Object Detection using Non-Stationary Camera

Amritesh
Department of Computer Science and Engineering
PES University
Bangalore, India
amriteshc101@icloud.com

Mayank Agrawal
Department of Electrical and Electronics
PES University
Bangalore, India
mayank1403@gmail.com

Amityush Amit
Department of Computer Science and Engineering
PES University
Bangalore, India
amityush1923@gmail.com

Dr. Shylaja SS
Department of Computer Science and Engineering
PES University
Bangalore, India
shylaja.sharath@pes.edu

***Abstract* — As technology is advancing and people are trying to find ways to automate tasks that are done on a day-to-day basis, it has been a constant motivation to build self-driving vehicles. A lot of research has already been done in this field and has been quite successful, the previous method of object/vehicle detection takes input from various sensors and tries to detect vehicles present in front. This paper focuses on detecting vehicles using only video inputs and time-efficient algorithms that can provide spontaneous outputs. We have used the You Only Look Once (version 3) approach for achieving our goal, the model is trained on COCO 2017 dataset. The COCO dataset was chosen because it has 80 classes and is very precise and gives the model a good amount of data to train on. We were able to achieve good results and very few false positives. The model is not only able to detect cars, rather it can detect most of the things that can be found on the road (like trucks, bicycles, traffic lights, etc).**

Keywords — vehicle detection, image localization, image recognition, you only look once approach, real-time object detection

I. Introduction

Object Detection, a computer vision technique is a task of detecting object instances. These objects belong to a certain class, for example, cars, humans, etc. Being an actively researched area of computer vision, object detection first identifies and then locates the objects. Object detection can be used to track, count, or accurately label objects depending on the type of identification and localization used.

A simple example for an object to be identified can be an image of a cat and a person. Here, the classes are cat and human. Earlier, object detection was incapable of detecting multiple objects in one image/frame. With the advancement and introduction of new algorithms, the current state-of-the-art models can detect multiple objects in one image/frame.

The biggest need for object detection in the field of AI is that it helps the computer visualize the world as we humans do which denotes that the computer can make proper decisions on these detected objects. The world of object detection is governed by two steps which are identification and localization.

Object detection is very different from image recognition, for example, in image recognition, an image containing a picture of a dog is labeled as “dog” and with two dogs in one image is still labeled as “dog”. Whereas, in object detection, each dog will be detected and a box will be drawn with labels as “dog” around each dog. Object detection is responsible for predicting where each object is and what class the label should have, whereas image recognition predicts what object is there in the image. Thus, this shows that object detection provides more accurate information than image recognition.

Object detection is different from other computer vision tasks like image recognition and image segmentation, with the latter creating a pixel-level understanding of a scene’s elements, because of its capability of locating objects in an image as well as a video which further aids us in counting them and tracking them.

With the idea that object detection can work like a human eye, which when combined with the memory and processing power of computers, can provide us with some great applications of object detection. Some of these include video surveillance, crowd counting, anomaly detection as well as self-driving cars.

One of the key technologies in the advancement of the advanced driver assistance system (ADAS) is object detection. It detects driving lanes by detecting cars and works on pedestrian detection, which provides us with safer autonomous vehicles on the roads.

The domain of self-driving cars requires a simple camera attached to it that can take inputs. Inputs can be taken through LiDar sensors, Sonars as well as GPS. The idea that the computer can take a simple video as an input and then do some calculations on it and provide us with results is still novel in the field. Today, the stationary cameras (adjusted at a fixed position) can be used to take inputs but with non-stationary cameras, many factors start affecting the input feed. The domain of stationary cameras being used to do predictions in the field of computer vision is well researched. The new research in this field lies in the input being taken from a non-stationary camera.

A variety of algorithms have been used in this field. Machine Learning models can be created from scratch which will include setting up your datasets and extracting features which are then forwarded to custom-built layers and weights in CNN. With the introduction of deep learning-based approaches, such as R-CNN or Yolo which

are based on Convolution Neural Network, models can automatically learn from images.

