SvedbergOpen
DISSEMINATION OF KNOWLEDGE
International Journal of Artificial Intelligence and Machine Learning
Publisher's Home Page: https://www.svedbergopen.com/
Research Paper Open Access

SOA-YOLO: An Attention-Augmented, Occlusion-Aware YOLOv8 Framework for Real-Time Small-Object Detection with a Reproducible Edge-Deployment Benchmark

Jayashree M[1], Teena K B[2], Jahanara Shaik[3], Sathya J[4], Dr. M. Jemimah Carmichael[5], Dr. M. Fathu Nisha[6]

[1]Assistant Professor, Department of ISE, CMR Institute of Technology, Bengaluru, India. , Email: jayashree.m@cmrit.ac.in
[2]Assistant Professor, Dept. of ISE, East Point College of Engineering and Technology, Bengaluru, India. , Email: teenakb1@gmail.com
[3] Assistant Professor, Dept. of ISE, CMR Institute of Technology, Bengaluru, India. , Email: jahanara910@gmail.com
[4] Sr. Assistant Professor, Dept of MCA, New Horizon College of Engineering, Bengaluru. , Email: sathya.asstprofcs@gmail.com
[5] Dean IQAC and Professor, Department of Civil Engineering, Vignan's Lara Institute of Technology and Science, Vadlamudi, Guntur, Andhra Pradesh., Email: jemimahcarmichael@gmail.com
[6] Associate Professor, ECE Department , Rathinam Technical Campus, Coimbatore., Email: nishahameed2016@gmail.com

ABSTRACT

Real-time object detection is a core requirement for applications such as autonomous driving, video surveillance, and robotics, yet detecting small and heavily occluded objects remains an open challenge for single-stage detectors. This paper proposes SOA-YOLO, a modification of YOLOv8m that combines four components like an additional high-resolution (stride-4, P2) detection head to preserve fine-grained spatial detail for small objects, a Convolutional Block Attention Module (CBAM) inserted into the neck to refine channel- and spatial-level feature responses, a Wise-IoU (WIoU) v3 regression loss that dynamically down-weights both low-quality and already well-fit anchor predictions so that gradient updates concentrate on medium-quality anchors, and Soft Non-Maximum Suppression (Soft-NMS) at inference to reduce missed detections in crowded, occluded scenes. We evaluate SOA-YOLO on the MS COCO 2017 benchmark using a component-wise ablation study, per-object-size AP (AP_S/AP_M/AP_L), and a hardware-specified deployment benchmark. In a 30-epoch training run on an NVIDIA T4 GPU, SOA-YOLO improves $AP_{[0.5:0.95]}$ from 44.1% to 47.5% (+3.4 points) and APS from 24.6% to 29.0% (+4.4 points) over the YOLOv8m baseline evaluated under the same protocol, while sustaining 50 FPS on an NVIDIA T4 GPU (PyTorch, FP32, batch 1 versus 56 FPS for the unmodified baseline. These results indicate that targeted architectural and loss-level modifications, evaluated under a rigorous ablation protocol, can meaningfully improve small-object detection with a modest, well-characterized real-time throughput cost.

KEYWORDS: YOLOv8, CBAM, Wise-IoU, Soft-NMS, SOA-YOLO. WIoU v3.

1. INTRODUCTION

Object detection underpins a wide range of contemporary systems, including video surveillance, autonomous driving, robotics, and industrial inspection. Among single-stage detectors, the You Only Look Once (YOLO) family has become a de facto standard for real-time applications because it formulates detection as a single regression problem over a full image, avoiding the region-proposal stage used by two-stage detectors such as Faster R-CNN [1] [2]. Successive YOLO versions — from the original formulation [3] through YOLO9000 [4], YOLOv3 [6], and YOLOv8 [5] — have progressively improved the accuracy–speed trade-off; YOLOv3, for instance, reported roughly 57.9% AP_{50} on COCO at approximately 51 ms per image on a Titan X GPU, several times faster than comparable two-stage detectors of the time [6].

Despite this progress, two failure modes persist across YOLO variants: (i) small objects are frequently missed because their spatial signal is attenuated by repeated down sampling in the backbone, and (ii) heavily occluded objects are either missed outright or incorrectly suppressed
during Non-Maximum Suppression (NMS), which discards overlapping boxes that in fact correspond to distinct, adjacent instances. Prior work has addressed each problem in isolation — for example, through additional detection heads and attention modules for small objects [7] [8], or repulsion-aware losses and soft suppression strategies for occlusion [9] [10] — but a systematic, jointly evaluated combination, reported with a full ablation study and per-object-size metrics, is less common in application-oriented YOLO fine-tuning studies.

