APSIPA Transactions on Signal and Information Processing, 2024, 13, e29

Overview Paper

YOLOv1 to YOLOv10: The Fastest and Most Accurate Real-time Object Detection Systems

Chien-Yao Wang[1,2*] and Hong-Yuan Mark Liao[1,2,3]

[1] *Institute of Information Science, Academia Sinica, Taiwan*
[2] *National Taipei University of Technology, Taiwan*
[3] *National Chung Hsing University, Taiwan*

ABSTRACT

This is a comprehensive review of the YOLO series of systems. Different from previous literature surveys, this review article re-examines the characteristics of the YOLO series from the latest technical point of view. At the same time, we also analyzed how the YOLO series continued to influence and promote real-time computer vision-related research and led to the subsequent development of computer vision and language models. We take a closer look at how the methods proposed by the YOLO series in the past ten years have affected the development of subsequent technologies and show the applications of YOLO in various fields. We hope this article can play a good guiding role in subsequent real-time computer vision development.

Keywords: YOLO, computer vision, real-time object detection

1 Introduction

Object detection is a fundamental computer vision task that can support a wide range of downstream tasks. For example, it can be used to assist instance

*Corresponding author: kinyiu@iis.sinica.edu.tw.

Received 17 August 2024; revised 20 September 2024; accepted 09 October 2024
ISSN 2048-7703; DOI 10.1561/116.20240058

segmentation, multi-object tracking, behavior analysis and recognition, face recognition, etc. Therefore, it has been a popular research topic in the past few decades. In recent years, due to the popularity of mobile devices, the ability to perform real-time object detection on the edge has become a necessary component for various real-world applications. Tasks belonging to such applications include autonomous driving, industrial robots, identity authentication, smart health care, visual surveillance, etc. Among the many real-time object detection algorithms, the YOLO (You Only Look Once) series (from v1 to v10) [2, 26, 37, 57, 82–84, 102, 105, 109] developed in recent years is particularly outstanding. It has greatly and extensively affected various research in the field of computer vision. This paper will review the YOLO family of technologies and their impact on the development of contemporary real-time computer vision systems.

The first deep learning-based method to achieve breakthrough success in the field of object detection was R-CNN [24]. R-CNN is a two-stage object detection method, which divides the object detection process into two stages: object proposal generation and object proposal classification. What R-CNN does is to first use selective search [99], which is commonly used in image processing, to extract proposals. At this stage, CNN is only used as a feature extractor to extract features of proposals. As for the recognition part, SVM [49] is used. The subsequent development of Fast R-CNN [23] and Faster R-CNN [85] respectively used SPPNet [48] to accelerate feature extraction and proposed Region Proposal Networks to gradually convert object detection into the end-to-end format. YOLO [82] was proposed by Joseph Radmon in 2015. It uses per gird prediction to complete object detection in one step. This is a groundbreaking approach that brings the field of real-time object detection to a whole new level. The subsequent development of classic one-stage object detection systems includes SSD [67], RetinaNet [62], FCOS [98], etc.

Although the one-stage object detection method can detect objects in real time, there is still a gap in accuracy from the two-stage object detection method. The one-stage detection systems such as RetinaNet [62] and YOLOv3 [84] have made significant progress on this issue, and they both achieved sufficient accuracy. YOLO series have become the most preferred method for industry and all academia and research centers that require real-time object analysis. In 2020, scaled-YOLOv4 [104] further designed a very effective object detection model scaling method. For the first time, the accuracy of the one-stage object detection method in the field of general object detection surpassed all contemporary two-stage object detection methods, and this achievement also led to many subsequent related research based on YOLO series methods.

In addition to object detection, YOLO series is also used in other computer vision fields as a basis for developing real-time systems. Currently in instance segmentation, pose estimation, image segmentation, 3D object detection, open-vocabulary object detection, etc., YOLO still plays a pivotal role in real-time systems.

In this review article, we will introduce the following issues in order:

- Introduction to the YOLO series methods and important related literature.
- The impact of the YOLO family of methods on the contemporary field of computer vision.
- Important methods for applying YOLO in different computer vision fields.

2 YOLO series

YOLO is synonymous with the most advanced real-time object detector of our time. The biggest difference between YOLO and traditional object detection systems is that it abandons the previous two-stage object detection method that requires first finding the locations where objects may be located in the image, and then analyzing the content of these locations individually. YOLO proposes a unified one-stage object detection method, and this method is streamlined and efficient, which makes YOLO widely used in various edge devices and real-time applications. Next we will introduce several representative YOLO versions, as listed in Table 1, and this literature review is different from the previous ones. We will put our emphasis on the state-of-the-art object detection methods and review the advantages and disadvantages of these methods.

2.1 YOLO (YOLOv1)

YOLOv1: Redmon *et al.* [82] was the first one who proposed the one-stage object detector in 2015, and the architecture of YOLOv1 is illustrated in Figure 1. As shown in the figure, an input image first passes through CNN for feature extraction, and then passes through two fully connected layers to obtain global features. Then, the aforementioned global features are reshaped back to the two-dimensional space for per grid prediction. YOLOv1 has the following important features:
One-Stage Object Detector. As shown in Figure 1, YOLOv1 directly classifies each grid of feature map, and also predicts B bounding boxes. Each bounding box will predict the object center (b_x, b_y), object size (b_w, b_h), and object score (b_{obj}) respectively. The one stage prediction method does not need to rely on the selective search that must be executed in the object proposal generation stage, which can avoid missed detections caused by insufficient manual design clues. In addition, the one-stage method can avoid the large number of parameters and calculations generated by fully connected layers

Table 1: List of YOLO series papers and publication time.

Model	Date	Publication	Citation
YOLO (YOLO1)	2015.06	CVPR 2016	[82]
YOLO9000 (YOLOv2)	2016.12	CVPR 2017	[83]
YOLOv3	2018.04	arXiv 2018	[84]
Gaussian YOLOv3	2019.03	ICCV 2019	[14]
YOLOv4	2020.01	arXiv 2020	[2]
YOLOv5	2020.05	–	[26]
Scaled-YOLOv4	2020.06	CVPR 2021	[104]
YOLOv5 r1.0	2020.06	–	[27]
PP-YOLO	2020.07	arXiv 2020	[68]
YOLOv5 r2.0	2020.07	–	[28]
YOLOv5 r3.0	2020.08	–	[29]
Scaled-YOLOv4 P6	2020.08	CVPR 2021	[104]
YOLOv5 r3.1	2020.10	–	[30]
YOLOv5 r4.0	2021.01	–	[31]
PP-YOLOv2	2021.04	arXiv 2021	[51]
YOLOv5 r5.0	2021.04	–	[32]
YOLOR	2021.05	JISE 2023	[110]
YOLOX	2021.07	arXiv 2021	[22]
YOLOv5 r6.0	2021.10	–	[33]
PP-PicoDet	2021.11	arXiv 2021	[119]
YOLOv5 r6.1	2022.02	–	[34]
PP-YOLOE	2022.03	arXiv 2022	[116]
YOLOv6	2022.06	–	[73]
YOLOv7	2022.07	CVPR 2023	[105]
YOLOv5 r6.2	2022.08	–	[35]
YOLOv6 2.0	2022.09	arXiv 2022	[57]
YOLOv7 AF	2022.11	CVPR 2023	[105]
DAMO-YOLO	2022.11	arXiv 2022	[117]
YOLOv5 r7.0	2022.11	–	[36]
YOLOv8	2023.01	–	[37]
YOLOv6 3.0	2023.01	arXiv 2023	[56]
YOLOv6 4.0	2023.04	OpenReview 2023	[58]
YOLO-NAS	2023.05	–	[93]
Gold-YOLO	2023.09	NeurIPS 2023	[103]
YOLOv8 r1.0	2024.01	–	[38]
YOLOv9	2024.02	ECCV 2024	[109]
YOLOv8 r2.0'	2024.04	–	[40]
YOLOv10	2024.05	NeurIPS 2024	[102]
YOLOv8 r2.0	2024.06	–	[39]
YOLOv8 r3.0	2024.09	–	[41]

in the second stage, and it can avoid the irregular operations required when connecting two stages of RoI operations. Therefore, YOLO's design can capture features and make predictions more timely and effectively. Below we will take a closer look at the most important concepts in YOLOv1, which are anchor-free bounding box regression, IoU-aware objectness, and global context features.

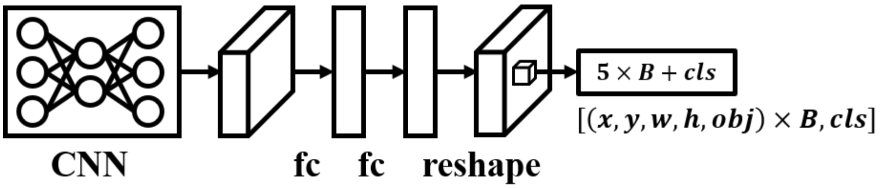


Figure 1: Architecture of YOLOv1. The architecture of YOLOv1 allows users to extract global features from fully-connected layers, predict B anchor-free bounding boxes at each position, and use IoU-aware objectness score as the filtering basis.

Anchor-free Bounding Box Regression. In Equation 1, YOLOv1 directly predicts the proportion of the length and width of the object in the entire image. Although the anchor-free method requires optimization of a large dynamic range of length and width, which makes convergence more difficult, it also has the advantage of being able to predict some special examples more accurately because it is not restricted by anchors.

$$\begin{aligned} b_x &= t_x + c_x, \\ b_y &= t_y + c_y, \\ b_w &= {t_w}^2, \\ b_h &= {t_h}^2 \end{aligned} \tag{1}$$

IoU-aware Objectness. In order to more accurately measure the quality of bounding box prediction, the method proposed by YOLOv1 is to predict the IoU value between a certain bounding box and the assigned ground-truth bounding box, and use this as the soft label of the objectness predicted by IoU-aware branch. Finally, the confidence score of bounding box is determined by the product of objectness score and classification probability.

Global Context Feature. To ensure that a grid doesn't only see the local feature and cause prediction errors, YOLOv1 uses fully connected layer to retrieve global context features. In such a design, no matter what the underlying CNN architecture is, each grid can see a sufficient range of features to predict the target object during prediction. Compared with fast R-CNN, this design effectively reduces background error by more than half.

