v7labs.com

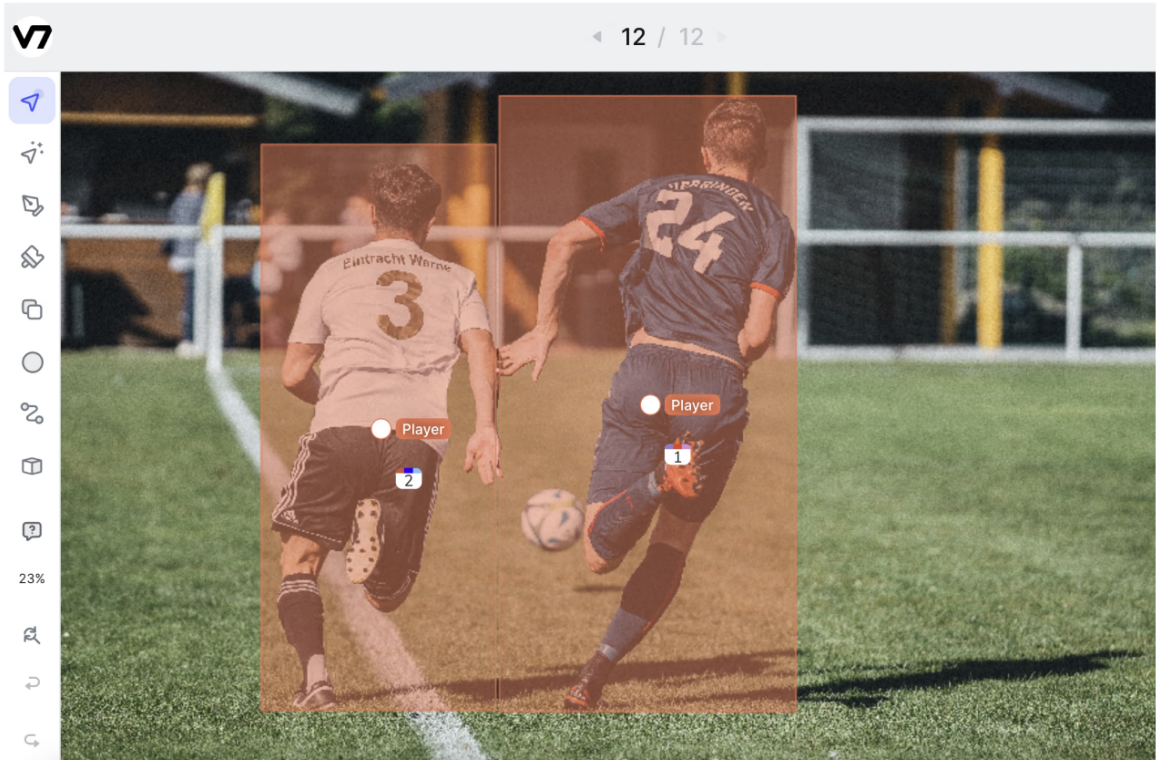
YOLO Algorithm for Object Detection Explained [+Examples]

18–22 minutes

Ready to streamline AI product deployment right away? Check out:

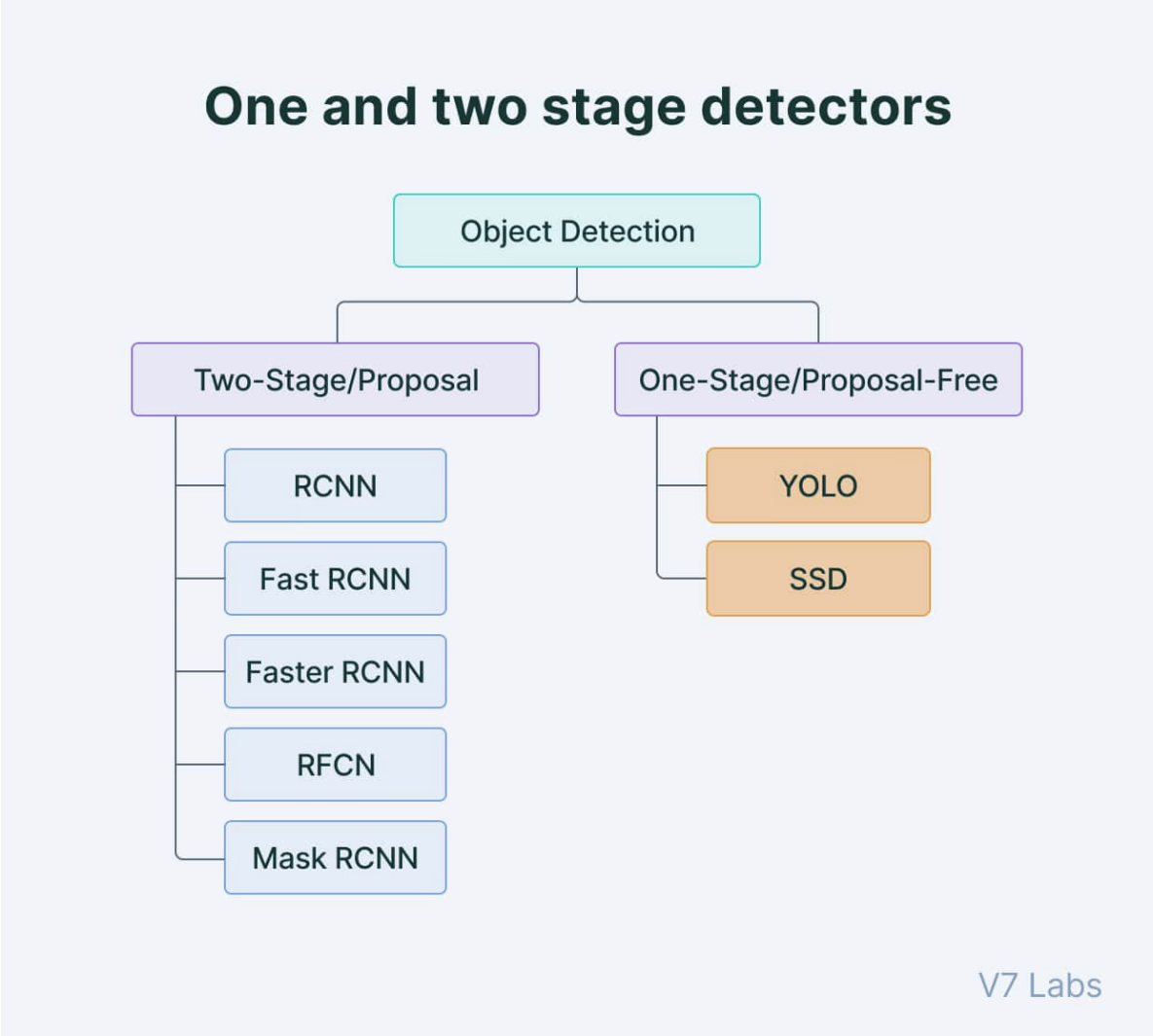
What is object detection?

