The Sparks Foundation

Function: IOT and Computer Vision

Task 1- Object Detection- Implement an object detector which identifies the classes of the objects in an image or video.

Submitted by: Kumari Soni (August 2021 Batch)

Note:

```
YOLO (You Only Look Once) is a very powerful and a fast algorithm in object detection. A strong
understanding of the algorithm is essential before we start to code.

Some important papers to start with -

There are three papers you need to go through (Maybe difficult to understand initially, but worth reading
it)

You Only Look Once: Unified, Real-Time Object Detection
YOLO9000: Better, Faster, Stronger
YOLOv3: An Incremental Improvement

We are going to use YOLO v3 for coding purpose in this repository.

Before going to code, we need to download some important YOLO files. It's the folder that's present in
this repository as yolo-coco

The three files that needs to be downloaded are -
    coco.names
    yolov3.cfg
    yolov3.weights

Download these files and save it inside a folder. Name the folder anything you wish.
Create a folder images and have some pictures inside it to test the object detection.
```

In [11]:

```python
# import all the required libraries
import cv2
import matplotlib.pyplot as plt

from utils import *
from darknet import Darknet
```

In [12]:

```python
# Set the location and name of the cfg file
cfg_file = './cfg/yolov3.cfg'

# Set the location and name of the pre-trained weights file
weight_file = './weights/yolov3.weights'

# Set the location and name of the COCO object classes file
namesfile = 'data/coco.names'

# Load the network architecture
m = Darknet(cfg_file)

# Load the pre-trained weights
m.load_weights(weight_file)

# Load the COCO object classes
class_names = load_class_names(namesfile)
```

```
Loading weights. Please Wait...100.00% Complete
```

