Engineering Science and Technology, an International Journal 70 (2025) 102161

Contents lists available at ScienceDirect

Engineering Science and Technology, an International Journal

journal homepage: www.elsevier.com/locate/jestch

Review

A comprehensive review on YOLO versions for object detection

Ayşe Aybilge Murat [a,*], Mustafa Servet Kiran [b]

[a] Department of Computer Engineering, Faculty of Engineering and Natural Sciences, Osmaniye Korkut Ata University, 80010 Osmaniye, Turkey
[b] Department of Computer Engineering, Faculty of Computer and Information Sciences, Konya Technical University, 42075, Konya, Turkey

ARTICLE INFO

Keywords:
Computer vision
Deep learning
Object detection
YOLO

ABSTRACT

The need for methods used for object detection has gained increasing momentum in recent years. Starting with traditional image processing techniques, this process has been facilitated by the addition of deep learning. Object detection is currently used in areas such as autonomous vehicles, disease diagnosis, robotic vision and industry. The types of systems that are predicted to be needed more and more in the age of developing technology are also increasing. In particular, YOLO (You Only Look Once), which is mostly preferred in real-time object detection, is preferred because it achieves high accuracy in a short time. This paper analyses the main versions of the YOLO algorithm since its first release. The paper systematically analyses the architectural differences between the versions of the YOLO algorithm, the strengths and weaknesses of the models and their contribution to performance. At the same time, in most of the previous studies on YOLO, a comprehensive comparison of the YOLOv9-v11 models is not presented and new architectural features are not evaluated. This review provides an in-depth analysis of the main versions from YOLOv1 to YOLOv11, including recent innovations such as NMS-free, Oriented Bounding Boxes (OBB), GELAN and PGI. This work is intended to be a useful guide for researchers and developers interested in the field.

1. Introduction

The skills and behaviours of individuals have always been a subject of research for science. The artificial systems produced have been influenced by the structure of humans. Particularly, our senses and perception abilities are noteworthy. Actions that individuals can perform effortlessly require machines to undergo extensive training and processes. Individuals can perceive quickly with their visual abilities and can detect objects. The field of computer vision has also been developed based on human visual capabilities and is designed for use in various areas. The process of extracting the desired information and making decisions by performing operations on images briefly explains computer vision [1]. Areas such as object detection, object segmentation, classification, pattern recognition, and image restoration are sub-branches of computer vision [2]. Computer vision has developed rapidly with machine learning and deep learning models [3]. Artificial intelligence techniques can perform operations on large image datasets quickly and with high accuracy.

The importance of object detection has increased for both static images and videos in proportion to the encountered problems. Speed plays a significant role in videos and real-time detection. Systems have been developed where speed is prioritized alongside accuracy criteria. Some of the preferred methods in real-time detection systems include Single Shot Detector (SSD) [4], RetinaNet [5], and YOLO. The reason for their preference is that operations can be performed through a single neural network. Although two-stage detectors achieve high accuracy, they are insufficient in terms of speed [6]. On the other hand, single-stage detectors aim for satisfactory accuracy and high speed [6]. This article examines the YOLO algorithm developed by Joseph Redmon and his colleagues in 2015. YOLO, which is widely preferred due to its results and wide range of applications, has been developed in several versions. Each new version has introduced architectural changes and various optimisations. There are several papers in the literature that examine this development of YOLO. One of these papers [7] analyses the versions of deep learning models used for object detection up to YOLOv4. The first of two similar papers [8] analysed YOLOX and YOLOR, including the YOLOv8 version, while [9] examined the same versions more superficially. Another comprehensive review was carried out on the main versions up to YOLOv8 [10,11] is another study that examines the same versions by giving examples of various usage areas. One of the recent

* Corresponding author at: Department of Computer Engineering, Faculty of Engineering and Natural Sciences, Osmaniye Korkut Ata University, Osmaniye, Turkey.
E-mail addresses: ayseaybilgemurat@osmaniye.edu.tr (A.A. Murat), mskiran@ktun.edu.tr (M.S. Kiran).

https://doi.org/10.1016/j.jestch.2025.102161
Received 15 January 2025; Received in revised form 18 June 2025; Accepted 27 July 2025
Available online 5 August 2025

studies [12] examined the YOLOv1-YOLOv10 versions and their usage areas with examples, but the architectures were not examined in depth. The most recent study on this subject [13] made a comparison including the YOLOv11 version and developed applications, but the architectures of YOLOv1-YOLOv8 versions were not analysed and a superficial comparison was presented. In the literature reviewed in this context, only one study was found in which the latest YOLOv11 version was included and the innovations it brought with it were compared from YOLOv8 to YOLOv11. A systematic and in-depth examination of the rapidly developing YOLO versions is important to fill the gaps in the literature. It has been observed that the architectural innovations and radical changes brought by the versions developed after YOLOv8 have not been sufficiently analysed. This paper aims to fill the gaps identified in previous studies by analysing all YOLOv1-YOLOv11 versions in a technical, systematic and unified comparison.

This article introduces image processing and object detection, followed by a comprehensive review of the YOLO versions. The architectures of the main versions from YOLOv1 to YOLOv11 and their developments throughout the process have been examined.

1.1. Image processing

Image processing can generally be described as a process that enables the extraction of meaningful information from images [14]. Information extraction is performed using techniques such as the sliding window method and mathematical operations on images [15]. As a result of the extracted information, outcomes such as object detection, classification, and segmentation can be achieved. Additionally, we can filter the image or remove unwanted noise. Traditional methods based on pixel processing have gradually been replaced by more advanced systems.

Faster and more effective solutions are needed when working with big data. Machine learning and deep learning have the ability to operate more easily in complex systems [16]. This allows for the desired features to be easily extracted from images. Among the most preferred deep learning models for image processing is Convolutional Neural Network (CNN) [17]. They process the image by extracting features from different layers [17]. Thus, they can achieve high accuracy in a short time with big data.

1.2. Object detection

Object detection, a subfield of computer vision, is the process of identifying specific objects using image processing techniques [18]. The location and class of the object in the image are determined. It is utilized in areas such as healthcare and industry [19–22], autonomous vehicles [23,24], and security systems [25,26]. Object detection has shown rapid development with deep learning models and algorithms [27]. By processing a large number of images, detailed information about objects can be accessed in a short time. There are two deep learning-based object detection methods: single-stage and two-stage detectors [28].

1.2.1. Single-stage and two-stage object detector

The two-stage detector consists of two different stages that generate proposal regions and perform classification [15]. Since the proposal regions detect the possible locations of objects in advance, the classification stage can undergo a more detailed detection process for the objects. This allows the detector to achieve a high accuracy rate. However, this process may extend the processing time. Examples of architectures for two-stage detectors include R-CNN [29], Fast R-CNN [30], and Faster R-CNN [31]. Although the two-stage detector achieves a satisfactory detection speed on static images, higher-speed detection is required for real-time detection. Single-stage detectors utilize a single network to detect and classify all possible locations. Since the process occurs on a single network, the detection speed is increased. Compared to two-stage detectors, single-stage detectors have a lower accuracy rate [32]. Examples of single-stage detectors include SSD [4], YOLO, RetinaNet [5], and EfficientDet [33]. This article examines the YOLO family, which achieves satisfactory results in terms of both speed and accuracy. In this paper, the YOLO family, which achieves satisfactory results in terms of both speed and accuracy and has a wide range of applications, is analysed.

2. YOLO

The YOLO algorithm, developed in 2015, has rapidly gained widespread adoption. Unlike two-stage detectors, it performs object detection on a single network. YOLO approaches object detection as a regression problem and calculates the probabilities of the objects contained within regions known as bounding boxes. It does not require a second stage for classification, enabling it to achieve high accuracy at a faster rate. Developed for real-time object detection, YOLO divides the image into an *SxS* grid and determines bounding boxes that encompass the objects. Confidence scores are computed to detect the objects within these bounding boxes. YOLOv1, YOLOv2, and YOLOv3 were developed by Redmon and colleagues, while other versions continue to be developed by different researchers. Fig. 1 illustrates the development timeline of YOLO since its initial release.

YOLO holds a significant place in real-time object detection and versions are being developed for use in a wide range of fields. One of the most commonly preferred fields is vehicle detection. YOLO has been utilized in systems for autonomous vehicles [34–36], parking systems [37], pedestrian detection [38], and license plate recognition [39,40]. It has also been employed in the detection of ships [41] and airplanes [42], making it a preferred choice in the defence industry due to its speed and accuracy. In addition to these, YOLO is widely used in security systems for detecting hazardous objects [43,44], facial recognition [45], and plays a crucial role in the early detection of fires [46].

YOLO also achieves high accuracy in detecting small objects. New models have been developed for detecting challenging objects, such as distant objects [42], medications [47] and circuit components [48].

In the field of medicine, YOLO has achieved remarkable success, being employed in cancer detection [49–51], polyp detection [52], fracture identification [53], and systems assisting experts in diagnosing skin disorders [54]. This has contributed significantly to the detection process, aiding medical professionals. YOLO is also used for sign language recognition [55,56], demonstrating its applications for the benefit of humanity across various domains.

In agriculture and industrial sectors, efforts have been made to increase efficiency. In agriculture and industrial sectors, YOLO has been utilized to enhance efficiency. Efforts have been made to detect [57–61] and classify fruits and vegetables and to automate harvest timing, aiming to optimize productivity. YOLO has been used in the packaging stages in the industrial sector [62].

To evaluate the performance of object detection algorithms in a comparable way, various evaluation criteria have been developed to measure specific data sets and model performance. The Microsoft COCO (Common Objects in Context) dataset [63], which is used in this context, is a frequently preferred dataset in computer vision systems for tasks such as object detection and object segmentation. It contains a total of 328,000 images and 2.5 million labels present in the images. The dataset consists of 91 different object categories, including vehicles, animals, and humans. Additionally, the dataset includes bounding boxes and segmentation masks for the images. Another commonly used dataset is the PASCAL VOC (Visual Object Classes) dataset [64], which contains a total of 11,530 images and consists of 20 different object categories. The images are divided into training and testing sets for model evaluation.

Various metrics are used to evaluate models in machine learning and deep learning. he confusion matrix is a tool used during the evaluation phase to display the number of correct or incorrect predictions compared to the actual results. Calculations are performed using the values obtained from this matrix to assess the model. Table 1 illustrates the confusion matrix.

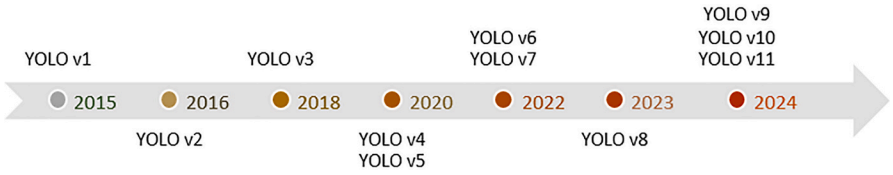

Fig. 1. YOLO timeline.

Table 1
Confusion matrix.

		Prediction Results	
		1	0
Actual Results	1	True Positive (TP)	False Negative (FN)
	0	False positive (FP)	True Negative (TN)

$$Precision = \frac{TP}{TP + FP} \tag{1}$$

$$\text{Re}call = \frac{TP}{TP + FN} \tag{2}$$

Precision and recall metrics are calculated as shown in equations (1) and (2). Precision indicates how many of the positives predicted by the model are actually positive. Recall, on the other hand, shows how many of the truly positive samples are predicted as positive. When multiple classes are to be predicted in object detection, precision values are calculated for each class. Different precision values arise for a class depending on the state of the objects. To eliminate this situation and achieve more consistent results, another metric used in object detection is Average Precision (AP) (equation (3). In this metric, a precision-recall curve is drawn for each class, and the average of the precision values in the area under the curve is referred to as AP. A higher AP value indicates better algorithm performance. The Mean Average Precision (mAP) value is obtained by taking the average of the AP values across all classes. The mAP value is obtained by averaging the AP values across all classes. Equation (4) shows how the calculation is performed. Here N is the total number of classes and AP_i is the AP value of the class.

$$AP = \int_0^1 p(r)dr \tag{3}$$

$$mAP = \frac{1}{N}\sum_{i=1}^{N} AP_i \tag{4}$$

Intersection over Union (IoU) is another important evaluation metric used in object detection. It assesses how accurately the locations of objects are predicted by measuring the similarity between the predicted and the actual bounding boxes. IoU is the ratio of the intersection of the actual bounding box and the predicted box to the union of these boxes (Fig. 2). This ratio produces a number between 0 and 1. The closer the obtained value is to 1, the more similar the predicted bounding box is to the actual bounding box, leading to the conclusion that it is a good prediction.

Non-Maximum Suppression (NMS) approach is utilized to retain the most accurate bounding boxes by eliminating irrelevant bounding boxes among the generated predictions. It removes predictions that fall below a certain threshold value among the boxes predicting the object in various sizes and regions (Fig. 3). This approach is employed in YOLO to achieve more accurate results in object detection. Object detection before and after NMS.

In addition to accuracy metrics such as IoU, Ap and mAP, we also use FLOPs, Params (M) and FPS metrics to provide a comprehensive evaluation of each YOLO version. FLOPs (floating point operations) is a metric that measures the computational complexity of the model and helps in the selection of resource-constrained hardware. It tells how many floating points are used for a single forward pass of the model during training. Models with low FLOPs require less computational power and are easier to use on edge devices. Params (M) is the number of weights learnt in the model (in the million band) and indicates the size of the model in terms of memory storage. A larger Params metric indicates more memory capacity. FPS (Frames Per Second) is the number of images the model can process per second and measures the extraction speed. It is one of the most important metrics used in real-time applications.

All metrics used are of great importance to understand the balance between models such as performance, efficiency, equipment suitability and to make fair comparisons.

2.1. YOLOv1

The original YOLO, was published as "You Only Look Once: Unified, Real-Time Object Detection," by Joseph Redmon and colleagues [65]. YOLO approached object detection as a regression problem. It detected objects by training them through a single network. YOLO primarily divides the image into an *SxS* grid. If the centre of the object to be detected corresponds to a grid cell, it detects the object. A grid cell predicts 'B'

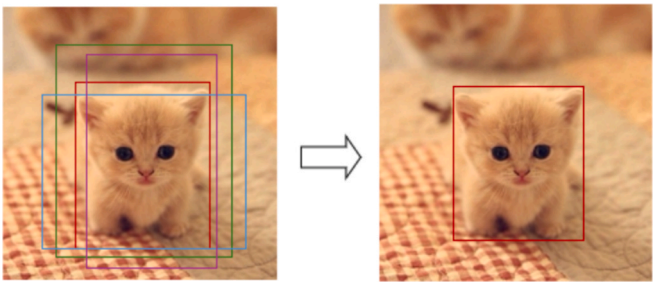

Fig. 3. Object detection before and after NMS.

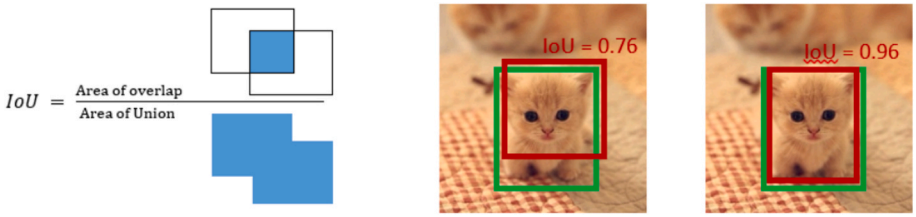

Fig. 2. Calculation of the IoU.

