YOLOv12: Attention-Centric Real-Time Object Detectors

Yunjie Tian
University at Buffalo
yunjieti@buffalo.edu

Qixiang Ye*
UCAS
qxye@ucas.ac.cn

David Doermann
University at Buffalo
doermann@buffalo.edu

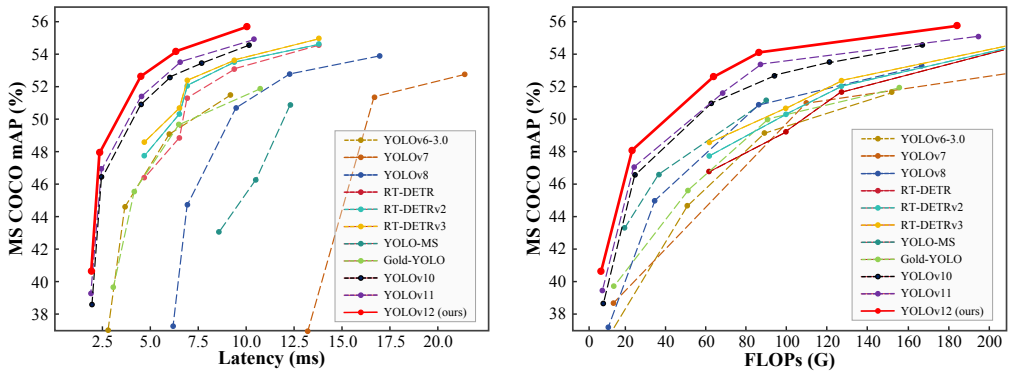

Figure 1: **Comparisons with other popular methods in terms of latency-accuracy (left) and FLOPs-accuracy (right) trade-offs.**

Abstract

Enhancing the network architecture of the YOLO framework has been crucial for a long time. Still, it has focused on CNN-based improvements despite the proven superiority of attention mechanisms in modeling capabilities. This is because attention-based models cannot match the speed of CNN-based models. This paper proposes an attention-centric YOLO framework, namely YOLOv12, that matches the speed of previous CNN-based ones while harnessing the performance benefits of attention mechanisms. YOLOv12 surpasses popular real-time object detectors in accuracy with competitive speed. For example, YOLOv12-N achieves 40.5% mAP with an inference latency of 1.62 ms on a T4 GPU, outperforming advanced YOLOv10-N / YOLO11-N by 2.0%/1.1% mAP with a comparable speed. This advantage extends to other model scales. YOLOv12 also surpasses end-to-end real-time detectors that improve DETR, such as RT-DETRv2 / RT-DETRv3: YOLOv12-X beats RT-DETRv2-R101 / RT-DETRv3-R101 while running faster with fewer computations and parameters. See more comparisons in Figure 1. Source code is available at `https://github.com/sunsmarterjie/yolov12`.

Real-time object detection has consistently attracted significant attention due to its low-latency characteristics, which provide substantial practicality [26, 31, 6, 19]. Among them, the YOLO series [51, 53, 52, 5, 32, 35, 64, 26, 67, 61, 31] has effectively established an optimal balance between latency and accuracy, thus dominating the field. Although improvements in YOLO have focused on areas such as loss functions [10, 80, 47, 46, 79, 38, 54], label assignment [25, 37, 68, 24, 81], network architecture design has remained a critical research priority [35, 64, 26, 67, 31]. Although

*Corresponding author.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

attention-centric models have been proven to possess more substantial modeling capabilities, even in small models [27, 21, 22, 57], most architectural designs continue to focus primarily on CNNs.

The primary reason for this situation lies in the inefficiency of the attention mechanism, which comes from two main factors: quadratic computational complexity and inefficient memory access operations of the attention mechanism (the latter being the main issue addressed by FlashAttention [16, 15]). As a result, under a similar computational budget, CNN-based architectures outperform attention-based ones by a factor of $\sim 3\times$ [41], which significantly limits the adoption of attention in YOLO systems.

This paper aims to tackle these challenges and establish an attention-centric YOLO framework, YOLOv12. We introduce three key improvements. **First**, we propose a simple yet efficient Area Attention module (A2), which preserves a large receptive field while efficiently reducing the computational complexity of attention, thus improving speed. Moreover, A2 supports flexible input sizes without constraints in window attention [42, 18] to accommodate window partitioning, allowing rectangular inference in YOLO. **Second**, we design Residual Efficient Layer Aggregation Networks (R-ELAN) to address optimization challenges in the designed attention-based models. Building on ELAN [64], R-ELAN introduces (i) a block-level residual design with scaling techniques and (ii) an improved feature aggregation strategy. **Third**, we refine attention-centric architectures to better integrate with the YOLO framework. Key modifications include: incorporating FlashAttention to mitigate memory access issue; using a decoupled projection strategy to construct q, k, and v during attention computation, thus avoiding redundant feature reorganization; removing positional encoding for a leaner design; reducing the MLP ratio from 4 to 1.5 to better balance attention and FFN computation; and decreasing the depth of stacked blocks to facilitate optimization.

Based on the designs outlined above, we develop a new family of real-time detectors with 5 model scales: YOLOv12-N, S, M, L, and X. We perform extensive experiments on standard object detection benchmarks following YOLO11 [31] without any additional tricks, demonstrating that YOLOv12 provides significant improvements over previous popular models in terms of latency-accuracy and FLOPs-accuracy trade-offs across these scales, as illustrated in Figure 1. For example, YOLOv12-N achieves 40.5% mAP, outperforming YOLOv10-N [61] by 2.0% mAP while maintaining a faster inference speed, and YOLO11-N [31] by 1.1% mAP with a comparable speed. This advantage remains consistent across other scale models. Compared to RT-DETRv2-R18 [78], YOLOv12-S is comparable and 47% faster in latency speed, requiring only 33% of its computations and 46% of its parameters. Compared to RT-DETRv3-R50 [43], YOLOv12-L is 0.4% mAP better and 15% faster, requiring only 61% of its computations and 63% of its parameters.

In summary, YOLOv12 contributes by: **(i)** introducing an attention-centric, efficient YOLO framework that challenges CNN dominance in the series, and **(ii)** achieving state-of-the-art results with fast inference and high accuracy without relying on pre-training or additional training techniques.

1 Related Work

Real-time Object Detectors. The YOLO series [51, 53, 52, 5, 32, 35, 64, 63, 11, 26, 67, 61, 31] has become the leading framework for real-time object detection. The early YOLO models [51, 53, 52] established the foundation primarily from a model design perspective. YOLOv4 [5] and YOLOv5 [32] introduced CSPNet [65], data augmentation, and multi-scale features, while YOLOv6 [35] further enhanced efficiency with BiC, SimCSPSPPF, *etc.* YOLOv7 [64] incorporated E-ELAN [66] for better gradient flow and various bag-of-freebies, while YOLOv8 [26] adopted the efficient C2f block for feature extraction. Recent versions, YOLOv9 [67], introduced GELAN for architectural optimization and PGI for training improvements, YOLOv10 [61] applied NMS-free training with dual assignments, and YOLOv11 [31] optimized speed and accuracy with C3K2 (a variant of GELAN) and lightweight depthwise separable convolutions. Other developments [73, 62, 75] further enhanced detection capabilities through improved backbones and head designs Beyond YOLO, RT-DETR [78] improved end-to-end detectors [9, 83, 45, 36] for real-time use through an efficient encoder and uncertainty-minimal query selection, with RT-DETRv2 [43] and RT-DETRv3 [69] further refinement with bag-of-freebies. Recent follow-ups [50, 29] continue to show compromising results.

Efficient Vision Transformers. Reducing the computational cost of global self-attention is the key to effectively applying vision transformers to downstream tasks. PVT [71] tackles this with multi-resolution stages and downsampling. Swin Transformer [42] restricts self-attention to local windows and shifts them to connect non-overlapping regions. Other approaches, like axial self-attention [28]

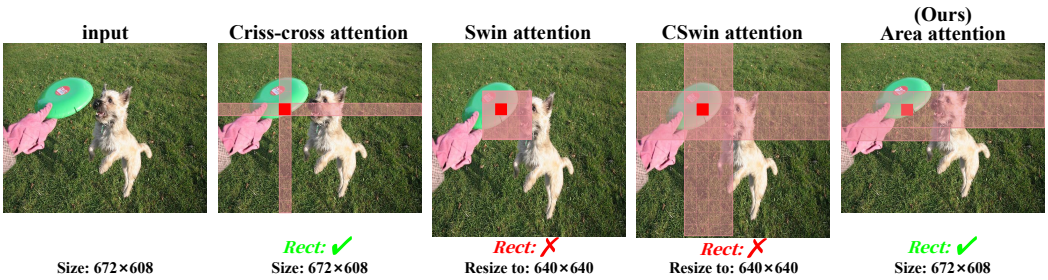

Figure 2: **Comparison of the representative local attention mechanisms with our area attention.** Area attention computes local attention over sequentially placed tokens without requiring input size constraints, enabling default distortion-free rectangular inference (*Rect*) in YOLO. Area attention enjoys efficient implementation, requiring only `flatten` and a `reshape` operations.

and criss-cross attention [30], compute attention within horizontal and vertical windows. CSWin Transformer [18] extends this with self-attention in cross-shaped windows. Furthermore, methods such as [14, 76] establish local-global relations, enhancing efficiency while reducing the reliance on global self-attention. Fast-iTPN [57] accelerates downstream task inference with token migration and gathering. Some methods [56, 70, 33] use linear attention to reduce complexity, but vision mamba models [82, 41] still struggle to achieve real-time speeds [41]. FlashAttention [16, 15] addresses high-bandwidth memory bottlenecks by optimizing I/O and reducing memory access for efficiency.