Object detection is a computer vision task that involves identifying and locating objects in images or videos. It is an important part of many applications, such as surveillance, self-driving cars, or robotics. Object detection algorithms can be divided into two main categories: single-shot detectors and two-stage detectors.



One of the earliest successful attempts to address the object detection problem using deep learning was the R-CNN (Regions with CNN features) model, developed by Ross Girshick and his team at Microsoft Research in 2014. This model used a combination of region proposal algorithms and convolutional neural networks (CNNs) to detect and localize objects in images.

Object detection algorithms are broadly classified into two categories based on how many times the same input image is passed through a network.



Single-shot object detection

Single-shot object detection uses a single pass of the input image to make predictions about the presence and location of objects in the image. It processes an entire image in a single pass, making them computationally efficient.

However, single-shot object detection is generally less accurate than other methods, and it's less effective in detecting small objects. Such algorithms can be used to detect objects in real time in resource-constrained environments.

YOLO is a single-shot detector that uses a fully convolutional neural network (CNN) to process an image. We will dive deeper into the YOLO model in the next section.

Two-shot object detection

Two-shot object detection uses two passes of the input image to make predictions about the presence and location of objects. The first pass is used to generate a set of proposals or potential object locations, and the second pass is used to refine these proposals and make final predictions. This approach is more accurate than single-shot object detection but is also more computationally expensive.

Overall, the choice between single-shot and two-shot object detection depends on the specific requirements and constraints of the application.

Generally, single-shot object detection is better suited for real-time applications, while two-shot object detection is better for applications where accuracy is more important.

Object detection models performance evaluation metrics

To determine and compare the predictive performance of different object detection models, we need standard quantitative metrics.

The two most common evaluation metrics are Intersection over Union (IoU) and the Average Precision (AP) metrics.

Intersection over Union (IoU)

Intersection over Union is a popular metric to measure localization accuracy and calculate localization errors in object detection models.

To calculate the IoU between the predicted and the ground truth bounding boxes, we first take the intersecting area between the two corresponding bounding boxes for the same object. Following this, we calculate the total area covered by the two bounding boxes— also known as the “Union” and the area of overlap between them called the “Intersection.”

The intersection divided by the Union gives us the ratio of the overlap to the total area, providing a good estimate of how close the prediction bounding box is to the original bounding box.

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

> Pro tip: Would you like to start annotating with bounding boxes? Check out 9 Essential Features for a Bounding Box Annotation Tool.

Average Precision (AP)

Average Precision (AP) is calculated as the area under a precision vs. recall curve for a set of predictions.

Recall is calculated as the ratio of the total predictions made by the model under a class with a total of existing labels for the class. Precision refers to the ratio of true positives with respect to the total predictions made by the model.

Recall and precision offer a trade-off that is graphically represented into a curve by varying the classification threshold. The area under this precision vs. recall curve gives us the Average Precision per class for the model. The average of this value, taken over all classes, is called mean Average Precision (mAP).

> Read more: Mean Average Precision (mAP) Explained: Everything You Need to Know

In object detection, precision and recall aren't used for class predictions. Instead, they serve as predictions of boundary boxes for measuring the decision performance. An IoU value > 0.5. is taken as a positive prediction, while an IoU value < 0.5 is a negative prediction.

What is YOLO?

You Only Look Once (YOLO) proposes using an end-to-end neural network that makes predictions of bounding boxes and class probabilities all at once. It differs from the approach taken by previous object detection algorithms, which repurposed classifiers to perform

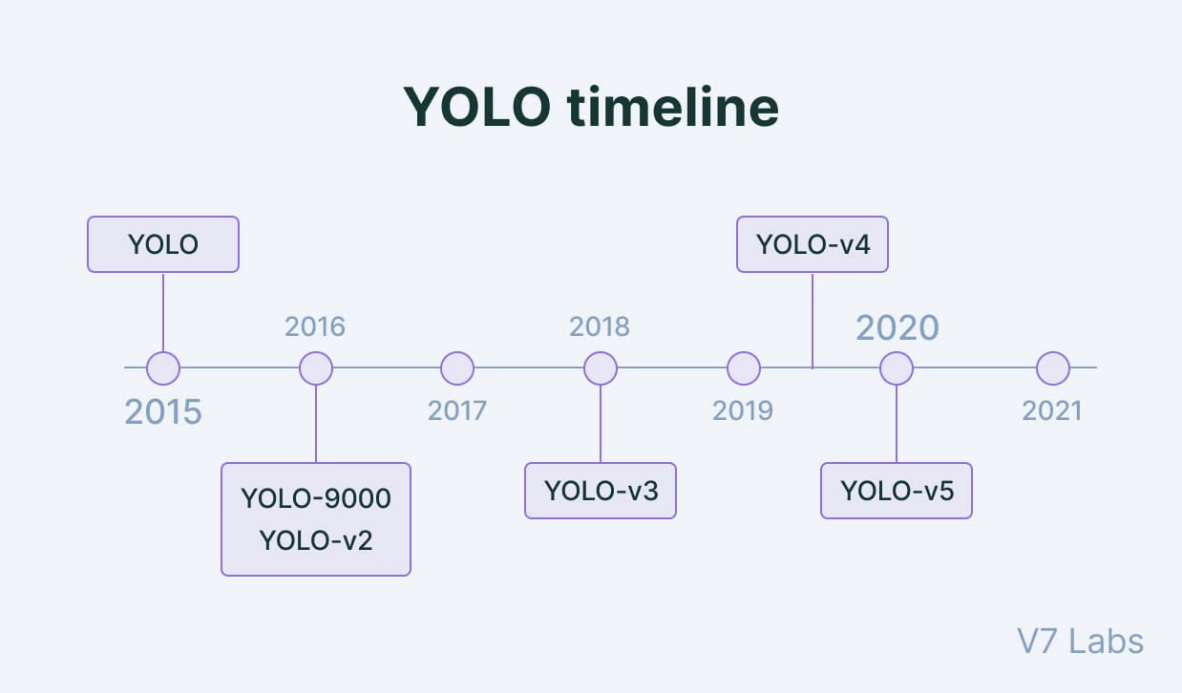
detection.

