Chapter 46
An Inter-Comparative Survey on State-of-the-Art Detectors—R-CNN, YOLO, and SSD

B. Bhavya Sree, V. Yashwanth Bharadwaj, and N. Neelima

Abstract In recent years, breakthrough enhancements in computer hardware and supercomputers made object detection a significant topic of research. Accurate object detection models are computationally expensive and are inefficient on simpler and limited configuration settings while faster models achieve real-time speed, work well on simpler configurations but fail to be accurate. There is always a trade-off between speed and accuracy. There is no clear-cut answer on which detector performs the best. The user will have to make a choice based on the requirement. This paper aims at analyzing numerous CNN-based object detection algorithms—R-CNN, Fast R-CNN, Faster R-CNN, You Only Look Once (YOLO), and Single Shot MutliBox Detector (SSD)—and make comparisons concerning performance, precision and speed and state as to which algorithm performs better under certain constraints. This enables the user to pick an object detector of his/her choice that better addresses the demands of an application.

46.1 Introduction

It is trivial for a human eye to distinguish, recognize, and classify objects in its view. However, it is hard for a machine to comprehend objects in real-world scenarios because they are highly adaptable and take in a variety of shapes, sizes, colors, and textures. Recent developments in computer vision and image processing, however, simplified the task of object detection. Object detection and tracking technology are

B. Bhavya Sree (✉) · V. Yashwanth Bharadwaj · N. Neelima
Amrita School of Engineering, Amrita Vishwa Vidyapeetham, Bengaluru, India
e-mail: bhavyasreebuddha@gmail.com

V. Yashwanth Bharadwaj
e-mail: yashwanthbharadwaj2@gmail.com

N. Neelima
e-mail: neelima.niz@gmail.com

A. N. R. Reddy et al. (eds.), *Intelligent Manufacturing and Energy Sustainability*,
Smart Innovation, Systems and Technologies 213,
https://doi.org/10.1007/978-981-33-4443-3_46

quite effective and finds numerous applications in self-driving cars, medical diagnosis, ball tracking in sports, video surveillance systems [1], etc. Object detection algorithms identify objects in a digital image or a video frame and fit a bounding box around with a label stating as to which class the object belongs. However, some objects may go undetected by sensors and this may be critical in the case of autonomous vehicles as they are to be working with 100 percent accuracy. For instance, there has been a case of death relating to a self-driving vehicle. Failing to sense, An Uber self-driving vehicle hit a pedestrian. Perception is, therefore, a life-or-death issue and cares for a lot more attention.

The advancements in deep learning and neural networks led to the discovery of state-of-the-art models like R-CNN, YOLO, SSD, etc. The fundamental duty of these detectors is to generate bounding boxes, estimate class probabilities, and assign a confidence score based on these probabilities. This paper gives an overview of how various object detection algorithms—R-CNN, Fast R-CNN, Faster R-CNN, YOLOv1, YOLOv2, YOLOv3, and SSD—work and differ on various grounds. These models are evaluated based on the mean Average Precision (mAP), test time, and memory specifications. The model that best fits the demands of your application can, henceforth, be selected and used in accordance.

46.2 Literature Review

46.2.1 R-CNN

Region-based CNN (R-CNN) [2, 3] is one of the various CNN-based object detection methods. To perform object detection, we need to know the class to which an object belongs, coordinates, and offset values of a bounding box. In R-CNN, a selective search (SS) algorithm employs a segmentation method to group adjoining pixels by color, texture, intensity, etc., and generate about 2k region proposals. Each proposal is warped into a fixed square size (227×227) and is fed into a CNN, AlexNet that makes use of five convolutional layers and two fully connected layers to extract a feature vector of dimension 4096×1. SVM makes use of this feature vector to classify objects in an image. It also outputs 4 offset values to enhance the exactness of a bounding box.

