Data Analytics Foundations ECE 9063
Human Emotion Classification Using Audio and Text Models

Sung Lin (Arthur) Chang, 251293243, *schan744@uwo.ca*
Tianze Zhang, 251322742, *tzhan648@uwo.ca*
Mingxin Bai, 251313990, *mbai25@uwo.ca*
Chaoyu Wang, 251316374, *cwan287@uwo.ca*
Yi Fei, 251279131, *yfei55@uwo.ca*

I. ABSTRACT

In this project, we plan to combine audio and language classification model through another Feedforward neural network in hopes of it capturing both audio mood expression and language sentiment and achieve a better performing model. We classified texts and audio samples into the five emotion categories (joy, sadness, fear, anger and neutral). For the language model we will use Bidirectional Encoder Representations from Transformers (BERT), a transfer learning model. As for the audio model, we will use CNN with 5 augmented features to better capture the tonal expression and frequency variation. For language and audio model (upstream model), we received respective average F-1 score of 83% and 66%, however, for the feedforward neural network (downstream model) the performance was poor due to:

- Downstream data has different data distribution than upstream model.
- Minimal downstream training data.
- Unclear classification boundary for audio and language for mood expression.

II. INTRODUCTION

In this project, we are aiming to use machine learning tools to better identify human emotion from speech and text data. More specifically, we attempt to predict different emotions, namely, joy, sadness, fear, anger and neutral from different sets of human audio and text data.

The reason we chose this direction is because many sectors in society need a tool like this to identify human emotions. For example, when doing customer satisfaction surveys and meetings with an important client, the client may not be speaking the truth about his or her feelings over the service or product, therefore, a tool like this could help the business to better understand their service or product. Additionally, the tool we are developing can be also used in medical therapeutic treatment, police interrogation, user experience feedback and virtual reality live emotion feedback.

III. BACKGROUND

Human emotions are complicated. An individual may express one mood in a language context and express a different mood in their tonal expression. This results in an unclear boundary of the actual expression of the individual. Our model consists of an audio classification model and language classification as upstream model. The audio model consists of a CNN of five audio features and the language model consists of BERT Cased with fine-tuning. We combine the individual model's output weight into a Feedforward neural network as the downstream model to generate predictions. This will allow the model to decide the audio and language weights that should be distributed into the decision of the mood classification. The metrics we use will be accuracy and F-1.

In the past, multitude of models were specifically designed to better classify emotion on either the context of the language or their mood, however, few of them tried to target both. This may be due to the lack of datasets currently available to the open-source community. Some of the most notable datasets are MOSI and MOSEI. These datasets consist of audio, language, and video as model inputs. Unfortunately, two of the major datasets (MOSI and MOSEI) have been closed to the public by CMU. To address the lack of datasets needed to train the downstream model, we created a new dataset with 100 samples consisting of male and female speakers of Asianic accents with varying contrasting expressions of language and tonal differences. As for the upstream model, the datasets will be further explained in the methodology section.

Most of the well-known models used to predict similar mood classification all involve the usage of all three inputs (audio, language, and video), resulting in an incomparable result to our classification task. In our task, we will have 5 moods (joy, sadness, fear, anger and neutral) of varying audio and language combination as input and 1 single mood output. The combination can consist of different emotional label for language and audio data, but it will only have one mood label.

IV. METHODOLOGY

We will first discuss the upstream Language and Audio Model before diving into the downstream Feedforward neural network model.

A. Audio Model

The Audio Model uses four different Kaggle datasets for training. These datasets contain recordings of different emotions (Joy, sadness, fear, anger and neutral). These data show below:

- Crowd-sourced Emotional Multimodal Actors Dataset (CREMA-D)[1]
- Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS)[2]
- Surrey Audio-Visual Expressed Emotion (SAVEE)[3]
- Toronto emotional speech set (TESS)[4]

1) Data Preparation:

In the first step, we need simple processing on four different datasets. Because the four datasets have different labels, but all contain the five labels we need. We will use a loop method to save the required labels and paths in the four datasets to four different new datasets. After that, save the above four datasets into a new CSV file. Subsequent data enhancement and feature extraction will be processed using this prepared new dataset.

