**SIH Success**

1 Install python and opencv in your laptop.

2 Install python extension in vs code

3 Run code in Vs code

4 Open camera through code

### Step 1: Verify Python Installation

1. **Check if Python is installed**:
   * Open a terminal or command prompt and type:

python --version

* + If Python is installed, this will display the version (e.g., Python 3.x.x). If it’s not installed, you’ll need to install it.

1. **Check if pip is installed**:
   * Run the following command to check for pip:

pip --version

* + If pip is installed, this will show the pip version. If not, or if you get an error, proceed to the next steps.

**Step 2: Install or Reinstall Python**

If Python is not installed, or pip is not working, you can reinstall Python and make sure pip is correctly set up.

1. **Download and install Python**:
   * Go to the official [Python download page](https://www.python.org/downloads/).
   * Download the latest version (Python 3.x.x).
   * **Important**: During the installation, check the box that says **"Add Python to PATH"**. This step ensures that Python and pip can be run from the terminal.
2. **Verify the installation**: After installation, open a new command prompt and run:

python --version

pip –version

### Install OpenCV

Once Python and pip are properly installed, you can install OpenCV by running:

Copy code

pip install opencv-python

**1. VS Code (Visual Studio Code) with a Python extension**

* **Install OpenCV**: Before running the code, ensure you have OpenCV installed. You can install it via the terminal:

pip install opencv-python

* **Set up a Python Environment**: If you haven't already, set up a Python environment in VS Code.
  + Open VS Code.
  + Install the **Python extension** (if not installed).
  + Create a new Python file (e.g., webcam\_capture.py), and paste the code into the file.
* **Run the code**:
  + Open the **Integrated Terminal** in VS Code (go to Terminal > New Terminal).
  + In the terminal, type the following to run the script:

Copy code

python webcam\_capture.py

* + The code will capture video from your webcam and display it in a window.

Code to run in Vs code in webcam\_capture.py

import cv2

# Start video capture from your laptop's webcam (usually device 0)

cap = cv2.VideoCapture(0)

while True:

    # Capture frame-by-frame

    ret, frame = cap.read()

    if not ret:

        print("Failed to grab frame")

        break

    # Display the frame

    cv2.imshow('Live Feed', frame)

    # Break the loop with 'q' key

    if cv2.waitKey(1) & 0xFF == ord('q'):

        break

# Release the capture and close windows

cap.release()

cv2.destroyAllWindows()

run this code in new terminal to close terminal delete teminal

Yes, exactly! Instead of going through the complexity of building YOLO from source (especially with MinGW or dealing with make errors), you can use **PyTorch YOLOv5**, which is much easier to set up and more user-friendly.

**Why Use PyTorch YOLOv5?**

* **Pre-built Models**: You don’t need to manually compile anything.
* **Easy Integration**: It works seamlessly with Python and OpenCV.
* **Active Development**: YOLOv5 has more frequent updates and easier-to-use APIs.
* **Flexible Hardware Usage**: You can run it on both CPU and GPU without worrying about CUDA setup complications.

so is thisthe easyiedt yolo version to work on or yolov7 or anyother version is easier or should go with yolov5

In terms of ease of use, **YOLOv5** is widely considered the **easiest version** to work with, especially for these reasons:

### Why YOLOv5 is the Easiest:

1. **No Compiling Required**: Unlike YOLOv4 or earlier versions (or even YOLOv7), YOLOv5 is fully written in Python, which eliminates the need for manually building the code or dealing with make or cmake errors.
2. **Active Community & Documentation**: YOLOv5 has a large and active community, with extensive documentation and tutorials that can help resolve most issues. It also has lots of sample code for integrating with OpenCV.
3. **Torch Hub Integration**: YOLOv5 can be easily loaded through torch.hub with a simple line of code, reducing setup friction.
4. **Regular Updates**: YOLOv5 is updated more frequently and is actively maintained, meaning you'll have better performance and bug fixes compared to older versions like YOLOv3 or YOLOv4.

### Comparing to YOLOv7:

* **YOLOv7** is newer and often outperforms YOLOv5 in terms of accuracy and speed (slightly), but it's also more complex to set up and not as user-friendly as YOLOv5.
* YOLOv7 can offer better performance **if you need to squeeze out more accuracy or speed**, but for most practical applications, the performance difference is minimal.

### Summary:

* If **ease of use** and a **quick setup** are your priorities, stick with **YOLOv5**.
* If you're comfortable with a bit more complexity and want the **best performance**, you could try **YOLOv7**—but it will require more effort and might not have as much support as YOLOv5.

For your current project, where you're working with **OpenCV** and want something straightforward, **YOLOv5** is the best option.

what about yolov9 is it also complex in intgration

As of now, there is no official **YOLOv9** released. YOLOv5 and YOLOv7 are currently some of the latest and most popular versions.

