**SPEECH EMOTION RECOGNITION**

**DATA SCIENCE PROJECT**

NAME: Alhan Charan Beshra

**PROJECT REPORT**

**Project Overview**

The Speech Emotion Recognition (SER) project aims to develop a system that can accurately identify and classify emotions from speech signals. This technology has applications in various fields, including customer service, mental health monitoring, and human-computer interaction. By leveraging deep learning techniques, we can analyze audio features to predict the speaker's emotional state.

**Dataset Information**

The dataset used is the Toronto Emotional Speech Set (TESS), which includes recordings of two actresses speaking 200 target words in seven different emotional states.

The dataset is organised such that each of the two female actor and their emotions are contain within its own folder. The format of the audio file is a WAV format.

Output Attributes:

* anger
* disgust
* fear
* happiness
* pleasant surprise
* sadness
* neutral

**Data Preprocessing**

**1. Loading the Data**

First, we load the dataset, which contains audio files categorized by different emotions.

**2. Normalization**

Normalization is applied to the audio data to ensure uniformity and to improve the performance and convergence of the neural network.

**3. Feature Extraction**

For audio data, extracting meaningful features is critical. We use Mel Frequency Cepstral Coefficients (MFCCs), which are widely used in speech and audio processing because they effectively represent the short-term power spectrum of sound.

**4. Label Encoding**

Since the labels (emotions) are categorical, they need to be converted into a numerical format. We use one-hot encoding for this purpose.

**5. Splitting the Data**

The data is split into training and validation sets to evaluate the model's performance.

These preprocessing steps ensure that the data is ready for model training, improving the model's performance and generalization capabilities.

**Model Building**

We use a Long Short-Term Memory (LSTM) network, which is a type of Recurrent Neural Network (RNN) well-suited for sequence data like audio signals.

The model architecture includes several layers:

* **LSTM Layer**: Captures the temporal dependencies in the audio data.
* **Dropout Layers**: Prevent overfitting by randomly setting a fraction of input units to 0 at each update during training.
* **Dense Layers**: Fully connected layers to learn the relationships between the extracted features.
* **Output Layer**: Uses a softmax activation function to classify the input into one of the seven emotion categories.

We then compile the model with the following configurations:

* **Loss Function**: categorical\_crossentropy
* **Optimizer**: adam
* **Metrics**: accuracy

**Training and Validation**

This involve training the model on a portion of the data and evaluating its performance on a separate validation set to ensure it generalizes well to unseen data.

**1. Data Splitting**

The first step is to split the data into training and validation sets. This ensures that we can evaluate the model's performance on unseen data and avoid overfitting.

* **X\_train**: Features for training the model.
* **X\_val**: Features for validating the model.
* **y\_train**: Labels for training the model.
* **y\_val**: Labels for validating the model.
* **test\_size**: Proportion of the data to be used for validation (20% in this case).
* **random\_state**: Ensures reproducibility by controlling the shuffling of the data.

**2. Model Training**

The model is trained using the training data. During training, the model learns to map the input features (MFCCs) to the corresponding emotion labels by adjusting the weights through backpropagation.

* **epochs**: Number of times the entire training dataset is passed through the model.
* **batch\_size**: Number of samples processed before the model is updated.
* **validation\_data**: Data used to evaluate the model's performance during training.

3. **Monitoring Training and Validation**

During training, it's important to monitor both training and validation metrics to understand how well the model is learning and to detect overfitting or underfitting.

* **Training Accuracy and Loss**: Indicate how well the model is learning on the training data.
* **Validation Accuracy and Loss**: Indicate how well the model is performing on unseen data.

**Evaluation**

This step involves generating predictions on the validation set, comparing them to the true labels, and using various metrics to assess the model’s performance.

**1. Generating Predictions**

First, we use the trained model to generate predictions on the validation set.

* **y\_pred**: Contains the predicted probabilities for each class for each sample in the validation set.

**2. Converting Predictions to Class Labels**

Since the model outputs probabilities, we convert these probabilities to class labels by selecting the class with the highest probability.

* **y\_pred\_classes**: Predicted class labels.
* **y\_val\_classes**: True class labels.

**Report**

The classification report and confusion matrix are displayed for evaluating the performance of a classification model. They provide insights into how well the model is performing for each class and help identify areas for improvement.

**1. Generating Classification Report**

The classification report includes metrics such as precision, recall, F1-score, and support for each class.

A screenshot of a graph

Description automatically generated

**2. Generating Confusion Matrix**

A confusion matrix provides a summary of prediction results on a classification problem. It helps visualize the performance of the model by showing how many instances of each class are correctly and incorrectly classified.

A screenshot of a computer screen

Description automatically generated

**Conclusion**

The Speech Emotion Recognition project successfully developed a model capable of classifying emotions from speech with a reasonable degree of accuracy. Further improvements can be made by exploring more advanced architectures, tuning hyperparameters, and increasing the dataset size.