YOLO27: An Overview of Dual-Scale Architecture and Query-Based Real-Time Computer Vision

Ranjan Sapkota
Cornell University, Ithaca, NY, USA
rs2672@cornell.edu

September 18, 2026

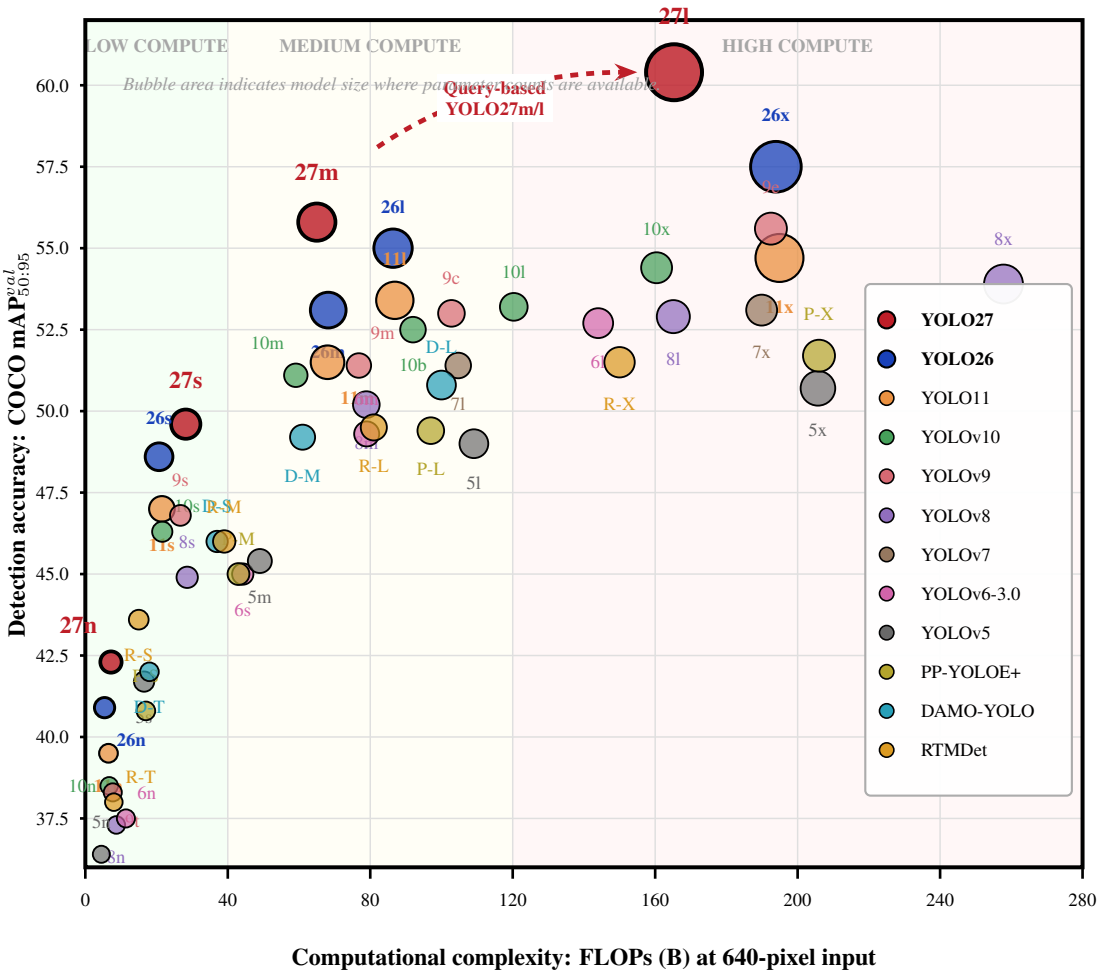

Figure 1: Accuracy–complexity landscape of YOLO27 and representative real-time detectors, comparing COCO mAP$^{val}_{50:95}$ against FLOPs; bubble size additionally represents model parameterization where consistently reported. YOLO27 results are preliminary.

ABSTRACT

This study presents a comprehensive overview of Ultralytics YOLO27 (also called YOLOv27), examining its scale-adaptive architecture, detection mechanisms, preliminary performance, and implications for real-time computer vision. Unlike conventional YOLO families that largely scale common architectures across model sizes, YOLO27 introduces capacity-dependent architectural specialization. YOLO27n/s employ streamlined convolutional neural networks with strengthened high-resolution features and dual-scale prediction, supporting one-to-many detection with non-maximum suppression (NMS) and optional one-to-one NMS-free inference. Conversely, YOLO27m/l adopt query-based transformer decoding for native end-to-end NMS-free detection, while YOLO27l further integrates UltraViT with deep-stage self-attention for enhanced global-context modeling. Foreground alignment supervision reduces discrepancies between dense training and one-to-one inference. Preliminary COCO benchmarks report 42.3–60.4 mAP at 640-pixel resolution and 0.62–2.32 ms TensorRT 11 FP16 latency, while YOLO27l reaches 61.2 mAP at 800 pixels. Beyond object detection, this overview examines instance and semantic segmentation, depth estimation, classification, pose estimation, oriented detection, deployment trade-offs, current limitations, and future opportunities spanning multimodal vision, edge AI, autonomous systems, and robotic perception, providing a perspective on YOLO27's evolving role within real-time intelligent visual perception systems.

1 Introduction

Object detection and segmentation constitute foundational components of modern computer vision, enabling intelligent systems to identify, localize, classify, and delineate multiple objects from complex visual observations, including individual images and continuous video streams [1, 2, 3]. These capabilities provide the perceptual foundation through which machines transform raw visual information into structured representations of objects, their spatial locations, categories, and, in the case of segmentation, pixel-level boundaries. Consequently, advances in detection and segmentation directly influence the ability of autonomous systems to understand dynamic environments and support subsequent tasks such as tracking, scene understanding, navigation, manipulation, and decision-making. Their practical significance extends across autonomous driving, robotics, precision agriculture, aerial intelligence, industrial inspection, surveillance, medical imaging, smart manufacturing, and edge artificial intelligence (AI), where perception models must operate under increasingly demanding real-world conditions [4, 5]. In these applications, achieving high predictive accuracy alone is insufficient; practical perception systems must simultaneously balance computational complexity, memory requirements, inference latency, robustness, and deployment efficiency, particularly when operating on resource-constrained edge devices or within time-critical autonomous platforms.

Among the numerous object-detection paradigms introduced during the past decade, the You Only Look Once (YOLO) family, whose architectural evolution is summarized in Table 1, has emerged as one of the most influential frameworks for real-time visual perception [6, 7, 8]. The original YOLO formulation fundamentally reframed object detection as a unified prediction problem, integrating object localization and category classification within a single inference pipeline rather than relying on computationally expensive multi-stage processing [6]. Subsequent YOLO generations have progressively incorporated improvements in backbone design, multi-scale feature representation, feature fusion, training strategies, detection heads, optimization, and deployment-oriented inference [7, 8]. This continuing architectural evolution has enabled YOLO-based detectors to maintain a strong emphasis on the accuracy–efficiency trade-off while expanding their relevance from conventional object detection toward broader real-time visual perception and increasingly complex autonomous vision systems.

Since the original YOLO formulation by Redmon et al. [6], successive generations (YOLOv2 [9], YOLOv3 [10], YOLOv4 [11], YOLOv5 (Source Link), YOLOv6 [12] , YOLOv7 [13], YOLOv8 (Source Link), YOLOv9 [14], YOLOv10 [15], YOLO11(Source Link), YOLOv12 [16], YOLOv13[17], and YOLO26 [18]) have introduced deeper feature extractors, multi-scale prediction, cross-stage feature aggregation, anchor-free detection, improved label assignment, attention mechanisms, end-to-end prediction, and deployment-oriented optimization [19]. These improvements have progressively transformed YOLO from a conventional single-stage detector into a broad computer vision ecosystem supporting object detection, instance segmentation, image classification, pose estimation, oriented object detection, and other dense prediction tasks. Modern YOLO development has therefore increasingly focused not only on maximizing benchmark accuracy, but also on finding favorable trade-offs among accuracy, latency, memory consumption, computational complexity, export compatibility, and deployment flexibility.

YOLO26 represented an important transition toward deployment-oriented real-time vision. Its design emphasized simplified regression, optional end-to-end prediction, reduced post-processing dependence, and efficient deployment across heterogeneous computing platforms [18]. However, scaling a single architectural design from extremely compact

Table 1: Representative architectural transitions leading to YOLO26 and YOLO27. The table distinguishes broader YOLO-named research developments from the recent Ultralytics model lineage.

Model	Year	Primary Paradigm	Representative Architectural Contribution	Role in Evolution
YOLOv1 [6]	2016	Single-stage CNN	Direct image-to-box/class regression	Established unified real-time detection
YOLOv2 [9]	2017	Anchor-based CNN	Anchor prediction and multi-scale training	Improved scale robustness
YOLOv3 [10]	2018	Multi-scale CNN	Darknet-53 and hierarchical prediction	Strengthened multi-resolution detection
YOLOv4 [11]	2020	CSP-based CNN	Feature aggregation and extensive training refinements	Improved accuracy–efficiency optimization
YOLOv6 [12]	2022	Efficient CNN	Deployment-oriented architecture and training	Hardware-oriented real-time detection
YOLOv7 [13]	2022	CNN	Trainable architectural and optimization refinements	Improved real-time efficiency
YOLOv9 [14]	2024	CNN/GELAN	Improved gradient and information propagation	Representation-oriented optimization
YOLOv10 [15]	2024	End-to-end CNN	Consistent dual assignments and NMS-free prediction	Transition toward direct inference
YOLOv12 [16]	2025	Attention-centric	Greater use of self-attention in real-time detection	CNN–attention convergence
YOLOv13 [17]	2025	Higher-order interaction	Enhanced relational feature modeling	Beyond pairwise feature interactions
YOLO26 [18]	2026	DFL-free dual-head CNN	NMS-free inference, Progressive Loss, STAL, MuSGD	Deployment and optimization simplification
YOLO27 (Source)	**2026**	**Scale-adaptive CNN/query**	**Dual-scale CNN n/s; query-based m/l; UltraViT in l**	**Architectural specialization by model scale**

edge models to high-capacity server models inevitably introduces compromises [19]. An architecture optimized for a nano model operating on embedded hardware may not necessarily represent the most effective design for a large model running on a high-performance GPU. Conversely, architectures capable of sophisticated global reasoning may impose unnecessary computational overhead when applied to highly constrained real-time systems.

Ultralytics YOLO27(Source Link) addresses this issue through a substantially different design philosophy. Rather than applying one homogeneous detector architecture across all model sizes, YOLO27 introduces a *scale-dependent dual-architecture strategy*. The compact YOLO27n and YOLO27s detection models retain a streamlined convolutional neural network (CNN) architecture optimized for real-time inference, whereas YOLO27m and YOLO27l transition toward query-based detection using transformer decoding and native non-maximum suppression (NMS)-free prediction. This separation represents an important architectural evolution: model scaling is no longer limited to varying network depth and width, but also determines the fundamental prediction paradigm used by the detector.

