Emotion in Motion: A Deep Learning Model for Speech Emotion Recognition

by MPG: Matan Gans, Peter Theodores, George Rusu

1. Introduction

The problem we are addressing is the difficulty of classifying human emotion, given variable length speech. Speech emotion recognition systems struggle with classifying emotion due to the abstract nature of emotion and the fact that human emotion can only be detected in small parts during long segments of speech.

The objective of this paper is to improve human-computer interaction efficiency by making the model aware of which time-frequency region of the utterance is most emotion-relevant. This is a classification problem based on the methodology described in the paper *Attention Based Fully Convolutional Network for Speech Emotion Recognition* by Zhang et al. (2019)

2. Methodology

We generally adhered to the proposed architecture outlined in Zhang et al. (2019), with some modifications based on our dataset, data preprocessing steps, and hyperparameter tuning.

2.1. Data

We used audio-only speech recordings from the open-sourced RAVDESS (Ryerson Audio-Visual Database of Emotional Speech and Song) dataset to train our model on, and extracted four different emotion classes. From each of these wav files, we developed a speech spectrogram representation with a series of 259 time steps representing average frequency values of the audio. We ended up with 510 such representations (divided into 134 Happy, 174 Angry, 52 Neutral, 150 Sad). Given the limited amount of data, we set up a five-fold cross validation sampling procedure (such that the model is trained five times under an 80-20 train-test split with a different section of the data being used as our test sample) so that we could average results over five unique test samples.

2.2. Model Architecture

Our model architecture is split into two parts: a Fully Convolutional Network (FCN), and an Attention Layer. The FCN is composed of 1 dimensional convolution layers, maxpool layers,

and nonlinear activation layers. We started with the paper's outlined values for channel size, kernel size, and stride size, however, because we had a reduction in the input dimension, we cut the channel size in half, and then treated channel size as a hyperparameter, tuning it until we saw our accuracies increase and our loss plots converge and display a decrease in fluctuation. So the 1D convolution layers have channel sizes 5, 10, 20, 30, have kernel sizes 11, 5, 3, 3, strides of 4, 1, 1, 1, and all have Rectified Linear Unit activations. All of the maxpool layers are identical, having pool sizes of 3 and stride length of 2. The final component of the FCN is a Dense layer with a tanh activation function.

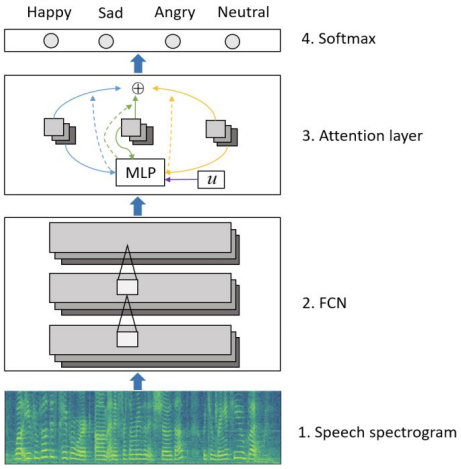

Figure 1: Proposed Architecture, as displayed in the paper

Our attention layer is used to construct an utterance emotion vector by extracting the elements most relevant to the utterance emotion and aggregating them. This is done because not all parts of the utterance, meaning not all average frequency values within the spectrogram, are equally significant. (1) shows each of the outputs of the FCN layer aligned in a C dimensional vector, to represent each part of the speech. (2) - (4) show the computations that our attention layer performs in order to produce the utterance emotion vector. (2) represents each annotation (a part of the utterance) being passed through a fully connected linear layer (with tanh activation) to produce an importance weight. Here, the vector u is a trainable variable vector of size (batch size x 4 x 1). The next step, after computing the importance weight, is to normalize it in (3) by using a scaled softmax function. When $\lambda = 1.0$, the function is a regular softmax function. The paper suggests using 0.3, but after finetuning our hyperparameters, we found that a value of 0.8 suited our adapted model better. This impacts the uniformity of the importance weights of the annotation vectors.

The final step involves us multiplying the annotation vector with the normalized weight.

$$\boldsymbol{A} = \{\boldsymbol{a}_1, \cdots, \boldsymbol{a}_L\}, \boldsymbol{a}_i \in \mathbb{R}^C \quad (1)$$

$$e_i = \boldsymbol{u}^T \tanh(\boldsymbol{W}\boldsymbol{a}_i + \boldsymbol{b}) \quad (2)$$

$$\alpha_i = \frac{\exp(\lambda e_i)}{\sum_{k=1}^{L} \exp(\lambda e_k)} \quad (3)$$

$$\boldsymbol{c} = \sum_{i=1}^{L} \alpha_i \boldsymbol{a}_i \quad (4)$$

Figure 2: Formulas for the Attention Mechanism, as displayed in the paper

Hyperparameters	
Learning Rate	0.001
Batch Size	51
Number of Epochs	200
Lamda (weight used in attention)	0.8

Figure 3: Hyperparameters used in our model

3. Results

Each trial is run on a dataset over 200 epochs. The unweighted accuracy is the percentage that the model predicts correctly from the test set. The weighted accuracy is the average of the percentage of samples of each emotion that the model predicted correctly.
We managed to get fairly high average accuracies, but the model occasionally falls into a trap of learning to predict only one emotion. This suboptimal strategy would likely be resolved with different hyperparameters, but every iteration of our model that we tried had this issue to some degree. Using the weighted average helped us examine when the model was learning bad strategies, and even though we weren't able to completely resolve this, we have a better understanding of why the model is doing this and why its performance varies. This problem is not visible when we only look at unweighted averages. This fact reinforces the idea that we need to examine results more carefully than just looking at an unweighted accuracy, as the discrepancy also hints at potential issues in the data distribution.