2) Data Visualization:

Before we perform data enhancement and feature extraction on the new data, we will use the drawing method to display the labels in the dataset and the amount of data contained in different labels in different colours. This graph can help us further understand our data.

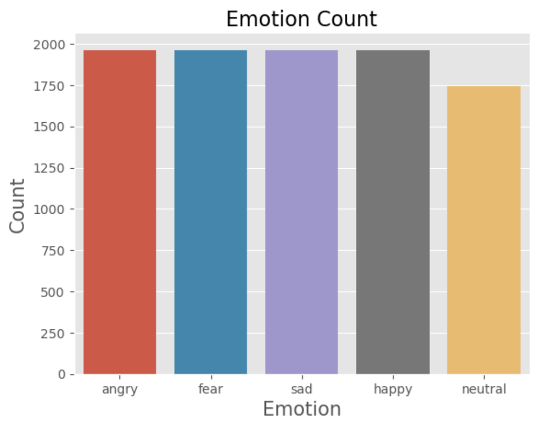


Fig. 1. Count of emotions

3) Data Augmentation:

Data augmentation is a technique to prevent models from being over-trained. Since our data is audio, we will augment the data by injecting noise, shifting time, and changing pitch and speed. To check which enhancement technique is more suitable for our dataset, the different data enhancements described above will be performed on the same segment of speech. When using these methods, the labels of the original training samples will remain unchanged. We found that injecting noise into speech, time-shifting, and changing pitch and speed can effectively enhance data, because it mimics the environmental factors that can affect the frequency and amplitude of the recording. These augmentations make our model invariant to these perturbations and make it more generalizable. The augmented data can effectively prevent the training model from over-fitting. Finally, the above data augmentation method will be used on the dataset after feature extraction.

4) Feature Extraction:

Feature extraction is the most important part before creating a model. But our model can't handle our audio directly. Therefore, we need to convert the audio data into a format the model can understand and perform feature extraction. We selected the following five features for extraction:

- Zero Crossing Rate
- Color spectrum
- Mel Frequency Cepstral Coefficients
- Mel Spectrogram
- Root Mean Square value

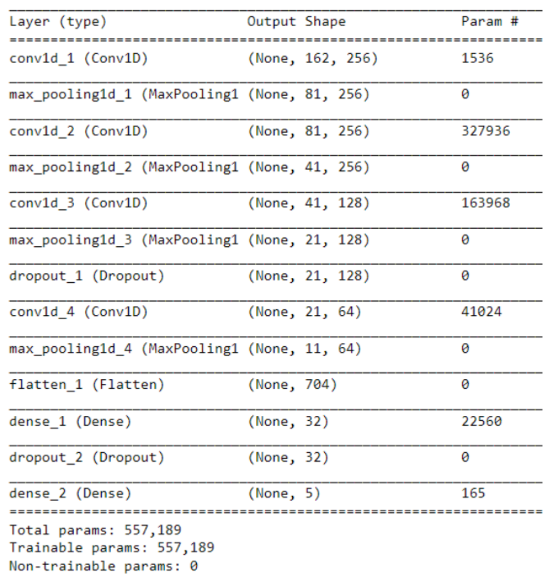
By the same method as above, the audio data can be converted into 162-bit features that can be used by the model and labelled accordingly. Then apply the data augmentation method mentioned in the previous step to the data. We extract the five features one by one and stack them horizontally into the same array. We use noise injection as one group and stretching and pitching as another group. Afterwards, the two datasets are subjected to feature extraction and vertically stacked. Finally, we converted each audio into three datasets without data enhancement, noise injection enhancement and stretching and pitching enhancement. Finally, we have usable data which contains 162-bit features and sentiment labels.

5) Data processing:

So far, we have extracted the data, and now we need to normalize and split the data for training and testing. Since this is a multi-class classification problem, we need to convert the label encoding into a One-Hot encoding. After we processed the data, we divided it into a 75% training set and a 25% test set. Next, we will normalize the two datasets using the Standard Scaler. Afterwards, the two datasets' data shapes are (27364, 162) and (7047, 162), but our model cannot use this data shape. Then change the shape of the two datasets to (27364, 162, 1) and (7047, 162, 1) for the model to use. So far, our data has been processed.

