applied sciences

MDPI

Review

From YOLOv1 to YOLO26: A Unified Comparative Review of the Evolution, Architecture, and Applications of YOLO-Based Object Detection Frameworks

Yong-Hwan Lee [1] and Wan-Bum Lee [2,*]

[1] Department of Game Contents, Wonkwang University, 460 Iksan-daero, Iksan 54538, Jeonbuk-do, Republic of Korea; hwany1458@empal.com
[2] Department of Computer Software Engineering, Wonkwang University, 460 Iksan-daero, Iksan 54538, Jeonbuk-do, Republic of Korea
* Correspondence: lwbwon@wku.ac.kr; Tel.: +82-10-3657-0607

Abstract

Object detection has evolved from classical feature-based approaches to deep learning-based frameworks, with the YOLO (You Only Look Once) family emerging as one of the most widely adopted paradigms for real-time object detection. Despite rapid architectural evolution from YOLOv1 to YOLO26, the literature remains fragmented and lacks a unified synthesis integrating architectural innovations, quantitative benchmarking, and domain-specific deployment considerations. This review presents a comprehensive longitudinal analysis of the YOLO lineage, spanning the evolution from Darknet-based backbones to hypergraph-enhanced correlation modeling in YOLOv13 and end-to-end NMS-free architectures in YOLO26, as well as the transition from anchor-based prediction to anchor-free, decoupled, and end-to-end detection heads. We synthesize source-reported metrics, including AP, FPS, FLOPs, and parameter counts, across both canonical and domain-specific benchmarks, while explicitly treating the resulting comparison as a cross-source synthesis rather than a hardware-normalized re-benchmark. Particular attention is given to developments reported during 2025–2026. Under their respective source protocols, YOLOv13 reports a 3.0-AP-point improvement over YOLO11-N (38.6 to 41.6 AP), whereas YOLO26 reports up to 43% faster CPU inference than YOLO11. Because both releases are relatively recent, these findings are presented as source-reported results and are accompanied by an explicit maturity caveat informed by emerging independent evaluations. Across seven application domains, we examine how architectural design choices relate to operational constraints, identify persistent challenges in domain generalization, small-object localization, and open-set detection, and propose future research directions centered on contextual representation, deployment-aware training, data-efficient learning, and reproducible, sustainability-aware benchmarking.

Keywords: You Only Look Once (YOLO); object detection; real-time detection; architectural evolution; end-to-end detection; NMS-free detection; hypergraph learning; edge deployment; benchmark synthesis; model selection

Check for updates

Academic Editor: João M. F. Rodrigues

Received: 13 July 2026
Revised: 5 October 2026
Accepted: 6 October 2026
Published: 20 October 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

1.1. Background and Motivation

Object detection is a fundamental computer vision task that involves identifying and localizing objects of interest within images or video frames [1]. In many practical domains,

including autonomous driving, robotics, aerial surveillance, medical imaging, and industrial inspection, real-time inference is a critical requirement: systems must process frames with minimal latency while maintaining sufficient detection accuracy [1–3]. Historically, object detection has evolved from handcrafted feature-based methods (e.g., Viola-Jones, HOG + SVM, and DPM) to deep learning-based detection frameworks. Two-stage detectors such as R-CNN [4], Fast R-CNN [5], and Faster R-CNN [6] first propose candidate object regions and then classify and refine them, achieving high accuracy but suffering from relatively high computational cost and latency.

To address these limitations, one-stage detectors, which perform localization and classification in a single forward pass, emerged as a practical alternative to two-stage pipelines in latency-sensitive applications [7,8]. Among these approaches, the YOLO (You Only Look Once) family has become one of the most influential detection frameworks owing to its favorable balance between inference speed and detection accuracy. The original YOLO (v1), introduced by Redmon et al. [8] in 2016, reformulated object detection as a unified regression problem, enabling real-time detection by directly predicting bounding boxes and class probabilities from full images. Across successive versions, the YOLO family has undergone continuous refinements to its architecture—including the backbone, neck, and detection head—as well as its training strategies, loss functions, anchor design, and multi-scale feature fusion, improving both computational efficiency and detection accuracy.

As object detection tasks diversify (e.g., small-object detection, dense scenes, domain shift), the demands on detectors evolve. YOLO variants have been adapted to operate on embedded/edge devices, to incorporate attention or transformer modules, to support multi-task outputs (e.g., segmentation, pose), and to optimize energy/latency trade-offs. The breadth and diversity of these developments highlight the need for a unified, systematic synthesis of how the YOLO family has evolved, what trade-offs each design entails, and where open challenges remain.

1.2. Research Context and Need for a Unified Review

Although numerous surveys have examined YOLO-based and one-stage object detection frameworks, most cover releases only through YOLOv7 or YOLOv8 and focus on selected architectural features or application domains rather than providing a comprehensive comparison across versions. For example, Terven et al. [9] present a comprehensive review from YOLOv1 through YOLOv8, describing network architectures, key enhancements, and training tricks. Ali and Zhang [2] similarly highlight application domains such as medical imaging, autonomous vehicles, and agriculture in the YOLO context. Jegham et al. [10] recently provided a benchmark and architectural review involving YOLOv9–v12 in addition to earlier versions. More broadly, surveys of object detection evolution (e.g., covering region-based and transformer-based methods [11]) place YOLO among many families but do not deeply contrast YOLO variants version-to-version [1]. The two-decade retrospective by Zou et al. [12] traces detector evolution from Viola-Jones through deep learning frameworks, while Liu et al. [13] surveyed generic deep-learning object detection encompassing detection frameworks, object proposal generation, context modeling, and evaluation metrics. More recently, Trigka and Dritsas [14] provided a comprehensive 2025 survey of machine learning techniques for object detection covering both traditional and contemporary deep learning approaches. A decade-spanning retrospective by Sapkota et al. [11] further underscores the need for an integrated, up-to-date synthesis.

Three important gaps in the existing literature motivate the present review. First, no existing survey provides a comprehensive treatment of the YOLO lineage through the latest 2025 releases: YOLOv13's hypergraph-based correlation modeling [15] and YOLO26's deployment-centric architectural simplification [16] represent qualitatively new

design directions that post-date all published comprehensive reviews, leaving practitioners without integrated guidance on when and why to adopt these latest variants. Second, existing surveys treat architectural evolution, quantitative benchmarking, and application deployment as largely separate concerns; few systematically trace how a specific structural innovation (e.g., anchor-free decoupled heads or NMS-free decoding) propagates into measurable benchmark improvements and, ultimately, into domain-specific deployment advantages. Third, benchmark numbers circulating in the literature are frequently repeated across secondary sources without verification against primary release papers, propagating inaccuracies; a review that enforces primary-source data discipline is needed to provide a reliable quantitative foundation.

To address these gaps, this review seeks to answer three central research questions: (1) How have YOLO's architectural modules—backbone, neck, and head—evolved across fourteen generations, and which transitions represent paradigm shifts rather than incremental refinements? (2) What do primary-source benchmarks reveal about the accuracy-latency-efficiency trade-offs across the lineage, particularly for the divergent 2025 releases? (3) How do architectural and algorithmic design choices map onto the operational constraints of real deployment domains, and what selection guidance follows for practitioners?

Overall, the literature still lacks a unified longitudinal review that spans YOLOv1 through YOLO26 and systematically integrates a consistent taxonomy, quantitative benchmarking, domain-level application mapping, and future research directions. Such a survey would:

- Clarify the evolution of YOLO by explaining how successive versions build on earlier designs.
- Uncover design trade-offs (accuracy vs. speed vs. complexity) across the lineage.
- Provide practitioners and researchers with a structured guide to selecting or extending YOLO variants for specific domains or constraints.
- Highlight underexplored challenges (e.g., generalization, robustness, domain shift, efficient deployment) that future YOLO-like systems could address.

Given that YOLO remains one of the most widely adopted frameworks for real-time object detection, a survey of this depth and breadth can fill a vital need in consolidating fragmented developments and pointing the way forward.

1.3. Objectives and Scope of This Review

The primary objective of this review is to provide a comprehensive generation-by-generation analysis of YOLO-based object detectors from YOLOv1, introduced in 2016, through YOLO26, unveiled in 2025 and formally documented in the 2026 technical report [16]. The review covers fourteen major architectural iterations, including the hypergraph-enhanced YOLOv13 [15] and the deployment-oriented YOLO26 [16], together with influential derivatives. The analysis is structured around three complementary dimensions:

1. Architectural and algorithmic evolution—cataloging changes in backbone networks, neck and head modules, anchor/anchor-free design, multi-scale fusion, activation and normalization strategies, loss functions, attention or transformer integrations, and optimization/training techniques.
2. Quantitative benchmarking and trade-off analysis—collecting performance metrics such as mAP (or variants), inference latency or FPS, parameter count, GFLOPs, memory footprint, and robustness measures across representative datasets (e.g., PASCAL VOC, MS COCO, and domain-specific datasets). We harmonize and re-interpret reported metrics where necessary to support fair cross-source com-

parison, while acknowledging that the resulting synthesis is not a hardware-normalized re-benchmarking.

3. Domain-level application mapping and adaptation strategies—surveying how different YOLO versions have been deployed or adapted across application domains and examining the associated challenges and design choices.

In addition, we identify persistent challenges, emerging limitations, and open research questions across YOLO's evolution, and we propose a future roadmap linking YOLO's lineage to upcoming trends (e.g., hybrid CNN–transformer detectors, model search/AutoML, multimodal detection, edge/energy-aware design).

Our scope includes both canonical YOLO releases (from original authors or recognized community-maintained forks) and significant research derivatives that are widely cited or practically adopted. We exclude minor variants or unpublished tweaks unless they introduce a novel design insight. This review does not attempt to provide exhaustive coverage of all one-stage object detectors, such as SSD and RetinaNet, or of transformer-only detectors; these methods are discussed only where they provide a relevant basis for comparison. Throughout this review we follow the original authors' designations: community/academic releases are written YOLOv2–YOLOv13, whereas Ultralytics releases use their official names YOLO11 and YOLO26. The phrase "YOLOv1 to YOLO26" denotes the lineage spanning these releases and does not imply the existence of intermediate versions YOLOv14–YOLOv25.

1.4. Contributions of This Work

This review offers six principal contributions to the object detection and real-time vision literature.

(1) Systematic architectural classification. We provide a systematic taxonomy of YOLO architectures from YOLOv1 (2016) to YOLO26 (2025), mapping their structural transitions—from Darknet-based backbones to hypergraph-enhanced correlation modeling, from anchor-driven to anchor-free paradigms, and from monolithic to decoupled and NMS-free heads—within a unified taxonomy that enables direct cross-version comparison.

(2) Cross-version quantitative benchmarking and trade-off analysis. We compile and normalize quantitative indicators—including mean Average Precision (mAP), frames per second (FPS), FLOPs, parameter counts, and memory footprint—from primary release papers and comparable surveys, presenting accuracy-latency-efficiency trade-offs across fourteen generations under explicitly documented measurement conventions.

(3) Domain-level application mapping. We conduct a multi-sector analysis of YOLO adoption across autonomous driving, industrial inspection, precision agriculture, biomedical imaging, robotics and drones, and consumer multimedia, synthesizing over one hundred concrete deployment case studies (Section 6) and identifying which architectural choices map to which domain constraints.

(4) Synthesis of challenges and emerging research directions. Through our cross-version and cross-domain synthesis, we identify persistent limitations—such as small-object localization, domain generalization, and open-set detection—and articulate a forward-looking research agenda encompassing hybrid attention architectures, end-to-end training, data-centric learning, and sustainability-aware benchmarking.

(5) Bridging theoretical advances and industrial deployment. We emphasize the practical relevance of YOLO's design philosophy for real-time embedded and safety-critical systems, distilling deployment guidance by hardware constraint, accuracy requirement, and regulatory context into an actionable playbook for practitioners.

(6) Evolutionary design-principle analysis. Beyond cataloging what changed in each generation, we distill why the lineage evolved as it did. Section 5 parses the YOLOv1-

to-YOLO26 trajectory into recurring bottleneck-response chains, identifies five recurrent design strategies, quantifies where redesign effort concentrated across architectural components, and explains the fundamental trade-offs—accuracy, localization precision, computational complexity, latency, memory, robustness, and deployment efficiency—that made those components the persistent loci of innovation.

In summary, this work extends prior reviews [9–11,17,18] by integrating architectural, quantitative, application-level, and design-principle perspectives within a single longitudinal framework spanning the complete YOLO lineage through the 2025 releases.

1.5. Organization of the Paper

The remainder of this paper is organized to provide a progressive overview of YOLO's evolution and research ecosystem. It begins by establishing the methodological foundation and continues toward in-depth analysis and synthesis.

Section 2 defines the methodology, data sources, and classification criteria used to curate and evaluate YOLO-related literature. Section 3 chronologically traces the architectural evolution of YOLO from its inception in 2016 to the latest YOLO26 (2025), highlighting design philosophies, structural shifts, and algorithmic refinements across 14 major iterations. Section 4 focuses on the key innovations underpinning YOLO's success—backbone transitions, neck-and-head optimizations, activation and loss-function evolution, and recent trends such as decoupled heads and attention-based modules. Section 5 steps back from the chronological and thematic accounts to distill the evolutionary design principles of the lineage, analyzing why each generation evolved, how recurring design strategies resolved successive bottlenecks, what characterizes each generation's design DNA, where redesign effort concentrated across architectural components, and why those components became the persistent targets of innovation. Section 6 expands the discussion to application domains, detailing how YOLO frameworks have been customized across diverse sectors, including autonomous driving, agriculture, manufacturing, healthcare, and XR/AR environments. Section 7 presents a comprehensive quantitative and qualitative comparison that synthesizes benchmark data from multiple studies and implementations. Section 8 analyzes open challenges and outlines future research directions, connecting YOLO's trajectory to emerging paradigms in AI vision. Finally, Section 9 concludes the review by summarizing key findings, theoretical implications, and practical recommendations for real-time object detection.

1.6. The YOLO Evolution at a Glance: A Reader's Roadmap

Before the detailed analysis begins, this subsection condenses the fourteen-generation trajectory into a four-era roadmap that maps each era to its defining breakthroughs and target deployment scenarios so that readers can locate any generation—and the sections analyzing it—at a glance. The eras correspond to the paradigm phases described in Section 3.

Foundation era (2016–2018; YOLOv1–v3 [8,19,20]). Defining breakthroughs: detection reformulated as single-pass regression, anchor priors, and three-scale FPN fusion. Target scenario: establishing that real-time detection is viable on commodity GPUs. Characteristic limitation: coarse grid partitioning and anchor priors constrained small-object recall and required dataset-specific anchor tuning. Analyzed in Sections 3.1 and 3.2.

Optimization and productization era (2020–2023; YOLOv4–v8 and major derivatives [21–27]). Defining breakthroughs: CSP modularity, training-time bag-of-freebies, structural reparameterization, anchor-free decoupled heads, and multi-task support. Target scenario: industrial production pipelines spanning cloud to embedded devices. Charac-

teristic limitation: NMS post-processing and growing backbone complexity introduced latency variability and export friction. Analyzed in Sections 3.1, 3.3 and 3.9.

Attention and refinement era (2024–early 2025; YOLOv9–v12 [28–31]). Defining breakthroughs: programmable gradient information, NMS-free decoding, and hybrid CNN-attention backbones. Target scenario: low-latency end-to-end deployment combined with improved contextual modeling for dense scenes. Characteristic limitation: attention modules raised memory-bandwidth pressure, and NMS-free decoding traded a small accuracy margin for end-to-end simplicity. Analyzed in Sections 3.4 and 3.5.

Bifurcation era (mid–late 2025; YOLOv13 and YOLO26 [15,16]). Defining breakthroughs: hypergraph-based high-order correlation modeling on the accuracy branch and DFL/NMS elimination with training-time innovation on the deployment branch. Target scenarios: occlusion-heavy, accuracy-critical perception versus CPU-class and mobile deployment. Characteristic limitation: both branches rest primarily on their source reports, and independent uniform-protocol evaluations are only beginning to emerge (Section 7.10). Analyzed in Sections 3.6 and 3.7, with design principles distilled in Section 5, domain fit in Section 6, and benchmarks in Section 7.

Readers seeking model-selection guidance may proceed directly to the evidence-bounded suitability summary in Section 6.7, the source-reported operating-point comparisons in Section 7.2, and the constraint-indexed deployment playbook in Section 9.3. Readers interested in why the lineage evolved as it did may turn to the design-principle synthesis in Section 5.

2. Overview of Object Detection and Methodological Framework

Object detection represents one of the fundamental problems in computer vision, aiming to localize and classify objects within digital images and video streams [1]. Since its early emergence in the 2000s, object detection has evolved from region-based feature engineering toward deep-learning-driven end-to-end detection frameworks capable of real-time inference. Understanding the broader landscape of object-detection algorithms provides important context for analyzing the evolution of the YOLO family, since YOLO itself was conceived as a reaction to the computational bottlenecks and architectural complexity of earlier two-stage detectors.

2.1. Two-Stage vs. One-Stage Detection Paradigms

Modern deep learning-based detectors are commonly categorized into two principal architectural paradigms—two-stage and one-stage approaches—reflecting distinct trade-offs between accuracy, latency, and engineering complexity [1,4–8]. Two-stage detectors, exemplified by the R-CNN family, follow a sequential processing pipeline consisting of (i) region proposal generation and (ii) classification plus bounding-box refinement. The original R-CNN introduced the use of convolutional features to classify region proposals generated by selective search [4]. Fast R-CNN improved efficiency by sharing convolutional features across proposals through region-of-interest (ROI) pooling [5], while Faster R-CNN integrated a Region Proposal Network (RPN) to generate proposals directly from feature maps, eliminating external proposal mechanisms [6]. These models achieved strong performance and became the dominant paradigm in early object-detection benchmarks, but their multi-stage inference introduced latency that limited real-time deployment in edge or embedded environments.

In contrast, one-stage detectors perform localization and classification within a single forward pass, eliminating the intermediate region-proposal stage. The Single Shot MultiBox Detector (SSD) was among the first to achieve near real-time speed while retaining reasonable accuracy by using multi-scale feature maps to detect objects of various sizes [7].

YOLOv1 further streamlined this concept by reformulating detection as a pure regression task, predicting both objectness and bounding boxes from the entire image at once [8]. Subsequent one-stage models have refined this concept through improved feature fusion and optimization, steadily closing the accuracy gap with two-stage detectors.

2.2. Comparison of Major Algorithms: R-CNN, SSD, RetinaNet, and EfficientDet

The evolution of object-detection frameworks between 2014 and 2020 reflects a persistent trade-off among accuracy, efficiency, and architectural complexity. The R-CNN lineage provided high precision but suffered from computational redundancy, as each region proposal required separate feature extraction. SSD simplified this process but remained limited in its ability to detect small objects reliably due to its limited receptive field and coarse feature resolution. To address the imbalance between foreground and background samples that plagued one-stage models, RetinaNet [32] introduced the Focal Loss, which down-weights easy negatives to focus training on hard, informative examples—substantially improving single-stage accuracy.

Building upon this lineage, EfficientDet [33] introduced a scalable and hardware-aware design using compound coefficient scaling to balance depth, width, and resolution, alongside a BiFPN (Bidirectional Feature Pyramid Network) that enhances feature reuse across levels. EfficientDet demonstrated that carefully balanced architectural scaling could achieve strong performance while maintaining computational efficiency, demonstrating that network optimization and architectural efficiency could rival complex multi-stage designs. However, despite these advancements, most of these frameworks remained constrained by a compromise between speed and accuracy, especially for real-time applications such as autonomous driving, UAV surveillance, or robotics. Beyond these mainstream families, alternative anchor-free keypoint-based approaches—CornerNet's paired-corner detection [34], CenterNet's direct center-point prediction [35], and keypoint-triplet detection [36]—reframe detection as keypoint estimation, directly predicting object corners or centers together with their sizes, a paradigm that later influenced the anchor-free designs of YOLOv8 and YOLOv10.

2.3. Technical Distinctiveness of YOLO

Against this historical backdrop, the YOLO family emerged as a highly influential framework in object detection. Unlike region-based pipelines, YOLO treated detection as a unified regression problem, simultaneously predicting bounding boxes and class probabilities from full-image features in a single network pass. This design philosophy substantially reduced inference latency while maintaining competitive detection performance, effectively enabling real-time object detection on commodity GPUs.

Beyond its single-shot formulation, YOLO's distinctiveness can be attributed to three key technical characteristics.

First, it employs a global receptive field: instead of processing local candidate regions, the entire image contributes to feature representation, leading to context-aware predictions and fewer background errors.

Second, YOLO introduced a grid-based prediction mechanism, dividing the image into $S \times S$ cells that each predict B bounding boxes with associated class probabilities—an approach that remains foundational in subsequent versions.

Third, its emphasis on lightweight optimization and modular design (e.g., Darknet, CSPDarknet, GELAN backbones) has allowed continuous scalability from high-end servers to resource-limited embedded devices.

As the algorithm evolved, YOLO integrated the strengths of prior detectors while innovating at multiple levels—anchor-box prediction (YOLOv2), multi-scale feature aggre-

gation (YOLOv3), CSP and PANet fusion (YOLOv4/v5), decoupled heads and RepVGG blocks (YOLOv6/v7), transformer-based attention and programmable gradient optimization (YOLOv9–v12) [28,29], hypergraph-enhanced correlation modeling (YOLOv13) [15], and most recently, end-to-end NMS-free inference optimized for edge deployment (YOLO26) [16]. This consistent trajectory toward faster, more accurate, and more adaptable models underpins YOLO's central position in modern computer vision research. Modern YOLO variants commonly initialize their backbones with ImageNet-pretrained [37] weights, leveraging the 1.28 million annotated images across 1000 categories as a foundation for transfer learning before fine-tuning on detection-specific datasets such as PASCAL VOC [38] and MS COCO [39].

2.4. Methodological Framework of the Review

Building upon the preceding discussion of object-detection paradigms, this section outlines the methodological framework adopted in the present review to systematically analyze YOLO's progression. The framework is intended to ensure that the comparative and historical analyses presented in later sections are supported by transparent and reproducible procedures.

2.4.1. Review Methodology and Scope

Informed by established review-reporting principles [40], this study is presented as a structured narrative review of the YOLO literature rather than as a formal systematic review or meta-analysis. The core literature search covered major YOLO releases and related derivatives published or publicly released between 2016 and September 2025. Targeted update searches were completed in August 2026 to incorporate the formal YOLO26 technical report [16], independent domain-specific evaluations of recent releases [41,42], secondary analysis [43], and broader multi-version benchmarking [44]. Relevant sources were identified through IEEE Xplore, ACM Digital Library, ScienceDirect, MDPI, arXiv, Google Scholar citation tracking, and official project repositories when implementation-specific information was not available in a paper. Because the evidence base includes papers, technical reports, official documentation, and repositories, all quantitative claims were retained with their original source and protocol rather than treated as results of a new standardized benchmark.

The primary methodological objective is to develop a longitudinal synthesis that not only records architectural progression but also correlates these design choices with quantitative improvements in detection accuracy, model size, and inference speed. Unlike previous YOLO surveys (e.g., [9,10,17]), which typically end with YOLOv8 or YOLO11, this work integrates the most recent versions (YOLOv9 through YOLO26) [15,16,28–30] and situates them within the broader paradigm of one-stage detection.

2.4.2. Selection Criteria for Included YOLO Studies

Studies were included for detailed analysis according to predefined inclusion criteria: each study had to (i) explicitly examine a YOLO version or recognized derivative, (ii) present empirical or architectural analysis, and (iii) report benchmark results or comparative performance on public datasets. Studies that mentioned YOLO only superficially or duplicated a retained preprint or journal version were excluded. Studies relying exclusively on inaccessible proprietary data were excluded from the quantitative synthesis when they provided neither reproducible methodology nor traceable benchmark definitions; however, clearly documented application studies were retained for qualitative discussion where appropriate. Each retained document was analyzed in full text and annotated with metadata, including YOLO version, architecture type, dataset, and evaluation metrics (mAP, FPS, FLOPs, parameter count). Each retained source was additionally assigned to one of four

evidence tiers—(i) official release papers and primary technical reports, (ii) independent peer-reviewed comparative studies, (iii) independent preprints, and (iv) application-specific studies—and quantitative statements in this review are attributed to their tier rather than synthesized across tiers with equal confidence. In particular, values available only at tier (i) for the 2025 releases are presented as source-reported claims pending independent replication. This selection strategy was intended to improve the consistency and reliability of the evidence base for subsequent comparative analysis.

For recent releases, the evidence was classified into three levels. Level 1 comprised primary release papers, official technical reports, and official documentation used to describe architecture and source-reported benchmarks [15,16]. Level 2 comprised independent studies that directly evaluated the relevant models under a stated protocol [41,42]. Level 3 comprised secondary reviews, perspectives, and deployment-oriented commentaries used only for contextual interpretation [43,45,46]. The broader multi-version benchmark in [44] covers earlier versions and is used as contextual evidence about non-monotonic cross-version performance; it is not treated as a direct reproduction of YOLOv13 or YOLO26. When Level 2 evidence was unavailable, the manuscript explicitly labels the corresponding result as source-reported and applies a maturity caveat.

2.4.3. Categorization of Models

To facilitate consistent comparison, YOLO-related models were organized into three analytical tiers: baseline, enhanced, and application-specific architectures. Baseline models represent official releases defining each generation's canonical structure, from YOLOv1 [8] to YOLO26 [16], spanning fourteen major iterations introduced between 2016 and 2025, with formal technical coverage extended through the 2026 YOLO26 report [16]. Enhanced models include derivatives such as YOLOX, YOLOR, and YOLO-NAS, which modify baseline structures to optimize accuracy, speed, or generalization. Application-specific variants comprise domain-tailored implementations like YOLO-Agri, YOLO-Med, or lightweight mobile versions (e.g., YOLOv5n, YOLOv8n) designed for embedded deployment. This hierarchical categorization provides conceptual clarity and facilitates structured comparison across model families and mirrors the natural diversification of YOLO research across theoretical and applied fronts.

2.4.4. Taxonomy of Analysis: Structural, Algorithmic, and Domain-Level Dimensions

Finally, this review adopts a three-axis taxonomy to analyze the evolution of the YOLO lineage. The structural axis captures network-level innovations—backbone changes (Darknet → CSPDarknet → GELAN → R-ELAN with hypergraph computation [15]), neck structures (FPN, PANet, E-ELAN, FullPAD [15]), and detection heads (anchor-based → anchor-free → decoupled → end-to-end NMS-free [16,30]). The algorithmic axis examines training methodologies, loss-function refinements (from MSE to CIoU and PGI), activation and normalization techniques, and optimization heuristics (CutMix, Mosaic, self-distillation). The domain-level axis focuses on YOLO's adaptability across application sectors, analyzing how domain-specific constraints (e.g., latency in robotics, accuracy in medical imaging, scalability in agriculture) shape the modification of core architectures.

Collectively, these dimensions provide a unified analytical framework linking architectural development, algorithmic refinement, and application-specific deployment for the remainder of this paper, linking YOLO's architectural innovations to measurable performance outcomes and domain-specific relevance.

3. Architectural Evolution of YOLO (YOLOv1 to YOLO26)

The YOLO family's evolution reflects a continuous interplay between structural innovation and practical constraints of real-time inference. In this section, we trace that trajectory version by version—highlighting the core architectural decisions, performance gains, and trade-offs—culminating in the newest YOLO26. Where possible, we align with the classifications established in Section 2 (baseline vs. enhanced vs. domain-specific) and the structural/algorithmic taxonomy. Figure 1 shows the release years and major architectural transitions, and Table 1 summarizes key architectural attributes across versions: backbone type, neck module, head scheme (anchor-based, anchor-free, decoupled, or end-to-end), attention or gradient modules, and representative performance metrics (mAP, FPS, latency).

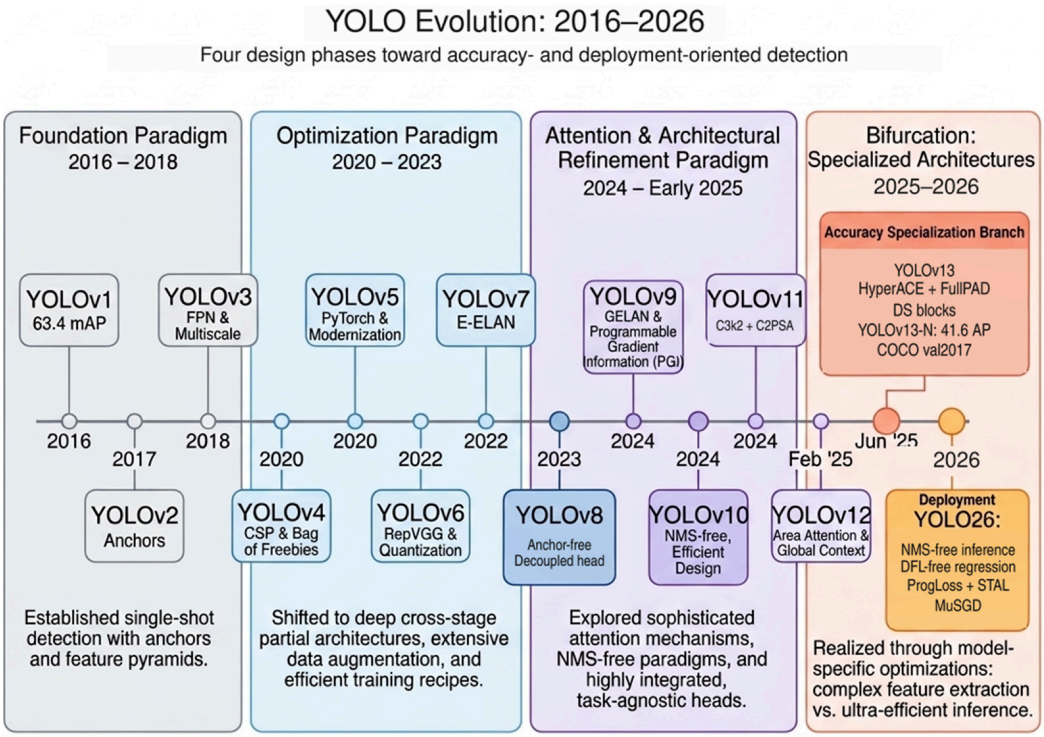


Figure 1. Evolution of the YOLO family from 2016 to 2026, organized into four broad design phases: foundation, optimization, attention and architectural refinement, and specialization for accuracy or deployment. Selected milestones highlight changes in feature extraction, multiscale fusion, prediction heads, and training strategies. The timeline extends through the formal YOLO26 technical report [16]. Individual accuracy values retain their original benchmark definitions and should not be interpreted as a common evaluation across generations.

3.1. YOLOv3 to YOLOv5: Multi-Scale Fusion and Feature Aggregation

YOLOv3 extended previous designs through the adoption of a deeper backbone architecture, Darknet-53, and employing multi-scale detection [20]. A key innovation was the introduction of three prediction scales corresponding to different feature-map resolutions, improving small-object detection. The architecture fused features from deeper and intermediate layers, enabling better contextual cues for bounding box regression. The loss functions and logistic classifiers were refined, but the overall prediction formulation continued to combine bounding-box regression with objectness prediction.

Table 1. Key Architectural Attributes and Innovations of YOLO Versions (v1 to YOLO26).

Version	Backbone/ Core Architecture	Neck/ Feature Fusion	Detection Head Type	Major Innovations/ New Modules	Key Strengths/ Technical Characteristics
YOLOv1 (2016) [8]	24 Conv + 2 FC (Darknet)	Single-Scale	Grid-based No anchors	End-to-end unified regression of class + bounding box	Extremely fast single-shot detection, limited recall for small/overlapping objects
YOLOv2 (YOLO9000) (2017) [19]	Darknet-19 (19 conv + 5 pool)	Passthrough layer for fine-grained features	Anchor-based Multi-scale	Batch normalization; Dimension clusters (k-means anchors); Multi-scale training; High-resolution classifier	Improved recall through anchor boxes; Better small object detection; Can detect 9000+ object classes (WordTree hierarchy)
YOLOv3 (2018) [20]	Darknet-53 (53 conv layers with residual connections)	Feature Pyramid Network (FPN) Multi-scale fusion	Anchor-based 3-scale detection	Multi-scale predictions (3 scales); Logistic regression for objectness; Independent logistic classifiers	Strong performance on small objects; Residual connections improve gradient flow; 3× detection scales (13 × 13, 26 × 26, 52 × 52)
YOLOv4 (2020) [21]	CSPDarknet53 (Cross-Stage Partial connections)	SPP (Spatial Pyramid Pooling) + PANet (Path Aggregation Network)	Anchor-based Coupled head	Mosaic data augmentation; Self-Adversarial Training (SAT); Mish activation; DropBlock; CIoU loss	Optimal balance between speed and accuracy; Bag-of-freebies and bag-of-specials training strategies; 43.5% mAP on COCO (65 FPS on V100)
YOLOv5 (2020–2022) [27]	CSPDarknet + Focus layer	PANet with CSP blocks	Anchor-based Coupled head	Auto-anchor calculation; Compound scaling (n/s/m/l/x variants); PyTorch native; Auto-augmentation; EMA model averaging	Industry-grade implementation; Easy deployment (ONNX, TensorRT, CoreML); Highly modular and customizable; Superior documentation and community support
YOLOv6 (2022) [25]	EfficientRep (Reparameterized backbone)	Rep-PAN (Reparameterized PANet)	Anchor-free Decoupled head	Self-distillation; Task Alignment Learning (TAL); Reparameterization for training-inference decoupling	Optimized for industrial deployment; Hardware-friendly operators; 52.5% mAP @ 121 FPS (T4, batch 32, for v6-L)
YOLOv7 (2023) [26]	E-ELAN (Extended Efficient Layer Aggregation Network)	E-ELAN based neck with concatenation-based feature fusion	Anchor-based coupled prediction head	Trainable bag-of-freebies; Coarse-to-fine lead head; RepConv reparameterization; Compound model scaling	YOLOv7-E6E reports 56.8 AP on COCO val2017 with 1280-pixel input and 36 FPS on V100 at batch size 1 [26]. E-ELAN and trainable bag-of-freebies support feature aggregation and training efficiency

Table 1. *Cont.*

Version	Backbone/ Core Architecture	Neck/ Feature Fusion	Detection Head Type	Major Innovations/ New Modules	Key Strengths/ Technical Characteristics
YOLOv8 (2023–2024) [22]	CSPDarknet with C2f modules (faster CSP bottleneck)	PAN-FPN with C2f blocks	Anchor-free Decoupled head	Anchor-free prediction; Task-aligned one-to-many assignment; Distribution Focal Loss (DFL); Multi-task support (detect/segment/ classify/pose)	Simplified architecture; Improved small object detection; 53.9% mAP @ 280 FPS (A100 for v8x); Unified framework for multiple vision tasks
YOLOv9 (2024) [29]	GELAN (Generalized Efficient Layer Aggregation Network)	GELAN-based multi-path aggregation	Anchor-free Decoupled head	Programmable Gradient Information (PGI); Reversible architecture for information retention; Auxiliary supervision branches	Addresses information bottleneck problem; 53.0% mAP @ 38 FPS (V100 for v9-C); Superior gradient flow management; Improved training stability for deep networks
YOLOv10 (2024) [30]	Spatial–channel decoupled downsampling; CIB and rank-guided block allocation	PAN/FPN-style fusion with efficiency-oriented blocks	NMS-free dual head with consistent one-to-many and one-to-one assignments	Lightweight classification head; large-kernel depthwise convolution; partial self-attention; NMS-free inference	The primary report gives 38.5 AP at 1.84 ms for YOLOv10-N and 54.4 AP at 10.70 ms for YOLOv10-X on NVIDIA T4 TensorRT FP16. No controlled quantization result or frame-time determinism metric is reported [30]
YOLO11 (2024) [31]	Ultralytics convolutional backbone with C3k2 blocks, ending in SPPF and C2PSA	PAN/FPN-style multi-scale fusion with C3k2 blocks	Anchor-free decoupled head	C3k2; SPPF; C2PSA with Position-Sensitive Attention blocks; multi-task support	YOLO11 is documented primarily through the official software documentation rather than a formal research paper. Detailed causal performance or memory-reduction claims are therefore not made [31]
YOLOv12 (2025) [28]	Attention-centric hierarchy with R-ELAN and Area Attention	R-ELAN-based multi-scale feature aggregation	Anchor-free decoupled head	Area Attention using l equal vertical or horizontal regions (default $l = 4$); residual scaling; FlashAttention; no positional encoding	YOLOv12-N reports 40.6 AP at 1.64 ms on NVIDIA T4 under the primary study protocol. Area Attention reduces regional attention cost but remains quadratic in the number of spatial tokens [28]

Table 1. *Cont.*

Version	Backbone/ Core Architecture	Neck/ Feature Fusion	Detection Head Type	Major Innovations/ New Modules	Key Strengths/ Technical Characteristics
YOLOv13 (2025) [15]	DS-Conv backbone with HyperACE blocks (Hypergraph-based Adaptive Correlation Enhancement)	FullPAD paradigm distributing hypergraph-enhanced features through three pipeline tunnels (backbone-to-neck, intra-neck, neck-to-head)	Anchor-free Decoupled head	HyperACE (learnable hyperedge construction with linear-complexity message passing O($N \cdot M$)); FullPAD (Full-Pipeline Aggregation-and-Distribution); Depthwise-separable (DS) blocks for parameter reduction; Global high-order correlation modeling beyond pairwise attention	First YOLO release to integrate hypergraph computation for multi-to-multi relationship modeling; YOLOv13-N reports 41.6 AP at 1.97 ms on T4 TensorRT FP16, with 2.5 M parameters and 6.4 GFLOPs; YOLOv13-X reports 54.8 AP. In the YOLOv13-S source ablation, the complete HyperACE–FullPAD configuration reaches 48.0 AP, compared with 47.1 AP when hypergraph-enhanced distribution is disabled and 47.6–47.8 AP when only one FullPAD pathway is retained [15]
YOLO26 (2025) [16]	Streamlined CSPDarknet-derived backbone with DFL removal	PAN-FPN with simplified feature aggregation	End-to-end NMS-free Decoupled head with TAL-based one-to-one assignment	DFL (Distribution Focal Loss) removal in favor of direct L1 regression; NMS-free inference via dual-head training (one-to-many auxiliary + one-to-one primary); Progressive Loss Balancing (ProgLoss); Small-Target-Aware Label Assignment (STAL); MuSGD optimizer (SGD-Muon hybrid)	Deployment-oriented design: across N/S/M/L/X scales the primary report gives 40.9–57.5 AP at 1.7–11.8 ms (T4 TensorRT), with up to 43% faster CPU ONNX inference for the nano scale; the one-to-one path trails the one-to-many path by 0.6–0.8 AP; STAL improves APS from 29.0 to 29.6 and MuSGD reaches 47.4 AP in 500 epochs versus 47.0 in 600 with SGD; multi-task support (detection/segmentation/pose/OBB/classification) with multi-backend export paths documented; controlled INT8 accuracy results are not reported in the primary paper

3.2. YOLOv1 and YOLOv2: Foundational Design and Early Optimizations

Introduced by Redmon et al. [8], YOLOv1 represented a significant conceptual shift in object detection by reformulating the task as a unified regression problem. It divides the input image into an S × S grid, with each grid cell predicting bounding boxes and class probabilities in a single pass. Architecturally, the model consisted of 24 convolutional layers followed by two fully connected layers. Although this architecture enabled real-time inference, it exhibited notable limitations in localization accuracy, object recall—particularly for small objects—and the detection of overlapping instances. The architecture also lacked multi-scale feature fusion, which constrained its ability to generalize across object sizes.

