IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 18 March 2023, accepted 3 April 2023, date of publication 10 April 2023, date of current version 13 April 2023.

*Digital Object Identifier 10.1109/ACCESS.2023.3266093*

TOPICAL REVIEW

# Object Detection Using Deep Learning, CNNs and Vision Transformers: A Review

**AYOUB BENALI AMJOUD AND MUSTAPHA AMROUCH**
Image et Reconnaissance de Forme-Systèmes Intelligents et Communicants (IRF-SIC) Laboratory, Faculty of Sciences, Ibn Zohr University, Agadir 80000, Morocco

Corresponding author: Ayoub Benali Amjoud (a05.benali@gmail.com)

This work was supported by the Centre National pour la Recherche Scientifique et Technique (CNRST), Morocco, through the Research Excellence Awards Program 2019–2022 under Grant 5UIZ2019.

**ABSTRACT** Detecting objects remains one of computer vision and image understanding applications' most fundamental and challenging aspects. Significant advances in object detection have been achieved through improved object representation and the use of deep neural network models. This paper examines more closely how object detection has evolved in the era of deep learning over the past years. We present a literature review on various state-of-the-art object detection algorithms and the underlying concepts behind these methods. We classify these methods into three main groups: anchor-based, anchor-free, and transformer-based detectors. Those approaches are distinct in the way they identify objects in the image. We discuss the insights behind these algorithms and experimental analyses to compare quality metrics, speed/accuracy tradeoffs, and training methodologies. The survey compares the major convolutional neural networks for object detection. It also covers the strengths and limitations of each object detector model and draws significant conclusions. We provide simple graphical illustrations summarising the development of object detection methods under deep learning. Finally, we identify where future research will be conducted.

**INDEX TERMS** Object detection, deep learning, review, convolutional neural networks, transformers, survey, neural networks.

## I. INTRODUCTION

Research and breakthroughs in object detection fall into two main periods. Before 2014, they were marked by traditional detection models, and after 2014 by models based on deep learning. Furthermore, due to the successful application of deep neural networks (DNNs) and convolutional neural networks (CNNs) [1], especially in recent years, the situation in many artificial intelligence fields has improved considerably. As a result, significant progress has been made in computer vision tasks such as classification, segmentation, and object detection [2]. Object detection involves image classification [1] and semantic and instance segmentation [2], [3]. Visual object detection is a process of image classification [3] and localization. This task becomes more complex than simple image classification or classification with localization, as an image usually contains several objects of different categories. It consists of locating the instances of an object in a given image and assigning each object instance a matching class label from a wide range of predefined classes. Deep learning-based object detection models using convolutional neural networks and transformers are now playing a pivotal role in the evolution of this domain. These models can provide vital information for the semantic understanding of images and videos. It has experienced a rapid rate of adoption in a variety of sectors. Examples include support for autonomous cars to navigate safely in traffic [4], [5], [6], [7], detection of abusive behavior [8], [9], facial detection [10], [11], human behavior analysis [12], [13], and medical imaging such as cancer detection [14], [15], robotics [16], [17], general image processing techniques such as cropping, orientation detection, and contrast enhancement [18], [19], [20], [21], remote sensing applications [214], [215], [216], and many other use cases [217], [218], [219]. Regarding future use cases for object detection, the possibilities are endless. To develop algorithms that can detect objects in a scene, we need to look

The associate editor coordinating the review of this manuscript and approving it for publication was Bing Li.

beyond shallow and deep CNNs. For a better understanding of the dynamics and interactions between objects in these visual scenes, it is necessary to use sequential and relational information modeling to connect objects both in time and space. However, before introducing and clarifying these advanced techniques, it is worthwhile first to understand the evolution of state-of-the-art object detectors, their limitations, and how they can be addressed. This paper presents an in-depth review of several approaches for solving the object detection task. We will explore and discuss the different frameworks used for object detection and the primary data sets and metrics applied to evaluate the detection. We describe the advantages and limitations of the most widely used convolutional neural networks, serving as a backbone for the leading object detection models. Initially, we cover algorithms from the anchor-based family for object detection, including two- and one-stage object detectors. We also review more sophisticated and faster algorithms based on anchor-free and transformer-based object detection approaches. Next, we elaborate on each approach's strengths and weaknesses by comparing the methods mentioned in the paper. Then, we shall provide a discussion of some future directions and prospects.

### A. COMPARISON WITH PREVIOUS REVIEWS

All previous studies [22], [35] were limited to an overview and comparison of a limited number of object detection models, although other models were available at their time. Most previous surveys followed the same method of dividing the models into two categories; two-stage and one-stage detectors. Moreover, some have just focused on one aspect of object detection. For example, some have studied the detection of salient objects [26], [30]. Others have studied the detection of small objects [33], [34], and others for tiny objects [31]. In [32], they review the learning strategies of object detector models. In this paper, we tried to cover all the detection models and approaches that depended on deep learning from 2013 to 2022, including the object detection models based on transformers published more recently. No previous work has comprehensively covered and analyzed the number of models we have listed. We also divided the detection models into four categories. The first concerns two-stage models based on anchors, the second relates to one-stage models based on anchors, the third refers to anchor-free methods, and the last category concerns transformer-based models.

### B. OUR CONTRIBUTIONS

The primary motivation of this work is to provide a comprehensive, detailed, and simplified overview through tables and figures of the past and current state of the field of object detection. This paper can be a starting point for researchers and engineers seeking to gain knowledge in this field, especially for those beginning their careers. They can learn about the current situation and contribute to advancing the field. Our contribution differs from previous ones regarding its focus and the number of models mentioned and covered. However, understanding any domain and developing new concepts necessitates knowledge of all existing concepts, including their pros and cons, particularly in a fast-developing field such as object detection. Our work brings some added value to the field of object detection. Therefore, it will provide researchers, especially those starting in this field or those interested in applying these techniques in other specific disciplines, such as healthcare, with an up-to-date, state-of-the-art overview of object detection.

1) We propose an up-to-date survey that covers older and more recently published object detection models.
2) We present the first review, which covers almost all object detection models based on deep learning.
3) We compare the different backbone networks object detectors use through their strengths, features, and limitations.
4) We suggest a research study outlining and investigating generic object detection approaches from the perspective of anchors and transformers.
5) We summarise the evolution and categories of object detection with deep learning in simplified charts, diagrams, and tables.
6) We outlined promising future directions in the field of object detection.

## II. TRADITIONAL OBJECT DETECTION METHODS

The first notable strides in object detection and image recognition began in 2001 when Paul Viola and Michael Jones designed an effective facial detection algorithm [36], a robust binary classifier built from multiple low classifiers. Their demonstration of faces detected in real-time on a webcam was the most impressive illustration of computer vision. In 2005, a new paper by Navneet Dalal and Bill Triggs was published. Their approach, based on the feature descriptor, Oriented Gradient Histograms (HOG) [37], outperformed existing pedestrian detection algorithms. In 2009 Felzenszwalb et al. developed the Deformable Part Model (DPM) [38], another crucial feature-based model. As a result, DPM has proven to be highly successful in object detection applications in which bounding boxes were applied to localize objects, as well as in template matching and other well-known object detection approaches used at the time. Several methods have already been developed to extract patterns from images and detect objects [39], [42]. All traditional methods tend to involve three parts: 1) The first step consists in inspecting the entire image at multiple positions and scales to generate candidate boxes with the use of methods like sliding window [43], [44], max-margin object detection, region proposal like the selective search algorithm [45]. Usually, with sliding windows, capturing several thousand windows in each image is usually necessary. Any costly calculation method used at this first level results in a prolonged process of scanning the entire image. Especially during training, several iterations on the training set are often necessary to include the selected

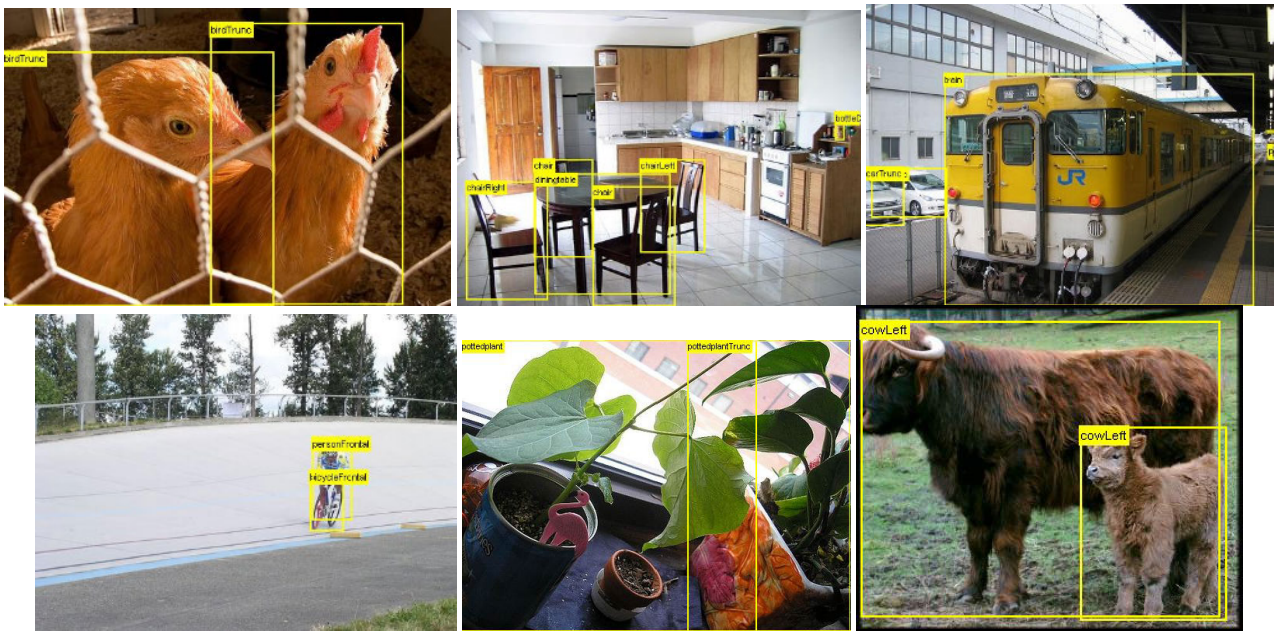


**FIGURE 1.** Samples from Pascal VOC 07.

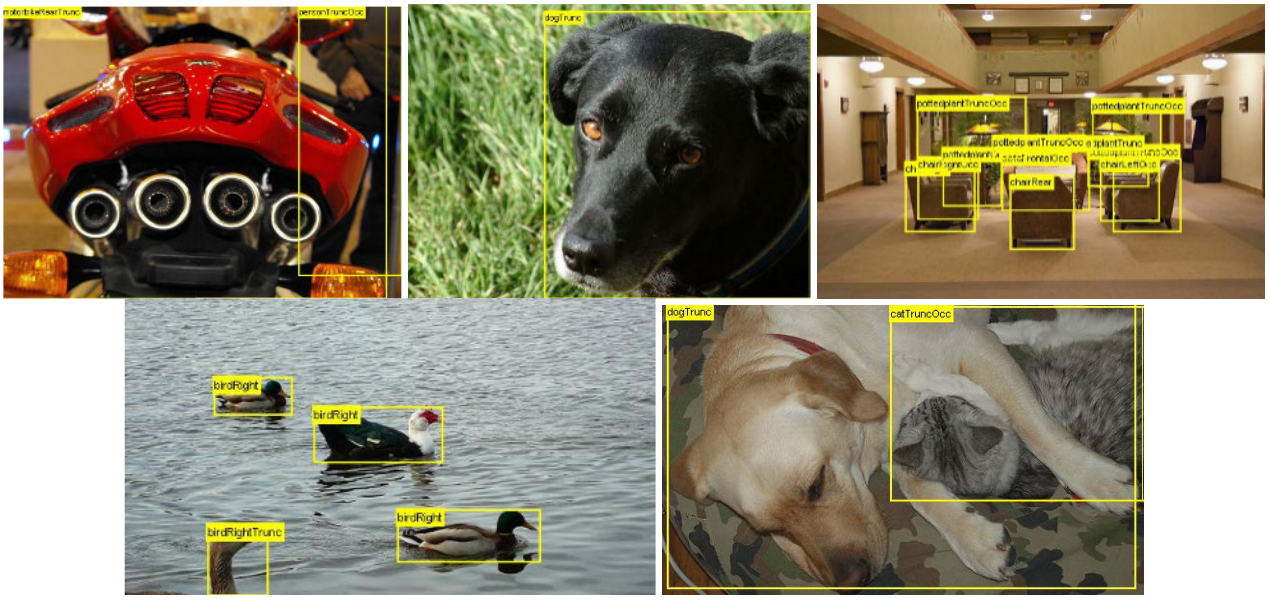


**FIGURE 2.** Samples from Pascal VOC 12.

"hard" negatives. 2) The second step, feature extraction, analyzes the generated regions to extract visual features or image patterns. With traditional object detection techniques, designing these features for the algorithm's performance is vital. To do this, we apply methods such as Haar-Like features [46], HOG [37], Scale-Invariant Feature Transform (SIFT) [47], Speeded Up Robust Feature (SURF) [48], and Binary Robust Independent Elementary Features (BRIEF) [49]. 3) Finally, the last step consists in classifying these entities, regardless of whether they contain an object or not, by using classification algorithms such as Support Vector Machine (SVM) [50], Adaboost [51], Deformable Part-based Model (DPM) [46] and K-Nearest Neighbors [52]. Three essential elements determine how well any object detection framework performs: the feature set, the classifier, the learning method, and the training set. In particular, most traditional methods that have been most efficient in recent PASCAL VOC detection challenges [53] have used several feature channels combined with detectors that include multiple aspects and mobile parts.

**FIGURE 3.** Samples from MS-COCO.

During 2008-2012, experiments conducted on PASCAL VOC using these traditional methods had become marginal, with minor improvements; this has highlighted the shortcomings of traditional detectors and the need to develop more robust approaches. The issue with traditional techniques such as those mentioned above that use sliding windows, for instance, where a rectangle of various sizes slides across the entire image trying to locate appropriate objects, requires a high level of computation effort, and it generates more duplicate windows. The work of the subjacent classifier crucially influences the overall output. Traditional approaches in object detection have been based on how we could manually design the features or the model according to our understanding. We attempt to search for patterns and edges through filtered images to describe them as features and classify them. Nevertheless, according to the most recent advances, it is most efficient to delegate such tasks to the computer so that they can learn for themselves. Following the ImageNet Large Scale Visual Recognition Competition (ILSVRC) launch in 2010 [54], the classification error rate for this competition was approximately 26% in 2011. After one year, in 2012, the error rate dropped to 16.4% due to a convolution neural network model called AlexNet [3]. Its architecture is close to Yann LeCun's LeNet-5 [55]. As a result, this was a critical opening for convolutional neural networks during this period. In the coming years and since 2012, convolution neural networks have won the battle, and the classification error rate for ILSVRC has been drastically reduced.

## III. DATASETS AND EVALUATION METRICS

Several datasets are available to support object detection challenges, and each object detection model is evaluated on these challenges' datasets. These datasets vary according to different perspectives regarding the number of images and outputs per image, the number of labeled classes, and image size. Some key performance metrics have been implemented for the spatial position and the predicted classes' accuracy.

### A. DATASETS

This paper compares all the object detection algorithms based on deep learning in the three most popular benchmark datasets. PASCAL VOC 2007, PASCAL VOC 2012, and Microsoft COCO, the ImageNet dataset, were not used due to their huge size, which necessitates a very high computing power for training.

#### 1) PASCAL VOC

PASCAL Visual Object Classification (PASCAL VOC) 2007 and 2012 is a familiar and widely used dataset for object detection with about 10,000 training and validation images with objects and bounding boxes. There are 20 different categories in the PASCAL VOC dataset.

#### 2) MS-COCO

The common Objects in COntext (COCO) dataset was developed by Microsoft and described in detail [56]. The COCO training, validation, and test sets include over 200,000 images and 80 object categories.

#### 3) ILSRVC

The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [220] is also one of the most well-known data sets in the object detection field. It started in 2010 as an annual challenge for object detection evaluation and continued

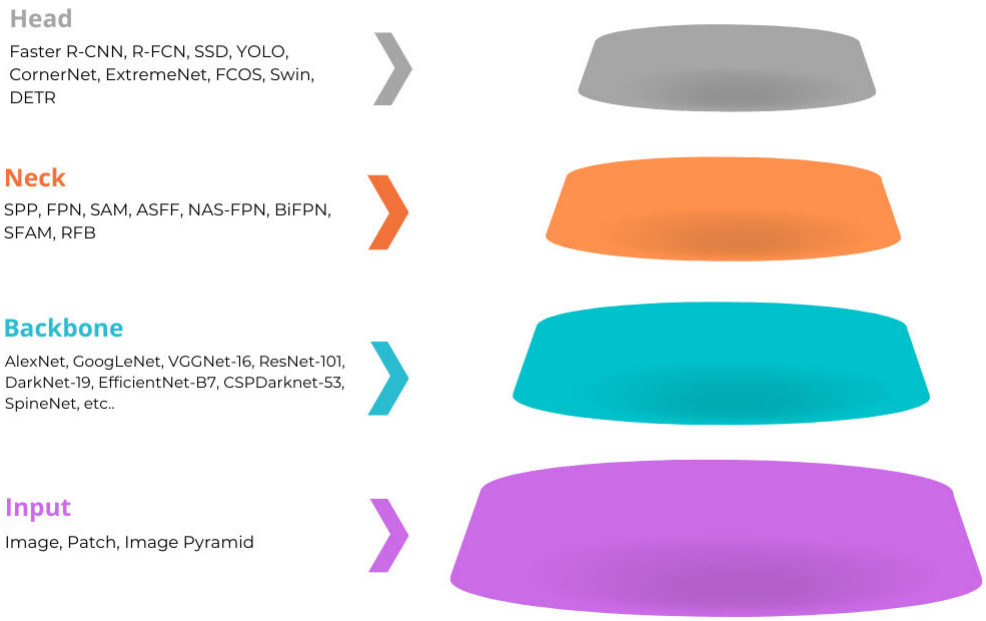


**FIGURE 4.** The components of an ordinary object detection model.

**TABLE 1.** An overview of methods, datasets, and evaluation metrics.

| Dataset | Total images | Classes | Train/Images | Train/Objects | Validation/Images | Validation/Objects | Test/Images |
|---|---|---|---|---|---|---|---|
| Pascal VOC 07 | 5011 | 20 | 2,501 | 6,301 | 2,510 | 6,307 | 4,952 |
| Pascal VOC 12 | 11,540 | 20 | 5,717 | 13,609 | 5,823 | 13,841 | 10,991 |
| MS-COCO | +328,000 | 80 | 118,287 | 860,001 | 5,000 | 36,781 | 40,670 |
| ILSRVC | +14M | 200 | 456,567 | 478,807 | 20,121 | 55,501 | 40,152 |
| Open Images | +9M | 600 | 1,743,042 | 14,610,229 | 41,620 | 204,621 | 125,436 |

until 2017. The dataset is composed of 1000 object classification classes making a total of more than 1 million images, of which half of which is dedicated to the detection task. There are about 200 object classes for the detection task.

#### 4) OPEN IMAGES

Open Images [221] is a dataset introduced by Google under the Creative Commons Attribution license. It comprises about 9.2 million labeled and unified ground-truth images and segmentation masks. This database has about 600 object classes with almost 16 million bounding boxes. It is considered one of the largest databases for object localization.

### B. EVALUATION METRICS

To evaluate the performance of object detection models, scientists have implemented several metrics to make the evaluation and comparison between these models more relevant and fairer. Several metrics, such as Intersection over Union (IoU), Frame Rate per Second (FPS), Precision, Recall, AUC, ROC, and RP curves, have been deployed. For example, a primary metric that is often expected in the field of object detection is IoU. IoU is a metric designed to measure the detection quality by calculating the difference between the ground truth annotations and the predicted bounding boxes. Usually, an object detection model generates several bounding boxes for each detected object. Through IoU and the threshold we set, we can eliminate some bounding boxes that fail to appear more accurate. An IoU value close to 1 indicates that the detection is more accurate.

$$IoU = \frac{\text{Area of union}}{\text{Area of intersection}}$$

As mentioned, Pascal VOC and MS-COCO are the reference datasets for testing and evaluating object detection models. Both challenges rely on mean average precision as the primary metric for evaluating object detector methods. However, there are still several differences in their definitions and implementations. An additional evaluation metric, mean average recall, is also applied for the MS-COCO Object Detection Challenge.

#### 1) MEAN AVERAGE PRECISION

The mAP value is the mean average precision of all K classes. The average precision (AP) is derived from the precision-recall curve, calculated for all unique recall levels. The method of computing AP by the PASCAL VOC challenge has changed since 2010. PASCAL VOC Challenge interpolates through all data points, compared to only 11 equidistant points. mAP evaluates the regression and classification accuracies.

#### 2) MEAN AVERAGE RECALL

The mAR value is the mean value of the RAs for all K classes. As AP, the average recall (AR) also represents a numerical metric to compare the detector's efficiency. AR is the mean recall on all IoU values within the [1, 0.5] interval and can be calculated as twice the area under the IoU recall curve.

**TABLE 2. An overview of methods, datasets, and evaluation metrics.**

| Ref | Dataset | Evaluation metric |
|---|---|---|
| [53] | The PASCAL VOC Challenge | mAP |
| [56] | The COCO Object Detection Challenge | mAP, mAR |

A standard object detection model is divided into four main parts: the input, the backbone, the neck, and the head. The input can be represented by a single image, a patch, or a pyramid of images. The backbone [57] can be a convolutional neural network like VGG [58], ResNet [59], EfficientNet [60], SpineNet [61], CSPDarkNet [62], etc. Then there is the neck which is a network found at the top of the backbone; this network is usually composed of many downstream and upstream paths such as FPN [63], NAS-FPN [64], ASFF [65], PAN [66] and BiFPN [67] or in the form of additional blocks such as SPP [68], RFB [69] and SAM [70]. As for the heads, they can be classified into two categories: those responsible for dense prediction, such as RetinaNet [71], YOLO [72], SSD [73], CornerNet [74], and FCOS [75]. And those responsible for sparse prediction like Faster R-CNN [76], Mask R-CNN [77], and RepDet [78].

## IV. BACKBONE NETWORKS FOR OBJECT DETECTION

Regarding object detection and building a robust object detector model, one of the most important factors that should be considered is the backbone network design. The backbone for object detection is a convolutional neural network designed to provide the foundation for an object detector. The backbone network's primary purpose is to extract features from the images before submitting them for further steps, such as the localization phase in object detection. There are several standard convolutional neural network backbones used by object detectors, including VGGNets, ResNets and EfficientNets, etc., which are pre-trained for classification tasks.

### A. ALEXNET

AlexNet [3] is a convolutional neural network (CNN) architecture developed in 2012. It consists of eight layers: five convolutional layers, two fully connected hidden layers, and one fully connected output 1000-way softmax classifier layer. AlexNet was the first CNN to win ImageNet Large Scale Visual Recognition Challenge and is a leading architecture for any object-detection task. It uses ReLU activation functions and local response normalization layers.

### B. VGGNETS

VGGNet [58] is a convolutional neural network architecture developed in 2014. It uses profound architecture with multiple convolutional and fully connected layers. It consists of five convolutional layers followed by three fully connected layers. The VGGNet architecture is known for its use of small convolutional filters ($3 \times 3$) and a very deep network with 16 to 19 layers. It uses ReLU activation functions and finishes with a softmax classifier. The main idea behind this architecture is to use very small filters ($3 \times 3$) to capture fine details in the images and stack multiple layers to increase the depth of the network; this way, it can learn more complex features.

### C. RESNETS

ResNet (Residual Network) [59] is an architecture designed and published in 2015. It is known for its ability to train profound networks without the problem of vanishing gradients, which is a common issue in very deep networks. The original paper on ResNet proposed five different sizes of the model: 18, 34, 50, 101 and 152 layers. Since then, many other variants of ResNet have been developed, such as ResNeXt and Wide Residual Networks (WRN). For example, ResNet-34 uses a plain network architecture inspired by VGG-19, adding shortcut connections. These shortcut connections allow the model to skip layers without affecting performance. The critical innovation of ResNet is the introduction of residual connections, which allow the network to learn the residual mapping between the input and the output of a layer rather than the original mapping. The residual connections allow the network to propagate gradients more quickly and allow for the training of much deeper networks. The resNet architecture uses a building block called "Residual Block," which contains multiple convolutional and batch normalization layers. The final layer is connected to a fully connected layer to classify the images.

### D. INCEPTION-RESNET

Inception-ResNet [228] is a convolutional neural architecture that builds on the Inception family of architectures developed by Google in 2016 but incorporates residual connections similar to the ResNet architecture to improve the flow of gradients and allow for the training of deeper networks. The Inception architecture is known for its use of multiple parallel convolutional and pooling layers, also called "Inception modules." Those modules extract features at different scales and then concatenate them before passing them to the next layer. It is 164 layers deep and trained on over a million images from the ImageNet database. The final layers are

**TABLE 3. Advantages and limitations of the object detector backbone.**

| Year | Backbone | Key features and advantages | Limitations |
|---|---|---|---|
| 2012 | AlexNet | -Introduction of consecutive convolutional layers.<br>-Great use of the downsampling.<br>-Non-linearity due to the use of Rectified Linear units.<br>-Fewer parameters and low computational complexity. | -Using large receptive fields.<br>-Low accuracy<br>-Memory-intensive due to overlapping blocks of pixels.<br>-Specific to certain applications. |
| 2014 | VGGNets | -Deep networks compared to AlexNet.<br>-Application of very small convolutional filters.<br>-Generalizes well across different datasets. | -A large number of parameters.<br>-Large size.<br>-Slower to train.<br>-Exploding gradient problem.<br>-Specific to particular applications. |
| 2015 | ResNets | -Introducing the identity and projection shortcut convolution to address the vanishing gradient problem.<br>-Application of batch normalization.<br>-Application of skip connections.<br>-Fewer parameters.<br>-Faster and smaller size compared to VGG.<br>-Generalizes well across different datasets.<br>-Fast training. | -Computationally heavy.<br><br>-Requires significantly more FLOPS than similar models.<br>-Complex architecture compared to VGGnets.<br>-Specific to particular applications. |
| 2016 | Inception-ResNet | - Application of residual inception blocks rather than Inception modules.<br>- Combining the Inception architecture with residual connections.<br>-Achieves better accuracy than Inception alone. | - Computationally expensive.<br><br>- Specific to certain applications and use cases. |
| 2019 | EfficientNets | -Introduction of the compound scaling method.<br>-Generalizes well across different datasets.<br>-Improved accuracy.<br>-More efficient. | - One-dimensional scaling.<br>-Higher computing and data movement costs. |
| 2015 | GoogLeNet | -Faster.<br>-Based on the Inception architecture [120][224]<br>-Application of dense modules.<br><br>-Not using fully connected layers.<br>-Fewer parameters and low computational complexity.<br>-Smaller pre-trained size. | -Requires more time for training.<br>-Complex architecture.<br>-Poor performance in face recognition compared to AlexNet, VGG-Face, and SqueezeNet. |
| 2019 | CSPResNeXt | -Address the duplicate gradient information problem.<br>-Reduces the memory footprint.<br>-Fewer parameters and low computational complexity.<br>-Better inference rate.<br>-Improve accuracy. | -The CSP block is complex.<br>-More raining time. |
| 2016 | DenseNet | -Application of dense connections to improve the gradient flow.<br>-Reduce the vanishing gradient problem.<br>-Reuse of features.<br>-Strengthening feature propagation<br>-Fewer parameters and low computational complexity. | -Replication of data several times.<br><br>-More memory usage. |
| 2018 | SENet | -Introduction of the attention mechanism.<br>-Application of the dynamic channel-wise feature recalibration.<br>-Improve the representational capacity of the network.<br>-Feature Recalibration.<br>-Lightweight. | -Slower than ResNets.<br>-Training time.<br><br>- Can be affected by noise in the input data.<br>-Not preserving the most activating pixels. |
| 2016 | Hourglass | -Stacked structure.<br>-Capturing both fine and coarse features<br>-More accurate in human pose estimation.<br>-Have a lightweight version. | -Complex architecture.<br>-High computational cost.<br>-Less accurate in some use cases. |
| 2020 | SpineNet | -Preserving spatial information<br>-Application of a scale-permuted network.<br>-Reduce the number of parameters.<br>-Detection of complex nonlinear relationships.<br>-More efficient. | -Complex architecture.<br>-Learning universal representations.<br>-Specific to certain applications and use cases.<br>-Needs more formal statistical training.<br>-Requires more evaluations across multiple applications and datasets. |
| 2020 | CSPDarknet | -Capturing both fine and coarse features.<br>-More efficient.<br>-Lightweight.<br><br>-Can be used in real-time applications | -The CSP block is complex<br>-More training time.<br>-Requires more evaluations across multiple applications and datasets. |
| 2022 | ConvNeXt | -Better accuracy and scalability.<br>- Fewer activation functions and normalization layers.<br><br>-Simple to fine-tune at different resolutions.<br>-Fully convolutional network.<br>-Outperforms ViTs and Swin Transformers. | -Slower and consume more memory.<br>-Depth-wise convolutions are slower and consume more memory than dense convolutions. |

connected to a fully connected layer to classify the images. The network has a similar architecture schema to Inception-v4, but the difference lies in their stems, Inception, and Residual blocks. The model has achieved excellent performance at a relatively low computational cost.

