INDEX

0.INTRODUCTION

Welcome to the Emotion Detection System!

In today's fast-paced world, where digital transformation is shaping industries and personal experiences alike, understanding emotions, sentiments, and human behaviour through technology has never been more important. Whether it is using advanced machine learning algorithms to detect sentiment in audio, creating human-computer interaction systems based on gestures, or analysing vast amounts of data for sentiment analysis, technology is bridging the gap between machines and human emotions.

This document is designed to guide you through the development and understanding of **emotion-based systems**, from the early conceptualization of models to real-world applications. We will explore cutting-edge techniques, practical implementations, and the theoretical foundation behind these technologies.

Objectives:

- To introduce the role of emotions in technology
- To present various tools and frameworks used in emotion detection
- To showcase real-world applications of sentiment analysis and emotional recognition in diverse fields
- To demonstrate how human-computer interaction is evolving through the understanding of emotions

1. Overview of Emotion and Sentiment Analysis

➢ What are Emotions and Sentiments?

Emotions and sentiments are central to human experience. While emotions are short-term and spontaneous reactions to events or stimuli, sentiments are more enduring, reflecting a long-term emotional stance towards objects, people, or events.

- **Emotions**: Brief and intense psychological responses that result from specific events, such as happiness, anger, sadness, or surprise.
- **Sentiments**: More stable, long-term opinions or attitudes, such as a favourable or unfavourable stance towards something.

Both emotions and sentiments are essential for understanding human interactions, which is why they have become focal points in fields such as psychology, marketing, and human-computer interaction.

➢ Why is Emotion and Sentiment Analysis Important?

In today's data-driven world, the ability to understand and analyse human emotions and sentiments can significantly impact various domains, including:

- **Customer Experience**: Companies can measure customer satisfaction by analysing reviews, social media, or direct feedback.
- **Healthcare**: Emotion detection in voice or facial expressions can help diagnose mental health conditions or assess the well-being of patients.
- **Marketing**: Sentiment analysis helps brands understand public perception and adjust campaigns accordingly.
- **Human-Computer Interaction (HCI)**: Emotional intelligence in systems leads to more personalized and adaptive user experiences, making devices and software more intuitive.

➢ Types of Emotion and Sentiment Analysis

The analysis of emotions and sentiments can be broken down into different methodologies:

- **Text-based Sentiment Analysis**: Used to determine the emotional tone behind a body of text, often categorized as positive, negative, or neutral.
- **Audio-based Emotion Recognition**: Analyses vocal patterns and tone to infer emotions such as happiness, sadness, or anger.
- **Facial Expression Recognition**: Uses computer vision to detect emotions based on facial features.
- **Gesture Recognition**: Identifies emotions through physical gestures, such as hand movements or posture.

➢ Key Technologies and Tools

Emotion and sentiment analysis leverage a variety of technologies, including:

- **Natural Language Processing (NLP)**: For text-based sentiment analysis.
- **Machine Learning**: To train models that classify emotions or sentiments based on data.
- **Deep Learning**: Neural networks are used for complex emotion recognition tasks such as voice or facial emotion detection.
- **Libraries and Frameworks**:
    - **DeepFace**: For facial recognition and emotion detection.
    - **Librosa**: For audio feature extraction in emotion analysis.
    - **NLTK, TextBlob**: For text-based sentiment analysis.

2. Technological Approaches to Emotion and Sentiment Analysis

➢ Overview of Emotion Recognition Techniques

Emotion and sentiment analysis rely on advanced algorithms and machine learning models to interpret human expressions, tone, and text. These techniques differ based on the medium being analysed — text, audio, or visual data. In this section, we will cover key technological approaches used in each area.

➢ Text-based Sentiment Analysis

Text-based sentiment analysis extracts subjective information from text, identifying whether the sentiment is positive, negative, or neutral. Natural Language Processing (NLP) and machine learning are the core technologies behind this approach.

Key Techniques:

- Lexicon-based Analysis: Uses predefined lists of words with associated sentiment scores to determine the sentiment of text.
- Machine Learning Models: Supervised learning models (e.g., Support Vector Machines, Naive Bayes) trained on labeled datasets to classify sentiment.
- Deep Learning: Models like Recurrent Neural Networks (RNNs) and Transformer-based models (e.g., BERT, GPT) allow for context-aware sentiment analysis, improving accuracy over traditional methods.

Libraries and Tools:

- NLTK: A powerful library for working with human language data.
- TextBlob: A simple tool for processing textual data and performing basic NLP tasks.

- VADER: A lexicon and rule-based sentiment analysis tool specifically attuned to sentiments expressed in social media.

➢ Audio-based Emotion Recognition

Human speech carries emotional information through tone, pitch, and pace, making audio a rich medium for emotion detection.

Key Techniques:

- Feature Extraction: Mel-frequency cepstral coefficients (MFCC), chroma features, and spectral contrast are extracted from audio to represent the signal in a way that a machine learning model can process.
- Machine Learning Models: After feature extraction, models like Support Vector Machines (SVM), Random Forests, or Neural Networks classify the emotion based on patterns in the audio data.
- Deep Learning for Audio: Convolutional Neural Networks (CNNs) and Long Short-Term Memory Networks (LSTMs) are widely used for complex audio emotion detection, especially when dealing with large datasets.

Libraries and Tools:

- Librosa: A Python library for music and audio analysis, widely used for feature extraction from audio data.
- OpenSMILE: A toolkit for audio analysis and feature extraction, widely used in emotion detection tasks.
- Praat: A tool for analyzing, synthesizing, and manipulating speech.

- **Visual-based Emotion Recognition**

Facial expressions are one of the most direct ways to detect human emotions. Modern systems rely on computer vision and machine learning to analyse faces and infer emotions.

Key Techniques:

- Facial Landmarks Detection: Detects key points on the face such as eyes, nose, and mouth, which are used to track facial movements.
- Convolutional Neural Networks (CNNs): CNNs are typically used for classifying emotions by processing pixel data from images or video frames.
- Facial Action Coding System (FACS): A system that categorizes facial expressions based on muscle movements and is often used to label training data for emotion recognition models.