6) Modeling constructor:

In this subsection, we propose our neural network specially designed for audio modelling tasks. We refer to the mainstream-related papers and models to adopt a similar CNN structure. Our model consists of four convolutional layers, four pooling layers and two fully connected layers. Hyperparameter sensitivity experiments are conducted to help us find the best combination composed of 4 groups of conv1d and max pooling layers with Relu as activation function to improve performance. The top



```
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_1 (Conv1D)            (None, 162, 256)          1536
_________________________________________________________________
max_pooling1d_1 (MaxPooling1 (None, 81, 256)           0
_________________________________________________________________
conv1d_2 (Conv1D)            (None, 81, 256)           327936
_________________________________________________________________
max_pooling1d_2 (MaxPooling1 (None, 41, 256)           0
_________________________________________________________________
conv1d_3 (Conv1D)            (None, 41, 128)           163968
_________________________________________________________________
max_pooling1d_3 (MaxPooling1 (None, 21, 128)           0
_________________________________________________________________
dropout_1 (Dropout)          (None, 21, 128)           0
_________________________________________________________________
conv1d_4 (Conv1D)            (None, 21, 64)            41024
_________________________________________________________________
max_pooling1d_4 (MaxPooling1 (None, 11, 64)            0
_________________________________________________________________
flatten_1 (Flatten)          (None, 704)               0
_________________________________________________________________
dense_1 (Dense)              (None, 32)                22560
_________________________________________________________________
dropout_2 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_2 (Dense)              (None, 5)                 165
=================================================================
Total params: 557,189
Trainable params: 557,189
Non-trainable params: 0
```

Fig. 2. Audio Model Hyperparameters

layers of our network are two fully connected layers, the first one is followed by Relu, and the other is Softmax. Softmax converts continuous data into discrete probabilities to better cooperate with our loss function – Categorical Cross-Entropy to fulfill single-label classification. We use the following formula to take the output probabilities (P) and measure the distance from the truth values. The final objective is to make that distance as little as possible (minimizing the Categorical Cross-Entropy loss).

$$Loss = -\sum_{i=1}^{outputsize} y_i \cdot \log \hat{y}_i \tag{1}$$

In practice, we learned that to increase a better ability to generalization, we need to make sure that the size of the training sample, N, satisfies the following condition:

$$N = O(\frac{w}{e}) \tag{2}$$

Where W is the total number of free parameters in our network, e represents the forecasting errors, and O(·) stands for "the order of quantity enclosed within". Because our model has 557,189 total parameters only in audio part and the training samples are way fewer compared to that quantity, we added two layers of dropout to prevent overfitting, one is between the third block and the fourth block of convolution and pooling layers, the other one is between the first fully connected layer and the second one.

The image above shows our model's workflow, and the image below shows our model's parameters in detail. The input of our model is a victor of 162*1 which is the output of the feature extraction after data argumentation, the output is the highest one among five possibilities that indicate how likely the input belongs to a specific

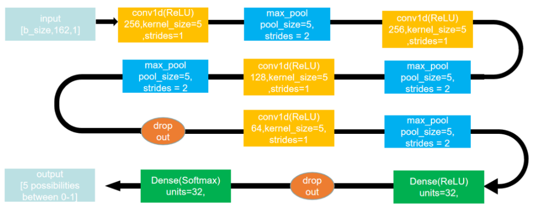


Fig. 3. Audio Model workflow

emotion. We choose 'Adam' as our model's optimizer and accuracy and f-1 score as the primary metrics. The learning rate is not fixed, this is because too high a learning rate can cause the optimizer to exceed the optimal value, while too low a learning rate can lead to too long a training time. It is difficult to find a static, highly effective and constant learning rate. Instead, we use ReduceLROnPlateau as callbacks to monitor loss. We set the model to ignore the first 2 epochs that do not improve the accuracy, and reduce the learning rate (by 40 percent) only if the metric still does not improve after the 3rd epoch while we set the minimum equals 1e-7 manually.