2 Approach

2.1 Efficiency Analysis

The attention mechanism, while highly effective in capturing global dependencies and facilitating tasks such as natural language processing [7, 17] and computer vision [23, 42], is inherently slower than CNNs. Two primary factors contribute to this speed discrepancy.

Complexity. First, the computational complexity of the attention operation scales quadratically with the input sequence length L. Specifically, for an input sequence with length L and feature dimension d, the computation of the attention matrix requires $O(L^2d)$ operations since each token attends to every other token. In contrast, the complexity of convolution operations in CNNs scales linearly with respect to the spatial or temporal dimension, *i.e.*, $O(kLd)$, where k is the kernel size and is typically much smaller than L. As a result, self-attention becomes computationally prohibitive, especially for significant inputs such as high-resolution images or long sequences.

Moreover, attention-based vision transformers often suffer from speed overhead due to their complex designs (*e.g.*, window partitioning / reverse in some window attentions [42, 18]) and additional modules (*e.g.*, positional encoding), leading to slower performance compared to CNNs [41]. In contrast, the design modules in this paper use simple and efficient operations to implement attention, maximizing speed and efficiency.

Computation. Second, the memory access patterns in attention computations are less efficient than in CNNs [16, 15]. In self-attention, intermediate maps such as the attention map (QK^T) and the softmax map ($L \times L$) must be transferred from high-speed GPU SRAM to high-bandwidth GPU memory (HBM) for further computation. The read/write speed of SRAM is more than 10 times faster, leading to significant memory access overhead and increased wall clock time.

2.2 Area Attention

A straightforward way to reduce the computational cost of vanilla attention is to use the linear attention mechanism [56, 70], which reduces the complexity from quadratic to linear. For a visual feature f with dimensions (n, h, d), where n is the number of tokens, h is the number of heads, and d is the head size, linear attention reduces the complexity from $2n^2hd$ to $2nhd^2$, cutting the computational cost as $n > d$. However, linear attention faces issues such as global dependency degradation [34],

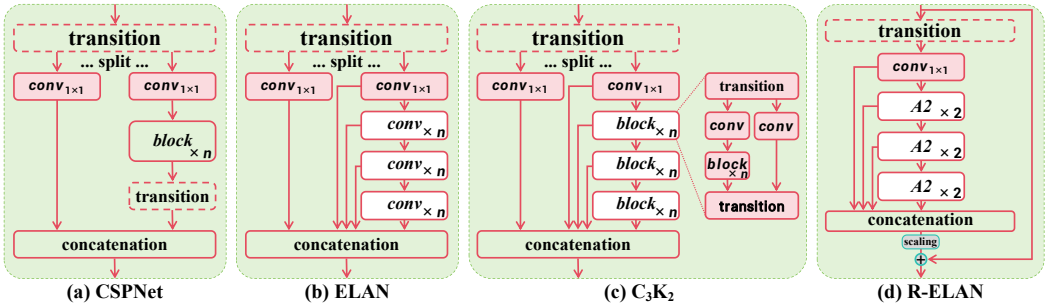

Figure 3: **The architecture comparison with popular modules** including (a): CSPNet [65], (b) ELAN [66], (c) C3K2 (a case of GELAN) [67, 31], and (d) the proposed R-ELAN (residual efficient layer aggregation networks).

instability [12], and distribution sensitivity [72]. Furthermore, the low-rank bottleneck [13, 3] provides limited speed improvements when applied to YOLO.

An alternative approach to effectively reduce complexity is the local attention mechanism (*e.g.*, criss-cross attention [30], axial attention [28], and window attention [42, 18]), as shown in Figure 2, which transforms global attention into local, thus reducing computational costs. However, partitioning the feature map into windows can introduce overhead or reduce the receptive field, impacting both speed and accuracy. Furthermore, for window attentions, such as Swin [42] and CSwin [18], they also suffer from additional constraints on input size to accommodate window partitioning. As shown in Figure 2, they cannot handle specific rectangular dimensions (native resolutions) commonly seen in the inference of YOLO (*e.g.*, 672×608) and must resize the input to window-friendly sizes (*e.g.*, 640×640), which somehow introduces image distortion and thereby degrades detection performance.

In this study, we propose a simple yet efficient area attention module (A2). The comparison of A2 with other local attentions is illustrated in Figure 2. Specifically, for a feature of size (H, W), we first perform a flatten operation to obtain a one-dimensional feature with $H \times W$ tokens, which is then evenly divided into l segments (areas) (so each area contains $\frac{H \times W}{l}$ sequentially placed tokens). Local attention is applied independently within each area (a simple reshape operation enables the transformation to the original size). This only requires a simple flatten and a reshape operation, leading to efficient implementation and faster speed. We empirically set the default value of l to 4 (or 8), thus reducing the receptive field to $\frac{1}{l}$ of the original size while maintaining a large receptive field. More importantly, A2 eliminates feature size constraints while only requiring the total token count $(H \times W)$ to be divisible by l. This design inherently supports rectangular inference (see Figure 2), the standard evaluation protocol in YOLO, thus achieving seamless compatibility with the YOLO framework. (See more details about A2 and architecture design in the Appendix.)

Regarding the memory access issue, we integrate FlashAttention [16, 15] into area attention to solve it. FlashAttention is already an infrastructure for many large language models [60, 1, 40] and diffusion models [55, 49, 4], and can provide similar benefits to YOLO.

2.3 Residual Efficient Layer Aggregation Networks

Efficient layer aggregation networks (ELAN) [64] are designed to improve feature aggregation. As shown in Figure 3 (b), ELAN splits the output of a transition layer (a convolution 1×1, processes each split through multiple modules, and then concatenates the outputs before applying another transition layer to align the dimensions. However, as noted in [64], this design can lead to instability. We argue that this architecture causes gradient blocking and lacks residual connections between input and output. In addition, incorporating the attention mechanism into the network introduces additional optimization challenges. Empirically, L- and X-scale models either fail to converge or remain unstable, even when using Adam or AdamW optimizers.

To address this issue, we propose residual efficient layer aggregation networks (R-ELAN), as shown in Figure 3 (d). We introduce a residual shortcut from input to output throughout the block, with a scaling factor (defaulting to 0.01). This design is inspired by the layer scaling [59] used in deep vision transformers. However, applying layer scaling for each attention area does not solve the optimization

Table 1: **Comparison with popular real-time object detectors.** All results are obtained using 640×640 inputs. †: pre-trained models are required.