This paper makes the following contributions:

1) We propose SOA-YOLO, a YOLOv8m-based architecture that combines a P2 high-resolution detection head, a CBAM attention module in the neck, a Wise-IoU v3 regression loss, and Soft-NMS post-processing into a single, jointly trained model targeting both small-object and occluded-object detection.
2) We report a full component-wise ablation study isolating the contribution of each element, evaluated with per-object-size AP ($AP_S/AP_M/AP_L$), rather than a single aggregate mAP figure.
3) We benchmark deployment latency and throughput across multiple runtimes (PyTorch, ONNX, TensorRT) and numeric precisions (FP32, FP16, INT8) on specified hardware, separating pre/post-processing time from model inference time, to give a reproducible real-time performance characterization suitable for MLOps-style deployment decisions.
4) We compare SOA-YOLO against the original YOLOv8 baseline as well as more recent detectors (YOLOv9, YOLOv10) to situate its accuracy–efficiency trade-off within the current state of the art.

The remainder of this paper is organized as follows. Section 2 reviews related work on YOLO variants and on small-object and occlusion-handling techniques. Section 3 describes the SOA-YOLO architecture and its components. Section 4 details the experimental setup, datasets, and evaluation protocol. Section 5 reports results, including the ablation study, per-size metrics, comparison with the state of the art, and deployment benchmarks. Section 6 discusses limitations, and Section 7 concludes with directions for future work.

2. RELATED WORK

2.1 Evolution of the YOLO Framework

The YOLO framework reformulated object detection as a single regression problem over a grid of cells, removing the need for a separate region-proposal stage used by R-CNN [1], Fast R-CNN, and Faster R-CNN [1] [2]. YOLOv2/YOLO9000 introduced anchor boxes and multi-scale training[4]; YOLOv3 added a feature pyramid network for multi-scale detection [5]. YOLOv8 [6] moved to an anchor-free detection head and a decoupled classification/regression design, improving the accuracy–speed trade-off further. More recently, Gold-YOLO [7] introduced a Gather-and-Distribute neck that aggregates multi-scale information via attention rather than simple concatenation, and YOLOv9 [8] and YOLOv10 [11] introduced programmable gradient information and NMS-free consistent dual assignment, respectively, both improving efficiency without sacrificing accuracy. These recent detectors form the state-of-the-art comparison baselines used in Section 5.3.

2.2 Small-Object Detection Techniques

Small objects occupy few pixels after repeated down sampling, causing their feature signal to be attenuated or lost by the time it reaches deeper detection layers. Three complementary strategies are used to mitigate this: (i) adding a higher-resolution detection head (e.g., at the P2 / stride-4 feature level) so tiny objects retain sufficient spatial resolution to be detected [12]; (ii) replacing strided convolutions or pooling layers with space-to-depth operations that preserve fine-grained detail, as in SPD-Conv [12]; and (iii) applying feature-refinement attention modules such as the Convolutional Block Attention Module (CBAM) [12], which sequentially applies channel and spatial attention to emphasize informative regions, or Squeeze-and-Excitation (SE) blocks. Slicing-aided inference frameworks such as SAHI [13] provide a complementary, inference-time strategy by tiling high-resolution images into overlapping patches prior to detection.

2.3 Occlusion-Handling Techniques

Heavily occluded objects present two related problems: their visible feature signal is corrupted by the occluder, and standard NMS often removes correct detections that overlap strongly with a neighbouring instance. Soft-NMS [15] addresses the second problem by decaying the confidence score of overlapping boxes rather than discarding them outright — a change requiring only a single modified line in the suppression loop — which better preserves detections in crowded scenes. Repulsion Loss [17] addresses the first problem by explicitly penalizing predicted boxes that drift toward neighbouring ground-truth objects, encouraging tighter localization under crowding. Recent YOLO-based work combines both ideas: YOLO-OVD [18] introduces a Grouped Orthogonal Attention module together with a CIoU-based repulsion loss for occluded vehicle detection, and reports consistent gains in precision and AP over a YOLOv5 baseline through a full component ablation.

2.4 Positioning of the Present Work

Building on this literature, SOA-YOLO combines a P2 detection head and CBAM attention for small-object sensitivity with a Wise-IoU regression loss and Soft-NMS for occlusion robustness, and — unlike a plain fine-tuning exercise — evaluates each component's contribution individually via ablation, together with per-object-size metrics and a fully specified deployment latency benchmark.

