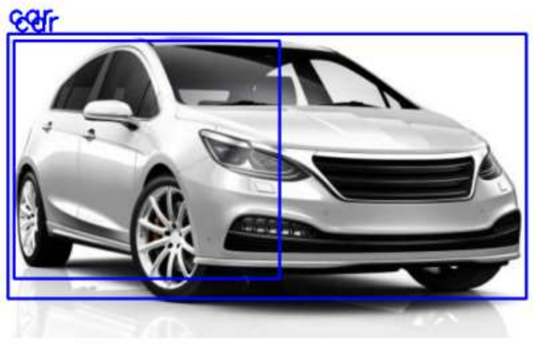
```
import torch
import torchvision
from torchvision.transforms import functional as F
import matplotlib.pyplot as plt
import numpy as np
import cv2
from PIL import Image
import os


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

Show hidden output

```
COCO_INSTANCE_CATEGORY_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 predict_image(image_path, model, threshold=0.5):
    image = Image.open(image_path).convert("RGB")
    image_tensor = F.to_tensor(image)

    with torch.no_grad():
        predictions = model([image_tensor])

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

    filtered_boxes = boxes[scores >= threshold]
    filtered_labels = labels[scores >= threshold]

    image_array = cv2.cvtColor(np.array(image), cv2.COLOR_RGB2BGR)

    for i, box in enumerate(filtered_boxes):
        label = COCO_INSTANCE_CATEGORY_NAMES[filtered_labels[i]]
        cv2.rectangle(image_array, (int(box[0]), int(box[1])), (int(box[2]), int(box[3])), (255, 0, 0), 2)
        cv2.putText(image_array, label, (int(box[0]), int(box[1]) - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.9, (255, 0, 0), 2)

    image_array = cv2.cvtColor(image_array, cv2.COLOR_BGR2RGB)
    plt.imshow(image_array)
    plt.axis("off")
    plt.show()


def detect_objects_in_dataset(image_folder):
    images = [os.path.join(image_folder, img) for img in os.listdir(image_folder) if img.endswith(('.jpg', '.png', '.jpeg'))]

    for image_path in images:
        print(f"Predicting for image: {image_path}")
        predict_image(image_path, model)


image_path = "/content/t2.jpg"
predict_image(image_path, model)
```

```
image_path = "/content/car12.jpg"
predict_image(image_path, model)
```



```
image_path = "/content/t4.jpg"
predict_image(image_path, model)
```

Start coding or generate with AI.