2.2 YOLO9000 (YOLOv2)

In addition to proposing many insightful new methods, Redmon and Farhadi [83] also integrate various existing techniques. They designed an object detector that combines high accuracy and speed, as shown in Figure 2. They converted the entire object detection architecture to full convolutional network. They

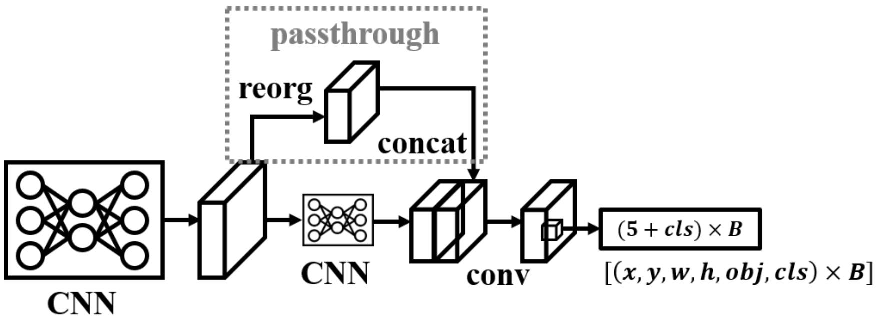


Figure 2: Architecture of YOLOv2. The architecture of YOLOv2 is to extract mix-resolution features by passthrough layer, and then predict the objects corresponding to B anchors for each position.

then combined high-resolution and low-resolution features, and finally use anchor-based for prediction. Due to its simple input and output formats, YOLOv2 is still one of the mainstream object detection methods commonly used in maintenance and development of many industrial scenes, especially on low-end devices with very limited computing resources. Below we will discuss the most essential parts of YOLOv2 respectively regarding dimension cluster, direct location prediction, fine-grained feature, resolution calibration, and joint training with WordTree issues.

Dimension Cluster. YOLOv2 proposed to use IoU distance as the basis to k-means clustering on ground-truth bounding boxes to obtain anchors. On the one hand, the anchor obtained by using dimension cluster can avoid the original manually set aspect ratio, which is difficult to learn the bounding box prediction of the object. On the other hand, it is also easier to converge than the bounding box regression of the anchor-free approach.

Direct Location Prediction. Faster R-CNN uses the anchor center as the basis to predict the offset between the object center and the anchor center. The above approach is very unstable during early training. YOLOv2 follows the object center regression method of YOLOv1 and directly predicts the true position of the object center based on the upper left corner of the grid responsible for predicting the object.

Fine-grained Feature. Passthrough layer predicts by reorganizing high-resolution features into lossless spatial-to-depth and combining them with low-resolution features. This can enhance fine-grained small object detection capabilities through high-resolution features while taking into account speed simultaneously.

Resolution Calibration. Since the backbone CNN often uses lower-resolution images for image classification pre-training than those used for object detection

training, the pre-trained model has never seen the state of larger objects. YOLOv2 uses the image classification pre-train of the same training size image, so that the object detection training process does not require additional learning of new size object information.
Joint Training with WordTree. YOLOv2 designed the training of group softmax using ImageNet with a similar hierarchy as WordTree, and then integrated the categories of COCO and ImageNet using Word- Tree. In the end, this technology requires joint training of ImageNet's image classification and COCO's object detection tasks. Because of the above design, YOLOv2 has the ability to detect 9000 categories of objects.

2.3 YOLOv3

YOLOv3 [84] was proposed by Redmon and Farhadi in 2018. They integrated the advanced technology of existing object detection and made corresponding optimizations to one-stage object detectors. As shown in Figure 3, in terms of architecture, YOLOv3 mainly combines FPN [61] to enable prediction of multiple scales at the same time. It also introduces the residual network architecture and designs DarkNet53. In addition, YOLOv3 also made significant changes to the label assignment task. The first change is that a ground truth will only be assigned to one anchor, while the second change is to change from soft label to hard label for IoU aware objectness. To this day YOLOv3 is still the most popular version of YOLO series. In what follows, let us detail the special designs of YOLOv3, namely prediction across scales, high GPU utility, and SPP.

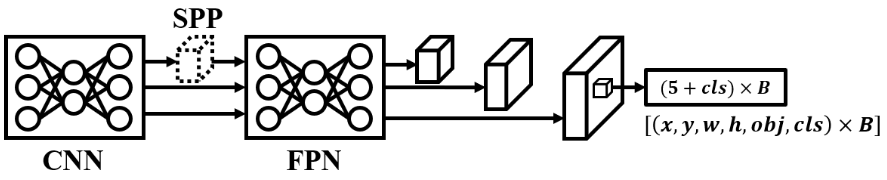


Figure 3: Architecture of YOLOv3, YOLOv5, and PP-YOLO. The design of YOLOv3 mainly changes the feature extraction to use SPP (optional) and FPN to extract multi-resolution features. The initial versions of YOLOv5 and PP-YOLO also follow this architecture.

Predictions Across Scales. YOLOv3 combines FPN to achieve prediction across scales, which can greatly improve the detection ability of small objects.
High GPU Utility. In the time of 2018, mainstream network architecture design focuses on reducing the amount of calculations and parameters. The design of Draknet53 has higher GPU hardware utilization than other architectures, so it has faster inference speed under the same amount of calculations.

Such a design has also led to subsequent architectural research focusing on actual hardware inference speed.

SPP. YOLOv1 uses fully connected layer to obtain global context features, while YOLOv2 uses passthrough layer to combine multiple resolution features. YOLOv3 designed multiple maximum pooling layers with a stride of 1 for kernel size from local to global. This design allows each grid to obtain multiple resolution features from local to global. SPP has been proven to be a simple, efficient method that can greatly improve accuracy.

2.4 Gaussian YOLOv3

Gaussian YOLOv3 [14] proposed a great way to significantly reduce the false positive of an object detection process. Gaussian YOLOv3 mainly changes the decoding method of the prediction head, and the method used is to convert the bounding box numerical regression problem into predicting its distribution. Its architecture is shown in Figure 4.

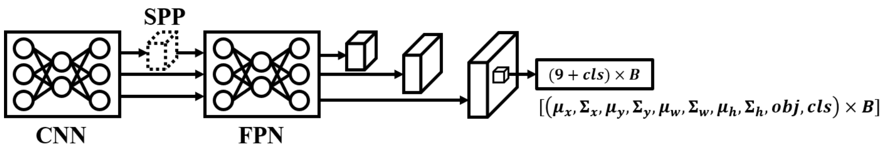

SPP
CNN
FPN
$(9 + cls) \times B$
$[(\mu_x, \Sigma_x, \mu_y, \Sigma_y, \mu_w, \Sigma_w, \mu_h, \Sigma_h, obj, cls) \times B]$

Figure 4: Architecture of Gaussian YOLOv3. Gaussian YOLOv3 changes regression head of bounding box to Gaussian distribution-based regressor.

Distribution-Based Bounding Box Regression. The distribution-based bounding box regression module included in the figure is the uncertainty of predicting bounding box (x, y, w, h) of a Gaussian distribution. This prediction method can significantly reduce the false positive of object detection.

2.5 YOLOv4

Since Joseph Redmon withdrew from computer vision research for some reason, subsequent versions of YOLO were mainly released on the open source platform GitHub. As for the publication time of the paper, it is later than the open source time. YOLOv4 [2] was submitted to Joseph Redmon as a draft by Alexey Bochkovskiy, in early April 2020, and was officially released on April 23 2020. YOLOv4 mainly integrates various technologies in different fields of computer vision in recent years to improve the learning effect of real-time object detectors. The architectural change of YOLOv4 is to replace FPN with PAN [66] and introduce CSPNet [106] as backbone, as shown in Figure 5. Most of the subsequent similar YOLO architectures followed this architecture.

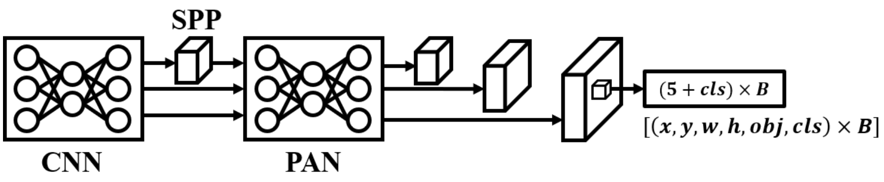


Figure 5: Architecture of YOLOv4 [2], Scaled-YOLOv4 [104], YOLOv5 r1–r7 [27–36], and PP-YOLOv2 [51]. YOLOv4 change the feature integration architecture to Path Aggregation Network (PAN) [66], and almost all subsequent YOLO versions adopted this design.

In view of the rapid innovation of deep learning technology YOLOv4 was not only developed based on DarkNet [1, 79], but also implemented on the most successful PyTorch YOLOv3 [25] at the time. YOLOv4 successfully demonstrated how to use one GPU to train an object detector that is as accurate as those trained with more than 128 GPUs, and at the same time has more than three times the inference speed. The excellent performance of YOLOv4 has also led to many subsequent object detection research. The following is a list of new features developed in YOLOv4.

Bag of Freebies. YOLOv4 introduces training techniques that only increase training time but do not affect inference time mainly including loss function, regularization methods, data augmentation methods, and label assignment methods.

Bag of Specials. YOLOv4 also introduces methods that only slightly affect the inference time but can greatly improve the accuracy, mainly including receptive field modules, attention mechanism, activation function, and normalization layers, and select useful combinations to add to the system.

Grid Sensitive Decoder. Users on the open source platform found that it was difficult to predict accurately when the object center was near the grid boundary. YOLOv4 analyzed the reason and found that the gradient from Sigmoid function would approach zero at extreme values. The developers of YOLOv4 then designed a decoding method as shown in Equation 2 to make the predicted target values fall within the effective gradient range.

$$\begin{aligned} b_x &= (1 + s_x)\sigma(t_x) - 0.5s_x + c_x, \\ b_y &= (1 + s_y)\sigma(t_y) - 0.5s_y + c_y, \\ b_w &= p_w e^{t_w}, \\ b_h &= p_h e^{t_h} \end{aligned} \tag{2}$$

Self-Adversarial Training. YOLOv4 also introduces self-adversarial sample generation training to enhance the robustness of the object detection system.

Training with Memory Sharing. YOLOv4 is also designed to allow GPU and CPU to share memory for storing the information required for gradient updates. This design allows the trained batch size no longer be limited by GPU memory.

2.6 Scaled-YOLOv4

In 2020, Wang *et al.* [104] continued the success achieved with YOLOv4 and continued to develop scaled-YOLOv4 that can be used on both edge and cloud. Thanks to the activity of the DarkNet and PyTorch YOLOv3 communities, scaled-YOLOv4 can abandon the pre-train steps required by ImageNet and directly use the train-from-scratch method to obtain high-quality object detection results. In terms of architecture, scaled-YOLOv4 has also introduced CSPNet into PAN, which can comprehensively improve the performance of speed, accuracy, number of parameters, and number of calculations. Scaled-YOLOv4 also designs model scaling methods for various edge devices and provides three types of models: P5, P6, and P7. In the training part, scaled-YOLOv4 uses the decoder and label assignment strategy proposed by the initial version of YOLOv5. Because of the various improvements mentioned above, scaled-YOLOv4 has achieved the highest accuracy and fastest inference speed of all object detectors. Below we list several unique designs of scaled-YOLOv4:

Compound Model Scaling. Previous model scaling methods only considered the integer hyperparameters of a given architecture. Scaled-YOLOv4 proposed a model scaling that simultaneously considers the input image resolution and receptive field matching, and uses the number of scaling model stages to design a more efficient architecture that can be applied to high-resolution images.

