Speech Emotion Recognition

Harshita Shukla (23n0257)

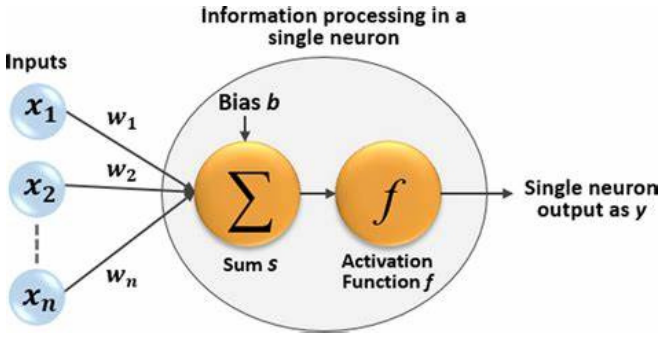
Concepts to Learn

- Neural Networks and Hyperparameter-Tuning
- Convolutional Neural Network
- Natural Language Processing
- Recurrent Neural Network
- LSTM and GRU

Neural Networks

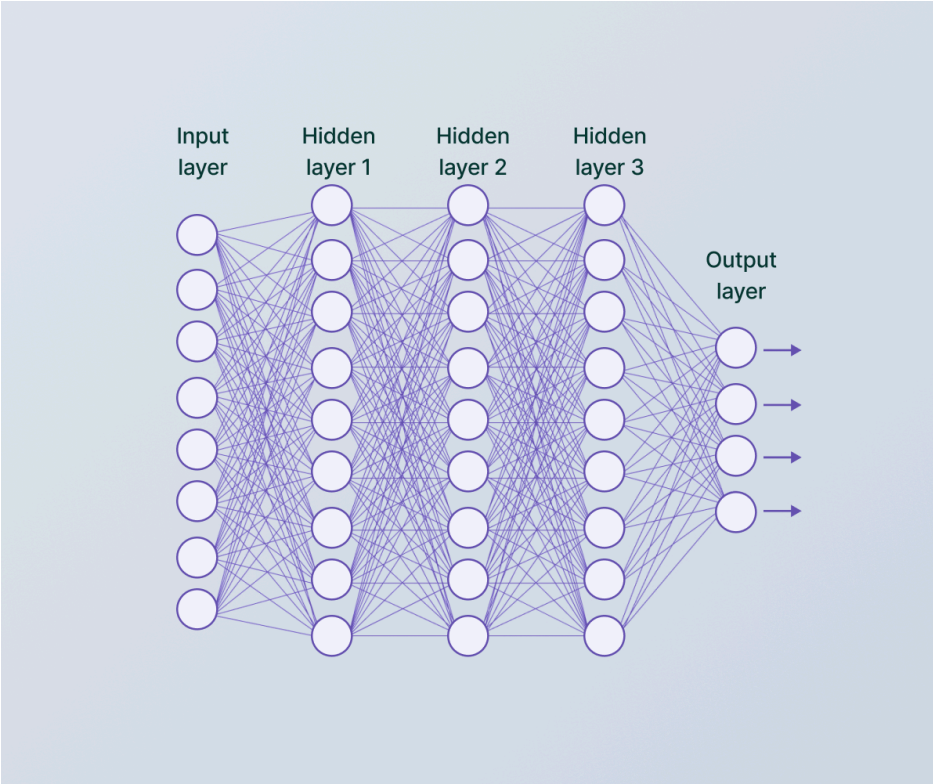
Neurons

Neuron is the smallest building block of Neural Networks. It receives inputs, each of which is multiplied by a corresponding weight, the weighted inputs are summed together, and the sum is then passed through an activation function (e.g., a step function, sigmoid, tanh or ReLU) to produce the output.



Neural Networks

Neural networks are computational models inspired by the human brain, composed of layers of interconnected neurons. The structure includes an input layer that receives data, one or more hidden layers that process inputs through weighted connections, and an output layer that produces the final result. Each connection between neurons has an associated weight that adjusts during training to minimize error.



