

Smart Exam proctoring 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 smartphone, multiple persons, books)
- Warning texts at detections of multiple persons or phone
- Displaying the detect/predicted image using the input captured image.

```
from google.colab import drive
drive.mount('/content/gdrive')
Mounted at /content/gdrive
ROOT_DIR = '/content/gdrive/My Drive/objDetection'
```

```
!pip install ultralytics
```

```
Collecting ultralytics
  Downloading ultralytics-8.3.228-py3-none-any.whl.metadata (37 kB)
Requirement already satisfied: numpy>=1.23.0 in /usr/local/lib/python3.12/dist-packages (from ultralytics) (2.0.2)
Requirement already satisfied: matplotlib>=3.3.0 in /usr/local/lib/python3.12/dist-packages (from ultralytics) (3.10.0)
Requirement already satisfied: opencv-python>=4.6.0 in /usr/local/lib/python3.12/dist-packages (from ultralytics) (4.12.0.88)
Requirement already satisfied: pillow>=7.1.2 in /usr/local/lib/python3.12/dist-packages (from ultralytics) (11.3.0)
Requirement already satisfied: pyyaml>=5.3.1 in /usr/local/lib/python3.12/dist-packages (from ultralytics) (6.0.3)
Requirement already satisfied: requests>=2.23.0 in /usr/local/lib/python3.12/dist-packages (from ultralytics) (2.32.4)
Requirement already satisfied: scipy>=1.4.1 in /usr/local/lib/python3.12/dist-packages (from ultralytics) (1.16.3)
Requirement already satisfied: torch>=1.8.0 in /usr/local/lib/python3.12/dist-packages (from ultralytics) (2.8.0+cu126)
Requirement already satisfied: torchvision>=0.9.0 in /usr/local/lib/python3.12/dist-packages (from ultralytics) (0.23.0+cu126)
Requirement already satisfied: psutil in /usr/local/lib/python3.12/dist-packages (from ultralytics) (5.9.5)
Requirement already satisfied: polars in /usr/local/lib/python3.12/dist-packages (from ultralytics) (1.31.0)
Collecting ultralytics-thop>=2.0.18 (from ultralytics)
  Downloading ultralytics_thop-2.0.18-py3-none-any.whl.metadata (14 kB)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib>=3.3.0->ultralytics)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.12/dist-packages (from matplotlib>=3.3.0->ultralytics)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.12/dist-packages (from matplotlib>=3.3.0->ultralytics)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib>=3.3.0->ultralytics)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.12/dist-packages (from matplotlib>=3.3.0->ultralytics)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib>=3.3.0->ultralytics)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.12/dist-packages (from matplotlib>=3.3.0->ultralytics)
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/lib/python3.12/dist-packages (from requests>=2.23.0->ultralytics)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.12/dist-packages (from requests>=2.23.0->ultralytics)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.12/dist-packages (from requests>=2.23.0->ultralytics)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.12/dist-packages (from requests>=2.23.0->ultralytics)
Requirement already satisfied: filelock in /usr/local/lib/python3.12/dist-packages (from torch>=1.8.0->ultralytics) (3.20.0)
Requirement already satisfied: typing_extensions>=4.10.0 in /usr/local/lib/python3.12/dist-packages (from torch>=1.8.0->ultralytics)
Requirement already satisfied: setuptools in /usr/local/lib/python3.12/dist-packages (from torch>=1.8.0->ultralytics) (75.2.0)
Requirement already satisfied: sympy>=1.13.3 in /usr/local/lib/python3.12/dist-packages (from torch>=1.8.0->ultralytics) (1.13.3)
Requirement already satisfied: networkx in /usr/local/lib/python3.12/dist-packages (from torch>=1.8.0->ultralytics) (3.5)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.12/dist-packages (from torch>=1.8.0->ultralytics) (3.1.6)
Requirement already satisfied: fsspec in /usr/local/lib/python3.12/dist-packages (from torch>=1.8.0->ultralytics) (2025.3.0)