(bounding boxes) and confidence scores. Each bounding box predicts 5 values: *x, y, w, h,* and the confidence score. The (x, y) coordinates represent the centre of the bounding box relative to the grid cell, while w and h represent the width and height of the bounding box concerning the entire image. The confidence score represents whether there is an object in the bounding box and the IoU value. Each grid cell predicts the conditional class probability, denoted as C, for the confidence score. Using these values, a tensor of $S \times S \times (B*5+C)$ is obtained. Here the value '5' represents the 5 parameters estimated for each box. In the paper, YOLO was evaluated on the Pascal VOC dataset, where the dataset's number of classes is C = 20, the number of bounding boxes per grid cell is $B = 2$, and the input image is divided into a 7*x*7 grid. Here the output size is $7 \times 7 \times 30$.

The YOLOv1 architecture was inspired by the GoogLeNet [66] model, and the details of the architecture are shown in Fig. 4. The architecture consisted of 24 convolutional layers and 2 fully connected layers. While the initial convolutional layers were used for feature extraction, the fully connected layers were responsible for predicting the probabilities and coordinates of the output. The 1×1 convolution layer was followed by the 3×3 convolution layer, which was used to reduce the features from the previous layers.

Convolutional layers were pre-trained with the ImageNet [67] dataset at a resolution of (224×224) and subsequently, the resolution was doubled for detection. In the pre-training phase, the first 20 convolutional layers, an average pooling layer and a fully connected layer were used. The DarkNet framework was employed during these processes [68]. Leaky activation functions were used in all layers except for the final one, where a linear activation function was applied. The final layer is also the one that predicts the bounding box coordinates. The paper also presented a faster approach called Fast-YOLO, which consists of 9 convolutional layers and processes 155 frames per second compared to YOLO's 45 frames per second.

In the model's output, the sum-squared error loss function was preferred by the authors as it is easy to optimize. In the loss function, the value of $\lambda_{\text{coord}} = 5$, was set to increase the loss caused by bounding box predictions, and $\lambda_{\text{noobj}} = 5$ was set to reduce the loss from boxes without objects. Equation (5) can be explained based on three loss stages: classification loss, localization loss, and confidence loss.

$$\begin{aligned}
&\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] + \lambda_{coord} \sum_{i=0}^{S^2} \\
&\times \sum_{j=0}^{B} 1_{ij}^{obj} \left[(\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] + \sum_{i=0}^{S^2} \\
&\times \sum_{j=0}^{B} 1_{ij}^{obj} (C_i - \hat{C}_i)^2 + \lambda_{noobj} \sum_{i=0}^{S^2} \\
&\times \sum_{j=0}^{B} 1_{ij}^{noobj} (C_i - \hat{C}_i)^2 + \sum_{i=0}^{S^2} 1_i^{obj} \sum_{c \in classes} (p_i(c) - \hat{p}_i(c))^2
\end{aligned} \tag{5}$$

The first two terms in the equation represent the localization loss, which indicates how far the predicted bounding box is from the actual position. These operations are performed when there is an object in the grid cell. The two terms following the localization loss in the equation represent the confidence loss. The first term is the loss function used when there is an object in the box, while the second term is used when there is no object. The last term expresses the classification loss. It is calculated for each class when there is an object in the grid.

There are two fundamental issues with the YOLO algorithm: the limited detection of nearby objects and difficulty in generalizing across objects. The reason for its inability to detect nearby objects is the predictability of two boxes with a single class within a grid cell. The issue with generalization arises because the algorithm is not accustomed to handling objects of varying aspect ratios due to the use of bounding boxes. This leads to a decrease in its success in detecting small and very large objects. In a study on the detection and classification of breast masses [69], it was shown that masses were detected with an accuracy of 99.7 %. In the study [70], where YOLOv1 architecture was used for garbage detection, modifications were made to the architecture and comparisons with the proposed method were presented. Here, it was observed that it was difficult to detect small objects.

2.2. YOLOv2

YOLOv2 or YOLO9000 was published as "YOLO9000: Better, Faster, Stronger" by Joseph Redmon and Ali Farhadi [71]. It derived its name from its ability to detect 9000 categories. The categories were obtained by combining the ImageNet and COCO datasets. In this way, object detection and classification tasks were trained together. To demonstrate how classification is performed on common objects, labelled images were used in object detection. In this section, a hierarchical

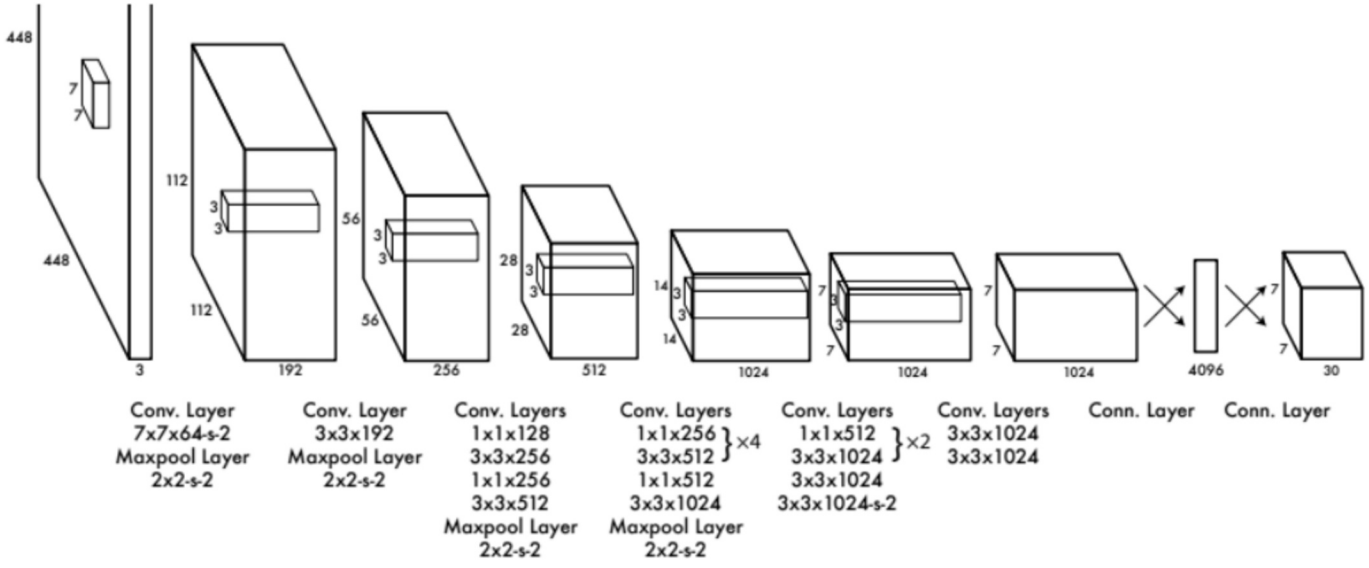

Fig. 4. YOLOv1 architecture [65].

classification tree was created using WordNet [72]. This allows for separate definitions for 'dog' and 'poodle'.

The DarkNet-19 architecture, inspired by the VGG [73] and Network in Network [74] models for YOLOv2, is shown in Table 2. DarkNet-19 consisted of 19 convolutional layers and 5 max-pooling layers. To reduce features, 3 × 3 and 1 × 1 convolutional layers were used. After each pooling layer, the number of channels was doubled.

This version of YOLO included several innovations in addition to its architecture. Batch normalization (BN) [75] was applied at each convolutional layer. This process accelerates the training process, enables the network to generalize better and is used to achieve consistent results. In this way, a mAP increases of 2 % was observed.

Unlike YOLOv1, images were fine-tuned at a resolution of 448 × 448 on ImageNet for 10 epochs. In this way, a *high-resolution classification network* was created and the mAP value increased by 4 %.

Fully connected layers were completely removed from the architecture, and *anchor boxes* were used for bounding box prediction. Anchor boxes are a set of boxes that are predetermined and of different sizes. A set of anchor boxes is defined for each grid cell. However, anchor boxes bring two issues: the manual selection of *box sizes* and the instability of the model during *direct location prediction*. While the model can learn to adjust the sizes of the boxes, determining the anchor box sizes using k-means clustering can make learning easier. The model can learn to adjust the size of the boxes, but the use of k-means clustering to determine the size of the anchor boxes at this stage may facilitate learning. For direct location prediction, the coordinates of the boxes are predicted relative to the position of the grid cell. There are boxes containing 5 predicted values $(t_x, t_y, t_w, t_h, t_o)$ for each grid cell (where the value of t_o is equal to the value of P_c). To find the coordinates of the bounding boxes from the center, we use the width and height values. Fig. 5 illustrates how the coordinates of the boxes are determined. Here, C_x and C_y represent the distance to the top left corner of the grid cell. The width and height values of the anchor box are P_w and P_h. Here the sigmoid function is used to normalise the values of t_x and t_y between 0–1. This allows the centre of the bounding box to remain inside the cell. $\sigma(t_x)$ and $\sigma(t_y)$ give the relative position of the bounding box with respect to the cell. The centre of the box b_x and b_y are transformed into coordinates normalised to the whole image by the given equations. Here the relative position and the cell position are summed to obtain the actual position.

Table 2
Darknet-19 for YOLOv2 architecture.

Type	Filters	Size/Stride	Output
Convolutional	32	3 × 3	224 × 224
Maxpool		2 × 2/2	112 × 112
Convolutional	64	3 × 3	112 × 112
Maxpool		2 × 2/2	56 × 56
Convolutional	128	3 × 3	56 × 56
Convolutional	64	1 × 1	56 × 56
Convolutional	128	3 × 3	56 × 56
Maxpool		2 × 2/2	28 × 28
Convolutional	256	3 × 3	28 × 28
Convolutional	128	1 × 1	28 × 28
Convolutional	256	3 × 3	28 × 28
Maxpool		2 × 2/2	14 × 14
Convolutional	512	3 × 3	14 × 14
Convolutional	256	1 × 1	14 × 14
Convolutional	512	3 × 3	14 × 14
Convolutional	256	1 × 1	14 × 14
Convolutional	512	3 × 3	14 × 14
Maxpool		2 × 2/2	7 × 7
Convolutional	1024	3 × 3	7 × 7
Convolutional	512	1 × 1	7 × 7
Convolutional	1024	3 × 3	7 × 7
Convolutional	512	1 × 1	7 × 7
Convolutional	1024	3 × 3	7 × 7
Convolutional	1000	1 × 1	7 × 7
Avgpool		Global	1000
Softmax			

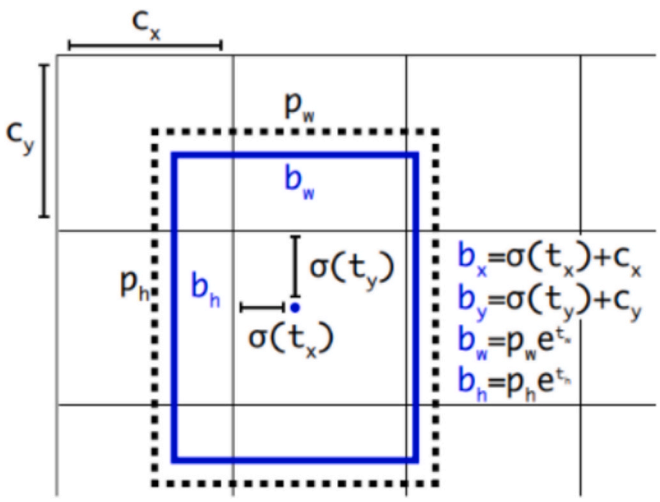

Fig. 5. Bounding box prediction [71].

The values b_w and b_h, which represent the actual width and height values, are obtained by multiplying the position of the anchor box and the estimated heights (for positivity) expanded by the exponential function. As a result, the mAP has increased by 5 %.

YOLOv2 added a transition layer with a 26 × 26 feature map to predict smaller objects while using a 13 × 13 feature map for larger objects. This transition layer, added to obtain *finer-grained features*, combines high and low-resolution features. As a result, the 26 × 26 × 512 feature map is transformed into a 13 × 13 × 2048 feature map, and through this combination, a size of 13 × 13 × 3072 is achieved, with the mAP value increasing by 1 %.

Since fully connected layers are not used, YOLOv2 can work with images of different sizes. A multiple of 32 was randomly selected for the size in every 10 iterations during the model training. The smallest size determined was 320*x*320, and the largest size was 608 × 608.

Performance has improved slightly with the anchor boxes approach, but accuracy rates are still not high. It still has limitations for the detection of small objects and performance is negatively affected in complex scenes. YOLOv2 was used for medical mask detection and an AP value of 81 % was obtained [76]. In another study [77], colour recognition was performed on objects such as traffic signs, and it was emphasised that performance may be degraded for low resolution or small objects.

2.3. YOLOv3

YOLOv3 was published by Joseph Redmon and Ali Farhadi under the title "YOLOv3: An Incremental Improvement" [78]. Unlike YOLOv2, the DarkNet-53 architecture was utilized (Table 3). This architecture was not as lightweight as previous versions but was faster than the algorithms being compared. YOLOv3 has also been successful in detecting small objects.

The network consisted of a total of 53 convolutional layers, with max-pooling layers removed and residual connection layers added after the convolutional layers. Features were extracted using the Feature Pyramid Network (FPN) architecture [79], which was inspired by these layers and upsampling. Additionally, 53 more convolutional layers were added on top of this backbone for object detection.

YOLOv3, similar to FPN, predicts boxes at 3 different grid sizes. It generates feature maps for the 3 boxes it predicts. The first of these feature maps has a grid structure of 13 × 13, as in YOLOv2. The second feature map has a grid structure of 26 × 26. The last feature map has a grid size of 52 × 52. The first and second feature maps are combined by upsampling. Subsequently, the result of this combination is merged with the last feature map by upsampling. Thanks to the outputs of different

Table 3
Darknet-53 for YOLOv3 architecture.

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3/2	128 × 128
	Convolutional	32	1 × 1	
1x	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3/2	64 × 64
	Convolutional	64	1 × 1	
2x	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3/2	32 × 32
	Convolutional	128	1 × 1	
8x	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3/2	16 × 16
	Convolutional	256	1 × 1	
8x	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3/2	8 × 8
	Convolutional	512	1 × 1	
4x	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

sizes obtained, the detection of small, medium, and large objects has been facilitated.

The version, like 'YOLOv2′, predicts bounding boxes with 4 values. The difference with YOLOv3 is that it uses logistic regression when predicting these boxes. It obtains an objectness score for each bounding box. If the previous bounding box overlaps more with the object than any other bounding box, it receives a value of 1. The system assigns a box for each base object. If a box is not assigned, a classification loss occurs.

Each box predicts the object inside the bounding box using a multi-labelled classifier. In this section, instead of using SoftMax, which stated that each box belonged to a single class, an independent logistic classifier was employed. Additionally, binary cross-entropy loss was utilized for classification during training.

Furthermore, the last convolutional layer added to the feature extractor in architecture is a three-dimensional tensor that computes the bounding box, class predictions, and objectness. With the COCO dataset, 3 boxes have been predicted at each scale. In the $N \times N \times [3^*(4+1+80)]$ tensor, '80′ represents the number of classes, '1′ represents the objectness prediction, '4′ represents the bounding box coordinate values, and the size of the *NxN* feature map is represented.

It has achieved better accuracy by using an FPN structure for multi-scale detection and a deeper architecture than previous versions. However, using a deeper architecture increased the size and complexity of the model. This resulted in a decrease in the extraction speed compared to previous versions. At the same time, the error rate is still high in complex scenes, and the model continues to struggle with very small objects. In a study on electronic components [80], it was found that complex scenes and a large number of components affect performance. Another study compared three versions of YOLO [54] on skin cancer diagnosis and reported that YOLOv2 was more successful. Another study for pill detection compared three different object detection methods and YOLOv3 achieved 80.69 % mAP [47].

