Exam Proctoring System

Online exams face significant challenges in maintaining academic integrity. Two complementary approaches—**Smart Exam Proctoring (Yolo and Haar Cascade based visual monitoring system)** and an **Audio-Visual Exam Proctoring System (hybrid real-time monitoring)**—provide robust solutions using OpenCV and Audio analytics.

● Yolo and Haar Cascade based visual monitoring system

- Google Colab's inbuilt functions for accessing webcam for Images and Video.
- OpenCV's Haar Cascade for face and eye detection.
- Ultralytics YoloV8 for objection detection (like phone, multiple persons, books).
- Warning texts at detections of multiple persons or phone.
- Displaying the detect/predicted image using the input captured image.

Link:
https://colab.research.google.com/drive/1RHZFIk9XWM3fHNzONVepMXr9MYuGufUG?usp=sharing

Stepwise description:

1. Install Ultralytics and import YOLO: Ultralytics creates cutting-edge, state-of-the-art (SOTA) YOLO models built on years of foundational research in computer vision and AI. Constantly updated for performance

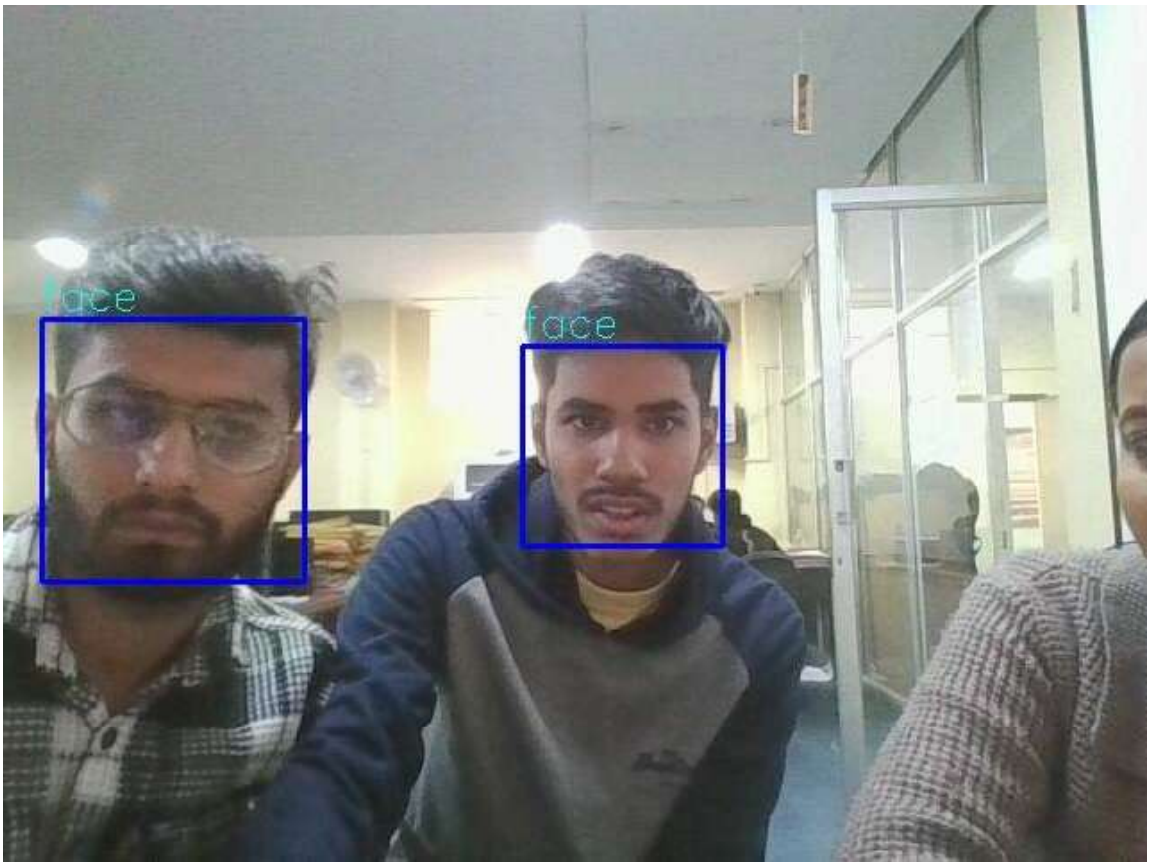
and flexibility, our models are fast, accurate, and easy to use. They excel at object detection, tracking, instance segmentation, image classification, and pose estimation tasks.

