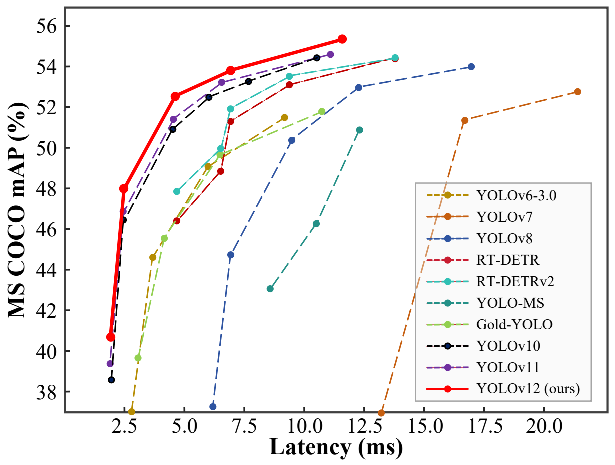
YOLOv12: Attention-Centric Real-Time Object Detectors

Yunjie Tian
University at Buffalo
yunjieti@buffalo.edu

Qixiang Ye
University of Chinese Academy of Sciences
qxye@ucas.ac.cn

David Doermann
University at Buffalo
doermann@buffalo.edu

github.com/sunsmarterjie/yolov12
-- Technical Report --



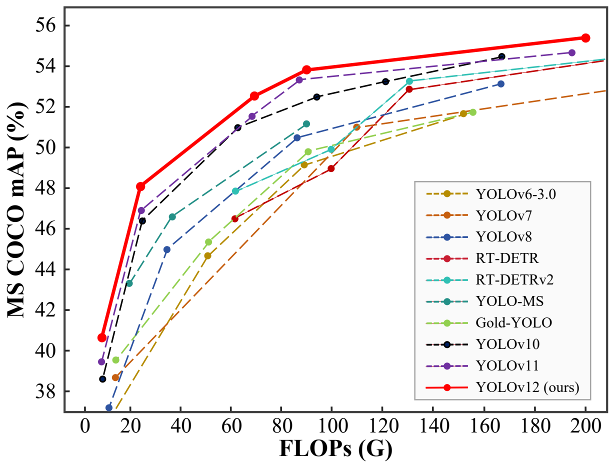


Figure 1. **Comparisons with other popular methods in terms of latency-accuracy (left) and FLOPs-accuracy (right) trade-offs.**

Abstract

Enhancing the network architecture of the YOLO framework has been crucial for a long time, but 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 all popular real-time object detectors in accuracy with competitive speed. For example, YOLOv12-N achieves 40.6% *mAP with an inference latency of* 1.64 *ms on a T4 GPU, outperforming advanced YOLOv10-N / YOLOv11-N by* 2.1%/1.2% *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-DETR / RT-DETRv2: YOLOv12-S beats RT-DETR-R18 / RT-DETRv2-R18 while running* 42% *faster, using only* 36% *of the computation and* 45% *of the parameters. More comparisons are shown in Figure 1.*

1. Introduction

Real-time object detection has consistently attracted significant attention due to its low-latency characteristics, which provide substantial practicality [4, 17, 24, 28]. Among them, the YOLO series [3, 24, 28, 29, 32, 45–47, 53, 57, 58] 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 [8, 35, 43, 44, 48, 67, 68], label assignment [22, 23, 34, 59, 69], network architecture design has remained a critical research priority [24, 28, 32, 57, 58]. Although attention-centric vision transformer (ViT) architectures have been proven to possess stronger modeling capabilities, even in small models [20, 21, 25, 50], 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 [13, 14]). As a result, under a similar computational budget, CNN-based architectures outperform attention-based ones by a factor of $\sim 3\times$ [38], which significantly limits the adoption of attention mechanisms in YOLO systems where high inference speed is critical.

This paper aims to address these challenges and further builds an attention-centric YOLO framework, namely YOLOv12. We introduce three key improvements. **First**, we propose a simple yet efficient area attention module (A2), which maintains a large receptive field while reducing the computational complexity of attention in a very simple way, thus enhancing speed. **Second**, we introduce the residual efficient layer aggregation networks (R-ELAN) to address the optimization challenges introduced by attention (primarily large-scale models). R-ELAN introduces two improvements based on the original ELAN [57]: **(i)** a block-level residual design with scaling techniques and **(ii)** a redesigned feature aggregation method. **Third**, we make some architectural improvements beyond the vanilla attention to fit the YOLO system. We upgrade traditional attention-centric architectures including: introducing FlashAttention to conquer the memory access issue of attention, removing designs such as positional encoding to make the model fast and clean, adjusting the MLP ratio from 4 to 1.2 to balance the computation between attention and feed forward network for better performance, reducing the depth of stacked blocks to facilitate optimization, and making use of convolution operators as much as possible to leverage their computational efficiency.

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 YOLOv11 [28] 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 40.6% mAP, outperforming YOLOv10-N [53] by 2.1% mAP while maintaining a faster inference speed, and YOLOv11-N [28] by 1.2% mAP with a comparable speed. This advantage remains consistent across other scale models. Compared to RT-DETR-R18 [66] / RT-DETRv2-R18 [40], YOLOv12-S is 1.5%/0.1% mAP better, while reports 42%/42% faster latency speed, requiring only 36%/36% of their computations and 45%/45% of their parameters.

In summary, the contributions of YOLOv12 are two-fold: 1) it establishes an attention-centric, simple yet efficient YOLO framework that, through methodological innovation and architectural improvements, breaks the dominance of CNN models in YOLO series. 2) without relying on additional techniques such as pretraining, YOLOv12 achieves state-of-the-art results with fast inference speed and higher detection accuracy, demonstrating its potential.

