modify the code to use your smile and the presence of crows feet around your eyes to determine the confidence level, you can adjust the calculation of the confidence percentage based on the following:

If your mouth is open and your teeth are visible, increase the confidence percentage by 20%.

If crows feet are detected around your eyes, increase the confidence percentage by 10%.

If both of the above conditions are met, increase the confidence percentage by 30%.

Add variables to keep track of the smile and crows feet detection.

Check if my mouth is open and my teeth are visible, then set the smile detection variable to True.

Check if crows feet are detected around my eyes, then set the crows feet detection variable to True.

Calculate the confidence percentage based on the smile and crows feet detection, and the number of detected eyes and smiles.

Add the confidence percentage to the text to display next to the recognized face.

Make sure to do it bit by bit because when you try to do it all at once you freeze, do it step by step explaining each section

here is the code:

import cv2

import numpy as np

# Load the pre-trained face, eye and smile detection models from OpenCV

face\_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade\_frontalface\_default.xml')

eye\_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade\_eye.xml')

smile\_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade\_smile.xml')

def recognize\_emotion\_and\_face():

"""

Recognize emotions and faces in a webcam video stream using OpenCV

Returns:

dict: a dictionary of emotions and faces information or None if an error occurs

"""

# Open a connection to the default webcam

try:

cap = cv2.VideoCapture(0)

if not cap.isOpened():

raise ValueError("Unable to open webcam")

except Exception as e:

print("Error: ", e)

return None

while True:

# Capture a frame from the webcam video stream

ret, frame = cap.read()

# Convert the frame to JPEG format

ret, image\_data = cv2.imencode('.jpg', frame)

# Display the captured frame

cv2.imshow('Captured Frame', frame)

# Check for key presses

if cv2.waitKey(1) == ord('q'):

break

# Process each detected face

gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)

faces = face\_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5)

for (x, y, w, h) in faces:

# Draw a rectangle around the face

cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)

# Crop the face region from the frame

face\_roi\_gray = gray[y:y+h, x:x+w]

face\_roi\_color = frame[y:y+h, x:x+w]

# Detect eyes in the face region

eyes = eye\_cascade.detectMultiScale(face\_roi\_gray, scaleFactor=1.1, minNeighbors=5)

# Detect smile in the face region

smiles = smile\_cascade.detectMultiScale(face\_roi\_gray, scaleFactor=1.1, minNeighbors=5)

# Check if both eyes and smile are detected

if len(eyes) > 1 and len(smiles) > 0:

# Draw rectangles around the eyes

for (ex, ey, ew, eh) in eyes:

cv2.rectangle(face\_roi\_color, (ex, ey), (ex+ew, ey+eh), (0, 0, 255), 2)

# Draw a rectangle around the smile

for (sx, sy, sw, sh) in smiles:

cv2.rectangle(face\_roi\_color, (sx, sy), (sx+sw, sy+sh), (255, 0, 0), 2)

# Search for crow's feet or wrinkles around the outer corners of the eyes

if ex > 0 and ey > 0:

# Define the search region around the eye

search\_region = gray[ey:ey+eh, ex:ex+ew]

# Apply a median blur to remove noise

search\_region = cv2.medianBlur(search\_region, 5)

# Search for crow's feet or wrinkles around the outer corners of the eyes

crow\_feet\_present = False

for i in range(1, len(search\_region)-1):

for j in range(1, len(search\_region[0])-1):

# Check for a change in brightness in the search region

brightness\_change = abs(int(search\_region[i][j+1]) - int(search\_region[i][j]))

if brightness\_change > 10:

crow\_feet\_present = True

break

brightness\_change = abs(int(search\_region[i+1][j]) - int(search\_region[i][j]))

if brightness\_change > 10:

crow\_feet\_present = True

break

if crow\_feet\_present:

break

# If crow's feet or wrinkles are detected, draw a green circle around the eye

if crow\_feet\_present:

center = (int(x+ex+ew/2), int(y+ey+eh/2))

radius = int(ew/2)

cv2.circle(face\_roi\_color, center, radius, (0, 255, 0), 2)

# Calculate the confidence percentage

confidence\_percent = round((len(eyes) + len(smiles)) / 3 \* 100)

# Determine the emotion label based on the detected face region

if confidence\_percent > 60:

emotion\_label = "Happy"

else:

emotion\_label = "Neutral"

# Display the emotion label and confidence percentage next to the recognized face

text = f"{emotion\_label} ({confidence\_percent}%)"

cv2.putText(frame, text, (x, y-10), cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, (255, 255, 255), 2)

# Show the processed frame

cv2.imshow('Processed Frame', frame)

# Release the webcam and close all windows

cap.release()

cv2.destroyAllWindows()

def main():

# Call the recognize\_emotion\_and\_face function

recognize\_emotion\_and\_face()

if \_\_name\_\_ == '\_\_main\_\_':

main()ss