The training of neural networks involves defining a loss function (MSE, Huber Loss, Binary Cross Entropy, Categorical Cross Entropy) to measure performance, computing gradients via backpropagation, and updating weights using optimization techniques. Optimization techniques are algorithms used to update the model's weights and biases based on the computed gradients. Common optimization techniques include:

- Gradient Descent (Mini-batch and stochastic)
- Momentum
- Adagrad
- NAG
- RMSprop

- Adam (most used)

Improving NN Performance

- To deal with **vanishing gradient problem** (while updating the weights, gradient becomes so small which prevents weights from changing), we use proper weight initialization techniques (**Xavier/Glorat** with tanh, sigmoid activation functions and **He** with ReLu).
- To deal with overfitting we use **Dropout layers**, **Regularization techniques** and **Early Stopping**.
- We normalize the input data and use **batch normalization** for normalizing activation vectors from hidden layers, this makes training faster and stable.
- **Learning rate scheduling**, hyperparameter tuning such as changing the number of hidden layers or no of neurons per layer, number of epochs and batch sizes also affect the performance.

Convolutional Neural Network

Convolutional Neural Networks (CNNs) are specialized artificial neural networks designed for processing structured grid data like images. They could be 1D, 2D and 3D. They are highly effective for tasks such as time series analysis, image and video recognition and classification, by leveraging the spatial structure of data to extract meaningful features and patterns.

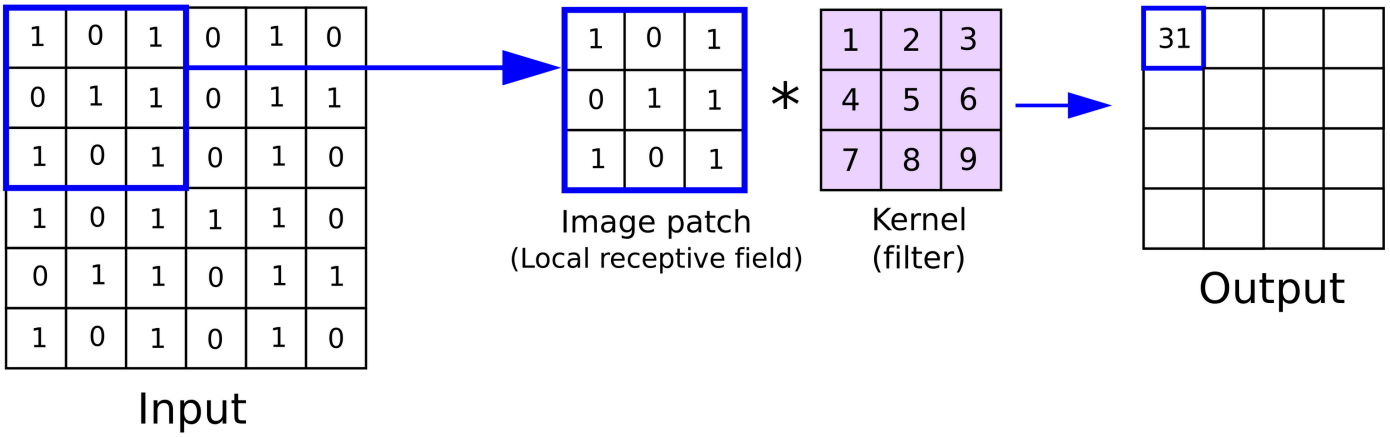
Structure

Input Layer:

- Receives raw pixel values of an image. For example, an image of size 256x256 with three color channels (RGB) has an input shape of (256, 256, 3).

Convolutional Layers:

- Apply convolutional filters to the input data, sliding over the image and performing a dot product between the filter and patches of the image. This captures local features such as edges, textures, and patterns. The output is called a feature map or activation map.

