EXPERIMENT 8

Face and Object Detection

Aim:

To perform Face and Object Detection using Haar Cascade and Object Detection using YOLO V5 Deep Learning Library.

Software/ Packages Used:

1. Google Colaboratory
2. Libraries used:
    - Opencv – python
    - Numpy
    - Matplotlib
    - tensorflow

Programs:

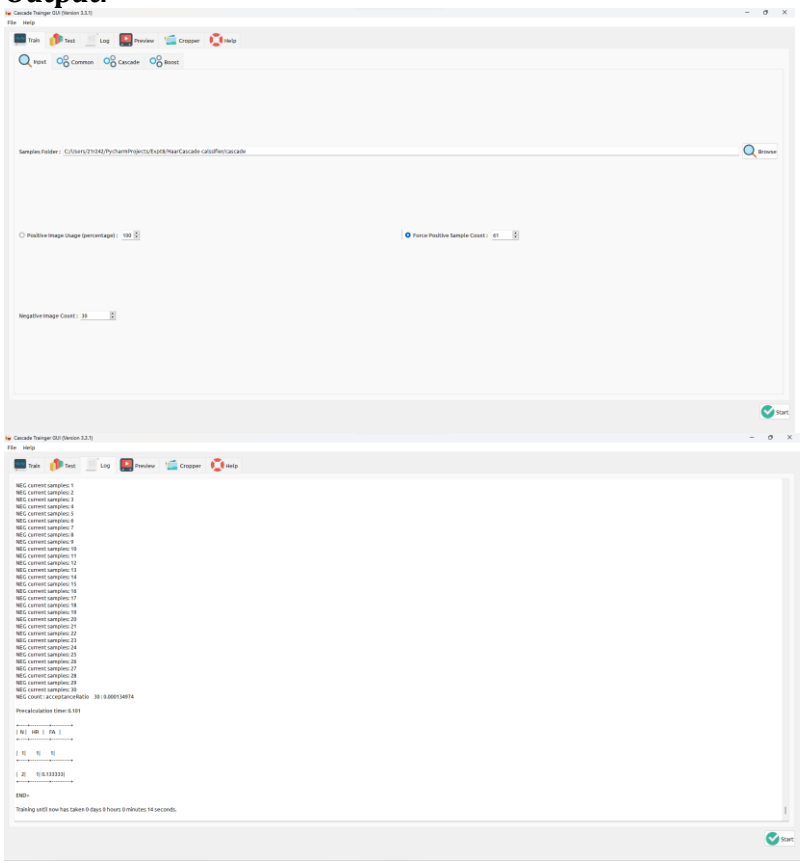
Haar Cascade Based Object Detection:

```
import cv2
import os
import time
#myPath = 'data/images'  # Rasbperry Pi:  '/home/pi/Desktop/data/images'
myPath = r'C:\Users\21r242\PycharmProjects\Expt8\image'
#cameraNo = 0
cameraBrightness = 180
moduleVal = 10  # SAVE EVERY iTH FRAME TO AVOID REPETITION
minBlur = 500  # SMALLER VALUE MEANS MORE BLURRINESS PRESENT
grayImage = False  # IMAGES SAVED COLORED OR GRAY
saveData = True  # SAVE DATA FLAG
showImage = True  # IMAGE DISPLAY FLAG
imgWidth = 180
imgHeight = 120
global countFolder
cap = cv2.VideoCapture(0,cv2.CAP_DSHOW)
cap.set(3, 640)
cap.set(4, 480)
cap.set(10, cameraBrightness)
count = 0
countSave = 0
def saveDataFunc():
   global countFolder
countFolder = 1
while os.path.exists(myPath + str(countFolder)):
   countFolder += 1
os.makedirs(myPath + str(countFolder))
if saveData: saveDataFunc()
```

