Speech Emotion Recognition

By Maria Atef

Abstract

Speech emotion recognition (SER) is a field of study that aims to recognize emotions from a person's speech. It has the potential to improve human-computer interaction and healthcare. In this research we review the state of the art in SER, including the use of deep learning models and new acoustic and temporal features. In this his research we build on the existing literature to develop the best-performing machine learning model to identify different emotions such as Happy, Sad, Angry, etc. Our target goal is to process human speech signals then display emotion. We are using Librosa package to extract features from the speech and for the dataset we are using the RAVDESS dataset which we use as an experimental dataset.

Keywords *Speech emotion recognition, RAVDESS, Librosa python library, Waveform, Mel spectrogram, Data augmentation techniques, Convolutional Neural Network (CNN), Decision Tree, K-Nearest Neighbors (KNN), Feature extraction, MFCC (Mel-Frequency Cepstral Coefficients), Data preprocessing, Standardization, One hot encoder, Train-test split, Model evaluation, Accuracy, Confusion matrix, Precision, Recall and F1-score .*

1.Introduction

Emotion recognition is increasingly vital in Human-Computer Interaction (HCI), facilitated by advancements in deep learning technology. This allows for the recognition of human emotions across speech, text, and facial expressions. Speech Emotion Recognition (SER) particularly holds promise in fields like psychological assessment, robotics, and mobile services. However, defining emotions presents challenges, with discrete models simplifying emotions into basic categories like 'happy' or 'angry', and dimensional models conceptualizing emotions as points in multidimensional emotional space. While current SER research primarily focuses on discrete models, the complexity of human emotions suggests a need for further exploration and refinement in emotion recognition systems.

2. Materials and Methods

2.1 Data

2.1.1 Data Description

The RAVDESS dataset was chosen for its comprehensive coverage of emotional states, including seven distinct categories such as Happy, Sad, Furious, Afraid, Disgust, Surprise, and a neutral baseline for each performer.

Data collection involved 24 trained performers, with an equal representation of 12 males and 12 females, ensuring gender balance in the dataset. (Fig1)

Recordings were conducted in a controlled setting, with standardized statements spoken in American English to maintain consistency across audio files.

The dataset includes two types of files: speech files comprising 1440 recordings and audio files comprising 1012 recordings, both adhering to a 16-bit bitrate and a 48 kHz sampling rate in WAV raw audio format to preserve original data integrity.

XXX-X-XXXX-XXXX-X/XX/$XX.00 ©20XX IEEE

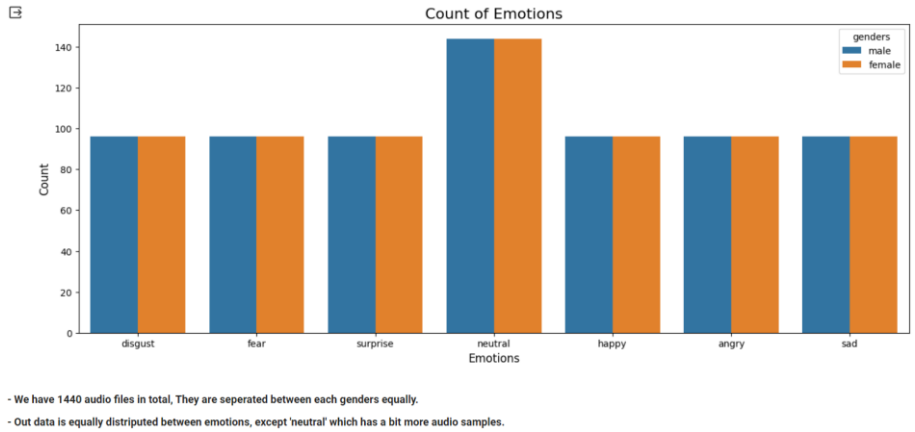

Figure 1: Out data is equally distributed between emotions, except 'neutral' which has a bit more audio samples.

2.1.2 visualize audio data

In the research methodology, an analysis technique was employed to visualize audio data, specifically aimed at examining emotional content. This technique involved plotting both the waveform and Mel spectrogram of audio recordings corresponding to different emotional states .Subsequently, two plots were generated side by side. The first plot displayed the waveform of the audio, representing amplitude variations over time. This waveform plot facilitated the visual inspection of the audio signal's shape and intensity, providing insights into its temporal characteristics.

