DETECTION OF OBJECTS IN A GIVEN IMAGE

ML06B1-3

Introduction

object detection is a technology which deals with computer vision and image processing that deals with detecting symmetric these object detection algorithms typically basically done with machine learning or deep learning to produce a meaningful result.

When we look at images or video we can just recognize as well as locate any object which surrounds us

So, our goal for this project is to replicate this intelligence using a computer. Here for this project we have used the yolo object detection technique with open cv and NumPy.

Description:

YOLO (you only look once) is a techniques to do object detection its more like an algorithm or strategy behind how the code is going to detect the object in image which is being provided.so what YOLO does is that it look at the entire image only once and goes through the network only once and detects object hence it's very fast then other techniques. Now the basic question arises is that how does this yolo technique works, this work in the following steps

1 - YOLO first take the input image.

2 – The framework then divide the input image into grid lets assume that this object detection divides into grid of 3X3.

3 - Image classification and localization and applied on each grid. YOLO then predicts the bounding boxes and their corresponding classes probabilities for object.

Libraries used:

```
import cv2
import numpy as np
```

1 **OpenCV -** OpenCV is an open source machine learning library which target the real time computer vision, it's written in C++ and its primary interface is also in C++. In python this library used to solve the computer vision problem in our project we have used the open cv for the purpose of image detection.

2 **NumPy** - NumPy is a general-purpose array processing package which delivers high performance multidimensional array object it generally used for scientific computation with python. NumPy can also be used as an efficient multi-dimensional container for the general data. This allow NumPy to seamlessly and speedily integrate with wide variety of database.

NumPy contains:

1 – A powerful n dimensional array object.

2- Has more supplicated function then other libraries present out there

3 – NumPy has tools to integrate c/c++ and also Fortran code.

Now we are going to load YOLO

```
# Load Yolo
net = cv2.dnn.readNet("yolov3.weights", "yolov3.cfg")
classes = []
with open("coco.names", "r") as f:
    classes = [line.strip() for line in f.readlines()]
layer_names = net.getLayerNames()
output_layers = [layer_names[i[0] - 1] for i in
net.getUnconnectedOutLayers()]
colors = np.random.uniform(0, 255, size=(len(classes), 3))
```

We have used the variable named as net and its purpose is to read the data which is being defined to read our file yolov3.weights", "yolov3.cfg which is our weights file and cfg file for

this project. Now we have defined a variable as classes which will return a copy of string with both leading and training character removed using strip (). Now we defined a variable layer name as net. getLayerNames() which return the list of type of layer used in this model and defined output layers as net.getUnconnectedOutLayers() which return the index of layer with unconnected output. Now we defined a variable named colors as np np.random.uniform (0, 255, size=(len(classes), 3)) so what it basically does is it generates the multiple random number that too with our desired shape.

Loading image:

```
# Loading image
img = cv2.imread("asp.jpg")
img = cv2.resize(img, None, fx=0.4, fy=0.4)
height, width, channels = img.shape
```

As we have already loaded the yolo in previous step now we are going to load the image and for that we use OpenCV library so first we defined a variable named img which is going to read our image file and store it in img then we resize the image file(.jpg extension) using cv2.resize.

Detecting object

```
# Detecting objects
blob = cv2.dnn.blobFromImage(img, 0.00392, (416, 416), (0, 0, 0), True,
crop=False)

net.setInput(blob)
outs = net.forward(output_layers)
```

Now for image detection we have used the variable blob and in that we defined by blob = cv2.dnn.blobFromImage(img, 0.00392, (416, 416), (0, 0, 0), True, crop=False) now what is does is that it derives the class encapsulates function of certain backends then what net.setInput(blob) does is that it set the new value for the layer output blob.Now net. Forward(output_layers) does is that it passes to compile output of layer, default runs forward pass for the whole network.

Displaying the output

Loop is applied in the sequence outs where each element (out) also undergoes for loop. The individual element detection which is again a sequence is sliced from the fifth position and assigned to scores.

```
for out in outs:
   for detection in out:
      scores = detection[5:]
```

Post this, using the function argmax from NumPy module, we assign the index of max value of the structure scores to class_id and the maximum value to confidence.

```
class_id = np.argmax(scores)
      confidence = scores [class_id]
```

The following code is executed once the given condition (confidence > 0.5) is satisfied. Using this piece of code, we compute the height and width of the object detected from the image in int.

```
center_x = int (detection [0] * width)
 center_y = int (detection [1] * height)
 w = int (detection [2] * width)
 h = int (detection [3] * height)
 x = int (center_x - w / 2)
 y = int (center_y - h / 2)
```

The rectangular coordinate values are computed using the above found values. These final values are put into a new list boxes. Along with it, the above found confidence and class_id are also appended to the created lists confidences and class_ids respectively.

```
boxes.append ([x, y, w, h])
        confidences.append (float (confidence))
        class_ids.append (class_id)
```

Displaying the identified objects on the screen:

In the below piece of code, we try to create a box around the objects that are identified and properly name them and display the main image on the screen with object identifiers drawn on it.

```
indexes = cv2.dnn.NMSBoxes(boxes, confidences, 0.5, 0.4)

print(indexes)

font = cv2.FONT_HERSHEY_COMPLEX

for I in range(len(boxes)):
```

```
        if i in indexes:
        x, y, w, h = boxes[i]
        label = str(classes[class_ids[i]])
        color = colors[i]
        cv2.rectangle(img, (x, y), (x + w, y + h), color, 2)
        cv2.putText(img, label, (x, y + 30), font, 0.65, color, 1)

cv2.imshow("Image", img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

We are getting the indexes that bound the identified objects. We later use those indexes in order to draw a box on the identified objects. We are setting font HERSHEY COMPLEX using the function provided by the OpenCV that is *cv2.fontname.*

We run a for loop with the range as the length of the list named boxes. We obtain the starting and ending indexes and also the width and height of the box we are supposed to draw. Also, we obtain the label of the corresponding object from the list. We set the color correspondingly. We use the function *cv2.rectangle* and pass it the image id, starting index, ending index and the color and thickness of the line. Afterwards we use the function *cv2.putText* to set the name of the object below the rectangle that was drawn around the identified object.

We use *cv2.imshow* to display the final image with identified objects properly marked and *cv2.waitkey(0)* is to keep the image displayed until the user wishes. And *cv2.destroyAllWindows()* will close the image output window once the user clicks the close button.

Conclusion and Result:

So, after implementing the above code we obtained our following desired result, the result clearly specifies or detect different object as shown in output.

Input:

So, we have given this image as input:

Output:

The output we found through this program is given:

Future scope of object detection:

so, after doing this project on object detection we understood that the task of finding object belongs to the classes of interest of computer vision and research. The ability to find the object is useful in many application such as self-driving car were object detection allows the car to detect different object so that the passenger can travel safely , Through a lot of progress has been made since the conception of the field of computer vision more then five decades ago as always there is a scope for the further improvement