Final Project Report:

A Survey of Deep Learning Based Object Detection

IEEE Access, vol. 7, pp. 128837-128868, 2019

Author: Pablo Ruiz

Abstract:

Object detection is one of the areas in computer vision that has seen large advances over the last decade. The introduction of Convolutional Neural Networks has given rise to both two-stage and one-stage detector models which exhibit very good performance and can be used for general and domain specific object detection. This paper provides a comprehensive review of the architecture, performance metrics, and performance of modern CNN object detection models.

1. Introduction

Object detection is at the forefront of computer vision research and large advances have been made in the last decade. Advancements in this field are significant because of its wide-ranging applications from autonomous driving, robotics, security/military, medical, etc. Recent advancements can be attributed to developments in convolutional neural networks (CNNs) which were made possible by rapidly increasing GPU/Accelerator computing power. Object detection as a whole is a field which deals with detecting instances of semantic objects of a certain class i.e., humans, dogs, cats, boats, cars, etc. in images or videos. To accomplish this it must first, detect regions of the image where individual objects are likely to be, surround them with a representative bounding box (BB), and then classify each region in a category with an associated confidence value as seen in Figure 1. Models can be trained for general object detection or domain specific object detection with a narrow focus on a single or a small group or related categories. This paper aims to explore the new developments in object detection models primarily those using convolutional neural networks as that is where most of the breakthroughs have taken place. Most relevant domain specific object detection models can be split into two categories based on their internal components, two-stage object detectors and one-stage object detectors. Two-stage object detectors are generally more complex and as a result are more accurate but slower and more computationally expensive both in training and in inference. One-stage object detectors are more streamlined and faster, but they come with limitations on accuracy and the number of objects instances that can be detected per image. To quantify the performance of each object detection model, relevant performance metrics such as Intersection Over Union (IOU) and Average Precision (AP) will also be presented. Finally, using performance data obtained from multiple other peer reviewed research papers, shown in Figure 14, the performance upsides and downsides of both two-stage and one-stage detector models will be shown and discussed.

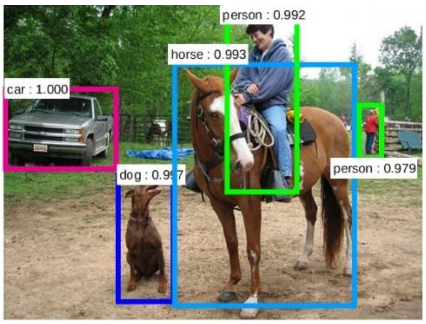

Figure 1. A sample image after undergoing the YOLOv3 one-stage object detection algorithm. [17]

2. Backbone Network

The backbone network of an object detection model is the core of model and has a large impact on the performance of the model as will be shown later in section 6. The backbone network is a CNN in charge of basic feature extraction for object detection tasks. It takes an image as input and outputs a feature map which can then be used in subsequent stages to identify individual regions of interest which might contain an object instance. Object detection models are usually designed to be modular in the sense that they can be made to work with any backbone network which can output the right size feature map to use as the input for the nest stage of the model. This allows a model to change its performance characteristics simply by changing the backbone network to one which prioritizes different features which are more appropriate for a specific semantic category or has a different balance of time/compute efficiency and accuracy. Generally, the quality of the feature map determines the ceiling of model performance therefore choosing an appropriate and powerful backbone network is vital. Several popular backbone networks currently in use are ResNet [9], ResNeXt [10], and AmoebaNet [11] which prioritize accuracy and ShuffleNet [12], SqueezeNet [13], and MobileNetV2 [14] which prioritize efficiency.

Convolutional Neural Networks (CNN) are a subset of artificial neural networks which are primarily used for image, speech and audio signal analysis. They are composed of layers of neurons/nodes primarily divided into an input convolution layer, an output fully connected layer and a series of hidden pooling or convolution layers in between. The first layers detect

simple features while the later layers detect larger objects or shapes. Generally, the more layers the better and more complex the extracted features but the performance suffers as a result. Convolution layers are the core of the CNN, and they apply a kernel to a subset of the input image using convolution and the resulting values form a new array which is the input for the subsequent layers. Pooling or down sampling layers also use a filter to reduce the dimensionality of the input, however they usually apply an averaging filter which takes the average of small subset of the input or a max filter which takes the largest value. Fully connected layers are different and connect every member of the input array to a member of the output array using a softmax activation function outputting a probability from 0 to 1. A representation of a backbone CNN can be seen in Figure 2.