B. Language Model

1) BERT (Bidirectional Encoder Representation from Transformers): BERT can be used to solve problems such as Sentiment Analysis and Text Summarization. All of those require a need for language understanding. We will use the pretrain BERT to be fine tuned learn our classification problem.

The Language Model uses three different datasets for training. These datasets contain text of different emotions (Joy, sadness, fear, anger and neutral). These data show below:

- Dailydialog [5]
- Emotion-stimulus [6]
- Isear [7]

2) Data Preparation: The dataset is a mixture of conversations, posts, and logs from social media and online chats. They have been labelled as 5 emotions and were already cleaned as text formatting. Therefore, for our preparation, we first cleaned the dataset through removal all Nans and dropped samples with length greater than five sentences (because our dataset only require audio of 3-5 seconds, which is less than 5 sentences of data). Afterward, for BERT to distinguish different emotions, we encoded 5 emotions. Joy is 0, sadness is 1, fear is 2, anger is 3, and neutral is 4.

3) BERT Input Representation:

BERT uses bi-directional parallel input to input the whole sentence into the model instead of inputting words one after another, which can fully utilize the performance of GPU and greatly improve the running efficiency of the model. At the same time, since the parallel input will lead to the loss of word position information in the text, BERT model needs to add an additional position

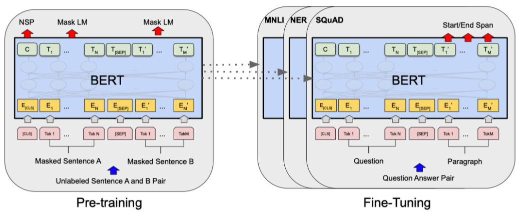


Fig. 4. Overall pre-training and fine-tuning procedures

encoding input to ensure that the position information will not be lost.

For different downstream tasks, BERT introduces two special symbols in the input layer, one is [CLS], which is placed at the top of the whole input text sequence, and the other is [SEP], which is placed in the middle of two sentences and at the end of the second sentence for tasks where the input is a sentence pair. The hidden state of [CLS] corresponding to the final output is used as a representation of the classification task, indicating the category of the input sentences in the case of single-sentence classification, and indicating that the two sentences are related/unrelated, similar meaning/opposite meaning in the case of sentence-pair classification.

The input vector of BERT consists of three parts, namely token embedding, segment embedding and position Embedding. Tokenization is performed before the subwords are embedded in token embedding. The WordPiece embedding method is used here to further split each word into more generic sub-words, which can avoid the problem of oversized vocabularies. Token embedding layer converts each segment into a fixed dimensional vector. Segment embedding is used to distinguish two sentences with only two values of 0 and 1. Sentence A is coded as 0, sentence B is coded as 1. If it is a single-sentence task, all codes are 0. Position embedding represents the position of each word in the sentence, and the embedding of the same word is different in different positions. Three embedding size are (batch_size, seq_length, hidden_size), and finally the three embeddings are summed up by element value, which represents the input of BERT encoding layer.

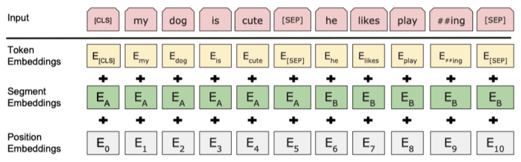

Input
[CLS] my dog is cute [SEP] he likes play ##ing [SEP]
Token Embeddings
E[CLS] Emy Edog Eis Ecute E[SEP] Ehe Elikes Eplay E##ing E[SEP]
Segment Embeddings
EA EA EA EA EA EA EB EB EB EB EB
Position Embeddings
E0 E1 E2 E3 E4 E5 E6 E7 E8 E9 E10

Fig. 5. Bert input representation

4) Pretraining:

The goal of pretraining is to let BERT learn what the language is and what is the context. BERT learns language by training two unsupervised tasks simultaneously, which are Masked Language Model (MLM) and Next Sentence Prediction (NSP) respectively.

