Practical 8

Aim: Write a program for object detection from the image/video

Code:

```
import torch
import torchvision
import pycocotools
from PIL import Image
holiday = Image.open("1.png")
holiday
kids_playing = Image.open("1.png")
kids_playing
```

```
from torchvision.transforms.functional import pil_to_tensor

holiday_tensor_int = pil_to_tensor(holiday)
kids_playing_tensor_int = pil_to_tensor(kids_playing)

holiday_tensor_int.shape, kids_playing_tensor_int.shape
holiday_tensor_int = holiday_tensor_int.unsqueeze(dim=0)
kids_playing_tensor_int = kids_playing_tensor_int.unsqueeze(dim=0)
holiday_tensor_int.shape, kids_playing_tensor_int.shape
print(holiday_tensor_int.min(), holiday_tensor_int.max())

holiday_tensor_float = holiday_tensor_int / 255.0
kids_playing_tensor_float = kids_playing_tensor_int / 255.0
print(holiday_tensor_float.min(), holiday_tensor_float.max())
```

tensor(0, dtype=torch.uint8) tensor(255, dtype=torch.uint8)
tensor(0.) tensor(1.)

```
from torchvision.models.detection import fasterrcnn_resnet50_fpn
object_detection_model = fasterrcnn_resnet50_fpn(pretrained=True,
progress=False)
```

```
object_detection_model.eval(); ## Setting Model for
Evaluation/Prediction
holiday_preds = object_detection_model(holiday_tensor_float)


holiday_preds
```

```
[{'boxes': tensor([[156.1355,  97.0599, 318.1713, 420.2654],
        [544.3697,  73.9455, 701.8228, 456.6855],
        [303.2911, 109.9369, 490.6133, 423.2640],
        [431.7032, 121.4213, 573.4266, 429.1568],
        [447.8835, 408.0948, 546.7186, 499.4369],
        [445.9314, 125.8446, 478.6230, 203.2442],
        [411.1901, 136.3124, 518.2003, 417.5601],
        [449.5498, 406.3989, 546.4171, 499.2476],
        [634.3514, 185.9923, 696.7591, 412.1613],
        [431.7467, 135.1377, 493.2739, 373.8511]], grad_fn=<StackBackward0>),
  'labels': tensor([ 1,  1,  1,  1, 37,  1,  1, 34,  1,  1]),
  'scores': tensor([0.9998, 0.9998, 0.9989, 0.9973, 0.9579, 0.2353, 0.1320, 0.1135, 0.0576,
        0.0520], grad_fn=<IndexBackward0>)}]
```

```
holiday_preds[0]["boxes"] =
holiday_preds[0]["boxes"][holiday_preds[0]["scores"] > 0.8]
holiday_preds[0]["labels"] =
holiday_preds[0]["labels"][holiday_preds[0]["scores"] > 0.8]
holiday_preds[0]["scores"] =
holiday_preds[0]["scores"][holiday_preds[0]["scores"] > 0.8]
holiday_preds
```

```
[{'boxes': tensor([[156.1355,  97.0599, 318.1713, 420.2654],
        [544.3697,  73.9455, 701.8228, 456.6855],
        [303.2911, 109.9369, 490.6133, 423.2640],
        [431.7032, 121.4213, 573.4266, 429.1568],
        [447.8835, 408.0948, 546.7186, 499.4369]], grad_fn=<IndexBackward0>),
  'labels': tensor([ 1,  1,  1,  1, 37]),
  'scores': tensor([0.9998, 0.9998, 0.9989, 0.9973, 0.9579], grad_fn=<IndexBackward0>)}]
```

```
kids_preds = object_detection_model(kids_playing_tensor_float)
kids_preds
```

```
[{'boxes': tensor([[156.1355,  97.0599, 318.1713, 420.2654],
        [544.3697,  73.9455, 701.8228, 456.6855],
        [303.2911, 109.9369, 490.6133, 423.2640],
        [431.7032, 121.4213, 573.4266, 429.1568],
        [447.8835, 408.0948, 546.7186, 499.4369],
        [445.9314, 125.8446, 478.6230, 203.2442],
        [411.1901, 136.3124, 518.2003, 417.5601],
        [449.5498, 406.3989, 546.4171, 499.2476],
        [634.3514, 185.9923, 696.7591, 412.1613],
        [431.7467, 135.1377, 493.2739, 373.8511]], grad_fn=<StackBackward0>),
  'labels': tensor([ 1,  1,  1,  1, 37,  1,  1, 34,  1,  1]),
  'scores': tensor([0.9998, 0.9998, 0.9989, 0.9973, 0.9579, 0.2353, 0.1320, 0.1135, 0.0576,
        0.0520], grad_fn=<IndexBackward0>)}]
```

```
kids_preds[0]["boxes"] = kids_preds[0]["boxes"][kids_preds[0]["scores"]
> 0.8]
kids_preds[0]["labels"] =
```

```
kids_preds[0]["labels"][kids_preds[0]["scores"] > 0.8]
kids_preds[0]["scores"] =
kids_preds[0]["scores"][kids_preds[0]["scores"] > 0.8]
kids_preds
```

```
[{'boxes': tensor([[156.1355,  97.0599, 318.1713, 420.2654],
        [544.3697,  73.9455, 701.8228, 456.6855],
        [303.2911, 109.9369, 490.6133, 423.2640],
        [431.7032, 121.4213, 573.4266, 429.1568],
        [447.8835, 408.0948, 546.7186, 499.4369]], grad_fn=<IndexBackward0>),
  'labels': tensor([ 1,  1,  1,  1, 37]),
  'scores': tensor([0.9998, 0.9998, 0.9989, 0.9973, 0.9579], grad_fn=<IndexBackward0>)}]
```

```
from pycocotools.coco import COCO
annFile='instances_val2017.json'
coco=COCO(annFile)
holiday_labels = coco.loadCats(holiday_preds[0]["labels"].numpy())
holiday_labels
```

```
loading annotations into memory...
Done (t=0.00s)
creating index...
index created!

[{'supercategory': 'person', 'id': 1, 'name': 'person'},
 {'supercategory': 'person', 'id': 1, 'name': 'person'},
 {'supercategory': 'person', 'id': 1, 'name': 'person'},
 {'supercategory': 'person', 'id': 1, 'name': 'person'},
 {'supercategory': 'sports', 'id': 37, 'name': 'sports ball'}]
```

```
kids_labels = coco.loadCats(kids_preds[0]["labels"].numpy())
kids_labels
```

