Multimodal Speech Emotion Recognition and Ambiguity Resolution

Gaurav Sahu
David R. Cheriton School of Computer Science
University of Waterloo
Ontario, Canada
gaurav.sahu@uwaterloo.ca

***Abstract*—Identifying emotion from speech is a non-trivial task pertaining to the ambiguous definition of emotion itself. In this work, we adopt a feature-engineering based approach to tackle the task of speech emotion recognition. Formalizing our problem as a multi-class classification problem, we compare the performance of two categories of models. For both, we extract eight hand-crafted features from the audio signal. In the first approach, the extracted features are used to train six traditional machine learning classifiers, whereas the second approach is based on deep learning wherein a baseline feed-forward neural network and an LSTM-based classifier are trained over the same features. In order to resolve ambiguity in communication, we also include features from the text domain. We report accuracy, f-score, precision and recall for the different experiment settings we evaluated our models in. Overall, we show that lighter machine learning based models trained over a few hand-crafted features are able to achieve performance comparable to the current deep learning based state-of-the-art method for emotion recognition.**

***Index Terms*—multimodal speech emotion recognition, machine learning, deep learning**

I. Introduction

Communication is the key to human existence and more often than not, we have to deal with ambiguous situations. For instance, the phrase "This is awesome" could be said under either happy or sad settings. Humans are able to resolve ambiguity in most cases because we can efficiently comprehend information from multiple domains (henceforth, referred to as modalities), namely, speech, text and visual. With the rise of deep learning algorithms, there have been multiple attempts to tackle the task of Speech Emotion Recognition (SER) as in [1] [2] and [3]. However, this rise has made practitioners rely more on the power of the deep learning models as opposed to using domain knowledge to construct meaningful features and building models that perform well as well as are interpretable. In this work, we explore the implication of hand-crafted features for SER and compare the performance of lighter machine learning models with the heavily data-reliant deep learning models. Furthermore, we also combine features from the textual modality to understand the correlation between different modalities and aid ambiguity resolution. More formally, we pose our task as a multi-class classification problem and employ the two classes of models to solve that. For both the approaches, we first extract hand-crafted features from the time domain of the audio signal and train the respective models.

In the first approach, we train traditional machine learning classifiers, namely, Random Forests, Gradient Boosting, Support Vector Machines, Naive-Bayes and Logistic Regression. In the second approach, we build a Multi-Layer Perceptron and an LSTM [4] classifier to recognize emotion given a speech signal. The models are evaluated on the *IEMOCAP* [5] dataset under different settings, namely, *Audio-only*, *Text-only* and *Audio + Text* [1].

The rest of the paper is organized as follows: Section II describes existing methods in the literature for the task of speech emotion recognition; Section III gives an overview of the dataset used in this work and the pre-processing steps applied before feature extraction; Section IV describes the proposed models and implementation details; Results are reported in Section V, followed by the conclusion and future

[1]Code for all the experiments is available at http://tinyurl.com/y55dlc3m

scope of this work in Section VI.

II. Literature Review

In this section, we review some of the work that has been done in the field of speech emotion recognition (SER). The task of SER is not new and has been studied for quite some time in literature. A majority of the early approaches ([6] [7]) used Hidden Markov Models (HMMs) [8] for identifying emotion from speech. Recent introduction of deep neural networks to the domain has also significantly improved the state-of-the-art performance. For instance, [3] and [9] use recurrent autoencoders to solve the task. Recently, methods have also been proposed to efficiently combine features from multiple domains, such as, Tensor Fusion Networks [10] and Low-Rank Matrix Multiplication [11], instead of trivial concatenation.

This work aims to provide a comparative study between 1) deep learning based models that are trained end-to-end, and 2) lighter machine learning and deep learning based models trained over handcrafted features. We also investigate the information residing in multiple modalities and how their combination affects the performance.

III. Dataset

In this work, we use the *IEMOCAP* [5] released in 2008 by researchers at the University of Southern California (USC). It contains five recorded sessions of conversations from ten speakers and amounts to nearly 12 hours of audio-visual information along with transcriptions. It is annotated with eight categorical emotion labels, namely, anger, happiness, sadness, neutral, surprise, fear, frustration and excited. It also contains dimensional labels such as values of the activation and valence from 1 to 5; however, they are not used in this work.