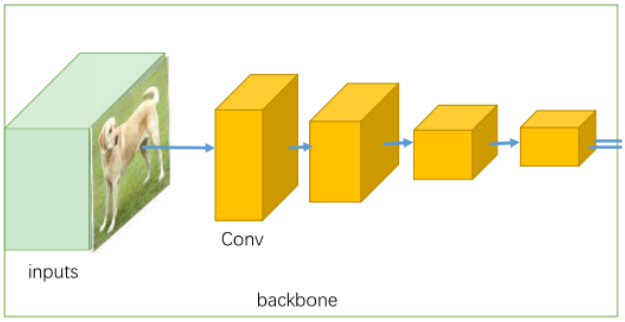

Figure 2 Structure of a backbone CNN. [4]

3. Two-Stage Object Detectors

Two-stage detectors were the first to be demonstrated by Girshick *et al.* in 2014 [7] and are generally employed in asynchronous applications where precision is more important than throughput. Due to their larger number of internal components their efficiency is low compared to one stage object detectors. The model proposed by Girshick is known as Region Based Convolutional Neural Network (R-CNN) and was subsequently improved a year later in 2015 by the same authors into what became known as Fast-CNN [15] and again two years later in 2017 into Faster-CNN [16].

3.1 Region Based Convolutional Neural Network (R-CNN)

R-CNN are composed of four distinct building blocks as seen in Figure 3. The first module is a category independent region proposal selective search algorithm which takes the image as an input and outputs several possible regions (usually ~2000) which contain an object of interest. Subsequently each of these regions must individually go through the backbone CNN to obtain a feature map. Next, the feature vector is run through a set of class specific Support Vector Machines (SVM) used for classification to find which one it has the highest correlation with. Finally, once the class has been determined a bounding box regressor is used to obtain a precise bounding box prediction. This workflow is illustrated in Figure 4.

The main problem with this approach as seen in Figure 4 is that the backbone feature extraction network must be ran once for every region proposal which is very inefficient. Furthermore, having a single SVM for each class included in the model is also very inefficient since every feature vector must go through every SVM before the best classification for it can be chosen.

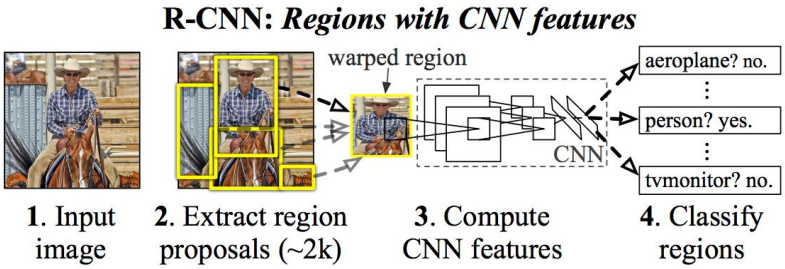

Figure 3. Distinct components of the R-CNN model. [6]

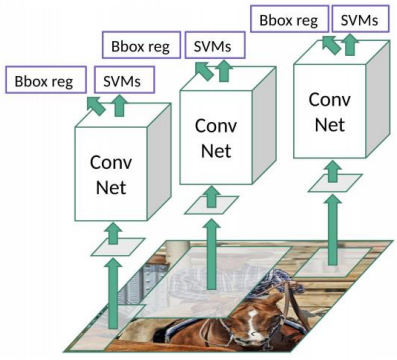

Figure 4. Workflow in an R-CNN model. [6]