The compact YOLO27n and YOLO27s models introduce four closely coupled mechanisms: (i) dual-scale detection rather than conventional three-scale prediction, (ii) increased representational capacity in early high-resolution feature stages, (iii) foreground alignment supervision during training, and (iv) support for both conventional one-to-many prediction with NMS and one-to-one NMS-free prediction. The medium and large models follow a different pathway. YOLO27m pairs a YOLO26-style convolutional backbone with a query-based transformer decoder, whereas YOLO27l combines the same general feature pyramid network/path aggregation network (FPN/PAN) neck philosophy with an UltraViT backbone containing self-attention in its deepest stage. The latter enables stronger global-context modeling while maintaining the multi-scale feature aggregation principles associated with YOLO-style detectors.

Importantly, YOLO27 does not expose this architectural diversity as additional complexity to the end user. The four detection scales operate through the same Ultralytics `YOLO` Python interface, and the appropriate training, validation, prediction, and export pathways are selected automatically according to the loaded model. This abstraction enables researchers and developers to move between compact CNN detectors and query-based transformer detectors without fundamentally modifying application code.

According to the available technical documentation, YOLO27 detection models span approximately 42.3-60.4 $\text{mAP}_{50:95}$ on the COCO validation benchmark at 640-pixel input resolution, with reported TensorRT 11 FP16 latency ranging from approximately 0.62 to 2.32 ms on an NVIDIA RTX PRO 6000 GPU. YOLO27l further reaches a reported 61.2 mAP at an increased 800-pixel input resolution, representing the first Ultralytics model reported to exceed 60 mAP on COCO (Source Link). These initials results suggest that YOLO27 is designed not simply as an incremental accuracy improvement, but as an exploration of how architecture itself should change according to model scale and deployment requirements.

Beyond object detection, the planned YOLO27 ecosystem extends toward instance segmentation, semantic segmentation, monocular depth estimation, image classification, pose/keypoint estimation, and oriented object detection. Detection, segmentation, pose, and oriented bounding-box variants are also intended to integrate with multi-object tracking through the existing prediction pipeline. Consequently, YOLO27 represents a broader transition from a detector-centric framework toward a unified real-time visual perception platform.

This paper provides insights into performance and key architectural enhancements introduced in YOLO27, with particular emphasis on its scale-dependent architecture, dual-scale detection mechanism, small-object representation,

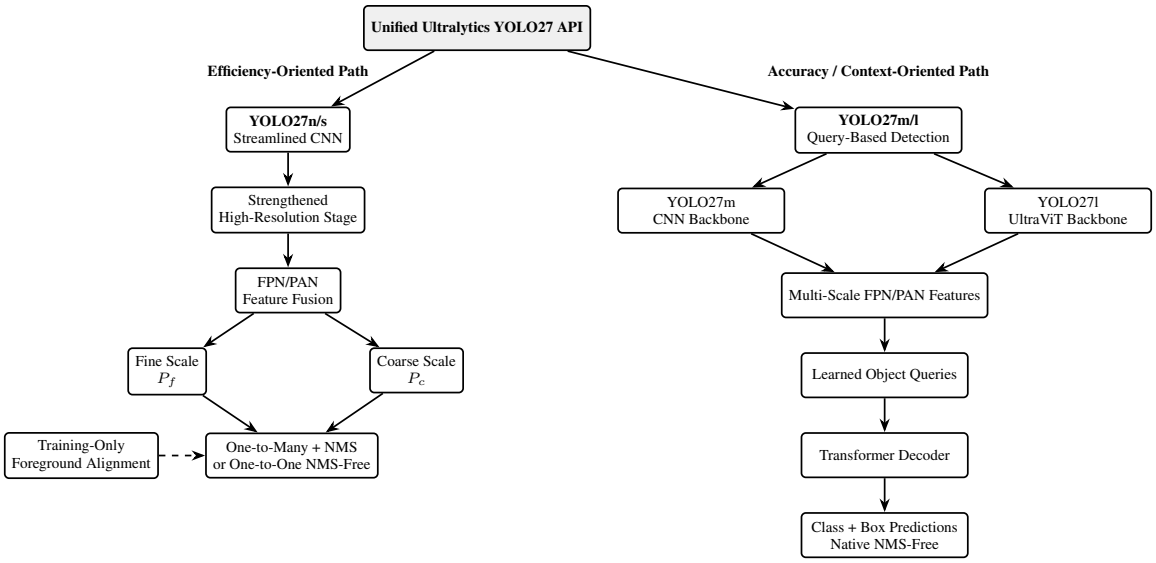

Figure 2: Scale-adaptive dual architecture of YOLO27. Compact YOLO27n/s detectors employ streamlined CNNs, strengthened high-resolution features, dual-scale prediction, foreground-alignment supervision, and either dense NMS-based or one-to-one NMS-free inference. YOLO27m/l instead use query-based transformer decoding; YOLO27m retains a CNN backbone whereas YOLO27l incorporates UltraViT with deep-stage self-attention.

foreground alignment supervision, query-based prediction, NMS-free inference, UltraViT backbone, multi-task capabilities, performance characteristics, and implications for edge and robotic AI. The paper further discusses the potential significance of YOLO27 for future real-time computer vision systems in which CNN efficiency and transformer-based contextual reasoning coexist within a unified model family.

2 Overview of the YOLO27 Model Family

YOLO27 introduces four principal model scales for object detection: YOLO27n, YOLO27s, YOLO27m, and YOLO27l. Unlike earlier YOLO families in which different scales were primarily obtained by compound adjustments of depth, width, and channel capacity, YOLO27 divides detection into two fundamentally different architectural regimes. The overall concept is illustrated in Fig. 2.

Let the model-scale variable be denoted by s. The YOLO27 detection architecture can be conceptually expressed as

$$\mathcal{A}_{27}(s) = \begin{cases} \mathcal{A}_{\text{CNN}}^{\text{dual}}, & s \in \{n, s\}, \\ \mathcal{A}_{\text{query}}, & s \in \{m, l\}, \end{cases} \tag{1}$$

where $\mathcal{A}_{\text{CNN}}^{\text{dual}}$ denotes the streamlined dual-scale CNN detector used by nano and small variants, and $\mathcal{A}_{\text{query}}$ denotes the query-based architecture used by medium and large variants. This formulation highlights the primary conceptual contribution of YOLO27: architectural specialization is determined by computational scale.

Table 2 summarizes these pathways. YOLO27n and YOLO27s prioritize minimal latency, reduced detection-head computation, and improved preservation of fine spatial features. YOLO27m introduces query-based NMS-free detection while retaining a convolutional backbone similar in philosophy to YOLO26. YOLO27l increases representational capacity substantially and integrates the UltraViT backbone, in which self-attention is introduced in the deepest feature-processing stage to capture global scene relationships.

An important distinction is that the dual-architecture split applies specifically to object detection. Other planned YOLO27 tasks retain the CNN-oriented architecture. This allows Ultralytics to introduce a more computationally intensive query-based paradigm for detection while preserving efficient and mature task-specific CNN heads for segmentation, depth, pose, classification, and oriented object detection.

The design can therefore be interpreted through three abstraction levels. First, the *user interface* remains unified. Second, the *internal architecture* changes according to model scale. Third, the *task head* changes according to the

Table 2: Architectural organization of the YOLO27 detection family (Source Link).

Model	Primary Design	Detection Scales	Prediction Paradigm	Backbone	NMS Behavior	Primary Strength	Target Deployment
YOLO27n	Streamlined CNN	2	Dense / one-to-one	CNN	Default NMS; optional NMS-free	Maximum efficiency	Edge / embedded
YOLO27s	Streamlined CNN	2	Dense / one-to-one	CNN	Default NMS; optional NMS-free	Speed–accuracy balance	Edge / drones
YOLO27m	Query-based	Multi-scale features	Object queries	YOLO26-style CNN	NMS-free	GPU balance	Production GPU
YOLO27l	Query-based hybrid	Multi-scale features	Object queries	UltraViT	NMS-free	Maximum accuracy	High-end GPU

visual prediction objective. This separation allows significant internal architectural diversification while preserving a consistent developer workflow.

3 Key Architectural Enhancements in YOLO27

YOLO27 introduces its most consequential change at the level of *model-family organization*. YOLO26 applies a common detection design across five n/s/m/l/x capacities. YOLO27 currently exposes four principal detection scales—n, s, m, and l—but divides them into two distinct architectural regimes (Ultralytics YOLO27). A compact representation is

$$\mathcal{A}_{27}(s) = \begin{cases} \mathcal{A}_{\text{CNN}}^{\text{dual}}, & s \in \{n, s\}, \\ \mathcal{A}_{\text{query}}, & s \in \{m, l\}. \end{cases} \tag{2}$$

Here, $\mathcal{A}_{\text{CNN}}^{\text{dual}}$ denotes the streamlined dual-scale CNN detector and $\mathcal{A}_{\text{query}}$ denotes query-based detection. This formulation changes the traditional meaning of model scaling. The medium and large configurations are not simply deeper or wider versions of the nano architecture; they employ a different detection mechanism.

3.1 Dual-Scale Detection for YOLO27n and YOLO27s

Conventional real-time object detectors frequently predict objects across three feature resolutions corresponding approximately to fine-, medium-, and coarse-scale representations [20, 21, 22, 23]. If the conventional prediction pyramid is denoted as

$$\mathcal{P}_3 = \{P_f, P_m, P_c\}, \tag{3}$$

where P_f, P_m, and P_c represent fine-, medium-, and coarse-resolution prediction levels, respectively, YOLO27n and YOLO27s instead employ

$$\mathcal{P}_{27}^{N/S} = \{P_f, P_c\}. \tag{4}$$

The intermediate prediction map is removed from the compact detection head as illustrated in Figure 3. This modification eliminates a significant portion of the computation associated with feature transformation, prediction convolutions, tensor movement, and decoding at the medium-resolution stage. For highly compact models, detection-head computation can represent a non-negligible proportion of total runtime; therefore, eliminating an entire prediction level provides an opportunity for meaningful latency reduction.

However, deleting a feature level without compensatory architectural adaptation could negatively affect object representation. YOLO27 consequently applies a fixed scaling mechanism to fused features to maintain balance between the two retained prediction levels. The exact implementation parameters may evolve before public release, but the conceptual operation can be represented as

$$\widetilde{F}_i = \alpha_i F_i, \qquad i \in \{f, c\}, \tag{5}$$

where F_i represents a fused feature map and α_i represents a fixed scale factor used to maintain appropriate feature magnitude across the surviving levels.

This represents an important change in computational allocation. Instead of uniformly distributing detection capacity across fine, intermediate, and coarse prediction maps, YOLO27n/s concentrate inference resources on two complementary representations: a high-resolution map responsible primarily for small and detailed targets, and a coarse map containing stronger semantic context for larger objects.

3.2 Strengthened High-Resolution Features for Small Objects

Small-object detection remains a persistent limitation of compact vision models [24, 25, 26]. A small object may occupy only a few pixels after image resizing, and repeated spatial downsampling can remove the visual evidence necessary for robust localization and classification. These difficulties are particularly severe in aerial imagery, precision agriculture, autonomous driving, surveillance, remote sensing, and mobile robotics, where operationally important objects may appear far from the camera.

YOLO27n/s address this problem by widening the early high-resolution feature stage. Consider an intermediate feature representation

$$F_h \in \mathbb{R}^{H \times W \times C}. \tag{6}$$

Increasing the effective channel capacity C while spatial dimensions H and W remain relatively large permits the network to encode a richer set of fine-grained textures, edges, local structures, and object boundaries before aggressive spatial reduction occurs.