2. Related Work

Real-time Object Detectors. Real-time object detectors have consistently attracted the community's attention due to their significant practical value. The YOLO series [3, 9, 24, 28, 29, 32, 45–47, 53, 54, 57, 58] has emerged as a leading framework for real-time object detection. The early YOLO systems [45–47] establish the framework for the YOLO series, primarily from a model design perspective. YOLOv4 [3] and YOLOv5 [29] add CSPNet [55], data augmentation, and multiple feature scales to the framework. YOLOv6 [32] further advance these with BiC and SimCSPSPPF modules for the backbone and neck, alongside anchor-aided training. YOLOv7 [57] introduce E-ELAN [56] (efficient layer aggregation networks) for improved gradient flow and various bag-of-freebies, while YOLOv8 [24] integrate a efficient C2f block for enhanced feature extraction. In recent iterations, YOLOv9 [58] introduce GELAN for architecture optimization and PGI for training improvement, while YOLOv10 [53] apply NMS-free training with dual assignments for efficiency gains. YOLOv11 [28] further reduces latency and increases accuracy by adopting the C3K2 module (a specification of GELAN [58]) and lightweight depthwise separable convolution in the detection head. Recently, an end-to-end object detection method, namely RT-DETR [66], improved traditional end-to-end detectors [7, 33, 37, 42, 71] to meet real-time requirements by designing an efficient encoder and an uncertainty-minimal query selection mechanism. RT-DETRv2 [40] further enhances it with bag-of-freebies. Unlike previous YOLO series, this study aims to build a YOLO framework centered around attention to leverage the superiority of the attention mechanism.

Efficient Vision Transformers. Reducing computational costs from global self-attention is crucial to effectively applying vision transformers in downstream tasks. PVT [61] addresses this using multi-resolution stages and downsampling features. Swin Transformer [39] limits self-attention to local windows and adjusts the window partitioning style to connect non-overlapping windows, balancing communication needs with memory and computation demands. Other methods, such as axial self-attention [26] and criss-cross attention [27], calculate attention within horizontal and vertical windows. CSWin transformer [16] builds on this by introducing cross-shaped window self-attention, computing attention along horizontal and vertical stripes in

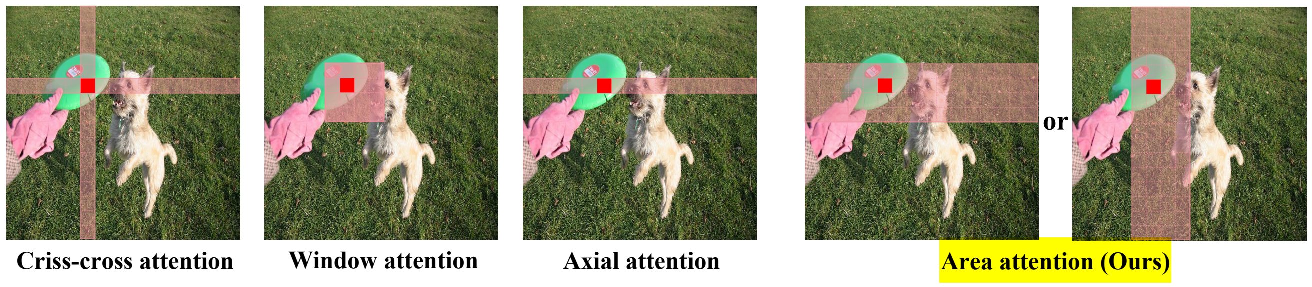


Figure 2. **Comparison of the representative local attention mechanisms with our area attention.** Area Attention adopts the most straightforward equal partitioning way to divide the feature map into l areas vertically or horizontally. (default is 4). This avoids complex operations while ensuring a large receptive field, resulting in high efficiency.

parallel. In addition, local-global relations are established in works such as [12, 64], improving efficiency by reducing reliance on global self-attention. Fast-iTPN [50] improves downstream task inference speed with token migration and token gathering mechanisms. Some approaches [31, 49, 60, 62] use linear attention to decrease the complexity of the attention. Although Mamba-based vision models [38, 70] aim for linear complexity, they still fall short of real-time speeds [38]. FlashAttention [13, 14] identifies high bandwidth memory bottlenecks that lead to inefficient attention computation and addresses them through I/O optimization, reducing memory access to enhance computational efficiency. In this study, we discard complex designs and propose a simple area attention mechanism to reduce the complexity of attention. Additionally, we employ FlashAttention to overcome inherent memory accessing problems of the attention mechanism [13, 14].

3. Approach

This section introduces YOLOv12, an innovation in the YOLO framework from the perspective of network architecture with attention mechanism.

3.1. Efficiency Analysis

The attention mechanism, while highly effective in capturing global dependencies and facilitating tasks such as natural language processing [5, 15] and computer vision [19, 39], is inherently slower than convolution neural networks (CNN). Two primary factors contribute to this discrepancy in speed.

Complexity. First, the computational complexity of the self-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 $\mathcal{O}(L^2d)$ 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.*, $\mathcal{O}(kLd)$, where k is the kernel size and is typically much smaller than L. As a result, self-attention becomes computationally prohibitive, especially for large inputs such as high-resolution images or long sequences.

Moreover, another significant factor is that most attention-based vision transformers, due to their complex designs (*e.g.*, window partitioning/reversing in Swin transformer [39]) and the introduction of additional modules (*e.g.*, positional encoding), gradually accumulate speed overhead, resulting in an overall slower speed compared to CNN architectures [38]. In this paper, the design modules utilize simple and clean operations to implement attention, ensuring efficiency to the greatest extent.

Computation. Second, in the attention computation process, memory access patterns are less efficient compared to CNNs [13, 14]. Specifically, during self-attention, intermediate maps such as the attention map (QK^T) and the softmax map $(L \times L)$ need to be stored from high-speed GPU SRAM (the actual location of the computation) to high bandwidth GPU memory (HBM) and later retrieved during the computation, and the read and write speed of the former is more than 10 times that of the latter, thus resulting in significant memory accessing overhead and increased wall-clock time[1]. Additionally, irregular memory access patterns in attention introduce further latency compared to CNNs, which utilize structured and localized memory access. CNNs benefit from spatially constrained kernels, enabling efficient memory caching and reduced latency due to their fixed receptive fields and sliding-window operations.

These two factors, quadratic computational complexity and inefficient memory accessing, together render attention mechanisms slower than CNNs, particularly in real-time or resource-constrained scenarios. Addressing these limitations has become a critical area of research, with approaches such as sparse attention mechanisms and memory-efficient approximations (*e.g.*, Linformer [60] or Performer [11]) aiming to mitigate the quadratic scaling.