3.2 Fast R-CNN

Fast R-CNN is an incremental improvement on R-CNN which addresses some of the bottlenecks identified in the previous section. Ultimately, it improves the performance both in inference speed and memory usage. The main structural change in Fast R-CNN was applying the backbone feature detection CNN to the whole image rather than to each region proposal individually. Thus, the feature detection CNN and the region proposal algorithm are run on the whole image in parallel and at the end the region proposals are overlaid on the feature map to identify the features in each region, as shown in Figure 5. Another smaller improvement was adding an additional region of interest pooling layer following in order to reduce the size of the feature map for each region to match the input size for the subsequent fully connected layers. Additionally, the SVM classification was replaced by a fully connected softmax layer which outputs a set of class probabilities for each region. Testing with the PASCAL VOC 2007 dataset, shown in Figure 7, shows similar precision performance of 66.9% (Fast-CNN) vs. 60% (CNN) with much better training times of 9.5hrs vs. 84hrs, a 9x improvement.

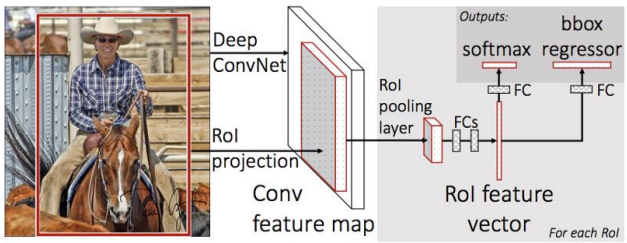

Figure 5. Distinct components of the Fast R-CNN model. [6]

3.3 Faster R-CNN

Faster R-CNN was the second incremental improvement on R-CNN and attempts to further improve performance beyond that of the previous iteration. The bottleneck in Fast R-

CNN became the region proposal generation because it was using a selective search algorithm. Faster R-CNN replaces this module with a separate CNN to predict region proposals in one pass. The new region proposal network is integrated into the backbone feature detection CNN and shares several features with it as seen in Figure 6. This change streamlines the process and integrates one of the main processes into a singular CNN, making it partially resemble the structure of one-stage object detectors as will be shown in the next section. This change alone was enough to dramatically improve the performance of the model. As shown in Figure 7 the inference performance of Faster R-CNN is faster than Fast R-CNN by a factor of 10 and R-CNN by a factor of 245, a massive improvement. The inference performance of Faster R-CNN is so good that it could almost be used in real time applications as noted by Girshick *et al.* [16].

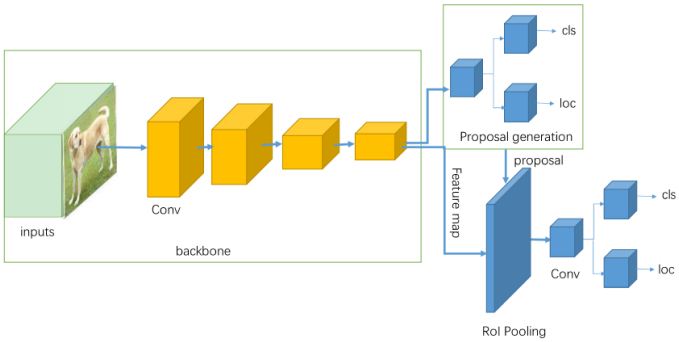

Figure 6. Structure of the Faster R-CNN model. [4]

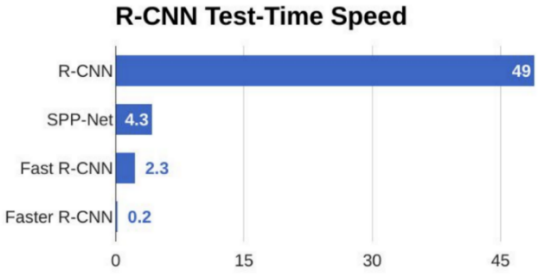

Figure 7. Observed inference performance in seconds using the PASCAL VOC 2007 dataset. [6]

4.One-Stage Object Detectors

The type of one-stage detector being explored in this paper, You Only Look Once (YOLO), was proposed by Redmon *et al.* in 2016 [8] and addresses the need for object detection in mobile platforms will limited power and/or compute such as robots or autonomous vehicles. The principal aim of these methods is to perform object detection in one shot with a single, in-line CNN.