One of the major algorithms in the domain of Object Detection is Yolo (You Only Look Once), an accurate framework and is supremely fast. Yolo works using already existing CNN techniques. Pre-trained and tested originally on the Pascal VOC dataset, Yolo has proved to be very efficient as well as very accurate. With newer datasets and pre-processed inputs, it can be made much more efficient.

Yolo is preferred over the other CNN models for object detection because it is a very fast algorithm, as fast as 45 frames per second. Its accuracy is almost similar to R-CNN models but R-CNN models are slower. R-CNN is a great method but it doesn't look at the whole input at once but rather localizes the parts that have the objects in them. Yolo follows a very different approach, it works on the whole input in one instance. In that very instance, Yolo predicts the bounding boxes as well as the classes.

To work on the whole input in one instance, Yolo divides the whole frame into smaller grids. It then predicts whether a particular object is present in each grid. The output is a simple bounding box dimension covered in that grid as well as the probability of each class in that grid.

For more accurate predictions, Yolo needs a better dataset to train on. One of the most popular datasets in the domain is the COCO 2017 Dataset with 80 classes for detection. For more accurate predictions, an increment in the number of grids can be done. For better results, we can use intersection over union and non-max suppression.

Yolo with the COCO dataset gives great results because the COCO dataset is a versatile dataset with classes up to 80 which gives really good predictions. Yolo is now being used in a lot of object-detection applications because of its reliability and speed.

With its high accuracy and run in real-time scenarios, Yolo has become popular. Yolo does real-time object detection and is a clever CNN. Yolo is trained to learn the generalized representation of common objects, which helps it outperform other CNN models. With 106 layers and no pooling layers which helps in downsampling the feature maps, Yolo has outperformed other models.

II. Related Work

Extensively talking, object location can be separated into AI-based methodologies and profound learning-based methodologies.

In more conventional ML-based methodologies, computer vision procedures are utilized to take a gander at different highlights of a picture, like the shading histogram or edges, to recognize gatherings of pixels that may have a place with an article. These highlights are then taken care of into a relapse model that predicts the area of the item alongside its mark.

Then again, profound learning-based methodologies utilize convolutional neural organizations (CNNs) to perform start to finish, unaided item recognition, in which highlights don't should be characterized and removed independently.

A. HOG Feature Extraction

HOG (which stands for Histogram of oriented Gradients) is a feature extracting tool, which is generally used in computer vision tasks for detecting objects. This feature has been used earlier for several different types of images for various feature extraction. This method emphasizes the structure or the shape of the object and predicts the object accordingly.

HOG feature extractor not only detects if the pixel is an edge or not, but rather it also provides us with the edge directions. Edge direction is predicted by breaking and extracting the gradient and the orientation of the edge provided. The orientation of the edge is determined in local portions of the image. The entire image is divided into smaller parts/regions and for each part, the orientation and the gradient are determined. Using these values, a histogram of each region is generated separately.

B. Convolutional Neural Network

Lately, profound models have been demonstrated to be more exact for characterization and identification across practically all item types. Utilizing Convolutional Neural Networks (CNNs) has empowered an information-driven way to deal with identification, limiting the work to configuration highlights, model items, and the need to depend on extra sensors. This method of convolution Neural Network provides good output but at the same time takes too much time.

Convolutional neural organizations are utilized to distinguish a vehicle. Methods like element combination are utilized to connect undeniable level highlights and low-level highlights and identify various sizes of vehicles on various highlights. To improve speed, we normally embrace completely convolutional engineering rather than completely associated layers.

CNN-based methodologies have shown rich portrayal influence and accomplished promising outcomes. R-CNN utilizes object propositions created by a specific hunt to prepare CNN for identification undertakings. Under the R-CNN system, SPP-Net and Fast R-CNN accelerate through creating area propositions on the map; these methodologies just need a PC once. Quicker R-CNN utilizes a locale proposition network rather than specific pursuit, at that point, it can prepare, start to finish and the speed and precision likewise improve. R-FCN attempts to lessen the calculation time with position-touchy score maps.

