Project Report On

“Object Detection using Deep Learning”

Submitted in Partial Fulfillment of the Requirements For

the award of the Degree

of

BACHELOR OF TECHNOLOGY IN

computer science and engineering

By

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

This is to certify that the Project work entitled **"Object Detection using Deep Learning"**, submitted by **A. NITHISH CHANDRA** towards partial fulfillment for the award of a Bachelor's Degree in Computer science Engineering

DECLARATION

This is to certify that the work reported in the present project titled **"Object detection using Deep Learning**" is a record of work done by us in the **Department of Computer science and Engineering,**

A. NITHISH CHANDRA

ABSTRACT

The ubiquitous and wide applications like scene understanding, video surveillance, and self-driving systems triggered vast research in the domain of computer vision in the most recent decade. Due to significant development in neural networks, especially deep learning, these visual recognition systems have attained remarkable performance. Object detection is one of these domains witnessing great success in computer vision. The Objective of the project is to detect objects using the You Only Look Once (YOLO) approach. This method has several advantages as compared to other object detection algorithms. In other algorithms like Convolutional Neural Network, Fast Convolutional Neural Network the algorithm will not look at the image completely but in YOLO the algorithm looks at the image completely by predicting the bounding boxes using convolutional network and the class probabilities for these boxes and detects the image faster as compared to other algorithms. This project demystifies the role of deep learning techniques based on convolutional neural networks for object detection.

CONTENTS

1. INTRODUCTION

To gain a complete image understanding, we should not only concentrate on classifying different images but also try to precisely estimate the concepts and locations of objects contained in each image. This task is referred to as object detection, which usually consists of different subtasks such as face detection, and pedestrian detection. As one of the fundamental computer vision problems, object detection is able to provide valuable information for semantic understanding of images and videos and is related to many applications, including image classification, human behavior analysis, face recognition, and **autonomous driving**. Meanwhile, inheriting from neural networks and related learning systems, the progress in these fields will develop neural network algorithms, and will also have great impact on object detection techniques which can be considered as learning systems. Object detection is a technology that detects the semantic objects of a class in digital images and videos. One of its real-time applications is **self-driving cars**. In this, our task is to detect multiple objects from an image. The most common object to detect in this application is the car, motorcycle, and pedestrian. There are various techniques for object detection, they can be split up into two categories, first is the algorithms based on Classifications. CNN and RNN come under this category. In this, we have to select the interested regions from the image and have to classify them using the Convolutional Neural Network. This method is very slow because we have to run a prediction for every selected region. The second category is the algorithms based on Regressions. **The YOLO** method comes under this category. In this, we won't select the interested regions from the image. Instead, we predict the classes and bounding boxes of the whole image at a single run of the algorithm and detect multiple objects using a single neural network. The YOLO algorithm is fast as compared to other classification algorithms. In real time our algorithm processes 45 frames per second. The YOLO algorithm makes localization errors but predicts less false positives in the background.

2. LITERATURE SURVEY

Object detection is a common term for computer vision techniques classifying and locating objects in an image. It plays an important role in computer vision, automatic vehicles, industrial automation etc. Deep learning in object detection is better than traditional target detection. Deep learning methods include Region proposal object detection algorithms wherein it generates region proposal networks and then classifies them. These include SPPnet, Region-based Convolutional Neural Networks, Fast RCNN, etc. Modern object detection is largely based on use of convolutional neural networks. Some of the most relevant system types today are R-FCN, Multibox Single Shot Detector (SSD) and YOLO (You Only Look Once).You Only Look Once: Unified, Real-Time Object Detection, was created by Joseph Redmon. The algorithm's prior work is on detecting objects using a regression algorithm. To get high accuracy and good predictions, YOLO was proposed. YOLO is popular because it achieves high accuracy while also being able to run in real-time. It is extremely fast. It sees the entire image during training and test time so it implicitly encodes contextual information about classes as well as their appearance. It learns generalizable representations of objects so that when trained on natural images and tested on artwork, the algorithm outperforms other top detection methods. These are the main reasons why YOLO was the chosen model for our project.