Libraries and Tools:

- OpenCV: A widely-used library for computer vision tasks, including face detection and facial landmarks identification.
- DeepFace: A deep learning-based library that provides face recognition, emotion analysis, and demographic insights.
- Dlib: A toolkit for detecting facial landmarks and for computer vision applications.

- **Multimodal Emotion Recognition**

To achieve a more holistic understanding of human emotions, multimodal emotion recognition combines different types of data, such as text, audio, and visual cues. This approach leverages the complementary nature of different data modalities to improve accuracy.

Key Techniques:

- Feature Fusion: Combines features from different data sources (text, audio, video) before feeding them into a unified machine learning or deep learning model.

- Ensemble Models: Uses multiple models trained on different data modalities and combines their predictions to improve the overall result.

➢ Emotion Recognition Frameworks

Several high-level frameworks and APIs are available that integrate these various techniques into easy-to-use systems for emotion detection and sentiment analysis.

- IBM Watson: A comprehensive set of AI services that include emotion detection and sentiment analysis from both text and audio.
- Google Cloud Natural Language API: Provides tools for sentiment analysis from text with powerful machine learning models.
- Microsoft Azure Cognitive Services: Offers APIs for text analytics and emotion detection through face recognition.

3. Deep Learning Models for Emotion Recognition

Deep learning has revolutionized the field of emotion recognition by enabling systems to learn complex patterns in large datasets. This section explores the key deep learning models and architectures that are most effective in analyzing emotions from text, audio, and visual data.

- **Overview of Deep Learning in Emotion Recognition**

Deep learning models are particularly suited for emotion recognition due to their ability to handle large amounts of data and automatically learn features from raw inputs like images, audio signals, and text. Unlike traditional machine learning models, deep learning networks do not require manual feature extraction, making them more efficient and scalable for real-world applications.

- **Convolutional Neural Networks (CNNs)**
  - **How CNNs Work**

Convolutional Neural Networks (CNNs) are primarily used for tasks involving images and visual data. CNNs consist of convolutional layers that automatically extract features like edges, shapes, and textures, which are crucial for recognizing facial expressions or identifying emotional cues in videos.

CNNs are widely applied in:

- **Facial Emotion Recognition**: CNNs can classify emotions by processing facial images frame by frame. Pre-trained models such as VGGFace or ResNet are often used for emotion recognition tasks.
- **Audio-based Emotion Recognition**: CNNs can also be applied to spectrograms of audio files, treating them as images. This approach has shown success in identifying emotions from voice patterns.

Popular CNN Architectures

- **VGGNet**: Known for its simplicity and effectiveness, VGGNet is commonly used for image classification and can be applied to facial emotion recognition tasks.
- **ResNet**: ResNet's residual connections help it tackle deeper networks, making it highly efficient in learning complex patterns in facial data.
- **EfficientNet**: Balancing performance and computational cost, EfficientNet can be applied to large-scale emotion detection tasks that require both speed and accuracy.

- **Recurrent Neural Networks (RNNs) and LSTMs**

How RNNs and LSTMs Work

Recurrent Neural Networks (RNNs) are designed to process sequential data, making them ideal for tasks involving time-series information, such as analyzing speech or videos. However, traditional RNNs struggle with long-term dependencies. Long Short-Term Memory (LSTM) networks, a variant of RNNs, address this issue by retaining important information over time.

LSTMs are widely used in:

- **Speech Emotion Recognition**: LSTMs can capture emotional cues from sequences of audio data by processing MFCC or other audio features over time.
- **Text-based Sentiment Analysis**: LSTMs are commonly applied in NLP to identify emotions and sentiments in text by preserving context over multiple words or sentences.

Bi-directional LSTMs

A Bi-directional LSTM processes input sequences in both forward and backward directions, improving accuracy in emotion detection by analyzing context from both past and future data points. This model is particularly effective in tasks like:

- **Audio-based Emotion Recognition**: Captures tonal and emotional changes across a conversation or speech.
- **Text Sentiment Analysis**: Identifies the emotional tone of a text by considering context from both the beginning and the end of sentences.

4. Emotion Recognition Model Overview

This model is designed to classify emotions based on audio inputs, utilizing the **RAVDESS dataset** for training and testing. The key steps involved in this process are:

1. **Librosa**: Librosa is a powerful Python library designed for music and audio analysis. It provides easy-to-use functions for loading audio files
2. **Data Preprocessing**: Audio files are loaded and processed to extract relevant features using **MFCC** and **STFT**.
3. **Feature Extraction**: The combination of MFCC and STFT features forms the input for training the model.
4. **Model Architecture**: A **Convolutional Neural Network (CNN)** with two Conv1D layers, followed by MaxPooling and Dropout layers, is used to extract temporal features from the audio data.
5. **Training and Evaluation**: The model is trained using categorical cross-entropy loss and evaluated based on accuracy, precision, recall, and F1 score.
6. **Emotion Prediction**: After training, the model can predict one of eight emotions: **neutral, calm, happy, sad, angry, fearful, disgust, and surprised**.

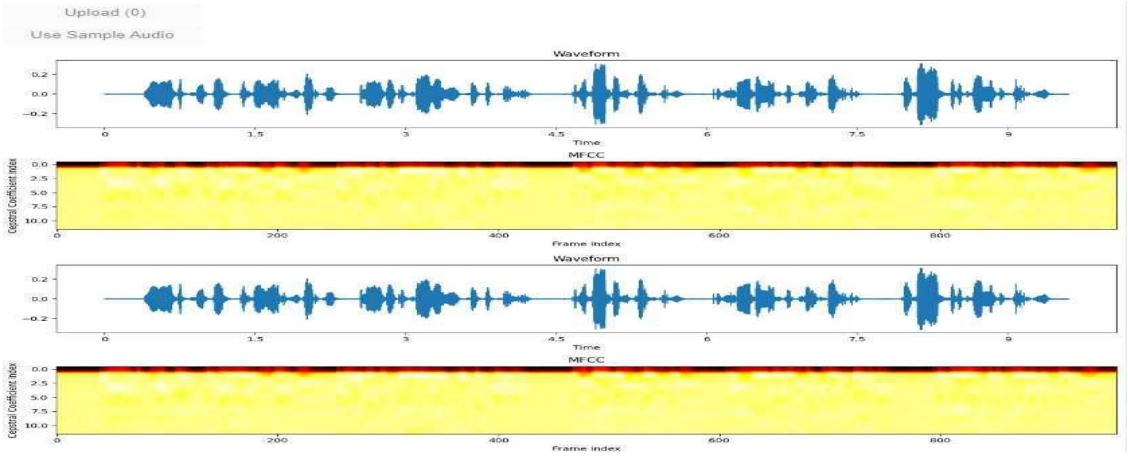
5. Feature Extraction: MFCC and STFT

Mel-Frequency Cepstral Coefficients (MFCC)

MFCC is one of the most widely used features in speech recognition and analysis because it captures the frequency characteristics of an audio signal in a way that aligns with human auditory perception. The MFCC process includes several steps, such as converting the audio signal into the frequency domain and applying filters on the mel scale, which simulates the human ear's sensitivity to different frequencies.