Hardware Friendly Architecture. Taking into account ShuffleNet- v2 [70] and HardNet's [8] analysis of hardware performance, the highly efficient CSPDark module and CSPOSA module were designed.

Naïve Once For All Model. Since scaled-YOLOv4 is trained in the mode of train-from-scratch, the problem of inconsistent resolution between the pre-trained models and the detection model no longer exists. However, the problem of inconsistency between user input images and training data still exists. The model scaling method proposed in scaled-YOLOv4 allows users to obtain the best accuracy without re-training during the inference stage, and only needs to remove the output of the corresponding stage.

2.7 YOLOv5

YOLOv5 [26] continues the design concept of PyTorch YOLO- v3 and has simplified and revised the overall architecture definition method. So far, there are about 10 different versions. The initial version is designed with an architecture similar to YOLOv3, while following EfficientDet's [95] model scaling pattern to provide models with different specifications. PyTorch YOLOv3 was developed from Erik's open source codes [63], so it uses GPL3 license, and subsequent versions are adjusted to the more strict AGPL3 license. YOLOv5 inherits many functions of PyTorch YOLOv3, such as using evolutionary algorithms for auto anchor and hyper-parameter search. When YOLOv5 [26] was open sourced, its performance was slightly worse than YOLOv3-SPP. After successively combining the CSPNet used by YOLOv4 and the CSPPAN used by scaled-YOLOv4, the first version of YOLOv5 r1.0 [27] was officially released in June 2020. Then the developers of YOLOv5 optimized both the speed-accuracy trade-off of the CSP fusion layer and the activation function, quoted YOLOR-based training hyper-parameters, used YOLOv5 r6.0 [33] in October 2021. The latest version of YOLOv5 is YOLOv5 r7.0 [36] Glenn released in November 2022. Due to the continuous maintenance and version updates by companies, YOLOv5 is currently the most popular YOLO development platform. Let us point out some distinctive features of YOLOv5 as follows:

Power-based Decoder. The width-height regression system of YOLOv3 uses exponential function to estimate offset, and this approach causes instability during training in some datasets. YOLOv5 proposed power-based decoder in Equation 3 to increase training stability. Since the output value range of power-based decoder is limited to a certain scaling range of anchor, there will be a theoretically bounded recall.

$$\begin{aligned} b_x &= 2\sigma(t_x) - 0.5 + c_x, \\ b_y &= 2\sigma(t_y) - 0.5 + c_y, \\ b_w &= p_w(2\sigma(t_w))^2, \\ b_h &= p_h(2\sigma(t_h))^2 \end{aligned} \tag{3}$$

Neighborhood Positive Samples. In order to make up for the deficiency caused by recall, YOLOv5 proposed to add more neighbor grids as positive samples. At the same time, in order to allow these neighbor grids to correctly predict the center point, they also enlarged the sigmoid scaling coefficient of the YOLOv4 center point decoder.

2.8 PP-YOLO

There are four versions of the PP-YOLO series, namely PP-YOLO [68], PP-YOLOv2 [51], PP-PicoDet [119], and PP-YOLOE [116]. PP-YOLO is improved based on YOLOv3. In addition to using a variety of YOLOv4 training techniques, it also adds CoordConv [65], Matrix NMS [111], and better ImageNet pre-trained model and other methods for improvement, while PP-YOLOv2 further introduces scaled-YOLOv4's CSPPAN and other mechanisms. PP-PicoDet uses neural architecture search as the basis to design the backbone, and introduces YOLOX's anchor-free decoder [22]. As for PP-YOLOE, it has made major changes, as shown in Figure 6. It modified RepVGG and designed CSPRepResStage and then used bounding box regression in TOOD's distribution-based regression process [20]. The YOLO series after YOLOv6 almost all follow the above format. Listed below are some design features of PP-YOLO series.

Neural Architecture Search. PP-PicoDet is an architecture designed for mobile devices. It combines ShuffleNetv2 [70] and GhostNet [46] for one shot neural architecture search.

Reparameterization Module. PP-YOLOE applies RepVGG [17] to CSP-Net, but removes the identity connection in the training phase.

Distribution-based Regression Raised. PP-YOLOE follows TOOD to use DFL [59] for bounding box regression. DFL is different from Gaussian YOLOv3 in that it does not need to limit the data to be Gaussian distribution, and can also directly learn the distribution of real data.

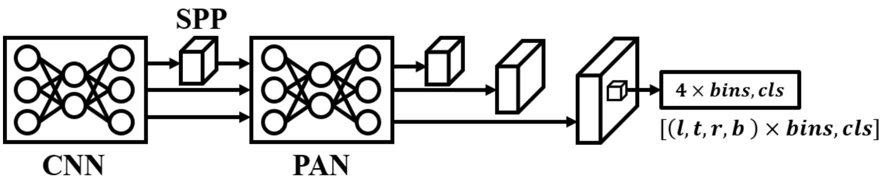


Figure 6: Architecture of PP-YOLOE [116], YOLOv6 2.0 [56], YOLOv7 AF [105], YOLOv8 [37], and YOLO-NAS [93]. PP-YOLOE changed bounding box regression head to TOOD's [20] anchor-free distribution-based regressor, and subsequent YOLO versions adopted this design.

2.9 YOLOR

YOLOR [110] is not an official version of the YOLO series, but its use of Latent Variable Model (LVM) as implicit knowledge encoder can significantly improve the detection effects of all YOLO series models, as shown in Figure 7. YOLOR's multi-task model has also been widely used in subsequent YOLO versions, and the advanced training technology it proposed has been continued and promoted in all subsequent versions. Below are some specially designed features of YOLOR.

Implicit Knowledge Modeling. YOLOR proposed three LVMs to encode implicit knowledge, including vector-based, neural network-based, and matrix factorization-based. The above three encoding methods can effectively enhance the feature alignment, prediction refinement, and multi-task learning capabilities of deep neural networks.

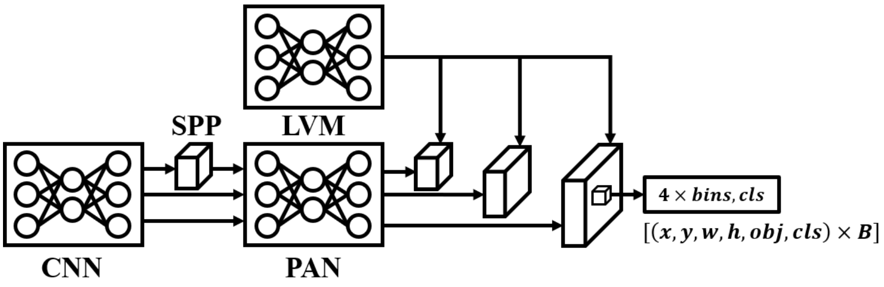


Figure 7: Architecture of YOLOR. YOLOR proposed refining bounding box regression head to Latent Variable Models (LVM).

Multi-task Model. YOLOR provides models that can perform object detection, image classification, and multiple object tracking at the same time, and it also provides pose estimation models that based on YOLO-Pose [71].
Advanced Training Technique. YOLOR developed advanced autoML technology [108], and its techniques for training hyperparameters are continued to be used in the latest version of YOLO series. YOLOR also uses large dataset pre-train, knowledge distillation, self-supervised learning, and self distillation technologies in its model. Until now, YOLOR trained using the above method is still the most accurate model of all YOLO series.

2.10 YOLOX

YOLOX [22], as shown in Figure 8, combined the most practical technologies at the time, mainly based on the CSPNet architecture [106] and FCOS' anchor-free head [98], improved OTA [21] and proposed the SimOTA dynamic label assignment method to replace the manual label assignment method that was easily confusing. Subsequent versions of YOLO also began to use or design different dynamic label assignment methods. The following describes the two most important features of YOLOX.
Decoupled Head. YOLOX uses decoupled head of FCOS, and this design makes classification and bounding box regression easier to learn.
Anchor-free Strikes Back. Due to the development of IoU-based loss, anchor-free head is no longer affected by the loss imbalance caused by the length and width of the object. With modern technology, anchor-free head can also be well trained. As for YOLOX, it takes FCOS' anchor-free head to achieve the planned objective.

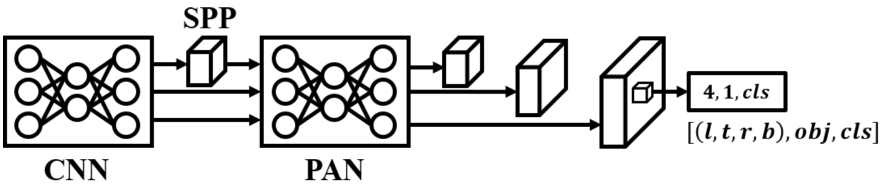


Figure 8: Architecture of YOLOX [22]. YOLOX proposed changing bounding box regression head to anchor-free regressor of FCOS [98], which also led to the development of the YOLO series towards anchor-free.

2.11 YOLOv6

The initial version of YOLOv6 [73] uses RepVGG [17] as the main architecture. In versions after version 2.0, such as Li *et al.* [56, 57], CSPNet [106] was introduced. YOLOv6 is a system specially designed for industry, so it has put a lot of effort into quantization issues. The contributions of YOLOv6 include using RepOPT [16] to make the quantized model more stable, and using quantization aware training (QAT) and knowledge distillation to enhance the accuracy of the quantized model. YOLOv6 version 3.0 [56] proposed a concept of anchor-aid training, as shown in Figure 9, to improve the accuracy of the system. Later in YOLOv6 version 4.0 [58], a lightweight architecture YOLOv6-lite based on depth-wise convolution was proposed to face lower-end computing devices. The following lists some of the unique features proposed by YOLOv6.

Reparameterizing Optimizer. YOLOv6 version 2.0 uses RepOPT to slow down the accuracy lost after model quantization.

Quantization Aware Training. In YOLOv6 version 2.0, QAT is used to improve the accuracy of the quantization model.

Knowledge Distillation. YOLOv6 version 2.0 uses self-distillation and channel-wise distillation respectively to improve model accuracy, and it also uses QAT to reduce the accuracy loss after model quantization.

Anchor-Aided Training. YOLOv6 version 3.0 proposed using anchor-based head to assist anchor-free head learning, as shown in Figure 9, to improve accuracy.

