Pre-Trained Speech Encoders for Emotion Recognition in Audio Data

Lucas Sha / lcsha , Christian Johann Martinez / cmarti88, Pir Servan Tutsi / ptutsi, Yihong Liu / yliu622

Brown University Department of Computer Science

1. Introduction

Robust speech emotion recognition (SER) models are an important technology. Such models currently have applications ranging from advanced smartphone assistants to virtual human simulations. However, people express their emotions very differently according to their demographic background—i.e ethnicity, personality type, gender, age. As a result, it can be difficult for SER models to generalize their comprehension of the nuances of emotional expression across a diverse group of people.

This project draws on the research paper *Personalized Adaptation with Pre-Trained Speech Encoders for Continuous Emotion Recognition* by authors Minh Tran and Yufeng Yin and supervised by Mohammad Soleymani. We chose this paper because the content of the work and the authors' intentions resonated with us. Human-computer interaction becomes an increasingly important area of research as artificial intelligence is knit into our day-to-day lives. Ensuring that artificial intelligence can understand the full diversity of human expression and behavior is critical to ensuring that it is beneficial and accessible to all.

The authors propose two techniques to surpass current SER model performance benchmarks on the MSP-Podcast dataset, as well as generalize better to unknown test speakers. Both techniques are either unsupervised, or make use of self-supervised pre-training processes built into the base model they use. The downstream fine-tuning process is supervised regression prediction.

(1) **Personalized Adaptive Pre-Training (PAPT)** is a variant of task-adaptive pre-training, wherein the pre-training process of a model is continued on new data. The authors propose additional feature engineering aspects to allow the model they work with to learn a personalized representation of speech.
(2) **Personalized Label Distribution Calibration Shift (PLDC)** is a method to manage label distribution shift. Label distribution shift occurs when models are evaluated on unfamiliar data with different predicted labels than those seen in training. PLDC corrects label distribution shift by giving the model a reference point to speakers with similar features seen in training to improve its inference.

Our project does not manage to fully re-implement the paper. We tested the feature engineering and architecture proposed by the authors, but pre-training proved unfeasible to re-implement for a variety of reasons. We did not implement PLDC. We had set up much of the necessary code infrastructure to do so in our preprocessing. However, we had an unfortunate last minute realization that the type of data we are working with is not compatible

with this method, and did not know how to create a substitute on our own. The limitations of our project scope and the pivots we had to make over its course will be covered extensively in **Challenges**. Irrespective of the difficulties, completing this project was an important learning experience and we hope the results will still be of some interest.

2. Methodology

a. Dataset

We use the dataset **CREMA-D** (Crowd-sourced Emotional Multimodal Actors Dataset). The dataset contains 7442 audio utterances split across 91 speakers. SpeakerID for the speakers are unique integers in the range [1001, 1091]. Utterances last for a few seconds each. Each clip consists of a speaker speaking one of 12 sentences. There are 6 categorical emotional labels for the data, and 4 intensity labels. All clip labeling is performed by crowd-sourced workers.

Sentences

- It's eleven o'clock (IEO).
- That is exactly what happened (TIE)
- I'm on my way to the meeting (IOM)
- I wonder what this is about (IWW)
- The airplane is almost full (TAI)
- Maybe tomorrow it will be cold (MTI)
- I would like a new alarm clock (IWL)
- I think I have a doctor's appointment (ITH)
- Don't forget a jacket (DFA)
- I think I've seen this before (ITS)
- The surface is slick (TSI)
- We'll stop in a couple of minutes (WSI)

Emotional Labels

- Happy (HAP)
- Sad (SAD)
- Fearful (FEA)
- Neutral (NEU)
- Angry (ANG)
- Disgust (DIS)

Intensity Labels

- Low (LO)

- Medium (MD)
- High (HI)
- Unknown (XX)

The dataset may be downloaded online. There is technically a pre-built TensorFlow loader for the dataset. However, we wanted to gain confidence in pipelining data from start to finish, especially since none of us had worked with audio data before. The raw wav files saved to one's computer after downloading are in the format 'SpeakerID_Sentence_EmotionLabel_IntensityLabel'. An example file name would thus be: 1001_IEO_DIS_LO.wav.

