Taiwanese Soap Opera Speech Emotion Recognition

Yeh, Ssu-Yung (葉思永), Lin, Yi-Ting (林奕廷) , Hung, Sheng-Hsiang (洪聖祥), Zhang, Hong-Qi (張紘齊), Hsu, Chun (許淳)

***Abstract*—We tried out different approaches to build convolutional neural network (CNN) models that can distinguish emotions with audio. The dataset we used for this project is entirely collected by us, which are clipped from Taiwanese soap operas. We initially built the categorical model from the ground up with five emotions and found out that the accuracy (43.9%) was way lower than we expected, so we relabeled the whole dataset into two labels, and thus we could train a binary model, which reached a better accuracy (70.9%). We then used a well-performing CNN model on GitHub as the base model and trained a transferred model, but it didn't perform well, with only 25% accuracy on average.**

Given the low accuracy of our model with our own dataset, we have done several experiments to analyze our dataset and compare it with the popular existing datasets. We ended up with three major causes. The first one is that the existing datasets apply specific tones, the second one is that emotions are highly complex, and the third one is the visual difference of the spectrogram between our dataset and the Toronto emotional speech set (TESS), which is a popular dataset online.

I. Introduction

We wanted to build a model that could distinguish emotions through audio, and we recalled that CNN was one of the popular methods for dealing with audio, so we adopted it. Since the topic of 'speech emotion recognition' (SER) was quite broad, we wanted to focus on a specific realm, and thus we decided to focus on Taiwanese soap opera lines, because they are well known for their dramatic expressions. Yet there weren't any existing dataset that we could acquire, hence we built our own.

II. Methods

A. Dataset

Our dataset was collected from Taiwanese dramas 'Taiwan Tornado' (台灣龍捲風) and 'Family Reunion' (一家團圓) on Youtube. In each video, we clipped different 3 second fragments based on 5 emotions: happy, sad, angry, fearful (worried included) and neutral. Since our topic was about using sound to distinguish emotions, our data was stored as .wav format, which contained all of the original audio elements.

We had 2 kinds of labels, categorical and binary. Categorical labels were 5 emotions as previously mentioned, and for binary one, we labeled happy and neutral as 'neutral' (non-negative); sad, angry, and fearful as 'negative'.

B. Data preprocessing

Since our dataset was relatively small, we applied some data augmentation techniques: adding noise, stretching, shifting, and pitching.

To extract features from the data, we used tools including Zero Crossing Rate (ZCR), Chromagram STFT, Root Mean Square (RMS), Mel-Frequency Cepstral Coefficients (MFCC), and Mel Spectrogram.

1. Zero-Crossing Rate

   The Zero-Crossing Rate (ZCR) of an audio frame is the rate of sign changes of the signal during the frame. In other words, it is the number of times the signal changes value, from positive to negative and vice versa, divided by the length of the frame [1] .

2. Chroma STFT

   The Chroma value of audio basically represents the intensity of the twelve distinctive pitch classes that are used to study music. They can be employed to differentiate the pitch class profiles between audio signals. Chroma STFT uses short-term Fourier transformation to compute Chroma features. STFT represents information about the classification of pitch and signal structure. It depicts the spike with high values (as evident from the color bar next to the graph) in low values (dark regions) [2].

3. Root Mean Square

   The Root Mean Square computes the root-mean-square (RMS) value for each frame.

4. Mel-Frequency Cepstral Coefficients (MFCC)

   Mel frequency cepstral coefficients are compact representations of the spectrum that are typically used to automatically identify speech and it is also used as a primary feature in many research areas that include audio signals [2].

5. Mel Spectrogram

   A Mel spectrogram is the collaboration of the Mel scale and spectrogram where the Mel scale represents the non-linear transformation of the frequency scale. In this case, the audio signal is first broken down into smaller frames and a Hamming window is applied on each frame. Then Discrete Fourier Transform (DFT) is applied to switch from the time domain to the frequency domain [2].

These feature extraction functions were provided in the audio and music analysis library: librosa [3]. We used hstack to concatenate these features into a 1*162 array per audio clip.

C. *Model building*

We tried two different types of classification methods: categorical and binary.