2.12 YOLOv7

YOLOv7 [105] introduces trainable auxiliary architectures that can be removed or integrated during the inference stage, including YOLOR [110], the recently popular RepVGG [17], and additional auxiliary losses. Architecturally, as shown in Figure 10, YOLOv7 uses ELAN [107] to replace the CSPNet used by YOLOv4, and proposes E-ELAN to design large models. YOLOv7 also provides

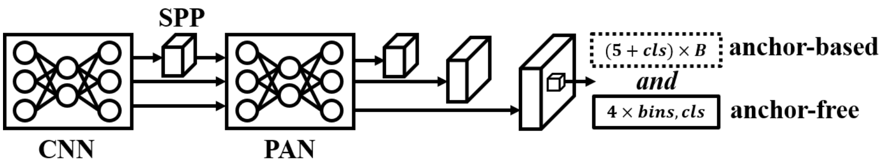


Figure 9: Architecture of YOLOv6 3.0 and YOLOv6 4.0. YOLOv6 3.0 proposed to guide the learning of anchor-free bounding box regression head with anchor-based bounding box regression head.

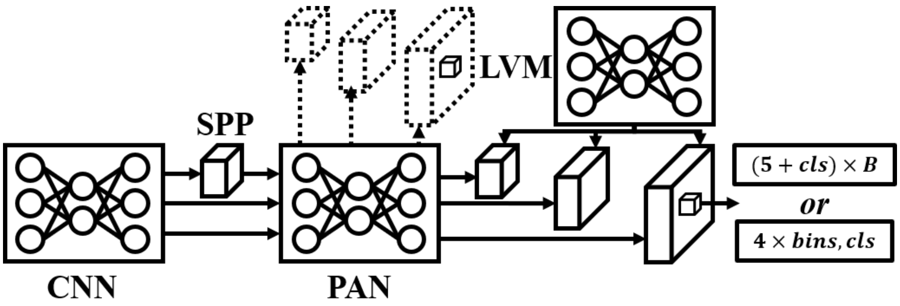


Figure 10: Architecture of YOLOv7 [105]. YOLOv7 guides learning with coarse-to-fine consistent auxliary head and inspires many consistent multiple prediction training research.

a variety of computer vision task-related models and supports anchor-based and anchor-free architectures. The features of YOLOv7 are listed below.
Make RepVGG Great Again. The reparameterization method proposed by RepVGG [17] allows simple network architectures to converge when deepening, but it cannot be effectively applied to modern popular deep network architectures. The planned RepConv technology proposed by YOLOv7 allows the reparameterization method to effectively bring gains to various residual-based and concatenation-based architectures.
Consistent Label Assignment. The auxiliary loss method used in the past will make the output targets of different branches inconsistent, which will lead to confusion and instability when performing training. In response to the maturity and popularity of the dynamic label assignment method, YOLOv7 proposed the consistent label assignment mechanism to maintain the consistency of the goals and feature learning directions of different branches.
Coarse to Fine Label Assignment. In the past, multi-stage refinement architectures, such as Cascade R-CNN [6] and HTC [9], required additional theoretical architectures to refine predictions step by step. YOLOv7 proposed the coarse-to-fine label assignment mechanism, which can directly use auxiliary

loss to guide the coarse-to-fine characteristics in the feature space, providing prediction refinement effects without changing the architecture.
Partial Auxiliary Loss. YOLOv7 allows some features to receive auxiliary information updates, and the remaining parts are still focused on target task learning. The developers of YOLOv7 found that this design has a good improvement effect on the main tasks.
Various Vision Tasks. YOLOv7 provides models including object detection, instance segmentation, and pose estimation, and has achieved real-time state-of-the-art in these tasks.

2.13 DAMO-YOLO

DAMO-YOLO [117] have proposed improved methods in terms of backbone architecture, feature integration, prediction head, and label assignment. The block diagram of DAMO-YOLO is shown in Figure 11. Its features are summarized as follows.
MAE-NAS. DAMO-YOLO uses MAE-NAS [92] to search CSPNet and ELAN to achieve a more efficient architecture.
Efficient GFPN. DAMO-YOLO disassembled the queen-fusion of GFPN [53] and retained the fusion layers of the trade-off that is designed for achieving the best speed and accuracy, so as to combine it with ELAN and design RepGFPN.
ZeroHead. DAMO-YOLO simplifies the complex decoupled head into a feature projection layer.
AlignedOTA. DAMO-YOLO proposed aligned OTA to solve the misalignment problem of classification prediction, regression prediction, and dynamic label assignment.

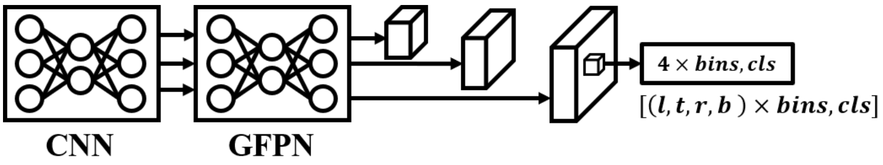


Figure 11: Architecture of DAMO-YOLO. DAMO-YOLO proposed changing the feature integration architecture to GFPN.

2.14 YOLOv8

YOLOv8 [37] is a refactored version of YOLOv5 [26], which updates the way the overall API is used and makes a lot of underlying code optimizations. It architecturally changes YOLOv7's ELAN, plus additional residual connection, while its decoder is the same as YOLOv6 2.0. It is not so much a new

YOLO version as it is a technology integration platform, and it basically integrates the APIs of multiple downstream tasks and connects them in series. YOLOv8 r2.0 [39] integrates the latest technologies such as YOLOv9 and YOLO World [13]. Its most recent version is YOLOv8 r3.0 [41], which is also known as YOLO11. It architecturally updates to combination of YOLOv9's Dark-ELAN and CSP-ELAN, and CSPNet and YOLOv10's PSA to achieve better speed-accuracy trade-off. Because program modification and API usage are not very intuitive, many developers have not yet switched to this platform. But for professional users, the performance improvements brought by optimization of the underlying program code have also attracted many R&D teams to use it. In what follows are two special features of YOLOv8:

Code Optimization. The optimization of the underlying program code released by YOLOv8 has brought about 30% improvement in training performance.

API for Down-stream Applications. YOLOv8 also provides a simple API to connect the detection model with various downstream tasks, such as segment anything, instance segmentation, pose estimation, multiple object tracking, etc.

2.15 YOLO-NAS

YOLO-NAS [93] did not reveal too many technical details. It mainly uses its own AutoNAC NAS to design the quantization friendly architecture and uses a multi-stage training process, including pre-training on Object365, COCO Pseudo-Labeled data, Knowledge Distillation (KD), and Distribution Focal Loss (DFL).

2.16 Gold-YOLO

Gold-YOLO [103] The overall architecture of Gold-YOLO is similar to that of YOLOv6 3.0. The main design is that the Gather-and-Distribute mechanism replaces PAN in the architecture, and masked image modeling is pre-trained during the training process.

Gather-and-Distribute Mechanism. The main architecture of Gather-and-Distribute is shown in Figure 12. It mainly collects features from each layer through two gather-and-distribute modules and integrates them into global features using transformers. The integrated global features will be distributed to the low-level and high-level layers respectively, and the distribution method uses the information injection module to integrate the global features with the features that have been distributed to layers.

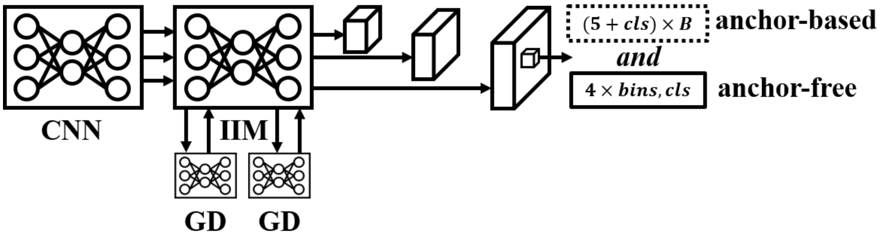


Figure 12: Architecture of Gold-YOLO. Gold-YOLO proposed adding global features to the feature integration architecture.

2.17 YOLOv9

YOLOv9 [109] proposed an important trustworthy technology – Programmable Gradient Information (PGI), whose architecture is shown in Figure 13. The design architecture in the figure can enhance the interpretability, robustness, and versatility of the model. The design of PGI is to use the concepts of reversible architecture and multi-level information to maximize the original data that the model can retain and the information needed to complete the target tasks. YOLOv9 extended ELAN to G-ELAN and used it to show how PGI can achieve excellent accuracy, stability and inference speed on models with low number of parameters. Several outstanding features of YOLOv9 are described below.

Auxiliary Reversible Branch. PGI exploits the properties of reversible architecture to solve the information bottleneck problem in deep neural networks. This is completely different from the general-purpose reversible architecture which simply maximizes the information to be retained. What PGI uses is to share the information retained by reversible architecture with the main branch in the form of auxiliary information. On the premise of retaining the information required for the target task, retain as much information as possible from the original data.

Multi-level Auxiliary Information. PGI proposed the concept of multi-level auxiliary information so that each layer of the main branch features retains the information required for all task objectives as much as possible. This can avoid the problem that past methods tend to lose important information at the shallow level, which in turn leads to the inability to obtain sufficient information at the deep level.

Generalize to Down-stream Tasks. Because PGI can maximize the retention of original data information, models trained by PGI achieve more robust performance in small datasets, transfer learning, multi-task learning, and adaptation to new downstream tasks.

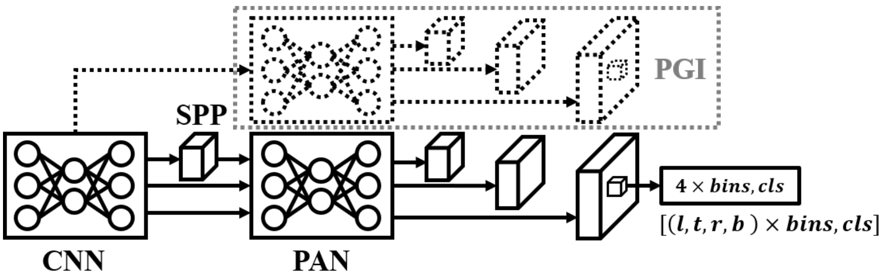


Figure 13: Architecture of YOLOv9. YOLOv9 proposed using auxiliary branches to help learning.

Generalize to Various Architectures. PGI can also be applied to other architectures, such as conventional CNN, depth-wise convolutional CNN, transformer, and different types of computer vision methods, such as anchor-based, anchor-free, post-processing free, etc. Therefore, PGI has absolutely superior versatility.

2.18 YOLOv10

The overall architecture of YOLOv10 [102] is shown in Figure 14, which is similar to YOLOv6 3.0, but the transformer-based module is added to enhance the extraction of global features. They changed the dual head to one-to-many and one-to-one matching, respectively. This change allows YOLO to do without post-processing likes the DETR-based method, and can directly obtain end-to-end object detection results. Next, we introduce some of the distinct features of YOLOv10.