- **n_mfcc parameter**: In our model, n_mfcc=45, which defines the number of MFCCs to extract. This value represents the number of coefficients used to represent the spectrum. Typically, the default is 13, but we've increased it to 45 to capture more detailed information about the emotions conveyed in the speech. This higher value allows us to retain more granular data, improving the model's learning ability.
- **Shape and Size of MFCC**: After extracting the MFCCs from an audio file, the output is averaged along the time axis, creating a 1D array of size 45 (since we specified n_mfcc=45). This is crucial for reducing the computational load while preserving the key frequency components.

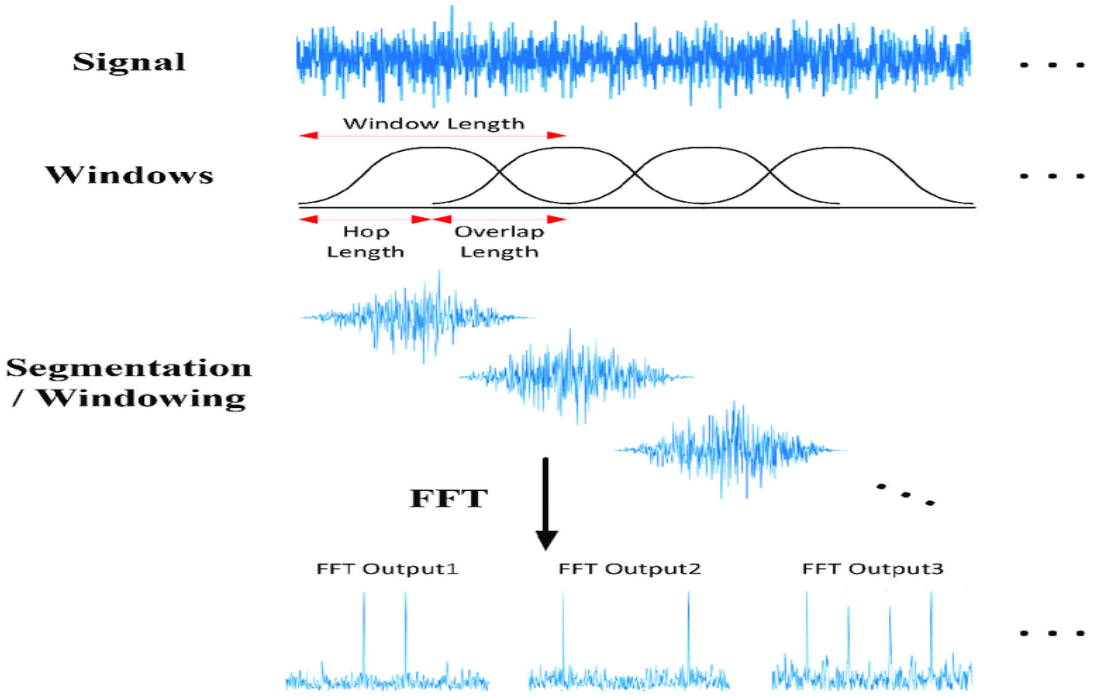
The MFCC output has a shape of (n_mfcc,), where n_mfcc is 45, giving us a fixed-length feature vector for each audio file.



Short-Time Fourier Transform (STFT)

STFT is another critical feature used in speech analysis. Unlike MFCC, which focuses on human-perceived frequencies, STFT provides a detailed time-frequency representation of the audio. It breaks down the audio into small time frames and applies the Fourier Transform to each, generating a matrix that represents how the frequency content of the signal evolves over time.

- **Shape and Size of STFT**: After applying the STFT, we obtain a 2D matrix where each row represents a frequency bin, and each column represents a time slice. We simplify this by averaging the frequency data over time to get a 1D array of size equal to the number of frequency bins.
- In our case, this reduces the complexity, creating a feature vector representing the overall energy distribution across frequency bins for the entire audio file.