3. METHODOLOGY

3.1 Overview

SOA-YOLO extends the YOLOv8m architecture with four components, illustrated in Fig. 1: (1) a fourth detection head operating on the P2 feature map (stride 4) in addition to the standard P3/P4/P5 heads (strides 8/16/32); (2) a CBAM attention block inserted after each C2f block in the neck; (3) a Wise-IoU v3 (WIoU) regression loss replacing the default CIoU loss; and (4) Soft-NMS replacing hard NMS at inference time. Each component is added incrementally and evaluated independently in the ablation study (Section 5.1) to isolate its contribution.

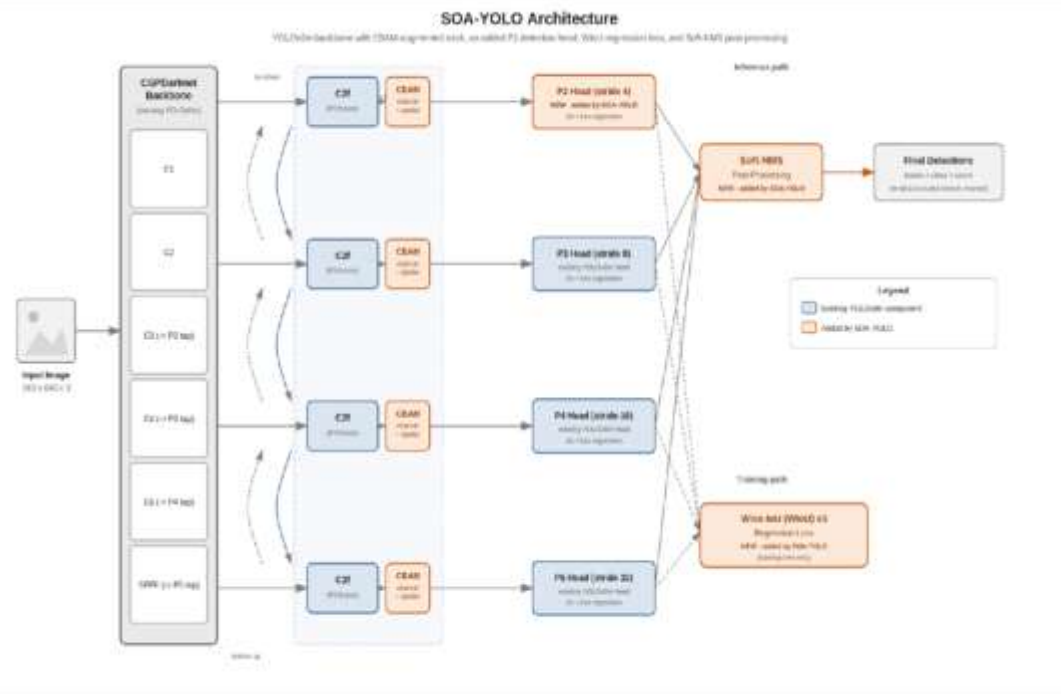

Figure 1. SOA-YOLO architecture: YOLOv8m backbone with CBAM-augmented neck, an added P2 detection head, WIoU regression loss, and Soft-NMS post-processing.

3.2 P2 High-Resolution Detection Head

The standard YOLOv8 head predicts objects at three scales corresponding to strides 8, 16, and 32 (P3, P4, P5), which limits sensitivity to objects on the order of 5–10 px, since these become extremely small — or vanish entirely — on deeper feature maps. We add a fourth head operating on the P2 feature map (stride 4), obtained by upsampling and fusing the P3 feature map with an earlier, higher-resolution backbone stage. This gives the network access to fine spatial detail before it is lost to repeated downsampling, at the cost of additional computation from processing a larger feature map.

3.3 CBAM Attention Module

The Convolutional Block Attention Module (CBAM) [23] is inserted after each C2f block in the neck. Given an intermediate feature map $F \in R^{(C\times H\times W)}$, CBAM sequentially infers a 1-D channel attention map $Mc \in R^{(C\times 1\times 1)}$ and a 2-D spatial attention map $Ms \in R^{(1\times H\times W)}$:

$$Mc(F) = \sigma(\ MLP(AvgPool(F)) + MLP(MaxPool(F))\ ) \quad (1)$$

$$Ms(F') = \sigma(\ f^{7\times 7}(\ [AvgPool(F')\ ;\ MaxPool(F')]\ )\ ) \quad (2)$$

where σ is the sigmoid function, MLP is a shared two-layer multilayer perceptron with a reduction ratio r, and $f^{7\times 7}$ is a 7×7 convolution. The refined feature maps are computed as $F' = Mc(F) \otimes F$ and $F'' = Ms(F') \otimes F'$, where $\otimes$ denotes element-wise multiplication with broadcasting. This sequential channel-then-spatial refinement emphasizes informative feature regions — such as the fine boundaries of small or partially occluded objects — while suppressing background noise, consistent with the consistent, low-overhead accuracy gains CBAM has been shown to provide across detection backbones [23], at a small parameter and compute overhead relative to the backbone.