Following a fundamentally different approach to object detection, YOLO achieved state-of-the-art results, beating other real-time object detection algorithms by a large margin.

While algorithms like Faster RCNN work by detecting possible regions of interest using the Region Proposal Network and then performing recognition on those regions separately, YOLO performs all of its predictions with the help of a single fully connected layer.

Methods that use Region Proposal Networks perform multiple iterations for the same image, while YOLO gets away with a single iteration.

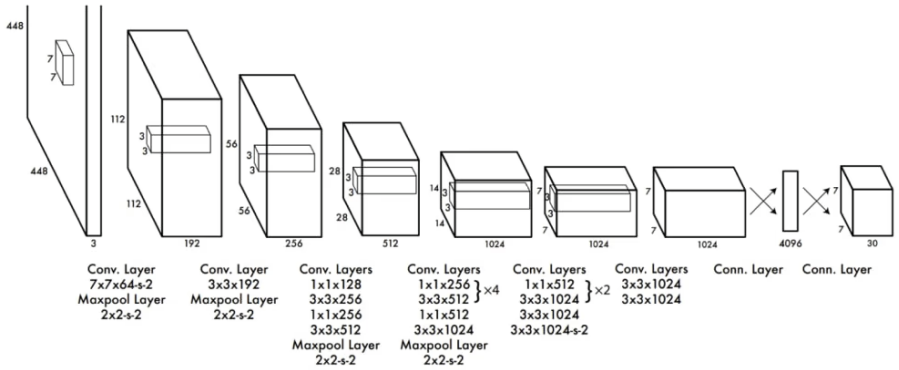
Several new versions of the same model have been proposed since the initial release of YOLO in 2015, each building on and improving its predecessor. Here's a timeline showcasing YOLO's development in recent years.



How does YOLO work? YOLO Architecture

The YOLO algorithm takes an image as input and then uses a simple

deep convolutional neural network to detect objects in the image. The architecture of the CNN model that forms the backbone of YOLO is shown below.



The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224 input image) and then double the resolution for detection.

The first 20 convolution layers of the model are pre-trained using ImageNet by plugging in a temporary average pooling and fully connected layer. Then, this pre-trained model is converted to perform detection since previous research showcased that adding convolution and connected layers to a pre-trained network improves performance. YOLO's final fully connected layer predicts both class probabilities and bounding box coordinates.

YOLO divides an input image into an S × S grid. If the center of an object falls into a grid cell, that grid cell is responsible for detecting that object. Each grid cell predicts B bounding boxes and confidence scores for those boxes. These confidence scores reflect how confident the model is that the box contains an object and how accurate it thinks the predicted box is.

YOLO predicts multiple bounding boxes per grid cell. At training time, we only want one bounding box predictor to be responsible for each object. YOLO assigns one predictor to be "responsible" for predicting an object based on which prediction has the highest current IOU with the ground

truth. This leads to specialization between the bounding box predictors. Each predictor gets better at forecasting certain sizes, aspect ratios, or classes of objects, improving the overall recall score.

One key technique used in the YOLO models is **non-maximum suppression (NMS)**. NMS is a post-processing step that is used to improve the accuracy and efficiency of object detection. In object detection, it is common for multiple bounding boxes to be generated for a single object in an image. These bounding boxes may overlap or be located at different positions, but they all represent the same object. NMS is used to identify and remove redundant or incorrect bounding boxes and to output a single bounding box for each object in the image.

Now, let us look into the improvements that the later versions of YOLO have brought to the parent model.

> Pro tip: Take a look at this list of 65+ Best Free Datasets for Machine Learning to find relevant data for training your models.

YOLO v2

YOLO v2, also known as YOLO9000, was introduced in 2016 as an improvement over the original YOLO algorithm. It was designed to be faster and more accurate than YOLO and to be able to detect a wider range of object classes. This updated version also uses a different CNN backbone called Darknet-19, a variant of the VGGNet architecture with simple progressive convolution and pooling layers.