The dataset is already split into multiple utterances for each session and we further split each utterance file to obtain `wav` files for each sentence. This was done using the start timestamp and end timestamp provided for the transcribed sentences. This results in a total of ∼10K audio files which are then used to extract features.

IV. Methodology

This section describes the data pre-processing steps followed by a detailed description of the features extracted and the two models applied to the classification problem.

TABLE I: Number of examples for each emotion

Class	Count
Angry	860
Happy	1309
Sad	2327
Fear	1007
Surprise	949
Neutral	1385
Total	7837

A. Data Pre-processing

a) Audio: A preliminary frequency analysis revealed that the dataset is not balanced. The emotions "fear" and "surprise" were under-represented and use upsampling techniques to alleviate the issue. We then merged examples from "happy" and "excited" classes as "happy" was under-represented and the two emotions closely resemble each other. In addition to that, we discard examples classified as "others"; they corresponded to examples that were labeled ambiguous even for a human. Applying the aforementioned operations resulted in 7837 examples in total. Final sample distribution for each of the emotions is shown in Table I.

b) Text: The available transcriptions were first normalized to lowercase and any special symbols were removed.

B. Feature Extraction

We now describe the handcrafted features used to train both, the ML- and the DL-based models.

1) Audio Features:

a) Pitch: Pitch is important because waveforms produced by our vocal cords change depending on our emotion. Many algorithms for estimating the pitch signal exist. We use the most common method based on *autocorrelation of center-clipped* frames [12]. Formally, the input signal $y[n]$ is center-clipped to give a resultant signal, $y_{clipped}[n]$:

$$y_{clipped}[n] = \begin{cases} y[n] - C_l, & \text{if } y[n] \geq C_l \\ 0, & \text{if } |y[n]| < C_l \\ y[n] + C_l, & \text{if } y[n] \leq C_l \end{cases} \quad (1)$$

Typically, C_l is nearly half the mean of the input signal and $[\cdot]$ denotes the discrete nature of the input

signal. Now, autocorrelation is calculated for the obtained signal $y_{clipped}$, which is further normalized and the peak values associated with the pitch of the given input $y[n]$. It was found that center-clipping the input signal resulted in more distinct autocorrelation peaks.

b) Harmonics: In the emotional state of anger or for stressed speech, there are additional excitation signals other than pitch ([13], [14]). This additional excitation is apparent in the spectrum as harmonics (see Figure 1) and cross-harmonics. We calculate harmonics using a median-based filter as described in [15]. First, the median filter is created for a given window size l, given by:

$$y[n] = median(x[n-k : n+k] | k = (l-1)/2) \quad (2)$$

where l is odd. For cases when l is even, the median is obtained as the mean of two values in the middle of the sorted list. This filter is then applied to S_h, the $h-$th frequency slice of a given spectrogram $\boldsymbol{S}$, to get harmonic-enhanced spectrogram frequency slice H_h as:

$$H_i = M(S_h, l_{harm}) \quad (3)$$

Here M is the median filter, i is the $i-$th time step and l_{harm} is the length of the harmonic filter.

c) Speech Energy: Since the energy of a speech signal can be related to its loudness, we can use it to detect certain emotions. Figure 2 shows the difference in energy levels of an "angry" signal v/s that of a "sad" signal. We use standard Root Mean Square Energy (RMSE) to represent speech energy using the equation:

$$E = \sqrt{\frac{1}{n}\sum_{i=1}^{n} y[i]^2} \quad (4)$$

RMSE is calculated frame by frame and we take both, the average and standard deviation as features.

d) Pause: We use this feature to represent the "silent" portion in the audio signal. This quantity is directly related to our emotions; for instance, we tend to speak very fast when excited (say, angry or happy, resulting in a low *Pause* value). The feature value is given by:

$$Pause = Pr(y[n] < t) \quad (5)$$

where t represents a carefully-chosen threshold of $\approx 0.4 * E$, E being the RMSE.

e) Central moments: Finally, we use the mean and standard deviation of the amplitude of the signal to incorporate a "summarized" information of the input.