C. Region-Based Convolutional Neural Networks

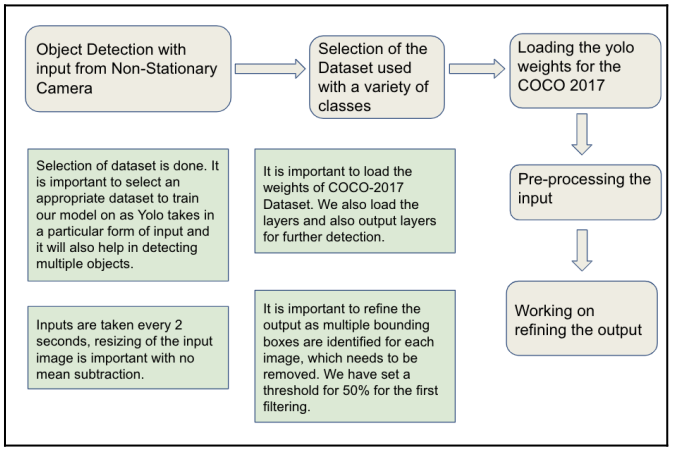
Region-based Convolutional Neural Networks, commonly known as R-CNN, is a group of AI models for computer vision and explicitly object identification. The R-CNN paper distributed in 2014, was the primary paper to show that CNN can prompt the elite in object discovery.

The technique accepts a picture as info and concentrates around 2000 (for say) district propositions from the picture. Each suggestion is then reshaped to a fixed size to be given as a commitment to the CNN.

The CNN removes a fixed-length highlight vector for every locale proposition. These highlights are utilized to group district propositions utilizing classification explicit direct SVM. The bounding boxes are refined utilizing jumping box relapse so the item is appropriately caught by the container. Even though R-CNN is acceptable at recognizing objects, it

has its inadequacies. This calculation is moderate and it takes around 47 secs to perform object discovery on a picture. Preparing isn't done in a solitary advance. There are various models for doing various parts which make the preparation interaction tedious.

III. METHODOLOGY



The methodology consists of three main parts: (1) loading the weights of the Yolo v3 trained on COCO Dataset and working on the output layers, (2) pre-processing the input that is provided through the model, (3) Refining the output and applying intersection over union and non-max suppression. The following subsections explain the same in detail.

A. Loading the weights of Yolo v3 trained on the COCO dataset and extracting the output layers

The Yolo version used is the Yolo v3. The training of the model was done on the COCO 2017 dataset. The COCO dataset was chosen because it has 80 classes and is very precise and gives the model a good amount of data to train on. A custom dataset was also planned to be used but the COCO dataset performed better and Yolo takes input for training in a particular format that the COCO dataset supports. Yolo requires weights for it to train on. COCO-dataset weights were added to the project.

In this part itself, we extract all the layers of the Neural network. Moreover, it is important to get all the unconnected layers as well. These unconnected layers are the output layers in the Yolo algorithm. The Yolo v3 model has 3 output layers (82, 94, 106). These layers will provide us with the output.

```
with open("./coco.names", 'r') as f:
    classes = [line.strip() for line in f.readlines()]
layers_name = net.getLayerNames()
output_layers = [layers_name[i[0]-1] for i in
net.getUnconnectedOutLayers()]
colors = np.random.uniform(0, 255, size=(len(classes), 3))
```

This helps us load all the layers of the Yolo in our project and gives us all the layers' names.

B. Pre-processing the input provided

As an assumption, we assume that each consecutive frame won't have a lot of changes or information, and thus, we try to reduce the load on the system by taking frames at certain time intervals. The current value is every 2 seconds one frame is collected and analyzed. This can be changed according to the usage.

Yolo can accept frames in all dimensions but it becomes a huge problem while training if the dimensions are not consistent. Thus, it is important to resize the image into a particular dimension.