2.4. YOLOv4

The YOLOv4, was published as "YOLOv4: Optimal Speed and Accuracy of Object Detection," by Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark [81]. The new features utilized in this version have introduced a more complex architecture. The authors aimed to achieve both fast and highly accurate results by experimenting with multiple techniques throughout the process.

Before YOLOv4, previous versions mentioned a backbone and head structure; however, subsequent versions incorporated a neck structure into this architecture for further enhancement. The architecture of the mentioned modern object detector, consisting of three parts, is illustrated in Fig. 6. Here, the backbone section is responsible for the feature extraction task from the input image. The neck section is responsible for merging feature maps of different sizes derived from the backbone. The aim is to improve the feature extraction process. The head section is responsible for performing object detection operations based on the features obtained from the previous two sections.

The development of YOLOv4 was based on two methods: Bag of Freebies (BoF) and Bag of Specials (BoS). These methods, which change the training strategy or training cost to improve model performance, utilize additional techniques. BoF is related to data augmentation and improves performance without increasing cost, while BoS involves post-processing techniques that, although slightly increasing the cost, enhance performance. Both methods were applied with different techniques for the backbone and neck. Initially, the architectures CSPDarknet53, CSPResNext50 [82], and EfficientNet-B3 [83] were compared to the backbone. The Cross Stage Partial Connections (CSP) here is an architectural design that allows the network to connect with different layers [84] and aids in more effective feature extraction. As a result of comparisons, CSPDarknet53, which includes 29 convolutional layers with a 3 × 3 filter and a 725 × 725 area, was selected for use in the backbone. Additionally, the Mish activation [85] was incorporated into the architecture. A Spatial Pyramid Pooling (SPP) [86] layer was added on top of CSPDarknet53 for use in the neck section. This layer increased the receptive field without reducing the speed of operation. Increasing the receiver area allows for better visualization of the object and its surroundings, thereby enabling more effective detection of objects of different sizes [87]. Additionally, a modified Path Aggregation Network (PANet) [88] was used instead of the FPN [79] used in YOLOv3 in the neck section. PANet is a bidirectional parameter aggregation method across different backbone sections at various detection levels [88]. Furthermore, a modified Spatial Attention Module (SAM) [89] block was utilized to assist in precise feature extraction and to highlight important regions in the images. The structures of PANet and SAM are shown in Fig. 7 and Fig. 8.

In addition to classical methods such as cropping, flipping and rotation, a mosaic data augmentation method combining four images was also used to augment the data [81]. This method has enabled the perception of objects outside their normal contexts. To reduce overfitting, DropBlock [90] regularization has been utilized. For use in the detector, CIoU loss [91] and CmBN (Cross mini-Batch Normalization) [75,81] were incorporated. Genetic algorithms have been employed to find the optimal hyperparameters. In addition to these methods, the SAT technique with 2 forward–backward stages was used to increase the robustness of the model [81]. The network modifies the training data using its own predictions, retrains with new images, and becomes resistant to its own errors. For the head section of YOLOv4, the anchor box structure of YOLOv3 was utilized. Fig. 9 shows the architecture of yolov4.

Adopting a different structure from previous architecture, YOLOv4 provides a balance of speed and accuracy by incorporating advanced features such as CSP, SPP and PANet. However, a large and complex architecture has been created, which may cause compatibility problems with end devices. In addition, although the data augmentation techniques used in the model have alleviated the generalization problem, the detection rate still decreases in complex scenes. Detection of small objects is still disadvantageous compared to large objects. In one study [92], YOLOv4 used for human detection from thermal images achieved 91 % MAP. In another study [93], it was preferred for the detection of harmful objects in the agricultural field, and it was stated that the efficiency may decrease in complex backgrounds. Similarly, it was observed in [94] that accuracy can be affected under dense vegetation and various

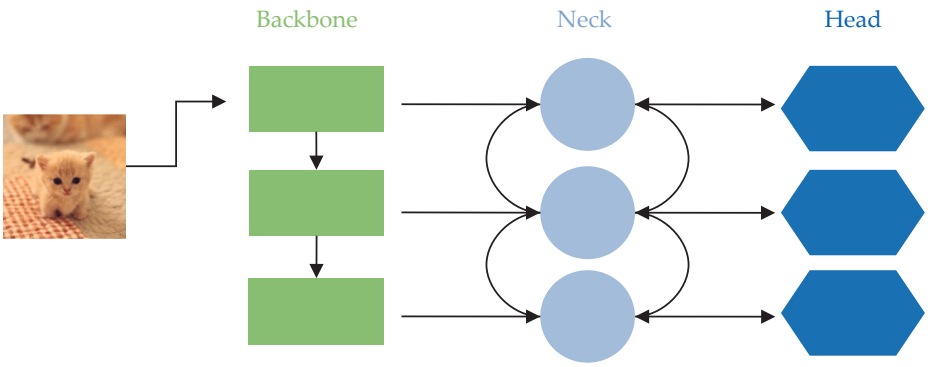

Fig. 6. Modern object detection architecture.

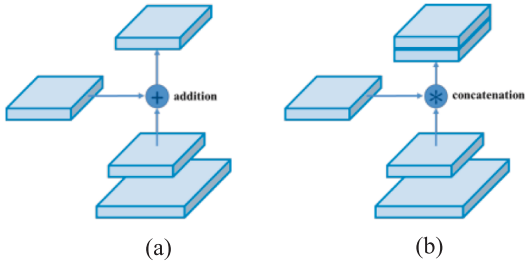

Fig. 7. a) PANet (b) modified PANet [81].

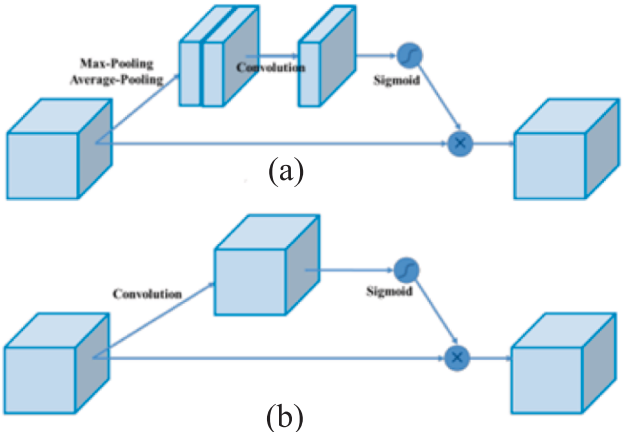

Fig. 8. (a) SAM (b) Modified SAM [81].

lighting conditions for the detection of endangered animals in wildlife. It has been reported that YOLOv4, which is used for ship fire detection, may be affected by the variable conditions of the marine environment in real-world problems [95].

2.5. YOLOv5

YOLOv5 [96] was published by Glenn Jocher, the founder of Ultralytics, but there is no official paper for it. In this version, instead of directly using the DarkNet architecture, the PyTorch library was utilized. This provided a lighter and more customisable structure. YOLOv5 aimed to be faster and easier to train than previous versions.

The backbone structure was created by implementing a modified CSPDarknet53 on PyTorch. A Spatial Pyramid Pooling Fast (SPPF) layer was added on top of the CSPDarknet53 in the neck section. SPPF is a faster version of the SPP layer used in YOLOv4. It combines feature maps at different scales to enhance the network's efficiency [86,97]. Upsampling layers were utilized to increase the resolution of the feature maps here. Additionally, a modified CSP-PANet structure was incorporated into the neck section. Batch Normalization (BN) and SiLU activation functions were employed in all convolutional layers used in the network. The YOLOv5 architecture is illustrated in Fig. 10.

Various data augmentation techniques were utilized in YOLOv5 to prevent overfitting and to achieve better generalization. These techniques include mosaic augmentation, copy-paste [98], HSV augmentation, random horizontal flipping, and MixUp [97,99]. The use of these techniques has been effective in the perception of small objects. For the detector, the CIoU loss function was utilized, while genetic algorithms were applied for hyperparameter selection. In the head part of the model, the anchor head structure used in YOLOv3 and YOLOv4 was preferred.

Various training strategies have been implemented to enhance the model's performance. During the training process, scaling the input images to different sizes positively affected the model's generalization ability. Similarly, a strategy called Exponential Moving Average (EMA), which takes the averages of the parameters used in previous steps, has also been employed to reduce the generalization error. Another strategy known as AutoAnchor statistically controls and optimizes the anchor boxes and ground truth boxes [97].

YOLOv5 contains 5 basic versions: YOLOv5n, YOLOv5s, YOLOv5m, YOLOv5l and YOLOv5x. These versions differ according to the hardware requirement, depth and width of the layers. Users can choose a version according to their own requirements.

The integration of YOLOv5 into end devices is simplified and a flexible and easy to access model is provided. Anchor-based architecture has limitations. While small objects are successfully detected in large-scale models such as YOLOv5x, object losses increase as the scale of the model decreases. In an industrial study [100], YOLOv3 and YOLOv5 were compared, and values of 80 % and 70 % were obtained respectively. A mAP value of 75.2 % was obtained in a study that wanted to detect small objects by combining YOLOv5 with transformers [101]. YOLOv5 was also preferred in end devices such as robots [102], and it was stated that the YOLOv5s model obtained more optimal results in the application. An improved YOLOv5 was used for defect detection in wind turbine blades and a mAP of 95.1 % was obtained [103].

2.6. YOLOv6

YOLOv6, titled "YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications" was published by Meituan Incorporates [104]. While it inherits certain features from previous versions, YOLOv6 introduced distinct backbone architecture. The authors explored various methods to improve performance and accuracy, surpassing previous versions. Additionally, they succeeded in enhancing inference speed with only a minimal reduction in enhancing inference speed with only a minimal reduction in efficiency. YOLOv6 is optimized for use in industrial applications.

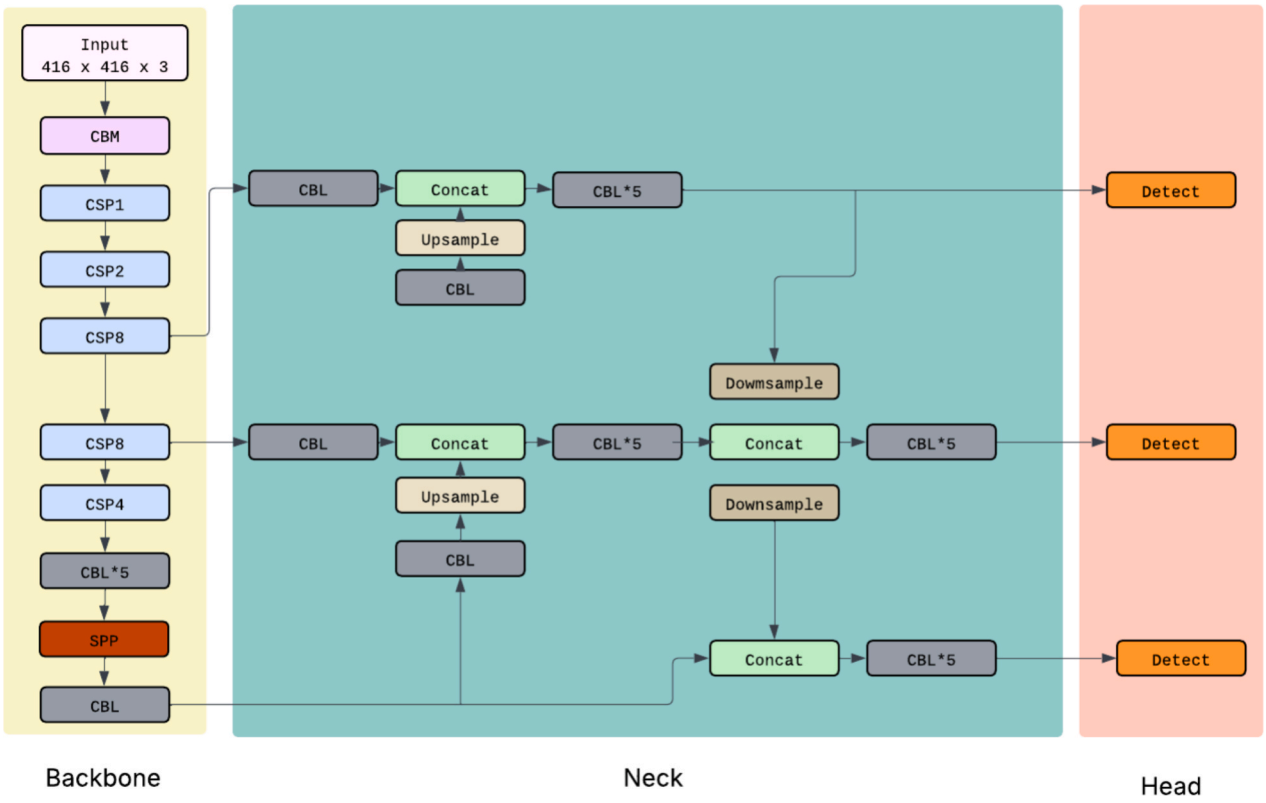

Fig. 9. YOLOv4 architecture.

The network design of YOLOv6 differs significantly from previous versions. In the backbone architecture, RepVGG [105] or CSPStackRep [84,104] blocks were utilized depending on the size of the model. RepVGG, or re-parameterizable VGG, employs a complex structure during training, enabling the model to learn more effectively. Once the training is completed, the complex structure is converted into a simpler network, allowing for faster inference. Since the RepBlock [105] structure poses challenges in terms of computational cost and scalability in large networks, it is preferred for smaller networks. For larger models, CSPStackRep blocks were employed [104]. During inference, each RepBlock is transformed into a stack, RepConv, consisting of a 3x3 convolutional layer and ReLu functions. In large models, due to scaling issues, RepVGG is not used on its own, but in the form of CSPStackRep blocks, an efficient version of the CSP block [104]. This block consists of binary RepConvs after ReLu activation functions and 3 1x1 convolution layers. This backbone architecture is referred to as EfficientRep.

In the neck of the model, the PANet structure from previous versions had been transformed into RepPAN, which was enhanced using either RepBlock or CSPStackRep. The RepPAN structure was used here by adjusting its width and depth according to the scale of the model. RepBlock for small models and CSPStackRep block for large models are integrated into the RepPAN structure. In the head of the model, an efficient decoupled head with a hybrid channel strategy was utilized. This structure was based on the detection head of YOLOv5; however, unlike YOLOv5, the classification and localization branches had been merged into a unified head. A hybrid-channel strategy is used to provide an efficient structure. The structure of the head scales with the structure of the spine and neck, reducing the computational cost. At the same time, the sensor is anchor-free, which is an approach where the distance from the anchor point to the four sides of the bounding boxes is estimated in advance. Furthermore, an anchor-free structure was adopted for the detector, providing a dynamic and flexible framework. The overall architecture of the model is illustrated in Fig. 11.

In YOLOv6, the authors tested the loss functions used in previous versions for this release as well. VariFocal Loss [106] was employed for classification loss, while SIoU [107] or GIoU [108] loss was preferred for regression loss. DFL (Distribution Focal Loss) [109], which is preferred for probability loss, is used to improve location estimation accuracy in the object detection phase, but it is used in YOLOv6 m and l architectures because it increases the computational load in small models. In addition, the TAL [110] (Task Alignment Learning) approach was used instead of the IoU approach to assign object labels, improving the quality of the label assignment and alleviating the misalignment problem. Additionally, the Task Alignment Learning (TAL) [110] approach was adopted to enhance the quality of label assignment and mitigate the issue of misalignment. Data augmentation techniques were also utilized to improve detection capabilities and overall performance.

Post-training quantization and self-distillation techniques were leveraged to enhance both speed and accuracy. The RepOptimizer [111] and channel-based distillation [112] methods were preferred here, and the information was learnt more efficiently during training. Furthermore, extending the training period contributed to improved efficiency for YOLOv6. The model is divided into eight sub-versions, ranging from YOLOv6n to YOLOv6-L6, to meet various requirements. The model achieved 57.2 % AP at 29 FPS.