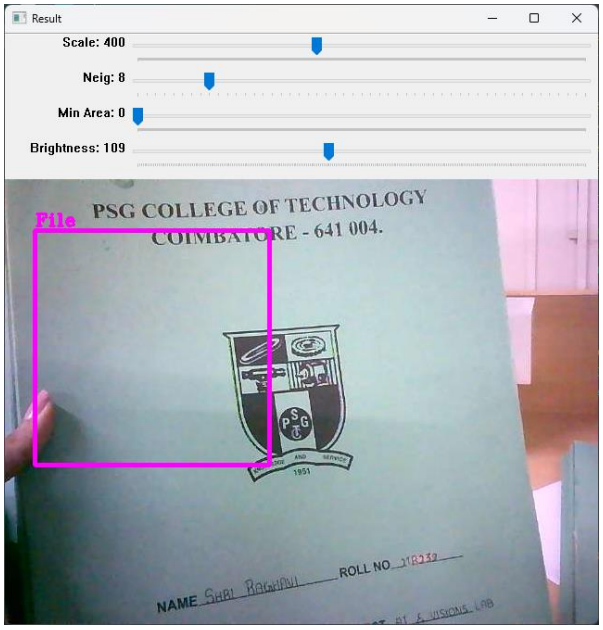
```
while True:
    success, img = cap.read()
    img = cv2.resize(img, (imgWidth, imgHeight))
    if grayImage: img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    if saveData:
        blur = cv2.Laplacian(img, cv2.CV_64F).var()
    if count % moduleVal == 0 and blur > minBlur:
        nowTime = time.time()
        cv2.imwrite(myPath + str(countFolder) +
            '/' + str(countSave) + "_" + str(int(blur)) + "_" + str(nowTime) + ".png", img)
        countSave += 1
    count += 1
    if showImage:
        cv2.imshow("Image", img)
#if cv2.waitKey(1) & amp; 0xFF == ord('q'):
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
cap.release()
cv2.destroyAllWindows()
import cv2
#path = 'Resources/haarcascades/haarcascade_frontalface_default.xml'  # PATH OF THE
CASCADE
path = r'C:\Users\21r242\PycharmProjects\Expt8\HaarCascade
calssifier\cascade\classifier\cascade.xml'
cameraNo = 0  # CAMERA NUMBER
#objectName = 'Arduino'  # OBJECT NAME TO DISPLAY
objectName = 'File'
frameWidth = 640  # DISPLAY WIDTH
frameHeight = 480  # DISPLAY HEIGHT
color = (255, 0, 255)
cap = cv2.VideoCapture(cameraNo)
cap.set(3, frameWidth)
cap.set(4, frameHeight)
def empty(a):
    pass
# CREATE TRACKBAR
cv2.namedWindow("Result")
cv2.resizeWindow("Result", frameWidth, frameHeight + 100)
cv2.createTrackbar("Scale", "Result", 400, 1000, empty)
cv2.createTrackbar("Neig", "Result", 8, 50, empty)
cv2.createTrackbar("Min Area", "Result", 0, 100000, empty)
cv2.createTrackbar("Brightness", "Result", 180, 255, empty)
# LOAD THE CLASSIFIERS DOWNLOADED
cascade = cv2.CascadeClassifier(path)
while True:
    # SET CAMERA BRIGHTNESS FROM TRACKBAR VALUE
    cameraBrightness = cv2.getTrackbarPos("Brightness", "Result")
    cap.set(10, cameraBrightness)
    # GET CAMERA IMAGE AND CONVERT TO GRAYSCALE
    success, img = cap.read()
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    # DETECT THE OBJECT USING THE CASCADE
    scaleVal = 1 + (cv2.getTrackbarPos("Scale", "Result") / 1000)
    neig = cv2.getTrackbarPos("Neig", "Result")
    objects = cascade.detectMultiScale(gray, scaleVal, neig)
    # DISPLAY THE DETECTED OBJECTS
    for (x, y, w, h) in objects:
        area = w * h
```

```
        minArea = cv2.getTrackbarPos("Min Area", "Result")
        if area > minArea:
            cv2.rectangle(img, (x, y), (x + w, y + h), color, 3)
            cv2.putText(img, objectName, (x, y - 5), cv2.FONT_HERSHEY_COMPLEX_SMALL, 1, color, 2)
            roi_color = img[y:y + h, x:x + w]

    cv2.imshow("Result", img)
    if cv2.waitKey(1) & 0xFF == ord('q'):
    # if cv2.waitKey(1) & amp: 0xFF == ord('q'):
        Break
```

Output:



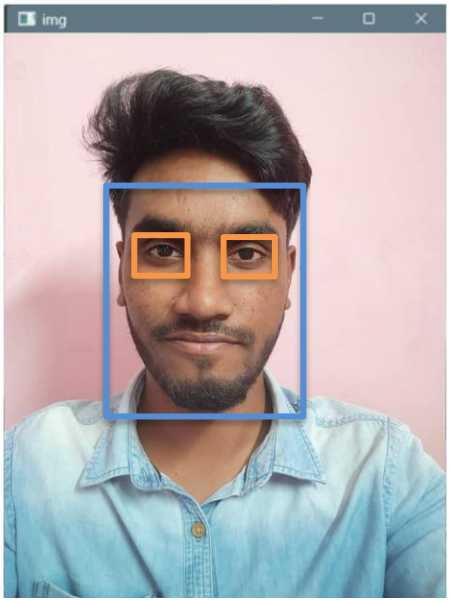
OUTPUT



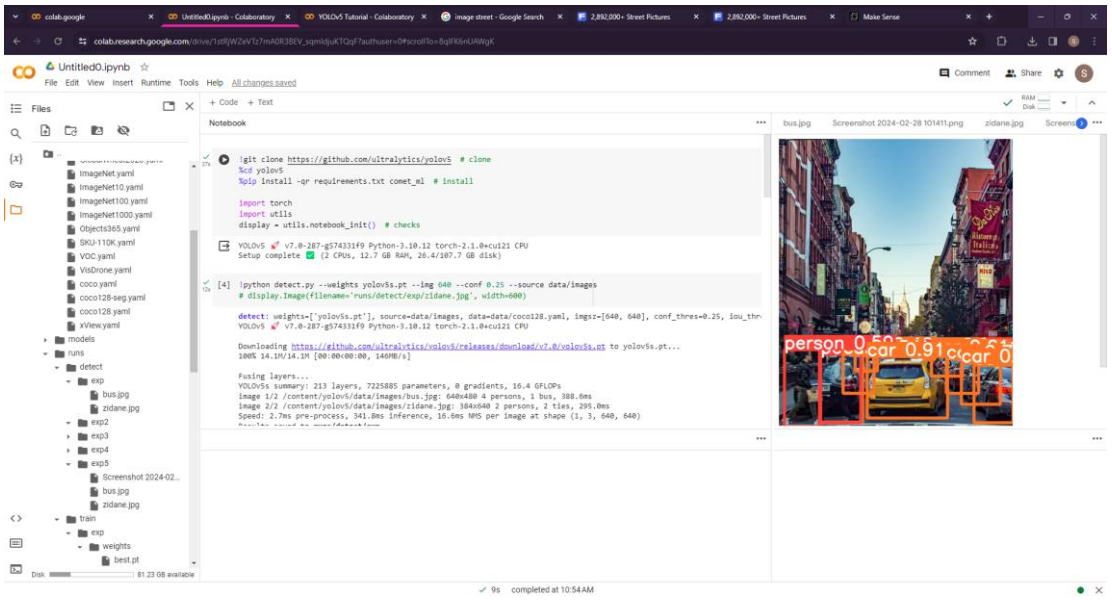
HAAR CLASSIFIER

```
import cv2

face_classifier = cv2.CascadeClassifier(cv2.data.haarcascades +
"haarcascade_frontalface_default.xml")
eye_classifier = cv2.CascadeClassifier(cv2.data.haarcascades + "haarcascade_eye.xml")
# capture frames from a camera
cap = cv2.VideoCapture(0,cv2.CAP_DSHOW)

# loop runs if capturing has been initialized.
while 1:

    # reads frames from a camera
    ret, img = cap.read()

    # convert to gray scale of each frames
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    # Detects faces of different sizes in the input image
    faces = face_classifier.detectMultiScale(gray, 1.3, 5)

    for (x, y, w, h) in faces:
        # To draw a rectangle in a face
        cv2.rectangle(img, (x, y), (x + w, y + h), (255, 255, 0), 2)
        roi_gray = gray[y:y + h, x:x + w]
        roi_color = img[y:y + h, x:x + w]

        # Detects eyes of different sizes in the input image
        eyes = eye_classifier.detectMultiScale(roi_gray)

        # To draw a rectangle in eyes
        for (ex, ey, ew, eh) in eyes:
            cv2.rectangle(roi_color, (ex, ey), (ex + ew, ey + eh), (0, 127, 255), 2)


 # Display an image in a window
    cv2.imshow('img', img)

    # Wait for Esc key to stop
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# Close the window
cap.release()
# De-allocate any associated memory usage
cv2.destroyAllWindows()
```

Output Image:



YOLO v5 Based Object Detection: (both for pretrained and custom data- (i/p - Video, Image, Live Video))




```
[1]  !git clone https://github.com/ultralytics/yolov5  # clone
     %cd yolov5
     %pip install -qr requirements.txt comet_ml  # install

     import torch
     import utils
     display = utils.notebook_init()  # checks
```

```
YOLOv5 🚀 v7.0-287-g574331f9 Python-3.10.12 torch-2.1.0+cu121 CPU
Setup complete ✅ (2 CPUs, 12.7 GB RAM, 26.4/107.7 GB disk)
```

```
# Train YOLOv5s on COCO128 for 3 epochs
!python train.py --img 640 --batch 2 --epochs 3 --data coco128.yaml --weights yolov5s.pt --cache
```