We used some basic visualization tools to understand the distribution of the raw audio data and speaker characteristics. The distribution for length of the audio samples in frames is given below. The audio sample rate is 22.05 kHz. So, the average audio sample duration is 40684.14/22050 = 1.845 seconds.

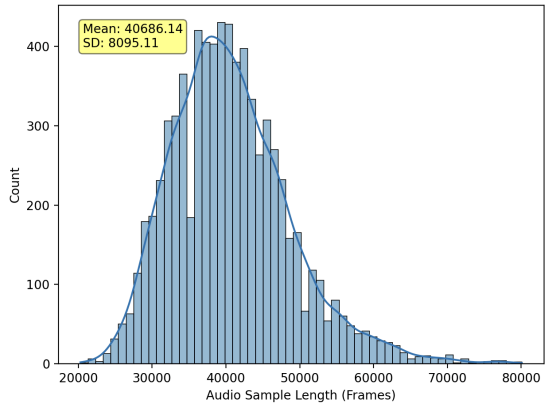

Figure 1. Distribution of audio sample frame lengths

We also examined the distribution of emotional labels for the data. Unlike some similarly annotated and structured datasets, CREMA-D has a near-even distribution of emotional labels.

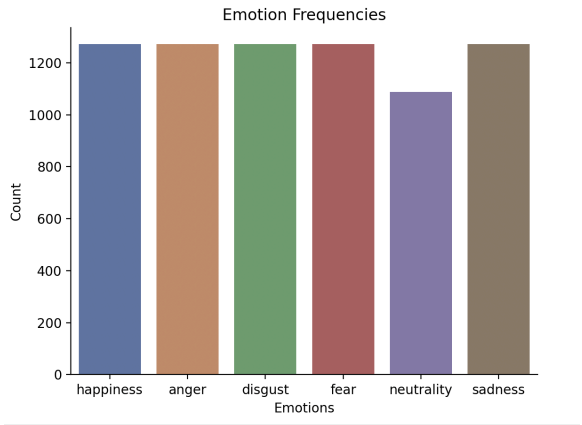

Figure 2. Distribution of emotional labels

Finally, out of curiosity, we looked at the distribution for ethnic demographics of the dataset, which are plotted below. There are only 91 total speakers, and the ethnic groups included are neither a comprehensive or evenly inclusive survey of those found across the world, though they are framed as such by the authors. This is definitely be an issue with the data.

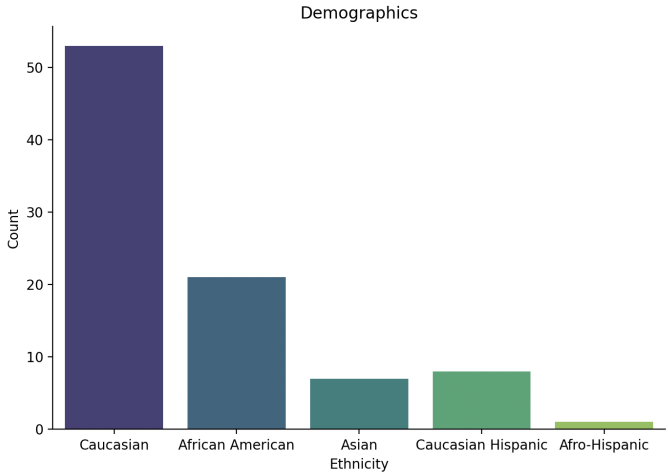

Figure 3. Distribution of demographic categories

b. Preprocessing

Our preprocessing pipeline creates two files: a csv functioning as an overall data table, and a JSON file used to map speaker IDs to speaker features extracted from the audio data. The csv has rows containing data corresponding to: Path | SpeakerID | Sentence | Emotional Label (OHE) | Intensity. Path is a path to the raw audio represented as a vector in .npy format. Intensity is the intensity where 0 is low, 1 is medium, 2 is high, and <unk> is unknown/XX. An example row might be: /…1001_IEO_DIS_LO.npy | 1001 | Sentence| [0 | 0 | 0 | 1 | 0 | 0 | 0] | 0

Our JSON file is created by iterating through the results of the CSV, creating a dictionary from SpeakerID to all audio utterances they are responsible for, then applying Mel Frequency Cepstral Coefficient (MFCC) analysis and mean-pooling to create speaker feature vector values to SpeakerID keys. MFCC analysis is a common technique in speech recognition; each coefficient represents a different level of frequency featurization within an audio sample and the coefficients together are meant to describe salient features of audio samples.