4.1 You Only Look Once

One-stage object detectors and particularly YOLO have become widely used in recent years and several further iterations such as YOLOv3 and YOLOv5 have been developed. In this paper the focus will be on the original model and its structure.

YOLO treats object detection as a regression problem and performs the whole process in a single CNN as shown in Figure 8. The first step is to split an image into a grid with predefined dimensions, usually 19x19. Each grid is responsible for finding K bounding boxes as shown in Figure 9 (BB with their center within the cell, outer edges can be anywhere in the image). An object is considered to lie in a specific grid if the center of its proposed bounding box lies in that grid. The probability for each class for each bounding box is calculated (Pc) using a fully connected softmax layer and the largest is chosen as its classification. Ultimately the intersection of bounding boxes is used to discard boxes with high overlap and select the one amongst them which has the highest Pc as shown in Figure 10. This process results in several limitations. First since it lacks a specific region proposal generation mechanism the number of region proposals which can be generated is limited by the number of grid squares and the number of bounding boxes within them. In practice, the detection limit for the YOLO tends to be around 100 BB per image. Additionally, it struggles particularly with images containing smaller objects in higher numbers.

The structure outlined above results in significantly higher inference performance (~45fps), ~80x faster than Fast R-CNN and ~6x faster than Faster R-CNN [4]. For example, in PASCAL VOC dataset YOLO obtains an accuracy of 63.4% mAP at 45fps, while Fast R-CNN obtains 70.0% mAP at 0.5fps and Faster R-CNN 73.2% mAP at 7fps [4].

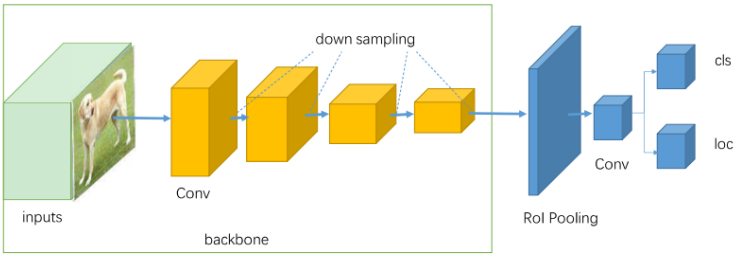

Figure 8. Structure of the YOLO model [4]

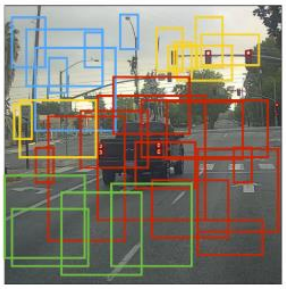

Figure 9. Bounding box proposals obtained through YOLO. [5]

Figure 10. Elimination of overlapping same class bounding box proposals and selection of the one with the highest confidence. [5]

4. Performance Metrics

The following metrics are widely used to objectively quantify the performance of a given object detection model on a specific dataset of images with a known ground truth which includes bounding box location, dimensions, and class.

5.1 Intersection Over Union

Intersection over union (IoU) is a term used to quantify the accuracy of a bounding box proposal with respect to a known ground truth bounding box. It is represented by a value between 0 and 1 where 1 would be a perfect match and 0 would represent two bounding boxes with no overlap. IoU is calculated by taking the area of overlap or intersection between two bounding boxes and dividing it by the total area covered by both bounding boxes or their union as shown graphically in Figure 11. Furthermore, usually to train a model or compare results between models a binary output which represents success or failure is desired. To obtain this from the continuous IoU result a threshold must be selected and applied as shown in Figure 12.