2. Model = 'yolov8n.pt' for object detection: !yolo task=detect mode=predict model=yolov8n.pt conf=0.25 source='photo.jpg'. YOLOv8 may be used directly in the Command Line Interface (CLI) with a yolo command for a variety of tasks

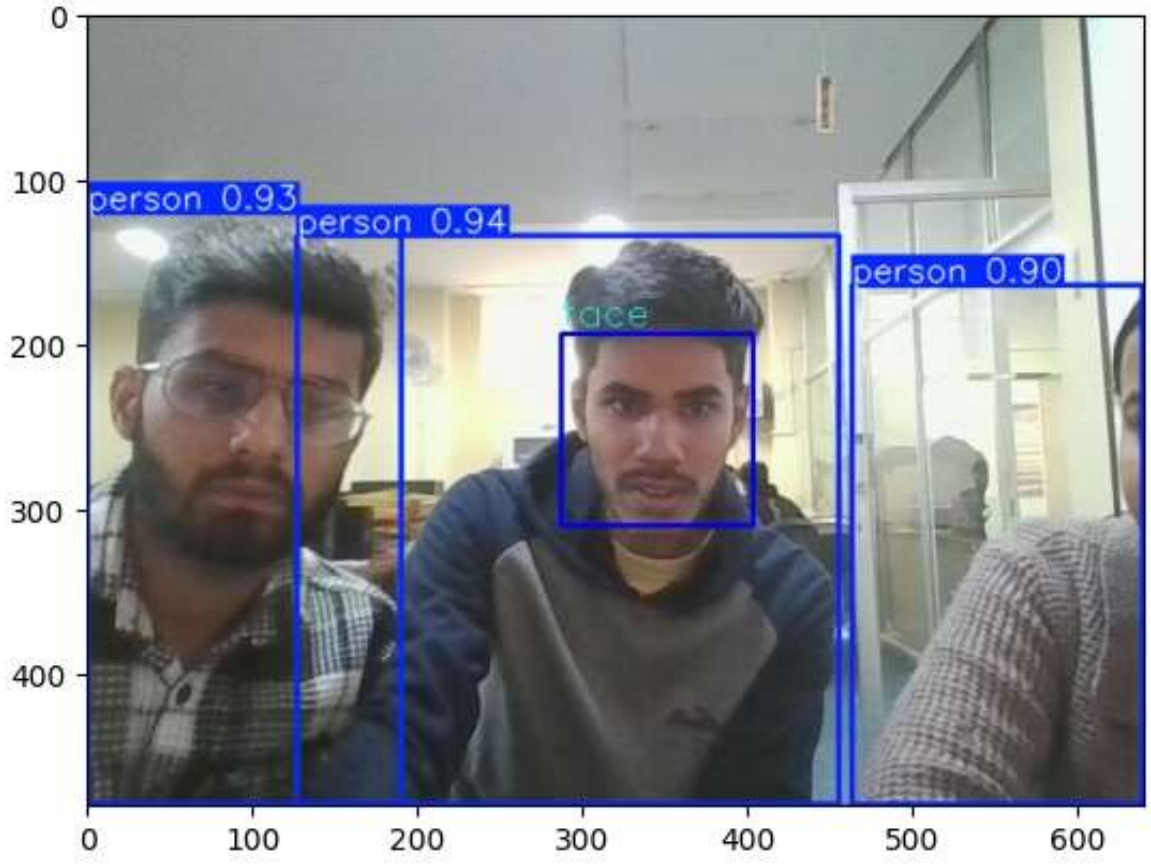
3. Google Colab: Access Webcam for Images and Video: We use the predefined functions by Colab for webcam access for Image and Video.
    - Function js_to_image(js_reply) to convert the JavaScript object into an OpenCV image
    - Function bbox_to_bytes(bbox_array) to convert OpenCV Rectangle bounding box image into base64 byte string to be overlayed on video stream
    - video_stream(): JavaScript to properly create our live video stream using our webcam as input
    - Function video_frame(label, bbox) to return data with label and bbox
    - Function take_photo(filename='photo.jpg', quality=0.8) to capture image from running video, apply Haar Cascade and save it to photo.jpg.