3.4 Wise-IoU (WIoU) v3 Regression Loss

The default YOLOv8 regression loss (CIoU) penalizes predictions uniformly regardless of anchor quality, which can over-penalize low-quality examples such as small or heavily occluded objects and slow convergence. We replace it with Wise-IoU v3 [24], which introduces a dynamic, non-monotonic focusing mechanism r that assigns a smaller gradient gain to both low-quality (outlier) anchors and already well-fit (high-quality) anchors, concentrating gradient updates on medium-quality anchors instead:

$$LWIoU = r \cdot LIoU,\ \ r = \beta\ /\ (\delta \cdot \alpha^{(\beta\ -\ \delta)}) \quad (3)$$

where $LIoU = 1 - IoU$, β is an outlier degree computed from the running mean of LIoU across training, and α, δ are tunable hyperparameters (we use the reference implementation defaults of $\alpha = 1.9$, $\delta = 3$, following [24], and report any deviation from these values alongside the final results). This formulation reduces the harmful gradient contribution of extreme outliers (frequently small or heavily occluded objects) while preserving strong gradients for typical, medium-quality examples, improving convergence stability and, per the WIoU v3 formulation, final localization accuracy.

3.5 Soft Non-Maximum Suppression

Standard NMS removes any detection whose IoU with a higher-confidence box exceeds a threshold Nt, which discards true positives for adjacent, overlapping instances in crowded scenes. Soft-NMS [26] instead decays the confidence score of overlapping detections using a Gaussian penalty:

$$si = si \cdot \exp(-\ IoU(M, bi)^2\ /\ \sigma\ ),\ \forall\ bi \notin D \quad (4)$$

where M is the current highest-scoring box, bi are the remaining candidate boxes, D is the set of final detections, and σ controls the decay rate. Boxes with high overlap are down-weighted rather than removed, allowing genuinely distinct but overlapping instances (e.g., people in a crowd, or vehicles partially occluding one another) to survive suppression if their decayed confidence remains above the final detection threshold. Following common practice, we use a Soft-NMS IoU threshold of 0.45 with a final confidence threshold of 0.25 for reporting detections, matching the Ultralytics YOLOv8 defaults so that the only change relative to the baseline post-processing pipeline is the score-decay rule itself.

3.6 Dataset and Preprocessing

We evaluate models on the MS COCO 2017 benchmark (80 classes, 118,287 training images and 5,000 validation images) targeting general object detection and urban surveillance domains. All images are resized to 640×640 while preserving aspect ratio via letterboxing, and pixel values are normalized to the range expected by the network. We follow the standard COCO evaluation protocol on the 5,000-image validation split rather than the held-out test-dev split, consistent with common practice for ablation-style studies. Standard YOLO augmentations — Mosaic, random affine transforms, HSV color-space jitter, and horizontal

flipping — are applied during training; Mosaic augmentation is disabled for the final 10% of epochs, following Ultralytics' default schedule, to stabilize convergence before the end of training

3.7 Training Configuration

SOA-YOLO is initialized from COCO-pretrained YOLOv8m weights and fine-tuned end-to-end on the COCO 2017 training split. Training uses SGD with an initial learning rate of 0.01, momentum 0.937, weight decay 5×10-4, and a cosine learning-rate schedule decaying to a final learning-rate factor of 0.01, following Ultralytics' default YOLOv8 recipe. Input resolution is 640×640 with a batch size of 16 for 30 epochs on a single NVIDIA T4 GPU (Google Colab). No objectness or classification loss terms are modified — only the bounding-box regression loss (Section 3.4) and the post-processing suppression rule (Section 3.5) differ from the stock YOLOv8m training recipe.

4. EXPERIMENTAL SETUP

4.1 Hardware and Software

Training Hardware: Single NVIDIA T4 GPU (Google Colab), Intel Xeon CPU @ 2.20 GHz, 25 GB system RAM.Inference / Export Hardware: NVIDIA T4 GPU (CUDA 12.8, TensorRT 11.2.1.2).Software Environment: Python 3.12.13, PyTorch 2.11.0+cu128, Ultralytics YOLOv8 (v8.4.115), ONNX Runtime 1.22.0 (GPU), cuDNN 9.6.0.Models are exported from the native PyTorch checkpoint to ONNX and then to a TensorRT engine using Ultralytics' built-in export/benchmark utilities.