3. YOLO MODEL

YOLO stands for You Only Look Once. It's an object detector that uses features learned by a deep convolutional neural network to detect an object. YOLO makes use of only convolutional layers, making it a fully convolutional network (FCN). It has 75 convolutional layers, with skip connections and up sampling layers. No form of pooling is used, and a convolutional layer is used to down sample the feature maps. This helps in preventing loss of low-level features.

4. YOLO ALGORITHM

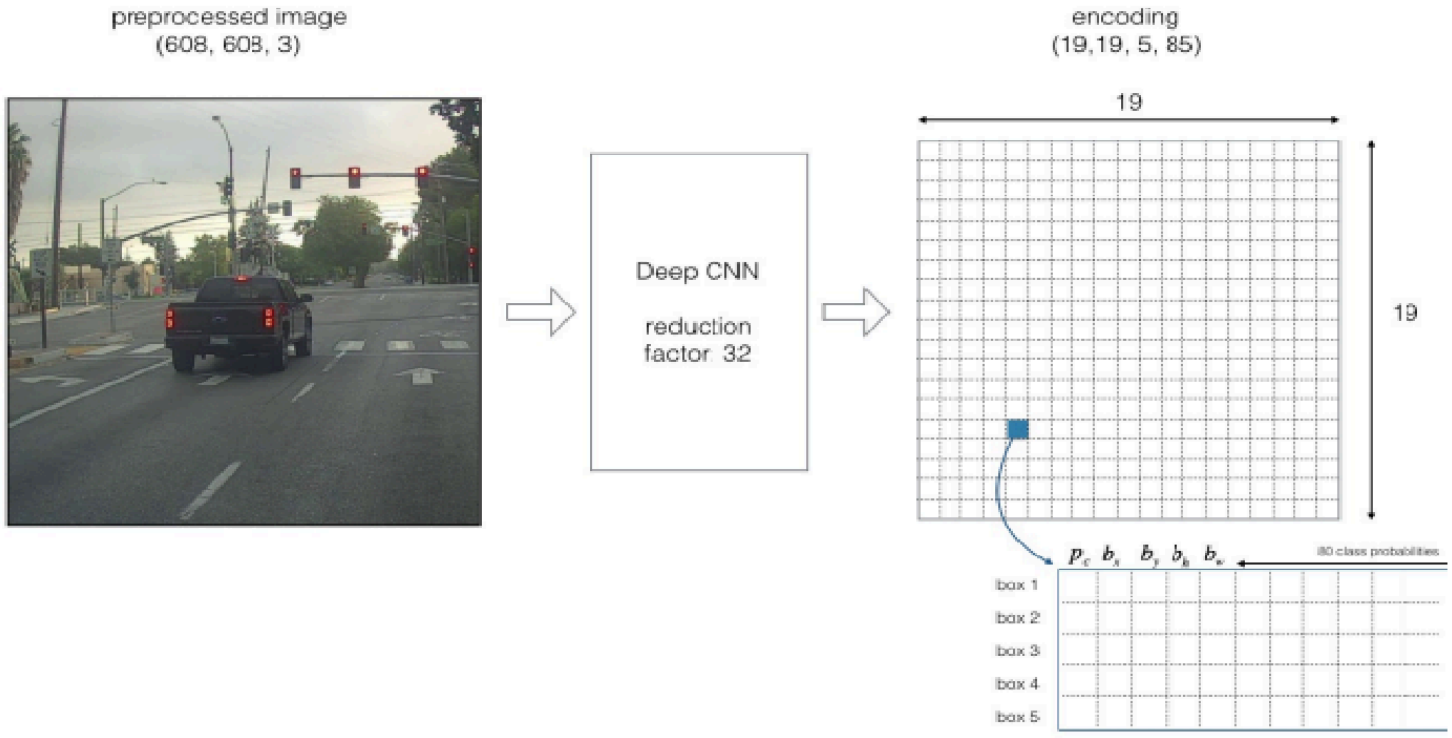
4.1 The Algorithm

YOLO algorithm gives a much better performance on all the parameters along with a high fps for real-time usage. YOLO algorithm is an algorithm based on regression, instead of selecting the interesting part of an Image, it predicts classes and bounding boxes for the whole image in **one run of the Algorithm.**