Feature Combination

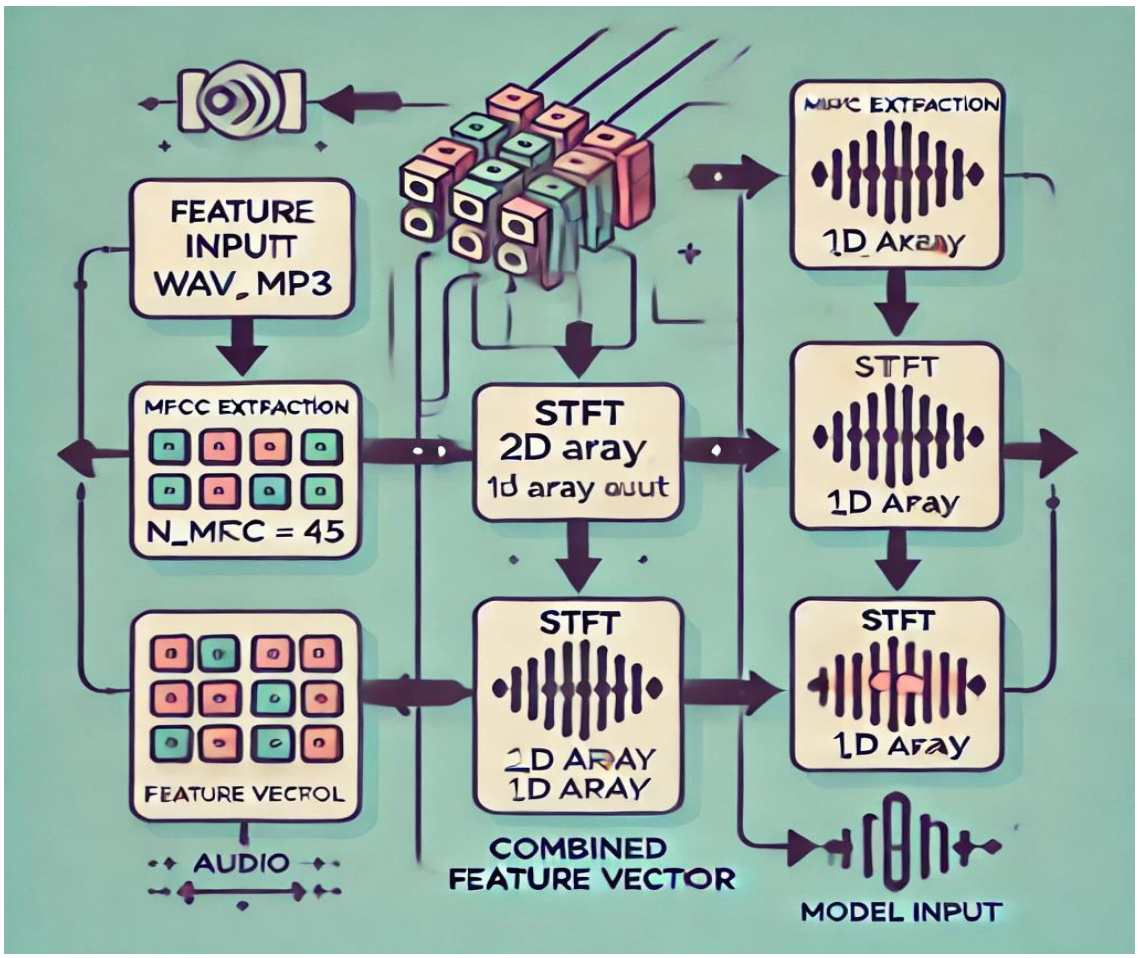
To enhance our model's performance, we combine MFCC and STFT features. MFCC captures perceptually meaningful information, while STFT focuses on the signal's detailed structure. By concatenating these two feature sets, we create a comprehensive representation of the audio file.

The final feature vector combines the 45-dimensional MFCC vector and the STFT vector, resulting in a longer, richer feature set that feeds into our model.

Data Shapes

For each audio file, the features are combined into a 1D array. This array is then reshaped for the model input, which requires 3D arrays where each data point's shape is (number of features, 1). Specifically:

- **Input Shape**: For the model, each input will have the shape (number_of_features, 1), where number_of_features = 45 (MFCC) + stft_vector_length.



6. Convolutional Architecture

Purpose of Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) are widely used in various tasks, especially in image and audio processing, due to their ability to automatically learn spatial hierarchies of features from input data. In the context of sentiment analysis from audio signals, CNNs help to:

- **Capture Local Patterns**: CNNs excel at recognizing local patterns in data, such as specific frequency components in audio signals, which are crucial for emotion recognition.
- **Parameter Sharing**: By using convolutional filters that are shared across the input space, CNNs reduce the number of parameters compared to fully connected networks, making them more efficient.
- **Translation Invariance**: The convolutional layers ensure that the model can recognize features regardless of their position in the input, which is particularly useful in audio where certain sounds can occur at different times.

Key Components and Parameters

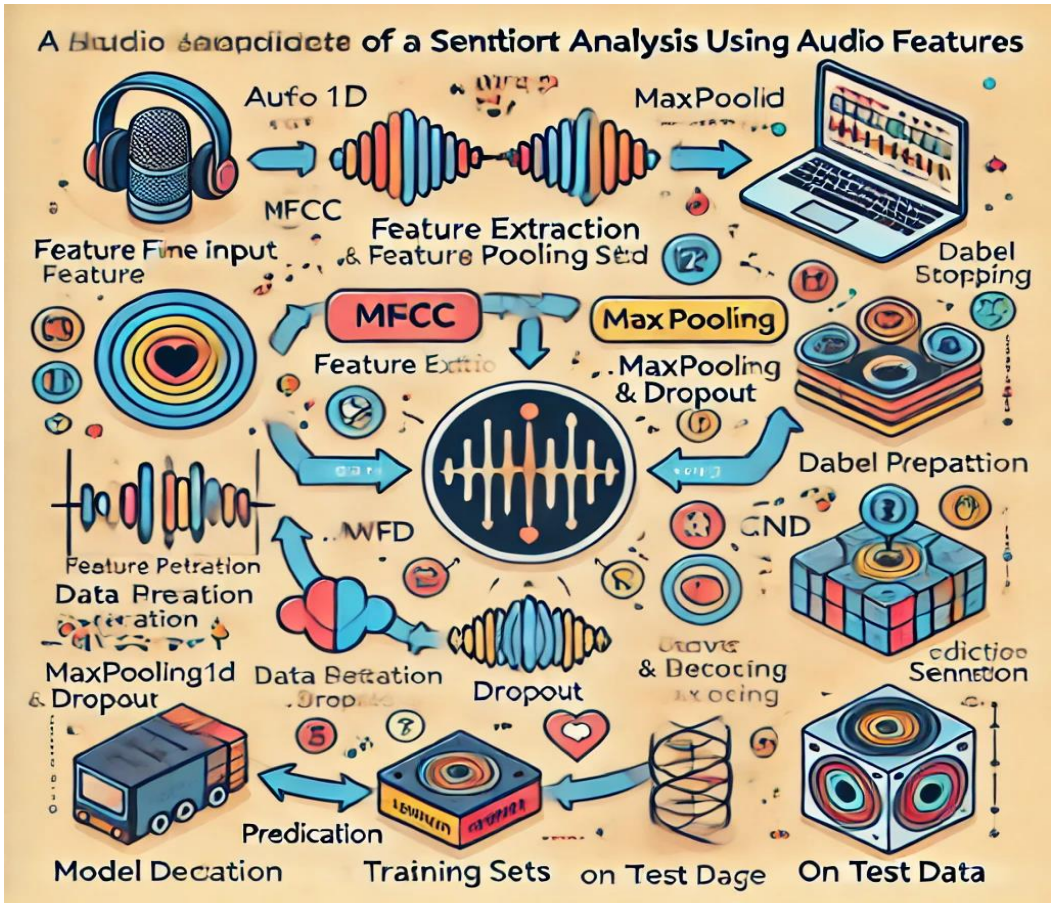
Here's a breakdown of the architecture defined in my model:

1. **Input Shape**
    - **Explanation**: The input shape corresponds to the dimensions of the extracted features, where ravdess_speech_data_array.shape[1] is the number of features (MFCCs and STFTs combined) for each audio sample, and 1 indicates a single channel.