Method	FLOPs (G)	#Param. (M)	$AP^{val}_{50:95}$ (%)	AP^{val}_{50} (%)	AP^{val}_{75} (%)	Latency (ms)
YOLOv8-N [26]	8.7	3.2	37.4	52.6	40.5	1.77
YOLOv10-N [61]	6.7	2.3	38.5	53.8	41.7	1.84
YOLO11-N [31]	6.5	2.6	39.4	55.3	42.8	1.5
YOLOv12-N (Ours)	**6.0**	**2.6**	**40.5**	**56.4**	**43.8**	**1.62**
YOLOv8-S [26]	28.6	11.2	45.0	61.8	48.7	2.33
RT-DETRv2-R18† [44]	60.0	20.0	47.9	64.9	–	4.58
YOLOv9-S [67]	26.4	7.1	46.8	63.4	50.7	–
YOLOv10-S [61]	21.6	7.2	46.3	63.0	50.4	2.49
YOLO11-S [31]	21.5	9.4	46.9	63.9	50.6	2.5
YOLOv12-S (Ours)	**19.5**	**9.1**	**47.8**	**64.9**	**51.3**	**2.44**
YOLOv8-M [26]	78.9	25.9	50.3	67.2	54.7	5.09
RT-DETRv2-R34† [44]	100.0	36.0	49.9	67.5	–	6.32
RT-DETRv3-R18† [44]	60.0	20.0	48.7	–	–	4.58
RT-DETRv3-R34† [69]	100.0	36.0	50.1	67.5	–	6.32
YOLOv9-M [67]	76.3	20.0	51.4	68.1	56.1	–
YOLOv10-M [61]	59.1	15.4	51.1	68.1	55.8	4.74
YOLO11-M [31]	68.0	20.1	51.5	68.5	55.7	4.7
YOLOv12-M (Ours)	**59.9**	**19.7**	**52.5**	**70.0**	**57.0**	**4.30**
YOLOv8-L [26]	165.2	43.7	53.0	69.8	57.7	8.06
RT-DETRv2-R50† [44]	136.0	42.0	53.4	71.6	–	6.90
RT-DETRv3-R50† [69]	136.0	42.0	53.4	–	–	6.90
YOLOv9-C [67]	102.1	25.3	53.0	70.2	57.8	–
YOLOv10-B [61]	92.0	19.1	52.5	69.6	57.2	5.74
YOLOv10-L [61]	120.3	24.4	53.2	70.1	58.1	7.28
YOLO11-L [31]	86.9	25.3	53.3	70.1	58.2	6.2
D-FINE-L† [50]	91	31	54.0	71.6	58.4	8.07
YOLOv12-L (Ours)	**82.6**	**26.6**	**53.8**	**71.1**	**58.7**	**5.89**
YOLOv8-X [26]	257.8	68.2	54.0	71.0	58.8	12.83
RT-DETRv2-R101† [44]	259.0	76.0	54.3	72.8	–	13.5
RT-DETRv3-R101† [69]	259.0	76.0	54.6	–	–	13.5
YOLOv10-X [61]	160.4	29.5	54.4	71.3	59.3	10.70
YOLO11-X [31]	194.9	56.9	54.6	71.6	59.5	11.3
D-FINE-X† [50]	202	62	55.8	73.7	60.2	12.89
YOLOv12-X (Ours)	**184.9**	**59.5**	**55.4**	**72.6**	**60.4**	**10.47**

challenge and introduces latency. This highlights that the convergence issue is not solely due to the attention mechanism but also the ELAN structure, validating the rationale behind R-ELAN design.

We also design a new feature aggregation approach, shown in Figure 3 (d). In the original ELAN, the input is passed through a transition layer, splitting it into two parts. Subsequent blocks process one part, and both parts are concatenated to produce the output. In contrast, our design uses a transition layer to adjust channel dimensions and produce a single feature map. This map is processed through subsequent blocks, followed by concatenation, forming a bottleneck structure. This method retains the original feature integration capability while reducing computational cost, parameters, and memory usage.

2.4 Architectural Improvements

Many attention-based vision transformers use plain-style architectures [20, 58, 2, 27, 23, 22], while we retain the hierarchical structure of previous YOLO versions [51, 53, 52, 5, 32, 35, 64, 26, 67, 61, 31]

and we will demonstrate its necessity. We simplify the architecture depth by removing the stacking of three blocks in the final stage of the backbone that are used most frequently in recent YOLO versions [26, 67, 61, 31], retaining only a single block. In addition, we retain the first two blocks, remove the third block, and replace all C3K2 blocks with R-ELAN blocks in the backbone. We used convolutions with the 2 and 4 groups in the second and third blocks, respectively.

Several default configurations of the vanilla attention mechanism have also been modified for better alignment with the YOLO system. These include adjusting the MLP ratio from 4 to 1.5 (or 2 for N / S / M scale models), removing positional encoding, and adding a separable convolution (7×7) (position perceiver) to enhance the ability of area attention to perceive positional information. The effectiveness of these changes will be demonstrated in Section 3.5.

Previous versions (*e.g.*, YOLOv10 [61] and YOLOv11 [31]) adopt a coupled projection strategy for constructing q, k, and v during attention calculation, where a single convolutional layer jointly projects the input and then splits the output into the three components. However, when used alongside the position perceiver, this coupling results in redundant reorganization of the v features, degrading inference efficiency. To mitigate this, we introduce a decoupled projection strategy that computes v separately from q and k. This design eliminates unnecessary processing on v, leading to an inference speedup of approximately 10%.

Table 2: **Comparison of YOLOv12 with previous versions in speed (GPU, CPU) and memory usage.** CUDA results are measured on T4 / RTX 3080 GPUs, with inference latency (ms) reported for FP32 and FP16. Memory usage (Mem.) is measured with the TensorRT model (bs = 1). All results are obtained using the same hardware.

Model	FLOPs (G)	Mem. (G)	CUDA FP32	CUDA FP16	CPU	mAP
YOLOv9-T [67]	8.2	0.16	4.0/2.3	2.4/1.5	40.1	–
YOLOv10-N [61]	6.7	0.19	3.0/1.6	1.7/1.0	32.4	38.5
YOLO11-N [31]	6.5	0.20	3.0/1.6	1.5/0.9	32.5	39.5
YOLOv12-N	6.0	0.15	3.0/1.6	1.6/1.0	38.7	40.5
YOLOv9-S [67]	26.4	0.19	7.3/3.7	3.3/1.9	85.6	46.8
YOLOv10-S [61]	21.6	0.23	6.3/2.6	2.6/1.3	70.1	46.3
YOLO11-S [31]	21.5	0.23	6.4/2.7	2.4/1.3	72.1	46.9
YOLOv12-S	19.5	0.18	6.5/2.7	2.5/1.3	83.3	47.8
YOLOv9-M [67]	76.3	0.24	16.7/6.3	5.6/2.7	186.5	51.4
YOLOv10-M [61]	59.1	0.29	13.5/5.3	4.8/2.4	161.5	51.1
YOLO11-M [31]	68.0	0.30	16.2/5.4	4.4/2.2	189.9	51.5
YOLOv12-M	59.9	0.24	14.9/5.2	4.3/2.1	213.2	52.5
YOLOv9-C [67]	102.1	0.29	20.9/7.4	6.2/2.8	272.6	53.0
YOLOv10-L [61]	120.3	0.30	23.8/8.0	7.3/3.4	294.8	53.2
YOLO11-L [31]	86.9	0.32	20.4/6.9	5.9/2.9	237.7	53.3
YOLOv12-L	82.6	0.27	20.6/6.9	6.0/2.9	299.5	53.8
YOLOv9-E [67]	189.0	0.69	48.6/15.5	15.5/6.5	499.7	55.6
YOLOv10-X [61]	160.4	0.40	35.0/10.7	10.4/4.5	410.1	54.4
YOLO11-X [31]	194.9	0.45	40.4/13.5	10.5/4.9	484.0	54.6
YOLOv12-X	184.9	0.37	40.6/13.6	10.5/4.9	524.6	55.4

3 Experiment

3.1 Experimental Setup

We validate YOLOv12 on the MS-COCO 2017 dataset [39]. The YOLOv12 family consists of 5 variants: YOLOv12-N, YOLOv12-S, YOLOv12-M, YOLOv12-L, and YOLOv12-X. All models are trained for 600 epochs using the SGD optimizer with an initial learning rate of 0.01, consistent with YOLO11 [31]. A linear learning rate decay schedule is adopted, with a linear warm-up for the first three epochs. Following the methodology in [61, 78], the latencies of all models are tested on a T4 GPU using TensorRT FP16. See the Appendix for more details and results.

3.2 Comparison with State-of-the-arts

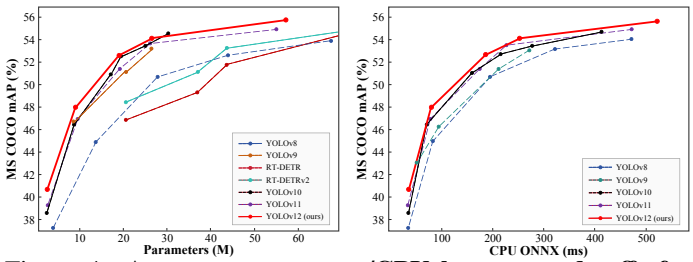

Figure 4: **Accuracy-parameter/CPU latency trade-offs for YOLOv12.**

We compare YOLOv12 with other detectors in Table 1, using performance data from their official reports (some achieve higher speed in our reproduction, see Table 2). **N-scale**: YOLOv12-N exceeds YOLOv8-N [67], YOLOv10-N [61], and YOLO11 [31] by up to 3.1% mAP, maintaining a similar or lower computational cost with 1.62 ms/image latency. **S-scale**: YOLOv12-S (19.5G FLOPs, 9.1M params) achieves 47.8 mAP

at 2.44 ms/image, outperforming YOLOv9-S [67], YOLOv10-S [61], and YOLO11-S [31]. Compared to RT-DETRv2-R18 [44], it offers better speed with lower computational cost. **M-scale**: YOLOv12-M (59.9G FLOPs, 19.7M params) achieves 52.5 mAP at 4.30 ms/image, outperforming YOLOv9-M [67], YOLOv10 [61], YOLO11 [31], and RT-DETRv2-R34 [44] / RT-DETRv3-R34 [69]. **L-scale**: YOLOv12-L exceeds YOLOv10-L [61] with 37.7G fewer FLOPs and YOLO11-L [31] by 0.5% mAP while maintaining a similar efficiency. It also outperforms RT-DETRv2-R50 [44] / RT-DETRv3-R50 [69] with fewer FLOPs, parameters, and faster speed. **X-scale**: YOLOv12-X exceeds YOLOv10-X [61] and YOLO11-X [31] by up to 1.0% mAP with similar efficiency and surpasses RT-DETRv2-R101 [43] / RT-DETRv3-R101 [69] with 28.6% fewer FLOPs, 21.7% fewer parameters, and 22.4% faster speed.