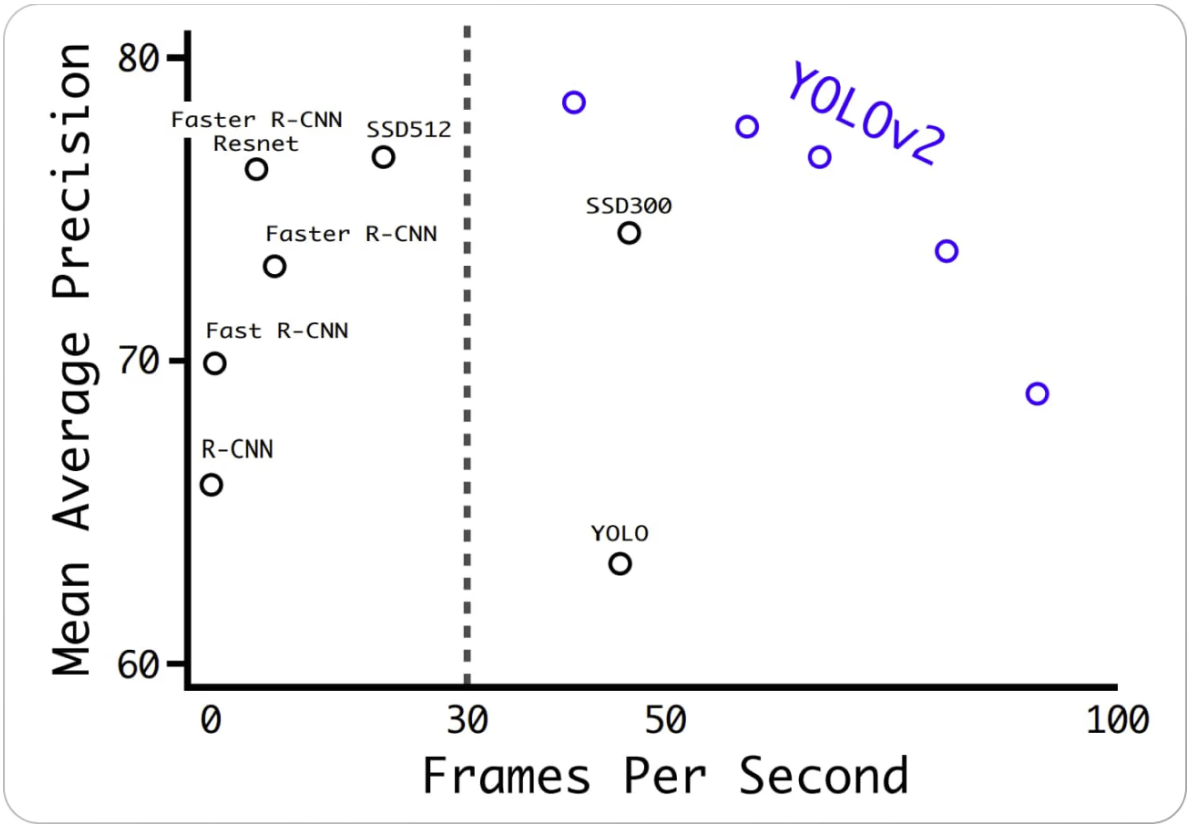
One of the main improvements in YOLO v2 is the use of anchor boxes. Anchor boxes are a set of predefined bounding boxes of different aspect ratios and scales. When predicting bounding boxes, YOLO v2 uses a combination of the anchor boxes and the predicted offsets to determine the final bounding box. This allows the algorithm to handle a wider range

of object sizes and aspect ratios.

Another improvement in YOLO v2 is the use of batch normalization, which helps to improve the accuracy and stability of the model. YOLO v2 also uses a multi-scale training strategy, which involves training the model on images at multiple scales and then averaging the predictions. This helps to improve the detection performance of small objects.

YOLO v2 also introduces a new loss function better suited to object detection tasks. The loss function is based on the sum of the squared errors between the predicted and ground truth bounding boxes and class probabilities.

The results obtained by YOLO v2 compared to the original version and other contemporary models are shown below.



Source: Paper

YOLO v3

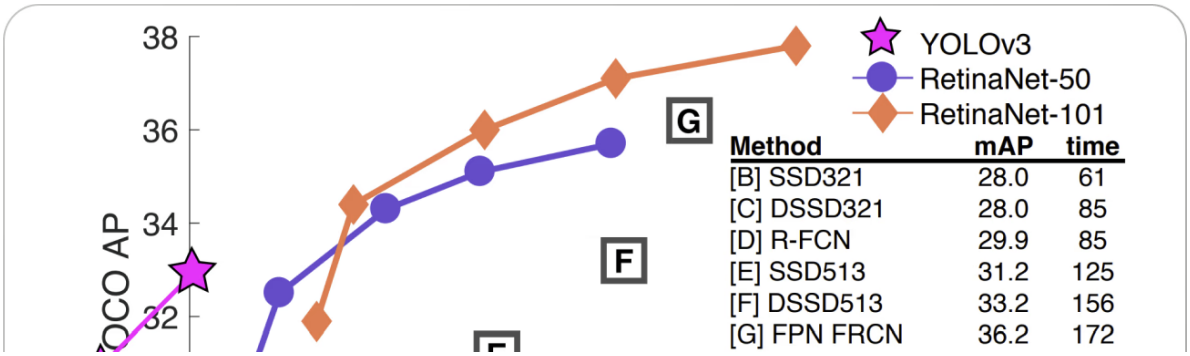
YOLO v3 is the third version of the YOLO object detection algorithm. It was introduced in 2018 as an improvement over YOLO v2, aiming to increase the accuracy and speed of the algorithm.

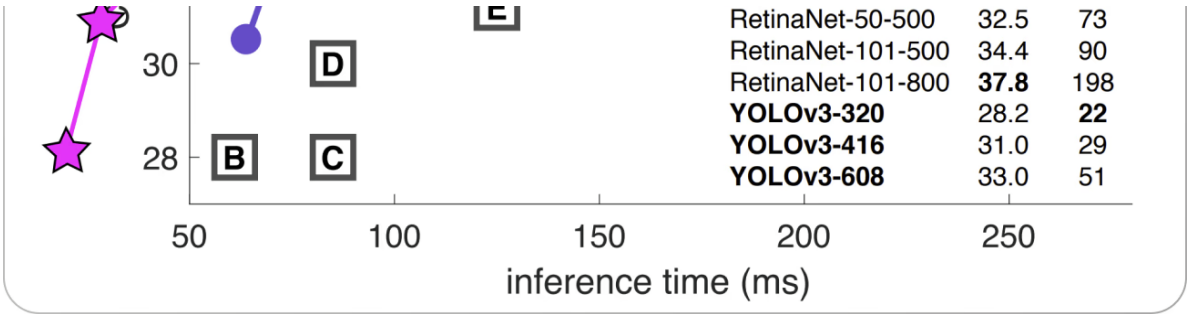
One of the main improvements in YOLO v3 is the use of a new CNN architecture called Darknet-53. Darknet-53 is a variant of the ResNet architecture and is designed specifically for object detection tasks. It has 53 convolutional layers and is able to achieve state-of-the-art results on various object detection benchmarks.

