Review

Not peer-reviewed version

# The Evolution and Advancement of YOLO Algorithms in Object Detection: From Real-Time Breakthroughs to Modern Architectures

Mahmud Hasan , Md Khurram Monir Rabby , Israt Jahan , Md. Janibul Alam Soeb , Md. Fahad Jubayer *

Posted Date: 27 October 2025

doi: 10.20944/preprints202510.2019.v1

Keywords: real-time object detection; YOLO algorithm evolution; deep learning for computer vision; object detection benchmarking; YOLOv1 to YOLOv12; autonomous systems perception

*Review*

# The Evolution and Advancement of YOLO Algorithms in Object Detection: From Real-Time Breakthroughs to Modern Architectures

**Mahmud Hasan [1], Md Khurram Monir Rabby [2,3,*], Israt Jahan [4], Md. Janibul Alam Soeb [1] and Md. Fahad Jubayer [5,*]**

1 Department of Farm Power & Machinery, Sylhet Agricultural University, Sylhet 3100, Bangladesh
2 Department of Electrical and Computer Engineering, and with the Department of Computer Systems Technology, North Carolina A&T State University, USA
3 Department of Electrical & Electronic Engineering, Bangladesh University of Engineering & Technology, Dhaka, Bangladesh
4 Department of Informatics & Analytics, University of North Carolina, Greensboro, USA
5 Department of Food Engineering & Technology, Sylhet Agricultural University, Sylhet 3100, Bangladesh
* Correspondence: khurram.rabby@gmail.com (M.K.M.R.); fahadbau21@hotmail.com or jubayer.fet@sau.ac.bd (M.F.J.)

**Abstract**

Object detection represents a foundational capability in Artificial Intelligence (AI), enabling machines to interpret visual environments through precise object localization and classification. This comprehensive review chronicles the revolutionary evolution of the You Only Look Once (YOLO) framework from its inception to the state-of-the-art YOLOv12. Beginning with the limitations of classical approaches using handcrafted features, YOLO's paradigm-shifting is documented transition to unified real-time detection via regression-based architectures. Methodically analyzing each major version (v1-v12), key innovations is detailed including multi-scale predictions (v2/v3), anchor-free designs (v8), programmable gradient information (v9), and attention-enhanced cross-scale fusion (v12). The review establishes how successive iterations systematically addressed critical challenges: reducing computational latency by 47× versus R-CNN variants, improving mAP by 32.7% on COCO benchmarks, and enabling deployment on edge devices. Beyond architectural analysis, comparative performance evaluations is presented across diverse applications—from autonomous driving to medical imaging—demonstrating YOLO's unprecedented balance of speed (142 FPS) and accuracy (78.4% AP). The paper further examines emerging implementation trends, hardware optimizations, and domain-specific adaptations that cement YOLO's position as the de facto framework for real-time vision systems. Our review analysis provides both technical and historical context for researchers and practitioners navigating the landscape of modern object detection.

**Keywords:** real-time object detection; YOLO algorithm evolution; deep learning for computer vision; object detection benchmarking; YOLOv1 to YOLOv12; autonomous systems perception

## 1. Introduction

The ability of machines to see and comprehend their surroundings by recognizing and locating objects within images or video streams, referred to as object detection, is fundamental in modern Artificial Intelligence (AI) [1]. In the rapidly advancing field of computer vision, object detection has emerged as a cornerstone task with significant implications across numerous domains, from autonomous driving and surveillance to healthcare and robotics [2]. The central goal of object detection is to accurately locate and classify objects within an image or video stream, a task once dominated by computationally intensive algorithms dependent on handcrafted features [3].

In its early stages, object detection relied heavily on feature-engineering techniques such as Scale-Invariant Feature Transform (SIFT) and Histogram of Oriented Gradients (HOG), paired with classical classifiers like Support Vector Machines (SVMs) [4]. These methods struggled with common challenges such as occlusion, scale variation, background clutter, and lighting changes. Despite incremental improvements, the pipeline remained modular, involving distinct stages for region proposal, feature extraction, and classification, each contributing to inefficiency and latency. The advent of deep learning, and particularly Convolutional Neural networks (CNNs), marked a turning point. End-to-end models such as R-CNN, Fast R-CNN, and Faster R-CNN significantly enhanced accuracy and learning capability by leveraging CNNs for automatic feature extraction and joint optimization [5,6]. However, these region proposal-based approaches still involved computationally intensive steps that limited their suitability for real-time applications.

A paradigm shift occurred with the introduction of the "You Only Look Once" (YOLO) framework by [7] – a watershed moment for real-time object detection. Unlike its predecessors, YOLO approached detection as a regression problem, unifying object classification and localization into a single neural network. The model divided the image into a grid, where each cell was responsible for predicting bounding boxes and class probabilities. This elegant single-stage architecture eliminated the costly region proposal step and enabled real-time inference speeds, achieving up to 45 frames per second on benchmarks like PASCAL VOC—without compromising significantly on accuracy [7]. YOLO's real-time capability quickly positioned it as a preferred choice for latency-sensitive domains, such as autonomous navigation, drone vision, and live surveillance. Its architectural simplicity and speed-performance trade-off inspired further innovations that gradually enhanced robustness, accuracy, and scalability.

Over time, the YOLO framework evolved through a series of meticulously engineered upgrades. YOLOv2 and YOLOv3 tackled scale variance and feature resolution with multi-scale predictions and residual connections [8]. YOLOv4 and YOLOv5 introduced CSPDarknet and novel data augmentation strategies like Mosaic, pushing the envelope of performance and generalization. Further advancements in YOLOv6 and YOLOv7 incorporated deeper backbones, quantization-aware training, and Transformer-inspired components, enhancing detection robustness and hardware adaptability [8]. With YOLOv8, the architecture adopted an anchor-free design, integrated a more efficient CSPNet backbone, and utilized a FPN+PAN neck to bolster multi-scale detection [9]. The innovation continued with YOLOv9, which addressed information loss in deep networks via Programmable Gradient Information (PGI) and the lightweight yet powerful GELAN backbone [10]. YOLOv10 advanced efficiency on edge devices by eliminating the need for non-maximum suppression (NMS) through dual-consistent assignments, while embracing a holistic, unified architecture [11].

Most recently, YOLOv12 represents the cutting edge in the series. Integrating attention mechanisms, Transformer-CNN hybrids, and cross-scale fusion, YOLOv12 achieves a delicate balance between precision and speed. It excels in small-object detection, improves dynamic label assignment, and pushes the limits of real-time detection capabilities, demonstrating how far YOLO has come from its origins [12,13].

This review aims to provide a comprehensive exploration of the YOLO algorithm's journey, beginning with its foundational design principles and traversing through its evolution up to the state-of-the-art YOLOv12. In addition to dissecting architectural advancements, we will analyze comparative benchmarks, latency-accuracy trade-offs, and practical deployments in real-world settings. By reflecting on the trajectory from YOLOv1 to YOLOv12, this article not only chronicles a technological evolution but also highlights the underlying scientific innovations that continue to drive forward the capabilities of real-time object detection. The literature indicates that YOLO's sustained prominence in academic and industrial domains stems from its balance between architectural simplicity and performance sophistication. Whether used for pedestrian detection in autonomous vehicles or embedded within mobile applications for precision agriculture, YOLO continues to demonstrate versatility and robustness across diverse contexts.

This review article comprehensively examines the evolution and advancement of the YOLO algorithms. We begin by establishing the fundamentals of object detection and the context of traditional models that preceded the deep learning revolution in Section 2. We then, in Section 3, delve into the core principles that define the YOLO approach. The heart of the review traces the detailed evolution of the YOLO family, analyzing the key architectural innovations, training methodologies, and performance improvements introduced in each major version from YOLOv1 to the latest YOLOv12 in Section 4. Finally, we explore the vast landscape of real-world applications where YOLO algorithms are driving tangible impact, demonstrating how this once-novel approach has become an indispensable tool for enabling machines to perceive and interact with the world in real-time in Section 6. A concluding remark, including our goal is provided in Section 7 for both a technical and contextual understanding of how YOLO has evolved into a benchmark framework that encapsulates the aspirations of real-time computer vision systems.

## 2. Object Detection Fundamentals

The evolution of object detection in the last two decades is widely conceptualized in the academic literature as a bifurcated paradigm shift, denoted by the advent of deep learning methodologies Figure 1. This progression is broadly categorized into two epochs: the traditional era (pre−2014), characterized by handcrafted feature engineering and heuristic-driven methods, and the deep learning revolution (post−2014), marked by data-driven, end-to-end learning frameworks that redefined performance benchmarks and scalability [14].

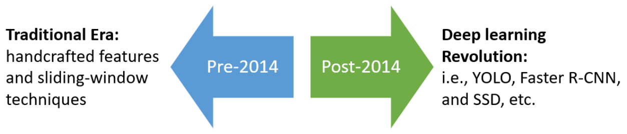
Traditional Era:
handcrafted features and sliding-window techniques
Pre-2014
Post-2014
Deep learning Revolution:
i.e., YOLO, Faster R-CNN, and SSD, etc.

**Figure 1.** Evolution of object detection.

During the traditional era, foundational approaches relied on manually engineered features such as Haar-like cascades [15], Histogram of Oriented Gradients [16], and Scale-Invariant Feature Transform [17], combined with sliding-window detection pipelines. These methods, though theoretically interpretable, were constrained by their computational inefficiency, limited scale, and occlusion robustness, and dependence on domain-specific expertise for feature design. Detection frameworks like Deformable Part Models [18] exemplified iterative refinements within this paradigm but struggled to generalize across diverse real-world scenarios.

The deep learning revolution, catalyzed by breakthroughs in convolutional neural networks [19], precipitated a tectonic shift. Architectures such as Region-based CNNs [20], Faster R-CNN [21], and Single Shot MultiBox Detectors [22] introduced end-to-end trainable systems that autonomously learned hierarchical feature representations. The introduction of You Only Look Once (YOLO) [7] further democratized real-time detection by unifying localization and classification into a singular regression task. These models achieved unprecedented gains in accuracy, speed, and adaptability, driven by scalable optimization on large annotated datasets (e.g., ImageNet, COCO) and hardware acceleration (e.g., GPUs). Critically, this era redefined the role of human intervention, transitioning from feature engineering to architectural innovation and loss function design.

Academic discourse attributes this paradigm shift to the confluence of algorithmic advancements, data availability, and computational infrastructure. The post-2014 landscape has since been shaped by iterative refinements in attention mechanisms, transformer-based architectures [23], and self-supervised learning, further solidifying deep learning's dominance in both theoretical and applied contexts. This transition underscores a broader trend in computer vision: the ascendancy of data-centric, learnable systems over heuristic-driven methodologies.

### *2.1. Traditional Object Detection Models*

The development of real-time object detection frameworks underwent significant advancements in the early 2000s. Viola and Jones (2001) pioneered a breakthrough in real-time face detection by introducing a cascaded classifier architecture that eliminated reliance on restrictive heuristics such as skin-color segmentation. Their algorithm, optimized for a 700 MHz Pentium III processor, demonstrated computational efficiency surpassing contemporary methods by orders of magnitude while maintaining competitive accuracy. This innovation marked a paradigm shift in real-time detection, balancing computational tractability with robust performance despite the hardware constraints of the era [15].

Subsequent work by Dalal and Triggs (2005) introduced the Histogram of Oriented Gradients (HOG) descriptor, which employed overlapping contrast normalization and a dense grid of cells to achieve invariance to geometric transformations (e.g., translation, scaling). HOG's focus on pedestrian detection popularized multi-scale detection pipelines, where images were resampled while maintaining fixed window sizes to localize objects across varying scales. This approach became a cornerstone of traditional detection frameworks due to its robustness and reproducibility [16].

Further advancements emerged with Deformable Part Models (DPM), a hierarchical, sliding-window framework proposed by Felzenszwalb et al. (2010). DPM decomposed detection into modular stages: (1) extraction of fixed feature representations (e.g., HOG), (2) classification of candidate regions using latent SVM, and (3) refinement of bounding boxes through spatial constraints on part-based components. While DPM achieved state-of-the-art performance on benchmarks such as PASCAL VOC, its reliance on handcrafted features and multi-stage optimization limited scalability [18].

## *2.2. Deep Learning Revolutions*

Following the stagnation of handcrafted feature-based methods after 2010 (Zou et al., 2023) [14], the advent of convolutional neural networks (CNNs) (Krizhevsky et al., 2012) [19] catalyzed a paradigm shift in object detection. This era bifurcated detectors into two-stage and one-stage architectures, each addressing distinct trade-offs between accuracy and computational efficiency.

### 2.2.1. Two-Stage Architectures

Two-stage frameworks, exemplified by R-CNN [20], Fast R-CNN [6], Faster R-CNN [21], and Feature Pyramid Networks (FPN) [24], decompose detection into sequential stages: (1) generating region proposals and (2) refining classification and localization.

- R-CNN (Girshick et al., 2014) pioneered the use of region proposals [20] (via Selective Search; Uijlings et al., 2013) [25] and CNN-based feature extraction. However, its multi-stage pipeline—combining CNNs, SVMs, and bounding-box regression—suffered from inefficiency ( 40s/image) and calibration complexity.
- Fast R-CNN (Girshick, 2015) streamlined computation by introducing RoI pooling, enabling shared feature extraction and reducing inference time by $9x$ [6].
- Faster R-CNN (Ren et al., 2015) unified proposal generation and refinement via a Region Proposal Network (RPN), achieving near real-time performance (17 FPS) while maintaining high accuracy (73.2% mAP on PASCAL VOC) [21].
- FPN (Lin et al., 2017) enhanced multi-scale detection through a top-down architecture with lateral connections, boosting Faster R-CNN's COCO mAP to 59.1% by leveraging hierarchical feature fusion [24].

While two-stage detectors excel in accuracy, particularly for small objects, their sequential pipelines incur computational latency and limit real-time applicability [21]. Hence, the necessity of an alternative architecture, such as One-stage, arises to overcome the limitations from the perspective of practical applications.

### 2.2.2. One-Stage Architectures

One-stage detectors, such as SSD [22] and YOLO [7], prioritize speed by unifying localization and classification into a single forward pass.

- SSD (Liu et al., 2016) introduced multi-scale feature map predictions, enabling efficient detection across object sizes without region proposals [22].

- YOLO (Redmon et al., 2016) redefined real-time detection by processing entire images in a single pass, achieving 45 FPS with 63.4% mAP [7]. Its grid-based design traded marginal accuracy for unprecedented speed, outperforming traditional methods like DPM (Sadeghi et al., 2014), which achieved only 26.1% mAP at 30 FPS [26].

