AI-Powered Mood-Based Music Recommender Using Facial Emotion Recognition

CODING

emotion_model_training.py

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping
import os
from collections import Counter
train_dir = r'C:\Users\megha\Desktop\Major Project\Major_code\emotion_dataset\train'
val_dir = r'C:\Users\megha\Desktop\Major Project\Major_code\emotion_dataset\test'
train_datagen = ImageDataGenerator(
rescale=1./255,
rotation_range=20,
zoom_range=0.2,
width_shift_range=0.2,
height_shift_range=0.2,
shear_range=0.2,
horizontal_flip=True
)
val_datagen = ImageDataGenerator(rescale=1./255)
img_size = (48, 48)
batch_size = 32
train_generator = train_datagen.flow_from_directory(
train_dir,
target_size=img_size,
```

```
color_mode='grayscale',
batch_size=batch_size,
class_mode='categorical'
51)
val_generator = val_datagen.flow_from_directory(
val_dir,
target_size=img_size,
color_mode='grayscale',
batch_size=batch_size,
class_mode='categorical'
)
def get_class_weights(generator):
counter = Counter(generator.classes)
max_count = float(max(counter.values()))
return {cls: max_count / count for cls, count in counter.items()}
class_weights = get_class_weights(train_generator)
model = Sequential([
Conv2D(32, (3, 3), activation='relu', input_shape=(48, 48, 1)),
MaxPooling2D(2, 2),
Conv2D(64, (3, 3), activation='relu'),
MaxPooling2D(2, 2),
Conv2D(128, (3, 3), activation='relu'),
MaxPooling2D(2, 2),
Flatten(),
Dense(256, activation='relu'),
Dropout(0.5),
Dense(train_generator.num_classes, activation='softmax')
])
model.compile(optimizer=Adam(learning_rate=0.0001), loss='categorical_crossentropy',
metrics=['accuracy'])
```

```
early_stop = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)
model.fit(
train_generator,
validation_data=val_generator,
epochs=10,
callbacks=[early_stop],
52class_weight=class_weights # Optional, helps in case of imbalance
)
model.save('emotion_model.h5')
print(" Model saved as 'emotion_model.h5'")
```

main.py

```
import cv2
import numpy as np
from keras.models import load_model
import webbrowser
from collections import Counter
import tkinter as tk
from tkinter import filedialog, simpledialog, messagebox
from PIL import Image, ImageTk
import sys
sys.stdout.reconfigure(encoding='utf-8')
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
emotion_model = load_model('emotion_model.h5')
emotion_labels = ['Angry', 'Disgust', 'Fear', 'Happy', 'Neutral', 'Sad', 'Surprise']
emotion_language_playlist = {
"Happy": {
"English": "https://youtube.com/playlist?list=PL0V5eyUGeaFkZ-
TJzPIi1QxM0loCYt5zI&si=5bM-TVjtYM-La2Aq",
"Hindi":
```

"https://youtube.com/playlist?list=PL0V5eyUGeaFnntAMWYo70UYb_YfyuDzd6&si=3_RnGx

5Gu9UbyjZt",

"Kannada":

"https://youtube.com/playlist?list=PL0V5eyUGeaFlIp9UyX_BRhChhz05ayVgn&si=O30YJAYj

MFP0CpbW",

"Telugu":"https://youtube.com/playlist?list=PL0V5eyUGeaFlXAoe7lKENH8WoG1ARA61o&

si=s_MUkmr4CQYzWmcV"

},

53"Sad": {

"English": "https://youtube.com/playlist?list=PL0V5eyUGeaFlJCgKFAY-

AKhMokQEvWHvl&si=_J4MieLDwyfj76xq",

"Hindi":

"https://youtube.com/playlist?list=PL0V5eyUGeaFmQtvkTHgpysKxGorTIBRZG&si=y-

7_C3EnmNlFns_V",

"Kannada": "https://youtube.com/playlist?list=PL0V5eyUGeaFlk8WrVunJcXlQrwdqp-

KYS&si=vNfAJw_4u1m3vHVw",

"Telugu":"https://youtube.com/playlist?list=PL0V5eyUGeaFmQVRCcWYBjHNbe1ibi__pM&

si=c8UaObrAaFvEmFOq"

},

"Angry": {

"English":

"https://youtube.com/playlist?list=PL0V5eyUGeaFnQwaI8U2zVkMPL1002pUvB&si=bTjCTA

6aZ9bXRAjd",

"Hindi":

"https://youtube.com/playlist?list=PL0V5eyUGeaFmvHxlcdb4mtnRuw6rU0Su6&si=ExRncBjS

gqX9Xzw2",