Requirement already satisfied: nvidia-cuda-nvrtc-cu12==12.6.77 in /usr/local/lib/python3.12/dist-packages (from torch>=1.8.0)
Requirement already satisfied: nvidia-cuda-runtime-cu12==12.6.77 in /usr/local/lib/python3.12/dist-packages (from torch>=1.8.0)
Requirement already satisfied: nvidia-cuda-cupti-cu12==12.6.80 in /usr/local/lib/python3.12/dist-packages (from torch>=1.8.0)
Requirement already satisfied: nvidia-cudnn-cu12==9.10.2.21 in /usr/local/lib/python3.12/dist-packages (from torch>=1.8.0->ultralytics)
Requirement already satisfied: nvidia-cublas-cu12==12.6.4.1 in /usr/local/lib/python3.12/dist-packages (from torch>=1.8.0->ultralytics)
Requirement already satisfied: nvidia-cufft-cu12==11.3.0.4 in /usr/local/lib/python3.12/dist-packages (from torch>=1.8.0->ultralytics)
Requirement already satisfied: nvidia-curand-cu12==10.3.7.77 in /usr/local/lib/python3.12/dist-packages (from torch>=1.8.0->ultralytics)
Requirement already satisfied: nvidia-cusolver-cu12==11.7.1.2 in /usr/local/lib/python3.12/dist-packages (from torch>=1.8.0->ultralytics)
Requirement already satisfied: nvidia-cusparse-cu12==12.5.4.2 in /usr/local/lib/python3.12/dist-packages (from torch>=1.8.0->ultralytics)
Requirement already satisfied: nvidia-cusparseelt-cu12==0.7.1 in /usr/local/lib/python3.12/dist-packages (from torch>=1.8.0->ultralytics)
Requirement already satisfied: nvidia-nccl-cu12==2.27.3 in /usr/local/lib/python3.12/dist-packages (from torch>=1.8.0->ultralytics)
Requirement already satisfied: nvidia-nvtx-cu12==12.6.77 in /usr/local/lib/python3.12/dist-packages (from torch>=1.8.0->ultralytics)
Requirement already satisfied: nvidia-nvjitlink-cu12==12.6.85 in /usr/local/lib/python3.12/dist-packages (from torch>=1.8.0->ultralytics)
Requirement already satisfied: nvidia-cufile-cu12==1.11.1.6 in /usr/local/lib/python3.12/dist-packages (from torch>=1.8.0->ultralytics)
Requirement already satisfied: triton==3.4.0 in /usr/local/lib/python3.12/dist-packages (from torch>=1.8.0->ultralytics) (3.4.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12/dist-packages (from python-dateutil>=2.7->matplotlib>=3.3.0)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /usr/local/lib/python3.12/dist-packages (from sympy>=1.13.3->torch>=1.8.0)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.12/dist-packages (from jinja2->torch>=1.8.0->ultralytics)
Downloading ultralytics-8.3.228-py3-none-any.whl (1.1 MB)
  1.1/1.1 MB 31.3 MB/s eta 0:00:00
Downloading ultralytics_thop-2.0.18-py3-none-any.whl (28 kB)
Installing collected packages: ultralytics-thop, ultralytics
Successfully installed ultralytics-8.3.228 ultralytics-thop-2.0.18
```

```

import kagglehub

# Download latest version
path = kagglehub.dataset_download("ultralytics/coco128")

Downloading from https://www.kaggle.com/api/v1/datasets/download/ultralytics/coco128?dataset\_version\_number=3...
100%|██████████| 6.66M/6.66M [00:00<00:00, 107MB/s]Extracting files...

import os
from ultralytics import YOLO

# model = YOLO("yolov8n.yaml")

# results = model.train(data = os.path.join(ROOT_DIR, "google_colab_config.yaml"), epochs = 1)

Creating new Ultralytics Settings v0.0.6 file ✓
View Ultralytics Settings with 'yolo settings' or at '/root/.config/Ultralytics/settings.json'
Update Settings with 'yolo settings key=value', i.e. 'yolo settings runs_dir=path/to/dir'. For help see https://docs.ultraly

x = !yolo task=detect mode=predict model=yolov8n.pt conf=0.25 source='bus.jpg'
x
print(x)
print(type(x))

['Ultralytics 8.3.228 🚀 Python-3.12.12 torch-2.8.0+cu126 CUDA:0 (Tesla T4, 15095MiB)', 'YOLOv8n summary (fused): 72 layers
<class 'IPython.utils.text.SList'>

```

Google Colab: Access Webcam for Images and Video

This notebook will go through how to access and run code on images and video taken using your webcam.