4. Haar Cascade Classifier: A simple object detection algorithm for face and eyes detection on images and video fetched from our webcam.
5. Main function implementation and logic:
    - Start streaming video from webcam and label 'Capturing…' for video
    - Initialize bounding box to empty
    - take_photo('photo.jpg') function to apply haar cascade to captured image and save it to photo.jpg.
    - Apply the YOLO command provided in point 2 on 'photo.jpg' for object detection.

- Convert the output of the YOLO command to string format and find keywords like ‘1 person’, ‘persons’, ‘cell phone’.
- If the string contains the keywords then output: 'One person in room', 'Warning: Two or more persons in room!', 'Warning: phone found!', respectively with blue or red(warning) color fonts.
- Show the predicted image.



HaarCascade Image with face detection.

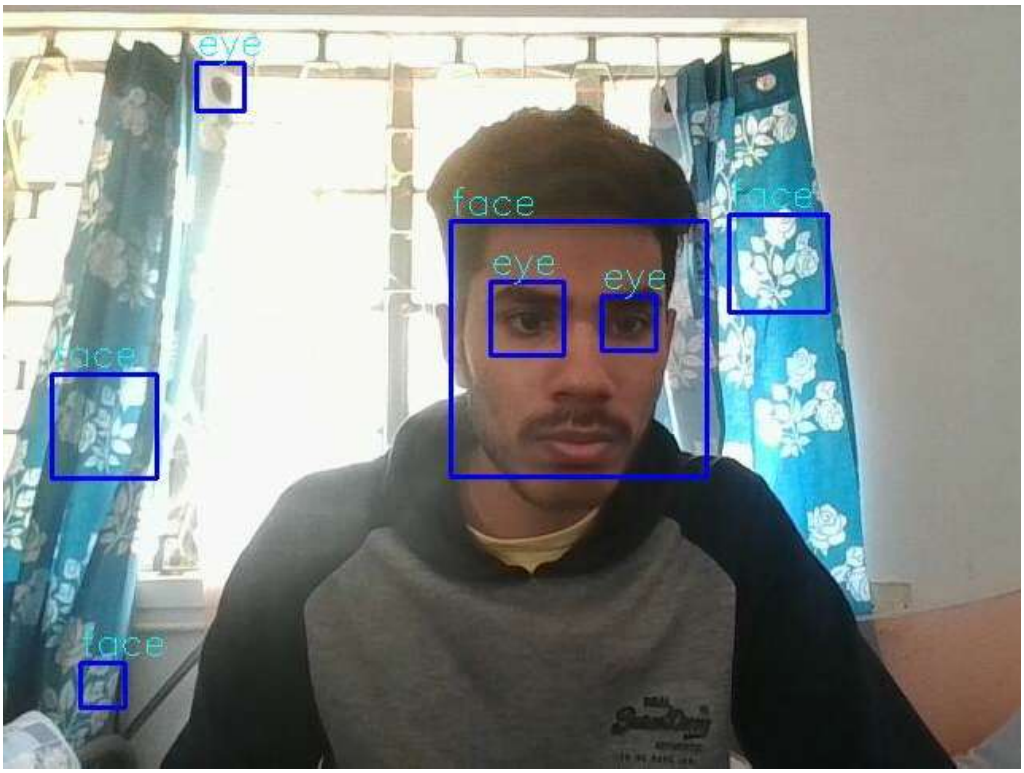


Final predicted image using [Yolo8n.pt] model.