3.3 Speed, Memory, and Efficiency

Table 2 compares the inference speed of YOLOv9 [67], YOLOv10 [61], YOLO11 [31], and YOLOv12 on T4 and RTX 3080 GPUs using FP32 and FP16 precision, as well as CPU (Intel Core i7-10700K @ 3.80GHz) speed and peak memory usage. All results are obtained on the same hardware to ensure fairness. The results show that YOLOv12 significantly outperforms YOLOv9 in inference speed, while matching the performance of YOLOv10 and YOLOv11. YOLOv12 requires less peak memory than the other models.

Figure 4 compares the speed and efficiency of YOLOv12. We present the accuracy-parameter trade-off, demonstrating that it outperforms popular methods, including YOLOv10, which has significantly fewer parameters. This underscores YOLOv12's ability to achieve higher accuracy with fewer parameters. We also include the accuracy-latency trade-off for YOLOv12 on a CPU (Intel Core i7-10700K @ 3.80GHz). As shown, YOLOv12 delivers superior performance and establishes superior boundaries, demonstrating greater efficiency across various metrics.

Table 3: **Ablation on the proposed area attention.** Compared to other methods (cross-cross (CC), Swin, and CSwin attentions), area attention helps YOLOv12-N/S/X models obtain better accuracy and run faster on GPU (CUDA) and CPU. CUDA results are measured on T4/RTX 3080. All results use the same hardware and exclude FlashAttention [16, 15].

Model	Atten. Type	Mem. (G)	CUDA (ms) FP32	CUDA (ms) FP16	CPU (ms)	$AP^{val}_{50:95}$
N	Global	0.21	6.1/2.7	3.1/1.6	66.3	41.7
	CC	0.17	4.7/2.3	2.4/1.4	72.9	39.5
	Swin	0.16	4.5/2.2	2.2/1.3	39.6	40.2
	CSwin	0.16	4.5/2.2	2.2/1.3	39.9	40.3
	A2	0.15	4.2/2.0	2.0/1.2	38.7	40.5
S	Global	0.30	12.5/4.8	5.2/2.4	144.3	49.0
	CC	0.20	9.5/3.9	3.8/2.1	90.6	46.6
	Swin	0.18	9.2/3.8	3.3/1.9	83.4	47.4
	CSwin	0.18	9.3/3.8	3.3/1.9	85.6	47.5
	A2	0.18	8.7/3.5	3.1/1.8	83.3	47.8
X	Global	0.68	74.2/31.4	26.3/10.8	913.0	56.2
	CC	0.40	56.2/23.6	16.6/8.0	572.3	54.7
	Swin	0.38	53.4/21.2	15.3/7.6	538.2	55.2
	CSwin	0.38	53.5/21.4	15.4/7.6	553.6	55.2
	A2	0.37	52.5/21.0	14.7/7.1	524.6	55.4

3.4 Ablation Studies

We perform ablation experiments to assess the effectiveness of area attention (A2) and E-ELAN in Table 3 and Table 4, respectively.

- **Area Attention.** In Table 3, evaluations are performed on YOLOv12-N/S/X models, measuring the GPU (CUDA) and CPU inference speed. CUDA results are obtained using identical T4 and RTX 3080 hardware to ensure fairness, while CPU performance is measured on an Intel Core i7-10700K @ 3.80GHz. Memory usage (Mem.) is measured with the TensorRT model (bs = 1). We compare the performance with other local attentions and adjust the number of chan-

Table 4: **Ablation on the proposed residual efficient layer aggregation networks (R-ELAN).** RA: Proposed feature integration; RS: Residual block; SL: Scaling factor for residuals; CS: Convergence status.

Model scale	RA	RS	SL	CS	FLOPs (G)	Param (M)	Mem. (G)	mAP	Lat.
N	✗	✗	–	✓	6.5	2.8	0.20	40.6	1.70
	✓	✗	–	✓	6.0	2.6	0.15	40.4	1.62
	✓	✓	0.01	✓	6.0	2.6	0.15	40.5	1.62
L	✗	✗	–	✗	–	–	–	–	–
	✓	✗	–	✗	–	–	–	–	–
	✗	✓	0.01	✓	87.7	27.2	0.31	53.9	6.15
	✓	✓	0.1	✓	82.6	26.6	0.27	53.7	5.89
	✓	✓	0.01	✓	82.6	26.6	0.27	53.8	5.89
X	✗	✗	–	✗	–	–	–	–	–
	✓	✗	–	✗	–	–	–	–	–
	✗	✓	0.01	✓	197.5	60.8	0.42	55.4	10.47
	✓	✓	0.1	✓	184.9	59.5	0.37	55.2	10.47
	✓	✓	0.01	✓	184.9	59.5	0.37	55.4	10.47

Table 5: **Diagnostic studies.** To save space, we only show the factor(s) to be diagnosed in each subtable. The default parameters are (unless otherwise specified) training for 600 epochs from scratch, using the YOLOv12-N model. All latency (Lat.) results are tested on a T4 GPU

.

Model	Lat.$^{\text{-DP}}$	Lat.
N	1.70	1.62
S	2.70	2.44
M	4.88	4.30
L	6.56	5.89
X	11.39	10.47

(a) Decoupled Projection

Method	mAP	Lat.
N/A	38.2	1.60
~~S_T~~	40.1	1.60
~~S_4~~	39.8	1.68
Ours	40.5	1.62

(b) Hierarchical Design

Ep.	mAP (N)	mAP (S)
300	39.2	47.0
500	40.3	47.6
600	40.5	47.8
800	41.1	48.1

(c) Training Epoch

kernel	mAP	Lat.
3	40.1	1.57
5	40.4	1.60
7	40.5	1.62
9	40.6	1.73

(d) Position Perceiver

Pos.	mAP	Lat.
RPE	40.2	1.77
APE	40.3	1.67
N/A	40.5	1.62

(e) Position Embedding

Model	mAP	Lat.
YOLOv10	38.5	1.68
YOLO11	39.4	1.49
YOLOv12	40.5	1.62

(f) FA for V10/11

Ratio (L)	mAP	Lat.
1.5	53.8	5.89
2.0	53.3	5.79
4.0	53.1	5.73

(g) MLP Ratio

FA	Lat. (N)	Lat. (S)
✗	2.00	3.12
✓	1.62	2.45

(h) FlashAttention

nels to ensure that all models have similar parameters and FLOPs. The results demonstrate significant efficiency and speedup with area attention. For example, with FP32 on RTX 3080, YOLOv12-N achieves a 1.2ms inference (RTX 3080) with a 40.5 mAP, surpassing the criss-cross (CC) [30], Swin [42], and CSwin [18] attentions, with only 0.15G inference memory usage. In particular, the Swin and CSwin attentions require a resize operation (to 640×640) during inference, which can cause a performance drop[2]. Performance gain is consistently observed across different models and hardware configurations. We do not use FlashAttention [16, 15] in this experiment because it would significantly reduce the speed difference.

• **R-ELAN.** In Table 4, evaluations are performed on YOLOv12-N/L/X models, revealing two key findings: **(i)** For smaller models such as YOLOv12-N, residual connections provide performance gain with negligible extra cost. For larger models (YOLOv12-L/X), residual connections are crucial for stable training, with a minimal scaling factor (0.01) for convergence. **(ii)** The proposed feature integration method reduces the complexity of the model (FLOPs and parameters) and the memory cost while maintaining accuracy, with only a slight (even without) performance drop.

3.5 Diagnosis & Visualization

We diagnose YOLOv12 designs in Tables 5a to 5h, using YOLOv12-N trained from scratch for 600 epochs, unless otherwise stated.

• **Decoupled Projection: Table 5a.** We compare the Decoupled Projection (DP) strategy to construct q, k, and v with previous methods (Lat.$^{\text{-DP}}$). DP consistently improves speed by around 10% in all scale models, as it avoids costly reorganization of the v feature during the A2 implementation, leading to better efficiency.