To understand the YOLO algorithm, first we need to understand what is actually being predicted. Ultimately, we aim to predict a class of an object and the bounding box specifying object location. Each bounding box can be described using four descriptors:

1. Center of the box (**bx, by**)
2. Width (**bw**)
3. Height (**bh**)
4. Value **c** corresponding to the class of an object

Along with that we predict a real number **pc**, which is the probability that there is an object in the bounding box. YOLO doesn't search for interested regions in the input image that could contain an object, instead it splits the image into cells, typically a 19x19 grid. Each cell is then responsible for predicting K bounding boxes.
Here we take K=5 and predict the possibility for 80 classes.



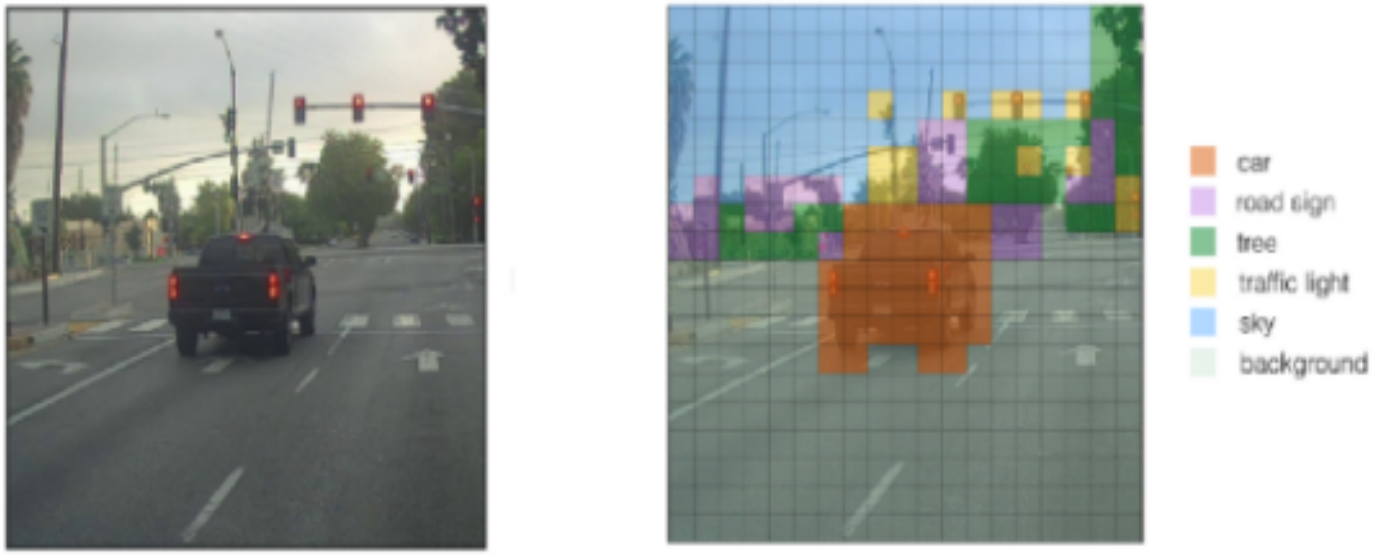
Here we take K=5 and predict possibility for 80 classes

4.2 What are Anchor boxes and their use-

Anchor boxes are chosen by exploring the training data to choose reasonable height/width ratios that represent the different classes. An Object is considered to lie in a specific cell only if the center coordinates of the anchor box lie in that cell. Due to this property the center coordinates are always calculated relative to the cell whereas the height and width are calculated relative to the whole Image size.

During the one pass of forward propagation, YOLO determines the probability that the cell contains a certain class. Probability that there is an object of certain class 'c'.The class with the maximum probability is chosen and assigned to that particular grid cell. Similar process happens for all the grid cells present in the image.