The second plot showcased the Mel spectrogram of the audio. This spectrogram, derived from the Short-Time Fourier Transform (STFT), depicted the distribution of signal energy across different frequencies over time. By converting the power spectrogram to decibels and displaying it on a Mel scale, the Mel spectrogram offered a more perceptually relevant representation of audio features, particularly for human speech. Both plots were annotated with the respective emotion associated with the audio data, enabling researchers to correlate visual patterns with emotional states. Additionally, axis labels and titles were provided for clarity and context. And we can see the differences between the seven target labels (fig 2)

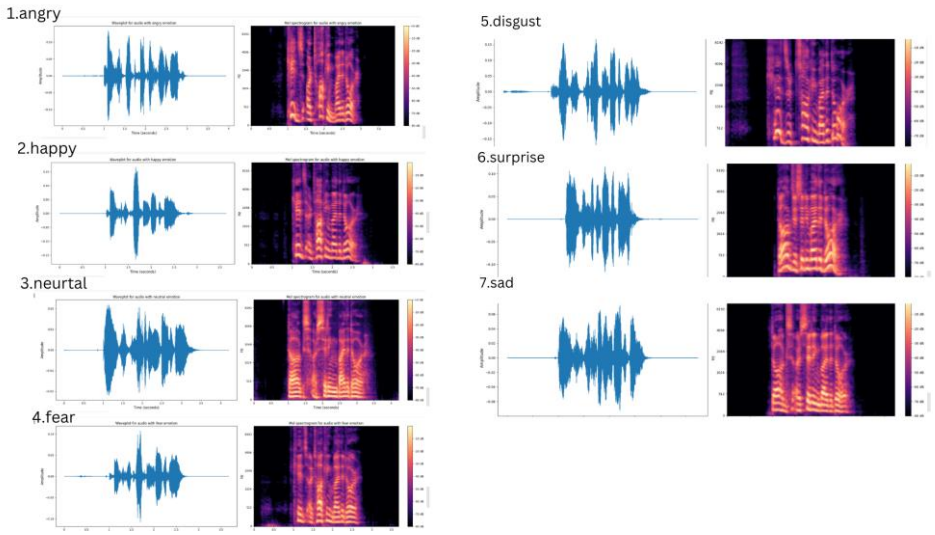

Fig 2: visualize audio data

3.Data Preprocessing

3.1 Data Augmentation

The augmentation techniques used in the provided code are common methods employed in the field of audio signal processing to increase the diversity of the training data and improve the robustness of machine learning models. By applying these techniques, models can learn to generalize better to unseen variations in the data and perform more effectively.

we use four augmentation techniques :

1. noise(data):

- This function adds random noise to the input audio data.

- `noise_amp` calculates the amplitude of the noise to be added based on a fraction of the maximum amplitude of the data.

- Noise is added to it using `np.random.normal()` function.

2. shift(data):

- This function shifts the audio data by a random number of samples within a certain range.

- `s_range` is calculated as a random integer within the range of -5 to 5 milliseconds.

- The `np.roll()` function is used to perform the shifting operation.

3. stretch(data, rate=0.8):

- This function stretches or compresses the audio data in time.

- It uses `librosa.effects.time_stretch()` function with the specified stretching rate.

4. pitch(data, sample_rate):

- This function applies pitch shifting to the audio data.

- It calculates the pitch shift amount (`pitch_change`) randomly.

- `librosa.effects.pitch_shift()` function is used to perform the pitch shifting.