• **Hierarchical Design: Table 5b.** Unlike other detection systems, such as Mask R-CNN [27, 2], where plain vision transformers produce substantial results, YOLOv12 behaves differently. Using a plain vision transformer (N/A) causes a significant performance drop, achieving only 38.2% mAP. A moderate adjustment, such as omitting the first (~~S_T~~) or fourth stage (~~S_4~~) while maintaining similar FLOPs, results in a slight performance degradation of 0.4% mAP and 0.7% mAP, respectively. Consistent with previous YOLO models, the hierarchical design remains the most effective, providing the best performance in YOLOv12.

• **Training Epochs: Table 5c.** We examine how varying the number of training epochs impacts performance (training from scratch). Although some existing YOLO detectors achieve optimal results after roughly 500 training epochs [26, 67, 61], YOLOv12 requires a more extended training

[2]If we resize the image to 640×640 for A2 inference, the speed remains unaffected, but the performance drops by about 0.2 in general, highlighting the importance of rectangular inference.

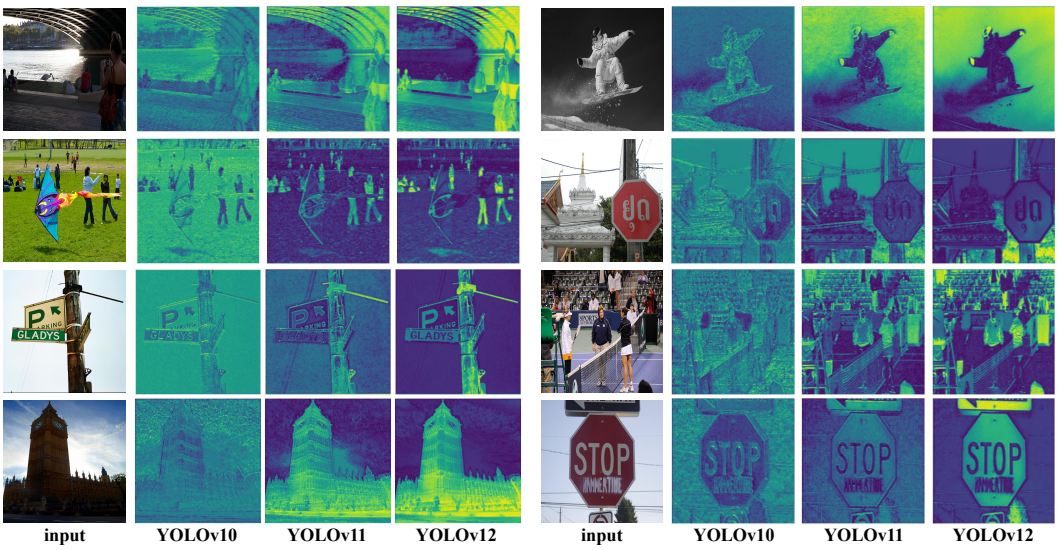

Figure 5: **Comparison of heat maps between YOLOv10 [61], YOLO11 [31], and the proposed YOLOv12.** YOLOv12 exhibits clearer object perception than YOLOv10 and YOLO11. All results use X-scale models. *Zoom in to compare details.*

period (about 600/800 epochs) to achieve peak performance, keeping the same configuration used in YOLO11 [31].

• **Position Perceiver: Tables 5d.** In the attention mechanism, we apply a separable convolution with a large kernel to the attention value v, adding its output to v@attn. This component, called the Position Perceiver, preserves the original positions of image pixels, helping the attention mechanism perceive positional information. While similar to the PSA module [61], we expand the convolution kernel, improving performance without affecting speed. Increasing the kernel size enhances performance but slows processing, with a significant slowdown at 9×9. Therefore, we set the default kernel size to 7×7.

• **Position Embedding: Tables 5e.** We examine the impact of common positional embeddings (RPE and APE) on performance. Interestingly, the best results are achieved without any positional embedding, leading to a cleaner architecture and faster inference.

• **FA for V10/11: Tables 5f.** This table uses the FlashAttention (FA) for YOLOv10 and YOLO11, including a few attention blocks. It can be seen that with FA, they also benefit from speed improvements. FA should serve as future infrastructure for the YOLO framework, much like in large language models.

• **MLP Ratio: Tables 5g.** In vanilla attention, the MLP ratio within the attention module is typically set to 4.0. However, YOLOv12 behaves differently. The table shows that varying the MLP ratio affects model size, so we adjust the feature dimensions for consistency. In particular, YOLOv12 performs better with an MLP ratio of 1.5, shifting the computational load toward the attention mechanism and emphasizing the importance of area attention.

• **FlashAttention: Tables 5h.** This table demonstrates the role of FlashAttention in YOLOv12, showing a 0.38ms speedup for YOLOv12-N and 0.67 ms for YOLOv12-S without additional costs.

Visualization: Heat Map Comparison. Figure 5 compares the heat maps of YOLOv12 with YOLOv10 [61] and YOLO11 [31]. These heat maps, extracted from the third stage of the backbones of X-scale models, highlight the activated regions, reflecting the model's object perception capability. As illustrated, YOLOv12 shows more defined object contours and better foreground activation than YOLOv10 and YOLO11, indicating improved perception. We explain that this improvement comes from the area attention mechanism, which captures a broader context and enables more precise foreground activation with its larger receptive field than CNN with its larger receptive field than CNN. We believe that this characteristic gives YOLOv12 a performance advantage.

4 Conclusion

This study introduces YOLOv12, which integrates an attention-centric design into the YOLO framework, achieving a state-of-the-art latency-accuracy trade-off. We propose a novel network that uses area attention to reduce computational complexity and R-ELAN to enhance feature aggregation for efficient inference. Key refinements to the vanilla attention mechanism further align it with YOLO's real-time constraints while maintaining high-speed performance. By combining area attention, R-ELAN, and architectural optimizations, YOLOv12 significantly improves accuracy and efficiency. Comprehensive ablation studies validate these innovations. This work challenges CNN-dominated YOLO designs and advances attention-based real-time object detection.

Limitations. YOLOv12 requires FlashAttention [16, 15], which currently supports Turing, Ampere, Ada Lovelace, or Hopper GPUs (*e.g.*, T4, Quadro RTX series, RTX20 series, RTX30 series, RTX40 series, RTX A5000/6000, A30/40, A100, H100, *etc.*).

5 Acknowledgment

This work was supported by National Natural Science Foundation of China (NSFC) under Grant 62225208 and 62450046 and CAS Project for CAS Project for Young Scientists in Basic Research under Grant No.YSBR-117.

References

[1] Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

[2] Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. Beit: Bert pre-training of image transformers. *arXiv preprint arXiv:2106.08254*, 2021.

[3] Srinadh Bhojanapalli, Chulhee Yun, Ankit Singh Rawat, Sashank Reddi, and Sanjiv Kumar. Low-rank bottleneck in multi-head attention models. In *International conference on machine learning*, pages 864–873. PMLR, 2020.

[4] Andreas Blattmann, Tim Dockhorn, Sumith Kulal, Daniel Mendelevitch, Maciej Kilian, Dominik Lorenz, Yam Levi, Zion English, Vikram Voleti, Adam Letts, et al. Stable video diffusion: Scaling latent video diffusion models to large datasets. *arXiv preprint arXiv:2311.15127*, 2023.

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

[6] Daniel Bogdoll, Maximilian Nitsche, and J Marius Zöllner. Anomaly detection in autonomous driving: A survey. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4488–4499, 2022.

[7] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[8] Alexander Buslaev, Vladimir I Iglovikov, Eugene Khvedchenya, Alex Parinov, Mikhail Druzhinin, and Alexandr A Kalinin. Albumentations: fast and flexible image augmentations. *Information*, 11(2):125, 2020.

[9] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer, 2020.

[10] Kean Chen, Weiyao Lin, Jianguo Li, John See, Ji Wang, and Junni Zou. Ap-loss for accurate one-stage object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(11):3782–3798, 2020.

[11] Yuming Chen, Xinbin Yuan, Ruiqi Wu, Jiabao Wang, Qibin Hou, and Ming-Ming Cheng. Yolo-ms: rethinking multi-scale representation learning for real-time object detection. *arXiv preprint arXiv:2308.05480*, 2023.

[12] Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*, 2020.

[13] Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*, 2020.

[14] Xiangxiang Chu, Zhi Tian, Yuqing Wang, Bo Zhang, Haibing Ren, Xiaolin Wei, Huaxia Xia, and Chunhua Shen. Twins: Revisiting the design of spatial attention in vision transformers. *Advances in Neural Information Processing Systems*, 34:9355–9366, 2021.

[15] Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*, 2023.

[16] Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359, 2022.

[17] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4171–4186, 2019.

[18] Xiaoyi Dong, Jianmin Bao, Dongdong Chen, Weiming Zhang, Nenghai Yu, Lu Yuan, Dong Chen, and Baining Guo. Cswin transformer: A general vision transformer backbone with cross-shaped windows. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12124–12134, 2022.