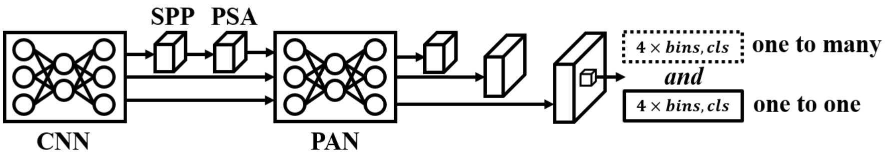


Figure 14: Architecture of YOLOv10 [102]. YOLOv10 proposed to use one-to-many prediction head to guide the learning of one-to-one prediction head.

Dual Label Assignment. Use the label assignment method similar to DATE [12], and add stop gradient operation to the one-to-one branch.
NMS-free Object Detection. The design of one-to-one matching mechanism enables the prediction process without relying on NMS for post-processing.

Rank-guided Block Design. Proposed to use rank to determine which stages use conventional convolution and which stages use depth-wise convolution.
Partial Self-attention. YOLOv10 combined CSPNet and Transformer and proposed the self-attention module.

2.19 Comparison of different YOLOs

The architecture of the entire YOLO series is summarized in Table 2. YOLOv1 innovatively proposed the first one-stage object detection architecture and was also the pioneer of anchor-free object detection. YOLOv3 and YOLOv4 respectively integrated the most advanced technologies at he time, allowing subsequent YOLO series versions to follow their main architecture. YOLOv3 designed ResNet-based backbone and FPN-based neck, while YOLOv4 designed CSPNet-based backbone and PAN-based neck. Gaussian YOLOv3 proposed the earliest and very successful distribution-based prediction head research, while YOLOX introduced FCOS's anchor-free head into the YOLO mechanism, and thus promoted the research of the YOLO series in the anchor-free direction. PP-YOLOE further used TOOD's distribution-based prediction head to enhance the design of anchor-free head, and thus became the subsequent mainstream prediction head, while YOLOv7 designed ELAN-based backbone and thus became today's mainstream architecture. In addition, YOLOv7 also designed the multiple consistent head training method to drive many related research. As for YOLOv10, it proposed to add the one-to-one matching mechanism to label assignment, and then designed end-to-end one-stage object detector.

Although each generation of YOLO versions continues to advance the accuracy-efficiency trad-off, it is worth noting that in addition to improvements in architecture, different versions of YOLO also seek improvements in training methods. For this reason, users need to have a deeper understanding of the nature of every method in order to choose the method that best suits them. In addition, YOLOv7 trained several different versions of YOLO systems using YOLOR's training strategy and compared them. The purpose of doing this is to allow users to more objectively observe the performance of different versions under the same conditions. Since the basic training strategies of different subsequent YOLO versions have almost not changed since YOLOR, we present the data of training different YOLOs under the same conditions in Table 3. From Table 3, we can clear see the differences in data between different versions of YOLO. However, since the inference time of TensorRT (TRT) is very sensitive to the layer number of the network, users must still choose the architecture that is most friendly to them in actual use.

In addition to the architecture, the prediction head and label assignment algorithms are also important factors that effect the object detection results. For example, when objects do not obscure each other, it is a better solution to

Table 2: Architecture of YOLO series. The **bold** font indicates changes that significantly affected subsequent versions.

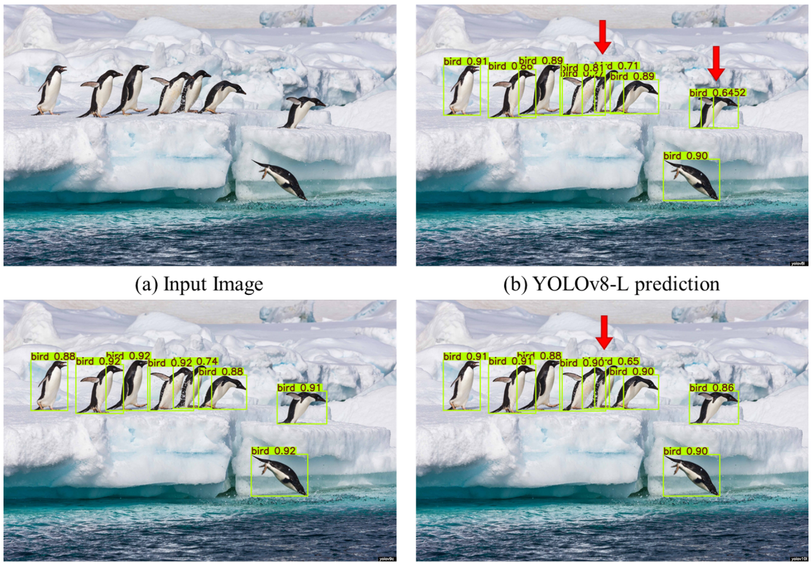
Model	Backbone	Neck	Head
YOLO (YOLO1)	**PlainNet-based**	**Fully-Connected**	**Anchor-free**
YOLO9000 (YOLOv2)	PlainNet-based	**Passthrough**	**Anchor-based**
YOLOv3	**ResNet-based**	**FPN-based**	Anchor-based
Gaussian YOLOv3	ResNet-based	FPN-based	**Anchor-based**
YOLOv4	**CSPNet-based**	**PAN-based**	Anchor-based
YOLOv5	ResNet-based	FPN-based	Anchor-based
Scaled-YOLOv4	CSPNet-based	PAN-based	Anchor-based
YOLOv5 r1.0	CSPNet-based	PAN-based	Anchor-based
PP-YOLO	ResNet-based	FPN-based	Anchor-based
YOLOv5 r2.0	CSPNet-based	PAN-based	Anchor-based
YOLOv5 r3.0	CSPNet-based	PAN-based	Anchor-based
Scaled-YOLOv4 P6	CSPNet-based	PAN-based	Anchor-based
YOLOv5 r3.1	CSPNet-based	PAN-based	Anchor-based
YOLOv5 r4.0	CSPNet-based	PAN-based	Anchor-based
PP-YOLOv2	ResNet-based	PAN-based	Anchor-based
YOLOv5 r5.0	CSPNet-based	PAN-based	Anchor-based
YOLOR	CSPNet-based	PAN-based	Anchor-based
YOLOX	CSPNet-based	PAN-based	**Anchor-free**
YOLOv5 r6.0	CSPNet-based	PAN-based	Anchor-based
PP-PicoDet	CSPNet-based	PAN-based	Anchor-free
YOLOv5 r6.1	CSPNet-based	PAN-based	Anchor-based
PP-YOLOE	CSPNet-based	PAN-based	**Anchor-free**
YOLOv6	PlainNet-based	PAN-based	Anchor-free
YOLOv7	**ELAN-based**	PAN-based	**Anchor-based**
YOLOv5 r6.2	CSPNet-based	PAN-based	Anchor-based
YOLOv6 2.0	CSPNet-based	PAN-based	Anchor-free
YOLOv7 AF	ELAN-based	PAN-based	Anchor-free
DAMO-YOLO	ELAN-based	PAN-based	Anchor-based
YOLOv5 r7.0	CSPNet-based	PAN-based	Anchor-based
YOLOv8	ELAN-based	PAN-based	Anchor-free
YOLOv6 3.0	CSPNet-based	PAN-based	Anchor-free
YOLOv6 4.0	CSPNet-based	PAN-based	Anchor-free
YOLO-NAS	ELAN-based	PAN-based	Anchor-free
Gold-YOLO	CSPNet-based	PAN-based	Anchor-free
YOLOv8 r1.0	ELAN-based	PAN-based	Anchor-free
YOLOv9	ELAN-based	PAN-based	Anchor-free
YOLOv8 r2.0'	ELAN-based	PAN-based	Anchor-free
YOLOv10	ELAN-based	PAN-based	**Anchor-free**
YOLOv8 r2.0	ELAN-based	PAN-based	Anchor-free
YOLOv8 r3.0	ELAN-based	PAN-based	Anchor-free

use label assignment methods such as YOLOv4 that can effectively improve recall. On the contrary, it is recommended to use the dynamic label assignment method developed after YOLOX. When the aspect ratio of an object is relatively fixed, it is more effective to use anchor-based prediction head. When the aspect ratio of an object is extreme, the anchor-free method is suitable.

Table 3: The performance of the main architecture of the YOLO series of papers on the COCO dataset. The **bold** font indicates Pareto optimal.

Model	#Param. (M)	FLOPs (G)	mAP (%)	T4, TRT (ms)
YOLOv4	**64.4**	**142.8**	**49.7**	–
Scaled-YOLOv4	**52.9**	**120.4**	**50.3**	–
YOLOR-CSP	**52.9**	**120.4**	**50.8**	–
YOLOv7	**36.9**	**104.7**	**51.2**	–
YOLOv5-L r6.2	46.5	109.1	49.0	–
YOLOv6-L 2.0	58.5	144.0	51.0	–
YOLOv7-L AF	43.6	130.5	**53.0**	**6.7**
YOLOv8-L	43.7	165.2	52.9	8.1
YOLOv6-L 3.0	59.6	150.7	51.8	7.9
YOLOv9-C	**25.3**	**102.1**	**53.0**	**6.1**
YOLOv9-TR	**14.1**	**67.5**	**53.1**	**5.9**
YOLOv10-B	19.1	92.0	52.5	**5.7**
YOLOv10-L	24.4	120.3	**53.2**	7.2
YOLOv8-L r3.0	25.3	86.9	**53.4**	6.2

As for YOLOv10 which proposed using one-to-one prediction head, it is easy to generate duplicated prediction boxes for the same object, so appropriate post-processing still needed. In Figure 15, we show prediction results and comparisons for the latest three YOLO versions.



(a) Input Image

(b) YOLOv8-L prediction

(c) YOLOv9-C prediction

(c) YOLOv10-L prediction

Figure 15: Prediction of latest versions of YOLO.

3 Impact of YOLO series

The YOLO series of algorithms have the characteristics of (1) relatively simple frame and (2) relatively easy deployment. In what follows we will describe these characteristics in detail.

3.1 Simpler

Simpler Frame. Based on the most forward-looking research on DeepMultiBox [19] and OverFeat [88], YOLO proposed a new way of one-stage object detection, and this new approach influenced many subsequent computer vision research. Before YOLO series was proposed, the tasks that originally required deep learning to perform dense prediction mainly included pixel-level tasks such as semantic segmentation and optical flow estimation. As for object detection, pose estimation and other instance-level tasks, most of them are split into multiple sub-tasks and predicted in the cascade way. After YOLO was proposed, many algorithms that originally needed to use multi-stage and bottom-up methods were suddenly converted to end-to-end, top-down, and one-stage methods. Examples of this sort include pose estimation and facial landmark detection that directly predict the bounding box and the relative positions of anchor points in the bounding box, as well as multi-object tracking that simultaneously detects and extracts re-identification features.
Simpler Deployment. YOLO does not use specially structured modules, so it is very easy to be deployed on a variety of computing devices. However, there are many special designs such as receptive field module and attention mechanism etc., which are very helpful in improving the accuracy of object detection, but it is not easy to design them into universal and simple modules. YOLO converts these special modules into modules with simple structures through clever design. For example, YOLOv3 proposed to use max-pooling with multiple resolutions to sweep the feature map with a stride of 1 to improve the SPP layer that is originally limited to a fixed input size, and the ASPP layer that requires dilated convolution. The above approach can greatly enhance the model's multi-resolution and global perception capabilities. As for YOLOv4, it proposed to use a convolution layer to replace the small network containing various pooling layers and fully connected layers in the attention module. The entire model of YOLOv4-tiny, YOLOv6 and YOLOv7 was even improved to the point where it only needs to be composed using 1×1 convolution, 3×3 convolution, and max-pooling. In addition, Darknet developed in C language by Joseph Redmon allows the training and inference process of YOLO without relying on additional software packages. The above-mentioned friendly deployment to existing equipment makes the YOLO series widely used in various practical systems.