### E. EFFICIENTNETS

EfficientNet [60] is a convolutional neural network and scaling method published in 2019 that uniformly scales all dimensions of depth/width/resolution using a compound scaling approach. This allows the network to balance accuracy and computational efficiency better. It uses a building block called mobile inverted bottleneck convolution (MBConv), which combines depthwise and pointwise convolutional layers. It is similar to MobileNetV2 and MnasNet but is slightly larger due to an increased FLOP budget. The final layers are connected to a fully connected layer to classify the images. EfficientNet-B0 is the base model with a similar architecture as other architectures such as ResNet and VGG, but as the number increases, such as EfficientNet-B7, the architecture becomes more complex, with more layers, more filters, and higher resolution input images.

### F. GOOGLENET

GoogLeNet [120], also known as Inception v1, is a convolutional neural network architecture based on the Inception architecture that Google developed in 2014. It utilizes Inception modules, allowing the network to choose the best filters for a given input. GoogLeNet is 22 layers deep, with 27 pooling layers, and consists of 9 inception blocks arranged into three groups with max-pooling in between, also called ''Inception modules.'' Those modules extract features at different scales and then concatenate them before passing them to the next layer and global average pooling at the end. The GoogLeNet architecture won the 2014 ImageNet Large Scale Visual Recognition Challenge (ILSVRC) competition.

### G. CSPRESNEXT

CSPResNeXt [62] is a convolutional neural network where the Cross Stage Partial Network (CSPNet) approach is applied to ResNeXt. CSPNet uses cross-stage partial connections to bypass some of the network's layers, improve the flow of gradients and allow for the training of deeper networks. A residual Network with Extreme cardinality or ResNeXt is an architecture that uses a building block called ''ResNeXt Block,'' which contains multiple branches of convolutional layers with different numbers of filters, allowing the network to learn features at different scales and increases the capacity of the network. CSPResNeXt is used as a feature extractor in YOLO v4 and partitions the feature map into multiple stages, allowing for better learning capability of CNNs.

### H. DENSENETS

DenseNet [229] is a network that uses dense connections between layers through Dense Blocks, which contain multiple convolutional layers, batch normalization, and ReLU activation. The final layers are connected to a fully connected layer for image classification. The architecture is characterized by its use of dense connections, which connect each layer to every other layer in a feed-forward fashion, which draws representational power from feature reuse instead of extremely deep or wide architectures. Each layer is connected directly with every other layer in the network, creating a dense connectivity pattern that allows the network to propagate the gradients through the network more efficiently and effectively, enabling the training of deeper networks. The architecture allows for a significant reduction in the number of parameters compared to traditional architectures.

### I. SENET

SENet [230] (Squeeze-and-Excitation Network) is a convolutional neural network architecture published in 2017. The architecture employs squeeze-and-excitation blocks to enable the network to perform dynamic channel-wise feature recalibration. This feature improves the feature representation capabilities of CNNs. The architecture uses a building block called ''SE block,'' which contains two sub-layers: a ''Squeeze'' layer, which reduces the dimensionality of the feature maps, and an ''Excitation'' layer, which adaptively recalibrates the feature maps. The Squeeze layer applies global average pooling to the feature maps to obtain a channel descriptor, which is then passed through a fully connected layer, also called a bottleneck layer, to reduce the dimensionality of the descriptor.

### J. HOURGLASS

Hourglass [231] architecture is a convolutional neural network (CNN) used for human pose estimation, object detection, and semantic segmentation tasks. The architecture is characterized by its repeated bottom-up and top-down processing, similar to an hourglass shape, which allows the network to learn the input's fine and coarse features. The Hourglass architecture consists of several modules stacked on top of each other. Each hourglass module is a sub-network consisting of convolutional and pooling layers at the top, followed by up-sampling and convolutional layers at the bottom that reconstruct the input feature maps. These modules are connected in a ''skip'' or ''residual'' connection fashion, allowing information to flow from one module to the next.

### K. SPINENET

SpineNet [61] is a convolutional neural network backbone with scale-permuted intermediate features and cross-scale connections learned on an object detection task developed by Google AI in 2020. It typically encodes an input image into a series of intermediate features with decreasing resolutions. The architecture of SpineNet is based on the idea of a ''sparse backbone,'' which is composed of a sequence of sparse convolutional layers, called ''spine layers,'' that are interleaved with dense layers called ''non-spine layers.'' The spine layers are lightweight, with fewer parameters and

computation, while the non-spine layers are more complex, with more parameters and computation.

### L. CSPDARKNET

CSPDarknet [199] is a convolutional neural network and backbone for object detection developed in 2020. It is based on the architecture of the Darknet, and It employs a CSPNet strategy to partition the feature map, which includes an activation function and attention mechanism. The architecture uses a series of CSP blocks with an increasing number of layers; the output of each block is concatenated with the output of the corresponding block in the previous stage; this allows the network to learn fine and coarse features the input. The final output is obtained by applying convolutional layers on the feature maps generated by the last CSP block.

### M. CONVNEXT

ConvNeXt [232] is a pure convolutional model inspired by the Vision Transformers design. ConvNeXt is built entirely from standard ConvNet modules. It retains the efficiency of standard ConvNet while being fully convolutional for learning and testing, making it simple to implement. ConvNeXt has fewer activation functions and normalization layers than other backbone networks and separates the downsampling layer. The model was evaluated on various vision tasks such as ImageNet classification and object detection. It showed higher performance in all major benchmarks. ConvNeXt uses convolutions that operate on a per-channel level by shuffling only the information in the spatial dimension. Depth convolutions are clustered convolutions where the number of clusters equals the number of input channels. In ConvNext, depth convolutions are used in MobileNet and later in EfficientNet.

## V. DATA AUGMENTATION

During training, models adopt different learning strategies such as localization refinement, data augmentation, cascade learning, and Imbalance sampling. Those strategies help the models work efficiently to improve the accuracy and execution time for both localization and classification tasks. For example, data augmentation is one of the most efficient techniques to improve model results because it does not add any inference complexity. However, still not commonly used due to the complexity of designing methods that can efficiently handle both the localization and classification by transferring strategies between the two. Several augmentation techniques include color space, cropping, rotation, translation, Kernel filters, Random erasing, noise injection, color jittering, mixing images, and flipping.

### A. COLOR SPACE

Color space is a popular augmentation method when dealing with digital RGB images. The augmentation is performed on the R, G, and B channels by selecting a single channel and adding two more zero matrices. It is also possible to apply other color enhancements, such as contrast, brightness, and saturation, by manipulating the values of the RGB matrices.

### B. ROTATION

The rotation method rotates the image to the left and right at an angle between 1° and 359°. The degree of rotation is an essential factor to consider in this method. The extent of rotation is chosen according to the image type and the problem to solve. For example, a slight rotation of about 20° is used in applications related to number detection, such as MNIST. Whereas using an extensive orientation, one risks losing the label value of the image.

### C. TRANSLATION

The basic concept of this augmentation is to shift the images in four directions, top, bottom, right and left. This technique allows the preservation of the spatial dimension of the image by filling the remaining space after translation with constant values like 0 and 255 or through random and Gaussian noise.

### D. CROPPING

The cropping technique is often used when a data set has different heights and widths. The technique is used to crop the central patch of each image. Random cropping is almost similar to the translation technique, except that translation preserves the spatial dimension of the image, whereas random cropping reduces the size of the image.

### E. KERNEL FILTERS

This augmentation technique is commonly used to clean or blur images by applying filters. The process of kernel filters is similar to that of convolutional neural networks. The idea is to either slide a matrix with a Gaussian blur filter onto the image to produce a blurred image or to slide the matrix with a high-contrast vertical or horizontal edge filter to produce a sharper image.

### F. RANDOM ERASING

Random erasing [222] is an augmentation method inspired by the dropout regularization mechanism and aims to solve the occlusion problem encountered in image recognition problems. Random erasing assists the model in avoiding the occlusion problem and overfitting by forcing the model to learn the defining features of the image. The operating mechanism of this method involves randomly selecting a patch in the image and masking it with average pixel values, zeros, or 255s.

## VI. ANCHOR-BASED DETECTORS

The anchor boxes represent a predefined collection of bounding boxes with selected widths and heights that reflect the widths and heights of the objects in the training data set. They also include, of course, various aspect ratios and scales found in the dataset. When detecting, the predetermined anchor boxes are arranged in a tiled pattern on the image. Moreover, the same anchors are constantly proposed on each image. Instead of predicting the boxes, the network predicts the probability and other attributes, such as background, intersection on union (IoU), and offsets for each tiled anchor box.

**TABLE 4. Advantages and limitations of two-stage detectors.**

| Year | Method | Advantages | Limitations |
|---|---|---|---|
| 2013 | R-CNN | -Simple to use.<br><br>-Application of convolutional neural networks for classification.<br>-It has formed a foundation for future developments. | -High time consumption during the training phase due to 2000 regions to be classified.<br>-Duplicated computations.<br><br>-Cannot be applied in real-time applications as it takes around 47 seconds for one test image.<br>-The selective search prevents the algorithm from learning in the regional proposal phase.<br>-The absence of an end-to-end training pipeline. |
| 2014 | SPPNet | -Multi-scale input to the feature extractor through applying the spatial pyramid layer.<br>-Faster than R-CNN due to the use of one convolutional layer. | - It is time-consuming because of the selective search mechanism.<br>-SPPNet does not provide an end-to-end learning architecture.<br>-The fine-tuning algorithm does not update the convolutional layers before SPP. |
| 2015 | Fast R-CNN | -Faster than SPPNet.<br>-Introducing ROI pooling layer for mapping various input sizes.<br>-Multi-task model by integrating bounding box regression along with classification.<br>-The ability to use an RPN +CNN pattern. | -Selective search algorithm.<br>- Region proposals turn into bottlenecks affecting its performance. |
| 2015 | Faster R-CNN | -With RPN instead of selective search, generating regional proposals requires significantly less time.<br>-Introducing anchor boxes.<br><br>-Multi-task loss.<br><br>-High performance in terms of accuracy.<br>-End-to-end learning. | -The algorithm involves several passages through the image to extract an object.<br>-Given that many separate sequential systems are available, however, the model's performance through time is influenced by the performance of previous systems.<br>- Difficulties detecting small objects due to using a single map of deep layer features for final prediction.<br>-The class imbalance needs to be correctly addressed. |
| 2016 | R-FCN | -Good speed/accuracy trade-off compared t previous detectors<br>-Handles translation variance when detecting objects. | -It does not support the global average pooling because the fully connected layer has been eliminated. |
| 2016 | FPN | -Detecting objects on different scales.<br>-The class imbalance is well handled. | -It only supports one-way information flow.<br>-FPN is a partial object detection system. |
| 2018 | PANet | - Preserving spatial information accurately<br><br>-Very fast and straightforward compared to Mask R-CNN, G-RMI, and RetinaNet<br>-Used in real-time detection models such as YOLOv4. | -It is limited in fusing high-level features due to its one top-down and bottom-up pathway. |
| 2019 | TridentNet | -Dealing with scale variation.<br>- The use of trident blocks and dilated convolutions produce large receptive fields. | - Very Slow |
| 2020 | SpineNet | -Great accuracy due to scale-permuted model.<br>-Can be used for image classification<br>- Can be used for real-time detection with SpineNet-49 and SpineNet-49S. | -Large training time |
| 2021 | Copy-Paste | -Greater accuracy<br><br>-Simple to integrate into any instance segmentation. | -Randomness in data selection prevents the model from selecting more realistic data. |

It returns a unique collection of predictions for each set anchor box. Generating bounding boxes can be described as follows: (1) Create thousands of candidate anchor boxes that best describe the objects' size, location, and shape. (2) Predict the offset for each bounding box. (3) Compute a loss function for each anchor box based on ground truth. (4) For each anchor box, compute the Intersection Over Union (IOU) to check which object's bounding box has the largest IOU. (5) When the probability is more significant than 0.5, notify the anchor box that it should detect the object with the highest IOU. and factor the prediction into the loss function. (6) If this probability is marginally less than 0.5, we instruct the anchor box not to learn from this sample since the prediction is ambiguous; otherwise, if the probability is remarkably less than 0.5, then the anchor box is likely to predict that no object is present. Finally, by using this process, we ensure that the model learns to identify only true objects. Using anchor boxes allows a network to detect multiple objects, objects of different scales, and overlapping objects. In object detection, anchor-based detectors define anchor boxes at each position in the feature map. The network predicts the probability of objects in each anchor box and then fits the size of the anchor boxes to fit the object. However, anchors require careful design and application in object detection frameworks. (a) The coverage ratio of the instance's location space is among the most critical factors

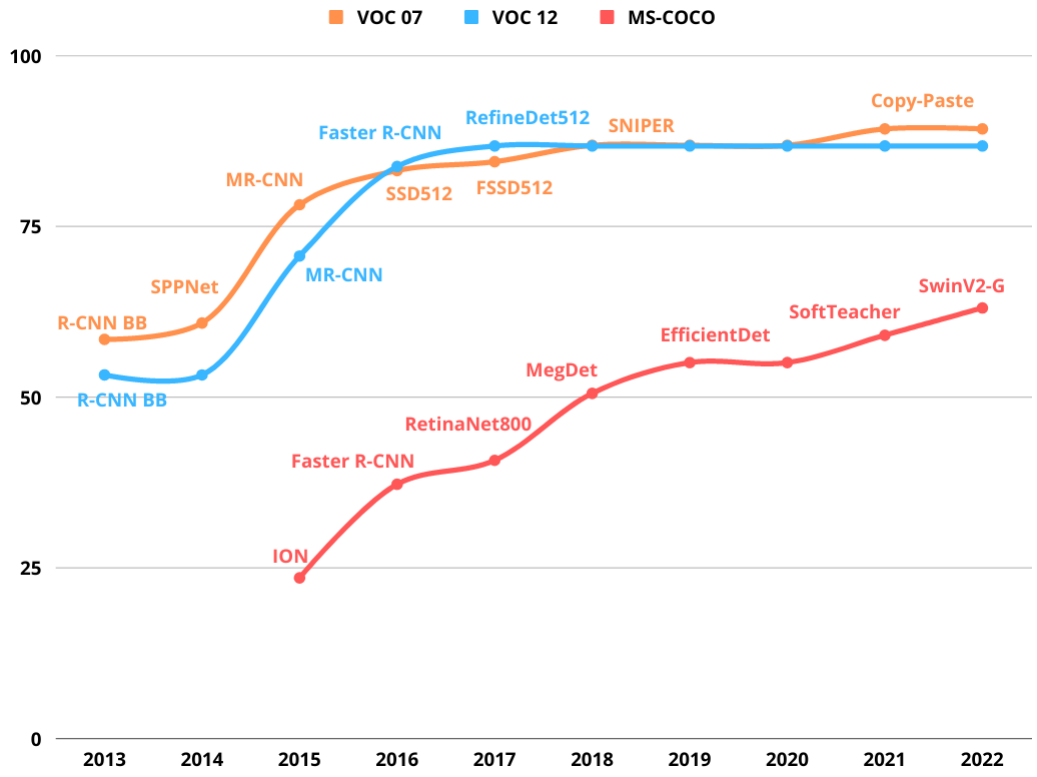


**FIGURE 5.** Accuracy evolution in the main object detection benchmarks.

in anchor design. To ensure a good recall rate, anchors are thoroughly engineered based on the statistics computed from the training/validation set [79], [80]. (b) Some design choices based on a particular dataset may not apply to other applications, which affects the generality [81]. (c) During the learning phase, the anchor-based approaches rely on intersection union (IoU) to define the positive/negative samples, thus adding extra computation and hyper-parameters for an object detection system [82]. Anchor-based object detection frameworks generally fall into two sections: two-stage, proposition-based detectors and one-stage, proposition-free methods. 1) Two-stage object detection. 2) One-stage object detection. The anchors serve as regression references and classification candidates for predicting proposals for two-stage detectors and final bounding boxes for one-stage detectors.

### *A. TWO-STAGE METHODS*

Region-based object detection algorithms were among the most widely used techniques for detecting objects in images. The first models in object detection start intuitively by searching the regions and then performing the classification. The two-stage methods are derived from the R-CNN methods that extract RoI using a selective search method [83] and then classify and regress them. Faster R-CNN [76] is the most well-known two-stage anchor-based detector reference. It uses a separate region proposal network (RPN) that generates RoI by modifying predefined anchor boxes and a region-based prediction network (R-CNN) [84], [85] to detect objects. Many models were subsequently introduced to improve its performance. For example, using bilinear interpolation, the Mask R-CNN [77] replaces the RoIPool layer with the RoIAlign layer. Other models look at different aspects to improve performance. For example, some target the whole architecture, such as [86] and [89], some use multi-scale learning and testing [90], [91], others feature fusion and enhancement [63], [92], the introduction of the new loss function and training [93], [95], and some employ better proposal and balance [96], [97]. In contrast, others apply context and attention mechanisms. Specific models also employ different learning strategies and loss functions.

#### 1) REGION-BASED CONVOLUTIONAL NEURAL NETWORKS (R-CNN)

Instead of processing many regions, the R-CNN [85] model developed by R. Girshick et al. in 2014 proposes many boxes in the image and checks whether one contains an object. The R-CNN relies on a selective search [83] method developed by J.R.R. Uijlings et al. in 2012, a variant of the exhaustive image search to extract these boxes from an image. These boxes are called regions. Selective search is used to extract 2000 region proposals; those candidate region proposals are cropped and resized to fit the input of the CNN feature extractor, where they extract a 4096-dimensional vector of features transmitted into several classifiers for class prediction. SVMs [50] are

assigned to each class to classify the object's occurrence in the proposed candidate region under a given feature vector. It also has a linear regressor that predicts four offset values to enhance the selected bounding boxes' accuracy and minimize localization errors. The R-CNN consisted of three simple steps: Scan the input image to detect objects that may be present using the selective search algorithm by proposing approximately 2000 candidate boxes. Then for every candidate box, we apply a CNN for feature extraction. The result of each CNN is transmitted to an SVM for classification and for a linear regressor to refine the object's bounding box. R-CNN is very easy to use but very slow.

#### 2) SPATIAL PYRAMID POOLING NETWORK (SPPNet)

Based on the concept of spatial pyramid matching (SPM) [98], SPPNet [68] is mainly an improved version of R-CNN [85]. SPPNet has implemented a specific CNN procedure known as Spatial Pyramid Pooling (SPP) during the passage of the convolutional layer and the fully connected layer. In transitioning from the convolutional layer to the fully connected layer, it proposes having multiple pooling layers at different scales instead of a single pooling layer often used as a standard in other methods. A selective search algorithm is applied by SPPNet to generate about 2,000 region proposals per image, just like R-CNN. Next, it extracts features directly from the whole image using ZFNet [99] only once. At the final conv layer, the feature maps delineated by each region proposal pass through the SPP layer, followed by the fully connected layer. Each bounding box has its SVM and bounding box regressor. SPPNet uses the SPP for every region proposal to pool the features of that region from the global feature volume to produce its fixed-length representation. SPP solves the problem of cropping the image before entering CNN at a fixed size, as in R-CNN with VGG [58], where image sizes are fixed (224*224). Thus, with SPP, the images can be used with different shapes. In contrast to R-CNN, SPPNet only deals with the image at the convolutional layers once, whereas R-CNN deals with the image at the convolutional layers at least 2000 times. As shown in Table 4, SPPNet is much faster and more accurate than R-CNN.

#### 3) FAST REGION-BASED CONVOLUTIONAL NETWORK (FAST R-CNN)

To solve some of the problems of R-CNN and SPPNet and to develop a faster object detection algorithm, Ross Girshick published a new paper named Fast R-CNN [84]. Comparing Fast R-CNN with SPP-net, one can observe that the SVM classifiers have been removed, and a regression and classification layer has been connected to the network. VGGNet is used instead of ZFNet, the region of interest (RIO) poling layer, rather than the SPP. On the other hand, Fast R-CNN is similar to the original R-CNN in many ways. However, two major additions have improved its detection speed: They extract the image features before proposing regions rather than forwarding the region proposals to the feature extractor. Thus, a single CNN is applied to the entire image rather than 2000 CNNs on 2000 regions. The SVM is also changed to a softmax layer, extending the neural network as a prediction model rather than building a new one. The primary CNN with several convolutional layers takes the entire image as input rather than applying a CNN for every region proposal. As a result, the region proposals are based on the last feature map. Therefore, they can build a single CNN for the entire image. Regions of interest (RoI) are detected by applying the selective search method to the feature maps produced. The proposal region is formally resized using an RoI pooling layer to obtain a valid region of interest that can be introduced into a fully connected layer. Fast R-CNN uses a softmax layer instead of many different SVMs to predict the class directly for each region proposal and the offset values for the bounding boxes. Therefore, we have only one neural network to train, compared to one neural network and many SVMs. Fast R-CNN uses a multi-task loss function that combines classification and regression losses. The classification loss is computed using the log loss function over two classes. The regression loss is computed using the L1 smooth loss function.

#### 4) FASTER REGION-BASED CONVOLUTIONAL NETWORK (FASTER R-CNN)

The three algorithms mentioned above, R-CNN, SPPNet, and Fast R-CNN, are based on a selective search to identify region proposals. Selective search is a slow and time-consuming method that impacts network performance and was proven to be the bottleneck of the entire process. Thus, the authors of Faster R-CNN [76] proposed a framework for object detection to replace the selective search algorithm and allow the network to discover region proposals. The Faster R-CNN point was that the region proposals depended on the image features previously calculated with the CNN forward passage (the first step in the classification). They have developed a region proposal network (RPN) [76] to generate region proposals directly, then predict bounding boxes. An RPN and Fast R-CNN model combined in Faster R-CNN [58]. Faster R-CNN takes the CNN feature maps and forwards them to the region proposal network. RPN utilizes a $3 \times 3$ sliding window that moves across these feature maps. Each sliding window location generates multiple potential regions and scores based on k fixed-ratio anchor boxes. Now we have bounding boxes in various shapes and sizes passed to the RoI pooling layer. Consequently, it is possible that region proposals may have no classes assigned to them after the RPN step. So, we can crop each proposal to make each proposal region include an object. That is what the RoI pooling layer is for. It extracts fixed-size feature maps for each anchor. These feature maps are then transmitted into a fully connected layer comprising a softmax and a linear regression layer. It then classifies the objects and predicts the bounding boxes for the detected objects. Only one CNN is applied in Faster R-CNN for region proposals and classification. Faster R-CNN is optimized for a multitask loss function comprising classification and regression loss.

**TABLE 5. Advantages and limitations of one-stage object detectors.**

| Year | Method | Advantages | Limitations |
|---|---|---|---|
| 2016 | SSD | -End-to-end training.<br>-Better accuracy than YOLO.<br>-Faster than Faster R-CNN.<br>-SSD512 outperforms Faster R-CNN.<br>-Multiple scale feature extraction. | -More time-consuming than YOLOv1.<br>-Less accurate than Faster R-CNN. |
| 2016 | YOLOv2 | -Fixed the limitations of yolov1.<br>-More efficient than Faster R-CNN and SSD in real-time applications.<br>-Multi-scale training. | - Less accurate than its competitors SSD and RetinaNet. |
| 2017 | RetinaNet | -Combine the power of Focal loss and FPN.<br>-Address class imbalance efficiently.<br>-Detecting objects of various scales.<br>-Faster than Faster R-CNN, R-FCN, and SSD. | -Slower than YOLOv1. |
| 2018 | YOLOv3 | -More apt to detect small objects.<br>-Multi-scale prediction.<br>-More efficient than SSD. | -Less efficient than RetinaNet. |
| 2018 | MegDet | -Less training time | -Cannot meet real-time detection requirements. |
| 2020 | EfficientDet | - Fast fusion of multi-scale features.<br>-High efficiency due to the use of efficient backbones. | -Cannot meet real-time detection requirements. |
| 2020 | PAA | -More accurate due to an optimized anchor assignment strategy. | -Cannot meet real-time detection requirements. |

The Region Proposal Network (RPN) is a Convolutional Neural Network that proposes regions. At the same time, the second network is a Fast R-CNN for feature extraction and outputting the Bounding Box and Class Labels. The RPN is optimized for the given multitask loss function.