[19] Douglas Henke Dos Reis, Daniel Welfer, Marco Antonio De Souza Leite Cuadros, and Daniel Fernando Tello Gamarra. Mobile robot navigation using an object recognition software with rgbd images and the yolo algorithm. *Applied Artificial Intelligence*, 33(14):1290–1305, 2019.

[20] Alexey Dosovitskiy. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[21] Yuxin Fang, Quan Sun, Xinggang Wang, Tiejun Huang, Xinlong Wang, and Yue Cao. Eva-02: A visual representation for neon genesis. *Image and Vision Computing*, 149:105171, 2024.

[22] Yuxin Fang, Quan Sun, Xinggang Wang, Tiejun Huang, Xinlong Wang, and Yue Cao. Eva-02: A visual representation for neon genesis. *Image and Vision Computing*, 149:105171, 2024.

[23] Yuxin Fang, Wen Wang, Binhui Xie, Quan Sun, Ledell Wu, Xinggang Wang, Tiejun Huang, Xinlong Wang, and Yue Cao. Eva: Exploring the limits of masked visual representation learning at scale. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19358–19369, 2023.

[24] Chengjian Feng, Yujie Zhong, Yu Gao, Matthew R Scott, and Weilin Huang. Tood: Task-aligned one-stage object detection. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3490–3499. IEEE Computer Society, 2021.

[25] Zheng Ge, Songtao Liu, Zeming Li, Osamu Yoshie, and Jian Sun. Ota: Optimal transport assignment for object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 303–312, 2021.

[26] Jocher Glenn. Yolov8. *https://github.com/ultralytics/ultralytics/tree/main*, 2023.

[27] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16000–16009, 2022.

[28] Jonathan Ho, Nal Kalchbrenner, Dirk Weissenborn, and Tim Salimans. Axial attention in multidimensional transformers. *arXiv preprint arXiv:1912.12180*, 2019.

[29] Shihua Huang, Zhichao Lu, Xiaodong Cun, Yongjun Yu, Xiao Zhou, and Xi Shen. Deim: Detr with improved matching for fast convergence. *arXiv preprint arXiv:2412.04234*, 2024.

[30] Zilong Huang, Xinggang Wang, Lichao Huang, Chang Huang, Yunchao Wei, and Wenyu Liu. Ccnet: Criss-cross attention for semantic segmentation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 603–612, 2019.

[31] Glenn Jocher. yolov11. *https://github.com/ultralytics*, 2024.

[32] Glenn Jocher, K Nishimura, T Mineeva, and RJAM Vilariño. yolov5. *https://github.com/ultralytics/yolov5/tree*, 2, 2020.

[33] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International conference on machine learning*, pages 5156–5165. PMLR, 2020.

[34] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International conference on machine learning*, pages 5156–5165. PMLR, 2020.

[35] Chuyi Li, Lulu Li, Yifei Geng, Hongliang Jiang, Meng Cheng, Bo Zhang, Zaidan Ke, Xiaoming Xu, and Xiangxiang Chu. Yolov6 v3. 0: A full-scale reloading. *arXiv preprint arXiv:2301.05586*, 2023.

[36] Feng Li, Hao Zhang, Shilong Liu, Jian Guo, Lionel M Ni, and Lei Zhang. Dn-detr: Accelerate detr training by introducing query denoising. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13619–13627, 2022.

[37] Shuai Li, Chenhang He, Ruihuang Li, and Lei Zhang. A dual weighting label assignment scheme for object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9387–9396, 2022.

[38] Xiang Li, Wenhai Wang, Lijun Wu, Shuo Chen, Xiaolin Hu, Jun Li, Jinhui Tang, and Jian Yang. Generalized focal loss: Learning qualified and distributed bounding boxes for dense object detection. *Advances in Neural Information Processing Systems*, 33:21002–21012, 2020.

[39] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer, 2014.

[40] Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

[41] Yue Liu, Yunjie Tian, Yuzhong Zhao, Hongtian Yu, Lingxi Xie, Yaowei Wang, Qixiang Ye, Jianbin Jiao, and Yunfan Liu. Vmamba: Visual state space model. *Advances in neural information processing systems*, 37:103031–103063, 2025.

[42] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022, 2021.

[43] Wenyu Lv, Yian Zhao, Qinyao Chang, Kui Huang, Guanzhong Wang, and Yi Liu. Rt-detrv2: Improved baseline with bag-of-freebies for real-time detection transformer. *arXiv preprint arXiv:2407.17140*, 2024.

[44] Wenyu Lv, Yian Zhao, Qinyao Chang, Kui Huang, Guanzhong Wang, and Yi Liu. Rt-detrv2: Improved baseline with bag-of-freebies for real-time detection transformer. *arXiv preprint arXiv:2407.17140*, 2024.

[45] Depu Meng, Xiaokang Chen, Zejia Fan, Gang Zeng, Houqiang Li, Yuhui Yuan, Lei Sun, and Jingdong Wang. Conditional detr for fast training convergence. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3651–3660, 2021.

[46] Kemal Oksuz, Baris Can Cam, Emre Akbas, and Sinan Kalkan. A ranking-based, balanced loss function unifying classification and localisation in object detection. *Advances in Neural Information Processing Systems*, 33:15534–15545, 2020.

[47] Kemal Oksuz, Baris Can Cam, Emre Akbas, and Sinan Kalkan. Rank & sort loss for object detection and instance segmentation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3009–3018, 2021.

[48] Haodong Ouyang. Deyo: Detr with yolo for end-to-end object detection. *arXiv preprint arXiv:2402.16370*, 2024.

[49] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4195–4205, 2023.

[50] Yansong Peng, Hebei Li, Peixi Wu, Yueyi Zhang, Xiaoyan Sun, and Feng Wu. D-fine: redefine regression task in detrs as fine-grained distribution refinement. *arXiv preprint arXiv:2410.13842*, 2024.

[51] J Redmon. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

[52] Joseph Redmon. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[53] 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.

[54] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. Generalized intersection over union: A metric and a loss for bounding box regression. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 658–666, 2019.

[55] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[56] Zhuoran Shen, Mingyuan Zhang, Haiyu Zhao, Shuai Yi, and Hongsheng Li. Efficient attention: Attention with linear complexities. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 3531–3539, 2021.

[57] Yunjie Tian, Lingxi Xie, Jihao Qiu, Jianbin Jiao, Yaowei Wang, Qi Tian, and Qixiang Ye. Fast-itpn: Integrally pre-trained transformer pyramid network with token migration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[58] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International conference on machine learning*, pages 10347–10357. PMLR, 2021.

[59] Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Hervé Jégou. Going deeper with image transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 32–42, 2021.

[60] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[61] Ao Wang, Hui Chen, Lihao Liu, Kai Chen, Zijia Lin, Jungong Han, and Guiguang Ding. Yolov10: Real-time end-to-end object detection. *arXiv preprint arXiv:2405.14458*, 2024.

[62] Chengcheng Wang, Wei He, Ying Nie, Jianyuan Guo, Chuanjian Liu, Yunhe Wang, and Kai Han. Gold-yolo: Efficient object detector via gather-and-distribute mechanism. *Advances in Neural Information Processing Systems*, 36:51094–51112, 2023.

[63] Chengcheng Wang, Wei He, Ying Nie, Jianyuan Guo, Chuanjian Liu, Yunhe Wang, and Kai Han. Gold-yolo: Efficient object detector via gather-and-distribute mechanism. *Advances in Neural Information Processing Systems*, 36, 2024.

[64] 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.

[65] Chien-Yao Wang, Hong-Yuan Mark Liao, Yueh-Hua Wu, Ping-Yang Chen, Jun-Wei Hsieh, and I-Hau Yeh. Cspnet: A new backbone that can enhance learning capability of cnn. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 390–391, 2020.

[66] Chien-Yao Wang, Hong-Yuan Mark Liao, and I-Hau Yeh. Designing network design strategies through gradient path analysis. *arXiv preprint arXiv:2211.04800*, 2022.

[67] Chien-Yao Wang, I-Hau Yeh, and Hong-Yuan Mark Liao. Yolov9: Learning what you want to learn using programmable gradient information. *arXiv preprint arXiv:2402.13616*, 2024.

[68] Jianfeng Wang, Lin Song, Zeming Li, Hongbin Sun, Jian Sun, and Nanning Zheng. End-to-end object detection with fully convolutional network. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 15849–15858, 2021.

[69] Shuo Wang, Chunlong Xia, Feng Lv, and Yifeng Shi. Rt-detrv3: Real-time end-to-end object detection with hierarchical dense positive supervision. In *2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 1628–1636. IEEE, 2025.

[70] Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

[71] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 568–578, 2021.