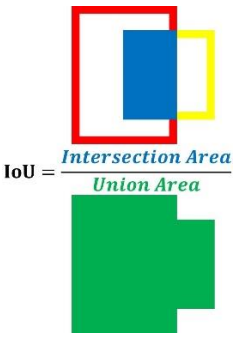

Figure 11. Graphical representation of Intersection over Union (IoU) [3]

$$class(IoU) = \begin{cases} Positive \rightarrow IoU \geq Threshold \\ Negative \rightarrow IoU < Threshold \end{cases}$$

Figure 12. Mathematical representation of thresholding to obtain a binary output from IoU values. [3]

5.2 Average Precision

There are two main metrics that can be derived from the IoU results from a model and the known ground truth from a dataset, precision and recall. Precision is a continuous value from zero to one and represents the confidence of the model when classifying a sample or of the classifications made how many were correct. Recall, also a continuous value from zero to

one, represents the number of samples correctly classified as positive out of the total or of the classifications possible how many were made correctly. A model that has high recall, but low precision would exhibit lots of false positives. On the other hand, a model with low recall but high precision would exhibit a lot of missed classifications but when it does find a bounding box it is almost always correct.

The average precision (AP) and mean average precision (mAP) are the main performance metrics used to compare model performance with a given dataset. To find the AP first a plot of the precision vs. the recall including every image in the dataset must be made for each class. The average precision is represented by the area under the curve as shown in Figure 13. The mean average precision is simply the mean of the average precision values across all the classes included in the object detection model.

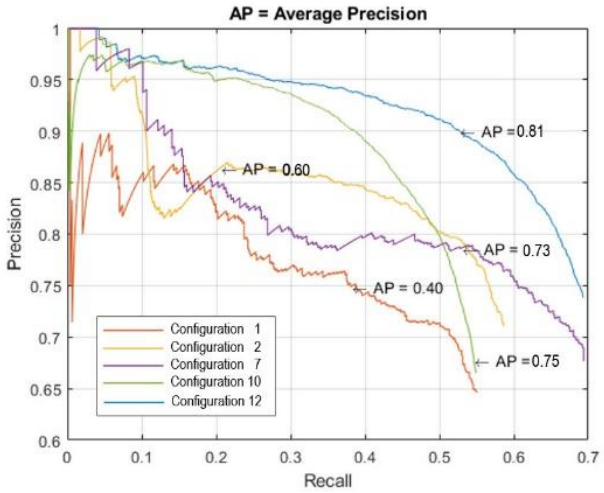

Figure 13 Graph representing the Precision-Recall curve of a single class in a dataset for several models. [1]

5. Performance

The table in Figure 14 represents a compilation of AP values for different object recognition models using different backbone networks and different datasets. This means that most of these values are not directly comparable to each other as it has been established that the backbone network can have significant effects on the performance of a model and furthermore

the data set and accompanying ground truth used to test a model can influence perceived performance through the AP performance metrics outlined above since all data sets are not equally as difficult or their ground truths equally as accurate. Therefore, it is important to note that the results in the table are a general indication of the performance of each model relative to each other.

TABLE 2. Detection results on the MS COCO test-dev dataset of some typical baselines. *AP*, AP_{50}, AP_{75} scores (%). AP_S:AP of small objects, AP_M: AP of medium objects, AP_L:AP of large objects. *DCNv2+Faster R-CNN models are trained on the 118k images of the COCO 2017 train set.