2. **Convolutional Layer**
    - **Parameters**:
        - **64**: The number of filters (kernels) used in this layer, determining how many feature maps will be created.
        - **kernel_size=3**: The size of each filter, which defines how many consecutive features will be combined during convolution. A size of 3 means the filter will look at 3 time steps at a time.
        - **activation='relu'**: The Rectified Linear Unit (ReLU) activation function is used to introduce non-linearity in the model, allowing it to learn complex patterns.
3. **Max Pooling Layer**
    - **Explanation**: This layer reduces the dimensionality of the feature maps by taking the maximum value over a specified window (here, 2 time steps). This helps in reducing computation and capturing the most important features.
4. **Dropout Layer**
    - **Parameters**:
        - **0.2**: The dropout rate indicates that 20% of the neurons will be randomly turned off during training. This regularization technique helps prevent overfitting by ensuring that the model does not become too reliant on any single feature.
5. **Flatten Layer**
    - **Explanation**: This layer converts the 2D feature maps into a 1D feature vector, preparing the data for the fully connected layers that follow.
6. **Dense Layers**
    - **First Dense Layer**:
        - **64**: The number of neurons in this layer, allowing the model to learn complex combinations of features.

- **activation='relu'**: Again, using ReLU to maintain non-linearity.

- **Output Dense Layer**:
  - **8**: The number of output neurons corresponds to the number of emotion classes being predicted (neutral, calm, happy, sad, angry, fearful, disgust, surprised).
  - **activation='softmax'**: This activation function is used in multi-class classification problems to output probabilities for each class, ensuring that the sum of the output probabilities equals 1.



7. The Model

Dataset :

Link : https://www.kaggle.com/datasets/uwrfkaggler/ravdess-emotional-speech-audio

Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS)

Speech audio-only files (16bit, 48kHz .wav) from the RAVDESS. Full dataset of speech and song, audio and video (24.8 GB) available from Zenodo. Construction and perceptual validation of the RAVDESS is described in our Open Access paper in PLoS ONE.

Check out our Kaggle Song emotion dataset.

Files

This portion of the 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. Speech emotions includes calm, happy, sad, angry, fearful, surprise, and disgust expressions. Each expression is produced at two levels of emotional intensity (normal, strong), with an additional neutral expression.

File naming convention

Each of the 1440 files has a unique filename. The filename consists of a 7-part numerical identifier (e.g., 03-01-06-01-02-01-12.wav). These identifiers define the stimulus characteristics:

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). NOTE: There is no strong intensity for the 'neutral' emotion.

- Statement (01 = "Kids are talking by the door", 02 = "Dogs are sitting by the door").
- Repetition (01 = 1st repetition, 02 = 2nd repetition).
- Actor (01 to 24. Odd numbered actors are male, even numbered actors are female).

Filename example: 03-01-06-01-02-01-12.wav

1. Audio-only (03)
2. Speech (01)
3. Fearful (06)
4. Normal intensity (01)
5. Statement "dogs" (02)
6. 1st Repetition (01)
7. 12th Actor (12)
   Female, as the actor ID number is even.

Code :

```
import os
import librosa
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from tensorflow.keras import layers, models, callbacks
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.layers import Dropout
```

```
def extract_mfcc(wav_file_name):
    y, sr = librosa.load(wav_file_name)
    mfcc = np.mean(librosa.feature.mfcc(y=y, sr=sr, n_mfcc=45).T, axis=0)
    return mfcc

def extract_stft(wav_file_name):
    y, sr = librosa.load(wav_file_name)
    stft = np.abs(librosa.stft(y))
    stft_mean = np.mean(stft, axis=1)  # Averaging along the time axis
    return stft_mean

def extract_features(wav_file_name):
    mfcc = extract_mfcc(wav_file_name)
    stft = extract_stft(wav_file_name)
    return np.concatenate((mfcc, stft))

ravdess_speech_labels = []
ravdess_speech_data = []

for dirname, _, filenames in os.walk(r'C:\Users\Aprajit Sharma\Desktop\Sentiment
Analysis\ravdess'):
    for filename in filenames:
        if filename.endswith(".wav"):
            ravdess_speech_labels.append(int(filename[7:8])-1)
            wav_file_name = os.path.join(dirname, filename)
            ravdess_speech_data.append(extract_features(wav_file_name))

ravdess_speech_data_array = np.asarray(ravdess_speech_data)
ravdess_speech_label_array = np.array(ravdess_speech_labels)

print("Feature data shape:", ravdess_speech_data_array.shape)
```

```
print("Labels shape:", ravdess_speech_label_array.shape)

# Prepare the data for training
labels_categorical = to_categorical(ravdess_speech_label_array, num_classes=8)

x_train, x_test, y_train, y_test = train_test_split(
    np.expand_dims(ravdess_speech_data_array, axis=-1),
    labels_categorical,
    test_size=0.30,
    random_state=9
)

print("x_train shape:", x_train.shape)
print("y_train shape:", y_train.shape)
print("x_test shape:", x_test.shape)
print("y_test shape:", y_test.shape)

# Define and compile the model
early_stopping = callbacks.EarlyStopping(
    monitor='loss',
    patience=15,
    restore_best_weights=True
)

def model_cnn():
    input_shape = (ravdess_speech_data_array.shape[1], 1)
    model = models.Sequential([
        layers.Conv1D(64, kernel_size=3, activation='relu', input_shape=input_shape),
        layers.MaxPooling1D(pool_size=2),
        layers.Dropout(0.2),
        layers.Conv1D(32, kernel_size=3, activation='relu'),
```

```
        layers.MaxPooling1D(pool_size=2),
        layers.Dropout(0.5),
        layers.Flatten(),
        layers.Dense(64, activation='relu'),
        layers.Dense(8, activation='softmax')
    ])

    model.compile(optimizer='nadam',
                  loss='categorical_crossentropy',
                  metrics=['accuracy'])

    return model

model_A = model_cnn()

# Train the model
history = model_A.fit(x_train, y_train,
                      validation_split=0.1,
                      epochs=100,
                      shuffle=True,
                      callbacks=[early_stopping])

# Emotion labels
emotions = {
    0: 'neutral',
    1: 'calm',
    2: 'happy',
    3: 'sad',
    4: 'angry',
    5: 'fearful',
    6: 'disgust',
```

```
    7: 'surprised'
}

# Prediction function
def predict(wav_filepath):
    test_point = extract_features(wav_filepath)
    test_point = np.reshape(test_point, newshape=(1, test_point.shape[0], 1))
    predictions = model_A.predict(test_point)
    print(emotions[np.argmax(predictions[0])])

# # Test the predict function
# predict('C:/Users/Aprajit Sharma/Desktop/Sentiment Analysis/ravdess/Actor_01/03-01-01-01-01-01-01.wav')

# Evaluate the model
from sklearn.metrics import precision_score, recall_score, f1_score, accuracy_score

predictions = model_A.predict(x_test)
y_pred = np.argmax(predictions, axis=1)
y_true = np.argmax(y_test, axis=1)

precision = precision_score(y_true, y_pred, average='weighted')
recall = recall_score(y_true, y_pred, average='weighted')
f1 = f1_score(y_true, y_pred, average='weighted')
accuracy = accuracy_score(y_true, y_pred)

print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
model_A.save('audio_sentiment_model.h5')
```