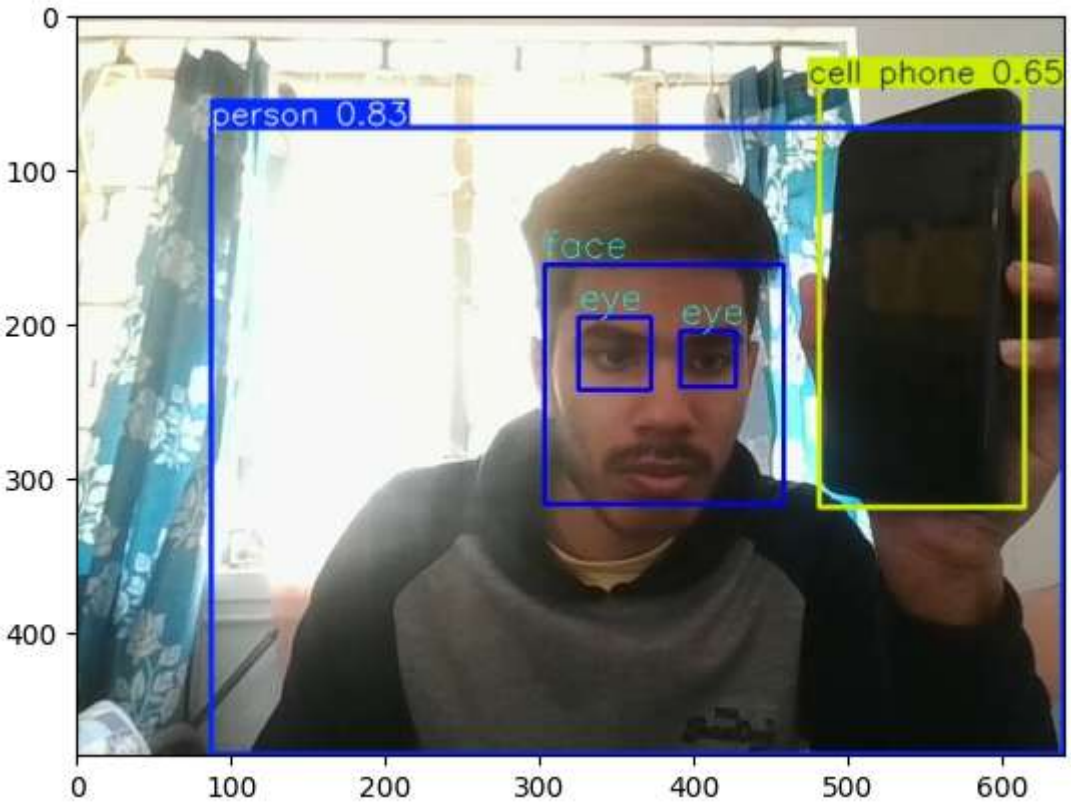
In the above image example, warning text: was generated as output because multiple persons were found in front of webcam, which is not fair in online examinations.

`Warning: Two or more persons in room!, Warning: phone found!` are some of the warning messages.

Example:



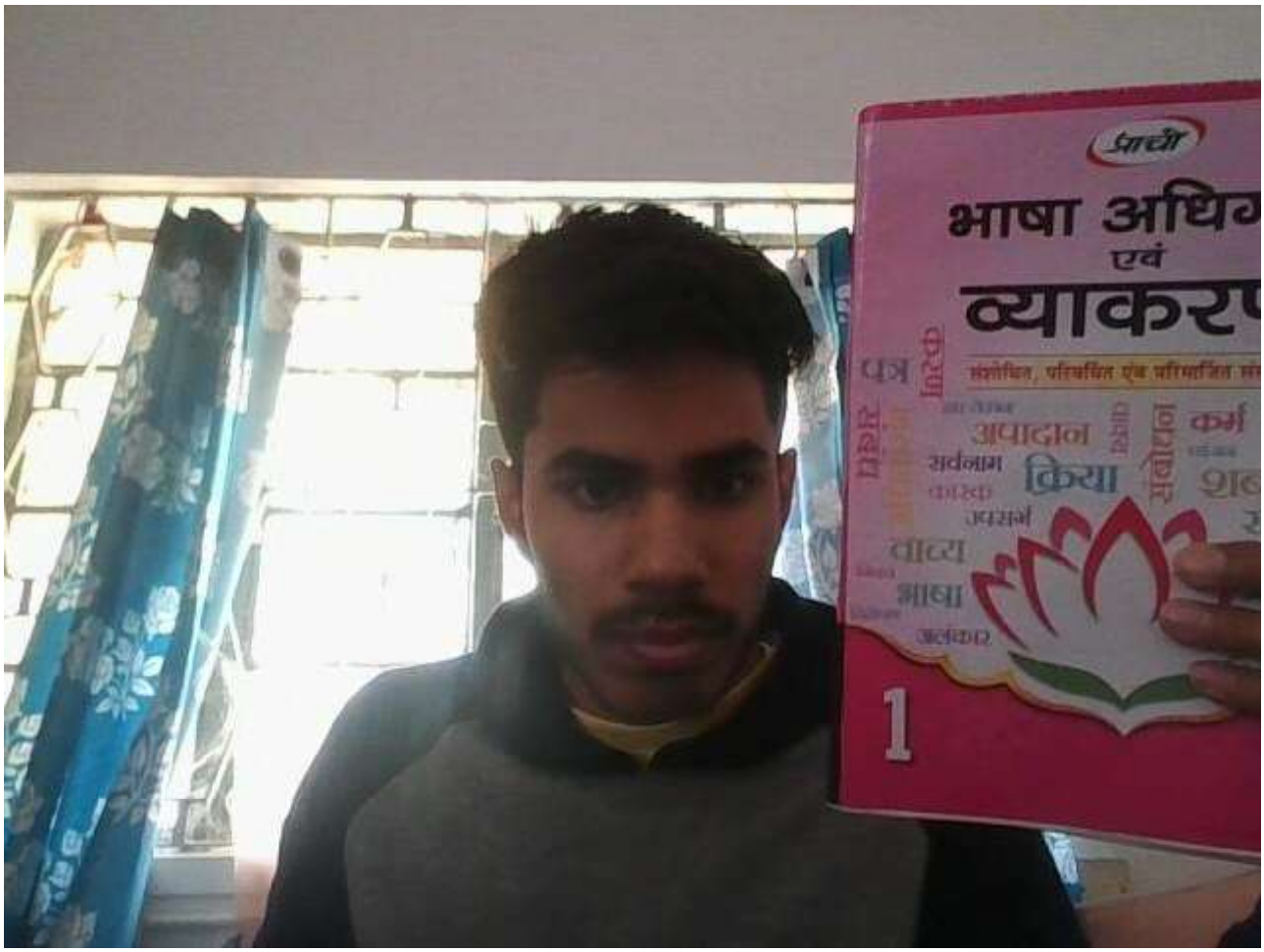
HaarCascade using both face and eyes detection may provide unnatural outputs sometimes!



