YOLO-ULM: Ultra-Lightweight Models for Real-Time Object Detection

Supplementary Material

1. Appendix

In this supplementary document, we first present the detailed hyperparameter configurations in Section 1.1. Section 1.2 provides a detailed performance comparison between YOLO-ULM-Turbo and YOLOv12-Turbo. Section 1.3 benchmarks the inference speeds of real-time detection models on different devices platforms.

1.1. Hyper-parameter

We follow the fine-tuning details of [2, 4, 5, 7], including training settings, loss parameters, and augmentation parameters. YOLO-ULM is trained from scratch using the SGD optimizer for 600 epochs. The SGD momentum and weight decay are set to 0.937 and 5×10^{-4}, respectively. The initial learning rate is 1×10^{-2}, linearly decreasing to 1×10^{-4}. For data augmentation, we employ Mosaic [1, 6], Mixup [8], and copy-paste augmentation [9], as in [4]. Specific details are shown in Tab. 1. All models are trained on 8 NVIDIA RTX 4090 GPUs, and following [2–4].

Table 1. Hyperparameters of YOLO-ULM on COCO dataset used in our experimental evaluation on RTX 4090 GPUs and AMD 7502 CPUs.

Hyperparameters	YOLO-ULM-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

1.2. Performance Comparison between YOLO-ULM-Turbo and YOLOv12-Turbo.

As shown in Tab. 3, YOLO-ULM-Turbo achieves faster speeds at various model scales than YOLOv12-Turbo.

Table 2. Comparison of inference latency on RTX 4090 GPUs and AMD 7502 CPUs.

Model	Scale	FLOPs↓	#Param.↓	Latency(GPU)↓	Latency(CPU)↓
YOLOv8	N	8.7	3.2	0.6	35.4
	S	28.6	11.2	0.7	74.6
	M	78.9	25.9	1.1	170.5
	L	165.2	43.7	1.6	300.2
	X	257.8	68.2	2.3	465.8
YOLOv9	S	26.4	7.1	1.2	88.4
	M	76.3	20.0	1.4	188.3
	C	102.1	25.3	1.5	238.0
YOLOv10	N	6.7	2.3	0.8	36.8
	S	21.6	7.2	0.9	68.1
	M	59.1	15.4	1.2	152.7
	B	92.0	19.1	1.4	198.8
	L	120.3	24.4	1.8	246.2
	X	160.4	29.5	2.3	348.7
YOLOv11	N	6.5	2.6	0.7	37.0
	S	21.5	9.4	0.8	68.6
	M	68.0	20.1	1.2	172.7
	L	86.9	25.3	1.6	205.2
	X	194.9	56.9	2.3	425.5
YOLOv12	N	6.5	2.6	1.0	44.2
	S	21.4	9.3	1.2	81.7
	M	67.5	20.2	1.6	185.3
	L	88.9	26.4	2.6	267.8
	X	199.0	59.1	3.6	508.2
YOLOv13	N	6.4	2.5	1.5	44.7
	S	20.8	9.0	1.6	75.9
	L	88.4	27.6	2.9	220.2
	X	199.2	64.0	3.9	412.0
YOLO-ULM	N	6.4	2.1	0.7	36.8
	S	21.2	7.4	0.9	63.8
	M	70.2	16.0	1.2	152.6
	L	95.1	22.8	1.7	201.0
	X	213.1	51.0	2.5	405.1
YOLO-ULM-Turbo	N	5.8	2.1	0.7	32.7
	S	18.8	7.2	0.9	55.5
	M	59.3	15.1	1.1	128.3
	L	81.6	21.6	1.6	167.8
	X	182.6	48.1	2.3	336.5

At 1.48ms / image latency, YOLO-ULM-Turbo-N attains 40.7% mAP with only 5.8G FLOPs and 2.1M parameters. YOLO-ULM-Turbo demonstrates superior inference speeds across all variants (N, S, M, L, X) compared to YOLOv12-Turbo variants, while achieving FLOPs reductions of 3.3%, 3.1%, 0.8%, 1.0%, and 3.4%, with parameter reductions reaching 16%, 20.9%, 23.0%, 18.5%, and 19.1% respectively.

1.3. Inference Speed on Different Devices

Tab. 2 presents a comparison of inference speed among different device platforms. We evaluate the FP16 precision of YOLOv8-v13, YOLO-ULM, and YOLO-ULM-Turbo on RTX 4090 GPUs. The YOLOv8-v13 architectures were benchmarked under Ultralytics unified framework (GitHub codebase) [3]. Both YOLO-ULM and YOLO-ULM-Turbo demonstrate higher inference speeds on RTX 4090 GPUs and CPU than baseline YOLOv12, while maintaining consistency with YOLOv10 and YOLOv11. For instance, YOLO-ULM-N and YOLO-ULM-Turbo-N (both latency are 0.7 ms / image) achieve 30% lower inference latency than YOLOv12-N with 1.0 ms / image latency. YOLO-ULM-S and YOLO-ULM-Murbo-S

Table 3. Comparison between YOLO-ULM-Turbo and YOLOv12-Turbo. All training settings are consistent with YOLO-ULM.