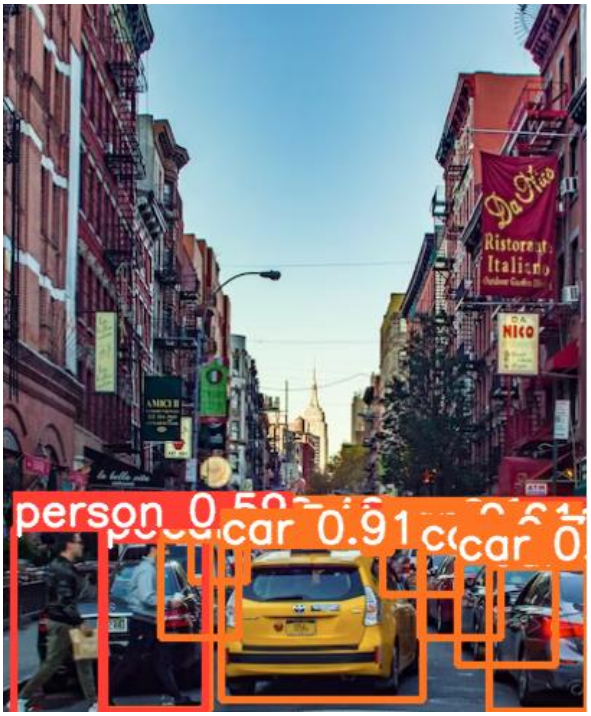
```
2024-02-28 04:24:19.014779: E external/local_xla/xla/stream_executor/cuda/cuda_dnn.cc:9261] Unable to register cuDNN
2024-02-28 04:24:19.014844: E external/local_xla/xla/stream_executor/cuda/cuda_fft.cc:607] Unable to register cuFFT
2024-02-28 04:24:19.016981: E external/local_xla/xla/stream_executor/cuda/cuda_blas.cc:1515] Unable to register cuBL
train: weights=yolov5s.pt, cfg=, data=coco128.yaml, hyp=data/hyps/hyp.scratch-low.yaml, epochs=3, batch_size=2, imgs
github: up to date with https://github.com/ultralytics/yolov5 ✅
YOLOv5 🚀 v7.0-287-g574331f9 Python-3.10.12 torch-2.1.0+cu121 CPU

hyperparameters: lr0=0.01, lrf=0.01, momentum=0.937, weight_decay=0.0005, warmup_epochs=3.0, warmup_momentum=0.8, wa
TensorBoard: Start with 'tensorboard --logdir runs/train', view at http://localhost:6006/
COMET WARNING: Comet credentials have not been set. Comet will default to offline logging. Please set your credentia
COMET INFO: Using '/content/yolov5/.cometml-runs' path as offline directory. Pass 'offline_directory' parameter into

Dataset not found ⚠️, missing paths ['/content/datasets/coco128/images/train2017']
Downloading https://ultralytics.com/assets/coco128.zip to coco128.zip...
100%|██████████| 6.66M/6.66M [00:00<00:00, 121MB/s]
Dataset download success ✅ (1.2s), saved to /content/datasets

                 from  n    params  module                                  arguments
  0                -1  1      3520  models.common.Conv                      [3, 32, 6, 2, 2]
  1                -1  1     18560  models.common.Conv                      [32, 64, 3, 2]
  2                -1  1     18816  models.common.C3                        [64, 64, 1]
  3                -1  1     73984  models.common.Conv                      [64, 128, 3, 2]
```

```
[4] !python detect.py --weights yolov5s.pt --img 640 --conf 0.25 --source data/images
    # display.Image(filename='runs/detect/exp/zidane.jpg', width=600)
```

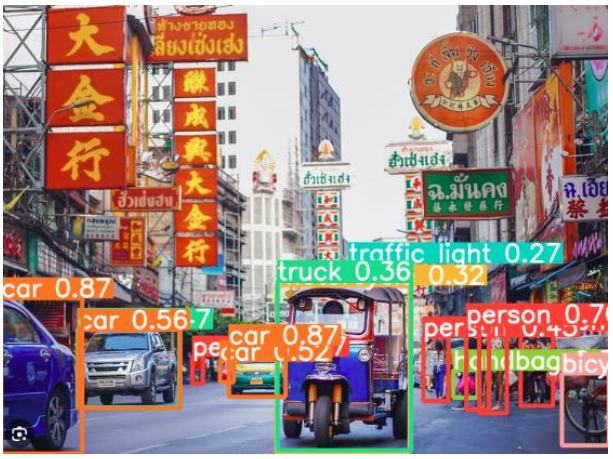
```
detect: weights=['yolov5s.pt'], source=data/images, data=data/coco128.yaml, imgsz=[640, 640], conf_thres=0.25, iou_thr
YOLOv5 🚀 v7.0-287-g574331f9 Python-3.10.12 torch-2.1.0+cu121 CPU

Downloading https://github.com/ultralytics/yolov5/releases/download/v7.0/yolov5s.pt to yolov5s.pt...
100% 14.1M/14.1M [00:00<00:00, 146MB/s]

Fusing layers...
YOLOv5s summary: 213 layers, 7225885 parameters, 0 gradients, 16.4 GFLOPs
image 1/2 /content/yolov5/data/images/bus.jpg: 640x480 4 persons, 1 bus, 388.6ms
image 2/2 /content/yolov5/data/images/zidane.jpg: 384x640 2 persons, 2 ties, 295.0ms
Speed: 2.7ms pre-process, 341.8ms inference, 16.6ms NMS per image at shape (1, 3, 640, 640)
Results saved to runs/detect/exp
```

