Machine Learning Methods for Speech Emotion Recognition: A Survey Study

Yenho Chen
The Center for Machine Learning
Georgia Institute of Technology
Atlanta, GA
yenho@gatech.edu

Hyeongju Choi
Electrical & Computer Engineering
Georgia Institute of Technology
Atlanta, GA
hchoi375@gatech.edu

Zaid Khan
Electrical & Computer Engineering
Georgia Institute of Technology
Atlanta, GA
zkhan78@gatech.edu

Mohammad Nikbakht
Electrical & Computer Engineering
Georgia Institute of Technology
Atlanta, GA
mohnikbakht@gatech.edu

***Abstract*—Speech and emotions are two of the most important modes of communication among human beings, which makes Speech Emotion Recognition (SER) a key component in Human-Computer Interaction (HCI) systems. In this work, we perform a comparative study of various ML-based emotion classification models using the audio-only modality from the Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS) dataset with 8 emotion classes. The ML models investigated in this paper include: logistic regression, multi-layer perceptron, support vector machines (SVM), and convolutional neural networks (CNN). For optimum feature selection, we utilize the backwards selection algorithm. Based on experimental results, our best-performing ML model was 2D-CNN, which obtained 48.9% accuracy, 47.9% precision, 48.9% recall, and 46.3% f1 score.**

***Index Terms*—Speech Emotion Classification; Audio Classification; Machine Learning; Deep Learning; Feature Selection**

I. Introduction

Speech Emotion Recognition (SER) is defined as detection of the emotional state of a person from their speech signal [1]. With advancements in Human Computer Interaction (HCI), SER has gained more attention since emotions are a major aspect of human interactions [2]. Many applications have been explored for SER such as quality measurement for call centers, aircraft pilot stress monitoring, quality improvement of gaming, and enhancing therapy sessions [3]–[5]. Emotion labeling is not unified but a common labeling known as the big six which consists of anger, disgust, fear, happiness, sadness, surprise, and neutral. However, studies have shown discrepancies between speaker and observer ratings resulting in unresolved challenges [2], [6].

We can split SER into two parts, feature extraction and classification. For feature extraction, global features from the entire signal and local features from the windowed signal have both been shown to be very effective for SER [2], [7], [8]. For classification, Gaussian Mixture models (GMM), hidden Markov model (HMM), support vector machine (SVM), Kernel Regression, K-nearest Neighbors (KNN), Maximum Likelihood Bayesian classifier (MLC), and Neural Networks (NN), have been previously explored by the literature [7], [9]–[11].

TABLE I
RAVDESS Database (Audio-only) attributes

Specification	Description
Number of Actors	24
Number of Male/Female Actors	12/12
Audio Sample-rate	48 kHz
Sample resolution	16 bit
Number of recordings	1440
Name of classes	calm, happy, sad, angry, fear, surprise, disgust and neutral.

The pandemic social restrictions led to a lack of interactions and psychological distress which affected the emotional and mental health of individuals impacted by the pandemic. Thus, a need for remote emotion monitoring is felt. The main goal of this project is to explore different machine learning algorithms on the SER task which can help address this need. In particular, we compare traditional statistical approaches to more modern deep learning methods based on the evaluation metrics to learn more about the structure of the data and the complexity of the SER task. An overall flowchart of the project is shown in Figure 1.

II. Methodology

We use the Ryerson Audio-Visual Database of Emotion Speech and Song (RAVDESS), an English language database commonly used to evaluate SER algorithms [12]. The database is gender-balanced containing 24 professional actors (12 female, 12 male), vocalizing two lexically-matched statements in a neutral North American accent. In this project, we use audio-only modality, which contains 1440 recordings samples. The output labels comprise of 8 emotions: *calm, happy, sad, angry, fear, surprise, disgust* and *neutral*. A summary of the RAVDESS audio-only data-set is shown in Table I.