Thanks to the architectural changes made in YOLOv6, the model has become compatible for low-resource end devices. Although the anchor-free structure eliminates limitations compared to previous versions, incorrect detection of objects in low contrast areas can be performed. In addition, the dataset size is expected to be large to achieve high accuracy in the model. In a study [113], different scaled versions of YOLOv6 were used to detect the clustering behavior of chickens, and it was reported that the limited size of the data set affected the performance when objects overlapped. In another study [114], a comparison of YOLOv5 and YOLOv6 was presented in the context of plant leaf disease detection, YOLOv5 achieved a faster result with a mAP value of 63.5 %.

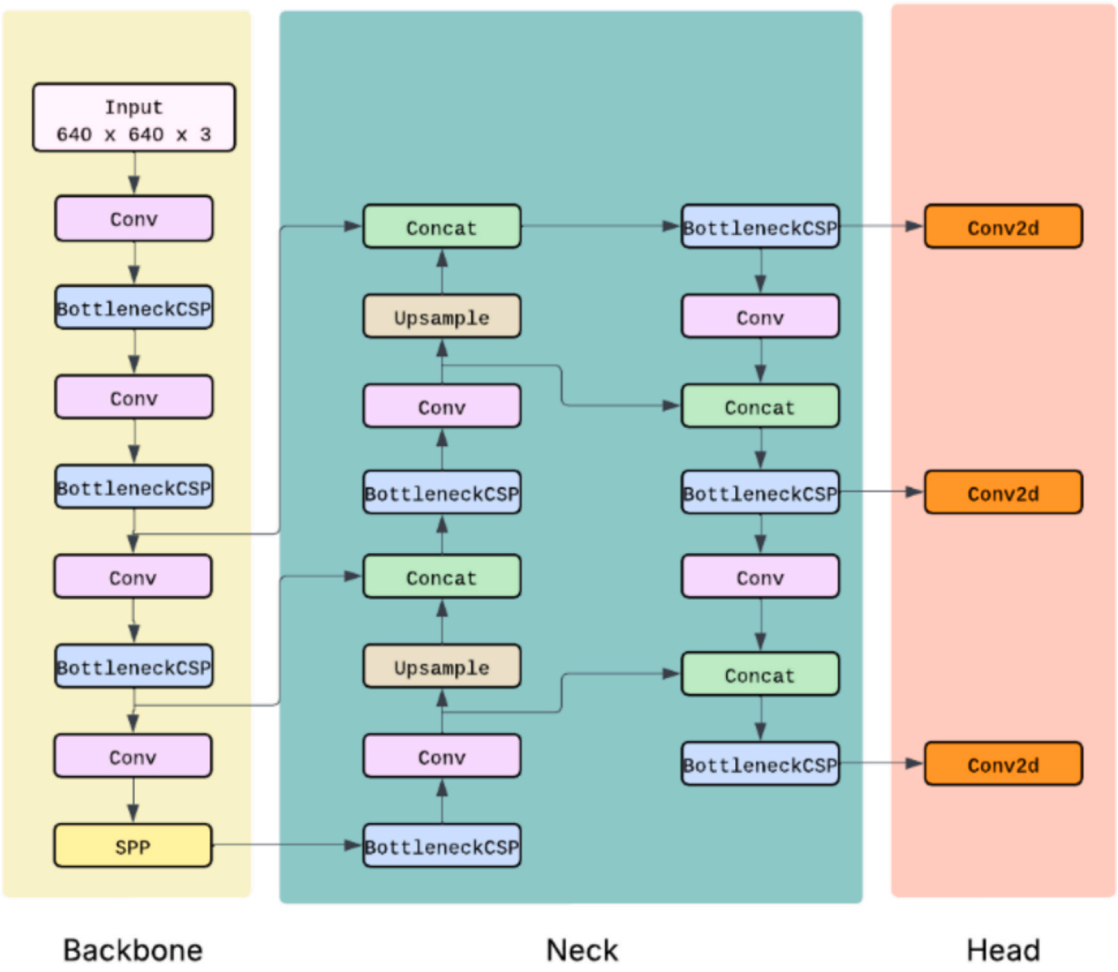

Fig. 10. YOLOv5 architecture.

2.7. YOLOv7

YOLOv7, titled "YOLOv7: Trainable Bag-of-Freebies Sets New State-of-the-Art for Real-Time Object Detectors," was published by the authors of YOLOv4 [115]. At the time of its release, it outperformed all object detectors operating between 5 FPS and 160 FPS [115]. The authors proposed an architectural design featuring a trainable Bag-of-Freebies (BoF) structure and novel methods, which enhanced training efficiency without increasing inference costs.

The architecture of YOLOv7 can be analysed in two parts: the E-ELAN structure and model scaling.

The E-ELAN (Extended Efficient Layer Aggregation Networks) [115] structure used in the architecture is an extended version of ELAN [116]. ELAN enables the aggregation of features from different layers, allowing more effective learning through a deeper network. While building a deep network, it manages the shortest and longest gradient paths. E-ELAN utilizes cardinalities of expansion, shuffle, and merge to ensure continuous learning of the network without disrupting the original gradient path. Although it modifies the architecture of the computational block, it does not entirely alter the architecture of the transition layer. Fig. 12 illustrates the ELAN and E-ELAN structures.

In YOLOv7, a concatenation-based model scaling strategy was employed for model scaling. The purpose of model scaling is to adjust certain attributes and create models at different scales to optimize inference speeds. Factors such as resolution, depth, and width are utilized during the scaling process. It is crucial to ensure that the model is scaled uniformly across all factors while preserving its structural integrity. To achieve this, YOLOv7 employs a concatenation-based strategy that allows each factor to be proportionally scaled in the same manner.

Another approach employed in YOLOv7, the trainable BoF structure, introduces several innovations. The re-parameterizable model structure presented in YOLOv6 has been modified and adapted for this version. Predicting that the RepConv [105] structure may not function effectively in all architectures, the authors designed a planned re-parameterization convolution. They removed the identity connection from the RepConv structure and named this new design RepConvN.

The head structure of YOLOv7 also differs from previous versions. YOLOv7 features a decoupled-head design, consisting of a primary head and an auxiliary head. While the primary head is responsible for the final output, the auxiliary head serves as a training assistant. This allows the primary head to focus more on information that has not yet been learned. The head structure is integrated with a dynamic label assignment system that assigns soft labels, enabling more effective learning for the model. The versions of YOLOv7 include YOLOv7-tiny, YOLOv7x, YOLOv7-W6, YOLOv7-E6, and YOLOv7-D6.

Additionally, specific details are introduced for the BoF. A topology is utilized where the batch normalization (BN) layer is directly connected to the convolutional layer. This allows the mean and variance of the batch normalization to be merged with the bias and weights of the convolutional layer during the inference phase [115]. The EMA model from YOLOv5 is also used in the final inference model. Fig. 13 shows the architecture of yolov7.

Although satisfactory results can be obtained for small data sets with modifications to the architecture, there may be difficulties in customizing the architecture due to the complexity of the architecture. In

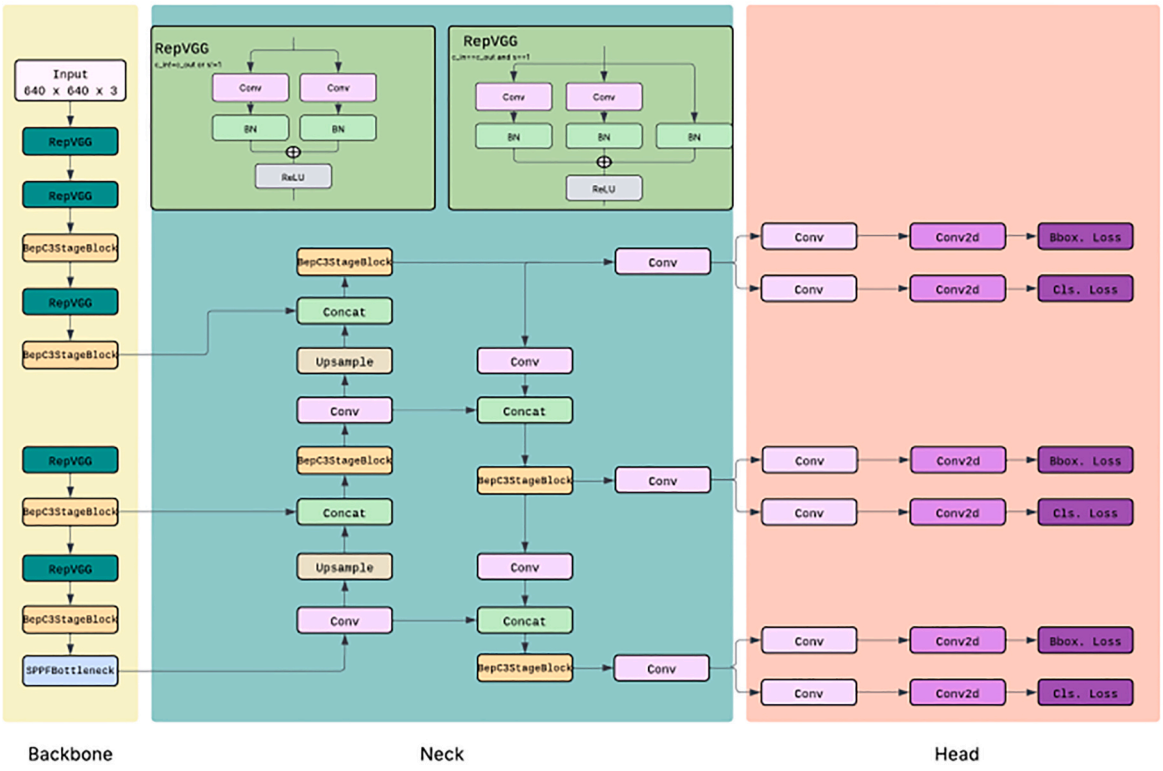

Fig. 11. YOLOv6 architecture.

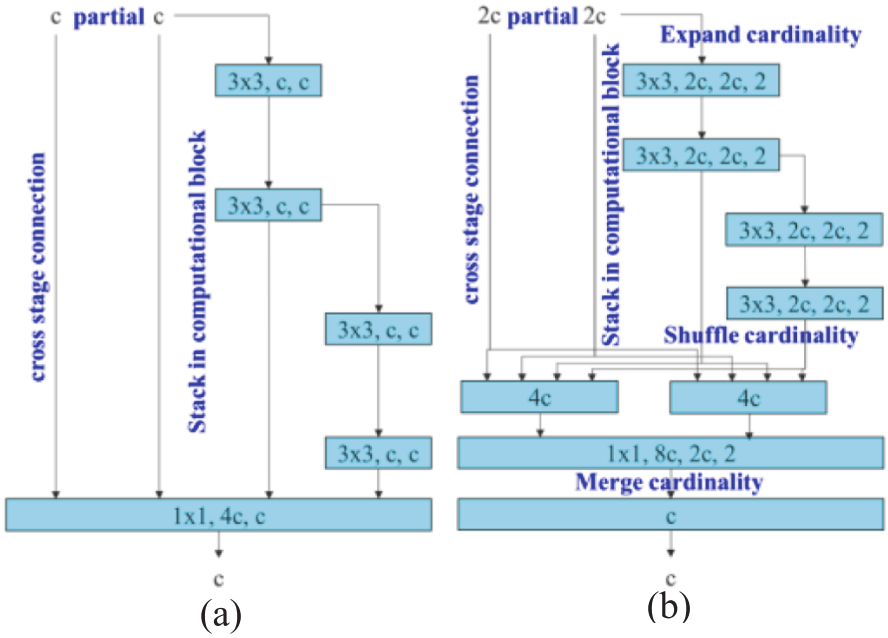

Fig. 12. (a) ELAN (b) E-ELAN.

addition, using large-scale models on end devices may not give effective results. In the safety helmet detection task [117], versions 5, 6, and 7 of YOLO were compared and YOLOv7 achieved better results. Similarly, another study made comparisons for road defect detection and YOLOv7 showed high performance with 79 % mAP value [118]. An example of its use on a mobile device is available in the literature [119]. In a paper [120] comparing models for the application of small-scale versions of fetal heart structures, YOLOv7 achieved an accuracy of 82.1 % with a limited data set.

2.8. YOLOv8

YOLOv8 [121] was released by Ultralytics, the developer of YOLOv5, and does not have an official paper. It surpasses all previous versions in terms of speed and accuracy. YOLOv8 features a new backbone and an anchor-free, decoupled head.

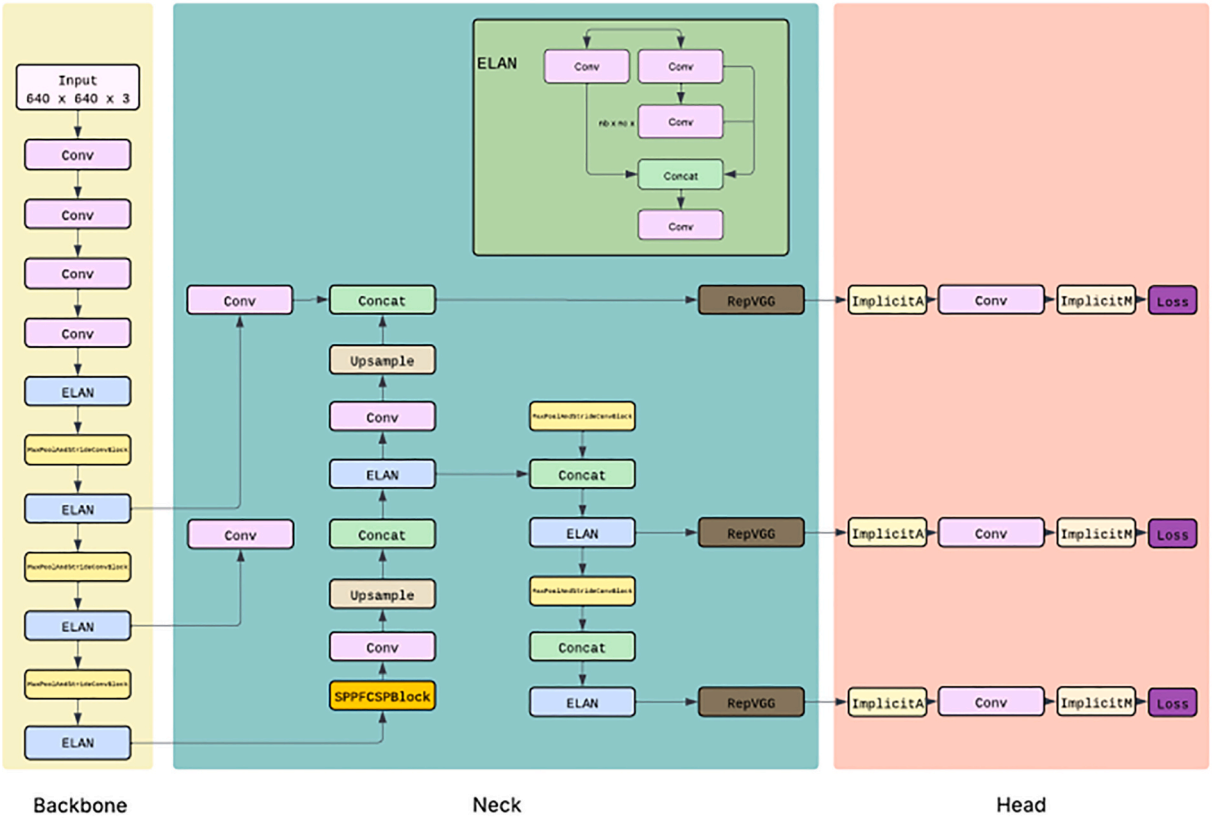

Fig. 13. YOLOv7 architecture.