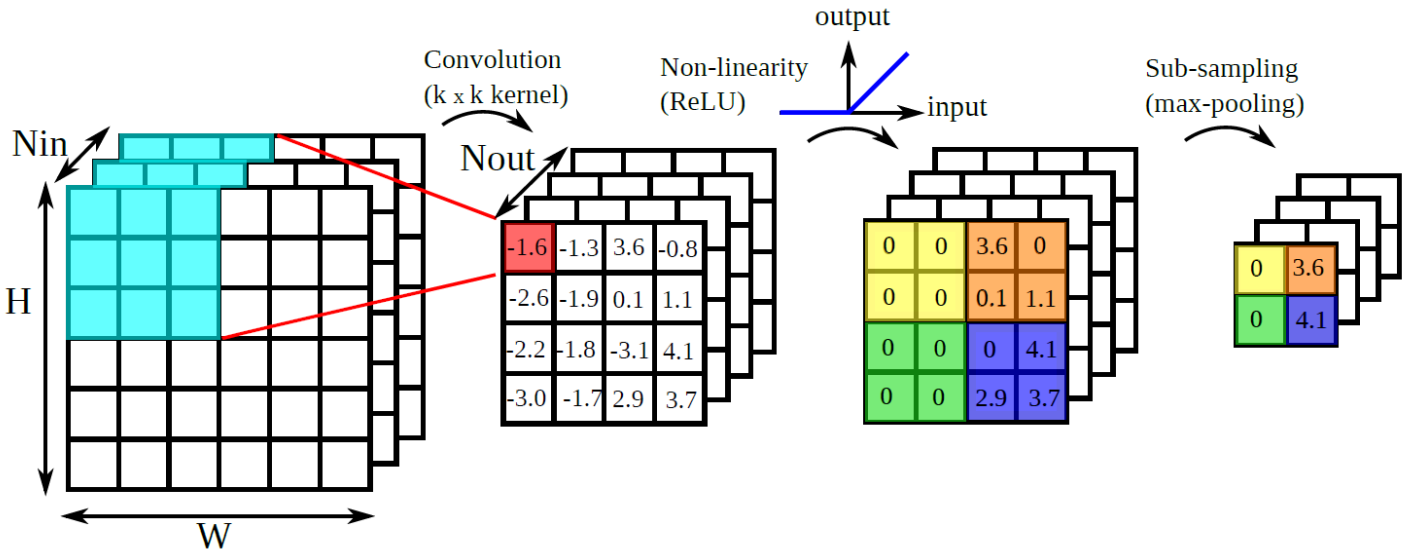


Activation Function:

- Applied after each convolution operation to introduce non-linearity. A common choice is ReLU (Rectified Linear Unit), which replaces negative pixel values with zero, maintaining positive values.

Pooling Layers:

- Reduce the spatial dimensions of the feature maps while retaining the most significant information. Max pooling, which takes the maximum value in each patch of the feature map, is the most common type. Pooling layers reduce computational load and the number of parameters and help make feature detection invariant to scale and orientation.

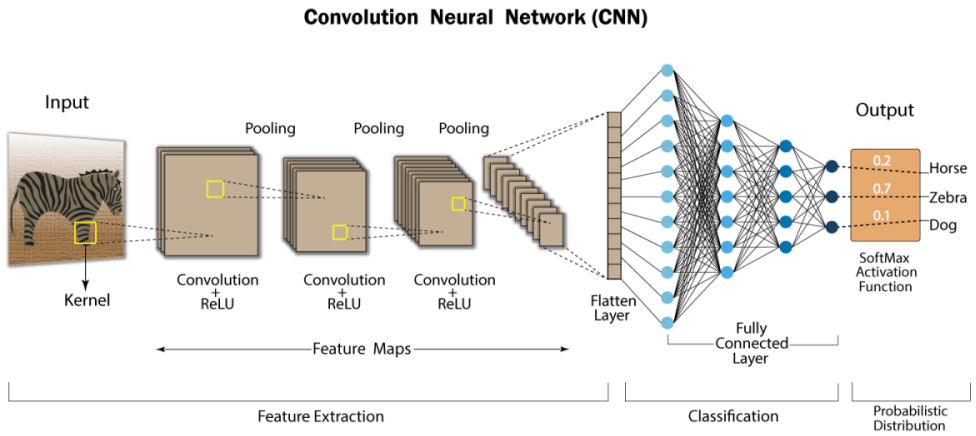


Fully Connected Layers:

- Perform high-level reasoning after several convolutional and pooling layers. They are similar to those in traditional neural networks and are used to output class scores or probabilities.