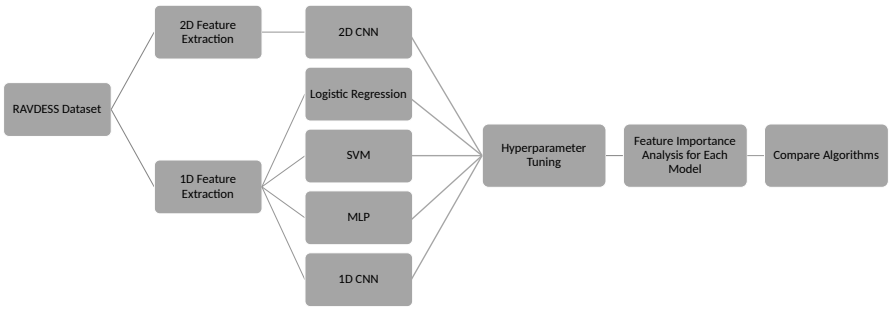


Fig. 1. Flow Chart visualization the model comparison procedure

A. Feature Extraction

Audio data is collected as a single time-series with high sampling rate. This prevents us from effectively using the raw data since the number of features will be much greater than the number of samples, which causes the learning problem to become ill-conditioned. To overcome this, we apply signal processing techniques to the audio samples and obtain features that exist in a much lower dimension. For local features, we divide audio signals into short time windows of length 50 ms with 50% overlap and compute Mel frequency cepstral coefficients (MFCCs), Mel-spectrogram, chromagram, and spectral contrast. By computing these features, we convert our raw audio signals into a perceptually meaningful space, representing the data in a way that's closer to how humans perceive the audio signals. Doing so has been shown to improve audio classifcation accuracy for SER [2], [5], [8]. Local features are all two dimensional matrices that span across time and respective feature channels of interest (i.e. frequency, energy, pitch) as shown in Figure 2. For global features, we compute mean, standard deviation, max, min, and range of each raw audio signal.

In order to train models with different architectures, we design two sets of features: one dimensional (1D) and two dimensional (2D) features. For 1D features, we average local feature channels over time to get a 1D vector and concatenate with the global features. A sample of the 1D feature vectors is shown in Figure 8. For 2D features, more complex multimodal techniques are required to incorporate both global and local features effectively. For this reason, we decide to include only local feature for 2D. A sample of the 2D feature images is shown in Figure 2. Furthermore, we transform all local features into a log scale which allows us to emphasize lower frequency features.

B. Dataset Split

We split our data samples into three sets: train, validation, and test. To prevent data leakage, we set aside actors 1-21 for training and validation, but held out actors 22-24 for a test set. This split was kept constant across all models to ensure a fair comparison. We standardize our data by z-scoring across the training population and using the same statistics when evaluating our models. We also account for imbalanced classes in the train set by oversampling less prevalent classes to prevent biased predictions based on uneven priors. For this dataset, only the neutral emotion required oversampling since there are exactly half as many sample as other emotions since there is no distinction between emotional intensity.

C. Feature Selection

By reducing the number of features used for our classification models, we can better pose our learning problem, speed up learning, improve model generalization, and enhance interpretability. We approach feature selection by implementing the backwards selection algorithm shown in Algorithm 1. We use the validation accuracy as the evaluation score throughout this procedure. The results of our selection algorithm for all models are shown in Table III.

Briefly, backwards selection begins by computing a baseline score on the full model, which includes all n features. For each step, we remove a single feature at a time, fit a new model, and compute its evaluation score. The set of features that yields the highest score is identified as the optimal subset for a particular step. We then repeat this procedure for $n-1$ features, constructing and evaluating reduced models, until all features are removed. Finally, we plot the backward steps against the evaluation metric and determine the optimal subset to be the step with the highest score. We show an example of backwards selection results in Figure 3a and see that by reducing the feature set, we can dramatically improve the performance of the MLP model.