YOLOv8 features a new backbone similar to YOLOv5, with a few modifications. The key change was the transformation of the three convolutional layers in the CSP structure into the C2f module. C2f, or Cross-Stage Partial Bottleneck with Two Convolutions, combined outputs from different stages, aiming to improve accuracy. The SPPF block used in YOLOv5 is also used here and the spatial context is learnt by combining maps of different scales. In this context, the model provides a deep and fast structure, and at the same time it is lightweight and can be easily operated on edge devices. The neck section is similar to YOLOv5, where C2f is used for information flow. The architecture of YOLOv8 is shown in Fig. 14.

Output in 3 different scales for small, medium and large objects as in previous versions. It has a decoupled head structure for classification and regression tasks. Since estimating the coordinate of the bounding box and class prediction at the same time may lead to inefficient learning, a decoupled-head structure has been adopted. See also, adopting an anchor-free approach, YOLOv8 directly predicts the centre of the object. This reduces the number of boxes that need to be predicted, allowing for faster NMS processing. As in previous versions, data augmentation techniques were also used in YOLOv8. Methods such as mosaic enlargement, CutMix, and MixUp were employed, but since their use throughout the entire training process negatively affected performance, they were not used in the last ten epochs. The DFL loss function [109] was used for classification loss, while the CIoU loss function is applied for bounding box loss. YOLOv8 has successfully tackled multiple tasks, including object detection, classification, segmentation, pose estimation, and tracking. It is available in five sub-versions: YOLOv8n, YOLOv8s, YOLOv8m, YOLOv8l, and YOLOv8x.

It has sufficient processing speed and accuracy rates for real-time detection. It adopts a user-friendly approach that is easy to use on end devices. Small data sets can be worked with, and satisfactory results can be obtained. In addition, noise may increase in very small data and prediction accuracy may decrease. In one study [122], YOLOv8 was tested on flying objects and a mAP value of 99.1 % was obtained. Another study for object detection in adverse weather conditions during automated driving was performed with YOLOv8 [123]. A customised version of YOLOv8 was developed for the detection of small objects in unmanned aerial vehicle (UAV) images [124]. Another study [125] compared YOLOv5, 6, 7 and v8 for underwater environments and reported the success of YOLOv8.

2.9. YOLOv9

YOLOv9, titled "YOLOv9: Learning What You Want to Learn Using Programmable Gradient Information," was published by Chien-Yao Wang, I-Hau Yeh, and Hong-Yuan Mark Liao [126]. Developed based on YOLOv7, this version focuses on reversible functions and information loss issues.

YOLOv9 can be examined in two main sections: PGI and GELAN architecture.

YOLOv9 focuses on the issue of information bottleneck [127] encountered in deep networks. Feature extraction and spatial transformation operations are performed on the input data through multiple layers. The phenomenon of losing information as the data progresses through the network is referred to as the information bottleneck. Several methods exist to alleviate this problem. In the reversible architecture discussed in this paper, additional layers are required for combining the feedback data, which increases the inference cost. A function's reversibility refers to the ability to reverse it without information loss, thereby reducing the likelihood of losing critical data. In masked modelling methods, the loss functions may conflict with each other, potentially leading to incorrect associations. To address such issues, the concept of Programmable Gradient Information (PGI) [126] was proposed by the authors. PGI ensures reliable gradient generation to prevent information loss, thus enhancing the performance and convergence ability of the model. PGI consists of three main components: the main branch, the

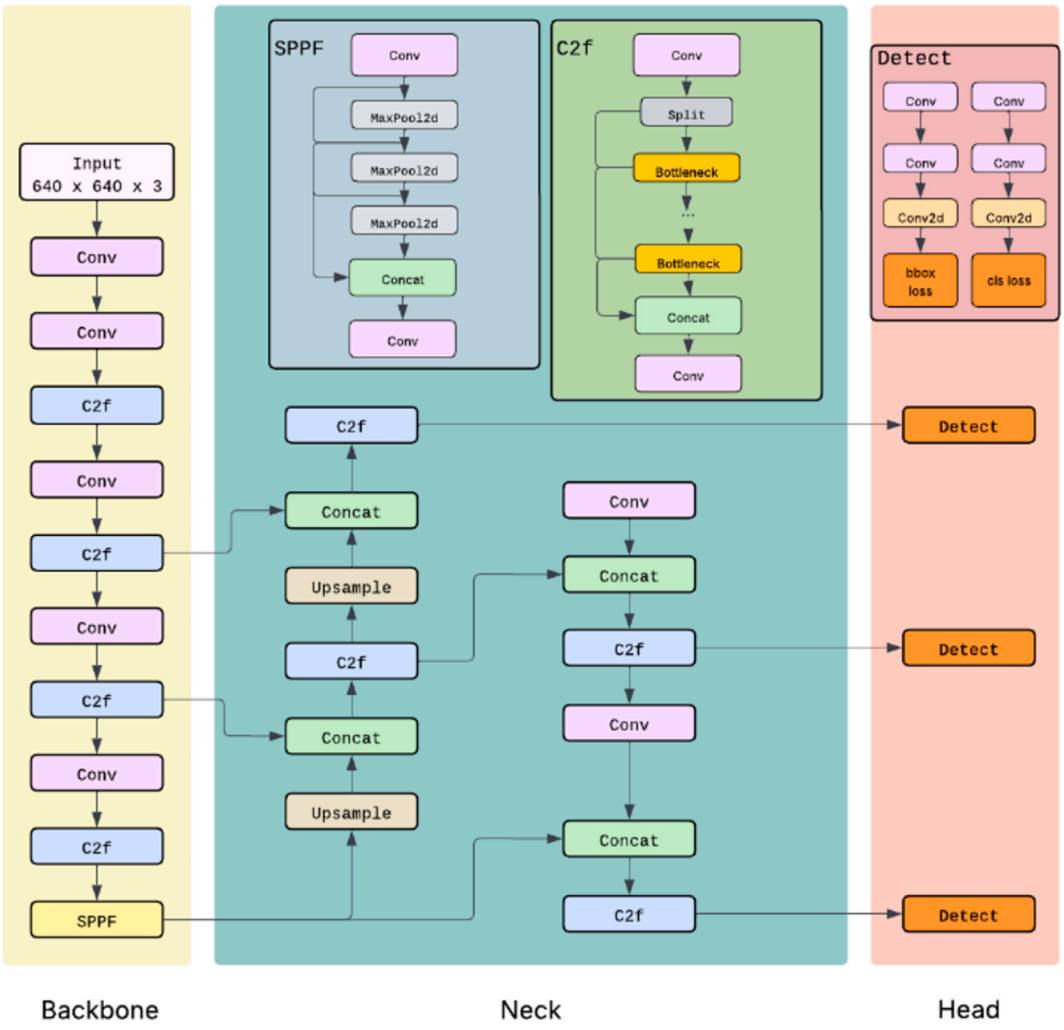

Fig. 14. YOLOv8 architecture.

auxiliary reversible branch, and multi-level auxiliary information. Since the reversible architecture is applied to the main branch during the inference process, the additional cost issue is resolved. It allows the model to be updated more accurately, but due to its slowing effect, it was only used during training. The auxiliary reversible branch is responsible for generating reliable gradients to prevent information loss caused by deepening networks. The multi-level auxiliary information component is used to combine gradients returned from different prediction blocks. Thanks to the PGI approach, accurate localisation of low-contrast objects and weak samples is ensured. In previous versions of YOLO, information loss may occur during learning with classical deep learning layers, while the problem of information loss is significantly alleviated by PGI. At the same time, while deepening the network in previous versions increased the inference cost, since PGI is used in the training phase, the cost remained constant in this sense. PGI has also managed to achieve higher accuracy on lightweight architectures thanks to gradients. Fig. 15 illustrates the PGI structure.

When designing the architecture, the ELAN structure shown in Fig. 12(a) was replaced by the GELAN (Generalized Efficient Layer Aggregation Network) [126] structure. GELAN is a lightweight architecture based on gradient path computation. In addition to the ELAN architecture, GELAN also incorporated the CSPNet [84] architecture. As in YOLOv7, the planned RepConv structure and CSPNet [84] blocks were used. The architecture, taking into account inference speed, accuracy, computational complexity, and the number of parameters, also allows users to select computational blocks as desired. In the architecture's head structure, the anchor-free head was optimized. Furthermore, by applying YOLOv7's auxiliary and leader head structure to PGI auxiliary supervision, they achieved good performance. The architecture of YOLOv9 is shown in Fig. 16. YOLOv9 has several subversions, including YOLOv9-s, YOLOv9-m, YOLOv9-c, and YOLOv9-e.

PGI has enabled YOLOv9 to detect multiple objects and improve performance in harsh environments. The number of parameters has been reduced by 46 % from the previous version YOLOv8 and an AP increase of 0.6 % has been achieved. Thanks to its lightweight architecture, ease of use in end devices has been ensured. By using YOLOv9 in an unmanned aerial vehicle, it can detect transport vehicles from the air for urban security [128]. It was seen in the study that YOLOv9, which is used to detect aircraft with satellite images, has a potential in providing security [129]. In a fire detection task [130], it achieved an MAP of 92 %, and in a challenging task such as bone fracture detection from X-ray images [53], it achieved an mAP of 65.62 %.

2.10. YOLOv10

YOLOv10, titled "YOLOv10: Real-Time End-to-End Object Detection," was published by Ao Wang, Hui Chen, Lihao Liu, and colleagues from Tsinghua University [131]. Based on YOLOv8, this version introduced a new approach that addresses the shortcomings of the model architecture and modifies post-processing techniques.

Computational load and inefficient parameter usage lead to a decrease in model performance. In the backbone section of YOLOv10,

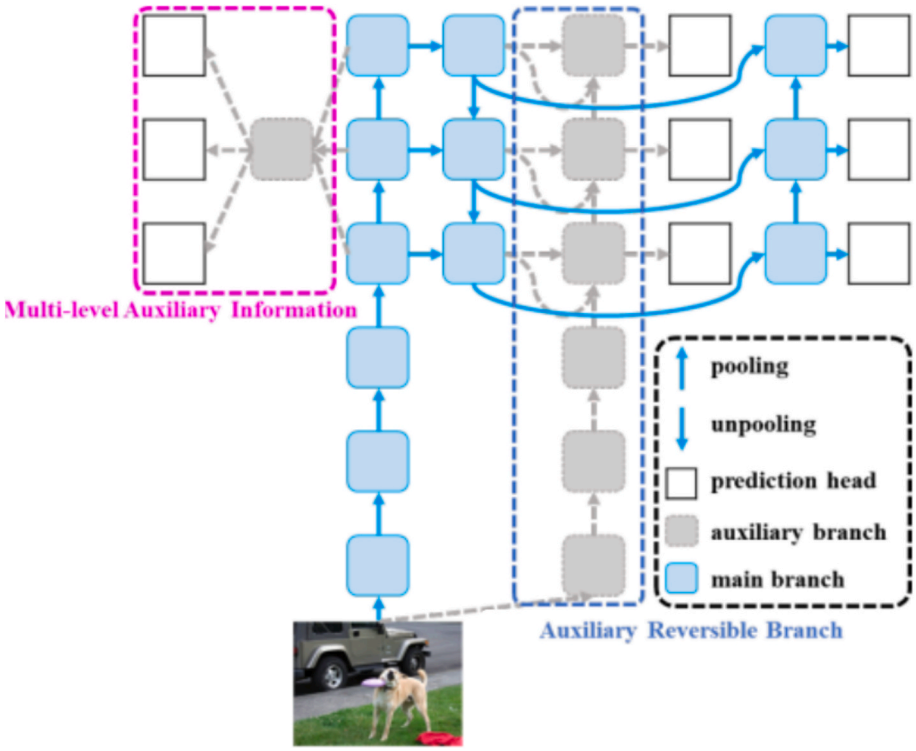

Fig. 15. Programmable Gradient Information (PGI).

which is based on YOLOv8, a modified version of CSPNet [84] was used. This reduced the computational load and improved the model's efficiency. In the neck section, the PANet [88] structure from previous versions were utilized. In the head section, an approach with both one-to-one and multi-head configurations were employed. One of the most significant changes in YOLOv10 is its NMS-free architecture. The post-processing technique, NMS, which removes redundant bounding boxes, negatively affects the inference process and hinders end-to-end deployment [131]. Therefore, the authors propose using a consistent binary assignment strategy with double-label assignments and a consistent matching metric during inference, without the need for NMS (Fig. 17). Here, double-label assignments were used for one-to-many predictions during training and one-to-one predictions during inference. Consistent binary assignments refer to matching the model's predictions with the true locations and classes, enabling consistent tracking of moving objects. Overall, this strategy ensures a unique bounding box assignment for each object, thus improving accuracy while reducing inference time.

The "holistic efficiency-accuracy-oriented model," which reduces computational load while enhancing detection capability, was introduced in the study. The concepts of efficiency and accuracy were examined in two parts, with various techniques proposed. For efficiency, a lightweight classification head was initially designed. A reduced architecture, consisting of 3×3 and 1×1 convolutions, was developed to minimize the number of parameters and operations without compromising performance. This approach aimed to lower the computational burden. Subsequently, spatial-channel decoupled downsampling was proposed. By separating spatial reduction and channel expansion processes, an efficient downsampling method was achieved. Pointwise convolutions were utilized for adjusting channel dimensions, while depthwise convolutions were used for spatial dimensions. This ensured maximum information retention while reducing computational overhead. Another proposed approach is a sequence-guided block design. Since deeper networks contain more stages, intrinsic ordering was employed to analyse redundancy across these stages. To address this complexity and redundancy, a sequence-guided block design was introduced. This design also presented a compact inverted block (CIB) structure for spatial and channel shuffling [131]. Efficiency was achieved by removing unnecessary information and reducing complexity.

For model accuracy, two concepts were introduced. The first concept, large-kernel convolutions, was utilized for small-scale models. While expanding the receptive field allowed for improved performance on larger objects, challenges were encountered with smaller objects. In deeper stages, 7×7 convolutions were employed, and an additional 3×3 convolution branch was introduced through structural re-parameterization for optimization. Self-attention [132] is used for global modelling but comes with issues related to computational complexity and memory usage. To address these issues, the second concept, the Partial Self-Attention (PSA) module [131], was designed. PSA split features into two parts, feeding only one part into the attention module before merging them again. By applying PSA in low-resolution stages, computational costs are reduced. As a result, learning capability is enhanced, improving both model accuracy and performance. Fig. 18 shows the YOLOv10 architecture. Like YOLOv8, it features N/S/M/L/X scales. Additionally, by increasing the width factor of the YOLOv10M version, the YOLOv10B version was also produced.

YOLOv10 contains too many new blocks, which increases the complexity of the model and makes it difficult to modify. In a study [133], when YOLOv9 and YOLOv10 were compared over a safety helmet detection application, YOLOv10 outperformed YOLOv9 by achieving 98 % mAP. YOLOv10 achieved 99.5 % mAP for the detection of tears in different types of fabrics [134], 87.6 % mAP in a pedestrian detection application for autonomous vehicles [135] and 94 % mAP in an application for detecting paediatric wrist fractures [136].