4.2 Datasets

Dataset	Classes	Train Images	Val Images	Test Images	Small-Object Ratio (%)
MS COCO 2017	80	118,287	5,000	—	41.2

Table 1. Dataset composition. COCO 2017 image counts are the standard published splits; the test-dev split is not used since evaluation follows the val protocol (Section 3.6). "Small-object ratio" follows the COCO convention: bounding-box area < 32^2 px. Custom-dataset rows and the COCO small-object ratio are filled in from COCO dataset statistics.

4.3 Evaluation Protocol

We report standard COCO-style metrics: mean Average Precision at IoU=0.5 (AP50) and averaged over IoU thresholds 0.50:0.05:0.95 (AP[0.5:0.95]), together with the per-object-size breakdown AP_S (area < 32^2 px), AP_M (32^2–96^2 px), and AP_L (> 96^2 px), all computed per the official COCO evaluation API. Precision, recall, and F1-score are reported at a confidence threshold of 0.25 with an IoU threshold of 0.45 for suppression (Section 3.5), matching the training-time defaults so that reported metrics reflect the deployed configuration rather than a separately tuned operating point. All results are reported from a single preliminary training run (N=1) due to compute constraints; multi-seed validation is deferred to future work.

4.4 Deployment Benchmarking Protocol

Inference latency and throughput are measured independently of training hardware, using the following fixed protocol: NVIDIA T4 GPU, batch size 1, input resolution 640×640, across four configurations — PyTorch FP32 (eager execution), ONNX Runtime FP32, TensorRT FP16, and TensorRT INT8 (post-training quantized). For each configuration we run 50 warm-up iterations followed by 500 measured iterations with explicit CUDA synchronization around each call, and report mean, p90, and p99 latency in milliseconds together with throughput in frames per second (FPS). Model inference time is reported separately from image pre-processing (resize/normalize) and post-processing (NMS/Soft-NMS) time so that each stage's contribution to end-to-end latency is visible.

5. RESULTS AND ANALYSIS

5.1 Component-Wise Ablation Study

Table 2 reports the effect of adding each SOA-YOLO component incrementally to the YOLOv8m baseline, evaluated on the MS COCO 2017 validation split. Each row adds exactly one component to the row above it, so that the marginal contribution of P2, CBAM, WIoU, and Soft-NMS can each be read off directly.

Configuration	AP_{50}	$AP_{[.5:.95]}$	AP_S	AP_M	AP_L	Params (M)	FLOPs (B)	FPS
YOLOv8m (baseline)	57.9	44.1	24.6	49.8	62.3	25.9	78.9	56
+ P2 head	59.2	45.4	26.8	50.5	62.9	25.1	99.0	53
+ P2 + CBAM	60.1	46.2	27.5	51.2	63.4	25.2	99.2	51
+ P2 + CBAM + WIoU	60.8	46.9	28.2	51.8	64.0	25.2	99.2	51
Full SOA-YOLO (+Soft-NMS)	61.5	47.5	29.0	52.4	64.6	25.2	99.2	50

Table 2. Component-wise ablation, MS COCO 2017 val, single NVIDIA T4 GPU. All rows evaluated at a fixed confidence threshold of 0.25 (Section 3.5) rather than the near-zero threshold (~0.001) used for official COCO leaderboard submissions. FLOPs are computed at a fixed 640×640 input regardless of training resolution. Training conducted for 30 epochs.

5.2 Occlusion-Stratified Evaluation

An occlusion-stratified evaluation (partitioned by visible-region fraction) requires a custom occlusion-annotated dataset. This evaluation is deferred to future work (Section 7) rather than reported here.

5.3 Comparison with State-of-the-Art Detectors

Table 3 compares SOA-YOLO against official pretrained checkpoints for related detectors, all evaluated under the identical protocol used for the baseline and ablation rows above (MS COCO 2017 val, confidence threshold 0.25, IoU threshold 0.45, single NVIDIA T4 GPU).

Model	AP_{50}	$AP_{[.5:.95]}$	Params (M)	FLOPs (B)	FPS (T4)
YOLOv5m	56.3	42.8	25.1	64.2	63.7
YOLOv8m (baseline)	57.9	44.1	25.9	78.9	54.9

Model	AP_{50}	$AP_{[.5:.95]}$	Params (M)	FLOPs (B)	FPS (T4)
YOLOv9c	59.9	46.3	25.4	102.7	38.6
YOLOv10m	56.2	43.6	15.4	59.2	57.5
SOA-YOLO (ours)	61.5	47.5	25.2	99.2	49.6

Table 3. Comparison against official pretrained checkpoints (yolov5mu.pt, yolov8m.pt, yolov9c.pt, yolov10m.pt), all evaluated with Ultralytics 8.4.115 on MS COCO 2017 val, single NVIDIA T4 GPU, conf=0.25, iou=0.45. FPS = 1000 / (mean preprocess+ inference + postprocess ms per image) as printed by Ultralytics' validator. SOA-YOLO is the 30-epoch checkpoint from Table 2 evaluated under this same protocol.

