Object Detection in Real Time

Avijit Bhattacharjee
Roll No:1803141003

June 2020

1 Introduction

Being human we can locate and detect objects in an image quickly and accurately. As our visual system is agile and authentic we can do complicated tasks like riding bikes with meager conscious. In this project I have used yolov3 where when we give image as input then as output we get that image appended with bounding boxes around the objects and class probabilities corresponding to that. As in this detector detection process is considered as a regression problem this detector is fast enough. Moreover, yolov3 looks at the complete image while training and testing therefore this conceal environmental knowledge in conjunction with the objects actualization.
In this project I have made a real time object detection system for detecting persons in an image or video. I have taken the pretrained weights of darknet53 to train the convolutional neural network for this purpose. As I am using yolov3 therefore, this detector uses multi-scale training method. Here I have used an approach which is mainly dependent on deep learning. The network for this problem has been trained on images taken from open images data set.

2 Brief Background about the problem

In this section I have explained the background of object detection in a nutshell.

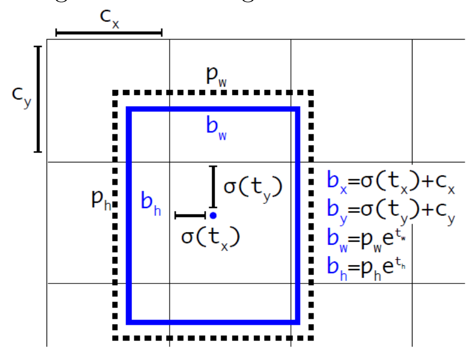
2.0.1 Predicting Bounding Box

Each bounding box gets four coordinate t_x, t_y, t_h, t_w predictions.
let c_x, c_y be the offset from the upper left corner point of the image.
let p_w, p_h be the width and height of the prior bounding box then one can the predictions from the formulas below ,

$$b_x = \sigma(t_x) + c_x; b_y = \sigma(t_y) + c_y; b_w = p_w e^{t_w}; b_h = p_h e^{t_h} \tag{1}$$

During training sumsquared error loss function has been used for the above four parameters. Moreover Logistic regression was used in each bounding box to predict objectness score in them. If the prior bounding box gets more intersection

Figure 1: bounding box coordinates



over union with the ground truth object then for that box objectness score is one.

2.0.2 Class Prediction

The algorithm predicts so many bounding boxes with multiclass prediction. While predicting the multiclass classification report the authors have used logistic classifiers independently instead of using softmax. This kind of multilebel classifer is helpful in problems where we have one label overlapping the other one.

2.0.3 Predictions Across scales

Concepts simmiler to feature pyramid was used to extract features from 3 different scales. Diversified features were used to make better predictions. Features were diversified by mapping features from n-2 th layer to n th layer by residual connections. Priors for the Bounding boxes were selected using K means clustering. They have divided 9 clusters into 3 scales.

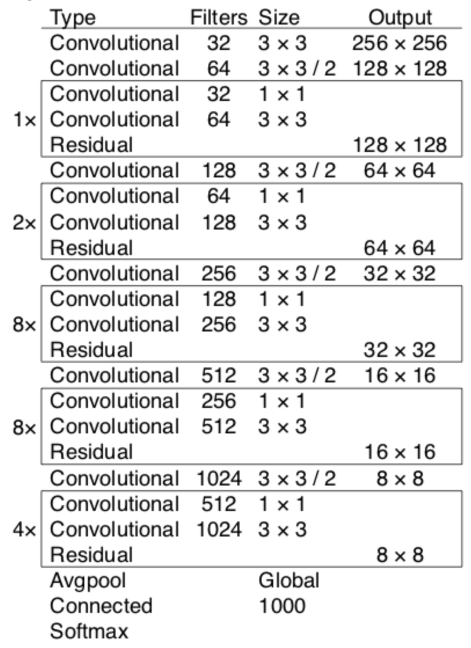
2.0.4 Feature Extractor

A new network was used by the authors in YOLOv3. They have used a mixed cocept of residual network, Darknet-19, YOLOv2. A succession of 3×3 and 1×1 have been used in this feature extractor with some shortcut connections, making this a 53 convolution layer network .

3 Work done

I have used one toolkit[1] that allows us to download any number images from any classes from the Open Images website. After that I have written a python code to convert labels and annotations from the format that the toolkit generates into the format that YOLOv3 takes in for its training. When we download the

