Ex.No: 8 Roll.No: 210701193

# PYTHON PROGRAM TO PERFORM OBJECT DETECTION WITH YOLOv3

Aim:

To perform object detection with YOLOv3 in python.

# Procedure:

1. Parse command-line arguments to choose between webcam, video, or image detection.
2. Load the pre-trained YOLO model, conﬁguration ﬁle, and COCO class names.
3. Load and resize the input image or video frame.
4. Convert the input image/frame to a blob for YOLO processing.
5. Perform forward pass through the YOLO network to detect objects.
6. Extract bounding boxes, class IDs, and conﬁdence scores from the YOLO output.
7. Apply non-maximum suppression to ﬁlter overlapping boxes.
8. Draw bounding boxes and labels on the detected objects in the image/frame.
9. Display the processed image/frame with detected objects.
10. Release video capture and close all OpenCV windows on exit.

# Code:

import cv2

import numpy as np import argparse import time

parser = argparse.ArgumentParser()

parser.add\_argument('--webcam', help="True/False", default=False)

parser.add\_argument('--play\_video', help="Tue/False", default=False) parser.add\_argument('--image', help="Tue/False", default=False) parser.add\_argument('--video\_path', help="Path of video ﬁle", default="videos/car\_on\_road.mp4")

parser.add\_argument('--image\_path', help="Path of image to detect objects", default="Images/bicycle.jpg")

parser.add\_argument('--verbose', help="To print statements", default=True) args = parser.parse\_args()

#Load yolo

def load\_yolo():

net = cv2.dnn.readNet("yolov3.weights", "yolov3.cfg") classes = []

with open("coco.names", "r") as f:

classes = [line.strip() for line in f.readlines()]

output\_layers = [layer\_name for layer\_name in net.getUnconnectedOutLayersNames()]

colors = np.random.uniform(0, 255, size=(len(classes), 3)) return net, classes, colors, output\_layers

def load\_image(img\_path): # image loading

img = cv2.imread(img\_path)

img = cv2.resize(img, None, fx=0.4, fy=0.4) height, width, channels = img.shape return img, height, width, channels

def start\_webcam():

cap = cv2.VideoCapture(0)

return cap

def display\_blob(blob): '''

Three images each for RED, GREEN, BLUE channel

'''

for b in blob:

for n, imgb in enumerate(b): cv2.imshow(str(n), imgb)

def detect\_objects(img, net, outputLayers):

blob = cv2.dnn.blobFromImage(img, scalefactor=0.00392, size=(320, 320), mean=(0, 0, 0), swapRB=True, crop=False)

net.setInput(blob)

outputs = net.forward(outputLayers) return blob, outputs

def get\_box\_dimensions(outputs, height, width): boxes = []

confs = [] class\_ids = []

for output in outputs: for detect in output:

scores = detect[5:]

class\_id = np.argmax(scores) conf = scores[class\_id]

if conf > 0.3:

center\_x = int(detect[0] \* width) center\_y = int(detect[1] \* height) w = int(detect[2] \* width)

h = int(detect[3] \* height)

x = int(center\_x - w/2) y = int(center\_y - h / 2)

boxes.append([x, y, w, h]) confs.append(ﬂoat(conf)) class\_ids.append(class\_id)

return boxes, confs, class\_ids

def draw\_labels(boxes, confs, colors, class\_ids, classes, img): indexes = cv2.dnn.NMSBoxes(boxes, confs, 0.5, 0.4)

font = cv2.FONT\_HERSHEY\_PLAIN

for i in range(len(boxes)): if i in indexes:

x, y, w, h = boxes[i]

label = str(classes[class\_ids[i]]) color = colors[i]

cv2.rectangle(img, (x,y), (x+w, y+h), color, 2) cv2.putText(img, label, (x, y - 5), font, 1, color, 1)

cv2.imshow("Image", img)

def image\_detect(img\_path):

model, classes, colors, output\_layers = load\_yolo() image, height, width, channels = load\_image(img\_path)

blob, outputs = detect\_objects(image, model, output\_layers)

boxes, confs, class\_ids = get\_box\_dimensions(outputs, height, width) draw\_labels(boxes, confs, colors, class\_ids, classes, image)

while True:

key = cv2.waitKey(1) if key == 27:

break

def webcam\_detect():

model, classes, colors, output\_layers = load\_yolo() cap = start\_webcam()

while True:

\_, frame = cap.read()

height, width, channels = frame.shape

blob, outputs = detect\_objects(frame, model, output\_layers)

boxes, confs, class\_ids = get\_box\_dimensions(outputs, height, width) draw\_labels(boxes, confs, colors, class\_ids, classes, frame)

key = cv2.waitKey(1) if key == 27:

break cap.release()

def start\_video(video\_path):

model, classes, colors, output\_layers = load\_yolo() cap = cv2.VideoCapture(video\_path)

while True:

\_, frame = cap.read()

height, width, channels = frame.shape

blob, outputs = detect\_objects(frame, model, output\_layers)

boxes, confs, class\_ids = get\_box\_dimensions(outputs, height, width) draw\_labels(boxes, confs, colors, class\_ids, classes, frame)

key = cv2.waitKey(1) if key == 27:

break cap.release()

if name == ' main ': webcam = args.webcam video\_play = args.play\_video

image = args.image if webcam:

if args.verbose:

print('---- Starting Web Cam object detection ')

webcam\_detect() if video\_play:

video\_path = args.video\_path if args.verbose:

print('Opening '+video\_path+" ")

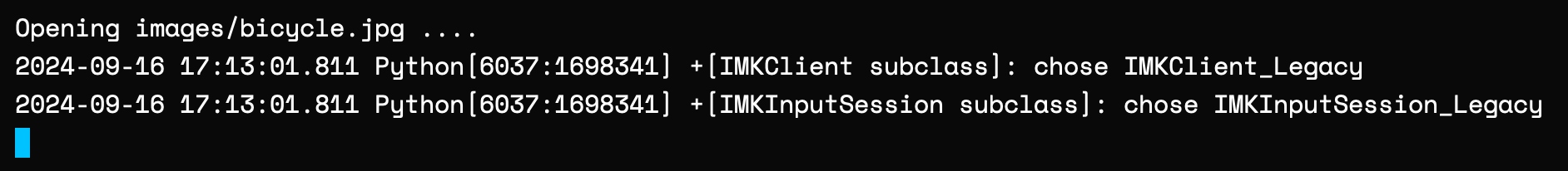
start\_video(video\_path) if image:

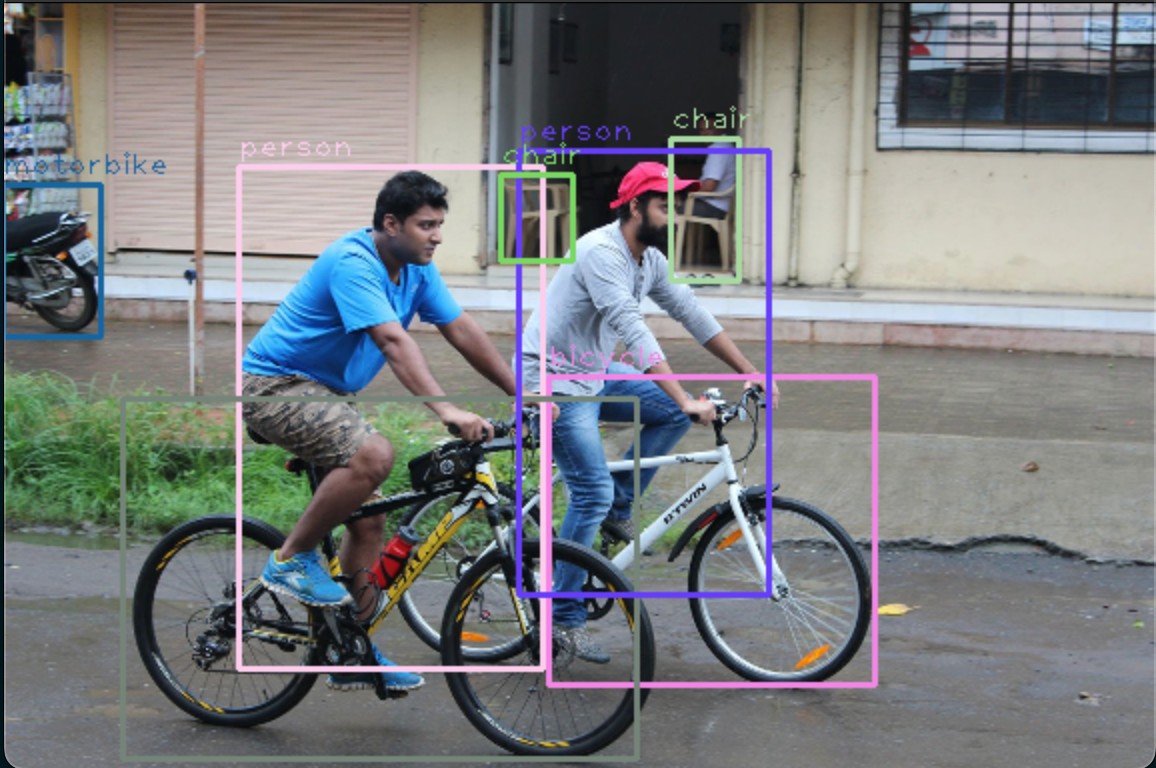
image\_path = args.image\_path if args.verbose:

print("Opening "+image\_path+" ")

image\_detect(image\_path) cv2.destroyAllWindows()

# Output:





Result:

Thus, to perform object detection using YOLOv3 in python has been completed successfully.