The YOLOv1 architecture comprises 24 convolutional layers followed by two fully connected layers that regress to a 7 × 7 × 30 output tensor encoding bounding boxes and class probabilities for each grid cell (Figure 2) [8]. This unified regression framework eliminated the region-proposal stage of contemporary two-stage detectors, achieving 45 FPS on a Titan X GPU—a 155× speedup over R-CNN—while attaining 63.4% mAP@0.5 on PASCAL VOC 2007 [8]. However, the coarse 7 × 7 grid and fixed two-box-per-cell constraint limited small-object localization, motivating the refinements introduced in YOLOv2.

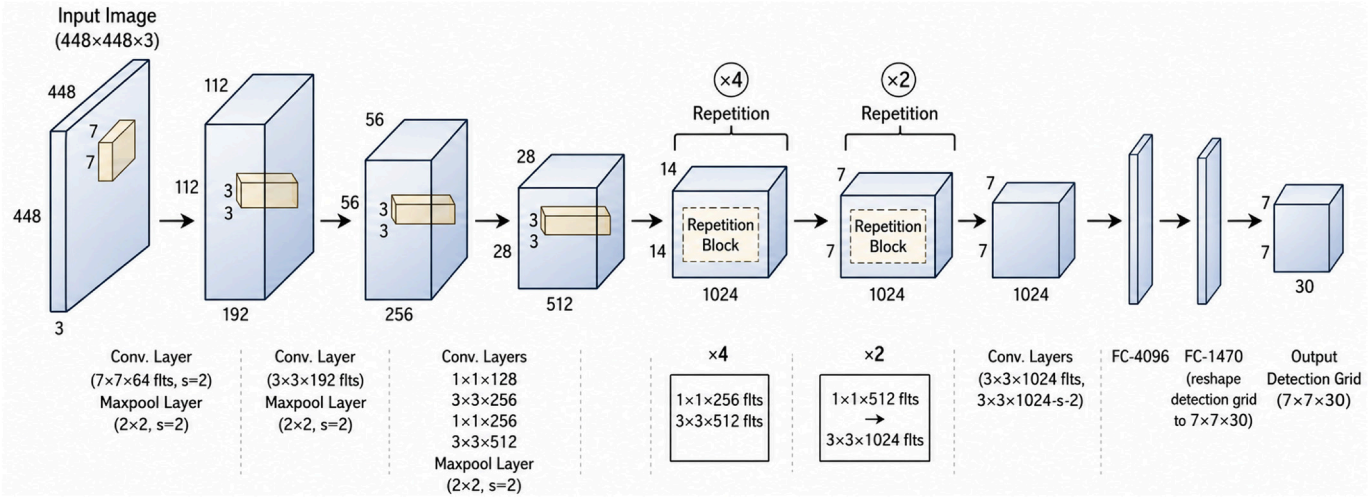


Figure 2. YOLOv1 Architecture: Unified single-shot detection pipeline comprising 24 convolutional layers (alternating 1 × 1 reduction and 3 × 3 extraction inspired by Network-in-Network) followed by two fully connected layers regressing to a 7 × 7 × 30 output tensor encoding bounding boxes and class probabilities. Adapted from Redmon et al. [8].

YOLOv2 (also known as YOLO9000) addressed many of these shortcomings by introducing anchor boxes, multi-scale training, and a new backbone network, Darknet-19, consisting of 19 convolutional layers plus 5 pooling layers [19]. YOLOv2 also used batch normalization, higher-resolution input, and dimension clustering to set anchor aspect ratios. These enhancements improved detection recall and localization precision. However, YOLOv2 still lacked deep feature aggregation and struggled with very small objects.

Architecturally, YOLOv2 integrates the Darknet-19 backbone with universally applied batch normalization [47], a passthrough layer for fine-grained feature preservation, and anchor-based prediction with k-means-derived priors (Figure 3) [19]. Multi-scale training across resolutions $\{320^2, \ldots, 608^2\}$ further enhances robustness to object-size variation. These refinements elevate mAP@0.5 from 63.4% (YOLOv1) to 76.8% on PASCAL VOC 2007 while maintaining 40–90 FPS, establishing YOLOv2 as the first YOLO variant to achieve competitive accuracy with contemporary two-stage detectors (Faster R-CNN: 73.2% mAP [6]) without sacrificing speed [19]. Detailed mechanisms—including *k*-means

clustering for anchor dimensions, passthrough spatial reorganization, and the 13 × 13 × 125 output formulation—are described in [19].

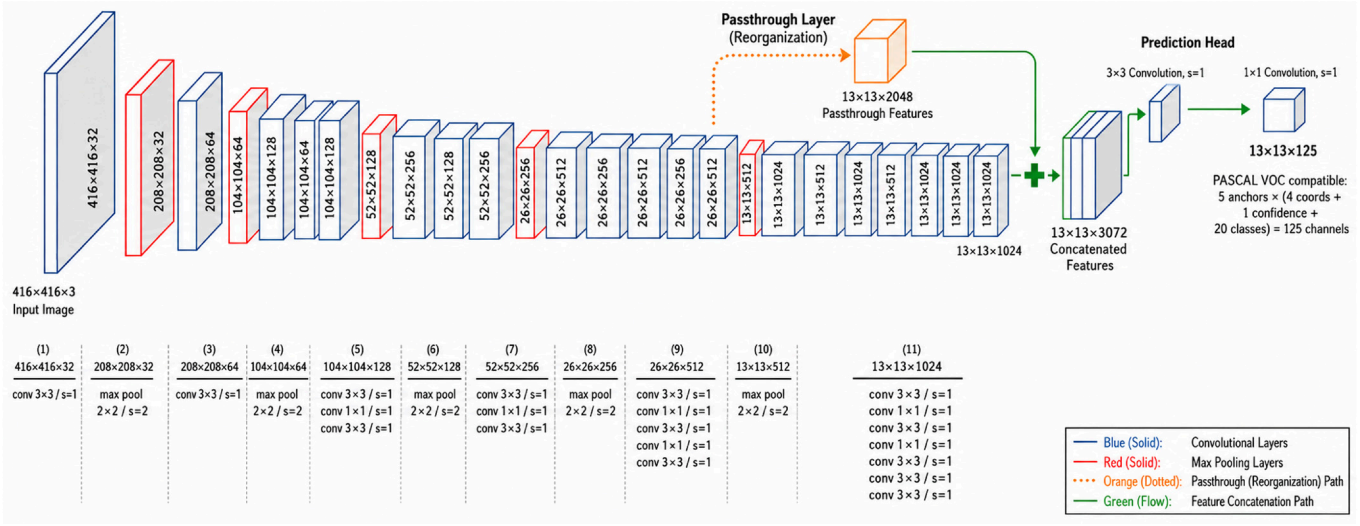


Figure 3. YOLOv2 Architecture: Darknet-19 backbone (19 convolutional layers and 5 max-pooling layers with universal batch normalization [47]) coupled with anchor-based detection head and passthrough layer for multi-scale feature fusion. Adapted from Redmon and Farhadi [19].

YOLOv3′s three-scale detection architecture integrates the Darknet-53 backbone with residual connections [48] and an FPN-inspired multi-scale prediction head (Figure 4) [49]. Three detection heads operate at 13 × 13, 26 × 26, and 52 × 52 resolutions, each employing three anchor boxes derived from *k*-means clustering, with a transition from softmax to logistic regression enabling multi-label classification [20]. In the 608 × 608 evaluation reported in [20], YOLOv3 achieves 33.0 AP, 57.9 AP50, and 18.3 APS on COCO; the same comparison lists 5.0 APS for YOLOv2. Detailed receptive-field analysis and the per-scale anchor assignment strategy are described in [20,49].

YOLOv4 [21] introduced CSPDarknet53 together with SPP (Spatial Pyramid Pooling) and PANet-based feature aggregation, significantly expanding the framework's representational capacity, combining channel-splitting and partial residual pathways (CSP) to reduce computation while preserving performance. YOLOv4 also embraced training optimizations like mosaic augmentation, Self-Adversarial Training (SAT), and DropBlock.

Three synergistic components define the YOLOv4 architecture (Figure 5): the CSP-Darknet53 backbone with Cross-Stage Partial connections for reduced computational redundancy [50], an SPP [51] + PANet [52] neck for multi-scale feature fusion, and a three-scale anchor-based detection head [21]. Training-time innovations include Mosaic augmentation, Self-Adversarial Training, Mish activation [53], and Complete-IoU loss [54]. YOLOv4 combines architectural and training modifications intended to improve the accuracy–speed trade-off [21]. Comparisons with EfficientDet, a one-stage detector, should preserve the evaluation conditions reported by each source [33]. Detailed CSP channel-splitting mechanics, SPP kernel configurations, and the bag-of-freebies/bag-of-specials taxonomy are presented in [21,50].

Although initially released outside the traditional peer-review process, YOLOv5 became one of the most widely adopted YOLO implementations through its robust open-source ecosystem and production-oriented tooling [2,9,22]. It introduced more flexible model scaling (nano, small, medium, large, and extra-large variants), improved deployment support (PyTorch, ONNX), and lighter structures for edge compatibility. Its neck and head modules were borrowed from YOLOv4 (PANet, CSP) but simplified.

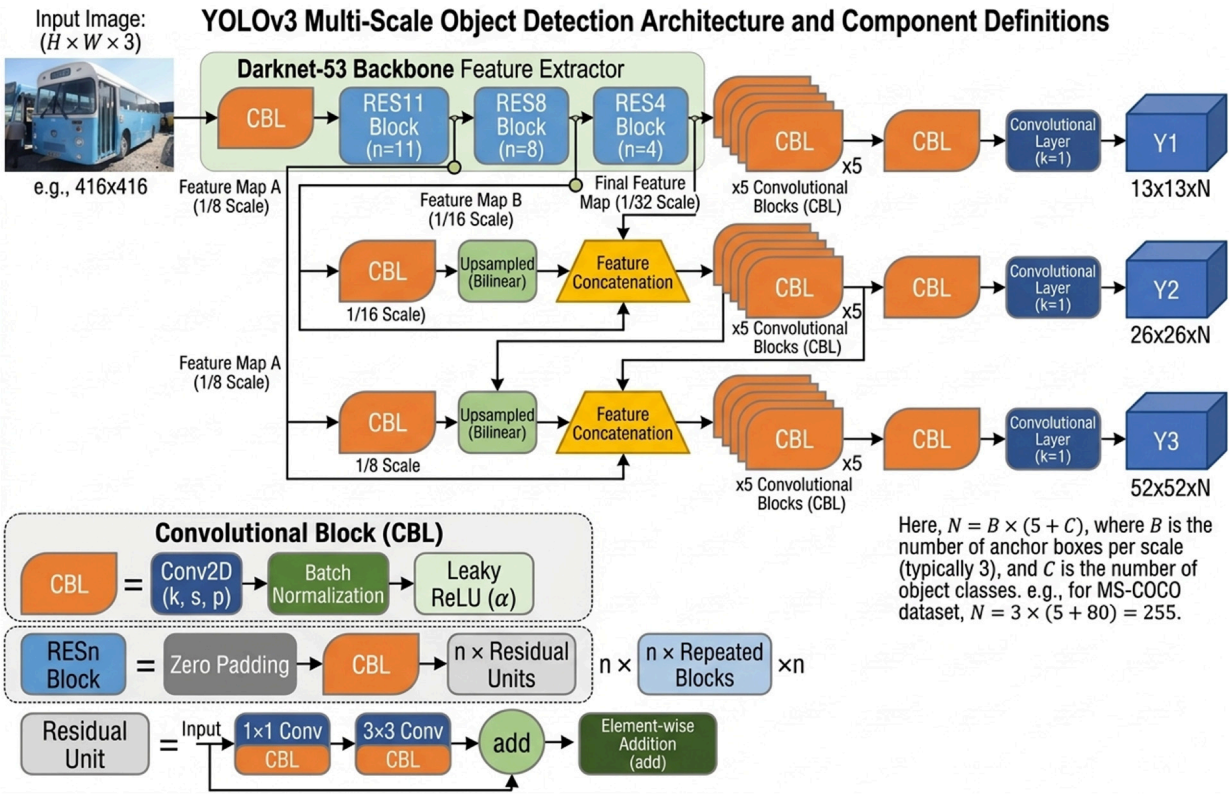


Figure 4. YOLOv3 Architecture: Multi-scale detection via FPN-inspired feature pyramid [49] with three detection heads at 13 × 13, 26 × 26, and 52 × 52 resolutions, employing Darknet-53 backbone with residual connections [48] for enhanced gradient propagation. Adapted from Redmon and Farhadi [20].

The YOLOv5 architecture is characterized by production-oriented refinements rather than architectural novelty (Figure 6) [27]. Key features include the Focus layer for efficient space-to-depth transformation, CSP-enhanced backbone and PANet neck, an anchor-based three-scale head with auto-anchor calculation, and compound scaling generating five variants $\{n, s, m, l, x\}$ for deployment flexibility. The PyTorch-native implementation supports multi-format export (ONNX, TensorRT, CoreML, TFLite), enabling broad edge-to-cloud deployment. YOLOv5x achieves 50.7% mAP@[0.5:0.95] on COCO with sub-13 ms latency on a V100 GPU (batch 1), surpassing YOLOv4's 43.5% while maintaining comparable throughput [22,27].

Collectively, these releases established the foundation for modern YOLO architectures, combining deeper feature representations, multi-scale detection, and increasingly sophisticated training strategies to narrow the accuracy gap with heavier detectors while preserving real-time throughput. Beyond conventional benchmarks, YOLOv2's anchor-based design proved adaptable to specialized domains—for example, combined with Kalman filtering for accurate zebrafish tracking in biological research [55]—foreshadowing the domain diversification examined in Section 6.

3.3. YOLOv6, YOLOv7, YOLOv8: Efficiency, Decoupling, and Modular Design

Beginning with YOLOv6 and continuing through YOLOv7, architectural development increasingly emphasized deployment efficiency and task-specific head decoupling. YOLOv6 introduced several architectural refinements specifically aimed at industrial deployment environments, focusing on streamlined operations for embedded inference [25].

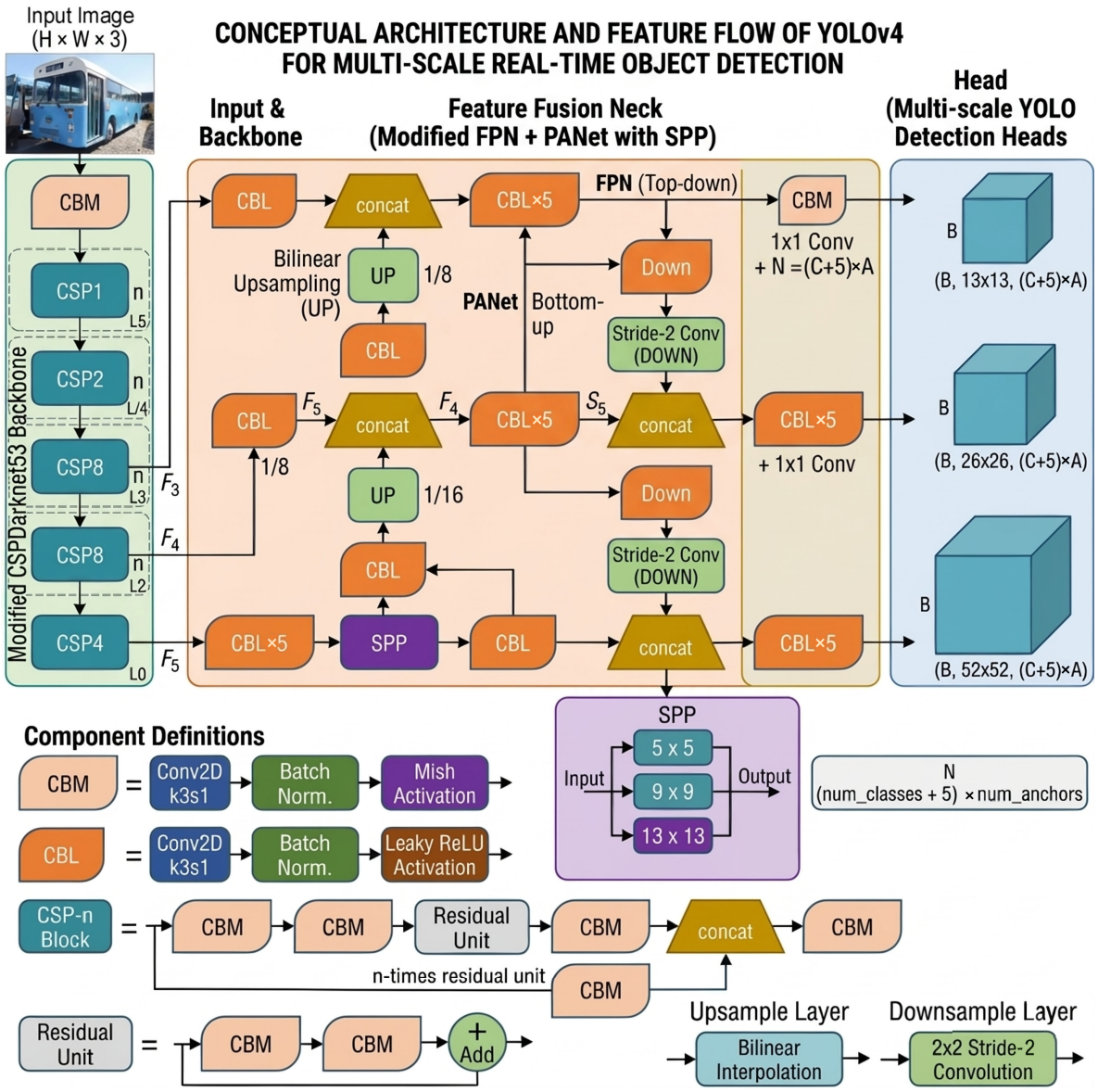


Figure 5. YOLOv4 Architecture: Three-stage design integrating CSPDarknet53 backbone [50] with Cross-Stage Partial connections, an SPP [51] and PANet [52] neck for multi-scale feature fusion, and a three-scale detection head augmented by comprehensive training optimizations (Mosaic augmentation, Mish activation [53], Complete-IoU loss [54]). Adapted from Bochkovskiy et al. [21].

YOLOv6 is optimized for industrial deployment through three key innovations (Figure 7): the EfficientRep backbone with RepVGG-style reparameterization [56] (a multi-branch training topology fused into a single-branch structure for inference), the Rep-PAN reparameterized neck, and a decoupled head separating classification and localization branches [25]. Task Alignment Learning (TAL) introduces dynamic label assignment jointly optimizing classification score and IoU prediction, while self-distillation provides richer supervisory signals than hard labels alone [25]. Hardware-friendly operator selection ensures predictable latency across diverse backends. YOLOv6-N achieves 35.9% mAP@[0.5:0.95] at 1.2 ms latency (NVIDIA T4, TensorRT, 802 FPS at batch 1), and YOLOv6-L attains 52.5% mAP@[0.5:0.95] at 121 FPS (batch 32) [25]. Detailed RepBlock fusion mathematics, TAL alignment metrics, and self-distillation training schedules are presented in [25,56].

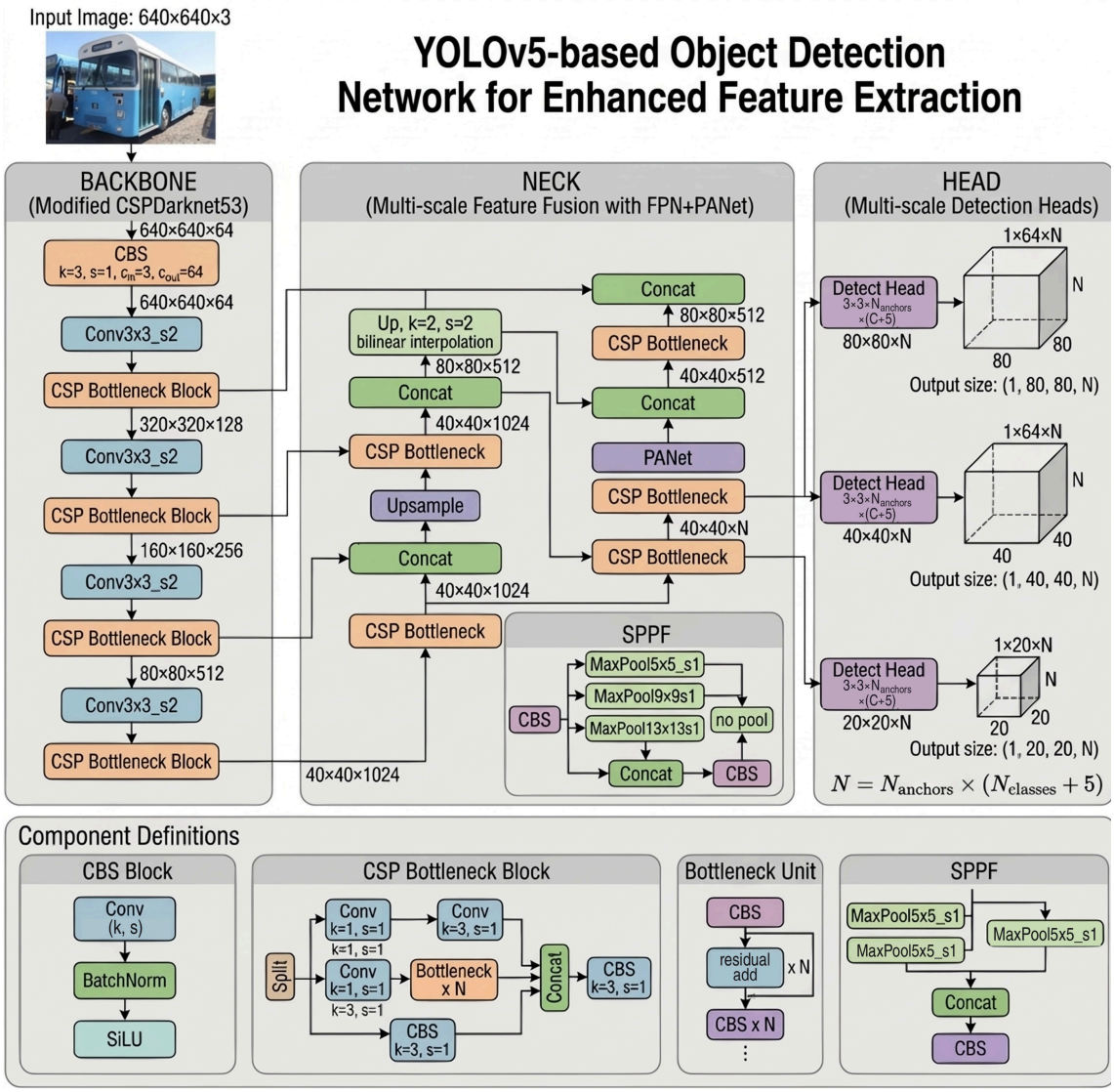


Figure 6. YOLOv5 Architecture: Production-grade modular design featuring the Focus layer for efficient space-to-depth transformation, CSP-enhanced backbone, PANet neck, and anchor-based three-scale head with auto-anchor calculation. Compound scaling generates five variants (n/s/m/l/x) supporting multi-format export (ONNX, TensorRT, CoreML, TFLite) for edge-to-cloud deployment flexibility. Adapted from Jocher et al. [27].

YOLOv7 introduced E-ELAN (Extended Efficient Layer Aggregation Network) to enhance feature diversity without increasing overhead [26], along with "trainable bag-of-freebies" strategies that improve performance without increasing inference cost [26].

The YOLOv7 architecture centers on the Extended Efficient Layer Aggregation Network (E-ELAN) backbone, which employs expand-shuffle-merge operations to achieve diverse gradient paths without the parameter growth of densely connected topologies (Figure 8) [26]. The neck retains a PANet structure with E-ELAN blocks, while the head incorporates RepConv reparameterization [56]. Trainable bag-of-freebies—including coarse-to-fine lead head supervision, implicit knowledge learning, and EMA normalization—improve training without inference cost. Compound scaling generates six variants (YOLOv7 through YOLOv7-E6E) using the source paper's compound scaling strategy. YOLOv7 achieves 51.4 AP on COCO test-dev at 161 FPS on a V100 under the paper's batch-1 protocol, and YOLOv7-E6E reaches 56.8% mAP@[0.5:0.95], under its separately

reported high-resolution protocol [26]. Detailed E-ELAN gradient flow analysis, compound scaling coefficients, and the complete bag-of-freebies enumeration are presented in [26].

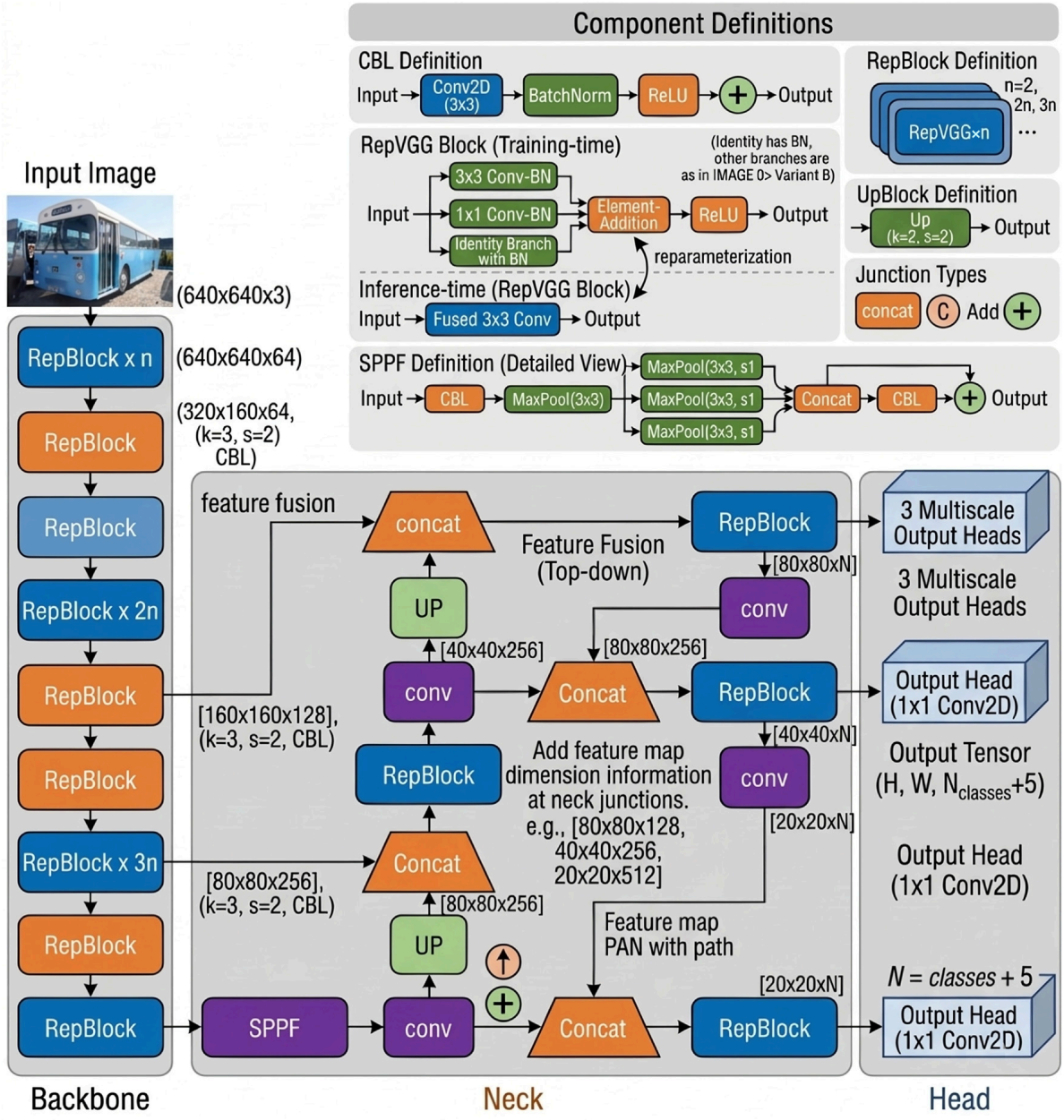


Figure 7. YOLOv6 Architecture: Industrial-deployment optimization featuring the EfficientRep backbone with RepVGG-style reparameterization [56] (a multi-branch training topology fused into a single-branch structure for inference), Rep-PAN reparameterized neck, decoupled classification-localization head, and Task Alignment Learning for dynamic label assignment. Adapted from Li et al. [25].

YOLOv8 further strengthened the framework's modular design philosophy while expanding support for multi-task vision applications [22]. It embraced an anchor-free approach, simplified head structures, and broader task support (detection, segmentation akin to Mask R-CNN [57], pose). Its design prioritized maintainability, flexibility, and ease of customization in production pipelines.

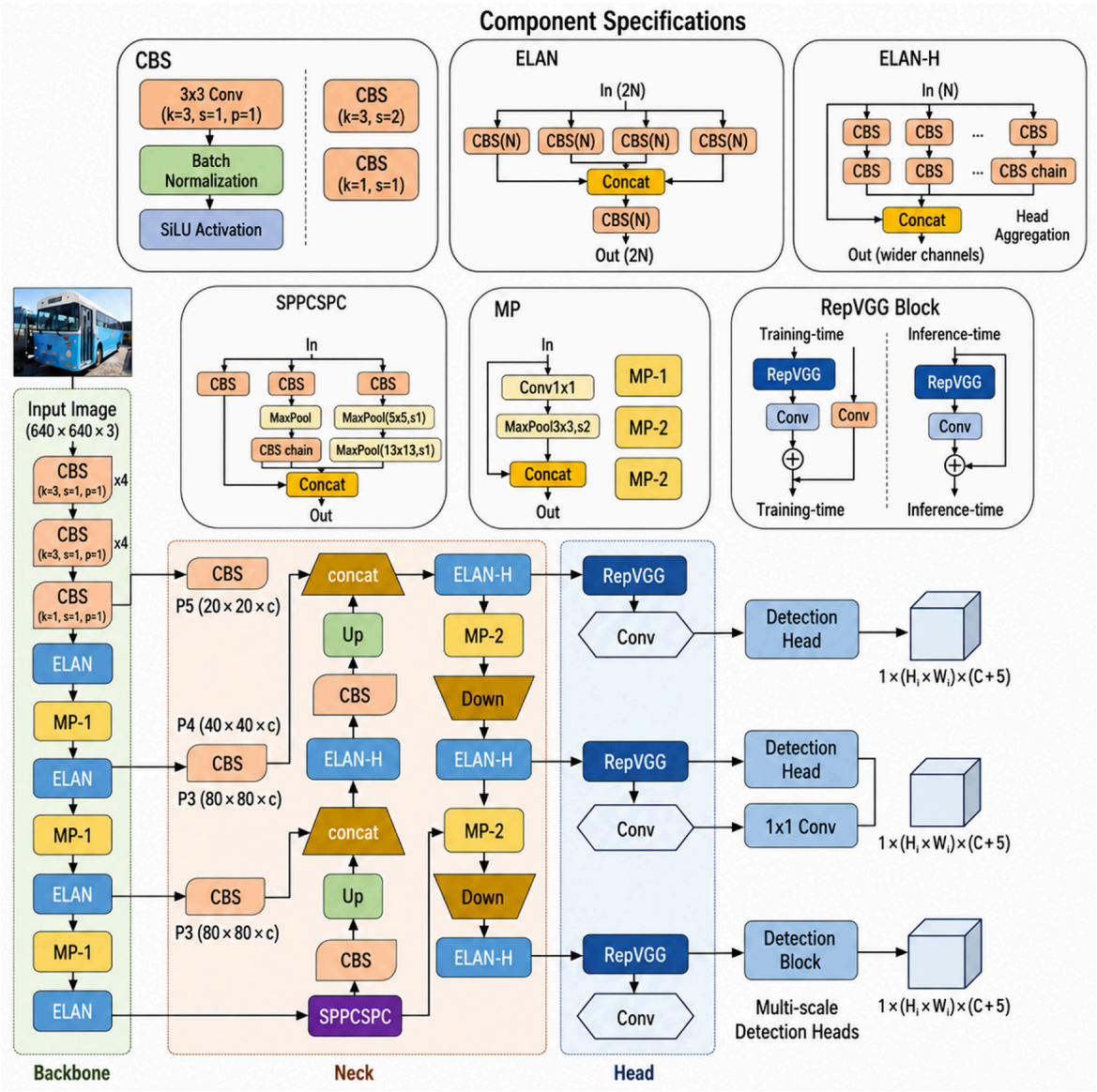


Figure 8. YOLOv7 Architecture: Extended Efficient Layer Aggregation Network (E-ELAN) backbone employing expand-shuffle-merge operations for diverse gradient paths, PANet neck with E-ELAN integration, RepConv reparameterization [56] in the detection head, and trainable bag-of-freebies pushing the accuracy-efficiency Pareto frontier. Adapted from Wang et al. [26].

YOLOv8 represents an important transition from anchor-based to anchor-free detection with task-agnostic multi-task support (Figure 9) [22]. Key components include the C2f (faster CSP bottleneck with two convolutions) backbone module replacing YOLOv5′s C3, a PAN-FPN neck with C2f integration, an anchor-free decoupled head with task-aligned assignment, and Distribution Focal Loss (DFL) for probabilistic bounding box regression. Multi-task heads—detection, instance segmentation (YOLOv8-Seg), and pose estimation (17 COCO keypoints)—branch from the shared backbone-neck. The official YOLOv8 documentation reports 53.9 AP for YOLOv8x on COCO val2017 at 640-pixel input, with an A100 TensorRT latency of 3.53 ms [22].

These variants represent a transitional phase in which accuracy improvements are balanced against architectural simplicity and practical deployment requirements.

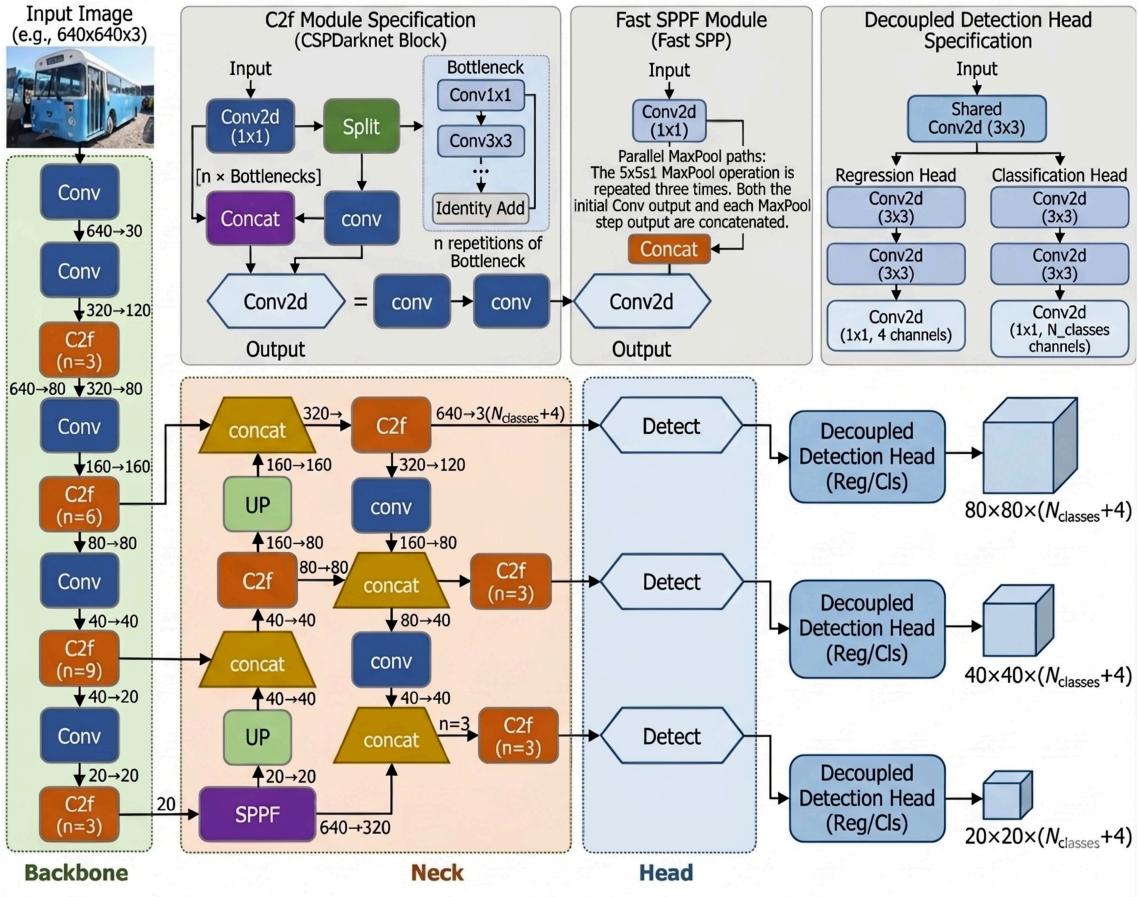


Figure 9. YOLOv8 Architecture: Anchor-free paradigm shift featuring the C2f (faster CSP bottleneck with two convolutions) backbone replacing YOLOv5's C3, PAN-FPN neck with C2f integration, anchor-free decoupled head with task-aligned assignment, and Distribution Focal Loss (DFL) for probabilistic bounding box regression. Task-agnostic backbone-neck supports multi-task outputs (detection, segmentation [22], pose estimation). Adapted from Jocher et al. [22].