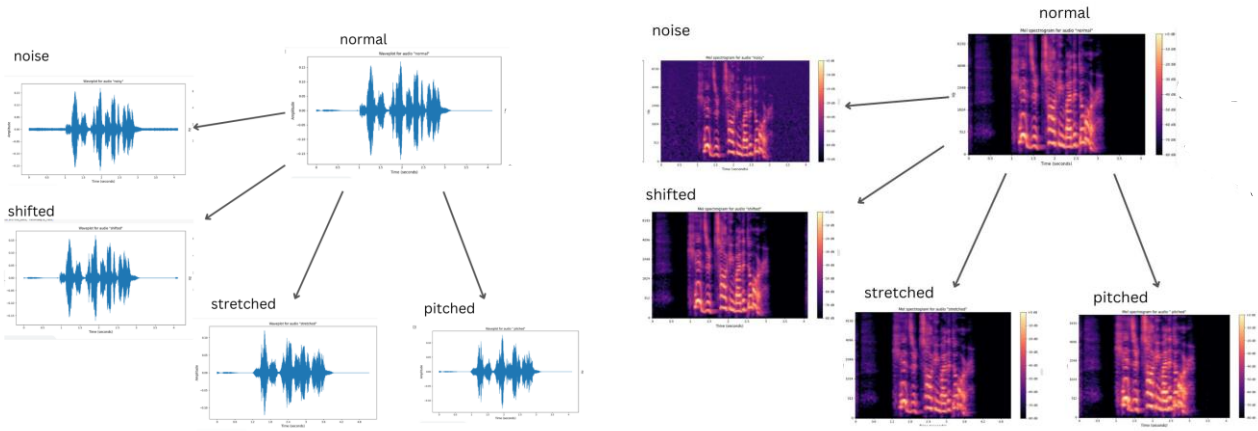

Fig 3,4: effect of data Augmentation on one example

3.2 Feature Extraction

There is several techniques .to make "extract_features" This function combines the ZCR, RMSE, and MFCC features

we select MFCC (Mel-Frequency Cepstral Coefficients) serve as a cornerstone in audio feature extraction due to their ability to capture essential aspects of sound. By aligning with the human auditory system's perception through the Mel scale, MFCCs offer a representation that closely mirrors how humans interpret sound frequencies. Moreover, they excel in dimensionality reduction, distilling complex audio signals into manageable sets of coefficients while preserving vital information. Their robustness in noisy environments further enhances their utility, ensuring reliable performance even in challenging acoustic conditions. Computationally efficient algorithms underpin the calculation of MFCCs, facilitating real-time processing tasks. Additionally, their inherent invariance to certain speech variations, such as speaker differences and speaking rates, underscores their versatility across various audio processing applications. In essence, MFCCs

provide a succinct yet comprehensive portrayal of spectral characteristics pivotal for tasks like speech recognition and speaker identification.

'get_features', is designed to extract various features from an audio file located at the specified path. It utilizes the librosa library to load the audio file with a defined duration and offset. The function then proceeds to extract features from the original audio data and from modified versions of it, including noisy, shifted, stretched, and pitched variants. Each set of features is obtained through the extract_features function. After all features are collected, they are padded to ensure uniform length across all feature sets. Finally, the function returns an array containing the extracted features, ready for further processing or analysis. Overall, this function provides a comprehensive approach to extracting diverse features from audio data, facilitating tasks such as audio classification or analysis.

'joblib' library to parallelize the extraction of features from audio files stored in 'df.path', each associated with an emotion label stored in 'df.emotions'. The 'process_feature' function is defined to extract features from each audio file, and then X and Y lists are populated with the extracted features and their corresponding emotion labels, respectively. The Parallel function from 'joblib' is employed to parallelize the feature extraction process across multiple CPU cores, enhancing efficiency. Finally, the extracted features and emotion labels are aggregated into X and Y lists, respectively, for further processing or analysis.

made a CSV file will contain the extracted features along with their associated emotion labels

3.3 data preprocesssing

-fill nans values with zeros ensures that the structure of the data remains intact and effective processing.

- encoding using 'one hot encoder' resulting numerical representation retains the interpretability of the original categories, providing an efficient and effective means to process categorical labels for training and inference tasks.

-spliting the data into train and test sets (80% train 20%test).

- standardize the features in both the training (`x_train`) and testing (`x_test`) datasets, which is a common preprocessing step in machine learning. Standardization involves subtracting the mean and scaling to unit variance, ensuring that all features have a similar scale. By applying the `fit_transform` method to the training data, the scaler computes the mean and standard deviation for each feature and standardizes the data accordingly. The same transformation is then applied to the testing data using the `transform` method, ensuring consistency between the two datasets. Enhancing model performance and convergence by standardizing feature values. (fig 6)