D. Models

There are two classes of algorithms that we investigate: 1) traditional statistical learning methods and 2) deep learning methods. First, we develop linear discriminative models such as logistic regression and support vector machines to establish the baseline performance of the SER classification task. These methods have been shown to be reasonably effective at audio sample classification tasks [13]. Furthermore, we can use the test results to determine if our task is linearly separable given our extracted features, or if more complex models are required. Second, we develop three neural network architectures: 1) multi-layer perceptron (MLP), 2) 1D convolutional (CNN), and 3) 2D convolutional networks (CNN). In recent works, neural networks have been shown to achieve state-of-the-art

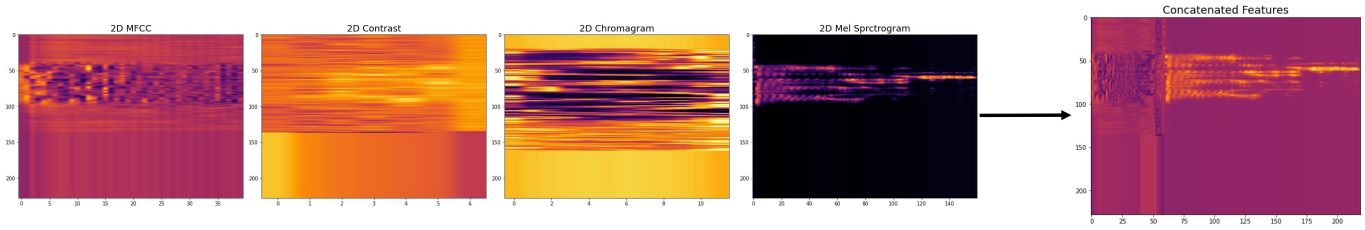


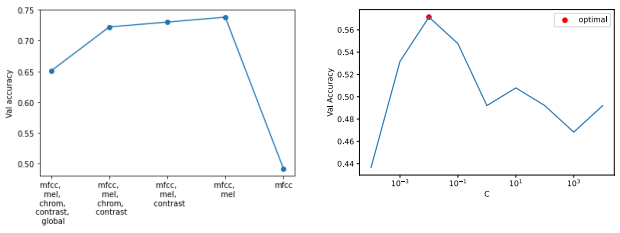
Fig. 2. 2D features extracted from one sample and concatenated

performance in audio classification tasks [14]. We use Scikit-Learn implementations of LR and SVM, and Pytorch to build all deep learning models.

1) Logistic Regression: The LR classifier is expected to learn a mapping from 1D features to 8-class output. The hyper-parameters that required tuning was the inverse regularization strength c. To find the best logistic regression configuration, we follow the hyperparameter search procedure from Section C in the Appendix to identify optimal values for c as well as the best feature subset while mininmizing computational costs. We find that the first backward step with $c = 0.01$ achieves the best validation performance as shown in Table III and Fig 3b.

2) Support Vector Machines: We explore SVMs to determine if maximum margin classifiers are more suitable for SER given our extracted features. We test two types of kernels: linear and radial basis function (RBF). For all models, we set up the inputs to be the concatenated 1D features and the outputs for the 8 emotion classes. The hyperparameters that required tuning was the inverse regularization strength c. We use the hyperparameter and feature selection procedure from Section C in the Appendix to find the best configuration for SVMs.

3) Multi-layer Perceptron: The MLP classifier is expected to learn a functional mapping between 1D features and 8-class output. We implement a 5-layer network as shown in Figure 5 to see if deeper models have the capacity of outperform the shallow linear models. Each layer uses includes batch normalization to stabilize their outputs before passing it through the activation function. We use ReLu as the activation function, which has been shown to speed up training for deep models [15]. Finally, we include dropout after each layer to mitigate overfitting and help with model generalization.



(a) (b)

Fig. 3. Feature selection and Hyperparameter tuning. a) Backwards feature selection steps for MLP. b) Hyperparameter search for Logistic Regression

After parameter tuning different combinations of 1D features were tested and compared through the backward selection algorithm (Algorithm 1) to pick the best set of features then the model was retrained on the whole training set and was tested on the test set and accuracy, precision, recall, and f1-score were recorded.

For training, we use the cross-entropy loss with the Adam optimizer with 1e-4 learning rate and He initialization. We use mini-batches of size 8 to introduce stochasticity during optimization so that we can avoid getting stuck in local optima. We optimize the model architecture by exploring a variety of configurations with different depths and size of layers. Once model is frozen, we perform feature selection with Algorithm 1, retrain the optimal configuration on the training set, and the evaluate model performance on the held-out test set.