3.4. YOLOv9, YOLOv10, YOLO11: Gradient Routing, End-to-End Detection, and Partial Attention

A subsequent stage in the evolution of the YOLO family is represented by YOLOv9, which introduced Programmable Gradient Information (PGI) and GELAN (Generalized Efficient Layer Aggregation Network) to better manage information flow and feature aggregation across layers [29]. These modules allowed more expressive gradient routing over layers, making training deeper models more stable.

YOLOv9 was designed to mitigate the information bottleneck problem commonly observed in deep neural networks through two principal innovations (Figure 10): Programmable Gradient Information (PGI) and the Generalized Efficient Layer Aggregation Network (GELAN) [29]. PGI introduces auxiliary reversible branches that preserve complete input information alongside the main pathway, enabling programmable gradient routing through learnable scaling parameters. GELAN extends E-ELAN [26] with cross-stage connections aggregating features across non-adjacent stages, providing multi-scale gradient flow without DenseNet-style memory overhead. Auxiliary detection heads at intermediate depths provide additional gradient signals during training and are removed at inference for zero overhead. YOLOv9-C reports 53.0 AP on COCO val2017 in the cited study [29]. The reported ablations examine GELAN and PGI under the study's own training and evaluation settings. Detailed mathematical formulations of reversible branch transformations, gradient routing mechanisms, and GELAN cross-stage aggregation are

presented in [29]. Detailed visualizations of YOLOv9′s gradient flow and ablation interpretability are documented in subsequent analyses [58,59], confirming PGI's role in stabilizing deep-network training.

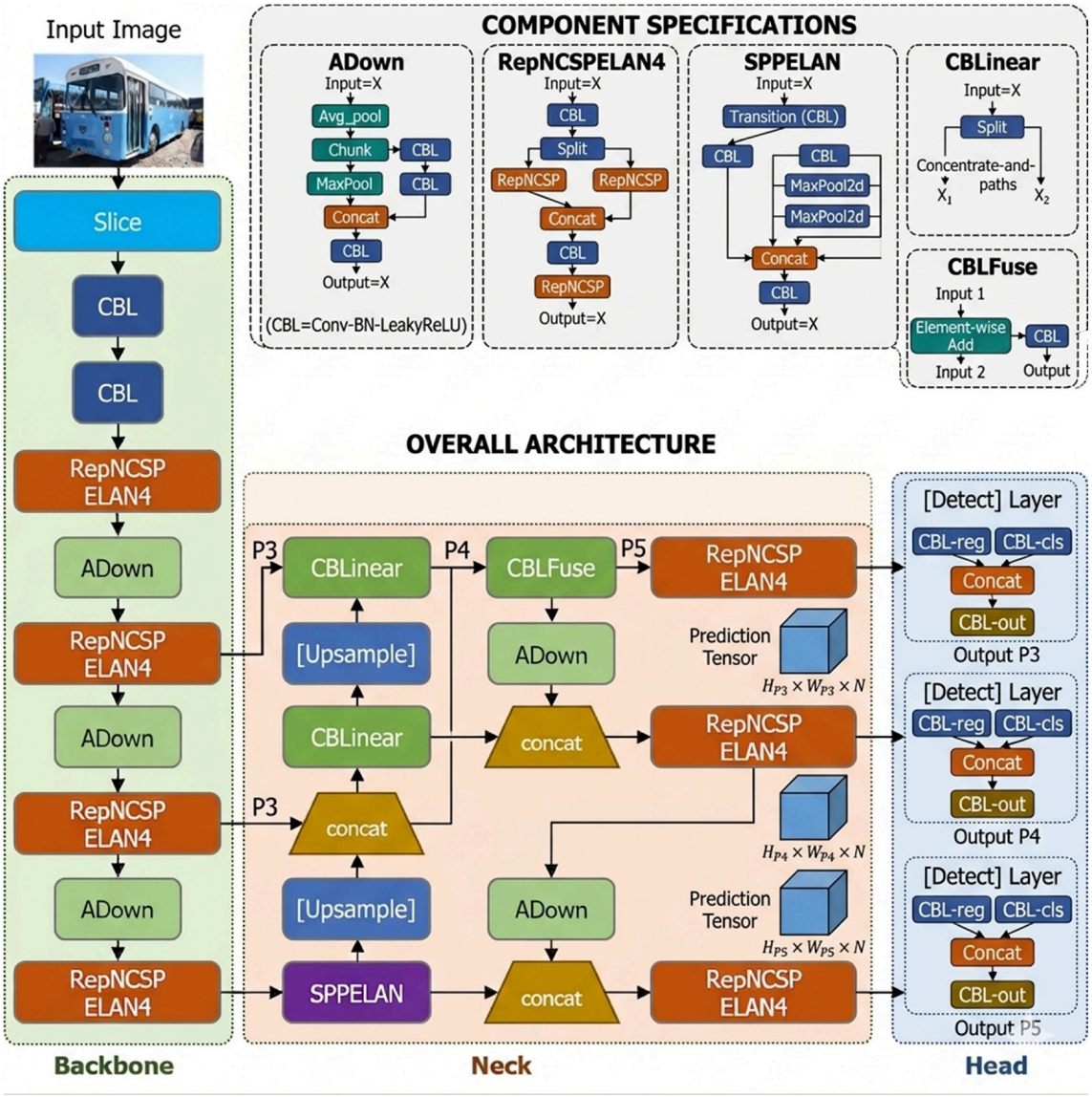


Figure 10. YOLOv9 Architecture: Programmable Gradient Information (PGI) introducing auxiliary reversible branches for complete input preservation, and the Generalized Efficient Layer Aggregation Network (GELAN) extending E-ELAN [26] with cross-stage connections across non-adjacent stages, jointly addressing the information bottleneck problem in deep networks while providing multi-scale gradient flow. Adapted from Wang et al. [29].

YOLOv10 emphasizes end-to-end efficiency through consistent dual-label assignment and holistic efficiency–accuracy co-design rather than model-specific quantization methods (Figure 11) [30]. Its principal innovations include a lightweight classification head, spatial–channel decoupled downsampling, rank-guided block design, large-kernel depthwise convolution, partial self-attention, and an NMS-free detection path. During training, consistent one-to-many and one-to-one assignments provide complementary supervision; at inference, the one-to-one path produces predictions without NMS.

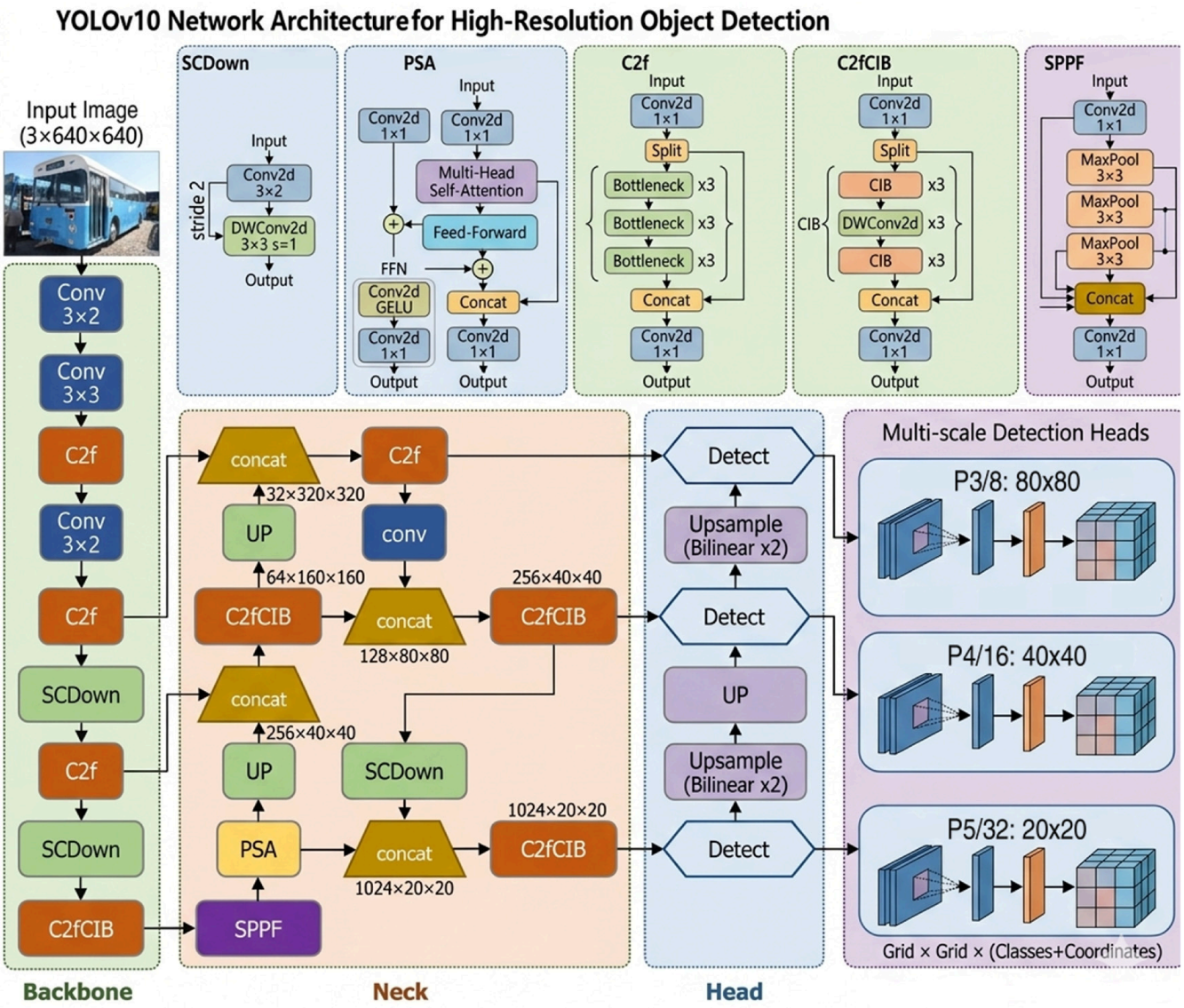


Figure 11. Conceptual overview of YOLOv10, organized into backbone, neck, and detection components. The module insets introduce spatial–channel decoupled downsampling, partial self-attention, C2f, compact inverted blocks, and spatial pyramid pooling. For a 640 × 640 input, the labeled detection scales are 80 × 80, 40 × 40, and 20 × 20, corresponding to strides of 8, 16, and 32. Consistent dual assignments support training, with the one-to-one branch used for NMS-free inference [30].

The official study reports 38.5 AP for YOLOv10-N at 1.84 ms and 54.4 AP for YOLOv10-X at 10.70 ms on NVIDIA T4 TensorRT FP16, with YOLOv10-B showing 46% lower latency and 25% fewer parameters than YOLOv9-C at comparable accuracy [30]. These results demonstrate low source-reported latency and a simplified end-to-end graph, but the study does not report frame-to-frame latency variance; accordingly, NMS-free inference should not be described as deterministic runtime.

YOLO11 maintains the conventional backbone-neck-head architecture while incorporating additional contextual modeling mechanisms (Figure 12) [31]. C3k2 is a configurable CSP-style block derived from C2f; depending on its configuration, the repeating unit can use a bottleneck, a C3k block, or a bottleneck paired with a PSABlock.

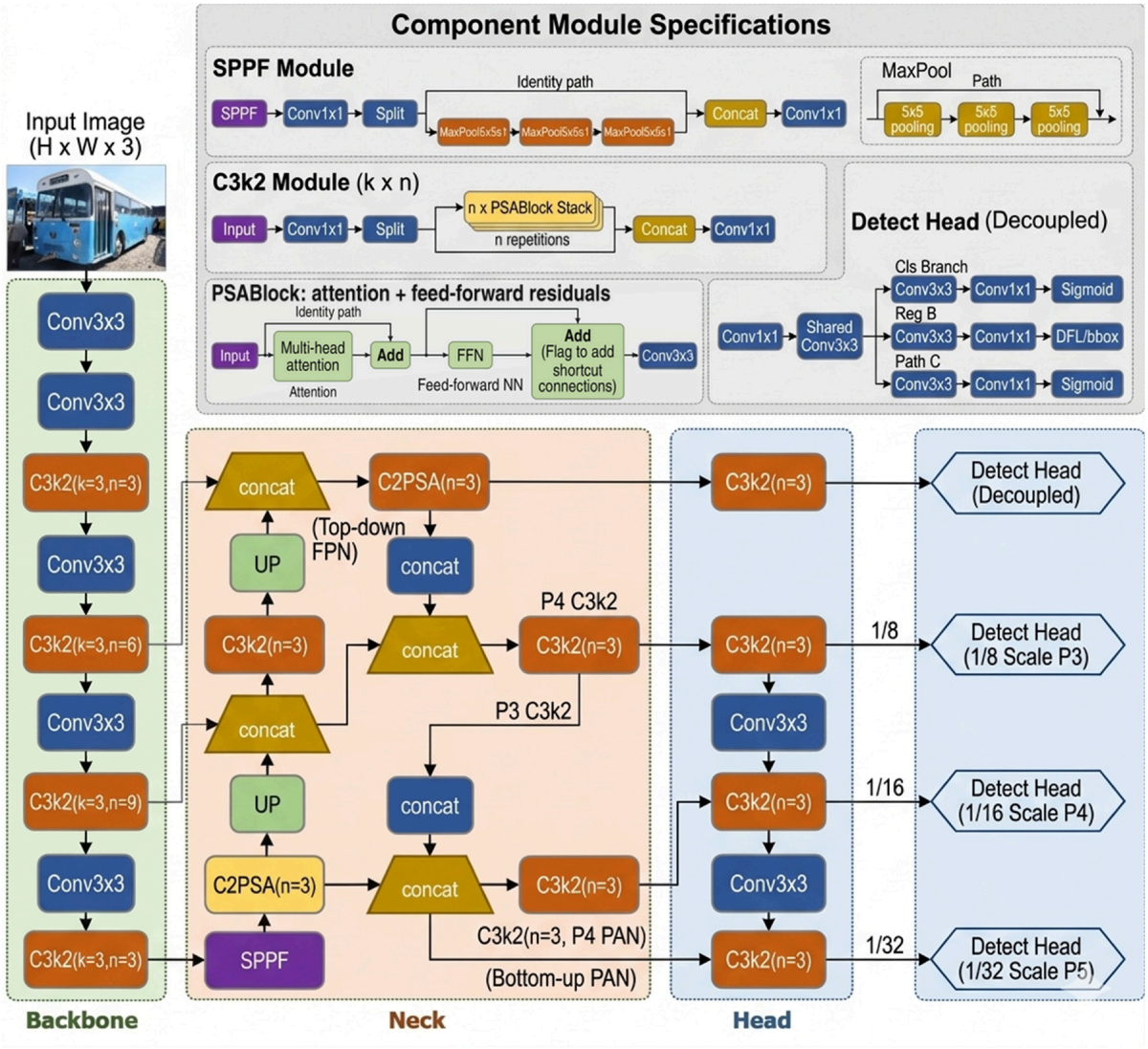


Figure 12. Conceptual overview of YOLO11 feature extraction, feature fusion, and detection, with insets highlighting selected processing modules. The PSABlock inset illustrates multi-head attention followed by residual addition, feed-forward processing, and a second residual addition. Features extracted at different resolutions are combined through top-down and bottom-up fusion to support object detection across scales. Decoupled prediction separates classification from bounding-box regression, while the module illustrations accompany the architectural discussion in the main text [31].

C2PSA is positioned after SPPF and divides the projected features into two channel groups. One group passes through stacked Position-Sensitive Attention (PSABlock) modules, each composed of multi-head attention and a two-layer feed-forward network with residual connections, while the other group provides a bypass path. The two groups are then concatenated and fused. This design adds contextual modeling without replacing the full convolutional hierarchy. Performance values should remain attached to the exact model scale and runtime reported in the source-specific comparison of Section 7.2.

Taken together, YOLOv9, YOLOv10, and YOLO11 emphasize gradient-aware learning, end-to-end optimization, and contextual feature modeling, respectively [29–31].

3.5. YOLOv12: Attention-Centric Real-Time Detection

YOLOv12 may be characterized as an attention-centric real-time detector built around Area Attention and Residual Efficient Layer Aggregation Networks (R-ELAN) (Figure 13) [28]. Area Attention divides a feature map into l equal regions along the vertical

or horizontal axis using a reshape operation; the default setting in the primary report is $l = 4$. Attention is computed independently within each region, retaining a larger receptive field than small local windows while reducing the arithmetic cost of full global attention. R-ELAN adds block-level residual connections with scaling and redesigns feature aggregation to stabilize optimization in attention-augmented large models. The architecture also adopts FlashAttention, removes positional encoding, reduces the MLP ratio, and uses convolution-based operators where appropriate.

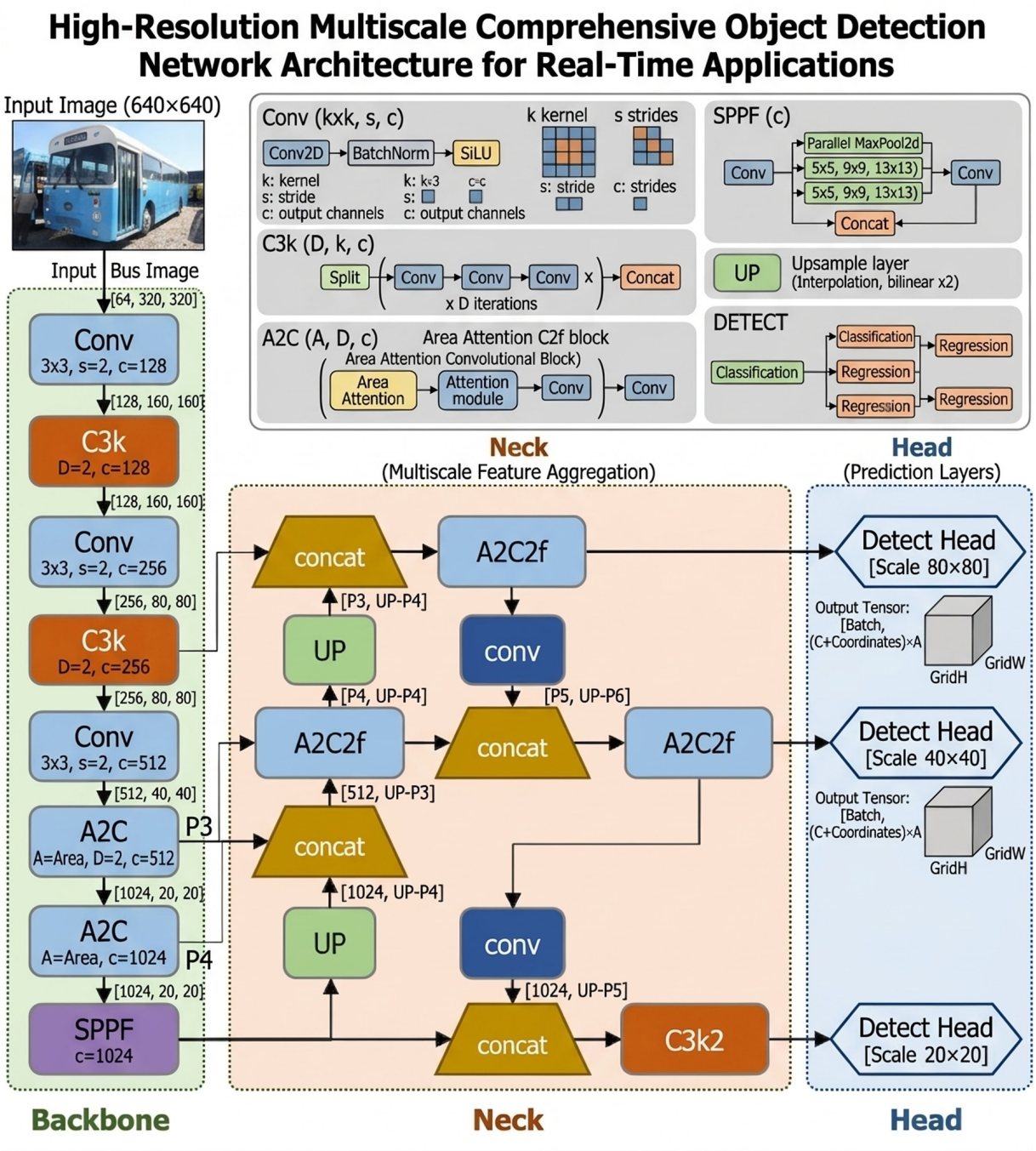


Figure 13. Conceptual overview of attention-based feature extraction and multiscale prediction in YOLOv12. The diagram emphasizes Area Attention, feature aggregation, and the organization of backbone, neck, and detection components. For the illustrated 640 × 640 input, the prediction scales are 80 × 80, 40 × 40, and 20 × 20. Area Attention models spatial relationships within feature-map regions, while multiscale fusion combines information across resolutions to support detection of objects with different sizes [28].

Under the source protocol, YOLOv12-N achieves 40.6 AP at 1.64 ms on an NVIDIA T4, corresponding to increases of 2.1 AP points over YOLOv10-N and 1.2 AP points over YOLO11-N [28]. These are source-reported comparisons under the evaluation configuration used by the YOLOv12 study.

Furthermore, YOLOv12 retains compatibility with multi-task outputs such as segmentation, classification, keypoint detection, and oriented bounding boxes. A comparative agricultural application study also showed that YOLOv12n achieved superior precision, recall, and mAP@0.5 relative to YOLO11 and v10 when trained on LLM-generated synthetic data for apple detection [60].

Consequently, YOLOv12 represents one of the earliest YOLO variants to incorporate attention mechanisms as a central architectural component. Its design may indicate a broader trend toward increasingly attention-oriented real-time detection architectures.

3.6. YOLOv13: Hypergraph-Enhanced Adaptive Visual Perception

In June 2025, YOLOv13 introduced a distinct architectural direction by extending conventional pairwise correlation modeling toward hypergraph-based high-order relationship learning through hypergraph computation [15]. While prior YOLO architectures captured local patterns or pairwise pixel dependencies, they lacked the capacity to model global high-order correlations among multiple spatial locations—a limitation that is particularly relevant in scenes with densely clustered objects, severe occlusion, and substantial variation in object scale. To address this limitation, YOLOv13 introduces three principal innovations: the Hypergraph-based Adaptive Correlation Enhancement (HyperACE) mechanism, the Full-Pipeline Aggregation-and-Distribution (FullPAD) paradigm, and depthwise-separable (DS) feature-extraction blocks.

One of the defining characteristics of YOLOv13 is its use of hypergraph-enhanced correlation modeling (Figure 14) [15]. HyperACE uses learnable hyperedges to model multi-way relationships across spatial locations and scales, while FullPAD distributes the resulting correlation-enhanced features through backbone-to-neck, intra-neck, and neck-to-head pathways. Depthwise-separable blocks replace large-kernel conventional blocks, reducing YOLOv13-N from 3.1 M to 2.5 M parameters and from 7.9 to 6.4 GFLOPs while retaining 41.6 AP in the source ablation. On COCO val2017, YOLOv13-N reports 41.6 AP at 1.97 ms on an NVIDIA T4 with TensorRT FP16, an increase of 3.0 AP points over YOLO11-N and 1.5 AP points over YOLOv12-N under the comparison reported in [15]. For YOLOv13-S, the complete HyperACE–FullPAD configuration reaches 48.0 AP; disabling hypergraph-enhanced distribution yields 47.1 AP, whereas configurations retaining only one FullPAD pathway achieve 47.6–47.8 AP. It should be noted that these findings originate from the primary release paper and therefore warrant further independent validation. An independent uniform-protocol study found that YOLOv13 did not uniformly outperform earlier versions on downstream tasks [41]; therefore, the maturity caveat in Section 7.10 applies.

3.7. YOLO26: End-to-End NMS-Free Detection for Edge Deployment

YOLO26 was unveiled at YOLO Vision 2025 in London and was subsequently documented in the official 2026 technical report by Jocher et al. [16]. This review treats [16] as the primary source for the architecture and benchmarks, while the analyses by Sapkota et al. [45], Sapkota and Karkee [46], and Chakrabarty [43] are used as complementary secondary evidence. Because YOLO26 is recent, independent uniform-protocol evaluations remain limited, and early domain-specific findings indicate that its advantages do not transfer uniformly across tasks [42]. YOLO26 places particular emphasis on architectural simplification and deployment-oriented training optimization. It provides an NMS-free

inference path, removes Distribution Focal Loss from box regression, and introduces Progressive Loss, Small-Target-Aware Label Assignment, and the MuSGD optimizer.

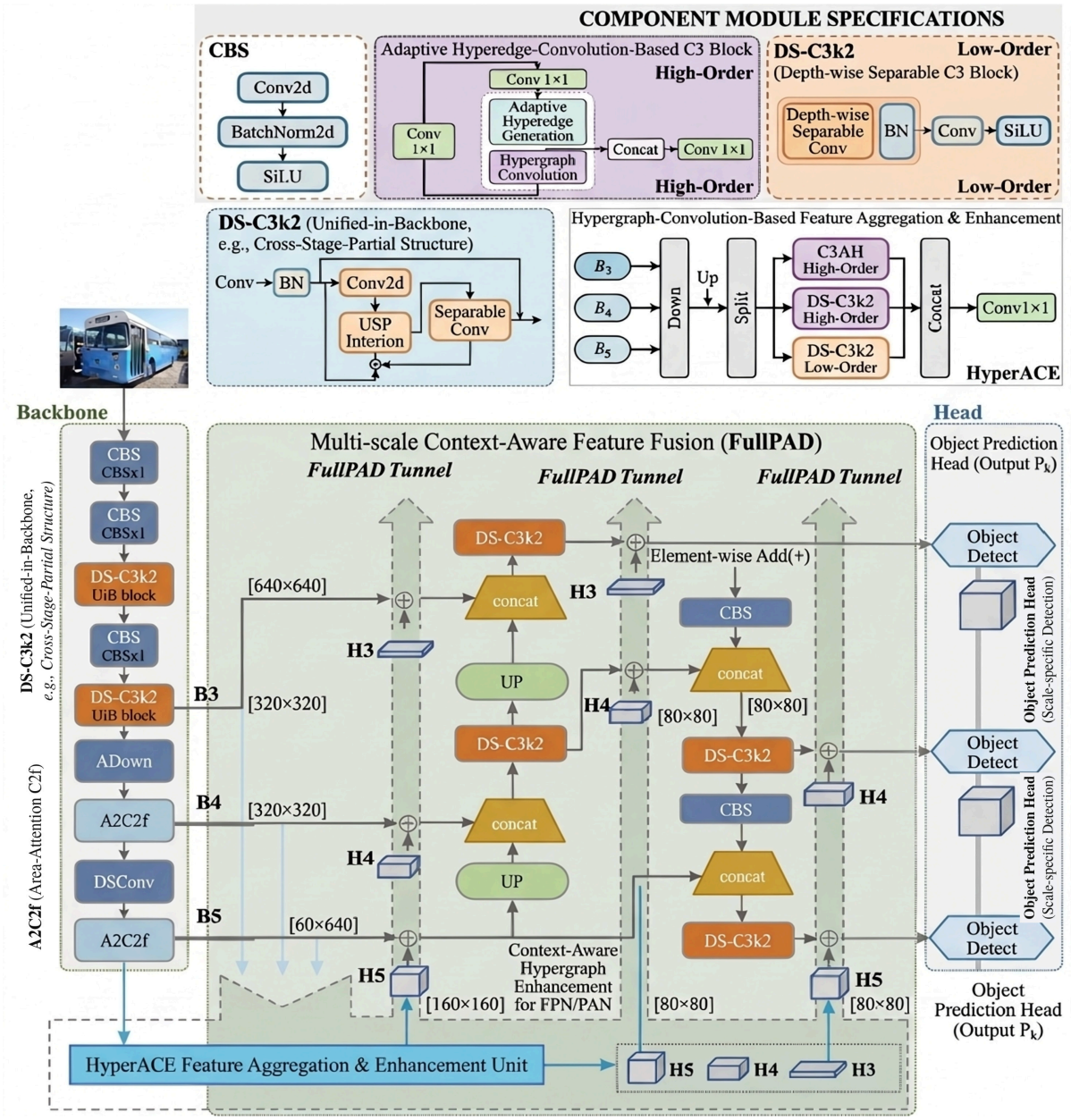


Figure 14. YOLOv13 architecture: hypergraph-enhanced adaptive visual perception for modeling high-order correlations beyond pairwise attention. HyperACE constructs learnable hyperedges that connect multiple visual elements, and FullPAD distributes the enhanced features through backbone-to-neck, intra-neck, and neck-to-head pathways. Depthwise-separable blocks reduce YOLOv13-N from 3.1 M to 2.5 M parameters and from 7.9 to 6.4 GFLOPs while retaining 41.6 AP in the source ablation. ⊕ denotes element-wise addition, and ⊙ denotes element-wise multiplication. Adapted from Lei et al. [15].

A central design objective of YOLO26 is the realization of NMS-free detection (Figure 15). Conventional dense detectors generate overlapping predictions that are filtered by NMS, adding a data-dependent post-processing stage and an additional IoU threshold. Instead, YOLO26 employs a dual-branch training strategy that couples a one-to-many auxiliary branch with a one-to-one branch based on Task-Aligned Assignment. At inference, the one-to-one branch can be used without NMS, simplifying the deployed graph. The primary

report gives 38.9 ms CPU ONNX latency for YOLO26-N at 640 × 640 and reports up to 43% higher CPU throughput than YOLO11-N under its Intel Xeon evaluation protocol [16]. This result is source-specific and does not establish frame-to-frame determinism, safety certification, or performance on untested edge devices.

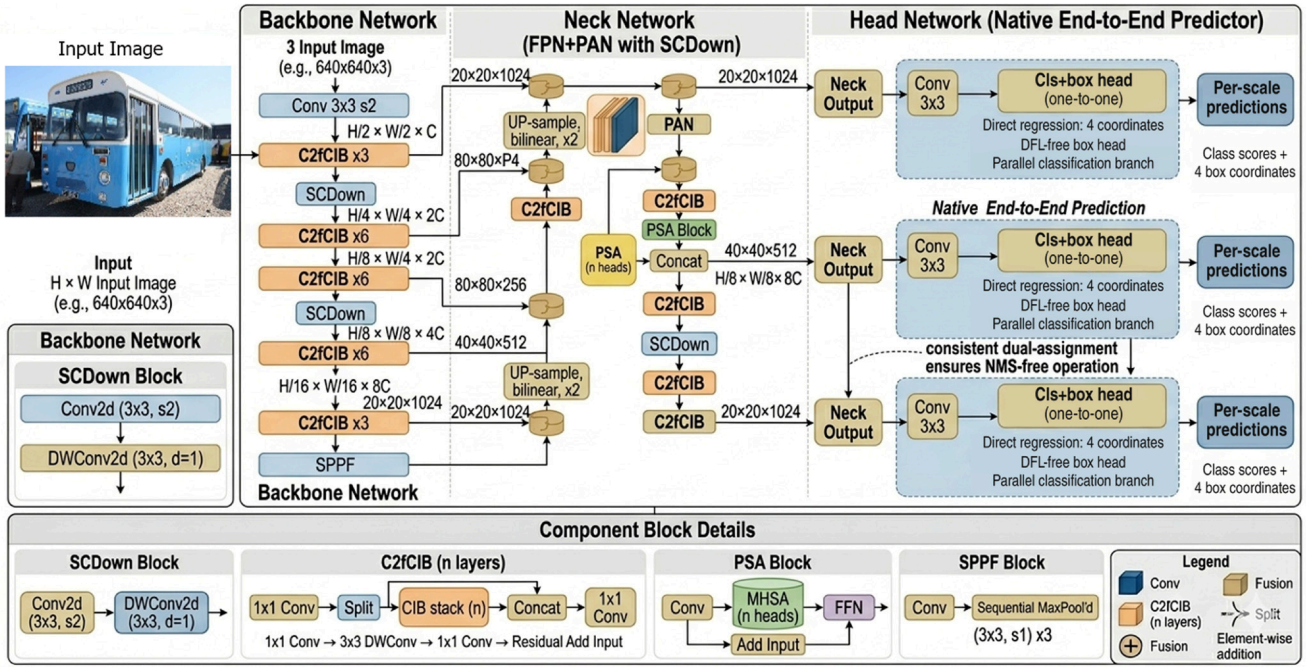


Figure 15. Conceptual overview of YOLO26 detection, emphasizing feature extraction, multiscale fusion, and end-to-end prediction. The three prediction panels highlight classification and direct box regression within the one-to-one branch, producing class scores and four box coordinates at each location. Removing Distribution Focal Loss (DFL) simplifies the regression output representation, while one-to-one prediction supports NMS-free inference. The accompanying module insets illustrate convolution, feature aggregation, and attention-related processing discussed in the main text [16].

YOLO26 removes Distribution Focal Loss and uses direct box regression with L1 supervision [16]. In the controlled YOLO26-S ablation, removing DFL changes AP from 46.0 to 46.3 at 640-pixel input and from 49.8 to 51.1 at 1280-pixel input. These effects are specific to the surrounding training configuration and should not be generalized as universal gains. Progressive Loss gradually shifts supervision emphasis from the one-to-many branch to the inference-time one-to-one branch. The best schedule changes the branch weights from (0.8, 0.2) at the start of training to (0.1, 0.9) at the end, increasing end-to-end AP from 46.4 to 46.7. STAL with s_{ref} = 16 increases overall AP from 46.6 to 46.8 and AP_S from 29.0 to 29.6. MuSGD combines Muon-style orthogonalized updates with SGD for multidimensional parameters and uses SGD for one-dimensional parameters; in the reported comparison, it reaches 47.4 AP after 500 epochs, compared with 47.0 AP after 600 epochs for SGD. These are source-specific ablation results [16].

Source-reported benchmarks characterize YOLO26's deployment-oriented design on COCO val2017. The main COCO results table of the YOLO26 report [16] lists 53.1 AP for the standard one-to-many head and 52.5 AP for the E2E one-to-one head of YOLO26-M, alongside a released-model T4 TensorRT10 time of 4.7 ms. Across N/S/M/L/X, standard-head AP ranges from 40.9 to 57.5, whereas E2E AP ranges from 40.1 to 56.9; reported T4 times range from 1.7 to 11.8 ms. These two AP columns should not be treated as interchangeable, and the source does not provide separate speed columns for both heads. YOLOv12-M reports 52.5 AP and 4.86 ms under the T4 TensorRT FP16 protocol of its own paper [28]; this is a separate source-specific operating point. YOLO26 supports detection,

instance segmentation, pose estimation, oriented bounding boxes and classification through task-specific heads and exposes export paths for multiple deployment backends [16]. The primary report does not provide a controlled INT8 accuracy study, frame-time variance measurements, or Jetson/mobile-device benchmarks; such claims therefore require separate platform-specific evidence.

3.8. Summary and Comparative Trends

Over nearly a decade of architectural development (2016–2025), with the formal literature base extended through 2026 [16], YOLO's architectural philosophy has progressed through distinct paradigm shifts:

- From purely convolutional, grid-based regression (v1, v2)
- toward deeper backbones and multi-scale fusion (v3–v5)
- to efficiency, decoupling, and modular design (v6–v8)
- toward gradient-aware and attention-augmented structures (v9–v11)
- through attention-centric architectures (v12)
- advancing to hypergraph-enhanced high-order correlation modeling (v13)
- and ultimately maturing toward deployment-centric simplification with end-to-end NMS-free inference (YOLO26)

This trajectory reflects evolving design priorities rather than a strictly monotonic progression in performance. Early versions emphasized deeper backbones and multi-scale representation; middle generations prioritized modularity, reparameterization, and anchor-free heads; recent releases explored programmable gradients, attention, hypergraph computation, and end-to-end NMS-free deployment. Source-reported comparisons show continuing improvements in particular accuracy or latency measures, but the magnitude and ordering depend on the dataset, training recipe, hardware, runtime, and model scale. Consequently, detection accuracy and deployment efficiency should be viewed as jointly optimized and evaluation-dependent objectives rather than universally co-improving properties.

From its inception, the YOLO family has exemplified a consistent trade-off between structural simplicity and detection precision.

- The early versions (v1–v2) established YOLO's regression-based paradigm, eliminating proposal stages and introducing anchors for spatial flexibility.
- The mid-generation models (v3–v5) introduced multi-scale fusion and CSP/PANet architectures, closing the performance gap with heavier detectors like Faster R-CNN.
- Subsequent releases (v6–v8) emphasized industrial efficiency, modularization, and anchor-free decoupled heads, bringing YOLO into mobile and embedded domains.
- Across YOLOv9–YOLO11, the principal developments include programmable gradient information, consistent dual assignments for end-to-end detection, and partial self-attention [29–31].
- With YOLOv12, the architecture was redefined as an attention-centric yet real-time framework, leveraging Area Attention and R-ELAN to capture global context efficiently.
- Finally, the 2025 releases bifurcated the lineage: YOLOv13 extended correlation modeling to hypergraph-based high-order reasoning, while YOLO26 pivoted toward deployment-centric simplification with end-to-end, NMS-free inference.

Together, these iterations illustrate a decade-long evolution from manually designed convolutional stacks to adaptive, modular detectors that integrate attention mechanisms —maintaining YOLO's identity as the benchmark for real-time, high-efficiency object detection.

