**AI Project**

**Emotion Based Music Recommender**

**Introduction:**

The "Emotion Based Music Recommender" is a Streamlit application that leverages computer vision and machine learning techniques to capture the user's emotion through facial and hand movements and recommend songs based on the detected emotion, preferred language, and singer. The application utilizes various libraries and technologies to provide an interactive and personalized music recommendation experience.

The core functionalities of the application include real-time emotion analysis from video streams, integration with a pre-trained emotion classification model, visualization of landmarks on video frames, and the ability to recommend songs based on the captured emotion and user inputs.

**Libraries used:**

**Streamlit:** The Streamlit library is used to create the user interface and define the app's layout. It provides functions for displaying text, input fields, buttons, and headers. Additionally, Streamlit is used to manage the session state, which keeps track of the app's state and variables.

**streamlit\_webrtc:** This library extends Streamlit and provides the webrtc\_streamer function, which enables real-time video streaming and processing in web applications. It is used in this code to stream video frames and apply the EmotionProcessor for emotion analysis.

**av:** The av library is used for working with audio and video data. In this code, it is used to convert the video frames to an av.VideoFrame object, which can be processed and returned as the output of the EmotionProcessor class.

**cv2 (OpenCV):** OpenCV is a powerful computer vision library that offers various functions for image and video processing. In this code, OpenCV is used to perform image manipulations, such as flipping the video frames horizontally (cv2.flip()). It is also used to draw landmarks on the frames using the drawing\_utils module (drawing.draw\_landmarks()). Additionally, OpenCV is used for color space conversion (cv2.cvtColor()).

**numpy:** NumPy is a fundamental library for numerical computing in Python. It provides support for efficient array operations and mathematical functions. In this code, NumPy is used to manipulate and reshape arrays of landmarks extracted from the video frames.

**mediapipe:** Mediapipe is a powerful cross-platform framework for building machine learning pipelines for multimodal applications. In this code, the holistic and hands modules from Mediapipe are used to detect face landmarks (res.face\_landmarks) and hand landmarks (res.left\_hand\_landmarks, res.right\_hand\_landmarks) in the video frames. The drawing\_utils module is used to visualize the detected landmarks on the frames.

**Keras:** Keras is a high-level deep learning library that provides a user-friendly API for building and training neural networks. In this code, Keras is used to load a pre-trained emotion classification model (load\_model()). The loaded model is then used to make predictions on the extracted landmarks.

**webbrowser:** The webbrowser module provides a high-level interface for displaying web-based documents. It is used in this code to open a web browser with a YouTube search query (webbrowser.open()) based on the user's selected language, emotion, and singer.

These libraries provide the necessary functionality for capturing video frames, analyzing emotions using landmark detection, making predictions with a pre-trained model, and interacting with web browsers to recommend songs based on user inputs.

**Implementation:**

import streamlit as st

from streamlit\_webrtc

import webrtc\_streamer

import av

import cv2

import numpy as np

import mediapipe as mp

from keras.models import load\_model

import webbrowser

**# Load emotion classification model and label information**

model = load\_model("model.h5")

label = np.load("labels.npy")

**# Initialize Mediapipe and drawing utilities**

holistic = mp.solutions.holistic

hands = mp.solutions.hands

holis = holistic.Holistic()

drawing = mp.solutions.drawing\_utils

**# Set up Streamlit application header**

st.header("Emotion Based Music Recommender")

**# Initialize session state**

**# Check if the "run" key exists in session state, and initialize it if not**

if "run" not in st.session\_state:

st.session\_state["run"] = "true"

try:

**# Attempt to load the previous emotion from file**

emotion = np.load("emotion.npy")[0]

except:

emotion = ""

**# Set the "run" key based on the presence of a saved emotion**

if not emotion:

st.session\_state["run"] = "true"

else:

st.session\_state["run"] = "false"

class EmotionProcessor:

**# Perform image processing steps**

def recv(self, frame):

frm = frame.to\_ndarray(format="bgr24")

**# Flip the frame horizontally**

frm = cv2.flip(frm, 1)

**# Process holistic landmarks**

res = holis.process(cv2.cvtColor(frm, cv2.COLOR\_BGR2RGB))

lst = []

if res.face\_landmarks:

**# Extract face landmarks**

for i in res.face\_landmarks.landmark:

lst.append(i.x - res.face\_landmarks.landmark[1].x)

lst.append(i.y - res.face\_landmarks.landmark[1].y)