[1]This problem has been solved by FlashAttention [13, 14], which will be directly adopted during model design

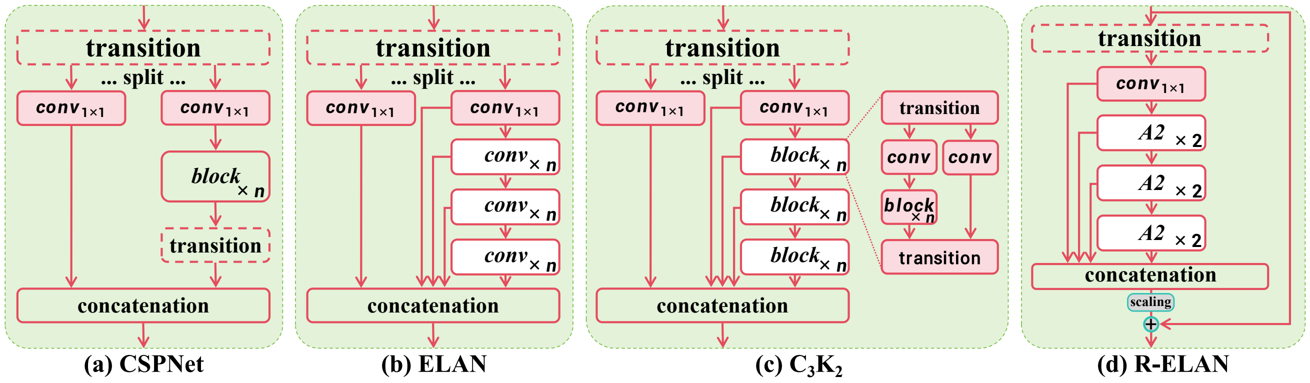


Figure 3. **The architecture comparison with popular modules** including (a): CSPNet [55], (b) ELAN [56], (c) C3K2 (a case of GELAN) [28, 58], and (d) the proposed R-ELAN (residual efficient layer aggregation networks).

3.2. Area Attention

A simple approach to reduce the computational cost of vanilla attention is to use the linear attention mechanism [49, 60], which reduces the complexity of vanilla attention 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 $2nhd^2$, decreasing the computational cost since $n > d$. However, linear attention suffers from global dependency degradation [30], instability [11], and distribution sensitivity [63]. Furthermore, due to the low-rank bottleneck [2, 10], it offers only limited speed advantages when applied to YOLO with input resolution of 640×640.

An alternative approach to effectively reduce complexity is the local attention mechanism (*e.g.*, Shift window [39], criss-cross attention [27], and axial attention [16]), 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. In this study, we propose the simple yet efficient area attention module. As illustrated in Figure 2, the feature map with the resolution of (H, W) is divided into l segments of size $(\frac{H}{l}, W)$ or $(H, \frac{W}{l})$. This eliminates explicit window partitioning, requiring only a simple reshape operation, and thus achieves faster speed. We empirically set the default value of l to 4, reducing the receptive field to $\frac{1}{4}$ of the original, yet it still maintains a large receptive field. With this approach, the computational cost of the attention mechanism is reduced from $2n^2hd$ to $\frac{1}{2}n^2hd$. We show that despite the complexity n^2, this is still efficient enough to meet the real-time requirements of the YOLO system when n is fixed at 640 (it increases n if the input resolution increases). Interestingly, we find that this modification has only a slight impact on performance but significantly improves speed.

3.3. Residual Efficient Layer Aggregation Networks

Efficient layer aggregation networks (ELAN) [57] are designed to improve feature aggregation. As shown in Figure 3 (b), ELAN splits the output of a transition layer (a 1×1 convolution), processes one split through multiple modules, then concatenates the all the outputs and applies another transition layer (a 1×1 convolution) to align dimensions. However, as analyzed by [57], this architecture can introduce instability. We argue that such a design causes gradient blocking and lacks residual connections from input to output. Furthermore, we build the network around the attention mechanism, which presents 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 problem, we propose residual efficient layer aggregation networks (R-ELAN), Figure 3 (d). In contrast, we introduce a residual shortcut from input to output throughout the block with a scaling factor (default to 0.01). This design is similar to layer scaling [52], which is introduced to build a deep vision transformer. However, applying layer scaling for each area attention will not conquer the optimization challenge and introduce slowdown on latency. This shows that the introduction of the attention mechanism is not the only reason for convergence but the ELAN architecture itself, which verifies the rationale behind our R-ELAN design.

We also design a new aggregation approach as shown in Figure 3 (d). The original ELAN layer processes the input of the module by first passing it through a transition layer, which then splits it into two parts. One part is further processed by subsequent blocks, and finally both parts are concatenated to produce the output. In contrast, our design applies a transition layer to adjust the channel dimensions and produces a single feature map. This feature map is then processed through subsequent blocks followed by the concatenation, forming a bottleneck structure. This approach

Table 1. **Comparison with popular state-of-the-art real-time object detectors. All results are obtained using** 640×640 **inputs.**