Below we have printed a PCA of the MFCC-derived feature vectors for our speakers, clustered according to gender and race.

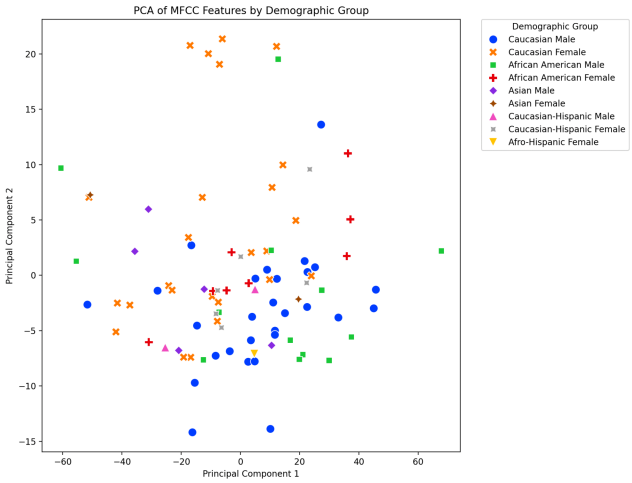

Figure 4. MFCC features clustering plot

c. Model, Training

As stated earlier, we were unable to fully re-implement the methodology of the paper. We will discuss what we did re-implement here, and our plan and execution for doing so. Our model architecture follows closely that of the authors' proposed architecture in the computational graph below. The two differences are that we did not continue the pre-training of HuBERT and that our dense layer count and optimal hidden-size architectures were different from theirs.

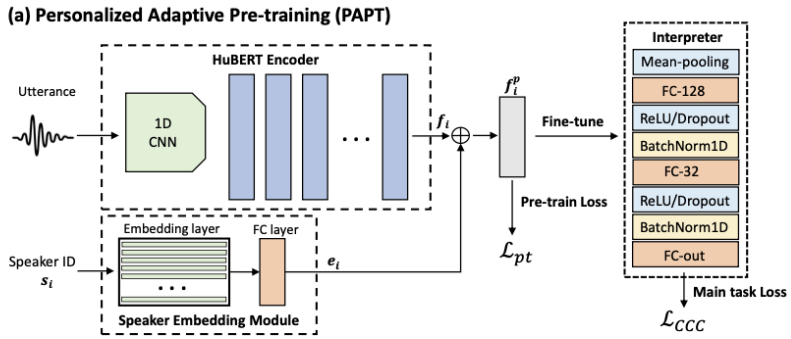

Figure 4. Computational graph for PAPT technique taken from the paper.

In terms of nitty-gritty implementation, we were very limited by computational power, but less so speed than memory. HuBERT is a very large transformer model. We could not directly load our samples into the HuBERT → Fine-tuning head pipeline directly. On Colab, this used all available RAM. On our own machines, it would result in a zsh: killed message which usually means all RAM was used in the process of running a program.

Instead, we had to use HuBERT like it was a secondary pre-processor. We loaded in batches 100 at a time with a script to create HuBERT embeddings of those batches of audio, then saved the HuBERT audio embeddings in a JSON mapping the original file path to the HuBERT embedding (only 1024 in size after mean-pooling so easy to store). This was no doubt an expensive and annoying process but it was necessary to deal with the limited computational power we had access to. From here, we made a data-loader for the JSON files with train-test-val being split in the rough ratio of 0.6-0.2-0.2.

The above text aside, the core of our methodology ended up being quite simple and definitely smaller than the original scope we wanted to do. As previously stated, this pivot in methodology and why we had to pivot will be discussed extensively in **Challenges**. We just ended up fine-tuning several versions of a model, and comparing their performances:

(1) Base
    (a) HuBERT audio encoding as a base, raw classification MLP head (256-64-6) without positional SpeakerID encoding
(2) Speaker 'Position-Encoded'
    (a) The architecture (we believe was) proposed by the authors; uses the SpeakerID as a 'vocabulary' for a model; embeds SpeakerID 'positionally' using tf.keras.layers.Embedding() and sums a dense transformed speaker embedding with the original audio embedding from HuBERT to improve accuracy
(3) Speaker Feature-Encoded
    (a) An architecture we tried for a while because we misunderstood(?) the authors; sums speaker feature vectors to the HuBERT audio embedding of utterances before putting the sum through the MLP head.
        (i) Did not work.