This enhancement is particularly complementary to the dual-scale detection mechanism. Removal of the intermediate prediction map reduces computational expense, while the widened early feature representation redirects part of the representational budget toward fine-scale visual information. Accordingly, YOLO27 does not simply remove one prediction head; it restructures where capacity is allocated within the compact detector.

The design (Figure 3) indicates a broader principle for efficient computer vision: computational reduction is most effective when accompanied by deliberate redistribution of representational capacity. Instead of uniformly shrinking all components of the network, YOLO27n/s attempt to preserve the features most relevant to the hardest detection regime, small objects, while simplifying portions of the architecture that contribute disproportionately to latency.

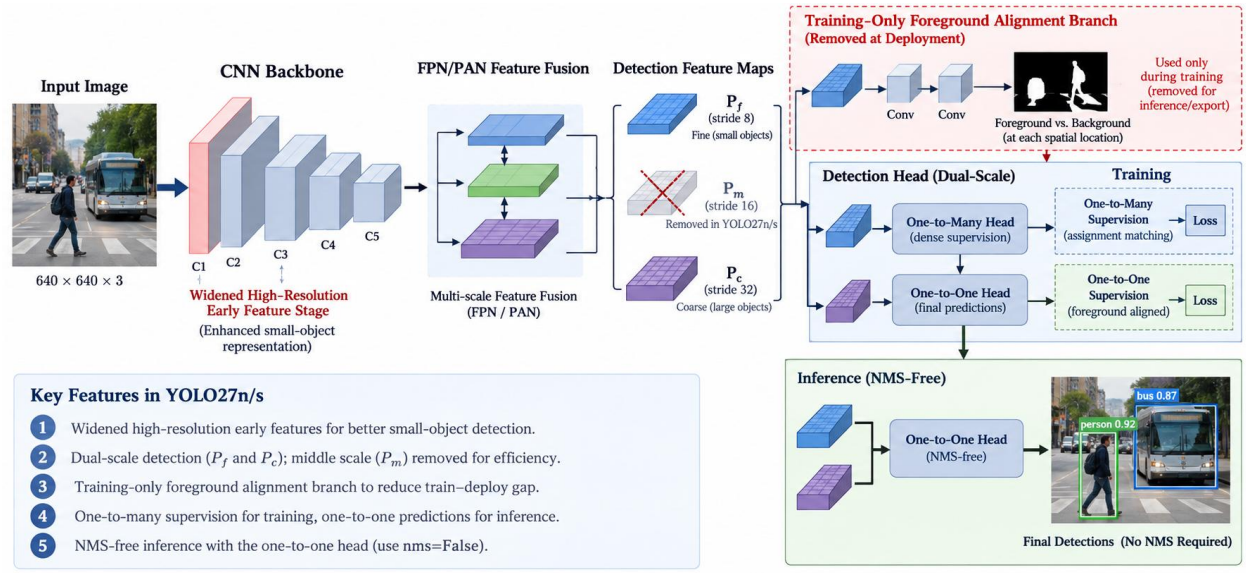

Figure 3: Detailed architecture of YOLO27n/s, illustrating the streamlined CNN backbone, widened high-resolution features, FPN/PAN fusion, and dual-scale prediction using fine (P_f) and coarse (P_c) feature maps. Training incorporates one-to-many supervision and a foreground-alignment branch, while deployment uses one-to-one predictions for efficient NMS-free inference.

3.3 Foreground Alignment Supervision

A major training-oriented innovation in YOLO27n/s is foreground alignment supervision. Modern end-to-end object detectors often use different supervision densities during training and deployment. Dense one-to-many supervision provides multiple positive learning signals for each ground-truth target and can improve optimization stability, whereas one-to-one prediction aims to associate each physical object with a unique final detection. The mismatch between these

objectives can create an accuracy gap between training-efficient dense prediction and deployment-efficient one-to-one inference.

YOLO27 introduces an additional lightweight branch during training that learns whether individual spatial locations correspond to foreground objects or background (Figure 4). Let z_i denote the feature representation at spatial location i. A conceptual binary foreground probability can be written as

$$p_i^{fg} = \sigma\left(g_\theta(z_i)\right), \tag{7}$$

where g_θ denotes the auxiliary foreground prediction branch and $\sigma(\cdot)$ is the sigmoid activation function. A generic foreground supervision term may be represented as

$$\mathcal{L}_{fg} = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \log(p_i^{fg}) + (1-y_i)\log(1-p_i^{fg})\right], \tag{8}$$

where $y_i \in \{0,1\}$ represents background or foreground assignment. This equation is presented as a conceptual abstraction of foreground/background supervision rather than a specification of the unreleased implementation.

The purpose of the auxiliary branch is to improve alignment between dense one-to-many training supervision and the one-to-one detection head used for direct prediction. Preliminary Ultralytics results report that the accuracy gap between these prediction modes decreases substantially relative to YOLO26. The documented gap of approximately 0.9 and 0.8 mAP for YOLO26n and YOLO26s, respectively, is reduced to approximately 0.4 mAP for the corresponding YOLO27 compact models.

This is particularly important because the foreground branch is a *training-only mechanism*. It is removed during inference and model export. Therefore, it can increase the quality of learned representations without increasing the deployed parameter graph or runtime latency. This follows an increasingly important principle in efficient deep learning: training may contain auxiliary complexity provided that this complexity can be structurally removed before deployment.

3.4 Dual Prediction Behavior in Compact YOLO27 Models

The N and S variants require careful interpretation with respect to NMS. YOLO27n/s do not exclusively operate as NMS-free detectors. Their standard detection pathway uses the conventional one-to-many head with NMS. The same models additionally expose a one-to-one prediction pathway that can be selected using `nms=False`, thereby enabling NMS-free inference.

This provides two operational modes:

$$\mathcal{D}_{N/S} = \begin{cases} \mathcal{H}_{1:M} + \text{NMS}, & \text{default}, \\ \mathcal{H}_{1:1}, & \texttt{nms=False}, \end{cases} \tag{9}$$

where $\mathcal{H}_{1:M}$ denotes a one-to-many prediction head and $\mathcal{H}_{1:1}$ denotes a one-to-one head capable of directly producing final detections.

This flexibility as illustrated in Figure 4 is valuable because conventional NMS remains computationally inexpensive for many compact deployment environments and may retain marginally stronger accuracy, whereas one-to-one NMS-free prediction simplifies the complete inference graph and may be preferable when deterministic end-to-end deployment or post-processing elimination is required.

3.5 Query-Based Detection in YOLO27m and YOLO27l

The architecture (Figure 4) changes more fundamentally for YOLO27m and YOLO27l. Rather than performing dense object prediction at every candidate spatial location and subsequently removing redundant detections, the larger models employ a fixed collection of learnable or dynamically refined object queries processed by a transformer decoder.

Let

$$Q^{(0)} = \{q_1^{(0)}, q_2^{(0)}, \ldots, q_{N_q}^{(0)}\} \tag{10}$$

represent an initial set of N_q object queries. Given image features F, transformer decoder layers iteratively refine these queries:

$$Q^{(k+1)} = \mathcal{T}_k \left(Q^{(k)}, F \right), \tag{11}$$

where $\mathcal{T}_k$ denotes the k-th query refinement operation. The final object representation is mapped directly into class confidence and bounding-box parameters:

$$(\hat{c}_i, \hat{b}_i) = \phi \left(q_i^{(K)} \right), \tag{12}$$

where $\hat{c}_i$ and $\hat{b}_i$ represent the predicted class and bounding box corresponding to final query i.

Because the query decoder is designed to produce a compact set of object-level hypotheses, duplicate detections can be suppressed through the architecture itself rather than by an external NMS stage. Consequently, YOLO27m/l provide native end-to-end NMS-free detection.

This approach brings YOLO27 conceptually closer to set-based transformer detectors while retaining the feature extraction and multi-scale fusion principles that have historically contributed to YOLO's real-time efficiency. It therefore represents a hybridization rather than a complete abandonment of the YOLO detection philosophy.

3.6 YOLO27m: CNN Backbone with Query-Based Decoder

YOLO27m occupies an important middle position within the family. Its backbone remains convolutional and is described as following the proven YOLO26-style design, whereas its detection stage transitions toward query-based transformer decoding.

This decomposition allows the medium model to exploit two complementary properties. Convolution retains efficient local feature extraction, translation-biased processing, and accelerator-friendly computation. The query decoder introduces global object-level reasoning and eliminates the requirement for NMS-based duplicate filtering.

The medium variant can therefore be interpreted as an architectural bridge between the fully streamlined CNN pathway of YOLO27n/s and the stronger hybrid transformer representation introduced by YOLO27l. This positioning also explains why YOLO27m is intended as the practical accuracy–latency "sweet spot" for production GPU inference.

3.7 YOLO27l and the UltraViT Backbone

YOLO27l introduces the most substantial representational change through the UltraViT backbone. While the network retains an FPN/PAN-style neck for multi-resolution feature aggregation, UltraViT introduces self-attention within the deepest backbone stage.

Convolution processes information primarily through local receptive fields, although increasingly deep layers can indirectly represent broad contextual information. Self-attention directly models relationships between spatial features. Given query, key, and value matrices Q, K, and V, scaled dot-product attention is generally expressed as

$$\text{Attention}(Q, K, V) = \text{softmax} \left(\frac{QK^T}{\sqrt{d_k}} \right) V, \tag{13}$$

where d_k denotes key dimensionality.

Applying attention primarily at deep feature stages is computationally attractive because spatial resolution has already been reduced considerably. The network can therefore obtain global contextual reasoning without incurring the cost of applying full self-attention to the original high-resolution image.

This design as illustrated in Figure 4 is especially relevant to crowded scenes, partially occluded targets, large objects, and context-dependent visual interpretation. YOLO27l thus combines convolutional efficiency, hierarchical multi-scale aggregation, transformer-style global reasoning, and query-based prediction within one architecture.

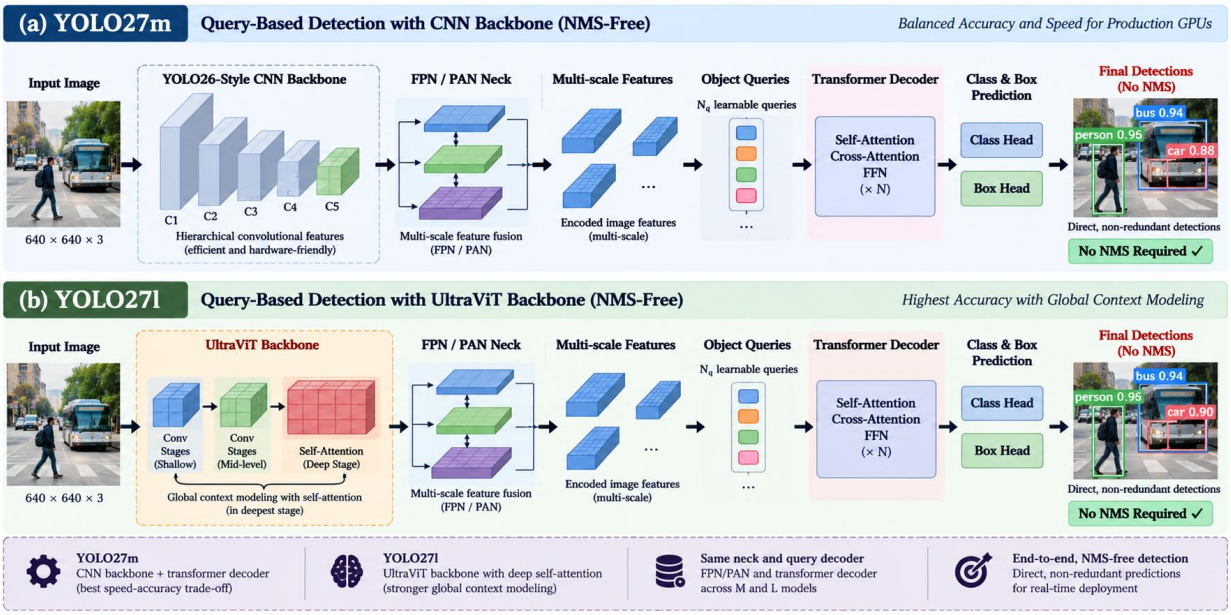

Figure 4: Comparison of YOLO27 query-based detection architectures. (a) YOLO27m combines a YOLO26-style convolutional backbone with FPN/PAN multi-scale feature fusion, learnable object queries, and transformer decoding to directly generate class and bounding-box predictions without NMS. (b) YOLO27l replaces the conventional CNN backbone with UltraViT, integrating convolutional shallow and intermediate stages with deep-stage self-attention for global-context modeling, while retaining the shared FPN/PAN neck and query decoder for accurate, end-to-end, non-redundant object detection.