46.2.2 Fast R-CNN

R-CNN [2] works quite convincingly. However, it is tedious to compute a feature vector for every region proposal and this in turn accounts for a lot more memory space—2000 feature vectors for 2000 region proposals. Furthermore, three models—CNN, SVM, and a bounding box regressor—are to be trained separately. The author

of R-CNN, therefore, has come up with a better algorithm nullifying the constraints on time and memory.

Fast R-CNN [4] seems like a better solution. This algorithm inputs an entire image and processes it through a set of convolutional and max-pooling layers to generate a convolutional feature map. This is where the difference lies. Fast R-CNN computes a feature map on an entire image, unlike R-CNN. A corresponding part of the feature map is extracted for each region proposal. The region of interest (RoI) pooling layer warps the region proposals into fixed-length feature vectors. The feature vectors are loaded into a sequence of fully connected (FC) layers that bifurcate into two output layers. One that uses a softmax classifier to predict the probability estimates over the complete set of object classes and the other to output four offset values for refined bounding box aspect ratio.

46.2.3 Faster R-CNN

However, R-CNN and Fast R-CNN are found ineffective in real time because they employ a selective search [5] technique, which being a tedious process hinders the network's efficiency. Hence, Shaoqing Ren et al. have come up with an improvised algorithm [6] that uses a Region Proposal Network (RPN) in estimating the object proposals. In Faster R-CNN, an entire image is fed into a deep layered network to generate a convolutional feature map. A mini-network inputs an $n \times n$ block of this convolutional feature map to generate region proposals as illustrated in Fig. 46.1. This feature is led into a pair of FC's—a regression layer (Reg.) and a classification layer (Class.). A maximum of 'k' proposals are generated at every sliding window's

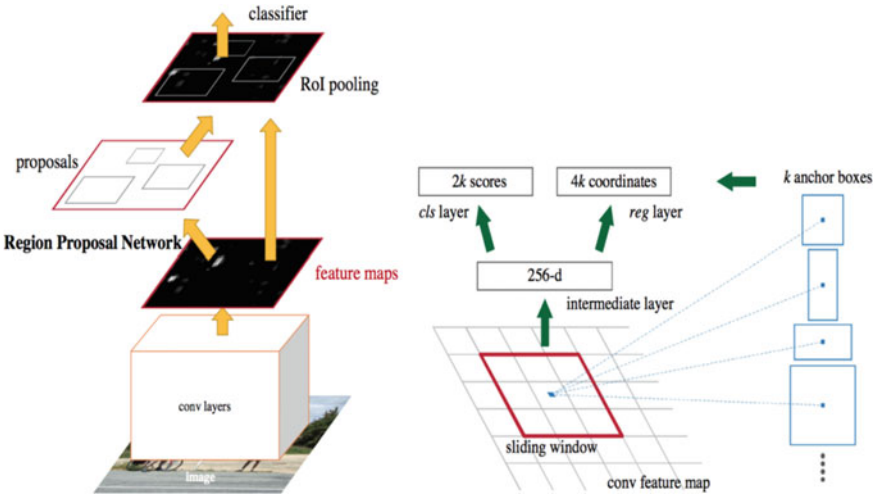

Fig. 46.1 Region proposal network. *Source* Ren et al. (2016)

location. The Reg. layer outputs $4 \times k$ encoded coordinates of k boxes and the Class. layer outputs $2 \times k$ probability estimates for the proposal constituting an object or not. The k proposals are parametrized correlative to k anchor boxes. This architecture is administered with an $n \times n$ convolutional layer succeeded by a pair of 1×1 convolutional layers meant for regression and classification.

46.2.4 YOLOv1

You Only Look Once (YOLO) is one of the most efficient algorithms in the history of object detection. R-CNN, Fast R-CNN, and Faster R-CNN employ region proposal methods to first generate region proposals and then load these proposed boxes into a classifier. These complex pipelines are time-consuming and so, they are not efficient in real-time. However, YOLO [7] is simpler and is extremely fast. You Only Look Once (YOLO) at an image to detect and classify an object. Its base network runs at a speed of 45 fps. Furthermore, its mean average precision (mAP) is twice more than other real-time systems.