```
"Kannada": "https://youtube.com/playlist?list=PL0V5eyUGeaFl_Bl6ued4-
XUBCkt6PhjzJ&si=gML_n6ORQ-CaZ-Nr",
"Telugu":"https://youtube.com/playlist?list=PL0V5eyUGeaFkRJSr2pYxmdQCBVWn0KK-
Y&si=v4U8v0uGF1oNiD6k"
},
"Fear": {
"English": "https://youtube.com/playlist?list=PL0V5eyUGeaFlZ8FM5d2ipr9-
7EIrCyxPy&si=0H0aHoo_hvl_x-s_",
"Hindi":
"https://youtube.com/playlist?list=PL0V5eyUGeaFn0_sjsb079YugfzpCBQdj3&si=HWZVx2
Kw
oLEEQ5Uu",
"Kannada":
"https://youtube.com/playlist?list=PL0V5eyUGeaFmsCp_nunZJzi1xtBt3Wjt1&si=3YqCdI5
XK
TljJOyH",
54"Telugu":"https://youtube.com/playlist?list=PL0V5eyUGeaFk-
0zQ3jIP3PGHiZtpSRUdY&si=eLe5s5fDsma7ouCu"
}
}
def play_youtube_music(emotion, language):
emotion_label = emotion.split(" ")[0].strip()
try:
playlist_url = emotion_language_playlist[emotion_label][language]
print(f"[♪] Playing {language} playlist for {emotion_label} mood...")
webbrowser.open(playlist_url)
except KeyError:
print(f"[X] No playlist found for emotion '{emotion_label}' in language '{language}'.")
def detect_emotion(frame):
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
faces = face_cascade.detectMultiScale(gray, 1.3, 5)
```

```
for (x, y, w, h) in faces:
roi_gray = gray[y:y+h, x:x+w]
roi_gray = cv2.resize(roi_gray, (48, 48))
roi = roi_gray.astype("float") / 255.0
roi = np.reshape(roi, (1, 48, 48, 1))
prediction = emotion_model.predict(roi, verbose=0)
print(f"Prediction: {prediction}") # Print all probabilities
max_index = int(np.argmax(prediction))
if max_index < len(emotion_labels):
confidence = prediction[0][max_index]
detected_emotion = f"{emotion_labels[max_index]} ({confidence:.2f})"
else:
detected_emotion = "Unknown"
return detected_emotion
return None
55def webcam_mode():
cap = cv2.VideoCapture(0)
frame_count = 0
collected_emotions = []
while True:
ret, frame = cap.read()
if not ret:
break
emotion = detect_emotion(frame)
banner_height = 40
cv2.rectangle(frame, (0, 0), (frame.shape[1], banner_height), (255, 182, 193), -1)
if emotion:
display_text = f"Emotion: {emotion}"
collected_emotions.append(emotion)
frame_count += 1
```