Table 3: Conceptual architectural differences between YOLO26 [18] and YOLO27 (Source Link).

Aspect	YOLO26	YOLO27
Model scaling	Common architecture scaled by model capacity	Architecture itself changes between compact N/S and larger M/L detectors
Compact detection	Conventional multi-scale YOLO design	Two-scale prediction with medium detection scale removed
Small-object strategy	Improved efficient feature representation	Widened early high-resolution stage plus preserved fine prediction map
Training alignment	One-to-many and one-to-one prediction pathways	Additional foreground alignment supervision explicitly reduces accuracy gap
NMS behavior	Optional one-to-one end-to-end inference	N/S support NMS default/NMS-free mode; M/L are query-based and NMS-free
Medium detector	CNN-style detector	CNN backbone coupled to transformer query decoder
Large detector	Scaled CNN-oriented architecture	UltraViT backbone + FPN/PAN + transformer query decoder
Global reasoning	Primarily hierarchical convolutional features	Explicit deep-stage self-attention in YOLO27l
Model family	N, S, M, L, X variants	N, S, M, L detection variants in current preview
Task ecosystem	Detection, segmentation, semantic, depth, classification, pose, OBB	Planned support for the same seven major visual tasks

4 Architectural Comparison of YOLO27 with YOLO26

YOLO27 represents a broader architectural departure from YOLO26 than would be implied by model numbering alone. YOLO26 primarily emphasized simplification of an efficient YOLO-style detector and optional one-to-one inference. YOLO27 instead explicitly recognizes that different model scales may benefit from different prediction mechanisms.

Table 3 summarizes the principal architectural transition.

The architectural philosophy behind YOLO27 can consequently be summarized as *specialize rather than uniformly scale*. The nano and small variants remove unnecessary computation and emphasize high-resolution spatial information. The medium model introduces object-query reasoning while preserving a convolutional feature extractor. The large model adds global self-attention and substantially greater capacity for accuracy-critical environments.

This progression suggests that the classical distinction between "CNN detectors" and "transformer detectors" is becoming less meaningful. Instead, real-time object detection increasingly involves selecting where convolution,

Table 4: Planned task coverage of the YOLO27 family. Checkmarks denote intended support according to the preview documentation; public availability may change before release (Source Link).

Family	Example Filenames	Task	Primary Output	Train	Val	Predict	Export
YOLO27	`yolo27n/s/m/l.pt`	Detection	Boxes + classes	✓	✓	✓	✓
YOLO27-seg	`yolo27n/s/m/l-seg.pt`	Instance segmentation	Instance masks	✓	✓	✓	✓
YOLO27-sem	`yolo27n/s/m/l-sem.pt`	Semantic segmentation	Pixel classes	✓	✓	✓	✓
YOLO27-depth	`yolo27n/s/m/l-depth.pt`	Depth estimation	Dense depth	✓	✓	✓	✓
YOLO27-cls	`yolo27n/s/m/l-cls.pt`	Classification	Class probabilities	✓	✓	✓	✓
YOLO27-pose	`yolo27n/s/m/l-pose.pt`	Pose estimation	Keypoints	✓	✓	✓	✓
YOLO27-obb	`yolo27n/s/m/l-obb.pt`	Oriented detection	Rotated boxes	✓	✓	✓	✓

attention, dense prediction, and object-query reasoning provide the greatest marginal benefit for a given computational budget.

5 YOLO27 Supported Vision Tasks

Although the two-path architecture is specific to object detection, the YOLO27 family is planned as a multi-task real-time computer vision framework. Current documentation identifies seven principal task groups: object detection, instance segmentation, semantic segmentation, depth estimation, image classification, pose estimation, and oriented object detection.

Table 4 summarizes the planned model naming convention and principal outputs.

5.1 Object Detection

Detection represents the task in which YOLO27 introduces its major architectural bifurcation. The compact N/S variants use the dual-scale CNN pathway, whereas M/L models use query-based detection. The primary output remains object category probabilities and bounding-box coordinates.

5.2 Instance and Semantic Segmentation

Instance segmentation associates a separate pixel-level mask with each detected object [27, 28], whereas semantic segmentation assigns categorical labels to image pixels regardless of individual object identity [29, 30, 31]. YOLO27 is intended to provide dedicated model families for both tasks. Importantly, these models use the CNN architecture rather than the query-based M/L detection design.

5.3 Depth Estimation

The inclusion of depth estimation broadens YOLO27 from categorical scene perception toward geometric scene understanding. A depth model estimates a dense map

$$D = f_\theta(I), \tag{14}$$

where I represents an RGB image and D represents predicted per-pixel depth. Such capability is particularly relevant for autonomous navigation, robotics, obstacle avoidance, AR/VR, and spatial perception.

5.4 Classification

YOLO27 classification models provide image-level semantic recognition rather than spatial localization. Maintaining classification within the same software ecosystem allows researchers to reuse model-loading, training, validation, and export interfaces across fundamentally different computer vision tasks.

Table 5: Preliminary YOLO27 object detection performance on the COCO validation set at 640-pixel input resolution. CPU latency is measured using ONNX Runtime FP32 on AMD EPYC 9655, while GPU latency is measured using TensorRT 11 FP16 on NVIDIA RTX PRO 6000. Detection speed measurements use the NMS-free prediction pathway.

Model	Input	$\text{mAP}_{50:95}$	CPU ONNX (ms)	GPU TRT11 (ms)	Params (M)	FLOPs (B)
YOLO27n	640	42.3	$\mathbf{16.1 \pm 1.6}$	$\mathbf{0.62 \pm 0.00}$	**3.0**	**7.2**
YOLO27s	640	49.6	33.2 ± 0.1	0.79 ± 0.00	11.8	28.2
YOLO27m	640	55.8	67.2 ± 0.3	1.39 ± 0.00	22.8	65.0
YOLO27l	640	**60.4**	149.6 ± 1.4	2.32 ± 0.00	72.3	165.3

5.5 Pose Estimation

Pose models predict structured keypoints corresponding to human joints, anatomical landmarks, mechanical components, or domain-specific landmarks. Such functionality supports human activity analysis, sports analytics, livestock monitoring, manipulation, and human-robot interaction.

5.6 Oriented Object Detection

Oriented bounding boxes represent objects using rotated rectangles rather than axis-aligned bounding boxes. This is particularly useful for aerial imagery, remote sensing, text detection, elongated agricultural structures, maritime imagery, and industrial components whose dominant orientation carries important geometric information.

In addition, YOLO27 detection, segmentation, pose, and oriented detection models are intended to integrate with tracking mode, enabling their outputs to support temporal multi-object association across video sequences.

6 Performance Benchmarking and Preliminary Results

The preliminary performance evaluation of YOLO27 spans seven computer vision tasks: object detection, instance segmentation, semantic segmentation, monocular depth estimation, image classification, pose estimation, and oriented object detection (OBB). These benchmarks provide a broader view of YOLO27 than detection accuracy alone and reveal how model scaling affects accuracy, latency, parameter count, and computational complexity across different visual prediction problems. Unless otherwise stated, GPU inference speed is measured on an NVIDIA RTX PRO 6000 using TensorRT 11 with FP16 precision, whereas CPU inference speed is measured on an AMD EPYC 9655 using ONNX Runtime with FP32 precision. The results reported in this section are preliminary research measurements and may change during final model development. Public model weights and complete reproduction instructions are expected to accompany the official release.

For detection specifically, reported speed measurements use the NMS-free prediction pathway with `nms=False`. This distinction is important for YOLO27n and YOLO27s because their default one-to-many prediction pathway uses NMS, whereas the reported detection latency corresponds to the one-to-one NMS-free head. Parameter and FLOP counts correspond to fused inference models after convolution-batch-normalization folding and removal of unused prediction branches. Consequently, training checkpoints can contain more parameters than the corresponding deployed inference graph.

6.1 COCO Object Detection Performance

Object detection performance is evaluated on the COCO validation set, which contains 80 object categories and remains one of the most widely adopted benchmarks for general-purpose object detection. Detection accuracy is reported using mean Average Precision averaged over intersection over union (IoU) thresholds from 0.50 to 0.95, denoted $\text{mAP}_{50:95}$ (Source Link). Table 5 summarizes the preliminary results at an input resolution of 640×640 pixels.

The progression in Table 5 demonstrates strong accuracy scaling across the YOLO27 family. Moving from YOLO27n to YOLO27s increases $\text{mAP}_{50:95}$ from 42.3 to 49.6, corresponding to a gain of 7.3 percentage points. YOLO27m further increases accuracy to 55.8 mAP, representing another 6.2-point improvement, while YOLO27l reaches 60.4 mAP, an additional 4.6-point gain. Thus, the complete transition from YOLO27n to YOLO27l provides an 18.1-point increase in COCO $\text{mAP}_{50:95}$.

The accuracy–latency relationship is visualized explicitly in Fig. 5(a). The figure is particularly informative because it places the four models in a common accuracy–speed space while simultaneously encoding parameter count through

bubble size. YOLO27n and YOLO27s occupy the low-latency region associated with the streamlined CNN architecture, whereas YOLO27m and YOLO27l occupy the higher-accuracy region associated with query-based detection. The conceptual boundary shown in Fig. 5(a) therefore represents not merely a change in model capacity, but a transition between two different detection paradigms.

