import cv2

import numpy as np

import time

import os

from google.colab.patches import cv2\_imshow

# Load YOLO model

def load\_yolo():

    # Load the YOLO network

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

    # Get the output layer names

    layer\_names = net.getLayerNames()

    output\_layers = [layer\_names[i - 1] for i in net.getUnconnectedOutLayers()]

    # Load class names

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

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

    return net, classes, output\_layers

# Process image for detection

def detect\_cars(image\_path):

    # Load the model

    net, classes, output\_layers = load\_yolo()

     # Load and prepare image

    img = cv2.imread(image\_path)

    # Check if image was loaded successfully

    if img is None:

        print(f"Error: Could not load image at path: {image\_path}")

        print(f"Current working directory: {os.getcwd()}")  # Print current directory for debugging

        return 0  # Return 0 cars detected if image loading failed

    height, width, channels = img.shape

    # Create blob from image

    blob = cv2.dnn.blobFromImage(img, 0.00392, (416, 416), (0, 0, 0), True, crop=False)

    # Set input to the network

    net.setInput(blob)

    # Get detections

    start\_time = time.time()

    outputs = net.forward(output\_layers)

    end\_time = time.time()

    print(f"YOLO took {end\_time - start\_time:.2f} seconds")

    # Process detections

    class\_ids = []

    confidences = []

    boxes = []

    for output in outputs:

        for detection in output:

            scores = detection[5:]

            class\_id = np.argmax(scores)

            confidence = scores[class\_id]

            # Filter detections for cars (class\_id 2 in COCO dataset)

            if confidence > 0.5 and class\_id == 2:  # 2 is car in COCO dataset

                # Object detected

                center\_x = int(detection[0] \* width)

                center\_y = int(detection[1] \* height)

                w = int(detection[2] \* width)

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

                # Rectangle coordinates

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

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

                boxes.append([x, y, w, h])

                confidences.append(float(confidence))

                class\_ids.append(class\_id)

    # Apply non-maximum suppression

    indexes = cv2.dnn.NMSBoxes(boxes, confidences, 0.5, 0.4)

    # Draw boxes

    colors = np.random.uniform(0, 255, size=(len(classes), 3))

    font = cv2.FONT\_HERSHEY\_SIMPLEX

    for i in range(len(boxes)):

        if i in indexes:

            x, y, w, h = boxes[i]

            label = f"Car: {confidences[i]:.2f}"

            color = colors[class\_ids[i]]

            cv2.rectangle(img, (x, y), (x + w, y + h), color, 2)

            cv2.putText(img, label, (x, y - 10), font, 0.5, color, 2)

   # Save and show the result

    cv2.imwrite("car\_detection\_result.jpg", img)

    # Use cv2\_imshow instead of cv2.imshow

    cv2\_imshow(img)

    cv2.waitKey(0)

    cv2.destroyAllWindows()

    return len(indexes)  # Return number of cars detected

# Main function to call detection

def main():

    image\_path = "traffic\_image.jpeg"  # Replace with your image path

    num\_cars = detect\_cars(image\_path)

    print(f"Total cars detected: {num\_cars}")

if \_\_name\_\_ == "\_\_main\_\_":

    main()

YOLO took 4.60 seconds



Total cars detected: 27