```
else:
display_text = "No face detected"
cv2.putText(frame, display_text, (10, 30), cv2.FONT_HERSHEY_SIMPLEX,
0.9, (0, 0, 0), 2, cv2.LINE_AA)
cv2.imshow('Emotion Detection - Webcam', frame)
if frame_count >= 50:
most_common = Counter(collected_emotions).most_common(1)[0][0]
# Removed emoji from print statement
print(f"Detected Emotion (final): {most_common}")
language = simpledialog.askstring("Language", "Enter preferred language
(English/Hindi/Kannada):")
play_youtube_music(most_common, language)
break
56if cv2.waitKey(1) & 0xFF == ord('q'):
break
cap.release()
cv2.destroyAllWindows()
def image_mode():
file_path = filedialog.askopenfilename(filetypes=[("Image files", "*.jpg *.png *.jpeg")])
if not file_path:
return
image = cv2.imread(file_path)
emotion = detect_emotion(image)
if emotion:
messagebox.showinfo("Detected Emotion", f"Emotion: {emotion}")
language = simpledialog.askstring("Language", "Enter preferred language
(English/Hindi/Kannada):")
play_youtube_music(emotion, language)
else:
messagebox.showerror("Error", "No face detected in the image.")
```

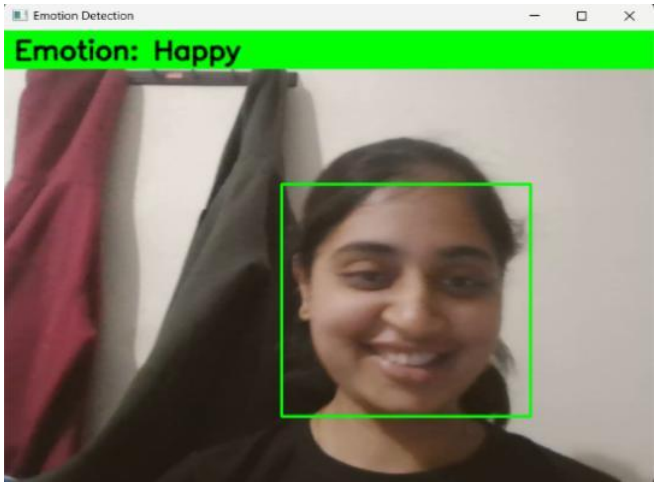
```
def main_gui():
root = tk.Tk()
root.title("?AI-Powered Mood-Based Music Recommender")
root.geometry("500x400")
