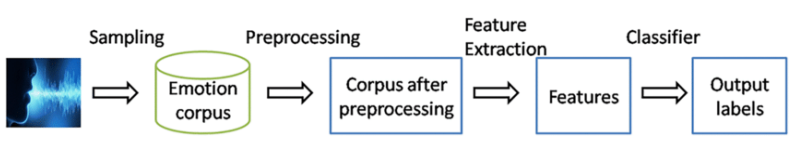
Speech Emotion Recognition (SER)

Introduction

Speech Emotion Recognition is a technology that analyzes speech to determine the emotions expressed by the speaker. It uses artificial intelligence and machine learning techniques to extract information related to emotions such as happiness, sadness, anger, fear, and surprise from various acoustic features such as pitch, duration, intensity, tone, timing, and prosody. Machine learning techniques are then applied to this information to determine the emotions associated with the speech.

Speech Emotion Recognition is a technology that uses machine learning techniques to analyze speech and extract information related to emotions. This technology has applications in various fields such as education, medicine, marketing, and entertainment, among others. It can improve the quality of communication, increase accuracy in speech recognition and voice typing, enhance interaction in electronic games and robots, detect individuals' mental states, and improve the quality of voice-related services such as phone calls and technical support.



Loading Dataset

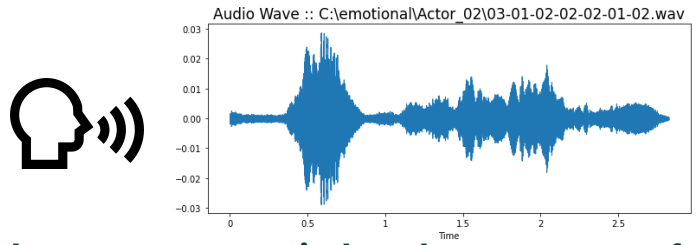
We introduced load_data function which takes a path as input and returns two lists: f_emotions and f_paths. The function loads the data from the specified path and extracts the emote and file path for each file in the directory. Then it returns emoticons and file paths as separate lists.

The get_emotion function takes a number as input and returns the corresponding emotion as a string. Uses a dictionary to map the entry number to a string representing the emotion.

Finally, the code calls the load_data function with the given path and maps the returned emoticons and file paths to the variable emoticons and paths, respectively.

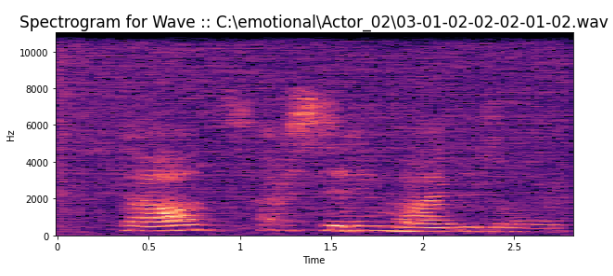
Read audio & Extracting features using the MFCC technique

read_audio: Loads an audio file from a given track and returns the audio data and sample rate.
draw_wave: Displays the audio wave of the selected audio file.



The voice wave of an audio file

drow_spectogram: Displays the spectrogram of the selected audio file.
add_noise: Adds random noise to the audio data.
Shift: Randomly shifts the audio data along the time axis.
Pitch: Changes the pitch of the audio data by a random factor.
strech: stretches the audio data by a factor.
feature_extraction: Extracts MFCC features from audio data using the librosa library.
Processing_audio: Applies the randomly selected audio processing function to the audio data.
get_features: Extract MFCC features from audio data after applying various processing functions.



Spectral diagram of the same audio file after adding the characteristic functions of the audio file

Dataset

Speech file (Audio_Speech_Actors_01-24.zip, 215 MB) contains 1440 files: 60 trials per actor x 24 actors = 1440.
The filename consists of a 7-part numerical identifier (e.g., 02-01-06-01-02-01-12.mp3 :

Filename identifiers
Modality (01 = full-AV, 02 = video-only, 03 = audio-only).
Vocal channel (01 = speech, 02 = song).
Emotion (01 = neutral, 02 = calm, 03 = happy, 04 = sad , 05 = angry, 06 = fearful, 07 = disgust, 08 = surprised).
Emotional intensity (01 = normal, 02 = strong).