Another improvement in YOLO v3 are anchor boxes with different scales and aspect ratios. In YOLO v2, the anchor boxes were all the same size, which limited the ability of the algorithm to detect objects of different sizes and shapes. In YOLO v3 the anchor boxes are scaled, and aspect ratios are varied to better match the size and shape of the objects being detected.

YOLO v3 also introduces the concept of "feature pyramid networks" (FPN). FPNs are a CNN architecture used to detect objects at multiple scales. They construct a pyramid of feature maps, with each level of the pyramid being used to detect objects at a different scale. This helps to improve the detection performance on small objects, as the model is able to see the objects at multiple scales.

In addition to these improvements, YOLO v3 can handle a wider range of object sizes and aspect ratios. It is also more accurate and stable than the previous versions of YOLO.





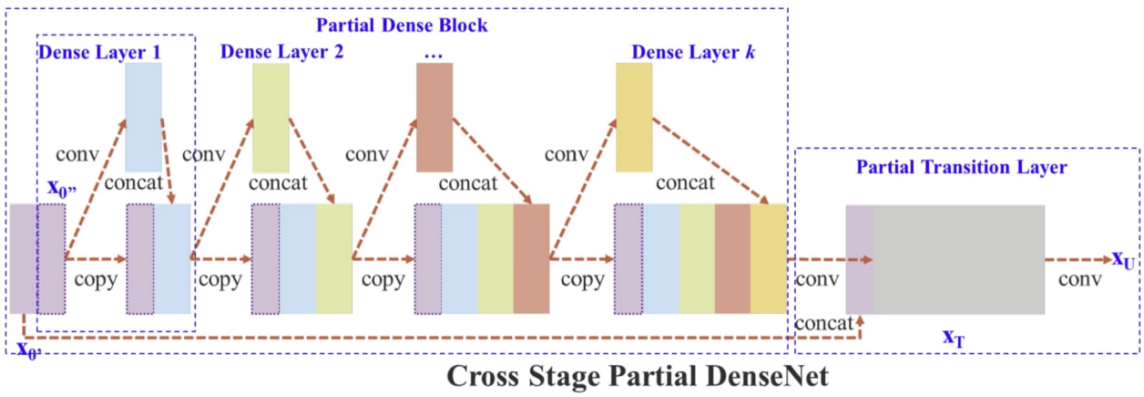
Comparison of the results obtained by YOLO v3. Source: Paper

YOLO v4

Note: Joseph Redmond, the original creator of YOLO, has left the AI community a few years before, so YOLOv4 and other versions past that are not his official work. Some of them are maintained by co-authors, but none of the releases past YOLOv3 is considered the "official" YOLO.

YOLO v4 is the fourth version of the YOLO object detection algorithm introduced in 2020 by Bochkovskiy et al. as an improvement over YOLO v3.

The primary improvement in YOLO v4 over YOLO v3 is the use of a new CNN architecture called CSPNet (shown below). CSPNet stands for "Cross Stage Partial Network" and is a variant of the ResNet architecture designed specifically for object detection tasks. It has a relatively shallow structure, with only 54 convolutional layers. However, it can achieve state-of-the-art results on various object detection benchmarks.



Architecture of CSPNet. Source: Paper

Both YOLO v3 and YOLO v4 use anchor boxes with different scales and aspect ratios to better match the size and shape of the detected objects. YOLO v4 introduces a new method for generating the anchor boxes, called "k-means clustering." It involves using a clustering algorithm to group the ground truth bounding boxes into clusters and then using the centroids of the clusters as the anchor boxes. This allows the anchor boxes to be more closely aligned with the detected objects' size and shape.

While both YOLO v3 and YOLO v4 use a similar loss function for training the model, YOLO v4 introduces a new term called "GHM loss." It's a variant of the focal loss function and is designed to improve the model's performance on imbalanced datasets. YOLO v4 also improves the architecture of the FPNs used in YOLO v3.