#### 5) REGION-BASED FULLY CONVOLUTIONAL NETWORK (R-FCN)

R-CNN-based detectors, such as Fast R-CNN or Faster R-CNN, detect objects in two phases. First, generate region proposals (ROI) and classify and localize objects from the ROI. These detectors save valuable time by sharing calculations of repeated convolutional features for object classification and region proposals. However, Faster R-CNN still contains several unshared R-CNN's fully connected layers that must be calculated for each of the hundreds of proposals. The Region-based Fully Convolutional Network (R-FCN) [100] is a framework that combines the two main phases in a single model to take into account both the detection of the object and its position simultaneously. It contains only convolutional layers that provide complete backpropagation for training and inference. As we have observed in the methods mentioned above, that region proposal is mainly generated by RPN. The ROI pooling is performed and passes across fully connected (FC) layers to classify and regress the bounding boxes. The post-ROI pooling is not shared between ROIs and takes a long time. As a result, the FC layers add more parameters to the model, which leads to more complexity. In R-FCN, there is still RPN for region proposals. However, unlike the R-CNN series, the FC layers after ROI pooling are eliminated. As an alternative, the objective complexity is moved before the ROI pooling to generate feature maps, each dedicated to detecting a category at a specific location. For instance, a feature map is dedicated to detecting a dog, another for detecting a car, Etc. These feature maps rely on an object's spatial localization, called position-sensitive score maps. After the ROI pooling, all these regions' proposals will use the same score maps to carry out the average voting, a simple computation. Consequently, there is no learning layer after the ROI layer; in other words, R-FCN is significantly faster than Faster R-CNN and has a highly reliable mAP. R-FCN operates as follows; the input image is processed by the backbone ResNet-101 [59] to generate feature maps. These feature maps are transmitted on the one hand to an RPN to produce RoI and, on the other hand, to a fully convolutional layer for generating a bank of position-sensitive score maps. To have a score map, $k^2$ $(C + 1)$, where $k^2$ is defined as the number of relative positions used to split an object in a grid. $C + 1$ is defined as the number of classes with a background. Afterward, on each ROI, they split it into the exact $k^2$ boxes or sub-regions as the scorecards. They check the score bank for each bin to ensure that it corresponds to the respective position of the object. In the upper left bin, for instance, we will search for the score maps that match the upper left corner of an object and average these values in the RoI region. The system performs this procedure for each class all over again. After each $k^2$ bin has a corresponding object value in each class, the $k^2$ bins are averaged to produce a unique score per class. They classify the RoI with a softmax on the remaining dimensional vector $C + 1$. They use convolution filters for the regression of the selection framework to generate the $k \times k \times (C + 1)$ score maps used for classification purposes. An additional convolution filter generates a four $\times$ k $\times$ k map based on the same feature maps. The loss function for R-FCN is defined on each RoI and is the summation of the cross-entropy loss and the box regression loss. The classification loss (Lcls) and bounding box regression loss (Lreg) are used in online hard example mining (OHEM).

#### 6) FEATURE PYRAMID NETWORKS (FPN)

The FPN [63] is not an object detector in itself. It is a feature detector that operates in combination with object detectors. For instance, with FPN, we can extract multiple feature map

layers and feed them into an RPN to detect objects. Compared to the feature extractor used in some frameworks like Faster R-CNN, FPN generates more layers of feature maps, multi-scale feature maps, and high-quality information than the standard feature pyramid used for object detection. Using FPN allows us to detect objects on various scales. The FPN consists of a bottom-up and a top-down pathway. The bottom-up pathway is the traditional convolution network for feature extraction and uses a ResNet [59]. The spatial resolution decreases as we move upwards; as more high-level features are detected, each layer's semantic value is enhanced. As a reference set of feature maps, the output of the last layer of each stage will be used to enhance the top-down pathway through the lateral connection. The top-down pathway allows for higher-resolution layers from a semantic-rich layer. Whereas the reconstructed layers are semantic, the locations of the objects after all sub-sampling and bottom-up sampling are inaccurate. The authors include lateral connections between the reconstructed layers and the associated feature maps to address this problem to predict the most appropriate locations. In the top-down pathway, an oversampling by a factor of 2 is performed on the spatial resolution using the nearest neighbor to simplify the process. For each lateral connection, feature maps of the same spatial size are merged from the bottom-up and top-down pathways. In more detail, the feature maps of the bottom-up pathway are convolved at $1 \times 1$ convolutions to minimize the channel size. Moreover, the bottom-up and top-down feature maps are combined by element-wise addition. Then, a $3 \times 3$ convolution is applied directly to each merged map to compute the final feature map, designed to minimize the frequency folding effect of oversampling. The final set of feature maps is called P2, P3, P4, P5, which refers to C2, C3, C4, C5, and both have the same spatial size, respectively.

#### 7) PANET

The Path Aggregation Network (PANet) [66] is a method mainly developed, for instance, segmentation, which inserts an additional upward path aggregation network above FPN. They provide an adaptive feature pooling that shortens the distance between the lower and topmost feature levels by grouping the features of all feature levels and avoiding arbitrarily assigned outputs. PANet allows the network to decide which features are useful. They use a complementary path to enhance the feature of each proposal by providing accurate localization signals in lower layers and generating a bottom-up augmentation. The PANet obtained an accuracy of 41.4 on the MS-COCO dataset compared to Mask R-CNN, which achieved only 36.4%. PANet uses ResNeXt-101 as a backbone.

#### 8) TRIDENTNET

The TridentNet model [89] proposes an approach to deal with the scale variations in object detection based on generating in-network scale-specific feature maps using uniform representational power. They build a parallel multi-branch architecture and apply scale-aware training, where each branch shares the same transformation parameters but with different receptive fields. The model applies a fast inference method with only one major branch to boost the model performance without using additional parameters and computations. The authors TridentNet achieved an mAP of 48.4 on the MS-COCO dataset with Resnet-101 as a backbone.

#### 9) SPINENET

SpineNet [61] is a classification and object detection model that uses Neural Architecture Search for learning in contrast to traditional encode-decoder architectures with scale-decreased backbone leading to ineffectiveness in generating multi-scale features. The SpineNet proposed method has a fixed stem network followed by scale-permuted intermediate features and cross-scale connections. The authors proposed many variants of SpineNet, such as SpineNet-49, SpineNet-143, and SpineNet-190. The latter obtained an AP of 52.2% on the MS-COCO dataset.

#### 10) COPY-PASTE

In [101], the authors applied the copy-paste data augmentation strategy and proved its effectiveness for object detection and instance segmentation. The copy-paste method chooses two images randomly and applies a random scale jittering and a horizontal flip. It generates new data by pasting objects from one image to another. In the final stage, they tune the ground truth annotations for the bounding boxes by eliminating all occluded objects. The authors provide a self-training Copy-paste where a supervised model is trained on labeled data, producing pseudo labels on unlabeled data. Combined with Cascade Eff-B7 NAS-FPN, they achieved an AP of 55.9% on the MS-COCO dataset.

### B. COMPARISON: TWO-STAGE DETECTORS

Table 4 lists a chronological comparison of the strengths and limitations of the two-step anchor-based detection methods mentioned earlier in this paper.

### C. ONE-STAGE METHODS

One-stage anchor-based detectors are characterized primarily by their computational and runtime efficiency. These models directly classify and regress predefined anchor boxes instead of using regions of interest. The SSD was this category's first well-known object detector [73]. The major challenge encountered in this type of detector is the imbalance between positive and negative samples. Several approaches and mechanisms have been implemented to overcome this problem, such as anchor refinement and matching [102], training from scratch [103], [104], multi-layer context information fusion [105], [107], and feature enrichment and alignment [69], [108], [111]. Other works have been directed toward developing new loss functions [79], [112] and new architectures [113], [114].

**TABLE 6. Advantages and limitations of anchor-free object detectors.**

| Year | Method | Advantages | Limitations |
|---|---|---|---|
| 2016 | YOLOv1 | -Very fast, it runs at 45 fps.<br>-End-to-end training.<br>-It has fewer localization errors compared to Faster R-CNN. | -Dealing with small objects.<br>-It likewise addresses the localization error of bounding boxes for small and large boxes.<br>-Difficulties in generalizing due to unseen aspect ratios.<br>-Coarse Features. |
| 2018 | CornerNet | -Competitive with traditional two-stage anchor-based detectors. | -Cannot meet real-time detection requirements. |
| 2019 | ExtremeNet | -It can also be used in instance segmentation. | -Cannot meet real-time detection requirements. |
| 2019 | RepDet | -End-to-end training.<br>-Effective and competitive with two-stage anchor-based detectors. | -Cannot meet real-time detection requirements. |
| 2019 | FSAF | -It can act as a module and be integrated with one-stage object detectors.<br>-Applies online feature selection instead to feature pyramids.<br>-Marginal computation cost<br>-Faster than RetinaNet | -Cannot meet real-time detection requirements. |
| 2019 | FCOS | -Simpler compared to other one-stage detectors.<br>-It detects objects using a per-pixel approach, as in the case of semantic segmentation.<br>-Can be used as a region proposal network i, Faster R-CNN | -Cannot meet real-time detection requirements. |
| 2020 | ATSS | -Increase the performance via the introduction of the Adaptive Training Sample Selection<br>-More accurate without using any overhead | -Cannot meet real-time detection requirements. |
| 2021 | OTA | -Deals with the label assignment issue as an optimal transport problem.<br>- More accurate than ATSS and FCOS | -Needs more time for training due to the Sinkhorn-Knopp Iteration algorithm<br>-Cannot meet real-time detection requirements. |
| 2022 | DSLA | -Deals with the inconsistency in object detection.<br>-Smooth label assignment<br>-The most accurate anchor-free detectors | -Cannot meet real-time detection requirements. |

#### 1) YOLOv2

YOLOv2, or YOLO9000 [80], published in 2017, is an object detection model capable of detecting more than 9,000 object categories in real-time. It has many updated features to fix the problems of the first version. The main improvements in YOLOv2 compared to YOLOv1 [72] are the application of batch normalization over the entire convolutional layers. Besides training 224 $\times$ 224 images, it uses 448 $\times$ 448 images to fine-tune the classification network over ten periods on ImageNet [115]. Using 416 $\times$ 416 images during training eliminates a pooling layer for better output resolution, removes all fully connected layers, and replaces them with anchor boxes for predicting bounding boxes. The model achieved 69.2% mAP and 88% recall with the anchor boxes; without them, it achieved 69.5% mAP and 81% recall. Although the mAP is slightly reduced, its recall has a high margin increase. As with Faster R-CNN [76], the anchor box sizes and scales were pre-set beforehand. YOLO9000 relies on k-means clustering to achieve interesting IOU scores because the standard Euclidean distance-based k-means often have additional errors when dealing with larger boxes. Using an IoU clustering approach with nine anchor boxes, Faster R-CNN obtained 60.9%, whereas YOLO900 achieved 67.2%. For YOLOv2, the location is defined by the logistic activation, thus reducing the value between 0 and 1, compared to YOLOv1, which has no constraints on the location prediction. YOLOv2 predicts multiple bounding boxes per grid cell. To compute the loss for the true positive, only one of them should be responsible for the object. For this purpose, the one with the highest IoU (intersection over union) with the ground truth is selected. YOLOv2 loss function has three parts: finding bounding-box coordinates, bounding-box score prediction, and class-score prediction. All of them are Mean-Squared error losses and are modulated by some scalar meta-parameter or IoU score between the prediction and ground truth.

#### 2) YOLO v3

The YOLO [72] algorithm uses a softmax function to convert the scores into probabilities equal to one. YOLOv3 [116] applies a multi-label classification, and the softmax layer is substituted with an independent logistic classifier to calculate the input's probability of being part of a particular label. Rather than applying the mean square error to compute the classification loss, YOLOv3 applies a binary cross-entropy loss for every label. In addition, it minimizes the cost complexity of calculations by bypassing the SoftMax function. It provides additional minor enhancements. It performs prediction at three scales, precisely by downsampling the input image dimensions by 32, 16, and 8, respectively. Darknet, in this version, has been extended to include 53 convolutional layers. Detections in several layers are a good solution for solving the problem of small object detection, a common concern with YOLOv2. YOLO v3 uses a total of 9 anchor boxes. Three per each scale. It relies on K-Means clustering to generate all nine anchors. Next, the anchors are identified in descending order of one dimension. The first scale allocates the three most prominent anchors, the second assigns the

following three anchors, and the third one the last three. YOLOv3 has more bounding boxes predicted than YOLOv2. For the same 416 × 416 image, YOLOv2 has 13 × 13 × 5 = 845 boxes; at every grid cell, a total of 5 boxes were detected with the use of 5 anchors, as opposed to YOLO v3, which predicted boxes at three distinct scales, totaling 10,647 predicted boxes for an image with the size of 416 × 416. In other words, it predicts ten times more boxes than the total predicted by YOLO v2. For each scale, every grid can predict three boxes using three anchors. Since there are three scales, nine anchor boxes are used. YOLOv3's loss function of YOLOv3 is defined from three aspects: the bounding box position error, the bounding box confidence error, and the classification prediction error between the ground truth and the predicted boxes. YOLOv3 predicts an objectness score for each bounding box using logistic regression. The first aspect of the loss function is the bounding box position error. The error is calculated by summing up the squared differences between predicted and true values of a bounding box's x, y, w, and h coordinates multiplied by a lambda coefficient that controls its importance to other losses. The second aspect is the bounding box confidence error which measures how confident YOLOv3 is that there is an object in a given bounding box. This term uses binary cross-entropy loss to calculate how well it predicts whether or not there is an object in a given cell. Finally, classification prediction error measures how well YOLOv3 predicts an object's class. It uses binary cross-entropy loss for each label.

#### 3) SSD

Single Shot MultiBox Detector (SSD) [73] is an object detection framework published after R-CNN and YOLO. It was developed by W. Liu et al. to predict bounding boxes and class probability in a one-time process using an end-to-end CNN architecture. It is typically faster than the faster R-CNN. The SSD allows a one-time shot to detect several objects in the image instead of the two shots required for the region proposal network methods listed in the previous section. As a consequence, SSD is considerably more time-saving compared to region-based approaches. An image is introduced as an input through a VGG-16 [58] network to extract feature maps. Several convolutional layers are added with different filter sizes (19 × 19, 10 × 10, 5 × 5, 3 × 3, and 1 × 1). These and the 38 × 38 feature map produced by conv4_3 of VGG are the feature maps that 3 × 3 convolution filters will process for each cell to make predictions. There are k-bounding boxes for each location in the feature maps. These k-boxes are of various sizes and aspect ratios. On each bounding box, we calculate the C class scores and four offsets about the original shape of the default bounding box. Each box has four parameters and a probability vector corresponding to the confidence given to each object class. SSD involves negative sampling to determine poor predictions. It applies the non-maximal suppression technique at the end of the model, like YOLO, to maintain the more appropriate boxes. Afterward, the Hard-Negative Mining (HNM) method is applied to ensure faster and more stable training. They select the negative examples according to the highest confidence value assigned to each default box and then select the high ones to ensure that the negative and positive ratio is below 3:1.

The SSD loss function combines localization and confidence loss. The localization loss is the mismatch between the ground truth box and the predicted boundary box. SSD only penalizes predictions from positive matches. Negative matches can be ignored. The confidence loss is a softmax loss over multiple confidence classes (c). During training, the set of default boxes and scales for detection is essential. The SSD uses smooth L1 loss as its regression loss function. It is a particular case of Huber Loss with $\delta = 1$. Smooth L1 loss combines L1 Loss and L2 Loss. When |a| is less than or equal to 1, it behaves like an L2 loss. One-hot encoding turns the label y into a probability distribution.

#### 4) RETINANET

RetinaNet [79] is a single-stage object detector such as SSD and YOLO that offers almost the same performance as two-stage detectors such as Faster R-CNN. This paper's significant contribution is a new loss function called a focal loss for classification, which has significantly increased accuracy. RetinaNet is a single, composite network consisting of the leading backbone network called Feature Pyramid Net, which relies on ResNet (ResNet50 or ResNet101) and two task-specific sub-networks. The backbone network calculates the convolutional feature map for the entire input image. The first subnetwork is used to classify the output of the backbone, while the second subnetwork network is used to perform bounding box regression using the backbone's output. Because of its fully convolutional structure, RetinaNet allows the network to take an image of random size and generates feature maps with proportional sizes at several levels in the feature pyramid. In the classification sub-network, a fully convolutional network is associated with each level of FPN. For each anchor A and K object class, it predicts how probably there will be objects in each spatial position. There are four 3 × 3 convolution layers with 256 filters in addition to ReLU activation [117]. A further 3 × 3 convolutional layers are applied with a K × A filter, followed by sigmoid activation at the outputs. Focal loss is applied as a loss function. For the subnetwork, parameters are shared at all levels. As a result, the shape of the output feature map has the following dimensions (W, H, KA), which correspond to the feature map width and height, and K. A denotes the object's class and anchor box values. The regression subnetwork is associated with each FPN feature map parallel to the classification subnetwork. The regression subnetwork is designed the same way as the classification subnetwork; the only difference is that the parameters are not shared, with the last convolution layer consisting of 3 × 3 and 4 filters. Therefore, the output feature map would be in the shape of (W, H, 4A).

RetinaNet utilizes a focal loss function to address class imbalance during training. RetinaNet's focal loss function

**TABLE 7. Advantages and limitations of transformer-based object detectors.**

| Year | Method | Advantages | Limitations |
|---|---|---|---|
| 2020 | DETR | -End-to-end training<br>-Simple architecture<br>-It does not necessitate a dedicated library.<br>-Can be used in panoptic segmentation.<br>-It achieves better results on large objects compared to Faster R-CNN due to the self-attention mechanism | -Slow convergence<br>-Cannot meet real-time detection requirements. |
| 2021 | SMCA | -Improve the slow convergence of DETR by introducing the spatially modulated co-attention mechanism.<br>-More accurate than DETR | -Cannot meet real-time detection requirements. |
| 2021 | Swin | -Great accuracy<br>-Good speed/accuracy trade-off<br>-Can be used for image classification and semantic segmentation. | -Cannot meet real-time detection requirements. |
| 2022 | Anchor DETR | -Better accuracy than DETR.<br>-Less training time than DETR<br>-Faster than other transformer-based detectors. | - Still cannot meet real-time detection requirements |
| 2022 | DESTR | -Outperforms transformer-based detectors that use single-scale features | -Cannot meet real-time detection requirements. |

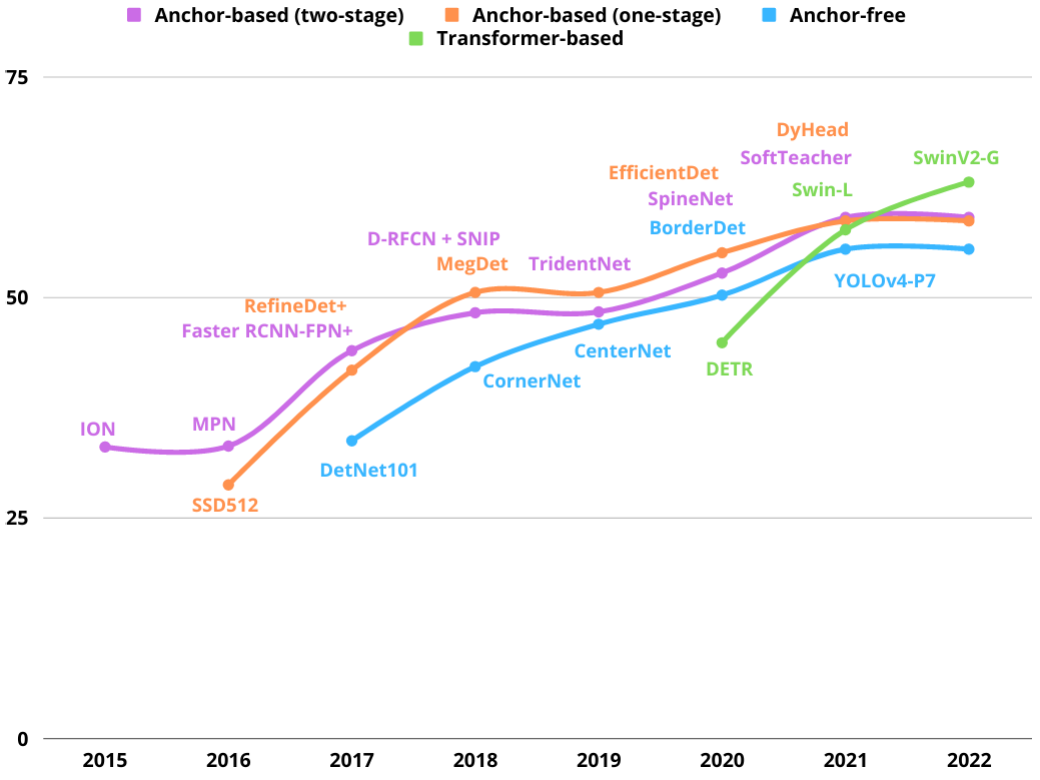


**FIGURE 6. Accuracy evolution of the main object detector families in MS-COCO.**

down-weights the loss assigned to well-classified examples, focusing the training on a sparse set of hard examples and preventing many easy negatives from overwhelming the detector during training.

### 5) MEGDET

MegDet [118] is a model that tackles the object detection task from the batch size factor. The authors propose a Large Mini-Batch size of 256 instead of 16 during training. They use a Cross-GPU batch normalization with 128 GPUs and a warmup learning rate policy to train the whole network in a suitable time. MegDet achieved an mAP of 50.6% on the MS-COCO dataset using the ResNet-50 as a backbone and the OHEM technique. They finished the model training in four hours. The MegDet paper does not describe a specific loss function by name. However, it mentions that the shape of the regression loss function (parameters of SmoothL1 Loss) is used in MegDet.

#### 6) EFFICIENTDET

EfficientDet [67] is an object detection model that relies on the pretrained EfficientNet [60] backbones, a weighted bidirectional feature network, and a personalized compound scaling technique. The bidirectional feature network takes the level features 3 to 7 from the efficient net and applies top-down and bottom-up bidirectional feature fusion. The class and box network weights are shared between all levels of features. EfficientDet7 achieved an AP of 52.2% on the MS-COCO dataset using the EfficientNet-B7 as a backbone. EfficientDet uses a focal loss function for dense object detection. However, the EfficientDet paper mentions that the detection head and loss function are replaced with a segmentation head and loss function to perform segmentation tasks.

#### 7) PAA

PAA [119], a model based on a new technique to assign anchors based on the likelihood optimization of the probability distribution, stands for probabilistic anchor assignment. It consists of computing scores of anchors and identifying positive and negative samples in a probabilistic way compared to the heuristic IoU challenging assignment, which makes the training process more difficult and time-consuming. The authors propose a score voting method for post-processing in dense object detection. PAA achieved an AP of 50.8% on the MS-COCO dataset and 53.5% on the same dataset for multi-scale testing. The authors tested the model with many backbones and obtained the best results using ResNeXt-32 × 8d-152-DCN.

#### 8) YOLOv5

YOLOv5[1] is a model in the You Only Look Once (YOLO) family. It is used for detecting objects and comes in four main versions: small (s), medium (m), large (l), and extra large (x), each offering progressively higher accuracy rates. YOLOv5 focuses on inference speed and accuracy, using compound-scaled object detection models trained on the COCO dataset for model ensembling and Test Time Augmentation. The algorithm only looks at an image once and detects all objects present and their location.YOLOv5 was introduced in 2020 by the same team that developed the original YOLO algorithm as an open-source project. It builds upon the success of previous versions and adds several new features and improvements.YOLOv5 uses a Convolutional Neural Network (CNN) backbone called CSPDarknet to form image features. These features are combined in the model neck, which uses a PANet (Path Aggregation Network) variant, and sent to the head. The model head then interprets the combined features to predict the class of an image. It also uses residual and dense blocks to enable information flow to the deepest layers. The architecture consists of three parts: backbone, neck, and head.

[1]https://github.com/ultralytics/yolov5

#### 9) YOLOv7

YOLOv7 [233] is a faster and more accurate real-time for computer vision tasks. Like Scaled YOLOv4 [199], YOLOv7 backbones do not use ImageNet pre-trained backbones. YOLOv7 weights are trained using Microsoft's COCO dataset, and no datasets or pre-trained weights are used. The official paper demonstrates how this improved architecture surpasses all previous YOLO versions and all other object detection models in terms of speed and accuracy. YOLOv7 improves speed and accuracy by introducing several architectural reforms. The larger models in the YOLO7 family are YOLOv7-X, YOLOv7-E6, YOLOv7-D6, and YOLOv7-E6E. Other variations include YOLOv7-X, YOLOv7-E6, and YOLOv7-D6, which were obtained by applying the proposed compound scaling method to scale up the depth and width of the entire model.

### D. COMPARISON: ONE-STAGE DETECTORS

Table 5 lists a chronological comparison of the strengths and limitations of the one-stage anchor-based detection methods mentioned earlier in this paper.

## VII. ANCHOR-FREE DETECTORS

### A. YOLOv1

YOLO [72] has a different approach to object detection. It captures the complete image in a single instance. It then predicts both the coordinates of the bounding boxes for regression and the class probabilities with only one network in one evaluation. Thus, his name is YOLO; you only look once. The power of the YOLO model ensures real-time predictions. The input image is split into an SxS grid of cells to perform detection. A single grid cell is supposed to predict every single object in the image, and this is where the object's center falls. Each cell will predict B potential bounding boxes with each bounding box's C class probabilities value, with a total of SxSxB boxes. Since the probability of most of these boxes is relatively small, the algorithm excludes those boxes that fall below a minimum specific probability threshold. A non-maximal suppression procedure is applied to all left boxes, removing all possible multiple detections and keeping the most accurate objects. A CNN based on the GoogLeNet [120] model, which includes the initial modules, has been applied. The network architecture includes 24 convolutional layers and two fully connected layers. The reduction layers of $1 \times 1$ filters, followed by convolutional $3 \times 3$ layers, replace the primary inception modules. As a result of the final layer, a tensor of S * S * (C + B * 5) is obtained that equals the predictions of each grid cell. The total estimate of probabilities for each class is called C. The number of anchor boxes in each cell is indicated by B, with an additional four coordinates and a confidence value for each cell. YOLO has three loss functions, one for the abjectness score and two others for the coordinates and classification errors. The latter is calculated when the abjectness score is greater than 0.5. YOLOv1 loss function is divided into three parts: the one

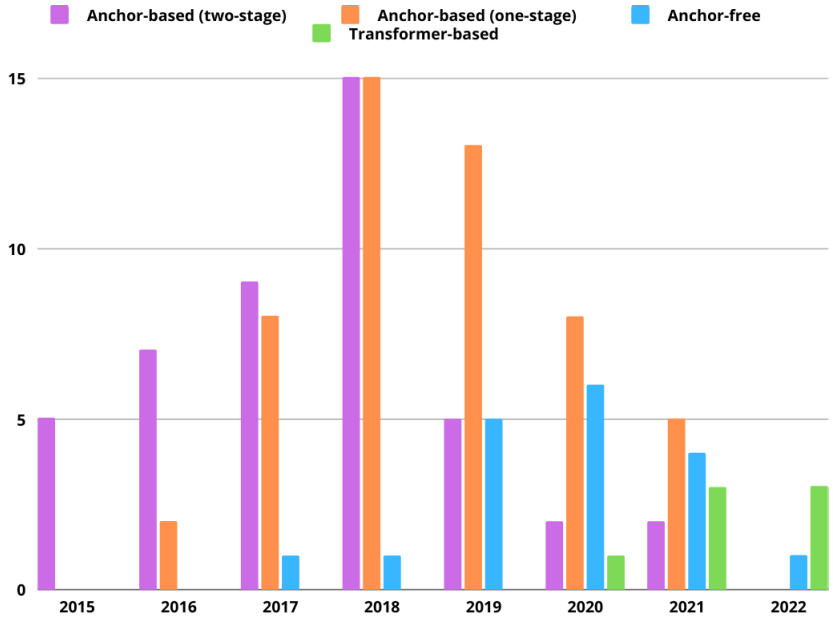


**FIGURE 7.** The number of state-of-the-art object detectors, by category, published in top journals and evaluated on MS-COCO.

responsible for finding the bounding-box coordinates, the bounding-box score prediction, and the class prediction. The final loss function is a sum of these three parts.