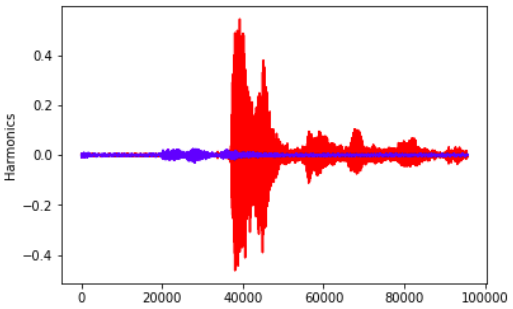


Fig. 1: Harmonics of angry (red) and sad (blue) audio signals

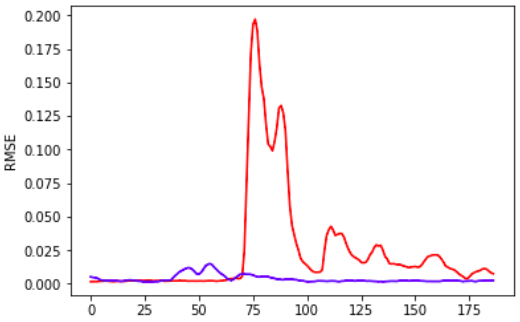


Fig. 2: RMSE plots of angry (red) and sad (blue) audio signals

2) Text Features::

a) Term Frequency-Inverse Document Frequency (TFIDF): TFIDF is a numerical statistic that shows the correlation between a word and a document in a collection or corpus. It consists of two parts:

- *Term Frequency:* It denotes how many times a word/token occurs in a document. The simplest choice is to use *raw count* of a token in a document (sentences, in our case).

- *Inverse Document Frequency:* This term is introduced to lessen the bias due to frequently occurring words in language such "the", "a" and "an". Usually, idf for a term t and a document D is defined as:

$$idf(t, D) = \log \frac{N}{|d \in D : t \in d|} \quad (6)$$

The denominator shows the frequency of documents containing the term t and N is the total number of documents.

Finally, TFIDF value for a term is calculated by taking the product of TF and IDF values.

C. Machine Learning Models:

This section describes the various ML-based classifiers considered in this work, namely, Random Forests, Gradient Boosting, Support Vector Machines, Naive-Bayes, and Logistic Regression

a) Random Forest (RF): Random forests are ensemble learners that operate by constructing multiple decision trees at training time and outputting the class that is the mode of the classes (classification) of the individual trees. It has two base working principles:

- Each decision tree predicts using a random subset of features [16]
- Each decision tree is trained with only a subset of training samples. This is known as bootstrap aggregating [17]

Finally, a majority vote of all the decision trees is taken to predict the class of a given input.

b) Gradient Boosting (XGB): XGB refers to eXtreme Gradient Boosting. It is an implementation of boosting that supports training the model in a fast and parallelized way. Boosting is another ensemble classifier combining a number of weak learners, typically decision trees. They are trained in a sequential manner, unlike RFs, using forward stagewise additive modeling. During the early iterations, the decision trees learned are simple. As training progresses, the classifier becomes more powerful because it is made to focus on the instances where the previous learners made errors. At the end of training, the final prediction is a weighted linear combination of the output from the individual learners [18].

c) Support Vector Machines (SVMs): SVMs are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. An SVM training algorithm essentially builds a non-probabilistic binary linear classifier (although methods such as Platt scaling [19] exist to use SVM in a probabilistic classification setting). It represents each training example as a point in space, mapped such that the examples of the separate categories are divided by a clear gap that is as wide as possible (this is usually achieved by minimizing the hinge loss). New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall. SVMs were originally introduced to perform linear classification; however, they can efficiently perform a non-linear classification using the kernel trick [20], implicitly mapping their inputs into high-dimensional feature spaces.

d) Multinomial Naive Bayes (MNB): Naive Bayes classifiers are a family of simple "probabilistic classifiers" based on applying Bayes' theorem with strong (naive) independence assumptions between the features. Under multinomial settings, the feature vectors represent the frequencies with which certain events have been generated by a multinomial $(p_1, \ldots, p_n)$ where p_i is the probability that event i occurs. MNB is very popular for document classification task in text [21] which too essentially is a multi-class classification problem.

e) Logistic Regression (LR): LR is typically used for binary classification problems [22], that is, when we have only two labels. In this work, LR is implemented in a one-vs-rest manner; six classifiers have been trained for each class and finally, we consider the class that is predicted with the highest probability.