For the categorical classification, we tried various types of model architecture including 2-D CNN model, 1-D CNN model, and Transfer learning in which 1-D CNN combined with the above-mentioned feature extraction methods was the prominent one.

Our proposed model is a sequential model consisting of three Conv1D layers followed by two dense layers among which the final layer is the output layer.

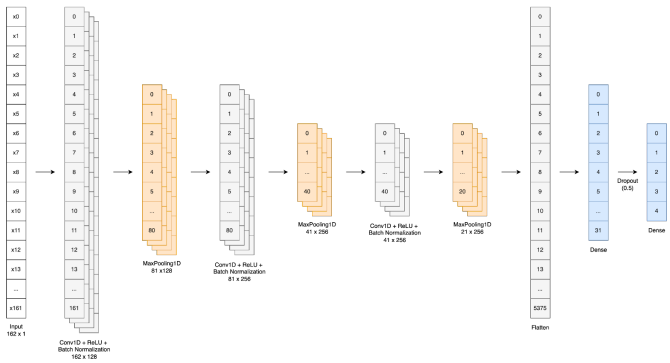

Fig. 1. Model architecture of our categorical model

For the binary classification, we adapted the 1-D CNN categorical model with some hyperparameters tweaked.

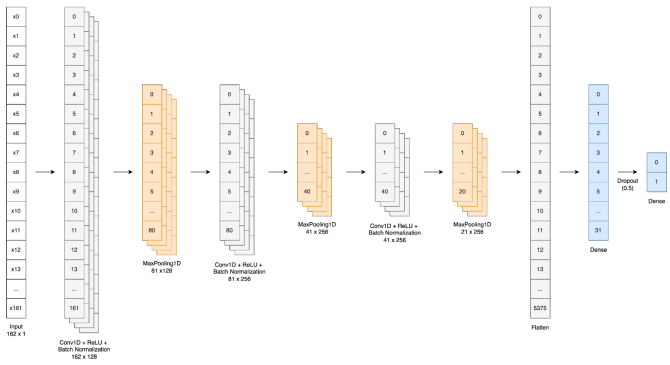

Fig. 2. Model architecture of our binary model

D. *Transfer learning*

Since there exist a lot of pre-trained models on the internet for SER, we wanted to see if these can help with our accuracy.

The two main qualities we wanted for the base model were popularity and simplicity. There were 130 public repositories on GitHub with the tag 'speech-emotion-recognition', so we looked into the first few ones, which are the most popular ones. Many of these popular projects applied advanced and complex libraries that we didn't have time and ability to dive into, especially those that focused on data preprocessing, since most of them required prior knowledge of the science of audio augmentation. Therefore we looked for the model that is the simplest so that we could have a full grasp of what we were doing, and we adopted MiteshPuthran's model [4] as our base model.

This base model, according to its documents, had more than 70% accuracy on 10 labels (five emotions and two genders combined). They also stated that it had 100% accuracy on gender classification. It consisted of six 1-D convolution layers, two dropout layers, and one flatten layer.

Layer (type)	Output Shape	Param #
conv1d_7 (Conv1D)	(None, 216, 128)	768
activation_8 (Activation)	(None, 216, 128)	0
conv1d_8 (Conv1D)	(None, 216, 128)	82048
activation_9 (Activation)	(None, 216, 128)	0
dropout_3 (Dropout)	(None, 216, 128)	0
max_pooling1d_2 (MaxPooling 1D)	(None, 27, 128)	0
conv1d_9 (Conv1D)	(None, 27, 128)	82048
activation_10 (Activation)	(None, 27, 128)	0
conv1d_10 (Conv1D)	(None, 27, 128)	82048
activation_11 (Activation)	(None, 27, 128)	0
conv1d_11 (Conv1D)	(None, 27, 128)	82048
activation_12 (Activation)	(None, 27, 128)	0
dropout_4 (Dropout)	(None, 27, 128)	0
conv1d_12 (Conv1D)	(None, 27, 128)	82048
activation_13 (Activation)	(None, 27, 128)	0
flatten_2 (Flatten)	(None, 3456)	0

Table 1. The structure of the base model

Data preprocessing and feature extraction for the base model are almost the same as our own model. It also adopts the library 'librosa' to load the audio and extract the features with MFCC.