3.2 Better

In addition to being lightweight and easy-to-use as described in the previous section, the YOLO series models also have some more advanced functions, such as better training techniques, better model scalability, and better model generalizability. Below we describe these functions in detail.

Better Training Technique. The training technology proposed by YOLO series models is not only more advanced but also complementary to the most advanced training technology currently available. Many studies in the past have mostly verified the proposed method on a foundation method, such as ResNet [47] and ViT [18] for image recognition, or faster R-CNN [85] and DETR [7] for object detection. However, most studies ignore whether the proposed methods are compatible with the current state-of-the-art methods and complement each other to promote the overall progress of the field. Since YOLOv2, the YOLO series has considered compatibility with the most advanced technologies when designing, and at the same time proposed new methods that can complement these technologies. In YOLOv3, YOLOv4, and PP-YOLO series of models, developers also try to analyze technologies that cannot be compatible with each other. This kind of attitude has an important guiding role for subsequent developers.

Better Model Scalability. The YOLO series models do not require special settings when performing model scaling. Scaled-YOLOv4 and YOLOv7 proposed some guidelines for model scaling, while YOLOv5 follows EfficientDet's model scaling method. These scaling methods are directly integrated into the framework, and this kind of design allows users to obtain stable and great performance no matter how they adjust the hyperparameters of model scaling.

Better Model Generalizability. The methods proposed by the YOLO series can be applied to many fields. For example, the concept of using prediction and ground truth to calculate metric proposed by YOLOv1 has been widely used in various soft label generation methods, while the method of using K-means as the initial anchor proposed by YOLOv2 has been extended to the pose estimation field. The WorldTree group softmax method proposed by YOLO is transformed into dealing with the imbalanced data distribution problem of long-tailed learning. SimOTA proposed by YOLOX is used as the basic method for various dynamic label assignment, and the hybrid label assignment method proposed by YOLOv7 is also widely used. The methods proposed by YOLO series are also applicable to a variety of architectures, for example, the CSPNet used by YOLOv4 not only shows excellent results on CNN, but has also been proven to work well with architectures such as Transformer [100], Graph Neural Networks [87], Spiking Neural Networks [96], and MAMBA [43]. The subsequent ELAN used by YOLOv7 has also been rapidly applied in various computer vision fields.

3.3 Faster

Faster Architecture. Another feature of the YOLO series is its very fast inference speed, mainly because its architecture is designed for the actual inference speed of the hardware. The designers of YOLOv3 found that even a simple 1×1 convolution and 3×3 convolution combined architecture, although it has a lower computational load, does not necessarily mean that it has an advantage in inference speed. Therefore, they designed DarkNet for real-time object detection. As for the designers of Scaled-YOLOv4, they referred to research including ShuffleNetv2 [70] and HarDNet [8], and further analyzed the criteria that need to be considered for high inference speed architecture for different levels of devices from edge to cloud. To achieve the same purpose, the developers of scaled-YOLOv4 designed Fully CSPOSANet and CSPDarkNet. As for the developers of YOLOv6, they used the efficient RepVGG as the backbone, while the designers of DAMO-YOLO used NAS technology to directly search for efficient architectures in CSPNet and ELAN.

3.4 Stronger

Stronger Adaptability. The YOLO series has gained great progress and response in the open source community. The training method integrated by Darknet and PyTorch YOLOv3 allows YOLO series to train object detectors without relying on ImageNet's pre-trained model. Due to the above reasons, the YOLO series can be easily applied to data in different domains without relying on a large number of training models corresponding to the domain. The above advantages enable the YOLO series to be widely used in various application domains. In addition, the YOLO series can also be easily applied to different datasets. For example, PyTorch YOLOv3 proposes to use evolutionary algorithms to automatically search for hyperparameters, which can be applied to different datasets. In addition, the improved anchor-free-based YOLO from YOLOX to PP-YOLOE allows YOLO to rely on fewer hyperparameters during training and can be used more widely in various application domains.
Stronger Capability. The YOLO series has excellent performance in a variety of computer vision tasks. For example, after being widely used in the field of real-time object detection, many other computer vision models based on YOLO have been developed, including YOLACT [3] instance segmentation model, JDE [113] multiple object tracking, and so on. Taking YOLOR as an example, it began to combine multiple tasks into the same model for prediction. It can perform image recognition, object detection, and multi-object tracking at the same time, and significantly improve the effect of multi-task joint learning. On the same task, YOLOv5 trains image recognition and object detection models separately. In addition, YOLOv7 also demonstrated outstanding performance in a variety of computer vision domains. At that time, it became the most

advanced method for real-time object detection, instance segmentation, and pose estimation. On the same issue, YOLOv8 additionally integrates tasks such as rotating object detection and pose estimation. In addition, YOLOv9 further combines YOLOv7 and YOLOR to extend the multi-task model to the visual-language domain.
Stronger Versatility. Since object detection is a necessary starting step for many practical applications, and as a top object detection method, YOLO's design is very suitable for matching with various downstream task models. In this regard, the design of PP-YOLO series is particularly outstanding, and this series can provide an integrated system for dozens of downstream tasks including face analysis, license plate recognition, multi-object tracking, traffic statistics, behavior analysis, etc.

4 YOLO for various computer vision tasks

The YOLO series systems have been widely used in many fields. In this section, we will introduce YOLO's representative works in other computer vision fields and explain the new designs either in architecture or methods completed by these representative works in order to achieve real-time performance.

4.1 Multiple Object Tracking

In the past, deep learning-based multiple object tracking related algorithms, such as Deep-SORT [114], needed to crop the detected object area from the original image after detecting objects, and then capture features through additional networks for tracking. ROLO [44] proposes objects directly detected by YOLO, and uses LSTM [50] for single object tracking. They proposed to use multiple LSTM to design MOLO and then perform multiple object tracking. JDE [113] proposes to output the re-ID features for object tracking while detecting objects. However, JDE's multi-scale dense prediction re-ID feature requires a large amount of calculations. In addition, a set of re-ID features will match multiple anchors, making it easy to confuse IDs. CSTrack [60] further combines JDE and FairMOT [123], and after integrating multi-scale features, only outputs re-ID features at one scale. This can achieve more accurate multi-object tracking effects.

4.2 Instance Segmentation

In the past, most instance segmentation prediction was performed separately for each detected object, so more complex segmentation network is required. YOLACT [3] and YOLACT++ [4] decompose the instance segmentation process into two steps, namely prototypes and coefficients, and only need to predict coefficients to use these prototypes to form the output instance segmentation results. Using the above method can greatly reduce the amount of operations required when instance segmentation is executed. YOLACTEdge [64] then pushes instance segmentation further to the video domain. The concept of using FeatFlowNet greatly reduces the number of features extracted by the backbone.

Another way to reduce the computation of instance segmentation prediction is to express binary mask in other ways, such as expressing mask in the form of polygon or polar coordinates. Although this expression method will cause some distortion, it can express the mask of the object in very few dimensions. Insta-YOLO [74] and Poly YOLO [52] are two examples to use the polygon form to predict the result of instance segmentation.

4.3 Automated Driving

YOLO series is also widely used in visual perception tasks in self-driving scenarios. YOLOP [115] and YOLOPv2 [45] respectively use CSPNet and ELAN as the main architecture for object detection, and therefore can be used for area detection and lane prediction. HybridNet [101], YOLOPv3 [121], and YOLOPX [122] are also modified by different versions of YOLO and perform self-driving tasks.

4.4 Human Pose Estimation

Human pose estimation can be viewed as additional spatial attributes for predicting object detection targets. Since keypoints do not necessarily fall in grids, additional decoder design is required. KAPAO [72] divides human pose estimation into human pose object and keypoint object expressions for prediction and combination. YOLO-Pose [71] directly predicts the regression value of the key relative to the center of the grid, and then execute human pose estimation. The above design can achieve pretty good results.

4.5 3D Object Detection

There are also some studies that generalize YOLO series from 2D to 3D. In addition to ComplexYOLO [91] which combines images and LIDAR as input, and Expandable YOLO [94] which uses RGB-D images as input, there is also YOLO 6D [97] and YOLO 3D [86] which simply use images as input.

4.6 Video Perception

YOLO series, which performs extremely well in real-time object detection in images, will naturally be applied to the video domain. Among them, YOLOV [89] and YOLOV++ [90] can be applied to video object detection. Alternatively, stream YOLO [118] can be used with streaming perception.

4.7 Face Detection

Face detection is one of the most popular subfields among the various possible application domains of object detection. The face detection models designed based on YOLO [11, 80, 120] also performs quite well in this field.

4.8 Image Segmentation

Due to the real-time and high-performance characteristics of YOLO, it has also begun to be combined with many foundation models and applied to new computer vision tasks. Fast-SAM [125] combines YOLO with SAM [55] and applies it to the general image segmentation task. The above combination can greatly improve the inference speed of the task model.

4.9 Open Vocabulary Object Detection

YOLO is also used in conjunction with visual language foundation models. Examples of this sort include YOLO-world [13] and Open-YOLO 3D [5], which combine YOLO and CLIP [81] methods and can be used to perform 2D and 3D open vocabulary object detection respectively.

4.10 Combine With Other Architecture

YOLO also demonstrates compatibility with a variety of deep neural network architectures. Architectures of this sort include ViT [75–78, 124], MAMBA [112], SNN [54, 69], GNN [10, 42], and KAN [15]. They can all be effectively combined with YOLO.

4.11 Summarization

Observing the development of YOLO series and thier applications in different fields, we summarize the possible future development trends of YOLO. First, various services integrated based on YOLO object detection. Examples of this sort are PP-YOLO and YOLOv8. They used YOLO as a real-time detector for various objects, and then used multi-object tracking methods to conduct more in-depth value-added analysis. Second, a multi-modal multi-task model designed with the YOLO architecture as the main body. Since YOLOR started working on multi-modal and multi-tasking YOLO, YOLOv7 integrates instance segmentation and pose estimation into the main YOLO version for the first time. By YOLOv9, multi-task models including visual-language tasks have also been developed more and more complete. We believe that real-time multi-modal multi-tasking based on YOLO will have further development in the future. The third is the combination of YOLO with large language models and foundation models. The combination of YOLO and SAM in FastSAM and the combination of YOLO and CLIP in YOLO-World are pioneers of this type of work. Due to the attention brought by these studies, it is conceivable that combining YOLO with large language models will be a direction worth exploring in the future.