### B. CORNERNET

CornerNet [74] is an object detection model that uses key points to detect the object bounding box. It uses a convolutional neural network to detect objects as paired keypoints from the top-left and bottom-right corners. Those corners are represented as heatmaps, one for the top-left corners and the other for the bottom-right corner. Each corner has only one ground truth positive location, while all the remaining locations are identified as negative. This technique prevents the model from using traditional anchors employed in other object detectors. The authors also propose a new type of pooling layer named corner pooling that aims to localize corners efficiently. CornerNet uses the Hourglass-104 backbone and achieved an accuracy of 40.5% in the MS-COCO dataset and 42.1% using multi-scale training in the same dataset. CornerNet uses associative embedding, where the network predicts similar embeddings for corners belonging to the same object and a loss function similar to the triplet loss. In addition, it proposes a new variant of focal Loss as a loss function, which dynamically adjusts the weights of each anchor box.

### C. EXTREMENET

ExtremeNet [121] uses a bottom-up approach to detect objects. They use a standard keypoint estimation network to identify the object's center point and its four extreme points: top, right-most, left, and bottom-most. These four extreme vital points are used as the object bounding box in a purely geometric way. The model uses the Hourglass-104 as a backbone and obtained an accuracy of 43.7% and 40.2% on the MS-COCO dataset for the single-scale and multi-scale testing, respectively. The ExtremeNet paper does not describe a specific loss function by name.

### D. REPPOINTS

RepPoints [78] stands for representative points, a technique representing objects as a set of sample points. Since the traditional bounding boxes provide a coarse localization and extraction, RepPoints use points to localize and identify objects. The reppoint technique does not use anchors to sample the space of bounding boxes. Instead, it learns to automatically process the ground truth localization and recognition targets by limiting the spatial extent within an object and identifying the semantically relevant local areas. The authors proposed object detection model is RPDet [78] based on the RepPoints representation combined with deformable convolution. RPDet used ResNet-101-DCN as a backbone and obtained an accuracy of 42.8% and 46.5% in multi-scale training and testing. RepPoints paper describes two sets of RepPoints, one driven by the points distance loss alone and the other by a combination of the points distance loss and the center-ness loss.

### E. FSAF

The authors propose a Feature Selective Anchor-Free (FSAF) module [122] to solve two problems faced in anchor-based single-shot detectors with feature pyramids; the heuristic-guided feature selection and the overlap-based anchor sampling. While training multi-level anchor-free branches, the FSAF module applies online feature selection while training

the multi-level anchor-free branches, improving baselines with tiny inference overhead. Each instance is linked to the appropriate feature level to optimize the network. The model encodes those instances following an anchor-free approach to learn the parameters for classification and regression. The authors experiment with applying the FSAF module with other anchor-based branches, such as the RetinaNet baseline, and yield excellent results with free inference overhead. The proposed model achieved a currency of 44.6% on the MS-COCO dataset. The FSAF paper does not describe a specific loss function by name. However, the FSAF module uses focal loss for non-ignoring regions and a 4-channel feature map for the bounding box regression subnet.

### F. FCOS

In addition to being an anchor-free detector, the Fully Convolutional One-Stage Detection (FCOS) [75] is also a proposal-free detector. Similar to semantic segmentation, FCOS relies on the per-pixel technique to detect objects, avoiding all the hyperparameters and the complexity of overlapping in training. FCOS uses Non-maximum suppression (NMS) for post-processing and filtering the bounding boxes, which improves accuracy. FCOS achieves an accuracy of 44.7% in MS-COCO using the ResNeXt-64 × 4d-101-FPN as a backbone. The authors use the FCOS as a region proposal network in two-stage object detectors, such as Faster R-CNN. The loss function used in FCOS combines three losses: focal loss for classification, IoU loss for regression, and center-ness loss. The focal loss addresses the class imbalance problem by down-weighting the easy examples and up-weighting the hard ones. The IoU loss measures the overlap between predicted bounding boxes and ground-truth boxes. The center-ness loss encourages the network to predict more accurate bounding boxes by penalizing predictions far from the objects' center.

### G. ATSS

Adaptive Training Sample Selection (ATSS) [123] is a method developed to deal with the gap between center-based anchor-free and one-stage anchor-based detectors, depending on how they define the positive and negative training samples. The proposed model can automatically define the positive and negative training samples based on the object's statistical characteristics. The positive and negative samples are used for classification, while the negative ones are for regression. The Adaptive Training Sample Selection technique, had no hyperparameters compared to previous techniques. The authors also mentioned that tiling multiple anchors per location is crucial during object detection. ATSS used the ResNets as a backbone and obtained the highest accuracy with ResNeXt-64 × 4d-101-DCN with 47.7% and 50.7% on multi-scale. The Adaptive Training Sample Selection (ATSS) method automatically selects positive and negative samples based on object characteristics using statistical characteristics to calculate dynamic thresholds. However, the ATSS paper does not describe a specific loss function by name.

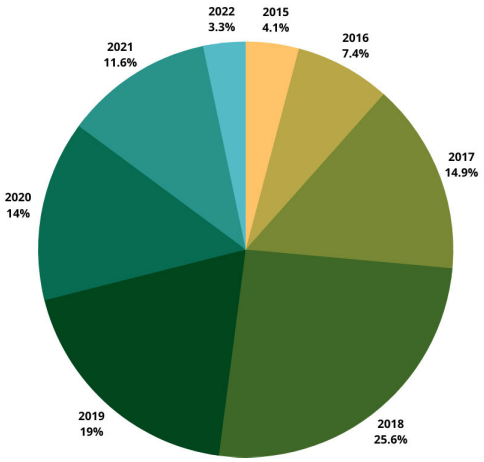


**FIGURE 8.** The percentage of object detection models published each year.

### H. OTA

The authors propose an Optimal Transport Assignment [124] technique based on the optimization theory. The model deals with the label-assigning stage in object detection as an Optimal Transport problem by defining the transportation between each anchor and ground truth pair. The technique uses a cost-effective transport of labels from ground-truth objects and backgrounds to anchors using the Sinkhorn-Knopp Iteration [125]. Based on the Intersection-over-Union values between the predicted bounding boxes and each ground truth, they provide a new simple estimation strategy to identify the positive labels each ground truth needs. Compared to previous one-stage detectors, OTA can cope with the assignment of ambiguous anchors by assigning them manually using hand-crafted rules before applying the optimal transport assignment. OTA achieves excellent results on the MS-COCO dataset with an accuracy of 49% and 51.5% on multi-scale testing. The authors tested their method with several backbones and obtained the best results using the ResNeXt-64 × 4d-101-DCN. The Optimal Transport Assignment (OTA) loss function is a label-assigning procedure in object detection that aims to transport labels from ground-truth objects and assign them to anchor boxes.

### I. DSLA

DSLA [126] stands for Dynamic Smooth Label Assignment and is a recent anchor-free detector published to solve the inconsistency problems in previous detectors. They improve the transition between positive and negative samples by improving the centeredness representation suggested in FCOS and providing an interval relaxation strategy. The Intersection-of Union is coupled with the smooth label with a value between 0 and 1 to supervise the classification branch, which is merged with the quality estimation branch, resulting in a more simplified anchor-free model with good localization quality. The IoU is predicted dynamically during training. The authors tested the DSLA model with several

backbones, such as Resnet-50, Resnet-50-DCN, ResNeXt-101-64 × 4d-DCN, and the Swin-S. With the Swin-S backbone, they achieved remarkable results on the MS-COO dataset, reaching 49.2%. DSLA improves the performance of detection models with adaptive label assignment algorithms and lower bounding box losses for those positive samples indicating more samples with higher-quality predicted boxes are selected as positives.

### J. YOLOv8

YOLOv8[2] is a state-of-the-art object detection, image classification, and instance segmentation model developed by Ultralytics. It is designed to be fast, accurate, and easy to use. YOLOv8 builds upon the success of previous YOLO versions and introduces new features and improvements to boost performance and flexibility further. It can be trained on large datasets and run on various hardware platforms, from CPUs to GPUs. One key feature of YOLOv8 is its extensibility. It supports all previous versions of YOLO, making it easy to switch between different versions and compare their performance. This makes YOLOv8 an ideal choice for users who want to take advantage of the latest YOLO technology while still being able to use their existing YOLO models.YOLOv8 includes numerous architectural and developer-convenience features, making it an appealing choice for a wide range of object detection and image segmentation tasks. The architecture of YOLOv8 changed from a simple version to a more complex one, with new convolutional layers and a new detection head. Compared to YOLOv5, it replaces the C3 module with the C2f module.

### K. COMPARISON: ANCHOR-FREE DETECTORS

Table 6 lists a chronological comparison of the strengths and limitations of the anchor-free object detection methods mentioned earlier in this paper.

## VIII. TRANSFORMER-BASED DETECTORS

### A. VIT

ViT, published in [127] and inspired by transformers in NLP tasks [128], [129], was the first object detection model to apply transformers directly to images instead of combining convolutional neural networks and transformers. ViT splits the image into patches by providing the sequence of linear embeddings of these patches as an input to a Transformer. The model processes the patches as a sequence of words like tokens processed in Natural Language Processing. A constant latent vector is used to flatten and map the patches to the vector size dimension with a trainable projection in all the transformer layers. They used an MLP with one hidden layer during the classification during the pre-training time and one single layer at the fine-tuning time. The ViT achieved the highest performance when trained on larger datasets when they were first published. The Vision Transformer (ViT) paper does not describe a specific loss function by name. However, the ViT model outputs hidden raw states without any specific head on top. It can be used as a building block for various computer vision tasks such as image classification.

[2]https://docs.ultralytics.com/

### B. DETR

The DEtection TRansformer (DETR) presented in [130] is the first end-to-end object detection model based on transformers. It consists of a pretrained CNN backbone and transformer. The model uses Resnets as a backbone to generate the lower dimensional features, which will then be formatted into a single set of features and added to a positional encoding, fed into a Transformer. The transformer creates an end-to-end trainable detector. The transformer is based on the original transformer [131]. It consists of an Encoder and a Decoder, removing hand-crafted modules like anchor generation. The transformer encoder takes image features and position encodings as input and directs the result to the decoder. The decoder processes those features and transmits the output into a fixed number of Prediction Heads, which consist of a predefined number of feed-forward networks. Each prediction head's output has a class and bounding box. Multi-head attentions in the decoder modify these object queries with encoder embeddings to generate results passed through multi-layer perceptrons to predict class and bounding boxes. DeTR uses bipartite matching loss to find the optimal one-to-one matching between detector output and padded ground truth. DETR generates a predefined number of predictions, each computed in parallel. DETR proposes a set-based global loss that forces unique predictions via bipartite matching. The DETR model approaches object detection as a direct set prediction problem and consists of a set-based global loss, which is the sum of the classification loss and the bounding box regression loss.

### C. SMCA

The SMCA model [132], published in 2021, was an alternative to improve the DETR model convergence. To train DETR from scratch, it needs about 500 epochs to achieve the best results. SMCA proposes a mechanism called Spatially Modulated Co-Attention to improve the convergence of DETR. The SMCA model only replaces the co-attention mechanism in the DETR decoder by applying location-aware co-attention. This new feature constraints co-attention responses to be high near initially estimated bounding box locations. Training SMCA takes only 108 epochs and achieves better results than the original DETR, and demonstrates potential processing of global information.

### D. SWIN

The Swin Transformer [133] seeks to provide a transformer-based backbone for computer vision tasks. The word Swin stood for Shifted window and was the first time to apply the shifted window concept used in CNN in transformers. It uses patches as in the ViT model by splitting the input images into multiple, non-overlapping patches and converting them into embeddings. Numerous Swin Transformer blocks are

**TABLE 8.** Comparative results on VOC 2007 test set (%).

| method | backbone | data | mAP | Year |
|---|---|---|---|---|
| ***two-stage anchor-based*** | | | | |
| R-CNN[85] | AlexNet | 07 | 54.2 | 2013 |
| R-CNN bb[85] | AlexNet | 07 | 58.5 | 2013 |
| R-CNN bb[68] | ZF5 | 07 | 59.2 | 2014 |
| SPP-net[68] | ZF5 | 07 | 60.9 | 2014 |
| R-CNN [85] | VGGNet-16 | 07 | 62.2 | 2013 |
| HyperNet[92] | AlexNet | 07+12 | 65.9 | 2016 |
| R-CNN bb[136] | VGGNet-16 | 07 | 66.0 | 2016 |
| G-CNN[94] | VGGNet-16 | 07 | 66.8 | 2016 |
| Fast R-CNN[76] | VGGNet-16 | 07 | 66.9 | 2015 |
| NoC[137] | VGGNet-16 | 07+12 | 68.8 | 2015 |
| Faster R-CNN[76] | VGGNet-16 | 07 | 69.9 | 2017 |
| Fast R-CNN[84] | VGGNet-16 | 07+12 | 70.0 | 2015 |
| NoC bb[137] | VGGNet-16 | 07+12 | 71.6 | 2015 |
| Faster R-CNN [76] | VGGNet-16 | 07+12 | 73.2 | 2017 |
| OHEM [138] | VGGNet-16 | 07+12 | 74.6 | 2016 |
| HyperNet [92] | VGGNet-16 | 07+12 | 74.8 | 2016 |
| SIN[139] | VGGNet-16 | 07+12 | 76.0 | 2018 |
| HyperNet [92] | VGGNet-16 | 07+12 | 76.3 | 2016 |
| Faster R-CNN [59] | ResNet-101 | 07+12 | 76.4 | 2016 |
| ION [140] | VGGNet-16 | 07+12 | 76.5 | 2015 |
| R-FCN [100] | ResNet-101 | 07 | 76.6 | 2016 |
| LocNet [141] | VGGNet-16 | 07+12 | 77.5 | 2016 |
| MR-CNN [142] | VGGNet-16 | 07+12 | 78.2 | 2015 |
| Faster R-CNN [76] | VGGNet-16 | 07+12+COCO | 78.8 | 2017 |
| ION [140] | VGGNet-16 | 07+12+S | 79.2 | 2015 |
| R-FCN [100] | ResNet-101 | 07+12 | 79.5 | 2016 |
| R-FCN multi-sc-train [100] | ResNet-101 | 07+12 | 80.5 | 2016 |
| CoupleNet [154] | ResNet-101 | 07+12 | 82.7 | 2017 |
| ACoupleNet [156] | ResNet-101 | 07+12+S | 83.1 | 2019 |
| R-FCN [100] | ResNet-101 | 07+12+COCO | 83.6 | 2016 |
| DCN+R-CNN [157] | ResNet-101+ResNet-152 | 07+12 | 84.0 | 2018 |
| Faster R-CNN+++ [59] | ResNet-101 | 07+12+COCO | 85.6 | 2016 |
| ACoupleNet [156] | ResNet-101 | 07+12+COCO* | 85.7 | 2019 |
| SNIPER [158] | ResNet101 | 07+12 | 86.9 | 2018 |
| Copy-Paste [101] | EfficientNet-B7 NAS-FPN | 07+12 | 88.6 | 2021 |
| Copy-Paste [101] | EfficientNet-B7 NAS-FPN | 07+12+COCO | **89.3** | 2021 |
| ***one-stage anchor-free*** | | | | |
| YOLOv1[72] | GoogLeNet[120] | 07+12 | 63.4 | 2016 |
| YOLOv1[72] | VGGNet-16 | 07+12 | **66.4** | 2016 |
| ***one-stage anchor-based*** | | | | |
| SSD300[73] | VGGNet-16 | 07 | 68.0 | 2016 |
| SSD512[73] | VGGNet-16 | 07 | 71.6 | 2016 |
| YOLOv2 352 [80] | Darknet-19 | 07+12 | 73.7 | 2016 |
| SSD300[73] | VGGNet-16 | 07+12 | 74.3 | 2016 |
| RON384 [106] | VGGNet-16 | 07+12 | 75.4 | 2017 |
| Shrivastava et al. [138] | VGGNet-16 | 07+12 | 76.4 | 2016 |
| YOLOv2 416[80] | Darknet-19 | 07+12 | 76.8 | 2015 |
| SSD512 [73] | VGGNet-16 | 07+12 | 76.8 | 2016 |
| SSD321 [105] | ResNet-101 | 07+12 | 77.1 | 2017 |
| SSD300 [73] | VGGNet-16 | 07+12 | 77.2 | 2016 |
| RON384++ [106] | VGGNet-16 | 07+12 | 77.6 | 2017 |
| DSOD300 [103] | DS/64-192-48-1[103] | 07+12 | 77.7 | 2017 |
| YOLOv2 480 [80] | Darknet-19 | 07+12 | 77.8 | 2016 |
| STDN300 [107] | DenseNet-169 | 07+12 | 78.1 | 2018 |
| Gidaris et al [142] | VGGNet-16 | 07+12 | 78.2 | 2015 |
| R-SSD300 [143] | VGGNet-16 | 07+12 | 78.5 | 2017 |
| YOLOv2 544 [80] | Darknet-19 | 07+12 | 78.6 | 2016 |
| DSSD321 [105] | ResNet-101 | 07+12 | 78.6 | 2017 |
| StairNet [144] | VGGNet-16 | 07+12 | 78.8 | 2017 |
| FSSD300 [145] | VGGNet-16 | 07+12 | 78.8 | 2017 |
| BlitzNet300 [146] | ResNet-50 | 07+12+S | 79.1 | 2017 |
| RUN300[147] | VGGNet-16 | 07+12 | 79.2 | 2018 |
| ESSD300 [148] | VGGNet-16 | 07+12 | 79.4 | 2018 |
| SSD300 [73] | VGGNet-16 | 07+12+COCO | 79.6 | 2016 |
| DFPR300 [114] | VGGNet-16 | 07+12 | 79.6 | 2018 |
| WeaveNet [149] | VGGNet-16 | 07+12 | 79.7 | 2017 |
| SSD512 [73] | VGGNet-16 | 07+12 | 79.8 | 2016 |
| RefineDet320 [102] | VGGNet-16 | 07+12 | 80.0 | 2017 |
| BPN320 [150] | VGGNet-16 | 07+12 | 80.3 | 2018 |

**TABLE 8.** *(Continued.)* **Comparative results on VOC 2007 test set (%).**

| method | backbone | data | mAP | Year |
|---|---|---|---|---|
| ***one-stage anchor-based [continued]*** | | | | |
| EFIPNet300 [151] | VGGNet-16 | 07+12 | 80.4 | 2019 |
| ScratchDet300 [104] | Root-ResNet-34 | 07+12 | 80.4 | 2019 |
| RFBNet300 [69] | VGGNet-16 | 07+12 | 80.5 | 2018 |
| ESCNet300 | VGGNet-16 | 07+12 | 80.5 | |
| SSD513 [105] | ResNet-101 | 07+12 | 80.6 | 2017 |
| PFPNet-R320 [113] | VGGNet-16 | 07+12 | 80.7 | 2018 |
| R-SSD512 [143] | VGGNet-16 | 07+12 | 80.8 | 2017 |
| FSSD512 [145] | VGGNet-16 | 07+12 | 80.9 | 2017 |
| STDN513 [107] | DenseNet-169 | 07+12 | 80.9 | 2018 |
| RUN512 [147] | VGGNet-16 | 07+12 | 80.9 | 2018 |
| SSD300 [73] | VGGNet-16 | 07+12+COCO | 81.2 | 2016 |
| EFGRNet320 [110] | VGGNet-16 | 07+12 | 81.4 | 2019 |
| DSSD513 [105] | ResNet-101 | 07+12 | 81.5 | 2017 |
| BlitzNet512 [146] | ResNet-50 | 07+12+S | 81.5 | 2017 |
| MSA-DNN300 [152] | VGGNet-16 | 07+12 | 81.5 | 2018 |
| SSD512 [73] | VGGNet-16 | 07+12+COCO | 81.6 | 2016 |
| DSOD300 [103] | DS/64-192-48-1 | 07+12+COCO | 81.7 | 2017 |
| DES512 [108] | VGGNet-16 | 07+12 | 81.7 | 2018 |
| HSD512 [153] | VGGNet-16 | 07+12 | 81.7 | 2019 |
| RefineDet512 [102] | VGGNet-16 | 07+12 | 81.8 | 2017 |
| EFIPNet512 [151] | VGGNet-16 | 07+12 | 81.8 | 2019 |
| BPN512 [150] | VGGNet-16 | 07+12 | 81.9 | 2018 |
| RFBNet512 [69] | VGGNet-16 | 07+12 | 82.2 | 2018 |
| PFPNet-R512 [113] | VGGNet-16 | 07+12 | 82.3 | 2018 |
| DFPR512 [114] | ResNet-101 | 07+12 | 82.4 | 2018 |
| FSSD300 [145] | VGGNet-16 | 07+12+COCO | 82.7 | 2016 |
| EFGRNet512 [110] | VGGNet-16 | 07+12 | 82.7 | 2019 |
| MSA-DNN512 [152] | VGGNet-16 | 07+12 | 82.9 | 2018 |
| HSD512 [153] | VGGNet-16 | 07+12 | 83.0 | 2019 |
| TripleNet512 [155] | ResNet-101 | 07+12 | 83.0 | 2019 |
| RefineDet320+ [102] | VGGNet-16 | 07+12 | 83.1 | 2017 |
| SSD512 [73] | VGGNet-16 | 07+12+COCO | 83.2 | 2016 |
| PFPNet-R320+ [113] | VGGNet-16 | 07+12 | 83.5 | 2018 |
| RefineDet512+ [102] | VGGNet-16 | 07+12 | 83.8 | 2017 |
| RetinaNet500+AP-Loss [112] | ResNet-101 | 07+12 | 83.9 | 2019 |
| RefineDet320 [102] | VGGNet-16 | 07+12+COCO | 84.0 | 2017 |
| ScratchDet300 [104] | Root-ResNet-34 | 07+12+COCO | 84.0 | 2019 |
| PFPNet-R512+ [113] | VGGNet-16 | 07+12 | 84.1 | 2018 |
| ScratchDet300+ [104] | Root-ResNet-34 | 07+12 | 84.1 | 2019 |
| FSSD512* [145] | VGGNet-16 | 07+12+COCO | 84.5 | 2017 |
| RetinaNet500+AP-Loss (MS) [112] | ResNet-101 | 07+12 | 84.9 | 2019 |
| DAFS320 [111] | VGGNet-16 | 07+12+COCO | 84.7 | 2019 |
| DAFS320+ [111] | VGGNet-16 | 07+12+COCO | 86.1 | 2019 |
| ScratchDet300+ [104] | Root-ResNet-34 | 07+12+COCO | **86.3** | 2019 |

then applied to the patches in 4 stages. Each successive stage reduces the number of patches to maintain hierarchical representation, compared to ViT, which uses patches of one size. These patches are converted linearly into C-dimensional vectors. It computes self-attention only within the local window as the transformer block comprises local multi-headed self-attention modules based on alternating shifted patch windows in successive blocks. Computation complexity becomes linear with image size in local self-attention, while a shifted window enables cross-window connection and reduces complexity. Each time the attention window shifts concerning the previous layer. Swin utilizes comparatively higher parameters than convolutional models.

### E. ANCHOR DETR

In [134], the authors propose an end-to-end object detection model based on transformers with a novel query design. Their novel query design is based on the anchor points to solve the absence of explicit physical meaning of learned object queries, which makes the optimization process difficult. The anchor points were used before in CNN-based detectors, and applying this mechanism lets the object query focus on the objects near the anchor points. The Anchor DETR model can predict multiple objects at one position. To optimize the complexity, they use an attention variant, Row-Column Decoupled Attention, that reduces the memory cost without sacrificing accuracy. The primary model uses ResNet-101 as the backbone with a DC5 feature and achieves an accuracy of 45.1% on MS-COCO with considerably fewer training epochs than DETR. The authors proposed anchor-free, RAM-free, and NMS-free variants.

### F. DESTR

DESTR [135], published recently, proposed solving some previous transformer problems, such as the Cross and self-attention mechanisms and the transformer's decoder content

**TABLE 9.** Comparative results on VOC 2012 test set (%).

| method | backbone | data | mAP | Year |
|---|---|---|---|---|
| ***two-stage anchor-based*** | | | | |
| R-CNN [85] | AlexNet | 12 | 49.6 | 2013 |
| R-CNN bb [85] | AlexNet | 12 | 53.3 | 2013 |
| R-CNN [85] | VGGNet-16 | 12 | 59.2 | 2013 |
| R-CNN bb [136] | VGGNet-16 | 12 | 62.4 | 2016 |
| Fast R-CNN [84] | VGGNet-16 | 12 | 65.7 | 2015 |
| G-RCNN | VGGNet-16 | 07++12 | 66.4 | 2016 |
| Faster R-CNN [76] | VGGNet-16 | 12 | 67.0 | 2017 |
| NoC [137] | VGGNet-16 | 07++12 | 67.6 | 2015 |
| Fast R-CNN [84] | VGGNet-16 | 07++12 | 68.4 | 2015 |
| NoC bb [137] | VGGNet-16 | 07++12 | 68.8 | 2015 |
| Faster R-CNN [76] | VGGNet-16 | 07++12 | 70.4 | 2017 |
| Fast R-CNN+YOLO [72] | VGGNet-16 | 07++12 | 70.7 | 2016 |
| MR_CNN_S_CNN [142] | VGGNet-16 | 12 | 70.7 | 2015 |
| HyperNet_SP [92] | VGGNet-16 | 07++12 | 71.3 | 2016 |
| HyperNet [92] | VGGNet-16 | 07++12 | 71.4 | 2016 |
| OHEM [138] | VGGNet-16 | 07++12 | 71.9 | 2016 |
| Shrivastava et al. [138] | VGGNet-16 | 07++12 | 72.6 | 2016 |
| SIN [139] | VGGNet-16 | 07++12 | 73.1 | 2018 |
| Faster R-CNN [59] | ResNet-101 | 07++12 | 73.8 | 2016 |
| MR CNN MORE DATA[142] | VGGNet-16 | 07++12 | 73.9 | 2015 |
| Gidaris et al [142] | VGGNet-16 | 07++12 | 73.9 | 2015 |
| Faster R-CNN [76] | VGGNet-16 | 07+12+COCO | 75.9 | 2017 |
| R-FCN multi-sc-train [100] | ResNet-101 | 07++12 | 77.6 | 2016 |
| OHEM++ [138] | VGGNet-16 | 07+12+COCO | 80.1 | 2016 |
| CoupleNet [154] | ResNet-101 | 07++12 | 80.4 | 2017 |
| ACoupleNet [156] | ResNet-101 | 07+12+S | 81.0 | 2019 |
| DCN+R-CNN [157] | ResNet-101+ResNet-152 | 07++12 | 81.2 | 2018 |
| R-FCN multi-sc-train [100] | ResNet-101 | 07++12+COCO | 82.0 | 2017 |
| Faster R-CNN+++ [59] | ResNet-101 | 07++12+COCO | **83.8** | 2016 |
| ***one-stage anchor-based*** | | | | |
| RON320 [106] | VGGNet-16 | 07++12 | 71.7 | 2017 |
| SSD300 [73] | VGGNet-16 | 07++12 | 72.4 | 2016 |
| RON384 [106] | VGGNet-16 | 07++12 | 73.0 | 2017 |
| YOLOv2 544 [80] | Darknet-19 | 07++12 | 73.4 | 2016 |
| RON320++ [106] | VGGNet-16 | 07++12 | 74.5 | 2017 |
| SSD512 [73] | VGGNet-16 | 07++12 | 74.9 | 2016 |
| SSD321 [105] | ResNet-101 | 07++12 | 75.4 | 2017 |
| RON384++ [106] | VGGNet-16 | 07++12 | 75.4 | 2017 |
| SSD300 [73] | VGGNet-16 | 12++07 | 75.8 | 2016 |
| DSSD321 [105] | ResNet-101 | 07++12 | 76.3 | 2017 |
| DSOD300 [103] | DS/64-192-48-1 | 07++12 | 76.3 | 2017 |
| R-SSD300 [143] | VGGNet-16 | 07++12 | 76.4 | 2017 |
| StairNet [144] | VGGNet-16 | 07++12 | 76.4 | 2017 |
| WeaveNet [149] | VGGNet-16 | 07++12 | 77.0 | 2017 |
| SSD300 [73] | VGGNet-16 | 07+12+COCO | 77.5 | 2016 |
| ION [140] | VGGNet-16 | 07+12+S | 76.4 | 2015 |
| PFPNet-R320 [113] | VGGNet-16 | 07++12 | 77.7 | 2017 |
| RefineDet320 [102] | VGGNet-16 | 07++12 | 78.1 | 2017 |
| SSD512 [73] | VGGNet-16 | 07++12 | 78.5 | 2016 |
| DFPR300 [114] | ResNet-101 | 07++12 | 78.7 | 2018 |
| BlitzNet300 [146] | ResNet-50 | 07++12 | 79.0 | 2017 |
| MSA-DNN300 [152] | VGGNet-16 | 07++12 | 79.2 | 2018 |
| SSD300 [73] | VGGNet-16 | 07+12+COCO | 79.3 | 2016 |
| DSOD300 [103] | VGGNet-16 | 07+12+COCO | 79.3 | 2017 |
| SSD512 [105] | ResNet-101 | 07++12 | 79.4 | 2017 |
| DSSD513 [105] | ResNet-101 | 07++12 | 80.0 | 2017 |

query initialization. The authors propose a new Detection Split Transformer that divides the content embedding estimation of cross-attention into two independent parts, one for the classification and the other for box regression embedding. By doing this, they let each cross-attention deal with its specific task. For the content query initialization, they use a mini-detector to learn the content and initialize the positional embedding of the decoder. It is equipped with heads for classification and regression embeddings. Finally, to account for pairs of adjacent object queries in the decoder, they augment the self-attention by the spatial context of the other query in the pair.