4) 1D CNN: Convolutional Neural Networks have been shown to have reasonable performance for the SER task [16]. Similar to the MLP, the 1D CNN models are expected to learn a mapping from 1D features to 8-class output. For our model, we use ReLu as the activation function and employed dropout layers to mitigate overfitting. We implement the same training and architecture search procedure as the MLP and find that the best model consisted of four convolutional layers as shown in Figure 7.

5) 2D CNN: 2D CNNs are commonly used in the literature for their ability to aggregate information locally across feature channels and time [17]. Our 2D CNN models are expected to learn a mapping between 2D features and 8-class outputs. After implementing the same training and architecture search procedure as the MLP, we find that the best model included 3 convolutional layers followed by average pooling layers as shown in Figure 6. Additionally, we use ReLu as the activation function and incorporate dropout layers to mitigate overfitting.

E. Evaluation Metrics

To compare the performance of these models we report classification accuracy, precision, recall, and F1 scores on all dataset splits in Table II. Accuracy is a simple metric, but only works well if false positives and negatives have similar cost. Depending on the applications of SER, if the these costs are very different, it may be better to use precision, recall, and F1 score as a more suitable metric. We combine error counts across classes so that we can obtain a single combined score for each model.

TABLE II
RESULT COMPARISON FOR DIFFERENT ML MODELS

ML Model	Train				Validation				Test			
	accuracy	precision	f1 score	recall	accuracy	precision	f1 score	recall	accuracy	precision	f1 score	recall
LR	0.710	0.707	0.705	0.707	0.571	0.583	0.557	0.571	0.422	0.476	0.419	0.422
SVM_{lin}	0.798	0.800	0.795	0.798	0.595	0.612	0.589	0.595	0.378	0.431	0.365	0.378
SVM_{rbf}	0.979	0.979	0.979	0.979	0.611	0.611	0.605	0.611	0.400	0.428	0.385	0.400
MLP	0.997	0.997	0.997	0.997	0.738	0.748	0.736	0.738	0.417	0.425	0.384	0.384
1D-CNN	1	1	1	1	0.746	0.769	0.748	0.748	0.356	0.364	0.320	0.320
2D-CNN	0.951	0.952	0.951	0.951	0.714	0.762	0.712	0.714	**0.489**	**0.479**	**0.463**	**0.489**

TABLE III
BACKWARDS SELECTION STEPS WITH OPTIMAL FEATURE IN BOLD.

ML Models	Full Model	Backward Step			
		step 1	step 2	step 3	step 4
LR	mfcc, mel, chrom, contrast, global	**mfcc, mel, chrom, contrast**	mfcc, mel, contrast	mfcc, mel	mfcc
SVM_{lin}	mfcc, mel, chrom, contrast, global	**mfcc, mel, chrom, global**	mfcc, mel, chrom	mfcc, mel	mfcc
SVM_{rbf}	mfcc, mel, chrom, contrast, global	mfcc, mel, chrom, global	**mfcc, mel, global**	mfcc, mel	mel
MLP	mfcc, mel, chrom, contrast, global	mfcc, mel, chrom, contrast	mfcc, mel, contrast	**mfcc, mel**	mfcc
1D CNN	mfcc, mel, chrom, contrast, global	mfcc, mel, chrom, contrast	mfcc, mel, chrom	**mfcc, chrom**	mfcc
2D CNN	mfcc, mel, chrom, contrast	mfcc, chrom, mel	**mfcc, chrom**	mfcc	N/A

III. RESULTS & DISCUSSIONS

We compare the performance of Logistic regression and SVM models to establish baseline performance for the SER task. Looking at Table II, we first see that all models are able to learn meaningfully from the data since they all achieve test accuracies much higher than chance. Although both SVM models have higher train and validation accuracies than logistic regression (LR), we see that logistic regression performed the best on the test set, suggesting that SVM models are experiencing overfitting. Interestingly, LR also outperforms the nonlinear RBF kernel SVM, suggesting that maximum marginal classifiers are poorly suited towards the SER task. Given these observations, we establish logistic regression as the baseline reference model.

We first compare the performance of models that use 1D features. Looking at the deep learning models, we see that 1D CNN outperforms MLP on the validation set, but shows worse test performance. Furthermore, we see that LR outperforms both 1D deep learning models on the test set, despite having a significantly worse score on the validation set. We notice that that both deep learning models achieve near perfect accuracy on the train set while LR does not, indicating that the 1D models are overfit.