Method	FLOPs (G)	#Param. (M)	$\mathbf{AP}^{val}_{50:95}$ (%)	$\mathbf{AP}^{val}_{50}$ (%)	$\mathbf{AP}^{val}_{75}$ (%)	Latency (ms)
YOLOv6-3.0-N [32]	11.4	4.7	37.0	52.7	–	2.69
Gold-YOLO-N [54]	12.1	5.6	39.6	55.7	–	2.92
YOLOv8-N [24]	8.7	3.2	37.4	52.6	40.5	1.77
YOLOv10-N [53]	6.7	2.3	38.5	53.8	41.7	1.84
YOLO11-N [28]	6.5	2.6	39.4	55.3	42.8	1.5
YOLOv12-N (Ours)	**6.5**	**2.6**	**40.6**	**56.7**	**43.8**	**1.64**
YOLOv6-3.0-S [32]	45.3	18.5	44.3	61.2	–	3.42
Gold-YOLO-S [54]	46.0	21.5	45.4	62.5	–	3.82
YOLOv8-S [24]	28.6	11.2	45.0	61.8	48.7	2.33
RT-DETR-R18 [66]	60.0	20.0	46.5	63.8	–	4.58
RT-DETRv2-R18 [41]	60.0	20.0	47.9	64.9	–	4.58
YOLOv9-S [58]	26.4	7.1	46.8	63.4	50.7	–
YOLOv10-S [53]	21.6	7.2	46.3	63.0	50.4	2.49
YOLO11-S [28]	21.5	9.4	46.9	63.9	50.6	2.5
YOLOv12-S (Ours)	**21.4**	**9.3**	**48.0**	**65.0**	**51.8**	**2.61**
YOLOv6-3.0-M [32]	85.8	34.9	49.1	66.1	–	5.63
Gold-YOLO-M [54]	87.5	41.3	49.8	67.0	–	6.38
YOLOv8-M [24]	78.9	25.9	50.3	67.2	54.7	5.09
RT-DETR-R34 [66]	100.0	36.0	48.9	66.8	–	6.32
RT-DETRv2-R34 [41]	100.0	36.0	49.9	67.5	–	6.32
YOLOv9-M [58]	76.3	20.0	51.4	68.1	56.1	–
YOLOv10-M [53]	59.1	15.4	51.1	68.1	55.8	4.74
YOLO11-M [28]	68.0	20.1	51.5	68.5	55.7	4.7
YOLOv12-M (Ours)	**67.5**	**20.2**	**52.5**	**69.6**	**57.1**	**4.86**
YOLOv6-3.0-L [32]	150.7	59.6	51.8	69.2	–	9.02
Gold-YOLO-L [54]	151.7	75.1	51.8	68.9	–	10.65
YOLOv8-L [24]	165.2	43.7	53.0	69.8	57.7	8.06
RT-DETR-R50 [66]	136.0	42.0	53.1	71.3	–	6.90
RT-DETRv2-R50 [41]	136.0	42.0	53.4	71.6	–	6.90
YOLOv9-C [58]	102.1	25.3	53.0	70.2	57.8	–
YOLOv10-B [53]	92.0	19.1	52.5	69.6	57.2	5.74
YOLOv10-L [53]	120.3	24.4	53.2	70.1	58.1	7.28
YOLO11-L [28]	86.9	25.3	53.3	70.1	58.2	6.2
YOLOv12-L (Ours)	**88.9**	**26.4**	**53.7**	**70.7**	**58.5**	**6.77**
YOLOv8-X [24]	257.8	68.2	54.0	71.0	58.8	12.83
RT-DETR-R101 [66]	259.0	76.0	54.3	72.7	–	13.5
RT-DETRv2-R101 [41]	259.0	76.0	54.3	72.8	–	13.5
YOLOv10-X [53]	160.4	29.5	54.4	71.3	59.3	10.70
YOLO11-X [28]	194.9	56.9	54.6	71.6	59.5	11.3
YOLOv12-X (Ours)	**199.0**	**59.1**	**55.2**	**72.0**	**60.2**	**11.79**

not only preserves the original feature integration capability, but also reduces both computational cost and parameter / memory usage.

3.4. Architectural Improvements

In this section, we will introduce the overall architecture and some improvements over the vanilla attention mechanism. Some of them are not initially proposed by us.

Many attention-centric vision transformers are designed with the plain-style architectures [1, 18, 19, 21, 25, 51], while we retain the hierarchical design of the previous YOLO systems [3, 24, 28, 29, 32, 45–47, 53, 57, 58] and will demonstrate the necessity of this. We remove the design of stacking three blocks in the last stage of the backbone, which is present in recent versions [24, 28, 53, 58]. Instead, we retain only a single R-ELAN block, reducing the total number of blocks and contributing to optimization. We inherit the first two stages of the backbone from YOLOv11 [28] and do not use the proposed R-ELAN.

Additionally, we modify several default configurations in the vanilla attention mechanism to better suit the YOLO system. These modifications include adjusting the MLP ratio from 4 to 1.2 (or 2 for the N- / S- / M-scale models) is used to better allocate computational resources for better performance, adopting `nn.Conv2d+BN` instead of `nn.Linear+LN` to fully exploit the efficiency of convolution operators, removing positional encoding, and introduce a large separable convolution (7×7) (namely position perceiver) to help the area attention perceive position information. The effectiveness of these modifications will be validated in Section 4.5.

4. Experiment

This section is divided into four parts: experimental setup, a systematic comparison with popular methods, ablation studies to validate our approach, and an analysis with visualizations to further explore YOLOv12.

4.1. Experimental Setup

We validate the proposed method on the MSCOCO 2017 dataset [36]. The YOLOv12 family includes 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, which remains the same as YOLOv11 [28]. We adopt a linear learning rate decay schedule and perform a linear warm-up for the first 3 epochs. Following the approach in [53, 66], the latencies of all models are tested on a T4 GPU with TensorRT FP16.

Baseline. We choose the previous version of YOLOv11 [28] as our baseline. The model scaling strategy is also consistent with it. We use several of its proposed C3K2 blocks (that is a special case of GELAN [58]). We do not use any more tricks beyond YOLOv11 [28].

4.2. Comparison with State-of-the-arts

We present a performance comparison between YOLOv12 and other popular real-time detectors in Table 1.

For N-scale models, YOLOv12-N outperforms YOLOv6-3.0-N [32], YOLOv8-N [58], YOLOv10-N [53], and YOLOv11 [28] by 3.6%, 3.3%, 2.1%, and 1.2% in mAP, respectively, while maintaining similar or even fewer computations and parameters, and achieving the fast latency speed of 1.64 ms/image.

For S-scale models, YOLOv12-S, with 21.4G FLOPs and 9.3M parameters, achieves 48.0 mAP with 2.61 ms/image latency. It outperforms YOLOv8-S [24], YOLOv9-S [58], YOLOv10-S [53], and YOLOv11-S [28] by 3.0%, 1.2%, 1.7%, and 1.1%, respectively, while maintaining similar or fewer computations. Compared to the end-to-end detectors RT-DETR-R18 [66] / RT-DETRv2-R18 [41], YOLOv12-S achieves beatable performance but with much better inference speed and less computational cost and fewer parameters.

For M-scale models, YOLOv12-M, with 67.5G FLOPs and 20.2M parameters, achieves 52.5 mAP performance and 4.86 ms/image speed. Compared to Gold-YOLO-M [54], YOLOv8-M [24], YOLOv9-M [58], YOLOv10 [53], YOLOv11 [28], and RT-DETR-R34 [66] / RT-DETRv2-R34 [40], YOLOv12-S enjoys superiority.