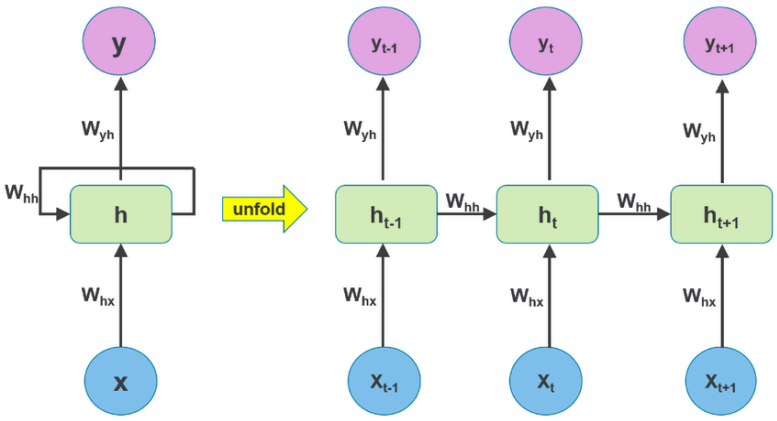
Output Layer:

- Provides the final predictions. Uses the sigmoid (binary classification) or SoftMax activation function (multiclass classification) to output a probability distribution over the classes.



Recurrent Neural Network

Recurrent Neural Networks (RNNs) are a class of neural networks designed to handle sequential data by maintaining a hidden state that captures information from previous time steps. This hidden state is updated iteratively as new data is processed, allowing the network to capture temporal dependencies and patterns in the data. Unlike traditional feedforward networks, RNNs have connections that loop back on themselves, enabling them to remember previous inputs. This makes RNNs particularly useful for tasks involving time-series data, natural language processing, and any domain where context or sequence is important.



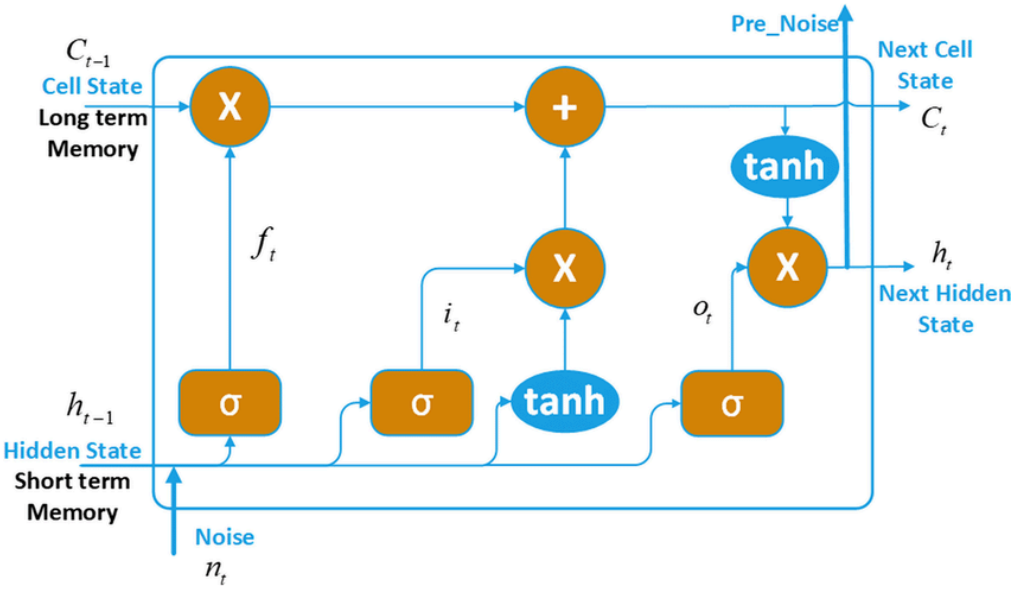
Types

- **Many to one**- sequential input with non-sequential output (time-series analysis)
- **One to many-** non-sequential input and sequential output (image-captioning, music generator)
- **Many to many-** input and output both sequential. Both sequences could be of same length (POS tagging, NER) or variable length (Machine-Translation)