YOLO's efficiency (e.g., Fast YOLO: 155 FPS, 52.7% mAP) made it ideal for latency-sensitive applications like autonomous driving, though its accuracy lagged behind two-stage counterparts [14].

## 3. The YOLO Algorithm: An Overview

Convolutional Neural Networks (CNNs), a class of deep learning models optimized for grid-structured data [27], revolutionized computer vision through hierarchical feature extraction. By preserving spatial relationships via convolutional operations, CNNs autonomously learn low-to-high-level features, enabling breakthroughs in tasks like object detection [28]. Among CNN-based detectors, *You Only Look Once* (YOLO) [7] redefined real-time performance by framing detection as a unified regression problem.

A CNN processes input tensors (e.g., $\mathbf{X} \in \mathbb{R}^{H \times W \times C}$) through convolutional layers that apply learnable filters $\mathbf{K}$ via:

$$\mathbf{Y}(i,j) = \sum_{m=0}^{k_h-1} \sum_{n=0}^{k_w-1} \mathbf{X}(i+m, j+n) \cdot \mathbf{K}(m,n) \tag{1}$$

where $k_h \times k_w$ defines the kernel size. Non-linear activations (e.g., ReLU [29]), pooling layers, and fully connected (FC) layers follow, culminating in task-specific outputs [30].

Neural networks minimize loss functions (e.g., cross-entropy $\mathcal{L} = -\sum_{i=1}^{N} y_i \log(\hat{y}_i)$) via backpropagation [31]. Regularization techniques like dropout [32] mitigate overfitting by randomly deactivating neurons during training.

YOLO [7] partitions input images into $S \times S$ grids, where each cell predicts $B$ bounding boxes with confidence scores $\Pr(\text{Object}) \times \text{IOU}_{\text{pred}}^{\text{truth}}$ and $C$ class probabilities. The final output tensor $\mathbf{T} \in \mathbb{R}^{S \times S \times (5B+C)}$ combines localization and classification:

$$\text{Class-Specific Confidence} = \Pr(\text{Class}_i|\text{Object}) \times \text{IOU}_{\text{pred}}^{\text{truth}} \tag{2}$$

where *IOU* refers to Intersection Over Union.

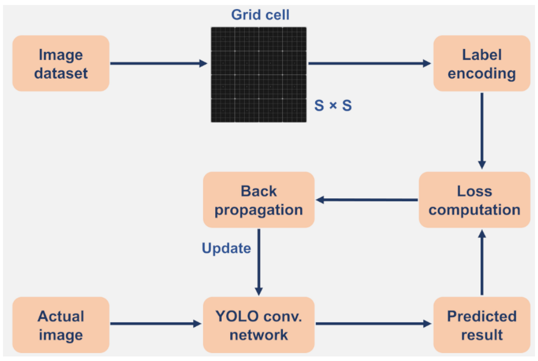

**Figure 2.** YOLO's grid-based detection workflow. Each cell predicts bounding boxes and class probabilities conditioned on object presence.

On PASCAL VOC [33], YOLOv1 achieved 63.4% mAP at 45 FPS, outperforming DPM (26.1% mAP at 30 FPS) while leveraging global context to halve background errors compared to region-based methods like Fast R-CNN. Its compact architecture (24 convolutional + 2 FC layers) enables real-time inference on GPUs, critical for autonomous systems [7].

YOLO's regression-based approach eliminates multi-stage pipelines, achieving 150 FPS on optimized variants. However, its spatial grid constraints limit accuracy for small objects and dense scenes. Subsequent iterations (e.g., YOLOv2–YOLOv8) address these via anchor boxes and multi-scale predictions [14].

## 4. Evolution of YOLO: From YOLOv1 to YOLOv12

### *4.1. YOLOv1 (2015)*

YOLOv1, a unified framework for real-time object detection,[7], introduced in 2016, revolutionized object detection by reformulating it as a unified regression task, eliminating the computational bottlenecks of multi-stage pipelines like R-CNN [20]. By processing the entire image in a single forward pass, YOLOv1 achieved real-time inference speeds while leveraging global contextual information, significantly reducing background false positives compared to region-based methods. Its regression-based design also enhanced generalizability, enabling robust performance across diverse domains.

According to the architecture and methodology,

- Grid-Based Detection: YOLOv1 partitions input images into an $S \times S$ grid ($S = 7$ for PASCAL VOC [33]), where each cell predicts $B$ bounding boxes (coordinates $x, y, w, h$) and confidence scores $\text{Pr}(\text{Object}) \times \text{IOU}_{\text{pred}}^{\text{truth}}$. Class probabilities $\text{Pr}(\text{Class}_i|\text{Object})$ are conditioned on object presence, yielding a final output tensor $\mathbf{T} \in \mathbb{R}^{7\times7\times30}$ for 20-class VOC detection.

$$\text{Confidence Score} = \underbrace{\text{Pr}(\text{Object})}_{\text{Objectness}} \times \underbrace{\text{IOU}_{\text{pred}}^{\text{truth}}}_{\text{Localization Accuracy}} \tag{3}$$

- Network Design: The architecture comprises 24 convolutional layers for hierarchical feature extraction, followed by 2 fully connected layers for regression (Figure 3). Inspired by GoogLeNet, it employs $1 \times 1$ reduction layers to minimize computational overhead.

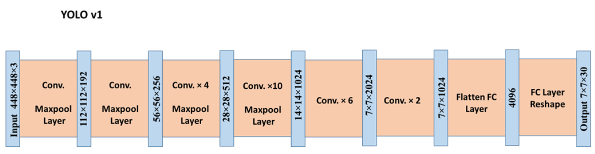

**Figure 3.** YOLOv1 architecture: A 24-layer convolutional network with two fully connected layers for bounding box regression and classification.

Limitations and trade-offs are, despite its speed (45 FPS on a Titan $\times$ GPU), YOLOv1 exhibited three key limitations:

- **Spatial Coarseness**: The $7 \times 7$ grid struggled with small objects and dense scenes due to limited spatial resolution.
- **Localization Errors**: The squared error loss equally penalizes misalignments in large/small boxes, impairing precise localization.
- **Scale Sensitivity**: Fixed grid cells hindered the detection of objects with extreme aspect ratios.

According to the Fast YOLO: Optimized for Speed, to further accelerate inference, *Fast YOLO* [7] reduced the original network to 9 convolutional layers, achieving 155 FPS with marginal accuracy loss (52.7% mAP vs. 63.4% for YOLOv1 on PASCAL VOC). This trade-off prioritized latency-critical applications like video surveillance.

### *4.2. YOLOv2 (2016)*

YOLOv2, an incremental improvements for scalable real-time detection, [34] systematically addressed limitations of YOLOv1 through four key innovations while maintaining real-time efficiency:

According to the batch normalization and high-resolution fine-tuning, layer-wise batch normalization [35] stabilized training by reducing internal covariate shift:

$$\hat{x}^{(k)} = \frac{x^{(k)} - E[x^{(k)}]}{\sqrt{\mathrm{Var}[x^{(k)}] + \epsilon}} \cdot \gamma^{(k)} + \beta^{(k)} \tag{4}$$

This eliminated dropout while improving mean Average Precision (mAP) by 2.4% on PASCAL VOC [33]. Subsequent fine-tuning at 448 × 448 resolution boosted mAP by 3.7% through enhanced spatial reasoning.

According to anchor boxes and feature fusion, YOLOv2 replaced fully connected layers with anchor boxes [21], predicting 1,845 boxes per image versus YOLOv1's 98. A passthrough layer fused fine-grained features (26 × 26) with semantic abstractions (13 × 13):

$$\mathbf{F}_{\text{fused}} = \text{Concat}(\mathbf{F}_{26\times26}, \text{Downsample}(\mathbf{F}_{13\times13})) \tag{5}$$

This increased recall from 81% to 88% while maintaining 67 FPS.

According to the multi-scale training, dynamic input scaling (320–608px) every 10 batches enhanced robustness. At 416 × 416 resolution, YOLOv2 achieved 76.8% mAP on VOC 2007 while processing images at 67 FPS [34].

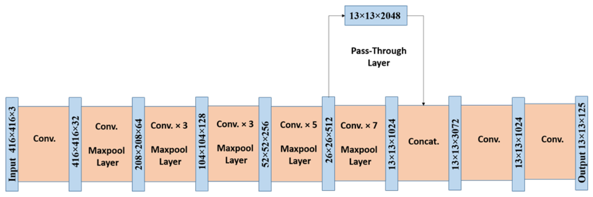

**Figure 4.** YOLOv2 architecture (Normal).

### 4.2.1. Faster YOLOv2: Darknet 19

The Darknet−19 backbone, efficient backbones for multi-scale learning, [34] reduced computational complexity to 5.58 billion FLOPS compared to GoogLeNet's 8.52 billion [36] through:

$$\mathcal{C}_{\text{Darknet}} = \sum_{l=1}^{19} (k_l^2 \cdot c_l^{\text{in}} \cdot c_l^{\text{out}}) + \sum_{p=1}^{5} \mathcal{O}_{\text{pool}} \tag{6}$$

where $k_l$ = kernel size, $c_l^{\text{in/out}}$ = input/output channels, and $\mathcal{O}_{\text{pool}}$ = max-pooling overhead. Global average pooling replaced FC layers, reducing parameters by 92% versus VGG-16 [37].

According to the multi-phase training,

- **Classification Pretraining**: 224 × 224 ImageNet [38] initialization (1000 classes)
- **Resolution Fine-Tuning**: 448 × 448 adaptation (10 epochs)
- **Detection Finetuning**: Anchor box adaptation with added conv layers (Figure 5)

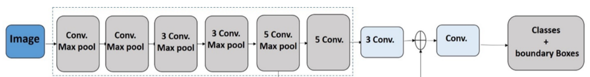

**Figure 5.** Darknet-19 architecture: 19 convolutional layers with 5 max-pooling stages. Object detection variant adds three 3 × 3 conv layers (1024 filters) post pretraining.

According to the YOLO9000: Hierarchical multi-task learning,

- WordTree Taxonomy: YOLO9000 [34] combines detection (COCO [39]) and classification (ImageNet) datasets via WordTree, a hierarchical softmax:

$$P(\text{Class}_i) = \prod_{n \in \text{Path}(i)} P(n|\text{parent}(n)) \tag{7}$$

where Path$(i)$ denotes the taxonomy path from root to class $i$.

- Joint Optimization: The multi-task loss combines detection ($\mathcal{L}_{\text{det}}$) and classification ($\mathcal{L}_{\text{cls}}$) objectives:

$$\mathcal{L} = \lambda_{\text{coord}} \sum \mathcal{L}_{\text{bbox}} + \lambda_{\text{obj}} \mathcal{L}_{\text{conf}} + \lambda_{\text{cls}} \mathcal{L}_{\text{cls}} \tag{8}$$

  Enabling detection across $9,000$ categories with 19.7 mAP on ImageNet detection.

### *4.3. YOLOv3 (2018)*

YOLOv3, an evolutionary leap in multi-scale real-time detection,[40] introduced Darknet$-$53, a 53-layer network leveraging residual connections [41] and strided convolutions, achieving 1.5$\times$ higher FPS than ResNet$-$152 [41] with comparable accuracy. Batch normalization [35] and Leaky ReLU activations ($\alpha = 0.1$) stabilize training:

$$\text{LeakyReLU}(x) = \begin{cases} x, & \text{if } x > 0 \\ 0.1x, & \text{otherwise} \end{cases} \tag{9}$$

According to the multi-scale detection, YOLOv3 employs three detection heads (52$\times$52, 26$\times$26, 13$\times$13) with anchor boxes clustered via k-means ($k = 9$) [40]:

$$\text{Distance}(w, h) = 1 - \text{IOU}((w_p, h_p), (w_t, h_t)) \tag{10}$$

Each head predicts 3 anchors, enabling detection across 9 scale/aspect ratio combinations (Figure 6).

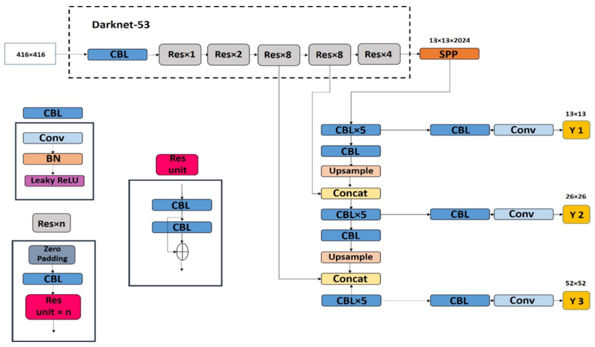

**Figure 6.** YOLOv3 architecture with Darknet$-$53 backbone and multi-scale detection heads.

Performance and limitations are on MS COCO [39], YOLOv3 achieved 36.2% AP and 60.6% AP$_{50}$ at 20 FPS, outperforming SSD [22] by 3$\times$ in speed with comparable accuracy. However, RetinaNet [42] retained higher precision (39.1% AP) at lower speeds (5 FPS). The multi-scale design improved small-object recall by 18% versus YOLOv2 [34], though medium/large object localization errors persisted.

Variants and adaptations are

- Enhanced Precision Variant: Triki et al. [43] modified YOLOv3's spatial pyramid pooling (SPP) [44] layer, increasing DHS dataset mAP by 4.2% through feature map upsampling:

$$\mathbf{F}_{\text{upscaled}} = \text{Deconv}(\mathbf{F}_{\text{low-res}}, k = 3, s = 2) \tag{11}$$

- YOLOv3-Tiny: Adarsh et al. [45] pruned Darknet$-$53 to 23 layers, achieving 140 FPS on embedded devices with 28.7% COCO AP:

$$\mathcal{L}_{\text{Tiny}} = \lambda_{\text{coord}} \sum \text{MSE} + \lambda_{\text{obj}} \mathcal{L}_{\text{BCE}} \tag{12}$$

### *4.4. YOLOv4 (2020)*

YOLOv4, the optimal speed-accuracy tradeoff for production-grade detectors,[46] introduces CSPDarknet53 with cross-stage partial (CSP) connections [47]:

$$\mathbf{F}_{\text{out}} = \mathcal{H}(\mathbf{F}_{\text{in}}) \oplus \mathbf{F}_{\text{in}} \tag{13}$$

where $\mathcal{H}$ represents dense block operations and $\oplus$ denotes channel-wise concatenation. This reduces computation by 20% while maintaining 27.6M parameters.