2.11. YOLOv11

YOLOv11 [137], the latest model released by Ultralytics, does not have an official paper. Like most of its predecessors, it has several subversions. YOLOv11, based on the versatile modelling introduced in YOLOv8, includes applications such as object detection, classification, segmentation, pose estimation, and oriented object detection (OBB –

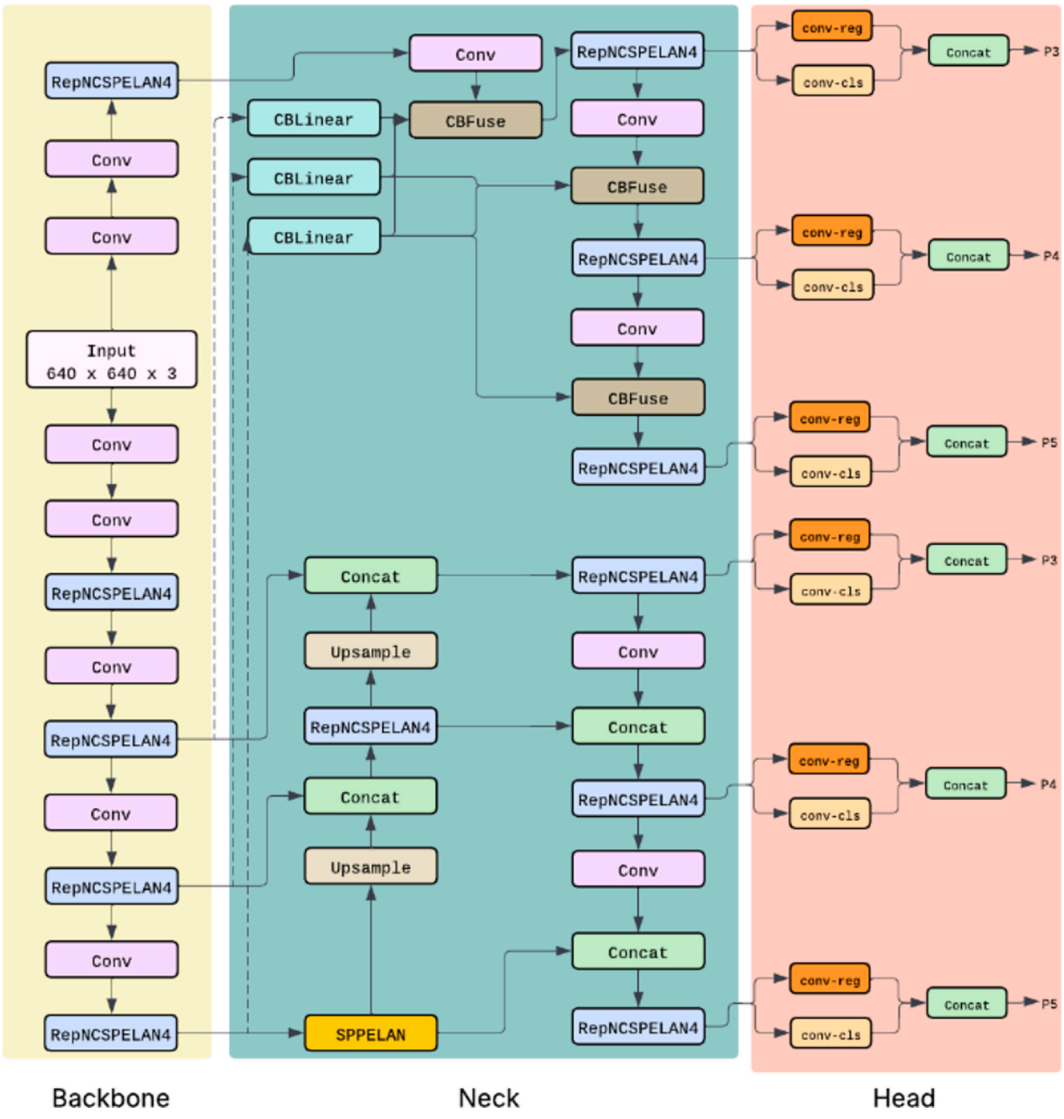

Fig. 16. YOLOv9 architecture.

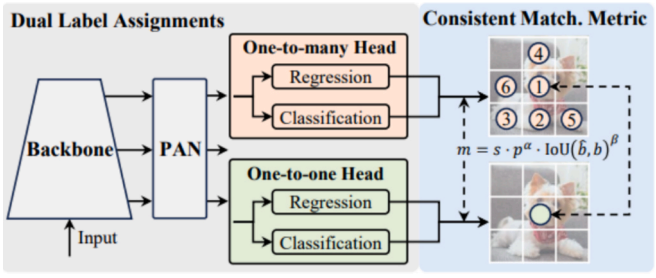

Fig. 17. Consistent binary assignments for NMS-free [100].

Oriented Bounding Boxes) [138]. YOLOv11 has achieved performance improvements with the addition of new blocks to the architecture.

YOLOv11, developed based on YOLOv8, employed a similar architecture but introducing several innovations. In the backbone section, five convolutional layers of different sizes were used to establish the foundation for feature extraction. The first innovation in the architecture was the introduction of the C3k2 layer. Replacing the C2f layer, a derivative of CSP used in previous versions, the C3k2 layer aimed to increase computational efficiency. This layer consists of two smaller convolutions instead of a large one, with K2 indicating the kernel size. The convolution works as a sliding window over a 2x2 region. While performance is maintained, the cost is reduced, and feature extraction is improved [137]. The SPPF block, first used in YOLOv5, was applied to the backbone in the same way. Subsequently, another innovation, the C2PSA layer, was proposed. The PSA [131] module used in YOLOv10 was combined with two convolutional layers. This allowed the model to focus its attention on critical regions, improving performance on low-resolution and small objects while reducing computational costs. The C3k2 and C2PSA layers in the neck section facilitated feature transmission to the head, which was designed similarly to the YOLOv8 architecture. The C3k2 module was employed across different depths and regions to construct the final detection head [137]. Fig. 19 shows the YOLOv11 architecture.

Another feature in YOLOv11 was the use of oriented object detection (OBB) [138]. The primary goal was to better represent the position and orientation of the object to be detected and tracked. Unlike ordinary bounding boxes, inclined boxes are used. A rotation angle parameter is added to the bounding box calculation to measure the angle between the box and the horizontal axis. The box shape adjusts based on the object's movement. By following the object's inclination, the box better represents the actual area covered by the object. This facilitates the detection and tracking of inclined objects in applications such as autonomous vehicles, satellite, and aerial imaging. YOLOv11 provides better analysis

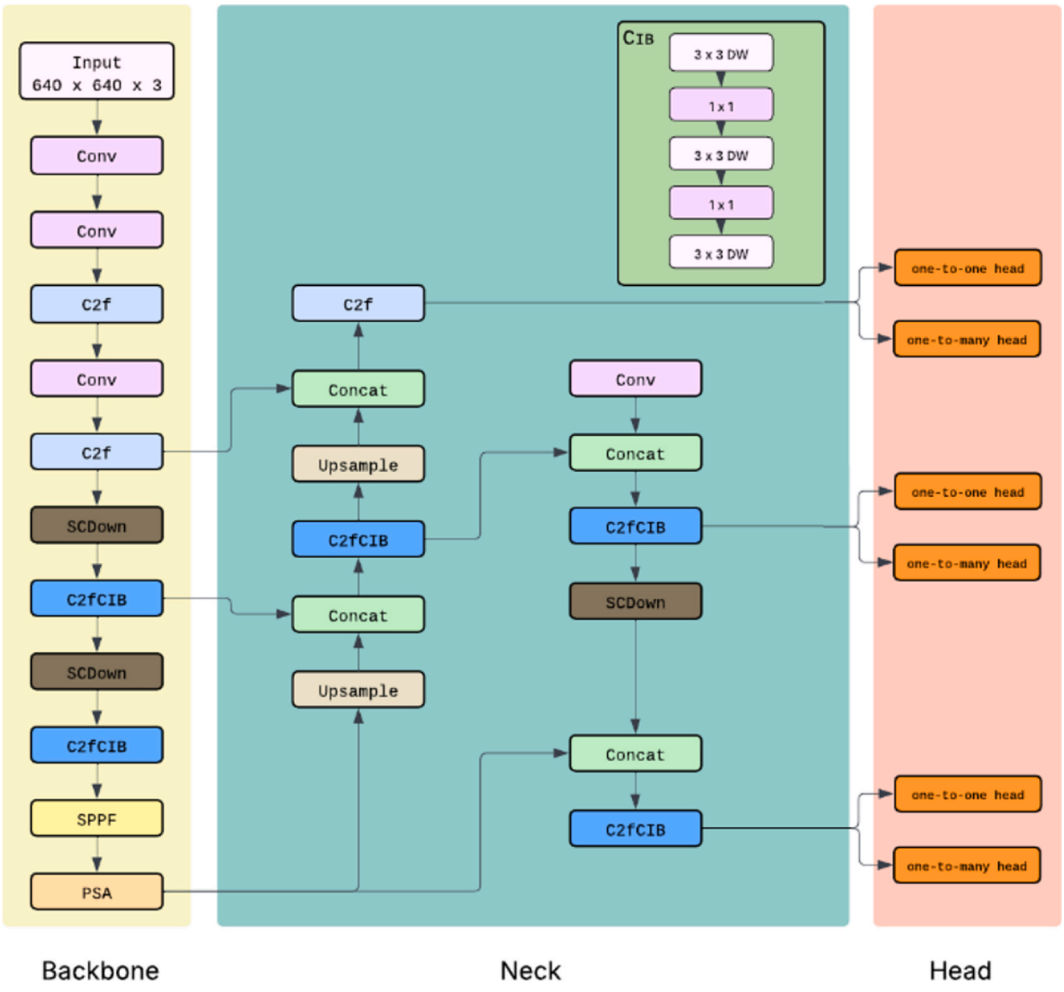

Fig. 18. YOLOv10 architecture.

and detection in complex images [137].

In a study [139], defects in solar panels were wanted to be detected and YOLOv5, YOLOv8, YOLOv11 models were compared in this context. YOLOv11 achieved the highest value with 93.4 % mAP but had difficulties in detecting rare defects. In another study, YOLOv8-YOLOv11 were compared for multiple weed detection, and it was reported that YOLOv11 achieved high accuracy and speed [140]. There is a YOLOv11 study developed for the detection of small objects by UAV and a mAP value of 45.3 % was obtained [141].

3. Discussion

Each version of YOLO has been built upon the previous one, adding new features and improvements. This paper examines the main versions of YOLO from YOLOv1 to YOLOv11. The evolving architectures of these versions over time have significantly impacted their speed and accuracy. Table 4 provides a performance summary of the versions analysed. This is based on the largest scale models. Table 5 summarises the architectural innovations of the versions.

Since the release of the first YOLO version, efforts have been made to address the shortcomings in real-time object detection. Before the original YOLO, object detection was performed by two-stage detectors. However, the speed required for real-time object detection was insufficient to overcome these challenges. The single-stage detector YOLO achieved high speed with almost a simple architecture. However, issues such as detecting small objects, difficulty in generalization, and multiple objects being present in the same bounding box led to a decrease in accuracy. Aiming to address these issues, YOLOv2 was introduced a year later. In this version, the generalization problem was attempted to be solved with the batch normalization approach and objects were predicted with higher accuracy by using anchor boxes. Transition layers were also added to alleviate the small object issue. YOLOv3 tried to balance accuracy and speed, employing an FPN structure for multi-scale object detection, which allowed for the fast and accurate detection of smaller objects. YOLOv4 used a modified architecture compared to previous versions. A neck structure was added, increasing computational complexity. However, by using data augmentation techniques and new strategies, accuracy was improved without sacrificing speed. While similar to its predecessor, YOLOv5 featured a lighter and more customizable architecture. A different strategy was used to alleviate the generalization problem, and multi-scale operation once again enhanced accuracy. Subversions were developed for users with different hardware capabilities, and this subversion tradition continued in all versions after YOLOv5. YOLOv6 adopted a different approach to increase speed and removed anchor boxes. Additionally, by designing an architecture that varied according to model size, it balanced speed with accuracy. High amounts of data are required for models to achieve good performance with high accuracy. However, YOLOv7 works efficiently even with small datasets due to its architecture, which does not require pre-trained weights. The decoupled head structure improved accuracy without increasing inference costs. YOLOv8 incorporates various applications and has a simplified structure. Since there are no anchor boxes, it provides fast object detection and uses different techniques for this. Previous versions, before YOLOv9, did not focus on the issue of information loss. YOLOv9 PGI and GELAN structures have been proposed to solve this problem. YOLOv9 proposed new techniques to address this issue

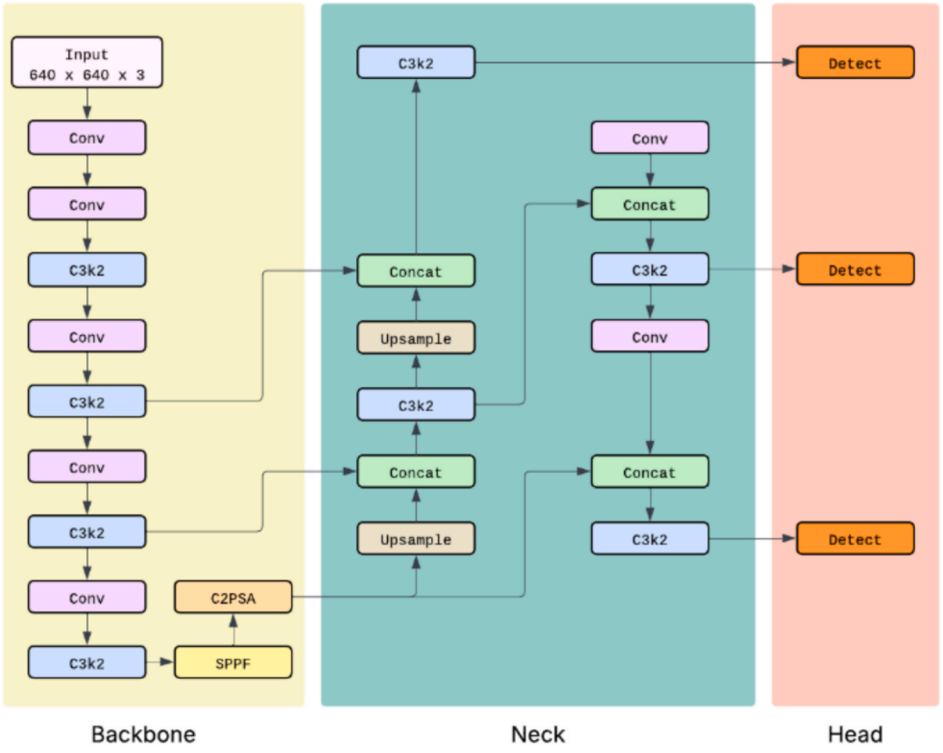

Fig. 19. YOLOv11 architecture.

Table 4
Comprehensive comparison of versions.

Versions	Date	Framework	Dataset	Input Size	mAP (%)	FPS (GPU)	FLOPs	Param (M)
YOLOv1	2015	Darknet	Pascal VOC	448	63.4	45	–	–
YOLOv2	2016	Darknet	Pascal VOC	416	76.8	67	–	–
YOLOv3	2018	Darknet	COCO	416	55.3	35	–	65.8
YOLOv4	2020	Darknet	COCO	416	43.5	65	–	64.4
YOLOv5	2020	Pytorch	COCO	640	50.7	101	205.7	86.7
YOLOv6	2022	Pytorch	COCO	640	52.8	121	144	58.5
YOLOv7	2022	Pytorch	COCO	640	56.8	114	189.9	71.3
YOLOv8	2023	Pytorch	COCO	640	53.9	82	257.8	68.2
YOLOv9	2024	Pytorch	COCO	640	55.6	73	189	57.3
YOLOv10	2024	Pytorch	COCO	640	54.4	120	160.4	29.5
YOLOv11	2024	Pytorch	COCO	640	54.7	240	194.9	56.9

Table 5
Comparison of innovations in versions.

