```
from  google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=Tr
```

```
from torchvision.models.detection import fasterrcnn_resnet50_fpn
from PIL import Image
import matplotlib.pyplot as plt
import matplotlib.patches as patches
import torchvision.transforms as transforms

detection_model = fasterrcnn_resnet50_fpn(pretrained=True, progress=False)

detection_model.eval();
```

```
/usr/local/lib/python3.10/dist-packages/torchvision/models/_utils.py:223: UserWarning: Arguments other than a weight enum or
  warnings.warn(msg)
```

```
sample_image = Image.open("../content/drive/My Drive/Colab Notebooks/img/cat_1.jpg")

transform = transforms.Compose([
    transforms.ToTensor(),
])

sample_image_tensor = transform(sample_image)
sample_image_tensor
```

```
tensor([[[0.0000, 0.0039, 0.0039,  ..., 0.0000, 0.0000, 0.0000],
         [0.0000, 0.0000, 0.0000,  ..., 0.0000, 0.0000, 0.0000],
         [0.0000, 0.0000, 0.0000,  ..., 0.0000, 0.0000, 0.0000],
         ...,
         [0.0000, 0.0000, 0.0000,  ..., 0.0039, 0.0000, 0.0000],
         [0.0000, 0.0000, 0.0000,  ..., 0.0039, 0.0039, 0.0000],
         [0.0000, 0.0000, 0.0000,  ..., 0.0039, 0.0039, 0.0000]],

        [[0.5098, 0.5137, 0.5137,  ..., 0.5647, 0.5608, 0.5647],
         [0.5098, 0.5098, 0.5098,  ..., 0.5647, 0.5608, 0.5647],
         [0.5098, 0.5098, 0.5098,  ..., 0.5647, 0.5608, 0.5647],
         ...,
         [0.4941, 0.4902, 0.4941,  ..., 0.6706, 0.6667, 0.6627],
         [0.4941, 0.4902, 0.4941,  ..., 0.6706, 0.6706, 0.6667],
         [0.4941, 0.4902, 0.4941,  ..., 0.6706, 0.6706, 0.6745]],

        [[0.7020, 0.7059, 0.7059,  ..., 0.7608, 0.7569, 0.7490],
         [0.7020, 0.7020, 0.7020,  ..., 0.7608, 0.7569, 0.7490],
         [0.7020, 0.7020, 0.7020,  ..., 0.7608, 0.7569, 0.7490],
         ...,
         [0.6902, 0.6863, 0.6902,  ..., 0.8627, 0.8588, 0.8549],
         [0.6902, 0.6863, 0.6902,  ..., 0.8627, 0.8627, 0.8588],
         [0.6902, 0.6863, 0.6902,  ..., 0.8627, 0.8627, 0.8627]]])
```

```
sample_image_tensor.shape
```

```
torch.Size([3, 667, 1000])
```

```
sample_image_tensor = sample_image_tensor.unsqueeze(dim=0)