Note : This model will be saved in the audio_sentiment_model.h5 file and can be used anywhere.

Output :

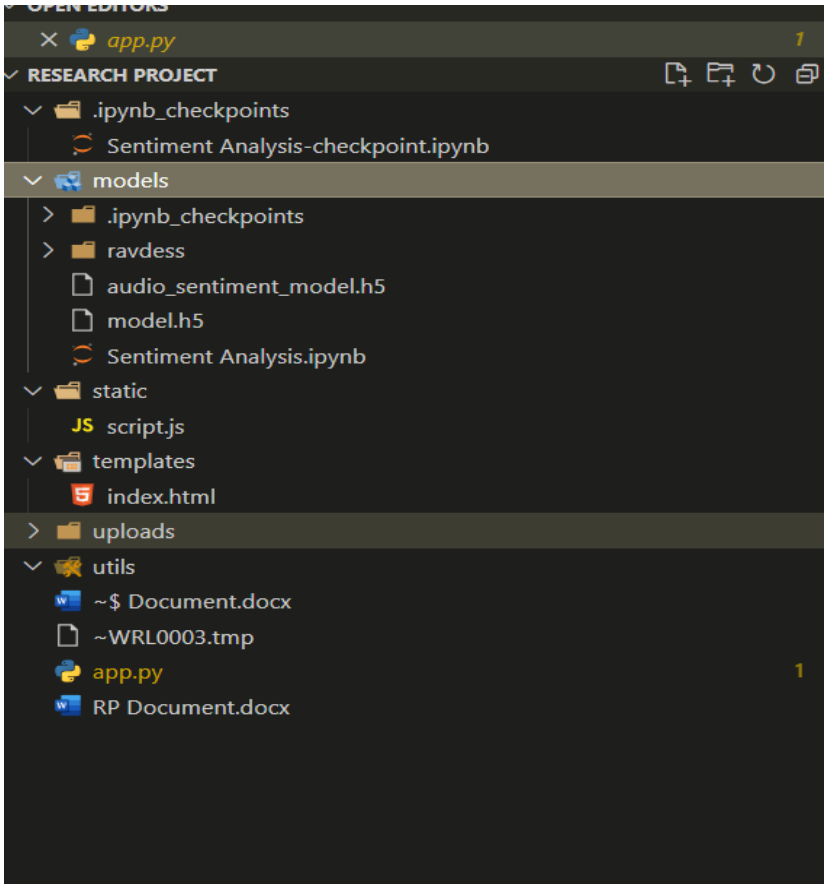
```
Feature data shape: (2880, 1070)
Labels shape: (2880,)
x_train shape: (2016, 1070, 1)
y_train shape: (2016, 8)
x_test shape: (864, 1070, 1)
y_test shape: (864, 8)
```

```
57/57 ━━━━━━━━━━━━━━━━━━━━ 2s 35ms/step - accuracy: 0.9573 - loss: 0.1387 - val_accuracy: 0.8366 - val_loss: 0.6418
Epoch 100/100
57/57 ━━━━━━━━━━━━━━━━━━━━ 2s 33ms/step - accuracy: 0.9418 - loss: 0.1871 - val_accuracy: 0.8713 - val_loss: 0.4552

27/27 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step
Accuracy: 0.8553240740740741
Precision: 0.8628777802259316
Recall: 0.8553240740740741
F1 Score: 0.8569297666750535
```

```
In [2]: predict('C:/Users/Aprajit Sharma/Desktop/Sentiment Analysis/ravdess/Actor_01/03-01-01-01-01-01-01.wav')

1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 125ms/step
neutral

In [3]: model_A.save('audio_sentiment_model.h5')

WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_model.keras')` or `keras.saving.save_model(model, 'my_model.keras')`.
```

Connecting the frontend:

File Structure ➔



Note: I have used Flask for serving the web page.