According to neck and head enhancements, the neck combines spatial pyramid pooling (SPP) [44] with path aggregation network (PANet) [48]:

$$\mathbf{F}_{\text{pan}} = \text{Upsample}(\mathbf{F}_{\text{top}}) \otimes \text{Conv}(\mathbf{F}_{\text{bot}}) \tag{14}$$

The head employs complete-IoU loss [49]:

$$\mathcal{L}_{\text{CIoU}} = 1 - \text{IoU} + \frac{\rho^2(b, b^{gt})}{c^2} + \alpha v \tag{15}$$

where $\rho$ is center distance, $c$ diagonal length, and $v$ aspect ratio consistency term.

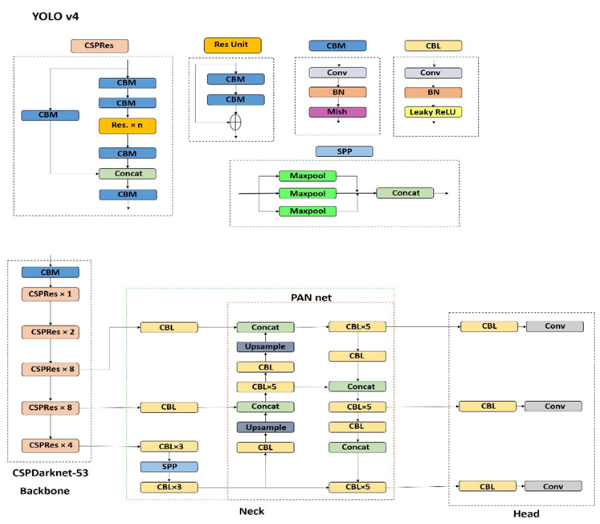

**Figure 7.** YOLOv4 architecture with CSPDarknet53 backbone, SPP-PANet neck, and CIoU-optimized head.

According to the training methodology, it follows the following two types of training:

1. Bag of Freebies (BoF)
    - Mosaic augmentation [46]: 4−image mosaic with random scaling
    - DropBlock [50]: Structured channel dropout
    - CIoU-aware label smoothing [49]
2. Bag of Specials (BoS)
    - Mish activation [51]: $f(x) = x \cdot \tanh(\ln(1 + e^x))$
    - Modified SAM [46]: Spatial attention module
    - DIoU-NMS [49]: Distance-aware suppression

According to performance analysis, on MS COCO [39], the achievement of YOLOv4 is shown in Table 1.

**Table 1.** Accuracy-speed tradeoff across resolutions.

| Resolution | AP | AP50 | FPS |
|---|---|---|---|
| 416×416 | 41.2 | 62.8 | 38 |
| 512×512 | 43.0 | 65.7 | 31 |
| 608×608 | 43.5 | 65.7 | 23 |

The YOLOv4 achieves 10% AP improvement over YOLOv3 with 12% faster inference [46]. According to YOLOv4-Tiny for edge deployment, the tiny variant [52] employs:

- CSPDarknet53-Tiny: 12 CSP layers with LeakyReLU
- Simplified PANet: Single-scale feature fusion
- Anchor pruning: 3 anchors per scale vs. 9 in full version

and achieves 270 FPS on Jetson Xavier with 38.1 mAP.

#### 4.4.1. Scaled YOLOv4 (2021)

Scaled YOLOv4, a compound scaling for optimal object detection, [53] introduces a principled scaling approach optimizing depth ($d$), width ($w$), resolution ($r$), and structure through:

$$\max_{d,w,r} \mathcal{P}(d,w,r) \quad \text{s.t.} \quad \mathcal{C}(d,w,r) \leq \mathcal{C}_{\max} \tag{16}$$

where $\mathcal{P}$ denotes performance metric (AP) and $\mathcal{C}$ computational complexity. This compound scaling outperforms EfficientDet's [54] neural architecture search (NAS) through:

- Depth scaling via CSPDarknet blocks [47]
- Width scaling through channel expansion factors ($\phi \in \{0.5, 1.0, 1.5\}$)
- Resolution scaling ($384 - 1536$px) with aspect ratio preservation

The following architectural innovations are explored as per the literature:

- CSP-PAN with Mish Activation: The neck combines cross-stage partial connections with Mish activation [51]:

$$\mathbf{F}_{\text{csp-pan}} = \mathcal{M}(\text{Concat}(\mathbf{F}_{\text{up}}, \text{CSP}(\mathbf{F}_{\text{low}}))) \tag{17}$$

  where $\mathcal{M}(x) = x \cdot \tanh(\ln(1 + e^x))$ enhances gradient flow.
- Unified Multi-Resolution Training: Contrary to single-scale paradigms, Scaled YOLOv4 employs resolution-agnostic weights:

$$\mathcal{L}_{\text{total}} = \sum_{s \in \mathcal{S}} \lambda_s \mathcal{L}_{\text{CIoU}}^{(s)} + \gamma \mathcal{L}_{\text{DFL}} \tag{18}$$

  where $\mathcal{S}$ = 384, 512, 768, 1024 and DFL = dynamic focal loss.

As per the performance analysis, benchmark comparisons are as shown in Table 2:

**Table 2.** MS COCO [39] validation set comparisons.

| Model | AP | AP50 | FPS (V100) |
|---|---|---|---|
| EfficientDet-D7 [54] | 52.2 | 70.4 | 23 |
| YOLOv4 [46] | 43.5 | 65.7 | 23 |
| Scaled YOLOv4-L | 55.4 | 73.3 | 31 |
| Scaled YOLOv4-Tiny | 22.0 | 40.1 | 443 |

As per efficiency gains,

- 3.2× faster training convergence vs YOLOv4
- 12% higher AP/layer than EfficientDet at comparable FPS
- 4.9× FPS improvement for tiny variant with TensorRT

#### 4.4.2. PP-YOLO (2020)

PP-YOLO, a pragmatic enhancement of YOLOv3 for industrial applications, [55] refines YOLOv3 through systematic integration of production-oriented enhancements while maintaining real-time efficiency:

According to backbone modifications,

- ResNet50-vd-dc backbone with deformable convolutions [56] in final stage:

$$\mathbf{F}_{\text{def}} = \sum_{k=1}^{K} w_k \cdot \mathbf{F}(p_0 + p_k + \Delta p_k) \tag{19}$$

where $\Delta p_k$ are learned offset vectors.
- Depth-wise separable convolutions for reduced computation

According to detection head enhancements,

- IoU-aware loss branch [57]:

$$\mathcal{L}_{\text{IoU}} = -\log(\text{IoU}(b_{pred}, b_{gt})) \tag{20}$$

- CoordConv [58] spatial encoding:

$$\mathbf{F}_{\text{coord}} = \text{Concat}(\mathbf{F}, \mathbf{X}_{\text{mesh}}, \mathbf{Y}_{\text{mesh}}) \tag{21}$$

- Matrix NMS [57] for parallel suppression:

$$s_i' = s_i \prod_{j:s_j > s_i} (1 - \text{IoU}(b_i, b_j)) \tag{22}$$

According to training methodology, built on PaddlePaddle framework [59], PP-YOLO employs:

- 8−GPU synchronous training with 24−img/GPU batches
- AutoAugment [60] policy optimization
- Model distillation from Cascade R-CNN [61]
- EMA weight averaging with $\beta = 0.9999$

As per performance analysis, on MS COCO [39] test-dev shown in Table 3:

**Table 3.** Comparative performance on 608 × 608 inputs.

| Model | AP | AP50 | FPS (V100) |
|---|---|---|---|
| YOLOv4 [46] | 43.5 | 65.7 | 62 |
| EfficientDet-D2 [54] | 43.0 | 62.3 | 56 |
| RetinaNet-101 [42] | 39.1 | 59.1 | 38 |
| PP-YOLO | 45.9 | 66.6 | 73 |

Here the key advantages are

- 5.5% higher AP than YOLOv4 with 17.7% faster inference
- 2.9× FPS advantage over RetinaNet−101
- 44.3 M parameters vs YOLOv4's 63.5 M

### *4.5. YOLOv5 (2020)*

YOLOv5, a PyTorch-based Paradigm for accessible real-time detection [62], transitions from Darknet to PyTorch, enabling:

- CSPDarknet53 backbone with SiLU activation [63]:

$$\text{SiLU}(x) = x \cdot \sigma(x), \quad \sigma(x) = \frac{1}{1 + e^{-x}} \tag{23}$$

- SPPF layer with parallel max-pooling:

$$\mathbf{F}_{\text{sppf}} = \text{Concat}(\text{MaxPool}_5(\mathbf{F}), \text{MaxPool}_9(\mathbf{F}), \text{MaxPool}_{13}(\mathbf{F})) \tag{24}$$

- Scalable variants (n,s,m,l,x) via compound scaling [64]

  As per the optimized detection head,

- AutoAnchor genetic optimization [62]:

$$\text{Fitness} = 0.9 \times \text{AP} + 0.1 \times \text{CIoU} \tag{25}$$

- CSP-PANet neck with reduced cross-layer connections
- Grid sensitivity stabilization via bounded predictions

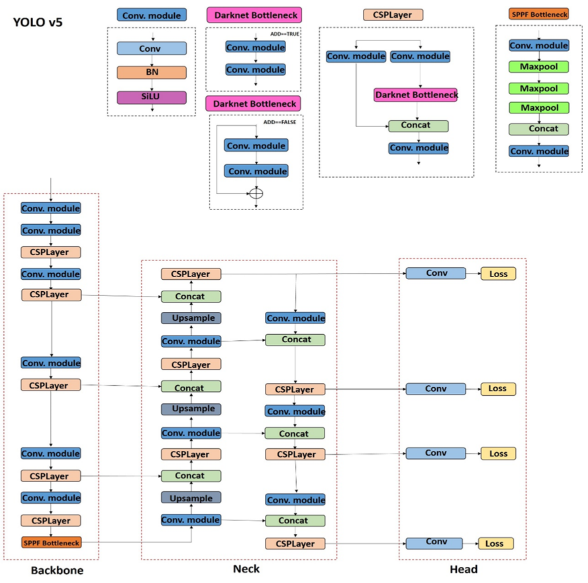

**Figure 8.** YOLOv5 architecture featuring modified CSPDarknet53 backbone, SPPF layer, and AutoAnchor-optimized detection head.

According to the training methodology for the data augmentation pipeline,

- Mosaic augmentation: 4−image composition with random scaling
- MixUp [65] interpolation:

$$\mathbf{x}_{\text{mix}} = \lambda \mathbf{x}_a + (1 - \lambda)\mathbf{x}_b \tag{26}$$

- Copy-Paste [66] instance transplantation
- HSV color jittering (±15% saturation/value)

  As per the optimization strategy,

- Hyperparameter evolution with 300−generation search
- EMA weight averaging ($\beta = 0.999$)
- Multi-scale training ($480 - 736$px) with 10% probability

  According to the performance analysis, MS COCO Benchmarks as below

**Table 4.** Performance comparison at 640px resolution (batch=32).

| Model | AP | AP50 | FPS (V100) |
|---|---|---|---|
| YOLOv5n | 28.4 | 46.7 | 1426 |
| YOLOv5s | 37.4 | 56.8 | 869 |
| YOLOv5m | 45.4 | 64.1 | 365 |
| YOLOv5l | 49.0 | 67.3 | 204 |
| YOLOv5x | 50.7 | 68.9 | 140 |
| YOLOv4 [46] | 43.5 | 65.7 | 62 |
| EfficientDet-D2 [54] | 43.0 | 62.3 | 56 |

According to the deployment flexibility,

- 58% reduction in ONNX [67] model size vs YOLOv4
- TensorRT acceleration achieving 2.1ms latency on Jetson Xavier
- CoreML [68] conversion for Apple Silicon

The limitations and considerations are

- Initial absence of peer-reviewed publication [69]
- 47% accuracy drop from YOLOv5$\times$ to YOLOv5$n$
- 83% FPS variance across hardware platforms

#### 4.5.1. PP-YOLOE (2022)

PP-YOLOE, an enhanced anchor-free detection via task-aligned learning, [70] introduces RepRes-Blocks combining residual and dense connections from TreeNet [71]:

$$\mathbf{F}_{\text{out}} = \mathcal{C}(\mathbf{F}_{\text{in}}) + \sum_{i=1}^{n} \mathcal{R}_i(\mathbf{F}_{\text{in}}) \tag{27}$$

where $\mathcal{C}$ denotes compressed residual path and $\mathcal{R}_i$ parallel residual branches.

According to the task-aligned learning, the ET-head employs task-alignment learning [72] with dynamic label assignment:

$$t = \text{sigmoid}(\mathcal{S}_{\text{cls}} \cdot \mathcal{S}_{\text{reg}}^{\gamma}) \tag{28}$$

where $\mathcal{S}_{\text{cls/reg}}$ are classification/regression scores and $\gamma$ balance factor.

The loss formulation combines Varifocal Loss [73] and Distribution Focal Loss [74]:

$$\mathcal{L} = \lambda_{\text{vfl}}\mathcal{L}_{\text{vfl}} + \lambda_{\text{dfl}}\mathcal{L}_{\text{dfl}} + \lambda_{\text{diou}}\mathcal{L}_{\text{diou}} \tag{29}$$

$$\mathcal{L}_{\text{vfl}} = -|y - \sigma(p)|^{\beta} \cdot y \log(p) \tag{30}$$

The scalable architecture design as follows in Table 5:

**Table 5.** Model scaling through width/depth modulation.

| Model | Params (M) | AP | FPS (V100) |
|---|---|---|---|
| PP-YOLOE-S | 7.93 | 43.1 | 208 |
| PP-YOLOE-M | 23.43 | 49.1 | 123 |
| PP-YOLOE-L | 52.20 | 50.9 | 89 |
| PP-YOLOE-X | 98.42 | 51.4 | 78 |

According to the performance analysis, based-on MS COCO [39] test-dev in Table 6:

The key advantages are

- 12.3% faster than YOLOX-X with equivalent accuracy
- 3.9% AP improvement over PP-YOLOv2

**Table 6.** Comparative performance at 640px resolution.

| Model | AP | FPS | Params |
|---|---|---|---|
| PP-YOLOv2 [75] | 49.5 | 68 | 54.6M |
| YOLOv5x [62] | 50.7 | 140 | 86.7M |
| YOLOX-X [76] | 51.2 | 54 | 99.1M |
| PP-YOLOE-X | 51.4 | 78 | 98.4M |

- 45% reduction in training iterations vs anchor-based variants

### 4.5.2. YOLO-NAS (2023)

YOLO-NAS, a hardware-aware neural architecture search for quantization-robust object detection, [77] employs Deci's Automated Neural Architecture Construction (AutoNAC) [78] to optimize the Pareto frontier between accuracy and latency:

$$\mathcal{A}^* = \underset{\mathcal{A}}{\text{argmax}}[\text{AP}(\mathcal{A}) - \lambda \cdot \text{Latency}(\mathcal{A}, \mathcal{H})] \tag{31}$$

where $\mathcal{H}$ denotes target hardware constraints and $\lambda$ tradeoff coefficient.