Long Short-Term Memory (LSTM)

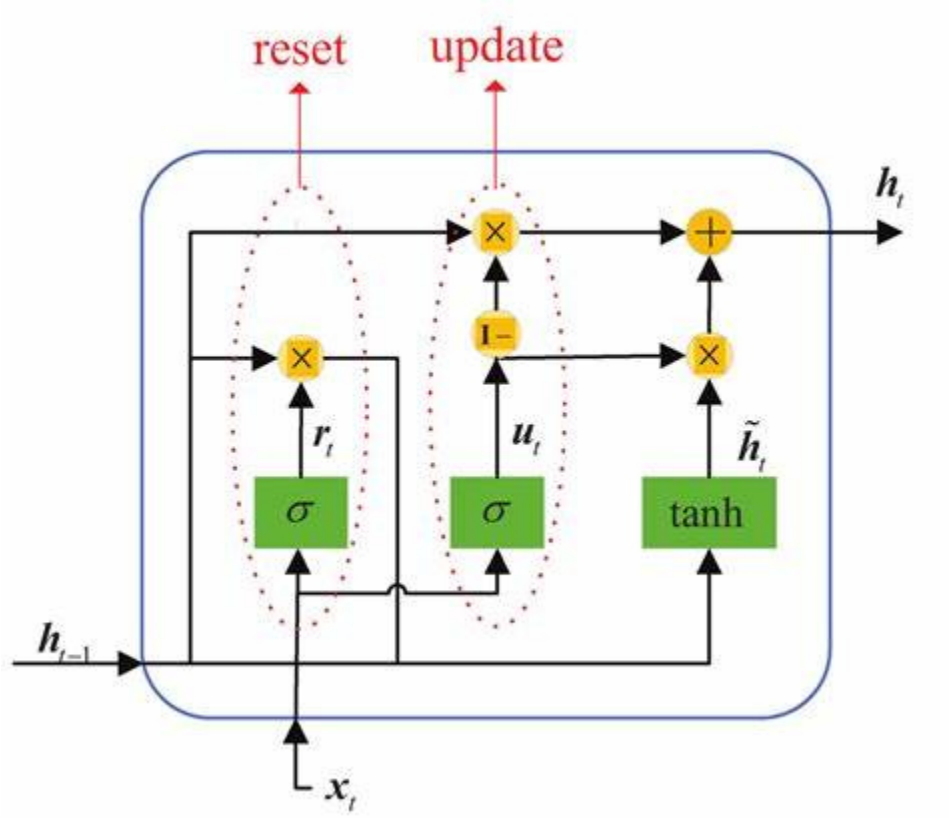
Long Short-Term Memory (LSTM) networks are a specialized type of Recurrent Neural Network (RNN) designed to handle long-range

dependencies in sequential data. Unlike traditional RNNs, LSTMs use a series of gates—input, forget, and output gates—to regulate the flow of information, allowing them to maintain and update a memory cell over long period. This gating mechanism helps prevent issues like vanishing gradients, making LSTMs effective for tasks involving long sequences. They are particularly useful in natural language processing, time-series prediction, and any application requiring the retention of contextual information over time.



Gated Recurrent Units (GRU)

Gated Recurrent Units (GRUs) are a type of Recurrent Neural Network (RNN) designed to capture dependencies in sequential data with fewer parameters than Long Short-Term Memory (LSTM) networks. GRUs use two gates—reset and update gates—to control the flow of information and maintain a hidden state, simplifying the architecture compared to LSTMs. The reset gate determines how much of the previous information to forget, while the update gate controls how much new information to incorporate. This design helps mitigate issues like vanishing gradients and makes GRUs efficient for tasks involving sequences. GRUs are particularly useful in applications like machine translation, speech recognition, and time-series analysis.



Natural Language Processing

NLP Definition: Natural Language Processing (NLP) is a field of artificial intelligence that focuses on the interaction between computers and human language, enabling machines to understand, interpret, and generate human language in a meaningful way.