Having trained the above classifiers, we take ensemble of the best performing classifiers and use it for comparison with the current state-of-the-art for emotion recognition on the IEMOCAP dataset.

D. Deep Learning Models

In this section, we describe the deep learning models used. Typically, Deep Neural Networks (DNNs) are trained in an end-to-end fashion and they are expected to "figure out" features completely on their own. However, training such a model can

take a lot of time as well as computational resources. In order to minimize the computational overhead, we directly feed the handcrafted features as input to these models and compare their performance with the traditional end-to-end trained counterparts. In this work, we implement two types of models:

a) Multi-Layer Perceptron (MLP): MLP belongs to a class of feed-forward neural network. It consists of at least three nodes: an input, a hidden and an output layer. All the nodes are interleaved with a non-linear activation function to stabilize the network during training time. Their expressive power increases as we increase the number of hidden layers upto a certain extent. Their non-linear nature allows them to distinguish data that is not linearly separable.

b) Long Short Term Memory (LSTM): LSTMs [4] were introduced for long-range context capturing in sequences. Unlike MLP, it has feedback connections that allow it to decide what information is important and what is not. It consists of a gating mechanism and there are three types of gates: input, forget and output. Their equations are mentioned below:

$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \tag{7}$$

$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \tag{8}$$

$$o_t = \sigma_g(W_o x_t + U_o g_h t - 1 + b_o) \tag{9}$$

$$c_t = f \cdot c_{t-1} + i_t \cdot \sigma_c(W_c x_t + U_c h_{t-1} + b_c) \tag{10}$$

$$h_t = o_t \cdot \sigma_h(c_t) \tag{11}$$

where initial values are $c_0 = 0$ and $h_0 = 0$ and $\cdot$ denotes the element-wise product, t denotes the time step (each element in a sequence belongs to one time step), x_t refers to the input vector to the LSTM unit, f_t is the forget gate's activation vector, i_t refers to the input gate's activation vector, o_t refers to the output gate's activation vector, h_t is the hidden state vector (which is typically used to map a vector from the feature space to a lower-dimensional latent space,) c_t is the cell state vector and W, U and b are weight and bias matrices which need to be learned during training. From figure 3, we see that an LSTM cell is able to keep track of hidden states at all time steps through the feedback mechanism.

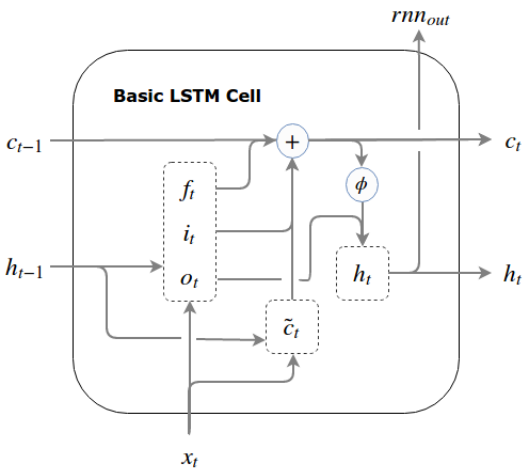


Fig. 3: Visualization of an LSTM cell

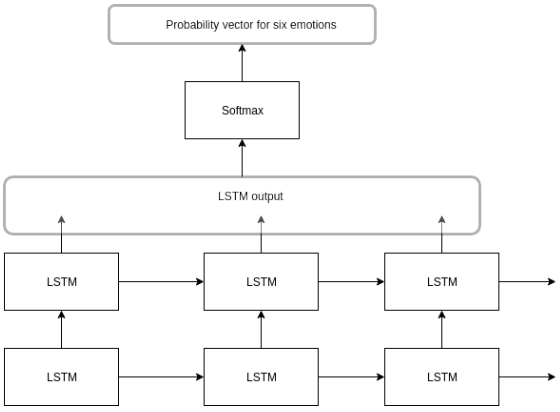


Fig. 4: LSTM classifier

Figure 4 shows the network implemented in this work. We feed the feature vectors as input to the network and finally pass the output of the LSTM network through a softmax layer to get probability scores for each of the six emotion classes. Since we are using feature vectors as input, we do not need another decoder network to transform it back from hidden to output space thereby reducing network size.