App.py ➔

import os

from flask import Flask, request, jsonify, render_template

from werkzeug.utils import secure_filename

import numpy as np

import librosa

from tensorflow.keras.models import load_model

```
# Load the trained model
model = load_model('models/model.h5')

# Create Flask app
app = Flask(__name__)
UPLOAD_FOLDER = 'uploads/'
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER

# Emotion labels
emotions = {
    0: 'neutral',
    1: 'calm',
    2: 'happy',
    3: 'sad',
    4: 'angry',
    5: 'fearful',
    6: 'disgust',
    7: 'surprised'
}

# Route for the main page
@app.route('/')
def index():
    return render_template('index.html')

# Route to handle audio file upload and prediction
@app.route('/predict', methods=['POST'])
def predict():
    if 'file' not in request.files:
        return jsonify({'error': 'No file provided'})
```

```
    file = request.files['file']
    if file.filename == '':
        return jsonify({'error': 'No selected file'})

    filename = secure_filename(file.filename)
    filepath = os.path.join(app.config['UPLOAD_FOLDER'], filename)
    file.save(filepath)

    # Extract features and predict
    features = extract_features(filepath)
    features = np.reshape(features, (1, features.shape[0], 1))
    predictions = model.predict(features)
    predicted_emotion = emotions[np.argmax(predictions[0])]

    return jsonify({'emotion': predicted_emotion})

# Feature extraction (same as in your notebook)
def extract_mfcc(wav_file_name):
    y, sr = librosa.load(wav_file_name)
    mfcc = np.mean(librosa.feature.mfcc(y=y, sr=sr, n_mfcc=45).T, axis=0)
    return mfcc

def extract_stft(wav_file_name):
    y, sr = librosa.load(wav_file_name)
    stft = np.abs(librosa.stft(y))
    stft_mean = np.mean(stft, axis=1)
    return stft_mean

def extract_features(wav_file_name):
    mfcc = extract_mfcc(wav_file_name)
    stft = extract_stft(wav_file_name)
```

```
    return np.concatenate((mfcc, stft))

if __name__ == "__main__":
    app.run(debug=True)
```

Script.js ➔

```
function uploadAudio() {
  let audioFile = document.getElementById("audioFile").files[0];

  if (!audioFile) {
    alert("Please select an audio file");
    return;
  }

  let formData = new FormData();
  formData.append("file", audioFile);

  fetch("/predict", {
    method: "POST",
    body: formData,
  })
    .then((response) => response.json())
    .then((data) => {
      document.getElementById("result").innerText =
        "Predicted Emotion: " + data.emotion;
    })
    .catch((error) => {
      console.error("Error:", error);
    });
}
```

Index.html ➔

```
<!DOCTYPE html>
<html lang="en">
 <head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>Sentiment Analysis</title>
  <style>
   body {
    background: linear-gradient(135deg, #f06, #4a90e2);
    color: white;
    font-family: Arial, sans-serif;
    display: flex;
    align-items: center;
    justify-content: center;
    height: 100vh;
    margin: 0;
   }

   .card {
    background-color: rgba(255, 255, 255, 0.1);
    border-radius: 10px;
    padding: 20px;
    box-shadow: 0 4px 20px rgba(0, 0, 0, 0.3);
    text-align: center;
   }

   h1 {
```

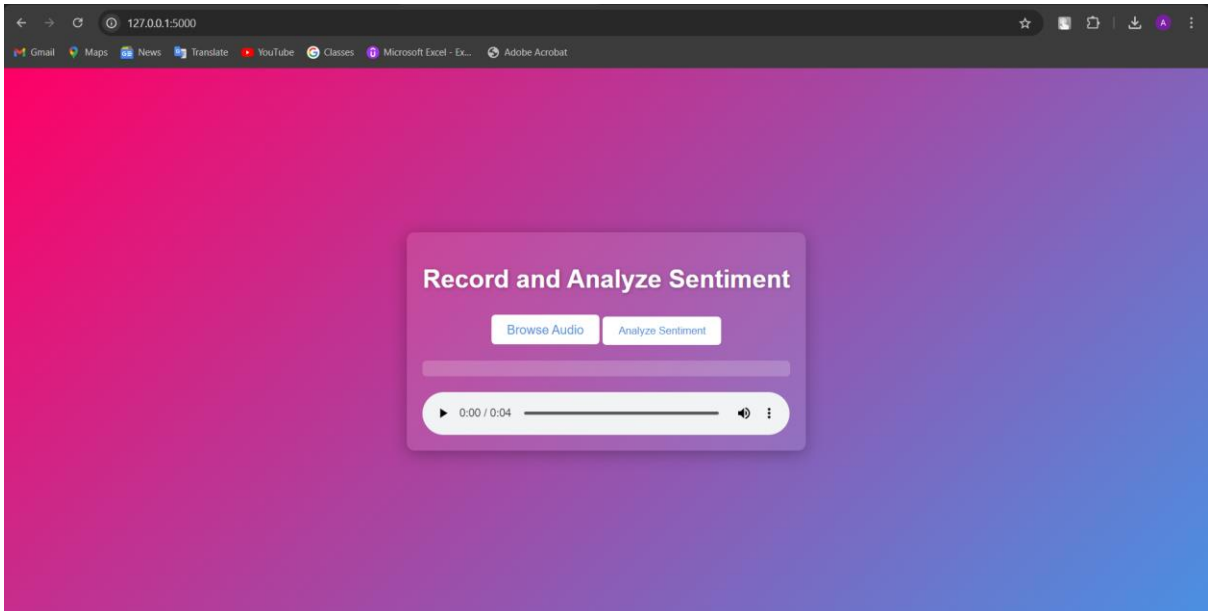
```
  margin-bottom: 20px;
  text-shadow: 1px 1px 5px rgba(0, 0, 0, 0.3);
}

input[type="file"] {
  display: none;
}

.custom-file-input {
  padding: 10px 20px;
  border: none;
  border-radius: 5px;
  background-color: #ffffff;
  color: #4a90e2;
  cursor: pointer;
  transition: background-color 0.3s;
  margin-bottom: 10px;
}

.custom-file-input:hover {
  background-color: #e2e2e2;
}

button {
  padding: 10px 20px;
  border: none;
  border-radius: 5px;
  background-color: #ffffff;
  color: #4a90e2;
  cursor: pointer;
  transition: background-color 0.3s;
```