Method	Data	Backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Fast R-CNN [29]	train	VGG-16	19.7	35.9	–	–	–	–
Faster R-CNN [8]	trainval	VGG-16	21.9	42.7	–	–	–	–
OHEM [40]	trainval	VGG-16	22.6	42.5	22.2	5.0	23.7	37.9
ION [41]	train	VGG-16	23.6	43.2	23.6	6.4	24.1	38.3
OHEM++ [40]	trainval	VGG-16	25.5	45.9	26.1	7.4	27.7	40.3
R-FCN [42]	trainval	ResNet-101	29.9	51.9	-	10.8	32.8	45.0
CoupleNet [43]	trainval	ResNet-101	34.4	54.8	37.2	13.4	38.1	52.0
Faster R-CNN G-RMI [44]	–	Inception-ResNet-v2	34.7	55.5	36.7	13.5	38.1	52.0
Faster R-CNN+++ [25]	trainval	ResNet-101-C4	34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN w FPN [15]	trainval35k	ResNet-101-FPN	36.2	59.1	39.0	18.2	39.0	48.2
Faster R-CNN w TDM [45]	trainval	Inception-ResNet-v2-TDM	36.8	57.7	39.2	16.2	39.8	52.1
Deformable R-FCN [38]	trainval	Aligned-Inception-ResNet	37.5	58.0	40.8	19.4	40.1	52.5
umd_det [46]	trainval	ResNet-101	40.8	62.4	44.9	23.0	43.4	53.2
Cascade R-CNN [47]	trainval35k	ResNet-101-FPN	42.8	62.1	46.3	23.7	45.5	55.2
SNIP [48]	trainval35k	DPN-98	45.7	67.3	51.1	29.3	48.8	57.1
Fitness-NMS [49]	trainval35k	ResNet-101	41.8	60.9	44.9	21.5	45.0	57.5
Mask R-CNN [11]	trainval35k	ResNeXt-101	39.8	62.3	43.4	22.1	43.2	51.2
DCNv2+Faster R-CNN [39]	train118k*	ResNet-101	44.8	66.3	48.8	24.4	48.1	59.6
G-RMI [44]	trainval32k	Ensemble of Five Models	41.6	61.9	45.4	23.9	43.5	54.9
YOLOv2 [30]	trainval35k	DarkNet-53	33.0	57.9	34.4	18.3	35.4	41.9
YOLOv3 [32]	trainval35k	DarkNet-19	21.6	44.0	19.2	5.0	22.4	35.5
*SSD300** [10]	trainval35k	VGG-16	25.1	43.1	25.8	6.6	22.4	35.5
RON384+++ [50]	trainval	VGG-16	27.4	49.5	27.1	–	–	–
SSD321 [34]	trainval35k	ResNet-101	28.0	45.4	29.3	6.2	28.3	49.3
DSSD321 [34]	trainval35k	ResNet-101	28.0	46.1	29.2	7.4	28.1	47.6
SSD512* [10]	trainval35k	VGG-16	28.8	48.5	30.3	10.9	31.8	43.5
SSD513 [34]	trainval35k	ResNet-101	31.2	50.4	33.3	10.2	34.5	49.8
DSSD513 [34]	trainval35k	ResNet-101	33.2	53.3	35.2	13.0	35.4	51.1
RetinaNet500 [33]	trainval35k	ResNet-101	34.4	53.1	36.8	14.7	38.5	49.1
RetinaNet800 [33]	trainval35k	ResNet-101-FPN	39.1	59.1	42.3	21.8	42.7	50.2
M2Det512 [35]	trainval35k	VGG-16	37.6	56.6	40.5	18.4	43.4	51.2
M2Det512 [35]	trainval35k	ResNet-101	38.8	59.4	41.7	20.5	43.9	53.4
M2Det800 [35]	trainval35k	VGG-16	41.0	59.7	45.0	22.1	46.5	53.8
RefineDet320 [36]	trainval35k	VGG-16	29.4	49.2	31.3	10.0	32.0	44.4
RefineDet512 [36]	trainval35k	VGG-16	33.0	54.5	35.5	16.3	36.3	44.3
RefineDet320 [36]	trainval35k	ResNet-101	32.0	51.4	34.2	10.5	34.7	50.4
RefineDet512 [36]	trainval35k	ResNet-101	36.4	57.5	39.5	16.6	39.9	51.4
RefineDet320+ [36]	trainval35k	VGG-16	35.2	56.1	37.7	19.5	37.2	47.0
RefineDet512+ [36]	trainval35k	VGG-16	37.6	58.7	40.8	22.7	40.3	48.3
RefineDet320+ [36]	trainval35k	ResNet-101	38.6	59.9	41.7	21.1	41.7	52.3
RefineDet512+ [36]	trainval35k	ResNet-101	41.8	62.9	45.7	25.6	45.1	54.1
CornerNet512 [51]	trainval35k	Hourglass	40.5	57.8	45.3	20.8	44.8	56.7
NAS-FPN [18]	trainval35k	RetinaNet	45.4	–	–	–	–	–
NAS-FPN [18]	trainval35k	AmoebaNet	48.0	–	–	–	–	–

1. Unless otherwise specified, *AP and AR are averaged over multiple Intersection over Union (IoU) values.* Specifically we use 10 IoU thresholds of .50:.05:.95. This is a break from tradition, where AP is computed at a single IoU of .50 (which corresponds to our metric $AP^{IoU=.50}$). Averaging over IoUs rewards detectors with better localization.