3. <u>Results</u>

a. Visualization

Below, we have attached three line plots showing training accuracy progression over batches for our models. We have also attached a table comparing model performance on validation and test sets. Hyperparameters were:

- *batch size* = 150
- *learning rate* = 1.1e-3
- *optimizer* = adam
- *epochs* = 20

The first MLP architecture we implemented, and which produced good results, was:

- *Dense* — 256
    - *ReLU* \ *Dropout* \ *Batchnorm*
- *Dense* — 64
    - *ReLU* \ *Dropout* \ *Batchnorm*
- *Dense* — 6
    - *Softmax*

(We discuss alternate architectures and tweaks in **Ablation Studies**)

In our table, the Average Validation Accuracy row header refers to average accuracy over the last 10 epochs of training.

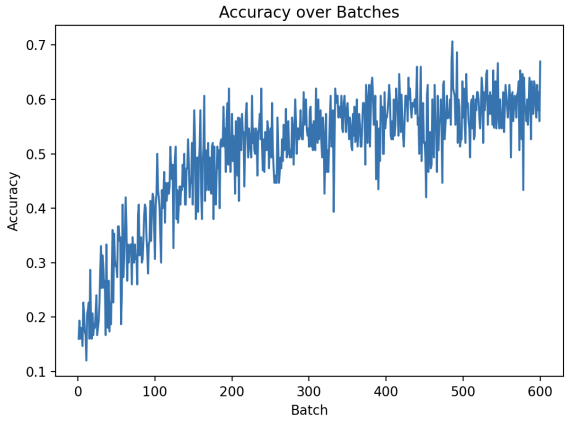

Figure 1. *Base Model*

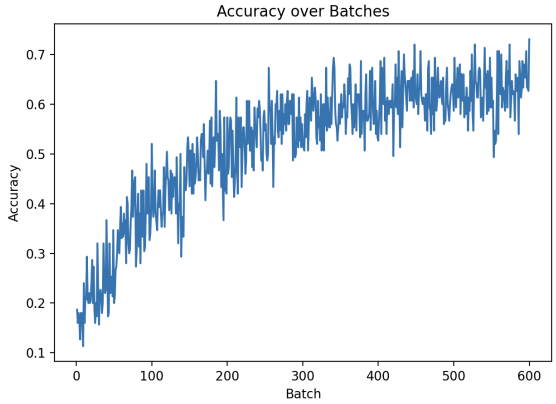

Figure 2. *Personalized Model*

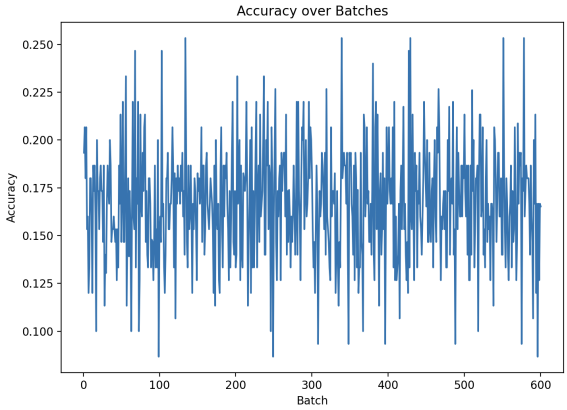

Figure 3. *MFCC Feature Model (Unsuccessful)*

Model	Average Validation Accuracy	Maximum Validation Accuracy	Test Accuracy	Test Loss
Base	0.566	0.5822	0.580	1.04
Personalized	**0.637**	**0.689**	**0.631**	**0.97**
MFCC	0.165	0.189	----	----

Table 1. Model performance on test and validation sets.

b. Analysis

We were pleasantly surprised by the results of speaker personalization. As shown in **Table 1,** the authors' speaker-personalization technique substantially boosted our emotion recognition accuracy on test and validation data. It also seems to stabilize training accuracy. Training accuracy remained relatively jittery across batches for both *Base Model* and *Personalized Model* as shown in **Figure 1** and **Figure 2**. However, personalized training reduced the range of fluctuation. Notice how in **Figure 1**, training accuracy often dips to low/mid 40% values well after epoch 300. In **Figure 2**, training accuracy never dips below 50% after epoch 300.