5 Conclusions

In this article, we introduce the evolution of the YOLO series over the years, review these technologies from the perspective of modern object detection technology, and point out the key contributions they made at each stage. We analyze YOLO's influence on the field of modern computer vision from aspects such as ease of use, accuracy improvement, speed improvement, and versatility in various fields. Finally, we introduce the YOLO-related models in various fields. The purpose is that through this review article, readers can not only be inspired by the development of the YOLO series, but also better understand how to develop various real-time computer vision methods. We also hope to provide readers an idea of the different tasks YOLO can be used for and possible future directions.

References

[1] AlexeyAB, "darknet", 2019, https://github.com/AlexeyAB/darknet.

[2] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection", *arXiv preprint arXiv:2004.10934*, 2020.

[3] D. Bolya, C. Zhou, F. Xiao, and Y. J. Lee, "YOLACT: Real-time instance segmentation", in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2019, 9157–66.

[4] D. Bolya, C. Zhou, F. Xiao, and Y. J. Lee, "YOLACT++ Better Real-Time Instance Segmentation", *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2022.

[5] M. E. A. Boudjoghra, A. Dai, J. Lahoud, H. Cholakkal, R. M. Anwer, S. Khan, and F. S. Khan, "Open-YOLO 3D: Towards Fast and Accurate Open-Vocabulary 3D Instance Segmentation", *arXiv preprint arXiv:2406.02548*, 2024.

[6] Z. Cai and N. Vasconcelos, "Cascade R-CNN: Delving into high quality object detection", in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, 6154–62.

[7] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers", in *Proceedings of the European conference on computer vision (ECCV)*, 2020, 213–29.

[8] P. Chao, C.-Y. Kao, Y.-S. Ruan, C.-H. Huang, and Y.-L. Lin, "HarDNet: A low memory traffic network", in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2019, 3552–61.

[9] K. Chen, J. Pang, J. Wang, Y. Xiong, X. Li, S. Sun, W. Feng, Z. Liu, J. Shi, W. Ouyang, *et al.*, "Hybrid task cascade for instance segmentation", in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, 4974–83.

[10] P. Chen, Y. Wang, and H. Liu, "GCN-YOLO: YOLO Based on Graph Convolutional Network for SAR Vehicle Target Detection", *IEEE Geoscience and Remote Sensing Letters*, 2024.

[11] W. Chen, H. Huang, S. Peng, C. Zhou, and C. Zhang, "YOLO-Face: a real-time face detector", *The Visual Computer*, 37, 2021, 805–13.

[12] Y. Chen, Q. Chen, Q. Hu, and J. Cheng, "DATE: Dual assignment for end-to-end fully convolutional object detection", *arXiv preprint arXiv:2211.13859*, 2022.

[13] T. Cheng, L. Song, Y. Ge, W. Liu, X. Wang, and Y. Shan, "YOLO-World: Real-time open-vocabulary object detection", in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024, 16901–11.

[14] J. Choi, D. Chun, H. Kim, and H.-J. Lee, "Gaussian YOLOv3: An accurate and fast object detector using localization uncertainty for autonomous driving", in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2019, 502–11.

[15] danielsyahputra, "KAN-YOLO", 2024, https://github.com/danielsyahputra/ultralytics.

[16] X. Ding, H. Chen, X. Zhang, K. Huang, J. Han, and G. Ding, "Re-parameterizing your optimizers rather than architectures", in *The International Conference on Learning Representations (ICLR)*, 2023.

[17] X. Ding, X. Zhang, N. Ma, J. Han, G. Ding, and J. Sun, "RepVGG: Making VGG-style ConvNets great again", in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, 13733–42.

[18] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, *et al.*, "An image is worth 16x16 words: Transformers for image recognition at

scale", in *The International Conference on Learning Representations (ICLR)*, 2021.

[19] D. Erhan, C. Szegedy, A. Toshev, and D. Anguelov, "Scalable object detection using deep neural networks", in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014, 2147–54.

[20] C. Feng, Y. Zhong, Y. Gao, M. R. Scott, and W. Huang, "TOOD: Task-aligned one-stage object detection", in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, 3490–9.

[21] Z. Ge, S. Liu, Z. Li, O. Yoshie, and J. Sun, "OTA: Optimal transport assignment for object detection", in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, 303–12.

[22] Z. Ge, S. Liu, F. Wang, Z. Li, and J. Sun, "YOLOX: Exceeding YOLO series in 2021", *arXiv preprint arXiv:2107.08430*, 2021.

[23] R. Girshick, "Fast R-CNN", in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015, 1440–8.

[24] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation", in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014, 580–7.

[25] J. Glenn, "YOLOv3 PyTorch", 2019, https://github.com/ultralytics/yolov3.

[26] J. Glenn, "YOLOv5 (2020.05)", 2020, https://github.com/ultralytics/yolov5.

[27] J. Glenn, "YOLOv5 release v1.0", 2020, https://github.com/ultralytics/yolov5/releases/tag/v1.0.

[28] J. Glenn, "YOLOv5 release v2.0", 2020, https://github.com/ultralytics/yolov5/releases/tag/v2.0.

[29] J. Glenn, "YOLOv5 release v3.0", 2020, https://github.com/ultralytics/yolov5/releases/tag/v3.0.

[30] J. Glenn, "YOLOv5 release v3.1", 2020, https://github.com/ultralytics/yolov5/releases/tag/v3.1.

[31] J. Glenn, "YOLOv5 release v4.0", 2021, https://github.com/ultralytics/yolov5/releases/tag/v4.0.

[32] J. Glenn, "YOLOv5 release v5.0", 2021, https://github.com/ultralytics/yolov5/releases/tag/v5.0.

[33] J. Glenn, "YOLOv5 release v6.0", 2021, https://github.com/ultralytics/yolov5/releases/tag/v6.0.

[34] J. Glenn, "YOLOv5 release v6.1", 2022, https://github.com/ultralytics/yolov5/releases/tag/v6.1.

[35] J. Glenn, "YOLOv5 release v6.2", 2022, https://github.com/ultralytics/yolov5/releases/tag/v6.2.

[36] J. Glenn, "YOLOv5 release v7.0", 2022, https://github.com/ultralytics/yolov5/releases/tag/v7.0.
[37] J. Glenn, "YOLOv8 (2023.01)", 2023, https://github.com/ultralytics/ultralytics.
[38] J. Glenn, "YOLOv8 release v8.1.0", 2024, https://github.com/ultralytics/ultralytics/releases/tag/v8.1.0.
[39] J. Glenn, "YOLOv8 release v8.2.0", 2024, https://github.com/ultralytics/ultralytics/releases/tag/v8.2.0.
[40] J. Glenn, "YOLOv8 release v8.2.0"', 2024, https://github.com/ultralytics/ultralytics/releases.
[41] J. Glenn, "YOLOv8 release v8.3.0", 2024, https://github.com/ultralytics/ultralytics/releases/tag/v8.3.0.
[42] M. Gong, R. Liu, and V. K. Asari, "YOLO-based GNN for multi-person pose estimation", in *Pattern Recognition and Tracking XXXV*, Vol. 13040, 2024, 124–31.
[43] A. Gu and T. Dao, "MAMBA: Linear-time sequence modeling with selective state spaces", *arXiv preprint arXiv:2312.00752*, 2023.
[44] Guanghan, "ROLO", 2016, https://github.com/Guanghan/ROLO.
[45] C. Han, Q. Zhao, S. Zhang, Y. Chen, Z. Zhang, and J. Yuan, "YOLOPv2: Better, faster, stronger for panoptic driving perception", *arXiv preprint arXiv:2208.11434*, 2022.
[46] K. Han, Y. Wang, Q. Tian, J. Guo, C. Xu, and C. Xu, "GhostNet: More features from cheap operations", in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, 1580–9.
[47] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition", in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, 770–8.
[48] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition", *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 37(9), 2015, 1904–16.
[49] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, and B. Scholkopf, "Support vector machines", *IEEE Intelligent Systems and their Applications*, 13(4), 1998, 18–28.
[50] S. Hochreiter and J. Schmidhuber, "Long short-term memory", *Neural Computation*, 9(8), 1997, 1735–80.
[51] X. Huang, X. Wang, W. Lv, X. Bai, X. Long, K. Deng, Q. Dang, S. Han, Q. Liu, X. Hu, *et al.*, "PP-YOLOv2: A practical object detector", *arXiv preprint arXiv:2104.10419*, 2021.

[52] P. Hurtik, V. Molek, J. Hula, M. Vajgl, P. Vlasanek, and T. Nejezchleba, "Poly-YOLO: higher speed, more precise detection and instance segmentation for YOLOv3", *Neural Computing and Applications*, 34(10), 2022, 8275–90.

[53] Y. Jiang, Z. Tan, J. Wang, X. Sun, M. Lin, and H. Li, "GiraffeDet: A heavy-neck paradigm for object detection", in *The International Conference on Learning Representations (ICLR)*, 2022.

[54] S. Kim, S. Park, B. Na, and S. Yoon, "Spiking-YOLO: spiking neural network for energy-efficient object detection", in *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, Vol. 34, No. 07, 2020, 11270–7.

[55] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, *et al.*, "Segment anything", in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2023, 4015–26.

[56] C. Li, L. Li, Y. Geng, H. Jiang, M. Cheng, B. Zhang, Z. Ke, X. Xu, and X. Chu, "YOLOv6 v3.0: A full-scale reloading", *arXiv preprint arXiv:2301.05586*, 2023.

[57] C. Li, L. Li, H. Jiang, K. Weng, Y. Geng, L. Li, Z. Ke, Q. Li, M. Cheng, W. Nie, *et al.*, "YOLOv6: A single-stage object detection framework for industrial applications", *arXiv preprint arXiv:2209.02976*, 2022.

[58] C. Li, B. Zhang, L. Li, L. Li, Y. Geng, M. Cheng, X. Xiaoming, X. Chu, and X. Wei, "YOLOv6: A single-stage object detection framework for industrial applications", 2023.

[59] 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", *Advances in Neural Information Processing Systems (NeurIPS)*, 33, 2020, 21002–12.

[60] C. Liang, Z. Zhang, X. Zhou, B. Li, S. Zhu, and W. Hu, "Rethinking the competition between detection and reid in multiobject tracking", *IEEE Transactions on Image Processing (TIP)*, 31, 2022, 3182–96.

[61] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection", in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, 2117–25.

[62] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection", in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017, 2980–8.

[63] E. Linder-Norén, "YOLOv3 PyTorch", 2018, https://github.com/eriklindernoren/PyTorch-YOLOv3.