We now compare our 2D model with the 1D models. We see that the 2D CNN outperforms all 1D models on the test set, achieving 49% accuracy in contrast to LR's 42%. The superior performance of the 2D model indicates that aggregating information over time and feature channels is greatly beneficial for the SER task. Finally, we demonstrate preliminary results that global features are not a necessary feature to include, since the 2D model does not use them but outperforms models that do. However, this claim would need to be further qualified with experiments that directly compare 2D models with and without global features.

IV. CONCLUSION & FUTURE DIRECTIONS

In this work, we compared a variety of machine learning algorithms with optimized architectures for SER. We demonstrate that the ability to learn from data that includes time-series features can improve classification performance. This is clear since for 1D features, the linear LR model achieves the best test performance while the deep learning models achieve sub-optimal performance as a result of overfitting. Additionally, the 2D CNN, which uses time-series features, outperforms all 1D models in all metrics, further supporting the effectiveness of time-series features. Furthermore, we observe that MFCC, mel-spectrogram, and chromagram are consistently included in the optimal subset as shown in Table III. This implies their importance for SER, which agrees findings from current literature [16]. Given these results, we propose the following future directions to improve model performance.

1) **Design time-series features for 1D models.** Since time-series features work well for 2D models, we expect that they should also work for 1D models.
2) **Implement cross-subject cross validation procedure.** By obtaining a better error estimate, we can develop models that are better suited towards cross-subject SER.
3) **Explore more modern feature selection methods.** Backwards selection can give an inaccurate estimate of the optimal subset due to the multiple testing hypothesis. We can improve our pipeline by exploring more advanced statistical methods for feature selection.
4) **Simplify the classification problem.** It has previously been shown that we can simplify the classification problem to 4 classes based on the 2D valence-arousal emotion space and dramatically improve SER performance [18].

V. Team Member contributions

- Yenho: Designed backwards selection procedure. Implemented Logistic regression and SVM classifiers. Wrote models intro section, logistic regression and SVM subsections, and discussion on traditional statistical models. Also wrote the conclusion.
- Hyeongju: Assisted with feature extraction and evaluated 2D-CNN model by training and testing the performance. Wrote abstract section and provided plots and information on 2D-CNN.
- Zaid: Implemented the pipeline for feature extraction and preprocessing. Ran simulations for 1D-CNN and MLP to gather performance metrics for each model. Contributed in report writing including feature selection section, backward selection table and final result table.
- Mohammad: Designed and programmed the deep learning models MLP, 1D CNN, 2D CNN (PyTorch), refined the feature extraction pipeline, and programmed a dataset wrapper for PyTorch DataLoader. Wrote introduction, deep learning models sections, evaluation metrics, and parts of discussion and future directions.

References

[1] M. Selvaraj, R. Bhuvana, and S. P. Karthik, "Human speech emotion recognition," vol. 8, pp. 311–323, 02 2016.

[2] E. Lieskovská, M. Jakubec, R. Jarina, and M. Chmulík, "A review on speech emotion recognition using deep learning and attention mechanism," *Electronics*, vol. 10, no. 10, 2021. [Online]. Available: https://www.mdpi.com/2079-9292/10/10/1163

[3] D. J. France, R. G. Shiavi, S. Silverman, M. Silverman, and M. Wilkes, "Acoustical properties of speech as indicators of depression and suicidal risk," *IEEE transactions on Biomedical Engineering*, vol. 47, no. 7, pp. 829–837, 2000.

[4] F. Burkhardt, J. Ajmera, R. Englert, J. Stegmann, and W. Burleson, "Detecting anger in automated voice portal dialogs." in *INTERSPEECH*, 2006.

[5] M. S. Hossain, G. Muhammad, B. Song, M. M. Hassan, A. Alelaiwi, and A. Alamri, "Audio–visual emotion-aware cloud gaming framework," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 25, no. 12, pp. 2105–2118, 2015.

[6] K. P. Truong, D. A. Van Leeuwen, and F. M. De Jong, "Speech-based recognition of self-reported and observed emotion in a dimensional space," *Speech communication*, vol. 54, no. 9, pp. 1049–1063, 2012.