[72] Yunyang Xiong, Zhanpeng Zeng, Rudrasis Chakraborty, Mingxing Tan, Glenn Fung, Yin Li, and Vikas Singh. Nyströmformer: A nyström-based algorithm for approximating self-attention. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 14138–14148, 2021.

[73] Xianzhe Xu, Yiqi Jiang, Weihua Chen, Yilun Huang, Yuan Zhang, and Xiuyu Sun. Damo-yolo: A report on real-time object detection design. *arXiv preprint arXiv:2211.15444*, 2022.

[74] Xianzhe Xu, Yiqi Jiang, Weihua Chen, Yilun Huang, Yuan Zhang, and Xiuyu Sun. Damo-yolo: A report on real-time object detection design. *arXiv preprint arXiv:2211.15444*, 2022.

[75] Zhiqiang Yang, Qiu Guan, Keer Zhao, Jianmin Yang, Xinli Xu, Haixia Long, and Ying Tang. Multi-branch auxiliary fusion yolo with re-parameterization heterogeneous convolutional for accurate object detection. *arXiv preprint arXiv:2407.04381*, 2024.

[76] Qihang Yu, Yingda Xia, Yutong Bai, Yongyi Lu, Alan L Yuille, and Wei Shen. Glance-and-gaze vision transformer. *Advances in Neural Information Processing Systems*, 34:12992–13003, 2021.

[77] Hongyi Zhang. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

[78] Yian Zhao, Wenyu Lv, Shangliang Xu, Jinman Wei, Guanzhong Wang, Qingqing Dang, Yi Liu, and Jie Chen. Detrs beat yolos on real-time object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16965–16974, 2024.

Table 6: **Hyperparameters for training the YOLOv12 family on COCO [39].**

Hyperparameters	N/S/M/L/X-Scale
Training Configuration	
Epochs	600
Optimizer	SGD
Momentum	0.937
Batch size	32×8
Weight decay	5×10^{-4}
Warm-up epochs	3
Warm-up momentum	0.8
Warm-up bias learning rate	0.0
Initial learning rate	10^{-2}
Final learning rate	10^{-4}
Learning rate schedule	Linear decay
Loss Parameters	
Box loss gain	7.5
Class loss gain	0.5
DFL loss gain	1.5
Augmentation Parameters	
HSV saturation augmentation	0.7
HSV value augmentation	0.4
HSV hue augmentation	0.015
Translation augmentation	0.1
Scale augmentation	0.5/0.9/0.9/0.9/0.9
Mosaic augmentation	1.0
Mixup augmentation	0.0/0.05/0.15/0.15/0.2
Copy-paste augmentation	0.1/0.15/0.4/0.5/0.6
Close mosaic epochs	10

[79] Zhaohui Zheng, Ping Wang, Wei Liu, Jinze Li, Rongguang Ye, and Dongwei Ren. Distance-iou loss: Faster and better learning for bounding box regression. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 12993–13000, 2020.

[80] Dingfu Zhou, Jin Fang, Xibin Song, Chenye Guan, Junbo Yin, Yuchao Dai, and Ruigang Yang. Iou loss for 2d/3d object detection. In *2019 international conference on 3D vision (3DV)*, pages 85–94. IEEE, 2019.

[81] Benjin Zhu, Jianfeng Wang, Zhengkai Jiang, Fuhang Zong, Songtao Liu, Zeming Li, and Jian Sun. Autoassign: Differentiable label assignment for dense object detection. *arXiv preprint arXiv:2007.03496*, 2020.

[82] Lianghui Zhu, Bencheng Liao, Qian Zhang, Xinlong Wang, Wenyu Liu, and Xinggang Wang. Vision mamba: Efficient visual representation learning with bidirectional state space model. *arXiv preprint arXiv:2401.09417*, 2024.

[83] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. *arXiv preprint arXiv:2010.04159*, 2020.

A More Details

Architecture Details. We present the detailed configuration of the overall YOLOv12 network architecture in Table 8. For the backbone, we stack only eight blocks. Except for the convolution layers of downsampling 3×3, the remaining components are two A2 blocks configured with 4 and 1 areas, respectively. We apply grouped convolutions to the downsampling layers with IDs 1 and 2 to save computations, using two groups ($g = 2$) and four groups ($g = 4$), respectively. For Neck,

Table 7: **Detailed performance of YOLOv12 on COCO.**

	$\mathbf{AP}^{val}_{50:95}$ (%)	$\mathbf{AP}^{val}_{50}$(%)	$\mathbf{AP}^{val}_{75}$ (%)	$\mathbf{AP}^{val}_{small}$ (%)	$\mathbf{AP}^{val}_{medium}$ (%)	$\mathbf{AP}^{val}_{large}$ (%)
YOLOv12-N	40.5	56.6	43.7	20.2	45.2	58.2
YOLOv12-S	47.8	64.9	51.5	29.7	53.0	65.3
YOLOv12-M	52.5	69.6	57.0	35.7	58.2	68.9
YOLOv12-L	53.8	71.0	58.6	36.9	59.4	71.0
YOLOv12-X	55.4	72.5	60.3	38.9	60.8	70.9

we follow the design of YOLOv11, replacing only the first three C3K2 modules with A2 blocks. However, we do not enable area attention in these blocks (A2 = False); instead, we retain only the feature integration design of R-ELAN. The Head design remains unchanged from YOLOv11.

To implement area attention in the A2 block, we use a convolutional layer 1×1 with a batch normalization layer to construct the projections for qk and v. Similarly, we use convolution with Batch Normalization layers for the output projection and position perceiver, which facilitates optimization. For the MLP module, we stack two consecutive 1×1 convolution layers with a non-linear activation layer (SiLU function) between them.

Table 8: Network configurations of YOLOv12.

ID	Module	Route	A2	$\mathbf{Num.}_{\mathbf{area}}$	Dims.	Depth	Size	Stride
BackBone								
0	Conv	–	–	–	64	1	3	2
1	Conv	0	–	–	128	1	3	2 (g = 2)
2	C3K2	1	-	-	256	2	-	1
3	Conv	2	–	–	256	1	3	2 (g = 4)
4	C3K2	3	True	-	512	2	–	–
5	Conv	4	–	–	512	1	-	2
6	A2 block	5	True	4	512	4	–	–
7	Conv	6	–	–	1024	1	3	2
8	A2 block	7	True	1	1024	4	–	–
Neck								
9	Up	8	–	–	1024	1	2	2
10	Concat	9, 6	–	–	1024	1	–	–
11	A2 block	10	False	–	512	2	–	–
12	Up	11	–	–	512	1	2	2
13	Concat	12, 4	–	–	512	1	–	–
14	A2 block	13	False	–	256	2	–	–
15	Conv	14	–	–	256	1	3	2
16	Concat	15, 11	–	–	256	1	–	–
17	A2 block	16	False	–	512	2	–	–
18	Conv	17	–	–	512	1	3	2
19	Concat	18, 8	–	–	512	1	–	–
20	C3K2	19	–	–	1024	2	–	–
Head								
21	Predict	14, 17, 20	–	–	–	–	–	–

Training Details. All YOLOv12 models are trained using the default SGD optimizer for 600 epochs. Following previous works [64, 26, 67, 61], the SGD momentum and weight decay are set to 0.937 and 5×10^{-4}, respectively. The initial learning rate is set to 1×10^{-2} and decays linearly to 1×10^{-4} throughout the training process. Data augmentations, including Mosaic [5, 64], Mixup [83], and copy-paste augmentation [77], are applied to enhance training. Following YOLOv11 [31], we adopt the Albumentations library [8]. Detailed hyperparameters are presented in Table 6. The N/S/M models are trained on 4× NVIDIA A6000 GPUs and the L/X models are trained on 8× NVIDIA A800 GPUs. Following established conventions [26, 67, 61, 31], we report the standard mean average

precision (mAP) on different object scales and IoU thresholds. In addition, we report the average latency in all images.

Result Details. We report more details of the YOLOv12 results in Table 7 including $AP^{val}_{50:95}$, AP^{val}_{50}, AP^{val}_{75}, AP^{val}_{small}, AP^{val}_{medium}, AP^{val}_{large}.

Table 9: **Comparative analysis of inference speed across different GPUs (RTX 3080, RTX A5000, and RTX A6000)**. Inference latency: milliseconds (ms) for FP32 and FP16 precision.

