Understanding Emotion Classification In Audio Data

Stanford CS224N Custom Project

Gaurab Banerjee **Emily Huang** **Allison Lettiere**
Department of Computer Science
Stanford University
{gbanerje,emily2h,agl12}@stanford.edu

Abstract

Spoken audio sentiment classification continues to be a difficult task in natural language processing. Currently, there does not exist a consensus on the best input features nor the best model architectures for neural emotion classification. In the following experiments, we employed the RAVDESS (Ryerson Audio-Visual Database of Emotional Speech and Song) dataset to probe speech sentiment classification. We demonstrate the outcome of a series of experiments to identify the optimal combination of neural model inputs for emotion classification, specifically examining MFCCs (Mel-frequency cepstral coefficients), Mel spectrogram, chromagram, spectral contrast, and Tonnetz representation. Second, we construct three new model architectures, integrating our results from the feature exploration. Our new model architectures consist of an extension of ResNet-50, an extension of DenseNet-201, and a novel combination of ResNet-50 and DenseNet-201 outputs along with custom architectural units that we call MultiNet. Our new proposed architectures and input feature combinations perform better than previous published results on the RAVDESS dataset, demonstrating an exciting novel contribution to the field of speech emotion classification. Finally, we demonstrate that MFCCs and Mel spectrograms are the most important audio representations for the audio sentiment classification task and that Tonnetz representation can aid with a minor boost in accuracy.

1 Key Information to include

- External collaborators (if you have any): Justin Wang
- Mentor (custom project only): Parastoo Abtahi and Jackie Yang (HCI Group)
- Sharing project: CS 224S

2 Introduction

Speech emotion recognition is a difficult task that is critical for many human-computer interaction applications. Accurate speech emotion detection systems open up new possibilities in virtual assistant conversational design. Currently, many virtual assistants ignore important sentiment and rely solely on speech transcription. To learn if the user is satisfied with its most recent action, a system must prompt the user for explicit confirmation, instead of implicitly inferring satisfaction based on user's vocal emotion. As an alternative, we envision a system that can use emotion detection to realize it has made a mistake without asking the user if it has done so.

While there has been some work in this space, much of the existing work relies on sentiment analysis of the transcribed text. This is not surprising as written sentiment analysis is accompanied by a significantly larger body of work [1] compared to raw audio and researchers have achieved near perfect accuracy on datasets such as Yelp & IMDb. Unfortunately, transcription discards a plethora of features present in audio.

Despite recent advances in machine learning and deep learning research, modern networks still struggle to achieve near perfect accuracy on speech emotion recognition tasks. As an example, the state-of-the-art accuracy on the Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS) dataset is 71.61% (human accuracy = 67%) [2, 3]. Previous works [4, 5, 2] demonstrate that a combination of advanced deep neural network techniques and an intentional selection of audio representations may help advance accuracy levels on three speech emotion recognition datasets.

Previous research outlines the advantages of past emotion recognition architectures, including Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and long-short term memory models (LSTMs). Previous exploration leaves substantial room for improvement with respect to audio feature representation in combination with model architecture selection. We aim to resolve some of the extreme ambiguity as to how to best represent input audio and construct a model optimized for emotion classification which can outperform the current state-of-the-art model's 71.61% test accuracy.

In this research, we perform an extensive search over various combinations of input features and model architectures to identify the combinations of features and existing architectures that achieve the highest validation accuracies and use these results to inform the construction of three new architectures with which we set a new state-of-the-art for RAVDESS emotion classification accuracy. Additionally, we show that classification accuracy varies based on gender. We further demonstrate that transfer learning using ImageNet trained models does work for the sentiment classification sub-task of audio classification problems. To our knowledge, this is the first granular investigation to empirically demonstrate the effects of specific audio input features on sentiment classification using a variety of model architectures.

3 Related Work

Issa et al.'s (2020) [2] top performing model (evaluated on RAVDESS) combines multiple models in an ensemble and uses the ensemble to output the final classification. The ensembling was especially interesting to us as it led us to consider approaches in which we would train models on separate sets of inputs and combine them to produce better overall classifications. We used their proposed model as a variant baseline and to determine whether their model was hyperoptimized to their input features.

The authors combine five different audio features (MFCCs, Mel spectrogram, chromagram, spectral contrast, and Tonnetz representation) to create a stacked input. This informed our input feature selection and our decision to compare all models using combinations of these five features. We did not find comprehensive research detailing relative effects of each of these features on accuracy.

Palanisamy et al. (2020), presents preliminary work on the use of CNN-based models for audio classification. ImageNet [6] has become a classic pretraining set for image models focused on classification. The authors examine whether standard image classification models pretrained on datasets like ImageNet can be used for audio classification tasks without significant fine-tuning.

Palanisamy et al. (2020) show that using pretrained weights for a standard model performs better than using randomly initialized weights. They show than an ensemble of ImageNet pretrained DenseNet achieves state-of-the-art on both the ESC-50 [7] and the UrbanSound8K [8] datasets and they establish that employing transfer learning assumptions on image models holds for spectrograms. These results informed our DenseNet & ResNet approaches to the audio sentiment classification task.

Luo et al. (2019) propose an utterance-based deep learning model with a parallel combination of CNN and LSTM based networks to obtain an Audio Sentiment Vector (ASV) [4]. In this network, the CNN branch of the network takes in spectrum graphs as input, while the LSTM branch takes spectral centroid, MFCC, and other traditional acoustic features as input. The authors merge the outputs of the LSTM and CNN branch into the ASV using attention, which acts as a regulator.

For the CNN branch, a ResNet was the best architecture empirically. Furthermore, they found that MFCC, spectral centroid, spectral contrast and chroma stft, produced the highest model accuracy which informs our feature selection. Finally, the authors show that their multimodal model performed better than the state-of-the-art model at the time [9], which only used MFCC as input.