For L-scale models, YOLOv12-L even surpasses YOLOv10-L [53] with 31.4G fewer FLOPs. YOLOv12-L beats YOLOv11 [28] by 0.4% mAP with comparable FLOPs and parameters. YOLOv12-L also outperforms RT-DERT-R50 [66] / RT-DERTv2-R50 [41] with faster speed, fewer FLOPs (34.6%) and fewer parameters (37.1%).

For X-scale models, YOLOv12-X significantly outperforms YOLOv10-X [53] / YOLOv11-X [28] by 0.8% and 0.6%, respectively, with comparable speed, FLOPs, and parameters. YOLOv12-X again beats RT-DETR-R101 [66] / RT-DETRv2-R101 [40] with faster speed, fewer FLOPs (23.4%) and fewer parameters (22.2%).

In particular, if the L- / X-scale models are evaluated using FP32 precision (which requires saving the model separately in FP32 format), YOLOv12 will achieve an improvement of $\sim 0.2\%$ mAP. This means that YOLOv12-L/X will report 33.9%/55.4% mAP.

4.3. Ablation Studies

- **R-ELAN.** Table 2 evaluates the effectiveness of the proposed residual efficient layer networks (R-ELAN) using YOLOv12-N/L/X models. The results reveal two key findings: **(i)** For small models like YOLOv12-N, residual connections do not impact convergence but degrade performance. In contrast, for larger models (YOLOv12-L/X), they are essential for stable training. In particular, YOLOv12-X requires a minimal scaling factor (0.01) to ensure convergence. **(ii)** The proposed feature integration method effectively reduces the complexity of the model in terms of FLOPs and parameters while maintaining comparable performance with only a marginal decrease.
- **Area Attention.** We conduct ablation experiments to validate the effectiveness of area attention, with results presented in Table 3. Evaluations are performed on YOLOv12-

Table 2. **Ablation on the proposed residual efficient layer aggregation networks (R-ELAN).** Vanilla: Uses the original ELAN design; Re-Aggre.: Employs our proposed feature integration method; Resi.: Utilizes the residual block technique; Scaling: The scaling factor for the residual connection.

Model	Vanilla	Re-Aggre.	Resi.	Scaling	Convergence	FLOPs (G)	#Param. (M)	$AP^{val}_{50:95}$ (%)
YOLOv12-N	✓	✗	✗	–	✓	6.9	2.7	40.8
	✗	✓	✗	–	✓	6.5	2.6	40.6
	✗	✗	✓	0.1	✓	6.5	2.6	40.3
YOLOv12-L	✓	✗	✗	–	✗	–	–	–
	✗	✓	✗	–	✗	–	–	–
	✗	✓	✓	0.1	✓	88.9	26.4	53.3
	✗	✓	✓	0.01	✓	88.9	26.4	53.7
	✗	✗	✓	0.01	✓	94.3	27.8	53.8
YOLOv12-X	✓	✗	✗	–	✗	–	–	–
	✗	✓	✗	–	✗	–	–	–
	✗	✓	✓	0.1	✗	–	–	–
	✗	✓	✓	0.01	✓	199.0	59.1	55.2
	✗	✗	✓	0.01	✓	211.3	62.3	55.3

Table 3. **Ablation on the proposed area attention**. With area attention (✓), YOLOv12-N/S/X models run significantly faster on both GPU (CUDA) and CPU. CUDA results are measured on RTX 3080/A5000. Inference latency: milliseconds (ms) for FP32 and FP16 precision. (All results are obtained without using FlashAttention [13, 14].)

Model	Area Attention	CUDA FP32	CUDA FP16	CPU
YOLOv12-N	✗	2.7/2.5	1.5/1.5	62.9
	✓	2.0/2.0	1.3/1.3	31.4
YOLOv12-S	✗	5.1/4.4	2.5/2.2	130.0
	✓	3.5/3.1	1.7/1.7	78.4
YOLOv12-X	✗	26.4/21.9	11.1/10.4	804.2
	✓	18.2/14.3	7.1/6.7	512.5

N/S/X models, measuring the inference speed on both the GPU (CUDA) and CPU. CUDA results are obtained using RTX 3080 and A5000, while CPU performance is measured on an Intel Core i7-10700K @ 3.80GHz. The results demonstrate a significant speedup with area attention (✓). For example, with FP32 on RTX 3080, YOLOv12-N achieves a 0.7ms reduction in inference time. This performance gain is consistently observed across different models and hardware configurations. We do not use FlashAttention [13, 14] in this experiment because it would significantly reduce the speed difference.

Table 4. **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 [58]	T	8.2	2.4/1.5	2.4/1.6	2.3/1.7
	S	26.4	3.7/1.9	3.4/2.0	3.5/1.9
	M	76.3	6.5/2.8	5.5/2.6	5.2/2.6
	C	102.1	8.0/2.9	6.4/2.7	6.0/2.7
	E	189.0	17.2/6.7	14.2/6.3	13.1/5.9
YOLOv10 [53]	N	6.7	1.6/1.0	1.6/1.0	1.6/1.0
	S	21.6	2.8/1.4	2.4/1.4	2.4/1.3
	M	59.1	5.7/2.5	4.5/2.4	4.2/2.2
	B	92.0	6.8/2.9	5.5/2.6	5.2/2.8
YOLOv11 [28]	N	6.5	1.6/1.0	1.6/1.0	1.5/0.9
	S	21.5	2.8/1.3	2.4/1.4	2.4/1.3
	M	68.0	5.6/2.3	4.5/2.2	4.4/2.1
	L	86.9	7.4/3.0	5.9/2.7	5.8/2.7
	X	194.9	15.2/5.3	10.7/4.7	9.1/4.0
YOLOv12	N	6.5	1.7/1.1	1.7/1.0	1.7/1.1
	S	21.4	2.9/1.5	2.5/1.5	2.5/1.4
	M	67.5	5.8/1.5	4.6/2.4	4.4/2.2
	L	88.9	7.9/3.3	6.2/3.1	6.0/3.0
	X	199.0	15.6/5.6	11.0/5.2	9.5/4.4

4.4. Speed Comparison

Table 4 presents a comparative analysis of inference speed across different GPUs, evaluating YOLOv9 [58], YOLOv10 [53], YOLOv11 [28], and our YOLOv12 on RTX 3080, RTX A5000, and RTX A6000 with FP32 and

Table 5. **Diagnostic studies.** We only show the factor(s) to be diagnosed in each subtable to save space. The default parameters are (unless otherwise specified): training for 600 epochs from scratch, using YOLOv12-N model.