[7] M. Deriche *et al.*, "A two-stage hierarchical bilingual emotion recognition system using a hidden markov model and neural networks," *Arabian Journal for Science and Engineering*, vol. 42, no. 12, pp. 5231–5249, 2017.

[8] D. Pravena and D. Govind, "Significance of incorporating excitation source parameters for improved emotion recognition from speech and electroglottographic signals," *International Journal of Speech Technology*, vol. 20, no. 4, pp. 787–797, 2017.

[9] X. Mao, L. Chen, and L. Fu, "Multi-level speech emotion recognition based on hmm and ann," in *2009 WRI World congress on computer science and information engineering*, vol. 7. IEEE, 2009, pp. 225–229.

[10] T. Seehapoch and S. Wongthanavasu, "Speech emotion recognition using support vector machines," in *2013 5th international conference on Knowledge and smart technology (KST)*. IEEE, 2013, pp. 86–91.

[11] H. M. Fayek, M. Lech, and L. Cavedon, "Evaluating deep learning architectures for speech emotion recognition," *Neural Networks*, vol. 92, pp. 60–68, 2017.

[12] S. R. Livingstone and F. 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*, vol. 13, no. 5, p. e0196391, 2018.

[13] J. P. D. Galileu and J. M. Ribeiro da Silva Tavares, "Urban sound event classification for audio-based surveillance systems," *Universidade Do Porto MSc Thesis*, 2020.

[14] H. Lu, H. Zhang, and A. Nayak, "A deep neural network for audio classification with a classifier attention mechanism," *ArXiv*, vol. abs/2006.09815, 2020.

[15] K. Hara, D. Saito, and H. Shouno, "Analysis of function of rectified linear unit used in deep learning," pp. 1–8, 2015.

[16] Y. Li, C. Baidoo, T. Cai, and G. A. Kusi, "Speech emotion recognition using 1d cnn with no attention," in *2019 23rd international computer science and engineering conference (ICSEC)*. IEEE, 2019, pp. 351–356.

[17] S. Kwon *et al.*, "A cnn-assisted enhanced audio signal processing for speech emotion recognition," *Sensors*, vol. 20, no. 1, p. 183, 2020.

[18] Y.-H. Yang and H. H. Chen, "Machine recognition of music emotion: A review," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 3, no. 3, pp. 1–30, 2012.

APPENDIX

A. 1D Features Additional Features

A diagram of the 1D features is included in Figure 8. This represents the 2D features with each feature channel averaged over time concatenated with the extracted global features.

B. Backward Selection Algorithm

The full backward selection algorithm is detailed in Algorithm 1.

C. Hyperparameter Search Procedure for Linear Models

To minimize computational complexity, we disentangle hyperparameter search from backwards selection using the following procedure. Although c is only optimal for the full model, we assume that it is within a reasonable range for the reduced models. Additionally, this procedure biases the optimal subset towards the full model, which can be favorable in the sense that the algorithm will only select a reduced subset if it can outperform the full model on an unoptimized set of hyperparameters.

1) **Identify a reasonable value for c.** We initially set c to be the optimal value for the full model. Although we recognize that c is only optimized for the full model, we assume that it is a good estimate for the reduced models. Furthermore, this biases our procedure towards the full model, and only eliminates features if there is a significant increase in performance.
2) **Perform backwards selection.** We use the fixed c value found previously for the entirety of feature selection. This allows us to determine the best subset of features for the model.
3) **Perform hyperparameter search on optimal subset.** We tested inverse regularization strength on a log scale such that $c \in [10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}, 10^{3}, 10^{4}]$. This gives us a coarse approximation of the optimal hyperparameters for a model built on the optimal subset.
4) **Retrain optimal configuration on all available data.** By retraining on both train and validation sets, we hope to achieve better performance on the test set.