Figure 14. Table with AP results for many object detection model, backbone, and data set combinations [4]

Additionally, it is important to note that thew subscript next to each AP column represents the threshold used to obtain the IoU binary output for each bounding box. Therefore, AP50 represents the AP using an IoU threshold of 0.50 and so on.

The most analogous results in this table which pertain to the Faster R-CNN and YOLO models are those highlighted in Figure 15. These results are somewhat comparable because although the backbone is different the data set being used in the same. This shows, as was implied previously, that Faster R-CNN is overall more accurate represented by the larger AP values across the board. However, it is important to note that the accuracy is relatively close and in many applications this tradeoff could be warranted depending on the value associated with the increased inference performance provided by the YOLO model.

Method	Data	Backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Faster R-CNN w FPN [15]	trainval35k	ResNet-101-FPN	36.2	59.1	39.0	18.2	39.0	48.2
YOLOv3 [32]	trainval35k	DarkNet-19	21.6	44.0	19.2	5.0	22.4	35.5

Figure 15. Subset of figure 15 which includes the most analogous results for the Faster R-CNN and YOLOv3 models. [4]

Finally, to complete the picture Figure 16 illustrates the difference in inference performance between the Faster R-CNN model, which as mentioned previously is the fastest of the two-stage detector R-CNN based models, and the YOLO model. When comparing minimum performance YOLO is 8 times faster than Faster R-CNN with 40fps and 5fps respectively. Furthermore, when comparing maximum performance YOLO is over 5 times faster than Faster R-CNN with 91fps and 17fps respectively. Additionally, it is important to note that the maximum throughput achieved by Faster R-CNN is still too low for most real time applications while that achieved by YOLO is more than adequate for most real time applications. This means that when dealing with real time, time sensitive applications YOLO or one-stage detector models like it are the only available choice.

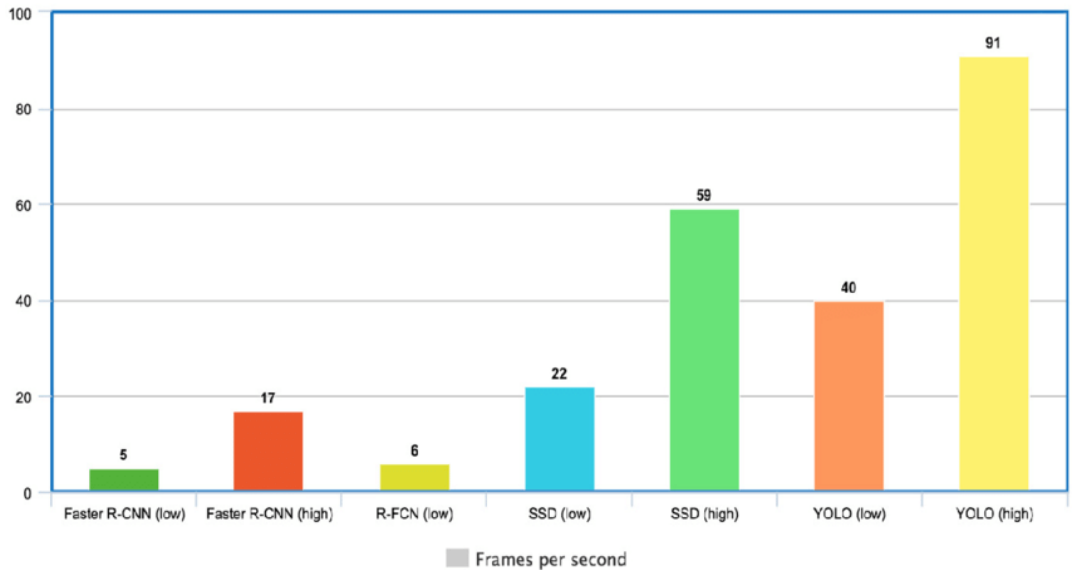

Figure 16. Graph showing the maximum and minimum throughput of different models measured in frames per second. [2]