Method	**AP$^{val}_{50:95}$**	**Latency**
Linear$_{+\text{LN}}$	40.5	1.68
Linear$_{+\text{BN}}$	39.5	1.70
Conv$_{+\text{BN}}$	40.6	1.64
Conv$_{+\text{LN}}$	40.3	1.66

(a) Attention Implementation

Method	**AP$^{val}_{50:95}$**	**Latency**
N/A	38.3	1.60
~~S_1~~	40.1	1.63
~~S_4~~	39.8	1.71
Ours	40.6	1.64

(b) Hierarchical Design

Epochs	**AP^{val} (N)**	**AP^{val} (S)**
300	39.5	47.0
500	40.3	47.8
600	40.6	48.0
800	40.4	47.7

(c) Training Epoch

kernel	**AP$^{val}_{50:95}$**	**Latency**
3×3	40.4	1.60
5×5	40.4	1.61
7×7	40.6	1.64
9×9	40.7	1.79

(d) Position Perceiver

Pos.	**AP$^{val}_{50:95}$**	**Latency**
RPE	40.3	1.76
APE	40.5	1.69
N/A	40.6	1.64

(e) Position Embedding

Area	**AP$^{val}_{50:95}$**	**Latency**
✗	40.8	1.70
✓	40.6	1.64

(f) Area Attention

Ratio (L)	**AP$^{val}_{50:95}$**	**Latency**
1.2	53.8	6.77
2.0	53.6	6.75
4.0	53.1	6.68

(g) MLP Ratio

FA	**Latency (N)**	**Latency (S)**
✗	1.92	3.02
✓	1.64	2.61

(h) FlashAttention

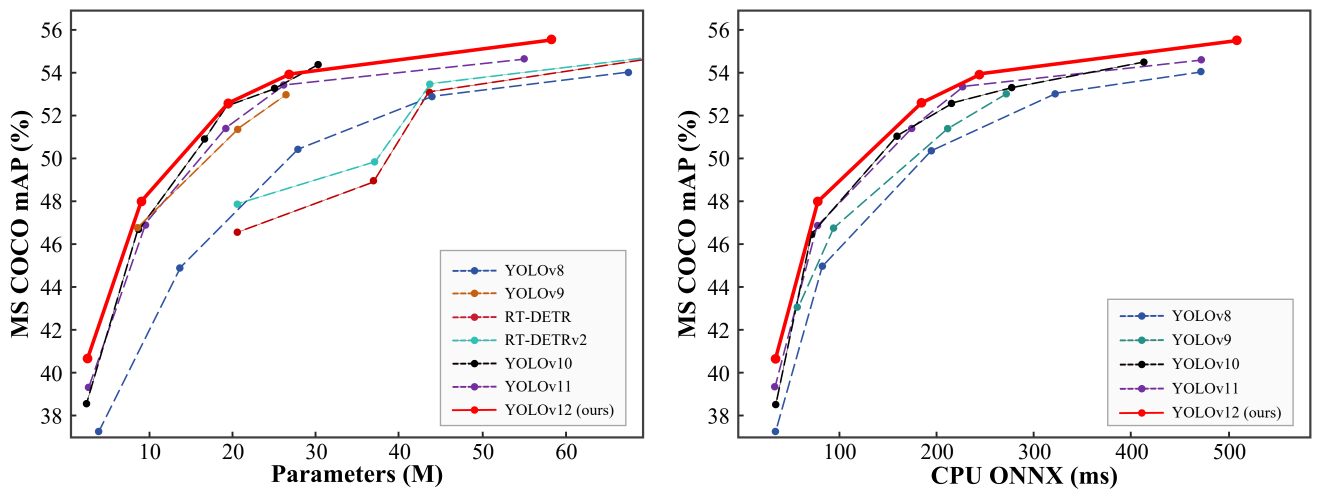


Figure 4. **Comparison with popular methods in terms of accuracy-parameters (left) and accuracy-latency trade-off on CPU (right).**

FP16 precision. To ensure consistency, all results are obtained on the same hardware and YOLOv9 [58] and YOLOv10 [53] are evaluated using the integrated codebase of ultralytics [28]. The results indicate that YOLOv12 achieves a significantly higher inference speed than YOLOv9 [58] while remaining on par with YOLOv10 [53] and YOLOv11 [28]. For example, on RTX 3080, YOLOv9 reports 2.4 ms (FP32) and 1.5 ms (FP16), while YOLOv12-N achieves 1.7 ms (FP32) and 1.1 ms (FP16). Similar trends hold across other configurations.

Figure 4 presents additional comparisons. The left subfigure illustrates the accuracy-parameter trade-off comparison with popular methods, where YOLOv12 establishes a dominant boundary beyond the counterparts, surpassing even YOLOv10, A YOLO version characterized by significantly fewer parameters, showcasing the efficacy of YOLOv12. We compare the inference latency of YOLOv12 with previous YOLO versions on a CPU in the right subfigure (All results were measured on an Intel Core i7-10700K @ 3.80GHz). As shown, YOLOv12 surpasses other competitors with a more advantageous boundary, highlighting its efficiency across diverse hardware platforms.

4.5. Diagnosis & Visualization

We diagnose the YOLOv12 designs in Tables 5a to 5h. Unless otherwise specified, we perform these diagnostics on YOLOv12-N, with a default training of 600 epochs from scratch.