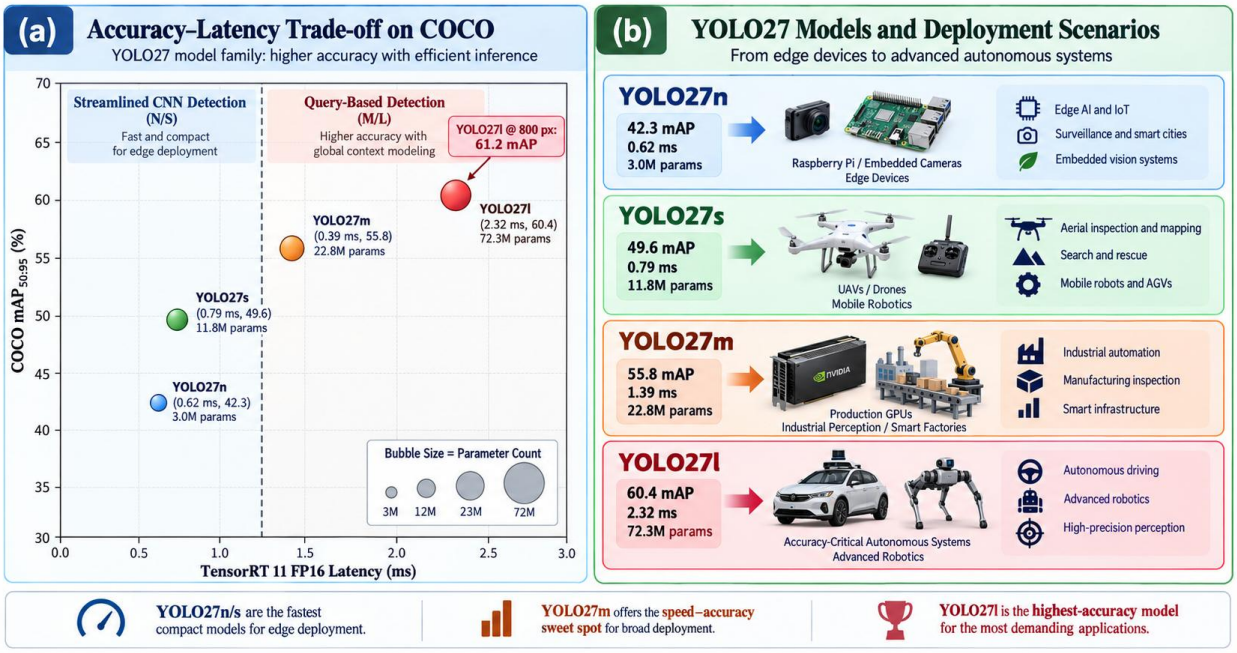

Figure 5: YOLO27 accuracy–latency and deployment landscape. (a) COCO $mAP_{50:95}$ versus TensorRT 11 FP16 latency highlights the architectural transition from streamlined CNN-based YOLO27n/s to query-based YOLO27m/l, with bubble size representing fused parameter count. The additional YOLO27l operating point at 800-pixel resolution reaches 61.2 mAP. (b) Representative deployment scenarios span embedded edge vision, UAVs and mobile robotics, production GPU-based industrial perception, and accuracy-critical autonomous systems.

GPU latency remains low relative to the observed accuracy increase. YOLO27n requires approximately 0.62 ms under the reported TensorRT configuration, while YOLO27s requires 0.79 ms. The query-based YOLO27m increases latency to 1.39 ms while reaching 55.8 mAP. YOLO27l reaches 60.4 mAP at approximately 2.32 ms. These values indicate that the larger models exchange additional compute for progressively stronger detection accuracy while remaining within a real-time GPU operating regime.

CPU scaling is substantially more pronounced. CPU latency rises from 16.1 ms for YOLO27n to 149.6 ms for YOLO27l, corresponding to approximately a 9.3$\times$ increase. This difference reinforces the importance of hardware-aware model selection: YOLO27l may remain highly efficient on a modern GPU while being considerably more computationally demanding for CPU-only deployment.

Model complexity follows a similar trend. The fused parameter count increases from 3.0 M for YOLO27n to 11.8 M, 22.8 M, and 72.3 M for YOLO27s, YOLO27m, and YOLO27l, respectively. FLOPs increase from 7.2 B to 165.3 B. The especially large increase between YOLO27m and YOLO27l reflects the accuracy-oriented design of the large model, including its substantially higher representational capacity and UltraViT-based backbone.

At a larger 800-pixel input resolution, YOLO27l is additionally reported to reach 61.2 $mAP_{50:95}$. This operating point is highlighted separately in Fig. 5(a). The result should not be treated as directly equivalent to the 640-pixel measurements because increasing input resolution changes both spatial information content and computational cost. Nevertheless, it demonstrates that YOLO27l can trade additional computation for higher accuracy and represents the first reported Ultralytics YOLO operating point above 60 mAP on COCO.

6.2 Instance Segmentation Performance on COCO

YOLO27 also extends its CNN architecture to instance segmentation, where each individual object requires both localization and pixel-level mask prediction. Table 6 reports COCO bounding-box and mask $mAP_{50:95}$.

Table 6: Preliminary YOLO27 instance segmentation performance on COCO at 640-pixel input resolution.

Model	Input	Box $mAP_{50:95}$	Mask $mAP_{50:95}$	CPU (ms)	GPU (ms)	Params (M)	FLOPs (B)
YOLO27n-seg	640	41.8	35.4	$\mathbf{22.0 \pm 2.2}$	$\mathbf{0.73 \pm 0.00}$	**3.0**	**11.2**
YOLO27s-seg	640	50.0	42.5	45.8 ± 0.1	0.96 ± 0.00	11.6	43.1
YOLO27m-seg	640	53.2	45.0	109.3 ± 0.9	1.46 ± 0.00	25.4	139.6
YOLO27l-seg	640	**57.7**	**47.9**	248.3 ± 1.3	2.86 ± 0.00	67.5	361.9

Table 7: Preliminary YOLO27 semantic segmentation performance on Cityscapes. The benchmark uses 1024×2048-pixel inputs and reports validation mIoU.

Model	Input	$mIoU_{val}$	CPU (ms)	GPU (ms)	Params (M)	FLOPs (B)
YOLO27n-sem	1024×2048	78.8	$\mathbf{95.6 \pm 3.0}$	$\mathbf{0.89 \pm 0.00}$	**2.0**	**34.0**
YOLO27s-sem	1024×2048	81.2	169.6 ± 0.9	1.43 ± 0.00	7.8	131.5
YOLO27m-sem	1024×2048	82.4	336.9 ± 2.8	2.71 ± 0.00	16.1	385.4
YOLO27l-sem	1024×2048	**83.8**	787.1 ± 2.2	6.43 ± 0.00	44.8	1085.8

Segmentation accuracy increases monotonically with model scale. Box $mAP_{50:95}$ rises from 41.8 for YOLO27n-seg to 57.7 for YOLO27l-seg, whereas mask mAP increases from 35.4 to 47.9. The persistent difference between box and mask mAP reflects the greater difficulty of recovering precise object boundaries compared with predicting enclosing bounding boxes. GPU latency nevertheless remains below 3 ms for all four reported variants under the benchmark configuration.

6.3 Semantic Segmentation Performance on Cityscapes

Semantic segmentation assigns a semantic class label to every image pixel, rather than separating individual object instances [32, 33, 34]. YOLO27 semantic segmentation models are evaluated on the Cityscapes benchmark using 1024×2048 input images and mean Intersection over Union (mIoU) as the primary accuracy metric. The preliminary results are summarized in Table 7.

As shown in Table 7, semantic segmentation accuracy improves consistently with model scale. YOLO27n-sem achieves 78.8% mIoU with only 2.0 M parameters, while YOLO27l-sem reaches 83.8% mIoU. This 5.0-point improvement is accompanied by a substantial increase in computational cost, from 34.0 to 1085.8 B FLOPs and from 0.89 to 6.43 ms GPU latency. These results illustrate that dense pixel-wise prediction at high spatial resolution produces a markedly steeper computational scaling regime than conventional object detection, particularly for the largest YOLO27 variant.

6.4 Monocular Depth Estimation

Monocular depth estimation extends YOLO27 beyond categorical recognition toward geometric scene understanding by predicting per-pixel scene depth from a single RGB image. The reported YOLO27 depth models use 768-pixel inputs and are evaluated using the δ_1 accuracy criterion on NYU Depth V2 (Source Link), KITTI-580 (Source Link), and a broader benchmark mean. Larger δ_1 values indicate that a greater fraction of predicted depths satisfy the accepted multiplicative agreement with ground truth. The reported results are summarized in Table 8.

Table 8 reveals that depth-estimation performance does not scale monotonically across every benchmark. YOLO27n-depth achieves the highest KITTI-580 δ_1 value of 0.8256, whereas YOLO27l-depth produces the strongest NYU result of 0.8711 and the highest overall benchmark mean of 0.7527. This dataset-dependent behavior indicates that greater model capacity does not necessarily guarantee uniformly stronger performance under domain variation. Accordingly, depth-model selection should consider cross-dataset generalization, deployment cost, and application environment rather than model scale alone.

6.5 Image Classification Complexity

YOLO27 classification models operate at 224×224 input resolution and represent the image-level recognition branch of the YOLO27 ecosystem. Unlike the detection, segmentation, pose, and depth benchmarks, the preliminary classification results currently emphasize computational characteristics rather than ImageNet Top-1 or Top-5 accuracy. Accordingly, Table 9 reports runtime, parameter count, and FLOPs without introducing unreported classification accuracy values.

Table 8: Preliminary YOLO27 monocular depth-estimation performance. δ_1 is reported for NYU Depth V2, KITTI-580, and the benchmark mean.

Model	Input	δ_1 NYU	δ_1 KITTI-580	δ_1 Mean	CPU (ms)	GPU (ms)	Params (M)	FLOPs (B)
YOLO27n-depth	768	0.8314	**0.8256**	0.7238	$\mathbf{47.2 \pm 4.0}$	$\mathbf{0.79 \pm 0.00}$	**5.4**	**49.3**
YOLO27s-depth	768	0.8682	0.7835	0.7454	72.1 ± 0.5	0.97 ± 0.00	13.0	77.3
YOLO27m-depth	768	0.8652	0.7746	0.7476	120.2 ± 0.4	1.34 ± 0.00	23.4	143.6
YOLO27l-depth	768	**0.8711**	0.8041	**0.7527**	253.7 ± 0.5	2.65 ± 0.00	59.3	341.8

Table 9: Preliminary computational characteristics of YOLO27 classification models at 224-pixel input resolution.

Model	CPU (ms)	GPU (ms)	Params (M)	FLOPs (B)
YOLO27n-cls	$\mathbf{1.9 \pm 0.0}$	$\mathbf{0.28 \pm 0.00}$	**2.9**	**0.5**
YOLO27s-cls	3.1 ± 0.1	0.31 ± 0.00	6.9	1.7
YOLO27m-cls	6.0 ± 0.6	0.39 ± 0.00	12.4	6.1
YOLO27l-cls	16.7 ± 0.1	0.69 ± 0.00	32.8	18.5

As summarized in Table 9, the classification variants are the computationally lightest YOLO27 models among the reported task families. YOLO27n-cls requires only 0.5 B FLOPs and approximately 0.28 ms of GPU inference time, whereas YOLO27l-cls increases to 18.5 B FLOPs and approximately 0.69 ms. Even the largest classification model therefore remains substantially less computationally demanding than dense high-resolution tasks such as semantic segmentation. This difference primarily reflects the simpler prediction objective of image-level classification, which produces a global class representation rather than spatially dense pixel-wise outputs.

6.6 Pose Estimation Performance on COCO

Pose estimation evaluates the ability of YOLO27 to localize structured keypoints. The COCO pose benchmark reports $\text{mAP}_{50:95}$ and mAP_{50} for the pretrained person category.