We cannot directly compare our results to the authors' for a few reasons beyond the fact that they pre-trained and we did not. Obviously, we are working with different types of data—the MSP-Podcast dataset is labeled with *continuous* Valence Arousal Regression values. CREMA-D is labeled with emotion *categories*. Though CREMA-D has an emotional intensity label which could be the target of a continuous regression model, this label was insufficiently crowd-sourced—the vast majority of data is simply labeled 'XX'/unknown emotional intensity. So, unfortunately, we could not use it as a regression target.

Additionally, the authors' dataset has a lot more speakers and more diverse groups than ours—we were working with a set of 91 speakers, while MSP-Podcast has 1409. The positive effect of speaker-personalization might compound the more speakers there are as the vocabulary grows richer and the model learns to encode subtle speaker-related characteristics better.

So while the idea of speaker-personalization encoding might be most effective when pre-trained into the model weights, it is still an effective feature-engineering technique that stabilizes training and improves accuracy.

c. Ablation Studies

We tried many different model architectures and hyperparameter changes to improve accuracy, some successful and unsuccessful. At first, we thought that the jitteriness of the training could be due to the limited dense layers in our neural network and the fact that we jump from a 1024 size audio embedding from HuBERT into a 256 layer immediately after. We thus experimented with a Modified Architecture simply packing on more layers:

- *Dense — 1024*
    - *ReLU \ Dropout \ Batchnorm*
- *Dense —* 256
    - *ReLU \ Dropout \ Batchnorm*
- *Dense —* 64
    - *ReLU \ Dropout \ Batchnorm*
- *Dense — 64*
    - *ReLU \ Dropout \ Batchnorm*
- *Dense —* 6
    - *Softmax*

The spread of accuracy over batches during training does appear to be a bit tighter overall with a deeper architecture, especially for the base model. (They are still relatively jittery though) Additionally, training metrics have improved for both the base and personalized models.

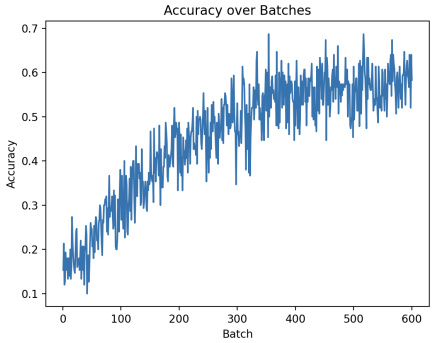

Figure 4. *Base Model*

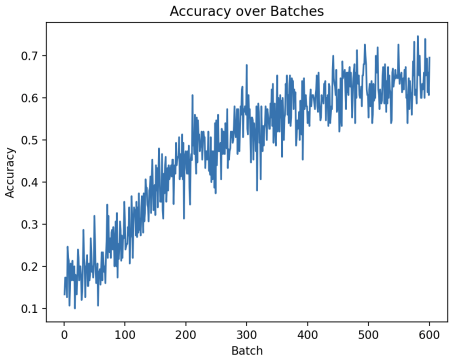

Figure 5. *Personalized Model*

Model	Average Validation Accuracy	Maximum Validation Accuracy	Test Accuracy	Test Loss
Base	0.587	0.610	0.618	0.998
Personalized	**0.632**	**0.671**	**0.649**	**0.947**

We also discovered in the process of tuning hyperparameters that our model is quite sensitive to the interplay of learning rate and batch size. The first batch size/learning rate combination that worked for us was 30/1e-4, respectively. However, batch size of 150 with the same learning rate showed substantially poorer performance. Batch size of 150 with a learning rate of 1e-3 had a bit poorer performance. Our final batch size of 150 with a learning rate of 1.1e-3 seemed to hit the sweet spot and give us the best results so far, though more hyperparameter tweaking could be done.

d. Discussion

We think fixing the jitteriness of our training plot could push the model to perform better. Even in the more stable personalized model, accuracy can still shift between 60 and 70% in late epochs suggesting it hasn't quite found the best representation yet. We are not entirely sure how good our model is relative to top results because the benchmarks for CREMA-D are quite empty. It seems like the last submitted benchmark is 60.2%, which is lower than ours. (Though maybe this is a 'home-grown' audio representation model, whereas we use the much more robust HuBERT as our baseline?)