Text Vectorization Techniques:

- **Bag of Words (BoW):** Represents text as a collection of word frequencies, ignoring grammar and word order.
- **Term Frequency-Inverse Document Frequency (TF-IDF):** Weighs word frequencies by their importance in the context of the entire corpus, reducing the impact of common words.
- **Word2Vec:** Converts words into dense vectors by capturing semantic relationships using neural networks.
- **GloVe (Global Vectors for Word Representation):** Generates word vectors by analyzing word co-occurrence statistics in a corpus.
- **FastText:** Extends Word2Vec by considering subword information, improving representation for rare words and morphological variants.

- **BERT (Bidirectional Encoder Representations from Transformers):** Provides contextual word embeddings by considering the context of words in both directions (left and right) in a sentence.

Properties of a Speech Dataset

Audio Files: The primary data format is usually WAV or MP3 files containing recorded speech.

Labels: Each audio file is associated with emotion labels, indicating the emotion expressed in the speech, such as happiness, sadness, anger, or neutral.

Features: Acoustic features derived from the audio signal, including various spectrograms and coefficients, that represent the speech characteristics needed for emotion recognition.

Features We Can Extract from a Speech Dataset

Mel-Frequency Cepstral Coefficients (MFCCs):

MFCCs capture the short-term power spectrum of a sound signal. They represent the timbral aspects of the audio and are critical for distinguishing phonetic content.

Mel Spectrogram:

A Mel spectrogram is a visual representation of the spectrum of frequencies in a sound signal over time, using Mel-scale frequencies. It provides a time-frequency analysis of the audio signal, highlighting how frequency components evolve over time. This feature helps in understanding the audio's temporal structure and tonal qualities.

Chroma Features:

Chroma features capture the energy of each pitch class in the audio signal. These features are derived from the pitch content and are useful for identifying harmonic content, which is beneficial in music and tonal speech analysis.

Spectral Contrast:

Spectral contrast measures the difference in amplitude between peaks and valleys in the sound spectrum. This feature helps in distinguishing between tonal and noisy sounds, providing additional context about the audio's texture and quality.

Model Building

Our model will analyze input speech data and give the emotion (among the 8 emotions) present in the speech audio file.

Data

We have used RAVDESS speech and song dataset and Tess dataset for building this model.

RAVDESS contains 1440 files: 60 trials per actor x 24 actors = 1440. The RAVDESS contains 24 professional actors (12 female, 12 male), vocalizing two lexically matched statements in a neutral North American accent. Eight emotions were selected for speech: neutral, calm, happy, sad, angry, fearful, surprise, and disgust

These stimuli were modeled on the Northwestern University Auditory Test №6 (NU-6; Tillman & Carhart, 1966). A set of 200 target words were spoken in the carrier phrase “Say the word ____’ by two actresses (aged 26 and 64 years)

In total we have 2800 files from TESS dataset, 1012 files from RAVDESS song and 1440 from RAVDESS speech dataset, summing up to 5252 files.