D. Hyperparameter Tuning Additional Figures

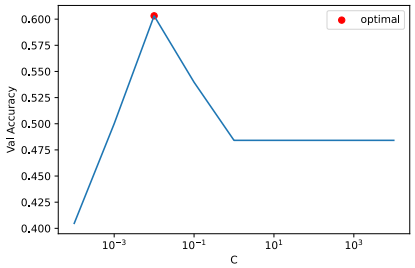
For completeness, results of hyperparameter search for the remaining statistical models are shown in Figures 4a and 4b. Notice that optimal hyperparameters can vary drastically based on the type of model, further qualifying the importance the need to select appropriate values.

E. Model Architectures

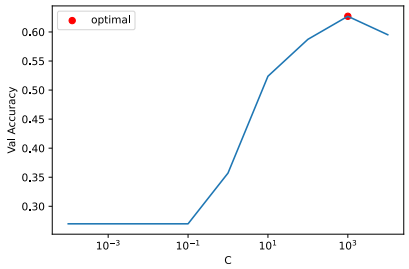
Model architecture diagrams for MLP, 2D CNN, and 1D CNN are shown in Figure 5, 6 and 7, respectively.

F. Confusion Matrices for Test Set

For completeness, we include Figure 9 to show the class-wise performance of each model.



(a)



(b)

Fig. 4. Hyperparameter Grid Search for the regularization strength a) SVM linear kernel b) SVM RBF kernel

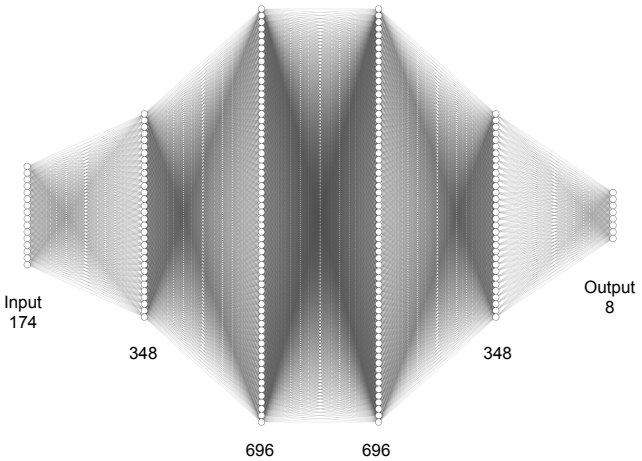


Fig. 5. MLP architecture

Algorithm 1 Backwards selection algorithm

Input: $X_n = [\{x_1, ..., x_n\}]_{i=1}^m$ feature set, $y = \{y_1, \ldots, y_m\}$ emotion labels, n total number of features, m number of data samples, $f(x;\theta)$ model, θ model parameters

Output: k Optimal feature subset

Initialize: $k \leftarrow \{1, \ldots, n\}$ all features, $j \leftarrow 1$ step number, $P_1 \leftarrow k$ backwards path, A step scores, a step scores for inner loop

$X_{k,\text{train}}, X_{k,\text{val}}, y_{\text{train}}, y_{\text{val}} \leftarrow \text{TrainTestSplit}(X_k, y)$

$\theta^* = \text{fit}(f(x;\theta), X_{k,\text{train}}, y_{\text{train}})$ ▷ fit full model

$A_1 \leftarrow \frac{1}{n_{\text{val}}} \sum_{i=1}^{m} \mathbb{I}[f(X_{k,\text{val}};\theta^*) = y_{\text{val}}]$ ▷ evaluate

while $n > 0$ **do**

  for $i \in \{1 \ldots, n\}$ **do** ▷ eliminate one feature at a time

    $k \leftarrow \{1, \ldots, n\}/i$

    $\theta^* = \text{fit}(f(x;\theta), X_{k,\text{train}}, y_{\text{train}})$ ▷ fit reduced model