The foremost step in YOLOv1 is to split an input image into an $s \times s$ grid and bring about s^2 grid cells. If an object's centroid falls within a grid cell, then that grid cell takes control in detecting that object. Each grid cell got a task of estimating the bounding boxes (BB), confidence parameters, and class probabilities (P_c). Mathematically, the output size happens to be $s \times s \times (BB \times 5 + P_c)$. Higher the confidence scores, more confident is the model in predicting that an object exists. What if the confidence score is zero? Then it is clear that the grid cell holds no object. A confidence score is computed by enacting an Intersection over Union (IoU) technique on the predicted box relative to a ground truth and if the value falls under a threshold, the detection would be straight away stated as a false negative (FN) detection. Even after threshold filtering, many boxes with higher objective scores and yet no sign of target objects are left out. To eliminate duplicate detection, a second filter called Non-Maximum Suppression (NMS) is used. YOLOv1's architecture emulates GoogleNet's model constituting a series of twenty-four convolutional layers adjoined by two fully connected layers. 1×1 layers are used in between to lessen the feature space.

46.2.5 YOLOv2

YOLOv1 performs fairly well. However, it will have to put up with a few shortcomings concerning localization and recall. Recall can be understood as the number of correct hits, and it is relatively low in comparison with other region proposal methods. YOLOv2 [8] is an improvised version over YOLOv1 and aims at enhancing object localization and recall.

Batch normalization would help enhance mAP by 2%. Furthermore, the network is trained on ImageNet images for 10 epochs at a higher resolution of 448 × 448. This in turn improves the mAP by 4%. The use of anchor boxes to localize objects enhances recall rate by 7%. However, there is a decrease in mAP by 0.3%. YOLOv2 [9] employs k-means clustering to instinctively discover good anchors rather than hand-picked anchor boxes as used in YOLOv1. This enhances the mAP by about 5%. Opting new image proportions arbitrarily for every 10 batches trains the network competently across various image dimensions.

46.2.6 YOLOv3

YOLOv3 is a modified version of YOLOv2. YOLOv2's 30 layered model architecture (Darknet-19 and 11 layers on top for object detection) often struggle to detect smaller objects as a result of downsampling the fine-grained features by the layers. To overcome this crisis, YOLOv3 [10] concatenates the existing feature map with the preceding layer's feature map and captures low-level features. YOLOv3 makes use of the model, Darknet-53 that originally has 53 layers with an additional 53 layers stacked onto the existing layers to perform object detection, attributing to its efficiency in locating smaller objects. However, this makes v3 slower in comparison with its other versions.

As depicted in Fig. 46.2, YOLOv3's object detection at three different output layers (8,294,106) and three different scales enhances its ability in detecting smaller objects. Logistic classifiers are used as an alternative to softmax for predicting the class and bounding box priors are constructed by employing a k-means clustering algorithm on the training dataset.

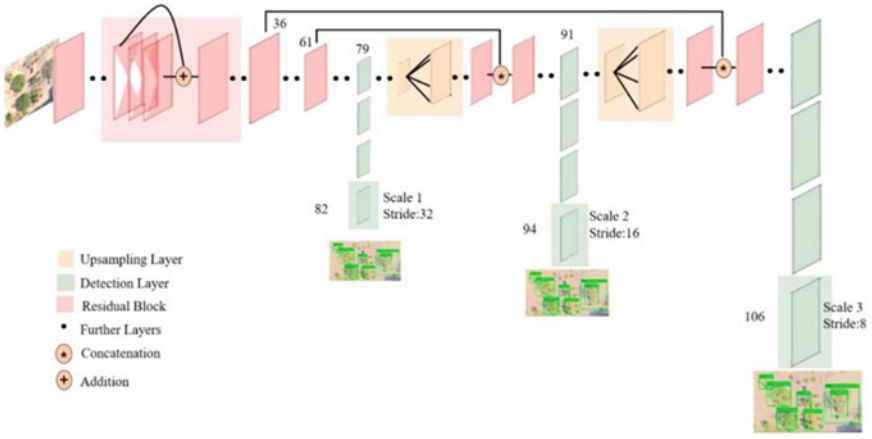

Fig. 46.2 Architecture of YOLOv3. *Source* Hossain and Lee (2019, p. 10)