3.9. Derived and Extended YOLO Models

Starting from the release of YOLOv5, the YOLO ecosystem diversified dramatically. A growing number of research groups and industrial laboratories began developing derivative architectures—collectively referred to as extended YOLO models—that built upon the foundational design principles of YOLO while introducing specialized modules to enhance performance, scalability, or adaptability to particular environments.

This section reviews the most influential variants, organized by generation, summarizing their architectural contributions, key characteristics, benchmark results, and broader significance to the evolution of real-time object detection.

A. YOLOv5 Variants

YOLOv5 (Ultralytics, 2020–2022) [27] established the modular foundation for YOLO derivatives, offering five compound-scaled configurations (n/s/m/l/x) and high-resolution extensions (P6, P7) up to 1280 × 1280. It standardized PyTorch-based implementation, AMP training, hyperparameter evolution, and multi-export formats (ONNX, TensorRT, CoreML). YOLOv5x achieves 50.7% mAP@[0.5:0.95] on COCO with sub-13 ms latency on a V100 GPU (batch 1), surpassing YOLOv4 [21] (43.5%) at comparable throughput, and became the de facto platform for subsequent research variants.

B. YOLOX (2021): Anchor-Free Reformulation

YOLOX (Megvii AI Lab) [23] introduced a major community-driven anchor-free reformulation following FCOS-style point regression [61]. Its main innovations include a decoupled head separating classification and localization, SimOTA dynamic label assignment, and strong data augmentation using Mosaic, MixUp [62], and Copy-Paste [63]. YOLOX-L reports 50.0 AP on COCO at 68 FPS on an NVIDIA V100 under its source protocol [23]. Anchor-free prediction and decoupled classification and localization branches also appear in later YOLO releases, including YOLOv8 [22]; however, the cited sources do not establish a direct causal lineage.

C. YOLOR (2021): Unified Representation Learning

YOLOR [64] combines explicit task-specific features with implicit representations within a shared CSP-based detection framework. This formulation supports several detection-related subtasks within a shared representation. YOLOR-P6 reports 52.6 AP on COCO test-dev in the original paper [64]. Comparisons with YOLOv5-P6 remain source-specific, and the cited sources do not establish that YOLOR directly influenced YOLO11 or YOLO26. Transformer-based variants such as YOLOS [65] reformulate detection as sequence prediction using transformer blocks built on layer normalization [66], in contrast to the batch-normalized convolutional stacks of the YOLO lineage, showing that Vision Transformers can perform object detection without a conventional YOLO-style convolutional detector. Lightweight backbones, including MobileNetV2 [67], MobileNetV3 [68], ShuffleNetV2 [69], and GhostNet [70], have also been integrated into YOLO-based detectors for mobile and embedded applications. These systems are discussed as related architectural directions rather than as direct descendants of YOLOR.

D. PP-YOLO and PP-YOLOE (2020–2022): Industrial-Grade Optimization

The PP-YOLO family emphasizes production-oriented detector development. PP-YOLOE [24] uses CSPRepResNet-based backbone and neck components, an efficient task-aligned head, and task-aligned learning. PP-YOLOE-L reports 51.4 AP on COCO test-dev and 78.1 FPS on a Tesla V100 under the source protocol. These source-reported results are not a hardware-normalized comparison with other YOLO releases.

E. Scaled-YOLOv4 and YOLOv4-CSP (2021): High-Resolution and Cross-Stage Extensions

Scaled-YOLOv4 [71] applies compound scaling across depth, width, and input resolution, following the broader scaling principles introduced by EfficientNet [72] and refined in

EfficientNetV2 [73]. Its P5, P6, and P7 variants use CSP-based residual layers and PANet-style multi-scale aggregation. Scaled-YOLOv4-P7 reports 55.5 AP on COCO test-dev at 1536-pixel input without test-time augmentation in the paper [71]. A direct latency advantage over EfficientDet is not inferred because the available values do not establish a matched hardware and runtime comparison. Related scaling strategies appear in later YOLO families, but the available sources support a design parallel rather than a direct causal claim.

F. YOLO-NAS (2023): AutoML and Neural Architecture Search

Neural architecture search has also been formulated using reinforcement learning [74]; this broader research direction should be distinguished from the proprietary AutoNAC procedure used to develop YOLO-NAS. YOLO-NAS (Deci AI) [75] was generated using the proprietary AutoNAC neural architecture search engine, which optimizes a model for specified task, data, hardware, and latency objectives. The official documentation describes quantization-aware RepVGG-style blocks and selective quantization intended to reduce accuracy degradation after INT8 conversion. On COCO val2017 at 640 × 640, the official model zoo reports 47.5, 51.55, and 52.22 AP for the S, M, and L variants, respectively [75]. Because the detailed search procedure is proprietary and the cited documentation does not establish a causal relationship with YOLO11, this review does not attribute specific gradient/evolutionary search mechanisms, cross-source FLOP reductions, or direct downstream architectural influence to YOLO-NAS.

G. YOLOv7 E6E and YOLOv8-Pose/Seg (2023): Task-Specific Extensions

Later variants expanded YOLO into multi-task learning. YOLOv7-E6E [26] reports 56.8 AP on COCO val2017 at 1280-pixel input, with 151.7 M parameters and 843.2 GFLOPs. The cited V100 benchmark reports 36 FPS at batch size 1; the corresponding inference time is approximately 27.78 ms. Concurrently, YOLOv8-Pose and YOLOv8-Seg [22] add heads for human-pose estimation and instance segmentation atop a shared CSP-PANet backbone, transitioning YOLO from a pure detector to a multi-task perception framework for robotics, autonomous systems, and AR applications.

H. Comparative Impact of Derived Variants and Summary of the Section

Table S1 summarizes selected YOLO derivatives and related real-time detectors in terms of architectural purpose, major mechanisms, and evidence-bounded significance. Table S2 lists source-reported operating points for selected derivative–baseline pairs. A numerical delta is not interpreted as a controlled gain when the training recipe, model scale, input resolution, evaluation split, runtime, or hardware differs. Similar mechanisms, such as anchor-free decoupled heads and architecture-search objectives, later appear in several official releases, but the available sources do not by themselves establish direct architectural influence.

Broadly, these derived models can be grouped by their contributions:

- Algorithmic Refinement: YOLOX (anchor-free), YOLOR (representation unification)
- Industrial Optimization: PP-YOLO, PP-YOLOE
- Scalable Design: Scaled-YOLOv4, YOLOv5 variants
- Automated Architecture Search: YOLO-NAS
- Task Diversification: YOLOv7-E6E, YOLOv8-Pose/Seg

These variants expanded the YOLO ecosystem across architectural refinement, industrial optimization, automated architecture search, high-resolution scaling, task diversification, and open-vocabulary detection. Several design principles recur across derivative and official families, including anchor-free prediction, decoupled heads, scalable model families, and context-enhancement modules. Recurrence does not by itself demonstrate direct adoption, and the review therefore treats these relationships as conceptual parallels unless a primary source explicitly documents lineage.

The derivative ecosystem is best understood as a set of source-reported operating points rather than a single ordered leaderboard. Academic, industrial, and AutoML-derived systems pursue different objectives, training protocols, input resolutions, hardware targets, and task definitions. Accordingly, model selection should be based on controlled application-specific evaluation rather than on cross-source percentage gains or approximate FPS values.

4. Key Architectural Innovations

Whereas Section 3 examined the chronological evolution of YOLO architectures, this section focuses on the key design innovations that have shaped the development of the YOLO family. Successive YOLO generations have contributed not only new architectural components but also evolving principles for balancing computational efficiency, representational capacity, and real-time performance. The following sections analyze these core innovations in detail—spanning the evolution of backbones, neck and head designs, optimization objectives, activation and normalization functions, and advanced training methodologies.

4.1. Backbone Evolution

The backbone network functions as the primary feature-extraction component, influencing both representational quality and computational complexity. Across YOLO's fourteen generations (2016–2025), backbone architectures have evolved from simple convolutional stacks to highly modular, gradient-aware, and attention-enhanced structures. Figure 16 visualizes this progression alongside the parallel evolution of neck and head modules. The backbone track highlights convolutional feature extractors, including GELAN in YOLOv9 and DS blocks in YOLOv13; PGI is identified as a training mechanism. The neck and feature-fusion track includes FPN, PAN-style aggregation, and YOLOv13′s HyperACE and FullPAD mechanisms. The head track, in turn, shifts from grid-based and anchor-based designs through decoupled and anchor-free heads to NMS-free end-to-end detection, culminating in YOLO26′s dual-head training with STAL. The selected developments in Figure 16 organize the module-level analysis in the following subsections; the arrows summarize design progression without establishing direct architectural inheritance.

The earliest generations (YOLOv1–v2 [8,19]) relied on lightweight convolutional architectures derived from the Darknet family. Darknet-19 comprised 19 convolutional layers and 5 pooling layers, designed for rapid inference but lacking depth for robust small-object detection. YOLOv3 [20] introduced Darknet-53, integrating ResNet-style residual connections [48] to enhance gradient propagation and feature reuse, enabling deeper networks and a substantial accuracy gain over YOLOv2.

YOLOv4 [21] adopted CSPDarknet53 [50], using Cross-Stage Partial connections to improve information and gradient flow while reducing redundant computation. YOLOv7′s E-ELAN [26] further developed multi-path feature aggregation, while YOLOv9 combined GELAN with Programmable Gradient Information to manage information and gradient propagation in deeper networks [29]. More recent generations have adopted a diverse range of mechanisms for enhancing contextual representation: YOLO11 adds a partial C2PSA attention branch [31], YOLOv12 uses region-partitioned Area Attention with an efficient attention implementation [28,76,77], and YOLOv13 introduces adaptive hypergraph computation [15]. Related channel and spatial attention mechanisms include SE [78], concurrent spatial and channel squeeze-and-excitation [79], and CBAM [80]. Because these modules were evaluated on different architectures, datasets, training schedules, and hardware, their gains and latency costs should remain attached to the individual source experiments rather than summarized as universal percentage ranges.

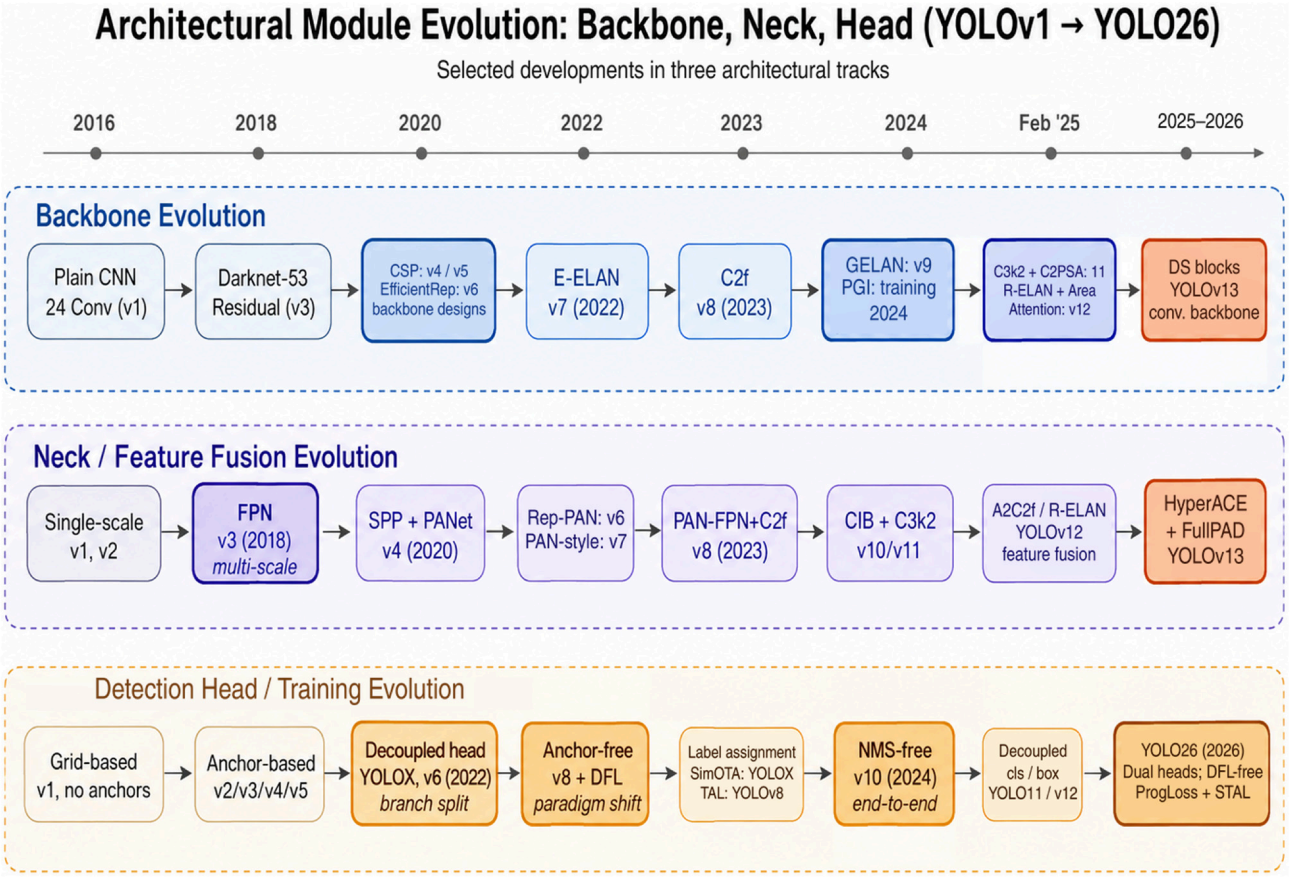


Figure 16. Selected architectural and training developments across YOLO generations from 2016 to 2026. Three horizontal tracks organize changes in backbone feature extraction, neck and feature fusion, and detection heads and training strategies. Later developments include HyperACE and FullPAD in YOLOv13 [15], and DFL-free regression and dual-head training in YOLO26 [16]. Arrows organize representative developments without establishing direct architectural inheritance; training mechanisms such as PGI and label assignment are distinguished from inference modules.

The most recent generations pursue distinct optimization priorities. YOLOv13 [15] introduces HyperACE and FullPAD and reports a 3.0-AP-point increase over YOLO11-N under the comparison in its primary paper. These mechanisms are motivated by the need to model higher-order relationships, but a shared cross-model benchmark for dense clustering, severe occlusion, or domain robustness has not been reported. YOLO26 [16] instead simplifies deployment by removing DFL and providing a one-to-one NMS-free inference path, together with Progressive Loss, STAL, and MuSGD. These architectural differences define candidates for further evaluation; they do not establish universal accuracy-first and deployment-first winners across application domains.

4.2. Neck and Head Optimization

The neck and head modules constitute the interface between feature extraction and prediction. They determine how multi-scale information is fused and how bounding boxes and class probabilities are predicted. Throughout the evolution of the YOLO lineage, these modules progressed from relatively simple feature-fusion mechanisms to increasingly sophisticated, decoupled, and attention-aware prediction architectures.

The early YOLOv1–v2 [8,19] models had no explicit neck, connecting high-level features directly to output layers. YOLOv3 [20] introduced Feature Pyramid Networks (FPN) [49] for multi-scale detection, while YOLOv4 [21] extended this with Path Aggregation Networks (PANet) [52] and Spatial Pyramid Pooling (SPP) [51] for enhanced cross-scale

context. From YOLOv5 [27] onward, PANet-style necks became standard, with compound-scaled variants (n/s/m/l/x) trading accuracy for speed. YOLOv7′s E-ELAN [26] provided improved connection patterns enabling longer gradient paths and denser feature reuse.

Meanwhile, the head design shifted from a monolithic, anchor-based formulation to an anchor-free, decoupled paradigm. Early YOLO generations relied on predefined anchor boxes for bounding-box regression, but these anchors often mismatched object scales and required dataset-specific tuning. YOLOv8 removed anchors entirely, predicting object centers and box dimensions directly via point regression in the spirit of CenterNet [35]. This design decoupled the classification and localization branches, improving training convergence and accuracy on small or irregularly shaped objects.

Recent architectures modify both feature processing and prediction. YOLOv9 [29] combines GELAN feature aggregation with anchor-free prediction; YOLO11 [31] uses C2PSA attention, and YOLOv12 [28] uses Area Attention in feature-processing modules. YOLOv10 [30] and YOLO26 [16] enable NMS-free end-to-end inference through one-to-one assignment. Removing NMS simplifies the deployed graph and avoids data-dependent post-processing, but it does not by itself establish deterministic end-to-end runtime; frame-time variability remains dependent on the hardware, runtime, load, and input pipeline. Collectively, these developments improve prediction flexibility while supporting a broader range of vision tasks (segmentation, keypoint estimation, oriented bounding boxes).

4.3. Loss Functions and Optimization Strategies

The design of optimization objectives has played a central role in the evolution of YOLO. YOLOv1 uses a weighted sum-of-squared-errors objective for coordinates, confidence, and class probabilities, with square-root parameterization of box width and height [8]. This formulation differs from the IoU-based localization losses used in later versions.

To address this, YOLOv2 [19] and YOLOv3 [20] introduced loss normalization and class-balanced weighting. A major transition occurred with the introduction of YOLOv4 [21], which progressed through Distance-IoU (DIoU) [81] and ultimately adopted Complete-IoU (CIoU) loss [54]—a distance-aware function jointly optimizing overlap area, center distance, and aspect-ratio consistency, providing faster convergence and higher localization accuracy.

Later generations refined this further: EIoU [82] simplified computation while maintaining performance; YOLOv9 introduced Programmable Gradient Information (PGI), which preserves reliable gradient signals through auxiliary reversible branches rather than defining a separate position-versus-shape learning-rate schedule [29]; and YOLO26 introduced Progressive Loss, which gradually shifts supervision from the dense one-to-many branch toward the inference-time one-to-one branch, improving end-to-end AP from 46.4 to 46.7 in the source ablation [16]. Optimization strategies also evolved—starting from YOLOv4 [21], training “bags of freebies” (cosine scheduling, label smoothing, the Adam optimizer [83] and its weight-decay-decoupled successor AdamW [84], mixed precision) improved stability without inference penalties, formalized into YOLOv7′s trainable bag-of-freebies [26]. Across successive YOLO generations, the evolution of optimization objectives reflects a consistent effort to improve localization quality and training stability: MSE (YOLOv1–v2) provided baseline regression but suffered from scale sensitivity, and IoU-family losses (GIoU [81], DIoU, CIoU [54]) progressively incorporated overlap-area, center-distance, and aspect-ratio consistency. Because these objectives were evaluated on different architectures, datasets, and training schedules, their reported effects should not be accumulated into a single cross-generation gain; source-specific numerical effects are

summarized in Section 7.6. Gradient-aware objectives (PGI [29], ProgLoss [16]) mark the field's shift from purely geometric alignment toward optimization-aware objective design.

4.4. Activation and Normalization Functions

Activation and normalization mechanisms represent important micro-architectural design elements that influence gradient propagation and representational capacity. Early YOLO models (v1–v2) used Leaky ReLU, chosen for its simplicity and computational speed. However, its limited dynamic range constrained representational capacity.

Activation choices in the YOLO lineage have evolved alongside backbone design but remain implementation-specific rather than generation-wide rules. Early Darknet-based models predominantly used Leaky ReLU; YOLOv4 adopted Mish in its CSPDarknet implementation; and SiLU became common in later PyTorch-based CNN backbones [85]. Attention-containing models may also use GELU within transformer-like submodules [86], but this does not imply that the entire detector family has replaced SiLU with GELU. The YOLO26 technical report does not attribute a controlled INT8-quantization result to its activation function, so no quantization-stability advantage is claimed here [16].

Batch Normalization continues to serve as the dominant normalization strategy across most convolutional YOLO backbones. Some implementations modify normalization placement, fusion, or training behavior, but the resulting effects should be reported only when a primary source provides a controlled ablation. Accordingly, this review treats activation and normalization as implementation-level design choices and avoids assigning universal AP gains across heterogeneous model families.

4.5. Training and Data Augmentation Techniques

Training methodology has become an increasingly important component of YOLO-based detector development. Early YOLO generations relied primarily on conventional geometric data augmentation techniques (flipping, scaling, color jitter), but from YOLOv4 onward, increasingly sophisticated data-centric strategies emerged to improve robustness and generalization across diverse deployment conditions.

Mosaic augmentation, introduced in YOLOv4, randomly combines four images into one during training, encouraging scale invariance and diverse context learning. CutMix [87] and MixUp [62] further blended image patches and pixel intensities, augmenting the network's ability to separate overlapping instances. YOLOv5 formalized auto-augmentation pipelines [88] and added self-distillation mechanisms, where teacher and student networks shared parameters for progressive refinement.

More recent YOLO generations have incorporated increasingly specialized optimization and label-assignment strategies. YOLO26 introduces MuSGD and STAL; in source-specific ablations, MuSGD reaches 47.4 AP after 500 epochs compared with 47.0 AP after 600 epochs for SGD, while STAL increases AP_S from 29.0 to 29.6 [16]. Mosaic, MixUp [62], CutMix [87], and Copy-Paste [63] are widely used augmentation methods, but their effects depend on the model, task, dataset, schedule, and baseline. Their contributions should therefore be reported only from controlled source-specific ablations and should not be accumulated into a universal +4.0–5.5 AP gain.

5. Evolutionary Design Principles of the YOLO Lineage: A Why–How–What–Where Analysis

Sections 3 and 4 examined the YOLO lineage from two complementary perspectives: a chronological account of each release and a thematic dissection of its component-level innovations. This section adopts a third, interpretive perspective. Rather than asking what changed, it parses the YOLOv1-to-YOLO26 trajectory into cause-and-effect chains and asks

five questions that jointly define the lineage's evolutionary logic. The first three concern drivers and outcomes: (1) why did each generation evolve—that is, which limitations, challenges, or emerging application requirements triggered its principal architectural changes; (2) how did successive generations respond—through which design strategies, mechanisms, and innovations; and (3) what, at the level of design DNA, characterizes each generation's architectural profile. The remaining two concern location and motivation: (4) where in the architecture—backbone, neck, head, feature representation, loss, label assignment, training strategy, or inference mechanism—was redesign effort repeatedly concentrated; and (5) why did these particular components, rather than others, become the persistent loci of innovation.

The analysis synthesizes the evidence already compiled in Table 1, Figure 16, and the benchmark synthesis of Section 7, but the framing and the conclusions drawn here constitute the authors' interpretive judgment rather than measured results; wherever a claim rests on source-reported numbers, the corresponding primary references are cited.

5.1. Why: Recurring Bottlenecks as Evolutionary Drivers

Most major transitions within the YOLO lineage appear to be associated with specific technical limitations identified in preceding generations. Table 2 summarizes these evolutionary relationships, pairing each transition with the limitation that triggered it, the design response that resolved it, and the source-reported net effect.

Table 2. Bottleneck-response reconstruction of the YOLO lineage. Each row pairs a generational transition with the exposed limitation of the predecessor, the principal design response, and the source-reported net effect. Effects are reported values from the cited primary sources under their respective evaluation conditions and do not represent paired, same-protocol before-and-after comparisons (see Section 7.1).

Transition	Exposed Bottleneck of Predecessor	Principal Design Response	Source-Reported Net Effect
v1 → v2	Rigid 7 × 7 grid and two-box-per-cell limit cause low recall and localization error [8]	Anchor priors via k-means, batch normalization, high-resolution classifier [19,47]	mAP@0.5 63.4 → 76.8 (VOC07) [19]
v2 → v3	Single-scale prediction misses small objects [19]	Three-scale FPN-style head on residual Darknet-53 [20,48,49]	AP_{small} 5.0 → 18.3 [20]
v3 → v4/v5	Gradient redundancy inflates training cost; no production tooling [21]	CSP partitioning, bag-of-freebies/specials, PyTorch productization [21,27,50]	43.5% → 50.7% mAP@[0.5:0.95] [21,27]
v5 → v6/v7	Train-inference structural mismatch; industrial latency targets [25]	Reparameterization, decoupled heads, TAL, trainable freebies [25,26,56]	52.5% at 121 FPS (batch 32, v6-L) [25]
v7 → v8	Dataset-specific anchor tuning; multi-task demand [22]	Anchor-free decoupled head, DFL, task-aligned assignment [22]	53.9% (v8x); unified multi-task platform [22]
v8 → v9	Information bottleneck destabilizes deep-network training [29]	PGI auxiliary reversible branches, GELAN [29]	53.0% with improved training stability [29]

Table 2. *Cont.*

Transition	Exposed Bottleneck of Predecessor	Principal Design Response	Source-Reported Net Effect
v9 → v10	NMS post-processing adds latency and hinders end-to-end deployment [30]	Consistent dual assignments with a one-to-one inference path; NMS-free decoding [30]	YOLOv10-B reports 46% lower latency and 25% fewer parameters than YOLOv9-C at comparable AP [30]
v10 → v11/v12	Pure CNNs lack long-range context [31]	C2PSA hybrid attention; Area Attention with R-ELAN [28,31]	51.5/52.5 AP (M-scale, source-reported) [16,28,31]
v12 → v13	Pairwise attention cannot model multi-to-multi correlations [15]	HyperACE hypergraph computation; FullPAD distribution [15]	41.6% (N-scale), +3.0 over YOLO11-N [15]
v11 → YOLO26	DFL/NMS operators impede export and CPU deployment [16]	DFL removal, native end-to-end head, ProgLoss, STAL, MuSGD [16]	Up to 43% faster CPU inference [16]

Taken together, Table 2 suggests what may be interpreted as a recurring evolutionary pattern within the YOLO lineage: a pendulum dynamic between accuracy-driven complexification and deployment-driven simplification. Architectural enrichment and deployment-oriented simplification recur across the reviewed releases. Anchors, multi-scale fusion, gradient routing, attention, and hypergraph modeling address representation or optimization goals, whereas reparameterization, anchor-free prediction, NMS-free decoding, and DFL removal address different efficiency or deployment constraints. This pattern is an interpretive synthesis; the sources do not establish that each enrichment step caused latency variance, export incompatibility, or quantization failure in a subsequent release. Until 2024 the two phases alternated within a single release line. The 2025 bifurcation into YOLOv13 [15] and YOLO26 [16] is, on this reading, not an anomaly but the pendulum's two phases executing in parallel for the first time: a structural response to a user base whose accuracy-critical and deployment-critical segments could no longer be served by a single compromise architecture. Independent multi-domain evidence that newer versions do not uniformly dominate their predecessors once training protocols are equalized [41,44] reinforces this interpretation—once a lineage spans divergent objectives, recency ceases to imply superiority for any particular deployment.

5.2. How: Five Recurrent Design Strategies

Although the lineage spans fourteen generations and dozens of named modules, the mechanisms through which these bottlenecks were addressed can, in our analysis, be broadly grouped into five recurring design strategies.

First, improved representational capacity per unit of computational cost. The backbone progression—residual connections (YOLOv3 [20,48]), cross-stage partial partitioning (YOLOv4 [21,50]), extended and generalized layer aggregation (YOLOv7 [26], YOLOv9 [29]), partial self-attention (YOLO11 [31], YOLOv12 [28]), and hypergraph correlation (YOLOv13 [15])—raised expressiveness per FLOP rather than raw capacity. This is why parameter counts across nano-scale variants remained nearly flat (1.9 M–3.2 M) while COCO mAP@[0.5:0.95] rose from 28.0 to 41.6 over the same period (see Section 7.2).

Second, information-flow engineering. The neck evolution from FPN [49] through PANet [52] to GELAN cross-stage aggregation [29] and FullPAD tri-tunnel distribution [15], together with Programmable Gradient Information [29], treats the network not as a stack

of layers but as a routing problem for features and gradients—an explicit response to the information-bottleneck failure mode of deep detectors.

Third, alignment of assignment and objective with the evaluation target. Label assignment progressed from hand-tuned and k-means anchor priors [19] through dynamic schemes (SimOTA [23], Task Alignment Learning [25]) to consistent one-to-one assignment [16,30] with small-target-aware extensions (STAL [16]); losses progressed from MSE [8] through the IoU family [54,81,82] and DFL [22] to curriculum-style ProgLoss [16]. Each step narrowed the gap between what training optimizes and what deployment measures.

Fourth—and in our assessment one of the lineage's most distinctive methodological characteristics to detector design at large—the systematic separation of train-time complexity from inference-time simplicity. Structural reparameterization (YOLOv6 [25], YOLOv7 [26], via RepVGG-style fusion [56]), auxiliary supervision branches removed at inference (YOLOv9 [29]), dual one-to-many/one-to-one heads in which only the simple branch survives deployment (YOLOv10 [30], YOLO26 [16]), and trainable bag-of-freebies [26] all purchase accuracy with training-time computation while leaving the deployed graph untouched. These strategies can improve the observed accuracy–latency trade-off under a specified protocol by allocating additional computation to training rather than inference.

Fifth, hardware-aware architectural and deployment co-design. Quantization-aware operator selection in explicitly documented quantization studies, architecture search under target-specific latency objectives such as YOLO-NAS [75], I/O-aware attention implementations such as FlashAttention [76,77], and export-driven removal of graph-hostile stages such as DFL and NMS in YOLO26 [16] treat the compiler, accelerator, and export format as first-class constraints. YOLOv10 [30] contributes efficiency-oriented operators and NMS-free inference, but its primary report should not be cited as controlled quantization evidence.

5.3. What: Generation-Level Design DNA

Table 1 and Figure 16 already catalog the component-level attributes of every generation; rather than repeating them, Table 3 compresses each generation to its dominant design principle and marks which of seven component families (backbone, neck, head, loss, label assignment, training methodology, post-processing) it substantially redesigned. This compact representation makes the lineage's design DNA directly comparable across generations and provides the raw material for the locational analysis in Section 5.4.

5.4. Where: The Concentration of Redesign Effort

Aggregating the final column of Table 3 highlights the architectural components that have received the greatest redesign attention across the fourteen canonical generations: the backbone was substantially redesigned in eleven generations, the neck in nine, the head in eight, training methodology in seven, label assignment in five, the loss formulation in five, and post-processing in only two.

Frequency alone, however, does not necessarily reflect impact. Backbone redesigns were frequent but predominantly incremental, driven by efficiency—each wave raising mAP-per-FLOP rather than shifting the detection paradigm. By contrast, the most substantial conceptual transitions appear to be concentrated in the head, assignment, and post-processing: the anchor-free transition (YOLOv8 [22]), NMS-free decoding (YOLOv10 [30]), and end-to-end dual-head training (YOLO26 [16]). Post-processing was redesigned in two generations in the coding used for Table 3. These NMS-free paths remove a data-dependent post-processing stage, but any reduction in end-to-end latency variance or export failures requires direct measurement [16,30].

Table 3. Dominant design principle and substantially redesigned component families per canonical generation. Component codes: B = backbone, N = neck, H = head, L = loss, A = label assignment, T = training methodology, P = post-processing. Assignments reflect the authors' synthesis of the primary release papers cited in Sections 3 and 4.

Version (Year)	Dominant Design Principle	Redesigned Components
YOLOv1 (2016) [8]	Detection as unified single-pass regression	B, H, L
YOLOv2 (2017) [19]	Statistical priors replace unconstrained regression	B, H, A, T
YOLOv3 (2018) [20]	Scale as a first-class design dimension	B, N, H, L
YOLOv4 (2020) [21]	Modular efficiency plus train-time freebies	B, N, L, T
YOLOv5 (2020) [27]	Productization and scaling discipline	T
YOLOv6 (2022) [25]	Inference-first structural rewrite	B, N, H, A, T
YOLOv7 (2022) [26]	Gradient-path density at zero inference cost	B, N, T
YOLOv8 (2023) [22]	Anchor-free multi-task platform	B, N, H, L, A
YOLOv9 (2024) [29]	Information preservation in depth	B, N, T
YOLOv10 (2024) [30]	End-to-end decoding	H, A, P
YOLO11 (2024) [31]	Pragmatic hybrid attention	B, N
YOLOv12 (2025) [28]	Attention as core, not accessory	B, N, H
YOLOv13 (2025) [15]	High-order correlation modeling	B, N
YOLO26 (2025) [16]	Deployment-first reduction	H, L, A, T, P

The location of a redesign can be related to measurable outcomes only within a matched evaluation. Neck and assignment modifications are commonly assessed through scale-stratified metrics such as APS; for example, the controlled YOLO26 STAL ablation increases APS from 29.0 to 29.6 while improving overall AP from 46.6 to 46.8 [16]. Head and assignment redesigns are motivated by crowded-scene performance, and post-processing removal simplifies the deployed graph, although its effect on end-to-end latency variability remains platform-dependent [16,30]. Thus, the redesigned component often reflects the primary bottleneck targeted by the corresponding modification, whereas the magnitude of the improvement must be established by a source-specific ablation rather than a cross-generation numerical sequence.

5.5. Why These Components: The Bottlenecks Behind the Loci of Innovation

The concentration pattern described in Section 5.4 may not be coincidental; it follows from which component dominates each fundamental trade-off axis. Detection accuracy at a fixed budget is governed primarily by the representational efficiency of the backbone and neck, which is why every accuracy wave began there. The returns, however, diminished over time: marginal gains shrank from double digits in the early era [19,20] to 1–3 points in the attention and hypergraph era [15,31], pushing later accuracy efforts toward correlation modeling rather than raw capacity. Localization precision depends on both regression supervision and label assignment. DFL-based regression predicts 4 K

logits for four coordinates with K bins per coordinate, whereas direct regression predicts four coordinates. This difference does not establish a universal 3–4× head-computation or latency overhead. The YOLO26 report evaluates DFL-free regression together with L1 supervision and training-time changes under its own protocol [16,22].

Computational complexity concentrates in the backbone, which dominates the FLOPs budget across the lineage and was therefore the first target of every efficiency wave (CSP partitioning [50], depthwise-separable blocks [15]). Latency contributions concentrate in the head and post-processing, where NMS runtime varies with prediction density [16,30]. Memory pressure concentrates in the neck and attention modules, motivating bandwidth-aware designs (cascaded SPPF [31], FlashAttention [76,77], regional attention partitioning [28]). Robustness under occlusion and clutter is governed by context-modeling capacity, which explains why attention and hypergraph mechanisms—rather than deeper convolution—are motivated by these conditions; because the surveyed primary sources do not share a common corruption protocol, such motivations are treated as design hypotheses requiring controlled stress testing rather than measured robustness rankings (Section 7.3) [15,28,31]. Deployment efficiency, finally, is governed by the operator set of the head and loss: components whose operations map poorly onto accelerator kernels or export formats (softmax-over-bins in DFL, data-dependent control flow in NMS) were ultimately eliminated regardless of their accuracy merits [16].

Taken together, a broader interpretation emerges. The components most persistently redesigned are those that either sit at the interface between the learned representation and the task definition (head, label assignment, post-processing) or dominate a hard resource budget (backbone for computation, neck for memory). Interface components encode non-differentiable heuristics—anchor matching, confidence thresholding, duplicate suppression—that gradient-based training cannot repair from within, so progress there required discrete paradigm shifts; budget-dominating components, by contrast, admitted continuous refinement. From this perspective, the evolution of the YOLO lineage may be viewed as a gradual transition from manually engineered heuristics toward increasingly learned and differentiable optimization mechanisms, executed under a strict real-time budget.

5.6. Summary

The five analytical questions introduced at the beginning of this section can be summarized as follows. Why: architectural evolution has generally been driven by concrete bottlenecks identified in preceding generations, with the aggregate trajectory following a complexification–simplification pendulum that bifurcated into parallel branches in 2025. How: through five recurring strategies—representational enrichment per FLOP, information-flow engineering, assignment–objective alignment, train-time/inference-time separation, and hardware co-design. What: each generation can be characterized by a dominant design principle (Table 3). Where: redesign effort concentrated in the backbone by frequency but in the head, assignment, and post-processing by paradigm impact. Why these components: because they either govern a dominant resource budget or encode the non-differentiable task interface. These principles organize the domain-level analysis in Section 6 and the quantitative synthesis in Section 7, generating hypotheses about architectural suitability that require validation on target data and hardware.

6. Application Domains of YOLO Frameworks

The practical significance of the YOLO family is most clearly demonstrated in real-world deployment environments, where operational constraints are diverse and often demanding. Real-time perception systems in safety-critical vehicles demand determinism

and extremely low latency, factory inspection lines require high recall for fine defects under uniform backgrounds, and agricultural environments confront nonstationary illumination and occlusion. Hospitals and laboratories impose strict requirements on sensitivity, explainability, and data governance; embodied agents and surveillance networks operate under power, thermal, and bandwidth constraints; and consumer multimedia or AR/VR workloads force tight, user-perceived latency budgets on diverse hardware. Across these application contexts, YOLO variants have been adopted not as interchangeable detectors but as distinct design solutions tailored to domain-specific requirements. This section analyzes those mappings, emphasizing how requirements translate into architectural and algorithmic choices and, ultimately, into deployment practices that meet operational requirements.

6.1. Autonomous Driving and Intelligent Transportation

In autonomous-driving environments, perception systems are evaluated as end-to-end pipelines in which even modest increases in latency may significantly influence downstream decision-making and operational safety. Detection systems in autonomous driving are typically evaluated along two primary dimensions: detecting small, distant, and often occluded instances (e.g., pedestrians and traffic signage), and doing so with consistent latency under variable environmental conditions. Recent YOLO variants report substantial gains on the canonical benchmarks KITTI [89], nuScenes [90], Waymo Open [91], and BDD100K. Rep-YOLOv5 [92] achieves 96.1% mAP at 202 FPS through structural reparameterization; YOLO-APD [93] reaches 77.7% mAP@[0.5:0.95] with >96% pedestrian recall in cluttered road geometries; YOLO-SPD [94] attains 97.85% precision on KITTI via Coordinate Attention; and PV-YOLO [95] compresses YOLOv8 for embedded automotive ECUs.

Traffic sign detection presents distinct challenges—signs typically $<32 \times 32$ pixels, severe class imbalance, partial occlusion—addressed by YOLO-PPA [96] (93.2% mAP@0.5 on GTSDB), TSD-YOLO [97] (Space-to-Depth for TT100K/CCTSDB), YOLOv8-CE [98] (occlusion-robust Coordinate Attention), and YOLO-ADual [99] (mobile-optimized). For adverse conditions, multispectral RGB-thermal fusion with middle-fusion YOLOv4 achieves the optimal accuracy-latency trade-off on KAIST and FLIR ADAS datasets [100]. Signal detection pipelines are deployed on automotive-grade SoCs such as NVIDIA Drive Orin and Qualcomm Snapdragon Ride [3,10].