After computing the above class probabilities, the image may look like this:



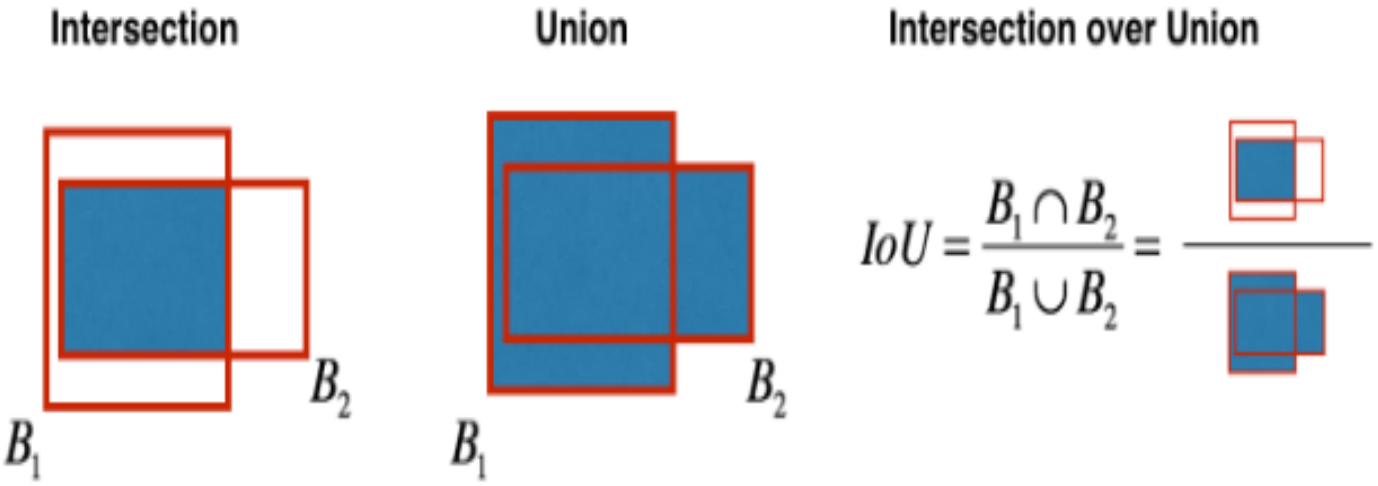
Various Classes of the image

This shows the before and after of predicting the class probabilities for each grid cell. After predicting the class probabilities, the next step is Non-max suppression. It helps the algorithm to get rid of the unnecessary anchor boxes, like you can see that in the figure below, there are numerous anchor boxes calculated based on the class probabilities.

AnchorBoxes

4.3 Non-max suppression

To resolve this problem Non-max suppression eliminates the bounding boxes that are very close by performing the IoU (Intersection over Union) with the one having the highest class probability among them.



IoU operation

It calculates the value of IoU for all the bounding boxes respective to the one having the highest class probability, it then rejects the bounding boxes whose value of IoU is greater than a threshold. It signifies that those two bounding boxes are covering the same object but the other one has a low probability for the same, thus it is eliminated.

Once done, algorithm finds the bounding box with next highest class probabilities and does the same process, it is done until we are left with all the different bounding boxes.

Before and After Non-Max-Suppression

After this, almost all of our work is done, the algorithm finally outputs the required vector showing the details of the bounding box of the respective class.

5. CODE IMPLEMENTATION

Execution of the entire code is done in the Jupyter notebook of the Anaconda distribution of python.

Steps followed for implementing yolo algorithm:

1. Import libraries and load weights file and input

In [1]:
```
import numpy as np
import cv2
import matplotlib.pyplot as plt
```

In [2]:
```
INPUT_FILE='C:/Users/Anagha/Desktop/group_project/cars/vid_4_940.jpg'
LABELS_FILE='data/coco.names'
CONFIG_FILE='cfg/yolov3.cfg'
WEIGHTS_FILE='yolov3.weights'
CONFIDENCE_THRESHOLD=0.3
```

- Numpy:

  It is a powerful library in python used for manipulating data in the form of an array. Since image data is stored in the form of an array of RGB pixel values stacked on top of each other, this library comes in handy.
- OpenCV:

  This library is widely used for dealing with processing image data and applying algorithms over it.
- Matplotlib:

  It is a powerful visualization tool that is used for plotting graphs and viewing images in the jupyter kernel itself.
- Then the yolo weights file and labels file downloaded from github are loaded.
- A confidence threshold value is set.