Versions	Backbone	Fundamental innovation	Anchor	NMS	Difference from previous versions
YOLOv1	Darknet-24	End-to-end detection	✘	✔	Detection over a single network
YOLOv2	Darknet-19	Anchor boxes, BN	✔	✔	More stable box prediction
YOLOv3	Darknet-53	Residual connections, FPN	✔	✔	Multi-scale forecasting
YOLOv4	CSPDarknet53	Neck structure, CSP, PANet, SPP, mosaic data augmentation	✔	✔	Deeper and data augmented structure
YOLOv5	v5CSPDarknet53	SPPF, data augmentation, EMA, AutoAnchor	✔	✔	PyTorch based flexible architecture
YOLOv6	EfficientRep	RepVGG, repPAN, efficient decoupled head, AnchorFree, TAL	✘	✔	Anchor-free construction, speed/performance balance
YOLOv7	v7E-ELAN	ELAN structure, model scaling, lead and auxiliary head	✘	✔	ELAN and auxiliary headings
YOLOv8	v8CSPDarknet53	C2f, pose estimate	✘	✔	Lightened csp blocks and simplified decoupled head, easy installation
YOLOv9	v9GELAN	PGI, GELAN	✘	✔	Gradient control, enhanced backbone
YOLOv10	v10CSPDarknet53	NMS-free, double label assignment, PSA	✘	✘	NMS-free, attention module
YOLOv11	v11CSPDarknet53	C3k2, C2PSA, OBB	✘	✘	Enhanced attention, oriented object detection

and, while there was no significant increase in the ratios, it produced a user-based system that balances inference speed and accuracy. YOLOv10 has increased model speed by changing the post-processing techniques used in previous versions on an attention-based basis. New techniques were suggested in the architecture to balance this with accuracy, ensuring a good speed-accuracy trade-off. YOLOv11, the latest

version based on YOLOv8, includes multi-functional applications. By adding C3k2 and C2PSA layers, feature extraction is improved while performance is preserved. The introduction of oriented object detection aims to enhance tracking and detection performance. In each new version, instead of solely focusing on increasing speed and accuracy, an effort has been made to balance these two concepts.

The performance of YOLO may vary according to the characteristics of the hardware to be used. parameter and flops metrics provide users with a preliminary information about the selection of the appropriate model for their hardware. High metrics usually require high GPU and RAM capacity. In this context, a version-based comparison can be made for the suitability of the model to be preferred for the hardware.

YOLO is used in a wide range of applications from industrial production to medical imaging, from security applications to autonomous systems. YOLO has shown a rapid development since its first release with its speed-accuracy balance. In each version, focal points have changed, and new techniques have been proposed. However, despite all these developments, some challenges remain to be overcome. Lighter architectures are needed for use on edge devices.

4. Future works

The rapid evolution of the object detection process for YOLO based models is presented in the paper. It is foreseen that new YOLO models with architectural innovations, different optimization techniques and lightweight structures for end devices will emerge in the future. In this process, generalization capabilities are expected to improve, and speed/accuracy ratios are expected to increase.

For complex backgrounds and overlapping objects, new attention modules can be developed to improve performance. At the same time, performance decreases that may be experienced due to the quality of the image in the detection of small objects can be prevented by image enhancement techniques.

Most models require large-scale data to achieve successful results. The labelling process of large-scale data is a time-consuming process. In the future, YOLO models may need to adopt self-learning approaches for rapid adaptation to real-world problems. The process of using both training and real-time applications can be realized in a meaningful way and in short periods of time. In addition, it may be recommended to use lightweighting techniques to facilitate its use on end devices.

The multitasking feature of YOLOv8 has been used to solve different problems. In the latest model, YOLOv11, the OBB detected the position and direction details of the object better. In this context, additional tasks can be added to the models and problem-specific innovations can be offered.

In conclusion, for the future of YOLO, it is not enough to improve only the accuracy metrics, but also requires a comprehensive improvement such as generalization capability, deployment strategies to end devices, and lightweight of the models.

CRediT authorship contribution statement

Ayşe Aybilge Murat: Writing – original draft, Visualization, Resources, Methodology, Writing – review & editing, Investigation. **Mustafa Servet Kiran:** Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] S.V. Mahadevkar, B. Khemani, S. Patil, K. Kotecha, D.R. Vora, A. Abraham, L. A. Gabralla, A review on machine learning styles in computer vision—techniques and future directions, IEEE Access 10 (2022) 107293–107329.

[2] D. Bhatt, C. Patel, H. Talsania, J. Patel, R. Vaghela, S. Pandya, K. Modi, H. Ghayvat, CNN variants for computer vision: history, architecture, application, challenges and future scope, Electronics 10 (20) (2021) 2470.

[3] A. Kumar, A. Kalia, A. Sharma, Object detection: a comprehensive review of the state-of-the-art methods, Internat. J. Next-Gen. Computing 11 (1) (2020).

[4] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, A.C. Berg, Ssd,, Single shot multibox detector, in: Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14, Springer, 2016, pp. 21–37..

[5] T.-Y. Lin, P. Goyal, R. Girshick, K. He, P. Dollár, Focal loss for dense object detection, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 2980–2988.

[6] V.K. Sharma, R.N. Mir, A comprehensive and systematic look up into deep learning based object detection techniques: a review, Comput. Sci. Rev. 38 (2020) 100301.

[7] T. Diwan, G. Anirudh, J.V. Tembhurne, Object detection using YOLO: challenges, architectural successors, datasets and applications, Multimed. Tools Appl. 82 (6) (2023) 9243–9275.

[8] J. Terven, D.-M. Córdova-Esparza, J.-A. Romero-González, A comprehensive review of yolo architectures in computer vision: from yolov1 to yolov8 and yolo-nas, Mach. Learn. Knowl. Extr. 5 (4) (2023) 1680–1716.

[9] Wang, X.; Li, H.; Yue, X.; Meng, L. A comprehensive survey on object detection YOLO. Proceedings http://ceur-ws. org ISSN 2023, 1613, 0073.

[10] A. Vijayakumar, S. Vairavasundaram, Yolo-based object detection models: A review and its applications, Multimedia Tools Appl. (2024) 1–40.

[11] M. Hussain, YOLOv1 to v8: unveiling each variant–a comprehensive review of YOLO, IEEE Access 12 (2024) 42816–42833.

[12] Alif, M. A. R.; Hussain, M. YOLOv1 to YOLOv10: A comprehensive review of YOLO variants and their application in the agricultural domain. *arXiv preprint arXiv:2406.10139*, **2024**.

[13] M.L. Ali, Z. Zhang, The YOLO framework: a comprehensive review of evolution, applications, and benchmarks in object detection, Computers 13 (12) (2024) 336.

[14] S. Uchida, Image processing and recognition for biological images, Dev. Growth Differ. 55 (4) (2013) 523–549.

[15] R. Kaur, S. Singh, A comprehensive review of object detection with deep learning, Digital Signal Process. 132 (2023) 103812.

[16] K. Sharifani, M. Amini, Machine learning and deep learning: a review of methods and applications, World Inform. Technol. Eng. J. 10 (07) (2023) 3897–3904.

[17] Y. Guo, Y. Liu, A. Oerlemans, S. Lao, S. Wu, M.S. Lew, Deep learning for visual understanding: a review, Neurocomputing 187 (2016) 27–48.

[18] L. Jiao, F. Zhang, F. Liu, S. Yang, L. Li, Z. Feng, R. Qu, A survey of deep learning-based object detection, IEEE Access 7 (2019) 128837–128868.

[19] N. Aishwarya, K.M. Prabhakaran, F.T. Debebe, M.S.S.A. Reddy, P. Pranavee, Skin cancer diagnosis with YOLO deep neural network, Procedia Comput. Sci. 220 (2023) 651–658.

[20] J. Xu, H. Ren, S. Cai, X. Zhang, An improved faster R-CNN algorithm for assisted detection of lung nodules, Comput. Biol. Med. 153 (2023) 106470.

[21] G. Li, X. Huang, J. Ai, Z. Yi, W. Xie, Lemon-YOLO: an efficient object detection method for lemons in the natural environment, IET Image Process. 15 (9) (2021) 1998–2009.

[22] N. Lamb, M.C. Chuah, A strawberry detection system using convolutional neural networks, in: 2018 IEEE International Conference on Big Data (Big Data), IEEE, 2018, pp. 2515–2520..

[23] R. Ravindran, M.J. Santora, M.M. Jamali, Multi-object detection and tracking, based on DNN, for autonomous vehicles: a review, IEEE Sens. J. 21 (5) (2020) 5668–5677.

[24] M. Hnewa, H. Radha, Object detection under rainy conditions for autonomous vehicles: a review of state-of-the-art and emerging techniques, IEEE Signal Process Mag. 38 (1) (2020) 53–67.

[25] R.M. Saji, N. Sobhana, Real time object detection using SSD for bank security. IOP Conference Series: Materials Science and Engineering, IOP Publishing, 2021. Vol. 1070, p 012060.

[26] Z. Liu, J. Li, Y. Shu, D. Zhang, Detection and recognition of security detection object based on YOLO9000, in: 2018 5th International Conference on Systems and Informatics (ICSAI), IEEE, 2018, pp. 278–282.

[27] L. Jiao, J. Zhao, A survey on the new generation of deep learning in image processing, IEEE Access 7 (2019) 172231–172263.

[28] Y. Xiao, Z. Tian, J. Yu, Y. Zhang, S. Liu, S. Du, X. Lan, A review of object detection based on deep learning, Multimed. Tools Appl. 79 (2020) 23729–23791.

[29] R. Girshick, J. Donahue, T. Darrell, J. Malik, Rich feature hierarchies for accurate object detection and semantic segmentation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 580–587.

[30] R. Girshick, Fast r-cnn, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 1440–1448.

[31] S. Ren, K. He, R. Girshick, J. Sun, Faster r-cnn: Towards real-time object detection with region proposal networks, Adv. Neural Inform. Process. Syst. (2015) 28..

[32] Y. Zhang, X. Li, F. Wang, B. Wei, L. Li, A comprehensive review of one-stage networks for object detection, in: 2021 IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC), IEEE, 2021, pp. 1–6.

[33] M. Tan, R. Pang, L.Q.V. Efficientdet, Scalable and efficient object detection, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 10781–10790.

[34] Benjumea, A.; Teeti, I.; Cuzzolin, F.; Bradley, A. YOLO-Z: Improving small object detection in YOLOv5 for autonomous vehicles. *arXiv preprint arXiv:2112.11798* **2021**.

[35] W. Liu, G. Ren, R. Yu, S. Guo, J. Zhu, L. Zhang, Image-adaptive YOLO for object detection in adverse weather conditions, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2022, pp. 1792–1800.

[36] P. Hurtik, V. Molek, J. Hula, M. Vajgl, P. Vlasanek, T. Nejezchleba, Poly-YOLO: higher speed, more precise detection and instance segmentation for YOLOv3, Neural Comput. & Applic. 34 (10) (2022) 8275–8290.

[37] Z. Xie, X. Wei, Automatic parking space detection system based on improved YOLO algorithm, in: 2021 2nd International Conference on Computer Science and Management Technology (ICCSMT), IEEE, 2021, pp. 279–285.

[38] W. Lan, J. Dang, Y. Wang, S. Wang, Pedestrian detection based on YOLO network model, in: 2018 IEEE international conference on mechatronics and automation (ICMA), IEEE, 2018, pp. 1547–1551.

[39] R.-C. Chen, Automatic license plate recognition via sliding-window darknet-YOLO deep learning, Image Vis. Comput. 87 (2019) 47–56.

[40] R. Laroca, E. Severo, L.A. Zanlorensi, L.S. Oliveira, G.R. Gonçalves, W. R. Schwartz, D. Menotti, A robust real-time automatic license plate recognition based on the YOLO detector, in: 2018 international joint conference on neural networks (ijcnn), IEEE, 2018, pp. 1–10.

[41] J.-H. Kim, N. Kim, Y.W. Park, C.S. Won, Object detection and classification based on YOLO-V5 with improved maritime dataset, J. Marine Sci. Eng. 10 (3) (2022) 377.

[42] V. Kharchenko, I. Chyrka, Detection of airplanes on the ground using YOLO neural network, in: 2018 IEEE 17th international conference on mathematical methods in electromagnetic theory (MMET), IEEE, 2018, pp. 294–297.

[43] S. Narejo, B. Pandey, D. Esenarro Vargas, C. Rodriguez, M.R. Anjum, Weapon detection using YOLO V3 for smart surveillance system, Math. Probl. Eng. 2021 (2021) 1–9.

[44] A.H. Ashraf, M. Imran, A.M. Qahtani, A. Alsufyani, O. Almutiry, A. Mahmood, M. Attique, M. Habib, Weapons detection for security and video surveillance using cnn and YOLO-v5s, CMC-Comput. Mater. Contin 70 (2022) 2761–2775.

[45] W. Chen, H. Huang, S. Peng, C. Zhou, C. Zhang, YOLO-face: a real-time face detector, Vis. Comput. 37 (2021) 805–813.

[46] F.M. Talaat, H. ZainEldin, An improved fire detection approach based on YOLO-v8 for smart cities, Neural Comput. & Applic. 35 (28) (2023) 20939–20954.

[47] L. Tan, T. Huangfu, L. Wu, W. Chen, Comparison of RetinaNet, SSD, and YOLO v3 for real-time pill identification, BMC Med. Inf. Decis. Making 21 (2021) 1–11.

[48] K. Tong, Y. Wu, I-YOLO: a novel single-stage framework for small object detection, Vis. Comput. (2024) 1–18.

[49] F. Prinzi, M. Insalaco, A. Orlando, S. Gaglio, S. Vitabile, A YOLO-based model for breast cancer detection in mammograms, Cogn. Comput. 16 (1) (2024) 107–120.

[50] G.H. Aly, M. Marey, S.A. El-Sayed, M.F. Tolba, YOLO based breast masses detection and classification in full-field digital mammograms, Comput. Methods Programs Biomed. 200 (2021) 105823.

[51] M.F. Almufareh, M. Imran, A. Khan, M. Humayun, M. Asim, Automated brain tumor segmentation and classification in MRI using YOLO-based deep learning, IEEE Access (2024).

[52] I. Pacal, A. Karaman, D. Karaboga, B. Akay, A. Basturk, U. Nalbantoglu, S. Coskun, An efficient real-time colonic polyp detection with YOLO algorithms trained by using negative samples and large datasets, Comput. Biol. Med. 141 (2022) 105031.

[53] C.T. Chien, R.Y. Ju, K.Y. Chou, J.S. Chiang, YOLOv9 for fracture detection in pediatric wrist trauma X-ray images, Electron. Lett 60 (11) (2024) e13248.

[54] Y. Nie, P. Sommella, M. O'Nils, C. Liguori, J. Lundgren, Automatic detection of melanoma with yolo deep convolutional neural networks, in: 2019 E-Health and Bioengineering Conference (EHB), IEEE, 2019, pp. 1–4.

[55] M. Alaftekin, I. Pacal, K. Cicek, Real-time sign language recognition based on YOLO algorithm, Neural Comput. & Applic. (2024) 1–16.

[56] Imran, A.; Hulikal, M. S.; Gardi, H. A. Real Time American Sign Language Detection Using Yolo-v9. *arXiv preprint arXiv:2407.17950* **2024**.

[57] Y. Bai, J. Yu, S. Yang, J. Ning, An improved YOLO algorithm for detecting flowers and fruits on strawberry seedlings, Biosyst. Eng. 237 (2024) 1–12.

[58] Y. Tian, G. Yang, Z. Wang, H. Wang, E. Li, Z. Liang, Apple detection during different growth stages in orchards using the improved YOLO-V3 model, Comput. Electron. Agric. 157 (2019) 417–426.

[59] G. Liu, J.C. Nouaze, P.L. Touko Mbouembe, J.H. Kim, YOLO-tomato: a robust algorithm for tomato detection based on YOLOv3, Sensors 20 (7) (2020) 2145.

