


app.py

```
1 from flask import Flask, request, jsonify, render_template, Response, send_file
2 from tensorflow.keras.models import load_model
3 from tensorflow.keras.preprocessing.image import img_to_array
4 import numpy as np
5 import cv2
6 import os
7
8 app = Flask(__name__)
9
10 # Load model and Haarcascade
11 model = load_model('model.h5')
12 face_classifier = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
13
14 # Emotion Labels
15 emotion_labels = ['Angry', 'Disgust', 'Fear', 'Happy', 'Neutral', 'Sad', 'Surprise']
16
17 # ===== Home Page =====
18 @app.route('/')
19 def home():
20     return render_template('index.html')
21
22 # ===== Image Upload and Detection =====
23 @app.route('/upload-image', methods=['POST'])
24 def upload_image():
25     try:
26         file = request.files['file']
27         if not file:
28             return jsonify({'error': 'No file uploaded.'})
29
30         # Read uploaded image
31         img_bytes = file.read()
32         nparr = np.frombuffer(img_bytes, np.uint8)
33         frame = cv2.imdecode(nparr, cv2.IMREAD_COLOR)
34
35         gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
36         faces = face_classifier.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5)
37
38         for (x, y, w, h) in faces:
39             roi_gray = gray[y:y+h, x:x+w]
40             roi_gray = cv2.resize(roi_gray, (48, 48), interpolation=cv2.INTER_AREA)
41             roi = roi_gray.astype('float') / 255.0
42             roi = img_to_array(roi)
43             roi = np.expand_dims(roi, axis=0)
44
45             prediction = model.predict(roi)[0]
46             label = emotion_labels[prediction.argmax()]
47             label_position = (x, y - 10)
```

```
48
49     # Draw rectangle and label
50     cv2.rectangle(frame, (x, y), (x+w, y+h), (0,255,255), 2)
51     cv2.putText(frame, label, label_position, cv2.FONT_HERSHEY_SIMPLEX, 1,
(0,255,0), 2)
52
53     # Save the output image temporarily
54     output_image_path = 'static/output.jpg'
55     cv2.imwrite(output_image_path, frame)
56
57     # Return the processed image
58     return send_file(output_image_path, mimetype='image/jpeg')
59
60 except Exception as e:
61     return jsonify({'error': str(e)})
62
63 # ===== Video Upload and Detection =====
64
65 # ===== Video Upload and Detection (Optimized) =====
66 @app.route('/upload-video', methods=['POST'])
67 def upload_video():
68     try:
69         file = request.files['file']
70         if not file:
71             return jsonify({'error': 'No video uploaded.'})
72
73         temp_video_path = 'temp_video.mp4'
74         file.save(temp_video_path)
75
76         cap = cv2.VideoCapture(temp_video_path)
77
78         # Get original video details
79         fourcc = cv2.VideoWriter_fourcc(*'mp4v')
80         width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
81         height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
82         fps = cap.get(cv2.CAP_PROP_FPS)
83
84         out = cv2.VideoWriter('static/output_video.mp4', fourcc, fps, (width, height))
85
86         frame_skip_rate = 5 #  Process 1 frame out of every 5 frames
87         frame_count = 0
88
89         while True:
90             ret, frame = cap.read()
91             if not ret:
92                 break
93
94             frame_count += 1
95             if frame_count % frame_skip_rate != 0:
```

```
96         out.write(frame)
97         continue
98
99         gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
100         faces = face_classifier.detectMultiScale(gray, scaleFactor=1.1,
minNeighbors=5)
101
102         for (x, y, w, h) in faces:
103             roi_gray = gray[y:y+h, x:x+w]
104             roi_gray = cv2.resize(roi_gray, (48, 48), interpolation=cv2.INTER_AREA)
105             roi = roi_gray.astype('float') / 255.0
106             roi = img_to_array(roi)
107             roi = np.expand_dims(roi, axis=0)
108
109             prediction = model.predict(roi)[0]
110             label = emotion_labels[prediction.argmax()]
111             label_position = (x, y - 10)
112
113             cv2.rectangle(frame, (x, y), (x+w, y+h), (0,255,255), 2)
114             cv2.putText(frame, label, label_position, cv2.FONT_HERSHEY_SIMPLEX, 1,
(0,255,0), 2)
115
116         out.write(frame)
117
118         cap.release()
119         out.release()
120         os.remove(temp_video_path)
121
122         # After video processed, show result page
123         return render_template('video_result.html')
124
125     except Exception as e:
126         return jsonify({'error': str(e)})
127
128 # ===== Real-Time Webcam Streaming =====
129 camera = cv2.VideoCapture(0)
130
131 def generate_frames():
132     while True:
133         success, frame = camera.read()
134         if not success:
135             break
136         else:
137             gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
138             faces = face_classifier.detectMultiScale(gray, 1.1, 5)
139
140             for (x, y, w, h) in faces:
141                 roi_gray = gray[y:y+h, x:x+w]
142                 roi_gray = cv2.resize(roi_gray, (48, 48), interpolation=cv2.INTER_AREA)
```

```
143         roi = roi_gray.astype('float') / 255.0
144         roi = img_to_array(roi)
145         roi = np.expand_dims(roi, axis=0)
146
147         prediction = model.predict(roi)[0]
148         label = emotion_labels[prediction.argmax()]
149         label_position = (x, y-10)
150
151         cv2.rectangle(frame, (x, y), (x+w, y+h), (0,255,255), 2)
152         cv2.putText(frame, label, label_position, cv2.FONT_HERSHEY_SIMPLEX, 1,
153         (0,255,0), 2)
154
155         ret, buffer = cv2.imencode('.jpg', frame)
156         frame = buffer.tobytes()
157
158         yield (b'--frame\r\n'
159               b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n')
160
161 @app.route('/video_feed')
162 def video_feed():
163     return Response(generate_frames(),
164                     mimetype='multipart/x-mixed-replace; boundary=frame')
165
166 @app.route('/realtime')
167 def realtime():
168     return render_template('realtime.html')
169
170 # ===== Run Flask App =====
171 if __name__ == "__main__":
172     app.run(host='0.0.0.0', port=5000)
```