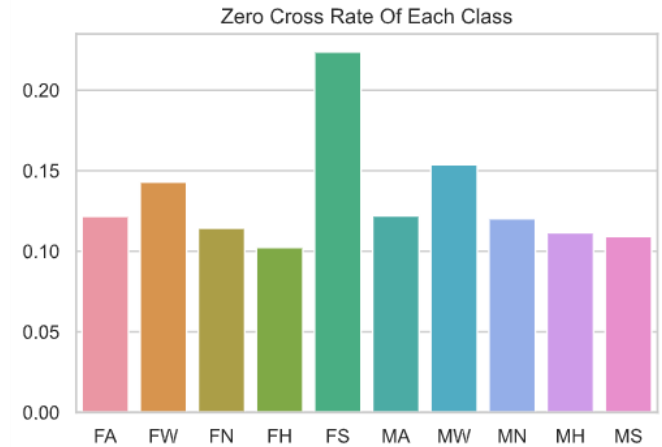
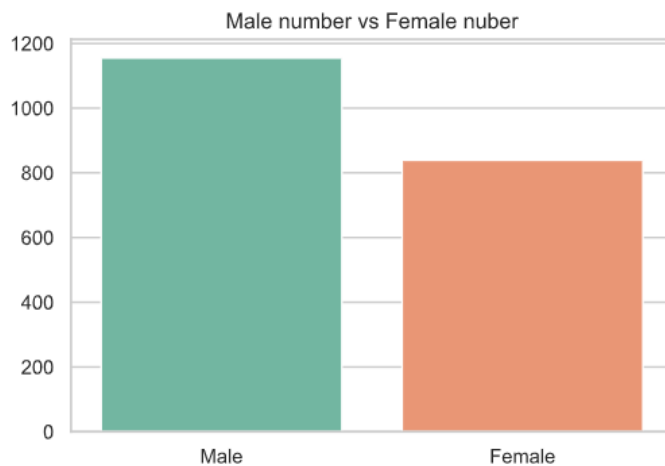
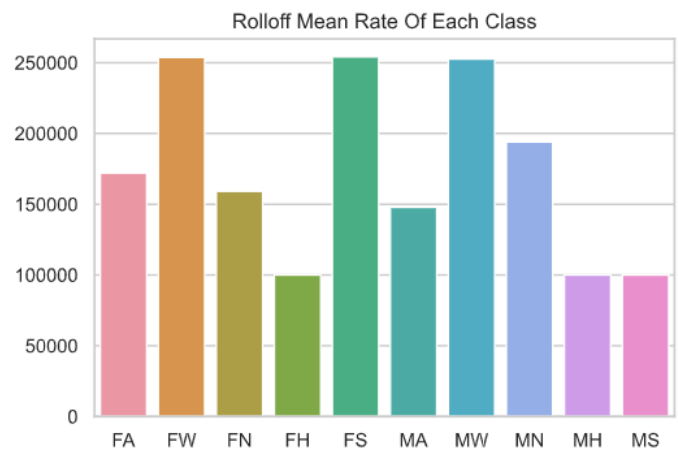
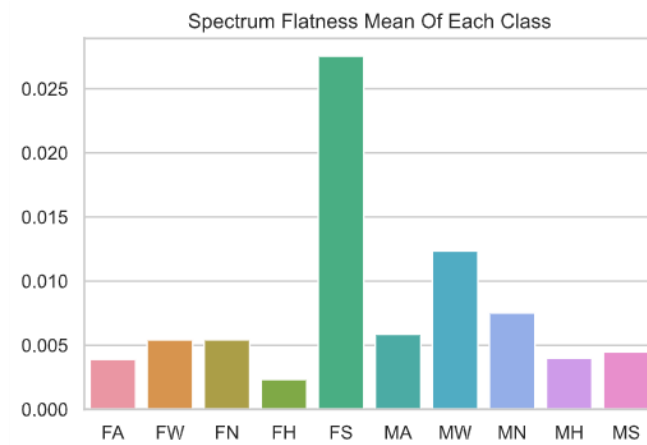
near to maximum frequency, all these features are extracted by librosa except the last feature which is gender, features are extracted mainly from these 2 plots above

After adding multiple inputs our network structure looks like this :