For this purpose of this tutorial we will be using OpenCV's Haar Cascade to do face detection and eye detection on our Webcam image and video.

```

# import dependencies
from IPython.display import display, Javascript, Image
from google.colab.output import eval_js
from base64 import b64decode, b64encode
import cv2
import numpy as np
import PIL
import io
import html
import time

```

Helper Functions

Below are a few helper function to make converting between different image data types and formats.

```

# function to convert the JavaScript object into an OpenCV image
def js_to_image(js_reply):
    """
    Params:
        js_reply: JavaScript object containing image from webcam
    Returns:
        img: OpenCV BGR image
    """
    # decode base64 image
    image_bytes = b64decode(js_reply.split(',')[1])
    # convert bytes to numpy array
    jpg_as_np = np.frombuffer(image_bytes, dtype=np.uint8)
    # decode numpy array into OpenCV BGR image
    img = cv2.imdecode(jpg_as_np, flags=1)

    return img

# function to convert OpenCV Rectangle bounding box image into base64 byte string to be overlayed on video stream
def bbox_to_bytes(bbox_array):
    """
    Params:
        bbox_array: Numpy array (pixels) containing rectangle to overlay on video stream.
    Returns:
        bytes: Base64 image byte string
    """

```

```
# convert array into PIL image
bbox_PIL = PIL.Image.fromarray(bbox_array, 'RGBA')
iobuf = io.BytesIO()
# format bbox into png for return
bbox_PIL.save(iobuf, format='png')
# format return string
bbox_bytes = 'data:image/png;base64,{}'.format((str(b64encode(iobuf.getvalue())), 'utf-8')))

return bbox_bytes
```

```
# JavaScript to properly create our live video stream using our webcam as input
def video_stream():
    js = Javascript('''
        var video;
        var div = null;
        var stream;
        var captureCanvas;
        var imgElement;
        var labelElement;

        var pendingResolve = null;
        var shutdown = false;

        function removeDom() {
            stream.getVideoTracks()[0].stop();
            video.remove();
            div.remove();
            video = null;
            div = null;
            stream = null;
            imgElement = null;
            captureCanvas = null;
            labelElement = null;
        }

        function onAnimationFrame() {
            if (!shutdown) {
                window.requestAnimationFrame(onAnimationFrame);
            }
            if (pendingResolve) {
                var result = "";
                if (!shutdown) {
                    captureCanvas.getContext('2d').drawImage(video, 0, 0, 640, 480);
                    result = captureCanvas.toDataURL('image/jpeg', 0.8)
                }
                var lp = pendingResolve;
                pendingResolve = null;
                lp(result);
            }
        }

        async function createDom() {
            if (div !== null) {
                return stream;
            }

            div = document.createElement('div');
            div.style.border = '2px solid black';
            div.style.padding = '3px';
            div.style.width = '100%';
            div.style.maxWidth = '600px';
            document.body.appendChild(div);

            const modelOut = document.createElement('div');
            modelOut.innerHTML = "<span>Status:</span>";
            labelElement = document.createElement('span');
            labelElement.innerText = 'No data';
            labelElement.style.fontWeight = 'bold';
            modelOut.appendChild(labelElement);
            div.appendChild(modelOut);

            video = document.createElement('video');
            video.style.display = 'block';
            video.width = div.clientWidth - 6;
            video.setAttribute('playsinline', '');
            video.onclick = () => { shutdown = true; };
            stream = await navigator.mediaDevices.getUserMedia(
                {video: { facingMode: "environment"}});
            div.appendChild(video);

            imgElement = document.createElement('img');
            imgElement.style.position = 'absolute';
        }
    ''')
    display(Javascript(js))
    display.HTML("""
        <div>
            <span>Status:</span>
            <img alt="Live video feed" style="width: 100%; height: auto; position: absolute; left: 0; top: 0; object-fit: cover;">
        </div>
    """)