• **Attention Implementation: Table 5a.** We examine two approaches to implementing attention. The convolution-based approach is faster than the linear-based approach due to the computational efficiency of convolution. Additionally, we explore two normalization methods (layer normalization (LN) and batch normalization (BN))

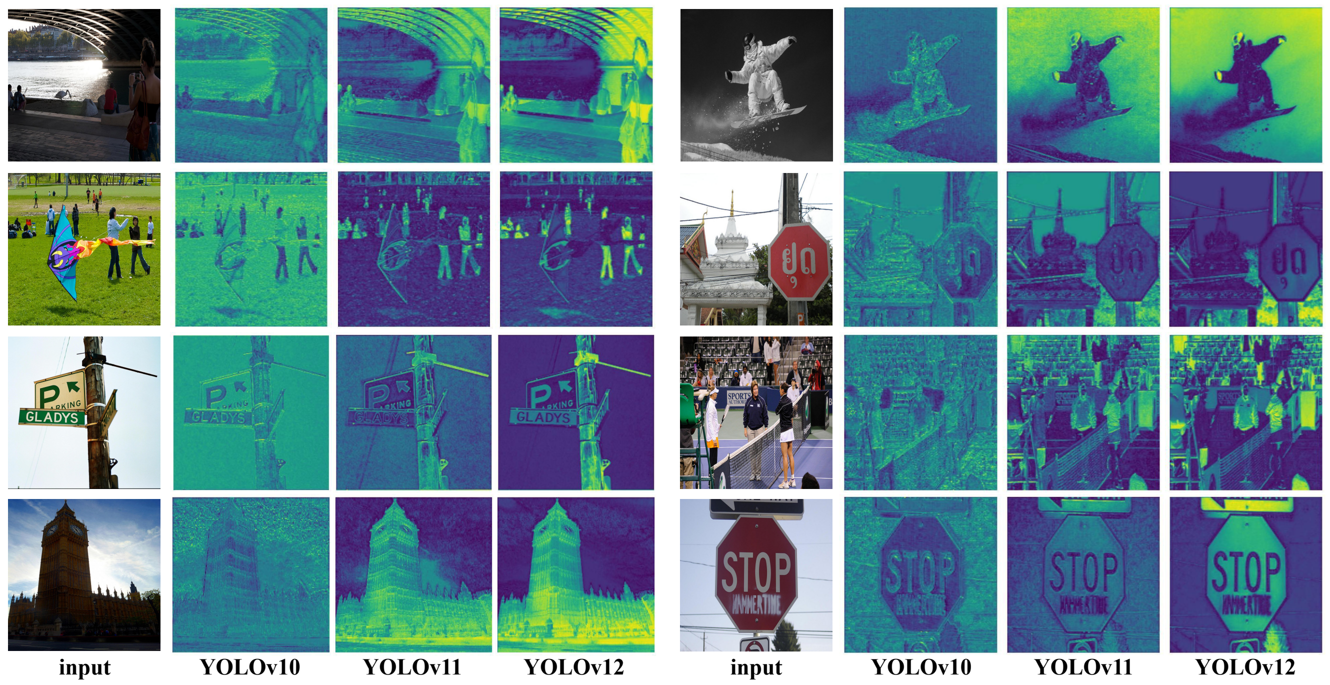


Figure 5. **Comparison of heat maps between YOLOv10 [53], YOLOv11 [28], and the proposed YOLOv12.** Compared to the advanced YOLOv10 and YOLOv11, YOLOv12 demonstrates a clearer perception of objects in the image. All the results are obtained using the X scale models. *Zoom in to compare the details.*

and find the result: although layer normalization is commonly used in attention mechanisms, it performs worse than batch normalization when used with convolution. Notably, this has already been used in the PSA module [53] and our finding is in line with its design.

- **Hierarchical Design: Table 5b.** Unlike other detection systems such as Mask R-CNN [1, 25], where the architectures of the plain vision transformers can produce strong results, YOLOv12 exhibits a different behavior. When using a plain vision transformer (N/A), the detector's performance drops significantly, achieving only 38.3% mAP. A more moderate adjustment, such as omitting the first (~~S_1~~) or fourth stage (~~S_4~~), while maintaining similar FLOPs by adjusting the feature dimensions, results in a slight performance degradation of 0.5% mAP and 0.8% mAP, respectively. Consistent with previous YOLO models, the hierarchical design remains the most effective, yielding 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 [24, 53, 58], YOLOv12 requires a more extended training period (about 600 epochs) to achieve peak performance, keeping the same configuration used in YOLOv11 [28].
- **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. We refer to this component as the Position Perceiver, as the smoothing effect of convolution preserves the original positions of image pixels, it helps the attention mechanism perceive positional information (This has already been used in the PSA module [53], but we expand the convolution kernel, achieving performance improvement without affecting speed). As shown in the table, increasing the convolution kernel size improves performance but gradually reduces speed. When the kernel size reaches 9×9, the slowdown becomes significant. Therefore, we set 7×7 as the default kernel size.
- **Position Embedding: Tables 5e.** We examine the impact of commonly used positional embeddings in most attention-based models (RPE: relative positional embedding; APE: absolute positional encoding) on performance. Interestingly, the best-performing configuration is achieved without any positional embedding, which brings cleaner architecture and faster inference latency.
- **Area Attention: Tables 5f.** In this table, we use the FlashAttention technique by default. This causes that while the area attention mechanism increases computational complexity (leading to performance gains), the resulting slowdown remains minimal. For further validation of area attention effectiveness, refer to Table 3.
- **MLP Ratio: Tables 5g.** In traditional vision trans-

Table 6. **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	40.6	56.7	43.8	20.2	45.2	58.4
YOLOv12-S	48.0	65.0	51.8	29.8	53.2	65.6
YOLOv12-M	52.5	69.6	57.1	35.7	58.2	68.8
YOLOv12-L	53.7	70.7	58.5	36.9	59.5	69.9
YOLOv12-X	55.2	72.0	60.2	39.6	60.7	70.9

formers, the MLP ratio within the attention module is generally set to 4.0. However, we observe different behavior with YOLOv12. In the table, varying the MLP ratio impacts the model size, so we adjust the feature dimensions to maintain overall model consistency. In particular, YOLOv12 achieves better performance with an MLP ratio of 1.2, diverging from conventional practices. This adjustment shifts the computational load more toward the attention mechanism, highlighting the importance of area attention.

• **FlashAttention: Tables 5h.** This table validates the role of FlashAttention in YOLOv12. It shows that FlashAttention accelerates YOLOv12-N by approximately 0.3ms and YOLOv12-S by around 0.4ms without other costs.

Visualization: Heat Map Comparison. Figure 5 compares the heat maps of YOLOv12 with state-of-the-art YOLOv10 [53] and YOLOv11 [28]. These heat maps, extracted from the third stage of the backbones of X-scale models, highlight the regions activated by the model, reflecting its object perception capability. As illustrated, compared to YOLOv10 and YOLOv11, YOLOv12 produces clearer object contours and more precise foreground activation, indicating improved perception. Our explanation is that this improvement comes from the area attention mechanism, which has a larger receptive field than convolutional networks and is therefore considered better at capturing the overall context, leading to more precise foreground activation. We believe that this characteristic gives YOLOv12 a performance advantage.