### G. COMPARISON: TRANSFORMER-BASED DETECTORS

Table 7 lists a chronological comparison of the strengths and limitations of the two-step anchor-based detection methods mentioned earlier in this paper.

**TABLE 9.** *(Continued.)* **Comparative results on VOC 2012 test set (%).**

| method | backbone | data | mAP | Year |
|---|---|---|---|---|
| ***one-stage anchor-based [continued]*** | | | | |
| SSD512 [73] | VGGNet-16 | 07+12+COCO | 80.0 | 2016 |
| RefineDet512 [102] | VGGNet-16 | 07++12 | 80.1 | 2017 |
| BlitzNet300 [146] | ResNet-50 | 07++12+S+COCO | 80.2 | 2017 |
| DES512 [108] | VGG-16 | 07++12 | 80.3 | 2018 |
| PFPNet-R512 [113] | VGGNet-16 | 07++12 | 80.3 | 2018 |
| RON384++ [106] | VGG-16 | 07+12+COCO | 80.7 | 2017 |
| ESCNet512 | VGGNet-16 | 07++12 | 80.9 | 2019 |
| DFPR512 [114] | ResNet-101 | 07++12 | 81.1 | 2018 |
| MSA-DNN512 [152] | VGGNet-16 | 07++12 | 81.3 | 2018 |
| TripleNet512 [155] | ResNet-101 | 07++12 | 81.9 | 2019 |
| FSSD300 [145] | VGGNet-16 | 07++12+COCO | 82.0 | 2017 |
| ScratchDet300 [104] | Root-ResNet-34 | 07++12+COCO | 82.1 | 2019 |
| SSD512 [73] | VGGNet-16 | 07++12+COCO | 82.2 | 2016 |
| RefineDet320+ [102] | VGGNet-16 | 07++12+COCO | 82.7 | 2017 |
| PFPNet-R320+ [113] | VGGNet-16 | 07++12 | 83.0 | 2018 |
| RetinaNet500+AP-Loss [112] | ResNet-101 | 07++12 | 83.1 | 2019 |
| RefineDet512+ [102] | VGGNet-16 | 07++12 | 83.5 | 2017 |
| PFPNet-R512+ [113] | VGGNet-16 | 07++12 | 83.7 | 2018 |
| BlitzNet512 [146] | ResNet-50 | 07++12+S+COCO | 83.8 | 2017 |
| DAFS320 [111] | VGGNet-16 | 07++12+COCO | 83.9 | 2019 |
| FSSD512 [145] | VGGNet-16 | 07++12+COCO | 84.2 | 2017 |
| RetinaNet500+AP-Loss multi-sc-train [112] | ResNet-101 | 07++12 | 84.5 | 2019 |
| ScratchDet300+ [104] | Root-ResNet-34 | 07++12+COCO | 86.3 | 2019 |
| RefineDet512 ++ [102] | VGGNet-16 | 07++12+COCO | 86.8 | 2017 |
| DAFS320+ [111] | VGGNet-16 | 07++12+COCO | **86.9** | 2019 |
| ***anchor-free*** | | | | |
| YOLOv1 [72] | GoogLeNet | 07++12 | **57.9** | 2016 |

## IX. PERFORMANCE ANALYSIS AND DISCUSSION

This section tests and compares all object detection models in the three benchmark databases in the object detection field. Pascal Voc 2007, Pascal Voc 2012 and MS-COCO. The column ''data'' in the following tables refer to training data.

### A. PASCAL VOC 2007

The results of the tests are listed in Table 8.

### B. PASCAL VOC 2012

The results of the tests are listed in Table 9.

### C. MS-COCO

The results of the tests are listed in Table 10.

### D. TESTING CONSUMPTION

All the frameworks listed below are tested using the Nvidia Titan X GPU (Maxwell architecture) for all experiments, facilitating speed comparison with earlier experiments, as they used the same GPU.

#### 1) PASCAL VOC07

The results of the tests are presented in Table 11.

#### 2) MS-COCO

The results of the tests are presented in Table 12.

### E. DISCUSSION

As we can observe in this survey, most of the tests were performed on the MS-COCO database. Indeed, its large size and rich annotations allow us to evaluate the models on a wide range of images and give a clear picture of how the models generalize. Tables 6 and 7 show that all the models that achieved the highest mAP on Pascal VOC 2007 fall into the anchor-based detectors. All the leading five models belong to the two-stage approach, except for ScratchDet++, which follows the one-stage approach. Copy-Paste achieved an mAP of 88.6% by combining EfficientNet-B7 and NAS-FPN as a backbone. Moreover, it reached an mAP of 89.3% when pre-training on MS-COCO. Copy and paste highlights the importance of copy-and-paste data augmentation. SNIPER, ScratchDet, ACoupleNet, and Faster R-CNN achieved the following mAPs: 86.9%, 86.3%, 85.7%, and 85.6%. Except for the Copy-Paste model, which uses EfficientNet-B7 NAS-FPN as its backbone, all other leading models use one of the following networks: ResNets, Root-Resnets, and VGGNets, proving the powerful performance of these models.

From Table 8, we remark that anchor-based detection methods are the models that scored the best mAPs on Pascal VOC 2012. We also notice that the one-stage anchor-based detectors surpass the two-stage anchor-based detectors, which was the opposite in the past. RefineDet512++ achieved the best mAP of 86.8% with pretraining on the MS-COCO dataset using VGGNet-16. In contrast, the highest mAP without pretraining on MS-COCO belongs to RetinaNet500 using AP-Loss and ResNet-101 as the backbone, with an mAP of 84.5% when applying the multi-scale testing. ScratchDet300+, FSSD512, and BlitzNet obtained an mAP of 86.3%, 84.2%, and 83.8%, respectively. Similar to Pascal VOC 2007 results, the main backbones that achieved the best

**TABLE 10.** MS COCO test-dev 2015 detection results.

| Method | Backbone | Data | mAP@.5 | mAP [.5,.95] | Year |
|---|---|---|---|---|---|
| ***two-stage anchor-based*** | | | | | |
| Fast R-CNN [76] | VGGNet-16 | train | 39.3 | 19.3 | 2015 |
| Fast R-CNN [84] | VGGNet-16 | train | 35.9 | 19.7 | 2015 |
| Fast R-CNN [140] | VGGNet-16 | train | 40.3 | 20.0 | 2015 |
| Fast R-CNN [140] | VGGNet-16 | train | 39.9 | 20.5 | 2015 |
| Faster R-CNN [76] | VGGNet-16 | train | 42.1 | 21.5 | 2017 |
| Faster R-CNN [76] | VGGNet-16 | trainval | 42.7 | 21.9 | 2017 |
| OHEM [138] | VGGNet-16 | trainval | 42.5 | 22.6 | 2016 |
| SIN [139] | VGGNet16 | train | 44.5 | 23.2 | 2018 |
| ION [140] | VGGNet16 | train | 43.2 | 23.6 | 2015 |
| ION [140] | VGGNet-16 | train+S | 46.3 | 24.6 | 2015 |
| OHEM++ [138] | VGGNet-16 | trainval | 45.9 | 25.5 | 2016 |
| Faster R-CNN w Cascade RCNN [87] | VGGNet-16 | train | 44.3 | 26.9 | 2018 |
| Faster R-CNN [59] | ResNet-101 | train | 48.4 | 27.2 | 2016 |
| NoC [137] | ResNet101 | train | 48.4 | 27.2 | 2015 |
| R-FCN [100] | ResNet-101 | train | 48.9 | 27.6 | 2016 |
| R-FCN multi-sc-train [100] | ResNet-101 | train | 49.1 | 27.8 | 2016 |
| MLKP [160] | ResNet101 | trainval35k | 52.4 | 28.6 | 2018 |
| R-FCN [100] | ResNet-101 | trainval | 51.5 | 29.2 | 2016 |
| R-FCN multi-sc-train [100] | ResNet-101 | trainval | 51.9 | 29.9 | 2016 |
| R-FCN w Cascade RCNN [87] | ResNet-50 | train | 49.9 | 30.9 | 2018 |
| R-FCN multi-sc-train, test [100] | ResNet-101 | trainval | 53.2 | 31.5 | 2016 |
| SMN [163] | VGGNet16 | trainval35k | 52.2 | 31.6 | 2017 |
| CoupleNet [154] | ResNet-101 | trainval | 53.5 | 33.1 | 2017 |
| ION [140] | VGGNet-16 | trainval35k+S | 55.7 | 33.1 | 2015 |
| MPN [164] | VGGNet16 | train | 51.9 | 33.2 | 2016 |
| R-FCN w Cascade RCNN [87] | ResNet-101 | train | 52.6 | 33.3 | 2018 |
| ACoupleNet [156] | ResNet-101 | trainval+S | 54.1 | 34.1 | 2019 |
| CoupleNet msc train [154] | ResNet-101 | trainval | 54.8 | 34.4 | 2017 |
| Faster R-CNN G-RMI [23] | Inception-ResNet-V2 | trainval | 55.5 | 34.7 | 2017 |
| Faster R-CNN+++ [59] | ResNet-101-C4 | trainval | 55.7 | 34.9 | 2016 |
| ACoupleNet mst train [156] | ResNet-101 | trainval+S | 55.7 | 35.4 | 2019 |
| RDSNet 600 [167] | ResNet-101 | trainval35k | 55.2 | 36.0 | 2019 |
| Faster R-CNN w/ FPN [63] | ResNet-101-FPN | trainval35k | 59.1 | 36.2 | 2016 |
| FPN FRCN [63] | ResNet-101 | trainval | 59.1 | 36.2 | 2016 |
| TensorMask [168] | ResNet-101 | trainval35k | 95.3 | 37.1 | 2019 |
| Faster R-CNN w/ TDM [169] | Inception-ResNet-v2-TDM | trainval | 57.8 | 37.3 | 2016 |
| TDM [169] | Inception-ResNet-v2-TDM | trainval | 57.8 | 37.3 | 2017 |
| Deformable R-FCN [170] | Aligned-Inception-Resnet | trainval | 58.0 | 37.5 | 2017 |
| RDSNet 800 [167] | ResNet-101 | trainval35k | 58.5 | 38.1 | 2019 |
| Mask R-CNN [77] | ResNet-101-FPN | trainval35k | 60.3 | 38.2 | 2017 |
| FPN [157] | ResNet-101 | trainval | 61.7 | 38.8 | 2018 |
| Gu et al [173] | ResNet-101 | trainval35k | 63.1 | 39.9 | 2018 |
| Relation Network [171] | DCN-101 | trainval35k | 58.6 | 39.0 | 2018 |
| DeepRegionlets [172] | ResNet-101 | trainval35k | 59.8 | 39.3 | 2018 |
| Mask R-CNN [77] | ResNeXt-101-FPN | trainval35k | 62.3 | 39.8 | 2017 |
| DetNet [175] | DetNet-59 | trainval35k | 62.1 | 40.3 | 2018 |
| IoU-Net [176] | ResNet-101 | trainval35k | 59.0 | 40.6 | 2018 |
| FPN [157] | ResNet-101+ResNet-152 | trainval | 64.4 | 40.7 | 2018 |
| Soft-NMS [178] | Aligned-Inception-ResNet | trainval | 62.8 | 40.9 | 2017 |
| SOD-MTGAN [181] | ResNet101 | trainval35k | 63.2 | 41.4 | 2018 |
| LH R-CNN [182] | ResNet101 | trainval35k | - | 41.5 | 2017 |
| G-RMI [23] | Ensemble of Five Models | trainval32k | 61.9 | 41.6 | 2017 |
| FPN-DCN [157] | ResNet-101 | trainval | 64.0 | 41.7 | 2018 |
| Fitness-NMS multi-sc-train [185] | ResNet-101 | trainval35k | 60.9 | 41.8 | 2018 |
| C-Mask RCNN [186] | ResNet-101 | trainval35k | 62.9 | 42.0 | 2018 |
| Cascade R-CNN [87] | ResNet101 | trainval35k | 62.1 | 42.8 | 2018 |
| Libra R-CNN [97] | ResNeXt101-FPN | train | 64.0 | 43.0 | 2019 |
| Revisiting RCNN [157] | ResNet-101+ResNet-152 | trainval | 66.1 | 43.1 | 2018 |
| Revisiting RCNN [157] | ResNet-101+ResNet-152 | trainval35k | 66.1 | 43.1 | 2018 |
| Faster RCNN-FPN+ | ResNet-101 | trainval35k | 63.9 | 44.0 | 2017 |
| PANet [66] | ResNeXt-101 | trainval35k | 65.0 | 45.0 | 2018 |
| D-RFCN + SNIP [90] | DPN-98 | trainval35k | 67.3 | 45.7 | 2018 |
| SNIPER [158] | ResNet101 | trainval35k | 67.0 | 46.1 | 2018 |
| TridentNet [89] | ResNet101-DCN | trainval35k | 67.6 | 46.8 | 2019 |
| Dynamic RCNN [205] | ResNet-101-DCN | trainval35k | 56.9 | 46.9 | 2020 |
| HTC [206] | ResNeXt-101-FPN | trainval35k | 63.9 | 47.1 | 2019 |
| HTC [206] | ResNeXt-101-64x4d | trainval35k | 66.5 | 47.2 | 2019 |
| PANet multi-sc-train [66] | ResNeXt-101 | trainval35k | 67.2 | 47.4 | 2018 |
| DCN-v2 multi-sc-test [223] | ResNet-101 | trainval35k | 67.9 | 46.0 | 2019 |
| D-RFCN + SNIP [90] | Ensemble of three networks | trainval35k | 69.7 | 48.3 | 2018 |

**TABLE 10.** *(Continued.)* **MS COCO test-dev 2015 detection results.**

| method | backbone | data | mAP@.5 | mAP [.5,.95] | year |
|---|---|---|---|---|---|
| ***two-stage anchor-based [continued]*** | | | | | |
| TridentNet [89] | ResNet101-DCN | trainval35k | 69.7 | 48.4 | 2019 |
| Dynamic RCNN multi-sc-train-test [205] | ResNet-101-DCN | trainval35k | 68.6 | 49.2 | 2020 |
| HTC + DCN multi-sc-train [170] | ResNeXt-101-64x4d | trainval35k | 70.3 | 50.8 | 2017 |
| TSD multi-sc-test [191] | SENet154 | trainval35k | 71.9 | 53.3 | 2020 |
| DetectoRS [225] | ResNeXt-101-64x4d | trainval35k | 71.6 | 53.3 | 2021 |
| CBNet multi-sc-test [226] | ResNet-152 | trainval35k | 71.9 | 53.3 | 2022 |
| DetectoRS multi-sc-test [225] | ResNeXt-101-64x4d | trainval35k | **74.2** | 55.7 | 2021 |
| SpineNet-190 (1536) [61] | SpineNet-49 | trainval35k | - | 52.5 | 2020 |
| SpineNet-190 (1280) [61] | SpineNet-49 | trainval35k | - | 52.8 | 2020 |
| Copy-paste [101] | Cascade Eff-B7 NAS-FPN | trainval35k | - | 56.0 | 2021 |
| SoftTeacher [211] | HTC++(Swin-L) | trainval35k | - | 59.1 | 2021 |
| SoftTeacher multi-sc-train [211] | HTC++(Swin-L) | trainval35k | - | **60.4** | 2021 |
| ***one-stage anchor-based*** | | | | | |
| YOLOv2 [80] | Darknet-19 | trainval35k | 44.0 | 21.6 | 2016 |
| SSD300 [73] | VGGNet-16 | trainval35k | 41.2 | 23.2 | 2016 |
| RON320 [106] | VGGNet-16 | trainval | 44.7 | 23.6 | 2017 |
| SSD300 [73] | VGGNet-16 | trainval35k | 43.1 | 25.1 | 2016 |
| RON384 [106] | VGGNet-16 | trainval | 46.5 | 25.4 | 2017 |
| RON320++ [106] | VGGNet-16 | trainval | 47.5 | 26.2 | 2017 |
| SSD512 [73] | VGGNet-16 | trainval35k | 46.5 | 26.8 | 2016 |
| DiCSSD300 [159] | VGG-16 | trainval35 | 46.3 | 26.9 | 2018 |
| FSSD300 [145] | VGGNet-16 | trainval35k | 47.7 | 27.1 | 2017 |
| RON384++ [106] | VGGNet-16 | trainval | 49.5 | 27.4 | 2017 |
| SSD321[105] | ResNet-101 | trainval35k | 45.4 | 28.0 | 2017 |
| DSSD321[105] | ResNet-101 | trainval35k | 46.1 | 28.0 | 2017 |
| STDN300 [107] | DenseNet-169 | trainval | 45.4 | 28.0 | 2018 |
| YOLOv3-320 [116] | Darknet-53 | trainval | 51.5 | 28.2 | 2018 |
| DES300 [108] | VGG-16 | trainval35 | 47.3 | 28.3 | 2018 |
| DFPR300 [114] | VGG-16 | trainval | 48.5 | 28.4 | 2018 |
| SSD512 [73] | VGGNet-16 | trainval35k | 48.5 | 28.8 | 2016 |
| DSOD300 [103] | DS/64-192-48-1 | trainval | 47.3 | 29.3 | 2017 |
| RefineDet320 [102] | VGGNet-16 | trainval35k | 49.2 | 29.4 | 2017 |
| BPN320 [150] | VGGNet-16 | trainval35k | 48.4 | 29.6 | 2018 |
| PFPNet-S300 [113] | VGGNet-16 | trainval35k | 49.6 | 29.6 | 2018 |
| EFIPNet300[151] | VGGNet16 | trainval35 | 48.8* | 30.0 | 2019 |
| RFBNet300 [69] | VGGNet16 | trainval35 | 49.3 | 30.3 | 2018 |
| RetinaNet400 [79] | ResNet-50 | trainval35k | 47.8 | 30.5 | 2017 |
| YOLOv3-416 [116] | Darknet-53 | trainval | 55.3 | 31.0 | 2018 |
| YOLACT-550[161] | D-53-FPN | trainval35 | 51.1 | 31.0 | 2019 |
| BlitzNet300 [146] | ResNet-50 | trainval | 49.7 | 31.1 | 2017 |
| SSD513 [105] | ResNet-101-SSD | trainval35k | 50.4 | 31.2 | 2017 |
| Rev-Dense [162] | VGGNet16 | trainval35 | 52.9 | 31.2 | 2018 |
| DFPR300 [114] | ResNet101 | trainval | 50.5 | 31.3 | 2018 |
| FSSD512 [145] | VGGNet-16 | trainval35k | 52.8 | 31.8 | 2017 |
| PFPNet-R32 [113] | VGGNet-16 | trainval35k | 52.9 | 31.8 | 2018 |
| STDN513 [107] | DenseNet169 | trainval | 51 | 31.8 | 2018 |
| RetinaNet400 [79] | ResNet-101 | trainval35k | 49.5 | 31.9 | 2017 |
| RefineDet320 [102] | ResNet-101 | trainval35k | 51.4 | 32.0 | 2017 |
| RUN512 [147] | VGGNet16 | trainval35 | 53.5 | 32.4 | 2018 |
| RetinaNet500 [79] | ResNet-50 | trainval35k | 50.9 | 32.5 | 2017 |
| ScratchDet300 [104] | Root-ResNet34 | trainval35k | 52 | 32.7 | 2019 |
| MSA-DNN320 [152] | ResNet101 | train2017 | 52.1 | 32.7 | 2018 |
| DES512 [108] | VGGNet16 | trainval35 | 53.2 | 32.8 | 2018 |
| RefineDet512 [102] | VGGNet-16 | trainval35k | 54.5 | 33.0 | 2017 |
| YOLOv3-608 [116] | Darknet-53 | trainval | 57.9 | 33.0 | 2018 |
| BPN512 [150] | VGGNet-16 | trainval35k | 53.1 | 33.1 | 2018 |
| DSSD513 [105] | ResNet-101-DSSD | trainval35k | 53.3 | 33.2 | 2017 |
| EFGRNet320 [110] | VGGNet16 | trainval35k | 53.4 | 33.2 | 2019 |
| DAFS320 [111] | ResNet101 | trainval35k | 52.7 | 33.2 | 2019 |
| PFPNet-S512 [113] | VGGNet-16 | trainval35k | 54.8 | 33.4 | 2018 |
| M2Det320 [165] | VGGNet16 | trainval35k | 52.4 | 33.5 | 2019 |
| HSD320 [153] | VGGNet16 | trainval35k | 53.2 | 33.5 | 2019 |
| YOLACT-700 [161] | ResNet-101-FPN | trainval35k | 54.3 | 33.7 | 2019 |
| RFBNet512 [69] | VGGNet16 | trainval35k | 54.2 | 33.8 | 2018 |
| EfficientDet-D0 (512) [67] | EfficientNet-B3 + BiFPN | trainval35k | 52.2 | 33.8 | 2020 |
| DRN512 [213] | VGGNet16 | trainval35k | 57.1 | 34.3 | 2018 |
| LRF [109] | ResNet-101 | trainval35k | 54.1 | 34.3 | 2019 |
| M2Det320 [165] | ResNet-101 | trainval35k | 53.5 | 34.3 | 2019 |
| RetinaNet500 [79] | ResNet-101 | trainval35k | 53.1 | 34.4 | 2017 |
| RFBNet512-E [69] | VGGNet16 | trainval35k | 55.7 | 34.4 | 2018 |