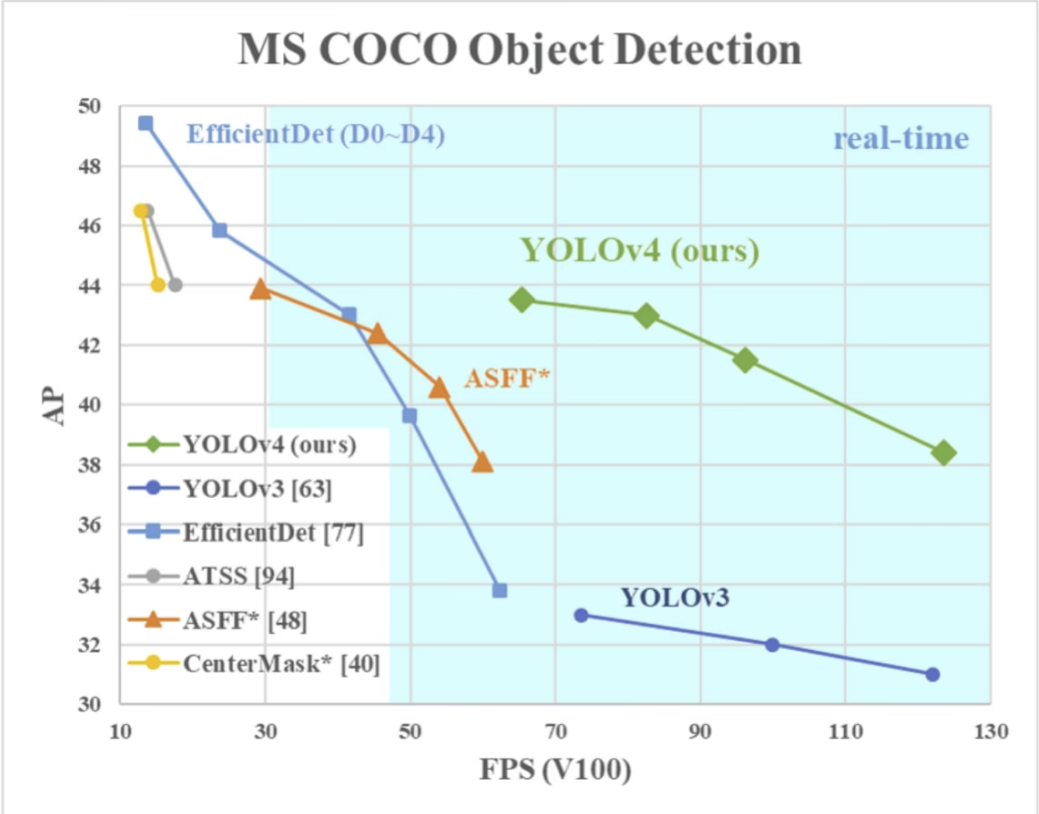


Figure 1: Comparison of the proposed YOLOv4 and other

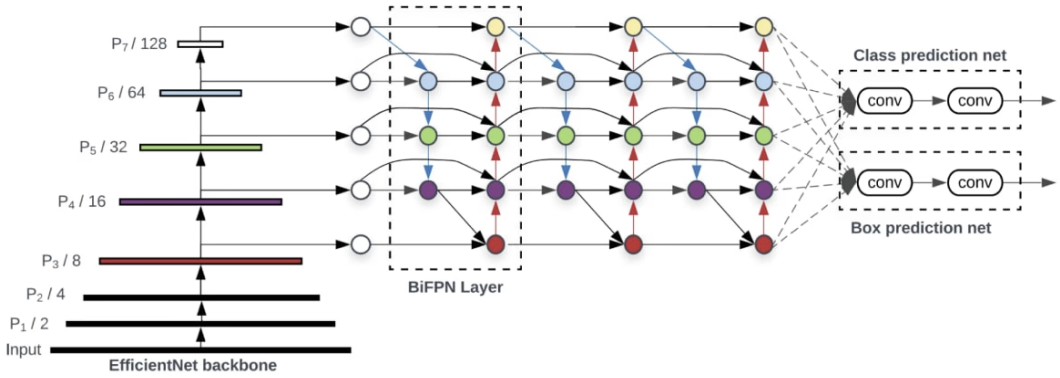
state-of-the-art object detectors. YOLOv4 runs twice faster than EfficientDet with comparable performance. Improves YOLOv3's AP and FPS by 10% and 12%, respectively.

Comparative performance of YOLO v4. Source: Paper

YOLO v5

YOLO v5 was introduced in 2020 by the same team that developed the original YOLO algorithm as an open-source project and is maintained by Ultralytics. YOLO v5 builds upon the success of previous versions and adds several new features and improvements.

Unlike YOLO, YOLO v5 uses a more complex architecture called EfficientDet (architecture shown below), based on the EfficientNet network architecture. Using a more complex architecture in YOLO v5 allows it to achieve higher accuracy and better generalization to a wider range of object categories.



EfficientDet architecture – It employs EfficientNet [39] as the backbone network, BiFPN as the feature network, and shared class/box prediction network. Both BiFPN layers and class/box net layers are repeated multiple times based on different resource constraints as shown in Table 1.

Architecture of the EfficientDet model. Source: Paper

Another difference between YOLO and YOLO v5 is the training data used to learn the object detection model. YOLO was trained on the PASCAL VOC dataset, which consists of 20 object categories. YOLO v5, on the other hand, was trained on a larger and more diverse dataset

called D5, which includes a total of 600 object categories.

YOLO v5 uses a new method for generating the anchor boxes, called "dynamic anchor boxes." It involves using a clustering algorithm to group the ground truth bounding boxes into clusters and then using the centroids of the clusters as the anchor boxes. This allows the anchor boxes to be more closely aligned with the detected objects' size and shape.

YOLO v5 also introduces the concept of "spatial pyramid pooling" (SPP), a type of pooling layer used to reduce the spatial resolution of the feature maps. SPP is used to improve the detection performance on small objects, as it allows the model to see the objects at multiple scales. YOLO v4 also uses SPP, but YOLO v5 includes several improvements to the SPP architecture that allow it to achieve better results.

YOLO v4 and YOLO v5 use a similar loss function to train the model. However, YOLO v5 introduces a new term called "CIoU loss," which is a variant of the IoU loss function designed to improve the model's performance on imbalanced datasets.

YOLO v6

YOLO v6 was proposed in 2022 by Li et al. as an improvement over previous versions. One of the main differences between YOLO v5 and YOLO v6 is the CNN architecture used. YOLO v6 uses a variant of the EfficientNet architecture called EfficientNet-L2. It's a more efficient architecture than EfficientDet used in YOLO v5, with fewer parameters and a higher computational efficiency. It can achieve state-of-the-art results on various object detection benchmarks. The framework of the YOLO v6 model is shown below.