Input Image

Output Image

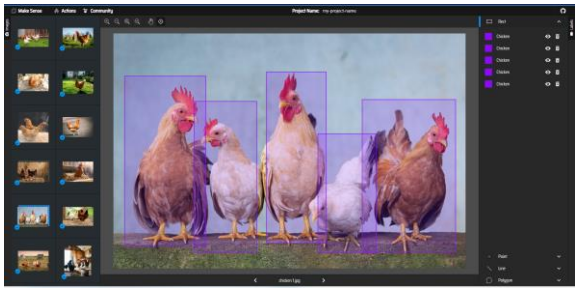


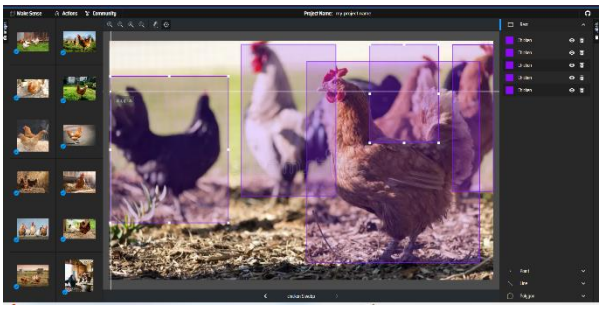
Input Image **Output Image**



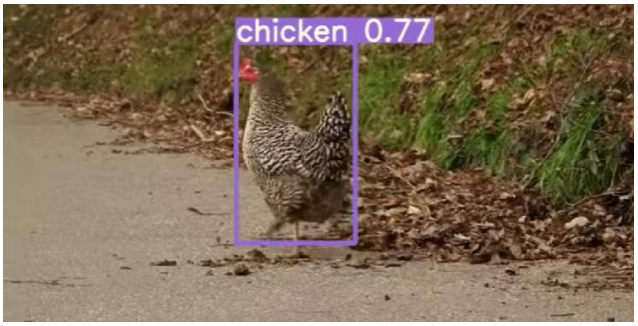
Live Video:

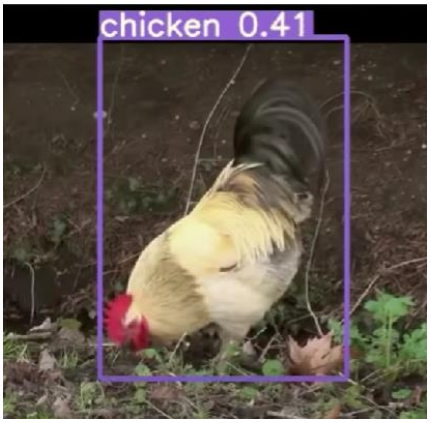
- Data set is collected and annotated using MakeSense Ai

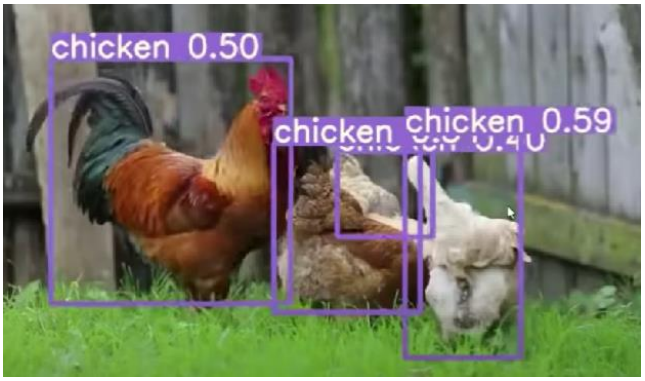




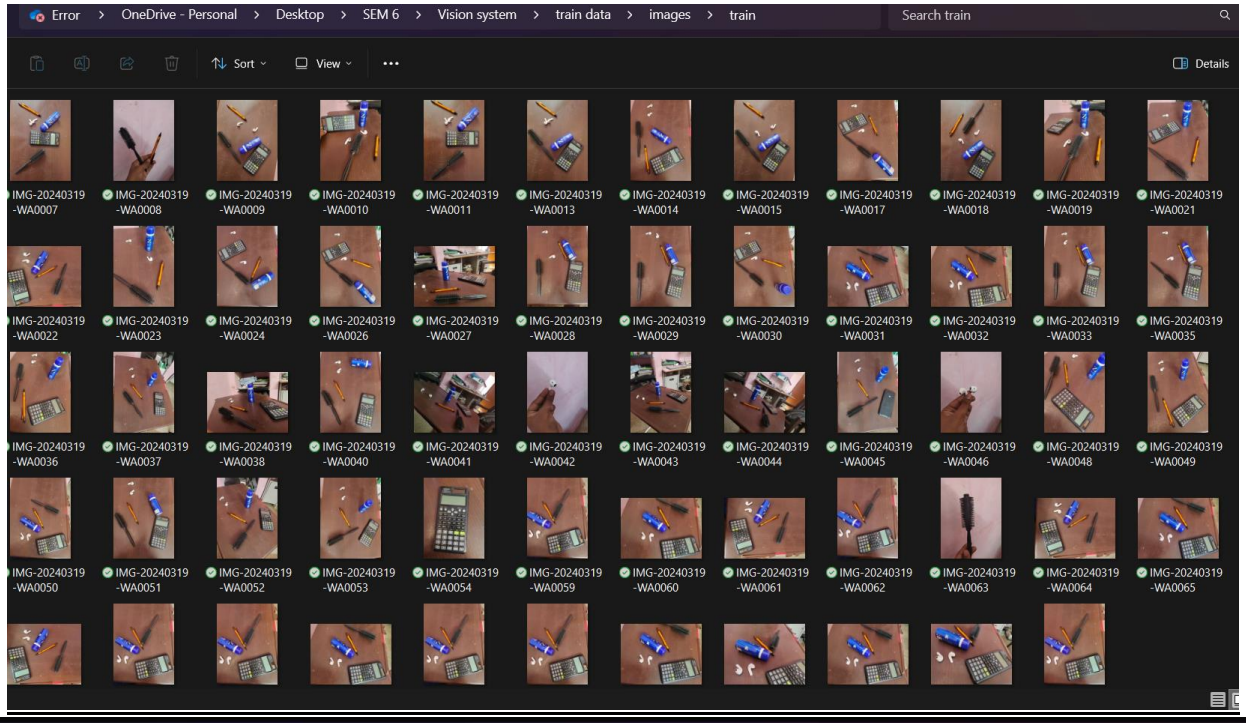
Output:

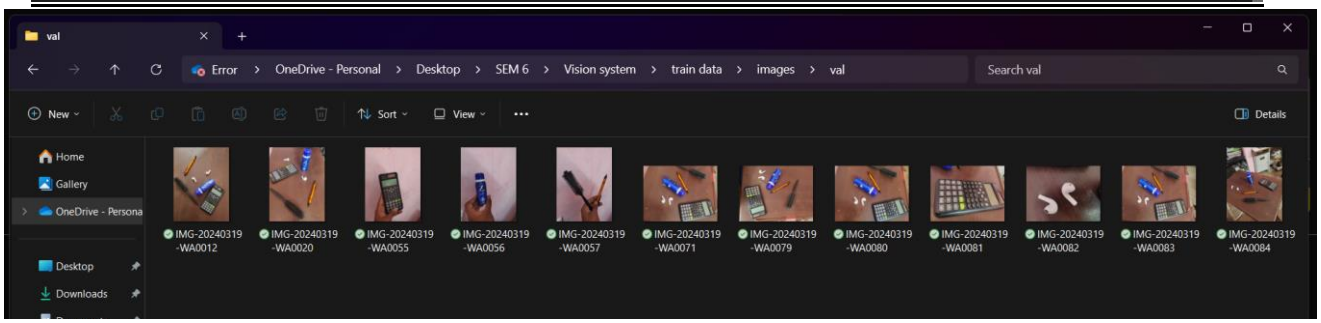


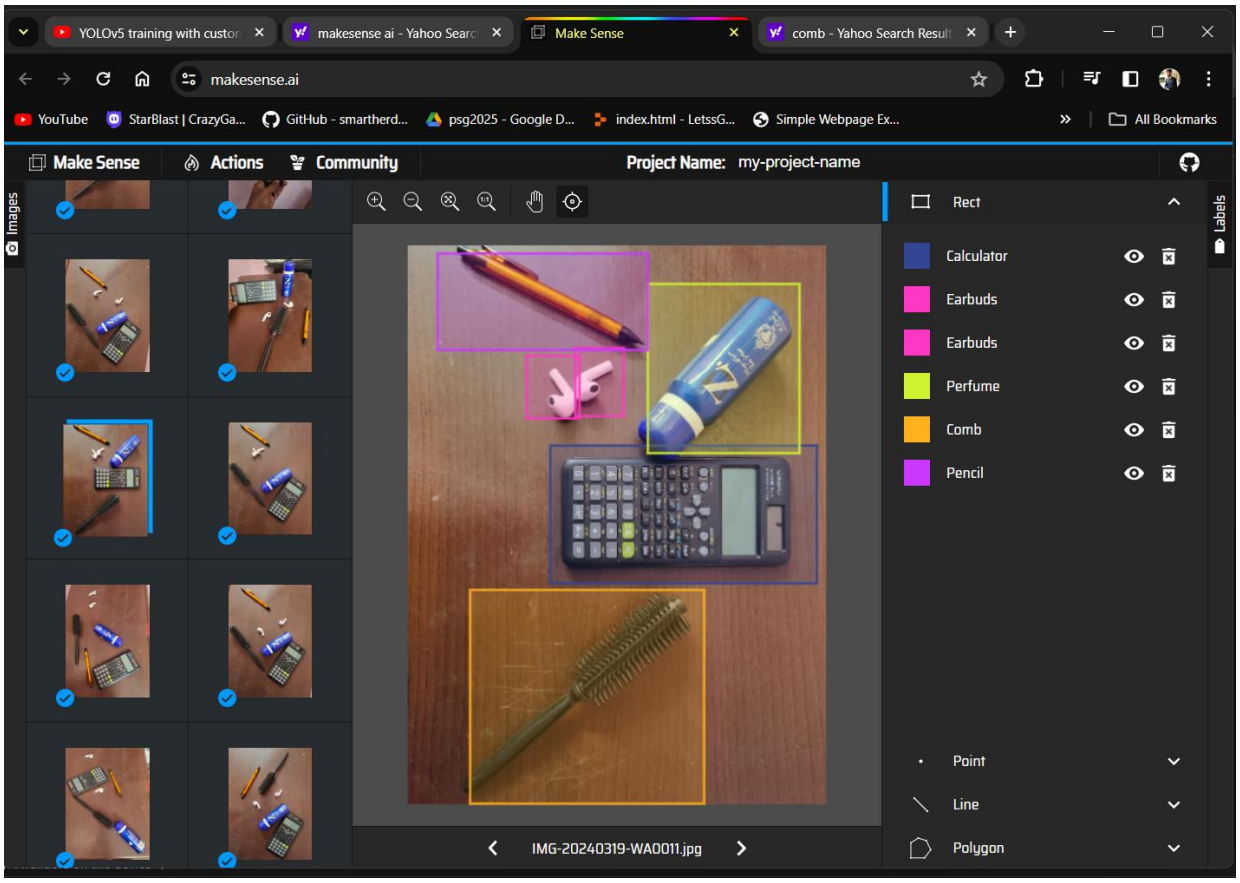


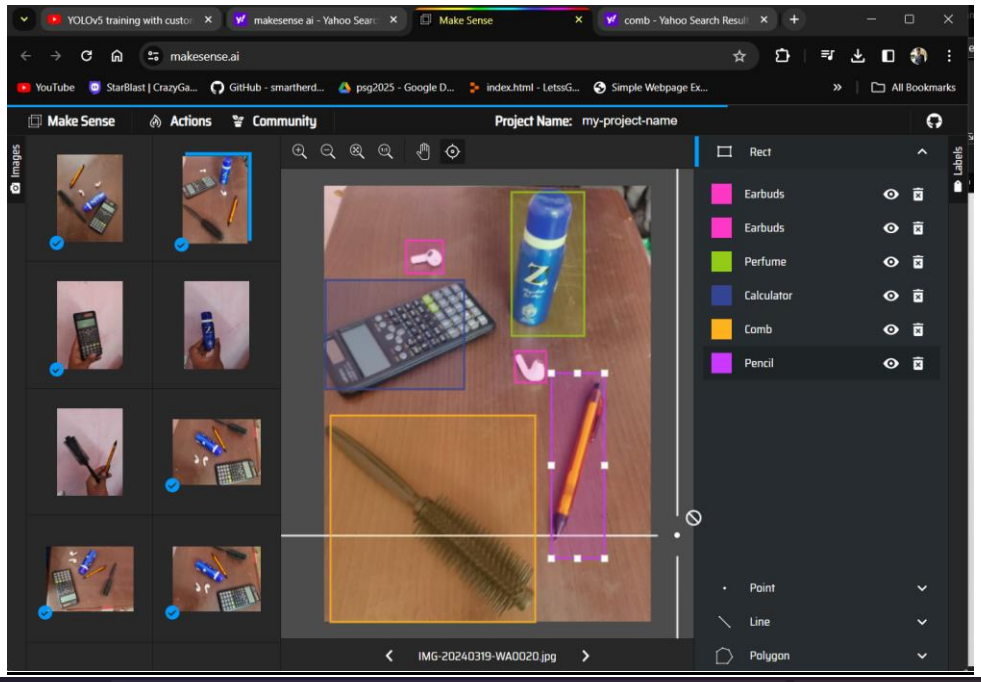


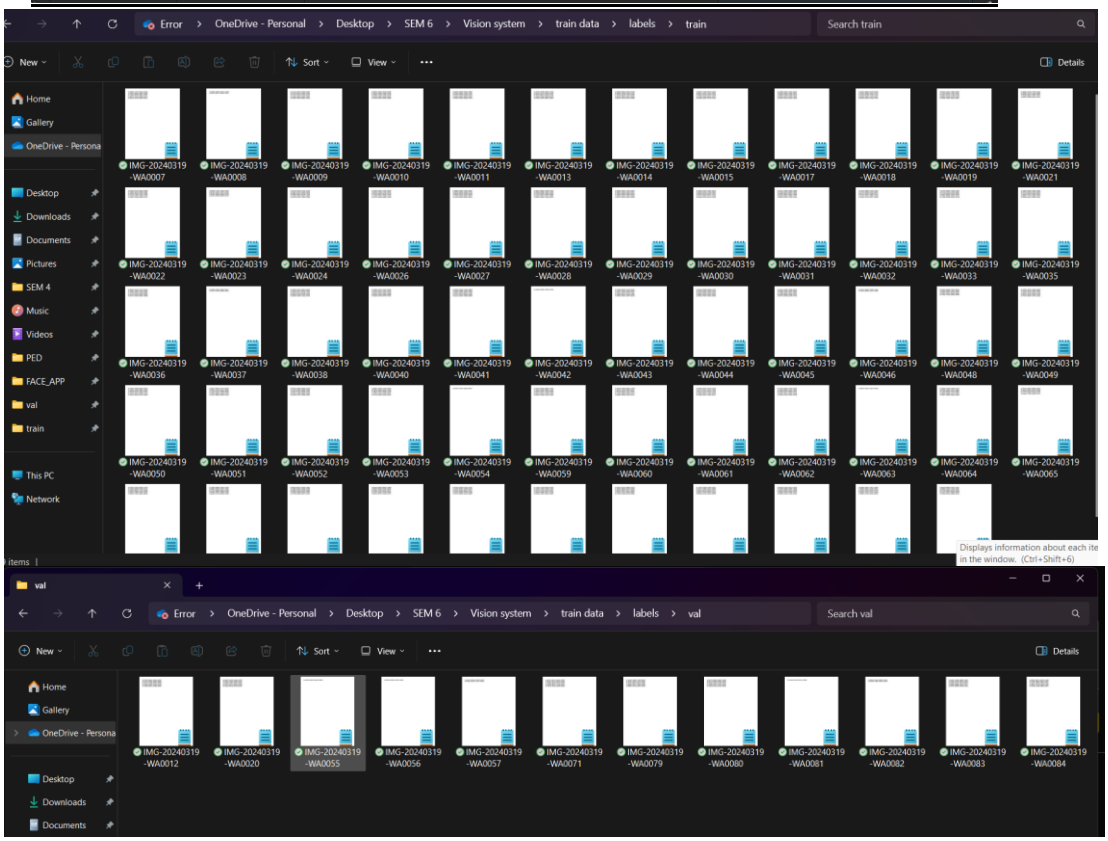
Custom data live video capture









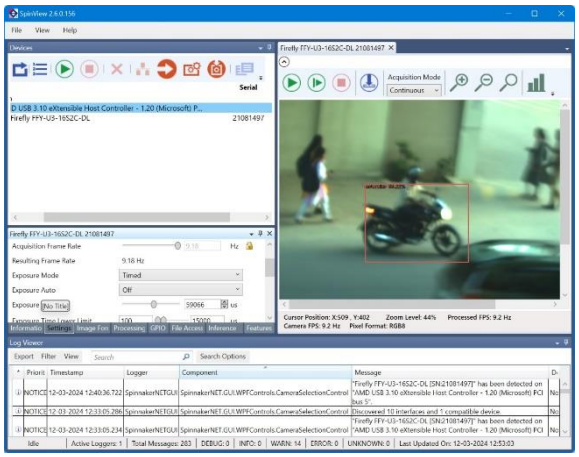


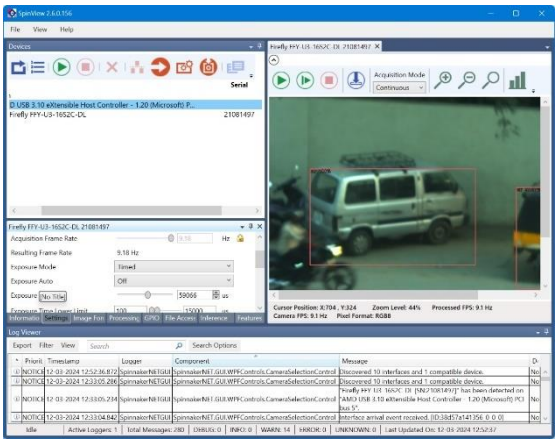
custom_data.yaml | Release Notes: 1.86.2

C: > Users > vs632 > Downloads > custom_data.yaml

```yaml
train: ../train data/images/train/# train images (relative to 'path') 128 images
val: ../train data/images/val # val images (relative to 'path') 128 images
nc: 5
# Classes
names: ['Calculator','Pencil','Comb','Earbuds','Perfume']
```

Object Detection using Deep Learning camera:





Post Lab Questions

1. **What are the key advantages of using Haar Cascades for face detection compared to other methods?**

Fast Processing: Haar cascades are computationally efficient, making them suitable for real-time applications.

Robustness: They can detect faces under various conditions such as changes in lighting, facial expressions, and minor occlusions.