Luo et al. demonstrate the value of merged outputs and selection of different input feature sets for each of the pre-merge architecture branches. This is the basis of our MultiNet architecture.

4 Approach

We first experimented with our input representations (Mel spectrogram, MFCC, spectral contrast, Tonnetz representation, and chromagram) to determine the relative importance of each. In total, there were 19 distinct combinations of input features tested: each individually, all five together, and various combinations of 2-4 features.

In order to enable this amount of experimentation efficiently, we built our code infrastructure to support a plug-n-play model with inspiration from Palanisamy et al. (2020) [5]. With our robust backend, feature sets and model choices could be controlled by editing just two parts of a configuration JSON file for each experiment. We reconstructed the Issa et al. codebase to adapt to our infrastructure, used the Torchvision model zoo for the DenseNet/ ResNet base architectures, and adapted an author-provided codebase for Vision Transformer. Our novel models, dataloader, and preprocessing pipeline were built from scratch.

Our two baselines were a simple logistic regression model and the network in [2]. We then tried our own simple five layer convolutional neural network shown in Fig. 7, a Vision Transformer (ViT) [10], ResNet-50 [11], and DenseNet-201 [12]. ViT, ResNet-50, and DenseNet-201 were all pretrained on ImageNet. We extended ResNet-50 and DenseNet-201 to create two of our novel architectures. Finally, we created a new variant of model architectures, MultiNet, which builds on [4].

The Vision Transformer is composed of 12 encoder blocks and one classification head. It takes as input a linear projection of flattened patches, which are spliced from the original image. We treat each feature or stacking of features as a N by T image, where N is the sum of the number of Mels, MFCCs, etc. dependent on the feature types used, and T is the input length representing time.

The simple convolutional network outlined in Fig. 7 takes as input $32 \times 1 \times 301 \times 250$ arrays. Each convolutional layer has 32 filters with kernel size 4×4, stride of 1, and no padding. The first block consists of a convolutional layer followed by a leaky ReLU activation. Next, a block consisting of a convolutional layer, leaky reLU activation, and batch normalization is repeated four times. The output is flattened and fed into a linear layer that has eight classes.

Our first new ImageNet architecture consisted of ResNet-50 trained on all five input features with three fully connected linear layers followed by ReLUs [13] built on top of the network. The second new ImageNet architecture consisted of DenseNet-201 trained on the Mel-scaled spectrograms, MFCCs, and Tonnetz representations with three fully connected linear layers followed by ReLUs built on top. We use leaky ReLU as suggested by [14] for its observed performance on neural network acoustic models.

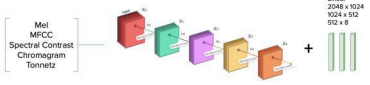

Figure 1: Stitched ResNet-50 Architecture

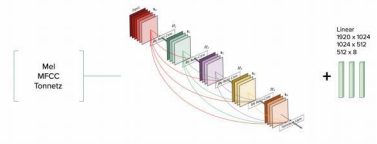

Figure 2: Stitched DenseNet-201 Architecture

4.1 MultiNet

To surpass the previous state-of-the-art performance, we constructed a novel architecture (MultiNet) that is a conglomeration of the ResNet and DenseNet model architectures and which takes inspiration from [4]. Let $X^1 = [X_1^1, X_2^1 ... X_m^1]$ represent the inputs to the Model A, and let $X^2 = [X_1^2, X_2^2 ... X_n^2]$ be inputs to the Model B. Specifically, X^1 and X^2 are the inputs that perform best with respect to each network. In the case of equivalent performance, we choose the smaller number of input features. In our experiments, the input features are the combination of all five input representations for ResNets and only the combination of the Mel spectrogram, MFCC, and Tonnetz representation for DenseNets. For our experiments specifically, let Model A be any DenseNet and Model B be any ResNet architecture. We experiment with ResNets 18 and 50 and DenseNets 121 and 201.

These individual architectures output $[I_1^1, I_2^1 ... I_p^1]$ and $[I_1^2, I_2^2 ... I_q^2]$, where p and q represent the final output dimensions of Model A and Model B, respectively. We concatenate these vectors to form $[I'_1, I'_2, ... I'_{p+q}]$, which is then fed into a downstream fully-connected architecture unit. For this unit,

we experiment with permutations of Dropout [15], Linear, and ReLU layers. The output of this final architectural unit is $[E_1, E_2...E_8]$ representing the eight emotion classes in RAVDESS.

During training, to preserve the information learned by the architectures individually, we transfer the ResNet and DenseNet weights from their respective best performing models and fine-tune the entire MultiNet. Notably, we do not separate the loss functions for each intermediate, parallelized model. In the case that ResNet and DenseNet capture different latent information, we believe combining their outputs should enable a downstream model to make more informed and accurate predictions.

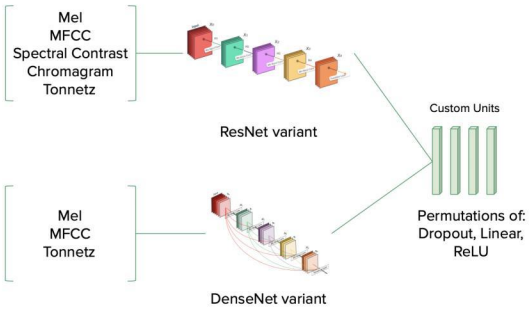

Figure 3: MultiNet Model Architecture

5 Experiments

5.1 Datasets

Our primary dataset is RAVDESS [3]. It consists of audio-visual recordings of speech and song utterances of 12 male & 12 female North American English speaking actors. Actors demonstrated each of these emotions: happy, sad, angry, fearful, surprise, disgust, calm, and neutral. We used only the .wav audio speech files, which yielded 1,440 samples.