**TABLE 10.** *(Continued.)* **MS COCO test-dev 2015 detection results.**

| method | backbone | data | mAP@.5 | mAP [.5,.95] | year |
|---|---|---|---|---|---|
| ***one-stage anchor-based [continued]*** | | | | | |
| EFIP512 [151] | VGGNet16 | trainval35k | 55.8 | 34.6 | 2019 |
| DFPR512 [114] | ResNet-101 | trainval35k | 54.3 | 34.6 | 2018 |
| EFIPNet512 [151] | VGGNet16 | trainval35k | 55.8 | 34.6 | 2019 |
| EfficientDet-D0 (512) [67] | EfficientNet-B0 | trainval35k | 53.0 | 34.6 | 2020 |
| RefineDet320+ [102] | VGGNet-16 | trainval35k | 56.1 | 35.2 | 2017 |
| PFPNet-R512 [113] | VGGNet-16 | trainval35k | 57.6 | 35.2 | 2018 |
| BPN320++ [150] | VGGNet-16 | trainval35k | 55.3 | 35.4 | 2018 |
| BlitzNet512 [146] | ResNet-50 | trainval | 55.5 | 35.8 | 2017 |
| YOLOv3-SPP [116] | Darknet-53 | trainval35k | 60.6 | 36.2 | 2018 |
| RefineDet512 [102] | ResNet-101 | trainval35k | 57.5 | 36.4 | 2017 |
| LRF 512 [109] | ResNet-101 | trainval35k | 58.5 | 37.3 | 2019 |
| RetinaNet500+AP-Loss [112] | ResNet-101 | trainval35k | 58.6 | 37.4 | 2019 |
| TripleNet [155] | ResNet101 | trainval35k | 59.3 | 37.4 | 2019 |
| MSA-DNN512 [152] | ResNet101 | train | 55.0 | 37.5 | 2018 |
| RefineDet512+ [102] | VGGNet-16 | trainval35k | 58.7 | 37.6 | 2017 |
| RetinaNet800 [79] | ResNet-101 | trainval | 57.5 | 37.8 | 2017 |
| PFPNet-R320+ [113] | VGGNet-16 | trainval35k | 60.0 | 37.8 | 2018 |
| BPN512++ [150] | VGGNet-16 | trainval35k | 58.0 | 37.9 | 2018 |
| MetaAnchor [81] | ResNet-50 | trainval35k | - | 37.9 | 2018 |
| RefineDet320+ [102] | ResNet-101 | trainval35k | 59.9 | 38.6 | 2017 |
| DAFS512 [111] | ResNet-101 | trainval35k | 58.9 | 38.6 | 2019 |
| EFGRNet512 [110] | ResNet101 | trainval35k | 58.8 | 39.0 | 2019 |
| RetinaNet800 [79] | ResNet-101-FPN | trainval35k | 59.1 | 39.1 | 2017 |
| ScratchDet300+ [104] | Root-ResNet34 | trainval35k | 59.2 | 39.1 | 2019 |
| PFPNet-R512+ [113] | VGGNet-16 | trainval35k | 61.5 | 39.4 | 2018 |
| ConRetinaNet [174] | ResNet-101 | trainval35k | 59.6 | 40.1 | 2019 |
| RetinaNet800 [79] | ResNeXt-101-FPN | trainval35k | 61.1 | 40.8 | 2017 |
| M2Det 800 [165] | VGGNet16 | trainval35k | 59.7 | 41.0 | 2019 |
| Cas-RetinaNet [179] | ResNet101 | trainval35k | 60.7 | 41.1 | 2019 |
| YOLOv4 [180] | CSPDarknet-53 | trainval35k | 62.8 | 41.2 | 2021 |
| GHM [183] | RetinaNet-FPN-ResNeXt-101 | trainval35k | 62.8 | 41.6 | 2019 |
| NATS [184] | ResNeXt101-32?4d | train | 64.3 | 41.6 | 2019 |
| RefineDet512+ multi-sc-train [102] | ResNet-101 | trainval35k | 62.9 | 41.8 | 2017 |
| RetinaNet500+AP-Loss multi-sc-train [112] | ResNet101 | trainval35k | 63.5 | 42.1 | 2019 |
| HSD768 [153] | ResNet101 | trainval35k | 61.2 | 42.3 | 2019 |
| RetinaMask800 [188] | ResNeXt-101-FPN-GN | trainval35k | 62.5 | 42.6 | 2020 |
| GFL [189] | ResNet50 | trainval35k | 61.2 | 42.9 | 2020 |
| EfficientDet-D2 [67] | Efficient-B2 | trainval35k | 62.3 | 43.0 | 2020 |
| YOLOv4 [180] | CSPDarknet-53 | trainval35k | 64.9 | 43.0 | 2021 |
| TSD [191] | ResNet-101 | trainval35k | 64.0 | 43.2 | 2020 |
| SABL [193] | ResNet-101 | trainval35k | 64.7 | 43.2 | 2020 |
| EFGRNet (MS) [110] | ResNet101 | trainval35k | 63.8 | 43.4 | 2019 |
| YOLOv4 608 [180] | CSPDarknet-53 | trainval35k | 65.7 | 43.5 | 2021 |
| MAL [194] | ResNet-101 | trainval35k | 62.8 | 43.6 | 2019 |
| ASFF (800) [65] | Darknet-53 | trainval35k | 64.1 | 43.9 | 2019 |
| M2Det512 multi-sc-train [165] | ResNet101 | trainval35k | 64.4 | 43.9 | 2019 |
| NoisyAnchor [195] | ResNeXt101 | trainval35k | 63.8 | 44.1 | 2020 |
| M2Det800 (MS) [165] | VGGNet16 | trainval35k | 64.6 | 44.2 | 2019 |
| TSP-RCNN [196] | ResNet101 | trainval35k | 63.8 | 44.8 | 2021 |
| PP-YOLO 608 [198] | ResNet50-vd-dcn | trainval35k | 65.2 | 45.2 | 2020 |
| MAL [194] | ResNeXt101 | trainval35k | 65.4 | 45.9 | 2019 |
| ATSS[123] | ResNet-101-DCN | trainval35k | 64.7 | 46.3 | 2020 |
| TSP-RCNN+ [196] | ResNet-101 | trainval35k | 66.0 | 46.5 | 2021 |
| AutoAssign [204] | ResNet-101 | trainval35k | 66.5 | 46.5 | 2020 |
| TSP-RCNN [196] | ResNet-101 | trainval35k | 66.2 | 46.6 | 2021 |
| TSP-RCNN [196] | ResNet-101-DCN | trainval35k | 66.7 | 47.4 | 2021 |
| ATSS [123] | ResNeXt-64x4d-101-DCN | trainval35k | 66.5 | 47.7 | 2020 |
| NAS-FPN 1280 [64] | AmoebaNet-DropBlock | trainval35k | - | 48.3 | 2019 |
| MegDet [118] | ResNet-50 | trainval35k | - | 50.6 | 2018 |
| GFLV2 [71] | ResNet-101-DCN | trainval35k | 69 | 50.6 | 2021 |
| ATSS multiscale testing [123] | ResNeXt-64x4d-101-DCN | trainval35k | 68.9 | 50.7 | 2020 |
| PAA [119] | ResNeXt-32x8d-152-DCN | trainval35k | 69.7 | 50.8 | 2020 |
| SEPC [227] | ResNeXt-101 | trainval35k | 69.7 | 50.8 | 2020 |
| EfficientDet-D7 [67] | EfficientNet-B6 | trainval35k | 71.4 | 52.2 | 2020 |
| GFLV2 multi-sc-train-test [71] | ResNet-101-DCN | trainval35k | 70.9 | 53.3 | 2021 |
| PAA MS [119] | ResNeXt-32x8d-152-DCN | trainval35k | 71.6 | 53.5 | 2020 |
| EfficientDet-D7(1536) [67] | EfficientNet-B6 | trainval35k | 72.4 | 53.7 | 2020 |
| EfficientDet-D7x (1536) [67] | EfficientNet-B7 | trainval35k | **74.3** | 55.1 | 2020 |
| YOLOR [209] | YOLOR-D6 | trainval35k | 73.3 | 55.4 | 2021 |
| DyHead [210] | Swin-L | trainval35k | - | 58.7 | 2021 |

**TABLE 10.** *(Continued.)* **MS COCO test-dev 2015 detection results.**

| method | backbone | data | mAP@.5 | mAP [.5,.95] | year |
|---|---|---|---|---|---|
| ***anchor-free*** | | | | | |
| DeNet101(wide) [166] | ResNet101 | trainval | 53.4 | 33.8 | 2017 |
| GA-Faster-RCNN [82] | ResNet-50 | trainval35k | 59.2 | 39.8 | 2019 |
| ExtremeNet [121] | Hourglass-104 | trainval35k | 55.5 | 40.2 | 2019 |
| CornerNet511 [74] | Hourglass104 | trainval35k | 56.5 | 40.5 | 2018 |
| CenterMask-Lite 600 [177] | VoVNet-39-FPN | trainval35k | - | 40.7 | 2020 |
| FSAF800 [122] | ResNet-101 | trainval35k | 61.5 | 40.9 | 2019 |
| RPDet [78] | ResNeXt-101 | trainval35k | 62.9 | 41.0 | 2019 |
| FoveaBox [187] | ResNeXt-101 | trainval35k | 61.9 | 42.1 | 2020 |
| CornerNet511 multi-sc-train [74] | Hourglass104 | trainval35k | 57.8 | 42.2 | 2018 |
| AB + FSAF 800 [122] | ResNeXt-64x4d-101-FPN | trainval35k | 63.8 | 42.9 | 2019 |
| FreeAnchor [190] | ResNet-101 | train | 62.2 | 43.1 | 2019 |
| FCOS [75] | ResNeXt-64x4d-101 | trainval35k | 62.8 | 43.2 | 2019 |
| Grid R-CNN [94] | ResNeXt-101 | trainval35k | 63.0 | 43.2 | 2019 |
| CornerNet-Lite [192] | Hourglass-54 | trainval35k | - | 43.2 | 2019 |
| ExtremeNet multi-sc-train [121] | Hourglass104 | train | 60.5 | 43.7 | 2019 |
| FoveaBox [187] | ResNeXt-101 | trainval35k | 63.5 | 43.9 | 2020 |
| TSP-FCOS [196] | ResNet-101 | trainval35k | 63.8 | 44.4 | 2021 |
| FCOS [75] | ResNeXt101-64?4d-FPN | trainval35k | 64.1 | 44.7 | 2019 |
| FreeAnchor [190] | ResNeXt101 | train | 64.3 | 44.9 | 2019 |
| CenterNet511 [197] | Hourglass-104 | trainval35k | 62.4 | 44.9 | 2019 |
| RPDet (MS) [78] | ResNet-101-DCN | trainval35k | 66.1 | 45.0 | 2019 |
| YOLOv4 [199] | CSPDarknet-53 | trainval35k | 64.1 | 45.5 | 2021 |
| CentripetalNet [203] | Hourglass-104 | trainval35k | 63.1 | 46.1 | 2020 |
| CenterMask [177] | VoVNet-99-FPN | trainval35k | - | 46.5 | 2020 |
| SAPD [202] | ResNet-101-DCN | trainval35k | 65.9 | 46.0 | 2020 |
| RPDet multi-sc-test [78] | ResNet101-DCN | trainval35k | 67.4 | 46.5 | 2019 |
| CenterNet511 multi-sc-train [197] | Hourglass104 | trainval35k | 64.5 | 47.0 | 2019 |
| SAPD [202] | ResNeXt-101-64x4d-DCN | trainval35k | 67.4 | 47.4 | 2020 |
| YOLOv4-CSP [199] | CSPDarknet-53s | trainval35k | 66.2 | 47.5 | 2021 |
| CentripetalNet (MS) [203] | Hourglass-104 | trainval35k | 65.1 | 48.0 | 2020 |
| BorderDet [207] | ResNeXt-64x4d-101-DCN | trainval35k | 67.1 | 48 | 2020 |
| OTA [124] | ResNeXt-64x4d-101-DCN | trainval35k | 67.6 | 49.2 | 2021 |
| DSLA [126] | Swin-S | trainval35k | 68.1 | 49.2 | 2022 |
| BorderDet multi-sc-test [207] | ResNeXt-64x4d-101-DCN | trainval35k | 68.9 | 50.3 | 2020 |
| OTA multi-sc-test [124] | ResNeXt-64x4d-101-DCN | trainval35k | 68.6 | 51.5 | 2021 |
| YOLOv4-P7 (1536) [199] | CSP-P7 | trainval35k | **73.4** | **55.5** | 2021 |
| ***transformer-based*** | | | | | |
| DETR-DC5+ [130] | ResNet101 | trainval35k | 64.7 | 44.9 | 2020 |
| Anchor DETR-DC5 [134] | ResNet-101 | trainval35k | 65.7 | 45.1 | 2022 |
| SMCA [132] | ResNet-50 | trainval35k | 65.5 | 45.6 | 2021 |
| SM-NAS: E5 [200] | Searched Backbone | trainval35k | 64.6 | 45.9 | 2019 |
| Conditional DETR-DC5 [201] | ResNet101 | trainval35k | 66.8 | 45.9 | 2021 |
| DESTR-DC5 [135] | ResNet-101 | trainval35k | 67.6 | 46.8 | 2022 |
| MAL multi-sc-train [194] | ResNeXt101 | trainval35k | 66.1 | 47.0 | 2019 |
| Deformable DETR [208] | ResNeXt-101 + DCN | trainval35k | 69.7 | 50.1 | 2021 |
| Deformable DETR multi-sc-train [208] | ResNeXt-101 + DCN | trainval35k | **71.9** | 52.3 | 2021 |
| Swin-L [133] | HTC++ | trainval35k | - | 57.7 | 2021 |
| Swin-L MS [133] | HTC++ | trainval35k | - | 58.7 | 2021 |
| Swin V2-G [212] | HTC++ | trainval35k | - | **63.1** | 2022 |

results were VGG networks, residual networks, and Root-ResNets.

For the models tested on the MS-COCO dataset, we can notice the intense competition between different approaches. The first four positions belong to different object detection approaches. So far, the Swin V2-G model based on transformers and the HTC++ backbone is the winner, with an mAP of 63.1%. Ranking second, we find Copy-Paste, which belongs to the anchor-based model family, with an mAP of 56.0%. Copy-Paste uses a combination of Cascade Eff-B7 and NAS-FPN. In third place, we find YOLOv4-P7, which falls into the anchor-free detector family with an mAP of 55.5%. YOLOv4-P7 uses the CSP-P7 network as its backbone. In fourth place, we have the EfficientDet-D7x model, which achieved an mAP of 55.1% and used the EfficientNet-B7 network as its backbone. EfficientDet-D7x belongs to the one-step anchor-based object detector family. In MS-COCO, the backbones that assisted in achieving an mAP greater than 50.0% are ResNets, ResNeXts, Efficient Nets, SpineNet, CSP, and HTC++.

Table 11 shows that all the fast object detection algorithms belong to the one-stage anchor-based approach family when implementing object detection models in a real-time environment. However, achieving high accuracy with many frames per second is difficult, as in the case of Fast YOLO, which achieved 155 FPS while obtaining only 55.7% mAP. We can

**TABLE 11. Comparison of testing consumption on VOC 07 test set.**

| method | backbone | data | input size | #boxes | mAP | fps |
|---|---|---|---|---|---|---|
| ***two-stage anchor-based*** | | | | | | |
| MR-CNN [142] | VGGNet-16 | 07+12 | 1000 × 600 | 250 | 78.2 | 0.03 |
| Fast R-CNN [84] | VGGNet-16 | 07+12 | 1000 × 600 | 2000 | 70.0 | 0.5 |
| HyperNet [92] | VGGNet-16 | 07+12 | 1000 × 600 | 100 | 76.3 | 0.88 |
| ION [140] | VGGNet-16 | 07+12 | 1000 × 600 | 4000 | 76.5 | 1.25 |
| Faster R-CNN [76] | ResNet-101 | 07+12 | 1000 × 600 | 300 | 76.4 | 2.4 |
| Faster R-CNN [76] | VGGNet-16 | 07+12 | 1000 × 600 | 300 | 73.2 | 7 |
| OHEM [138] | VGGNet-16 | 07+12 | 1000 × 600 | 300 | 46.6 | 7 |
| CoupleNet [154] | ResNet-101 | 07+12 | 1000 × 600 | 300 | **82.7** | 8.2 |
| R-FCN [41] | ResNet-101 | 07+12 | 1000 × 600 | 300 | 80.5 | 9 |
| Faster R-CNN | ZFNet | 07+12 | 1000 × 600 | 300 | 62.1 | **18** |
| ***one-stage anchor-based*** | | | | | | |
| DSSD [105] | ResNet-101 | 07+12 | 513 × 513 | 43688 | 81.5 | 5.5 |
| SSD [105] | ResNet-101 | 07+12 | 513 × 513 | 43688 | 80.6 | 6.8 |
| DSSD [105] | ResNet-101 | 07+12 | 321 × 321 | 17080 | 78.6 | 9.5 |
| SSD [105] | ResNet-101 | 07+12 | 321 × 321 | 17080 | 77.1 | 11.2 |
| RON384 [106] | VGGNet-16 | 07+12 | 384 × 384 | 30600 | 75.4 | 15 |
| R-SSD [143] | VGGNet-16 | 07+12 | 512 × 512 | 24564 | 80.8 | 16.6 |
| DSOD300 [103] | DS/64-192-48-1 | 07+12 | 300 × 300 | 8732 | 77.7 | 17.4 |
| SSD512 [73] | VGGNet-16 | 07+12 | 512 × 512 | 24564 | 79.8 | 19 |
| SSD | VGGNet16 | 07+12 | 512 × 512 | 24564 | 76.8 | 19 |
| BlitzNet [146] | ResNet-50 | 07+12 | 512 × 512 | 32766 | 81.5 | 19.5 |
| PFPNet-R512 [113] | VGGNet-16 | 07+12 | 512 × 512 | 16320 | **82.3** | 24 |
| BlitzNet [146] | ResNet-50 | 07+12 | 300 × 300 | 45390 | 79.1 | 24 |
| RefineDet512 [102] | VGGNet-16 | 07+12 | 512 × 512 | 16320 | 81.8 | 24.1 |
| ESSD [148] | VGGNet-16 | 07+12 | 300 × 300 | - | 79.4 | 25 |
| PFPNet-S512 [113] | VGGNet-16 | 07+12 | 512 × 512 | 24564 | 81.8 | 26 |
| PFPNet-R320 [113] | VGGNet-16 | 07+12 | 320 × 320 | 6375 | 80.7 | 33 |
| R-SSD [143] | VGGNet-16 | 07+12 | 300 × 300 | 8732 | 78.5 | 35 |
| PFPNet-S300 [113] | VGGNet-16 | 07+12 | 300 × 300 | 8732 | 79.9 | 39 |
| RUN [147] | VGGNet-16 | 07+12 | 300 × 300 | - | 79.2 | 40 |
| RefineDet320 [102] | VGGNet-16 | 07+12 | 320 × 320 | 6375 | 80.0 | 40.3 |
| SSD300 | VGGNet16 | 07+12 | 300 × 300 | 8732 | 74.3 | 46 |
| SSD [73] | VGGNet-16 | 07+12 | 300 × 300 | 8732 | 77.2 | 46 |
| WeaveNet [149] | VGGNet-16 | 07+12 | 320 × 320 | - | 79.7 | 50 |
| DES [108] | VGGNet-16 | 07+12 | 300 × 300 | - | 79.7 | 76.8 |
| EFIPNet[151] | VGGNet-16 | 07+12 | 300 × 300 | - | 80.4 | **111** |
| YOLOv2 [80] | Darknet-19 | 07+12 | 544 × 544 | 845 | 78.6 | 40 |
| YOLOv2 [80] | Darknet-19 | 07+12 | 480 × 480 | - | 77.8 | 59 |
| YOLOv2 [80] | Darknet-19 | 07+12 | 416 × 416 | - | 76.8 | 67 |
| YOLOv2 [80] | Darknet-19 | 07+12 | 352 × 352 | - | 73.7 | 81 |
| YOLOv2 [80] | Darknet-19 | 07+12 | 288 × 288 | - | 69.0 | 91 |
| ***anchor-free*** | | | | | | |
| YOLO [72] | GoogleNet [120] | 07+12 | 448 × 448 | 98 | **63.4** | 45 |
| Fast YOLO [72] | GoogleNet [120] | 07+12 | 448 × 448 | 98 | 52.7 | **155** |

spot, for example, that a model like EFIPNet managed to have a balance. EFIPNet achieved an mAP of 80.4% and an impressive FPS of 111 and used VGGNet-16 as its backbone. RefineDet320 achieved an mAP of 80.0% and 40 FPS and used VGGNet as a backbone.

According to Table 12, we can observe that all the fast object detection models belong to the anchor-based single-step object detection models. In addition, we can see that some models have successfully balanced detection accuracy and runtime speed. For example, YOLOv4, which uses CSPDarknet-53, achieved an mAP of 41.2% with 54 FPS. EfficientDet-D2, which uses the Efficient-B2 backbone, achieved an mAP of 43.0% with 41.7 FPS. Furthermore, no two-stage object detector model has performed well in real-time. (FPS > 30). RDSNet has 17 FPS and an mAP of 36.0%. In comparison, the anchor-free detectors such as CornerNet or ATSS could only attain 4.4 and 7 FPS, respectively. Therefore, we conclude that the anchor-based one-step detectors are still the fastest.

Figure 5, shows the evolution of the accuracy in the three datasets: VOC07, VOC21, and MS-COCO, between 2013 and 2022. The figure also displays the winning detection model for each year within each dataset. For VOC07 and VOC12, the accuracy is presented by mAP, while for MS-COCO, it is by mAP [.5,.95]. The chart shows that the accuracy has evolved in VOC07 from 58.5% in 2013 through the Model R-CNN BB to 89.3% in 2021 through the Copy-Paste model. This means an increase of more than 30%. The same in VOC12, with an increase in accuracy of over 33% during the same period. While in MS-COCO, there was an improvement in accuracy of 40% between 2015, with a value of 23.6 through ION and 63.1 in 2022 through the SwinV2-G model. We also note that accuracy is improved every year in the MS-COCO dataset. Thus, for example, in VOC12, the

**TABLE 12. Comparison of testing consumption on MS-COCO test set.**

| method | backbone | data | mAP@.5 | mAP [.5,.95] | fps |
|---|---|---|---|---|---|
| ***transformer-based*** | | | | | |
| DETR-DC5+ [130] | ResNet101 | trainval35k | **64.7** | **44.9** | **10** |
| ***anchor-free*** | | | | | |
| AB +FSAF 800 [122] | ResNeXt-64x4d-101-FPN | trainval35k | 63.8 | 42.9 | 2.8 |
| SAPD [202] | ResNeXt-101-64x4d-DCN | trainval35k | 67.4 | 47.4 | 4.5 |
| FSAF800 [122] | ResNet-101 | trainval35k | 61.5 | 40.9 | 5.6 |
| YOLOv4-P7 (1536) | CSP-P7 | trainval35k | **73.4** | **55.5** | **17** |
| CornerNet511 [74] | Hourglass104 | trainval35k | 56.5 | 40.5 | 4.4 |
| ***two-stage anchor-based*** | | | | | |
| Mask R-CNN [75] | ResNeXt-101-FPN | trainval35k | 62.3 | 39.8 | 3.3 |
| Fitness-NMS multi-sc-train [185] | ResNet-101 | trainval35k | 60.9 | 41.8 | 5.0 |
| Faster R-CNN w/ FPN [63] | ResNet-101-FPN | trainval35k | 59.1 | 36.2 | 6 |
| OHEM++ [138] | VGGNet-16 | trainval | 45.9 | 25.5 | 7 |
| Cascade R-CNN[87] | ResNet101 | trainval35k | 62.1 | 42.8 | 7.1 |
| CoupleNet msc train [154] | ResNet-101 | trainval | 54.8 | 34.4 | 8.2 |
| R-FCN multi-sc-train [100] | ResNet-101 | trainval | 51.9 | 29.9 | 9 |
| SABL [193] | ResNet-101 | trainval35k | 64.7 | 43.2 | 13 |
| RDSNet 600 [167] | ResNet-101 | trainval35k | **55.2** | **36.0** | **17** |
| ***one-stage anchor-based*** | | | | | |
| RetinaNet800 [79] | ResNet-101 | trainval | 57.5 | 37.8 | 5.1 |
| DSSD513 [105] | ResNet-101-DSSD | trainval35k | 53.3 | 33.2 | 5.5 |
| ATSS [123] | ResNeXt-64x4d-101-DCN | trainval35k | **66.5** | **47.7** | 7 |
| DSSD321 [105] | ResNet-101 | trainval35k | 46.1 | 28.0 | 9.5 |
| RetinaNet500 [79] | ResNet-101 | trainval35k | 53.1 | 34.4 | 11.1 |
| M2Det 800 [165] | VGGNet16 | trainval35k | 59.7 | 41.0 | 11.8 |
| YOLOv3-608 [116] | Darknet-53 | trainval | 57.9 | 33.0 | 20 |
| YOLOv3-SPP [116] | Darknet-53 | trainval35k | 60.6 | 36.2 | 20 |
| M2Det320 [165] | ResNet-101 | trainval35k | 53.5 | 34.3 | 21.7 |
| SSD512 [73] | VGGNet-16 | trainval35k | 48.5 | 28.8 | 22 |
| RefineDet512 [102] | VGGNet-16 | trainval35k | 54.5 | 33.0 | 22.3 |
| PFPNet-R512 [113] | VGGNet-16 | trainval35k | 57.6 | 35.2 | 24 |
| RFBNet512-E[69] | VGGNet16 | trainval35k | 55.7 | 34.4 | 24.3 |
| ASFF (800) [65] | Darknet-53 | trainval35k | 64.1 | 43.9 | 29 |
| LRF 512 [109] | ResNet-101 | trainval35k | 58.5 | 37.3 | 31.3 |
| PFPNet-R32 [113] | VGGNet-16 | trainval35k | 52.9 | 31.8 | 33 |
| RFBNet512 [69] | VGGNet16 | trainval35k | 54.2 | 33.8 | 33.3 |
| M2Det320 [165] | VGGNet16 | trainval35k | 52.4 | 33.5 | 33.4 |
| EFIPNet512[151] | VGGNet16 | trainval35k | 55.8 | 34.6 | 34 |
| DAFS512 [111] | VGGNet-16 | trainval35k | 52.9 | 33.8 | 35 |
| RefineDet320 [102] | VGGNet-16 | trainval35k | 49.2 | 29.4 | 38.4 |
| DiCSSD300 [159] | VGGNet-16 | trainval35k | 46.3 | 26.9 | 40.8 |
| EfficientDet-D2[67] | Efficient-B2 | trainval35k | 62.3 | 43.0 | 41.7 |
| SSD300 [73] | VGGNet-16 | trainval35k | 43.1 | 25.1 | 43 |
| LRF [109] | ResNet-101 | trainval35k | 51.1 | 34.3 | 52.6 |
| YOLOv4 [180] | CSPDarknet-53 | trainval35k | 62.8 | 41.2 | 54 |
| RFBNet300 [69] | VGGNet16 | trainval35k | 49.3 | 30.3 | **66.7** |

accuracy has stayed the same since 2017, remaining at the value of 86.8% realized by RefineDet. Likewise, in VOC07, the accuracy has only increased by 2.4% since 2018 with the introduction of Copy-Paste.

Figure 6 shows the evolution of different types of object detection models in the MS-COO dataset between 2015 and 2022. It can be seen that anchor-based two-stage models were the first to be evaluated in MS-COO in 2015, followed by anchor-based one-stage in 2016, anchor-free in 2017, and transform-based in 2020. So far, the most successful family is the transform-based with SwinV2-G, followed by the anchor-based two-stage with SoftTeacher, then the anchor-based one-stage with DyHead, and finally, the anchor-free one-stage detectors with YOLOv4-P7. We note a difference of more than 7% between the best transformer-based detector, SwinV2-G, and the best anchor-free detector, YOLOv4-P7. The anchor-based two-stage increased by 26%, starting with ION with an accuracy of 33.1 in 2015, reaching 59.1 in 2021 with the SoftTeacher model. For the anchor-based one-stage detectors, in 2016, SSD achieved an accuracy of 28.8%, and in 2021 DyHead achieved an accuracy of 87.7%, representing an enhancement of 30%. DetNet101, a model of the anchor-free detector family, reached an accuracy of 33.8% in 2017, and in 2021 YOLOv4-P7 increased the accuracy by more than 21%, reaching 55.5%. The most recently published transformer-based detectors achieved the best results with SwinV2-G in 2022 with an accuracy of 63.1%, while the first pure model based on transformers, DETR, achieved only 44.9% in 2020.

Figure 7 illustrates the number of detection models evaluated in MS-COCO by each detector family between 2015 and 2022. We find that 2018 was the most productive year with

more than 30 published models, of which half were anchor-based two-stage models, and the other half were anchor-based one-stage methods, and with the publication of only one anchor-free model. We also notice that anchor-based two-stage methods dominated the literature between 2015 and 2018 with more than 36 published models, whereas between 2018 and 2020, more than 36 anchor-based one-stage models were published. One can also spot that anchor-based models have evolved from 2015 to 2018. After 2018 they start losing proportion towards other detection families, such as anchor-free and transformer-based detectors. For example, more than 15 different models of the anchor-based two-stage family were introduced in 2018, while just one year later, only five models were released. In 2020, only two models were released, while more than six anchor-free detectors were released in the same year. As soon as they appeared in 2020, the transform-based detectors continuously expanded.

Figure 8 shows that about half of the detection models based on deep learning and evaluated in the MS-COCO dataset were introduced between 2018 and 2019. Then after 2019, the number of published models decreased yearly, with a value of 14% in 2020, 11.6% in 2021, and 3.3% in 2022.

## X. CONCLUSION AND FUTURE DIRECTIONS

In this paper, we presented an overview of the current state of object detection based on deep learning. We have provided the most detailed survey covering dozens of object detection models. We divided the models into four main approaches: two-stage anchor-based detectors, one-stage anchor-based detectors, anchor-free detectors, and transformer-based detectors. We tested and evaluated all models in major object detection databases such as Pascal VOC and MS-COCO. We determined that single-stage detectors have improved and rival two-stage detectors' accuracy. Furthermore, with the emergence of transformers in vision tasks, transformer-based detectors have achieved peak results, such as Swin-L and Swin V2, which achieved an mAP of 57.7% and 63.1%, respectively, in the MS-COCO dataset.

Object detection is an active area of research that is constantly evolving, and there are several promising future directions that researchers are exploring.