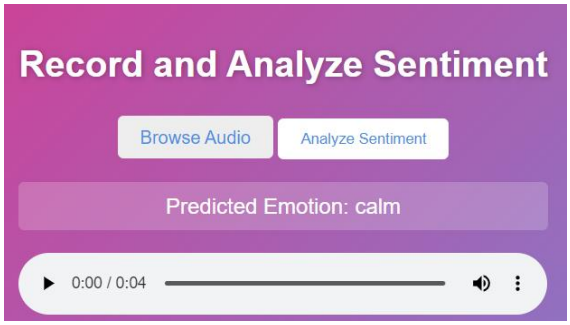
```
      margin-top: 10px;
    }

    button:hover {
      background-color: #e2e2e2;
    }

    #result {
      margin-top: 20px;
      font-size: 1.2em;
      background-color: rgba(255, 255, 255, 0.2);
      padding: 10px;
      border-radius: 5px;
    }

    audio {
      margin-top: 20px;
      width: 100%;
      border-radius: 5px;
    }
  </style>
</head>
<body>
  <div class="card">
    <h1>Record and Analyze Sentiment</h1>
    <label for="audioFile" class="custom-file-input">Browse Audio</label>
    <input
      type="file"
      id="audioFile"
      accept=".wav"
      onchange="previewAudio()"
```

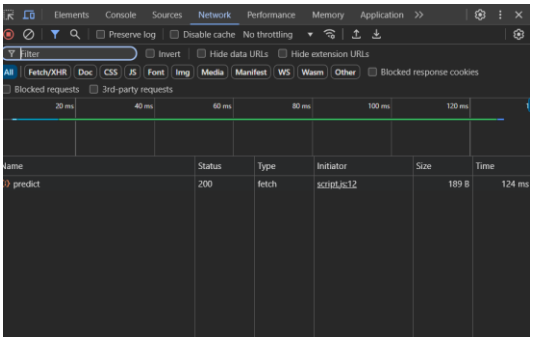
```
    />
    <button onclick="uploadAudio()">Analyze Sentiment</button>
    <p id="result"></p>
    <audio id="audioPreview" controls style="display: none">
      Your browser does not support the audio element.
    </audio>
  </div>

  <script src="{{ url_for('static', filename='script.js') }}"></script>
  <script>
    function previewAudio() {
      const fileInput = document.getElementById("audioFile");
      const audioPreview = document.getElementById("audioPreview");

      const file = fileInput.files[0];
      if (file) {
        const objectURL = URL.createObjectURL(file);
        audioPreview.src = objectURL;
        audioPreview.style.display = "block";
      }
    }
  </script>
 </body>
</html>
```

Output ➔

```
 * Running on http://127.0.0.1:5000
INFO:werkzeug:Press CTRL+C to quit
INFO:werkzeug: * Restarting with stat
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` wi
ll be empty until you train or evaluate the model.
WARNING:werkzeug: * Debugger is active!
INFO:werkzeug: * Debugger PIN: 536-891-044
```







8. Future Scope

The current sentiment analysis project leverages audio files to evaluate emotions through machine learning models. However, the future scope of this project presents exciting opportunities for enhancement and innovation. One significant avenue is the integration of live data input, which would enable real-time sentiment analysis. This enhancement could be particularly valuable in applications such as customer service, mental health monitoring, and interactive gaming, where immediate feedback is crucial.

Implementing live data would involve capturing audio input directly from users through a microphone. By employing libraries like sounddevice or pyaudio, we can stream audio data and process it in real-time, allowing the system to analyse emotions as they are expressed. This capability could further enhance user engagement and provide timely insights.

Additionally, future work could explore the expansion of the model to include a broader range of emotions, beyond the eight categories currently analysed. By utilizing larger and more diverse datasets, the model could improve its accuracy and robustness in detecting nuanced emotional states.

Another area for development is the improvement of the user interface (UI) on the frontend. Implementing features such as visual feedback on emotional analysis and sentiment scoring can enhance the user experience. Moreover, incorporating more detailed reporting and analytics can provide users with deeper insights into the audio data analysed.

Finally, the project could benefit from collaborations with experts in psychology and linguistics to better understand the interplay between verbal and non-verbal cues in emotional expression. This interdisciplinary approach can lead to the development of a more comprehensive sentiment analysis system that accounts for context and varying communication styles.

Overall, the future scope of this project is promising, with numerous possibilities for improving functionality, accuracy, and user engagement.

9. Acknowledgement

I would like to express my sincere gratitude to everyone who has contributed to the success of this sentiment analysis project. First and foremost, I extend my heartfelt thanks to my faculty advisor, whose guidance and support have been invaluable throughout this endeavour. Their insights into machine learning and audio processing have significantly shaped the direction of my work.

I would also like to acknowledge the contributions of my peers, who provided feedback and encouragement during the development process. Collaborative discussions helped refine my ideas and enhance the overall quality of the project. Special thanks to my colleagues in the Smart Systems club for their collaborative spirit and enthusiasm, which inspired me to push the boundaries of what I could achieve.

I am also grateful to the developers of the Librosa library, which served as a fundamental tool for audio analysis in this project. Their dedication to open-source software has made powerful audio processing techniques accessible to researchers and developers worldwide.

Lastly, I would like to thank my family and friends for their unwavering support and encouragement. Their belief in my abilities motivated me to pursue this project with passion and determination. This project stands as a testament to the collaborative efforts and support of those around me, and I am grateful for each contribution.

10. Bibliography

 Librosa Documentation. (2023). Retrieved from https://librosa.org/doc/latest/index.html
This official documentation provides a comprehensive overview of the Librosa library, detailing its installation, functions, and applications in audio analysis.

 TensorFlow Documentation. (2023). Retrieved from https://www.tensorflow.org/api_docs
This documentation covers the TensorFlow library, explaining its functionalities and offering guidance for building and training machine learning models, including neural networks for audio analysis.

 Fletcher, H. & G. K. (2021). *Introduction to Audio Analysis: A MATLAB Approach*. Academic Press.
This book offers a comprehensive introduction to audio analysis, focusing on the fundamentals and practical applications in the field, making it suitable for both beginners and advanced practitioners.

 Borkowski, D. (2017). *Digital Audio Signal Processing*. Wiley.
This book delves into digital audio signal processing techniques, covering various algorithms and their applications, including feature extraction and sound synthesis.

 Vasilakis, C. (2020). "Understanding Convolutional Neural Networks: A Primer." Medium. Retrieved from https://medium.com/@chris.vasilakis/understanding-convolutional-neural-networks-a-primer-4ed3b43f156

Thank You!