

Real Time Object Detection Using YOLO-V3

Abstract:	2
Introduction:	2
YOLO:	2
What are YOLO Versions?	3
YOLO-V1:	3
YOLO-V2:	3
What's new in YOLO-V3?	4
Project Architecture(YOLO V3 - Architecture):	4
Steps for object Detection using YOLO v3:	4
Architecture:	5
Loss Function/Evaluation Metrics of YOLO-V3:	6
IOU Loss:	6
MAP(Mean Average Precision):	7
Run the model:	7
Summary and Analysis:	8

Abstract:

Object detection is a technique that entails the classification and localisation of objects. It's a model that's been trained to recognise the presence and location of a variety of things. This may be used for both static photos and videos in real-time. To evaluate the exciting progress of object detection technique on autonomous driving, we created BDD100K, the largest driving video dataset with 100K videos and 10 tasks. Geographic, environmental, and weather diversity are all present in the dataset, which is important for training models that are less likely to be startled by unexpected situations. We are considering YOLO(You-Only-Look-Once)-V3 to detect objects in the images and real time video. Our findings reveal that existing models require particular training procedures to perform such diverse tasks.

Introduction:

Due to its vast use and technological progress, object detection has been increasingly popular in recent years. This task has been investigated in a variety of academic and practical contexts, including surveillance safety, autonomous driving, traffic monitoring, and robot vision. Object detection technology has been fast evolving with the advancement of deep convolutional neural networks and the rise of GPU processing speed. Object detection finds the object, classifies it into several categories, and then localises it by drawing bounding boxes around it.

YOLO:

YOLO is a recent invention in the world of object detection. The RCNN series framework has a number of drawbacks, including the fact that the entire network cannot perform end-to-end training, the intermediate training procedure necessitates a large amount of memory to retain some variables, and the computation speed is slow, among others. The YOLO algorithm proposes a novel approach that converts the object detection problem into a regression problem. It converts the target bounding box and its classified categories in numerous locations of images while producing the input image.



What are YOLO Versions?

There are three main variations of YOLO, they are YOLOv1, YOLOv2, and YOLOv3.

YOLO-V1:

Yolo-V1 was the first time the concept of a single-stage detector was seen. Batch normalisation (BN) and leaky ReLU activations, both of which were relatively new at the time, were used in the architecture. It struggles to detect close objects as each grid can propose only 2 bounding boxes. It has bad performance in localization of boxes as bounding boxes are learning totally from data.

YOLO-V2:

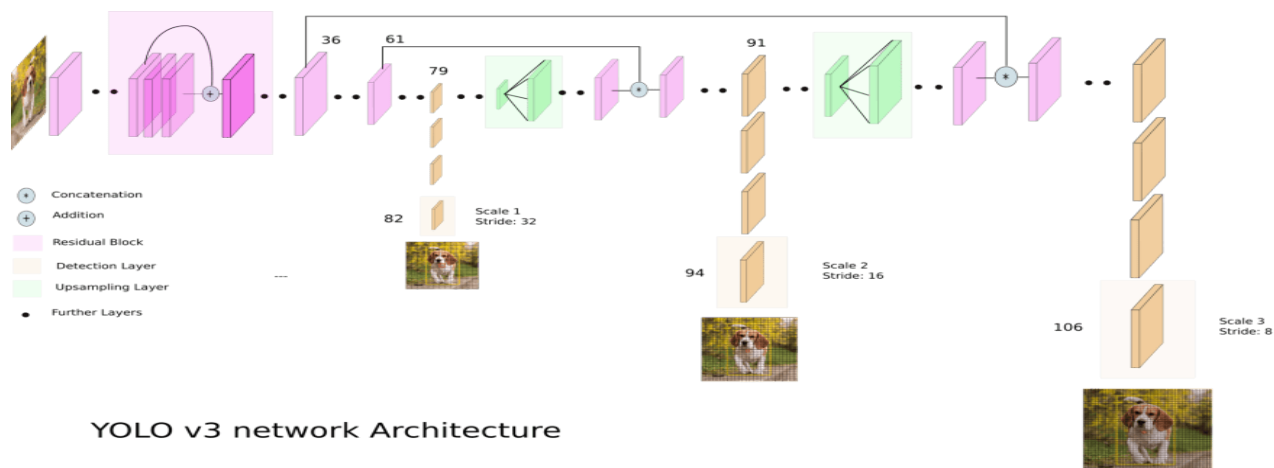
In Yolo-V2 fully connected layer has been removed at the end. This made the architecture fully resolution-independent. It is not suitable for small objects as each grid can only detect one object. It used DarkNet-19 as the model architecture. It consists of 19 layers deep.