Masked Language Model (MLM): BERT takes in sentences with random words replaced with masks. The goal is to output these mask tokens (each token is a word). In simple terms, it is to randomly dig out several blanks in the sentence and let the model predict the original words of these blanks. The strategy adopted in the pre-training of BERT is to randomly select 15% of the words in the sentence for processing, where these 15% words are treated differently in the ratio of 80%-10%-10%: 80% of the words are replaced with [MASK] to indicate the part to be predicted, 10% are replaced with random words, and 10% keep the original words unchanged. This training process allows the model to learn the representation of the distribution of words in context, and in addition replacing with 1.5% random (10% of 15%) does not impair the model's comprehension. This filling-the-blank process helps BERT to understand the bi-directional context with a sentence.

Next Sentence Prediction (NSP): BERT takes in two sentences, and BERT determines if the second sentence follows the first sentence. This helps BERT understand context across different sentences.

For every word, we get the token embedding from the pre-trained word piece embeddings add the position and segment embeddings to account for the ordering of the input. These are then passed into BERT, which under the hood is a stack of transformer encoders, and outputs word vectors for MLM and a binary value for NSP. The word vectors are then converted into a distribution to train using cross entropy loss. Once training is complete, BERT has some notion of language.

With the use of MLM and NSP, BERT could get a good understanding of the language and context.

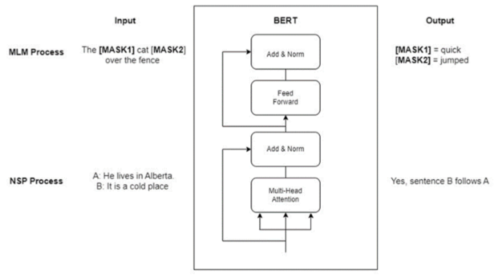


Fig. 6. BERT pre-training process

5) Fine Tuning:

The goal of fine tuning is to train BERT on very specific NLP tasks. Since Transformer uses a self-attention mechanism, it allows the BERT model to be applied to a wide variety of downstream tasks. In our case, we need to replace fully connected output layers of the network with a fresh set of output layers that can basically output the classification for the text. Then we can perform supervised training using our dataset. It would not take long because only output parameters are learned from scratch. The rest of the model parameters are just slightly fine-tuned. As a result, training time is fast.

C. Downstream merged model

1) Data Preparation:

As aforementioned in the background, the datasets that contain both audio and language context (MOSI and MOSEI) are no longer available to the public and are not readily available in our research, we will resort to creating our own dataset from scratch. We created 100 samples of evenly distributed (20 samples for each mood) of 3-5 seconds audio with meaningful sentences that correspond to each mood. For each mood, the 20 sample of data breaks down to 10 samples of where audio and sentences are aligned in mood, and 10 samples of them having contrasting mood expression. The reason why we use contrasting mood input for audio and sentences is because the mood a person is feeling does not always align between their spoken words and tonal output.

The two examples can be seen below to better understand their differences.

- Audio and sentences align in mood (Joy): (Tone: excited) It is finally winter break!
- Audio and sentences does not align in mood (Joy) (Audio: neutral, Language: excited): (Tone: calm and quiet, because the individual is in a library) I can't believe I aced the final!

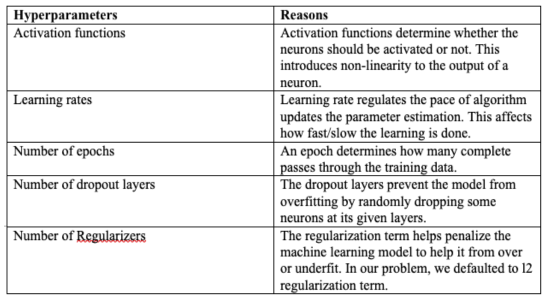
After we have created the dataset, we then split the dataset into 80:20 training: testing set. The audio and sentences data are then fed through the upstream BERT and audio model to get their corresponding last layer output (both have relu applied). From BERT, we will get 5 parameters matching the 5 mood output specified and from the audio model we have 15 parameters. The 15 parameters (5*3) come from 3 different types of audios being augmented from a single audio sample. The three types of audios are: original, noise injected, and stretched audio. For further explanation, please check out data processing in the audio section of methodology. In combination, there are a total of 20 input data that comes from the output of the BERT and audio model.