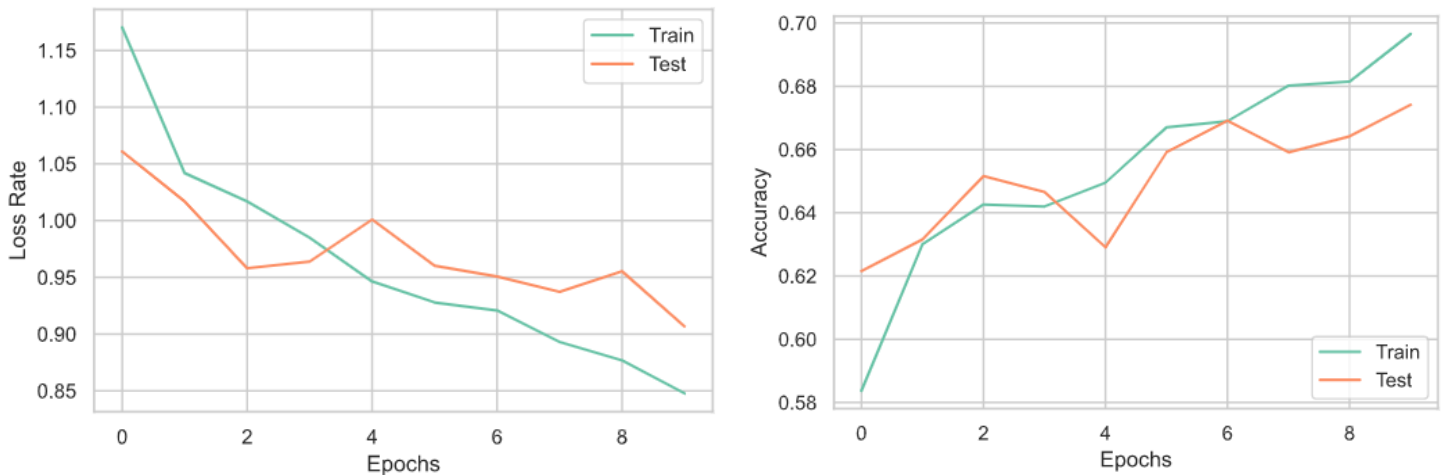


Here you can see the 4 features we used in the second input layer and the distribution of data in each class according to the respecting feature (spectrum contrast had a really low variance between in the classes but still it was important to us because it did clearly show some meaning for

each class we just have to amplify the variance somehow, I have put the before and after of its amplification process)

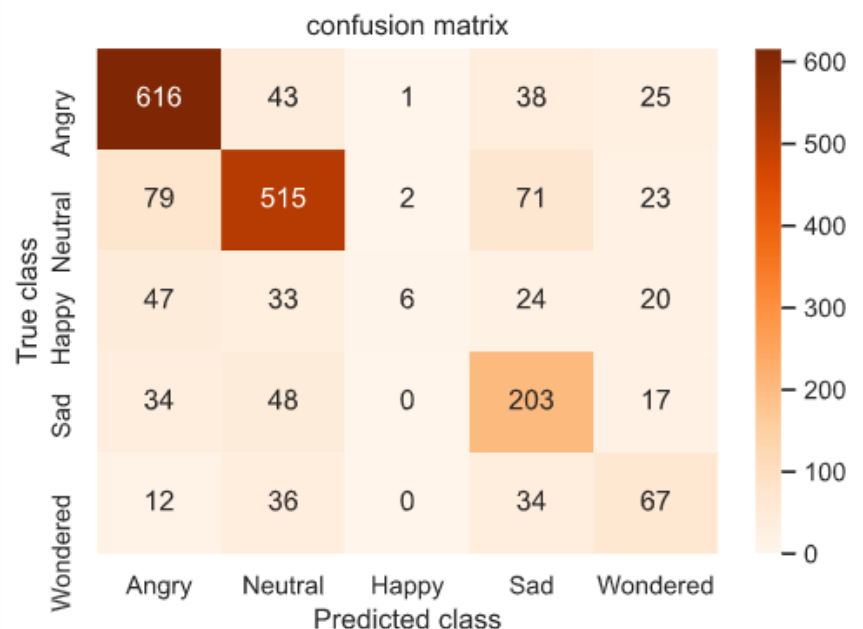


This Bidirectional version had a more promising results than the single input ones, here is the results :



As it nearly gave us around 67 percent accuracy on validate data, we can still make one simple modification to the LSTM structure to make it better