The researchers noted that in human labeling, 'calm' was often misidentified as 'happy', and 'sad' was misidentified as 'neutral' or 'calm'. These human classification errors provide a baseline which we should aim to overcome in our neural systems.

Using the state-of-the-art model's input data [2] as inspiration, we preprocessed the raw audio data into normalized Mel-scaled spectrograms, Mel-frequency Cepstral Coefficients, chromagrams, spectral contrast features, and Tonnetz representations using a custom preprocessing framework. In forming each of these features, we used 2048 FFTs, a hop length of 512, and set the max frequency to 4096 Hz. For the Mel spectrogram, we set the number of Mel bands to 256. We generated 20 MFCCs. Finally, we zero-padded each of our features' time dimension to a length of 250.

In summary, our input features were images and output labels were the eight different emotion labels listed above. We split our preprocessed data into an 80-10-10 split and stratified splits by emotion, intensity, statement, repeat, and gender to prevent distribution mismatch.

5.2 Evaluation

We primarily used validation and test accuracy to evaluate success. We selected validation accuracy as an evaluation metric because it is employed in previous literature to compare model success. Additionally, we know the human accuracy level on the RAVDESS dataset, and thus wanted to be able to gauge our success relative to human performance. Finally, it allows us to decompose accuracy by subclass, such as gender, and report on discrepancies between groups. We also use confusion matrices to better visualize per class and between class accuracies.

5.3 Experimental Details

For all of the initial networks except ViT, we trained with Adam optimization and used cross entropy loss [16]. For ViT, we used the AdamW optimizer provided by Hugging Face's Transformers package [17]. The learning rate, weight decay, and number of epochs for each trained model is in Table 1.

We trained ViT for 40 epochs originally, opting for 400 epochs after noticing how loss had not converged after 40 epochs. In general, to conserve training time and compute, we froze all of the model's weights except for the classification head. However, for each combination of features tested on ViT, we trained an additional model where the last of the 12 encoder blocks was unfrozen.

For both the novel stitched ResNet and stitched DenseNet networks, we performed an extensive hyperparameter search over learning rates [10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}] and weight decays [10^{-2}, 10^{-3}, 10^{-4}]. For both the stitched ResNet and stitched DenseNet, the optimal hyperparameter pair was

Model	Learning Rate	Weight Decay	Batch Size	# of Training Epochs
Logistic Regression	10^{-2}	10^{-3}	32	100
Five-Layer CNN	10^{-4}	10^{-3}	32	80
Issa et al.	10^{-5}	10^{-6}	16	700
Vision Transformer	$5 * 10^{-5}$	0	32	400
ResNet-50	10^{-4}	10^{-3}	32	40
DenseNet-201	10^{-4}	10^{-3}	32	40

Table 1: Model Hyperparameters

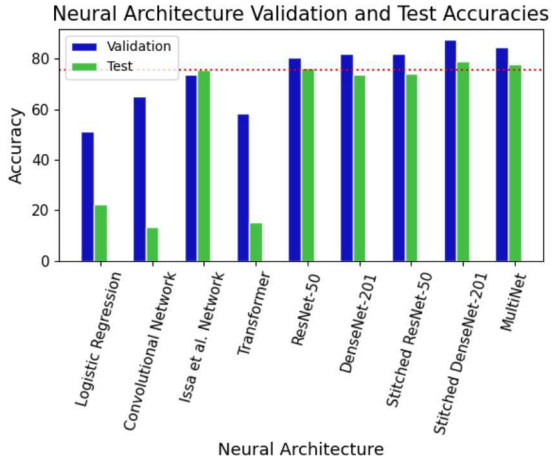

Figure 4: Validation Accuracy and Test Accuracy For All Model Architectures

learning rate of 10^{-6}, weight decay 10^{-3}. Both models were trained with learning rate 10^{-6}, weight decay 10^{-3}, and batch size 32 for 40 epochs with cross entropy loss and Adam optimization.

For MultiNet, we tested learning rates 10^{-3} and 10^{-4}. We conducted 11 unique experiments with MultiNet that were all based on different MultiNet architectures with hyperparameters held constant. We further tested on a wide range of DenseNet and ResNet output sizes as well as intermediate linear layer in and out sizes. For DenseNet and ResNet output sizes, our values ranged between 32 to 1024. First linear layer in sizes were a function of the sum of the ResNet and DenseNet out sizes, and linear layer out sizes were extremely varied. Our dropout layers had varied rates between 0.5 and 0.9. Table 5 lists the various pieces of MultiNet architecture.

The stopping criteria was to train until convergence or <1% average improvement over 10 epochs.

5.4 Results

Our full feature-architecture grid search results are listed in Tables 2, 3, and 4 of the Appendix and MultiNet model results are in Table 6.

The best logistic regression model was created using Mel-scaled spectrogram, MFCCs, Chromagram, Spectral contrast feature, and Tonnetz representation as input features. This model achieved 51.38% validation accuracy and 22.22% test accuracy. The best CNN model was created using Mel-scaled spectrogram, MFCCs, and Spectral contrast as input features. This model achieved 65.28% validation accuracy and 13.19% test accuracy. The network in [2] achieved maximum validation accuracy of 71.61% using Mel-scaled spectrogram, MFCCs, Chromagram, Spectral contrast features, and 75.69% test accuracy with MFCCs, Chromagram, and Tonnetz representation. The vision transformer (ViT), with all but the classification head frozen, achieved a 34.03% maximum validation and 15.28% test accuracy with all five features as inputs. With the last encoder block unfrozen, ViT reached a 58.33% maximum validation and 15.28% test accuracy with Mel-scaled spectrogram, MFCCs, and Tonnetz representation as input features. ResNet-50 reached 80.56% validation accuracy and 76.39% test accuracy with Mel-scaled spectrogram, MFCCs, Chromagram, Spectral contrast feature, and Tonnetz

representation as input features. DenseNet-201 reached 81.94% validation accuracy and 73.61% test accuracy with Mel-scaled spectrogram and MFCCs as input features.