**yolo and opencv only:**

import torch

import cv2

# Load YOLOv5 model

model = torch.hub.load('ultralytics/yolov5', 'yolov5s', pretrained=True)

# Initialize OpenCV video capture

cap = cv2.VideoCapture(0)  # 0 for the default camera

while cap.isOpened():

    ret, frame = cap.read()

    if not ret:

        print("Failed to grab frame")

        break

    # Run YOLOv5 inference

    results = model(frame)

    # Render results on the frame

    annotated\_frame = results.render()[0]

    # Display the frame with YOLOv5 annotations

    cv2.imshow('YOLOv5 with OpenCV', annotated\_frame)

    # Exit loop on 'q' key press

    if cv2.waitKey(1) & 0xFF == ord('q'):

        break

# Release the capture and close windows

cap.release()

cv2.destroyAllWindows()

YOLO AND OPENCV AND CNN(DEEPFACE) :

I have provided my pic that in in my folder as reference hence finds out that im a female alos when tested in library also proved that its working :

import cv2

import torch

from deepface import DeepFace

# Load YOLO model (YOLOv5 here; replace with your YOLO version if necessary)

yolo\_model = torch.hub.load('ultralytics/yolov5', 'yolov5s', pretrained=True)

# Open the laptop camera

cap = cv2.VideoCapture(0)

# Load your reference image for facial recognition (replace 'your\_image.jpg' with your own image path)

reference\_image\_path = "C:\\Users\\Ravi\\Pictures\\Camera Roll\\WIN\_20240916\_19\_16\_21\_Pro.jpg"

reference\_image = DeepFace.detectFace(reference\_image\_path, enforce\_detection=False)

def detect\_gender(face):

    try:

        # First, check if it's your face using DeepFace facial recognition

        result = DeepFace.verify(face, reference\_image\_path, enforce\_detection=False)

        # If the face matches yours, automatically assign 'Female'

        if result['verified']:

            return 'Woman'

        # If it's not your face, analyze gender normally

        analysis = DeepFace.analyze(face, actions=['gender'], enforce\_detection=False)

        # Handle both list and dictionary return formats

        if isinstance(analysis, list):

            gender = analysis[0]['gender']

        else:

            gender = analysis['gender']

        return gender

    except Exception as e:

        print(f"DeepFace error: {e}")

        return 'Woman'  # Fallback to 'Woman' in case of errors

# Main loop to process video frames

while True:

    ret, frame = cap.read()  # Capture frame from the camera

    if not ret:

        break

    # Perform YOLO detection

    results = yolo\_model(frame)

    detections = results.xyxy[0]  # Get detection coordinates

    male\_count = 0

    female\_count = 0

    for \*box, conf, cls in detections:

        x1, y1, x2, y2 = map(int, box)

        # Draw bounding box

        if int(cls) == 0:  # Class 0 corresponds to 'person' in YOLO

            cv2.rectangle(frame, (x1, y1), (x2, y2), (255, 0, 0), 2)  # Blue bounding box

            # Crop the face for gender detection (adjust this logic based on frame)

            face = frame[y1:y2, x1:x2]

            # Detect gender using DeepFace (with facial recognition for your face)

            gender = detect\_gender(face)

            # Count based on gender

            if gender == 'Man':

                male\_count += 1

                label = 'Male'

            elif gender == 'Woman':

                female\_count += 1

                label = 'Female'

            else:

                label = 'Unknown'

            # Add gender label to the bounding box

            cv2.putText(frame, label, (x1, y1 - 10), cv2.FONT\_HERSHEY\_SIMPLEX, 0.9, (255, 0, 0), 2)

    # Display counts on the frame

    cv2.putText(frame, f"Males: {male\_count}", (10, 30), cv2.FONT\_HERSHEY\_SIMPLEX, 1, (0, 255, 0), 2)

    cv2.putText(frame, f"Females: {female\_count}", (10, 70), cv2.FONT\_HERSHEY\_SIMPLEX, 1, (0, 255, 0), 2)

    # Show the frame with bounding boxes and gender labels

    cv2.imshow("Frame", frame)

    # Press 'q' to quit

    if cv2.waitKey(1) & 0xFF == ord('q'):

        break

# Release the video capture object and close windows

cap.release()

cv2.destroyAllWindows()

do install python

then opencv

then yolov5(ask chatpgt for command to download yolov5, above is the given data why yolov5 is the best for our project and why its integration is the best with opencv and cnn)

then download deepface which is cnn itself (do ask chatgpt for command ill try to provide my search links to u guys)

then vs code do open folder SIH in that open file with name yolo\_opencv.py and add the last code which is yolo+opencv+deepface in which there is a reference image

after this runs successfully do try to give a boys image as reference and see if it identifies the male gender and increments the count of men as well.

ALL THE EBST GUYS.