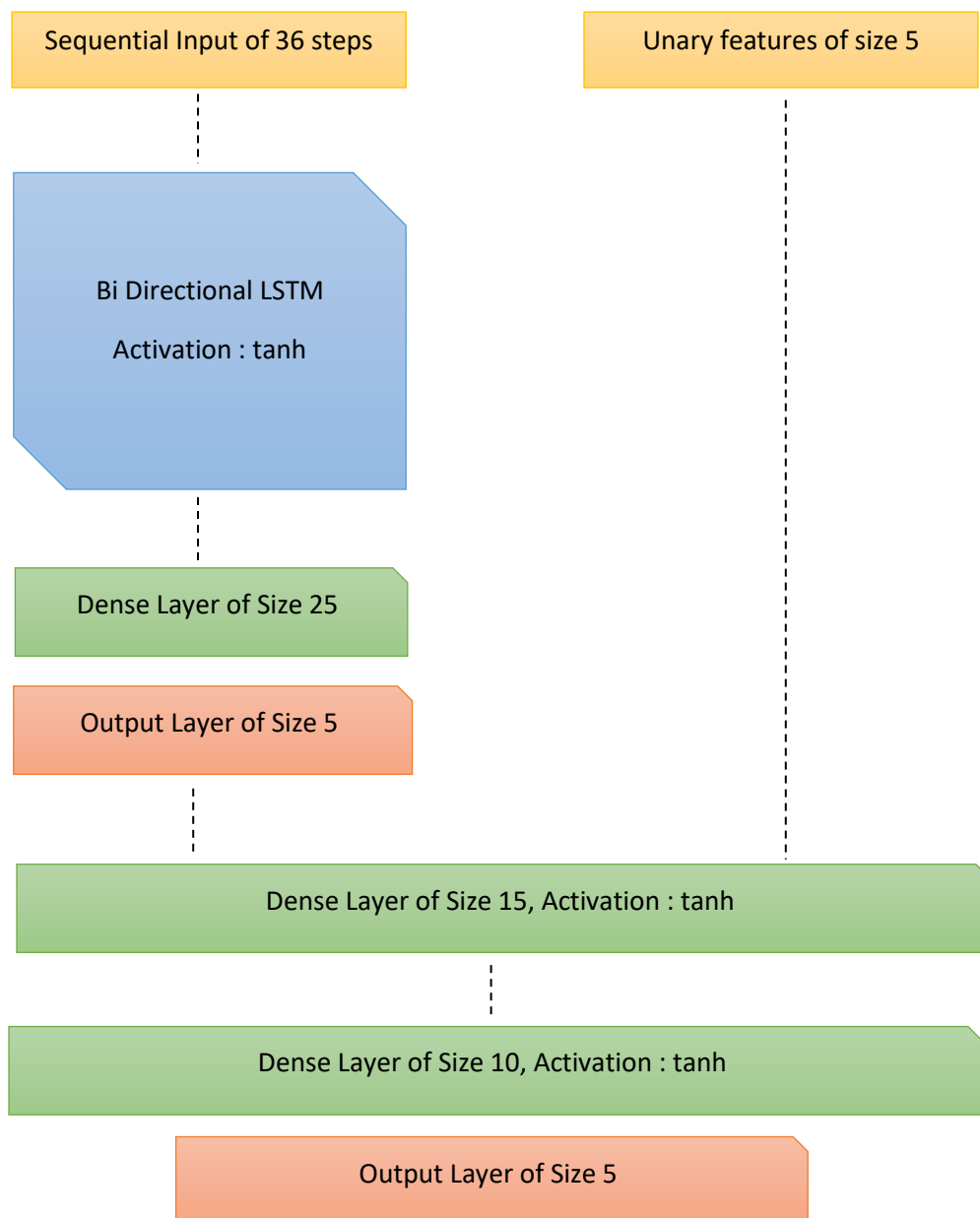
As you can see we have achieved a better network though yet Happy class doesn't get any good predictions, for this problem we test an assumption which as follow



The assumption is that if we let the LSTM does its works and let an ANN to classify the LSTM outputs alone and then we add the unary features to the ANN output and feed it to yet another ANN we'll get more chance

for dominated classes like happy, mainly because the LSTM has an output of 36 which is strictly more the unary feature sizes (5) so if we let the LSTM classify its outputs into 5 classes (this may not necessarily be our 5 desired classes) and now we concatenate unary inputs and the LSTM-ANN output we'll have a vector of 10 size for the last ANN classifier and hopefully this will make the chance of dominated classes better)