What's new in YOLO-V3?

You Only Look Once, Version 3 (YOLOv3) is a real-time object detection system that recognises specific things in films, live feeds, and photos. To detect an object, YOLO employs features learned by a deep convolutional neural network. It is built with Darknet-53 which contains 52 convolutions. It contains skip connections like ResNet and 3 prediction heads like FPN each processing image at a different spatial compression.

This uses a non-max suppression technique which makes sure that the object detection algorithm only detects each object once and it discards any false detections, it then gives out the recognized objects along with the bounding boxes.

Project Architecture(YOLO V3 - Architecture):



Steps for object Detection using YOLO v3:

1. The input is batch of images shape of $(m, 416, 416, 3)$
2. YoloV3 passes the input image into Convolutional neural network(CNN)
3. We get output with a list of bounding boxes along with recognized classes. Each bounding box is represented by 6 numbers (pc, bx, by, bh, bw, c) . If we expand c into an 80-dimensional vector, each bounding box is represented by 85 numbers.
4. Finally, we perform the IoU (Intersection over Union) and Non-Max Suppression to avoid selecting overlapping boxes.

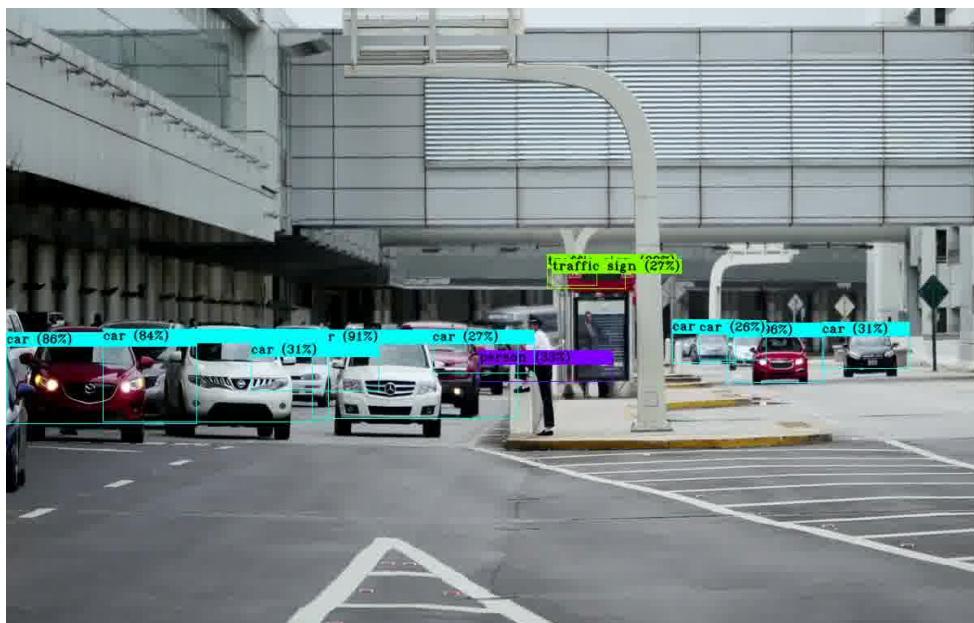
Architecture:

1. It has a variant of Darknet, which has a 53 layer network trained on ImageNet.
2. For detection task, more 53 layers are included giving us total 106 layer fully convolutional underlying architecture
3. Here detection is carried out by using 1×1 detection kernels on feature maps of three different sizes at three different locations throughout the network.
4. The detection kernel is $1 \times 1 \times (B \times (5 + C))$. B is for the amount of bounding boxes a feature map cell can predict, "5" stands for the four bounding box attributes and one object confidence, and C stands for the number of classes.
5. It uses binary cross-entropy for calculating the loss for each label
6. Hyper Parameters used are class_threshold, Non-max suppression threshold, input_shape and input_height

It is basically a three scale detection algorithm i.e., detection takes place at the following layers:

1. The 82nd layer is the first. The stride of the 81st tier is 32. If the image is 416 by 416 pixels, we will get a 13×13 feature map. Our 1×1 detection kernel produces a detection feature map of $13 \times 13 \times 255$ for each detection.
2. Second, at the 94th layer where we obtain a detection feature map of $26 \times 26 \times 255$.
3. Third, at the 106th layer where we obtain a detection feature map of $52 \times 52 \times 255$.