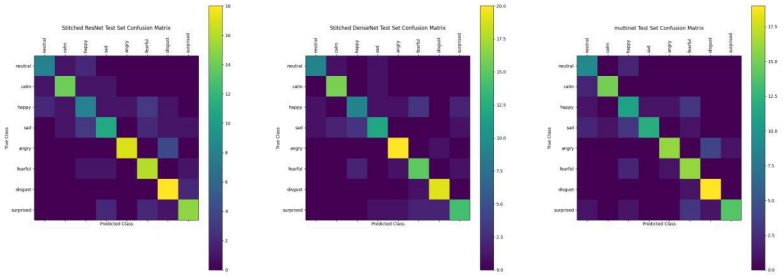

Figure 5: Test Set Confusion Matrices. Left To Right: Stitched ResNet-50, Stitched DenseNet-201, MultiNet. Labels In Order: neutral, calm, happy, sad, angry, fearful, disgust, surprised

We surpassed our previous models' validation accuracies for all three novel, ImageNet based architectures. For the stitched ResNet model, we achieved 81.94% validation accuracy and 74.30% test accuracy. For the stitched DenseNet model, we achieved 87.5% validation and 79.17% test accuracy. For the best MultiNet, we reached 84.72% validation and 77.78% test accuracy. Accuracies are shown in Fig. 4, and the true and false positive rates for each emotion class, are in Fig. 5.

6 Analysis

These results indicate that the combination of features and neural architecture choices can have a substantial impact on the emotion classification task as we hypothesized.

6.1 Feature Representations

Specifically, we found three input features that consistently outperformed all other features combinations across network architectures: the Mel-spectrogram, the MFCC, and the Tonnetz representation. Below, we provide insight into why they perform so well, and what information they provide.

- The Mel-spectrogram transforms a spectrogram non-linearly to reflect how a human perceives relative frequencies. This last part is important because how a human perceives relative frequencies often reflects how a human perceives emotion, which makes it the appropriate starting point for a model to learn latent emotion representations from.
- The MFCC is essentially a compressed, decorrelated version of the Mel-spectrogram, primarily used for Gaussian Mixture Models. Oftentimes it is sufficient to train stronger classifiers like CNNs with just the Mel-spectrogram since the CNN should be able to learn the complex representations within the spectrogram itself. However, we find that with limited data, our models do not have enough training examples to learn such representations. Including MFCCs provides a head start for the model to learn latent representations from.
- The Tonnetz representation splits frequencies into tonal centroids, specifically a 6-dimensional basis representing the perfect fifth, minor third, and major third each as two-dimensional coordinates. While Mel-scaled representations are commonly used to capture timbral aspects of music, they provide poor resolution of pitches and pitch classes [18], so this representation provides the model information about pitch and harmony.

6.2 Model Performances

In our baseline logistic regression and the simple CNN architectures, we see that we do not reach human accuracy levels on the task. Both of these architectures lack the complex parameter representations that are necessary to emulate many of the neural connections in the human brain.

Interestingly, while ResNet-50 has approximately 5 million more trainable parameters than DenseNet-201, it did not perform as well in either our initial experiments or stitched experiments. Furthermore, it appears that the structural differences in the two networks, stemming from the dense connections in DenseNet, had a sizable impact on the number of features and types of features required for adequate training and validation. This might suggest why the addition of several dense layers in our stitched models led to higher accuracies.

Compared to similarly sized models, ViT fell significantly short of the mark. By freezing the entire model except for the classification head, ViT was unable to fit to the training data, earning a 38.02% accuracy at best. By unfreezing the last encoder block, ViT overfit to the training data. The highest train accuracy across all the different input features we tried was 72.92%, while the highest validation accuracy was 58.33%. No matter whether the last encoder block was unfrozen or not, ViT could not generalize, earning a test accuracy slightly better than chance. Much like Transformers in general, ViT suffered due to the lack of data. Transformers trained on text data train optimally when given 58 million sentence pairs, and research has shown that decreasing the training corpus to 16 million sentence pairs significantly decreases performance [19]. From another point a view, given that we are working with a little more than 800 training examples and an architecture with 86 million parameters, it isn't hard to see that the model could have easily memorized those examples.

As opposed to what we had expected, MultiNet did not perform better than the sum of its parts. Indeed, final MultiNet test accuracy is nearly right in the middle between DenseNet-201 and ResNet-50 test accuracy. This would suggest that poorer performing models may actually diminish performance of a merged model. Potential solutions here could look at weighting the different outputs of the early models differently, splitting loss functions, or adding basin hopping (Monte Carlo minimization) to randomly perturb the model's location in the loss landscape in case it is stuck in a local minimum. Interestingly, both ResNet-18 and DenseNet-121 performed nearly as well or better than their more complex counterparts and had smoother training curves which leads us to believe that our dataset was too small and our learned functions were not approximating our validation sets well, even if training accuracy was hitting 100%. Additionally, we were surprised to notice that the addition of regularization through dropout did not help validation performance, nor did the addition of multiple linear + ReLU layers. The best final architectural unit turned out to be a single linear layer. Of all of the different experiments we ran, the best overall model was actually DenseNet-121 and ResNet-18, both with output sizes of 32 which were then stitched together and fed through a linear layer of in size 32 and out size 8. This model achieved 84.72% validation accuracy. As shown in Fig. 6, most of our ImageNet training curves looked quite stable. However, validation accuracy and loss curves had abundant noise regardless of model complexity, regularization, or hyperparameters. We hypothesize this is due to the size of our dataset. Since the validation set only has 100 examples, and the training set has 800 examples, we believe our learned functions do not truly approximate our data and that this could be mitigated by significant dataset expansion.