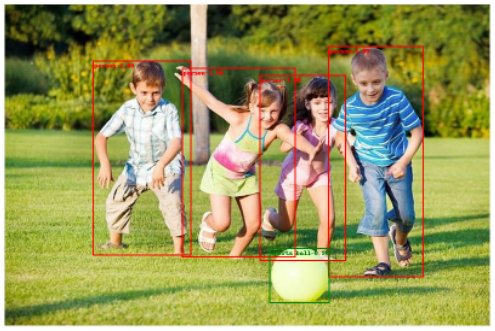
```
[{'supercategory': 'person', 'id': 1, 'name': 'person'},
 {'supercategory': 'person', 'id': 1, 'name': 'person'},
 {'supercategory': 'person', 'id': 1, 'name': 'person'},
 {'supercategory': 'person', 'id': 1, 'name': 'person'},
 {'supercategory': 'sports', 'id': 37, 'name': 'sports ball'}]
```

```
from torchvision.utils import draw_bounding_boxes

holiday_annot_labels = ["{}-{:.2f}".format(label["name"], prob) for
label, prob in zip(holiday_labels,
holiday_preds[0]["scores"].detach().numpy())]

holiday_output = draw_bounding_boxes(image=holiday_tensor_int[0],
                                     boxes=holiday_preds[0]["boxes"],
                                     labels=holiday_annot_labels,
                                     colors=["red" if label["name"]=="person"
else "green" for label in holiday_labels],
                                     width=2
```

```
                                )
holiday_output.shape
```

torch.Size([3, 546, 826])

```
from torchvision.transforms.functional import to_pil_image
to_pil_image(holiday_output)
```



```
from torchvision.utils import draw_bounding_boxes

kids_annot_labels = ["{}-{:.2f}".format(label["name"], prob) for label,
prob in zip(kids_labels, kids_preds[0]["scores"].detach().numpy())]

kids_output = draw_bounding_boxes(image=kids_playing_tensor_int[0],
                                  boxes=kids_preds[0]["boxes"],
                                  labels=kids_annot_labels,
                                  colors=["red" if label["name"]=="person"
else "green" for label in kids_labels],
                                  width=2,
                                  font_size=16,
                                  fill=True
                                 )
to_pil_image(kids_output)
```