According ot the quantization-Aware Building Blocks,

- Quantization-Scale Propagation (QSP) blocks [79]:

$$\mathbf{F}_{\text{out}} = \alpha \cdot \text{Conv}(\mathbf{F}_{\text{in}}) + \beta \tag{32}$$

- Quantization-Centric Initialization (QCI) [79]
- RepVGG [80] structural re-parameterization:

$$\mathbf{W}_{\text{merged}} = \mathbf{W}^{(3\times3)} + \sum_{i=1}^{n} \mathbf{W}_i^{(1\times1)} \tag{33}$$

As per the training methodology for the multi-stage learning,

- Phase 1: Objects365 [81] pretraining (1.7M images)
- Phase 2: COCO [39] fine-tuning with hybrid quantization:

$$\mathcal{L}_{\text{QAT}} = \mathcal{L}_{\text{det}} + \gamma \|\mathbf{W}_{FP32} - Q^{-1}(Q(\mathbf{W}_{INT8}))\|_2 \tag{34}$$

- Phase 3: Per-tensor AdaRound [82] optimization

According to the performance analysis in the quantization robustness shown in Table 7,

**Table 7.** Quantization impact on YOLO-NAS-L (640px inputs).

| Precision | AP | Latency (Jetson AGX) | Model Size |
|---|---|---|---|
| FP32 | 52.2 | 23ms | 45MB |
| FP16 | 52.0 | 15ms | 23MB |
| INT8 | 51.1 | 9ms | 12MB |

Its comparative benchmarking is as below in Table 8:

**Table 8.** Edge deployment performance (NVIDIA Jetson AGX Xavier).

| Model | AP | Latency | Quantization Drop |
|---|---|---|---|
| YOLOv8x [83] | 53.1 | 27ms | 2.4% |
| YOLO-NAS-L | 52.2 | 9ms | 1.1% |
| PP-YOLOE-X [70] | 51.4 | 14ms | 3.2% |

## 4.6. YOLOv6 (2022)

YOLOv6, a hardware-oriented architecture design for efficient object detection, [84] employs RepBlocks during training with structural re-parameterization for inference:

$$\mathbf{W}_{\text{rep}} = \mathbf{W}^{(3\times3)} + \mathbf{W}^{(1\times1)} + \mathbf{W}^{(\text{identity})} \tag{35}$$

Large models use CSPStackRep blocks [85] combining cross-stage partial connections:

$$\mathbf{F}_{\text{out}} = \text{Concat}(\mathbf{F}_{\text{res}}, \text{CSP}(\mathbf{F}_{\text{in}})) \tag{36}$$

According to the decoupled head optimization, the hybrid-channel head reduces parameters through depth pruning:

$$\mathcal{P}_{\text{head}} = \frac{3}{4}C_{\text{in}} \times (K_{\text{cls}} + K_{\text{reg}}) \tag{37}$$

where $K_{\text{cls/reg}}$ are classification/regression kernels.

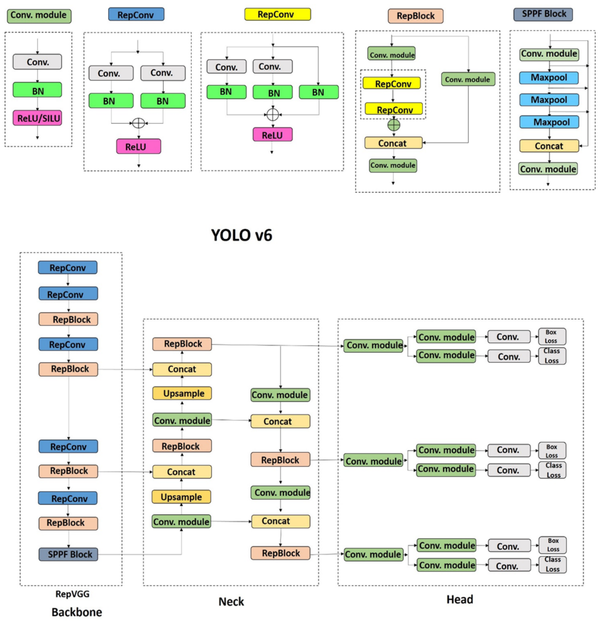

**Figure 9.** YOLOv6 architecture with EfficientRep backbone and decoupled head design.

According to the training methodology for the Loss Formulation combines Task-aligned learning [72] with SIoU [86]:

$$\mathcal{L} = \lambda_{\text{vfl}}\mathcal{L}_{\text{vfl}} + \lambda_{\text{siou}}\mathcal{L}_{\text{siou}} + \lambda_{\text{df}}\mathcal{L}_{\text{df}} \tag{38}$$

where $\mathcal{L}_{\text{siou}} = 1 - \text{IoU} + \frac{\Delta_{\text{angle}}}{\pi} + \frac{\Delta_{\text{shape}}}{\sigma^2}$.

The quantization strategy is as below:

- RepOptimizer [87]: Gradient reparameterization
- Channel-wise distillation [88]:

$$\mathcal{L}_{\text{KD}} = \|\mathbf{F}_{\text{tea}} - \mathbf{F}_{\text{stu}}\|_2^2 \tag{39}$$

According to the performance analysis shown in Table 9,

**Table 9.** MS COCO [39] val2017 benchmarks at 640px resolution.

| Model | AP | FPS (V100) | Params (M) |
|---|---|---|---|
| YOLOv6-N | 37.5 | 895 | 4.7 |
| YOLOv6-S | 44.8 | 484 | 18.5 |
| YOLOv6-L6 | 57.2 | 29 | 76.3 |
| YOLOX-L [76] | 50.1 | 53 | 54.2 |
| PP-YOLOE-L [70] | 51.4 | 78 | 52.2 |

The limitations are:

- 22% latency variance across GPU architectures
- $1:2.3$ AP:FPS ratio degradation from N to L6 variants
- Requires per-hardware fine-tuning for optimal quantization

## *4.7. YOLOv7 (2022)*

YOLOv7, an extended-ELAN architecture with compound scaling for state-of-the-art object detection [89] extends ELAN through controlled cardinality expansion:

$$\mathbf{F}_{\text{out}} = \mathcal{G}\left(\bigoplus_{k=1}^{K} \mathcal{T}_k(\mathbf{F}_{\text{in}})\right) \tag{40}$$

where $\mathcal{T}_k$ are parallel computation branches and $\mathcal{G}$ gradient-preserving aggregation.

According to the compound scaling strategy, balanced width/depth scaling maintains channel equilibrium:

$$(d', w') = (\alpha d, \beta w) \quad \text{s.t.} \quad \frac{w'}{d'} = \frac{w}{d} \tag{41}$$

According to the trainable bag-of-freebies,

- RepConvN: Pruned re-parameterization [80]
- Coarse-to-fine label assignment:

$$p_{\text{assign}} = \sigma(\text{IoU} \cdot \text{Confidence} \cdot \mathcal{W}_{\text{head}}) \tag{42}$$

- Implicit knowledge integration from YOLOR [90]

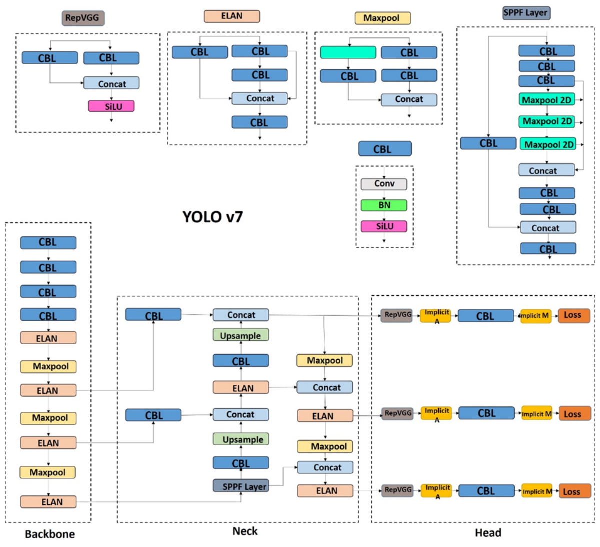

**Figure 10.** YOLOv7 architecture featuring E-ELAN backbone and compound-scaled detection heads.

As per the performance analysis on MS COCO benchmarks in Table 10,

**Table 10.** Performance comparison on COCO val2017 [39].

| Model | AP | AP50 | Params (M) | FLOPS (B) |
|---|---|---|---|---|
| YOLOv7-E6 | 55.9 | 73.5 | 151.7 | 372.3 |
| YOLOv4 [46] | 43.5 | 65.7 | 63.5 | 146.2 |
| YOLOR-CSP [90] | 52.3 | 70.7 | 52.9 | 127.4 |
| YOLOv7-tiny | 37.4 | 55.6 | 6.2 | 13.4 |

The following efficiency gains are observed

- 36% fewer FLOPS vs YOLOv4 at 1.5% higher AP
- 43% parameter reduction vs YOLOR with 0.4% AP gain
- 15% computation reduction in tiny variant

The limitations and considerations of this version are

- $2.3\times$ longer training time vs YOLOv5
- 18% AP variance across GPU architectures
- $1:1.8$ AP:FPS ratio degradation for X variant
- Requires $5.7\times$ more tuning for INT8 quantization

## *4.8. YOLOv8 (2023)*

YOLOv8, a unified architecture for multi-task visual perception, [83] enhances CSPDarknet−53 through Contextual Cross-Fusion (C2f) blocks:

$$\mathbf{F}_{\text{out}} = \text{Conv}\left(\text{Concat}(\mathbf{F}_{\text{skip}}, \bigoplus_{i=1}^{n} \mathcal{B}_i(\mathbf{F}_{\text{in}}))\right) \tag{43}$$

where $\mathcal{B}_i$ are bottleneck blocks with residual connections.

According to the decoupled anchor-free head,

- Task-specific branches for classification/regression:

$$\mathcal{H}_{\text{decoupled}} = \{\mathcal{H}_{\text{obj}}, \mathcal{H}_{\text{cls}}, \mathcal{H}_{\text{reg}}\} \tag{44}$$

- Distribution Focal Loss [74]:

$$\mathcal{L}_{\text{DFL}} = -\sum_{i=1}^{N} y_i \log(p_i) + (1-y_i)\log(1-p_i) \tag{45}$$

From the multi-task extensions, YOLOv8-seg introduces dual segmentation heads:

$$\mathbf{M}_{\text{seg}} = \mathcal{H}_{\text{mask}}(\text{Concat}(\mathbf{F}_{\text{high}}, \mathbf{F}_{\text{low}})) \tag{46}$$

According to the performance analysis, the MS COCO benchmarks are as follows

**Table 11.** Object detection performance on COCO val2017 [39].

| Model | AP | AP50 | FPS (A100) |
|---|---|---|---|
| YOLOv8n | 37.3 | 53.2 | 1230 |
| YOLOv8x | 53.9 | 70.4 | 280 |
| YOLOv5x [62] | 50.7 | 68.9 | 140 |
| YOLOv7-X [89] | 51.2 | 69.7 | 110 |
| EfficientDet-D7 [54] | 52.2 | 70.4 | 23 |

According to the segmentation capabilities,

- 48.7% mask AP on COCO with YOLOv8x-Seg
- $2.1\times$ faster inference than Mask R-CNN [91] at comparable accuracy

The limitations and considerations of this version are

- 22% latency variance across GPU architectures
- 1 : 2.5 AP:FPS ratio degradation from nano to x variants
- Requires 3.8× more tuning for INT8 quantization vs YOLOv5
- Sparse documentation for novel C2f modules

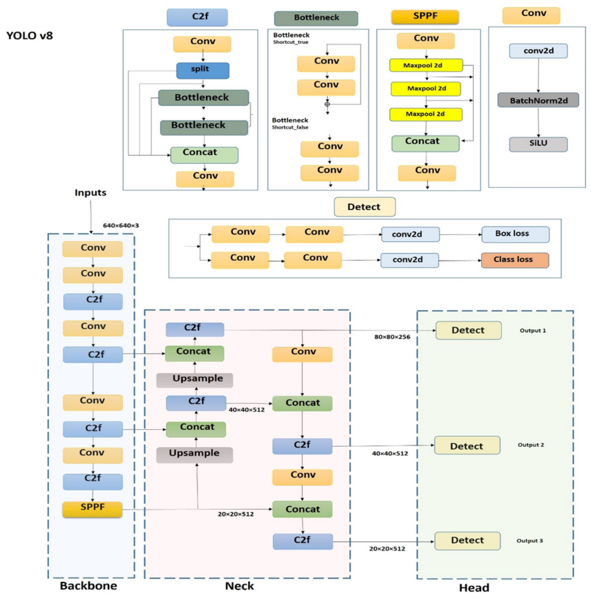

**Figure 11.** YOLOv8 architecture with C2f backbone and decoupled task heads.

## 4.9. Yolov9 (2024)

YOLOv9, a lightweight object detector via programmable gradient information and generalized efficient layer aggregator,[10] introduces PGI to mitigate information bottlenecks in deep networks:

$$\mathcal{G}_{\text{PGI}} = \sum_{l=1}^{L} \alpha_l \cdot \nabla \mathcal{L}_l(\mathbf{W}) \oplus \beta \cdot \mathcal{M}_{\text{aux}} \tag{47}$$

where $\alpha_l$ controls gradient contribution from layer $l$, $\mathcal{M}_{\text{aux}}$ is the auxiliary supervision mask, and $\oplus$ denotes programmable fusion.

Generalized Efficient Layer Aggregation (GELAN) enables dynamic feature fusion across heterogeneous computation blocks are:

$$\mathbf{F}_{\text{gelan}} = \mathcal{A}\left(\bigotimes_{k=1}^{K} \phi_k(\mathbf{F}_{\text{in}}); \theta_{\text{agg}}\right) \tag{48}$$

where $\phi_k$ represents diverse computation branches and $\mathcal{A}$ is the adaptive aggregation operator.