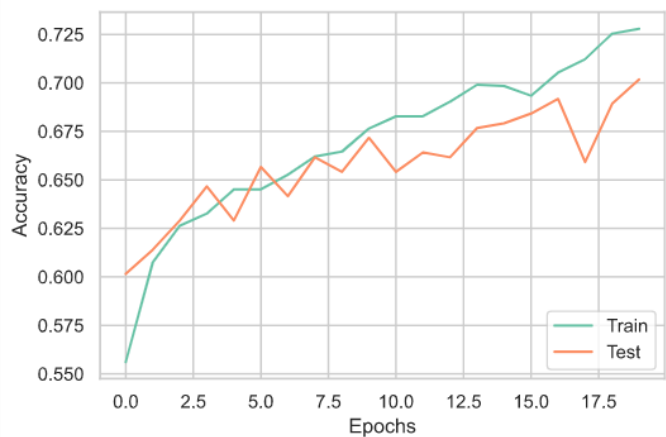
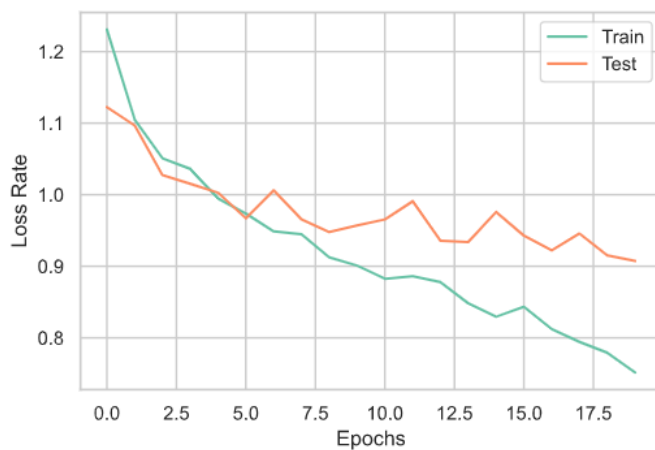
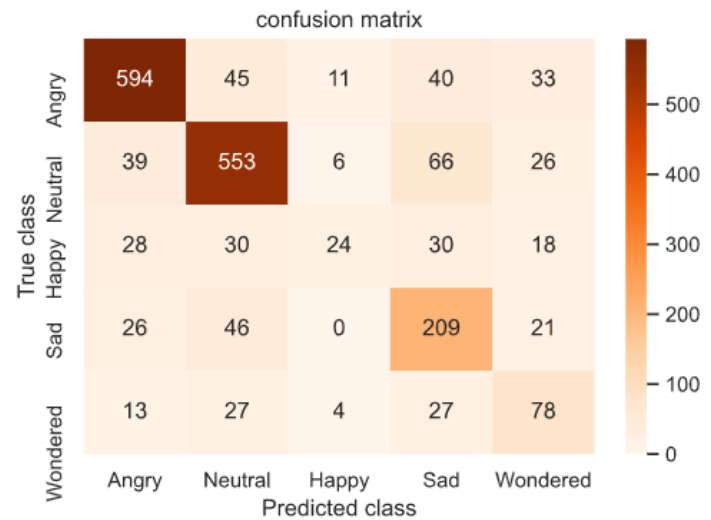
By doing that we get this network :



And the result we got is :

Though what we hoped didn't practically happen but the overall accuracy improved by around 3.5 percent and we reached 70 percent accuracy

Worthy of a try tough :D



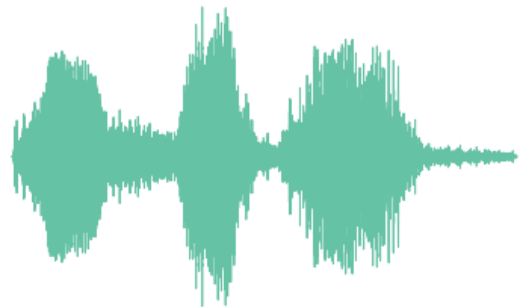
CNNs :

This approach was a mere act of satisfying curiosity and is obvious that in these cases when we have high number of time-steps to track (not necessarily this data set as it's audio samples wasn't that long but still long enough for the long term memory networks to work better than a CNN) the LSTM yields a more promising result

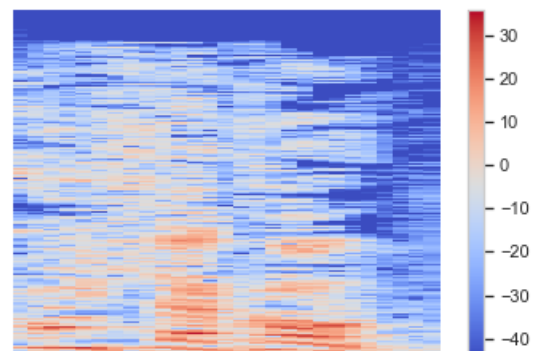
I tried The CNN on 3 different data sets : for creating these I plotted them in the first code and saved the figures without the axis and cropped them all before use in CNN