Across the surveyed studies, several recurring architectural preferences can be observed: decoupled and anchor-free heads for flexible localization [22,26,29]; YOLOX [23] under aspect-ratio diversity; NMS-free paths, as in YOLOv10 [30] and YOLO26 [16], for a simpler end-to-end deployment graph; and attention-augmented backbones, as in YOLO11 and YOLOv12 [28,31], for richer contextual representation. The 2025 releases offer complementary directions: YOLOv13 [15] is motivated by high-order correlation modeling in dense pedestrian clusters and heavy occlusion, whereas YOLO26 [16] emphasizes DFL-free regression, NMS-free inference, multi-backend export, and up to 43% faster source-reported CPU inference. These characteristics identify candidates for automotive evaluation but do not substitute for target-platform measurements of worst-case latency, thermal behavior, failure rates, and redundancy requirements.

Evaluation extends beyond aggregate AP to distance-aware recall, occlusion-conditioned breakdowns, and latency under load. Joint detection-tracking benchmarks such as MOT17 and MOT20 provide system-level metrics, and vehicle classification and tracking pipelines [101] exemplify this coupling in intelligent transportation. Despite substantial progress, challenges related to rare-event coverage, geographic and weather-related distribution shift, and vulnerability to physical or digital perturbations remain unresolved [102]. The magnitude of each degradation varies with the dataset, corruption protocol, sensor, model, and region; therefore, no universal 15–25% geographic penalty or 20–40%

long-tail penalty is inferred from the cited evidence. Evaluation should report subgroup- and condition-specific AP, recall, calibration, and failure rates, together with domain-randomization or adaptation results under a documented protocol.

6.2. Industrial and Manufacturing Inspection

Industrial inspection environments impose requirements that differ substantially from those of conventional consumer-vision applications: inspection lines seek near-zero miss rates for narrowly defined defect classes at high line speed, tolerating high-resolution (often 4 K+) inputs while respecting deterministic control constraints. PCB defect detection—where defects (missing holes, mouse bites, open circuits, shorts) appear as sub-32-pixel features—has advanced through PCB-YOLO [103] (Coordinate Attention with Weighted-IoU on HRIPCB), YOLO-RLC [104] (97.3% mAP@0.5 at 76.7 FPS, +4.1% over YOLOv5-S), MS-YOLO [105] (YOLOv10-based PCB defect detection with multiscale edge strengthening and improved SPDConv), and lightweight GhostNet-based variants for edge cameras [106]. Steel surface inspection on NEU-DET spans six defect categories: SS-YOLO [107] achieves 97% mAP@0.5 while cutting FLOPs 79.3%; YOLO-SDS [108], ELS-YOLO [109], and SSA-YOLO [110] add StarNet backbones, SCSA attention, and shared convolutional detection, respectively.

Welding defect detection is safety-critical in the automotive and shipbuilding industries: X-ray-based porosity/crack detection [111] and a Slim-Neck lightweight architecture achieving 98.8% precision and 99.2% recall on cup-rupture detection [112] enable closed-loop process control. Fabric inspection requires near-perfect recall under high intra-class variability—systematically reviewed across YOLO v1–v11 [113], with DCFE-YOLO [114] adding Dynamic Snake Convolution for elongated defects. Food-packaging inspection addresses rare defects at high production rates [115], while pharmaceutical-packaging studies report YOLO-based blister and fill-level inspection [116]. When such systems create, modify, retain, or transmit regulated electronic records, compliance considerations should be assessed separately under FDA 21 CFR Part 11 [117] and EU GMP Annex 11 for computerized systems [118].

Contemporary deployment candidates include PP-YOLOE [24] as an industrial baseline, P6/P7 variants [27,71] for high-resolution inspection, YOLOv10 [30] for NMS-free inference, and YOLO-NAS [75] where a quantization-aware architecture matches the target hardware. YOLO26 [16] adds a DFL-free head, NMS-free inference, multi-backend export, and improved source-reported CPU latency; however, its Progressive Loss and STAL results were established on COCO rather than on 100:1 industrial defect imbalance or micro-defect benchmarks. YOLOv13 [15] provides a candidate architecture for studying high-order relationships among distributed defect patterns. All of these choices require validation on the actual line camera, PLC timing budget, defect distribution, and export runtime.

One of the most persistent challenges is severe class imbalance with scarce positive samples; mitigation strategies include CAD-derived synthetic data, illumination-controlled acquisition, generative augmentation, and augmented negatives. Adapting models across production lines remains challenging, and self-supervised pretraining and few-shot learning are promising approaches.

6.3. Agriculture and Precision Farming

Agricultural perception systems operate under highly variable field conditions: illumination varies across hours and seasons, plant geometry produces pervasive occlusion, and computation must often run onboard drones or rovers with tight power budgets. Fruit detection has emerged as one of the most extensively studied agricultural applications of YOLO-based detectors: YOLOv5-RACF [119] integrated with ROS, LiDAR mapping,

and robotic grasping achieves 98.7% mAP@0.5 with ±2 mm diameter accuracy in 0.13 s; S-YOLO [120] optimizes greenhouse tomato detection; STRAW-YOLO [121] localizes strawberry picking-points via keypoint regression; and YOLO-Tomato [122] remains a foundational reference. A systematic review across YOLOv3–v9 [123] documents YOLOv9′s GELAN/PGI reducing parameters 15% versus YOLOv8x while improving AP 1.7%. Sapkota et al. demonstrated the use of YOLOv8 with shape-fitting techniques to detect and size immature green apples [124]. In a separate study, an improved YOLOv12 model trained on LLM-generated synthetic data outperformed YOLO11 and YOLOv10 in precision, recall, and mAP@0.5 [60].

Weed detection enables 70–90% herbicide reduction through selective spraying (YOLOWeeds [125], CottonWeeds [126], zero-shot UAV generalization across species [127]). Disease and pest monitoring increasingly leverages UAV imagery: YOLO v5/v8/NAS variants achieve 73.7% mAP on challenging leaf-disease benchmarks [128]; lightweight variants enable smartphone diagnosis across coriander, citrus, tea, rice, and coffee [129]; and the PDT dataset with YOLO-DP targets dense tree-crop pests [130]. Livestock monitoring spans YOLOv2–v10 with mAP above 90% for pigs and cattle [131]: swine behavior detection with Mixed ELAN [132], individual cattle identification at 99.79% mAP [133], YOLO-BoT behavior tracking at 93.6% accuracy and 31.2 FPS [134], YOLOv8m/v10m feeding-behavior comparison [135], PBR-YOLO piglet tracking [136], and 400+ poultry studies covering welfare and disease early warning [137]. UAV-based cereal multi-disease detection identifies YOLOv5 as the most efficient real-time architecture [138].

For hardware-constrained platforms, NAS-generated or compact architectures can be shortlisted when their operators are supported by the target runtime [75]. YOLOv12-N provides an attention-centric candidate [28], while YOLOv13 and YOLO26 offer different source-reported design priorities [15,16]. YOLO26 provides a simplified DFL-free and NMS-free graph, but the primary report does not benchmark low-cost smartphones or agricultural diagnostic workloads. Smartphone deployment should therefore be treated as a candidate use case requiring direct measurement of accuracy, end-to-end latency, energy, memory, and thermal stability on the selected device. Evaluation should also include application-specific economic measures such as herbicide reduction, yield improvement, and labor savings where relevant. Domain shift across cultivars, growth stages, environmental conditions, and acquisition platforms continues to limit model generalization; controlled synthetic augmentation, diffusion-based generation [60], and on-site calibration should be evaluated under documented protocols, with continual learning considered when field conditions evolve.

6.4. Healthcare and Biomedical Imaging

In clinical imaging, detection performance must be evaluated not only in terms of accuracy but also with respect to sensitivity, specificity, traceability, and clinical reliability: datasets are smaller, class imbalance is severe, and missed polyps or lesions can have consequences that extend well beyond benchmark performance. Colorectal polyp detection—where early removal directly reduces mortality—has been studied extensively. Representative methods include semantic feature-enhanced YOLOv5 with Coordinate Attention on Kvasir-SEG, CVC-ClinicDB, ETIS, and CVC-ColonDB [139], the latter validated through WM-DOVA localization maps [140]; YOLO-V8m reaching 95.6% precision, 91.7% recall, and 92.4% F1 across five public datasets [141]; LightAttn-YOLO-V8 [142]; and YOLO11n with LOF preprocessing [143], increasingly paired with YOLOv8-Seg [22] for polyp size measurement in adenoma surveillance.

Pulmonary lesion detection targets small (3–10 mm) nodules with subtle contrast and is evaluated predominantly on ChestX-ray8 (108,948 images, 8 disease categories) [144].

Representative methods include Fast-YOLO for pneumonia [145], NSEC-YOLO with adaptive noise suppression [146], YOLOv12-DSV at 73.5% mAP50 [147], PSO-optimized YOLOv6 [148], and ELCT-YOLO for CT lung tumors [149], with hepatic benchmarks such as LiTS [150] providing complementary abdominal evaluation; meta-analyses support YOLO-based radiograph analysis as an effective clinical second reader [151]. Brain tumor detection in MRI spans YOLO-NeuroBoost with Inner-GIoU and CBAM [80,152], improved YOLO11 [153], automated segmentation-classification [154], a 2020–2024 systematic review [155], and BGF-YOLO achieving state-of-the-art on Br35H [156]. Skin lesion analysis builds on YOLOv3 + GrabCut on the ISIC archive [157,158], hyperspectral conversion across YOLOv5–v11 [159], and smartphone-based Health-of-Things melanoma screening [160]. Mammographic mass detection benefits from Transformer-augmented YOLOv5 under limited training data [161]; cell-level applications include bone marrow differential counts [162] and TW-YOLO blood-cell detection using multi-scale feature fusion [163]; retinal screening addresses sub-50-pixel microaneurysms via YOLO on OCT [164] and YOLOv8/YOLOv9 segmentation of diabetic-retinopathy features in fundus images [165].

Architectural selection is often influenced by modality-specific requirements and clinical workflow constraints: YOLO11′s attention modules [31] may motivate application-specific evaluation in medical imaging, but their clinical sensitivity and false-positive rates require modality-specific evidence; YOLOv10′s NMS-free graphs [30] suit radiology PACS environments; YOLOv13 [15] holds promise for histopathology where hypergraph correlation captures relationships among adjacent cells; and YOLO26 [16] is a candidate for device-oriented deployment because its primary report documents DFL-free regression, an NMS-free inference path, multi-backend export, and source-reported CPU/T4 latency—although medical-grade FPGAs, INT8 accuracy retention, and intraoperative latency variance are not evaluated in that report and must be established in application-specific studies. Metrics extend to AUC, sensitivity at fixed false-positive operating points, and reader studies against clinician baselines [166]. Regulatory requirements related to privacy, security, and auditability often favor deployment strategies that provide greater control over data governance; open challenges include cross-institutional domain shift, label noise, regulatory compliance (FDA 510(k), CE-MDR, HIPAA), and subgroup fairness. Federated learning [167], differential privacy [168], and uncertainty calibration are being investigated to address privacy and reliability challenges; clinical suitability requires application-specific validation.

6.5. Robotics, Drones, and Surveillance

Robotics and surveillance applications emphasize ultra-low-latency perception, stable detection under egomotion, and tight coupling with trackers, SLAM systems, or planners. UAV-based small-object detection—with VisDrone serving as a canonical benchmark and objects often $< 32 \times 32$ pixels—has advanced through SOD-YOLO's P2-level detection head [169], LRDS-YOLO with SegNext Attention [170], on-drone EdGS-YOLOv8 [171], SOD-YOLOv8 for aerial and traffic scenes [172], and the lightweight EL-YOLO [173]. Wildlife monitoring combines ecology with conservation: the WAID dataset with SE-YOLO [174], BuckTales multi-UAV antelope tracking [175], DEAL-YOLO achieving 93.3% mAP@0.5 on WAID with sub-1 M parameters [176], YOLO adaptations for endangered species (e.g., WilDect-YOLO) [177], and FDM-YOLO thermal-RGB fusion for varying illumination [178].

Fire and disaster detection addresses wildfire prevention amid climate change: optimized YOLOv5 smoke detection [179], Fire-PPYOLOE with large kernels [180], YOLOv7x/YOLOv8s at 80.4% mAP across ground and aerial imagery [181], YOLOGX forest fire detection [182], and ForestGuard integrating YOLO with Federated Learning [167]

across distributed drone fleets [183]. Underwater robotics contends with color distortion and camouflage: UCOEA clustering [184], an attention-enhanced YOLOv7 for underwater object detection [185], aquaculture YOLOv7 at 77% precision/recall [186], DLFE-YOLO at 84.4% mAP@0.5 on URPC2020 [187], and YOLO-Starfish for ocean ranches [188]. Surveillance and SAR complete the picture: dual-stream RGB/IR YOLO11n distinguishing drones from birds at 0.989/0.988 mAP@0.5 on the VIP Cup 2025 validation data and supporting payload identification [189], YOLOv9-based intrusion alerts [190], and YOLO-FEDER FusionNet gaining +62.8 percentage points mAP@0.5 in camouflaged SAR scenes [191]. Manipulation and grasping run lightweight variants (YOLOv5-n/s [27], YOLOv8-n [22]) on Raspberry Pi and Jetson platforms, with YOLOX + ByteTrack for smooth tracklets [23], keypoint-based orientation-aware grasping [121], and ORB-SLAM semantic landmarks [192].

Edge robotics is a relevant target for YOLO26, but the primary report should be represented by the hardware it actually evaluates. YOLO26-N reports 40.9 AP for the standard head and 40.1 AP for the E2E head, alongside released-model times of 38.9 ms for CPU ONNX and 1.7 ms for T4 TensorRT at 640 × 640 [16]; the same report does not provide simultaneous-stream results or frame-time variance measurements. YOLOv13-N reports 41.6 AP at 1.97 ms T4 TensorRT FP16 in its own evaluation, and its hypergraph correlation is motivated by occlusion and clutter, although a controlled cross-model occlusion benchmark is not available [15]. Because software versions and measurement procedures differ, these values should be treated as source-specific operating points rather than a direct device ranking. A hybrid configuration that runs YOLO26 continuously with periodic YOLOv13 verification is a prospective system design requiring experimental validation. Proper evaluation reports HOTA, MOTA, and IDF1 alongside mAP where tracking is involved, together with latency under load, energy, and thermal behavior under the application's actual occlusion and motion conditions. Open-set conditions—unknown classes, weather and time-of-day shifts—continue to challenge closed-world detectors, with on-device distillation and continual adaptation as emerging responses.

6.6. Multimedia, AR/VR, and Smart Devices

Consumer-facing and immersive applications introduce a unique set of requirements related to latency, computational efficiency, and user experience: detection must share compute with rendering pipelines, integrate with platform runtimes (CoreML on iOS, NNAPI on Android), and satisfy latency budgets often below 16 ms to match display refresh rates. Mobile deployment depends on export and runtime support. YOLOv5 provides export options for multiple deployment formats [27], while the Ultralytics ExecuTorch integration supports mobile deployment of YOLO26 [193]. Mobile-device performance depends strongly on the export graph, delegate, numerical precision, and thermal state; a follow-up overview reports real-time smartphone operation for compact variants only at a qualitative level [46], and neither it nor the primary report provides controlled benchmarks for specific smartphone SoCs or NPU accelerators, so device-specific latency figures should be reported only after direct measurement with the exported model and a documented runtime configuration.

AR/VR systems require tight integration between perception and rendering pipelines operating at 60–120 Hz to maintain a responsive and stable user experience. Hand tracking predominantly pairs YOLO-based detection with 21-point landmark extraction: reviewed across AI-driven gesture systems [194], simplified by YOLO11′s integrated pose estimation [31], demonstrated in multi-user VR gaming combining YOLOv8 with MediaPipe [195], and extended to XR across Oculus Quest, Apple Vision Pro, and HoloLens 2 [196]. The NMS-free inference paths of YOLOv10 [30] and YOLO26 [16] remove a data-

dependent post-processing stage from the deployed graph; however, the primary reports do not provide frame-to-frame variance measurements, so XR suitability should be verified using end-to-end frame time, jitter percentiles, and missed-frame rate on the target headset. Smart-camera and IoT applications often use nano-scale detectors, including YOLOv5n [27], YOLOv8n [22], YOLOv10-N [30], and YOLO26-N [16], to support on-device inference where privacy, bandwidth, and latency are important. The cited sources do not establish a deployment count across millions of consumer devices; therefore, adoption scale should not be quantified without a dedicated market or deployment source. Retail analytics spans YOLO-based foot-traffic analysis [197], vision-based queue-length estimation [198], automated retail checkout with deep tracking [199], and unmanned-store customer identification using YOLOv4 together with a Siamese network [200], executed on in-store edge processors. These systems illustrate distinct retail tasks and should be evaluated using task-specific measures such as counting error, identification accuracy, checkout error rate, and end-to-end latency on the in-store processor. Sports analytics integrates YOLOv5/v8 with multi-object tracking for player tracking and tactical analysis [201], and open-vocabulary extensions such as YOLO-World [202] enable zero-shot detection of natural-language categories for content tagging.

For flagship hardware with dedicated NPUs, YOLO11 [31], YOLOv12 [28], and YOLOv13 [15] are the accuracy-oriented candidates; for mass-market deployment across price tiers, YOLO26 [16] is a strong candidate because its primary report documents a DFL-free head, NMS-free inference, ExecuTorch/TFLite/CoreML export paths, and up to 43% faster CPU ONNX inference than YOLO11-N on the tested platform—although no universal INT8 accuracy-retention or deterministic-latency guarantee is inferred, and device selection should rest on measured accuracy after export, latency distributions, energy, memory, and thermal stability on each target. Open challenges include sub-100 mW always-on detection, privacy-preserving on-device learning, and integration with multimodal foundation models for richer scene interpretation.

6.7. Cross-Domain Synthesis and Best Practices

Figure 17 summarizes the type of evidence available for representative model families across the application domains reviewed in Sections 6.1–6.6. The matrix distinguishes documented evaluations, including task-specific derivatives (D), inferred architectural relevance (A), and cells without a selected evidence mapping (NR); these categories do not rank models or identify a primary recommendation. Recent models such as YOLOv13 and YOLO26 have limited accumulated deployment evidence, so their positions should be treated as hypotheses for controlled evaluation rather than as validated domain leadership.

Four recurring challenges emerge consistently across the application domains surveyed in this review. Small-target detection appears in distant-pedestrian [93], defect-inspection [105], agricultural [125], medical [164], and aerial [169] settings. Reported strategies include P2-level heads [169], Space-to-Depth modules [97,105], and Small-Target-Aware Label Assignment. In the YOLO26 source ablation, STAL increases AP_S from 29.0 to 29.6, a gain of 0.6 AP points [16]. Occlusion occurs in crowded intersections [93], fruit clusters [120,121], dense cells [162], and vegetation-hidden targets [174]. Context-enhancement mechanisms such as HyperACE [15], C2PSA [31], Area Attention [28], and CBAM [80] are relevant candidates, but their benefits must be established through controlled domain-specific tests. Class imbalance affects manufacturing [115], rare pathologies, and minority weed species [125,126]; focal loss [32], controlled sampling, and carefully validated synthetic generation [60] are possible responses. Distribution shift across geographies, institutions, cultivars, and production lines should be reported for each dataset and protocol rather than summarized as a universal 15–25% degradation. Mitigation may

include external-domain evaluation, adaptation, domain randomization, and federated learning where appropriate [167,183].

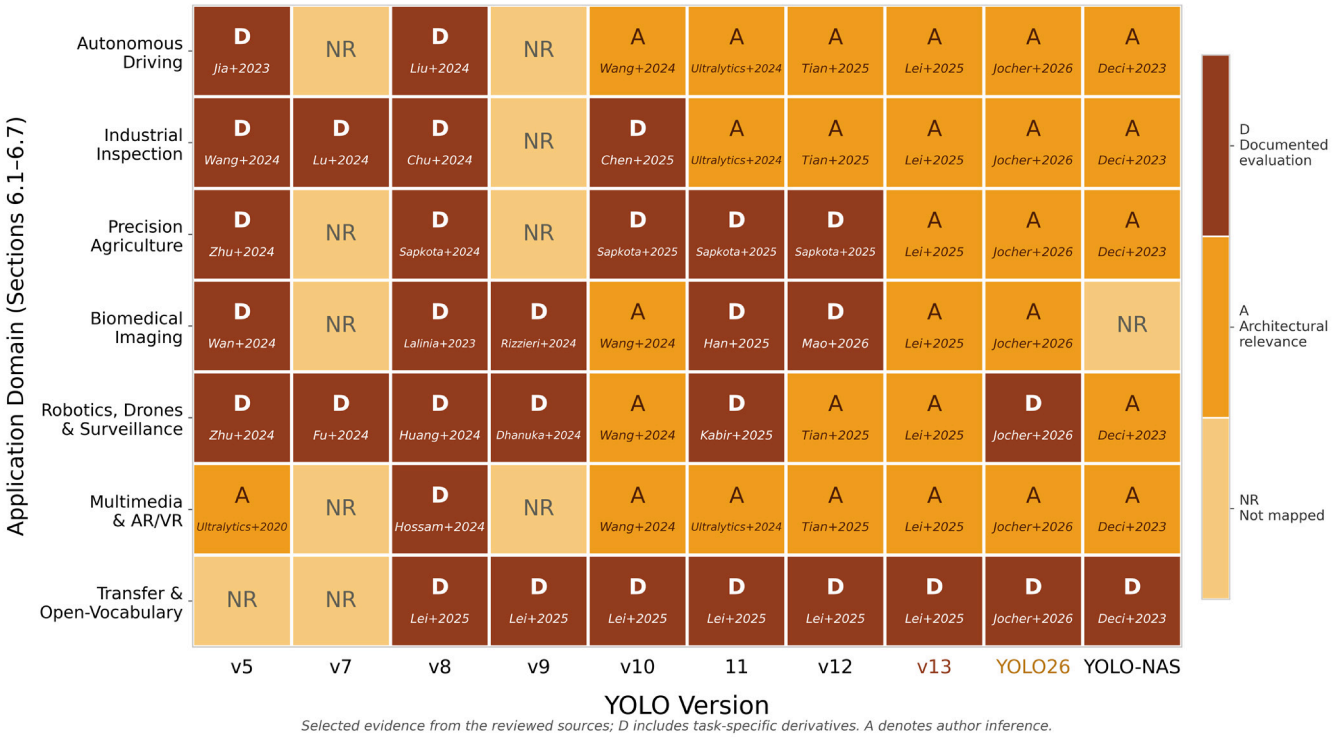


Figure 17. Application-domain evidence map for ten representative YOLO families. D denotes a reported evaluation of a model family or a task-specific derivative; A denotes author-inferred architectural relevance; NR denotes no selected mapping in this figure. Colors distinguish evidence categories rather than suitability rankings. D does not imply operational deployment, independent replication, or superiority. The final row groups distinct evaluation settings: COCO-to-Pascal-VOC transfer evaluation for YOLOv8–YOLOv13 [15], open-vocabulary evaluation of the YOLOE-26 extension [16], and developer-reported fine-tuning on Roboflow-100 for YOLO-NAS [75]. These settings are not directly comparable. The map is illustrative rather than exhaustive, and model selection requires target-specific validation. Sources by row: autonomous driving [92,95]; industrial inspection [104,105,107,108]; precision agriculture [60,119,124]; biomedical imaging [139,141,147,153,165]; robotics, drones, and surveillance [16,119,171,185,189,190]; multimedia and AR/VR [197]; transfer and open-vocabulary [15,16,75]. Cells marked A refer to the architecture sources [15,16,27,28,30,31,75].

Hardware-aware selection should begin with model scale, operator support, export compatibility, and direct measurement on the target platform. Nano and small variants can be shortlisted for mobile or edge evaluation, while larger variants may be appropriate for server-class GPUs. The YOLO26 report provides CPU ONNX and T4 TensorRT measurements [16] but does not report controlled benchmarks on low-cost smartphones, Jetson devices, or serverless-cloud environments. YOLO-NAS [75] provides an architecture-search approach for target-specific constraints without establishing a universal efficiency advantage. Multi-task frameworks, including YOLOv8 [22], YOLO11 [31], and YOLO26 [16], can reduce duplicated feature extraction when several tasks share a backbone, but the end-to-end system benefit remains workload-dependent and must be measured.

Looking ahead, the ecosystem appears likely to evolve along two complementary trajectories: continued refinement of inference-time efficiency for large-scale production AI (YOLO26 [16]) and increasingly sophisticated contextual reasoning for accuracy-critical applications (YOLOv13 [15]) —both increasingly integrated with foundation models (YOLO-World [202]), generative synthetic data [60], and federated continual learning [167,183].

For practitioners, the central takeaway is that successful deployment depends on aligning architectural choice, training strategy, evaluation methodology, and operational constraints within a unified decision-making framework.

7. Unified Evaluation of YOLO: Accuracy, Robustness, and Deployment Efficiency

Meaningful empirical comparison requires careful consideration of the experimental conditions under which individual results were reported. Following the methodological conventions of comparable surveys [9–11], the tables and figures in this section compile results from primary release papers, official documentation, and selected follow-up evaluations [15,16,46] rather than reporting new experiments. Hardware, runtime, batch size, precision, and input resolution are retained wherever available. Because substantial differences exist across datasets, hardware platforms, runtimes, and evaluation settings, the resulting analysis should be interpreted as a cross-source synthesis rather than a hardware-normalized benchmark. Figures 18 and 19 therefore present source-reported operating points as scatter plots and should not be interpreted as a global Pareto frontier or a controlled head-to-head ranking.

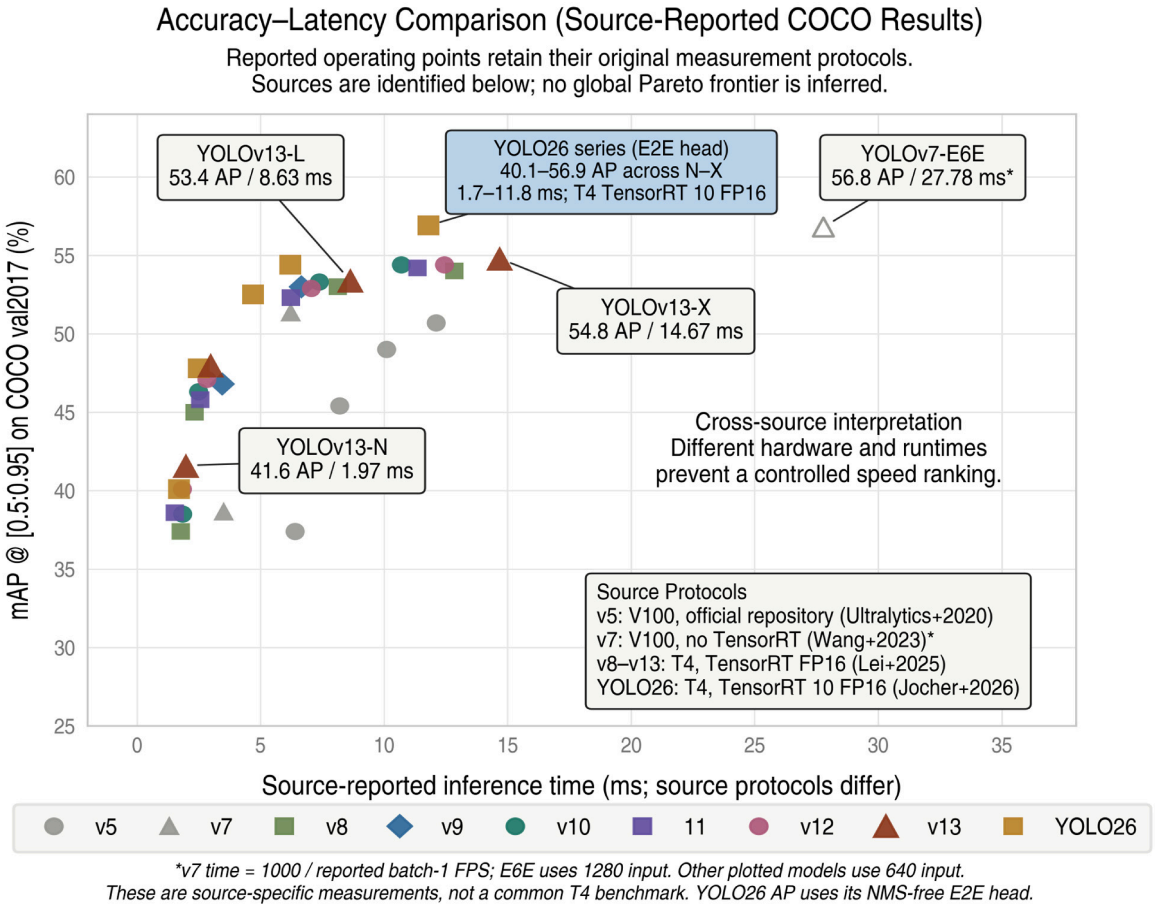


Figure 18. Source-reported COCO val2017 AP versus inference time for representative YOLO models. Colors and markers identify model families; the inset lists their source protocols. YOLOv5 values are from the legacy anchor-based checkpoints in the official repository [27], using V100 batch-1 measurements that exclude NMS. YOLOv7 times are derived as 1000 divided by the reported batch-1 FPS on V100 without TensorRT [26]. YOLOv8–YOLOv13 points retain the main COCO comparison in [15]. YOLO26 points use the E2E AP values of the main COCO results table in [16]. Input size is 640 except for YOLOv7-E6E at 1280. The plot does not define a hardware-normalized global Pareto frontier.

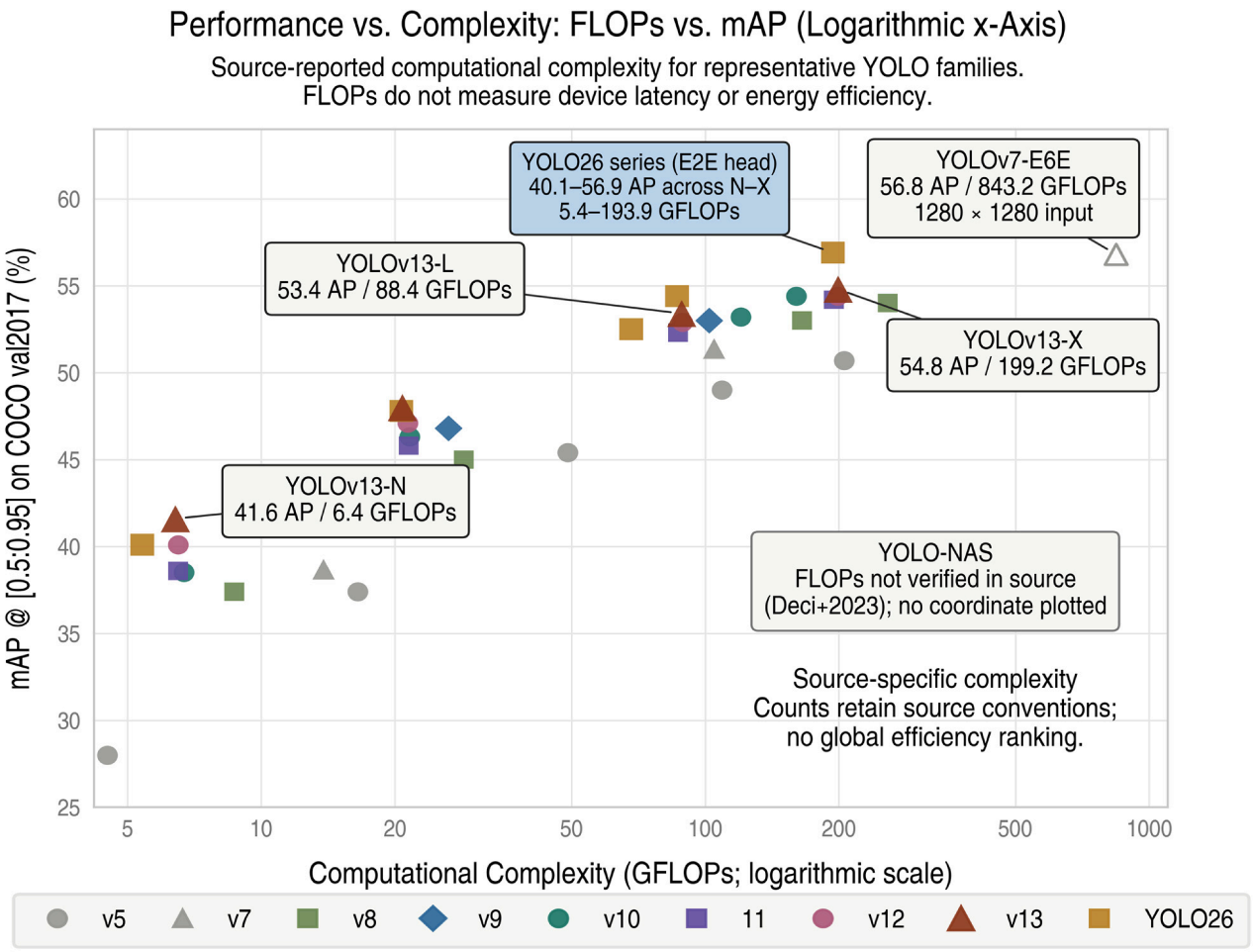


Figure 19. Source-reported COCO val2017 AP versus nominal FLOPs on a logarithmic horizontal axis. Colors and markers identify model families. Operation counts retain the conventions of their sources [15,16,26,27]. YOLOv7-E6E uses 1280 input and is shown with an open marker; the other points use 640 input. YOLO26 values use E2E AP. YOLO-NAS is retained in an explanatory note but has no plotted coordinate because a matching FLOP value was not verified in [75]. FLOPs do not measure latency or energy, and no global efficiency ranking is asserted.

7.1. Evaluation Protocol Conventions

The compiled benchmarks follow the COCO evaluation framework [39]; however, dataset splits, input resolutions, timing procedures, and runtime environments remain source-dependent. In Table 4, source C reports test-dev AP, whereas the other source groups report val2017 AP; nominal input size is 640 except for YOLOv7-E6E at 1280. Hardware and runtime details are specified in the source notes, and unspecified settings are not assumed to be shared. The tabulated metrics are AP at IoU 0.5, AP averaged over IoU thresholds 0.5–0.95, inference time, FPS, parameter count, nominal GFLOPs and estimated FP32 parameter storage. For image-quality-sensitive deployments (medical imaging preprocessing, autonomous-driving visibility assessment), additional metrics, including the Structural Similarity Index (SSIM) [203], complement detection-specific scores by quantifying visual fidelity under compression or low-light conditions. Reproducibility considerations—random seed protocols, framework versions, dataset checksums, and hyperparameter configurations—follow the conventions documented in Detectron2 [204], MMDetection [205], and the Ultralytics framework [22].

Table 4. Source-reported performance and model size for representative YOLO-family detectors. Letters A–H identify the sources and evaluation conditions in the notes. AP is expressed on a 0–100 scale; YOLO26 entries distinguish standard-head and E2E AP. An asterisk marks a calculated conversion between inference time and FPS. Estimated FP32 parameter storage is calculated from the reported parameter count and peak memory is not measured. Differences in dataset split, input handling and timing protocol preclude a hardware-normalized ranking across all rows.