```

```

imgElement.style.zIndex = 1;
imgElement.onclick = () => { shutdown = true; };
div.appendChild(imgElement);

const instruction = document.createElement('div');
instruction.innerHTML =
    '<span style="color: red; font-weight: bold;">' +
    'When finished, click here or on the video to stop this demo</span>';
div.appendChild(instruction);
instruction.onclick = () => { shutdown = true; };

video.srcObject = stream;
await video.play();

captureCanvas = document.createElement('canvas');
captureCanvas.width = 640; //video.videoWidth;
captureCanvas.height = 480; //video.videoHeight;
window.requestAnimationFrame(onAnimationFrame);

return stream;
}
async function stream_frame(label, imgData) {
if (shutdown) {
    removeDom();
    shutdown = false;
    return '';
}
var preCreate = Date.now();
stream = await createDom();

var preShow = Date.now();
if (label != "") {
    labelElement.innerHTML = label;
}

if (imgData != "") {
    var videoRect = video.getClientRects()[0];
    imgElement.style.top = videoRect.top + "px";
    imgElement.style.left = videoRect.left + "px";
    imgElement.style.width = videoRect.width + "px";
    imgElement.style.height = videoRect.height + "px";
    imgElement.src = imgData;
}

var preCapture = Date.now();
var result = await new Promise(function(resolve, reject) {
    pendingResolve = resolve;
});
shutdown = false;

return {'create': preShow - preCreate,
        'show': preCapture - preShow,
        'capture': Date.now() - preCapture,
        'img': result};
}
''')

display(js)

def video_frame(label, bbox):
    data = eval_js('stream_frame("{}", "{}")'.format(label, bbox))
    return data

```

Haar Cascade Classifier

For this tutorial we will run a simple object detection algorithm called Haar Cascade on our images and video fetched from our webcam. OpenCV has a pre-trained Haar Cascade face detection model.

```

# initialize the Haar Cascade face detection model
face_cascade = cv2.CascadeClassifier(cv2.samples.findFile(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml'))
eye_cascade = cv2.CascadeClassifier(cv2.samples.findFile(cv2.data.haarcascades + 'haarcascade_eye.xml'))
# smile_cascade = cv2.CascadeClassifier(cv2.samples.findFile(cv2.data.haarcascades + 'haarcascade_smile.xml'))

def take_photo(filename='photo.jpg', quality=0.8):
    js = Javascript('''
        async function takePhoto(quality) {
            const div = document.createElement('div');
            const capture = document.createElement('button');

```

```

capture.textContent = 'Capture';
div.appendChild(capture);

const video = document.createElement('video');
video.style.display = 'block';
const stream = await navigator.mediaDevices.getUserMedia({video: true});

document.body.appendChild(div);
div.appendChild(video);
video.srcObject = stream;
await video.play();

// Resize the output to fit the video element.
google.colab.output.setIframeHeight(document.documentElement.scrollHeight, true);

// Wait for Capture to be clicked.
await new Promise((resolve) => capture.onclick = resolve);

const canvas = document.createElement('canvas');
canvas.width = video.videoWidth;
canvas.height = video.videoHeight;
//canvas.setAttribute = 'person'
canvas.getContext('2d').drawImage(video, 0, 0);
stream.getVideoTracks()[0].stop();
div.remove();
return canvas.toDataURL('image/jpeg', quality);
}
...)
display(js)

# get photo data
data = eval_js('takePhoto({})'.format(quality))
# get OpenCV format image
img = js_to_image(data)
# grayscale img
gray = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)
print(gray.shape)
# get face bounding box coordinates using Haar Cascade
faces = face_cascade.detectMultiScale(gray)
# eyes = eye_cascade.detectMultiScale(gray)
# smile = smile_cascade.detectMultiScale(gray)
# draw face bounding box on image
for (x,y,w,h) in faces:
    img = cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
    cv2.putText(img, 'face', (x,y-5), cv2.FONT_HERSHEY_SIMPLEX , 0.8, (255,255,0), 1)

# for (x,y,w,h) in eyes:
#     img = cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
#     cv2.putText(img, 'eye', (x,y-5), cv2.FONT_HERSHEY_SIMPLEX , 0.8, (255,255,0), 1)

# save image
cv2.imwrite(filename, img)

return filename