2) Feedforward neural network Model:

The Feedforward neural network model is the simplest and first model ever created for the artificial neural network. It works by passing the output from one neuron to another with activation functions, weights, and biases included. The weights and biases are then updated by gradient descent to finish one epoch of training. It is because of this property that allows it to be a universal approximator that can generalize to any trainable regression and classification problem. Because of this property and our need to test the model's combination on a simpler model to see if it is even feasible, we selected this algorithm.

To start, we initialized a baseline model with one input, hidden, and output layer. The activation function used is sigmoid for the hidden layer and SoftMax for the final layer. However, the model did not perform well as we faced an exploding/vanishing gradient problem and the validation loss shot up.

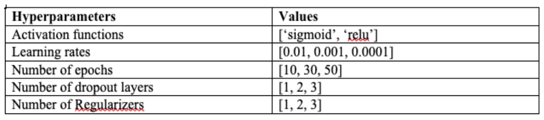
To better our model, we selected the number of layers and neurons to be defaulted at 5 layers and (20, 15, 10, 7, and 5) neurons. This is because the general guideline from Applications of Deep Neural Networks with Keras [8], where a model shouldn't exceed 5 layers unless it is extremely complex. The number of neurons is also a guidelines "The number of nodes is generally set at a combination of   of input neurons + output neurons." We did not put these hyper-parameters as tunable parameters primarily due to the time complexity needed for grid-search. We then used the following hyper-parameter to use with grid-search: activation functions, learning rate, number of epochs, number of dropout layers, and number of regularizers. The reason why we have selected the following to tune are as follows:



Hyperparameters	Reasons
Activation functions	Activation functions determine whether the neurons should be activated or not. This introduces non-linearity to the output of a neuron.
Learning rates	Learning rate regulates the pace of algorithm updates the parameter estimation. This affects how fast/slow the learning is done.
Number of epochs	An epoch determines how many complete passes through the training data.
Number of dropout layers	The dropout layers prevent the model from overfitting by randomly dropping some neurons at its given layers.
Number of Regularizers	The regularization term helps penalize the machine learning model to help it from over or underfit. In our problem, we defaulted to l2 regularization term.

Fig. 7. Reason for selecting above mentioned hyper-parameters to tune

In the figure below, we implemented grid-search to search the following hyper-parameter values. The grid-search found the following terms to be the best performing values: activations with relu on each layer prior to the last layer, 0.001 learning rate, 50 epochs, 1 dropout layer, and 1 l2 regularization term.



Hyperparameters	Values
Activation functions	['sigmoid', 'relu']
Learning rates	[0.01, 0.001, 0.0001]
Number of epochs	[10, 30, 50]
Number of dropout layers	[1, 2, 3]
Number of Regularizers	[1, 2, 3]

Fig. 8. Best hyper-parameter values base on Grid-search

V. EVALUATION/RESULTS

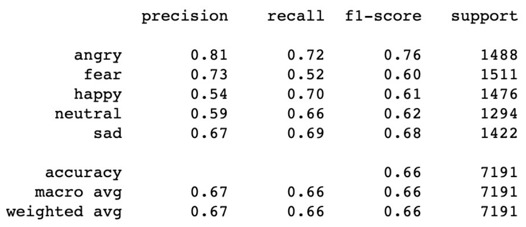
The language, audio, and merged models' evaluation/result will be discussed below:

A. Audio model evaluation

In order to evaluate the performance of the model scientifically and effectively, besides f1-score, we also choose the accuracy as one of the evaluation indexes based on a balanced dataset, in line with the mainstream works in the field of audio emotion forecasting. The calculation formula of accuracy is as follows.

$$accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{3}$$

From Fig. 9 and 10, We can see our model is more accurate in predicting angry emotions, reaching 81% precision, 72% recall and 76% f1-score, and the confusion matrix also proves what we achieve. It is because that audio files of specific angry emotion is quite different from other emotions audios in the forms of data representation like speaking speed or tone.



```
              precision    recall  f1-score   support

       angry       0.81      0.72      0.76      1488
        fear       0.73      0.52      0.60      1511
       happy       0.54      0.70      0.61      1476
     neutral       0.59      0.66      0.62      1294
         sad       0.67      0.69      0.68      1422

    accuracy                           0.66      7191
   macro avg       0.67      0.66      0.66      7191
weighted avg       0.67      0.66      0.66      7191
```