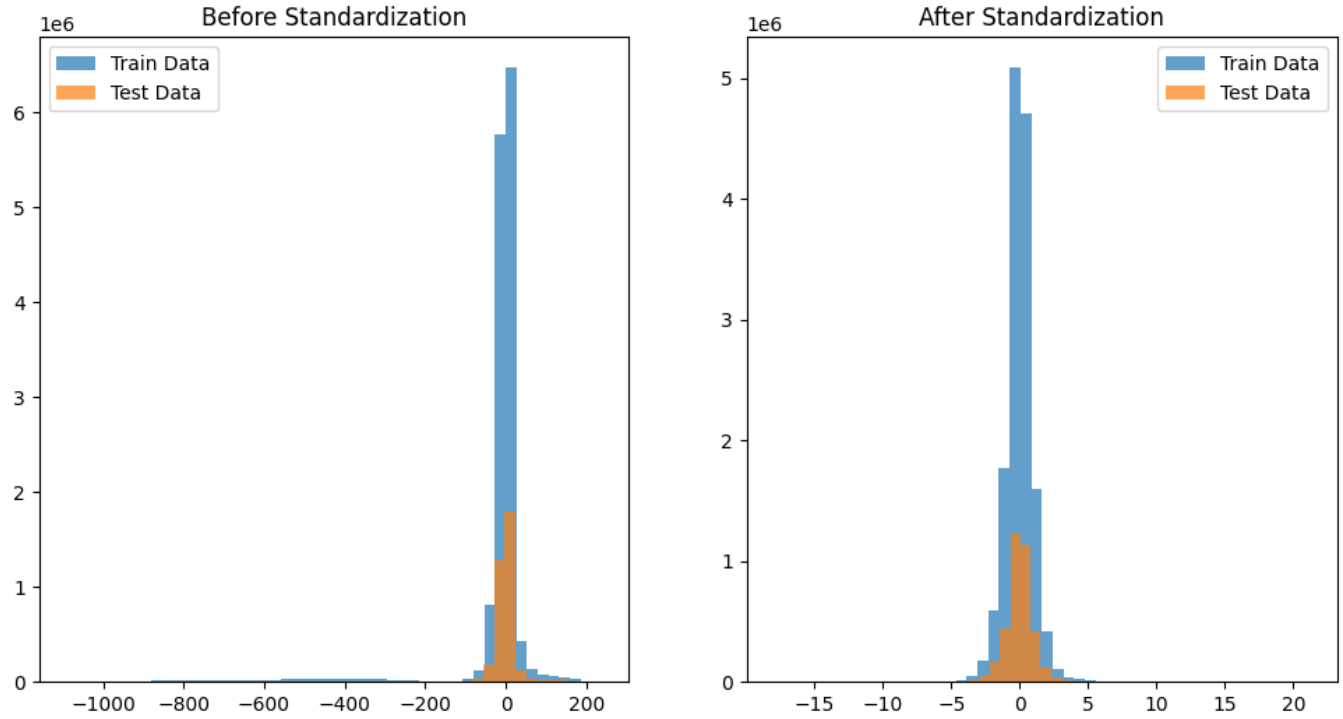

fig 6: Preprocessing standardize

4. model

We will discuss differents models

4.1 Decision Tree

derivative Decision Tree Classifier (dt_model) using the Gini impurity criterion for splitting nodes, allowing for an unlimited depth (max_depth=None), and fixing the random state for reproducibility. After training the model on the training data (x_train and y_train), it evaluates the model's performance on both the training and test sets. The training set score indicates how well the model fits the training data, while the test set score measures its performance on unseen data. In this case, the training set score is perfect (1.000), suggesting potential overfitting, while the test set score is considerably lower (0.415), indicating poor generalization to unseen data. This suggests that the model might be too complex and is not effectively capturing the underlying patterns in the data, leading to overfitting. so we have to change the model

4.2 K-Nearest Neighbors

K-Nearest Neighbors Classifier (knn) with n_neighbors set to 4, indicating that it will consider the labels of the four nearest neighbors to classify each data point. After training the model on the training data (x_train and y_train), it predicts labels for the test data (x_test) using the trained model. The training set score indicates the accuracy of the model on the training data, while the test set score measures its performance on unseen data. In this case, the training set score is 0.673, and the test set score is 0.494. These scores suggest that the model performs moderately well on the training set but shows only slightly better performance than random guessing on the test set. This indicates a potential need for further model optimization or exploration of other algorithms to achieve better generalization to unseen data. or use another model

So, we notice that Conventional machine learning struggles with audio recognition classification. Transitioning to deep learning is necessary. Deep learning's automatic pattern extraction from complex data offers promise for audio classification. Leveraging deep neural networks' hierarchical learning enables more accurate models for audio data. Thus, embracing deep learning improves performance in audio recognition tasks.

4.3 Deep Learning model

Instead of traditional machine learning models, we employ a deep learning approach, specifically utilizing a Convolutional Neural Network (CNN) model built using the Keras Sequential API, to achieve superior results as appear in (fig 7)

4.3.1 Model Architecture:

4.3.1.1 Input Layer (Convolutional Layer):

The Conv1D layer performs 1D convolution on the input time series data. In this case, it has 64 filters, each with a kernel size of 3. These filters slide over the input data to capture local patterns, with the ReLU (Rectified Linear Unit) activation function applied to introduce non-linearity.