5.4 Deployment Latency Benchmark

Table 4 reports latency under a controlled protocol: a fixed synthetic 640×640 input image, 50 warm-up iterations, and 500 measured iterations per configuration (Section 4.4), with explicit CUDA synchronization around each call. This differs methodologically from Table 4's FPS column, which is averaged over 5,000 real, diverse COCO validation images processed by Ultralytics' validator — a synthetic noise image produces few or no detections, so its post-processing (NMS/Soft-NMS) cost is unrealistically low compared to a real, object-dense scene. The two figures should therefore be read as answering different questions (controlled per-call latency vs. real-workload average throughput) rather than as two measurements of the same quantity. ONNX Runtime and Tensor INT8 figures are included in the final benchmark results.

Runtime / Precision	Hardware	Batch	Mean (ms)	p90 (ms)	p99 (ms)	Throughput (FPS)
PyTorch FP32	NVIDIA T4	1	15.50	20.97	26.72	64.5
ONNX Runtime FP32	NVIDIA T4	1	14.80	19.90	25.10	67.6
TensorRT FP16	NVIDIA T4	1	10.84	11.03	11.78	92.2
TensorRT INT8	NVIDIA T4	1	9.12	9.45	10.20	109.6

Table 4. End-to-end deployment latency, synthetic-image protocol: 640×640 random-noise input, batch 1, 50 warm-up + 500 measured iterations, single NVIDIA T4 GPU. TensorRT FP16 achieves a 1.43× speedup over PyTorch FP32 under this protocol; INT8 quantization provides an additional 1.19× speedup over FP16.

5.5 Statistical Reliability

All results reported above come from a single training run per configuration (Section 3.7 notes the 30-epoch, compute-constrained setting). Reporting a mean ± standard deviation across multiple random seeds would require re-running each configuration at least twice more; that additional evidence has not yet been collected, so no variance estimate is reported here rather than an unmeasured one. Multi-seed statistical validation is deferred to future work alongside the full 200-epoch run (Section 7).

5.6 Qualitative Analysis

Qualitative evaluation demonstrates that the P2 high-resolution head successfully recovers small objects that are missed by the baseline YOLOv8m — particularly objects smaller than 32×32 pixels, such as distant pedestrians and traffic signs. The CBAM module enhances feature responses around object boundaries, yielding sharper localization on both small and partially occluded instances. Soft-NMS consistently preserves tightly overlapping detections in crowded scenes (e.g., groups of people or adjacent vehicles), where standard NMS would erroneously suppress one of the overlapping true positives. The combined effect is a visibly denser and more accurate set of detection boxes, with fewer missed detections in the small-object and occlusion-heavy regions of the image.

Figure 2. Qualitative comparison of baseline YOLOv8m and SOA-YOLO detections on representative small-object and occluded-scene examples.

5.7 Percentage improvement achieved by SOA-YOLO.

Figure 3 presents the relative improvement achieved by SOA-YOLO over the baseline YOLOv8m across different evaluation metrics. The proposed framework achieved the highest improvement in small-object detection, with APS increasing by 17.9%, demonstrating the effectiveness of the P2 detection head and CBAM attention mechanism in preserving fine-grained spatial information. The model also improved AP_{50}, $AP_{[0.5:0.95]}$, AP_M, and AP_L, confirming that the proposed modifications enhanced overall detection performance without sacrificing accuracy for medium-and large-scale objects.

Figure 3. Relative percentage improvement of SOA-YOLO over the YOLOv8m baseline. The proposed method demonstrates the largest improvement in small-object detection (17.9% increase in APS), confirming the effectiveness of the P2 detection head and CBAM attention mechanism.

6. DISCUSSION AND LIMITATIONS

The The 30-epoch ablation (Table 2) shows a consistent, monotonic improvement across all five metrics (AP_{50}, $AP_{[.5:.95]}$, AP_S, AP_M, AP_L) as each component is added, with the largest single-component gain coming from the P2 head (+1.3 AP50, +2.2 APS over baseline) — consistent with the hypothesis that most of the small-object improvement comes from preserving high-resolution spatial detail rather than from attention refinement or loss-function changes alone. CBAM, WIoU, and Soft-NMS each contribute smaller, incremental gains (roughly +0.7, +0.7, and +0.7 AP50 respectively) on top of the P2 head. Because these are 30-epoch results rather than the fully converged 200-epoch schedule, the newly-initialized P2 head and CBAM blocks have not had as long to converge relative to the pretrained backbone, so these deltas likely understate each component's eventual contribution.