1) **Speed-accuracy trade-off:** Increasing the accuracy of an object detection algorithm requires more computational resources and longer processing times. Decreasing the accuracy can lead to faster processing times but lower detection performance. Therefore, researchers consistently aim to improve the accuracy and speed of object detection algorithms by using more efficient architectures and training methods to enable real-time and low-power applications, especially in complex scenes with occlusions or cluttered backgrounds.
2) **Tiny object detection:** Tiny object detection is a specific case of object detection focusing on detecting and localizing very small objects in images or videos. It remains challenging because extracting information from small objects with only a few pixels is difficult. These objects may be so small that they are barely visible or partially occluded by other objects in the scene. Tiny object detection has many potential applications, such as detecting small animals in wildlife monitoring, identifying minor defects in manufacturing processes, and medical imaging.
3) **3D object detection:** With the increasing availability of 3D sensors, there is a growing interest in 3D object detection. Unlike 2D object detection, which estimates the location and size of objects in a two-dimensional image, 3D object detection involves estimating objects' position, orientation, and dimensions in three-dimensional space. 3D object detection can be helpful in applications such as augmented reality, robotics, and autonomous driving, where accurate knowledge of the 3D environment is necessary for navigation and interaction with the physical world.
4) **Multi-modal object detection:** involves detecting objects from multiple visual and textual sources, such as images, videos, and audio, enabling more comprehensive and accurate object detection in complex scenarios. Multi-modal detection can be helpful in applications such as autonomous driving, where multiple sensors detect objects around a vehicle.
5) **Few-shot learning:** Few-shot learning is an area of research that aims to develop algorithms to learn to detect objects from just a few examples. This is particularly useful when collecting large amounts of labeled data is difficult or expensive. Those models will work with limited data or in low-resource settings.

Overall, the future of object detection using deep learning is promising, with many exciting developments for future research.

## REFERENCES

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015, doi: 10.1038/nature14539.
[2] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 9, pp. 1627–1645, Sep. 2010, doi: 10.1109/TPAMI.2009.167.
[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, vol. 25, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Red Hook, NY, USA: Curran Associates, 2012, pp. 1097–1105. Accessed: Oct. 22, 2019. [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf
[4] C. Chen, A. Seff, A. Kornhauser, and J. Xiao, "DeepDriving: Learning affordance for direct perception in autonomous driving," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Santiago, Chile, Dec. 2015, pp. 2722–2730, doi: 10.1109/ICCV.2015.312.
[5] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3D object detection network for autonomous driving," 2016, *arXiv:1611.07759*. Accessed: Oct. 21, 2019.
[6] S. Ramos, S. Gehrig, P. Pinggera, U. Franke, and C. Rother, "Detecting unexpected obstacles for self-driving cars: Fusing deep learning and geometric modeling," 2016, *arXiv:1612.06573*. Accessed: Oct. 21, 2019.
[7] J. Ni, K. Shen, Y. Chen, W. Cao, and S. X. Yang, "An improved deep network-based scene classification method for self-driving cars," *IEEE Trans. Instrum. Meas.*, vol. 71, pp. 1–14, 2022, doi: 10.1109/TIM.2022.3146923.

[8] C. Nobata, J. Tetreault, A. Thomas, Y. Mehdad, and Y. Chang, "Abusive language detection in online user content," in *Proc. 25th Int. Conf. World Wide Web*, Montreal, QC, Canada, Apr. 2016, pp. 145–153, doi: 10.1145/2872427.2883062.

[9] A.-M. Founta, D. Chatzakou, N. Kourtellis, J. Blackburn, A. Vakali, and I. Leontiadis, "A unified deep learning architecture for abuse detection," 2018, *arXiv:1802.00385*. Accessed: Oct. 21, 2019.

[10] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Santiago, Chile, Dec. 2015, pp. 3730–3738, doi: 10.1109/ICCV.2015.425.

[11] W. Liu, I. Hasan, and S. Liao, "Center and scale prediction: Anchor-free approach for pedestrian and face detection," *Pattern Recognit.*, vol. 135, Mar. 2023, Art. no. 109071, doi: 10.1016/j.patcog.2022.109071.

[12] M. Baccouche, F. Mamalet, C. Wolf, C. Garcia, and A. Baskurt, "Sequential deep learning for human action recognition," in *Human Behavior Understanding*, A. A. Salah and B. Lepri, Eds. Berlin, Germany: Springer, 2011, pp. 29–39, doi: 10.1007/978-3-642-25446-8_4.

[13] Z. Sun, Q. Ke, H. Rahmani, M. Bennamoun, G. Wang, and J. Liu, "Human action recognition from various data modalities: A review," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 3, pp. 3200–3225, Mar. 2022, doi: 10.1109/TPAMI.2022.3183112.

[14] A. A. Cruz-Roa, J. E. A. Ovalle, A. Madabhushi, and F. A. G. Osorio, "A deep learning architecture for image representation, visual interpretability and automated basal-cell carcinoma cancer detection," in *Medical Image Computing and Computer-Assisted Intervention—MICCAI 2013*, C. Salinesi, M. C. Norrie, and Ó. Pastor, Eds. Berlin, Germany: Springer, 2013, pp. 403–410, doi: 10.1007/978-3-642-40763-5_50.

[15] A. B. Nassif, M. A. Talib, Q. Nasir, Y. Afadar, and O. Elgendy, "Breast cancer detection using artificial intelligence techniques: A systematic literature review," *Artif. Intell. Med.*, vol. 127, May 2022, Art. no. 102276, doi: 10.1016/j.artmed.2022.102276.

[16] I. Lenz, H. Lee, and A. Saxena, "Deep learning for detecting robotic grasps," *Int. J. Robot. Res.*, vol. 34, nos. 4–5, pp. 705–724, Apr. 2015, doi: 10.1177/0278364914549607.

[17] Z. Zhou, L. Li, A. Fürsterling, H. J. Durocher, J. Mouridsen, and X. Zhang, "Learning-based object detection and localization for a mobile robot manipulator in SME production," *Robot. Comput.-Integr. Manuf.*, vol. 73, Feb. 2022, Art. no. 102229, doi: 10.1016/j.rcim.2021.102229.

[18] W. Wang, X. Wu, X. Yuan, and Z. Gao, "An experiment-based review of low-light image enhancement methods," *IEEE Access*, vol. 8, pp. 87884–87917, 2020, doi: 10.1109/ACCESS.2020.2992749.

[19] G. Guo, H. Wang, C. Shen, Y. Yan, and H.-Y.-M. Liao, "Automatic image cropping for visual aesthetic enhancement using deep neural networks and cascaded regression," *IEEE Trans. Multimedia*, vol. 20, no. 8, pp. 2073–2085, Aug. 2018, doi: 10.1109/TMM.2018.2794262.

[20] A. B. Amjoud and M. Amrouch, "Transfer learning for automatic image orientation detection using deep learning and logistic regression," *IEEE Access*, vol. 10, pp. 128543–128553, 2022, doi: 10.1109/ACCESS.2022.3225455.

[21] K. Aurangzeb, S. Aslam, M. Alhussein, R. A. Naqvi, M. Arsalan, and S. I. Haider, "Contrast enhancement of fundus images by employing modified PSO for improving the performance of deep learning models," *IEEE Access*, vol. 9, pp. 47930–47945, 2021, doi: 10.1109/ACCESS.2021.3068477.

[22] W. Zhiqiang and L. Jun, "A review of object detection based on convolutional neural network," in *Proc. 36th Chin. Control Conf. (CCC)*, Jul. 2017, pp. 11104–11109, doi: 10.23919/ChiCC.2017.8029130.

[23] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, and K. Murphy, "Speed/accuracy trade-offs for modern convolutional object detectors," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Honolulu, HI, USA, Jul. 2017, pp. 3296–3297, doi: 10.1109/CVPR.2017.351.

[24] S. Agarwal, J. O. D. Terrail, and F. Jurie, "Recent advances in object detection in the age of deep convolutional neural networks," 2018, *arXiv:1809.03193*.

[25] Z.-Q. Zhao, P. Zheng, S.-T. Xu, and X. Wu, "Object detection with deep learning: A review," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 11, pp. 3212–3232, Nov. 2019, doi: 10.1109/TNNLS.2018.2876865.

[26] A. Borji, M.-M. Cheng, Q. Hou, H. Jiang, and J. Li, "Salient object detection: A survey," *Comput. Vis. Media*, vol. 5, no. 2, pp. 117–150, Jun. 2019, doi: 10.1007/s41095-019-0149-9.

[27] Z. Zou, K. Chen, Z. Shi, Y. Guo, and J. Ye, "Object detection in 20 years: A survey," 2019, *arXiv:1905.05055*.

[28] L. Liu, W. Ouyang, X. Wang, P. Fieguth, J. Chen, X. Liu, and M. Pietikäinen, "Deep learning for generic object detection: A survey," *Int. J. Comput. Vis.*, vol. 128, pp. 261–318, 2020, doi: 10.1007/s11263-019-01247-4.

[29] G. Huang, I. Laradji, D. Vazquez, S. Lacoste-Julien, and P. Rodriguez, "A survey of self-supervised and few-shot object detection," 2021, *arXiv:2110.14711*.

[30] W. Wang, Q. Lai, H. Fu, J. Shen, H. Ling, and R. Yang, "Salient object detection in the deep learning era: An in-depth survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 6, pp. 3239–3259, Jun. 2022, doi: 10.1109/TPAMI.2021.3051099.

[31] K. Tong and Y. Wu, "Deep learning-based detection from the perspective of small or tiny objects: A survey," *Image Vis. Comput.*, vol. 123, Jul. 2022, Art. no. 104471, doi: 10.1016/j.imavis.2022.104471.

[32] X. Wu, D. Sahoo, and S. C. H. Hoi, "Recent advances in deep learning for object detection," *Neurocomputing*, vol. 396, pp. 39–64, Jul. 2020, doi: 10.1016/j.neucom.2020.01.085.

[33] N.-D. Nguyen, T. Do, T. D. Ngo, and D.-D. Le, "An evaluation of deep learning methods for small object detection," *J. Electr. Comput. Eng.*, vol. 2020, Apr. 2020, Art. no. e3189691, doi: 10.1155/2020/3189691.

[34] Y. Liu, P. Sun, N. Wergeles, and Y. Shang, "A survey and performance evaluation of deep learning methods for small object detection," *Expert Syst. Appl.*, vol. 172, Jun. 2021, Art. no. 114602, doi: 10.1016/j.eswa.2021.114602.

[35] 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, doi: 10.1109/ACCESS.2019.2939201.

[36] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Kauai, HI, USA, Dec. 2001, pp. I-511–I-518, doi: 10.1109/CVPR.2001.990517.

[37] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, San Diego, CA, USA, Jun. 2005, pp. 886–893, doi: 10.1109/CVPR.2005.177.

[38] P. Felzenszwalb, D. McAllester, and D. Ramanan, "A discriminatively trained, multiscale, deformable part model," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Anchorage, AK, USA, Jun. 2008, pp. 1–8, doi: 10.1109/CVPR.2008.4587597.

[39] S. Ullman, M. Vidal-Naquet, and E. Sali, "Visual features of intermediate complexity and their use in classification," *Nature Neurosci.*, vol. 5, no. 7, pp. 682–687, Jul. 2002, doi: 10.1038/nn870.

[40] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray, "Visual categorization with bags of keypoints," in *Proc. Workshop Stat. Learn. Comput. Vis. ECCV*, May 2004, vol. 1, nos. 1–22, pp. 1–2.

[41] F.-F. Li and P. Perona, "A Bayesian hierarchical model for learning natural scene categories," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, vol. 2, San Diego, CA, USA, Jun. 2005, pp. 524–531, doi: 10.1109/CVPR.2005.16.

[42] J. Sivic, B. C. Russell, A. A. Efros, A. Zisserman, and W. T. Freeman, "Discovering objects and their location in images," in *Proc. 10th IEEE Int. Conf. Comput. Vis. (ICCV)*, vol. 1, Beijing, China, Oct. 2005, pp. 370–377, doi: 10.1109/ICCV.2005.77.

[43] S. Agarwal and D. Roth, "Learning a sparse representation for object detection," in *Computer Vision—ECCV 2002*, A. Heyden, G. Sparr, M. Nielsen, and P. Johansen, Eds. Berlin, Germany: Springer, 2002, pp. 113–127, doi: 10.1007/3-540-47979-1_8.

[44] H. Schneiderman and T. Kanade, "Object detection using the statistics of parts," *Int. J. Comput. Vis.*, vol. 56, no. 3, pp. 151–177, Feb. 2004, doi: 10.1023/B:VISI.0000011202.85607.00.

[45] K. E. A. van de Sande, J. R. R. Uijlings, T. Gevers, and A. W. M. Smeulders, "Segmentation as selective search for object recognition," in *Proc. Int. Conf. Comput. Vis.*, Barcelona, Spain, Nov. 2011, pp. 1879–1886, doi: 10.1109/ICCV.2011.6126456.

[46] R. Lienhart and J. Maydt, "An extended set of Haar-like features for rapid object detection," in *Proc. Int. Conf. Image Process.*, Rochester, NY, USA, 2002, pp. I-900–I-903, doi: 10.1109/ICIP.2002.1038171.

[47] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, Nov. 2004, doi: 10.1023/B:VISI.0000029664.99615.94.

[48] H. Bay, T. Tuytelaars, and L. Van Gool, "SURF: Speeded up robust features," in *Computer Vision—ECCV 2006*, A. Leonardis, H. Bischof, and A. Pinz, Eds. Berlin, Germany: Springer, 2006, pp. 404–417, doi: 10.1007/11744023_32.

[49] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "BRIEF: Binary robust independent elementary features," in *Computer Vision—ECCV 2010*, K. Daniilidis, P. Maragos, and N. Paragios, Eds. Berlin, Germany: Springer, 2010, pp. 778–792, doi: 10.1007/978-3-642-15561-1_56.
[50] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, Sep. 1995, doi: 10.1007/BF00994018.
[51] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *J. Comput. Syst. Sci.*, vol. 55, no. 1, pp. 119–139, Aug. 1997, doi: 10.1006/jcss.1997.1504.
[52] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Trans. Inf. Theory*, vol. IT-13, no. 1, pp. 21–27, Jan. 1967, doi: 10.1109/TIT.1967.1053964.
[53] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal visual object classes (VOC) challenge," *Int. J. Comput. Vis.*, vol. 88, no. 2, pp. 303–338, Jun. 2010, doi: 10.1007/s11263-009-0275-4.
[54] *ImageNet Large Scale Visual Recognition Competition 2010 (ILSVRC2010)*. Accessed: Oct. 22, 2019. [Online]. Available: http://www.image-net.org/challenges/LSVRC/2010/
[55] *MNIST Demos on Yann LeCun's Website*. Accessed: Oct. 22, 2019. [Online]. Available: http://yann.lecun.com/exdb/lenet/
[56] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár, "Microsoft COCO: Common objects in context," 2014, *arXiv:1405.0312*. Accessed: Oct. 26, 2019.
[57] A. B. Amjoud and M. Amrouch, "Convolutional neural networks backbones for object detection," in *Image and Signal Processing*, A. El Moataz, D. Mammass, A. Mansouri, and F. Nouboud, Eds. Cham, Switzerland: Springer, 2020, pp. 282–289, doi: 10.1007/978-3-030-51935-3_30.
[58] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*. Accessed: Oct. 22, 2019.
[59] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Las Vegas, NV, USA, Jun. 2016, pp. 770–778, doi: 10.1109/CVPR.2016.90.
[60] M. Tan and Q. V. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," 2019, *arXiv:1905.11946*. Accessed: Aug. 2, 2022.
[61] X. Du, T.-Y. Lin, P. Jin, G. Ghiasi, M. Tan, Y. Cui, Q. V. Le, and X. Song, "SpineNet: Learning scale-permuted backbone for recognition and localization," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 11589–11598, doi: 10.1109/CVPR42600.2020.01161.
[62] C.-Y. Wang, H.-Y. M. Liao, Y.-H. Wu, P.-Y. Chen, J.-W. Hsieh, and I.-H. Yeh, "CSPNet: A new backbone that can enhance learning capability of CNN," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2020, pp. 1571–1580, doi: 10.1109/CVPRW50498.2020.00203.
[63] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," 2016, *arXiv:1612.03144*. Accessed: Oct. 8, 2019.
[64] G. Ghiasi, T.-Y. Lin, and Q. V. Le, "NAS-FPN: Learning scalable feature pyramid architecture for object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 7029–7038, doi: 10.1109/CVPR.2019.00720.
[65] S. Liu, D. Huang, and Y. Wang, "Learning spatial fusion for single-shot object detection," 2019, *arXiv:1911.09516*.
[66] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path aggregation network for instance segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 8759–8768, doi: 10.1109/CVPR.2018.00913.
[67] M. Tan, R. Pang, and Q. V. Le, "EfficientDet: Scalable and efficient object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 10778–10787, doi: 10.1109/CVPR42600.2020.01079.
[68] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," 2014, *arXiv:1406.4729*.
[69] S. Liu, D. Huang, and Y. Wang, "Receptive field block net for accurate and fast object detection," in *Computer Vision—ECCV 2018*. Berlin, Germany: Springer-Verlag, Sep. 2018, pp. 404–419, doi: 10.1007/978-3-030-01252-6_24.
[70] S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon, "CBAM: Convolutional block attention module," in *Computer Vision—ECCV 2018*, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds. Cham, Switzerland: Springer, 2018, pp. 3–19, doi: 10.1007/978-3-030-01234-2_1.
[71] X. Li, W. Wang, X. Hu, J. Li, J. Tang, and J. Yang, "Generalized focal loss v2: Learning reliable localization quality estimation for dense object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 11627–11636, doi: 10.1109/CVPR46437.2021.01146.
[72] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Las Vegas, NV, USA, Jun. 2016, pp. 779–788, doi: 10.1109/CVPR.2016.91.
[73] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," 2015, *arXiv:1512.02325*.
[74] H. Law and J. Deng, "CornerNet: Detecting objects as paired keypoints," *Int. J. Comput. Vis.*, vol. 128, no. 3, pp. 642–656, Mar. 2020, doi: 10.1007/s11263-019-01204-1.
[75] Z. Tian, C. Shen, H. Chen, and T. He, "FCOS: Fully convolutional one-stage object detection," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 9626–9635, doi: 10.1109/ICCV.2019.00972.
[76] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, Jun. 2017, doi: 10.1109/TPAMI.2016.2577031.
[77] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," 2017, *arXiv:1703.06870*.
[78] Z. Yang, S. Liu, H. Hu, L. Wang, and S. Lin, "RepPoints: Point set representation for object detection," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 9656–9665, doi: 10.1109/ICCV.2019.00975.
[79] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, "Focal loss for dense object detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 2, pp. 318–327, Feb. 2020, doi: 10.1109/TPAMI.2018.2858826.
[80] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," 2016, *arXiv:1612.08242*. Accessed: Oct. 8, 2019.
[81] T. Yang, X. Zhang, Z. Li, W. Zhang, and J. Sun, "MetaAnchor: Learning to detect objects with customized anchors," in *Proc. 32nd Int. Conf. Neural Inf. Process. Syst.* Red Hook, NY, USA: Curran Associates, 2018, pp. 318–328.
[82] J. Wang, K. Chen, S. Yang, C. C. Loy, and D. Lin, "Region proposal by guided anchoring," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 2960–2969, doi: 10.1109/CVPR.2019.00308.
[83] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders, "Selective Search for Object Recognition," *Int. J. Comput. Vis.*, vol. 104, pp. 154–171, Apr. 2013, doi: 10.1007/s11263-013-0620-5.
[84] R. Girshick, "Fast R-CNN," 2015, *arXiv:1504.08083*. Accessed: Oct. 8, 2019.
[85] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," 2013, *arXiv:1311.2524*. Accessed: Oct. 8, 2019.
[86] Z. Cai, Q. Fan, R. S. Feris, and N. Vasconcelos, "A unified multi-scale deep convolutional neural network for fast object detection," in *Computer Vision—ECCV 2016*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Cham, Switzerland: Springer, 2016, pp. 354–370, doi: 10.1007/978-3-319-46493-0_22.
[87] Z. Cai and N. Vasconcelos, "Cascade R-CNN: Delving into high quality object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Salt Lake City, UT, USA, Jun. 2018, pp. 6154–6162, doi: 10.1109/CVPR.2018.00644.
[88] H. Lee, S. Eum, and H. Kwon, "ME R-CNN: Multi-expert R-CNN for object detection," 2017, *arXiv:1704.01069*.
[89] Y. Li, Y. Chen, N. Wang, and Z.-X. Zhang, "Scale-aware trident networks for object detection," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 6053–6062, doi: 10.1109/ICCV.2019.00615.
[90] B. Singh and L. S. Davis, "An analysis of scale invariance in object detection—SNIP," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Salt Lake City, UT, USA, Jun. 2018, pp. 3578–3587, doi: 10.1109/CVPR.2018.00377.
[91] M. Najibi, B. Singh, and L. Davis, "AutoFocus: Efficient multi-scale inference," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 9744–9754, doi: 10.1109/ICCV.2019.00984.

[92] T. Kong, A. Yao, Y. Chen, and F. Sun, "HyperNet: Towards accurate region proposal generation and joint object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Las Vegas, NV, USA, Jun. 2016, pp. 845–853, doi: 10.1109/CVPR.2016.98.

[93] Y. He, C. Zhu, J. Wang, M. Savvides, and X. Zhang, "Bounding box regression with uncertainty for accurate object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 2883–2892, doi: 10.1109/CVPR.2019.00300.

[94] M. Najibi, M. Rastegari, and L. S. Davis, "G-CNN: An iterative grid based object detector," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Las Vegas, NV, USA, Jun. 2016, pp. 2369–2377, doi: 10.1109/CVPR.2016.260.

[95] X. Wang, A. Shrivastava, and A. Gupta, "A-fast-RCNN: Hard positive generation via adversary for object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 3039–3048, doi: 10.1109/CVPR.2017.324.

[96] Z. Tan, X. Nie, Q. Qian, N. Li, and H. Li, "Learning to rank proposals for object detection," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 8272–8280, doi: 10.1109/ICCV.2019.00836.

[97] J. Pang, K. Chen, J. Shi, H. Feng, W. Ouyang, and D. Lin, "Libra R-CNN: Towards balanced learning for object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 821–830, doi: 10.1109/CVPR.2019.00091.

[98] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, vol. 2, New York, NY, USA, Jun. 2006, pp. 2169–2178, doi: 10.1109/CVPR.2006.68.

[99] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Computer Vision—ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham, Switzerland: Springer, 2014, pp. 818–833, doi: 10.1007/978-3-319-10590-1_53.

[100] J. Dai, Y. Li, K. He, and J. Sun, "R-FCN: Object detection via region-based fully convolutional networks," in *Advances in Neural Information Processing Systems*, vol. 29, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Eds. Red Hook, NY, USA: Curran Associates, 2016, pp. 379–387. Accessed: Apr. 10, 2020. [Online]. Available: http://papers.nips.cc/paper/6465-r-fcn-object-detection-via-region-based-fully-convolutional-networks.pdf

[101] G. Ghiasi, Y. Cui, A. Srinivas, R. Qian, T.-Y. Lin, E. D. Cubuk, Q. V. Le, and B. Zoph, "Simple copy-paste is a strong data augmentation method for instance segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 2917–2927, doi: 10.1109/CVPR46437.2021.00294.

[102] S. Zhang, L. Wen, X. Bian, Z. Lei, and S. Z. Li, "Single-shot refinement neural network for object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Salt Lake City, UT, USA, Jun. 2018, pp. 4203–4212, doi: 10.1109/CVPR.2018.00442.

[103] Z. Shen, Z. Liu, J. Li, Y.-G. Jiang, Y. Chen, and X. Xue, "DSOD: Learning deeply supervised object detectors from scratch," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Venice, Italy, Oct. 2017, pp. 1937–1945, doi: 10.1109/ICCV.2017.212.

[104] R. Zhu, S. Zhang, X. Wang, L. Wen, H. Shi, L. Bo, and T. Mei, "ScratchDet: Training single-shot object detectors from scratch," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 2263–2272, doi: 10.1109/CVPR.2019.00237.

[105] C.-Y. Fu, W. Liu, A. Ranga, A. Tyagi, and A. C. Berg, "DSSD: Deconvolutional single shot detector," 2017, *arXiv:1701.06659*. Accessed: Oct. 8, 2019.

[106] T. Kong, F. Sun, A. Yao, H. Liu, M. Lu, and Y. Chen, "RON: Reverse connection with objectness prior networks for object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Honolulu, HI, USA, Jul. 2017, pp. 5244–5252, doi: 10.1109/CVPR.2017.557.

[107] P. Zhou, B. Ni, C. Geng, J. Hu, and Y. Xu, "Scale-transferrable object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Salt Lake City, UT, USA, Jun. 2018, pp. 528–537, doi: 10.1109/CVPR.2018.00062.

[108] Z. Zhang, S. Qiao, C. Xie, W. Shen, B. Wang, and A. L. Yuille, "Single-shot object detection with enriched semantics," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Salt Lake City, UT, USA, Jun. 2018, pp. 5813–5821, doi: 10.1109/CVPR.2018.00609.

[109] T. Wang, R. M. Anwer, H. Cholakkal, F. S. Khan, Y. Pang, and L. Shao, "Learning rich features at high-speed for single-shot object detection," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 1971–1980, doi: 10.1109/ICCV.2019.00206.

[110] J. Nie, R. M. Anwer, H. Cholakkal, F. S. Khan, Y. Pang, and L. Shao, "Enriched feature guided refinement network for object detection," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Seoul, South Korea, Oct. 2019, pp. 9536–9545, doi: 10.1109/ICCV.2019.00963.

[111] S. Li, L. Yang, J. Huang, X.-S. Hua, and L. Zhang, "Dynamic anchor feature selection for single-shot object detection," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 6608–6617, doi: 10.1109/ICCV.2019.00671.

[112] K. Chen, J. Li, W. Lin, J. See, J. Wang, L. Duan, Z. Chen, C. He, and J. Zou, "Towards accurate one-stage object detection with AP-loss," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Long Beach, CA, USA, Jun. 2019, pp. 5114–5122, doi: 10.1109/CVPR.2019.00526.

[113] S.-W. Kim, H.-K. Kook, J.-Y. Sun, M.-C. Kang, and S.-J. Ko, "Parallel feature pyramid network for object detection," in *Computer Vision—ECCV 2018*, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds. Cham, Switzerland: Springer, 2018, pp. 239–256, doi: 10.1007/978-3-030-01228-1_15.

[114] T. Kong, F. Sun, W. Huang, and H. Liu, "Deep feature pyramid reconfiguration for object detection," in *Computer Vision—ECCV 2018*, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds. Cham, Switzerland: Springer, 2018, pp. 172–188, doi: 10.1007/978-3-030-01228-1_11.

[115] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Miami, FL, USA, Jun. 2009, pp. 248–255, doi: 10.1109/CVPR.2009.5206848.

[116] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," 2018, *arXiv:1804.02767*.

[117] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. 27th Int. Conf. Mach. Learn.*, Madison, WI, USA, Jun. 2010, pp. 807–814.

[118] C. Peng, T. Xiao, Z. Li, Y. Jiang, X. Zhang, K. Jia, G. Yu, and J. Sun, "MegDet: A large mini-batch object detector," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 6181–6189, doi: 10.1109/CVPR.2018.00647.

[119] K. Kim and H. S. Lee, "Probabilistic anchor assignment with IoU prediction for object detection," in *Computer Vision—ECCV 2020*, A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, Eds. Cham, Switzerland: Springer, 2020, pp. 355–371, doi: 10.1007/978-3-030-58595-2_22.