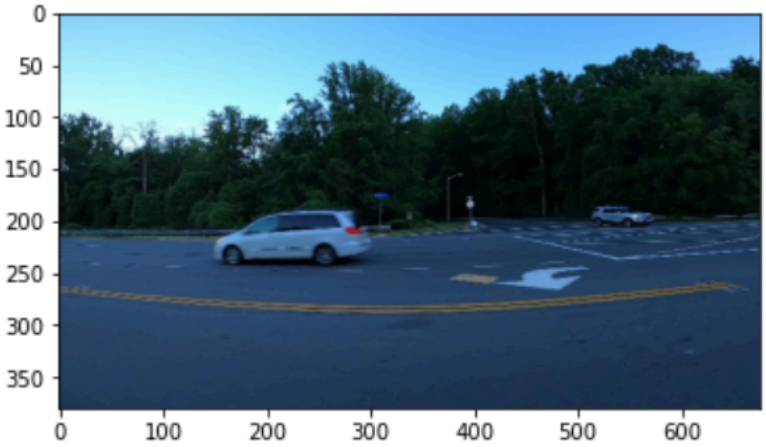
2. Load the input image

   INPUT:

```
In [62]: #Loading the imput image
         image = cv2.imread(INPUT_FILE)
```

```
In [63]: import matplotlib.pyplot as plt
         plt.imshow(image[:,:,::-1])
```

```
Out[63]: <matplotlib.image.AxesImage at 0x1e2167deee0>
```



The loaded image is visualized using matplotlib.

3. Create a deep neural network model using the yolo weights and configuration files.

```
In [5]: #creating the model
        net = cv2.dnn.readNetFromDarknet(CONFIG_FILE, WEIGHTS_FILE)
```

```
In [6]: net
```

```
Out[6]: <dnn_Net 000001E22FF6C2B0>
```

4. Filter the output anchor box predictions based on the confidence threshold.

```
In [70]: # initialize our lists of detected bounding boxes, confidences, and class IDs, respectively
         boxes = []
         confidences = []
         classIDs = []
```

In [71]:
```
# loop over each of the layer outputs
for output in layerOutputs:
    # loop over each of the detections
    for detection in output:
        # extract the class ID and confidence (i.e., probability) of the current object detection
        scores = detection[5:]
        classID = np.argmax(scores)
        confidence = scores[classID]

        # filter out weak predictions by ensuring the detected
        # probability is greater than the minimum probability (threshold)
        if confidence > CONFIDENCE_THRESHOLD:
            # scale the bounding box coordinates back relative to the size of the image, keeping in mind that YOLO actually
            # returns the center (x, y)-coordinates of the bounding
            # box followed by the boxes' width and height
            box = detection[0:4] * np.array([W, H, W, H])
            (centerX, centerY, width, height) = box.astype("int")

            # use the center (x, y)-coordinates to derive the top and
            # and left corner of the bounding box
            x = int(centerX - (width / 2))
            y = int(centerY - (height / 2))

            # update our list of bounding box coordinates, confidences, and class IDs
            boxes.append([x, y, int(width), int(height)])
            confidences.append(float(confidence))
            classIDs.append(classID)
```

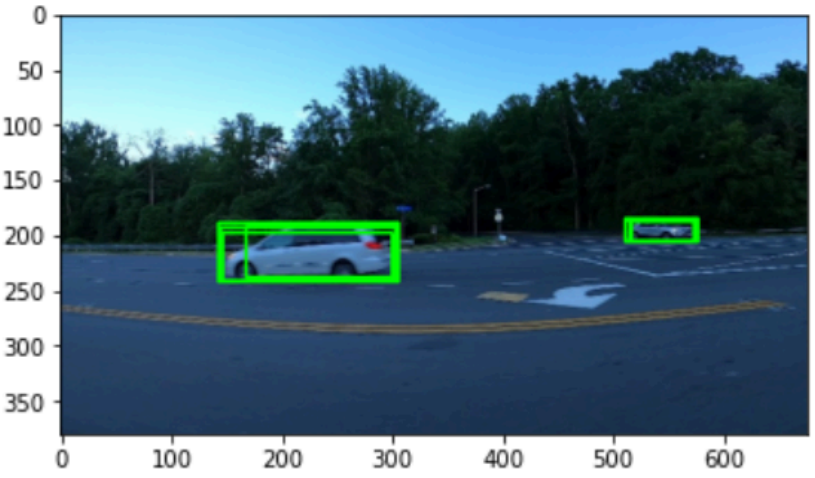
This block of code loops over each anchor box detection, and filters out those bounding box coordinates that are associated with a probability or confidence value greater than the threshold confidence, in this case, 0.3.