46.2.7 SSD

Unlike other region proposal algorithms like Faster R-CNN that requires two distinct steps in detecting objects—one for estimating region proposals, other for detecting objects in each region proposal, SSD [11] takes one shot in detecting multiple objects in an image. It uses VGG-16 model architecture to extract feature maps. 3×3 convolutional filter is used on these feature maps to find the bounding boxes and class scores of objects. SSD makes use of distinctly scaled feature maps to precisely detect larger and smaller objects in an image. Low-resolution feature maps would work sufficiently well on larger objects. However, high-resolution feature maps would be necessary for detecting smaller objects. Default bounding boxes are chosen manually beforehand to enclose a broad range of real-time objects. Different resolution filters use boxes with various aspect ratios (1, 2, 3, 1/2, 1/3). To find a perfect bounding box for an object, a technique called matching strategy is used, which states that a default box with IoU higher than a threshold (say 0.5) concerning the ground truth is a positive match. NMS is used on top of this to remove possible duplicates.

46.3 Experimental Analysis

The proposed algorithm, YOLOv3 is trained on COCO and PASCAL VOC datasets, and the experimental results are compared among various object detectors—R-CNN, Fast R-CNN, Faster R-CNN, YOLOv1, YOLOv2, and SSD.

46.3.1 R-CNN, Fast R-CNN, and Faster R-CNN

R-CNN generates about 2000 region proposals for a single image and the detection time is about 49 s as inferred from Table 46.1. This is not effective for real-world examples. Fast R-CNN is, therefore, devised to take in an entire image as input. This saves up a lot of training time and is quite efficient in a way. However, selective search

Table 46.1 Comparison among region-based neural network detectors

		R-CNN	Fast R-CNN	Faster R-CNN
Method	Training data	SS	SS	RPN
mAP (%)	07	58.5	66.9	69.9
	12	53.3	65.7	67.0
	07 + 12	–	68.4	73.2
	07 + 12 + COCO	–	–	75.9
Detection time (s)		49	2.32	0.2

Table 46.2 Performance measure on Pascal VOC dataset. *Source* Redmon and Farhadi (2018, p. 4) [10]

Method	Input proportions	# Boxes	Mean average precision (%)	Frame rate (FPS)
Faster R-CNN (VGG-16)	1000 × 600	6000	73.20	7
YOLO (VGG-16)	448 × 448	98	66.40	21
SSD	300 × 300	8732	74.30	46
SSD	512 × 512	24,564	76.80	19

[5] constitutes a major part of the training period. Hence, R-CNN and Fast R-CNN are ineffective in real-time. Faster R-CNN replaces selective search and instead uses a region proposal network (RPN) that will help process an image in just 0.2 s and is, therefore, worthy of usage in real time.

46.3.2 Faster R-CNN, YOLO, and SSD

Faster R-CNN works well. However, its FPS is pretty low in comparison with normal standards. If we care for real-time speed, SSD and YOLO are at the rescue. YOLO and SSD are state of the art models that are capable of achieving a higher frame rate. As inferred from Table 46.2, YOLOv1′s mAP is the least and it fails in detecting minor objects. However, SSD300 outruns all other detectors while maintaining a fair FPS—real-time speed.

Table 46.3 depicts how YOLOv2 outperforms prior object detection methods concerning accuracy and speed. Look at the way it runs for different resolutions of input images for an effortless exchange between accuracy and speed. At low resolution, i.e., at 288 × 288, YOLOv2 runs at almost 90 FPS. This feature makes it worthy for usage in high FPS video streams and compact GPU's. At higher resolution, YOLOv2 is very much accurate with an mAP, 78.6, and an acceptable real-time speed, 40 FPS (Table 46.4).