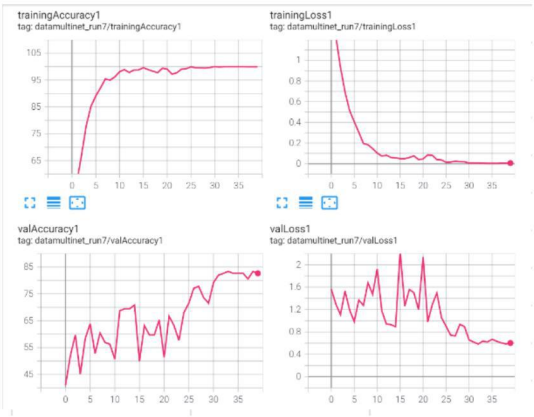

Figure 6: Best MultiNet Training Curves

Across all of our experiments, we found that the Mel-Spectogram and Mel-frequency cepstral coefficients were undoubtedly the most valuable extracted features from our initial audio clips. This implies that knowledge about individual frequencies, the frequencies' amplitudes, and phoneme representation are pertinent to doing well on our task. Furthermore, our explorations indicate the value of using a visual image-based representation of audio data and the superiority of large, ImageNet pretrained models. We do additional complexity analysis in Fig. 11 where we show that there is a definite tradeoff being made between computational resources and a few percent of accuracy. Issa et al. (2020) perform nearly as well as us, but using significantly less computation.

From the confusion matrices (Fig. 5) for the test set results on the stitched ResNet-50, stitched DenseNet-201, and MultiNet, we can see that anger and disgust are classified with the highest accuracy and neutral, happy and sad have the lowest classification accuracy. These patterns follow previous behavior in literature analyzing emotion classification, specifically with relations to the ambiguity between neutral and sad.

6.3 Gender

As the RAVDESS dataset is also labeled by gender, one of the post-hoc analyses we did was looking more closely at gender differences among each model. For the holdout evaluation set, we can see discrepancies in performance across gender more clearly in Table 7 and Fig. 9. Notably, the ResNet and stitched ResNet architectures both had test accuracies for women that were over 20 points greater than the accuracies for men. For other models, there was also a large discrepancy, though not as large - for example, the DenseNet architectures both had a nearly 10 point difference. Across all models, performance for women was an average of 10.5 points higher. For baseline architectures like logistic regression and the simple convolutional network, while accuracy was higher for women, the difference was not as substantial.

While it is not immediately clear why this discrepancy exists, this may be something to look at in future analysis. An initial investigation shown in Fig. 10 suggests a relationship between the complexity of a model (determined by number of parameters) and the size of the gap in accuracy. However, without additional runs of each model, we cannot conclude a relationship here.

7 Conclusion

Overall, we have identified the importance of three speech audio representations: MFCCs, the Mel-Spectogram, and the Tonnetz Representation for emotion classification. We have also identified the importance of image-based features and image models for performing well on speech sentiment classification. We have provided three novel architectures, all of which are expansions on or combinations of previous image models (ResNet and DenseNet), that increase validation accuracy on the emotion label of the RAVDESS dataset. We have been able to analyze a variety of different network structures and increase the field's understanding of the interactions between these structures and different audio features. Lastly, we have advanced on the state-of-the-art for our dataset in the emotion classification task.

The main limitation of our work is the inclusion of only a single, generally small speech dataset for training and evaluating our models. Without further testing and the introduction of additional data, we cannot conclude that the results will extrapolate to all other speech data. Additionally, the RAVDESS dataset is limited to two short utterances, both spoken in English. Going forward, we would like to test that our findings generalize to datasets which contain more expressive utterances and other languages to better understand the implications for the broader speech audio sentiment classification objective.

To address some of these limitations and other anomalies we saw during experimentation, there are a few next steps that we can take. As a first step, as we saw in Fig. 6, to address the noise in the validation accuracy and loss curves, we may want to do bootstrapping in order to get error bars to better understand the curves. Additionally, to explore the gender imbalance in performance, we would like to perform spectral augmentation as proposed in [20], and to shift the pitch of male voices to see what effect that may have on the accuracies of the different models.

In the future, we would also like to use these research findings to classify voice assistant responses based on user emotions. We wish to conduct transfer learning on our best models to identify whether a voice assistant has executed a user's intended task based on the user's emotion when responding to the voice assistant.

References

[1] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*, 2019.

[2] Dias Issa, M Fatih Demirci, and Adnan Yazici. Speech emotion recognition with deep convolutional neural networks. *Biomedical Signal Processing and Control*, 59:101894, 2020.

[3] Steven R Livingstone and Frank A Russo. The ryerson audio-visual database of emotional speech and song (ravdess): A dynamic, multimodal set of facial and vocal expressions in north american english. *PloS one*, 13(5):e0196391, 2018.

[4] Ziquan Luo, Hua Xu, and Feiyang Chen. Audio sentiment analysis by heterogeneous signal features learned from utterance-based parallel neural network. In *Association for the Advancement of Artificial Intelligence (AAAI)*, 2019.

[5] Kamalesh Palanisamy, Dipika Singhania, and Angela Yao. Rethinking cnn models for audio classification. *arXiv preprint arXiv:2007.11154*, 2020.

[6] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[7] Karol J Piczak. Esc: Dataset for environmental sound classification. In *Proceedings of the 23rd ACM international conference on Multimedia*, pages 1015–1018, 2015.