4.3.1.2 Pooling Layer:

Following the convolutional layer, a MaxPooling1D layer is employed to downsample the output, reducing the computational complexity and the risk of overfitting. A pool size of 2 is specified, meaning that the maximum value within each 2-element window is retained.

4.3.1.3 Flattening Layer:

The Flatten layer reshapes the output from the previous layer into a one-dimensional array, preparing it to be fed into the subsequent fully connected layers.

4.3.1.4 Dense Layers:

A Dense layer with 64 neurons and ReLU activation follows the flattened output. This layer enables the network to learn complex patterns by providing connections between all neurons of the previous layer and the current layer.

Dropout regularization is applied after the first Dense layer to randomly drop 30% of the neurons during training. This technique helps prevent overfitting by introducing redundancy and reducing interdependency among neurons.

The final Dense layer has 7 neurons, corresponding to the number of classes in the classification task, with softmax activation. Softmax converts the raw output into probabilities, facilitating multi-class classification by assigning a probability distribution over the classes.

4.3.2 Model Compilation:

4.3.2.1 Optimizer:

The Adam optimizer is used to update the model weights based on the gradients of the loss function. Adam is well-suited for tasks with large datasets and high-dimensional parameter spaces, as it adapts the learning rates for each parameter individually.

4.3.2.2 Loss Function:

Categorical cross-entropy loss is chosen as the loss function, which measures the dissimilarity between the true class distribution and the predicted class probabilities. It is commonly used for multi-class classification tasks.

4.3.2.3Metrics:

The model's performance during training is evaluated using accuracy, which measures the proportion of correctly classified samples out of the total number of samples.

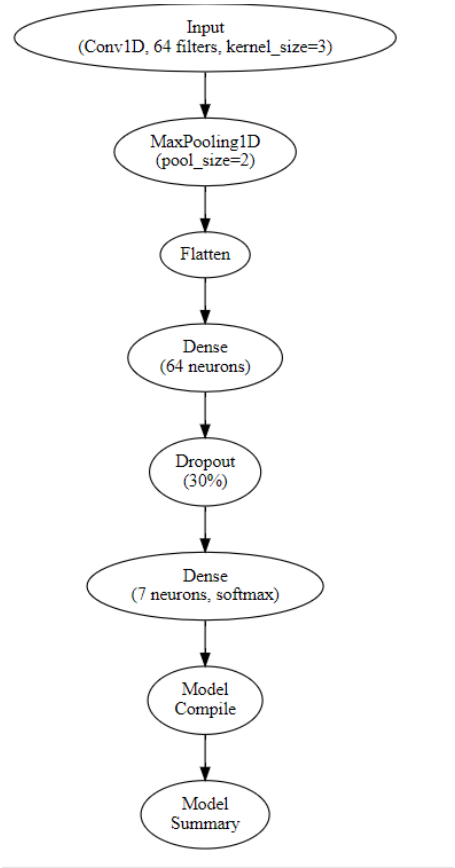

Fig 7: CNN model flow chart

5. Evaluation

5.1 CNN accuracy

The training accuracy of the model is approximately 97.27%, while the test accuracy stands at approximately 72.5%. These metrics serve as indicators of the model's performance in generalizing to unseen data, highlighting its efficacy beyond the training dataset. Moreover, when compared to conventional machine learning methods, such as decision trees and k-nearest neighbors, the attained accuracy demonstrates notable improvement, suggesting the model's superior predictive capabilities.

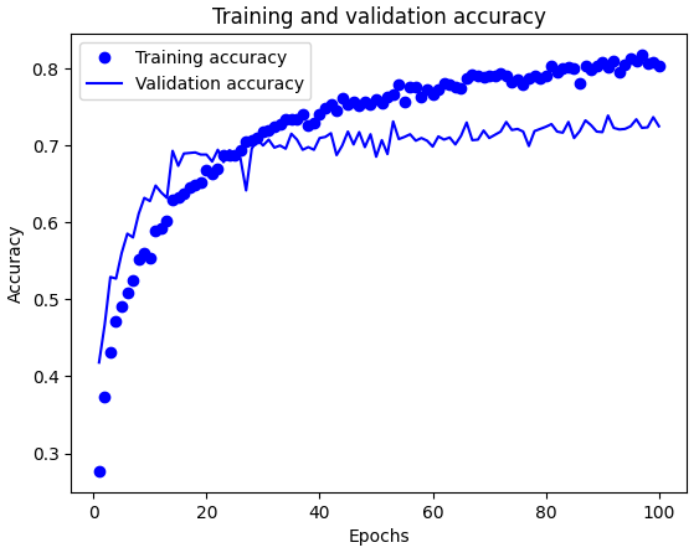

Fig 8: evaluation the model with epochs

The graph illustrates a consistent increase in accuracy across epochs for both the training and validation datasets. (Fig 8)

5.2 confusion matrix

The classification report presents a comprehensive assessment of a model's performance across multiple classes in a multiclass classification task.