Fig. 9. Evaluation on different emotions

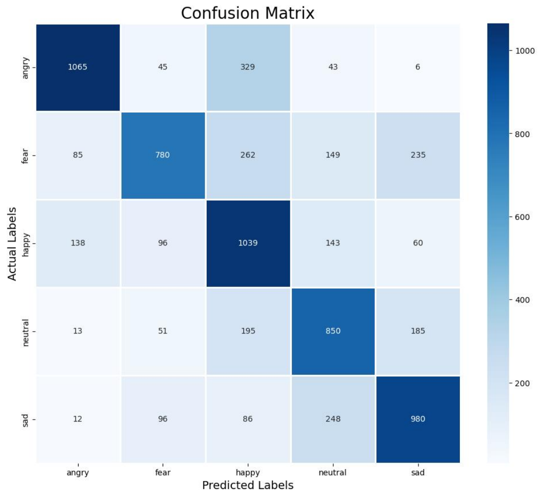


Fig. 10. Confusion matrix of our audio model

It is evident from Fig. 11 that accuracy of our test dataset is 65%, which is better than the most of related works. Still we can improve the performance by going deeper in the model, applying more augmentation techniques, extracting other valuable features or stacking them using various formats in our future work.

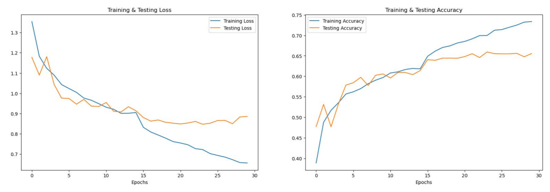


Fig. 11. Accuracy Loss of our audio model

B. Language model evaluation

To evaluate the pre-trained BERT model, we use f1-score to measure accuracy, because F1-score gives consideration of both precision and recall. The formula of how f1-score gives consideration to precision and recall is shown as following:

$$f1 - score = 2 * (\frac{precision * recall}{precision + recall}) \tag{4}$$

Based on the f1-score, which is shown below for each emotion and weighted average, we can see that, we got higher score when classify the large-fluctuated emotion such as joy and fear, which is reasonable because such large-fluctuated emotion contains more distinct differences among all moods. While for other moods, namely, sadness, anger, neutral, we got a relatively lower score, which is around 82%. The overall weighted f1-score is 83% with 3393 data. A table of pre-trained BERT Accuracy is shown as below:

	F1-score	Support
Joy	0.86	707
Sadness	0.82	676
Fear	0.86	679
Anger	0.81	638
Neutral	0.81	638
	Overall Weighted avg	Support
Overall	0.83	3393

Table:1 Evaluation Table of BERT among all features

C. Downstream merged model

1) Observations of data output prior to the downstream model: After the data were fed through the upstream models (BERT and audio), we checked the accuracy for training and validation dataset for both outputs to see what will be fed into the downstream model. The accuracy for BERT is around 55 percent and for audio model is around 22 percent. Initially, we were concerned on the performance of the upstream model, but then we realized the low accuracy were a result of two problems:

Different data distribution: The training data we had for the upstream model was significantly different than the ones we have created. For example, the audio data used to train were predominantly actors of North American accents and heir recordings are much more dramatic than our version. In contrast, we had Asianic accents, and our mood expressions were a lot flatter. With our audio model taking consideration of frequency range and tonal variation, it is clear why the model may not perform well on our dataset.

Contrasting label: As mentioned above, our dataset includes samples that have differing labels for audio and sentence. For example, the audio data alone can have a sadness label while sounding happy, because the language label might be the determining factor of why the audio, sentence combination is labeled as sadness. 50 of our 100 samples are labeled in such fashion.

2) Downstream model Evaluation: Although we were worried about the issue of "garbage in, garbage out." We still created a downstream model, because we thought the problem may be predominately caused by contrasting label and it may be resolved by combining the output for language and audio model.