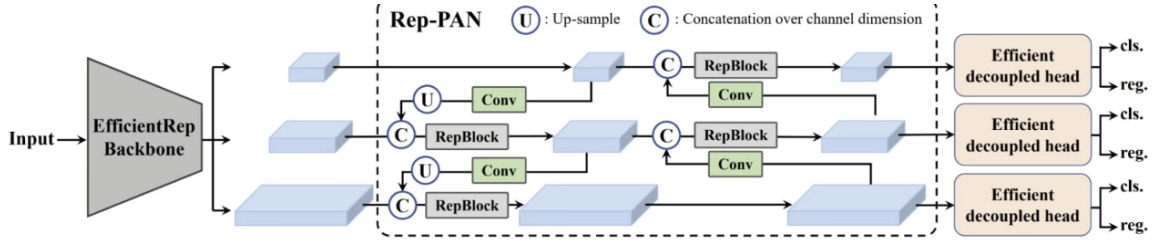
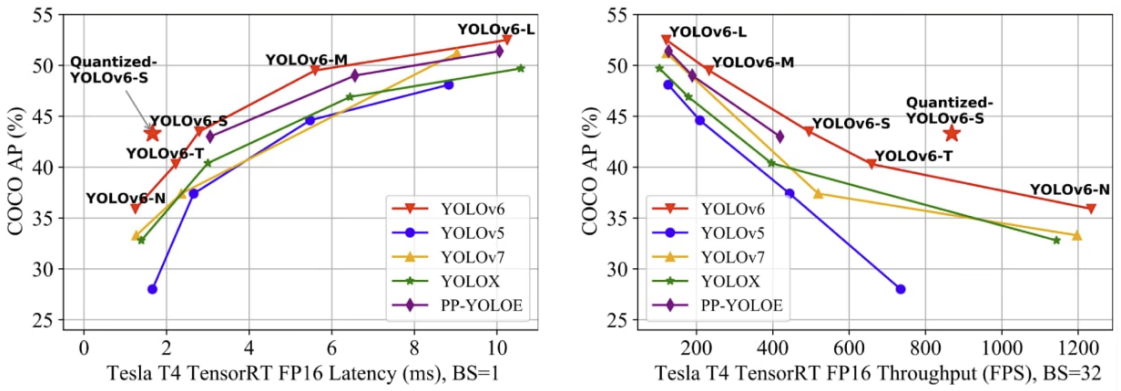


Figure 2: The YOLOv6 framework (N and S are shown). Note for M/L, RepBlocks is replaced with CSPStackRep.

Overview of YOLO v6. Source: Paper

YOLO v6 also introduces a new method for generating the anchor boxes, called "dense anchor boxes."

The results obtained by YOLO v6 compared to other state-of-the-art methods are shown below.



Comparison of state-of-the-art efficient object detectors. Both latency and throughput (at a batch size of 32) are given for a handy reference. All models are test with TensorRT 7 except that the quantized model is with TensorRT 8.

What's new with YOLO v7?

YOLO v7, the latest version of YOLO, has several improvements over the previous versions. One of the main improvements is the use of anchor boxes.

Anchor boxes are a set of predefined boxes with different aspect ratios that are used to detect objects of different shapes. YOLO v7 uses nine anchor boxes, which allows it to detect a wider range of object shapes

and sizes compared to previous versions, thus helping to reduce the number of false positives.

Here is YOLO v7 in action:

A key improvement in YOLO v7 is the use of a new loss function called “focal loss.” Previous versions of YOLO used a standard cross-entropy loss function, which is known to be less effective at detecting small objects. Focal loss battles this issue by down-weighting the loss for well-classified examples and focusing on the hard examples—the objects that are hard to detect.

YOLO v7 also has a higher resolution than the previous versions. It processes images at a resolution of 608 by 608 pixels, which is higher than the 416 by 416 resolution used in YOLO v3. This higher resolution allows YOLO v7 to detect smaller objects and to have a higher accuracy overall.

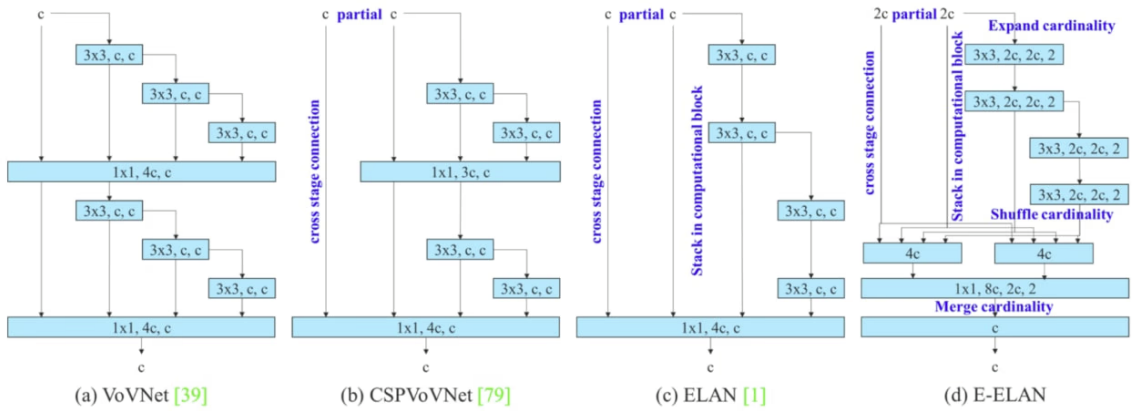
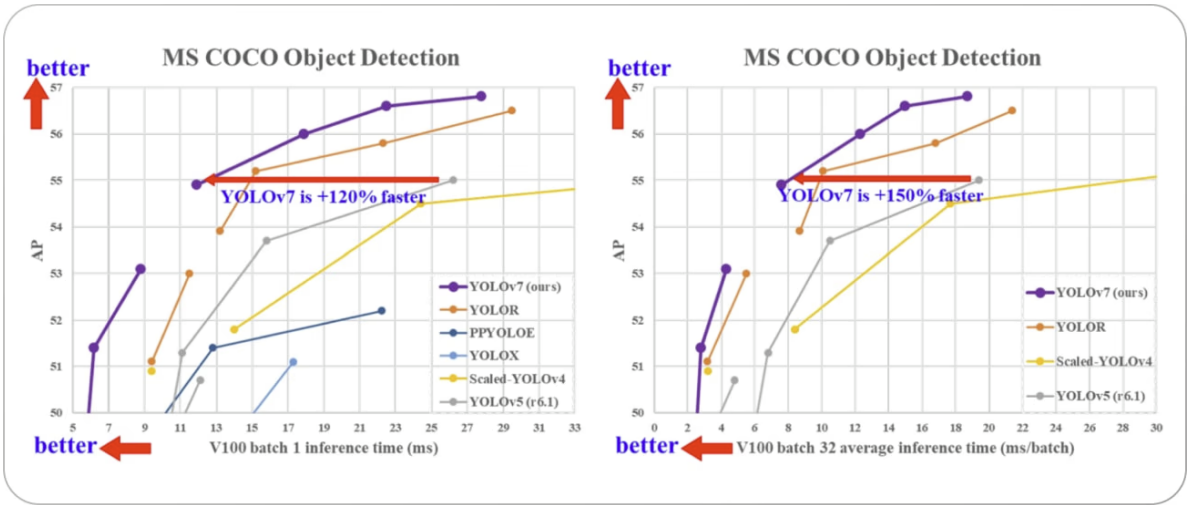


Figure 2: Extended efficient layer aggregation networks. The proposed extended ELAN (E-ELAN) does not change the gradient transmission path of the original architecture at all, but use group convolution to increase the cardinality of the added features, and combine the features of different groups in a shuffle and merge cardinality manner. This way of operation can enhance the features learned by different feature maps and improve the use of parameters and calculations.