Model	AP_{50}	AP50:95 Standard/E2E [1]	Inference Time (ms) [2]	FPS [2]	Parameters (M)	FLOPs (G)	Estimated FP32 Parameter Storage (MB) [3]	Source
YOLOv5n	45.7	28.0	6.3	159 *	1.9	4.5	7.6	A [27]
YOLOv5s	56.8	37.4	6.4	156 *	7.2	16.5	28.8	A [27]
YOLOv5m	64.1	45.4	8.2	122 *	21.2	49.0	84.8	A [27]
YOLOv5l	67.3	49.0	10.1	99 *	46.5	109.1	186.0	A [27]
YOLOv5x	68.9	50.7	12.1	83 *	86.7	205.7	346.8	A [27]
YOLOX-S	-	40.5	9.8	102 *	9.0	26.8	36.0	B [23]
YOLOX-M	-	46.9	12.3	81 *	25.3	73.8	101.2	B [23]
YOLOX-L	-	49.7	14.5	69 *	54.2	155.6	216.8	B [23]
YOLOv7-Tiny-SiLU	56.7	38.7	3.50 *	286	6.2	13.8	24.8	C [26]
YOLOv7	69.7	51.4	6.21 *	161	36.9	104.7	147.6	C [26]
YOLOv7-E6E	74.4	56.8	27.78 *	36	151.7	843.2	606.8	C [26]
YOLOv8n	52.6	37.4	1.77	565 *	3.2	8.7	12.8	D [15]
YOLOv8s	61.8	45.0	2.33	429 *	11.2	28.6	44.8	D [15]
YOLOv8m	67.2	50.3	5.09	196 *	25.9	78.9	103.6	E [28]
YOLOv8l	69.8	53.0	8.13	123 *	43.7	165.2	174.8	D [15]
PP-YOLOE-S	60.5	43.1	4.81 *	208	7.9	17.4	31.6	C [26]
PP-YOLOE-M	66.5	48.9	8.13 *	123	23.4	49.9	93.6	C [26]
PP-YOLOE-X	69.9	52.2	22.22 *	45	98.4	206.6	393.6	C [26]
YOLOv9-C	70.2	53.0	6.64	151 *	25.3	102.1	101.2	D [15]
YOLOv9-E	72.8	55.6	-	-	57.3	189.0	229.2	F [29]
YOLOv10-N	53.8	38.5	1.84	543 *	2.3	6.7	9.2	E [28]
YOLOv10-S	63.0	46.3	2.49	402 *	7.2	21.6	28.8	E [28]
YOLOv10-M	68.1	51.1	4.74	211 *	15.4	59.1	61.6	E [28]
YOLO-NAS-S	-	47.5	3.21	312 *	-	-	-	G [75]
YOLO-NAS-M	-	51.55	5.85	171 *	-	-	-	G [75]
YOLO-NAS-L	-	52.22	7.87	127 *	-	-	-	G [75]
YOLO11-N	54.2	38.6	1.53	654 *	2.6	6.5	10.4	D [15]
YOLO11-S	62.6	45.8	2.56	391 *	9.4	21.5	37.6	D [15]
YOLO11-M	68.5	51.5	4.7	213 *	20.1	68.0	80.4	E [28]
YOLOv12-N	56.0	40.1	1.83	546 *	2.6	6.5	10.4	D [15]
YOLOv12-S	64.2	47.1	2.82	355 *	9.3	21.4	37.2	D [15]
YOLOv12-M	69.6	52.5	4.86	206 *	20.2	67.5	80.8	E [28]
YOLOv13-N [15]	57.8	41.6	1.97	508 *	2.5	6.4	10.0	D [15]
YOLOv13-S	65.2	48.0	2.98	336 *	9.0	20.8	36.0	D [15]
YOLOv13-L	70.9	53.4	8.63	116 *	27.6	88.4	110.4	D [15]
YOLOv13-X	72.0	54.8	14.67	68 *	64.0	199.2	256.0	D [15]

Table 4. *Cont.*

Model	AP_{50}	AP50:95 Standard/E2E [1]	Inference Time (ms) [2]	FPS [2]	Parameters (M)	FLOPs (G)	Estimated FP32 Parameter Storage (MB) [3]	Source
YOLO26-N [16]	-	40.9/40.1	1.7	588 *	2.4	5.4	9.6	H [16]
YOLO26-S	-	48.6/47.8	2.5	400 *	9.5	20.7	38.0	H [16]
YOLO26-M	-	53.1/52.5	4.7	213 *	20.4	68.2	81.6	H [16]
YOLO26-L	-	55.0/54.4	6.2	161 *	24.8	86.4	99.2	H [16]
YOLO26-X	-	57.5/56.9	11.8	85 *	55.7	193.9	222.8	H [16]

Notes: [1]. AP50 denotes AP at IoU 0.5; AP50:95 averages AP over IoU thresholds 0.5–0.95. In the five YOLO26 rows, the AP50:95 entry is standard/E2E (one-to-many with NMS/one-to-one without NMS). All other rows contain one AP value under the source conditions below. [2]. * Calculated value. For source C, the source reports FPS, and time is calculated as 1000/FPS, rounded to two decimals. For the other timed rows, FPS is calculated as 1000/reported milliseconds, rounded to the nearest integer. These reciprocals are not independently measured throughput. Conversions use the original reported number, not a previously rounded conversion. [3]. Estimated FP32 parameter storage (MB) = 4 × reported parameters (M), with decimal MB (10^6 bytes), rounded to one decimal. This is an estimate from the rounded parameter count and excludes activations, workspaces, input/output tensors and runtime overhead. FLOPs are source-reported nominal operation counts, not measurements of a particular GPU or precision. - indicates that the selected source does not provide the metric. YOLOX AP50, YOLOv9-E timing/FPS and YOLO-NAS AP50/parameters/FLOPs/storage are omitted for this reason. No metric is imputed. The main COCO results table of the YOLO26 technical report [16] provides one released-model timing column alongside two AP columns; the present table does not infer a separate latency for each head. A [27]—Ultralytics YOLOv5 official repository, legacy Pretrained Checkpoints. COCO val2017; P5 checkpoints, nominal 640 input; V100, batch 1; NMS excluded. The source does not specify a precision/runtime version for this table. B [23]—YOLOX official repository, Benchmark, Standard Models. COCO val2017 AP; 640 input; V100 latency. Runtime, precision, batch size and timing boundaries are not specified beside the selected repository table; no additional conditions are assumed. C [26]—YOLOv7 paper (main comparison table), V100 comparison block. COCO test-dev AP and AP50; V100, batch 1, without TensorRT. Nominal size 640, except YOLOv7-E6E at 1280. The paper notes letterbox resizing for timing. PP-YOLOE rows retain the values reported in this comparison table; per-model runtime and precision are not standardized here. D [15]—YOLOv13 paper (main COCO comparison table and Section IV-A). COCO val2017; 640 input; T4, TensorRT FP16. The comparison reproduces YOLO11 and YOLOv12 v1.0 under the authors' settings. Runtime version, batch size and identical post-processing boundaries are not specified in the selected table. E [28]—YOLOv12 paper (main results table and experimental settings). COCO val2017; 640 input; T4, TensorRT FP16. All retained cells come from the same comparison row. YOLOv12-M is the original v1.0 model, not the later Turbo model. Runtime version, batch size and identical timing boundaries are not specified in the selected table. F [29]—YOLOv9 official repository, Performance table. COCO val2017; 640 input. AP50, AP, parameters and FLOPs are provided; inference time and FPS are not. No latency is inferred from parameter count or another model. G [75]—SuperGradients official YOLONAS.md, model table. COCO val2017; 640 input; NVIDIA T4. Non-INT8 rows are used. The selected source does not identify their runtime version, precision, batch size or timing boundaries, and does not tabulate AP50, parameter count or FLOPs. H [16]—YOLO26 paper (main COCO results table and the timing description of its first figure). COCO val2017; 640 input; T4, TensorRT10; the paper describes T4 timing as FP16. AP is shown as standard/E2E. The source table provides one released-model speed column, not two separately measured timings for the two heads; batch size is not specified there.

The surveyed studies differ considerably with respect to dataset selection, training configuration, runtime environment, hardware platform, and performance-measurement methodology. Missing protocol details are stated explicitly in the source notes; they are not filled using settings from another model or study. Normalization conventions also differ across detection frameworks. Batch normalization is common in convolutional detector components, while transformer components and other detection frameworks may use layer normalization [66] or group normalization [206]. Group normalization reduces sensitivity to the training batch size [206]. Table 4 retains identified rows from official repositories and published comparison tables, without averaging or selecting the highest value across

sources. A dash denotes a metric not reported in the selected source, and an asterisk identifies an arithmetic conversion between inference time and FPS. Independent multi-domain reproduction studies reinforce the need for this discipline: when YOLO versions are retrained and evaluated under a single uniform protocol, newer versions do not always outperform their predecessors, and version rankings can deviate substantially from those implied by primary-release reports [41,44]. Consequently, all reported values should be interpreted within the context of their original evaluation protocols.

7.2. Accuracy-Speed-Size Trade-Offs

Table 4 compiles 41 representative YOLO-family model variants, including YOLOX [23], PP-YOLOE [24] and YOLO-NAS [75]. The letter beside each model identifies the source and evaluation conditions in the table notes. Values for the YOLOv8, YOLOv9-C, YOLOv10, YOLO11, YOLOv12 and YOLOv13 rows follow the identified comparison tables in [15,28]; other rows use the sources specified in notes A–C and F–H. Scaled-YOLOv4 [71] is discussed elsewhere in this review but is not tabulated here. AP50 is retained only where the selected source reports it. YOLO26 AP is presented separately for the standard and end-to-end heads, alongside the released-model timing reported in [16]. The reported results should not be interpreted as a unified ranking across heterogeneous evaluation protocols.

Within the YOLOv13 primary report, YOLOv13-N reaches 57.8 AP_50 and 41.6 AP with 2.5 M parameters and 1.97 ms T4 TensorRT FP16 latency, while YOLOv13-X reaches 54.8 AP with 64.0 M parameters [15]. These values describe the operating points reported by that study. Comparisons with YOLO11 and YOLOv12 should remain tied to the protocol used in the YOLOv13 paper and should not be interpreted as a hardware-normalized Pareto shift across all sources.

Figure 18 plots source-reported COCO AP against inference time for representative model variants. Its points follow the sources and validation settings identified in the figure caption, rather than reproducing every row of Table 4. In particular, Figure 18 uses validation AP for YOLOv7 and E2E AP for YOLO26, whereas Table 4 identifies the YOLOv7 test-dev results and displays both YOLO26 head-specific AP values. Runtime versions and timing boundaries also differ across sources. Accordingly, the visualization should be interpreted as a descriptive comparison of reported operating points rather than as a hardware-normalized global Pareto frontier.

Figure 19 plots source-reported COCO AP against nominal FLOPs on a logarithmic horizontal axis. Its points retain the source-specific validation results identified in the caption; they are not a plot of every row in Table 4. For YOLO26, the figure uses E2E AP, which is explicitly distinguished from standard-head AP in Table 4. FLOPs indicate arithmetic complexity but do not measure latency, memory traffic, energy or accelerator utilization. The plot therefore shows the range of reported compute scales rather than a hardware-normalized efficiency ranking.

7.3. Robustness and Generalization

Real-world deployment frequently exposes detectors to conditions that differ substantially from those represented in benchmark datasets. Common sources of performance degradation include illumination variation, motion blur, sensor noise, compression artifacts, partial occlusion, crowding, and distribution shift [10,11]. However, the surveyed studies use different datasets, corruption severities, preprocessing pipelines, training regimes, and metrics. A model-by-model retention ranking cannot therefore be reconstructed reliably from these cross-source results.

Table 5 summarizes the mechanisms by which each stress condition affects a detector and the measurements needed for defensible evaluation. This evidence-bounded format

avoids assigning numerical robustness scores to models that were not tested under an identical protocol. Architectural mechanisms such as attention, hypergraph correlation, or NMS-free decoding may motivate hypotheses about robustness, but those hypotheses require controlled corruption benchmarks before they can be treated as comparative evidence.

Table 5. Evidence-bounded framework for robustness evaluation. This conceptual synthesis does not rank models or report retention percentages because the surveyed studies use heterogeneous protocols.

Stress Condition	Typical Failure Mechanism	Defensible Evaluation	Relevant Mitigation
Low illumination	Reduced signal-to-noise ratio, color shift, loss of local contrast	Clean vs. low-light AP at fixed illumination/severity; calibration and object-scale breakdown	Exposure-aware augmentation, low-light preprocessing, sensor fusion
Motion blur	Boundary smearing and feature displacement, especially for small objects	AP by blur kernel/trajectory and object speed; temporal consistency for video	Motion-aware augmentation, shorter exposure, temporal modeling
Sensor noise	False texture and unstable confidence under low SNR	AP and calibration by noise type and severity using the same checkpoint	Noise augmentation, denoising, sensor-specific calibration
Occlusion/crowding	Missing evidence, overlapping instances, assignment ambiguity	AP by visible fraction, APS, missed detections, crowd-specific datasets	Context modeling, scale-aware assignment, targeted sampling
JPEG/compression	Blocking and ringing artifacts, confidence drift	AP and calibration by codec and quality setting; bandwidth-accuracy curve	Compression-aware training, bitrate-adaptive preprocessing
Domain shift	Changes in geography, sensor, weather, or institutional acquisition	External-domain AP, calibration, subgroup analysis, drift monitoring	Domain adaptation, continual evaluation, federated or privacy-preserving updates

Robust evaluation practices should include both clean-condition and corruption-sensitive performance metrics using the same trained checkpoint, dataset split, corruption implementation, and severity schedule. Results should include absolute AP, relative retention, confidence calibration, and subgroup or object-scale breakdowns. For video applications, temporal stability and tracking metrics should be added. Under these requirements, robustness becomes a reproducible property of a model–data–runtime configuration rather than an intrinsic ranking inferred from architecture labels.

7.4. Small Objects, Rare Classes, and Crowded Scenes

Applications such as intelligent transportation [92–94] and aerial surveillance [169,170,174] are dominated by small or distant targets where standard detection metrics conceal substantial performance heterogeneity across object scales. Following the COCO evaluation protocol, AP breakdowns by object scale (AP_S for small objects with area $< 32^2$, APM for medium 32^2–96^2, APL for large $>96^2$) reveal architectural sensitivities to scale and density that aggregate mAP cannot expose. Table 6 compiles these breakdowns from primary sources. Anchor-free prediction and decoupled heads are relevant design choices, but their effects on small-object AP and crowded-scene performance depend on the model, assignment strategy, dataset, and evaluation protocol. Table 6 should therefore be read as a source-specific scale breakdown rather than proof of a universal architectural advantage.

Table 6. S-scale COCO val2017 AP values reproduced as tabulated in the YOLO26 technical report [16]; YOLO26 values correspond to the standard (one-to-many) head. This source-reported comparison is not an independent uniform re-benchmark by the present authors; APS, APM, and APL denote scale-specific AP rather than recall or miss rates.

Model	AP	AP_{Small}	AP_{Medium}	AP_{Large}
YOLOv9-S	46.8	25.7	49.9	61.0
YOLOv10-S	46.3	26.8	51.0	63.8
YOLO11-S	47.0	29.0	51.7	64.4
YOLOv12-S	48.0	29.8	53.2	65.6
YOLO26-S	48.6	29.5	53.2	65.8

Aggregate AP can conceal substantial differences among small, medium, and large objects. Table 6 reproduces the S-scale comparison as tabulated in the YOLO26 technical report [16]. The table provides a traceable reporting source, but it should not be interpreted as independent retraining of all baseline models; the values retain the evaluation provenance assembled by [16].

The comparison shows that small-object performance does not improve monotonically with release year. APS rises from 25.7 for YOLOv9-S and 26.8 for YOLOv10-S to 29.0 for YOLO11-S and 29.8 for YOLOv12-S, whereas YOLO26-S reports 29.5 despite achieving the highest overall AP among the listed YOLO variants. These observations further support the broader conclusion that architectural progress is inherently multi-objective: a newer detector may improve overall accuracy or deployment simplicity without dominating every object scale. Crowd-specific performance is not inferred here because the reporting source does not provide a common crowd metric for these rows. APS measures precision–recall performance across IoU thresholds rather than a missed-object proportion. Quantitative precision and recall complements, including a small-target Ships/Vessels comparison, are reported separately in Section 7.7; those dataset-level values are not COCO-area-filtered miss rates.

7.5. Resource Efficiency and Deployment Friendliness

Beyond detection accuracy, deployment decisions are increasingly influenced by energy efficiency, thermal behavior, memory utilization, and quantization robustness. Quantization may improve deployment efficiency, but its effects depend on calibration data, operator coverage, runtime, hardware, and the accuracy metric. The YOLO26 primary report does not provide a controlled INT8 detection table, and the follow-up overview [46] discusses quantization robustness only qualitatively; consequently, this review does not impute INT8 accuracy deltas or latencies for specific models. Table 7 instead defines the minimum evidence needed for a reproducible deployment comparison: quantized results should be included only when the FP32/FP16 and INT8 engines are generated from the same checkpoint, evaluated on the same dataset, and accompanied by a complete calibration and runtime manifest.

CPU and GPU performance characteristics should be evaluated independently because they reflect different deployment objectives. The YOLO26 primary report gives 38.9 ms CPU ONNX latency for YOLO26-N and reports up to 43% higher CPU throughput than YOLO11-N under its Intel Xeon protocol [16]. YOLOv13 reports 39.97 ms CPU ONNX latency for YOLOv13-N on an Intel Xeon Platinum 8352V [15]. Because the processors, runtimes, and measurement procedures are not identical, these values should be presented as source-specific operating points. Neither primary detection report provides a controlled Jetson benchmark, frame-time variance, or a universal edge-platform speedup.

Table 7. Minimum evidence required for reporting quantized YOLO deployment results. This table is a reporting checklist, not a model-performance ranking.

Evidence Item	Required Reporting Fields	Reason
Model identity	Checkpoint hash, model scale, task head, input size	Prevents comparison of non-equivalent graphs
Accuracy baseline	Dataset/split, metric, FP32 or FP16 AP, confidence/NMS settings	Defines the reference from which quantization loss is computed
Quantization	PTQ or QAT, bit width, per-channel/per-tensor scheme, excluded operators	Quantization methods are not interchangeable
Calibration	Dataset, image count, sampling strategy, preprocessing	Calibration distribution can materially affect INT8 accuracy
Runtime	Backend, version, precision, batch size, warm-up, repetitions	Latency depends on software configuration
Hardware	Device, power mode, clocks, thermal state, memory	Enables reproducibility and energy interpretation
Statistics	Median and tail latency, variance, throughput, energy, thermal behavior	Mean FPS alone can conceal unstable deployment behavior

YOLOv13 and YOLO26 represent different design priorities, but the available primary evidence does not support a shared dense-scene, occlusion, Jetson, or quantization benchmark. Table 8 therefore separates reported facts from untested deployment hypotheses. YOLOv13 introduces HyperACE and FullPAD for high-order correlation modeling [15], whereas YOLO26 removes DFL, provides native NMS-free inference, and changes branch supervision and small-target assignment during training [16]. Their source-reported latency and AP values can be listed as operating points, but a definitive application ranking requires evaluation under the same model scale, runtime, hardware, dataset, and corruption protocol.

Table 8 supports an evidence-bounded shortlist rather than a categorical recommendation. YOLOv13 can be evaluated when high-order contextual modeling is relevant, whereas YOLO26 can be evaluated when a DFL-free and NMS-free deployment graph is desirable. The final choice must be made through controlled evaluation using the same dataset, model scale, input size, runtime, precision, and hardware. A hybrid system combining both families is a prospective research design and should not be presented as a practical solution until it is experimentally validated.

7.6. Ablation Studies and Module Contributions

Table 9 summarizes representative source-specific ablation studies reported for YOLOv13 and YOLO26 [15,16]. For YOLOv13-S, the complete HyperACE–FullPAD configuration reaches 48.0 AP compared with 47.1 AP when hypergraph-enhanced distribution is disabled, while single-path FullPAD configurations reach 47.6–47.8 AP [15]. For YOLO26, the table reports controlled results for DFL removal, Progressive Loss, STAL, and MuSGD [16]. These effects should not be interpreted as cumulative across distinct model families or experimental configurations and should not be generalized to unrelated architectures, datasets, or training schedules.

Table 8. Evidence-bounded comparison of YOLOv13 and YOLO26. NR indicates that the primary detection report does not provide the specified evaluation.

Dimension	YOLOv13 [15]	YOLO26 [16]	Interpretation
Primary design goal	High-order correlation through HyperACE and FullPAD	DFL-free, NMS-free, multi-task deployment-oriented design	Different architectural objectives
Nano COCO operating point	41.6 AP; 2.5 M; 6.4 G; 1.97 ms T4 FP16	40.9 AP (standard one-to-many); 40.1 AP (E2E one-to-one); 2.4 M parameters; 5.4 GFLOPs; source-reported T4 TRT10 latency: 1.7 ms	Source-specific; not a controlled head-to-head test
CPU ONNX nano latency	39.97 ms on Intel Xeon Platinum 8352V	38.9 ms under the CPU ONNX protocol reported in [16]	Source-specific CPU values; processor and runtime details differ, so no direct speed ranking is inferred
Controlled dense/occlusion benchmark	NR	NR	No cross-model robustness ranking is supported
Controlled INT8 detection table	NR	NR	Do not infer quantization loss from architecture alone
Selection rule	Candidate when high-order contextual modeling is relevant	Candidate when DFL-free and NMS-free deployment is relevant	Final choice requires target-specific measurement

Table 9. Source-specific ablation findings for YOLOv13 and YOLO26. Each delta is computed only within the model scale, dataset, and training protocol of the cited source; rows are not cumulative across model families.

Model and Ablation	Baseline	Modified Result	Reported Effect
YOLOv13-S: complete HyperACE–FullPAD configuration [15]	47.1 AP	48.0 AP	+0.9 AP points
YOLOv13-S: single versus complete FullPAD routing [15]	47.6–47.8 AP	48.0 AP	+0.2–0.4 AP points
YOLOv13-N: depthwise-separable blocks [15]	41.6 AP; 3.1 M parameters; 7.9 GFLOPs	41.6 AP; 2.5 M parameters; 6.4 GFLOPs	AP retained; parameters and FLOPs reduced
YOLO26-S: DFL removal at 640 pixels [16]	46.0 AP	46.3 AP	+0.3 AP points
YOLO26: Progressive Loss [16]	46.4 E2E AP	46.7 E2E AP	+0.3 AP points
YOLO26: STAL [16]	46.6 AP; 29.0 AP_S	46.8 AP; 29.6 AP_S	+0.2 AP and +0.6 AP_S
YOLO26: MuSGD [16]	47.0 AP after 600 epochs	47.4 AP after 500 epochs	+0.4 AP points; 16.7% fewer epochs

These results identify source-specific component effects but do not constitute a unified cross-generation ablation, because architectures and training protocols differ between the YOLOv13 and YOLO26 studies.

7.7. Error Analysis and Failure Modes

Systematic error analysis distinguishes false positives, missed detections, and localization errors [9–11]. Table 10 separates two within-study comparisons rather than pooling heterogeneous datasets. Panel A retains the six-model TSL hand-gesture benchmark [41]; Panel B adds the lightweight models from the Ships/Vessels evaluation in [10], a single-class task in which the target objects are relatively small. Model names follow the respective sources; matching scale labels does not imply equal parameter counts.

Table 10. Source-reported detection metrics and calculated precision and recall complements. (**A**) TSL hand-gesture benchmark [41], reported as means ± standard deviations over three random seeds. (**B**) Ships/Vessels benchmark [10], using the nano- and tiny-scale rows of the Ships/Vessels results figure in [10]. The panels are separate comparisons, and their values must not be pooled.

Model	Precision P	Recall R	mAP@[0.5:0.95]	$100(1-P)$ (%)	$100(1-R)$ (%)
Panel A. TSL hand-gesture benchmark [41]					
YOLOv8-N	0.961 ± 0.007	0.956 ± 0.016	0.829 ± 0.011	3.9 ± 0.7	4.4 ± 1.6
YOLOv9-T	0.975 ± 0.002	0.966 ± 0.004	0.876 ± 0.003	2.5 ± 0.2	3.4 ± 0.4
YOLOv10-N	0.965 ± 0.003	0.965 ± 0.010	0.859 ± 0.006	3.5 ± 0.3	3.5 ± 1.0
YOLO11-N	0.965 ± 0.002	0.958 ± 0.007	0.844 ± 0.003	3.5 ± 0.2	4.2 ± 0.7
YOLOv12-N	0.974 ± 0.007	0.960 ± 0.004	0.857 ± 0.003	2.6 ± 0.7	4.0 ± 0.4
YOLOv13-N	0.964 ± 0.008	0.953 ± 0.005	0.861 ± 0.005	3.6 ± 0.8	4.7 ± 0.5
Panel B. Ships/Vessels benchmark [10] (single class; small targets)					
YOLOv5un	0.635	0.532	—	36.5	46.8
YOLOv8n	0.655	0.533	—	34.5	46.7
YOLOv9t	0.647	0.516	—	35.3	48.4
YOLOv10n	0.584	0.487	—	41.6	51.3
YOLO11n	0.574	0.510	—	42.6	49.0
YOLOv12n	0.571	0.505	—	42.9	49.5

Notes: P, R, and mAP@[0.5:0.95] use a 0–1 scale. Complement columns are calculated percentages, not additional experiments. In Panel A, the standard deviations are multiplied by 100 under the linear transformation. Panel B values are calculated from the source's rounded P and R; uncertainty estimates are not reported for these rows. Panel B describes missed detections on a small-target dataset, not a COCO-area-filtered small-object miss rate. Neither panel supports extrapolation to untested versions such as YOLO26. In Panel B, "—" indicates that mAP is omitted from this comparison, not that it is unavailable in the source.

To quantify error-related trends, we calculate the complements of the published precision and recall values: $E_P = 100(1 - P)$ and $E_R = 100(1 - R)$. For count-based metrics, $P = TP/(TP + FP)$ and $R = TP/(TP + FN)$, so these complements represent the percentage of predictions that are false positives and the percentage of ground-truth objects that are missed, respectively. For class-averaged metrics, the complements retain the source's averaging convention and are not reconstructed pooled counts. E_P is not the background-normalized false-positive rate $FP/(FP + TN)$, and E_P and E_R do not share a denominator; they must not be summed into an error-category distribution. The comparisons retain the sources' evaluation operating points; a common numerical confidence threshold across all models is not established here.

In Panel A, the precision complement decreases from 3.9% for YOLOv8-N to 2.5% for YOLOv9-T but subsequently varies between 2.6% and 3.6%. The recall complement is lowest for YOLOv9-T (3.4%) and highest for YOLOv13-N (4.7%). Neither complement

decreases monotonically with version number, and these descriptive differences do not establish statistical significance.

In Panel B, the precision complement ranges from 34.5% (YOLOv8n) to 42.9% (YOLOv12n), and the recall complement ranges from 46.7% (YOLOv8n) to 51.3% (YOLOv10n). From YOLOv8n to YOLOv12n, the two complements increase by 8.4 and 2.8 percentage points, respectively, and the recall complement changes non-monotonically (46.7%, 48.4%, 51.3%, 49.0%, and 49.5%). For the six lightweight models listed in Panel B, the recall complement ranges from 46.7% to 51.3% under the source's evaluation protocol and does not decrease monotonically with version number. These are task-specific trends under the source protocol, not evidence that newer architectures inherently cause more errors; because the source does not report recall for a fixed object-area threshold, the results should not be generalized to all COCO-defined small objects.

The separate error-category analysis in [41] reports 3.4 $\pm$ 0.7% missed detections, 2.6 $\pm$ 0.5% false positives, and 11.2 $\pm$ 1.3% localization errors for YOLOv9-T only. These diagnostic rates are retained as reported and are presented separately from the calculated complements in Table 10 because their equivalence in definition and aggregation has not been established. Table 6 provides complementary COCO scale-specific AP evidence; APS is not recall and cannot be converted into a miss rate as 100—APS. Together, these results quantify error-related trends and size-specific accuracy while making explicit that a common size-conditioned error decomposition across all versions is not yet available. Future benchmarking should release such a decomposition for every checkpoint under a shared protocol.

7.8. Reproducibility, Safety, and Governance

Reproducible evaluation requires versioned datasets, environment hashes, published seeds, and accessible training/evaluation scripts. Where models operate in safety-critical contexts (autonomous driving [92,93], medical imaging [139,145,152]), governance artifacts—dataset documentation [207], model cards [208], bias and robustness audits, rollback and monitoring procedures—are not optional. Quantization and graph transformations must be validated with golden test suites so that numerical drift does not escape into production. The principles articulated here, combined with the compiled benchmarks reproduced from primary sources, are intended to support rigorous evaluation practice rather than to replace original experimental verification on the practitioner's specific deployment hardware, target domain, and operational requirements. Practitioners deploying any YOLO variant in production should validate primary-source benchmarks against their own hardware and data, particularly given the documented sensitivity of detection performance to hardware-specific operator implementations, library versions, and quantization toolchain choices.

7.9. Synthesis and Practical Guidance

The compiled results in Figures 18 and 19 provide a source-reported shortlist rather than a universal ranking. NMS-free families such as YOLOv10 [30] and YOLO26 [16] are candidates when post-processing simplicity is important; attention-augmented and hypergraph-based families are candidates when contextual representation is a central design concern [15,28,31]. These architectural motivations should not be converted into claims of robustness per millisecond or domain superiority without a controlled evaluation. Final selection should compare candidate checkpoints on the target data and hardware using accuracy, scale-specific AP, latency distributions under load, energy, memory, thermal behavior, calibration, and application-specific failure metrics.

7.10. Critical Discussion

Trade-offs and open caveats. Architectural evolution should not be interpreted as evidence of monotonic performance improvement. Anchor-free and one-to-one assignment strategies alter supervision and decoding, but their effects depend on the dataset, class imbalance, model scale, and training configuration. C2PSA, Area Attention, and HyperACE provide different mechanisms for contextual representation; however, controlled cross-model robustness comparisons under common occlusion, low-light, blur, or corruption protocols remain limited. NMS-free decoding removes a data-dependent post-processing stage but does not by itself establish lower frame-time variance or deterministic end-to-end latency. Evidence for the recent releases comprises primary reports [15,16], secondary analyses [43,45,46], independent domain-specific evaluations [41,42], and broader multi-version context [44]. A peer-reviewed uniform-protocol benchmark spanning YOLOv8n through YOLOv13n found that YOLOv9t, rather than the newest release, achieved the strongest overall accuracy on a gesture-detection task [41]. An independent document-domain evaluation found that YOLO11 outperformed the evaluated YOLO26 variants on form-element detection and that the larger YOLO26 variant showed a pronounced precision-recall imbalance [42]. Broader multi-version evidence likewise indicates that newer releases do not necessarily outperform earlier generations under all evaluation conditions [44]. Results for YOLOv13 and YOLO26 should therefore be interpreted as source-reported operating points rather than settled domain-general conclusions.

8. Future Directions and Open Challenges

The YOLO family has evolved from a streamlined convolutional detector into a modular ecosystem spanning gradient-aware aggregation, anchor-free decoupled heads, attention-augmented backbones, and—most recently—hypergraph-based correlation modeling [15] alongside deployment-centric architectural simplification [16]. Nevertheless, several important challenges and opportunities remain. Real-world deployments increasingly demand temporal reasoning, multimodal fusion, end-to-end training without hand-tuned post-processing, and principled methods for measuring robustness, uncertainty, fairness, and environmental impact. This section distills a research agenda that goes beyond incremental accuracy gains to emphasize capability, calibration, and governance under realistic constraints.

8.1. Convergence with Transformers and Foundation Models

Contemporary object-detection architectures are increasingly influenced by advances in transformer-based vision models [209]. Related backbone research includes ConvNeXt, which revisits convolutional design [210], MobileViT, which combines convolutional and transformer representations for mobile vision [211], and convolutional stems that improve the optimization of vision transformers [212]. These studies provide architectural context rather than evidence of direct adoption by a particular YOLO release. DETR [213] introduced end-to-end set prediction with bipartite matching, while later work used deformable attention [214] and other training refinements to improve optimization. RT-DETR [215] shows that Transformer-based detection can be designed for real-time constraints. Within the YOLO family, YOLOv12 uses region-partitioned Area Attention [28], YOLOv13 introduces hypergraph-based high-order correlation modeling [15], and YOLO26 combines a dual-head detector with DFL-free regression and deployment-oriented training components [16]. Hierarchical attention in Swin Transformer [216], token-based representation in ViT [217], and vision-language-action modeling [218] provide broader research context, but the cited sources do not establish direct causal lineage from these methods to a specific YOLO release. The architectural directions represented by YOLOv13 and

YOLO26 may provide useful foundations for future research in temporal, multimodal, and three-dimensional perception, but those extensions remain research directions rather than demonstrated model capabilities. MuSGD combines Muon-style orthogonalized updates with SGD and should not be described as an adaptive-learning-rate method [16].

8.2. Beyond Images: Video, Temporal Memory, and Multimodal/3D Perception

Many safety-critical deployment scenarios involve continuous video streams rather than isolated images. These scenarios also require handling re-identification, occlusion, and motion blur. Future YOLO architectures may increasingly incorporate temporal information as a central design consideration: lightweight temporal memory (recurrent keys/values, sliding-window attention), motion-aware label assignment, and end-to-end training that couples detection with association (joint detect-track) to reduce reliance on separately engineered integration code. Similar opportunities are likely to emerge in multimodal perception systems: Camera-LiDAR/radar fusion via bird's-eye-view (BEV) transformation can leverage cross-attention at the neck, trained under multi-view consistency. Extending detectors into 3D (or 2.5D BEV) should preserve YOLO's strengths: vectorized decoding, minimal post-processing, and on-device feasibility. Evaluation of these approaches should extend beyond mAP to ID switch rates, long-range AP, and degraded-condition performance across night/weather and compression [174,178,191].

8.3. End-to-End Decoding and Training Efficiency

NMS-free set-prediction approaches include Sparse R-CNN [219], while YOLOv10 [30] and YOLO26 [16] provide NMS-free YOLO-style inference paths. These designs remove a data-dependent post-processing stage and simplify deployment graphs, but the primary reports do not establish production validation or frame-to-frame latency stability across devices. YOLO26 jointly trains one-to-many and one-to-one branches and progressively increases emphasis on the inference-time branch [16]. Extending these strategies to related tasks such as instance segmentation, panoptic segmentation, and multi-object tracking remains an important area for future investigation requiring task-specific validation. Efficiency should continue to be investigated at the algorithmic level, through lighter attention and dynamic computation; at the architectural level, through hardware-aware search with latency or energy objectives [75,220]; and at the numerical level, through mixed precision, quantization, sparsity, and low-rank adaptation. Compiler and accelerator co-design should be evaluated for each target runtime. Removing NMS or DFL, favoring kernel-supported operators, and exposing static shape or range information may simplify execution, but their latency and accuracy effects must be measured. Future work may explore differentiable mechanisms for duplicate prediction control while maintaining deployment efficiency and reporting end-to-end median and tail latency, jitter, throughput under load, and export accuracy.

8.4. Data-Centric Learning: Self-Supervision, Semi-Supervision, and Synthetic Data

Architectural advances alone are unlikely to sustain long-term progress without corresponding improvements in data quality and diversity. Self-supervised pretraining (masked image modeling, teacher-student distillation) delivers stronger features for downstream detection at modest fine-tuning cost. Semi-supervised learning (confidence-aware pseudo-labels with strong/weak augmentation) offers linear data scaling when annotation is scarce. An increasingly important research direction involves simulation-based and generative data synthesis: rendering pipelines and diffusion models can generate rare or safety-critical scenarios (adverse weather, corner-case poses) with explicit control over pose, lighting, and occlusion. Recent work by Sapkota et al. [60] demonstrates LLM-generated synthetic data for agricultural detection achieving superior precision and recall over real-data-only

baselines. A rigorous program pairs synthesis with domain randomization and domain adaptation, evaluated under stress tests (ImageNet-C/COCO-C-style corruptions) to avoid overfitting to visually plausible synthetic images that do not adequately represent real-world conditions. Within the YOLO ecosystem, Mosaic and CutMix remain widely adopted augmentation strategies; one promising direction involves more sophisticated curriculum-based approaches to data governance and sample selection—sampling strategies that expose tails, balance scales, and correct label noise on the fly.

8.5. Robustness, Uncertainty, Fairness, and Privacy

Detection accuracy alone is insufficient to guarantee trustworthy deployment. Robustness should be systematically evaluated under corruption, occlusion, and distribution-shift conditions, with detectors instrumented to surface uncertainty—through energy-based OOD detection [221,222], Bayesian approximation [223], or deep ensembles [224]—that downstream planners can use. Conformal risk control and post hoc calibration can turn scores into operational guarantees. Open-set and OOD detection—recognizing unknown categories or out-of-scope environments—is central for long-lived systems; energy-based and contrastive methods offer practical baselines. Fairness assessment should extend beyond classification tasks and encompass detection-specific failure modes (as demonstrated by Gender Shades [225]) into detection contexts, where errors correlate with skin tone, clothing, sensor placement, and geographic representation [102]. Privacy considerations further motivate federated learning [167,183] with differential privacy [168] in regulated domains such as medical and financial applications, as well as on-device inference where feasible [16]. These constraints should be incorporated from the outset rather than addressed retrospectively.

Published evaluations document fault cases for the newest YOLO variants in the form of task-, domain-, and object-scale-specific performance limitations, showing that a newer release or a higher aggregate AP does not by itself guarantee robust behavior. (i) In an independent document-domain benchmark, YOLO11 outperformed the evaluated YOLO26 variants on form-element detection, and the larger YOLO26 variant showed a pronounced precision–recall imbalance [42]. (ii) Under a single uniform retraining protocol spanning YOLOv8-N through YOLOv13-N, YOLOv9-T rather than the newest release achieved the strongest overall accuracy on a gesture-detection task, and even this best-performing model retained measurable missed-detection, false-positive, and localization errors (Table 10, Panel A, for detection metrics; Section 7.7 for the separate error-category analysis) [41]. (iii) Across 33 datasets from 11 domains, the ODverse33 benchmark showed that later releases from YOLOv5 to YOLO11 do not uniformly outperform earlier ones, with YOLOv9 rather than a later version attaining the highest small-object accuracy [44]. (iv) In the YOLO26 technical report itself, YOLO26-S records a lower small-object AP than YOLOv12-S despite the highest overall AP among the listed S-scale YOLO variants (Section 7.4, Table 6) [16]. These findings motivate targeted robustness evaluation but do not, by themselves, establish performance under controlled image corruptions.

Several measurement practices can also make the robustness of the newest variants appear more adequate than the evidence supports. First, neither the YOLOv13 nor the YOLO26 primary report provides a controlled dense-scene or occlusion benchmark (Table 8), and no standardized corruption or adversarial benchmark results for these variants were identified in the reviewed literature; in the absence of such evidence, clean COCO accuracy alone should not be treated as evidence of robustness under these conditions. Second, robustness is sometimes attributed to architectural labels such as attention, hypergraph correlation, or NMS-free decoding without a controlled test; as discussed in Section 7.3, such mechanisms motivate hypotheses rather than evidence. Third, cross-source retention

comparisons that mix different checkpoints, corruption implementations, severity schedules, and metrics can produce unfair rankings (Section 7.3, Table 5). Fourth, aggregate summaries such as detection counts or mean confidence can conceal instance-level failures, because a missed object may be offset by an additional box that has no counterpart in the reference result, and the removal of low-confidence detections can leave the mean confidence of the remaining boxes unchanged or even higher.

Figure 20 illustrates these issues with a self-simulated example. The official YOLO26n checkpoint [16] was applied at a confidence threshold of 0.25 to a single street scene photographed by the authors, and four deterministic degradations corresponding to the stress conditions in Table 5 (low illumination, motion blur, sensor noise, and partial occlusion) were applied to the same image. The degradations comprised gamma-based darkening applied to normalized pixel intensities (γ = 2.8, gain = 0.45), a normalized 21-pixel horizontal motion-blur kernel, zero-mean additive Gaussian noise (σ = 32 on the 0–255 intensity scale; NumPy random seed = 0), and a uniform gray patch with intensity 114, centered at (0.50 W, 0.46 H) and spanning 32% of the original image width W and height H. Degradations were applied before inference. BGR images were passed to the Ultralytics YOLO prediction interface using the supplied yolo26n.pt checkpoint, which is configured for end-to-end detection. Apart from this threshold, the script used the prediction interface's default preprocessing and inference settings. Each degraded result was matched one-to-one to the clean-input detections by greedy class-aware IoU matching (IoU $\geq$ 0.5, highest overlap first). Low illumination and occlusion each caused the same small background object to be missed; low illumination, motion blur, and sensor noise each introduced a detection absent from the clean input; and motion blur and sensor noise reduced the mean confidence from 0.84 to 0.67 even though no clean-input detection was lost. Under low illumination, the detection count remained unchanged because the missed object was offset by a new box, and under partial occlusion, the mean confidence of the three remaining detections rounded to the same value as that of the clean input (0.84) despite one unmatched clean-input detection. Because the occlusion patch fully covers the small background object, panel (e) illustrates the effect of removing visual evidence rather than a failure to detect a still-visible target. This single-image simulation is a qualitative illustration rather than a controlled benchmark; it motivates, but does not replace, the per-instance, severity-controlled evaluation protocol outlined in Table 5.