Though I tried to look around these 3 data sets to manually find any meaning in them, there seams that human eye cant catch those kind of correlations :\

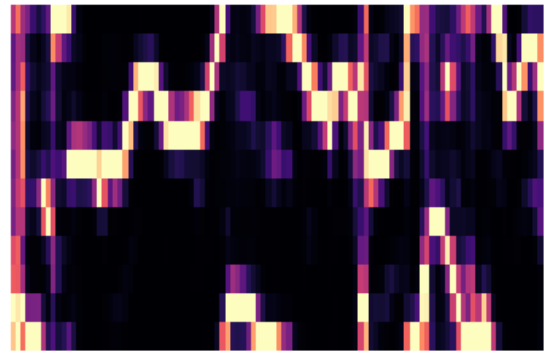
- 1- Wave form: which is basically a demonstration of frequency intensity of each time step



- 2- Spectogrma : which is essentially
A visualization of frequency spectrum with respect to time in simplere words it's showing us how much of each frequency is present at each moment (t = x axis) this chart is also called sonographs or voiceprint



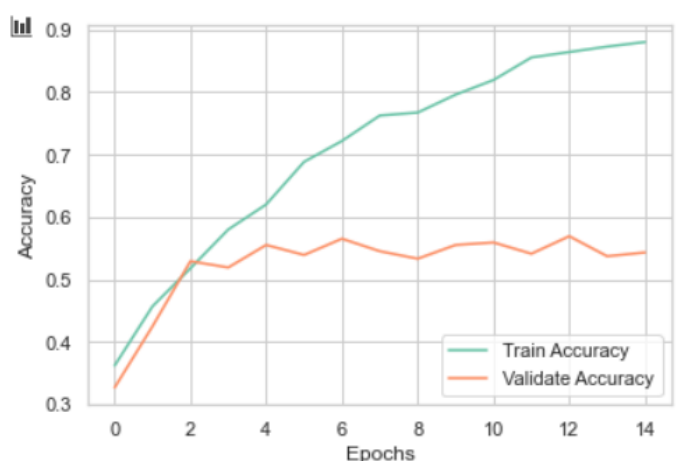
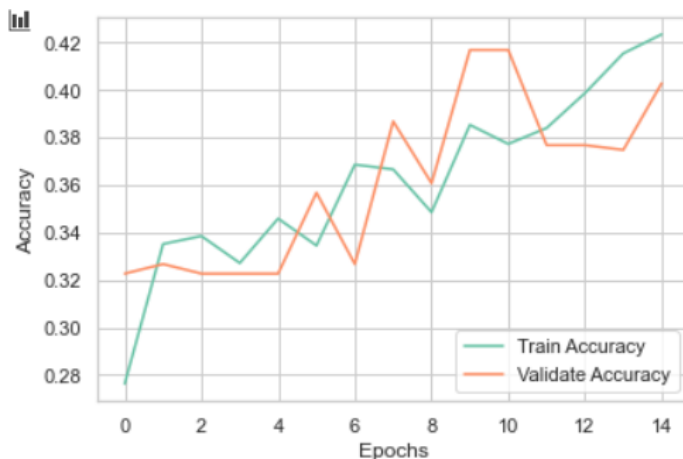
3- Chromagram : this chart shows (on y axis) each o notes and their presence rate in each time step(on x axis), for example in first row we can see how much the note 'B also called Si in Latin' was present at each frame



I the CNN with these 3 data sets (now that I think about it using a 3 channeled CNN which has all three fed at the same time could had better results :\ but too late to do any other modifications to the code)

Though they all had a accuracy of around 55 percent and almost all of data set we're getting predicted to be either A or N , but there is 2 assumptions which came out actually surprisingly true, and helpful

1- On using CNN in these data set, using a pooling layer does messes up the results, because by doing max pooling and mean pooling we'll be denying the information which actually defines a sound wave and that is the fluctuations



Here you can see 2 charts, one with pooling and the other without, True, we're getting overfit in the right chart which is the version without the use of Pooling, but if we were to stop at epoch 2 or 3, and even now, we have a validation accuracy of 55 but with pooling it would be 39

The reason could potentially be that of the fact that the dominated classes lose their detail in pooling after each backpropagation due to be dominated with A or N details