Models	Accuracy	F1	Precision	Recall
RF	56.0	**56.0**	57.2	**57.3**
XGB	55.6	**56.0**	56.9	56.8
SVM	33.7	15.2	17.4	21.5
MNB	31.3	9.1	19.6	17.2
LR	33.4	14.9	17.8	20.9
MLP	41.0	36.5	42.2	35.9
LSTM	43.6	43.4	53.2	40.6
ARE (4-class)	56.3	-	54.6	-
E1 (4-class)	56.2	45.9	**67.6**	48.9
E1	**56.6**	55.7	57.3	**57.3**

(a) Audio-only setting

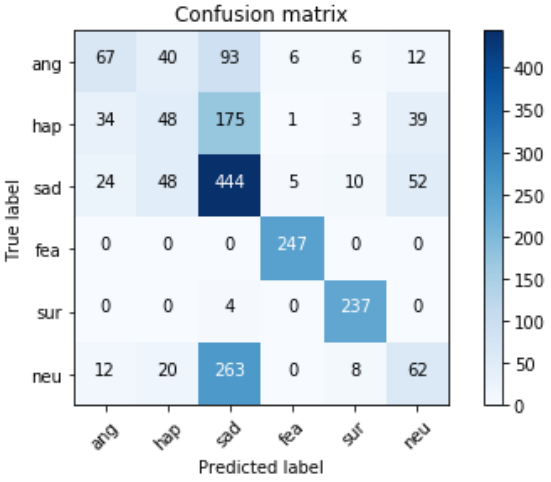
Models	Accuracy	F1	Precision	Recall
RF	62.2	60.8	65.0	62.0
XGB	56.9	55.0	70.3	51.8
SVM	62.1	61.7	62.5	**63.5**
MNB	61.9	62.1	**71.8**	58.6
LR	64.2	64.3	69.5	62.3
MLP	60.6	61.5	62.4	63.0
LSTM	63.1	62.5	65.3	62.8
TRE (4-class)	**65.5**	-	63.5	-
E1 (4-class)	63.1	61.4	**67.7**	59.0
E2	64.9	**66.0**	71.4	63.2

(b) Text-only setting

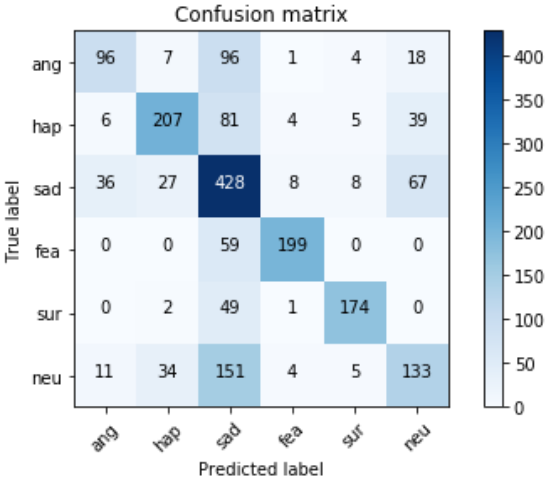
Models	Accuracy	F1	Precision	Recall
RF	65.3	65.8	69.3	65.5
XGB	62.2	63.1	67.9	61.7
SVM	63.4	63.8	63.1	65.6
MNB	60.5	60.3	70.3	57.1
MLP	66.1	68.1	68.0	69.6
LR	63.2	63.7	66.9	62.3
LSTM	64.2	64.7	66.1	65.0
MDRE (4-class)	**75.3**	-	71.8	-
E1 (4-class)	70.3	67.5	**73.2**	65.5
E2	70.1	**71.8**	72.9	**71.5**