One factor which we hypothesize could make a significant difference is more dynamic batching when feeding into the HuBERT model. We pad everything to 48000 frames and truncate everything above to 48000. If we

dynamically batched by size, grouping videos with similar lengths together and feeding them in, this could result in more robust representations of audio on the tails of the length distribution. We have heard that HuBERT should be capable of taking in varying sequence lengths, but its preprocessors in HuggingFace naturally pad audio and our attempts to input varying sequence lengths (e.g using RaggedTensor) have resulted in extremely cryptic errors we could not debug. So, we think length-based batching could improve the accuracy of our predictions for sure to avoid overpadding smaller sequences and having to truncate longer ones too much.

4. Challenges

We faced many challenges over the course of the project, which is why our end result was not quite what we imagined/hoped we would do at the beginning. The biggest challenge was being exposed to and competently pruning research papers for good, reproducible ideas and results. No one in our group had done a research project that involved working with existing literature before. We had to pivot many times due to not having a lot of this relevant experience.

We assumed that the authors' **PAPT** methodology would be reproducible. However, there were a lot of obstacles which got in our way:

(1) **PAPT** requires continuing the pre-training of Meta's model HuBERT. However, HuBERT's self-supervised pre-training architecture uses a proprietary model to generate pseudo-labels, so any attempt to continue re-training it must either have access to this proprietary model or replicate it.

(2) There was no way for us to feasibly implement this pseudo-label generation. The authors' source code is incomplete. There were no files showing how they re-implemented the HuBERT pseudo-label generation method for **PAPT**. Either someone from Meta was willing to give them access to the HuBERT pseudo-label model or they implemented a very successful imitation of it behind the scenes.
    (a) If case 1, we can't do anything.
    (b) If case 2, trying to re-implement pseudo-label generation from scratch without any paper guidance or source code would be quite a bit beyond the scope of what we can do!

(3) Even if we had access to pseudo-label generation, pre-training HuBERT is really really computationally expensive as it is a multi billion parameter foundation model; this is well beyond a medium compute project.

So, having to refactor our project plans according to what is feasible and what is not, and having to discover that along the way and pivot appropriately, was the biggest overall challenge of this project.

Other challenges involved the rough quality of datasets in speech audio recognition. MSP-Podcast is a very unique dataset in that it is annotated with continuous emotion measures rather than only simple classifications. We were not able to access MSP-Podcast in time due to Brown University's data transfer agreement requiring a fair

amount of downtime. So, we decided to use the fully publicly available CREMA-D. CREMA-D is objectively a less expansive and well-labeled dataset. It does not contain any Valence Arousal Dominance numerical labels.

Nonetheless, we hoped to use CREMA-D's intensity labels as a continuous regression measure, and to implement the authors' **PLDC** method (which requires numerical data/distributions, not categorical ones) using these labels. However, the intensity labels for CREMA-D are mostly unannotated, which threw a wrench in our plans. We tried to think about some ways we could do something similar to **PLDC** for categorical datasets. For example, looking at the mode label for speakers with similar features to the evaluated speaker, and then shifting the probabilities assigned to the evaluated speaker's emotion classification distribution to match that searched up distribution better. But this seemed to be hackish and difficult to formulate/justify (much less implement) well, so we decided not to try a last-minute self-implementation of **PLDC** adapted to categorical data.

We also simply misunderstood some things that cost us a lot of time, which on reflection is all part of the learning process. For example, the authors discuss both speaker embeddings and speaker features, and we assumed that they meant the same thing. That is, speaker features are calculated as the mean of features of utterances for a given speaker, and this is also how speaker embeddings are calculated. This is why we tried out the *MFCC Features* model. We spent a fair amount of time trying to scale the features properly to get this to work until we realized embedding meant literally using speakerID sets like tokens for words in a vocabulary and using them in an embedding table!

Likewise, our inexperience with using the HuBERT model from HuggingFace, assuming it could easily take in inputs of varying sequence lengths, killed a lot of time spent trying to prepare our data for the model. Our first time processing our data, we vastly overpadded the data to max length by allowing the autoprocessor model associated with it to work on default setting. Most of our features were basically 0 as a result, and we had to undergo a lengthy process of going back and repadding our entire dataset (which took another 2 hours).