**# Extract left hand landmarks**

if res.left\_hand\_landmarks:

for i in res.left\_hand\_landmarks.landmark:

lst.append(i.x - res.left\_hand\_landmarks.landmark[8].x)

lst.append(i.y - res.left\_hand\_landmarks.landmark[8].y)

else:

**# If left hand landmarks are not available, fill with zeros**

for i in range(42):

lst.append(0.0)

**# Extract right hand landmarks**

if res.right\_hand\_landmarks:

for i in res.right\_hand\_landmarks.landmark:

lst.append(i.x - res.right\_hand\_landmarks.landmark[8].x)

lst.append(i.y - res.right\_hand\_landmarks.landmark[8].y)

else:

**# If right hand landmarks are not available, fill with zeros**

for i in range(42):

lst.append(0.0)

**# Convert the landmarks to numpy array**

lst = np.array(lst).reshape(1, -1)

**# Make emotion prediction**

pred = label[np.argmax(model.predict(lst))]

print(pred)

cv2.putText(frm, pred, (50, 50), cv2.FONT\_ITALIC, 1, (255, 0, 0), 2)

np.save("emotion.npy", np.array([pred]))

**# Visualize landmarks**

drawing.draw\_landmarks(frm, res.face\_landmarks, holistic.FACEMESH\_TESSELATION,

landmark\_drawing\_spec=drawing.DrawingSpec(color=(0,0,255), thickness=-1, circle\_radius=1),

connection\_drawing\_spec=drawing.DrawingSpec(thickness=1))

drawing.draw\_landmarks(frm, res.left\_hand\_landmarks, hands.HAND\_CONNECTIONS)

drawing.draw\_landmarks(frm, res.right\_hand\_landmarks, hands.HAND\_CONNECTIONS)

**#Returning the processed frames**

return av.VideoFrame.from\_ndarray(frm, format="bgr24")

**#Streamlit UI and Music Recommendation**

lang = st.text\_input("Language")

singer = st.text\_input("singer")

if lang and singer and st.session\_state["run"] != "false":

webrtc\_streamer(key="key", desired\_playing\_state=True,

video\_processor\_factory=EmotionProcessor)

btn = st.button("Recommend me songs")

if btn:

if not(emotion):

st.warning("Please let me capture your emotion first")

st.session\_state["run"] = "true"

else:

webbrowser.open(f"https://www.youtube.com/results?search\_query={lang}+{emotion}+song+{singer}")

np.save("emotion.npy", np.array([""]))

st.session\_state["run"] = "false"

The code after the EmotionProcessor class defines the Streamlit user interface and music recommendation functionality.

The lang and singer variables capture user inputs for the language and singer preferences.

The if condition checks if both lang and singer are provided and the "run" key in the session state is not set to "false".

Inside the if block, the webrtc\_streamer function is called to initiate video streaming with the EmotionProcessor as the video processor factory.

The btn variable represents a button for recommending songs.

If the button is clicked, the code checks if the emotion is not empty. If it's empty, a warning message is displayed to capture the emotion first, and the "run" key is set to "true". Otherwise, a YouTube search URL is generated based on the language, emotion, and singer inputs. The emotion is then cleared, the "run" key is set to "false", and the URL is opened in a web browser.

**Conclusion:**

In conclusion, the provided code implements an Emotion-Based Music Recommender using a combination of computer vision and web technologies. It utilizes the Streamlit library for creating the user interface and the Streamlit Webrtc package for video streaming. The code incorporates various libraries such as OpenCV, NumPy, Mediapipe, and Keras for image processing, hand and facial landmark detection, and emotion classification.

The main functionality of the code revolves around an EmotionProcessor class that receives video frames, applies image processing techniques to detect facial and hand landmarks, and predicts the user's emotion based on the landmarks. The detected emotion is displayed on the video frame using text overlay. The code also draws the detected landmarks on the frame for visual feedback.

The user interface allows the user to input their preferred language and singer. If both inputs are provided and the "run" state is not set to "false", the code initiates video streaming with the EmotionProcessor as the video processor factory. Additionally, there is a button to recommend songs. When clicked, it checks if the user's emotion is captured. If not, a warning message is displayed, and the "run" state is set to "true". Otherwise, it generates a YouTube search URL based on the language, emotion, and singer inputs, opens the URL in a web browser, clears the emotion data, and sets the "run" state to "false".

Overall, this code combines computer vision, machine learning, and web technologies to create an interactive music recommender system based on the user's emotion.

**Submitted By:**

* Hassam Hameed (F2020266481)
* Waleed Waheed Butt (F2020266469)
* Farrukh Aziz (F2020266248)