Pose estimation exhibits consistent scaling with model capacity. $\text{mAP}_{50:95}$ improves from 58.0 for YOLO27n-pose to 72.2 for YOLO27l-pose, representing a 14.2-point gain. At the less restrictive mAP_{50} criterion, performance increases from 84.1 to 91.3. The YOLO27m-pose configuration may provide an attractive intermediate operating point, achieving 69.0 $\text{mAP}_{50:95}$ at approximately 1.29 ms GPU latency.

6.7 Oriented Object Detection on DOTA

Oriented object detection extends conventional axis-aligned bounding boxes by predicting object orientation [35, 36, 35], making it particularly relevant to aerial imagery [37, 38, 39], remote sensing [40, 41, 42], industrial inspection [43, 44], maritime perception [45, 46], and other domains containing elongated or arbitrarily rotated targets. Table 11 reports preliminary results on DOTA v1 using 1024-pixel inputs.

The OBB results exhibit a notably different scaling pattern from standard COCO detection. Increasing model size from YOLO27n-obb to YOLO27s-obb provides a 1.8-point $\text{mAP}_{50:95}$ gain, whereas increasing from the small to medium model provides only 0.1 point. YOLO27l-obb reaches the highest reported value of 57.2. This comparatively modest scaling indicates that larger parameter counts do not necessarily translate into proportional improvements for oriented detection and that task-specific geometric constraints may become increasingly important.

6.8 Cross-Task Performance and Computational Scaling

Taken together, Tables 5–11 demonstrate that YOLO27 scaling behavior is strongly task dependent. Increasing model capacity consistently improves standard detection, instance segmentation, semantic segmentation, and pose estimation, but the marginal return varies substantially. Depth estimation and oriented detection show more nuanced behavior, indicating that model scale alone cannot explain cross-domain generalization.

A compact cross-task summary is provided in Table 12. Because different tasks use different datasets, input resolutions, and evaluation metrics, the numerical values in this table should *not* be interpreted as directly comparable accuracy

Table 10: Preliminary YOLO27 pose estimation performance on COCO at 640-pixel input resolution.

Model	Input	$\text{mAP}_{50:95}$	mAP_{50}	CPU (ms)	GPU (ms)	Params (M)	FLOPs (B)
YOLO27n-pose	640	58.0	84.1	$\mathbf{19.6 \pm 0.1}$	$\mathbf{0.67 \pm 0.00}$	**3.1**	**9.2**
YOLO27s-pose	640	64.3	87.0	36.6 ± 0.2	0.80 ± 0.00	11.2	31.1
YOLO27m-pose	640	69.0	89.9	76.6 ± 0.2	1.29 ± 0.00	22.8	86.2
YOLO27l-pose	640	**72.2**	**91.3**	181.1 ± 0.7	2.45 ± 0.00	61.7	242.3

Table 11: Preliminary YOLO27 oriented object detection performance on DOTA v1 at 1024-pixel input resolution.

Model	Input	$\text{mAP}_{50:95}$	mAP_{50}	CPU (ms)	GPU (ms)	Params (M)	FLOPs (B)
YOLO27n-obb	1024	54.0	80.3	$\mathbf{42.6 \pm 0.2}$	$\mathbf{0.83 \pm 0.00}$	**3.0**	**19.7**
YOLO27s-obb	1024	55.8	81.6	88.2 ± 0.6	1.30 ± 0.00	10.9	76.6
YOLO27m-obb	1024	55.9	82.6	188.0 ± 0.7	1.92 ± 0.00	22.8	221.7
YOLO27l-obb	1024	**57.2**	**82.7**	455.8 ± 1.1	4.33 ± 0.00	68.7	622.6

scores. Instead, the table summarizes the best reported model within each task family and the associated computational operating point.

The summary reveals three broader trends. First, the large variant provides the strongest reported primary accuracy metric for most tasks. Second, dense high-resolution prediction, particularly semantic segmentation and OBB, requires substantially greater computation than classification or standard detection. Third, the optimal deployment model cannot be determined by accuracy alone because latency and resource scaling differ substantially between tasks.

6.9 Interpreting Parameters, FLOPs, and Fused Inference Models

The parameter and FLOP values reported above correspond to fused inference models rather than complete training graphs. During deployment, convolution and batch-normalization operations can be algebraically fused such that

$$\mathbf{y} = \text{BN}\left(\mathbf{W} * \mathbf{x} + \mathbf{b}\right) \quad \longrightarrow \quad \mathbf{W}' * \mathbf{x} + \mathbf{b}', \tag{15}$$

where $\mathbf{W}'$ and $\mathbf{b}'$ incorporate the learned batch-normalization parameters. This transformation reduces runtime graph complexity without changing the represented inference function.

YOLO27 additionally removes prediction branches that are not required by the selected deployment pathway. The training-only foreground-alignment mechanism of YOLO27n/s is an important example: it contributes supervision during optimization but does not contribute computational cost after removal for inference and export. Consequently, comparisons based exclusively on training checkpoint parameter counts may overestimate actual deployment complexity.

FLOPs should likewise be interpreted cautiously. Although FLOPs provide a hardware-independent approximation of arithmetic complexity, they do not fully characterize memory access, kernel-launch overhead, parallelism, operator fusion, attention efficiency, or accelerator utilization. The relationship between FLOPs and latency is therefore non-linear, particularly when comparing CNN and transformer-based architectures.

6.10 Accuracy-Latency Scaling

A simplified detection-efficiency indicator can be defined as

$$\eta_{\text{det}} = \frac{\text{mAP}_{50:95}}{T_{\text{inf}}}, \tag{16}$$

where T_{inf} denotes inference latency. Although this ratio can provide a compact indication of accuracy obtained per unit inference time, it should not replace comprehensive benchmarking because latency depends on hardware, precision, batch size, input resolution, preprocessing, post-processing, memory transfer, and inference-engine optimization.

The trade-off is visualized in Fig. 5(a), where the compact YOLO27n/s models occupy the efficiency-oriented region and YOLO27m/l shift toward higher accuracy. Importantly, Fig. 5(a) also shows that the architectural transition occurs

Table 12: Cross-task summary of the highest reported YOLO27 variant within each preliminary benchmark. Metrics are task-specific and are not directly comparable across rows.

Task	Dataset	Best Variant	Primary Metric	Score	Input	GPU (ms)	Params (M)
Detection	COCO	YOLO27l	$mAP_{50:95}$	60.4	640	2.32	72.3
Instance Seg.	COCO	YOLO27l-seg	Mask $mAP_{50:95}$	47.9	640	2.86	67.5
Semantic Seg.	Cityscapes	YOLO27l-sem	$mIoU_{val}$	83.8	1024×2048	6.43	44.8
Depth	NYU / multi-benchmark	YOLO27l-depth	δ_1 mean	0.7527	768	2.65	59.3
Pose	COCO	YOLO27l-pose	$mAP_{50:95}$	72.2	640	2.45	61.7
OBB	DOTA v1	YOLO27l-obb	$mAP_{50:95}$	57.2	1024	4.33	68.7
Classification	ImageNet	–	Accuracy not reported here	–	224	0.28–0.69	2.9–32.8

without a discontinuous increase in GPU latency: YOLO27m reaches 55.8 mAP at 1.39 ms despite transitioning to query-based detection.

This observation supports the scale-adaptive design philosophy of YOLO27. Rather than forcing a single architecture across every computational budget, the family retains streamlined CNN detection where latency is most critical and introduces query-based reasoning where additional model capacity can produce greater accuracy benefits.

6.11 From Benchmark Performance to Deployment Selection

Benchmark performance becomes practically meaningful only when considered relative to deployment requirements. Fig. 5(b) translates the numerical accuracy–latency trade-off into representative application scenarios. YOLO27n is positioned for embedded cameras, edge AI, and resource-constrained devices; YOLO27s provides greater capacity for UAVs, mobile robots, and real-time aerial perception; YOLO27m represents a speed–accuracy compromise for production GPUs, industrial automation, and manufacturing inspection; and YOLO27l targets accuracy-critical autonomous systems and advanced robotic perception.

This interpretation is consistent with the performance distribution in Fig. 5(a): the increase in accuracy is accompanied by a progressive increase in parameter count and latency. Thus, Fig. 5(a) provides the quantitative basis for the deployment mapping shown in Fig. 5(b).

The deployment scenarios in Fig. 5(b), however, should be interpreted as representative rather than prescriptive. Actual model choice must be validated on the target hardware using the intended image resolution, precision, batch size, and application-specific workload. For example, a YOLO27m model may be preferable to YOLO27s on a sufficiently powerful on-board GPU, whereas YOLO27n may remain the only practical choice for strictly power-constrained embedded hardware.

6.12 Reproducibility and Benchmarking Protocol

Reproducibility is essential for validating the preliminary YOLO27 results after public release. The reported detection accuracy is intended to be reproducible through the standard Ultralytics validation interface. A representative validation command is shown below.

YOLO27 COCO Validation

```
yolo val model=yolo27n.pt data=coco.yaml
```

Scientifically rigorous comparisons should control the input resolution, numerical precision, batch size, warm-up procedure, inference engine, hardware platform, preprocessing, and post-processing. For YOLO27n/s detection models, the selected prediction head should additionally be reported because accuracy and latency may differ between the default one-to-many pathway with NMS and the one-to-one NMS-free pathway.

For detection, the benchmark values reported in Table 5 correspond to the NMS-free pathway. Consequently, future studies attempting to reproduce these measurements should explicitly maintain the same prediction configuration and report the complete hardware and inference settings.

6.13 Training and Inference Examples

YOLO27 preserves the unified Ultralytics interface for training, validation, prediction, and export. Pretrained PyTorch `*.pt` weights and model configuration `*.yaml` files can be passed through the same `YOLO()` interface. A representative Python workflow is shown below.

YOLO27 Training and Inference

```python
from ultralytics import YOLO

# Load pretrained model
model = YOLO("yolo27n.pt")

# Run inference
results = model("path/to/bus.jpg")

# Train on COCO8
results = model.train(
    data="coco8.yaml",
    epochs=100,
    imgsz=640
)
```

The same workflow can be executed through the Ultralytics command-line interface. To maintain a compact single-column presentation, the prediction and training commands are separated across lines.

YOLO27 Command-Line Interface

```bash
# Prediction
yolo predict model=yolo27n.pt \
    source=path/to/bus.jpg

# Training
yolo train model=yolo27n.pt \
    data=coco8.yaml \
    epochs=100 imgsz=640
```

For custom datasets, the same interface can be retained by replacing the dataset specification with the corresponding `data.yaml` configuration file.

YOLO27 Custom-Dataset Training

```python
from ultralytics import YOLO

# Load pretrained model
model = YOLO("yolo27n.pt")

# Run inference
results = model("path/to/image.jpg")

# Train on custom dataset
results = model.train(
    data="data.yaml",
    epochs=100,
    imgsz=640
)
```

For YOLO27n/s detection models, the one-to-one NMS-free prediction pathway can be selected using `nms=False`. This configuration should be reported explicitly in experimental studies because the default compact-model prediction pathway uses the one-to-many head followed by NMS.

YOLO27 NMS-Free Inference

```
results = model.predict(
    source="path/to/image.jpg",
    nms=False
)
```

Overall, the preliminary benchmarks indicate that YOLO27 should be interpreted as a family of deployment-specific operating points rather than as a single detector. The compact N/S models emphasize efficiency and high-throughput inference, whereas the M/L detection models progressively trade additional computation for query-based reasoning and higher accuracy. Meanwhile, segmentation, semantic segmentation, depth estimation, classification, pose estimation, and OBB demonstrate the broader applicability of YOLO27 as a unified real-time visual perception framework. The quantitative trends reported in Tables 5–12, together with the accuracy–deployment landscape in Fig. 5, reinforce the central design principle of YOLO27: model architecture and computational capacity should be selected jointly according to the visual task, available hardware, and intended deployment environment.