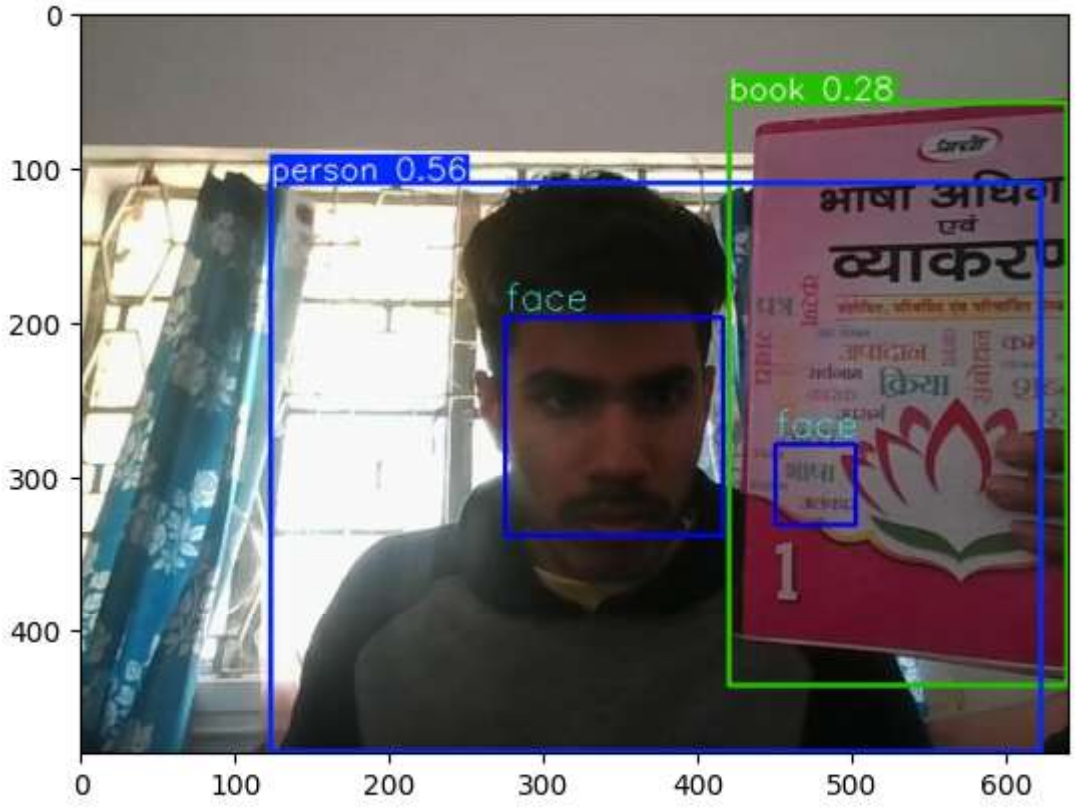
Predicted image using Haar Cascade and YOLO.

```
(480, 640)
Saved to photo.jpg
['Ultralytics 8.3.228 🚀 Python-3.12.12 torch-2.8.0+cu126 CUDA:0 (Tesla T4,
15095MiB)', 'YOLOv8n summary (fused): 72 layers, 3,151,904 parameters, 0
gradients, 8.7 GFLOPs', '', 'image 1/1 /content/photo.jpg: 480x640 1 person,
1 cell phone, 81.2ms', 'Speed: 3.1ms preprocess, 81.2ms inference, 3.7ms
postprocess per image at shape (1, 3, 480, 640)', 'Results saved to
\x1b[1m/content/runs/detect/predict4\x1b[0m', '💡 Learn more at
https://docs.ultralytics.com/modes/predict']
One person in room
Warning: phone found!
```

