

Ex No: 8

Date: 13-09-2024

OBJECT DETECTION WITH YOLO3

AIM:

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

PROCEDURE:

1. Download and load the dataset.
2. Perform analysis and preprocessing of the dataset.
3. Build an object detection model with YOLO3 using Keras/TensorFlow.
4. Compile and fit the model.
5. Perform prediction with the test dataset.
6. Calculate performance metrics.

PROGRAM:

```
import cv2

import numpy as np

import matplotlib.pyplot as plt

# Function to display images in Jupyter Notebook
def display_image(image):

    image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

    plt.figure(figsize=(10, 6))

    plt.imshow(image_rgb)

    plt.axis('off')

    plt.show()

# Load YOLO model
```

```
def load_yolo():  
    net = cv2.dnn.readNet("yolov3.weights", "yolov3.cfg") # Ensure you have these files in the  
correct path  
    classes = []  
    with open("coco.names", "r") as f: # Ensure 'coco.names' is in the correct path  
        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
```

Load and preprocess image

```
def load_image(img_path):  
    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
```

Detect objects in an image

```
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
```

Get bounding box dimensions for detected objects

```

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: # Confidence threshold
                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

# Draw labels on detected objects
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)
display_image(img) # Display the image in Jupyter Notebook
```

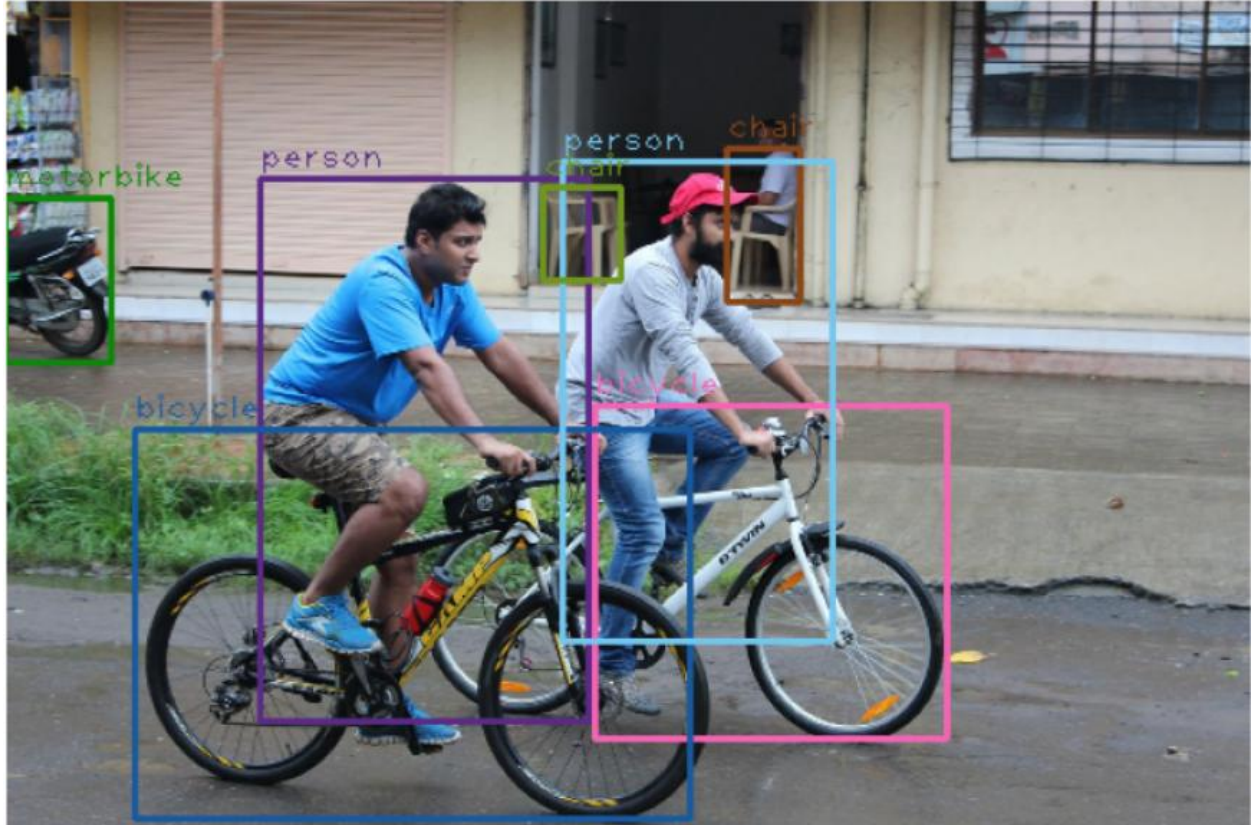
Main function to detect objects in an image

```
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)
```

Manually set the image path and call the detection function

```
image_path = "Images/bicycle.jpg" # Update this path to your test image location
image_detect(image_path)
```

OUTPUT:



RESULT:

Thus, an object detection model with YOLO3 using Keras/TensorFlow was successfully implemented.