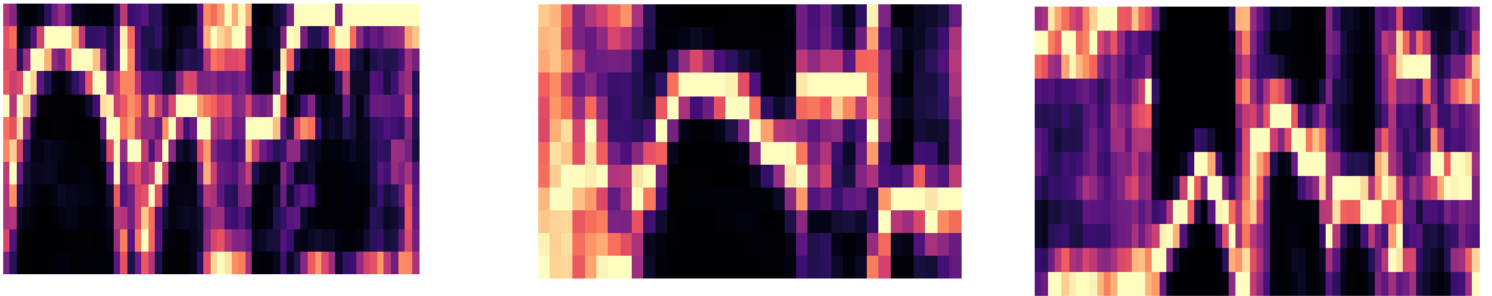


2- We assume that using chromagram we'll have a chance of detecting wondered class because of this pattern that I found in the data set by manually wandering in the pictures, bear with me and look at these pictures : Turns out that pictures labeled with 'W' had these 'very clear sin like waves' comparing to other classes which were more messy

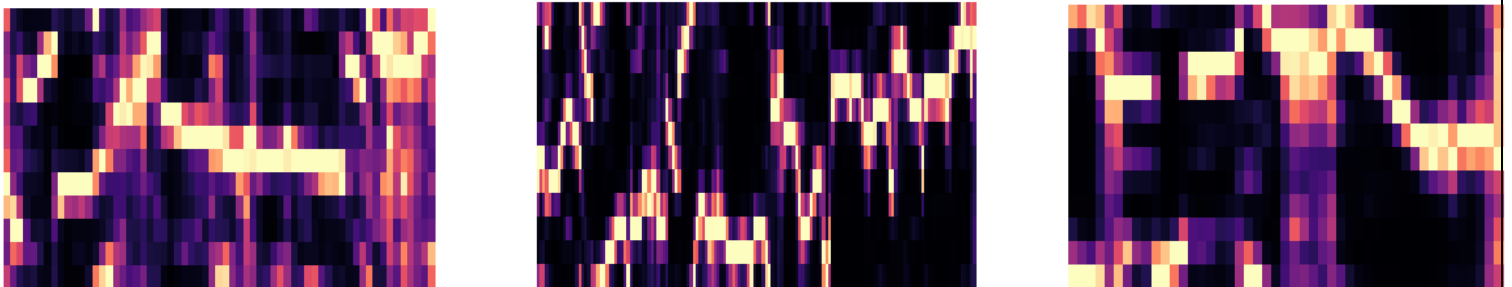


This kind of means that when a person is wondered about something the voice they make contain an octave of musical notes back and forth for a sound like 'aoooo' or 'weooooow' which kind of contain all the musical notes in an octave

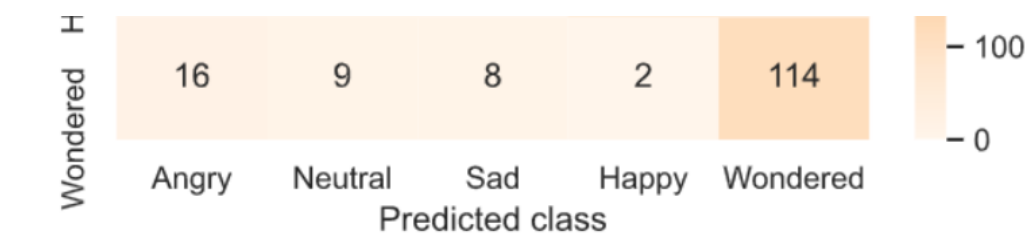
There are some other examples of 'W' class shown below



Comparing to anger or other classes :



And the results show that its actually true :



Well, ain't that interesting !! :)