sample_image_tensor.shape
```

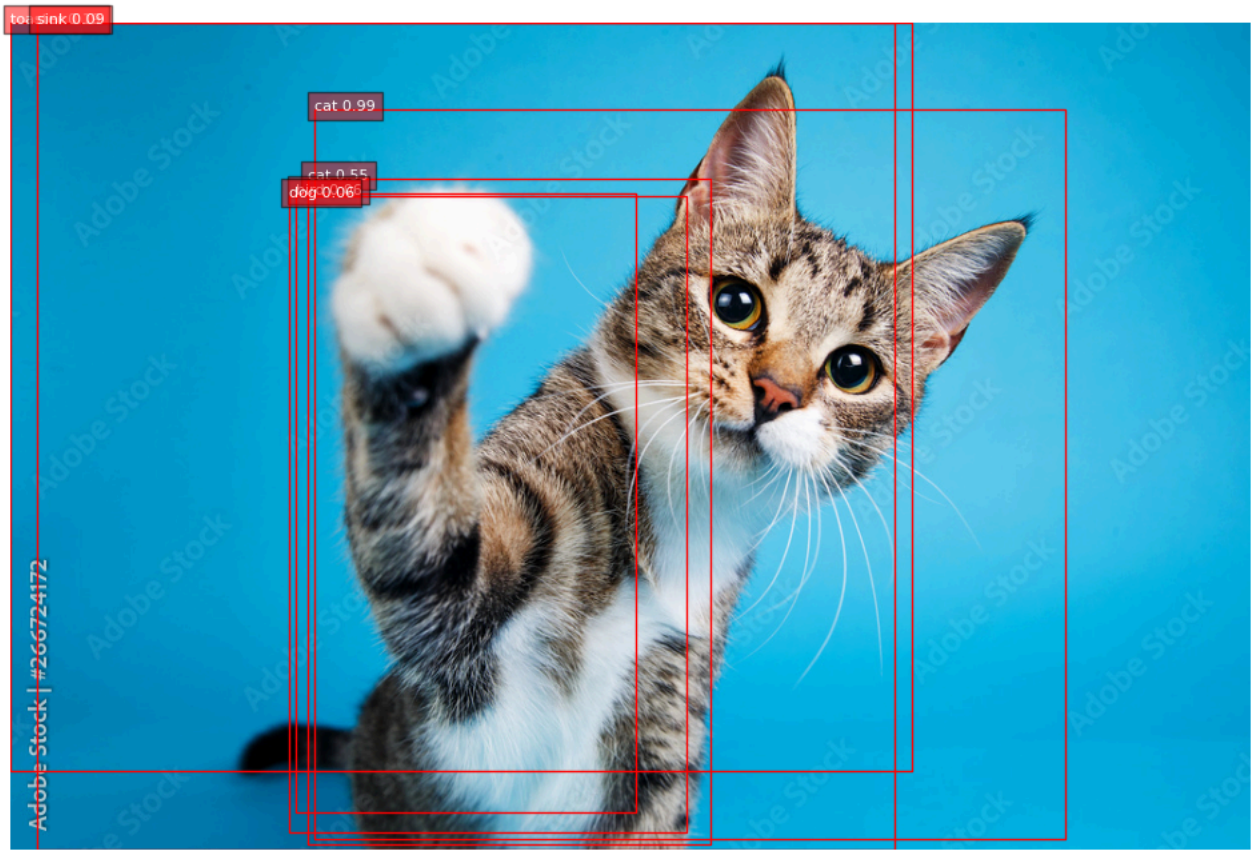
```
torch.Size([1, 3, 667, 1000])
```

```
preds = detection_model(sample_image_tensor)
preds
```

```
[{'boxes': tensor([[244.8693,  69.8805, 850.1993, 658.3933],
         [239.7491, 126.1456, 563.9388, 662.0167],
         [  0.0000,   0.0000, 726.8479, 602.6778],
         [ 20.6079,   0.0000, 712.1937, 667.0000],
         [229.0835, 137.7528, 503.4952, 636.4664],
         [223.7636, 139.9935, 544.3611, 652.7441]], grad_fn=<StackBackward0>),
  'labels': tensor([17, 17, 72, 73, 16, 18]),
  'scores': tensor([0.9935, 0.5474, 0.1891, 0.0905, 0.0578, 0.0553],
         grad_fn=<IndexBackward0>)}]
```

```
NAMES_COCO = [
    '','__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'
]

img = sample_image_tensor.squeeze().detach().cpu().numpy()
img = img.transpose(1, 2, 0)

fig, ax = plt.subplots(1, figsize=(12, 9))

ax.imshow(img)

for box, label, score in zip(preds[0]['boxes'], preds[0]['labels'], preds[0]['scores']):
    x1, y1, x2, y2 = box.detach().cpu().numpy()
    label_name = NAMES_COCO[label.item()]
    rect = patches.Rectangle((x1, y1), x2 - x1, y2 - y1, linewidth=1, edgecolor='r', facecolor='none')
    ax.add_patch(rect)
    plt.text(x1, y1, f'{label_name} {score.item():.2f}', color='white', fontsize=8, bbox=dict(facecolor='red', alpha=0.5))

plt.axis('off')
plt.show()
```




```
fixed_threshold = 0.8

img = sample_image_tensor.squeeze().detach().cpu().numpy()
img = img.transpose(1, 2, 0)

fig, ax = plt.subplots(1, figsize=(12, 9))

ax.imshow(img)

for box, label, score in zip(preds[0]['boxes'], preds[0]['labels'], preds[0]['scores']):
    if score.item() > fixed_threshold:
        x1, y1, x2, y2 = box.detach().cpu().numpy()

        label_index = label.item()
```

```
        label_name = NAMES_COCO[label_index]
        rect = patches.Rectangle((x1, y1), x2 - x1, y2 - y1, linewidth=1, edgecolor='r', facecolor='none')
        ax.add_patch(rect)
        plt.text(x1, y1, f'{label_name} {score.item():.2f}', color='white', fontsize=8, bbox=dict(facecolor='red', alpha=0.5

plt.axis('off')
plt.show()
```