(c) Audio+Text setting

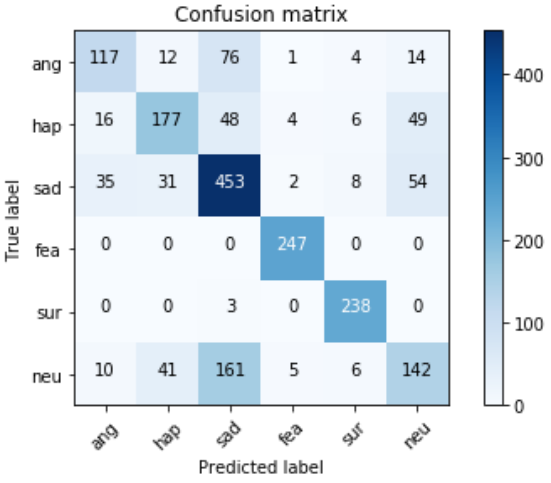
Fig. 5: Performance of different models; E1: Ensemble (RF + XGB + MLP); E2: Ensemble (RF + XGB + MLP + MNB + LR)



(a) E1, Audio-only setting



(b) E2, Text-only setting



(c) E2, Audio+Text setting

Fig. 6: Confusion Matrices of the our ensemble models; E1: Ensemble (RF + XGB + MLP); E2: Ensemble (RF + XGB + MLP + MNB + LR)

E. Experiments

Here, we describe the three different settings we conducted our experiments in:

- *Audio-only:* In this setting, we train all the classifiers using only the audio feature vectors described earlier.
- *Text-only:* In this setting, we train all the classifiers using only the text feature vectors (TFIDF vectors)
- *Audio+Text:* In this setting, we fuse the feature vectors from the two modalities. There have been some methods proposed to fuse vectors efficiently from multiple modalities but we simply concatenate the feature vectors from audio and text to obtain the combined feature vectors. Through this experiment, we would be able to infer how much information is contained in each of the modalities and how does fusion influence the model's performance.

F. Implementation Details

In this section, we describe the implementation details adopted in this work.

- We use `librosa` [23], a Python library, to process the audio files and extract features from them.
- We use `scikit-learn` and `xgboost` [24] [25], the machine learning libraries for Python, to implement all the ML classifiers (RF, XGB, SVM, MNB, and LR) and the MLP.
- We use `PyTorch` [26] to implement the LSTM classifiers described earlier.
- In order to regularize the hidden space of the LSTM classifiers, we use a *shut-off* mechanism, called dropout [27], where a fraction of neurons are not used for final prediction. This is shown to increase the robustness of the network and prevent overfitting.

We randomly split our dataset into a train (80%) and test (20%) set. The same split is used for all the experiments to ensure a fair comparison. The LSTM classifiers were trained on an NVIDIA Titan X GPU for faster processing. We stop the training when we do not see any improvement in validation performance for $>$10 epochs. Here, one epoch refers to one iteration over all the training samples. Different batch sizes were used for different models. Hyperparameters for the all the models under the three experiment settings could be found in the released repository.

G. Evaluation Metrics:

In this section, we first describe the various evaluation metrics used and report results for the three experiment settings.

a) Accuracy: This refers to the percentage of test samples that are classified correctly.

b) Precision: This measure tells us out of all predictions, how many are actually present in the ground truth (a.k.a. labels). It is calculated using the formula:

$$Precision = \frac{tp}{tp + fp} \tag{12}$$

c) Recall: This measure tells us how many correct labels are present in the predicted output. It is calculated using the formula:

$$Precision = \frac{tp}{tp + fn} \tag{13}$$

Here, tp, fp, and fn stand for true positive, false positive and false negative respectively. We can compute these values from the confusion matrix.

d) F-score: It is defined as the harmonic mean of precision and recall. This measure was included as accuracy is not a complete measure of a model's predictive power but F-score is since it is more normalized.

We compare our best performing models with the current state-of-the-art as mentioned in [2]. They employ three types of recurrent encoders, namely, ARE, TRE and MDRE denoting Audio-, Text- and Multimodal Dual- Recurrent Encoders respectively. It is important to mention that [2] only considers four emotions for classification, namely, angry, happy, sad and neutral as opposed to six in our case. In order to present a fair comparison of our method with theirs, we also run the experiments for the four classes (models with code 4-class in Figure 5).