```

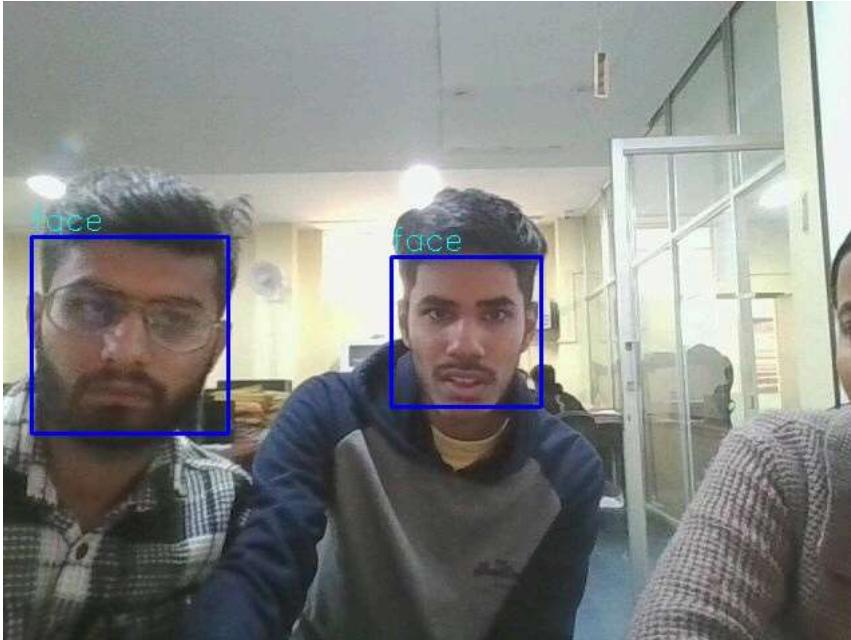
```

try:
    filename = take_photo('photo.jpg')
    print('Saved to {}'.format(filename))

    # Show the image which was just taken.
    display(Image(filename))
except Exception as err:
    # Errors will be thrown if the user does not have a webcam or if they do not
    # grant the page permission to access it.
    print(str(err))

```

(480, 640)
Saved to photo.jpg



```
# Source - https://stackoverflow.com/questions/287871/how-do-i-print-colored-text-to-the-terminal
# Posted by joeld
# Retrieved 11/5/2025, License - CC-BY-SA 4.0

class bcolors:
    OKBLUE = '\033[94m'
    WARNING = '\033[93m'
    FAIL = '\033[91m'
    BOLD = '\033[1m'
    UNDERLINE = '\033[4m'

print(bcolors.FAIL + "Text in red!")
```

Text in red!

Double-click (or enter) to edit

```
import time
# start streaming video from webcam
video_stream()
# label for video
label_html = 'Capturing...'
# initialize bounding box to empty
bbox = ''
count = 0
while True:
    js_reply = video_frame(label_html, bbox)
    if not js_reply:
        break

    img = js_to_image(js_reply["img"])
    filename = take_photo('photo.jpg')
    print('Saved to {}'.format(filename))
    #!yolo task=detect mode=predict model=yolov8n.pt conf=0.25 source='photo.jpg'
    x = !yolo task=detect mode=predict model=yolov8n.pt conf=0.25 source='photo.jpg'
    #print(x)
    s = str(x)
    print(s)
    word1 = '1 person'
    word2 = 'persons'
    word3 = 'cell phone'
    if word1 in s:
        print(bcolors.OKBLUE + 'One person in room')
    elif word2 in s:
        print(bcolors.FAIL + 'Warning: Two or more persons in room!')
    if word3 in s:
        print(bcolors.FAIL + 'Warning: phone found!')
    #print(type(s))
    #display(Image('/content/runs/detect/predict35/photo.jpg'))
    time.sleep(2)
```

```
(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
Warning: Two or more persons in room!']
```

```
idx = s.find('predict')
idx

from skimage import io
import matplotlib.pyplot as plt

# reading the sample image from a url
location = '/content/runs/detect/predict' + s[idx+7: idx+8] + '/photo.jpg'
print(location)
image = io.imread(location)
plt.imshow(image)
plt.show()

# image2 = cv2.imread(location)
# plt.imshow(image2)
# plt.show()
```

/content/runs/detect/predict4/photo.jpg