[64] H. Liu, R. A. R. Soto, F. Xiao, and Y. J. Lee, "YOLACTEdge: Real-time instance segmentation on the edge", in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021, 9579–85.

[65] R. Liu, J. Lehman, P. Molino, F. Petroski Such, E. Frank, A. Sergeev, and J. Yosinski, "An intriguing failing of convolutional neural networks and the CoordConv solution", *Advances in Neural Information Processing Systems (NeurIPS)*, 31, 2018.

[66] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path aggregation network for instance segmentation", in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, 8759–68.

[67] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector", in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016, 21–37.

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

[69] X. Luo, M. Yao, Y. Chou, B. Xu, and G. Li, "Integer-Valued Training and Spike-Driven Inference Spiking Neural Network for High-performance and Energy-efficient Object Detection", in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2024.

[70] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, "ShuffleNet v2: Practical guidelines for efficient cnn architecture design", in *Proceedings of the European conference on computer vision (ECCV)*, 2018, 116–31.

[71] D. Maji, S. Nagori, M. Mathew, and D. Poddar, "YOLO-Pose: Enhancing YOLO for multi person pose estimation using object keypoint similarity loss", in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2022, 2637–46.

[72] W. McNally, K. Vats, A. Wong, and J. McPhee, "Rethinking keypoint representations: Modeling keypoints and poses as objects for multi-person human pose estimation", in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2022, 37–54.

[73] meituan, "YOLOv6 (2022.06)", 2022, https://github.com/meituan/YOLOv6.

[74] E. Mohamed, A. Shaker, A. El-Sallab, and M. Hadhoud, "Insta-YOLO: Real-time instance segmentation", *arXiv preprint arXiv:2102.06777*, 2021.

[75] H. Ouyang, "DEYO: DETR with YOLO for End-to-End Object Detection", *arXiv preprint arXiv:2402.16370*, 2024.

[76] H. Ouyang, "DEYO: DETR with YOLO for step-by-step object detection", *arXiv preprint arXiv:2211.06588*, 2022.

[77] H. Ouyang, "DEYOv2: Rank feature with greedy matching for end-to-end object detection", *arXiv preprint arXiv:2306.09165*, 2023.

[78] H. Ouyang, "DEYOv3: DETR with YOLO for Real-time Object Detection", *arXiv preprint arXiv:2309.11851*, 2023.

[79] pjreddie, “darknet”, 2018, https://github.com/pjreddie/darknet.

[80] D. Qi, W. Tan, Q. Yao, and J. Liu, “YOLO5Face: Why reinventing a face detector”, in *Proceedings of the European Conference on Computer Vision Workshops (ECCVW)*, 2022, 228–44.

[81] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, *et al.*, “Learning transferable visual models from natural language supervision”, in *International Conference on Machine Learning (ICML)*, 2021, 8748–63.

[82] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection”, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, 779–88.

[83] J. Redmon and A. Farhadi, “YOLO9000: better, faster, stronger”, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, 7263–71.

[84] J. Redmon and A. Farhadi, “YOLOv3: An incremental improvement”, *arXiv preprint arXiv:1804.02767*, 2018.

[85] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards real-time object detection with region proposal networks”, *Advances in Neural Information Processing Systems (NeurIPS)*, 28, 2015.

[86] ruhyadi, “YOLO3D”, 2022, https://github.com/ruhyadi/YOLO3D.

[87] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, “The graph neural network model”, *IEEE Transactions on Neural Networks*, 20(1), 2008, 61–80.

[88] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, “Overfeat: Integrated recognition, localization and detection using convolutional networks”, in *The International Conference on Learning Representations (ICLR)*, 2014.

[89] Y. Shi, N. Wang, and X. Guo, “YOLOV: Making still image object detectors great at video object detection”, in *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, Vol. 37, No. 2, 2023, 2254–62.

[90] Y. Shi, T. Zhang, and X. Guo, “Practical Video Object Detection via Feature Selection and Aggregation”, *arXiv preprint arXiv:2407.19650*, 2024.

[91] M. Simony, S. Milzy, K. Amendey, and H.-M. Gross, “Complex-YOLO: An Euler-region-proposal for real-time 3D object detection on point clouds”, in *Proceedings of the European Conference on Computer Vision Workshops (ECCVW)*, 2018.

[92] Z. Sun, M. Lin, X. Sun, Z. Tan, H. Li, and R. Jin, “MAE-Det: Revisiting maximum entropy principle in zero-shot nas for efficient object detection”, in *Proceedings of the International Conference on Machine Learning (ICML)*, 2022.

[93] super-gradients, “YOLO-NAS”, 2023, https://github.com/Deci-AI/super-gradients/blob/master/YOLONAS.md.

[94] M. Takahashi, Y. Ji, K. Umeda, and A. Moro, “Expandable YOLO: 3D object detection from RGB-D images”, in *International Conference on Research and Education in Mechatronics (REM)*, 2020, 1–5.

[95] M. Tan, R. Pang, and Q. V. Le, “EfficientDet: Scalable and efficient object detection”, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, 10781–90.

[96] A. Tavanaei, M. Ghodrati, S. R. Kheradpisheh, T. Masquelier, and A. Maida, “Deep learning in spiking neural networks”, *Neural Networks*, 111, 2019, 47–63.

[97] B. Tekin, S. N. Sinha, and P. Fua, “Real-time seamless single shot 6D object pose prediction”, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, 292–301.

[98] Z. Tian, C. Shen, H. Chen, and T. He, “FCOS: A simple and strong anchor-free object detector”, *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 44(4), 2020, 1922–33.

[99] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders, “Selective search for object recognition”, *International Iournal of Computer Vision (IJCV)*, 104, 2013, 154–71.

[100] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need”, *Advances in Neural Information Processing Systems (NeurIPS)*, 30, 2017.

[101] D. Vu, B. Ngo, and H. Phan, “HybridNets: End-to-end perception network”, *arXiv preprint arXiv:2203.09035*, 2022.

[102] A. Wang, H. Chen, L. Liu, K. Chen, Z. Lin, J. Han, and G. Ding, “YOLOv10: Real-time end-to-end object detection”, *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[103] C. Wang, W. He, Y. Nie, J. Guo, C. Liu, K. Han, and Y. Wang, “Gold-YOLO: Efficient Object Detector via Gather-and-Distribute Mechanism”, *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[104] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, “Scaled-YOLOv4: Scaling cross stage partial network”, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, 13029–38.

[105] 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”, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023, 7464–75.

[106] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2020, 390–1.

[107] C.-Y. Wang, H.-Y. M. Liao, and I.-H. Yeh, "Designing network design strategies through gradient path analysis", *Journal of Information Science and Engineering (JISE)*, 39(4), 2023, 975–95.

[108] C.-Y. Wang, H.-Y. M. Liao, I.-H. Yeh, Y.-Y. Chuang, and Y.-L. Lin, "Exploring the power of lightweight YOLOv4", in *roceedings of the IEEE International Conference on Computer Vision Workshops (ICCVW)*, 2021, 779–88.

[109] C.-Y. Wang, I.-H. Yeh, and H.-Y. M. Liao, "YOLOv9: Learning what you want to learn using programmable gradient information", in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2024.

[110] C.-Y. Wang, I.-H. Yeh, and H.-Y. M. Liao, "You only learn one representation: Unified network for multiple tasks", *Journal of Information Science and Engineering (JISE)*, 39(2), 2023, 691–709.

[111] X. Wang, R. Zhang, T. Kong, L. Li, and C. Shen, "SOLOv2: Dynamic and fast instance segmentation", *Advances in Neural Information Processing Systems (NeurIPS)*, 33, 2020, 17721–32.

[112] Z. Wang, C. Li, H. Xu, and X. Zhu, "Mamba YOLO: SSMs-Based YOLO For Object Detection", *arXiv preprint arXiv:2406.05835*, 2024.

[113] Z. Wang, L. Zheng, Y. Liu, Y. Li, and S. Wang, "Towards real-time multi-object tracking", in *Proceedings of the European conference on computer vision (ECCV)*, 2020, 107–22.

[114] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric", in *IEEE International Conference on Image Processing (ICIP)*, 2017, 3645–9.

[115] D. Wu, M.-W. Liao, W.-T. Zhang, X.-G. Wang, X. Bai, W.-Q. Cheng, and W.-Y. Liu, "YOLOP: You only look once for panoptic driving perception", *Machine Intelligence Research*, 19(6), 2022, 550–62.

[116] S. Xu, X. Wang, W. Lv, Q. Chang, C. Cui, K. Deng, G. Wang, Q. Dang, S. Wei, Y. Du, *et al.*, "PP-YOLOE: An evolved version of YOLO", *arXiv preprint arXiv:2203.16250*, 2022.

[117] X. Xu, Y. Jiang, W. Chen, Y. Huang, Y. Zhang, and X. Sun, "DAMO-YOLO: A report on real-time object detection design", *arXiv preprint arXiv:2211.15444*, 2022.

[118] J. Yang, S. Liu, Z. Li, X. Li, and J. Sun, "Real-time object detection for streaming perception", in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, 5385–95.

[119] G. Yu, Q. Chang, W. Lv, C. Xu, C. Cui, W. Ji, Q. Dang, K. Deng, G. Wang, Y. Du, *et al.*, “PP-PicoDet: A better real-time object detector on mobile devices”, *arXiv preprint arXiv:2111.00902*, 2021.

[120] Z. Yu, H. Huang, W. Chen, Y. Su, Y. Liu, and X. Wang, “YOLO-Facev2: A scale and occlusion aware face detector”, *Pattern Recognition*, 155, 2024, 110714.

[121] J. Zhan, J. Liu, Y. Wu, and C. Guo, “Multi-Task Visual Perception for Object Detection and Semantic Segmentation in Intelligent Driving”, *Remote Sensing*, 16(10), 2024, 1774.

[122] J. Zhan, Y. Luo, C. Guo, Y. Wu, J. Meng, and J. Liu, “YOLOPX: Anchor-free multi-task learning network for panoptic driving perception”, *Pattern Recognition*, 148, 2024, 110152.

[123] Y. Zhang, C. Wang, X. Wang, W. Zeng, and W. Liu, “FairMOT: On the fairness of detection and re-identification in multiple object tracking”, *International Journal of Computer Vision (IJCV)*, 129, 2021, 3069–87.

[124] Z. Zhang, X. Lu, G. Cao, Y. Yang, L. Jiao, and F. Liu, “ViT-YOLO: Transformer-based YOLO for object detection”, in *Proceedings of the IEEE International Conference on Computer Vision Workshops (ICCVW)*, 2021, 2799–808.

[125] X. Zhao, W. Ding, Y. An, Y. Du, T. Yu, M. Li, M. Tang, and J. Wang, “Fast segment anything”, *arXiv preprint arXiv:2306.12156*, 2023.