8.6. Benchmarking, Reproducibility, and Sustainability

Sustainable scientific progress depends on transparent methodologies and comparable evaluation procedures. The field would benefit from standardized evaluation protocols specifically designed for YOLO-family detectors that fix resolution, precision, post-processing, and evaluation scripts across versions and variants—a need highlighted by the data-accuracy challenges documented in cross-source benchmark synthesis (Section 7). Reproducibility artifacts—dataset hashes, seeds, environment manifests, ablation matrices—should accompany any reported gain. Given the scale of modern training runs, sustainability considerations are becoming increasingly important as training and deployment costs continue to grow: report energy consumption and estimated emissions alongside mAP [226–228], and prefer Green AI recipes [229] that reduce wall-clock and hardware wattage without compromising robustness. In practice, the most impactful advances are likely to be those that improve accuracy and efficiency while reducing energy and deployment costs.



Figure 20. Illustrative self-simulated robustness fault cases of the official YOLO26n checkpoint [16] (confidence threshold 0.25) on a single street scene photographed by the authors: (**a**) clean input; (**b**) low illumination; (**c**) motion blur; (**d**) additive sensor noise; (**e**) partial occlusion. Each degraded result is matched to the clean-input detections by class-aware IoU matching (IoU $\geq$ 0.5): solid orange boxes denote detections, dashed blue boxes denote clean-input detections that are missed, and solid purple boxes denote detections absent from the clean input. Panel titles report the number of detections, their mean confidence, and the missed and new counts. The figure is a qualitative illustration of the failure modes and measurement pitfalls discussed in Section 8.5, not a controlled benchmark (see Table 5). Here, "missed" and "new" denote unmatched predictions relative to the clean-input reference, not ground-truth false negatives or false positives. Mean confidence is computed over all detections retained at the confidence threshold in each panel and is reported to two decimal places.

8.7. Outlook

A decade of YOLO research suggests that real-time performance and strong detection accuracy can often be jointly optimized under appropriate design and deployment conditions, although the achieved trade-off remains dependent on the evaluation protocol and deployment environment. Future progress is likely to depend on the careful integration of contextual modeling, end-to-end optimization, data-centric learning, robustness evaluation, governance, and energy-aware design. Under these conditions, the YOLO family is well positioned to remain an influential framework for real-time visual perception across a broad range of application domains rather than a universal default for every application.

9. Conclusions and Practical Recommendations

This review has examined the evolution of the YOLO family from a compact single-shot detector into a diverse ecosystem of real-time object-detection architectures spanning fourteen major generations (YOLOv1 through YOLO26) and numerous derivative variants. We traced the architectural evolution of the lineage (Section 3), synthesized its key innovations—including hypergraph-based correlation modeling [15] and deployment-centric simplification [16] (Section 4)—and distilled its evolutionary design principles through a why–how–what–where analysis (Section 5). We then surveyed domain-specific deployments across seven application areas with over one hundred concrete case studies spanning autonomous driving, manufacturing, agriculture, healthcare, robotics, multimedia, and cross-domain practice (Section 6), and compiled benchmark evidence from primary sources [9–11,15,16,46] with explicit attention to data accuracy and source provenance (Section 7). Building on the forward-looking agenda in Section 8, this concluding

section distills the core takeaways and offers concrete guidance for researchers and practitioners facing real deployment constraints.

9.1. Integrated Perspective

Across fourteen major generations introduced between 2016 and 2025, with formal literature coverage extending through the 2026 YOLO26 report, the YOLO ecosystem repeatedly revisited information routing, multi-scale fusion, assignment, decoding, and optimization. CSP, ELAN-family aggregation, attention, and hypergraph mechanisms represent different approaches to feature reuse and contextual modeling. Anchor-free and decoupled heads simplify several aspects of prediction, but their accuracy and convergence effects remain protocol-dependent.

The 2025 releases illustrate increasingly divergent design priorities within the YOLO ecosystem. YOLOv13 reports a 3.0-AP-point increase over YOLO11-N under its source comparison and introduces HyperACE and FullPAD [15]. YOLO26 reports up to 43% higher CPU throughput than YOLO11 under its source protocol and combines NMS-free inference with DFL-free regression, Progressive Loss, STAL, and MuSGD [16]. Although these operating points are promising, their broader practical significance will require continued independent validation across diverse datasets and deployment environments; they should not be interpreted as a globally Pareto-optimal family or as universal domain winners.

9.2. What the Evidence Supports in Practice

One of the clearest insights emerging from the surveyed literature is that detector performance depends on the interaction of architecture, optimization strategy, training data, assignment mechanisms, and deployment conditions rather than on any single architectural component. Source-specific ablations show benefits from combinations such as HyperACE with FullPAD in YOLOv13 [15] and L1 supervision, STAL, Progressive Loss, and MuSGD in YOLO26 [16]; these effects should remain attached to their original baselines and protocols rather than being generalized as universal percentage gains. From a deployment perspective, NMS-free architectures simplify inference pipelines, although their benefits remain dependent on runtime, hardware, and application context. Attention and hypergraph mechanisms provide explicit ways to model broader context, yet robustness under occlusion, low illumination, or corruption must be demonstrated through controlled stress tests. Likewise, NAS and quantization-aware design can target hardware constraints, but their benefit should be reported using paired accuracy, latency, energy, and calibration evidence on the selected platform. The practical recommendation is therefore procedural: practitioners should shortlist models by architectural fit, reproduce them under a common evaluation protocol, and make final decisions based on application-specific error costs and resource limits.

9.3. Deployment Playbook by Constraint

Because no single detector is optimal for all deployment scenarios, we summarize guidance as narrative recommendations rather than rigid prescriptions, reflecting the cross-domain patterns observed in Section 6 (Figure 17) and the quantitative trade-offs compiled in Section 7 (Figures 18 and 19). Each recommendation below identifies a starting shortlist rather than a final answer: as emphasized in Sections 7.8 and 7.9, deployment decisions should be finalized only after validating candidate models on the target hardware and data of the specific application.

Edge and mobile environments with stringent latency, power, or memory constraints: Shortlist compact models with deployment-oriented graphs, including YOLOv10 and YOLO26, together with NAS-generated or lightweight alternatives where appropriate [16,30,75]. Do not assume INT8 readiness from the family name alone: export each

candidate, calibrate it with representative data, and evaluate accuracy, median and tail latency, energy, memory, and thermal stability on the target device. Keep input at 480–640 where feasible; apply dynamic resolution and event-driven sampling under thermal pressure, and validate with latency-under-load tests rather than isolated FPS. Smart cameras, IoT vision, and mobile applications consistently benefit from this profile [193,230].

Cluttered scenes and small or distant targets: shortlist models whose architecture and source evidence are relevant to contextual and multi-scale representation, including YOLO11 [31], YOLOv12 [28], and YOLOv13 [15]. Use scale-aware sampling and augmentation only after validating them on the target data. Report AP_S, recall by object size, and crowd- or occlusion-specific metrics rather than relying only on aggregate AP. UAV-based wildlife monitoring [174,176], dense urban surveillance, and crowded traffic [92,93] exemplify settings in which these measurements are relevant. The final model should be selected through controlled evaluation rather than architectural expectation alone.

High-resolution inspection and defect-analysis applications: Prefer P6/P7 configurations or scaled variants with deeper PAN/SPP necks; tile at inference to preserve spatial detail. Evaluate miss rate and PPV/NPV at native resolution. Industrial defect detection [103,107], medical imaging [141,152], and high-resolution medical image analysis [162,165] follow this pattern.

Safety-critical and regulated deployment environments: Optimize for calibration and traceability as much as raw accuracy: report sensitivity at a fixed false-positive rate, provide model cards and audit logs, fix the exported graphs and their operator-set versions and numerical precision, and run reference tests after quantization or graph fusion. Favor on-premises or on-device inference where privacy dictates (medical [139,151], autonomous driving [92,102], financial). FDA 510(k) and CE-MDR compliance pathways apply for medical contexts.

Latest-generation selection (2025 models): YOLOv13 and YOLO26 represent different source-reported design priorities rather than universally dominant application choices. YOLOv13 introduces HyperACE and FullPAD for high-order correlation modeling and is a candidate where dense, occluded, or multi-scale scenes are central—urban autonomous driving [93], crowded surveillance, dense robotics, complex agricultural detection [60]—at the cost of moderate reported latency (roughly 8–15 ms on T4 for the larger scales) [15]. YOLO26 emphasizes DFL-free regression, NMS-free inference, branch-focused training, streamlined export (ONNX, TensorRT, CoreML, TFLite, OpenVINO), and up to 43% faster source-reported CPU inference, making it a candidate for production systems prioritizing deployment reliability—mobile robotics, IoT cameras, industrial inspection, automotive ECUs, CPU-only deployment [16]. Use these properties to define a shortlist, then select a scale and inference path through controlled application-specific evaluation; hybrid operation involving both families should be described as a prospective design unless experimentally validated.

9.4. Limits, Risks, and How to Mitigate Them

Despite substantial progress, several challenges continue to affect real-world deployment across application domains. Distribution shift arising from differences in geography, weather, sensors, institutions [151], cultivars, and production lines remains a persistent challenge; its magnitude varies by dataset and protocol and should not be summarized as a universal percentage. Mitigation strategies include explicit external-domain evaluation, adaptation, continual monitoring, federated learning where appropriate [167,183], and controlled corruption testing. Open-set and out-of-distribution scenarios continue to challenge most closed-world object-detection frameworks; detectors should expose uncertainty or abstention signals and may be paired with energy-based [221,222] or contrastive OOD

screening. Data quality issues—including label noise, class imbalance, and annotation inconsistency—should be addressed through documented curation, active learning, self- or semi-supervision, and carefully validated synthetic data [60]. Physical adversarial perturbations remain a separate deployment risk [102]. Without standardized protocols and energy or thermal measurements, apparent gains may conceal reproducibility and deployment regressions.

This review has several limitations. First, the comparative tables synthesize values from primary reports and follow-up studies without independently retraining all models on standardized hardware; differences in software, precision, batch size, and measurement procedure remain. Second, the core structured literature search covered publications through September 2025 and was supplemented by targeted updates completed in August 2026, including the formal YOLO26 report [16], independent domain-specific evaluations [41,42], secondary analysis [43], and broader multi-version benchmarking [44]. Third, the analysis prioritizes English-language publications. Fourth, the domain-evidence matrix in Figure 17 is an author-curated qualitative synthesis rather than a controlled comparative experiment. Consequently, all model-selection guidance requires revalidation on the target data, hardware, and operational setting.

9.5. Research Outlook and Actionable Next Steps

Several promising research directions emerge from the synthesis presented in this review. Hybrid CNN-attention architectures with tokenized pyramids and localized attention may support longer-range reasoning, but their accuracy and latency must be evaluated under a common protocol [15,28,31]. End-to-end decoding with one-to-one assignment is exemplified by YOLO26 [16], but production validation across devices and domains remains an open requirement. Future data-centric research should address rare-event coverage through self- and semi-supervision, simulation, diffusion-based synthesis [60], and curriculum-based sampling. Trustworthiness research should prioritize calibrated uncertainty, out-of-distribution handling, fairness audits [225], and privacy-preserving or federated learning where appropriate [167,183]. Open-vocabulary detection [202] and vision-language-action modeling [218] offer directions beyond fixed-class recognition. Future benchmark studies should report energy consumption and environmental impact in a more systematic and transparent manner [226–228].

9.6. Reproducibility and Resource Release

To facilitate reproducibility and independent verification, we recommend the release of the following artifacts: (i) training and evaluation scripts with fixed seeds; (ii) versioned dataset manifests and checksums; (iii) export graphs (ONNX/TensorRT/CoreML/TFLite) with op-set versions and precision flags; (iv) measurement harnesses for latency-under-load, energy per 1 k frames, and thermal rise; and (v) ablation matrices that isolate module contributions. Collectively, these resources can improve transparency, reproducibility, and cumulative progress within the object-detection community, accelerating both academic and industrial progress. The data-accuracy discipline applied in Section 7—reporting primary-source values with explicit citations and acknowledging cross-source synthesis limitations—offers a template for survey authors aiming to support rigorous downstream evaluation rather than perpetuate unverifiable performance claims. The comparison tables in this review retain citations to the underlying publications and official implementations.

9.7. Closing Remarks

The evolution of the YOLO family over the past decade demonstrates that real-time object detection can continue to advance in capability while maintaining practical deployment efficiency. The continued relevance of YOLO stems from its ability to continuously refine

feature representation, optimization strategies, inference mechanisms, and deployment-oriented design. The emergence of YOLOv13 and YOLO26 highlights two complementary directions—high-order contextual modeling and deployment-oriented architectural simplification—whose relative value remains application- and protocol-dependent. As evaluation standards continue to evolve toward greater reproducibility, robustness, transparency, sustainability, and hardware awareness, the YOLO family is well positioned to remain a foundational framework for real-time visual perception research and deployment across the application domains surveyed in this review.

Supplementary Materials: The following supporting information can be downloaded at: https://www.mdpi.com/article/10.3390/app162010386/s1, Table S1: Selected YOLO derivatives and related real-time detectors; Table S2: Quantitative Performance Gaps between Major YOLO Derivatives and the Official YOLO Release of the Same Era.

Author Contributions: Conceptualization and methodology, Y.-H.L. and W.-B.L.; software, Y.-H.L. and W.-B.L.; validation, W.-B.L.; investigation, W.-B.L.; data curation, Y.-H.L.; writing—original draft preparation, Y.-H.L.; writing—review and editing, Y.-H.L. and W.-B.L.; visualization, W.-B.L.; funding acquisition, W.-B.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by Wonkwang University in 2025.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: No new experimental datasets were created in this review. Except for the illustrative single-image simulation in Figure 20, quantitative results were taken from the cited primary papers, official documentation, and technical reports, or calculated from those reported values using the formulas stated in the corresponding tables. In particular, the precision and recall complements in Table 10 are arithmetic transformations of source-reported metrics and are not new experimental measurements. The source photograph and the script used to generate Figure 20 are available from the corresponding author upon reasonable request. The curated comparison tables used in the review are provided in the article and Supplementary Materials. Third-party datasets, model weights, and implementations remain subject to the licenses and access policies of their original providers.

Acknowledgments: During the preparation of this manuscript, the authors used Nano Banana 2 (Google Gemini image generation model) for rendering the schematic diagrams and converting the vector-based figures into publication-quality PNG images. This tool was used solely for visual drafting and figure rendering, not for generating results, producing benchmark values, or shaping the scholarly interpretation of the reviewed literature. The authors have reviewed, verified, and edited all generated outputs—including the accuracy of every figure against the underlying data reported in the cited primary sources—and take full responsibility for the content of this publication.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

ADAS	Advanced Driver-Assistance System
AP	Average Precision
AR/VR/MR	Augmented/Virtual/Mixed Reality
BN	Batch Normalization
C2f	Cross-Stage Partial Bottleneck with Two Convolutions (Faster)
C2PSA	C2 module with Partial Self-Attention
CBAM	Convolution Block Attention Module
CIB	Compact Inverted Block

CIoU	Complete Intersection over Union
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CSP	Cross-Stage Partial (network)
CT	Computed Tomography
DETR	Detection Transformer
DFL	Distribution Focal Loss
DIoU	Distance Intersection over Union
DS-Conv	Depthwise-Separable Convolution
ECU	Electronic Control Unit
E-ELAN	Extended Efficient Layer Aggregation Network
EIoU	Efficient Intersection over Union
FDA	U.S. Food and Drug Administration
FLOPs	Floating-Point Operations
FP16/FP32	16-bit/32-bit Floating-Point Precision
FPN	Feature Pyramid Network
FPS	Frames Per Second
FullPAD	Full-Pipeline Aggregation-and-Distribution
GAN	Generative Adversarial Network
GELAN	Generalized Efficient Layer Aggregation Network
GIoU	Generalized Intersection over Union
GPU	Graphics Processing Unit
HIPAA	Health Insurance Portability and Accountability Act
HOTA	Higher-Order Tracking Accuracy
HyperACE	Hypergraph-based Adaptive Correlation Enhancement
IDF1	Identification F1 Score
INT8	8-bit Integer Quantization
IoT	Internet of Things
IoU	Intersection over Union
LiDAR	Light Detection and Ranging
LLM	Large Language Model
LWIR	Long-Wave Infrared
mAP	mean Average Precision
MOT	Multi-Object Tracking
MOTA	Multiple Object Tracking Accuracy
MRI	Magnetic Resonance Imaging
MuSGD	Muon-Stochastic Gradient Descent (hybrid optimizer)
NAS	Neural Architecture Search
NMS	Non-Maximum Suppression
NPU	Neural Processing Unit
OBB	Oriented Bounding Box
OCT	Optical Coherence Tomography
ONNX	Open Neural Network Exchange
OOD	Out-of-Distribution
PANet	Path Aggregation Network
PCB	Printed Circuit Board
PGI	Programmable Gradient Information
PLC	Programmable Logic Controller
PRISMA	Preferred Reporting Items for Systematic Reviews and Meta-Analyses
ProgLoss	Progressive Loss Balancing
PTQ	Post-Training Quantization
QAT	Quantization-Aware Training
R-CNN	Region-based Convolutional Neural Network
R-ELAN	Residual Efficient Layer Aggregation Network

ROI	Region-Of-Interest
ROS	Robot Operating System
RPN	Region Proposal Network
SAR	Search and Rescue
SE	Squeeze-and-Excitation
SGD	Stochastic Gradient Descent
SimOTA	Simplified Optimal Transport Assignment
SLAM	Simultaneous Localization and Mapping
SoC	System on Chip
SPP/SPPF	Spatial Pyramid Pooling (Fast)
SSD	Single Shot MultiBox Detector
STAL	Small-Target-Aware Label Assignment
TAL	Task Alignment Learning
UAV	Unmanned Aerial Vehicle
ViT	Vision Transformer
VLM	Vision-Language Model
YOLO	You Only Look Once

References

1. Sun, Y.; Sun, Z.; Chen, W. The evolution of object detection methods. *Eng. Appl. Artif. Intell.* **2024**, *133*, 108458. [CrossRef]
2. Ali, M.L.; Zhang, Z. The YOLO Framework: A Comprehensive Review of Evolution, Applications, and Benchmarks in Object Detection. *Computers* **2024**, *13*, 336. [CrossRef]
3. Hussain, M. YOLOv1 to v8: Unveiling Each Variant—A Comprehensive Review of YOLO. *IEEE Access* **2024**, *12*, 42816–42833. [CrossRef]
4. Girshick, R.; Donahue, J.; Darrell, T.; Malik, J. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014. [CrossRef]
5. Girshick, R. Fast R-CNN. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 7–13 December 2015. [CrossRef]
6. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 1137–1149. [CrossRef] [PubMed]
7. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.-Y.; Berg, A.C. SSD: Single Shot MultiBox Detector. In *European Conference on Computer Vision*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2016; Volume 9905, pp. 21–37. [CrossRef]
8. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016. [CrossRef]
9. Terven, J.; Córdova-Esparza, D.-M.; Romero-González, J.-A. A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS. *Mach. Learn. Knowl. Extr.* **2023**, *5*, 1680–1716. [CrossRef]
10. Jegham, N.; Koh, C.Y.; Abdelatti, M.; Hendawi, A. YOLO Evolution: A Comprehensive Benchmark and Architectural Review of YOLOv12, YOLO11, and Their Previous Versions. *arXiv* **2025**, arXiv:2411.00201v4. [CrossRef]
11. 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. *Artif. Intell. Rev.* **2025**, *58*, 274. [CrossRef]
12. Zou, Z.; Chan, K.; Shi, Z.; Guo, Y.; Ye, J. Object Detection in 20 Years: A Survey. *Proc. IEEE* **2023**, *111*, 257–276. [CrossRef]
13. Liu, L.; Ouyang, W.; Wang, X.; Fieguth, P.; Chen, J.; Liu, X.; Pietikäinen, M. Deep Learning for Generic Object Detection: A Survey. *Int. J. Comput. Vis.* **2020**, *128*, 261–318. [CrossRef]
14. Trigka, M.; Dritsas, E. A Comprehensive Survey of Machine Learning Techniques and Models for Object Detection. *Sensors* **2025**, *25*, 214. [CrossRef] [PubMed]
15. Lei, M.; Li, S.; Wu, Y.; Hu, H.; Zhou, Y.; Zheng, X.; Ding, G.; Du, S.; Wu, Z.; Gao, Y. YOLOv13: Real-Time Object Detection with Hypergraph-Enhanced Adaptive Visual Perception. *arXiv* **2025**, arXiv:2506.17733. [CrossRef]
16. Jocher, G.; Qiu, J.; Liu, M.; Lyu, S.; Akyon, F.C.; Kalfaoglu, M.E. Ultralytics YOLO26: Unified Real-Time End-to-End Vision Models. *arXiv* **2026**, arXiv:2606.03748. [CrossRef]

17. Xu, J.-H.; Li, J.-P.; Zhou, Z.-R.; Lv, Q.; Luo, J. A Survey of the Yolo Series of Object Detection Algorithms. In Proceedings of the International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP), Chengdu, China, 14–16 December 2024. [CrossRef]
18. Lee, Y.-H.; Kim, H.-J. Comparative Analysis of YOLO Series (from V1 to V11) and Their Application in Computer Vision. *Korean Soc. Semicond. Disp. Technol.* **2024**, *23*, 190–198.
19. Redmon, J.; Farhadi, A. YOLO9000: Better, Faster, Stronger. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017. [CrossRef]
20. Redmon, J.; Farhadi, A. YOLOv3: An Incremental Improvement. *arXiv* **2018**, arXiv:1804.02767. [CrossRef]
21. Bochkovskiy, A.; Wang, C.-Y.; Liao, M. YOLOv4: Optimal Speed and Accuracy of Object Detection. *arXiv* **2020**, arXiv:2004.10934. [CrossRef]
22. Ultralytics. Explore Ultralytics YOLOv8. Available online: https://docs.ultralytics.com/models/yolov8/ (accessed on 5 March 2026).
23. Ge, Z.; Liu, S.; Wang, F.; Li, Z.; Sun, J. YOLOX: Exceeding YOLO Series in 2021. *arXiv* **2021**, arXiv:2107.08430. [CrossRef]
24. 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* **2022**, arXiv:2203.16250. [CrossRef]
25. Li, C.; Li, L.; Jiang, H.; Weng, K.; Geng, Y.; Li, L.; Ke, Z.; Li, Q.; Cheng, M.; Nie, W.-Q.; et al. YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications. *arXiv* **2022**, arXiv:2209.02976. [CrossRef]
26. Wang, C.-Y.; Bochkovskiy, A.; Liao, M. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. *arXiv* **2023**, arXiv:2207.02696. [CrossRef]
27. Ultralytics. YOLOv5: Pretrained Checkpoints. Available online: https://github.com/ultralytics/yolov5#pretrained-checkpoints (accessed on 10 September 2026).
28. Tian, Y.; Ye, Q.; Doermann, D. YOLOv12: Attention-Centric Real-Time Object Detectors. *arXiv* **2025**, arXiv:2502.12524. [CrossRef]
29. Wang, C.-Y.; Yeh, I.-H.; Liao, M. YOLOv9: Learning What You Want to Learn using Programmable Gradient Information. *arXiv* **2024**, arXiv:2402.13616. [CrossRef]
30. Wang, A.; Chen, H.; Liu, L.; Chen, K.; Lin, Z.; Han, J.; Ding, G. YOLOv10: Real-Time End-to-End Object Detection. *arXiv* **2024**, arXiv:2405.14458. [CrossRef]
31. Ultralytics. Ultralytics YOLO11. Available online: https://docs.ultralytics.com/models/yolo11/ (accessed on 5 July 2026).
32. Lin, T.-Y.; Goyal, P.; Girshick, R.; He, K.; Dollár, P. Focal Loss for Dense Object Detection. *IEEE Trans. Pattern Anal. Mach. Intell.* **2020**, *42*, 318–327. [CrossRef] [PubMed]
33. Tan, M.; Pang, R.; Le, Q.V. EfficientDet: Scalable and Efficient Object Detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020. [CrossRef]
34. Law, H.; Deng, J. CornerNet: Detecting Objects as Paired Keypoints. In *European Conference on Computer Vision*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2018; Volume 11218, pp. 765–781. [CrossRef]
35. Zhou, X.; Wang, D.; Krähenbühl, P. Objects as Points (CenterNet). *arXiv* **2019**, arXiv:1904.07850. [CrossRef]
36. Duan, K.; Bai, S.; Xie, L.; Qi, H.; Huang, Q.; Tian, Q. CenterNet: Keypoint Triplets for Object Detection. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Republic of Korea, 27 October–2 November 2019. [CrossRef]
37. Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; Fei-Fei, L. ImageNet: A large-scale hierarchical image database. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, 20–25 June 2009. [CrossRef]
38. Everingham, M.; Van Gool, L.; Williams, C.K.I.; Winn, J.; Zisserman, A. The Pascal Visual Object Classes (VOC) Challenge. *Int. J. Comput. Vis.* **2010**, *88*, 303–338. [CrossRef]
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. In *European Conference on Computer Vision*; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2014; Volume 8693, pp. 740–755. [CrossRef]
40. Moher, D.; Liberati, A.; Tetzlaff, J.; Altman, D.G. Preferred Reporting Items for Systematic Re- views and Meta-Analyses: The PRISMA Statement. *PLoS Med.* **2009**, *6*, e1000097. [CrossRef] [PubMed]
41. Gao, Y.; Luo, W.; Zhang, S.; Ahmad, N.S.; Wang, X.; Goh, P. Benchmarking YOLOv8 to YOLOv13 for Robust Hand Gesture Recognition in Human–Robot Interaction. *Sci. Rep.* **2025**, *15*, 40043. [CrossRef] [PubMed]
42. Baral, G.; Zhou, J. AutoFormBench: Benchmark Dataset for Automating Form Understanding. *arXiv* **2026**, arXiv:2603.29832. [CrossRef]
43. Chakrabarty, S. YOLO26: An Analysis of NMS-Free End to End Framework for Real-Time Object Detection. *arXiv* **2026**, arXiv:2601.12882. [CrossRef]
44. Jiang, T.; Zhong, Y. ODverse33: Is the New YOLO Version Always Better? A Multi Domain Benchmark from YOLO v5 to v11. *arXiv* **2025**, arXiv:2502.14314. [CrossRef]
45. Sapkota, R.; Harsha Cheppally, R.; Sharda, A.; Karkee, M. YOLO26: Key Architectural Enhancements and Performance Benchmarking for Real-Time Object Detection. *arXiv* **2025**, arXiv:2509.25164. [CrossRef]

46. Sapkota, R.; Karkee, M. Ultralytics YOLO Evolution: An Overview of YOLO26, YOLO11, YOLOv8 and YOLOv5 Object Detectors. *arXiv* **2025**, arXiv:2510.09653. [CrossRef]
47. Ioffe, S.; Szegedy, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *Proceedings of the International Conference on Machine Learning, Lille, France, 6–11 July 2015*; PMLR: Cambridge, MA, USA, 2015; pp. 448–456. Available online: https://proceedings.mlr.press/v37/ioffe15.html (accessed on 5 October 2026).
48. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition (ResNet). In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016. [CrossRef]
49. Lin, T.-Y.; Dollár, P.; Girshick, R.; He, K.; Hariharan, B.; Belongie, S. Feature Pyramid Networks for Object Detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017. [CrossRef]
50. Wang, C.-Y.; Liao, H.-Y.M.; Wu, Y.-H.; Chen, P.-Y.; Hsieh, J.-W.; Yeh, I.-H. CSPNet: A New Backbone that can Enhance Learning Capability of CNN. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, Seattle, WA, USA, 14–19 June 2020. [CrossRef]
51. He, K.; Zhang, X.; Ren, S.; Sun, J. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **2015**, *37*, 1904–1916. [CrossRef] [PubMed]
52. Liu, S.; Qi, L.; Qin, H.; Shi, J.; Jia, J. Path Aggregation Network for Instance Segmentation (PANet). In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018. [CrossRef]
53. Misra, D. Mish: A Self Regularized Non-Monotonic Activation Function. *arXiv* **2019**, arXiv:1908.08681. [CrossRef]
54. Zheng, Z.; Wang, P.; Ren, D.; Liu, W.; Ye, R.; Hu, Q. Enhancing Geometric Factors in Model Learning and Inference for Object Detection and Instance Segmentation. *IEEE Trans. Cybern.* **2022**, *52*, 8574–8586. [CrossRef] [PubMed]
55. de Oliveira Barreiros, M.; de Oliveira Dantas, D.; de Oliveira Silva, L.C.; Ribeiro, S.; Kardec Barros, A. Zebrafish tracking using YOLOv2 and Kalman filter. *Sci. Rep.* **2021**, *11*, 3219. [CrossRef] [PubMed]
56. Ding, X.; Zhang, X.; Ma, N.; Han, J.; Ding, G.; Sun, J. RepVGG: Making VGG-Style ConvNets Great Again. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*; IEEE: New York, NY, USA, 2021. [CrossRef]
57. He, K.; Gkioxari, G.; Dollár, P.; Girshick, R. Mask R-CNN. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017. [CrossRef]
58. Fan, S. Explaining YOLOv9 Object Detection with Attribution-Based Heatmap Methods. In Proceedings of the IEEE International Conference on Electrical, Automation and Computer Engineering, Changchun, China, 29–31 December 2024. [CrossRef]
59. Hidayatullah, P.; Tubagus, R. Yolov9 Architecture Explained. Available online: https://article.stunningvisionai.com/yolov9-architecture (accessed on 8 July 2026).
60. Sapkota, R.; Karkee, M. Improved YOLOv12 with LLM-Generated Synthetic Data for Enhanced Apple Detection and Benchmarking Against YOLOv11 and YOLOv10. *IFAC-PapersOnLine* **2025**, *59*, 239–244. [CrossRef]
61. Tian, Z.; Shen, C.; Chen, H.; He, T. FCOS: Fully Convolutional One-Stage Object Detection. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Republic of Korea, 27 October–2 November 2019. [CrossRef]
62. Zhang, H.; Cisse, M.; Dauphin, Y.N.; Lopez-Paz, D. Mixup: Beyond Empirical Risk Minimization. In *Proceedings of the International Conference on Learning Representations*; ICLR: Appleton, WI, USA, 2018. Available online: https://openreview.net/forum?id=r1Ddp1-Rb (accessed on 5 October 2026).
63. Ghiasi, G.; Cui, Y.; Srinivas, A.; Qian, R.; Lin, T.-Y.; Cubuk, E.D. Simple Copy-Paste is a Strong Data Augmentation Method for Instance Segmentation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021. [CrossRef]
64. Wang, C.-Y.; Yeh, I.-H.; Liao, H.-Y.M. You Only Learn One Representation: Unified Network for Multiple Tasks. *arXiv* **2021**, arXiv:2105.04206. [CrossRef]
65. Fang, Y.; Liao, B.; Wang, X.; Fang, J.; Qi, J.; Wu, R.; Niu, J.; Liu, W. You Only Look at One Sequence: Rethinking Transformer in Vision through Object Detection. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 26183–26195.
66. Ba, J.L.; Kiros, J.R.; Hinton, G.E. Layer Normalization. *arXiv* **2016**, arXiv:1607.06450. [CrossRef]
67. Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L.-C. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–23 June 2018. [CrossRef]
68. Howard, A.; Sandler, M.; Chu, G.; Chen, L.-C.; Chen, B.; Tan, M.; Wang, W.; Zhu, Y.; Pang, R.; Vasudevan, V.; et al. Searching for MobileNetV3. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Republic of Korea, 27 October–2 November 2019. [CrossRef]
69. Ma, N.; Zhang, X.; Zheng, H.-T.; Sun, J. ShuffleNet V2: Practical Guidelines for Efficient CNN Architecture Design. In *Proceedings of the European Conference on Computer Vision Lecture Notes in Computer Science*; European Conference on Computer Vision; Springer: Berlin/Heidelberg, Germany, 2018; Volume 11218, pp. 122–138. [CrossRef]
70. Han, K.; Wang, Y.; Tian, Q.; Guo, J.; Xu, C.; Xu, C. GhostNet: More Features from Cheap Operations. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 13–19 June 2020. [CrossRef]

71. Wang, C.-Y.; Bochkovskiy, A.; Liao, M. Scaled-YOLOv4: Scaling Cross Stage Partial Network. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021. [CrossRef]
72. Tan, M.; Le, Q. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In *Proceedings of the International Conference on Machine Learning, Long Beach, CA, USA, 9–15 June 2019*; PMLR: Cambridge, MA, USA, 2019; pp. 6105–6114. Available online: https://proceedings.mlr.press/v97/tan19a.html (accessed on 5 October 2026).
73. Tan, M.; Le, Q. EfficientNetV2: Smaller Models and Faster Training. In *Proceedings of the International Conference on Machine Learning (ICML), Virtual, 18–24 July 2021*; PMLR: Cambridge, MA, USA, 2021; pp. 10096–10106. Available online: https://proceedings.mlr.press/v139/tan21a.html (accessed on 5 October 2026).
74. Zoph, B.; Le, Q. Neural Architecture Search with Reinforcement Learning. In *Proceedings of the International Conference on Learning Representations, Toulon, France, 24–26 April 2017*; ICLR: Appleton, WI, USA, 2017. Available online: https://openreview.net/forum?id=r1Ue8Hcxg (accessed on 5 October 2026).
75. Deci, A.I. YOLO-NAS: A Next-Generation Object Detection Foundation Model. SuperGradients Official Documentation, 2023. Available online: https://github.com/Deci-AI/super-gradients/blob/master/YOLONAS.md (accessed on 8 July 2026).
76. Dao, T.; Fu, D.Y.; Ermon, S.; Rudra, A.; Ré, C. FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness. In *Advances in Neural Information Processing Systems*; Curran Associates, Inc.: Red Hook, NY, USA, 2022; Volume 35, Available online: https://openreview.net/forum?id=H4DqfPSibmx (accessed on 5 October 2026).
77. Dao, T. FlashAttention-2: Faster Attention with Better Parallelism and Work Partitioning. *arXiv* **2023**, arXiv:2307.08691. [CrossRef]
78. Hu, J.; Shen, L.; Sun, G. Squeeze-and-Excitation Networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018. [CrossRef]
79. Guha Roy, A.; Navab, N.; Wachinger, C. Concurrent Spatial and Channel 'Squeeze & Excitation' in Fully Convolutional Networks. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2018; Volume 11070, pp. 421–429. [CrossRef]
80. Woo, S.; Park, J.; Lee, J.-Y.; Kweon, I.S. CBAM: Convolutional Block Attention Module. In *European Conference on Computer Vision*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2018; Volume 11211, pp. 3–19. [CrossRef]
81. Zheng, Z.; Wang, P.; Liu, W.; Li, J.; Ye, R.; Ren, D. Distance-IoU Loss: Faster and Better Learning for Bounding Box Regression. *AAAI Conf. Artif. Intell.* **2020**, *34*, 12993–13000. [CrossRef]
82. Zhang, Y.-F.; Ren, W.; Zhang, Z.; Jia, Z.; Wang, L.; Tan, T. Focal and Efficient IOU Loss for Accurate Bounding Box Regression. *arXiv* **2022**, arXiv:2101.08158v2. [CrossRef]
83. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. *arXiv* **2015**, arXiv:1412.6980. [CrossRef]
84. Loshchilov, I.; Hutter, F. Decoupled Weight Decay Regularization (AdamW). In *Proceedings of the International Conference on Learning Representations, New Orleans, LA, USA, 6–9 May 2019*; ICLR: Appleton, WI, USA, 2019. Available online: https://openreview.net/forum?id=Bkg6RiCqY7 (accessed on 5 October 2026).
85. Ramachandran, P.; Zoph, B.; Le, Q.V. Searching for Activation Functions. *arXiv* **2017**, arXiv:1710.05941. [CrossRef]
86. Hendrycks, D.; Gimpel, K. Gaussian Error Linear Units (GELUs). *arXiv* **2016**, arXiv:1606.08415. [CrossRef]
87. Yun, S.; Han, D.; Chun, S.; Oh, S.J.; Yoo, Y.; Choe, J. CutMix: Regularization Strategy to Train Strong Classifiers with Localizable Features. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Republic of Korea, 27 October–2 November 2019. [CrossRef]
88. Cubuk, E.D.; Zoph, B.; Mane, D.; Vasudevan, V.; Le, Q.V. AutoAugment: Learning Augmentation Strategies from Data. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019. [CrossRef]
89. Geiger, A.; Lenz, P.; Urtasun, R. Are we ready for autonomous driving? The KITTI vision benchmark suite. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, 16–21 June 2012. [CrossRef]
90. Caesar, H.; Bankiti, V.; Lang, A.H.; Vora, S.; Liong, V.E.; Xu, Q.; Krishnan, A.; Pan, Y.; Baldan, G.; Beijbom, O. NuScenes: A multimodal dataset for autonomous driving. *arXiv* **2020**, arXiv:1903.11027. [CrossRef]
91. Sun, P.; Kretzschmar, H.; Dotiwalla, X.; Chouard, A.; Patnaik, V.; Tsui, P.; Guo, J.; Zhou, Y.; Chai, Y.; Caine, B.; et al. Scalability in Perception for Autonomous Driving: Waymo Open Dataset. *arXiv* **2020**, arXiv:1912.04838. [CrossRef]
92. Jia, X.; Tong, Y.; Qiao, H.; Li, M.; Tong, J.; Liang, B. Fast and accurate object detector for autonomous driving based on improved YOLOv5. *Sci. Rep.* **2023**, *13*, 9711. [CrossRef] [PubMed]
93. Joctum, A.; Kandiri, J. YOLO-APD: Enhancing YOLOv8 for Robust Pedestrian Detection on Complex Road Geometries. *Int. J. Comput. Trends Technol.* **2025**, *73*, 108. [CrossRef]
94. Farhat, W.; Ben Rhaiem, O.; Faiedh, H.; Souani, C. A novel architecture for efficient pedestrian detection in autonomous vehicles. *Connect. Sci.* **2025**, *37*, 2529261. [CrossRef]
95. Liu, Y.; Huang, Z.; Song, Q.; Bai, K. PV-YOLO: A lightweight pedestrian and vehicle detection model based on improved YOLOv8. *Digit. Signal Process.* **2024**, *156*, 104857. [CrossRef]