[8] Justin Salamon, Christopher Jacoby, and Juan Pablo Bello. A dataset and taxonomy for urban sound research. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 1041–1044, 2014.

[9] Soujanya Poria, Erik Cambria, Devamanyu Hazarika, Navonil Majumder, Amir Zadeh, and Louis-Philippe Morency. Context-dependent sentiment analysis in user-generated videos. In *Proceedings of the 55th annual meeting of the association for computational linguistics (volume 1: Long papers)*, pages 873–883, 2017.

[10] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2020.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[12] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks, 2018.

[13] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*, 2015.

[14] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3. Citeseer, 2013.

[15] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[16] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[17] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain

Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics.

[18] Brian Mcfee, Colin Raffel, Dawen Liang, Daniel Ellis, Matt Mcvicar, Eric Battenberg, and Oriol Nieto. librosa: Audio and music signal analysis in python. *Proceedings of the 14th Python in Science Conference*, 2015.

[19] Martin Popel and Ondřej Bojar. Training tips for the transformer model. *The Prague Bulletin of Mathematical Linguistics*, 110, 03 2018.

[20] Daniel S. Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D. Cubuk, and Quoc V. Le. Specaugment: A simple data augmentation method for automatic speech recognition. *Interspeech 2019*, Sep 2019.

A Appendix

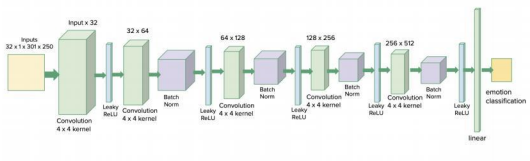

Figure 7: Convolutional Neural Network Model Architecture

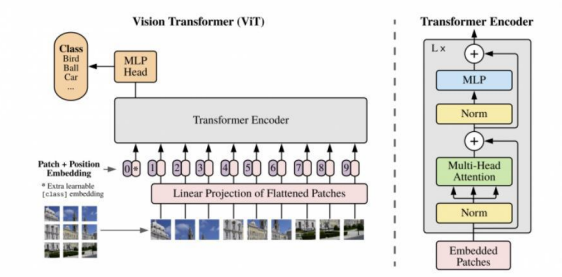

Figure 8: Vision Transformer (ViT) Architecture

Features					Values	Logistic Regression (avg)	CNN (full)	DenseNet	ResNet	ViT UF (ViT F)	Issa et al.
MEL	MFCC	CHR	SC	TON							
					Epoch	53	22	17	38	40 (40)	684
					Validation Accuracy	51.38	59.72	81.94	80.56	54.86 (34.03)	73.61
1	1	1	1	1	Validation Loss	1.87	1.42	0.5451	0.6945	1.344 (1.893)	1.0816
					Train Accuracy	62.326	99.91	98.87	100	82.92 (27.26)	91.05
					Train Loss	1.0336	0.0520	0.04641	0.004723	0.8097 (1.895)	0.5739
					Epoch		52				620
					Validation Accuracy		60.42				66.67
1	1	1	1	0	Validation Loss	*	1.39	*	*	*	1.10
					Train Accuracy		96.7				88.80
					Train Loss		0.2178				0.61
					Epoch		39				665
					Validation Accuracy		61.81				70.83
1	1	1	0	1	Validation Loss	*	1.21	*	*	*	1.1193
					Train Accuracy		95.57				91.77
					Train Loss		0.2485				0.5671
					Epoch		68				682
					Validation Accuracy		56.94				67.36
1	1	0	1	1	Validation Loss	*	1.639	*	*	*	1.1603
					Train Accuracy		97.74				89.46
					Train Loss		0.2008				0.5961
					Epoch		58				99
					Validation Accuracy		50.69				49.306
1	0	1	1	1	Validation Loss	*	1.795	*	*	*	1.6730
					Train Accuracy		64.06				43.57
					Train Loss		0.9909				1.5746
					Epoch		27				541
					Validation Accuracy		61.81				69.44
0	1	1	1	1	Validation Loss	*	1.134	*	*	*	1.0739
					Train Accuracy		99.57				89.62
					Train Loss		0.1483				0.6044
					Epoch	19	14	38	37	40 (40)	22
					Validation Accuracy	39.583	23.61	71.53	71.53	44.44 (36.04)	28.472
1	0	0	0	0	Validation Loss	1.93	20.0533	1.076	9701	1.554 (1.741)	1.946
					Train Accuracy	36.198	76.215	99.57	99.91	70.75 (38.02)	24.86
					Train Loss	1.62	2.1717	0.01443	0.015	0.8518 (1.607)	1.8467

Table 2: Experiments of 6 models across combinations of 5 features, constant hyperparameters per model