Sample Number	Unweighted Accuracy	Weighted Accuracy
1	59.8%	50.2%
2	72.5%	61.5%
3	36.3%	25.0%
4	59.8%	47.8%
5	27.5%	25.0%
Average	**51.2%**	**41.9%**

Figure 4: Unweighted and Weighted Accuracies Over 5 Samples (Following a 5-Fold Cross Validation Mechanism)

4. Challenges

Over the course of this project, we encountered several major challenges that resulted in suboptimal model performance and hindered our model's overall accuracy. The most pressing and impactful challenge we faced related to preprocessing. Our first and most radical deviation from the approach outlined in Zhang et al. (2019), as was described in the previous sections, was the reduction in input dimensions. and processing the raw waveforms from the data set. The paper outlines a general technique called Hamming for how to extract and isolate spectrograms to be analyzed by the FCN and the attention layer. The process of hamming, and the subsequent transformation of the resulting spectrogram windows using a Discrete Fourier Transform (DFT), are part of advanced digital signal processing methods used for convolution. We were unfamiliar with hamming, and upon researching the signal library in python, we found the

signal.spectrogram function that handled hamming. The resulting spectrograms then were passed through a DFT to create equally spaced spectrogram waves for clear analysis.

However when we tried to use these 2D tensors as inputs to our FCN (where input tensor is of size Frequency x Time) we ran into issues pertaining to size and performance. After expanding the tensor dimensions to include a third dimension, (Frequency x Time x Channel Size) as described in the paper, we ran into issues with how the attention layer was handling each individual element of the utterance. Zhang et al. (2019) did not provide any sort of guidance or reference to any other specific approach or library detailing how the data should have been processed, so we decided to represent our data in averages. Instead of using input tensors with every frequency at each timestep, we decided to average the frequency response for all valid timesteps in the entire speech utterance. This resulted in us no longer having to work with both frequency and time domains, reducing the input to the FCN to one dimension. This then prompted us to change the model accordingly, relying on one dimensional convolution, and updating how the attention layer would extract the resulting output of the FCN. Where Zhang et al. iterate through a C-sized (channel size) collection of Frequency x Time tensors, we had to operate on a C-sized collection of averaged F tensors.

Once our model began training we noticed that our loss and accuracy were not corresponding in logical, meaningful ways. We noticed that loss would sporadically increase, and that it would not converge at times. Our solution to this was drastically reducing the channel size of our convolutional layers and removing the fifth and final layer. Our model then improved in both accuracy and the losses began to converge to singular values.

5. Reflection

Our project turned out reasonably well, as our trained network is able to identify the emotion of the samples around 51% of the time, which is significantly better than guessing randomly. We were not able to reach the accuracy achieved in the paper, which makes sense as we had to deviate from their implementation given the time constraint on preprocessing as mentioned earlier. Nonetheless, our model allowed us to achieve our target goal, as we implemented a convolutional network using attention and got reasonably high accuracies. We did not meet our stretch goal of achieving the same accuracy as the paper or implementing multi-head attention.

Although our model was adapted and we implemented a modified version of the architecture detailed in the paper, our model performed to our expectations. Given that the target accuracies in the paper hovered around an average of 67%, we feel that our adapted model performs well enough when compared to the original proposed architecture.

The first, and perhaps most radical pivot we had to make was in dealing with the raw data and our decision to preprocess it. In the paper, the researchers based their model and classifications on sound pitch, operating in the frequency and time domains. However, due to

time constraints, we were unable to extract and transform our raw waves into the frequency and time domains, so we opted for using average frequency values given by the mel-spectrogram. This would mean that, when convolving and computing attention, we would have 1 less dimension to work with, since the proposed architecture in the paper has grid-based audio representations with frequency on one dimension and time on the other. Thus, because we had less inputs to work with, we had to scale down our learnable parameters. We did this by reducing the kernel sizes of all of the layers, and then ultimately by removing the 5th and final convolution layer.

5.1. Next Steps

If we had more time, we could further improve our preprocessing as described above to adhere more closely to what was done in the paper. This definitely would lead to a better trained model with more parameters to learn on, but given the time constraints for this project, we had to simplify the preprocessing step. Additionally, given more time, we could inspect the data more closely and make sure that we do not have over- or under-represented emotion classes, or search for other datasets that would give us a larger sample and potentially allow us to overcome the issues in the Results section. We initially considered the IEMOCAP dataset that was used in the research paper that we were basing our project on, but decided to start off with RAVDESS given limited time for preprocessing and training. However, next steps for this project would include using a larger and more involved dataset.

Our biggest takeaways are that, while CNNs with attention are suitable for this classification task of categorizing emotions from speech, there is much optimization required to get strong results from such a model. We gained an appreciation for the challenges of training models on small datasets, and learned not to underestimate the importance of having a good amount of knowledge about our data and its distribution before running such tasks.

6. Code Repository

Our codebase is accessible on GitHub at: https://github.com/matangans23/EmotionInMotion