Figure 2: darknet53 network architecture



	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

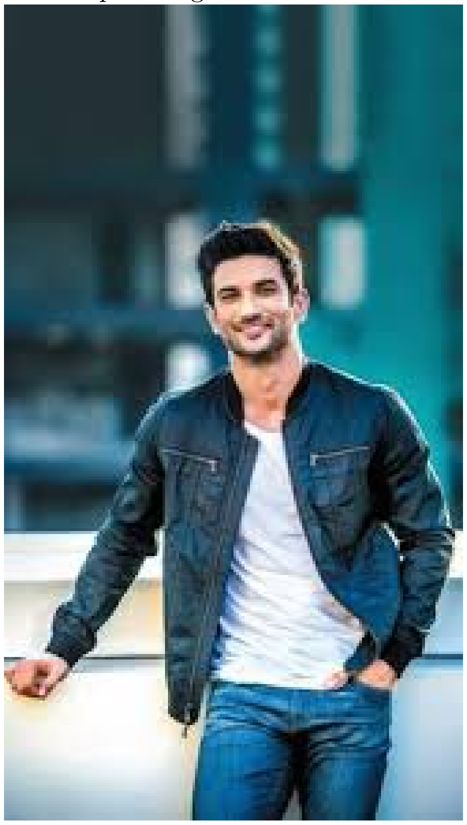
data from Open Images website we get top left and bottom right coordinates of a bounding box which is not allowed for the training of YOLOv3. This allows me to use the data I have downloaded to be trained by YOLOv3. After this I have cloned the darknet network from AlexeyAB's repository. Then from darknet I have downloaded the yolov3.cfg file and customized that according to my training requirements. I have changed max batches to 4000. As in Alexy's github profile it is mentioned that max batches should be (classes $\times$ 2000) but not less than 4000. Moreover, steps has been changed to 80 and 90 percent of max batches. Also I have used (number of classes $+5) \times 3$ formula to select the number of filters in the convolutional layers. After this I have created a custom obj files in the data folder in darknet which helped me in custom object detection. obj.names has the name of the classes in same order as used while downloading and obj.data is the file referring where all the data is.

4 Details of the Data used for performing experiments

After forking the toolkit for downloading the data sets [1] I have downloaded the Person classes images with labels and annotations from the following link.

`https://storage.googleapis.com/openimages/web/visualizer/index.html?set=train&type=detection&c=%2Fm%2F01g317`.

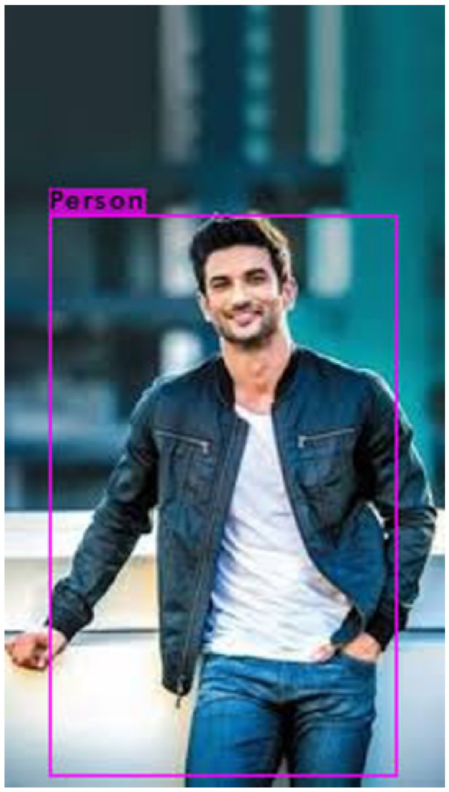
Figure 3: input image for the custom detector



5 Results and Analysis

I have set max steps in the custom configuration file as 4000 but as we can see in the fig 5 that the training losses started to saturate after 1800 iterations. Therefore to avoid overfitting I have stopped training the detector after 2100 iterations. Using these weights I have applied this custom detector on fig 3 and got the output as fig 4. It predicted the out put with 85 percent of accuracy in 91 millisecond with just 2GB of GPU provided in google colab. As can be seen from fig 5 the custom detector is getting losses below one after 2100 iterations which is considerable for use. I have also used a short video clip captured on this Holi in our campus for detection scheme. From the output video it is quite clear that this custom detector is able to detect Persons in background. Even there were so many people standing close to each other and some were playing football, this custom detector is able to detect every person accurately.

Figure 4: output image from the custom detector



6 conclusion

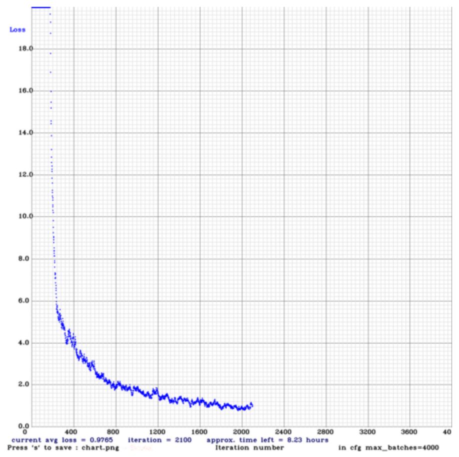
In this project I have built a competent object detector which achieves comparable results with the current state-of-the art detection systems[2, 3, 4].
In future I would like to use this kind of custom detector with different convolutional neural networks for human detection using Closed-circuit television.

References

[1] A. Vittorio, "Toolkit to download and visualize single or multiple classes from the huge open images v4 dataset," https://github.com/EscVM/OIDv4_ToolKit, 2018.

[2] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

Figure 5: training losses vs number of iterations



[3] J. Redmon and A. Farhadi, "Yolo9000: better, faster, stronger," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7263–7271.

[4] ——, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.