Pre-trained Models: Haar cascades come with pre-trained models for face detection, saving time and computational resources during implementation.

Simple Implementation: Haar cascades are relatively simple to implement and understand, making them accessible to developers with varying levels of expertise.

2. **What are the difference between ANN & CNN.**

Feature	Artificial Neural Network (ANN)	Convolutional Neural Network (CNN)
Architecture	Consists of interconnected layers of neurons, including input, hidden, and output layers.	Employs specialized layers such as convolutional, pooling, and fully connected layers.
Feature Learning	Not specifically designed for feature extraction from structured data like images.	Specifically designed for feature extraction from images, utilizing convolutional layers.
Parameter Sharing	Each neuron in one layer is connected to every neuron in the subsequent layer, leading to a large number of parameters.	Utilizes parameter sharing and local connectivity through convolutional layers, reducing the number of parameters.
Application	Widely used in various tasks such as classification, regression, and clustering across different domains.	Particularly effective in image recognition, object detection, and tasks involving spatial data analysis.

3. **For the following image perform the convolution operation. Also perform Max pooling, Min pooling and Average pooling on the input image.**

6	5	4	3	2	1
7	6	5	4	3	2
8	7	6	5	4	3
9	8	7	6	5	4
10	9	8	7	6	5
10	10	9	8	7	6

3	2	4
2	0	2
4	2	3

Convolution Output:

[132 110 88 66]
[154 132 110 88]
[176 154 132 110]
[194 176 154 132]

Max Pooling With pool size(2x2)

[7,5,3]
[9,7,5]
[10,9,7]

Min Pooling With pool size(2x2)

[5,3,1]
[7,5,3]
[9,7,5]

Average Pooling With pool size(2x2)

[6,4,2]
[8,6,4]
[9.75,8,6]

Department of RAE			
Criteria	**Excellent (75% - 100%)**	**Good (50 – 75%)**	**Poor (<50%)**
Preparation (30)			
Performance (30)			
Evaluation (20)			
Report (20)			
Sign:		**Total (100)**	

Result:

Thus Face Detection using Haar Cascade and Object Detection using Yolo V5 were performed.