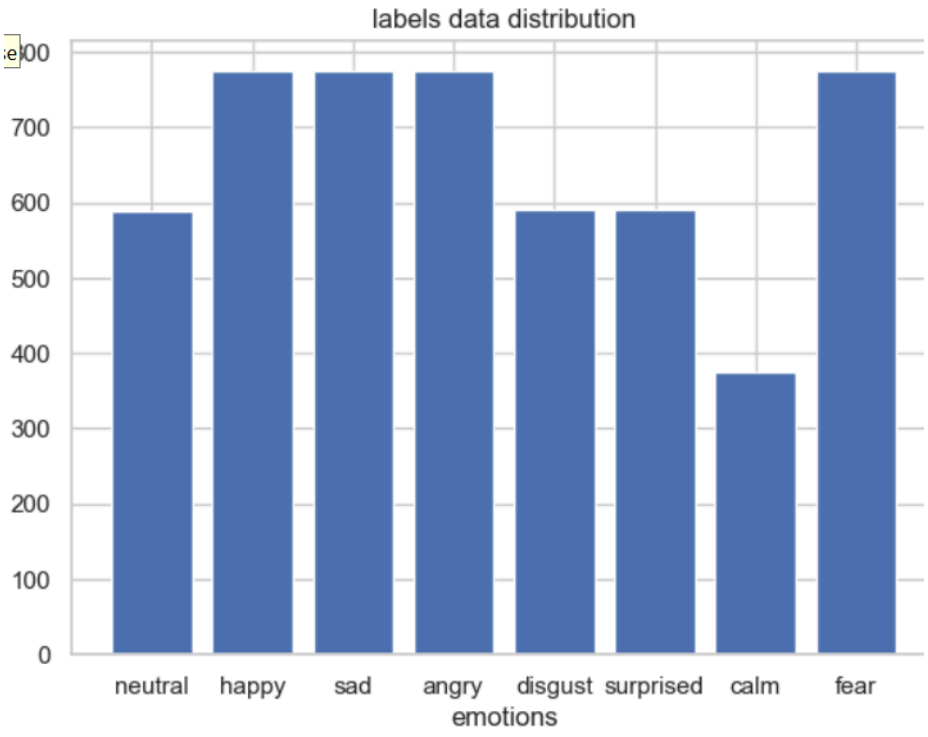
Data Preparation and Feature Extraction

Label Extraction and Mapping

- Extracted labels from the RAVDESS speech and song audio files.
- Mapped these labels to their respective emotions.
- Performed a similar extraction and mapping for the TESS dataset.

Sample Count Analysis

- Counted the number of samples for different emotions across datasets.
- Emotions identified: Neutral, Calm, Happy, Sad, Angry, Fear, Disgust, Surprised.
- After merging datasets, created a bar chart to visualize the distribution of samples for each emotion.
- The chart indicated that the imbalance among classes is minimal, allowing us to proceed with the dataset.



Audio Length Analysis

- Analyzed the length of audio files in each dataset.
- The lengths varied within a specific range.

Feature Extraction

- Extracted Mel and MFCC features from the audio files of each dataset.
- Obtained 13 Mel features and 128 MFCC features for each audio file.
- Constructed a DataFrame to concatenate the features from all datasets.

Data Preparation

- Split the dataset into training and testing sets.
- Encoded the output labels using label encoding.
- Normalized the features in the training and testing sets.

Model Development

Model Architecture

- Designed a CNN model with the following layers:
  - Conv1D layer and ReLu for two times
  - Max Pooling and A Dropout layer
  - Again Conv1D, ReLu , Max Pooling and Dropout layer
  - Batch Normalization, flattening, dense layer with dropout layer
  - SoftMax activation function in the output layer

Training Configuration

- Used the Adam optimizer with a learning rate of 0.001.

Performance Evaluation

- Achieved an accuracy of 76.02%.
- Obtained an F1 score of 76.97%.

Strategies to Improve Model Accuracy

To enhance the accuracy of our model, we can balance the class distribution for each emotion to prevent biases. Exploring different feature extraction techniques might improve the input data quality, and experimenting with various models could reveal more effective architectures. Also, hyperparameter tuning can optimize the model's performance, leading to better results.

Note

Not all concepts mentioned have been used in the final model, but they can be beneficial for building different models to perform similar tasks.

Resources

https://youtube.com/playlist?list=PL-wATfeyAMNqIee7cH3q1bh4QJFAaeNv0&si=0-en7dgNAVSFjEZ-

https://www.youtube.com/playlist?list=PLKnIA16_RmvYuZauWaPlRTC54KxSNLtNn

https://www.youtube.com/playlist?list=PLKnIA16_RmvZo7fp5kkIth6nRTeQQsjfX

https://www.coursera.org/learn/neural-networks-deep-learning?action=enroll

https://www.coursera.org/learn/deep-neural-network?specialization=deep-learning

https://www.coursera.org/learn/nlp-sequence-models?action=enroll

https://youtube.com/playlist?list=PLqnslRFeH2UrcDBWF5mfPGpqQDSta6VK4&si=hWXNK549af8p-vHr

https://tarun-lohex.medium.com/speech-emotion-recognition-258e88826a98