7. Conclusion

Object detection has experienced great advancements in the last decade. Modern object detection methods rely heavily on CNN due to their fast performance and relative adaptability. The main downsides of CNN are their heavy compute and data requirements for training. Both one-stage and two-stage detectors are modular and can make use of different feature detection "backbone" CNN to fine tune their performance for a specific task. One-stage detectors offer faster inference but make tradeoffs in the number of objects detected, precision, and small object performance. There are applications for both methods with one-stage detectors excelling in mobile and real-time applications while two-stage detectors excel in applications where precision is critical, for example medical and other asynchronous applications. Given current performance levels and considering the performance of one-stage detectors can be expected to continue improving, for most applications their large increase in inference performance overshadows their relatively small accuracy limitations.

8. References

1. “Average precision (AP) of the model on the MATLAB dataset, evaluations ...” [Online]. Available: https://www.researchgate.net/figure/6-Average-Precision-AP-of-the-model-on-the-Matlab-Dataset-evaluations-done_fig23_337745095.

2. “Comparison of frames processed per second (FPS) implementing the faster ...” [Online]. Available: https://www.researchgate.net/figure/Comparison-of-frames-processed-per-second-FPS-implementing-the-Faster-R-CNN-R-FCN-SSD_fig6_342570032.

3. “Evaluating object detection models using mean average precision,” *KDnuggets*. [Online]. Available: https://www.kdnuggets.com/2021/03/evaluating-object-detection-models-using-mean-average-precision.html.

4. L. Jiao, F. Zhang, F. Liu, S. Yang, L. Li, Z. Feng, and R. Qu, “A survey of Deep Learning-based object detection,” *IEEE Access*, vol. 7, pp. 128837–128868, 2019.

5. Manishgupta, “Yolo - you only look once,” *Medium*, 30-May-2020. [Online]. Available: https://towardsdatascience.com/yolo-you-only-look-once-3dbdbb608ec4.

6. R. Gandhi, “R-CNN, fast R-CNN, Faster R-CNN, YOLO - object detection algorithms,” *Medium*, 09-Jul-2018. [Online]. Available: https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e.

7. R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” 2014 IEEE Conference on Computer Vision and Pattern Recognition, 2014.

8. J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

9. K. He, G. Gkioxari, P. Dollar, and R. Girshick, “Mask R-CNN,” 2017 IEEE International Conference on Computer Vision (ICCV), 2017.

10. S. Xie, R. Girshick, P. Dollar, Z. Tu, and K. He, “Aggregated residual transformations for deep neural networks,” 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

11. G. Ghiasi, T.-Y. Lin, and Q. V. Le, “Nas-FPN: Learning Scalable Feature Pyramid Architecture for Object Detection,” 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019.

12. X. Zhang, X. Zhou, M. Lin, and J. Sun, “ShuffleNet: An extremely efficient convolutional neural network for mobile devices,” 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018.

13. F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, “Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <0.5MB model size,” arXiv.org, 04-Nov-2016. [Online]. Available: https://arxiv.org/abs/1602.07360. [Accessed: 14-Dec-2022].

14. M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, “MobileNetV2: Inverted residuals and linear bottlenecks,” 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018.

15. R. Girshick, “Fast R-CNN,” 2015 IEEE International Conference on Computer Vision (ICCV), 2015.

16. S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards real-time object detection with region proposal networks,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 6, pp. 1137–1149, 2017.

17. R. Patil, “How to use tensorflow object detection API on windows,” Medium, 31-Jan-2018. [Online]. Available: https://medium.com/@rohitrpatil/how-to-use-tensorflow-object-detection-api-on-windows-102ec8097699. [Accessed: 14-Dec-2022].