In [13]:

```python
# Print the neural network used in YOLOv3
m.print_network()
```

```
layer     filters    size              input                output
    0 conv     32  3 x 3 / 1   416 x 416 x   3   ->   416 x 416 x  32
    1 conv     64  3 x 3 / 2   416 x 416 x  32   ->   208 x 208 x  64
    2 conv     32  1 x 1 / 1   208 x 208 x  64   ->   208 x 208 x  32
    3 conv     64  3 x 3 / 1   208 x 208 x  32   ->   208 x 208 x  64
    4 shortcut 1
    5 conv    128  3 x 3 / 2   208 x 208 x  64   ->   104 x 104 x 128
    6 conv     64  1 x 1 / 1   104 x 104 x 128   ->   104 x 104 x  64
    7 conv    128  3 x 3 / 1   104 x 104 x  64   ->   104 x 104 x 128
    8 shortcut 5
```

```
  9 conv     64  1 x 1 / 1   104 x 104 x 128   ->   104 x 104 x  64
 10 conv    128  3 x 3 / 1   104 x 104 x  64   ->   104 x 104 x 128
 11 shortcut 8
 12 conv    256  3 x 3 / 2   104 x 104 x 128   ->    52 x  52 x 256
 13 conv    128  1 x 1 / 1    52 x  52 x 256   ->    52 x  52 x 128
 14 conv    256  3 x 3 / 1    52 x  52 x 128   ->    52 x  52 x 256
 15 shortcut 12
 16 conv    128  1 x 1 / 1    52 x  52 x 256   ->    52 x  52 x 128
 17 conv    256  3 x 3 / 1    52 x  52 x 128   ->    52 x  52 x 256
 18 shortcut 15
 19 conv    128  1 x 1 / 1    52 x  52 x 256   ->    52 x  52 x 128
 20 conv    256  3 x 3 / 1    52 x  52 x 128   ->    52 x  52 x 256
 21 shortcut 18
 22 conv    128  1 x 1 / 1    52 x  52 x 256   ->    52 x  52 x 128
 23 conv    256  3 x 3 / 1    52 x  52 x 128   ->    52 x  52 x 256
 24 shortcut 21
 25 conv    128  1 x 1 / 1    52 x  52 x 256   ->    52 x  52 x 128
 26 conv    256  3 x 3 / 1    52 x  52 x 128   ->    52 x  52 x 256
 27 shortcut 24
 28 conv    128  1 x 1 / 1    52 x  52 x 256   ->    52 x  52 x 128
 29 conv    256  3 x 3 / 1    52 x  52 x 128   ->    52 x  52 x 256
 30 shortcut 27
 31 conv    128  1 x 1 / 1    52 x  52 x 256   ->    52 x  52 x 128
 32 conv    256  3 x 3 / 1    52 x  52 x 128   ->    52 x  52 x 256
 33 shortcut 30
 34 conv    128  1 x 1 / 1    52 x  52 x 256   ->    52 x  52 x 128
 35 conv    256  3 x 3 / 1    52 x  52 x 128   ->    52 x  52 x 256
 36 shortcut 33
 37 conv    512  3 x 3 / 2    52 x  52 x 256   ->    26 x  26 x 512
 38 conv    256  1 x 1 / 1    26 x  26 x 512   ->    26 x  26 x 256
 39 conv    512  3 x 3 / 1    26 x  26 x 256   ->    26 x  26 x 512
 40 shortcut 37
 41 conv    256  1 x 1 / 1    26 x  26 x 512   ->    26 x  26 x 256
 42 conv    512  3 x 3 / 1    26 x  26 x 256   ->    26 x  26 x 512
 43 shortcut 40
 44 conv    256  1 x 1 / 1    26 x  26 x 512   ->    26 x  26 x 256
 45 conv    512  3 x 3 / 1    26 x  26 x 256   ->    26 x  26 x 512
 46 shortcut 43
 47 conv    256  1 x 1 / 1    26 x  26 x 512   ->    26 x  26 x 256
 48 conv    512  3 x 3 / 1    26 x  26 x 256   ->    26 x  26 x 512
 49 shortcut 46
 50 conv    256  1 x 1 / 1    26 x  26 x 512   ->    26 x  26 x 256
 51 conv    512  3 x 3 / 1    26 x  26 x 256   ->    26 x  26 x 512
 52 shortcut 49
 53 conv    256  1 x 1 / 1    26 x  26 x 512   ->    26 x  26 x 256
 54 conv    512  3 x 3 / 1    26 x  26 x 256   ->    26 x  26 x 512
 55 shortcut 52
 56 conv    256  1 x 1 / 1    26 x  26 x 512   ->    26 x  26 x 256
 57 conv    512  3 x 3 / 1    26 x  26 x 256   ->    26 x  26 x 512
 58 shortcut 55
 59 conv    256  1 x 1 / 1    26 x  26 x 512   ->    26 x  26 x 256
 60 conv    512  3 x 3 / 1    26 x  26 x 256   ->    26 x  26 x 512
 61 shortcut 58
 62 conv   1024  3 x 3 / 2    26 x  26 x 512   ->    13 x  13 x1024
 63 conv    512  1 x 1 / 1    13 x  13 x1024   ->    13 x  13 x 512
 64 conv   1024  3 x 3 / 1    13 x  13 x 512   ->    13 x  13 x1024
 65 shortcut 62
 66 conv    512  1 x 1 / 1    13 x  13 x1024   ->    13 x  13 x 512
 67 conv   1024  3 x 3 / 1    13 x  13 x 512   ->    13 x  13 x1024
 68 shortcut 65
 69 conv    512  1 x 1 / 1    13 x  13 x1024   ->    13 x  13 x 512
 70 conv   1024  3 x 3 / 1    13 x  13 x 512   ->    13 x  13 x1024
 71 shortcut 68
 72 conv    512  1 x 1 / 1    13 x  13 x1024   ->    13 x  13 x 512
 73 conv   1024  3 x 3 / 1    13 x  13 x 512   ->    13 x  13 x1024
 74 shortcut 71
 75 conv    512  1 x 1 / 1    13 x  13 x1024   ->    13 x  13 x 512
 76 conv   1024  3 x 3 / 1    13 x  13 x 512   ->    13 x  13 x1024
 77 conv    512  1 x 1 / 1    13 x  13 x1024   ->    13 x  13 x 512
 78 conv   1024  3 x 3 / 1    13 x  13 x 512   ->    13 x  13 x1024
 79 conv    512  1 x 1 / 1    13 x  13 x1024   ->    13 x  13 x 512
 80 conv   1024  3 x 3 / 1    13 x  13 x 512   ->    13 x  13 x1024
 81 conv    255  1 x 1 / 1    13 x  13 x1024   ->    13 x  13 x 255
 82 detection
 83 route  79
 84 conv    256  1 x 1 / 1    13 x  13 x 512   ->    13 x  13 x 256
 85 upsample           * 2    13 x  13 x 256   ->    26 x  26 x 256
 86 route  85 61
 87 conv    256  1 x 1 / 1    26 x  26 x 768   ->    26 x  26 x 256
 88 conv    512  3 x 3 / 1    26 x  26 x 256   ->    26 x  26 x 512
 89 conv    256  1 x 1 / 1    26 x  26 x 512   ->    26 x  26 x 256
 90 conv    512  3 x 3 / 1    26 x  26 x 256   ->    26 x  26 x 512
 91 conv    256  1 x 1 / 1    26 x  26 x 512   ->    26 x  26 x 256
 92 conv    512  3 x 3 / 1    26 x  26 x 256   ->    26 x  26 x 512
 93 conv    255  1 x 1 / 1    26 x  26 x 512   ->    26 x  26 x 255
 94 detection
 95 route  91
 96 conv    128  1 x 1 / 1    26 x  26 x 256   ->    26 x  26 x 128
 97 upsample           * 2    26 x  26 x 128   ->    52 x  52 x 128
 98 route  97 36
 99 conv    128  1 x 1 / 1    52 x  52 x 384   ->    52 x  52 x 128
100 conv    256  3 x 3 / 1    52 x  52 x 128   ->    52 x  52 x 256
101 conv    128  1 x 1 / 1    52 x  52 x 256   ->    52 x  52 x 128
102 conv    256  3 x 3 / 1    52 x  52 x 128   ->    52 x  52 x 256
103 conv    128  1 x 1 / 1    52 x  52 x 256   ->    52 x  52 x 128
104 conv    256  3 x 3 / 1    52 x  52 x 128   ->    52 x  52 x 256
```

```
105 conv     255  1 x 1 / 1    52 x  52 x 256   ->    52 x  52 x 255
106 detection
```