If a comparison is to be made between SSD and YOLOv3 on COCO, at an input resolution, 320 × 320, YOLOv3 runs about three times faster than SSD achieving almost the same mAP (28%).

46.4 Conclusion

This paper makes a comparative analysis among various CNN-based object detection algorithms from the very basic R-CNN to the very recent path-breaking SSD putting forth various aspects on speed, accuracy, and memory. R-CNN and Fast R-CNN

Table 46.3 Performance w.r.to Pascal VOC 2007

Method	Training data	Mean average precision (%)	Frame rate (FPS)
Fast R-CNN	07 + 12	70.00	0.5
Faster R-CNN VGG16	07 + 12	73.20	7
Faster R-CNN residual net	07 + 12	76.40	5
YOLO	07 + 12	63.40	45
SSD-300	07 + 12	74.30	46
SSD-500	07 + 12	76.80	19
YOLOv2 (288 × 288)	07 + 12	69.00	91
YOLOv2 (352 × 352)	07 + 12	73.70	81
YOLOv2 (416 × 416)	07 + 12	76.80	67
YOLOv2 (480 × 480)	07 + 12	77.80	59
YOLOv2 (544 × 544)	07 + 12	78.60	40

Table 46.4 Performance on COCO dataset

Algorithm	Mean average precision (%)	Time (in ms)
SSD (321 × 321)	28.00	61
SSD (513 × 513)	31.20	125
YOLOv3 (320 × 320)	28.20	22
YOLOv3 (416 × 416)	31.00	29
YOLOv3 (608 × 608)	33.00	51

take about 49 s and 2.3 s, respectively, to process an image. It's quite high for real-world applications. Faster R-CNN is comparatively better in respect of detection time. However, YOLO and SSD are state-of-the-art models and work efficiently with real-time speeds. SSD300 runs at 59FPS exceeding the existing state-of-the-art YOLOv1's 45FPS. YOLOv2 is faster and can run over different image proportions providing a smooth barter between speed and accuracy. YOLOv3 and SSD can work well on smaller objects. Various constraints stated in this study might help the user choose an algorithm that better serves the application.

References

1. J.V.V.S.N. Raju, P. Rakesh, Neelima. N, Driver drowsiness monitoring system, in *Smart Innovation, Systems and Technologies (SIST)* vol. 169 (2019), pp. 675–683
2. R. Girshick, J. Donahue, T. Darrell, J. Malik, Rich feature hierarchies for accurate object detection and semantic segmentation, in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2014). ISBN 978-1-4799–5118-5
3. K. Madan, K. Bhanu Anusha, P. Pavan Kalyan, N. Neelima, Research on different classifiers for early detection of lung nodules. Int. J. Recent Technol. Eng. 1037–1040 (2019)
4. R. Girshick, Fast R-CNN, in *IEEE International Conference on Computer Vision (ICCV)* (2015). ISSN 2380-7504
5. J.R. Uijlings, K.E. van de Sande, T. Gevers, A.W. Smeulders, Selective search for object recognition. Int. J. Comput. Vision (IJCV)
6. S. Ren, K. He, R. Girshick, J. Sun, Faster R-CNN: towards real-time object detection with region proposal networks, in *Advances in Neural Information Processing Systems (NIPS 2015)*, vol. 28 (2015), pp. 91–99
7. J. Redmon, S. Divvala, R. Girshick, A. Farhadi, You only look once: unified, real-time object detection (2016), pp. 1–10
8. J. Redmon, A. Farhadi, YOLO9000: better, faster, stronger, in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017), pp. 6517–6525
9. D. Foley, R. O'Reilly, An evaluation of convolutional neural network models for object detection in images on low-end devices, in *AICS, 2018*, vol. 2259 (2018), pp. 1–12
10. J. Redmon, A. Farhadi, YOLOv3: an incremental improvement (2018) pp. 1–6
11. W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, A.C. Berg, SSD: single shot multibox detector, in *EECV 2016* (2016), pp. 21–37