Then, the bounding box coordinates are calculated based on their centre coordinate values.

5. After this step, we are left with multiple bounding boxes with confidence greater than the threshold.

In [73]:
```
for i in boxes:
    x,y,w,h=i
    cv2.rectangle(image, (x, y), (x + w, y + h), [0,255,0], 2)
plt.imshow(image[:,:,::-1])
```

Out[73]:
```
<matplotlib.image.AxesImage at 0x1e216ea3c40>
```



To bring this down to one single bounding box for one object, we apply non-maxima suppression.

```
In [24]: # apply non-maxima suppression to suppress weak, overlapping bounding boxes
         idxs = cv2.dnn.NMSBoxes(boxes, confidences, CONFIDENCE_THRESHOLD,CONFIDENCE_THRESHOLD)
```

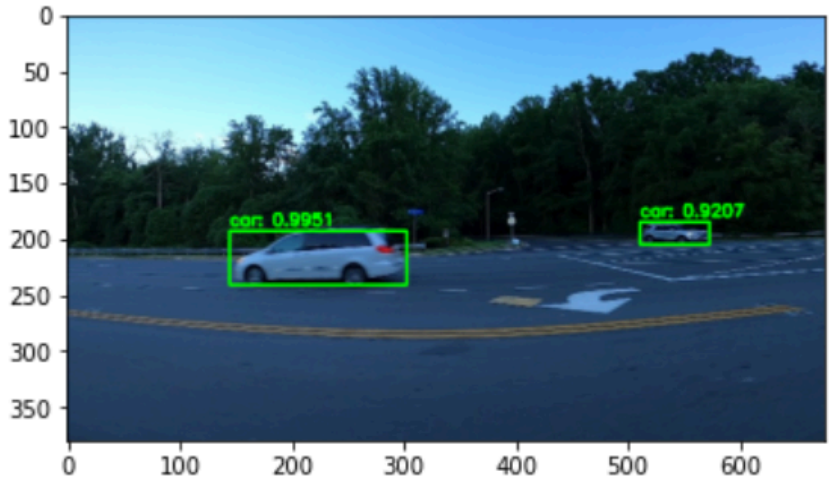
6. After this, we draw the final bounding box output on the image.

```
cv2.rectangle(image, (x, y), (x + w, y + h), [0,255,0], 2)
text = "{}: {:.4f}".format(LABELS[classIDs[i]], confidences[i])
cv2.putText(image, text, (x, y - 5), cv2.FONT_ITALIC, 0.5, [0,255,0], 2)
```

OUTPUT:

The output we obtain is as follows

```
In [26]: import matplotlib.pyplot as plt
         plt.imshow(image[:,:,::-1])
Out[26]: <matplotlib.image.AxesImage at 0x1e21528b790>
```



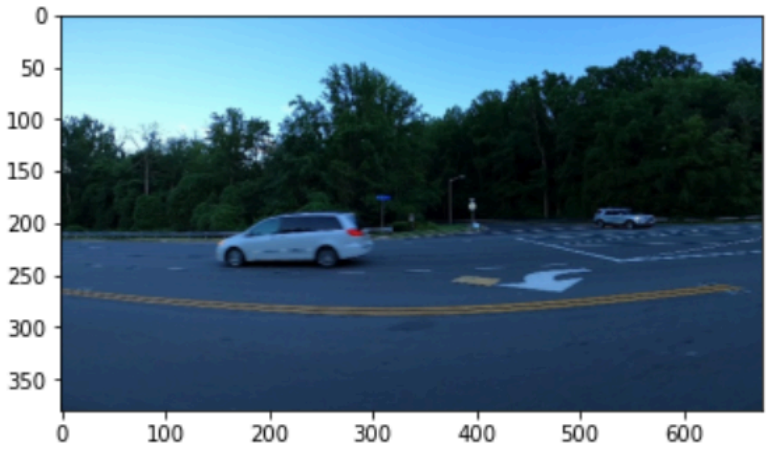
6. RESULTS

Input given:

```
In [62]: #Loading the imput image
         image = cv2.imread(INPUT_FILE)
```

```
In [63]: import matplotlib.pyplot as plt
         plt.imshow(image[:,:,::-1])
```