[120] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Boston, MA, USA, Jun. 2015, pp. 1–9, doi: 10.1109/CVPR.2015.7298594.

[121] X. Zhou, J. Zhuo, and P. Krähenbühl, "Bottom-up object detection by grouping extreme and center points," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 850–859, doi: 10.1109/CVPR.2019.00094.

[122] C. Zhu, Y. He, and M. Savvides, "Feature selective anchor-free module for single-shot object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 840–849, doi: 10.1109/CVPR.2019.00093.

[123] S. Zhang, C. Chi, Y. Yao, Z. Lei, and S. Z. Li, "Bridging the gap between anchor-based and anchor-free detection via adaptive training sample selection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 9756–9765, doi: 10.1109/CVPR42600.2020.00978.

[124] Z. Ge, S. Liu, Z. Li, O. Yoshie, and J. Sun, "OTA: Optimal transport assignment for object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 303–312, doi: 10.1109/CVPR46437.2021.00037.

[125] P. A. Knight, "The Sinkhorn–Knopp algorithm: Convergence and applications," *SIAM J. Matrix Anal. Appl.*, vol. 30, no. 1, pp. 261–275, Jan. 2008, doi: 10.1137/060659624.

[126] H. Su, Y. He, R. Jiang, J. Zhang, W. Zou, and B. Fan, "DSLA: Dynamic smooth label assignment for efficient anchor-free object detection," *Pattern Recognit.*, vol. 131, Nov. 2022, Art. no. 108868, doi: 10.1016/j.patcog.2022.108868.

[127] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16 × 16 words: Transformers for image recognition at scale," 2020, *arXiv:2010.11929*.
[128] S. Rao, Y. Li, R. Ramakrishnan, A. Hassaine, D. Canoy, J. Cleland, T. Lukasiewicz, G. Salimi-Khorshidi, and K. Rahimi, "An explainable transformer-based deep learning model for the prediction of incident heart failure," *IEEE J. Biomed. Health Informat.*, vol. 26, no. 7, pp. 3362–3372, Jul. 2022, doi: 10.1109/JBHI.2022.3148820.
[129] A. B. Amjoud and M. Amrouch, "Automatic generation of chest X-ray reports using a transformer-based deep learning model," in *Proc. 5th Int. Conf. Intell. Comput. Data Sci. (ICDS)*, Oct. 2021, pp. 1–5, doi: 10.1109/ICDS53782.2021.9626725.
[130] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," 2020, *arXiv:2005.12872*.
[131] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017, *arXiv:1706.03762*. Accessed: Mar. 4, 2021.
[132] P. Gao, M. Zheng, X. Wang, J. Dai, and H. Li, "Fast convergence of DETR with spatially modulated co-attention," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 3601–3610, doi: 10.1109/ICCV48922.2021.00360.
[133] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted Windows," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 9992–10002, doi: 10.1109/ICCV48922.2021.00986.
[134] Y. Wang, X. Zhang, T. Yang, and J. Sun, "Anchor DETR: Query design for transformer-based object detection," 2021, *arXiv:2109.07107*.
[135] L. He and S. Todorovic, "DESTR: Object detection with split transformer," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 9367–9376, doi: 10.1109/CVPR52688.2022.00916.
[136] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Region-based convolutional networks for accurate object detection and segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 1, pp. 142–158, Jan. 2016, doi: 10.1109/TPAMI.2015.2437384.
[137] S. Ren, K. He, R. Girshick, X. Zhang, and J. Sun, "Object detection networks on convolutional feature maps," 2015, *arXiv:1504.06066*. Accessed: Oct. 8, 2019.
[138] A. Shrivastava, A. Gupta, and R. Girshick, "Training region-based object detectors with online hard example mining," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Las Vegas, NV, USA, Jun. 2016, pp. 761–769, doi: 10.1109/CVPR.2016.89.
[139] Y. Liu, R. Wang, S. Shan, and X. Chen, "Structure inference net: Object detection using scene-level context and instance-level relationships," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 6985–6994, doi: 10.1109/CVPR.2018.00730.
[140] S. Bell, C. L. Zitnick, K. Bala, and R. Girshick, "Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Las Vegas, NV, USA, Jun. 2016, pp. 2874–2883, doi: 10.1109/CVPR.2016.314.
[141] S. Gidaris and N. Komodakis, "LocNet: Improving localization accuracy for object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Las Vegas, NV, USA, Jun. 2016, pp. 789–798, doi: 10.1109/CVPR.2016.92.
[142] S. Gidaris and N. Komodakis, "Object detection via a multi-region & semantic segmentation-aware CNN model," 2015, *arXiv:1505.01749*. Accessed: Oct. 8, 2019.
[143] J. Jeong, H. Park, and N. Kwak, "Enhancement of SSD by concatenating feature maps for object detection," 2017, *arXiv:1705.09587*. Accessed: Oct. 8, 2019.
[144] S. Woo, S. Hwang, and I. S. Kweon, "StairNet: Top-down semantic aggregation for accurate one shot detection," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Lake Tahoe, NV, USA, Mar. 2018, pp. 1093–1102, doi: 10.1109/WACV.2018.00125.
[145] Z. Li and F. Zhou, "FSSD: Feature fusion single shot multibox detector," 2017, *arXiv:1712.00960*. Accessed: Oct. 8, 2019.
[146] N. Dvornik, K. Shmelkov, J. Mairal, and C. Schmid, "BlitzNet: A real-time deep network for scene understanding," 2017, *arXiv:1708.02813*. Accessed: Oct. 8, 2019.
[147] K. Lee, J. Choi, J. Jeong, and N. Kwak, "Residual features and unified prediction network for single stage detection," 2017, *arXiv:1707.05031*.
[148] L. Zheng, C. Fu, and Y. Zhao, "Extend the shallow part of single shot multibox detector via convolutional neural network," 2018, *arXiv:1801.05918*.
[149] Y. Chen, J. Li, B. Zhou, J. Feng, and S. Yan, "Weaving multi-scale context for single shot detector," 2017, *arXiv:1712.03149*.
[150] X. Wu, D. Zhang, J. Zhu, and S. C. H. Hoi, "Single-shot bidirectional pyramid networks for high-quality object detection," 2018, *arXiv:1803.08208*. Accessed: Oct. 8, 2019.
[151] Y. Pang, T. Wang, R. M. Anwer, F. S. Khan, and L. Shao, "Efficient featurized image pyramid network for single shot detector," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 7328–7336, doi: 10.1109/CVPR.2019.00751.
[152] K. Song, H. Yang, and Z. Yin, "Multi-scale attention deep neural network for fast accurate object detection," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 29, no. 10, pp. 2972–2985, Oct. 2019, doi: 10.1109/TCSVT.2018.2875449.
[153] J. Cao, Y. Pang, J. Han, and X. Li, "Hierarchical shot detector," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 9704–9713, doi: 10.1109/ICCV.2019.00980.
[154] Y. Zhu, C. Zhao, J. Wang, X. Zhao, Y. Wu, and H. Lu, "CoupleNet: Coupling global structure with local parts for object detection," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Venice, Italy, Oct. 2017, pp. 4146–4154, doi: 10.1109/ICCV.2017.444.
[155] J. Cao, Y. Pang, and X. Li, "Triply supervised decoder networks for joint detection and segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 7384–7393, doi: 10.1109/CVPR.2019.00757.
[156] Y. Zhu, C. Zhao, H. Guo, J. Wang, X. Zhao, and H. Lu, "Attention CoupleNet: Fully convolutional attention coupling network for object detection," *IEEE Trans. Image Process.*, vol. 28, no. 1, pp. 113–126, Jan. 2019, doi: 10.1109/TIP.2018.2865280.
[157] B. Cheng, Y. Wei, H. Shi, R. Feris, J. Xiong, and T. Huang, "Revisiting RCNN: On awakening the classification power of faster RCNN," in *Computer Vision—ECCV 2018*. Berlin, Germany: Springer-Verlag, Sep. 2018, pp. 473–490, doi: 10.1007/978-3-030-01267-0_28.
[158] B. Singh, M. Najibi, and L. S. Davis, "SNIPER: Efficient multi-scale training," in *Proc. 32nd Int. Conf. Neural Inf. Process. Syst.* Red Hook, NY, USA: Curran Associates, 2018, pp. 9333–9343.
[159] W. Xiang, D.-Q. Zhang, H. Yu, and V. Athitsos, "Context-aware single-shot detector," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Mar. 2018, pp. 1784–1793, doi: 10.1109/WACV.2018.00198.
[160] H. Wang, Q. Wang, M. Gao, P. Li, and W. Zuo, "Multi-scale location-aware kernel representation for object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 1248–1257, doi: 10.1109/CVPR.2018.00136.
[161] D. Bolya, C. Zhou, F. Xiao, and Y. J. Lee, "YOLACT: Real-time instance segmentation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 9156–9165, doi: 10.1109/ICCV.2019.00925.
[162] Y. Xin, S. Wang, L. Li, W. Zhang, and Q. Huang, "Reverse densely connected feature pyramid network for object detection," in *Computer Vision—ACCV 2018*, C. V. Jawahar, H. Li, G. Mori, and K. Schindler, Eds. Cham, Switzerland: Springer, 2019, pp. 530–545, doi: 10.1007/978-3-030-20873-8_34.
[163] X. Chen and A. Gupta, "Spatial memory for context reasoning in object detection," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 4106–4116, doi: 10.1109/ICCV.2017.440.
[164] S. Zagoruyko, A. Lerer, T.-Y. Lin, P. O. Pinheiro, S. Gross, S. Chintala, and P. Dollár, "A multipath network for object detection," 2016, *arXiv:1604.02135*.
[165] Q. Zhao, T. Sheng, Y. Wang, Z. Tang, Y. Chen, L. Cai, and H. Ling, "M2Det: A single-shot object detector based on multi-level feature pyramid network," in *Proc. 33rd AAAI Conf. Artif. Intell. 31st Innov. Appl. Artif. Intell. Conf., 9th AAAI Symp. Educ. Adv. Artif. Intell.* Honolulu, HI, USA: AAAI Press, 2019, pp. 9259–9266, doi: 10.1609/aaai.v33i01.33019259.
[166] L. Tychsen-Smith and L. Petersson, "DeNet: Scalable real-time object detection with directed sparse sampling," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Venice, Italy, Oct. 2017, pp. 428–436, doi: 10.1109/ICCV.2017.54.
[167] S. Wang, Y. Gong, J. Xing, L. Huang, C. Huang, and W. Hu, "RDSNet: A new deep architecture for reciprocal object detection and instance segmentation," 2019, *arXiv:1912.05070*.

[168] X. Chen, R. Girshick, K. He, and P. Dollar, "TensorMask: A foundation for dense object segmentation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 2061–2069, doi: 10.1109/ICCV.2019.00215.

[169] A. Shrivastava, R. Sukthankar, J. Malik, and A. Gupta, "Beyond skip connections: Top-down modulation for object detection," 2016, *arXiv:1612.06851*. Accessed: Oct. 8, 2019.

[170] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei, "Deformable convolutional networks," 2017, *arXiv:1703.06211*. Accessed: Oct. 8, 2019.

[171] H. Hu, J. Gu, Z. Zhang, J. Dai, and Y. Wei, "Relation networks for object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 3588–3597, doi: 10.1109/CVPR.2018.00378.

[172] H. Xu, X. Lv, X. Wang, Z. Ren, N. Bodla, and R. Chellappa, "Deep regionlets for object detection," in *Computer Vision—ECCV 2018*, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds. Cham, Switzerland: Springer, 2018, pp. 827–844, doi: 10.1007/978-3-030-01252-6_49.

[173] J. Gu, H. Hu, L. Wang, Y. Wei, and J. Dai, "Learning region features for object detection," in *Computer Vision—ECCV 2018*, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds. Cham, Switzerland: Springer, 2018, pp. 392–406, doi: 10.1007/978-3-030-01258-8_24.

[174] T. Kong, F. Sun, H. Liu, Y. Jiang, and J. Shi, "Consistent optimization for single-shot object detection," 2019, *arXiv:1901.06563*.

[175] Z. Li, C. Peng, G. Yu, X. Zhang, Y. Deng, and J. Sun, "DetNet: Design backbone for object detection," in *Computer Vision—ECCV 2018*, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds. Cham, Switzerland: Springer, 2018, pp. 339–354, doi: 10.1007/978-3-030-01240-3_21.

[176] B. Jiang, R. Luo, J. Mao, T. Xiao, and Y. Jiang, "Acquisition of localization confidence for accurate object detection," in *Computer Vision—ECCV 2018*. Berlin, Germany: Springer-Verlag, Sep. 2018, pp. 816–832, doi: 10.1007/978-3-030-01264-9_48.

[177] Y. Lee and J. Park, "CenterMask: Real-time anchor-free instance segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 13903–13912, doi: 10.1109/CVPR42600.2020.01392.

[178] N. Bodla, B. Singh, R. Chellappa, and L. S. Davis, "Soft-NMS—Improving object detection with one line of code," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 5562–5570, doi: 10.1109/ICCV.2017.593.

[179] H. Zhang, H. Chang, B. Ma, S. Shan, and X. Chen, "Cascade RetinaNet: Maintaining consistency for single-stage object detection," 2019, *arXiv:1907.06881*.

[180] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal speed and accuracy of object detection," 2020, *arXiv:2004.10934*.

[181] Y. Bai, Y. Zhang, M. Ding, and B. Ghanem, "SOD-MTGAN: Small object detection via multi-task generative adversarial network," in *Computer Vision—ECCV 2018*, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds. Cham, Switzerland: Springer, 2018, pp. 210–226, doi: 10.1007/978-3-030-01261-8_13.

[182] Z. Li, C. Peng, G. Yu, X. Zhang, Y. Deng, and J. Sun, "Light-head R-CNN: In defense of two-stage object detector," 2017, *arXiv:1711.07264*.

[183] B. Li, Y. Liu, and X. Wang, "Gradient harmonized single-stage detector," in *Proc. 33rd AAAI Conf. Artif. Intell. 31st Innov. Appl. Artif. Intell. Conf. 9th AAAI Symp. Educ. Adv. Artif. Intell.* Honolulu, HI, USA: AAAI Press, 2019, pp. 8577–8584, doi: 10.1609/aaai.v33i01.33018577.

[184] J. Peng, M. Sun, Z.-X. Zhang, T. Tan, and J. Yan, "Efficient neural architecture transformation search in channel-level for object detection," in *Proc. 33rd Int. Conf. Neural Inf. Process. Syst.* Red Hook, NY, USA: Curran Associates, 2019, pp. 14335–14344.

[185] L. Tychsen-Smith and L. Petersson, "Improving object localization with fitness NMS and bounded IoU loss," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Salt Lake City, UT, USA, Jun. 2018, pp. 6877–6885, doi: 10.1109/CVPR.2018.00719.

[186] Z. Chen, S. Huang, and D. Tao, "Context refinement for object detection," in *Computer Vision—ECCV 2018*, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds. Cham, Switzerland: Springer, 2018, pp. 74–89, doi: 10.1007/978-3-030-01237-3_5.

[187] T. Kong, F. Sun, H. Liu, Y. Jiang, L. Li, and J. Shi, "FoveaBox: Beyound anchor-based object detection," *IEEE Trans. Image Process.*, vol. 29, pp. 7389–7398, 2020, doi: 10.1109/TIP.2020.3002345.

[188] C.-Y. Fu, M. Shvets, and A. C. Berg, "RetinaMask: Learning to predict masks improves state-of-the-art single-shot detection for free," 2019, *arXiv:1901.03353*.

[189] X. Li, W. Wang, L. Wu, S. Chen, X. Hu, J. Li, J. Tang, and J. Yang, "Generalized focal loss: Learning qualified and distributed bounding boxes for dense object detection," in *Proc. 34th Int. Conf. Neural Inf. Process. Syst.* Red Hook, NY, USA: Curran Associates, 2020, pp. 21002–21012.

[190] X. Zhang, F. Wan, C. Liu, R. Ji, and Q. Ye, "FreeAnchor: Learning to match anchors for visual object detection," in *Proc. Adv. Neural Inf. Process. Syst.* Red Hook, NY, USA: Curran Associates, 2019, pp. 1–9. Accessed: Oct. 21, 2019.

[191] G. Song, Y. Liu, and X. Wang, "Revisiting the sibling head in object detector," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 11560–11569, doi: 10.1109/CVPR42600.2020.01158.

[192] H. Law, Y. Teng, O. Russakovsky, and J. Deng, "CornerNet-lite: Efficient keypoint based object detection," 2019, *arXiv:1904.08900*.

[193] J. Wang, W. Zhang, Y. Cao, K. Chen, J. Pang, T. Gong, J. Shi, C. C. Loy, and D. Lin, "Side-aware boundary localization for more precise object detection," in *Computer Vision—ECCV 2020*, A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, Eds. Cham, Switzerland: Springer, 2020, pp. 403–419, doi: 10.1007/978-3-030-58548-8_24.

[194] W. Ke, T. Zhang, Z. Huang, Q. Ye, J. Liu, and D. Huang, "Multiple anchor learning for visual object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 10203–10212, doi: 10.1109/CVPR42600.2020.01022.

[195] H. Li, Z. Wu, C. Zhu, C. Xiong, R. Socher, and L. S. Davis, "Learning from noisy anchors for one-stage object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 10585–10594, doi: 10.1109/CVPR42600.2020.01060.

[196] Z. Sun, S. Cao, Y. Yang, and K. Kitani, "Rethinking transformer-based set prediction for object detection," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 3591–3600, doi: 10.1109/ICCV48922.2021.00359.

[197] K. Duan, S. Bai, L. Xie, H. Qi, Q. Huang, and Q. Tian, "CenterNet: Keypoint triplets for object detection," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 6568–6577, doi: 10.1109/ICCV.2019.00667.

[198] X. Long, K. Deng, G. Wang, Y. Zhang, Q. Dang, Y. Gao, H. Shen, J. Ren, S. Han, E. Ding, and S. Wen, "PP-YOLO: An effective and efficient implementation of object detector," 2020, *arXiv:2007.12099*.

[199] C.-Y. Wang, A. Bochkovskiy, and H.-Y.-M. Liao, "Scaled-YOLOv4: Scaling cross stage partial network," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 13024–13033, doi: 10.1109/CVPR46437.2021.01283.

[200] L. Yao, H. Xu, W. Zhang, X. Liang, and Z. Li, "SM-NAS: Structural-to-modular neural architecture search for object detection," 2019, *arXiv:1911.09929*.

[201] D. Meng, X. Chen, Z. Fan, G. Zeng, H. Li, Y. Yuan, L. Sun, and J. Wang, "Conditional DETR for fast training convergence," 2021, *arXiv:2108.06152*.

[202] C. Zhu, F. Chen, Z. Shen, and M. Savvides, "Soft anchor-point object detection," in *Computer Vision—ECCV 2020*, A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, Eds. Cham, Switzerland: Springer, 2020, pp. 91–107, doi: 10.1007/978-3-030-58545-7_6.

[203] Z. Dong, G. Li, Y. Liao, F. Wang, P. Ren, and C. Qian, "CentripetalNet: Pursuing high-quality keypoint pairs for object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 10516–10525, doi: 10.1109/CVPR42600.2020.01053.

[204] B. Zhu, J. Wang, Z. Jiang, F. Zong, S. Liu, Z. Li, and J. Sun, "AutoAssign: Differentiable label assignment for dense object detection," 2020, *arXiv:2007.03496*.

[205] H. Zhang, H. Chang, B. Ma, N. Wang, and X. Chen, "Dynamic R-CNN: Towards high quality object detection via dynamic training," in *Computer Vision—ECCV 2020*, A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, Eds. Cham, Switzerland: Springer, 2020, pp. 260–275, doi: 10.1007/978-3-030-58555-6_16.

[206] K. Chen, W. Ouyang, C. C. Loy, D. Lin, J. Pang, J. Wang, Y. Xiong, X. Li, S. Sun, W. Feng, Z. Liu, and J. Shi, "Hybrid task cascade for instance segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 4969–4978, doi: 10.1109/CVPR.2019.00511.

[207] H. Qiu, Y. Ma, Z. Li, S. Liu, and J. Sun, "BorderDet: Border feature for dense object detection," in *Computer Vision—ECCV 2020*. Berlin, Germany: Springer-Verlag, Aug. 2020, pp. 549–564, doi: 10.1007/978-3-030-58452-8_32.

[208] X. Zhu, W. Su, L. Lu, B. Li, X. Wang, and J. Dai, "Deformable DETR: Deformable transformers for end-to-end object detection," 2020, *arXiv:2010.04159*.

[209] C.-Y. Wang, I.-H. Yeh, and H.-Y. M. Liao, "You only learn one representation: Unified network for multiple tasks," 2021, *arXiv:2105.04206*.
[210] X. Dai, Y. Chen, B. Xiao, D. Chen, M. Liu, L. Yuan, and L. Zhang, "Dynamic head: Unifying object detection heads with attentions," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 7369–7378, doi: 10.1109/CVPR46437.2021.00729.
[211] M. Xu, Z. Zhang, H. Hu, J. Wang, L. Wang, F. Wei, X. Bai, and Z. Liu, "End-to-end semi-supervised object detection with soft teacher," 2021, *arXiv:2106.09018*.
[212] Z. Liu, H. Hu, Y. Lin, Z. Yao, Z. Xie, Y. Wei, J. Ning, Y. Cao, Z. Zhang, L. Dong, F. Wei, and B. Guo, "Swin transformer v2: Scaling up capacity and resolution," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 11999–12009, doi: 10.1109/CVPR52688.2022.01170.
[213] X. Chen, J. Yu, S. Kong, Z. Wu, and L. Wen, "Joint anchor-feature refinement for real-time accurate object detection in images and videos," 2018, *arXiv:1807.08638*.
[214] Z. Wang, J. Guo, C. Zhang, and B. Wang, "Multiscale feature enhancement network for salient object detection in optical remote sensing images," *IEEE Trans. Geosci. Remote Sens.*, vol. 60, 2022, Art. no. 5634819, doi: 10.1109/TGRS.2022.3224815.
[215] A. Shafique, G. Cao, Z. Khan, M. Asad, and M. Aslam, "Deep learning-based change detection in remote sensing images: A review," *Remote Sens.*, vol. 14, no. 4, p. 871, Feb. 2022.
[216] S. Shokirov, T. Jucker, S. R. Levick, A. D. Manning, T. Bonnet, M. Yebra, and K. N. Youngentob, "Habitat highs and lows: Using terrestrial and UAV LiDAR for modelling avian species richness and abundance in a restored woodland," *Remote Sens. Environ.*, vol. 285, Feb. 2023, Art. no. 113326.
[217] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 2, no. 1, pp. 41–50, Feb. 2018, doi: 10.1109/TETCI.2017.2772792.
[218] S. Majchrowska, A. Mikołajczyk, M. Ferlin, Z. Klawikowska, M. A. Plantykow, A. Kwasigroch, and K. Majek, "Deep learning-based waste detection in natural and urban environments," *Waste Manage.*, vol. 138, pp. 274–284, Feb. 2022, doi: 10.1016/j.wasman.2021.12.001.
[219] A. Bhattacharyya, D. Bhaik, S. Kumar, P. Thakur, R. Sharma, and R. B. Pachori, "A deep learning based approach for automatic detection of COVID-19 cases using chest X-ray images," *Biomed. Signal Process. Control*, vol. 71, Jan. 2022, Art. no. 103182, doi: 10.1016/j.bspc.2021.103182.
[220] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015, doi: 10.1007/s11263-015-0816-y.
[221] A. Kuznetsova, H. Rom, N. Alldrin, J. Uijlings, I. Krasin, J. Pont-Tuset, S. Kamali, S. Popov, M. Malloci, A. Kolesnikov, T. Duerig, and V. Ferrari, "The open images dataset V4," *Int. J. Comput. Vis.*, vol. 128, no. 7, pp. 1956–1981, Jul. 2020, doi: 10.1007/s11263-020-01316-z.
[222] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang, "Random erasing data augmentation," in *Proc. AAAI*, Apr. 2020, vol. 34, no. 7, pp. 13001–13008, doi: 10.1609/aaai.v34i07.7000.
[223] X. Zhu, H. Hu, S. Lin, and J. Dai, "Deformable ConvNets v2: More deformable, better results," 2018, *arXiv:1811.11168*.
[224] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2818–2826, doi: 10.1109/CVPR.2016.308.
[225] S. Qiao, L.-C. Chen, and A. Yuille, "DetectoRS: Detecting objects with recursive feature pyramid and switchable atrous convolution," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 10208–10219, doi: 10.1109/CVPR46437.2021.01008.
[226] T. Liang, X. Chu, Y. Liu, Y. Wang, Z. Tang, W. Chu, J. Chen, and H. Ling, "CBNet: A composite backbone network architecture for object detection," *IEEE Trans. Image Process.*, vol. 31, pp. 6893–6906, 2022, doi: 10.1109/TIP.2022.3216771.
[227] X. Wang, S. Zhang, Z. Yu, L. Feng, and W. Zhang, "Scale-equalizing pyramid convolution for object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 13356–13365, doi: 10.1109/CVPR42600.2020.01337.
[228] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-ResNet and the impact of residual connections on learning," in *Proc. 31st AAAI Conf. Artif. Intell.*, San Francisco, CA, USA, Feb. 2017, pp. 4278–4284.
[229] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Honolulu, HI, USA, Jul. 2017, pp. 2261–2269, doi: 10.1109/CVPR.2017.243.
[230] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 7132–7141, doi: 10.1109/CVPR.2018.00745.
[231] A. Newell, K. Yang, and J. Deng, "Stacked hourglass networks for human pose estimation," in *Computer Vision—ECCV 2016*, vol. 9912, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Cham, Switzerland: Springer, 2016, pp. 483–499, doi: 10.1007/978-3-319-46484-8_29.
[232] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, "A ConvNet for the 2020s," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 11966–11976, doi: 10.1109/CVPR52688.2022.01167.
[233] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," 2022, *arXiv:2207.02696*.

**AYOUB BENALI AMJOUD** received the engineering degree in information systems and computer networks from the Faculty of Science and Technology of Marrakech, Cadi Ayyad University, in 2017. He is currently pursuing the Ph.D. degree with Image et Reconnaissance de Forme-Systèmes Intelligents et Communicants (IRF-SIC) Laboratory, Faculty of Sciences of Agadir, Ibn Zohr University. He worked with national and international organizations and held research and development positions. His research interests include deep learning in computer vision, pattern recognition, and image captioning. He received the Grant of Excellence from the National Centre for Scientific and Technical Research of Morocco. This grant is awarded to the most outstanding student researchers in Morocco.

**MUSTAPHA AMROUCH** received the master's degree in mathematics and applied computer science and the Ph.D. degree in computer vision from the Faculty of Sciences, Ibn Zohr University, Agadir, Morocco, in 2007 and 2012, respectively. He is currently a Professor and a Researcher with Image et Reconnaissance de Forme-Systèmes Intelligents et Communicants (IRF-SIC) Laboratory, Ibn Zohr University. His research interests include understanding the principles behind machine learning and computer vision, and improving and applying their algorithms to build real applications.