YOLOv3 is capable of detecting at three sizes due to the above three different detection feature maps. The first map of 16×16 is used for detection of large objects, 26×26 for medium objects and 52×52 for small objects. It is also very accurate to detect very small objects.



Here from the figure also we can observe that our detection algorithm is capable of detecting images at very large size and even at accurate size

Loss Function/Evaluation Metrics of YOLO-V3:

YOLO-V3 Loss function contains three parts:


1. Loss of Bounding Box Coordinates
2. IOU Error loss
3. MAP

Bounding Box Coordinates loss is evaluated by Euclidean Loss which computes the square root of the difference of the bounding box parameters squared. This doesn't consider area of the overlapping bounding box includes only the difference of the parameters left, right, top, bottom

IOU Loss:

IOU Error loss works only when the predicted bounding boxes overlap with the ground truth box. IOU loss would not provide any moving gradient for non-overlapping cases. It is simply an evaluation metric which is used to measure accuracy of an object detector on a particular dataset.

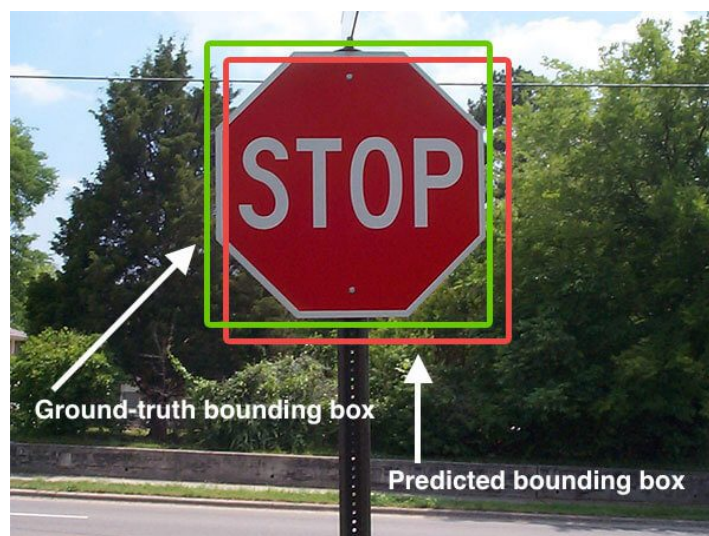
IOU can be determined via:

$$IoU = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$


To apply IOU to evaluate an object detector we need:

1. The ground-truth bounding boxes (i.e., the hand labeled bounding boxes from the testing set that specify where in the image our object is).
2. The predicted bounding boxes from our model.

Below is an visual example of a ground-truth bounding box versus a predicted bounding box



In the below figure I have included examples of good and bad Intersection over Union scores.

Predicted bounding boxes that heavily overlap with the ground-truth bounding boxes have higher scores than those with less overlap. This makes Intersection over Union an excellent metric for evaluating custom object detectors



MAP(Mean Average Precision):

MAP compares the ground truth bounding box to the detected box and returns a score. The higher the score the more accurate the model will be in its detections.

Run the model:

You can run and evaluate the model by following the commands mentioned in code folder of project github folder

Summary and Analysis:

Object detection is an important and challenging problem in computer vision, which has been widely concerned by people. With the advancement of deep learning technology, great changes have taken place in the field of target detection. We introduced BDD100 data detection with Yolo-V3. Yolo-V3 is a good detector. Its fast and it's accurate.

References

1. Girshick, Ross. 2016. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. arXiv:1506.01497v3.
2. Redmon, Joseph. 2018, *YOLOv3: An Incremental Improvement*, <https://arxiv.org/abs/1804.02767>