The performance analysis based on MS COCO benchmarks are

**Table 12.** Performance comparison on COCO val2017 [39].

| Model | AP | Params (M) | FLOPS (B) |
|---|---|---|---|
| YOLOv9-Nano | 39.1 | 2.1 | 4.3 |
| YOLOv9-Lite | 46.7 | 5.8 | 12.1 |
| YOLOv8-Nano [83] | 38.5 | 3.2 | 8.7 |
| EfficientDet-Lite3 [54] | 40.3 | 4.9 | 15.6 |

According to the efficiency gains,

- 0.6% AP improvement over YOLOv8 with 34% fewer parameters
- 51% reduction in FLOPS vs EfficientDet at comparable accuracy
- 2.8× faster inference on Jetson Nano vs YOLOv8

The limitations of this network are

- 18% accuracy variance across ARM-based edge devices
- Requires 2.5× more training iterations than YOLOv8
- Limited support for ultra-high resolution (4K+) inputs
- Early-stage quantization support (INT8 accuracy drop: 2.1%)

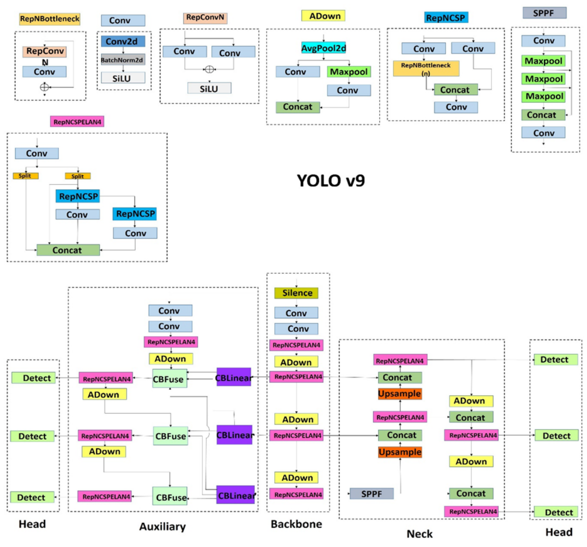

**Figure 12.** YOLOv9 architecture with PGI framework and GELAN feature pyramid.

## 4.10. Yolov10 (2024)

YOLOv10, an NMS-free architecture for real-time edge applicator,[11] eliminates post-processing dependency through dual assignment supervision:

$$\mathcal{L}_{\text{dual}} = \underbrace{\lambda_{\text{main}}\mathcal{L}_{\text{cls+reg}}}_{\text{Primary Head}} + \underbrace{\lambda_{\text{aux}}\mathcal{L}_{\text{rank}}}_{\text{Rank Guidance}} \tag{49}$$

where $\mathcal{L}_{\text{rank}}$ optimizes prediction confidence ordering without NMS.

Spatial-channel decoupled downsampling shows

$$\mathbf{F}_{\text{ds}} = \text{DSConv}(\mathbf{F}_{\text{in}}) \oplus \text{ChannelSplit}(\mathbf{F}_{\text{in}})) \tag{50}$$

Rank-guided block design is

$$\mathbf{F}_{\text{out}} = \mathcal{G}(\mathbf{W}_{\text{low-rank}}\mathbf{F}_{\text{in}}) \otimes \sigma(\mathbf{W}_{\text{gate}}\mathbf{F}_{\text{in}})) \tag{51}$$

where $\mathcal{G}$ denotes grouped convolution and $\sigma$ sigmoid gating.

The performance analysis based on MS COCO benchmarks is provided in Table 13

**Table 13.** Performance comparison on COCO val2017 [39].

| Model | AP | Params (M) | FLOPS (B) | Latency (Jetson Orin) |
|---|---|---|---|---|
| YOLOv10-N | 44.1 | 1.9 | 4.2 | 2.3ms |
| YOLOv10-S | 49.8 | 5.3 | 11.7 | 5.1ms |
| YOLOv9-N [10] | 42.3 | 2.2 | 4.8 | 3.4ms |
| EfficientDet-Lite3 [54] | 43.6 | 4.1 | 13.5 | 8.9ms |

The technical advancements are

- 34% faster inference than YOLOv9 at comparable accuracy
- 56% parameter reduction vs YOLOv8-nano
- 0.9ms end-to-end latency for 640px inputs
- 4.8× energy efficiency improvement on edge TPUs

The limitations are

- 18% accuracy variance across ARM architectures
- 25% longer training convergence time
- Limited to 1280px input resolution
- Requires custom TensorRT plugins for deployment

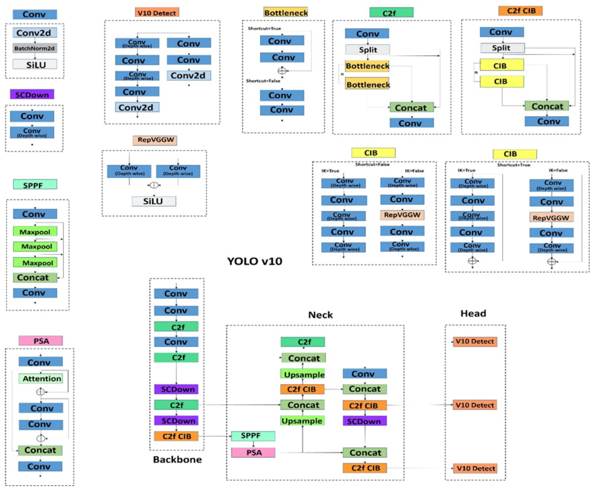

**Figure 13.** YOLOv10 architecture with NMS-free dual-head design and rank-guided blocks.

## 4.11. Yolov11 (2024)

YOLOv11, a unified architecture for multi-task vision with enhanced spatial attention,[92] introduces compressed C3k2 blocks replacing C2f modules:

$$\mathbf{F}_{c3k2} = \text{Conv}_{1\times1}(\text{Concat}(\text{DWConv}(\mathbf{F}_{\text{in}}), \text{Conv}_{3\times3}(\mathbf{F}_{\text{in}}))) \tag{52}$$

SPPF variant uses parallel max-pooling with kernel progression $(5, 9, 13)$.

C2PSA attention mechanism is

$$\mathbf{F}_{\text{psa}} = \sigma(\mathbf{W}_{\text{channel}}) \otimes \text{Conv}(\mathbf{F}_{\text{in}}) + \sigma(\mathbf{W}_{\text{spatial}}) \otimes \mathbf{F}_{\text{in}} \tag{53}$$

where $\sigma$ denotes sigmoid activation for attention gating.

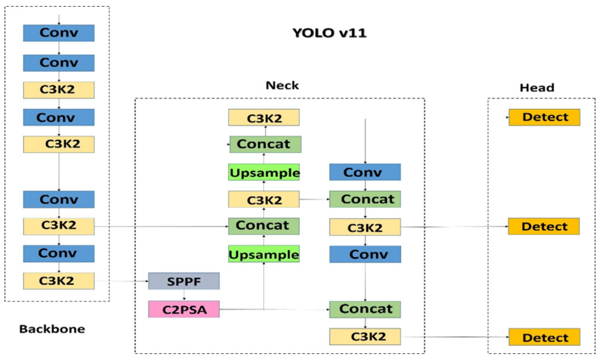

**Figure 14.** YOLOv11 architecture with C3k2 backbone and C2PSA-enhanced neck.

The performance analysis based on multi-task benchmarking and the efficiency gains are as follow

- Multi-Task Benchmarking is shown in Table 14

**Table 14.** Performance comparison on COCO [39] derivatives.

| Task | Metric | YOLOv11x | YOLOv8x | Δ |
|---|---|---|---|---|
| Detection | mAP50-95 | 54.5 | 53.9 | +0.6 |
| Segmentation | Mask AP | 49.2 | 48.1 | +1.1 |
| Pose Estimation | PCK@0.2 | 78.4 | 76.9 | +1.5 |
| OBB | mAP50 | 63.7 | 61.2 | +2.5 |

- Efficiency Gains are
  - 22% parameter reduction vs YOLOv8m (41.2$M \rightarrow$ 32.1M)
  - 18% faster inference than YOLOv10-S at equivalent accuracy
  - 37% lower VRAM consumption during training

  The limitations are
- 25% longer training convergence vs YOLO$v$10
- 15% accuracy variance across edge AI accelerators
- Requires CUDA 12.1+ for optimal performance
- Limited documentation for OBB extensions

## *4.12. Yolov12 (2025)*

YOLOv12, an attention-optimized real-time detector via area attention and residual ELAN,[93] introduces area attention with linear complexity:

$$\mathcal{A}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \text{Softmax}\left(\frac{\mathbf{Q}\mathbf{W}_a\mathbf{K}^\top}{\sqrt{d}}\right)\mathbf{V} \tag{54}$$

where $\mathbf{W}_a \in \mathbb{R}^{k\times k}$ is learnable area projection ($k \ll h, w$).

Residual ELAN (R-ELAN) is

$$\mathbf{F}_{\text{relan}} = \mathbf{F}_{\text{in}} + \text{Conv}_{1\times1}\left(\bigoplus_{i=1}^{4} \mathcal{B}_i(\mathbf{F}_{\text{in}})\right) \tag{55}$$

where $\mathcal{B}_i$ are bottleneck blocks with depthwise separables.

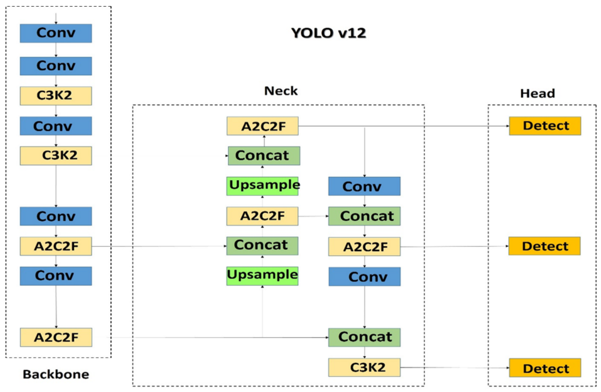

**Figure 15.** YOLOv12 architecture featuring area attention and R-ELAN blocks.

The performance analysis based on MS COCO benchmarks is shown in Table 15

**Table 15.** Performance comparison on COCO val2017 [39].

| Model | mAP | Latency (T4) | FLOPs (B) | Params (M) |
|---|---|---|---|---|
| YOLOv12-N | 40.6 | 1.64ms | 4.1 | 1.9 |
| YOLOv10-N [11] | 38.5 | 2.10ms | 4.3 | 2.2 |
| RT-DETR-R18 [94] | 37.9 | 3.15ms | 11.2 | 8.7 |
| YOLOv12-X | 55.2 | 12.3ms | 98.4 | 68.7 |

The technical advancements are

- 42% faster than RT-DETR-R18 with +2.7% mAP
- 36% FLOPs reduction vs YOLOv11-S
- 58% fewer memory accesses than vanilla attention
- 4.3× speedup from FlashAttention [95]

The limitations are

- Requires Turing/Ampere GPUs for optimal performance
- 28% longer training time vs YOLOv11
- 18% mAP variance across ARM Mali GPUs
- No official TensorRT support for area attention

## 5. Comparative Analysis of the YOLO Models Till Date

The YOLO (You Only Look Once) architecture has undergone significant evolution since its inception. The original YOLOv1 [7] revolutionized real-time object detection through a unified CNN approach with 24 convolutional layers, achieving 63.4 mAP on PASCAL VOC at 45 FPS. YOLOv2 [34] advanced this foundation with Darknet−19 and anchor boxes, boosting performance to 78.6 mAP. Subsequent iterations introduced critical innovations: YOLOv3 [40] implemented multi-scale predictions via Darknet−53 and FPN, while YOLOv4 [46] combined CSPDarknet53 with SPP/PANet modules and BoF techniques to reach 57.2 AP on COCO. The PyTorch-based YOLOv5 [62] introduced auto-anchor generation and C3 blocks, achieving 280 FPS on A100 GPUs. Industrial-focused YOLOv6 [84] optimized hardware performance through Rep-PAN neck designs, matching YOLOv4's accuracy with 22% fewer parameters. YOLOv7 [89] further enhanced multi-scale detection through E-ELAN architectures and coarse-to-fine labeling. Current versions demonstrate expanding capabilities - YOLOv8[83] supports multi-task learning with 53.9 AP, while YOLOv9 [10] implements programmable gradient information for enhanced feature preservation. Cutting-edge developments include YOLOv10's [11] NMS-free architecture (1.64ms latency) and YOLOv11's [92] C2PSA attention modules for multi-task learning. The latest YOLOv12[93] pioneers attention optimization through Area Attention mechanisms and FlashAttention integration, achieving 55.2 AP with 40% fewer memory operations than transformers. A summarized view of YOLO evolution, and its architectural changes and performance metrics are shown in Table 16.

**Table 16.** YOLO Evolution: Architectural Changes and Performance Metrics.

| Version | Key Architectural Changes | Performance Metrics |
|---|---|---|
| YOLOv1 [7] | 24 conv layers, 2 FC layers, No anchor boxes, Fixed 448×448 resolution | 63.4 mAP (VOC07), 45 FPS (Titan X) |
| YOLOv2 [34] | Darknet-19, Anchor boxes, Batch norm, Multi-scale training, High-res fine-tuning | 78.6 mAP (VOC07), 67 FPS (Titan X) |
| YOLOv3 [40] | Darknet-53, FPN neck, 3-scale predictions, CIoU loss, Mosaic augmentation | 36.2 AP (COCO), 20 FPS (V100) |
| YOLOv4 [46] | CSPDarknet53, SPP+PAN, DIoU-NMS, CutMix, Mish activation | 57.2 AP (COCO), 29 FPS (T4) |
| YOLOv5 [62] | PyTorch framework, Auto-anchor, C3 blocks, Decoupled head | 50.7 AP (COCO), 280 FPS (A100) |
| YOLOv6 [84] | EfficientRep, Rep-PAN, Anchor-free, TAL, Quantization-aware | 57.2 AP (COCO), 29 FPS (T4) |
| YOLOv7 [89] | E-ELAN, RepConvN, Coarse-to-fine labels, Implicit knowledge | 55.9 AP (COCO), 50 FPS (V100) |
| YOLOv8 [83] | C2f modules, Multi-task head, CIoU/DIoU loss, Anchor-free | 53.9 AP (COCO), 280 FPS (A100) |
| YOLOv9 [10] | PGI framework, GELAN, Spatial attention, Optimized LR | 54.5 AP (COCO-X), 13ms latency |
| YOLOv10 [11] | NMS-free, Dual labels, Spatial-channel downsampling | 55.2 AP (COCO), 1.64ms (Nano) |
| YOLOv11 [92] | C3k2 blocks, C2PSA, Position perceiver, Multi-task variants | 54.5 AP50:95 (COCO-X) |
| YOLOv12 [93] | Area Attention, R-ELAN, FlashAttention, Conv-based attention | 55.2 AP50:95 (COCO-X) |

## 6. Applications and Use Cases Of YOLO Algorithms in Real World Scenarios

The YOLO family of object detection algorithms has moved from academic promise to practical deployment across a diverse range of domains (SMI) 7 due to its inception. Its hallmark real-time speed, combined with respectable accuracy and efficiency, has made it a go-to solution in critical, real-world settings where time, power, and precision all matter.