Model	Scale	FLOPs (G)	RTX 3080	A5000	A6000
YOLOv9 [67]	T	8.2	2.4/1.5	2.4/1.5	2.3/1.5
	S	26.4	3.7/1.9	3.3/1.9	3.3/1.8
	M	76.3	6.3/2.7	5.4/2.4	5.1/2.4
	C	102.1	7.4/2.8	6.4/2.6	6.0/2.6
	E	189.0	15.5/6.5	14.0/6.1	12.9/5.7
YOLOv10 [61]	N	6.7	1.6/1.0	1.6/1.0	1.6/1.0
	S	21.6	2.8/1.3	2.4/1.3	2.4/1.2
	M	59.1	5.3/2.4	4.3/2.3	4.2/2.1
	B	92.0	6.7/2.8	5.4/2.5	5.1/2.6
	X	160.4	10.7/4.5	7.2/3.5	6.8/3.2
YOLOv11 [31]	N	6.5	1.6/0.9	1.6/0.9	1.5/0.9
	S	21.5	2.7/1.3	2.3/1.3	2.3/1.3
	M	68.0	5.4/2.2	4.4/2.1	4.3/2.0
	L	86.9	6.9/2.9	5.7/2.6	5.6/2.5
	X	194.9	13.5/4.9	10.4/4.5	8.9/3.9
YOLOv12	N	6.0	1.6/1.0	1.6/1.0	1.6/1.0
	S	19.5	2.7/1.3	2.3/1.3	2.3/1.3
	M	59.9	5.2/2.1	4.4/2.1	4.3/2.1
	L	82.6	6.9/2.9	5.8/2.6	5.7/2.5
	X	184.9	13.6/4.9	10.7/4.6	9.5/4.0

Table 10: Comparison of YOLOv12 series with more lightweight or stronger detectors including DEYO [48], DAMO-YOLO [74], and recent D-FINE [50].

Model	#Param. (M)	FLOPs (G)	$AP^{val}_{50:95}$	Latency (ms)
DEYO-tiny [48]	4.0	8.0	37.6	2.01
YOLOv12-N (Ours)	**2.6**	**6.0**	**40.5**	**1.62**
DAMO-YOLO-T [74]	8.5	18.1	42.0	2.21
DAMO-YOLO-S [74]	16.3	37.8	46.0	3.18
DEYO-S [48]	14.0	26.0	45.8	3.34
YOLOv12-S (Ours)	**9.1**	**19.5**	**47.8**	**2.44**
DAMO-YOLO-M [74]	28.2	61.8	49.2	4.57
DAMO-YOLO-L [74]	42.1	97.3	50.8	6.48
DEYO-M [48]	33.0	78.0	50.7	7.14
YOLOv12-M (Ours)	**19.7**	**59.9**	**52.5**	**4.30**
YOLOv7 [64]	36.9	104.7	51.2	17.03
D-FINE-L [50]	31	91	54.0	8.07
YOLOv12-L (Ours)	**26.6**	**82.6**	**53.8**	**5.89**
D-FINE-X [50]	62	202	55.8	12.89
YOLOv12-X (Ours)	**59.5**	**184.9**	**55.4**	**10.47**

B More Comparisons

Latency Comparison on Various GPUs. Table 9 presents a comparative analysis of inference speed across different GPUs, evaluating YOLOv9 [67], YOLOv10 [61], YOLOv11 [31], and our YOLOv12 on RTX 3080, RTX A5000, and RTX A6000 with FP32 and FP16 precision. To ensure consistency, all results are obtained on the same hardware, and YOLOv9 [67] and YOLOv10 [61] are evaluated using the integrated codebase of Ultralytics [31]. Across all tested models, FP16 inference is significantly faster than FP32, often reducing latency by more than 50%. The inference speed generally improves as we move from RTX 3080 to A6000. **N-Scale:** YOLOv12-N achieves similar latencies (1.6 ms for FP32 and 1.0-1.1 ms for FP16), matching or slightly outperforming their YOLOv10 and YOLOv11 counterparts. **S-Scale:** YOLOv12-S maintains lower latency than YOLOv9-S and YOLOv10-S while achieving superior FLOPs efficiency. **M/L-Scale:** YOLOv12-M and YOLOv12-L demonstrate competitive speed, with FP16 latencies close to their counterparts, while offering improved accuracy. **X-Scale:** YOLOv12-X achieves 13.6 ms (FP32) and 4.9 ms (FP16) on RTX 3080, matching YOLOv11-X and YOLOv10-X in efficiency.

Comparison with Other Detectors. Table 10 presents a comparison with other lightweight and state-of-the-art real-time object detectors, such as DAMO-YOLO [74], YOLOv7 [64], DEYO [48], and D-FINE [50]. **N-scale**: YOLOv12-N achieves 40.5% mAP, surpassing DEYO-tiny while requiring lower computational cost (6.0G FLOPs vs. 8.0G) and achieving faster inference (1.62ms vs. 2.01ms). **S-scale**: YOLOv12-S (9.1M parameters, 19.5G FLOPs) achieves 47.8% mAP, outperforming DAMO-YOLO-S and DEYO-S with a better balance between accuracy and efficiency. **M-scale**: YOLOv12-M (19.7M parameters, 59.9G FLOPs) achieves 52.5% mAP, outperforming DAMO-YOLO-M and DEYO-M while being more efficient. **L-scale**: YOLOv12-L achieves 53.8% mAP with 82.6G FLOPs, surpassing DAMO-YOLO-L and YOLOv7 while maintaining a faster inference speed. **X-scale**: YOLOv12-X (59.5M parameters, 184.9G FLOPs) achieves 55.4% mAP, surpassing D-FINE-X while being more computationally efficient and faster.

C Contributions & Broader Impact

Contributions. This work effectively incorporates attention-centric architectures as the core backbone of the YOLO system, achieving state-of-the-art performance. To this end, we make three key contributions:

1. **Area Attention Module (A2).** We propose a simple yet efficient Area Attention module that maintains a large receptive field while reducing the computational complexity of attention, significantly improving inference speed.
2. **Residual Efficient Layer Aggregation Networks (R-ELAN).** To address optimization challenges in attention-based models, especially at scale, we design R-ELAN. It builds on ELAN [64] and introduces: **(i)** a block-level residual design with scaling techniques and **(ii)** an improved feature aggregation strategy.
3. **Optimization for YOLO Integration.** We refine attention-centric architectures to better integrate with the YOLO framework. Key modifications include: incorporating FlashAttention to mitigate memory access issue; using a decoupled projection strategy to construct q, k, and v during attention computation, thus avoiding redundant feature reorganization; removing positional encoding for a leaner design; reducing the MLP ratio from 4 to 1.5 to better balance attention and FFN computation; and decreasing the depth of stacked blocks to facilitate optimization.

Broader Impact. This study breaks the dominance of CNN architectures in YOLO systems by utilizing the proposed attention mechanism to achieve one of the most advanced YOLO object detectors. It opens up new follow-up research directions, such as transferring other successful techniques from attention mechanisms to further enhance this framework, unlocking greater potential.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction provide a comprehensive overview of the background and motivation of this study, effectively outlining its main contributions point-by-point, thus accurately reflecting the paper's scope and significance.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and essential assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation, as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We primarily focused on discussing the limitations associated with this study in section 4.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: The paper includes the full set of assumptions and correct proofs for each theoretical result, which mainly focuses on the receptive field and the complexity of some attentions.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: All information regarding the key contribution of this paper is reproducible and the code will be open-sourced.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: All code, data, and models are open sources with detailed reproducible instructions.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: The paper specifies detailed experimental configurations in Section 3.1 and more details are provided in Appendix, providing readers with essential information to comprehend the results. Following established conventions in the field of real-time object detection, the evaluation protocol encompasses standard practices commonly found in the relevant literature, ensuring readers can refer to established methodologies.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: We did not include an analysis of the statistical significance of the experiments mainly due to the prohibitively expensive training cost of YOLO models and our limited computing resources. However, we have provided the code, hyperparameters, and random seeds used in our experiments to facilitate the reproducibility of our findings. We would like to point out that, due to the extensive amount of training data, the statistical patterns of

the experiment results are likely to remain consistent across different trials. Consequently, reporting error bars or other information about statistical significance is not a common practice in studies developing YOLO models.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The N/S/M scale models are trained on a 4 × A6000 GPU server and L/X models are trained on a 8 × A800 GPU server.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: After carefully reviewing the referenced document, we certify that the research conducted in the paper conforms, in every respect, with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: The paper primarily focuses on real-time object detection using publicly available datasets that have undergone thorough validation without societal impact.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The proposed models are real-time object detection models trained on a benchmark dataset MSCOCO. This dataset has been extensively used in the computer vision community and has undergone comprehensive safety risk assessments.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

Justification: In the paper, we specified the datasets and code sources used (e.g., ultralytics), and provided appropriate citations in the reference section. Additionally, we ensured transparency by including the sources of any modified code files, making the changes traceable.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: We have included the code, along with detailed usage instructions, in the Appendix. After the review process is completed, we will make the code publicly available to the community.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: This study does not involve any crowdsourcing experiments or research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: No crowdsourcing experiments or research with human subjects were involved in this study. All experiments were conducted using code and GPU servers.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [No]

    Justification: This study focuses on the real-time object detection field and does not involve any usage of LLMs.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.