    $a_{i,n} \leftarrow \frac{1}{n_{\text{val}}} \sum_{i=1}^{m} \mathbb{I}[f(X_{k,\text{val}};\theta^*) = y_{\text{val}}]$ ▷ evaluate

  end for

  $k \leftarrow \{1, \ldots, n\}/\text{argmax}a_n$ ▷ record subset with best score

  $A_j \leftarrow \text{max}a_n$

  $P_j \leftarrow k$ ▷ record path

  $j \leftarrow j + 1$

  $n \leftarrow n - 1$

end while

$j^* \leftarrow \text{argmax}A$

$k \leftarrow P_{j^*}$ ▷ choose optimal subset

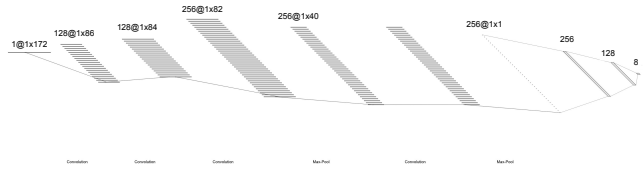


Fig. 6. 2D-CNN architecture

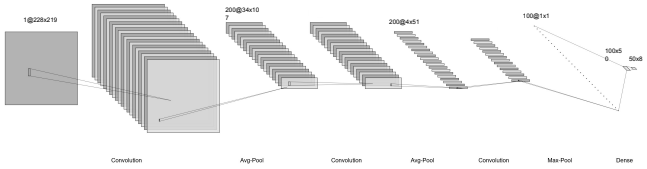


Fig. 7. 1D-CNN architecture

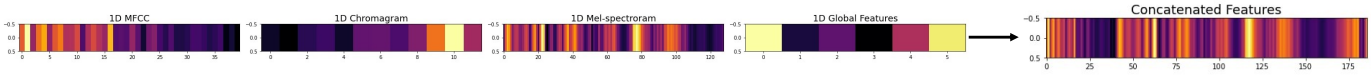
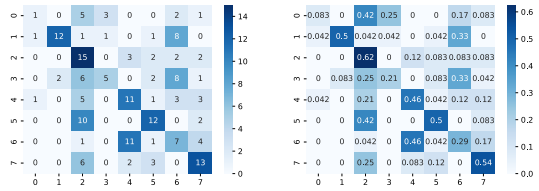
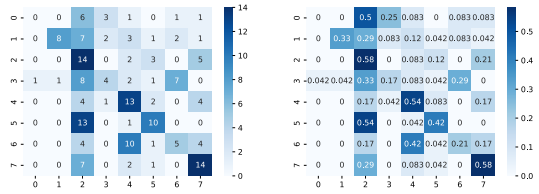


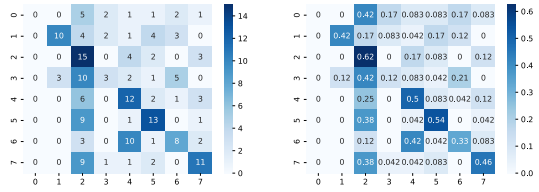
Fig. 8. 1D CNN features and concatenation



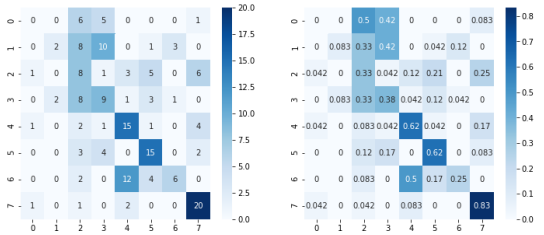
(a)



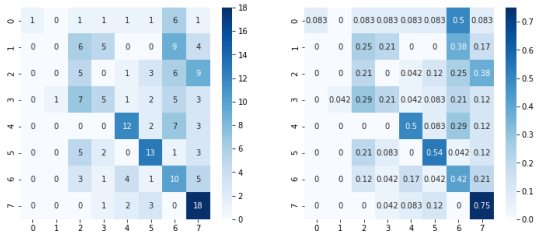
(b)



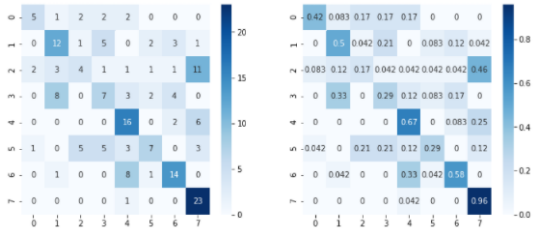
(c)



(d)



(e)



(f)

Fig. 9. Confusion matrices for held-out test set a) Logistic Regression b) SVM linear kernel c) SVM RBF kernel d) MLP e) 1D CNN f) 2D CNN