5. Conclusion

This study introduces YOLOv12, which successfully adopts an attention-centric design that traditionally is considered inefficient for real-time requirements, into the YOLO framework, achieving a state-of-the-art latency-accuracy trade-off. To enable efficient inference, we propose a novel network that leverages area attention to reduce computational complexity and residual efficient layer aggregation networks (R-ELAN) to enhance feature aggregation. Furthermore, we refine key components of the vanilla attention mechanism to better align with YOLO's real-time constraints while maintaining high-speed performance. As a result, YOLOv12 achieves state-of-the-art performance by effectively combining area attention, R-ELAN, and architectural optimizations, leading to significant improvements in both accuracy and efficiency. Comprehensive ablation studies further validate the effectiveness of these innovations. This study challenges the dominance of CNN-based designs in YOLO systems and advances the integration of attention mechanisms for real-time object detection, paving the way for a more efficient and powerful YOLO system.

Table 7. **Hyperparameters for fine-tuning the YOLOv12 family on COCO [36].**

Hyperparameters	**YOLOv12-N/S/M/L/X**
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

6. Limitations

YOLOv12 requires FlashAttention [13, 14], 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.*).

7. More Details

Fine-tuning Details. By default, all YOLOv12 models are trained using the SGD optimizer for 600 epochs. Following previous works [24, 53, 57, 58], 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 [3, 57], Mixup [71], and copy-paste augmentation [65], are applied to enhance training. Following YOLOv11 [28], we adopt the Albumentations library [6]. Detailed hyperparameters are presented in Table 7. All models are trained on 8× NVIDIA A6000 GPUs. Following established conventions [24, 28, 53, 58], 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. We recommend reviewing more details at the official code: https://github.com/sunsmarterjie/yolov12.

Result Details. We report more details of the results in Table 6 including $\text{AP}^{val}_{50:95}$, AP^{val}_{50}, AP^{val}_{75}, AP^{val}_{small}, AP^{val}_{medium}, AP^{val}_{large}.

References

[1] Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. Beit: Bert pre-training of image transformers. *arXiv preprint arXiv:2106.08254*, 2021. 6, 9

[2] 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

[3] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020. 1, 2, 6, 11

[4] 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. 1

[5] 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. 3

[6] 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. 11

[7] 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. 2

[8] 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. 1

[9] 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. 2

[10] 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. 4

[11] 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. 3, 4

[12] 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. 3

[13] Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*, 2023. 2, 3, 7, 11

[14] 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. 2, 3, 7, 11

[15] 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. 3

[16] 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. 2, 4

[17] 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. 1

[18] Alexey Dosovitskiy. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 6

[19] 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 Con-*

ference on Computer Vision and Pattern Recognition, pages 19358–19369, 2023. 3, 6

[20] 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. 1

[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. 1, 6

[22] 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. 1

[23] 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. 1

[24] Jocher Glenn. Yolov8. *https://github.com/ultralytics/ultralytics/tree/main*, 2023. 1, 2, 5, 6, 9, 11

[25] 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. 1, 6, 9

[26] Jonathan Ho, Nal Kalchbrenner, Dirk Weissenborn, and Tim Salimans. Axial attention in multidimensional transformers. *arXiv preprint arXiv:1912.12180*, 2019. 2

[27] 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. 2, 4

[28] Glenn Jocher. yolov11. *https://github.com/ultralytics*, 2024. 1, 2, 4, 5, 6, 7, 8, 9, 10, 11

[29] Glenn Jocher, K Nishimura, T Mineeva, and RJAM Vilariño. yolov5. *https://github.com/ultralytics/yolov5/tree*, 2, 2020. 1, 2, 6

[30] 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. 4

[31] 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. 3

[32] 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. 1, 2, 5, 6

[33] 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. 2

[34] 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. 1

[35] 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. 1

[36] 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. 6, 10

[37] Shilong Liu, Feng Li, Hao Zhang, Xiao Yang, Xianbiao Qi, Hang Su, Jun Zhu, and Lei Zhang. Dab-detr: Dynamic anchor boxes are better queries for detr. *arXiv preprint arXiv:2201.12329*, 2022. 2

[38] Yue Liu, Yunjie Tian, Yuzhong Zhao, Hongtian Yu, Lingxi Xie, Yaowei Wang, Qixiang Ye, Jianbin Jiao, and Yunfan Liu. Vmamba: Visual state space model. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. 2, 3

[39] 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. 2, 3, 4

[40] 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. 2, 6

[41] 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. 5, 6

[42] 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. 2

[43] 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. 1

[44] 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. 1

[45] J Redmon. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016. 1, 2, 6

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

[47] 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. 1, 2, 6

[48] 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. 1

[49] 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. 3, 4

[50] Yunjie Tian, Lingxi Xie, Jihao Qiu, Jianbin Jiao, Yaowei Wang, Qi Tian, and Qixiang Ye. Fast-itpn: Integrally pretrained transformer pyramid network with token migration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024. 1, 3

[51] 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. 6

[52] 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. 4

[53] 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. 1, 2, 5, 6, 7, 8, 9, 10, 11

[54] 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. 2, 5, 6

[55] 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. 2, 4

[56] 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. 2, 4

[57] 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. 1, 2, 4, 6, 11

[58] 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. 1, 2, 4, 5, 6, 7, 8, 9, 11

[59] 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. 1

[60] Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020. 3, 4

[61] 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. 2

[62] Enze Xie, Junsong Chen, Yuyang Zhao, Jincheng Yu, Ligeng Zhu, Yujun Lin, Zhekai Zhang, Muyang Li, Junyu Chen, Han Cai, et al. Sana 1.5: Efficient scaling of training-time and inference-time compute in linear diffusion transformer. *arXiv preprint arXiv:2501.18427*, 2025. 3

[63] 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*, pages 14138–14148, 2021. 4

[64] 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. 3

[65] Hongyi Zhang. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017. 11

[66] 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. 2, 5, 6

[67] 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*, pages 12993–13000, 2020. 1

[68] 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. 1

[69] 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. 1

[70] 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. 3

[71] 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. 2, 11