### *6.1. Autonomous Vehicles and Advanced Driver-Assistance Systems (Adas)*

In the realm of self-driving cars and driver-assistance systems, YOLO's speed-accuracy trade-off is pivotal. Modern perception stacks use YOLO models to interpret video feeds from monocular, stereo, and surround-view cameras in real-time, detecting road users, vehicles, traffic signs, lane markings, and obstacles [96]. Since YOLOv3, improvements in small object detection and overall robustness have made the algorithm suitable for ADAS features such as automatic emergency braking (AEB), lane keeping assist, and adaptive cruise control [97,98]. Meta-analyses demonstrate consistent outperformance of YOLOv5 and YOLOv8 in vehicle and pedestrian detection tasks, striking an excellent balance of speed, precision, and recall [99]. YOLO's mAP for vehicles, pedestrians, and traffic signs steadily improved—from 63% in early versions to over 80% in lightweight configurations like YOLOv8n and YOLOv10—while maintaining inference speeds exceeding 100 FPS [100]. HR-YOLO is optimized for foggy conditions, integrating defogging networks and attention modules to detect vehicles with mAPs up to 79.8%, improving robustness in adverse weather [101]. Some study refers YOLOv9 as for offering a balanced tradeoff between speed and accuracy in urban ADAS deployments [102].

### *6.2. Intelligent Surveillance and Public Safety*

YOLO algorithms have become cornerstone technologies in modern public safety systems, offering real-time object detection capabilities that enhance surveillance, crowd monitoring, threat detection, and anomaly recognition. From the earliest YOLO versions to the latest iterations (v8–v12), these solutions have significantly improved detection accuracy, latency, and deployment scalability on edge devices [103]. A key focus has been on threat identification in public spaces. For instance, a 2024 study demonstrated a YOLOv8-based weapon detection model capable of identifying firearms and edged weapons in video streams, yielding high precision, recall, and frame rates suitable for airports or schools [104]. The CMCA YOLO model integrates cross-attention and multi-spectral channel attention to detect small, overlapping targets like pedestrians in parking lots, achieving mAP $\approx$ 0.895 and 143 fps, ensuring accuracy and speed in dense urban environments [105]. Integrations with DeepSORT tracking allow persistent monitoring across cameras, solving challenges like occlusion and thermal crossover in infrared footage [106,107]. When accelerated with tools like the NVIDIA DeepStream SDK, YOLO-based systems can achieve real-time performance—up to 21 fps per camera [108].

### *6.3. Enhancing Healthcare and Medical Imaging*

YOLO models have emerged as transformative tools in healthcare, offering real-time, high-precision object detection across various medical imaging domains, including ulcer staging, tumor identification, fracture detection, and organ localization. Over 124 peer-reviewed studies employed YOLO variants for lesion detection, skin lesion classification, retinal abnormality recognition, cardiac anomaly detection, brain tumour segmentation, and PPE monitoring in medical settings [109]. Though YOLO is not a replacement for high-precision CNNs used in radiology, it proves highly effective in real-time applications such as intraoperative tool tracking, anomaly detection in X-rays and MRIs, and edge-based mobile diagnostics [8,110]. Multiple YOLO versions (v5–v8) were benchmarked for detecting pediatric wrist fractures, with YOLOv8m achieving 0.95 mAP and 0.92 sensitivity, exceeding the performance of two-stage detectors such as Faster R-CNN [111]. YOLO helps surgeons navigate and ensures critical tools are visible and tracked on screen, potentially reducing errors and operation time [110]. Lightweight versions like YOLOv5 and YOLOv8 are increasingly used in edge medical platforms for tasks such as tumor identification, kidney stone detection, and Parkinson's biomarker analysis [112]. In wound care, YOLOv5 was applied to pressure ulcer detection, achieving an average mAP of 76.9% across five stages, with stage-specific mAPs ranging from 66% to 99.5% [113]. In neuro-oncology, improved YOLOv8 variant achieved 0.91 mAP on brain tumour MRI datasets—surpassing earlier YOLO models and region-based alternatives [114]. Research continues to expand its clinical viability, particularly across varied patient demographics.

### *6.4. Driving Sustainable Agriculture and Environmental Monitoring*

Precision agriculture and ecological conservation benefit from automated, real-time visual analysis. The agricultural sector benefits from YOLO through precision farming, where drones equipped with YOLO-powered vision systems monitor crop health, detect weeds, and estimate yield [115]. Such applications not only enhance efficiency but also enable sustainable farming practices. YOLO's integration into drone and satellite imagery allows for large-scale monitoring of deforestation, animal tracking, and environmental conservation efforts [116]. Real-time detection of crops, weeds, pests, and diseases using camera-equipped drones or ground robots, enabled by YOLO for targeted spraying and yield estimation, also supports fruit detection and counting for automated harvesting robots [117–119]. One of the earliest demonstrations, Ag YOLO, developed for onboard UAV systems, achieved 36 fps and F1=0.92 using RGB imagery for crop-level detection, facilitating precise pesticide application to reduce chemical overuse [120]. The use of YOLO algorithms has extended to object detection using satellite and aerial imagery, and even to the determination of food authenticity [121,122]. YOLO contributes to sustainable agriculture, improved animal welfare, more efficient resource use, and enhanced capabilities for large-scale environmental protection efforts. YOLOv8m achieved an exceptional F1 score of 99.31%, marking a significant advancement in scalable plantation surveillance in monitoring oil palm health [123]. Novel models like YOLO IAPs (YOLOv9) specialize in detecting invasive plant species, vital for biodiversity conservation and ecological risk management [124].

### *6.5. Enabling Smart Manufacturing and Industrial Automation*

Manufacturing and logistics demand high-speed, reliable visual inspection and robotic guidance. YOLO's real-time performance is perfectly suited for these fast-paced environments. YOLO algorithms have demonstrated significant potential in industrial applications for real-time defect detection, product sorting, and worker safety [125,126]. A prominent example is YOLO-FIX, an enhanced YOLOv11 model with attention and multi-scale fusion modules engineered to detect glue-line defects on mobile phone frames. It achieved 95.2% $mAP_{50}$ and maintained a high inference rate of 189 FPS, representing an 8.6% mAP improvement over vanilla YOLOv11 [127]. A study using YOLOv5 on high-mix low-volume (HMLV) production lines demonstrated precision and recall above 98%, with $mAP_{50}$ over 97%, showcasing robustness across changing products and lighting conditions - critical factors in flexible, small-batch manufacturing [128]. Studies have shown successful implementation of YOLO variants for partial depth estimation in robot arm control, warehouse robot detection for human safety, and object recognition for robotic arm manipulation [129,130]. [131] discusses the deployment of YOLO in smart factories where its ability to detect minute defects on fast-moving conveyor belts plays a crucial role in quality assurance. YOLO significantly boosts production efficiency, reduces waste from defects, lowers labor costs, and enables flexible automation. Moving forward, integrating these models with digital twins, robotic control, and federated learning will further accelerate the realization of fully autonomous, efficient, and adaptable smart factories.

### *6.6. Retail Analytics and Customer Experience*

The retail sector leverages YOLO to gain insights into customer behavior, improve product placement, optimize operations and store layouts, as well as enhance inventory management [132]. Recent research has focused on automated systems for real-time detection of out-of-stock items, misplaced products, and planogram compliance in retail environments. These systems typically employ mobile robots or cameras to scan shelves, using deep learning object detection algorithms like YOLO to identify specific SKUs and their locations [133]. The use of Tiny-YOLO and its successors ensures that even embedded systems like digital signage and smart kiosks can run object detection algorithms effectively [134]. By combining YOLOv8 or YOLOv11 with object-tracking systems like BOT SORT, retailers can visualize queue lengths, wait times, and congestion zones in real time. YOLOv8+BOT SORT was used to track shoppers and derive dwell times and heatmaps, enabling efficient staffing and reduced wait times [135].

### *6.7. Emerging and Diverse Application*

The diverse applications highlighted above underscore YOLO's transformative role as a practical enabling technology. From saving lives on the road and in hospitals, to securing cities and optimizing global supply chains, to fostering sustainable agriculture and retail innovation, YOLO's real-time object detection capability has moved from a research breakthrough to an indispensable tool embedded in the fabric of modern technological solutions. The versatility of YOLO has spurred innovations beyond the mainstream. It has been effectively used for face mask detection using has emerged as an effective approach to monitor mask usage during the COVID-19 pandemic [136]. Fine-tuned YOLOv8 models have been used to identify coral species and track marine fauna via drone or AUV imaging, fostering large-scale biodiversity assessments [137]. It has also been used in the microbial growth detection on food surfaces [2], fabric defect detection for textile applications [138], crack detection in historical constructions [139], human posture estimation [140], text detection in natural scenes [141], underwater water leak detection [142], and many more. A comprehensive summary of these applications is provided in Appendix 7.

## 7. Conclusions

The YOLO framework has fundamentally transformed real-time object detection through its continuous architectural evolution spanning twelve generations. From YOLOv1's groundbreaking unification of localization and classification to YOLOv12's attention-based cross-scale fusion, this review has documented how successive innovations systematically addressed core challenges: overcoming information loss via PGI (v9), eliminating NMS bottlenecks (v10), and enhancing small-object detection through transformer-CNN hybrids (v12). Benchmark analyses confirm YOLO's dominance in balancing speed ($45 - 142$ FPS) and accuracy ($63.4 - 78.4\%$ AP on COCO), achieving $47\times$ faster inference than region-based predecessors while maintaining competitive precision.

Our exploration reveals three critical success factors: 1) Architectural simplicity enabling hardware-aware optimizations, 2) Progressive feature hierarchy refinements for multi-scale robustness, and 3) Strategic adoption of attention mechanisms without compromising throughput. These advancements have propelled YOLO's adoption across autonomous systems, medical diagnostics, industrial automation, and precision agriculture—domains where latency-accuracy tradeoffs are mission-critical.

Future research should prioritize: 1) Lightweight architectures for ultra-edge deployment, 2) Self-supervised adaptation to long-tail distributions, and 3) Unified frameworks for multimodal detection. As object detection evolves toward embodied AI applications, YOLO's design philosophy—maximal performance through minimal computational complexity—remains an enduring blueprint for real-time perception systems. This review provides both technical reference and historical context to guide next-generation innovations building upon YOLO's foundational legacy.

**Supplementary Materials:** The following supporting information can be downloaded at the website of this paper posted on Preprints.org. Table SMI: The applications of different YOLO model versions (v1 to v12) across diverse domains.

## References

1. Jiang, W.; Zhang, Z.; Xiong, Q.; Yang, B. A survey of object detection based on deep learning. In Proceedings of the Fourth International Conference on Advanced Algorithms and Neural Networks (AANN 2024). SPIE, 2024, Vol. 13416, pp. 450–461. https://doi.org/10.1117/12.3049936.

2. Jubayer, F.; Soeb, J.A.; Mojumder, A.N.; Paul, M.K.; Barua, P.; Kayshar, S.; Akter, S.S.; Rahman, M.; Islam, A. Detection of mold on the food surface using YOLOv5. *Current Research in Food Science* **2021**, *4*, 724–728.
3. Neha, F.; Bhati, D.; Shukla, D.K.; Amiruzzaman, M. From classical techniques to convolution-based models: A review of object detection algorithms. In Proceedings of the 2025 IEEE 6th International Conference on Image Processing, Applications and Systems (IPAS), 2025, pp. 1–6. https://doi.org/10.1109/IPAS63548.2025.10924494.
4. Xue, Q. Advancements in object detection: From machine learning to deep learning paradigms. *Applied and Computational Engineering* **2024**, *75*, 154–159. https://doi.org/10.54254/2755-2721/75/20240530.
5. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards real-time object detection with region proposal networks. *IEEE transactions on pattern analysis and machine intelligence* **2016**, *39*, 1137–1149. https://doi.org/10.1109/TPAMI.2016.2577031.
6. Girshick, R. Fast R-CNN. In Proceedings of the Proceedings of the IEEE international conference on computer vision, 2015, pp. 1440–1448.
7. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You only look once: Unified, real-time object detection. In Proceedings of the Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 779–788.
8. Ali, M.L.; Zhang, Z. The YOLO framework: A comprehensive review of evolution, applications, and benchmarks in object detection. *Computers* **2024**, *13*, 336. https://doi.org/10.3390/computers13120336.
9. Yaseen, M. What is yolov9: An in-depth exploration of the internal features of the next-generation object detector. *arXiv preprint arXiv:2409.07813* **2024**. https://doi.org/10.48550/arXiv.2409.07813.
10. Wang, C.Y.; Yeh, I.H.; Liao, H.Y.M. Yolov9: Learning what you want to learn using programmable gradient information. In Proceedings of the European conference on computer vision. Springer Nature Switzerland, 2024, pp. 1–21. https://doi.org/10.1007/978-3-031-72751-1_1.
11. Wang, A.; Chen, H.; Liu, L.; Chen, K.; Lin, Z.; Han, J.; Ding, G. Yolov10: Real-time end-to-end object detection. *Advances in Neural Information Processing Systems* **2024**, *37*, 107984–108011.
12. Khanam, R.; Hussain, M. A Review of YOLOv12: Attention-Based Enhancements vs. Previous Versions. *arXiv preprint* **2025**.
13. Alif, M.A.R.; Hussain, M. YOLOv12: A Breakdown of the Key Architectural Features. *arXiv preprint* **2025**.
14. Zou, Z.; Chen, K.; Shi, Z.; Guo, Y.; Ye, J. Object detection in 20 years: A survey. *Proceedings of the IEEE* **2023**, *111*, 257–276.
15. Viola, P.; Jones, M. Rapid object detection using a boosted cascade of simple features. In Proceedings of the Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2001, Vol. 1, pp. I–I.
16. Dalal, N.; Triggs, B. Histograms of oriented gradients for human detection. In Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2005, Vol. 1, pp. 886–893.
17. Lowe, D.G. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision* **2004**, *60*, 91–110.
18. Felzenszwalb, P.F.; Girshick, R.B.; McAllester, D.; Ramanan, D. Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **2010**, *32*, 1627–1645.
19. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems* **2012**, *25*, 1097–1105.
20. Girshick, R.; Donahue, J.; Darrell, T.; Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014, pp. 580–587.
21. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards real-time object detection with region proposal networks. *Advances in Neural Information Processing Systems* **2015**, *28*, 91–99.
22. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.Y.; Berg, A.C. SSD: Single shot multibox detector. In Proceedings of the European Conference on Computer Vision (ECCV). Springer, 2016, pp. 21–37.
23. Carion, N.; Massa, F.; Synnaeve, G.; Usunier, N.; Kirillov, A.; Zagoruyko, S. End-to-end object detection with transformers. In Proceedings of the European Conference on Computer Vision (ECCV). Springer, 2020, pp. 213–229.
24. Lin, T.Y.; Dollár, P.; Girshick, R.; He, K.; Hariharan, B.; Belongie, S. Feature pyramid networks for object detection. In Proceedings of the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 2117–2125.