We followed the data preprocessing procedures of the base model and replaced the last dense layer, which had 10 nodes for the 10 labels, with a new dense layer of five nodes and with ReLU as the activation function.

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 216, 1)]	0
sequential (Sequential)	(None, 3456)	411008
dense_1 (Dense)	(None, 5)	17285

Table 2. Structure of the entire transferred model, where 'sequential' is the base model

III. RESULTS

A. *Categorical*

The categorical model we trained achieved 43.9% accuracy on testing data.

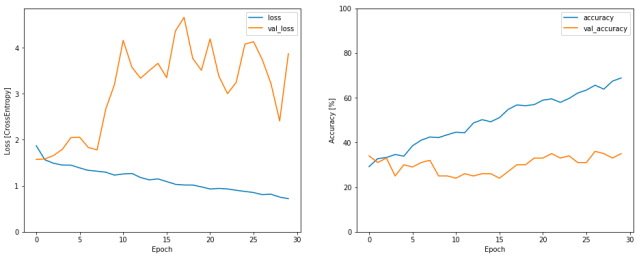

Fig. 3. Loss and accuracy curve of the categorical model

From the table below, we can see that "angry" and "neutral" emotions had the highest F1-score. The model identified these two emotions better.

Label	Precision	Recall	F1-score
Angry	0.50	0.73	0.59
Fearful	0.38	0.30	0.33
Happy	0.40	0.14	0.21
Neutral	0.63	0.83	0.72
Sad	0.53	0.37	0.43

Table 3. Classification report of the categorical model

B. *Binary*

The binary model we trained achieved 70.9% accuracy on testing data.

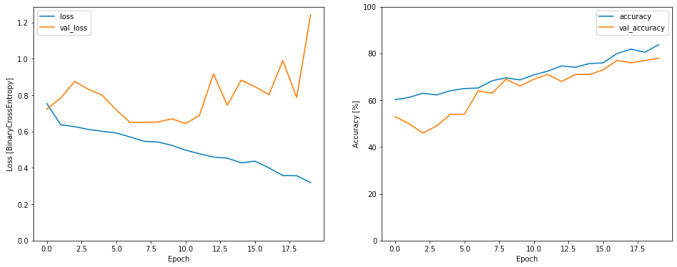

Fig. 4. Loss and accuracy curve of the categorical model

From the table below, we can see that data labeled "intense" had the highest F1-score. This echoes the results of the categorical classification, in which "angry" was one of the best-performing emotions.

Label	Precision	Recall	F1-score
Neutral	0.75	0.44	0.55
Intense	0.70	0.90	0.79

Table 4. Classification report of the binary model

C. *Transferred model*

The transferred model got merely 25% accuracy on validation data on average, which was underwhelming compared to the base model.

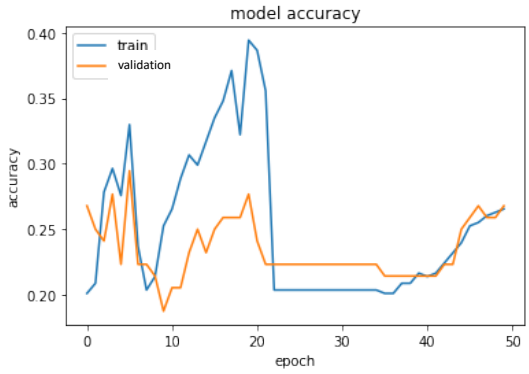

Fig. 5. The accuracy curve of the transferred model

Although in the figure above it seems like the accuracy is gradually increasing, we had actually tried out different hyperparameters, and the best it could do was around 25% for validation data. More epochs could help the training accuracy, but not for validation.

We also tried fine-tuning the model, and it did help, but as the figure below shows, it took a lot of epochs to get only a few improvements, so it basically lost the whole point of transfer learning, since the model had lost the base model's weights. Therefore we ruled out this model.