Change in the layer aggregation scheme of YOLO v7 for efficient object feature learning. Source: Paper

One of the main advantages of YOLO v7 is its speed. It can process images at a rate of 155 frames per second, much faster than other state-of-the-art object detection algorithms. Even the original baseline YOLO

model was capable of processing at a maximum rate of 45 frames per second. This makes it suitable for sensitive real-time applications such as surveillance and self-driving cars, where higher processing speeds are crucial.



Comparison in performance and inference speed of YOLO v7 with contemporary state-of-the-art real-time object detectors. AP = Average Precision. Source: Paper

Regarding accuracy, YOLO v7 performs well compared to other object detection algorithms. It achieves an average precision of 37.2% at an IoU (intersection over union) threshold of 0.5 on the popular COCO dataset, which is comparable to other state-of-the-art object detection algorithms. The quantitative comparison of the performance is shown below.

Model	#Param.	FLOPs	Size	AP^{val}	AP^{val}_{50}	AP^{val}_{75}	AP^{val}_{S}	AP^{val}_{M}	AP^{val}_{L}
YOLOv4 [3]	64.4M	142.8G	640	49.7%	68.2%	54.3%	32.9%	54.8%	63.7%
YOLOR-u5 (r6.1) [81]	46.5M	109.1G	640	50.2%	68.7%	54.6%	33.2%	55.5%	63.7%
YOLOv4-CSP [79]	52.9M	120.4G	640	50.3%	68.6%	54.9%	34.2%	55.6%	65.1%
YOLOR-CSP [81]	52.9M	120.4G	640	50.8%	69.5%	55.3%	33.7%	56.0%	65.4%
YOLOv7	36.9M	104.7G	640	**51.2%**	**69.7%**	**55.5%**	**35.2%**	**56.0%**	**66.7%**
improvement	-43%	-15%	-	+0.4	+0.2	+0.2	+1.5	=	+1.3
YOLOR-CSP-X [81]	96.9M	226.8G	640	52.7%	**71.3%**	57.4%	36.3%	57.5%	68.3%
YOLOv7-X	71.3M	189.9G	640	**52.9%**	71.1%	**57.5%**	**36.9%**	**57.7%**	**68.6%**
improvement	-36%	-19%	-	+0.2	-0.2	+0.1	+0.6	+0.2	+0.3
YOLOv4-tiny [79]	6.1	6.9	416	24.9%	42.1%	25.7%	8.7%	28.4%	39.2%
YOLOv7-tiny	6.2	5.8	416	**35.2%**	**52.8%**	**37.3%**	**15.7%**	**38.0%**	**53.4%**
improvement	+2%	-19%	-	+10.3	+10.7	+11.6	+7.0	+9.6	+14.2
YOLOv4-tiny-3l [79]	8.7	5.2	320	30.8%	47.3%	32.2%	**10.9%**	31.9%	51.5%
YOLOv7-tiny	6.2	3.5	320	**30.8%**	**47.3%**	**32.2%**	10.0%	**31.9%**	**52.2%**
improvement	39%	49%	-	-	-	-	0.9	-	+0.7

improvement	-39%	-49%	-	=	=	=	-0.9	=	+0.7
YOLOR-E6 [81]	115.8M	683.2G	1280	55.7%	73.2%	60.7%	40.1%	**60.4%**	69.2%
YOLOv7-E6	97.2M	515.2G	1280	**55.9%**	**73.5%**	**61.1%**	**40.6%**	60.3%	**70.0%**
improvement	-19%	-33%	-	+0.2	+0.3	+0.4	+0.5	-0.1	+0.8
YOLOR-D6 [81]	151.7M	935.6G	1280	56.1%	73.9%	61.2%	**42.4%**	60.5%	69.9%
YOLOv7-D6	154.7M	806.8G	1280	56.3%	73.8%	61.4%	41.3%	60.6%	70.1%
YOLOv7-E6E	151.7M	843.2G	1280	**56.8%**	**74.4%**	**62.1%**	40.8%	**62.1%**	**70.6%**
improvement	=	-11%	-	+0.7	+0.5	+0.9	-1.6	+1.6	+0.7

Source: Paper

However, it should be noted that YOLO v7 is less accurate than two-stage detectors such as Faster R-CNN and Mask R-CNN, which tend to achieve higher average precision on the COCO dataset but also require longer inference times.

Limitations of YOLO v7

YOLO v7 is a powerful and effective object detection algorithm, but it does have a few limitations.

1. YOLO v7, like many object detection algorithms, struggles to detect small objects. It might fail to accurately detecting objects in crowded scenes or when objects are far away from the camera.
2. YOLO v7 is also not perfect at detecting objects at different scales. This can make it difficult to detect objects that are either very large or very small compared to the other objects in the scene.
3. YOLO v7 can be sensitive to changes in lighting or other environmental conditions, so it may be inconvenient to use in real-world applications where lighting conditions may vary.
4. YOLO v7 can be computationally intensive, which can make it difficult to run in real-time on resource-constrained devices like smartphones or other edge devices.

YOLO v8

At the time of writing this article, the release of YOLO v8 has been confirmed by Ultralytics that promises new features and improved performance over its predecessors. YOLO v8 boasts of a new API that will make training and inference much easier on both CPU and GPU devices and the framework will support previous YOLO versions. The developers are still working on releasing a scientific paper that will include a detailed description of the model architecture and performance.