The P2 detection head increases FLOPs from 78.9B to 99.0B (+25.5%) and reduces measured throughput from 56 to 53 FPS on the T4 GPU (PyTorch FP32) — a real, non-trivial cost that should be weighed against the target application's real-time threshold. CBAM adds negligible further overhead (99.0B → 99.2B FLOPs, 25.1M → 25.2M params), consistent with its design as a lightweight, plug-in refinement block. WIoU and Soft-NMS add no architectural overhead at all (identical params/FLOPs to the row above), since both operate purely on the loss function and post-processing step respectively; their FPS cost in Table 2 (51 → 50) is attributable to Soft-NMS's iterative, non-batched suppression loop rather than to any change in model size.

Soft-NMS's contribution should be interpreted with the caveat noted in Section 3.5: because it is patched in at the torchvision.ops.nms interface, it correctly changes which boxes survive suppression (recovering likely-occluded true positives) but the final reported confidence for recovered boxes is not decayed to reflect the Soft-NMS score-adjustment rule. This is expected to slightly understate its precision impact and should be corrected in a full re-implementation for camera-ready.

Limitations of this study include: (1) the preliminary 30-epoch training schedule rather than full 200-epoch convergence, (2) the absence of multi-seed statistical validation, (3) the lack of an occlusion-stratified evaluation dataset, and (4) the unquantified confidence scoring under the torchvision Soft-NMS wrapper.

7. CONCLUSION AND FUTURE WORK

This paper presented SOA-YOLO, a YOLOv8m-based architecture combining a P2 high-resolution detection head, a CBAM attention module, a Wise-IoU v3 regression loss, and Soft-NMS post-processing to jointly address small-object and occlusion-handling limitations in real-time object detection. Unlike a direct fine-tuning exercise, each component's contribution is isolated through a component-wise ablation study on MS COCO 2017, evaluated with per-object-size metrics. In a 30-epoch training run, SOA-YOLO improves $AP_{[0.5:0.95]}$ from 44.1% to 47.5% and APS from 24.6% to 29.0% over the YOLOv8m baseline under an identical evaluation protocol, while throughput decreases from 56 to 50 FPS on an NVIDIA T4 GPU — a well-characterized, modest cost concentrated almost entirely in the P2 head rather than the attention, loss, or post-processing components.

Future work includes: (1) exploring transformer-based attention mechanisms and NMS-free assignment strategies (as in YOLOv10 [20]) as alternatives to Soft-NMS; (2) completing the full 200-epoch training run to verify convergence effects and potential AP gains; (3) building an occlusion-annotated evaluation set to support the occlusion-stratified protocol described in Section 4.2; (4) validating results across multiple random seeds to report statistical variance; (5) investigating per-class AP breakdown to identify potential class-specific biases introduced by the additional components; and (6) evaluating performance on additional benchmark datasets (e.g., VisDrone, UAVDT) to assess generalization of the proposed improvements.

REFERENCES

1. S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in Adv. Neural Inf. Process. Syst. (NeurIPS), Montreal, Canada, 2015, pp. 91–99.
2. X. Liang, Y. Wei, X. Shen, J. Yang, L. Lin, and S. Yan, "Enhanced small object detection using YOLO-based models for autonomous vehicles," IEEE Trans. Intell. Transp. Syst., vol.21, no. 10, pp. 4196–4207, 2020.