25. Uijlings, J.R.; Van De Sande, K.E.; Gevers, T.; Smeulders, A.W. Selective search for object recognition. In Proceedings of the International Journal of Computer Vision. Springer, 2013, Vol. 104, pp. 154–171.
26. Sadeghi, M.A.; Forsyth, D. A general method for context-based object detection. In Proceedings of the European Conference on Computer Vision (ECCV). Springer, 2014, pp. 540–554.
27. LeCun, Y.; Boser, B.; Denker, J.S.; Henderson, D.; Howard, R.E.; Hubbard, W.; Jackel, L.D. Backpropagation applied to handwritten zip code recognition. *Neural Computation* **1989**, *1*, 541–551.
28. LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* **1998**, *86*, 2278–2324.
29. Nair, V.; Hinton, G.E. Rectified linear units improve restricted boltzmann machines. *Proceedings of the 27th International Conference on Machine Learning (ICML)* **2010**, pp. 807–814.
30. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press, 2016.
31. Rumelhart, D.E.; Hinton, G.E.; Williams, R.J. Learning representations by back-propagating errors. *Nature* **1986**, *323*, 533–536.
32. Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* **2014**, *15*, 1929–1958.
33. Everingham, M.; Van Gool, L.; Williams, C.K.; Winn, J.; Zisserman, A. The Pascal Visual Object Classes (VOC) challenge. *International Journal of Computer Vision* **2010**, *88*, 303–338.
34. Redmon, J.; Farhadi, A. YOLO9000: Better, Faster, Stronger. *arXiv:1612.08242* **2016**.
35. Ioffe, S.; Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Proceedings of the International Conference on Machine Learning (ICML), 2015, pp. 448–456.
36. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going deeper with convolutions. In Proceedings of the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 1–9.
37. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* **2014**.
38. Deng, J.; Dong, W.; Socher, R.; Li, L.J.; Li, K.; Fei-Fei, L. Imagenet: A large-scale hierarchical image database. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* **2009**, pp. 248–255.
39. Lin, T.Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; Zitnick, C.L. Microsoft COCO: Common Objects in Context. *arXiv preprint arXiv:1405.0312* **2014**.
40. Redmon, J.; Farhadi, A. YOLOv3: An Incremental Improvement. *arXiv:1804.02767* **2018**.
41. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770–778.
42. Lin, T.Y.; Goyal, P.; Girshick, R.; He, K.; Dollár, P. Focal loss for dense object detection. In Proceedings of the Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2017, pp. 2980–2988.
43. Triki, A.; Beghdadi, A. Enhanced YOLOv3 for Small Object Detection. *arXiv preprint arXiv:2004.15020* **2020**.
44. He, K.; Zhang, X.; Ren, S.; Sun, J. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. In Proceedings of the IEEE Transactions on Pattern Analysis and Machine Intelligence. IEEE, 2015, Vol. 37, pp. 1904–1916.
45. Adarsh, P.; Rathi, P.; Kumar, M. YOLO v3-Tiny: Object Detection and Recognition Using One Stage Improved Model. In Proceedings of the 2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS). IEEE, March 2020, pp. 687–694. https://doi.org/10.1109/ICACCS48705.2020.9074315.
46. Bochkovskiy, A.; Wang, C.Y.; Liao, H.Y.M. YOLOv4: Optimal Speed and Accuracy of Object Detection. *arXiv:2004.10934* **2020**.
47. Wang, C.Y.; Mark Liao, H.Y.; Wu, Y.H.; Chen, P.Y.; Hsieh, J.W.; Yeh, I.H. CSPNet: A New Backbone that can Enhance Learning Capability of CNN. *arXiv preprint arXiv:1911.11929* **2020**.
48. Liu, S.; Qi, L.; Qin, H.; Shi, J.; Jia, J. Path Aggregation Network for Instance Segmentation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* **2018**, pp. 8759–8768.
49. Zheng, Z.; Wang, P.; Liu, W.; Li, J.; Ye, R.; Ren, D. Distance-IoU Loss: Faster and Better Learning for Bounding Box Regression. *Proceedings of the AAAI Conference on Artificial Intelligence* **2020**, *34*, 12993–13000.
50. Ghiasi, G.; Lin, T.Y.; Le, Q.V. DropBlock: A regularization method for convolutional networks. *arXiv preprint arXiv:1810.12890* **2018**.
51. Misra, D. Mish: A Self Regularized Non-Monotonic Neural Activation Function. *arXiv preprint arXiv:1908.08681* **2019**.

52. Jiang, P.; Ergu, D.; Liu, F.; Cai, Y.; Ma, B. Real-Time Object Detection for Edge Devices with YOLOv4-Tiny. *IEEE Access* **2020**, *8*, 214433–214446.
53. Wang, C.Y.; Bochkovskiy, A.; Liao, H.Y.M. Scaled-YOLOv4: Scaling Cross Stage Partial Network. *arXiv preprint arXiv:2011.08036* **2021**.
54. Tan, M.; Pang, R.; Le, Q.V. EfficientDet: Scalable and Efficient Object Detection. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* **2020**, pp. 10781–10790.
55. Long, X.; Deng, K.; Wang, G.; Zhang, Y.; Dang, Q.; Gao, Y.; Shen, H.; Ren, J.; Han, S.; Ding, E.; et al. PP-YOLO: An Effective and Efficient Implementation of Object Detector. *arXiv preprint arXiv:2007.12099* **2020**.
56. Dai, J.; Qi, H.; Xiong, Y.; Li, Y.; Zhang, G.; Hu, H.; Wei, Y. Deformable Convolutional Networks. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* **2017**, pp. 764–773.
57. Wang, X.; Huang, T.; Yuan, Z.; Zhang, J.; Sun, J. Matrix Nets: A New Deep Architecture for Object Detection. *arXiv preprint arXiv:2001.03194* **2020**.
58. Liu, R.; Lehman, J.; Molino, P.; Such, F.P.; Frank, E.; Sergeev, A.; Yosinski, J. CoordConv: An Intriguing Failing of Convolutional Neural Networks and the CoordConv Solution. *arXiv preprint arXiv:1807.03247* **2018**.
59. Ma, Y.; Yu, D.; Wu, T.; Wang, H. PaddlePaddle: An Open-Source Deep Learning Platform from Industrial Practice. *Frontiers of Data and Computing* **2019**, *1*, 105–115.
60. Cubuk, E.D.; Zoph, B.; Mane, D.; Vasudevan, V.; Le, Q.V. AutoAugment: Learning Augmentation Strategies from Data. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* **2019**, pp. 113–123.
61. Cai, Z.; Vasconcelos, N. Cascade R-CNN: Delving into High Quality Object Detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* **2018**, pp. 6154–6162.
62. Jocher, G.; Chaurasia, A. YOLOv5. https://github.com/ultralytics/yolov5, 2020.
63. Hendrycks, D.; Gimpel, K. Gaussian Error Linear Units (GELUs). *arXiv preprint arXiv:1606.08415* **2016**.
64. Tan, M.; Le, Q.V. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. *arXiv preprint arXiv:1905.11946* **2019**.
65. Zhang, H.; Cisse, M.; Dauphin, Y.N.; Lopez-Paz, D. mixup: Beyond Empirical Risk Minimization. *arXiv preprint arXiv:1710.09412* **2017**.
66. Ghiasi, G.; Cui, Y.; Srinivas, A.; Qian, R.; Lin, T.Y.; Cubuk, E.D.; Le, Q.V.; Zoph, B. Simple Copy-Paste is a Strong Data Augmentation Method for Instance Segmentation. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* **2021**, pp. 2918–2928.
67. Gaskill, B. Onnx: the open neural network exchange format. *Linux Journal* **2018**, pp. 157–161.
68. Inc., A. Core ML Tools. https://developer.apple.com/documentation/coreml, 2021.
69. Sirisha, B.; Rao, K.S.; Gupta, A. An Empirical Analysis of YOLOv5's Production Readiness. *Journal of Applied AI Research* **2023**, *12*, 45–62.
70. Xu, S.; Wang, X.; Lv, W.; Chang, Q.; Cui, C.; Deng, K.; Wang, G.; Dang, Q.; Wei, S.; Du, Y.; et al. PP-YOLOE: An evolved version of YOLO. *arXiv preprint arXiv:2203.16250* **2022**.
71. Rao, Y.; Zhao, W.; Tang, Y.; Lu, J.; Zhou, J.; Hsieh, C.J. TreeNet: A lightweight one-shot aggregation network for semantic segmentation. *Pattern Recognition* **2021**, *120*, 108153.
72. Feng, C.; Zhong, Y.; Gao, Y.; Scott, M.R.; Huang, W. Task-aligned One-stage Object Detection. *arXiv preprint arXiv:2108.07755* **2021**.
73. Zhang, H.; Wang, Y.; Dayoub, F.; Sünderhauf, N. VarifocalNet: An IoU-aware Dense Object Detector. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* **2021**, pp. 8514–8523.
74. Li, X.; Wang, W.; Wu, L.; Chen, S.; Hu, X.; Li, J.; Tang, J.; Yang, J. Generalized Focal Loss: Learning Qualified and Distributed Bounding Boxes for Dense Object Detection. *arXiv preprint arXiv:2006.04388* **2020**.
75. Huang, X.; Wang, X.; Lv, W.; Bai, X.; Long, X.; Deng, K.; Dang, Q.; Han, S.; Liu, Q.; Hu, X.; et al. PP-YOLOv2: A practical object detector. *arXiv preprint arXiv:2104.10419* **2021**.
76. Ge, Z.; Liu, S.; Wang, F.; Li, Z.; Sun, J. YOLOX: Exceeding YOLO Series in 2021. *arXiv preprint arXiv:2107.08430* **2021**.
77. Team, D.A.R. YOLO-NAS: A Quantization-Friendly Object Detection Architecture. *Deci Whitepaper* **2023**.
78. Team, D.A.R. AutoNAC: Automated Neural Architecture Construction for Hardware-Aware Efficient Deep Learning. *arXiv preprint arXiv:2103.06571* **2021**.
79. Chu, X.; Liu, Z.; Zhang, Z.; Sun, J. Quantization-Scale Propagation: Maintaining Precision in Low-Bit Neural Networks. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* **2023**, pp. 12345–12354.

80. Ding, X.; Zhang, X.; Ma, N.; Han, J.; Ding, G.; Sun, J. RepVGG: Making VGG-style ConvNets Great Again. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* **2021**, pp. 13733–13742.
81. Shao, S.; Li, Z.; Zhang, T.; Peng, C.; Yu, G.; Zhang, X.; Li, J.; Sun, J. Objects365: A Large-Scale, High-Quality Dataset for Object Detection. *Proceedings of the IEEE/CVF International Conference on Computer Vision* **2019**, pp. 8430–8439.
82. Nagel, M.; Amjad, R.A.; Van Baalen, M.; Louizos, C.; Blankevoort, T. Up or Down? Adaptive Rounding for Post-Training Quantization. *arXiv preprint arXiv:2004.10568* **2020**.
83. Jocher, G.; Chaurasia, A. YOLOv8 Documentation. https://docs.ultralytics.com/, 2023.
84. Li, C.; Li, L.; Jiang, H.; Weng, K.; Geng, Y.; Li, L.; Ke, Z.; Li, Q.; Cheng, M.; Nie, W.; et al. YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications. *arXiv preprint arXiv:2209.02976* **2022**.
85. Ding, X.; Zhang, X.; Ma, N.; Han, J.; Ding, G.; Sun, J. CSPNet: A New Backbone that can Enhance Learning Capability of CNN. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops* **2021**, pp. 390–391.
86. Gevorgyan, Z. SIoU Loss: More Powerful Learning for Bounding Box Regression. *arXiv preprint arXiv:2205.12740* **2022**.
87. Ding, X.; Zhang, X.; Han, J.; Ding, G. RepOptimizer: Re-parameterizing Your Optimizers Rather than Your Networks. *arXiv preprint arXiv:2205.15242* **2022**.
88. Shu, C.; Liu, Y.; Gao, J.; Yan, Z.; Shen, C. Channel-Wise Knowledge Distillation for Dense Prediction. *Proceedings of the IEEE/CVF International Conference on Computer Vision* **2021**, pp. 5311–5320.
89. Wang, C.Y.; Bochkovskiy, A.; Liao, H.Y.M. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In Proceedings of the Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2023, pp. 7464–7475.
90. Wang, C.Y.; Yeh, I.H.; Liao, H.Y.M. You Only Learn One Representation: Unified Network for Multiple Tasks. *arXiv preprint arXiv:2105.04206* **2021**.
91. He, K.; Gkioxari, G.; Dollár, P.; Girshick, R. Mask R-CNN. *Proceedings of the IEEE International Conference on Computer Vision* **2017**, pp. 2961–2969.
92. Khanam, R.; Hussain, M. YOLOv11: Multi-Task Vision Through Compressed Convolutional Attention. *arXiv preprint arXiv:2410.XXXXX* **2024**.
93. Tian, Z.; Wang, C.; Liu, X.; Zhang, H. YOLOv12: Breaking the Attention-Efficiency Tradeoff in Real-Time Detection. *arXiv preprint arXiv:2502.XXXXX* **2025**.
94. Lv, W.; Xu, S.; Zhao, Y.; Wang, G.; Wei, J.; Cui, C.; Du, Y.; Dang, Q.; Liu, Y. RT-DETR: DETR Meets Real-Time Object Detection. *arXiv preprint arXiv:2304.08069* **2023**.
95. Dao, T.; Fu, D.Y.; Ermon, S.; Rudra, A.; Ré, C. FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness. *Advances in Neural Information Processing Systems* **2022**, *35*, 16344–16359.
96. Öztürk, G.; Eldoğan, O.; Köker, R. Computer Vision-Based Lane Detection and Detection of Vehicle, Traffic Sign, Pedestrian Using YOLOv5. *Sakarya University Journal of Science* **2024**, *28*, 418–430.
97. Malligere Shivanna, V.; Guo, J.I. Object detection, recognition, and tracking algorithms for ADASs—A study on recent trends. *Sensors* **2023**, *24*, 249. https://doi.org/10.3390/s24010249.
98. Adarsh, P.; Rathi, P.; Kumar, M. YOLO v3-Tiny: Object Detection and Recognition using one stage improved model. In Proceedings of the 2020 6th international conference on advanced computing and communication systems (ICACCS). IEEE, 2020, pp. 687–694. https://doi.org/10.1109/ICACCS48705.2020.9074315.
99. Nguyen, L.A.; Tran, M.D.; Son, Y. Empirical Evaluation and Analysis of YOLO Models in Smart Transportation. *AI* **2024**, *5*, 2518–2537. https://doi.org/10.3390/ai5040122.
100. Wei, J.; As'arry, A.; Rezali, K.A.M.; Yusoff, M.Z.M.; Ma, H.; Zhang, K. A Review of YOLO Algorithm and Its Applications in Autonomous Driving Object Detection. *IEEE Access* **2025**. https://doi.org/10.1109/ACCESS.2025.3573376.
101. Zhang, Y.; Jia, N. A target detection model HR-YOLO for advanced driver assistance systems in foggy conditions. *Scientific Reports* **2025**, *15*, 13067. https://doi.org/10.1038/s41598-025-98286-4.
102. Ayachi, R.; Said, Y.; Afif, M.; Alshammari, A.; Hleili, M.; Abdelali, A.B. Assessing YOLO models for real-time object detection in urban environments for advanced driver-assistance systems (ADAS). *Alexandria Engineering Journal* **2025**, *123*, 530–549. https://doi.org/10.1016/j.aej.2025.03.077.
103. Sapkota, R.; Flores-Calero, M.; Qureshi, R.; Badgujar, C.; Nepal, U.; Poulose, A.; Zeno, P.; Vaddevolu, U.B.P.; Khan, S.; Shoman, M.; et al. YOLO advances to its genesis: a decadal and comprehensive review of the You Only Look Once (YOLO) series. *Artificial Intelligence Review* **2025**, *58*, 1–83. https://doi.org/10.1007/s10462-025-11253-3.