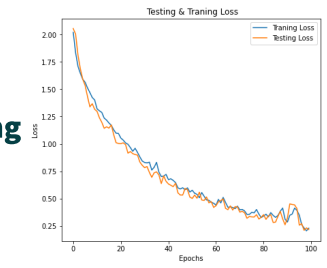
Training the model

A model was made to train the neural network using the audio data. The goal of this model is to analyze the feelings of the speakers in the audio data using artificial intelligence techniques.
Audio data is loaded from files in tracks and emotions. The program converts the audio into features using get_features, and adds the relevant features and emotions to the X and Y lists respectively.
OneHotEncoder is used to convert emoticons into a single hot encoder, which is stored in Y.
Then, the training and test data are split into x_train, x_test, y_train, and y_test using the train_test_split function in the sklearn library, with 20% of the data allocated to testing.
The training and test data are formatted so that depth is added to make it 3D using np.expand_dims, and the matrix axes are swapped using np.swapaxes to match the expected input shape of the model.
The model is built using sequential and multiple layers including TimeDistributed, LSTM and Dense are added to improve performance and emotion analysis. Adam is used as an optimizer to train the model and categorical_crossentropy is used as a loss function.
ReduceLROnPlateau and EarlyStopping are used as monitors to improve stopping training.
Finally, the model is trained using fit, and the training history is stored in the past for later reference.

Testing the model

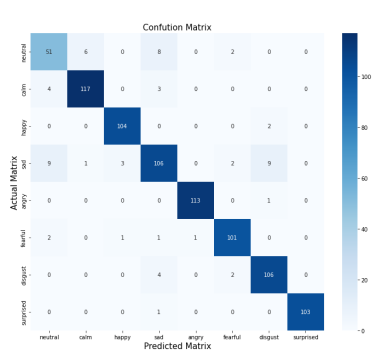
After training the model, it is tested using model.evaluate to calculate the accuracy of the model.
The training and testing loss and accuracy are stored in train_loss, test_loss, train_accuracy, and test_accuracy, respectively.
Two plots are created using matplotlib to show the training and testing loss and accuracy over time.
The final model is saved using model.save.
The saved model is loaded using load_model.
The model is tested using predict to make emotion predictions.
The predictions and actual emotions are stored in a pd.DataFrame and displayed using head().

Testing & Training Loss

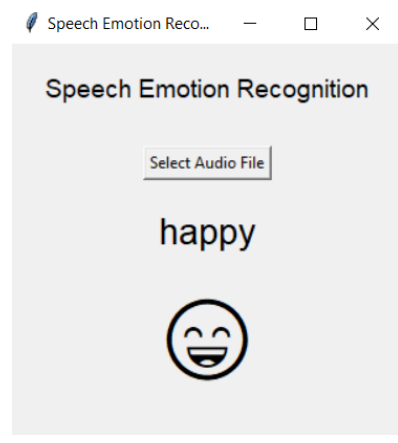


Testing & Training Accuracy

Confusion matrix was created using sklearn's confusion_matrix function to compare predicted feelings with actual feelings in the test data.
The emotions in the array are categorized using get_emotion to get the emotion rating corresponding to each category in encoder.categories_[0].
Then the confusion matrix is displayed using sns.heatmap.
The array was annotated with annotations=True, and the color scheme was set to "Blues" with cmap="Blues". The title, x-axis label, and y-axis label are also set using plt.title, plt.xlabel, and plt.ylabel, respectively. The figsize parameter of plt.figure is used to set the size of the plot.



Confusion matrix



The final output is a program that is able to analyze a person's feelings through their voice through a pre-recorded audio file.

Team members:

1- Ahmed Talat Abd El Mohsen Ali
221101084
2- Maged Mohamed Beltagy
221101048

Instructor: Prof. Ahmed Emam