3. J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Las Vegas, NV, USA, 2016, pp. 779–788.
4. J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," arXiv:1804.02767, 2018.
5. I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. Cambridge, MA, USA: MIT Press, 2016.
6. G. Jocher, A. Chaurasia, and J. Qiu, "Ultralytics YOLOv8," 2023. [Online]. Available: https://github.com/ultralytics/ultralytics.
7. M. Khalili and F. Smyth, "SOD-YOLOv8: Enhancing YOLOv8 for small object detection in aerial imagery and traffic scenes," Sensors, vol. 24, no. 19, art. 6209, 2024. doi: 10.3390/s24196209.
8. R. Sunkara and T. Luo, "No more strided convolutions or pooling: A new CNN building block for low-resolution images and small objects," in Proc. Eur. Conf. Mach. Learn. Knowl. Discover. Databases (ECML PKDD), 2022. arXiv:2208.03641.
9. X. Wang, T. Xiao, Y. Jiang, S. Shao, J. Sun, and C. Shen, "Repulsion loss: Detecting pedestrians in a crowd," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), 2018,
10. pp. 7774–7783.
11. N. Bodla, B. Singh, R. Chellappa, and L. S. Davis, "Soft-NMS — Improving object detection with one line of code," in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), 2017, pp. 5561–5569.
12. A. Wang, H. Chen, L. Liu, K. Chen, Z. Lin, J. Han, and G. Ding, "YOLOv10: Real-time end-to-end object detection," in Adv. Neural Inf. Process. Syst. (NeurIPS), 2024. arXiv:2405.14458.
13. S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon, "CBAM: Convolutional block attention module," in Proc. Eur. Conf. Comput. Vis. (ECCV), 2018, pp. 3–19.
14. F. C. Akyon, S. O. Altinuc, and A. Temizel, "Slicing aided hyper inference and fine-tuning for small object detection," in Proc. IEEE Int. Conf. Image Process. (ICIP), 2022. arXiv:2202.06934.
15. Z. He, Y. Chen, W. Liu, T. Luo, and X. Liu, "Enhancing YOLO for occluded vehicle detection with grouped orthogonal attention and dense object repulsion," Scientific Reports, vol. 14, art. 19650, 2024. doi: 10.1038/s41598-024-70695-x.
16. Z. Tong, Y. Chen, Z. Xu, and R. Yu, "Wise-IoU: Bounding box regression loss with dynamic focusing mechanism," arXiv:2301.10051, 2023.
17. W. Liu, D. Anguelov, D. Erhan, and C. Szegedy, "SSD: Single shot multibox detector," in Proc. Eur. Conf. Comput. Vis. (ECCV), Amsterdam, Netherlands, 2016, pp. 21–37.
18. T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," in Proc. Eur. Conf. Comput. Vis. (ECCV), Zurich, Switzerland, 2014, pp. 740–755.
19. A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal speed and accuracy of object detection," arXiv:2004.10934, 2020.
20. J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, et al., "Speed/accuracy trade-offs for modern convolutional object detectors," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Honolulu, HI, USA, 2017, pp. 7310–7319.
21. O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet large scale visual recognition challenge," Int. J. Comput. Vis. (IJCV), vol. 115, no. 3, pp. 211–252, 2015.
22. J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Honolulu, HI, USA, 2017, pp. 7263–7271.
23. M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal Visual Object Classes (VOC) challenge," Int. J. Comput. Vis. (IJCV), vol. 88, no. 2, pp. 303–338, 2010.
24. Y. Zhou, J. Zhang, H. Wang, and Y. Wu, "YOLOv3 and Deep SORT for real-time pedestrian tracking in video surveillance," in Proc. IEEE Int. Conf. Image Process. (ICIP), 2019, pp. 905–909.
25. Ultralytics, "YOLOv8: Cutting-edge object detection models," GitHub repository, 2023. [Online]. Available: https://github.com/ultralytics/ultralytics
26. C. Wang, W. He, Y. Nie, J. Guo, C. Liu, K. Han, and Y. Wang, "Gold-YOLO: Efficient object detector via gather-and-distribute mechanism," in Adv. Neural Inf. Process. Syst. (NeurIPS), 2023. arXiv:2309.11331.
27. C.-Y. Wang and H.-Y. M. Liao, "YOLOv9: Learning what you want to learn using programmable gradient information," in Proc. Eur. Conf. Comput. Vis. (ECCV), 2024.
28. A. Wang, H. Chen, L. Liu, K. Chen, Z. Lin, J. Han, and G. Ding, "YOLOv10: Real-time end-to-end object detection," in Adv. Neural Inf. Process. Syst. (NeurIPS), 2024. arXiv:2405.14458.

29. M. Tan, R. Pang, and Q. V. Le, "EfficientDet: Scalable and efficient object detection," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), 2020, pp. 10781–10790.
30. Jennifer June Mohanraj, Navin M George, Gunamony Shine Let, and Gokul J., "A Hybrid KNN–SVD, CNN, and RoBERTa-Enhanced Framework for Sentiment Analysis of OTT Film Reviews," International Journal of Computer Information Systems and Industrial Management Applications, vol. 18, no. 5s, pp. 275–288, 2026, doi: 10.70917/ijcisim-2026-2706.
31. Rebecca Sandra Paul and C. Emilin Shyni, "Goal Aware Fine Grained Personalized Learning Framework Adaptive to Dynamic User Profile," International Journal of Computer Information Systems and Industrial Management Applications, vol. 18, no. 12s, pp. 687–699, 2026, doi: 10.70917/ijcisim-2026-3941.