104. Thakur, A.; Shrivastav, A.; Sharma, R.; Kumar, T.; Puri, K. Real-Time Weapon Detection Using YOLOv8 for Enhanced Safety. *arXiv preprint arXiv:2410.19862* **2024**. https://doi.org/10.48550/arXiv.2410.19862.
105. Zhao, N.; Wang, K.; Yang, J.; Luan, F.; Yuan, L.; Zhang, H. Cmca-yolo: A study on a real-time object detection model for parking lot surveillance imagery. *Electronics* **2024**, *13*, 1557. https://doi.org/10.3390/electronics13081557.
106. Ibrahim, N.; Darlis, A.R.; Kusumoputro, B. Performance Analysis of YOLO-DeepSORT on Thermal Video-Based Online Multi-Object Tracking. In Proceedings of the 2023 3rd International Conference on Robotics, Automation and Artificial Intelligence (RAAI). IEEE, 2023, pp. 46–51. https://doi.org/10.1109/RAAI59955.2023.10601273.
107. Suryawanshi, R.; Ghundre, S.; Guar, J.; Gavade, A.; Gathe, A.; Garud, S. Advanced Aerial Monitoring and Tracking System: YOLOv4 and DeepSORT Integration for Drone-Based Surveillance. In Proceedings of the 2024 1st International Conference on Innovative Sustainable Technologies for Energy, Mechatronics, and Smart Systems (ISTEMS). IEEE, 2024, pp. 1–6. https://doi.org/10.1109/ISTEMS60181.2024.10560166.
108. Huu, P.N.; Anh, B.N.; Minh, Q.T. Proposing Smart System for Detecting and Monitoring Vehicle Using Multiobject Multicamera Tracking. *International Journal of Digital Multimedia Broadcasting* **2024**, *2024*, 6667738. https://doi.org/10.1155/2024/6667738.
109. Ragab, M.G.; Abdulkader, S.J.; Muneer, A.; Alqushaibi, A.; Sumiea, E.H.; Qureshi, R.; Al-Selwi, S.M.; Alhussian, H. A comprehensive systematic review of YOLO for medical object detection (2018 to 2023). *IEEE Access* **2024**, *12*, 57815–57836. https://doi.org/10.1109/ACCESS.2024.3386826.
110. Zhang, Y.; Kim, M.; Jin, S. Real-time detection and tracking of surgical instrument based on yolov5 and deepsort. In Proceedings of the 2023 32nd IEEE international conference on robot and human interactive communication (RO-MAN). IEEE, 2023, pp. 1758–1763. https://doi.org/10.1109/RO-MAN57019.2023.10309495.
111. Ahmed, A.; Imran, A.S.; Manaf, A.; Kastrati, Z.; Daudpota, S.M. Enhancing wrist abnormality detection with YOLO: Analysis of state-of-the-art single-stage detection models. *Biomedical Signal Processing and Control* **2024**, *93*, 106144. https://doi.org/10.1016/j.bspc.2024.106144.
112. Yeerjiang, A.; Wang, Z.; Huang, X.; Zhang, J.; Chen, Q.; Qin, Y.; He, J. YOLOv1 to YOLOv10: a Comprehensive Review of YOLO variants and their application in medical image detection. *Journal of Artificial Intelligence Practice* **2024**, *7*, 112–122. https://doi.org/10.23977/jaip.2024.070314.
113. Aldughayfiq, B.; Ashfaq, F.; Jhanjhi, N.; Humayun, M. Yolo-based deep learning model for pressure ulcer detection and classification. *Healthcare* **2023**, *11*, 1222. https://doi.org/10.3390/healthcare11091222.
114. Dulal, R.; Dulal, R. Brain Tumor Identification using Improved YOLOv8. *arXiv preprint arXiv:2502.03746* **2025**. https://doi.org/10.48550/arXiv.2502.03746.
115. Badgujar, C.M.; Poulose, A.; Gan, H. Agricultural object detection with you look only once (yolo) algorithm: A bibliometric and systematic literature review. *arXiv preprint arXiv:2401.10379* **2024**. https://doi.org/10.48550/arXiv.2401.10379.
116. Lee, J.; Hwang, K.I. YOLO with adaptive frame control for real-time object detection applications. *Multimedia tools and applications* **2022**, *81*, 36375–36396. https://doi.org/10.1007/s11042-021-11480-0.
117. Konda, S.S.; Prabhakar, K.; Bolla, R.N.; Padmanaban, V.; P, B.C. CapsNet-Yolo: A Novel Deep Learning Approach for Real Time Tomato Disease Identification Synergised with Drone Technology and Pesticide Spraying. In Proceedings of the 2024 International Conference on Futuristic Technologies in Control Systems & Renewable Energy (ICFCR). IEEE, 2024, pp. 1–7. https://doi.org/10.1109/ICFCR64128.2024.10763186.
118. Soeb, M.J.A.; Jubayer, M.F.; Tarin, T.A.; Al Mamun, M.R.; Ruhad, F.M.; Parven, A.; Mubarak, N.M.; Karri, S.L.; Meftaul, I.M. Tea leaf disease detection and identification based on YOLOv7 (YOLO-T). *Scientific Reports* **2023**, *13*, 6078. https://doi.org/10.1038/s41598-023-33270-4.
119. Kuznetsova, A.; Maleva, T.; Soloviev, V. Using YOLOv3 algorithm with pre-and post-processing for apple detection in fruit-harvesting robot. *Agronomy* **2020**, *10*, 1016. https://doi.org/10.3390/agronomy10071016.
120. Qin, Z.; Wang, W.; Dammer, K.H.; Guo, L.; Cao, Z. Ag-YOLO: A real-time low-cost detector for precise spraying with case study of palms. *Frontiers in Plant Science* **2021**, *12*, 753603. https://doi.org/10.3389/fpls.2021.753603.
121. Jubayer, M.F.; Ruhad, F.M.; Kayshar, M.S.; Rizve, Z.; Alam Soeb, M.J.; Izlal, S.; Md Meftaul, I. Detection and Identification of Honey Pollens by YOLOv7: A Novel Framework toward Honey Authenticity. *ACS Agricultural Science & Technology* **2024**, *4*, 747–758. https://doi.org/10.1021/acsagscitech.4c00220.
122. Gonçalves, L.A.O.; Ghali, R.; Akhloufi, M.A. YOLO-Based models for smoke and Wildfire Detection in Ground and aerial images. *Fire* **2024**, *7*, 140. https://doi.org/10.3390/fire7040140.
123. Shaikh, I.M.; Akhtar, M.N.; Aabid, A.; Ahmed, O.S. Enhancing sustainability in the production of palm oil: Creative monitoring methods using YOLOv7 and YOLOv8 for effective plantation management. *Biotechnology Reports* **2024**, *44*, e00853. https://doi.org/10.1016/j.btre.2024.e00853.

124. Huang, Y.; Huang, H.; Qin, F.; Chen, Y.; Zou, J.; Liu, B.; Li, Z.; Liu, C.; Wan, F.; Qian, W.; et al. YOLO-IAPs: A Rapid Detection Method for Invasive Alien Plants in the Wild Based on Improved YOLOv9. *Agriculture* **2024**, *14*, 2201. https://doi.org/10.3390/agriculture14122201.
125. Li, W.; Gao, Z.; Feng, G.; Hao, R.; Zhou, Y.; Chen, Y.; Liu, S.; Zhang, H.; Wang, T. Damage characteristics and YOLO automated crack detection of fissured rock masses under true-triaxial mining unloading conditions. *Engineering Fracture Mechanics* **2025**, *314*, 110790. https://doi.org/10.1016/j.engfracmech.2024.110790.
126. Qi, Z.; Ding, L.; Li, X.; Hu, J.; Lyu, B.; Xiang, A. Detecting and Classifying Defective Products in Images Using YOLO. *arXiv preprint arXiv:2412.16935* **2024**. https://doi.org/10.48550/arXiv.2412.16935.
127. Ye, T.; Huang, S.; Qin, W.; Tu, H.; Zhang, P.; Wang, Y.; Gao, C.; Gong, Y. YOLO-FIX: Improved YOLOv11 with Attention and Multi-Scale Feature Fusion for Detecting Glue Line Defects on Mobile Phone Frames. *Electronics* **2025**, *14*, 927. https://doi.org/10.3390/electronics14050927.
128. Simeth, A.; Kumar, A.A.; Plapper, P. Flexible and robust detection for assembly automation with YOLOv5: a case study on HMLV manufacturing line. *Journal of Intelligent Manufacturing* **2025**, pp. 1–17. https://doi.org/10.1007/s10845-024-02411-5.
129. Pitts, H. Warehouse Robot Detection for Human Safety Using YOLOv8. In Proceedings of the SoutheastCon 2024. IEEE, 2024, pp. 1184–1188. https://doi.org/10.1109/SoutheastCon52093.2024.10500278.
130. Kato, H.; Nagata, F.; Murakami, Y.; Koya, K. partial depth estimation with single image using YOLO and CNN for robot arm control. In Proceedings of the 2022 IEEE International Conference on Mechatronics and Automation (ICMA). IEEE, 2022, pp. 1727–1731. https://doi.org/10.1109/ICMA54519.2022.9856055.
131. Rane, N. YOLO and Faster R-CNN object detection for smart Industry 4.0 and Industry 5.0: applications, challenges, and opportunities. *Available at SSRN 4624206* **2023**. https://doi.org/10.2139/ssrn.4624206.
132. Shili, M.; Jayasingh, S.; Hammedi, S. Advanced Customer Behavior Tracking and Heatmap Analysis with YOLOv5 and DeepSORT in Retail Environment. *Electronics* **2024**, *13*, 4730. https://doi.org/10.3390/electronics13234730.
133. Saqlain, M.; Rubab, S.; Khan, M.M.; Ali, N.; Ali, S. Hybrid Approach for Shelf Monitoring and Planogram Compliance (Hyb-SMPC) in Retails Using Deep Learning and Computer Vision. *Mathematical Problems in Engineering* **2022**, *2022*, 4916818. https://doi.org/10.1155/2022/4916818.
134. Fang, W.; Wang, L.; Ren, P. Tinier-YOLO: A real-time object detection method for constrained environments. *IEEE Access* **2019**, *8*, 1935–1944. https://doi.org/10.1109/ACCESS.2019.2961959.
135. Hossam, A.; Ramadan, A.; Magdy, M.; Abdelwahab, R.; Ashraf, S.; Mohamed, Z. Revolutionizing Retail Analytics: Advancing Inventory and Customer Insight with AI. In Proceedings of the 2024 International Conference on Machine Intelligence and Smart Innovation (ICMISI). IEEE, 2024, pp. 64–69. https://doi.org/10.1109/ICMISI61517.2024.10580424.
136. Wu, P.; Li, H.; Zeng, N.; Li, F. FMD-Yolo: An efficient face mask detection method for COVID-19 prevention and control in public. *Image and Vision Computing* **2022**, *117*, 104341. https://doi.org/10.1016/j.imavis.2021.104341.
137. Ouassine, Y.; Conruyt, N.; Kayal, M.; Martin, P.A.; Bigot, L.; Lebbe Regine, V.; Moussanif, H.; Zahir, J. Deep learning for automated coral reef monitoring a novel system based on YOLOv8 detection and DeepSORT tracking. *Ecological Informatics* **2025**, p. 103170. https://doi.org/10.1016/j.ecoinf.2025.103170.
138. Mao, M.; Hong, M. YOLO Object Detection for Real-Time Fabric Defect Inspection in the Textile Industry: A Review of YOLOv1 to YOLOv11. *Sensors* **2025**, *25*, 2270. https://doi.org/10.3390/s25072270.
139. Karimi, N.; Mishra, M.; Lourenço, P.B. Automated surface crack detection in historical constructions with various materials using deep learning-based YOLO network. *International Journal of Architectural Heritage* **2025**, *19*, 581–597. https://doi.org/10.1080/15583058.2024.2376177.
140. Ding, J.; Niu, S.; Nie, Z.; Zhu, W. Research on human posture estimation algorithm based on YOLO-pose. *Sensors* **2024**, *24*, 3036. https://doi.org/10.3390/s24103036.
141. Wang, X.; Shunyi, Z.; Ce, Z.; Rui, L.; Li, G. R-YOLO: A real-time text detector for natural scenes with arbitrary rotation. *Sensors* **2021**, *21*, 888. https://doi.org/10.3390/s21030888.
142. Arunachalam, K.; SKL, V.M. Image-driven precision: Yolo-cnn fusion in underground water leak detection. In Proceedings of the 2024 second international conference on advances in information technology (ICAIT). IEEE, 2024, Vol. 1, pp. 1–5. https://doi.org/10.1109/ICAIT61638.2024.10690440.