Ex No: 8

OBJECT DETECTION WITH YOLO3

Aim:

To build an object detection model with YOLO3 using Keras/TensorFlow.

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)
```

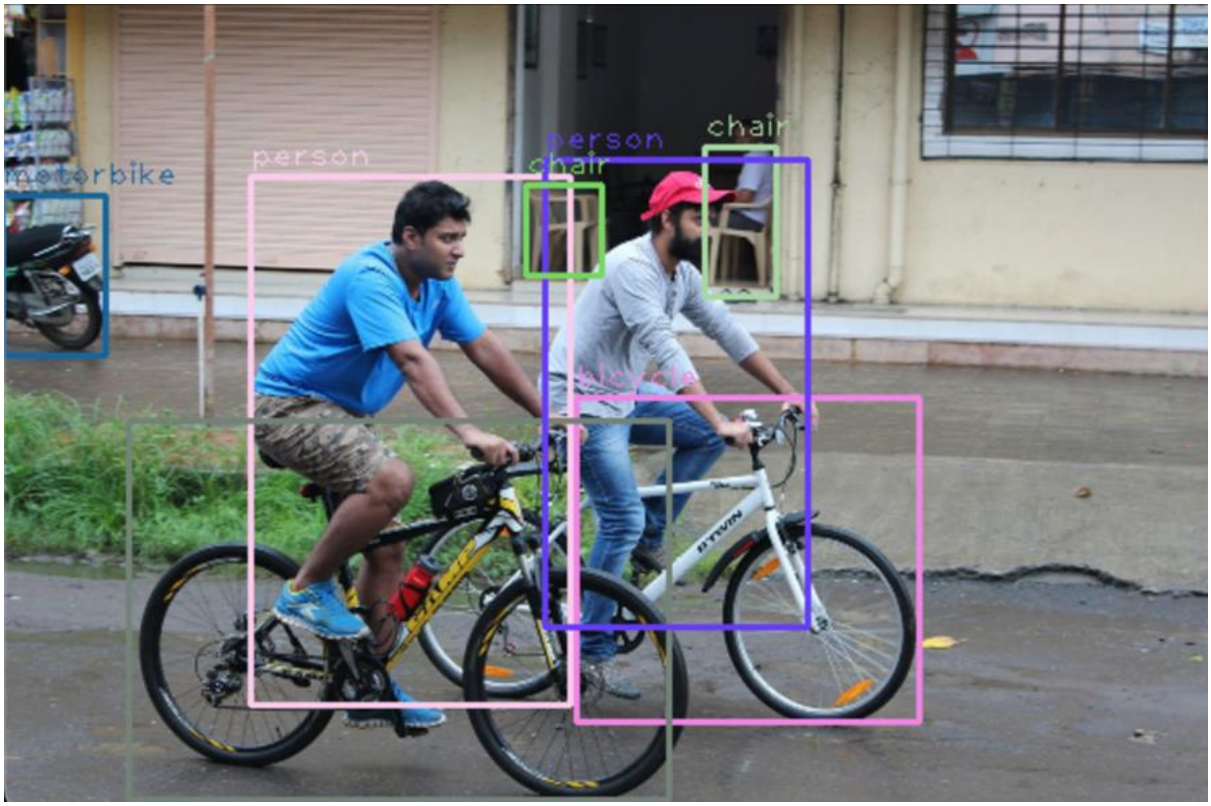
```
        cv2.imshow("Image", img)
def image_detect(img_path):
        model, classes, colors, output_layers = load_yolo()
        image, height, width, channels = load_image(img_path)
        blob, outputs = detect_objects(image, model, output_layers)
        boxes, confs, class_ids = get_box_dimensions(outputs, 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:

```
Opening images/bicycle.jpg ....
2024-09-16 17:13:01.811 Python[6037:1698341] +[IMKClient subclass]: chose IMKClient_Legacy
2024-09-16 17:13:01.811 Python[6037:1698341] +[IMKInputSession subclass]: chose IMKInputSession_Legacy
```



Result:

Thus the program to build an object detection model with YOLO3 using Keras/TensorFlow has been executed successfully.