V. Results

In this section, we discuss the performance of models described in Section IV.

From Figure 5, we can see that our simpler and lighter ML models either outperform or are

comparable to the much heavier current state-of-the-art on this dataset. A more detailed analysis follows:

a) Audio-only results: Results are especially interesting for this setting. Performance of LSTM and ARE reveals that deep models indeed need a lot of information to learn features as the LSTM classifier trained on eight-dimensional features achieves very low accuracy as compared to the end-to-end trained ARE. However, neither of them are able to beat the lighter E1 model (Ensemble of RF, XGB and MLP) which was trained on the eight-dimensional audio feature vectors. A look at the confusion matrix (Fig. 6a) reveals that detecting "neutral" or distinguishing between "angry", "happy" and "sad" is the most difficult for the model.

b) Text-only results: We observe that the performance of all the models for this setting is similar. This could be attributed to the richness of TFIDF vectors known to capture word-sentence correlation. We see from the confusion matrix (Fig. 6b) that our text-based models are able to distinguish the six emotions fairly well along with the end-to-end trained TRE. We observe that "sad" is the toughest for textual features to identify very clearly.

c) Audio+Text results: We see that combining audio and text features gives us a boost of $\sim$14% for all the metrics. This is clear evidence of the strong correlation between text and speech features. Also, this is the only case when the recurrent encoders seem to perform slightly better in terms of accuracy but at the cost of precision. The lower performance of E1 maybe be attributed to the trivial fusion method (concatenation) we use as simple concatenation for an ML model would still contain a lot of modality-specific connections instead of the desired inter-modal connections. The promising result here is that combining features from both the modalities indeed helped to resolve the ambiguity observed for modality-specific models as shown in Fig. 6c. We can say that the textual features helped in correct classification of "angry" and "happy" classes whereas the audio features enabled the model to detect "sad" better.

Overall, we can conclude that our simple ML methods are very robust to have achieved comparable performance even though they are modeled to predict six-classes as opposed to four in previous works.

A. Most Important Features:

In this section, we investigate which features contribute the most during prediction in this classification task. We chose the XGB model for this study and rank the eight audio features. We see that *Harmonic*, which is directly related to the excitation in signals, contributes the most. It is interesting to see that "silence" attributing to *Pause*, is almost as significant as standard deviation of the autocorrelated signal (related to pitch). The low contribution of central moments is expected as a signal is very diverse and an global/coarse feature would be unable to identify the nuances present in it.

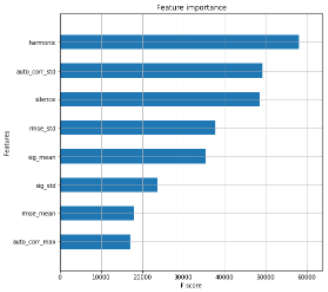


Fig. 7: Most important Audio features

VI. Conclusion and Future Work

In this work, we tackle the task of speech emotion recognition and study the contribution of different modalities towards ambiguity resolution on the IEMOCAP dataset. We compare, both, ML- and DL-based models and show that even lighter and more interpretable ML models can achieve performance close to DL-based models. We show that ensembling multiple ML models also lead to some improvement in the performance. We only extract a handful of time-domain features from audio signals. The audio feature-space could be made even richer if we could include some frequency-domain features too such as Mel-Frequency Cepstral Coefficients (MFCC) [28], Spectral Roll-off and additional time-domain features such as Zero Crossing Rate (ZCR) [29]. Also, better fusion methods such as TFN [10] and LMF [11] could be employed for combining speech and text vectors more effectively. It would also be interesting to see the scaling in the performance of ML models v/s DL models if include more data.

REFERENCES

[1] Z. Huang, M. Dong, Q. Mao, and Y. Zhan, "Speech emotion recognition using cnn," in *Proceedings of the 22nd ACM international conference on Multimedia*, pp. 801–804, ACM, 2014.

[2] S. Yoon, S. Byun, and K. Jung, "Multimodal speech emotion recognition using audio and text," in *2018 IEEE Spoken Language Technology Workshop (SLT)*, pp. 112–118, IEEE, 2018.