Using the fine-tuned value described above, we resulted in the accuracy and loss below. The accuracy of 38 percent is relatively low; however, this result makes sense due to the issues mentioned above. Another issue that was observed is the issue of unclear classification decision boundary. For example, given an audio model predicts a sample to be "joy" with 80 percent confidence and the language model predicts it as "sadness" with 75 percent confidence. It is true a model should be able to draw decision boundary based on the confidence of both models, however, a major issue here is the amount of incorrect labeled sample (even without contrasting label) provided by the upstream model. As a result, the model performed poorly despite its possibility to learn.

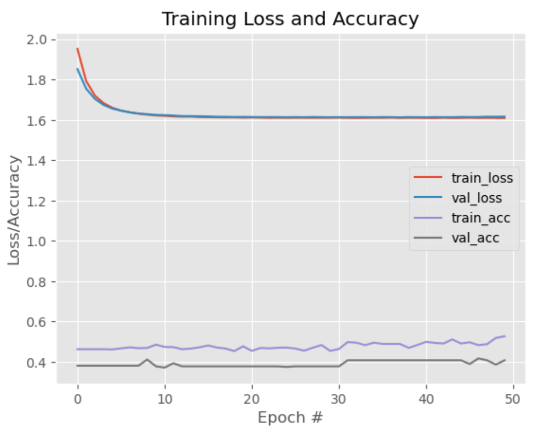


Fig. 12. Overall pre-training and fine-tuning procedures

VI. CONCLUSIONS

Individually, the audio and language models (upstream models) both performed well in their respective tasks, with their F-1 score being 83 % and 66%. However, when the Feedforward neural network (downstream model) was was incapable of capturing both model's meaning due to differing data distribution from upstream model, minimal downstream training data (only 100 samples), and unclear classification boundary as described above. In the future, when these three problems are addressed, the downstream model will likely perform much better than the current iteration. As for the upstream models, a wider data distribution and more features could be augmented in the audio model for a more comprehensive capture.

Once the merged model become feasible, we can use them as a key algorithm in many service industries like phone customer service. With our model, they can redirect customers based on their most likely emotions. Angry customers can be redirected to the complaint center, happy customers to sales and sad sounding customers to the support center, or they can just use the emotions as the service rating feedback.

References

[1] Cao, H., Cooper, D. G., Keutmann, M. K., Gur, R. C., Nenkova, A., Verma, R. (2014). Crema-d: Crowd-sourced emotional multimodal actors dataset. IEEE transactions on affective computing, 5(4), 377-390.

[2] Steven R. Livingstone, &amp; Frank A. Russo. (2019). RAVDESS Emotional speech audio [Data set]. Kaggle. https://doi.org/10.34740/KAGGLE/DSV/256618

[3] Haq, S., & Jackson, P. (2010). Machine Audition: Principles, Algorithms and Systems, chapter Multimodal Emotion Recognition. IGI Global, Hershey PA, 398-423.

[4] Dupuis, K., & Pichora-Fuller, M. K. (2010). Toronto emotional speech set (tess)-younger talker_happy.

[5] Yanran Li, Hui Su, Xiaoyu Shen, Wenjie Li, Ziqiang Cao, and Shuzi Niu. DailyDialog: A Manually Labelled Multi-turn Dialogue Dataset. IJCNLP 2017. [pdf] [arXiv] [dataset]

[6] Diman Ghazi, Diana Inkpen & Stan Szpakowicz (2015). "Detecting Emotion Stimuli in Emotion-Bearing Sentences". Proceedings of the 16th International Conference on Intelligent Text Processing and Computational Linguistics (CICLing 2015), Cairo, Egypt.

[7] Scherer, K.R. and Wallbotth, G. (1994) Evidence for Universality and Cultural Var-iation of Differential Emotion Response Patterning. Journal of Personality and So-cial Psychology, 66, 310-328. https://doi.org/10.1037/0022-3514.66.2.310

[8] Machine Learning (cs.LG), Artificial Intelligence (cs.AI), FOS: Computer and information sciences, FOS: Computer and information sciences, I.2.