Features					Values	Logistic Regression (avg)	CNN (full)	DenseNet	ResNet	ViT UF (ViT F)	Issa et al.
MEL	MFCC	CHR	SC	TON							
					Epoch	43	68		15		685
					Validation Accuracy	47.92	56.94		70.83		68.75
0	1	0	0	0	Validation Loss	1.62	1.3385	*	1.075	*	1.1235
					Train Accuracy	51.3	93.06		99.39		92.37
					Train Loss	1.346	0.6287		0.02495		0.5013
					Epoch	18	59		21		26
					Validation Accuracy	21.53	28.86		44.44		25.694
0	0	1	0	0	Validation Loss	2.04	3.450	*	0.1974	*	1.9968
					Train Accuracy	16.23	100		99.83		20.47
					Train Loss	2.06	0.000135		0.01244		2.0828
					Epoch	40	26		38		
					Validation Accuracy	26.39	35.42		41.67		
0	0	0	1	0	Validation Loss	1.998	2.892	*	0.2349	*	*
					Train Accuracy	21.528	100		100		
					Train Loss	1.961	0.000857		0.001937		
					Epoch	17	15		21		
					Validation Accuracy	24.31	36.805		36.11		
0	0	0	0	1	Validation Loss	2.041	2.6695	*	0.01569	*	*
					Train Accuracy	17.19	100		99.83		
					Train Loss	2.054	0.001265		2.576		
					Epoch			36	22		437
					Validation Accuracy			81.94	75		61.11
1	1	0	0	0	Validation Loss	*	*	0.6315	0.953	*	1.3047
					Train Accuracy			100	99.13		82.01
					Train Loss			0.0062145	0.03764		0.8024
					Epoch		48				573
					Validation Accuracy		61.81				68.75
0	1	1	0	0	Validation Loss	*	1.384	*	*	*	1.1473
					Train Accuracy		100				92.10
					Train Loss		0.05265				0.5490
					Epoch		24			40 (40)	488
					Validation Accuracy		63.89			47.22 (30.56)	65.972
0	1	0	1	0	Validation Loss	*	1.061	*	*	1.703 (1.831)	1.18
					Train Accuracy		98.87			87.93 (34.98)	86.63
					Train Loss		0.1716			0.4746 (1.80)	0.6283

Table 3: Experiments of 6 models across combinations of 5 features, constant hyperparameters per model

Features					Values	Logistic Regression (avg)	CNN (full)	DenseNet	ResNet	ViT UF (ViT F)	Issa et al.
MEL	MFCC	CHR	SC	TON							
0	1	0	0	1	Epoch Validation Accuracy Validation Loss Train Accuracy Train Loss	*	27 61.81 1.193 99.91 0.09849	*	*	*	642 68.06 1.1417 88.83 0.5591
1	1	1	0	0	Epoch Validation Accuracy Validation Loss Train Accuracy Train Loss	*	*	33 78.47 0.7184 99.91 0.01333	33 70.83 0.8823 99.74 0.0093477	40 (40) 48.61 (33.33) 1.647 (1.84) 86.2 (33.59) 0.4835 (0.4835)	537 68.75 1.1669 84.60 0.7133
1	1	0	1	0	Epoch Validation Accuracy Validation Loss Train Accuracy Train Loss	*	30 65.28 1.235 100 0.02582	24 75.69 0.8222 99.57 0.02455	32 76.39 0.7467 99.91 0.0086066	*	*
1	1	0	0	1	Epoch Validation Accuracy Validation Loss Train Accuracy Train Loss	61 49.3056 2.379 60.503 1.079	11 53.472 1.541 91.146 0.3413	24 85.42 0.58 100 0.0074489	34 79.17 0.7323 100 0.0048535	40 (40) 58.33 (33.33) 1.255 (1.907) 88.63 (29.34) 0.4134 (1.922)	653 68.06 1.1452 87.37 0.6149
0	1	1	0	1	Epoch Validation Accuracy Validation Loss Train Accuracy Train Loss	*	*	*	*	*	580 72.22 1.1004 93.75 0.5129

Table 4: Experiments of 6 models across combinations of 5 features, constant hyperparameters per model

Key	D1	D2	D3	R1	R2
Model	DenseNet-201 Output size: 1024	DenseNet-201 Output size: 32	DenseNet-121 Output size: 32	ResNet-50 Output size: 1024	ResNet-50 Output size: 32
Key	R3	A1	A2	A3	A4
Model	ResNet-18 Output size: 32	Dropout Linear (2048,512) ReLU Linear (512,8)	Linear In- 2048 Out- 8	Linear In- 64 Out- 8	Dropout Linear 64,8)

Table 5: MultiNet Architecture Selected Layer Options

Model A	Model B	Final Architectural Unit	Notes	Validation Accuracy
D1	R1	A1	Dropout 0.5 LR = 1e-3	74.31
D1	R1	A1	Dropout 0.9 LR = 1e-4	75.00
D1	R1	A1	Dropout 0.5 LR = 1e-4	74.31
D1	R1	A2	Dropout 0.5 LR = 1e-4	79.17
D2	R2	A3	Dropout 0.5 LR = 1e-4	79.17
R2	D2	A3	Dropout 0.5 LR = 1e-3	79.17
D3	R3	A3	Dropout 0.5 LR = 1e-3	83.33
D2	R2	A4	Dropout 0.5 LR = 1e-3	80.56
D3	R3	A3	Dropout 0.5 LR = 1e-3	84.72
D2	R2	A3	Dropout 0.5 LR = 1e-3	81.25
D2	D2	A3	Dropout 0.5 LR = 1e-3	77.28

Table 6: Selected MultiNet Results

Models	Test Acc	Test Loss	Test Acc (W)	Test Loss (W)	Test Acc (M)	Test Loss (M)
Logistic Regression	22.22	9.16	22.86	8.51	21.62	10.13
CNN (full dataset)	13.19	2.09	15.07	2.09	11.27	2.11
Transformer	15.28	2.12	16.44	2.10	14.08	2.16
DenseNet	73.61	0.95	78.08	0.85	69.01	1.30
ResNet	76.39	0.78	86.30	0.45	66.20	0.96
Issa et al.	75.69	0.96	80.82	0.82	70.42	1.11
DenseNet ++	79.17	0.90	83.56	0.69	74.65	1.06
ResNet ++	74.31	0.79	84.93	0.60	63.38	0.79
MultiNet	77.78	0.79	82.19	0.55	73.24	1.08
Average	56.40	2.06	61.14	1.85	51.54	2.30