Each version of Yolo has a different input image size. Weights of the trained model can be chosen accordingly. The input size used in the project is 320x320. It is important to perform mean subtraction on the image which helps combat illumination changes, hence becoming a major step in pre-processing for CNN models.

```
blob = cv2.dnn.blobFromImage(img, scalefactor =
0.00392, size = (320,320), mean = (0,0,0), swapRB =
True, crop = False)
net.setInput(blob)
outputs = net.forward(output_layers)
```

As mentioned, the Yolo model used requires 320x320 sized input images, so resizing is done. It is also important to scale the image after performing mean subtraction. The scale factor set is a general standard scale factor that is used by the industry. As OpenCV accepts input in the BGR channel and means the subtraction is performed in the RGB channel, to remove this discrepancy we swap R and B channels. This pre-processed input is then passed on to the CNN layers of YOLO.

C. Refining the output

Every Object detection model will have multiple detections for the same object with varying confidences. Yolo is no different. It is now important to decide on the class the object detected belongs to, which is provided by Yolo using the probabilities it provided in the output. The class with the highest probability is the class that the object belongs to. Yolo will also detect multiple bounding boxes for the same object. To avoid this we apply multiple methods. First of all, a threshold of 50% is set to allow only those outputs which has a confidence level higher than 50%.

```
scores = detect[5:]
class_id = np.argmax(scores)
conf = scores[class_id]
if(conf>0.5):
    center_x = int(detect[0]*width)
    center_y = int(detect[1]*height)
    w = int(detect[2]*width)
    h = int(detect[3]*height)
    x = int(center_x - w/2)
    y = int(center_y - h/2)
```

There are multiple bounding boxes still present for the same object. To avoid this, Intersection over Union and Non-Max-Suppression is performed. Intersection over Union (IOU), an evaluation metric provides us to compare how our model works against the provided dataset. It is the ratio of the area common between the predicted bounding

box and the actual bounding box to their union. To ideally capture the best bounding box that denotes the object, non-max-suppression is used. For non-max-suppression, (1) the confidence level of each box is required, (2) the IOU value is required. The highest confidence interval box is selected at first, then the process of IOU is performed to differentiate between different objects, the boxes with high IOU values will be discarded as there is a very high chance that these bounding boxes are duplicate ones. The IOU value is generally set to 40%. This provides us with one single box for each object detected.

IV. Results

The model was tested on a variety of videos, and the results were images with bounding boxes around the objects. To check the preciseness of the model, we judged each frame from each video and gathered four information from each frame,

- Total Objects in the video
- Number of objects detected
- Number of true positives
- The number of false positives.

Using the information gathered, we calculated the accuracy of the model.

The videos selected to test the model's accuracy were standard videos from the internet. The videos are taken with non-stationary cameras placed on a car dashboard. The output gathered from the model is as follows.

With the tests, the information gathered was 88.52% of objects were detected with 82.29% being true positive and 6.22% being a false positive.

The multiple false positives found by the model was because of -

- The input frame from the video is not very clear which results in false positives.
- More objects are not being detected because of the lack of a training dataset.
- Many false positives are detected because the training dataset has not covered those edge cases.

The project was focused on detecting vehicles, but because of the advanced COCO Dataset used in the model, the model is capable of detecting other objects such as traffic lights and pedestrians. The accuracy of detecting traffic lights is 44.44%. The low accuracy is because the number of training datasets is quite low.

Detected Objects	88.52%
True Positives	82.29%
False Positives	6.22%

V. Future Work

Various transformations, tests, and examinations have been left for the future because of the absence of time and accessible calculation power. There can be many more additions to the model that we have used and explained above. A few of the ideas that we think would be a great addition are -

- Traffic light detection, our model currently detects if there is a traffic light present in the image/frame. Detecting the color of traffic signals and making decisions based on that would certainly increase the usage.
- Decision-making based on pedestrians/roadblocks. The model detects humans or other factors that might appear on the road, leading to some sort of unwanted event. Using these predictions important decisions can be made.
- Traffic sign detection, the model currently ignores the traffic signs that appear on the path like speed limit, bump ahead, go slow sign, etc. These things can also be detected by doing some minor changes. Adding these changes will improve the decisions that can be made.
- Modifications can be made in the model to track the detected vehicles in the frame. Due to lack of computation power we took each frame at an interval of 2 seconds, if the video is taken continuously and some changes are made, then the model can be used to track vehicles as well.