Model	FLOPs(G)↓	#Param.(M)↓	$AP^{val}_{50:95}$(%)↑	AP^{val}_{50}(%)↑	AP^{val}_{75}(%)↑	Latency(ms)↓
YOLOv12-Turbo-N	6.0	2.5	40.4	55.9	43.5	1.6
YOLO-ULM-Turbo-N	**5.8**	**2.1**	**40.7**	**56.7**	**44.0**	**1.48**
YOLOv12-Turbo-S	19.4	9.1	47.6	64.5	51.5	2.42
YOLO-ULM-Turbo-S	**18.8**	**7.2**	**47.7**	**64.7**	**51.8**	**2.39**
YOLOv12-Turbo-M	59.8	19.6	52.5	69.9	57.1	4.27
YOLO-ULM-Turbo-M	**59.3**	**15.1**	**52.4**	**69.8**	**57.2**	**4.23**
YOLOv12-Turbo-L	82.4	26.5	53.8	71.0	58.6	5.83
YOLO-ULM-Turbo-L	**81.6**	**21.6**	**53.8**	**71.1**	**58.8**	**5.75**
YOLOv12-Turbo-X	184.6	59.3	55.4	72.5	60.3	10.38
YOLO-ULM-Turbo-X	**178.4**	**48.0**	**55.3**	**72.6**	**60.4**	**10.21**

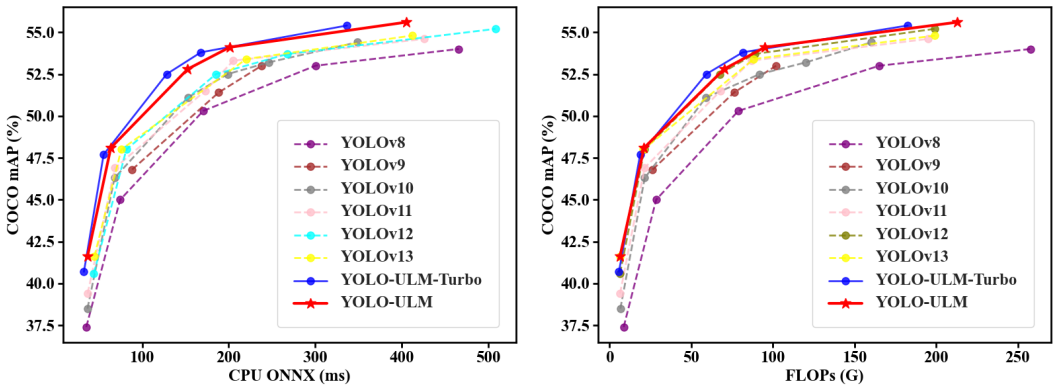

Figure 1. Comparison with popular methods in terms of accuracy-latency trade-off on CPU (left) and accuracy-FLOPs (right).

achieve a 43.8% reduction in inference latency relative to YOLOv13-S. YOLOv12-X shows 3.6 ms / image, while YOLO-ULM-X and YOLO-ULM-Turbo-X reach 2.5 ms / image and 2.3 ms / image, respectively. Consistent improvements persist across S, M, and L variants. Tab. 2 also compares inference speeds on the AMD EPYC 7502 32-Core Processor CPU. We evaluate the performance of our models and other competitors using OpenCV Deep Neural Network (DNN). YOLO-ULM-Turbo achieves state-of-the-art latency reduction across model scales, with its YOLO-ULM-Turbo-L variant demonstrating 37.3% lower inference time (167.8 ms / image) than YOLOv12 (267.8 ms / image) on CPU. Other variants also demonstrate similar reductions in inference latency.

As shown in Fig. 1 (left), the trade-off between accuracy and latency on CPU is presented in the form of a line graph. Meanwhile, FLOPs of different variants were also compared in Fig. 1 (right). YOLO-ULM and YOLO-ULM-Turbo also show state-of-the-art trade-offs in terms of performance and efficiency.

References

[1] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection, 2020. 1

[2] Jocher Glenn. Yolov8. https://github.com/ultralytics/ultralytics/tree/main, 2023. 1

[3] Glenn Jocher. Yolov11. https://github.com/ultralytics, 2024. 1

[4] Yunjie Tian, Qixiang Ye, and David Doermann. Yolov12: Attention-centric real-time object detectors, 2025. 1

[5] Ao Wang, Hui Chen, Lihao Liu, Kai Chen, Zijia Lin, Jungong Han, and Guiguang Ding. Yolov10: Real-time end-to-end object detection, 2024. 1

[6] 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 (CVPR)*, pages 7464–7475, 2023. 1

[7] Chien-Yao Wang, I-Hau Yeh, and Hong-Yuan Mark Liao. Yolov9: Learning what you want to learn using programmable gradient information, 2024. 1

[8] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization, 2018. 1

[9] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection, 2021. 1