Precision, representing the ratio of correctly predicted instances to all instances predicted as belonging to a specific class, ranges from 0.63 to 0.83 across classes. For example, the "angry" class achieves a precision of 0.83, indicating that 83% of instances classified as "angry" were accurately identified.

Recall, measuring the ratio of correctly predicted instances to all actual instances of a class, varies from 0.55 to 0.87. For instance, the "neutral" class has a recall of 0.87, indicating that 87% of actual "neutral" instances were correctly classified by the model.

The F1-score, a harmonic mean of precision and recall, ranges from 0.59 to 0.83 across classes. For instance, the "happy" class has an F1-score of 0.59, reflecting a balance between precision and recall for predicting instances of "happy."

Support indicates the number of actual occurrences of each class in the dataset, ranging from 177 to 287 instances per class.

The overall accuracy of the model on the entire dataset is 0.73 (73%).

The macro average precision, recall, and F1-score are 0.72, 0.71, and 0.71, respectively, providing an average assessment across all classes, considering each class equally.

The weighted average precision, recall, and F1-score are 0.73, 0.72, and 0.72, respectively, accounting for class distribution by giving more weight to classes with a larger number of instances.

(fig 9) visual representation of the confusion matrix, where each cell's color intensity corresponds to the number of instances classified correctly or incorrectly by the model.

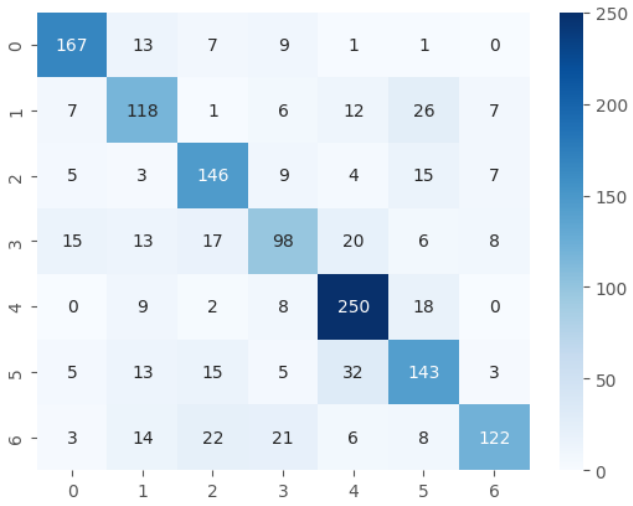

Fig 9: heatmap

6.References:

- *Head Fusion: Improving the Accuracy and Robustness of Speech Emotion Recognition on the IEMOCAP and RAVDESS Dataset*

Mingke Xu; Fan Zhang; Wei Zhang *HTTPS://IEEEXPLORE.IEEE.ORG/DOCUMENT/9381872*

- *SPEECH AND MUSIC CLASSIFICATION USING SPECTROGRAM BASED STATISTICAL DESCRIPTORS AND EXTREME LEARNING MACHINE PUBLISHED:*

28 NOVEMBER 2018 VOLUME 78, PAGES 15141–15168, (2019) *HTTPS://LINK.SPRINGER.COM/ARTICLE/10.1007/S11042-018-6899-Z*

- *Voice Recognition Algorithms using Mel Frequency Cepstral Coefficient (MFCC) and Dynamic Time Warping (DTW) Techniques Lindasalwa Muda, Mumtaj Begam and I. Elamvazuthi*

- *Decision tree-based acoustic models for speech recognition Masami Akamine & Jitendra Ajmera* *https://link.springer.com/article/10.1186/1687-4722-2012-10*
- *An analysis of convolutional neural networks for speech recognition Jui-Ting Huang; Jinyu Li; Yifan Gong* *https://ieeexplore.ieee.org/abstract/document/7178920*
- *Theoretical analysis of an alphabetic confusion matrix* ***https://link.springer.com/article/10.3758/BF03213026***