Table 7: Test accuracy and loss by model, split among gender

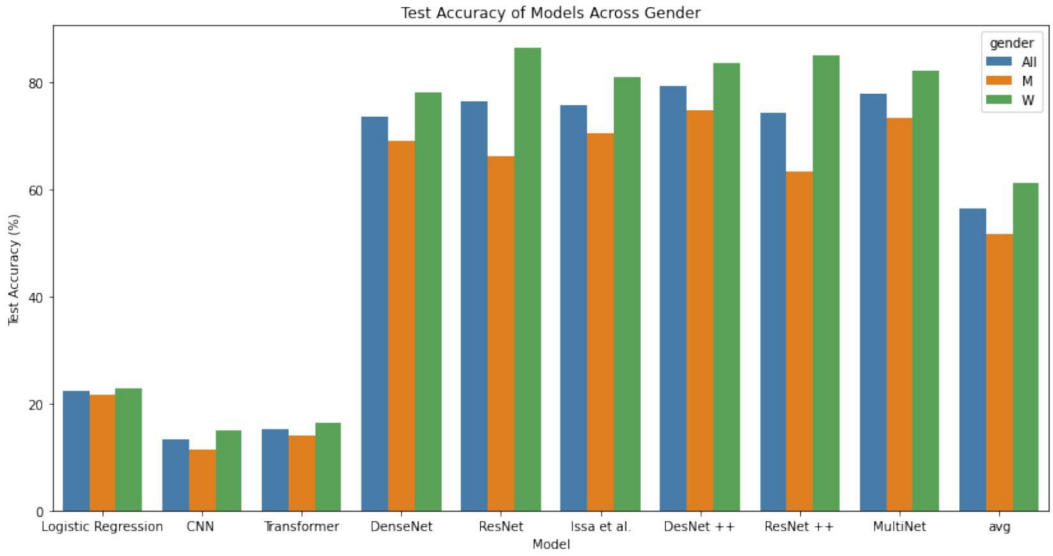

Figure 9: Test accuracy of each model by gender.

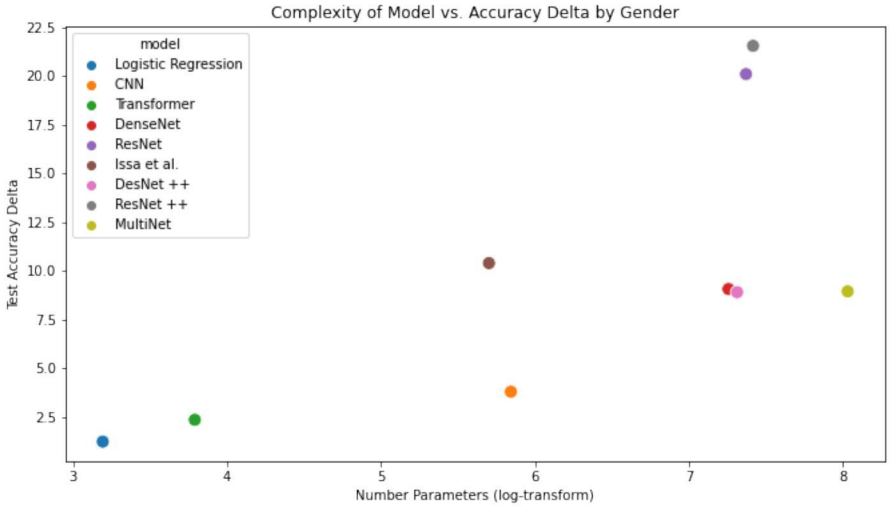

Figure 10: Log transform of number of parameters on the x-axis, plotted against the difference in test accuracy by gender ($\Delta acc_w - acc_m$).

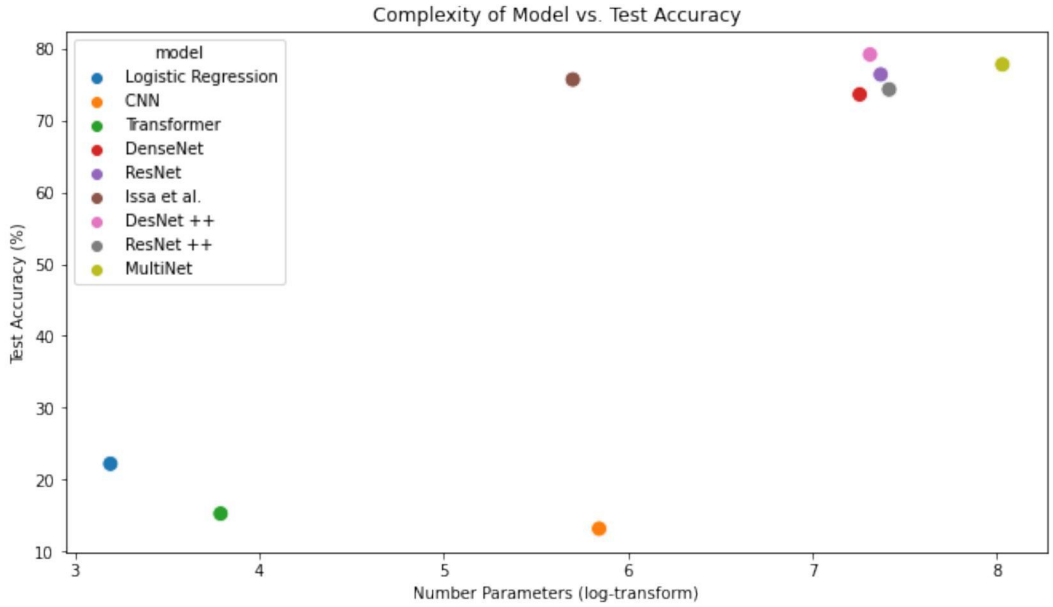

Figure 11: A plot of the number of parameters log transformed of each model versus its test accuracy.