```
# Counting the total number of the items in the images

import torch
import torchvision
import cv2
import numpy as np
from torchvision.transforms import functional as F
from google.colab.patches import cv2_imshow
from collections import Counter

model = torchvision.models.detection.fasterrcnn_resnet50_fpn(pretrained=True)
model.eval()

coco_cat_names = ['__background__', 'person', 'bicycle', 'car', 'motorcycle', 'airplane', 'bus',
                  'train', 'truck', 'boat', 'traffic light', 'fire hydrant', 'stop sign',
                  'parking meter', 'bench', 'bird', 'cat', 'dog', 'horse', 'sheep', 'cow',
                  'elephant', 'bear', 'zebra', 'giraffe', 'backpack', 'umbrella', 'handbag',
                  'tie', 'suitcase', 'frisbee', 'skis', 'snowboard', 'sports ball', 'kite',
                  'baseball bat', 'baseball glove', 'skateboard', 'surfboard', 'tennis racket',
                  'bottle', 'wine glass', 'cup', 'fork', 'knife', 'spoon', 'bowl', 'banana',
                  'apple', 'sandwich', 'orange', 'broccoli', 'carrot', 'hot dog', 'pizza',
                  'donut', 'cake', 'chair', 'couch', 'potted plant', 'bed', 'dining table',
                  'toilet', 'tv', 'laptop', 'mouse', 'remote', 'keyboard', 'cell phone',
                  'microwave', 'oven', 'toaster', 'sink', 'refrigerator', 'book', 'clock',
                  'vase', 'scissors', 'teddy bear', 'hair drier', 'toothbrush']

def calculate_iou(box1, box2):
    """Calculate Intersection over Union (IoU) between two bounding boxes."""
    x1, y1, x2, y2 = box1
    x1_p, y1_p, x2_p, y2_p = box2

    # Calculate the intersection
    inter_x1 = max(x1, x1_p)
    inter_y1 = max(y1, y1_p)
    inter_x2 = min(x2, x2_p)
    inter_y2 = min(y2, y2_p)

    inter_area = max(0, inter_x2 - inter_x1) * max(0, inter_y2 - inter_y1)

          ulate the union
         rea = (x2 - x1) * (y2 - y1)
    box2_area = (x2_p - x1_p) * (y2_p - y1_p)
    union_area = box1_area + box2_area - inter_area

    if union_area == 0:
        return 0
    return inter_area / union_area

def detect_obj(path, threshold=0.5, specific_item=None):
    img = cv2.imread(path)
    if img is None:
        print(f'Error: Could not load image from {path}. Please check the file path and ensure the image exists.')
        return None, None

    original_img = img.copy()
    img_tensor = F.to_tensor(img)

    with torch.no_grad():
        pred = model([img_tensor])

    boxes = pred[0]['boxes'].cpu().numpy()
    labels = pred[0]['labels'].cpu().numpy()
    scores = pred[0]['scores'].cpu().numpy()

    object_counts = Counter()  # To store count of detected objects

    for i, box in enumerate(boxes):
        if scores[i] >= threshold:
            label_index = labels[i]
            if 0 <= label_index < len(coco_cat_names):
                label = coco_cat_names[label_index]
            else:
                label = f"Unknown Label ({label_index})"
                print(f"Warning: Encountered unknown label index: {label_index}")

            score = scores[i]

            start = (int(box[0]), int(box[1]))
            end = (int(box[2]), int(box[3]))
            cv2.rectangle(original_img, start, end, (0, 255, 0), 2)
            cv2.putText(original_img, f"{label}: {score:.2f}", start, cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 0, 0), 2)
            # Count the detected object
            object_counts[label] += 1

    # Feature 4: Count Overlapping Objects
    overlaps = 0
    for i in range(len(boxes)):
        for j in range(i + 1, len(boxes)):
            if calculate_iou(boxes[i], boxes[j]) > 0.5:
                overlaps += 1
    print(f"\nNumber of overlapping objects: {overlaps}")

    # Feature 6: Summarize Items by Category
    movable_objects = ["car", "truck", "bicycle", "motorcycle", "bus", "train"]
    stationary_objects = ["chair", "bench", "dining table", "bed", "toilet", "couch"]

    movable_count = sum(object_counts[item] for item in movable_objects if item in object_counts)
    stationary_count = sum(object_counts[item] for item in stationary_objects if item in object_counts)

    print("\nSummary by Category:")
    print(f"Movable Objects: {movable_count}")
    print(f"Stationary Objects: {stationary_count}")

    # If a specific item is requested, filter the count for that item
    if specific_item:
        specific_item_count = object_counts.get(specific_item, 0)
        print(f"\nCount of '{specific_item}': {specific_item_count}")

    # Show the count of all detected objects
    print("\nObject counts:")
    for label, count in object_counts.items():
        print(f"{label}: {count}")

    return original_img, object_counts

if __name__ == '__main__':
    path = '/content/traffic_new.webp'  # Make sure this path is correct
    detected_image, object_counts = detect_obj(path)

    if detected_image is not None:
        cv2_imshow(detected_image)  # Display image in Colab

    # Example: Get the count of a specific item (e.g., 'cat')
    # detect_obj(path, specific_item="cat")
```

```
/usr/local/lib/python3.10/dist-packages/torchvision/models/_utils.py:208: UserWarning: The parameter 'pretrained' is deprecated sinc
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/torchvision/models/_utils.py:223: UserWarning: Arguments other than a weight enum or `None`
  warnings.warn(msg)
Downloading: "https://download.pytorch.org/models/fasterrcnn_resnet50_fpn_coco-258fb6c6.pth" to /root/.cache/torch/hub/checkpoints/
100%|██████████| 160M/160M [00:01<00:00, 127MB/s]

Number of overlapping objects: 12

Summary by Category:
Movable Objects: 63
Stationary Objects: 0

Object counts:
car: 56
truck: 5
bus: 2
traffic light: 1
```