root.configure(bg="#fce4ec") # light pink background
title = tk.Label(root, text="Mood-Based Music Recommender", font=("Helvetica", 18,
"bold"),
bg="#fce4ec", fg="#880e4f")
title.pack(pady=30)
57webcam_btn = tk.Button(root, text="? Use Webcam", font=("Helvetica", 14),
bg="#f06292",
fg="white",
width=20, height=2, relief="raised", bd=3, command=webcam_mode)
webcam_btn.pack(pady=15)
upload_btn = tk.Button(root, text="? Upload Image", font=("Helvetica", 14), bg="#ba68c8",
fg="white",
width=20, height=2, relief="raised", bd=3, command=image_mode)
upload_btn.pack(pady=15)
footer = tk.Label(root, text="Developed with ?", bg="#fce4ec", fg="#6a1b9a",
font=("Helvetica", 10))
footer.pack(side="bottom", pady=20)
root.mainloop()
if __name__ == "__main__":
main_gui()
```

OUTPUT

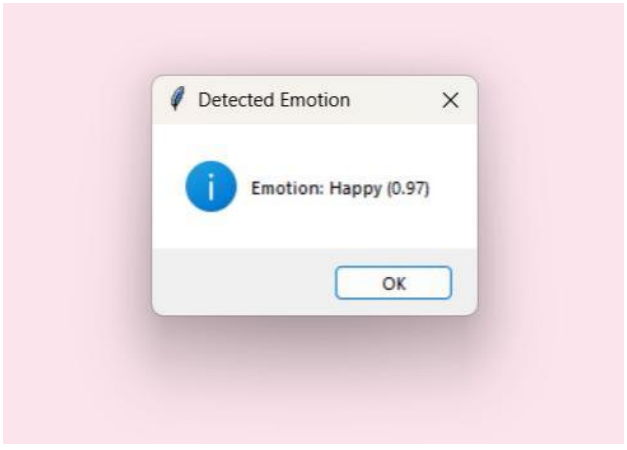
1. Model Prediction

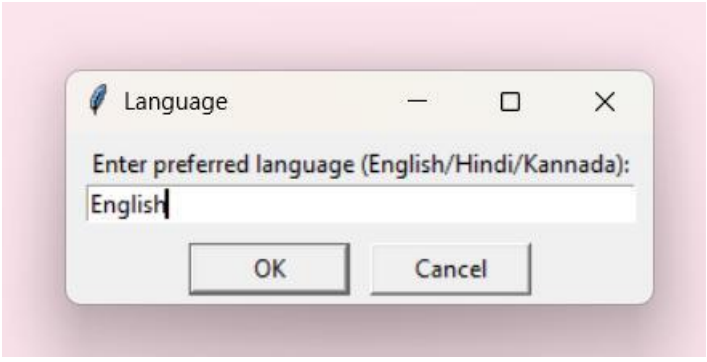
```
Prediction: [[0.00730692 0.01720635 0.00500907 0.9660098  0.00137707 0.00110199
  0.00198884]]
[♪] Playing English playlist for Happy mood...
```

2. Emotion Detection

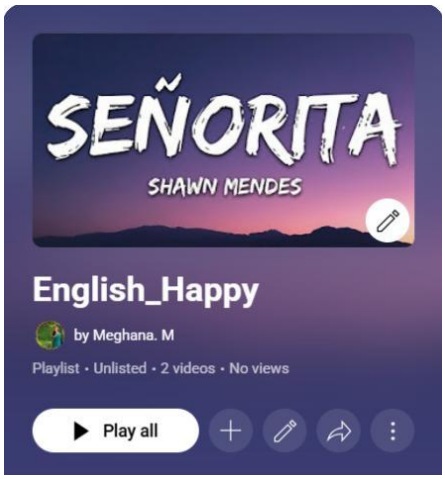


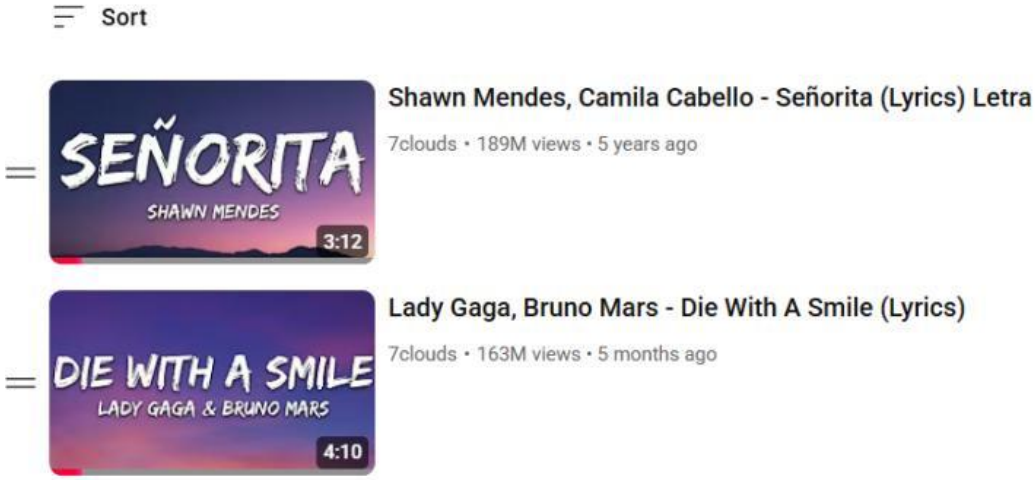
3. Detected Emotion and Language





4. Music Recommendation





5. User Interface