[60] R. Gai, N. Chen, H. Yuan, A detection algorithm for cherry fruits based on the improved YOLO-v4 model, Neural Comput. & Applic. 35 (19) (2023) 13895–13906.

[61] Z. Xue, R. Xu, D. Bai, H. Lin, YOLO-tea: a tea disease detection model improved by YOLOv5, Forests 14 (2) (2023) 415.

[62] D.-L. Pham, T.-W. Chang, A YOLO-based real-time packaging defect detection system, Procedia Comput. Sci. 217 (2023) 886–894.

[63] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, C. L. Zitnick, Microsoft coco: Common objects in context, in: Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13, Springer, 2014, pp. 740–755.

[64] M. Everingham, L. Van Gool, C.K. Williams, J. Winn, A. Zisserman, The pascal visual object classes (voc) challenge, Int. J. Comput. Vis. 88 (2010) 303–338.

[65] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, You only look once: Unified, real-time object detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 779–788.

[66] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 1–9.

[67] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, Imagenet large scale visual recognition challenge, Int. J. Comput. Vis. 115 (2015) 211–252.

[68] Redmon, J. *Darknet: Open source neural networks in c.* . 2013–2016. http://pjreddie.com/darknet/ (accessed 02.06.2024).

[69] M.A. Al-Masni, M.A. Al-Antari, J.-M. Park, G. Gi, T.-Y. Kim, P. Rivera, E. Valarezo, M.-T. Choi, S.-M. Han, T.-S. Kim, Simultaneous detection and classification of breast masses in digital mammograms via a deep learning YOLO-based CAD system, Comput. Methods Programs Biomed. 157 (2018) 85–94.

[70] A. Ye, B. Pang, Y. Jin, J. Cui, A YOLO-based neural network with VAE for intelligent garbage detection and classification, in: Proceedings of the 2020 3rd International Conference on Algorithms, Computing and Artificial Intelligence, 2020, pp. 1–7.

[71] J. Redmon, A. Farhadi, YOLO9000: better, faster, stronger, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 7263–7271.

[72] G.A. Miller, WordNet: a lexical database for English, Commun. ACM 38 (11) (1995) 39–41.

[73] Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* **2014**.

[74] Lin, M.; Chen, Q.; Yan, S. Network in network. *arXiv preprint arXiv:1312.4400* **2013**.

[75] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, in: International Conference on Machine Learning, PMLR, 2015, pp. 448–456.

[76] M. Loey, G. Manogaran, M.H.N. Taha, N.E.M. Khalifa, Fighting against COVID-19: a novel deep learning model based on YOLO-v2 with ResNet-50 for medical face mask detection, Sustain. Cities Soc. 65 (2021) 102600.

[77] X. Zhang, Z. Qiu, P. Huang, J. Hu, J. Luo, Application research of YOLO v2 combined with color identification, in: 2018 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), IEEE, 2018, pp. 138–1383.

[78] Redmon, J.; Farhadi, A. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767* **2018**.

[79] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, S. Belongie, Feature pyramid networks for object detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 2117–2125.

[80] J. Li, J. Gu, Z. Huang, J. Wen, Application research of improved YOLO V3 algorithm in PCB electronic component detection, Appl. Sci. 9 (18) (2019) 3750.

[81] Bochkovskiy, A.; Wang, C.-Y.; Liao, H.-Y. M. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934* **2020**.

[82] S. Xie, R. Girshick, P. Dollár, Z. Tu, K. He, Aggregated residual transformations for deep neural networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 1492–1500.

[83] M. Tan, Q. Le, Efficientnet: Rethinking model scaling for convolutional neural networks, in: International Conference on Machine Learning, PMLR, 2019, pp. 6105–6114.

[84] C.-Y. Wang, H.-Y.-M. Liao, Y.-H. Wu, P.-Y. Chen, J.-W. Hsieh, 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, 2020, pp. 390–391.

[85] Misra, D. Mish: A self regularized non-monotonic activation function. *arXiv preprint arXiv:1908.08681* **2019**.

[86] K. He, X. Zhang, S. Ren, J. Sun, Spatial pyramid pooling in deep convolutional networks for visual recognition, IEEE Trans. Pattern Anal. Mach. Intell. 37 (9) (2015) 1904–1916.

[87] S. Liu, D. Huang, Receptive field block net for accurate and fast object detection, in: Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 385–400.

[88] S. Liu, L. Qi, H. Qin, J. Shi, J. Jia, Path aggregation network for instance segmentation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 8759–8768.

[89] S. Woo, J. Park, J.-Y. Lee, K.I.S. Cbam, Convolutional block attention module, in: Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 3–19.

[90] G. Ghiasi, T.-Y. Lin, Q.V. Le, Dropblock: a regularization method for convolutional networks, Adv. Neural Inf. Proces. Syst. 31 (2018).

[91] Z. Zheng, P. Wang, W. Liu, J. Li, R. Ye, D. Ren, Distance-IoU loss: Faster and better learning for bounding box regression, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2020, pp. 12993–13000.

[92] P. Kannadaguli, YOLO v4 based human detection system using aerial thermal imaging for UAV based surveillance applications, in: 2020 International Conference on Decision Aid Sciences and Application (DASA), IEEE, 2020, pp. 1213–1219.

[93] M. Lippi, N. Bonucci, R.F. Carpio, M. Contarini, S. Speranza, A. Gasparri, A yolo-based pest detection system for precision agriculture, in: 2021 29th Mediterranean Conference on Control and Automation (MED), IEEE, 2021, pp. 342–347.

[94] A.M. Roy, J. Bhaduri, T. Kumar, K. Raj, WilDect-YOLO: an efficient and robust computer vision-based accurate object localization model for automated endangered wildlife detection, Eco. Inform. 75 (2023) 101919.
[95] H. Wu, Y. Hu, W. Wang, X. Mei, J. Xian, Ship fire detection based on an improved YOLO algorithm with a lightweight convolutional neural network model, Sensors 22 (19) (2022) 7420.
[96] Jocher, G. *Ultralytics YOLOv5*. 2020. https://github.com/ultralytics/yolov5 (accessed 22.12.2024).
[97] Jocher, G. *YOLOv5 architecture by Ultralytics* 2020. https://docs.ultralytics.com/yolov5/tutorials/architecture_description/ (accessed 02.06.2024).
[98] G. Ghiasi, Y. Cui, A. Srinivas, R. Qian, T.-Y. Lin, E.D. Cubuk, Q.V. Le, B. Zoph, Simple copy-paste is a strong data augmentation method for instance segmentation, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 2918–2928.
[99] Zhang, H.; Cisse, M.; Dauphin, Y. N.; Lopez-Paz, D. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412* **2017**.
[100] S. Schneidereit, A.M. Yarahmadi, T. Schneidereit, M. Breuß, M. Gebauer, YOLO-based Object detection in industry 4.0 Fischertechnik model environment, in: Proceedings of SAI Intelligent Systems Conference, Springer, 2023, pp. 1–20.
[101] Z. Guo, C. Wang, G. Yang, Z. Huang, G. Li, Msft-yolo: improved yolov5 based on transformer for detecting defects of steel surface, Sensors 22 (9) (2022) 3467.
[102] M. Idrissi, A. Hussain, B. Barua, A. Osman, R. Abozariba, A. Aneiba, T. Asyhari, Evaluating the forest ecosystem through a semi-autonomous quadruped robot and a hexacopter uav, Sensors 22 (15) (2022) 5497.
[103] R. Zhang, C. Wen, Sod-yolo: a small target defect detection algorithm for wind turbine blades based on improved YOLOV5, Adv. Theor. Simul. 5 (7) (2022) 2100631.
[104] Li, C.; Li, L.; Jiang, H.; Weng, K.; Geng, Y.; Li, L.; Ke, Z.; Li, Q.; Cheng, M.; Nie, W. YOLOv6: A single-stage object detection framework for industrial applications. *arXiv preprint arXiv:2209.02976* **2022**.
[105] X. Ding, X. Zhang, N. Ma, J. Han, G. Ding, S.J. Repvgg, Making vgg-style convnets great again, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 13733–13742.
[106] H. Zhang, Y. Wang, F. Dayoub, S.N. Varifocalnet, An iou-aware dense object detector, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 8514–8523.
[107] Gevorgyan, Z. SIoU loss: More powerful learning for bounding box regression. *arXiv preprint arXiv:2205.12740* **2022**.
[108] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, S. Savarese, Generalized intersection over union: a metric and a loss for bounding box regression, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 658–666.
[109] X. Li, W. Wang, L. Wu, S. Chen, X. Hu, J. Li, J. Tang, J. Yang, Generalized focal loss: Learning qualified and distributed bounding boxes for dense object detection, Adv. Neural Inf. Proces. Syst. 33 (2020) 21002–21012.
[110] C. Feng, Y. Zhong, Y. Gao, M.R. Scott, Huang, W. Tood, Task-aligned one-stage object detection, in: 2021 IEEE/CVF International Conference on Computer Vision (ICCV), IEEE Computer Society, 2021, pp. 3490–3499.
[111] Ding, X.; Chen, H.; Zhang, X.; Huang, K.; Han, J.; Ding, G. Re-parameterizing your optimizers rather than architectures. *arXiv preprint arXiv:2205.15242* **2022**.
[112] C. Shu, Y. Liu, J. Gao, Z. Yan, C. Shen, Channel-wise knowledge distillation for dense prediction, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 5311–5320.
[113] R.B. Bist, S. Subedi, X. Yang, L. Chai, A novel YOLOv6 object detector for monitoring piling behavior of cage-free laying hens, AgriEngineering 5 (2) (2023) 905–923.
[114] E. Iren, Comparison of yolov5 and yolov6 models for plant leaf disease detection, Eng. Technol. Appl. Sci. Res. 14 (2) (2024) 13714–13719.
[115] C.-Y. Wang, A. Bochkovskiy, 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, 2023, pp. 7464–7475.
[116] Wang, C.-Y.; Liao, H.-Y. M.; Yeh, I.-H. Designing network design strategies through gradient path analysis. *arXiv preprint arXiv:2211.04800* **2022**.
[117] N.D.T. Yung, W. Wong, F.H. Juwono, Z.A. Sim, Safety helmet detection using deep learning: Implementation and comparative study using YOLOv5, YOLOv6, and YOLOv7, in: 2022 International Conference on Green Energy, Computing and Sustainable Technology (GECOST), IEEE, 2022, pp. 164–170.
[118] N.I.M. Yusof, A. Sophian, H.F.M. Zaki, A.A. Bawono, A.H. Embong, A. Ashraf, Assessing the performance of YOLOv5, YOLOv6, and YOLOv7 in road defect detection and classification: a comparative study, Bull. Electr. Eng. Inform. 13 (1) (2024) 350–360.
[119] P.C. Kusuma, B. Soewito, Multi-object detection using yolov7 object detection algorithm on mobile device, J. Appl. Eng. Technol. Sci. (JAETS) 5 (1) (2023) 305–320.
[120] A.I. Sapitri, S. Nurmaini, M.N. Rachmatullah, B. Tutuko, A. Darmawahyuni, F. Firdaus, D.P. Rini, A. Islami, Deep learning-based real time detection for cardiac objects with fetal ultrasound video, Inf. Med. Unlocked 36 (2023) 101150.
[121] Jocher, G.; Chaurasia, A.; Qiu, J. *Ultralytics YOLOv8* 2023. https://github.com/ultralytics/ultralytics (accessed 02.06.2024).
[122] Reis, D.; Kupec, J.; Hong, J.; Daoudi, A. Real-time flying object detection with YOLOv8. *arXiv preprint arXiv:2305.09972* **2023**.
[123] D. Kumar, N. Muhammad, Object detection in adverse weather for autonomous driving through data merging and YOLOv8, Sensors 23 (20) (2023) 8471.
[124] G. Wang, Y. Chen, P. An, H. Hong, J. Hu, T. Huang, UAV-YOLOv8: a small-object-detection model based on improved YOLOv8 for UAV aerial photography scenarios, Sensors 23 (16) (2023) 7190.
[125] B. Gašparović, G. Mauša, J. Rukavina, J. Lerga, Evaluating Yolov5, Yolov6, Yolov7, and Yolov8 in underwater environment: Is there real improvement?, in: 2023 8th International Conference on Smart and Sustainable Technologies (SpliTech) IEEE, 2023, pp. 1–4.
[126] Wang, C.; Yeh, I.; Liao, H. YOLOv9: Learning what you want to learn using programmable gradient information. arXiv 2024. *arXiv preprint arXiv:2402.13616*.
[127] N. Tishby, N. Zaslavsky, Deep learning and the information bottleneck principle, in: 2015 IEEE information theory workshop (itw), IEEE, 2015, pp. 1–5.
[128] M. Bakirci, I. Bayraktar, YOLOv9-enabled vehicle detection for urban security and forensics applications, in: 2024 12th International Symposium on Digital Forensics and Security (ISDFS), IEEE, 2024, pp. 1–6.
[129] M. Bakirci, I. Bayraktar, Transforming aircraft detection through LEO satellite imagery and YOLOv9 for improved aviation safety, in: 2024 26th International Conference on Digital Signal Processing and its Applications (DSPA), IEEE, 2024, pp. 1–6.
[130] L. Song, Y. Wu, W. Zhang, Research on fire detection based on the YOLOv9 algorithm, in: 2024 IEEE 4th International Conference on Electronic Technology, Communication and Information (ICETCI), IEEE, 2024, pp. 1398–1406.
[131] Wang, A.; Chen, H.; Liu, L.; Chen, K.; Lin, Z.; Han, J.; Ding, G. YOLOv10: Real-Time End-to-End Object Detection. *arXiv preprint arXiv:2405.14458* **2024**.
[132] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, Adv. Neural Inf. Proces. Syst. 30 (2017).
[133] F.Y. A'la, N. Firdaus, H. Imaduddin, Precision in safety: YOLOv9 vs. YOLOv10 for helmet image detection, in: 2024 International Visualization, Informatics and Technology Conference (IVIT), IEEE, 2024, pp. 159–164.
[134] M. Mao, A. Lee, M. Hong, Efficient fabric classification and object detection using YOLOv10, Electronics (2079–9292) 13 (19) (2024).
[135] Y. Li, W. Leong, H. Zhang, YOLOv10-based real-time pedestrian detection for autonomous vehicles, in: 2024 IEEE 8th International Conference on Signal and Image Processing Applications (ICSIPA), IEEE, 2024, pp. 1–6.
[136] Ahmed, A.; Manaf, A. Pediatric wrist fracture detection in x-rays via yolov10 algorithm and dual label assignment system. *arXiv preprint arXiv:2407.15689* **2024**.
[137] Jocher, G.; Qiu, J. *Ultralytics YOLO11*. 2024. https://github.com/ultralytics/ultralytics (accessed 10.11.2024).
[138] M. Zand, A. Etemad, M. Greenspan, Oriented bounding boxes for small and freely rotated objects, IEEE Trans. Geosci. Remote Sens. 60 (2021) 1–15.
[139] R. Khanam, T. Asghar, M. Hussain, Comparative performance evaluation of yolov5, yolov8, and yolov11 for solar panel defect detection, Solar 5 (2025) 6. MDPI.
[140] A. Sharma, V. Kumar, L. Longchamps, Comparative performance of YOLOv8, YOLOv9, YOLOv10, YOLOv11 and Faster R-CNN models for detection of multiple weed species, Smart Agric. Technol. 9 (2024) 100648.
[141] B. Dewangan, M. Srinivas, LIGHT-YOLOv11: an efficient small object detection model for UAV images, in: 2025 IEEE 14th International Conference on Communication Systems and Network Technologies (CSNT), IEEE, 2025, pp. 557–563.