[3] K. Han, D. Yu, and I. Tashev, "Speech emotion recognition using deep neural network and extreme learning machine," in *Fifteenth annual conference of the international speech communication association*, 2014.

[4] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[5] C. Busso, M. Bulut, C.-C. Lee, A. Kazemzadeh, E. Mower, S. Kim, J. N. Chang, S. Lee, and S. S. Narayanan, "Iemocap: Interactive emotional dyadic motion capture database," *Language resources and evaluation*, vol. 42, no. 4, p. 335, 2008.

[6] A. Nogueiras, A. Moreno, A. Bonafonte, and J. B. Mariño, "Speech emotion recognition using hidden markov models," in *Seventh European Conference on Speech Communication and Technology*, 2001.

[7] B. Schuller, G. Rigoll, and M. Lang, "Hidden markov model-based speech emotion recognition," in *2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03).*, vol. 2, pp. II–1, IEEE, 2003.

[8] L. R. Rabiner and B.-H. Juang, "An introduction to hidden markov models," *ieee assp magazine*, vol. 3, no. 1, pp. 4–16, 1986.

[9] Y. Kim, H. Lee, and E. M. Provost, "Deep learning for robust feature generation in audiovisual emotion recognition," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 3687–3691, IEEE, 2013.

[10] A. Zadeh, M. Chen, S. Poria, E. Cambria, and L.-P. Morency, "Tensor fusion network for multimodal sentiment analysis," *arXiv preprint arXiv:1707.07250*, 2017.

[11] Z. Liu, Y. Shen, V. B. Lakshminarasimhan, P. P. Liang, A. Zadeh, and L.-P. Morency, "Efficient low-rank multimodal fusion with modality-specific factors," *arXiv preprint arXiv:1806.00064*, 2018.

[12] M. Sondhi, "New methods of pitch extraction," *IEEE Transactions on audio and electroacoustics*, vol. 16, no. 2, pp. 262–266, 1968.

[13] H. Teager and S. Teager, "Evidence for nonlinear sound production mechanisms in the vocal tract," in *Speech production and speech modelling*, pp. 241–261, Springer, 1990.

[14] G. Zhou, J. H. Hansen, and J. F. Kaiser, "Nonlinear feature based classification of speech under stress," *IEEE Transactions on speech and audio processing*, vol. 9, no. 3, pp. 201–216, 2001.

[15] D. Fitzgerald, "Harmonic/percussive separation using median filtering," 2010.

[16] Y. Amit, D. Geman, and K. Wilder, "Joint induction of shape features and tree classifiers," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, no. 11, pp. 1300–1305, 1997.

[17] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996.

[18] J. Friedman, T. Hastie, and R. Tibshirani, *The elements of statistical learning*, vol. 1. Springer series in statistics New York, 2001.

[19] J. Platt *et al.*, "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods," *Advances in large margin classifiers*, vol. 10, no. 3, pp. 61–74, 1999.

[20] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[21] A. M. Kibriya, E. Frank, B. Pfahringer, and G. Holmes, "Multinomial naive bayes for text categorization revisited," in *Australasian Joint Conference on Artificial Intelligence*, pp. 488–499, Springer, 2004.

[22] G. King and L. Zeng, "Logistic regression in rare events data," *Political analysis*, vol. 9, no. 2, pp. 137–163, 2001.

[23] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and music signal analysis in python," in *Proceedings of the 14th python in science conference*, pp. 18–25, 2015.

[24] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, *et al.*, "Scikit-learn: Machine learning in python," No. Oct, pp. 2825–2830, 2011.

[25] T. Chen, "Scalable, portable and distributed gradient boosting (gbdt, gbrt or gbm) library, for python, r, java, scala, c++ and more. runs on single machine, hadoop, spark, flink and dataflow," 2014.

[26] Facebook, "Pytorch," 2017.

[27] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[28] S. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *IEEE transactions on acoustics, speech, and signal processing*, vol. 28, no. 4, pp. 357–366, 1980.

[29] F. Gouyon, F. Pachet, O. Delerue, *et al.*, "On the use of zero-crossing rate for an application of classification of percussive sounds," in *Proceedings of the COST G-6 conference on Digital Audio Effects (DAFX-00), Verona, Italy*, 2000.