In [14]:

```
# Set the default figure size
plt.rcParams['figure.figsize'] = [24.0, 14.0]

# Load the image
img = cv2.imread('./images/city_scene.jpg')

# Convert the image to RGB
original_image = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

# We resize the image to the input width and height of the first layer of the network.
resized_image = cv2.resize(original_image, (m.width, m.height))

# Display the images
plt.subplot(121)
plt.title('Original Image')
plt.imshow(original_image)
plt.subplot(122)
plt.title('Resized Image')
plt.imshow(resized_image)
plt.show()
```

In [15]:

```
# Set the NMS threshold
nms_thresh = 0.6
```

In [16]:

```
# Set the IOU threshold
iou_thresh = 0.4
```

In [17]:

```
# Set the default figure size
plt.rcParams['figure.figsize'] = [24.0, 14.0]

# Load the image
img = cv2.imread('./images/city_scene.jpg')

# Convert the image to RGB
original_image = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

# We resize the image to the input width and height of the first layer of the network.
resized_image = cv2.resize(original_image, (m.width, m.height))

# Set the IOU threshold. Default value is 0.4
iou_thresh = 0.4

# Set the NMS threshold. Default value is 0.6
nms_thresh = 0.6

# Detect objects in the image
boxes = detect_objects(m, resized_image, iou_thresh, nms_thresh)

# Print the objects found and the confidence level
print_objects(boxes, class_names)

#Plot the image with bounding boxes and corresponding object class labels
plot_boxes(original_image, boxes, class_names, plot_labels = True)
```

```
It took 9.304 seconds to detect the objects in the image.
```

```
Number of Objects Detected: 28

Objects Found and Confidence Level:

1. person: 0.999996
2. person: 1.000000
3. car: 0.707236
4. truck: 0.933031
5. car: 0.658085
6. truck: 0.666982
7. person: 1.000000
8. traffic light: 1.000000
9. person: 1.000000
10. car: 0.997369
11. bus: 0.998023
12. person: 1.000000
13. person: 1.000000
14. person: 1.000000
15. person: 1.000000
16. person: 1.000000
17. traffic light: 1.000000
18. traffic light: 1.000000
19. handbag: 0.997282
20. traffic light: 1.000000
21. car: 0.989741
22. traffic light: 1.000000
23. traffic light: 0.999999
24. person: 0.999999
25. truck: 0.715036
26. traffic light: 1.000000
27. person: 0.999993
28. person: 0.999996
```