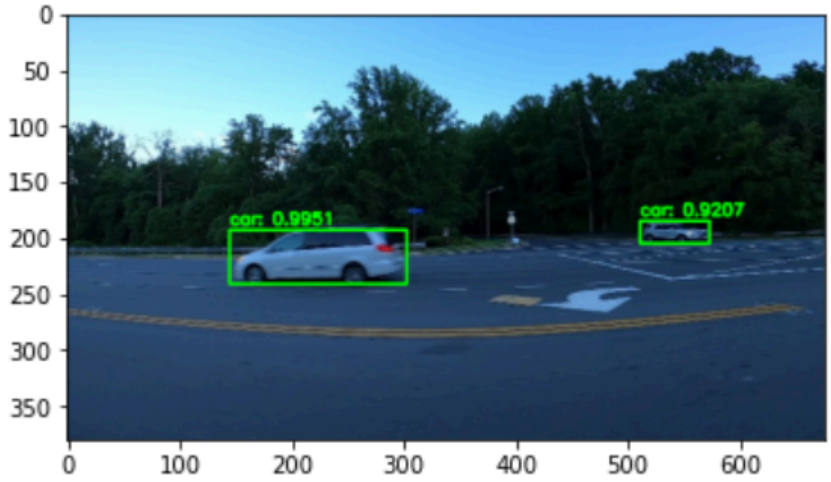
```
Out[63]: <matplotlib.image.AxesImage at 0x1e2167deee0>
```



Output obtained:

```
In [26]: import matplotlib.pyplot as plt
         plt.imshow(image[:,:,::-1])
```

```
Out[26]: <matplotlib.image.AxesImage at 0x1e21528b790>
```



7. CONCLUSION

Object detection is considered the foremost step in deployment of self-driving cars and robotics. In this paper, we demystified the role of deep learning techniques based on CNN for object detection. We proposed a YOLO algorithm for the purpose of detecting objects using a single neural network. This algorithm is generalized, it outperforms different strategies. The algorithm is simple to build and efficient. YOLO accesses the entire image in predicting boundaries. And also it predicts fewer false positives in background areas. Compared to other classifier algorithms this algorithm is much more efficient and fastest algorithm to use in real time.

8. REFERENCES

[1] Zhong-Qiu Zhao, Member, IEEE, Peng Zheng, Shou-tao Xu, and Xindong Wu, Fellow, IEEE
"Object Detection with Deep Learning: A Review "
https://ieeexplore.ieee.org/document/8627998

[2] Ajeet Ram Pathaka, Manjusha Pandeya, Siddharth Rautaraya
"Application of Deep Learning for Object Detection", International Conference on Computational Intelligence and Data Science (ICCIDS 2018)
https://www.sciencedirect.com/science/article/pii/S0386111219301566

[3] Geethapriya. S, N. Duraimurugan, S.P. Chokkalingam
"Real-Time Object Detection with Yolo"
https://www.ijeat.org/wp-content/uploads/papers/v8i3S/C11240283S19.pdf

[4] Gomathy Nayagam Meenakshi Sundaram
"A survey on Real time Object Detection and Tracking Algorithms"
https://www.researchgate.net/publication/274569172_A_survey_on_Real_time_Object_Detection_and_Tracking_Algorithms

[5]https://www.researchgate.net/publication/274569172_A_survey_on_Real_time_Object_Detection_and_Tracking_Algorithms

[6] https://machinelearningmastery.com/how-to-perform-object-detection-with-yolov3-in-keras/

[7] https://www.fritz.ai/object-detection/#:~:text=Object%20detection%20is%20a%20computer,all%20while%20 accurately%20labeling%20them.