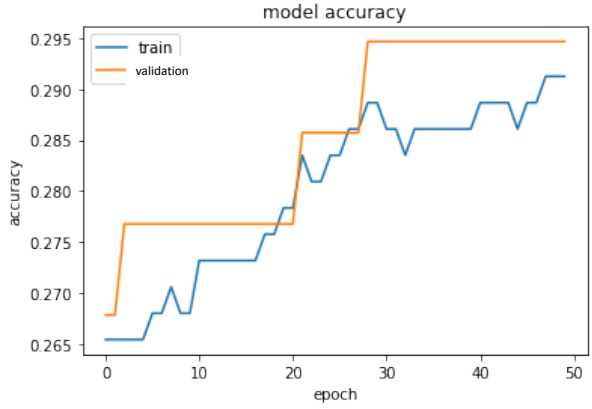

Fig. 6. The accuracy curve of the transferred model during fine-tuning

IV. DISCUSSION/CONCLUSION

We wanted to get an explanation on why our models didn't perform well whereas other models out there did, and below are the three:

1. Existing datasets have fixed tones for certain emotions

   We found out that most of the existing datasets had fixed tones for certain emotions, which doesn't apply in real life scenarios. We want to note that most of these datasets are built by paying actors/ actresses to do voice acting with designated emotions.

   Take TESS dataset for example, a, b, and c are three data that are all labeled 'happy', and we believe most people can easily pick up the specific

tone, but not the real ‘happiness’. We maintain that the well performing models, mostly trained with these existing datasets, are picking up the tones instead of the emotions. Yet in the real world, people don’t always use the same tone for the same emotion, and therefore our models can’t do the same with our own dataset. Although our data are from dramas, they are still pretty realistic.

2. Complexity of emotions

Furthermore, we also concluded that emotions are complex, and with only the audio, it is not that easy to distinguish emotions, even for humans. A is one of our collected data, and the original clip is from here. We believe that most people don’t feel happiness from the audio, but with the video, it is quite easy to get the answer. Then we realized that without facial expressions, the atmosphere, or context, it is relatively difficult to tell the real emotion.

3. Visible difference in spectrogram

Additionally, we wanted to see if the existing datasets were also difficult to distinguish for models. We then trained our CNN model again, but with the TESS dataset, and surprisingly, it got 98% accuracy. Therefore, we analyzed the difference between our dataset and the existing dataset TESS. To visualize the audio dataset, we plot the spectrogram by first calculating the short time fourier transform since we feed the spectrogram to the model.

We picked two emotions, neutral and fearful, as examples to compare the difference between our dataset and TESS dataset.

Figure 7 contains the spectrograms of nine neutral emotions from TESS and Figure 8 of nine fearful emotions also from TESS. We can easily distinguish the two emotions by spectrograms with our bare eyes since there are some unique features and regular patterns for different emotions.

Yet that is not the case for our dataset.

Figure 9 contains spectrograms of nine neutral emotions from our dataset and Figure 10 of nine fearful emotions also from our dataset. In our dataset, we can hardly distinguish the spectrogram between neutral and fearful emotion since there are no regular patterns and no unique features.

We picked fearful emotion from TESS and found out there exist some common features, which is the black line we framed in 9 spectrograms (Figure 11). We assumed that CNN can learn the features in the spectrogram so it can classify fearful emotion well.

For more visualization of different emotions from TESS and our dataset, we have made more comparisons, please refer to here [5].