96. Zhang, J.; Zhang, W.; Tan, C.; Li, X.; Sun, Q. YOLO-PPA based Efficient Traffic Sign Detection for Cruise Control in Autonomous Driving. In *Proceedings of the 2024 International Conference on Industrial Automation and Robotics (IAR)*; Association for Computing Machinery: New York, NY, USA, 2024; pp. 8–13. [CrossRef]
97. Du, S.; Pan, W.; Li, N.; Dai, S.; Xu, B.; Liu, H.; Xu, C.; Li, X. TSD-YOLO: Small traffic sign detection based on improved YOLO v8. *IET Image Process.* **2024**, *18*, 2884–2898. [CrossRef]
98. Luo, Y.; Ci, Y.; Zhang, H.; Wu, L. A YOLOv8-CE-based real-time traffic sign detection and identification method for autonomous vehicles. *Digit. Transp. Saf.* **2024**, *3*, 82–91. [CrossRef]
99. Fang, S.; Chen, C.; Li, Z.; Zhou, M.; Wei, R. YOLO-ADual: A Lightweight Traffic Sign Detection Model for a Mobile Driving System. *World Electr. Veh. J.* **2024**, *15*, 323. [CrossRef]
100. Roszyk, K.; Nowicki, M.R.; Skrzypczyński, P. Adopting the YOLOv4 Architecture for Low-Latency Multispectral Pedestrian Detection in Autonomous Driving. *Sensors* **2022**, *22*, 1082. [CrossRef] [PubMed]
101. Ahn, H.; Lee, Y.-H. Vehicle Classification and Tracking Based on Deep Learning. *J. Web Eng.* **2022**, *21*, 1283–1294. [CrossRef]
102. Eykholt, K.; Evtimov, I.; Fernandes, E.; Li, B.; Rahmati, A.; Xiao, C.; Prakash, A.; Kohno, T.; Song, D. Robust Physical-World Attacks on Deep Learning Visual Classification. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–23 June 2018; pp. 1625–1634. [CrossRef]
103. Wei, Z.; Yang, F.; Zhong, K.; Yao, L. PCB-YOLO: Enhancing PCB surface defect detection with coordinate attention and multi-scale feature fusion. *PLoS ONE* **2025**, *20*, e0323684. [CrossRef] [PubMed]
104. Wang, Y.; Huang, J.; Sharid Kayes Dipu, M.; Zhao, H.; Gao, S.; Zhang, H.; Lv, P. YOLO-RLC: An Advanced Target-Detection Algorithm for Surface Defects of Printed Circuit Boards Based on YOLOv5. *Comput. Mater. Contin.* **2024**, *80*, 4973–4995. [CrossRef]
105. Chen, W.; Ke, X.; Meng, S. Small defect detection in printed circuit boards based on the multiscale edge strengthening and an improved YOLOv10. *Sci. Rep.* **2025**, *15*, 36445. [CrossRef] [PubMed]
106. Yu, S.; Pan, F.; Zhang, X.; Zhou, L.; Zhang, L.; Wang, J. A lightweight detection algorithm of PCB surface defects based on YOLO. *PLoS ONE* **2025**, *20*, e0320344. [CrossRef] [PubMed]
107. Lu, J.; Yu, M.; Liu, J. Lightweight strip steel defect detection algorithm based on improved YOLOv7. *Sci. Rep.* **2024**, *14*, 13267. [CrossRef] [PubMed]
108. Chu, Y.; Yu, X.; Rong, X. A Lightweight Strip Steel Surface Defect Detection Network Based on Improved YOLOv8. *Sensors* **2024**, *24*, 6495. [CrossRef] [PubMed]
109. Zhang, Z.; Zhong, G.; Ding, P.; He, J.; Zhang, J.; Zhu, C. ELS-YOLO: Efficient Lightweight YOLO for Steel Surface Defect Detection. *Electronics* **2025**, *14*, 3877. [CrossRef]
110. Huang, X.; Zhu, J.; Huo, Y. SSA-YOLO: An improved YOLO for Hot-Rolled Strip Steel Surface Defect Detection. *IEEE Trans. Instrum. Meas.* **2024**, *73*, 5040017. [CrossRef]
111. Zuo, Y.; Wang, J.; Song, J. Application of YOLO Object Detection Network in Weld Surface Defect Detection. In Proceedings of the IEEE Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER), Jiaxing, China, 27–31 July 2021; pp. 704–710. [CrossRef]
112. Qin, Y.; Zhao, S. Lightweight YOLO-Based Online Inspection Architecture for Cup Rupture Detection in the Strip Steel Welding Process. *Machines* **2026**, *14*, 40. [CrossRef]
113. 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. [CrossRef] [PubMed]
114. Zhou, L.; Ma, B.; Dong, Y.; Yin, Z.; Lu, F. DCFE-YOLO: A novel fabric defect detection method. *PLoS ONE* **2025**, *20*, e0314525. [CrossRef] [PubMed]
115. Liu, G.; Zhang, S.; Wang, L.; Li, X.; Li, G. Research on mechanical automatic food packaging defect detection model based on improved YOLOv5 algorithm. *PLoS ONE* **2025**, *20*, e0321971. [CrossRef] [PubMed]
116. Vijayakumar, A.; Vairavasundaram, S.; Koilraj, J.A.S.; Rajappa, M.; Kotecha, K.; Kulkarni, A. Real-time visual intelligence for defect detection in pharmaceutical packaging. *Sci. Rep.* **2024**, *14*, 18811. [CrossRef] [PubMed]
117. U.S. Food and Drug Administration. Part 11, Electronic Records; Electronic Signatures—Scope and Application: Guidance for Industry. U.S. Department of Health and Human Services, Food and Drug Administration. 2003. Available online: https://www.fda.gov/regulatory-information/search-fda-guidance-documents/part-11-electronic-records-electronic-signatures-scope-and-application (accessed on 25 August 2026).
118. European Commission. EudraLex—The Rules Governing Medicinal Products in the European Union, Volume 4, Annex 11: Computerised Systems. European Commission, Brussels. 2011. Available online: https://health.ec.europa.eu/system/files/2016-11/annex11_01-2011_en_0.pdf (accessed on 25 August 2026).
119. Zhu, F.; Zhang, W.; Wang, S.; Jiang, B.; Feng, X.; Zhao, Q. Apple-Harvesting Robot Based on the YOLOv5-RACF Model. *Biomimetics* **2024**, *9*, 495. [CrossRef] [PubMed]
120. Sun, X. Enhanced tomato detection in greenhouse environments: A lightweight model based on S-YOLO with high accuracy. *Front. Plant Sci.* **2024**, *15*, 1451018. [CrossRef] [PubMed]

121. Ma, Z.; Dong, N.; Gu, J.; Cheng, H.; Meng, Z.; Du, X. STRAW-YOLO: A detection method for strawberry fruits targets and key points. *Comput. Electron. Agric.* **2025**, *230*, 109853. [CrossRef]
122. Liu, G.; Nouaze, J.C.; Lyonel, P.; Mbouembe, T.; Kim, J.H. YOLO-Tomato: A Robust Algorithm for Tomato Detection Based on YOLOv3. *Sensors* **2020**, *20*, 2145. [CrossRef] [PubMed]
123. Liu, S.; Xue, J.; Zhang, T.; Lv, P.; Qin, H.; Zhao, T. Research progress and prospect of key technologies of fruit target recognition for robotic fruit picking. *Front. Plant Sci.* **2024**, *15*, 1423338. [CrossRef] [PubMed]
124. Sapkota, R.; Ahmed, D.; Churuvija, M.; Karkee, M. Immature Green Apple Detection and Sizing in Commercial Orchards using YOLOv8 and Shape Fitting Techniques. *IEEE Access* **2024**, *12*, 43436–43452. [CrossRef]
125. Dang, F.; Chen, D.; Lu, Y.; Li, Z. YOLOWeeds: A novel benchmark of YOLO object detectors for multi-class weed detection in cotton production systems. *Comput. Electron. Agric.* **2023**, *205*, 107655. [CrossRef]
126. Saini, P.; Nagesh, D. CottonWeeds: Empowering precision weed management through deep learning and comprehensive dataset. *Crop Prot.* **2024**, *181*, 106675. [CrossRef]
127. Belissent, N.; Peña, J.M.; Mesías-Ruiz, G.A.; Shawe-Taylor, J.; Pérez-Ortiz, M. Transfer and zero-shot learning for scalable weed detection and classification in UAV images. *Knowl.-Based Syst.* **2024**, *292*, 111586. [CrossRef]
128. Radočaj, D.; Radočaj, P.; Plaščak, I.; Jurišić, M. Evolution of Deep Learning Approaches in UAV-Based Crop Leaf Disease Detection: A Web of Science Review. *Appl. Sci.* **2025**, *15*, 778. [CrossRef]
129. Gupta, C.; Gill, N.S.; Gulia, P.; Duhan, S.; Alduaiji, N.; Shukla, P.K.; Dwivedi, A. Deep vision in agriculture: Assessing the function of YOLO in the classification of plant leaf diseases (PLDs). *BioData Min.* **2025**, *18*, 91. [CrossRef] [PubMed]
130. Zhou, M.; Xing, R.; Han, D.; Qi, Z.; Li, G. PDT: UAV Target Detection Dataset for Pests and Diseases Tree. *arXiv* **2024**, arXiv:2409.15679. [CrossRef]
131. Rajagukguk, R.A.; Lee, S.-Y.; Park, J.-Y.; Daniel, K.F.; Lee, C.-R.; Chen, Z.; Liu, D.; Norton, T.; Park, J.; Hong, S.-W. Deep learning for visual animal monitoring (detection, tracking, pose estimation, and behavior classification): A comprehensive review. *Smart Agric. Technol.* **2025**, *12*, 101539. [CrossRef]
132. Lee, J.-H.; Choi, Y.H.; Lee, H.-S.; Park, H.J.; Hong, J.S.; Lee, J.H.; Sa, S.J.; Kim, Y.M.; Kim, J.E.; Jeong, Y.D.; et al. Enhanced Swine Behavior Detection with YOLOs and a Mixed Efficient Layer Aggregation Network in Real Time. *Animals* **2024**, *14*, 3375. [CrossRef] [PubMed]
133. Jahn, S.; Schmidt, G.; Bachmann, L.; Kammerer, R.; Louton, H.; Schütz, A.K.; Homeier-Bachmann, T. Individual behavior tracking of heifers by using object detection algorithm YOLOv4. *Front. Anim. Sci.* **2024**, *5*, 1499253. [CrossRef]
134. Tong, L.; Fang, J.; Wang, X.; Zhao, Y. Research on Cattle Behavior Recognition and Multi-Object Tracking Algorithm Based on YOLO-BoT. *Animals* **2024**, *14*, 2993. [CrossRef] [PubMed]
135. Guarnido-Lopez, P.; Ramirez-Agudelo, J.-F.; Denimal, E.; Benaouda, M. Programming and Setting Up the Object Detection Algorithm YOLO to Determine Feeding Activities of Beef Cattle: A Comparison between YOLOv8m and YOLOv10m. *Animals* **2024**, *14*, 2821. [CrossRef] [PubMed]
136. Luo, Y.; Lin, K.; Xiao, Z.; Lv, E.; Wei, X.; Li, B.; Zeng, H.L.Z. PBR-YOLO: A lightweight piglet multi-behavior recognition algorithm based on improved YOLOv8. *Smart Agric. Technol.* **2025**, *10*, 100785. [CrossRef]
137. Paneru, B.; Dhungana, A.; Dahal, S.; Ritz, C.W.; Kim, W.; Liu, T.; Chai, L. Computer vision models for precision poultry farming: A narrative review of behavioral and welfare monitoring studies. *Poult. Sci.* **2026**, *105*, 106887. [CrossRef] [PubMed]
138. Mahmood, A.; Anwar, W.; Sattar, H.; Hassan, S.R.; Sheraz, M.; Chuah, T.C. Deep learning framework using UAV imagery for multi-disease detection in cereal crops. *Sci. Rep.* **2025**, *16*, 3339. [CrossRef] [PubMed]
139. Wan, J.-J.; Zhu, P.-C.; Chen, B.-L.; Yu, Y.-T. A semantic feature enhanced YOLOv5-based network for polyp detection from colonoscopy images. *Sci. Rep.* **2024**, *14*, 15478. [CrossRef] [PubMed]
140. Bernal, J.; Sánchez, F.J.; Fernández-Esparrach, G.; Gil, D.; Rodrí Guez, C.; Vilariño, F. WM- DOVA maps for accurate polyp highlighting in colonoscopy: Validation vs. saliency maps from physicians. *Comput. Med. Imaging Graph.* **2015**, *43*, 99–111. [CrossRef] [PubMed]
141. Lalinia, M.; Sahafi, A. Colorectal polyp detection in colonoscopy images using YOLO-V8 network. *Signal Image Video Process.* **2023**, *18*, 2047–2058. [CrossRef]
142. Zare, O.; Beigzadeh, M.; Bzuayene, A.A.; Arslan, E.; Haghvirdizadeh, H.; Yaser Bozorgi Rad, S.; Abbasi, A.; Hassannataj Joloudari, J.; Gaftandzhieva, S. LightAttn-YOLO-V8: An efficient colorectal polyp detection with lightweight attention mechanism based on YOLO. *Sci. Rep.* **2026**, *16*, 18100. [CrossRef] [PubMed]
143. Huang, J.; Zeng, J.; Peng, J.; Lyu, M.; Liu, S. Endoscopic colorectal polyp detection based on improved YOLOv8. In Proceedings of the International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC), Orlando, FL, USA, 15–19 July 2024.
144. Wang, X.; Peng, Y.; Lu, L.; Lu, Z.; Bagheri, M.; Summers, R.M. ChestX-ray8: Hospital-scale Chest X-ray Database and Benchmarks on Weakly-Supervised Classification and Localization of Common Thorax Diseases. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017. [CrossRef]

145. Xie, Y.; Zhu, B.; Jiang, Y.; Zhao, B.; Yu, H. Diagnosis of pneumonia from chest X-ray images using YOLO deep learning. *Front. Neurorobotics* **2025**, *19*, 1576438. [CrossRef] [PubMed]
146. Zhang, X.; Liu, L.; Yang, X.; Liu, L.; Peng, W. NSEC-YOLO: Real-time lesion detection on chest X-ray with adaptive noise suppression and global perception aggregation. *J. Radiat. Res. Appl. Sci.* **2025**, *18*, 101281. [CrossRef]
147. Mao, M.; Hong, C.; Zhang, Y.; Huang, H.; Chu, J.; Fu, L. Research on lung nodule detection in X-ray plain films based on improved YOLOv12 model. *Sci. Rep.* **2026**, *16*, 11667. [CrossRef] [PubMed]
148. Goel, L.; Patel, P. Improving YOLOv6 using advanced PSO optimizer for weight selection in lung cancer detection and classification. *Multimed. Tools Appl.* **2024**, *83*, 78059–78092. [CrossRef]
149. Ji, Z.; Zhao, J.; Liu, J.; Zeng, X.; Zhang, H.; Zhang, X.; Ganchev, I. ELCT-YOLO: An Efficient One-Stage Model for Automatic Lung Tumor Detection based on CT Images. *Mathematics* **2023**, *11*, 2344. [CrossRef]
150. Bilic, P.; Christ, P.; Vorontsov, E.; Chlebus, G.; Chen, H.; Dou, Q.; Fu, C.-W.; Han, X.; Heng, P.-A.; Hesser, J.; et al. The Liver Tumor Segmentation Benchmark (LiTS). *arXiv* **2019**, arXiv:1901.04056. [CrossRef]
151. Elmujtba, M.; Essa, A. Diagnostic accuracy of AI in chest radiography for pneumonia and lung cancer: A meta-analysis. *Eur. J. Radiol. Open* **2025**, *15*, 100701. [CrossRef] [PubMed]
152. Chen, A.; Lin, D.; Gao, Q. Enhancing brain tumor detection in MRI images using YOLO-NeuroBoost model. *Front. Neurol.* **2024**, *15*, 1445882. [CrossRef] [PubMed]
153. Han, W.; Dong, X.; Wang, G.; Ding, Y.; Yang, A. Application and improvement of YOLO11 for brain tumor detection in medical images. *Front. Oncol.* **2025**, *15*, 1643208. [CrossRef] [PubMed]
154. Fahaad Almufareh, M.; Imran, M.; Khan, A.; Humayun, M.; Asim, M. Automated Brain Tumor Segmentation and Classification in MRI using YOLO-based Deep Learning. *IEEE Access* **2024**, *12*, 16189–16207. [CrossRef]
155. Bouhafra, S.; El Bahi, H. Deep Learning Approaches for Brain Tumor Detection and Classification Using MRI Images (2020 to 2024): A Systematic Review. *J. Imaging Inform. Med.* **2025**, *38*, 1403–1433. [CrossRef] [PubMed]
156. Kang, M.; Ting, C.-M.; Ting, F.F.; Phan, R.C.-W. BGF-YOLO: Enhanced YOLOv8 with Multiscale Attentional Feature Fusion for Brain Tumor Detection. *arXiv* **2023**, arXiv:2309.12585. [CrossRef]
157. Codella, N.; Rotemberg, V.; Tschandl, P.; Celebi, M.E.; Dusza, S.; Gutman, D.; Helba, B.; Kalloo, A.; Liopyris, K.; Marchetti, M.; et al. Skin Lesion Analysis Toward Melanoma Detection 2018: A Challenge Hosted by the International Skin Imaging Collaboration (ISIC). *arXiv* **2019**, arXiv:1902.03368. [CrossRef]
158. Ünver, H.M.; Ayan, E. Skin Lesion Segmentation in Dermoscopic Images with Combination of YOLO and GrabCut Algorithm. *Diagnostics* **2019**, *9*, 72. [CrossRef] [PubMed]
159. Huang, N.-C.; Mukundan, A.; Karmakar, R.; Syna, S.; Chang, W.-Y.; Wang, H.-C. Novel Snapshot-Based Hyperspectral Conversion for Dermatological Lesion Detection via YOLO Object Detection Models. *Bioengineering* **2025**, *12*, 714. [CrossRef] [PubMed]
160. Da Costa Nascimento, J.J.; Marques, A.G.; Rodrigues, Y.O.A.; Severiano, G.F.B.; Rodrigues, I.d.S.; Dourado, C., Jr.; Souza, L.F.D.F. Health of Things Melanoma Detection System—Detection and segmentation of melanoma in dermoscopic images applied to edge computing using deep learning and fine-tuning models. *Front. Commun. Netw.* **2024**, *5*, 1376191. [CrossRef]
161. Coşkun, D.; Karaboğa, D.; Baştürk, A.; Akay, B.; Nalbantoğlu, Ö.U.; Doğan, S.; Paçal, İ.; Altın Karagöz, M. A Comparative Study of YOLO Models and a Transformer-Based YOLOv5 Model for Mass Detection in Mammograms. *Turk. J. Electr. Eng. Comput. Sci.* **2023**, *31*, 1294–1313. [CrossRef]
162. Lewis, J.E.; Shebelut, C.W.; Drumheller, B.R.; Zhang, X.; Shanmugam, N.; Attieh, M.; Horwath, M.C.; Khanna, A.; Smith, G.H.; Gutman, D.A.; et al. An Automated Pipeline for Differential Cell Counts on Whole-Slide Bone Marrow Aspirate Smears. *Mod. Pathol.* **2023**, *36*, 100003. [CrossRef] [PubMed]
163. Zhang, D.; Bu, Y.; Chen, Q.; Cai, S.; Zhang, Y. TW-YOLO: An Innovative Blood Cell Detection Model Based on Multi-Scale Feature Fusion. *Sensors* **2024**, *24*, 6168. [CrossRef] [PubMed]
164. Ardelean, A.-I.; Ardelean, E.-R.; Marginean, A. Can YOLO Detect Retinal Pathologies? A Step Towards Automated OCT Analysis. *Diagnostics* **2025**, *15*, 1823. [CrossRef]
165. Rizzieri, N.; Dall'Asta, L.; Ozoliņš, M. Diabetic Retinopathy Features Segmentation without Coding Experience with Computer Vision Models YOLOv8 and YOLOv9. *Vision* **2024**, *8*, 48. [CrossRef]
166. Yuan, Z.; Yan, Y.; Sonka, M.; Yang, T. Large-Scale Robust Deep AUC Maximization: A New Surrogate Loss and Empirical Studies on Medical Image Classification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*; IEEE: New York, NY, USA, 2021. [CrossRef]
167. McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; Agüeray Arcas, B. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*; PMLR: Cambridge, MA, USA, 2017; pp. 1273–1282. Available online: https://proceedings.mlr.press/v54/mcmahan17a.html (accessed on 5 October 2026).
168. Dwork, C.; Roth, A. The Algorithmic Foundations of Differential Privacy. *Found. Trends Theor. Comput. Sci.* **2014**, *9*, 211–487. [CrossRef]

169. Wang, P.; Zhao, J. SOD-YOLO: Enhancing YOLO-Based Detection of Small Objects in UAV Imagery. *arXiv* **2025**, arXiv:2507.12727. [CrossRef]
170. Han, Y.; Wang, C.; Luo, H.; Wang, H.; Chen, Z.; Xia, Y.; Yun, L. LRDS-YOLO enhances small object detection in UAV aerial images with a lightweight and efficient design. *Sci. Rep.* **2025**, *15*, 22627. [CrossRef] [PubMed]
171. Huang, M.; Mi, W.; Wang, Y. EDGS-YOLOv8: An improved YOLOv8 Lightweight UAV Detection Model. *Drones* **2024**, *8*, 337. [CrossRef]
172. Khalili, B.; Smyth, A.W. SOD-YOLOv8—Enhancing YOLOv8 for Small Object Detection in Aerial Imagery and Traffic Scenes. *Sensors* **2024**, *24*, 6209. [CrossRef] [PubMed]
173. Xue, C.; Xia, Y.; Wu, M.; Chen, Z.; Cheng, F.; Yun, L. EL-YOLO: An efficient and lightweight low-altitude aerial objects detector for onboard applications. *Expert Syst. With Appl.* **2024**, *256*, 124848. [CrossRef]
174. Mou, C.; Liu, T.; Zhu, C.; Cui, X. WAID: A Large-Scale Dataset for Wildlife Detection with Drones. *Appl. Sci.* **2023**, *13*, 10397. [CrossRef]
175. Naik, H.; Yang, J.; Das, D.; Crofoot, M.C.; Rathore, A.; Sridhar, V.H. BuckTales: A Multi-UAV Dataset for Multi-Object Tracking and Re-Identification of Wild Antelopes. In *Advances in Neural Information Processing Systems 37*; Datasets and Benchmarks Track; NeurIPS: Red Hook, NY, USA, 2024. [CrossRef]
176. Naidu, A.P.; Gosalia, H.; Gakhar, I.; Rathore, S.S.; Didwania, K.; Verma, U. DEAL-YOLO: Drone-based Efficient Animal Localization using YOLO. *arXiv* **2025**, arXiv:2503.04698. [CrossRef]
177. Roy, A.M.; Bhaduri, J.; Kumar, T.; Raj, K. WilDect-YOLO: An efficient and robust computer vision-based accurate object localization model for automated endangered wildlife detection. *Ecol. Inform.* **2023**, *75*, 101919. [CrossRef]
178. Gao, P.; Zhong, D.; Qi, Q.; Ling, C.; Qiu, C.; Wang, B.; Du, X.; Gao, M. FDM-YOLO: Real-time small-target UAV wildlife detection via attention-guided cross-modality fusion. *Ecol. Inform.* **2026**, *95*, 103697. [CrossRef]
179. Mukhiddinov, M.; Abdusalomov, A.B.; Cho, J. A Wildfire Smoke Detection System Using Unmanned Aerial Vehicle Images Based on the Optimized YOLOv5. *Sensors* **2022**, *22*, 9384. [CrossRef] [PubMed]
180. Yu, P.; Wei, W.; Li, J.; Du, Q.; Wang, F.; Zhang, L.; Li, H.; Yang, K.; Yang, X.; Zhang, N.; et al. Fire-PPYOLOE: An Efficient Forest Fire Detector for Real-Time Wild Forest Fire Monitoring. *J. Sens.* **2024**, *2024*, 2831905. [CrossRef]
181. 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. [CrossRef]
182. Li, C.; Du, Y.; Zhang, X.; Wu, P. YOLOGX: An improved forest fire detection algorithm based on YOLOv8. *Front. Environ. Sci.* **2025**, *12*, 1486212. [CrossRef]
183. Aldughayfiq, B.; Alrashdi, I.; Tawfeek, M.A.; Jamel, L.; Shaban, W.M.; Talaat, F.M. Real-Time Forest Fire Detection, Monitoring, and Mitigation to Safeguard Humanity Using YOLO and Federated Learning. *Int. J. Distrib. Sens. Netw.* **2025**, *2025*, 4707734. [CrossRef]
184. Shen, X.; Yuan, G.; Wang, H.; Fu, X. Unsupervised clustering optimization-based efficient attention in YOLO for underwater object detection. *Artif. Intell. Rev.* **2025**, *58*, 219. [CrossRef]
185. Fu, J.; Tian, Y. Improved YOLOv7 Underwater Object Detection Based on Attention Mechanism. *Eng. Lett.* **2024**, *32*, 1377–1384.
186. Tomas, J.P.; Cacayan, J.; Francisco, J.; Reonal, P.M.; Tamara, A. Enhanced Aquaculture Monitoring: Real-time Fish Detection Underwater with YOLOv7. In Proceedings of the International Conference on Machine Learning and Machine Intelligence (MLMI), Osaka, Japan, 2–4 August 2024; 2024. [CrossRef]
187. Lin, Y.; Zhang, Y.; Yang, D.; Zhou, Y.; Wang, L. DLFE-YOLO: An enhanced framework for underwater object detection based on YOLOv8. *Intell. Mar. Technol. Syst.* **2025**, *3*, 37. [CrossRef]
188. Gong, R.; Xu, J.; Zheng, Z.; Zhang, D. YOLO-Starfish: Fish object detection learning complex underwater features. *Sci. Rep.* **2026**, *16*, 13964. [CrossRef] [PubMed]
189. Kabir, S.; Rifti, I.A.; Tabib, H.M.S.; Rahman, M.; Sadi, S.I.; Adil, H.; Rumi, A.M.S.; Haider, C.M.R. SpectraSentinel: LightWeight Dual-Stream Real-Time Drone Detection, Tracking and Payload Identification. *arXiv* **2025**, arXiv:2507.22650. [CrossRef]
190. Dhanuka, E.; Dhavakumar, P. Automated Email and SMS Notification System for Unauthorized Entry using YOLOv9. In Proceedings of the International Conference on Communication and Signal Processing, Melmaruvathur, India, 12–14 April 2024. [CrossRef]
191. Lenhard, T.; Weinmann, A.; Jäger, S.; Koch, T. YOLO-FEDER FusionNet: A Novel Deep Learning Architecture for Drone Detection. *arXiv* **2024**, arXiv:2406.11641. [CrossRef]
192. Mur-Artal, R.; Montiel, J.M.M.; Tardos, J.D. ORB-SLAM: A Versatile and Accurate Monocular SLAM System. *IEEE Trans. Robot.* **2015**, *31*, 1147–1163. [CrossRef]
193. Ultralytics. Deploy YOLO26 on Mobile & Edge with ExecuTorch. Ultralytics Documentation. Available online: https://docs.ultralytics.com/integrations/executorch/ (accessed on 8 July 2026).

194. Thakur, M.M.; Dhanvij, S.; Jawade, S.; Satdeve, A.; Khaire, P. A Review on Real Time AI-Driven Virtual Hand Gesture Control. *Int. Res. J. Eng. Technol.* **2025**, *12*. Available online: https://www.irjet.net/archives/V12/i12/IRJET-V12I1234.pdf (accessed on 8 July 2026).
195. Gkoutzios, S.; Vrigkas, M. A Multi-User Virtual Reality Game Based on Gesture Recognition from RGB Information. *Comput. Animat. Virtual Worlds* **2025**, *36*, e70063. [CrossRef]
196. Fumić, M.; Livada, Č. Bridging the Physical and Virtual Worlds: A Hand Tracking Gesture Recognition System for XR Applications. In *International Conference on Organization and Technology of Maintenance*; Lecture Notes in Networks and Systems; Springer: Cham, Switzerland, 2024; Volume 866, pp. 53–74. [CrossRef]
197. Hossam, A.; Ramadan, A.; Magdy, M.; Abdelwahab, R.; Ashraf, S.; Mohamed, Z. Revolutionizing Retail Analytics: Advancing Inventory and Customer Insight with AI. *arXiv* **2024**, arXiv:2405.00023. [CrossRef]
198. Yeşilyurt, M.E.; Güzel, M.S.; Sezer, E. A Comparison of Machine Learning Methods for Queue Length Detection. *Commun. Fac. Sci. Univ. Ank. Ser. A2-A3 Phys. Sci. Eng.* **2024**, *66*, 132–139. [CrossRef]
199. Shoman, M.; Aboah, A.; Morehead, A.; Duan, Y.; Daud, A.; Adu-Gyamfi, Y. A Region-Based Deep Learning Approach to Automated Retail Checkout. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), New Orleans, LA, USA, 19–20 June 2022. [CrossRef]
200. Horng, S.-J.; Huang, P.-S. Building Unmanned Store Identification Systems Using YOLOv4 and Siamese Network. *Appl. Sci.* **2022**, *12*, 3826. [CrossRef]
201. Cioppa, A.; Giancola, S.; Deliege, A.; Kang, L.; Zhou, X.; Cheng, Z.; Ghanem, B.; Van Droogenbroeck, M. SoccerNet-Tracking: Multiple Object Tracking Dataset and Benchmark in Soccer Videos. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), New Orleans, LA, USA, 19–20 June 2022. [CrossRef]
202. Cheng, T.; Song, L.; Ge, Y.; Liu, W.; Wang, X.; Shan, Y. YOLO-World: Real-Time Open-Vocabulary Object Detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 16–22 June 2024. [CrossRef]
203. Wang, Z.; Bovik, A.C.; Sheikh, H.R.; Simoncelli, E.P. Image Quality Assessment: From Error Visibility to Structural Similarity (SSIM). *IEEE Trans. Image Process.* **2004**, *13*, 600–612. [CrossRef] [PubMed]
204. Wu, Y.; Kirillov, A.; Massa, F.; Lo, W.-Y.; Girshick, R. Detectron2. GitHub Repository. 2019. Available online: https://github.com/facebookresearch/detectron2 (accessed on 8 July 2026).
205. Chen, K.; Wang, J.; Pang, J.; Cao, Y.; Xiong, Y.; Li, X.; Sun, S.; Feng, W.; Liu, Z.; Xu, J.; et al. MMDetection: Open MMLab Detection Toolbox and Benchmark. *arXiv* **2019**, arXiv:1906.07155. [CrossRef]
206. Wu, Y.; He, K. Group Normalization. In *Proceedings of the European Conference on Computer Vision (ECCV)*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2018; Volume 11217, pp. 3–19. [CrossRef]
207. Gebru, T.; Morgenstern, J.; Vecchione, B.; Wortman Vaughan, J.; Wallach, H.; Daumé, H., III; Crawford, K. Datasheets for Datasets. *Commun. ACM* **2021**, *64*, 86–92. [CrossRef]
208. Mitchell, M.; Wu, S.; Zaldivar, A.; Barnes, P.; Vasserman, L.; Hutchinson, B.; Spitzer, E.; Inioluwa, D.; Raji, I.D.; Gebru, T. Model Cards for Model Reporting. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*; Association for Computing Machinery: New York, NY, USA, 2019; pp. 220–229. [CrossRef]
209. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is All You Need. *Adv. Neural Inf. Process. Syst. (NeurIPS)* **2017**, *30*, 5998–6008. Available online: https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html (accessed on 5 October 2026).
210. Liu, Z.; Mao, H.; Wu, C.-Y.; Feichtenhofer, C.; Darrell, T.; Xie, S. A ConvNet for the 2020s (ConvNeXt). In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA, 27 September 2022. [CrossRef]
211. Mehta, S.; Rastegari, M. MobileViT: Light-weight, General-purpose, and Mobile-friendly Vision Transformer. In *Proceedings of the International Conference on Learning Representations (ICLR), Virtual, 25–29 April 2022*; ICLR: Appleton, WI, USA, 2022. Available online: https://openreview.net/forum?id=vh-0sUt8HlG (accessed on 5 October 2026).
212. Xiao, T.; Singh, M.; Mintun, E.; Darrell, T.; Dollár, P.; Girshick, R. Early Convolutions Help Transformers See Better. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 30392–30400.
213. Carion, N.; Massa, F.; Synnaeve, G.; Usunier, N.; Kirillov, A.; Zagoruyko, S. End-to-End Object Detection with Transformers. In *European Conference on Computer Vision*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2020; Volume 12346, pp. 213–229. [CrossRef]
214. Zhu, X.; Su, W.; Lu, L.; Li, B.; Wang, X.; Dai, J. Deformable DETR: Deformable Transformers for End-to-End Object Detection. In *Proceedings of the International Conference on Learning Representations, Vienna, Austria, 3–7 May 2021*; ICLR: Appleton, WI, USA, 2021. Available online: https://openreview.net/forum?id=gZ9hCDWe6ke (accessed on 5 October 2026).
215. Zhao, Y.; Lv, W.; Xu, S.; Wei, J.; Wang, G.; Dang, Q.; Liu, Y.; Chen, J. DETRs Beat YOLOs on Real-time Object Detection. *arXiv* **2023**, arXiv:2304.08069. [CrossRef]

216. Liu, Z.; Lin, Y.; Cao, Y.; Hu, H.; Wei, Y.; Zhang, Z. Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, QC, Canada, 10–17 October 2021. [CrossRef]
217. Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.R.; Gelly, S.; et al. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale (ViT). In *Proceedings of the International Conference on Learning Representations, Vienna, Austria, 3–7 May 2021*; ICLR: Appleton, WI, USA, 2021. Available online: https://openreview.net/forum?id=YicbFdNTTy (accessed on 5 October 2026).
218. Ma, Y.; Song, Z.; Zhuang, Y.; Hao, J.; King, I. A Survey on Vision-Language-Action Models for Embodied AI. *IEEE Trans. Neural Netw. Learn. Syst.* **2026**, *37*, 3031–3051. [CrossRef] [PubMed]
219. Sun, P.; Zhang, R.; Jiang, Y.; Kong, T.; Xu, C.; Zhan, W.; Tomizuka, M.; Li, L.; Yuan, Z.; Wang, C.; et al. Sparse R-CNN: End-to-End Object Detection with Learnable Proposals. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 20–25 June 2021. [CrossRef]
220. He, Y.; Lin, J.; Liu, Z.; Wang, H.; Li, L.-J.; Han, S. AMC: AutoML for Model Compression and Acceleration on Mobile Devices. In *Proceedings of the European Conference on Computer Vision*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2018; Volume 11211, pp. 815–832. [CrossRef]
221. Chalapathy, R.; Chawla, S. Deep Learning for Anomaly Detection: A Survey. *arXiv* **2019**, arXiv:1901.03407. [CrossRef]
222. Hendrycks, D.; Gimpel, K. A Baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks. *arXiv* **2018**, arXiv:1610.02136. [CrossRef]
223. Gal, Y.; Ghahramani, Z. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. In *Proceedings of the International Conference on Machine Learning*; PMLR: Cambridge, MA, USA, 2016; pp. 1050–1059. Available online: https://proceedings.mlr.press/v48/gal16.html (accessed on 5 October 2026).
224. Lakshminarayanan, B.; Pritzel, A.; Blundell, C. Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles. *arXiv* **2017**, arXiv:1612.01474. [CrossRef]
225. Buolamwini, J.; Gebru, T. Gender Shades: Intersectional Accuracy Disparities in Commercial Gender Classification. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*; PMLR: Cambridge, MA, USA, 2018; pp. 77–91. Available online: https://proceedings.mlr.press/v81/buolamwini18a.html (accessed on 5 October 2026).
226. Strubell, E.; Ganesh, A.; McCallum, A. Energy and Policy Considerations for Deep Learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, Florence, Italy, 28 July–2 August 2019*; Association for Computational Linguistics: Stroudsburg, PA, USA, 2019; pp. 3645–3650. [CrossRef]
227. Lacoste, A.; Luccioni, A.; Schmidt, V.; Dandres, T. Quantifying the Carbon Emissions of Machine Learning. *arXiv* **2019**, arXiv:1910.09700. [CrossRef]
228. Patterson, D.; Gonzalez, J.; Le, Q.; Liang, C.; Munguia, L.-M.; Rothchild, D.; So, D.; Texier, M.; Dean, J. Carbon Emissions and Large Neural Network Training. *arXiv* **2021**, arXiv:2104.10350. [CrossRef]
229. Schwartz, R.; Dodge, J.; Smith, N.A.; Etzioni, O. Green AI. *Commun. ACM* **2020**, *63*, 54–63. [CrossRef]
230. Shuvo, M.M.H.; Islam, S.K.; Cheng, J.; Morshed, B.I. Efficient Acceleration of Deep Learning Inference on Resource-Constrained Edge Devices: A Review. *Proc. IEEE* **2023**, *111*, 42–91. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.