Example:



Input Image

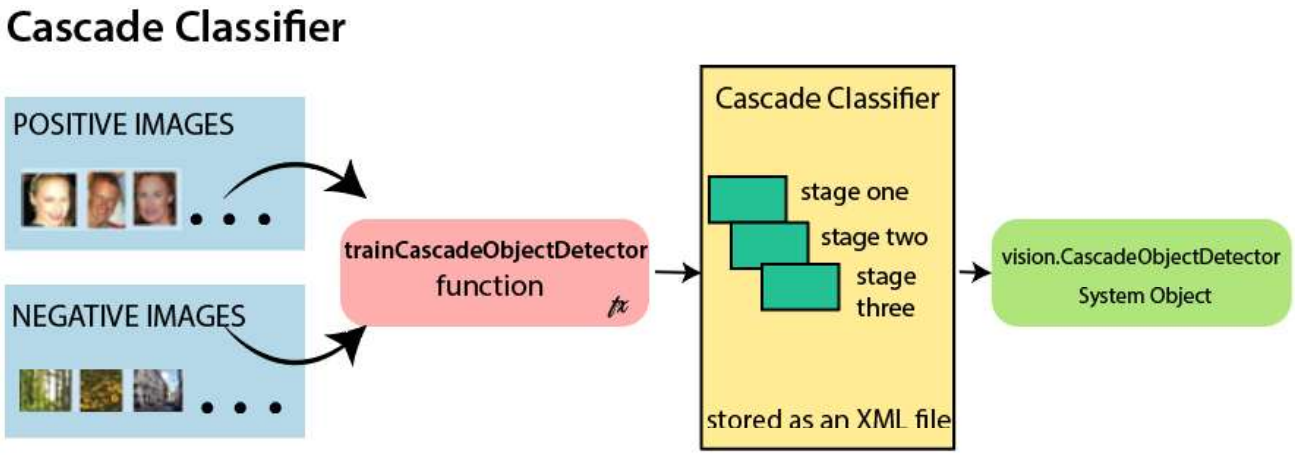


Predicted image using Haar Cascade and YOLO.

```
(480, 640)
Saved to photo.jpg
['Ultralytics 8.3.228 🚀 Python-3.12.12 torch-2.8.0+cu126 CUDA:0 (Tesla T4,
15095MiB)', 'YOLOv8n summary (fused): 72 layers, 3,151,904 parameters, 0
gradients, 8.7 GFLOPs', '', 'image 1/1 /content/photo.jpg: 480x640 1 person,
1 book, 50.7ms', 'Speed: 1.9ms preprocess, 50.7ms inference, 2.2ms
postprocess per image at shape (1, 3, 480, 640)', 'Results saved to
\x1b[1m/content/runs/detect/predict7\x1b[0m', '💡 Learn more at
https://docs.ultralytics.com/modes/predict']
One person in room
Warning: Study material Found!
```

Haar Cascade Classifier:

- A Haar cascade classifier is a machine learning-based object detection algorithm that uses a cascade of simple features to efficiently detect objects in images or videos.
- It is trained on positive (containing the object) and negative (not containing the object) images, and then uses a series of stages with learned features to quickly scan images and identify potential objects in real-time.
- Haar cascades are fast and can work in real-time, but they are less accurate than modern methods and can produce false positives.



YOLO Objection Detection:

- YOLO (You Only Look Once) works by dividing an image into a grid, and each grid cell is responsible for predicting bounding boxes and class probabilities for objects within its area.
- It uses a single convolutional neural network (CNN) to process the entire image in one pass, making it fast for real-time applications.
- The process includes extracting features, making bounding box and class predictions, and then using non-maximum suppression to filter out duplicate detections.