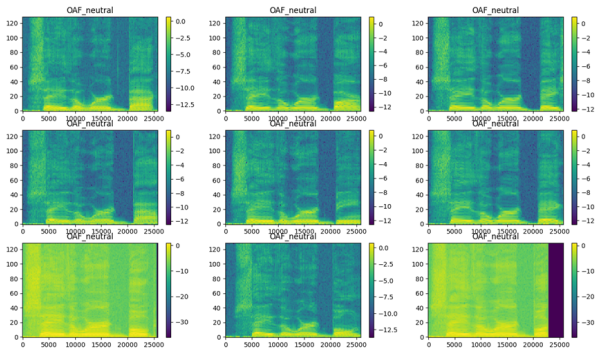

Fig. 7. Spectrograms of neutral emotions from TESS

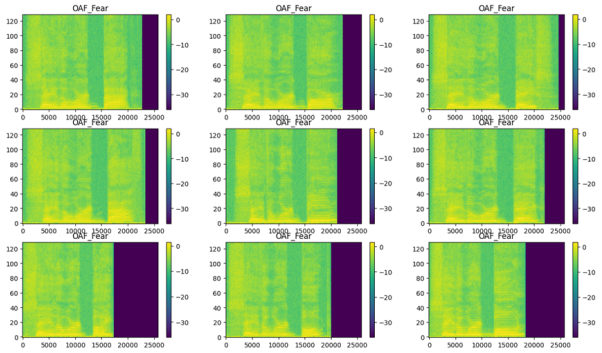

Fig. 8. Spectrograms of fearful emotions from TESS

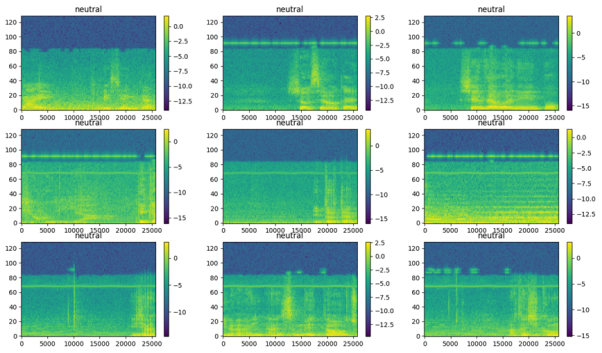

Fig. 9. Spectrograms of neutral emotions from our dataset

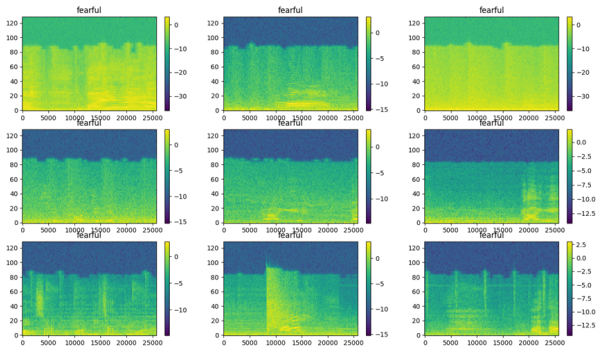

Fig. 10. Spectrograms of fearful emotions from our dataset

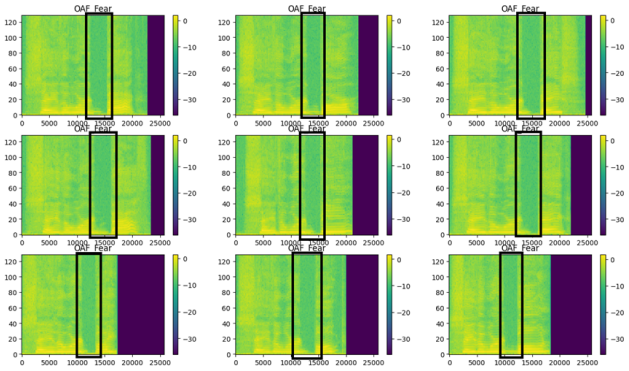

Fig. 11. Spectrograms of fearful emotions from TESS have common features

V. AUTHOR CONTRIBUTION STATEMENTS

- Lin, Yi-Ting (林奕廷) 20%: Categorical and binary model, data preprocessing
- Hsu, Chun (許淳) 20%: Data relabeling
- Hung, Sheng-Hsiang (洪聖祥) 20%: Categorical model, data analysis
- Zhang, Hong-Qi (張紘齊) 20%: Data collection, experiment on existing datasets
- Yeh, Ssu-Yung (葉思永) 20%: Transfer learning, proposal & report

VI. REFERENCES

1. Theodoros Giannakopoulos, Aggelos Pikrakis, “Introduction to Audio Analysis ”, Chapter 4 - Audio Features, Pages 59-103, Academic Press, 2014.
2. Das Joy, Ghosh Arka, Pal Abhijit, Dutta Sumit, Chakrabarty Amitabha, “Urban Sound Classification Using Convolutional Neural Network and Long Short Term Memory Based on Multiple Features”, 2020 Fourth International Conference On Intelligent Computing in Data Sciences (ICDS), DOI:10.1109/ICDS50568.2020.9268723, November 2020.
3. “librosa: audio and music processing in Python ” [Online] Available: https://librosa.org/
4. https://github.com/MiteshPuthran/Speech-Emotion-Analyzer
5. https://github.com/Sunnyhong0326/ML-analyze/blob/main/analyze.ipynb