7 Model Selection and Deployment Considerations

The architectural diversity of YOLO27 implies that model selection should not be based solely on parameter count. Instead, users should select models according to deployment hardware, latency requirements, target-object scale, scene complexity, and the relative importance of accuracy.

7.1 YOLO27n: Edge and Embedded Perception

YOLO27n is the smallest model and is intended for resource-constrained applications in which latency, energy consumption, and memory footprint dominate system design. Representative deployment environments include embedded cameras, IoT vision systems, low-power robotic platforms, lightweight UAVs, and potentially Raspberry Pi- or Jetson-class devices.

Its dual-scale prediction head and strengthened high-resolution features make it especially interesting for edge scenarios containing small objects. The model's relatively small parameter count of approximately 3.0 million also facilitates model storage and memory-efficient inference.

7.2 YOLO27s: Real-Time Mobile Vision

YOLO27s provides increased representational capacity while preserving the compact CNN pathway. Its approximately 49.6 mAP preliminary COCO performance and sub-millisecond reported TensorRT latency position it as a candidate for mobile robotics, drones, high-frame-rate smart cameras, and other applications where additional accuracy is needed without transitioning to the heavier query-based architecture.

7.3 YOLO27m: Production GPU Sweet Spot

YOLO27m is particularly important because it introduces query-based detection while maintaining a relatively compact model size. Its approximately 22.8 million fused parameters and 55.8 mAP reported performance indicate a significant increase in accuracy over the compact variants without approaching the computational scale of YOLO27l.

The combination of a convolutional backbone and transformer query decoder makes YOLO27m suitable for production GPU environments requiring high throughput, native NMS-free inference, and stronger object-level reasoning.

7.4 YOLO27l: Accuracy-Critical Systems

YOLO27l targets applications where accuracy is prioritized but real-time GPU execution remains necessary. Such applications may include autonomous vehicles, advanced robotic manipulation, high-resolution aerial perception, industrial inspection, intelligent transportation, medical vision, and complex environmental monitoring.

The introduction of UltraViT is particularly relevant in these scenarios because global context may help distinguish objects in highly cluttered or partially occluded scenes. The model's reported 60.4 mAP at 640 pixels and 61.2 mAP at 800 pixels position it as the highest-accuracy model within the preliminary YOLO27 family.

Table 13: Recommended YOLO27 variants for different deployment objectives.

Model	Design Priority / Architecture	Target Deployment
YOLO27n	Maximum speed; dual-scale CNN	Edge/IoT devices
YOLO27s	Speed–accuracy; dual-scale CNN	UAVs/mobile robots
YOLO27m	Balanced performance; CNN + query decoder	Production vision
YOLO27l	Maximum accuracy; UltraViT + query decoder	High-end autonomous perception

8 Unified Training, Inference, and Export Interface

One practical strength of YOLO27 is that architectural differences remain abstracted behind the same high-level Ultralytics interface. A user is not required to manually instantiate a transformer decoder for YOLO27m/l or configure the dual-scale prediction pathway for YOLO27n/s. Instead, the model checkpoint determines the appropriate pipeline automatically.

After public release, the standard Python interface is expected to follow the form:

YOLO27 Training and Inference Example

```python
from ultralytics import YOLO

# Load a pretrained YOLO27 model
model = YOLO("yolo27n.pt")

# Inference
results = model("path/to/image.jpg")

# Custom-dataset training
results = model.train(
    data="data.yaml",
    epochs=100,
    imgsz=640
)
```

For compact N/S detection models, the one-to-one NMS-free prediction path can be selected according to the YOLO27 interface configuration. Conceptually,

YOLO27 NMS-Free Inference Example

```python
results = model.predict(
    source="path/to/image.jpg",
    nms=False
)
```

can be used to request the NMS-free prediction head where supported.

The same general interface is intended to control model validation and export. This abstraction is significant from a software engineering perspective because it separates architectural innovation from user-facing complexity. A research group can benchmark nano, small, medium, and large models through largely identical scripts even though the underlying detectors employ different architectural principles.

For custom datasets, this also simplifies transfer learning. A researcher can initially develop a pipeline with YOLO27n for rapid experimentation and subsequently substitute YOLO27m or YOLO27l when greater accuracy is required, with limited modification of the surrounding training code.

9 Implications for Real-Time Computer Vision

9.1 Scale-Adaptive Architecture as a New YOLO Design Principle

The most consequential contribution of YOLO27 may not be an individual module but rather the decision to use different detection paradigms at different model scales. Conventional compound scaling assumes that a well-designed architecture remains near-optimal when proportionally enlarged or reduced. YOLO27 challenges this assumption.

At low computational budgets, convolution and highly optimized dense prediction remain difficult to outperform in terms of raw latency. At larger computational budgets, however, query-based prediction and global contextual reasoning become increasingly attractive because the additional computation can generate substantial accuracy gains.

YOLO27 therefore suggests a general formulation:

$$\mathcal{A}^* = f(B, L, M, A), \tag{17}$$

where the optimal architecture $\mathcal{A}^*$ depends on computational budget B, latency constraint L, memory availability M, and required accuracy A. Under this interpretation, architecture is itself a deployment variable rather than a fixed structure subjected only to scaling.

9.2 Implications for Small-Object Detection

The combination of dual-scale prediction and strengthened high-resolution representation is particularly relevant to applications dominated by small objects. In precision agriculture, examples include fruitlets, flowers, pests, disease symptoms, buds, and small plant structures. In aerial robotics, pedestrians, vehicles, animals, and infrastructure elements may occupy only a small number of pixels. In industrial inspection, defects can similarly represent extremely small portions of an image.

YOLO27's design suggests that small-object performance does not necessarily require preserving every prediction scale. Instead, computational resources can potentially be concentrated on a stronger high-resolution pathway while intermediate prediction computation is reduced.

9.3 Implications for Robotics

Robotic perception is characterized by strict latency requirements because predictions directly influence control decisions [47, 48]. A perception pipeline with high average accuracy but unpredictable latency can be less useful than a slightly less accurate model with deterministic response time [49, 50].

YOLO27 provides potentially useful operating points across the robotic computing hierarchy. Compact variants may support on-board UAV or mobile-robot perception, whereas medium and large variants may operate on workstation-class GPUs mounted on autonomous platforms or connected through high-bandwidth edge computing.

Native NMS-free query detection is particularly relevant when inference pipelines must be integrated tightly with control systems, as removing variable post-processing stages can simplify latency characterization. However, end-to-end system latency must still include image acquisition, preprocessing, GPU transfer, inference, output decoding, communication, planning, and control.

9.4 Implications for Edge AI

YOLO27n/s continue the edge-oriented direction established by previous Ultralytics models while introducing a more explicitly efficiency-driven detection head. Removing one prediction scale directly reduces tensor operations and memory movement, which are important considerations for edge accelerators whose performance may be constrained by memory bandwidth rather than arithmetic throughput alone.

The widened high-resolution stage also illustrates that edge optimization does not necessarily mean uniformly reducing network capacity. Strategic redistribution of computation may yield a better accuracy–latency trade-off than simple depth or width reduction.

9.5 Implications for Hybrid CNN–Transformer Vision

YOLO27m/l demonstrate the continuing convergence of CNN and transformer-based visual perception. Convolution provides efficient inductive biases for local structure, whereas attention provides flexible global relationships. Rather than selecting one paradigm exclusively, YOLO27 distributes them according to representational need.

YOLO27l is especially illustrative: early and intermediate processing can exploit efficient local feature extraction, while self-attention is introduced at a deeper, lower-resolution stage where global reasoning becomes computationally practical. The resulting features are then integrated through a familiar FPN/PAN-style neck before query-based detection. This hybrid strategy may represent an increasingly common architecture for future real-time perception systems.

10 Limitations and Open Research Questions

Despite promising preliminary results, several limitations of YOLO27 require further investigation. First, as an upcoming model family, its pretrained weights, final architectural specifications, ablation studies, and reproducibility protocols are not yet fully available; consequently, reported benchmarks may change. Second, comparisons between YOLO27n/s and YOLO27m/l confound model capacity with detection paradigm, motivating controlled comparisons of dense and query-based heads under matched computational budgets. Third, dual-scale detection requires scale-specific evaluation, particularly for small, intermediate-sized, and irregular objects potentially affected by removal of the intermediate feature level. Fourth, foreground-alignment supervision should be validated beyond COCO using UAV, agricultural, industrial, medical, and remote-sensing datasets exhibiting occlusion, imbalance, and annotation noise. Fifth, controlled ablations are needed to separate YOLO27l gains attributable to UltraViT self-attention, query decoding, and increased capacity. Finally, deployment studies should extend beyond high-end GPUs to embedded platforms, reporting latency, memory footprint, energy consumption, thermal behavior, and sustained throughput.

11 Conclusion

YOLO27 represents an important architectural transition within the Ultralytics YOLO family, moving beyond uniform model scaling toward a scale-dependent design strategy. YOLO27n/s prioritize efficient inference through streamlined CNNs, strengthened high-resolution features, dual-scale detection, and foreground-alignment supervision, whereas YOLO27m/l adopt query-based transformer decoding for native NMS-free prediction. YOLO27m retains a convolutional backbone, while YOLO27l incorporates UltraViT and deep-stage self-attention for enhanced global-context modeling. Preliminary COCO results demonstrate 42.3–60.4 $\text{mAP}_{50:95}$ at 640-pixel resolution with reported TensorRT 11 FP16 latency of 0.62–2.32 ms, while YOLO27l reaches 61.2 mAP at 800 pixels. Its extension across segmentation, depth estimation, classification, pose estimation, and oriented detection further positions YOLO27 as a broader real-time visual perception framework.

Future YOLO architectures may advance toward adaptive computation, vision–language and open-vocabulary perception, temporal memory, multimodal sensing, world-model integration, and hardware-aware optimization. These developments could strengthen perception for robotics, autonomous vehicles, UAVs, and embodied AI. Nevertheless, YOLO27's preliminary findings require independent benchmarking, controlled ablations, robustness evaluation, domain-specific validation, and cross-hardware energy–memory profiling to establish generalizability and quantify the individual contributions of its architectural innovations.

[1]

Declartaions

The authors declare that they have no competing interests.

Acknowledgment

The authors acknowledge Ultralytics for making preliminary information and documentation regarding YOLO27 publicly available. The architectural descriptions, model characteristics, supported tasks, and preliminary performance

[1] As the complete YOLO27 model specifications and pretrained weights are not yet publicly available, this overview reflects currently available architectural and preliminary benchmarking information. We will comprehensively update the manuscript, including architectural details, model configurations, weights, controlled benchmarks, ablation analyses, and deployment evaluations, once the complete YOLO27 models, technical specifications, and official weights are released.

results concerning YOLO27 presented in this manuscript are based solely on the official documentation and information provided by Ultralytics. The authors have not independently validated the unreleased YOLO27 models or their reported performance at the time of writing. Accordingly, this manuscript will be updated with complete architectural details, model configurations, pretrained weights, and additional experimental evaluations once the full YOLO27 models, technical specifications, and official weights are publicly released.