5. <u>Reflection</u>

This was (for some of us) one of the most rewarding course experiences we've had, despite the challenges we faced and disappointments about not implementing the full scope of what we wanted to do. We implemented at least our base/target goal (re-implementing the fine-tuning model, comparing results thoroughly to base) once we had pivoted to our final methodology plan. Our reach goal of implementing PLDC for testing inference ended up not being possible in the first place due to the nature of the data we are working with. We think the project turned out ok in the end, but we value the learning opportunity involved with it more than the result we got out of this particular project.

Our model basically works the way we imagined it would in terms of achieving a decent prediction. We wish it were easier to compare to both other models' performances and the authors'. Without any additional pre-training, we achieved test/validation accuracy increases in the ballpark of 5 - 10% over the base unpersonalized model using the speaker personalization method. At the same time, the authors hypothesized that mere speaker personalization/feature-engineering in fine-tuning might have less of an advantage over base fine-tuning the more speakers a dataset contains, thus prompting them to engage in pre-training too.

I will comment regarding model performance that it's impressive that it can identify the emotion accurately ~64% of the time, because we tried to identify a few of the audio samples and got a lot of them wrong. In fact, we confused labels like 'happiness' and 'fearful', 'sadness' and 'neutrality'. Part of this might be that we are using the audio only portion of the dataset. The original samples were, I believe, video clips instead of .wav files, and having a richer multimodal representation of emotion would probably make both our accuracy and (a good) model's accuracy much better.

The main thing we wish we would have done differently if we were to do this project again is to have immediately gotten advice on what parts of the project we can and should re-implement. It was definitely not super fun to have to pivot a lot as described in **Challenges**. We might also have just not re-implemented the paper and used CREMA-D's video part to try to create a multi-modal emotion recognition on our own.

If we had more time, we would have waited for the data transfer agreement to be completed and thereby gotten access to MSP-Podcast. Audio recognition datasets are few and generally not great in quality because it seems to currently be a smaller field than other NLP work or CV projects. We started requesting data a bit late so we decided not to risk the bureaucracy, but MSP-Podcast would have helped us a lot.

It is simply a better dataset as previously mentioned. It also gives us a good point of reference to the authors. With more time, we could try to see if we could, even without pre-training (since no matter how much time we have, this is probably too computationally expensive and we still have no means by which to generate pseudo-labels), achieve similar results to the authors by means of other feature engineering and techniques.

Finally, we would be able to implement both the authors' training/test split and their PLDC technique. The authors split their training/test data specifically so that some tests would have no speakers from the set of speakers the model was trained on. They did this to test inference to unheard speakers. The authors were able to split it this way because they had 1409 well sampled speakers; if we split, say, 10 speakers from our 91 total who had never been seen before, there's definitely a possibility that we just removed, say, half of the African-American speakers or all of our Asian speakers from the dataset! The PLDC situation has already been explained.

The big takeaway from this project for us goes back to the big challenge—know how to identify what parts of a paper are good research methodology, which are reproducible, and be able to pivot your aims well depending on that. Another big takeaway is to *look into your data, feature distributions* thoroughly before deciding how to preprocess it. We overpadded our data and made the entire thing noise the first time because we didn't look at the lengths of our data and notice that it's probably much smarter to make a max audio length cutoff (say, 48000 frames) and pad/truncate to there rather than allow the processor to noisily auto pad everything to the max length of 80080 frames. These are challenges unique to implementing a personal research project rather than working with the pre-defined scope and largely already processed data of a homework assignment but they were well worth going through.

6. References

Tran, Minh, et al. "Personalized Adaptation with Pre-Trained Speech Encoders for Continuous Emotion Recognition." *arXiv.Org*, 5 Sept. 2023, arxiv.org/abs/2309.02418.

7. Related Work

Sridhar, Kusha, and Carlos Busso. "Unsupervised Personalization of an Emotion Recognition System: The Unique Properties of the Externalization of Valence in Speech." arXiv.Org, 19 Jan. 2022, arxiv.org/abs/2201.07876.[1]

No public implementations found

[1] A paper cited by the authors of 'Personalized Adaptation'; we briefly discussed the paper in our first check-in but did not use any ideas from it.