The utilization of different kinds of individual portrayals and wellness capacities could be researched since they affect the outcomes towards the end.

VI. Conclusion

Taking into account significant learning and convolution, this report uses YOLOv3 to set up the object detection model and improve acknowledgment precision. Object detection applications in areas, for example, media, retail, producing, advanced mechanics, and so forth need models to be quick. In any case, YOLOv3 is additionally exact. This makes it the best model to pick in this kind of use where speed is significant in light of the fact that the items should be progressively or on the grounds that the information is excessively huge. Applications like autonomous vehicles or self-driving cars require object detection with very high precision and image localization in very little time. YOLOv3 is a good object detection model, at least for now.

References

[1] Vermesan, Ovidiu, and Joël Bacquet, eds. Cognitive Hyperconnected Digital Transformation: Internet of Things Intelligence Evolution. River Publishers, 2017.

[2] Dimiccoli, Mariella. "Computer Vision for Egocentric (First-Person) Vision." In Computer Vision for Assistive Healthcare, pp. 183-210. Academic Press, 2018.

[3] Sathi, Arvind. "Cognitive Things in an Organization." In Cognitive (Internet of) Things, pp. 41-59. Palgrave Macmillan, New York, 2016.

[4] Berg, Alexander C., and Jitendra Malik. "Shape matching and object recognition." In Toward Category-Level Object Recognition, pp. 483-507. Springer, Berlin, Heidelberg, 2006.

[5] Ketkar, Nikhil. "Training Deep Learning Models." In Deep Learning with Python, pp. 215-222. Apress, Berkeley, CA, 2017.

[6] Zhang, F. K., Feng Yang, and Ce Li. "Fast vehicle detection method based on improved YOLOv3." Computer Engineering and Applications 55, no. 02 (2019): 12-20.

[7] Sun, Zehang, Ronald Miller, George Bebis, and David DiMeo. "A real-time precrash vehicle detection system." In Sixth IEEE Workshop on Applications of Computer Vision, 2002.(WACV 2002). Proceedings., pp. 171-176. IEEE, 2002.

[8] Yan, Gang, Ming Yu, Yang Yu, and Longfei Fan. "Real-time vehicle detection using histograms of oriented gradients and AdaBoost classification." Optik 127, no. 19 (2016): 7941-7951.

[9] Chen, Baisheng, Yunqi Lei, and Wangwei Li. "A novel background model for real-time vehicle detection." In Proceedings 7th International Conference on Signal Processing, 2004. Proceedings. ICSP'04. 2004., vol. 2, pp. 1276-1279. IEEE, 2004.

[10] Zhang, Xinyu, Hongbo Gao, Chong Xue, Jianhui Zhao, and Yuchao Liu. "Real-time vehicle detection and tracking using improved histogram of gradient features and Kalman filters." International Journal of Advanced Robotic Systems 15, no. 1 (2018): 1729881417749949.

[11] Chen, Shaobin, and Wei Lin. "Embedded system real-time vehicle detection based on improved YOLO network." In 2019 IEEE 3rd Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC), pp. 1400-1403. IEEE, 2019.

[12] Bimbraw, Keshav. "Autonomous cars: Past, present and future a review of the developments in the last century, the present scenario and the expected future of autonomous vehicle technology." In 2015 12th international conference on informatics in control, automation and robotics (ICINCO), vol. 1, pp. 191-198. IEEE, 2015.

[13] Hu, Hou-Ning, Qi-Zhi Cai, Dequan Wang, Ji Lin, Min Sun, Philipp Krahenbuhl, Trevor Darrell, and Fisher Yu. "Joint monocular 3D vehicle detection and tracking." In Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 5390-5399. 2019.

[14] Chen, Xiaozhi, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. "Multi-view 3d object detection network for autonomous driving." In Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, pp. 1907-1915. 2017.