References

[1] Zhong-Qiu Zhao, Peng Zheng, Shou-tao Xu, and Xindong Wu. Object detection with deep learning: A review. *IEEE transactions on neural networks and learning systems*, 30(11):3212–3232, 2019.

[2] Zhengxia Zou, Keyan Chen, Zhenwei Shi, Yuhong Guo, and Jieping Ye. Object detection in 20 years: A survey. *Proceedings of the IEEE*, 111(3):257–276, 2023.

[3] Chhavi Rana et al. Artificial intelligence based object detection and traffic prediction by autonomous vehicles–a review. *Expert Systems with Applications*, 255:124664, 2024.

[4] Zohaib Khan, Yue Shen, and Hui Liu. Objectdetection in agriculture: A comprehensive review of methods, applications, challenges, and future directions. *Agriculture*, 15(13):1351, 2025.

[5] Ranjan Sapkota, Dawood Ahmed, and Manoj Karkee. Comparing yolov8 and mask r-cnn for instance segmentation in complex orchard environments. *Artificial Intelligence in Agriculture*, 13:84–99, 2024.

[6] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[7] Ranjan Sapkota, Marco Flores-Calero, Rizwan Qureshi, Chetan Badgujar, Upesh Nepal, Alwin Poulose, Peter Zeno, Uday Bhanu Prakash Vaddevolu, Sheheryar Khan, Maged Shoman, et al. Yolo advances to its genesis: a decadal and comprehensive review of the you only look once (yolo) series. *Artificial Intelligence Review*, 58(9):274, 2025.

[8] Ranjan Sapkota, Zhichao Meng, Martin Churuvija, Xiaoqiang Du, Zenghong Ma, and Manoj Karkee. Comprehensive performance evaluation of yolov12, yolo11, yolov10, yolov9 and yolov8 on detecting and counting fruitlet in complex orchard environments. *Agriculture Communications*, page 100125, 2026.

[9] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017.

[10] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[11] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020.

[12] Chuyi Li, Lulu Li, Hongliang Jiang, Kaiheng Weng, Yifei Geng, Liang Li, Zaidan Ke, Qingyuan Li, Meng Cheng, Weiqiang Nie, et al. Yolov6: A single-stage object detection framework for industrial applications. *arXiv preprint arXiv:2209.02976*, 2022.

[13] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7464–7475, 2023.

[14] Chien-Yao Wang, I-Hau Yeh, and Hong-Yuan Mark Liao. Yolov9: Learning what you want to learn using programmable gradient information. In *European conference on computer vision*, pages 1–21. Springer, 2024.

[15] Ao Wang, Hui Chen, Lihao Liu, Kai Chen, Zijia Lin, Jungong Han, et al. Yolov10: Real-time end-to-end object detection. *Advances in Neural Information Processing Systems*, 37:107984–108011, 2024.

[16] Yunjie Tian, Qixiang Ye, and David Doermann. Yolov12: Attention-centric real-time object detectors. *arXiv preprint arXiv:2502.12524*, 2025.

[17] Mengqi Lei, Siqi Li, Yihong Wu, Han Hu, You Zhou, Xinhu Zheng, Guiguang Ding, Shaoyi Du, Zongze Wu, and Yue Gao. Yolov13: Real-time object detection with hypergraph-enhanced adaptive visual perception. *arXiv preprint arXiv:2506.17733*, 2025.

[18] Glenn Jocher, Jing Qiu, Mengyu Liu, Shuai Lyu, Fatih Cagatay Akyon, and Muhammet Esat Kalfaoglu. Ultralytics yolo26: unified real-time end-to-end vision models. *arXiv preprint arXiv:2606.03748*, 2026.

[19] Ranjan Sapkota and Manoj Karkee. Ultralytics yolo evolution: An overview of yolo27, yolo26, yolo11, yolov8, and yolov5 object detectors for computer vision and pattern recognition. *arXiv preprint arXiv:2510.09653*, 2025.

[20] Licheng Jiao, Mengjiao Wang, Xu Liu, Lingling Li, Fang Liu, Zhixi Feng, Shuyuan Yang, and Biao Hou. Multiscale deep learning for detection and recognition: A comprehensive survey. *IEEE Transactions on Neural Networks and Learning Systems*, 36(4):5900–5920, 2024.

[21] Rezaul Karim, He Zhao, Richard P Wildes, and Mennatullah Siam. Med-vt: Multiscale encoder-decoder video transformer with application to object segmentation. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6323–6333. IEEE, 2023.

[22] Prabodh Kumar Sahoo, Manoj Kumar Panda, Upasana Panigrahi, Ganapati Panda, Prince Jain, Md Shabiul Islam, and Mohammad Tariqul Islam. An improved vgg-19 network induced enhanced feature pooling for precise moving object detection in complex video scenes. *Ieee Access*, 12:45847–45864, 2024.

[23] Yue Wang, Lu Zhang, Pingping Zhang, Yunzhi Zhuge, Junfeng Wu, Hong Yu, and Huchuan Lu. Learning local-global representation for scribble-based rgb-d salient object detection via transformer. *IEEE Transactions on Circuits and Systems for Video Technology*, 34(11):11592–11604, 2024.

[24] Ali Aldubaikhi and Sarosh Patel. Advancements in small-object detection (2023–2025): Approaches, datasets, benchmarks, applications, and practical guidance. *Applied Sciences*, 15(22):11882, 2025.

[25] Muhamad Muzammul and Xi Li. Comprehensive review of deep learning-based tiny object detection: challenges, strategies, and future directions. *Knowledge and Information Systems*, 67(5):3825–3913, 2025.

[26] Aref Miri Rekavandi, Shima Rashidi, Farid Boussaid, Stephen Hoefs, Emre Akbas, and Mohammed Bennamoun. Transformers in small object detection: A benchmark and survey of state-of-the-art. *ACM Computing Surveys*, 58(3):1–33, 2025.

[27] Abdul Mueed Hafiz and Ghulam Mohiuddin Bhat. A survey on instance segmentation: state of the art. *International journal of multimedia information retrieval*, 9(3):171–189, 2020.

[28] Kai Chen, Jiangmiao Pang, Jiaqi Wang, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jianping Shi, Wanli Ouyang, et al. Hybrid task cascade for instance segmentation. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4969–4978. IEEE, 2019.

[29] Gabriela Csurka and Florent Perronnin. An efficient approach to semantic segmentation. *International Journal of Computer Vision*, 95(2):198–212, 2011.

[30] Yanming Guo, Yu Liu, Theodoros Georgiou, and Michael S Lew. A review of semantic segmentation using deep neural networks. *International journal of multimedia information retrieval*, 7(2):87–93, 2018.

[31] Yujian Mo, Yan Wu, Xinneng Yang, Feilin Liu, and Yujun Liao. Review the state-of-the-art technologies of semantic segmentation based on deep learning. *Neurocomputing*, 493:626–646, 2022.

[32] Gabriela Csurka, Riccardo Volpi, and Boris Chidlovskii. Semantic image segmentation: Two decades of research. *Foundations and Trends in Computer Graphics and Vision*, 14(1-2):1–162, 2022.

[33] Anurag Arnab and Philip HS Torr. Pixelwise instance segmentation with a dynamically instantiated network. In *2017 IEEE conference on computer vision and pattern recognition (CVPR)*, pages 879–888. IEEE, 2017.

[34] Chengxiang Yin, Jian Tang, Tongtong Yuan, Zhiyuan Xu, and Yanzhi Wang. Bridging the gap between semantic segmentation and instance segmentation. *IEEE Transactions on Multimedia*, 24:4183–4196, 2021.

[35] Kun Wang, Zi Wang, Zhang Li, Ang Su, Xichao Teng, Erting Pan, Minhao Liu, and Qifeng Yu. Oriented object detection in optical remote sensing images using deep learning: a survey: K. wang et al. *Artificial Intelligence Review*, 58(11):350, 2025.

[36] Mohsen Zand, Ali Etemad, and Michael Greenspan. Oriented bounding boxes for small and freely rotated objects. *IEEE Transactions on Geoscience and Remote Sensing*, 60:1–15, 2021.

[37] Gui-Song Xia, Xiang Bai, Jian Ding, Zhen Zhu, Serge Belongie, Jiebo Luo, Mihai Datcu, Marcello Pelillo, and Liangpei Zhang. Dota: A large-scale dataset for object detection in aerial images. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3974–3983. IEEE, 2018.

[38] Jian Ding, Nan Xue, Yang Long, Gui-Song Xia, and Qikai Lu. Learning roi transformer for oriented object detection in aerial images. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2844–2853. IEEE, 2019.

[39] Huiying Wang, Chunping Wang, Qiang Fu, Dongdong Zhang, Renke Kou, Ying Yu, and Jian Song. Cross-modal oriented object detection of uav aerial images based on image feature. *IEEE Transactions on Geoscience and Remote Sensing*, 62:1–21, 2024.

[40] Long Wen, Yu Cheng, Yi Fang, and Xinyu Li. A comprehensive survey of oriented object detection in remote sensing images. *Expert Systems with Applications*, 224:119960, 2023.

[41] Cong Zhang, Jingran Su, Yakun Ju, Kin-Man Lam, and Qi Wang. Efficient inductive vision transformer for oriented object detection in remote sensing imagery. *IEEE Transactions on Geoscience and Remote Sensing*, 61:1–20, 2023.

[42] Dawen Yu and Shunping Ji. A new spatial-oriented object detection framework for remote sensing images. *IEEE Transactions on Geoscience and Remote Sensing*, 60:1–16, 2021.

[43] Kai Yao, Alberto Ortiz, and Francisco Bonnin-Pascual. A dcnn-based arbitrarily-oriented object detector with application to quality control and inspection. *Computers in Industry*, 142:103737, 2022.

[44] Yi Yu and Feipeng Da. Phase-shifting coder: Predicting accurate orientation in oriented object detection. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13354–13363. IEEE, 2023.

[45] Jiaying Lin, Phillip Diekmann, Christian-Eike Framing, Rene Zweigel, and Dirk Abel. Maritime environment perception based on deep learning. *IEEE Transactions on Intelligent Transportation Systems*, 23(9):15487–15497, 2022.

[46] Sol Han, Dongwook Lee, Jonghwi Kim, and Jinwhan Kim. A survey on maritime perception datasets and technologies for autonomous surface vessels. *Intelligent Service Robotics*, 19(2):25, 2026.

[47] Davide Falanga, Suseong Kim, and Davide Scaramuzza. How fast is too fast? the role of perception latency in high-speed sense and avoid. *IEEE Robotics and Automation Letters*, 4(2):1884–1891, 2019.

[48] Davide Falanga, Philipp Foehn, Peng Lu, and Davide Scaramuzza. Pampc: Perception-aware model predictive control for quadrotors. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–8. IEEE, 2018.

[49] Liangkai Liu, Zheng Dong, Yanzhi Wang, and Weisong Shi. Prophet: Realizing a predictable real-time perception pipeline for autonomous vehicles. In *2022 IEEE Real-Time Systems Symposium (RTSS)*, pages 305–317. IEEE, 2022.

[50] Tobias Blass, Arne Hamann, Ralph Lange, Dirk Ziegenbein, and Björn B Brandenburg. Automatic latency management for ros 2: Benefits, challenges, and open problems. In *2021 IEEE 27th Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 264–277. IEEE, 2021.