Appendix of "DETRs Beat YOLOs on Real-time Object Detection"

1. Experimental Settings

Dataset and metric. We conduct experiments on COCO [8] and Objects365 [11], where RT-DETR is trained on COCO `train2017` and validated on COCO `val2017` dataset. We report the standard COCO metrics, including AP (averaged over uniformly sampled IoU thresholds ranging from 0.50-0.95 with a step size of 0.05), AP$_{50}$, AP$_{75}$, as well as AP at different scales: AP$_S$, AP$_M$, AP$_L$.

Implementation details. We use ResNet [5, 6] pretrained on ImageNet [2, 10] as the backbone and the learning rate strategy of the backbone follows [1]. In the hybrid encoder, AIFI consists of 1 Transformer layer and the fusion block in CCFF consists of 3 *RepBlock*s. We leverage the uncertainty-minimal query selection to select top 300 encoder features to initialize object queries of the decoder. The training strategy and hyperparameters of the decoder almost follow DINO [13]. We train RT-DETR with the AdamW [9] optimizer using four NVIDIA Tesla V100 GPUs with a batch size of 16 and apply the exponential moving average (EMA) with $ema\_decay = 0.9999$. The $1\times$ configuration means that the total epoch is 12, and the final reported results adopt the $6\times$ configuration. The data augmentation applied during training includes *random_{color distort, expand, crop, flip, resize}* operations, following [12]. The main hyperparameters of RT-DETR are listed in Table A (refer to RT-DETR-R50 for detailed configuration).

Item	Value
optimizer	AdamW
base learning rate	1e-4
learning rate of backbone	1e-5
freezing BN	True
linear warm-up start factor	0.001
linear warm-up steps	2000
weight decay	0.0001
clip gradient norm	0.1
ema decay	0.9999
number of AIFI layers	1
number of *RepBlock*s	3
embedding dim	256
feedforward dim	1024
nheads	8
number of feature scales	3
number of decoder layers	6
number of queries	300
decoder npoints	4
class cost weight	2.0
α in class cost	0.25
γ in class cost	2.0
bbox cost weight	5.0
GIoU cost weight	2.0
class loss weight	1.0
α in class loss	0.75
γ in class loss	2.0
bbox loss weight	5.0
GIoU loss weight	2.0
denoising number	200
label noise ratio	0.5
box noise scale	1.0

Table A. Main hyperparameters of RT-DETR.

2. Comparison with Lighter YOLO Detectors

To adapt to diverse real-time detection scenarios, we develop lighter scaled RT-DETRs by scaling the encoder and decoder with ResNet50/34/18 [5]. Specifically, we halve the number of channels in the *RepBlock*, while leaving other components unchanged, and obtain a set of RT-DETRs by adjusting the number of decoder layers during inference. We compare the scaled RT-DETRs with the S and M models of YOLO detectors in Table B. The number of decoder layers used by scaled RT-DETR-R50/34/18 during training is 6/4/3 respectively, and Dec^k indicates that k decoder layers are used during inference. Our RT-DETR-R50-Dec$^{2-5}$ outperform all M models of YOLO detectors in both speed and accuracy, while RT-DETR-R18-Dec^2 outperforms all S models. Compared to the state-of-the-art M model (YOLOv8-M [4]), RT-DETR-R50-Dec^5 improves accuracy by 0.9% AP and increases FPS by 36%. Compared to the state-of-the-art S model (YOLOv6-S [7]), RT-DETR-R18-Dec^2 improves accuracy by 0.5% AP and increases FPS by 18%. This shows that RT-DETR is able to outperform the lighter YOLO detectors in both speed and accuracy by simple scaling.

3. Large-scale Pre-training for RT-DETR

We pre-train RT-DETR on the larger Objects365[11] dataset and then fine-tune it on COCO to achieve higher performance. As shown in Table C, we perform experiments on RT-DETR-R18/50/101 respectively. All three models are pre-trained on Objects365 for 12 epochs, and RT-DETR-R18 is fine-tuned on COCO for 60 epochs, while RT-DETR-R50 and RT-DETR-R101 are fine-tuned for 24 epochs. Experimental results show that RT-DETR-R18/50/101 is improved

Model	**#Epochs**	**#Params (M)**	**GFLOPs**	**FPS$_{bs=1}$**	**AP^{val}**	**AP$^{val}_{50}$**	**AP$^{val}_{75}$**	**AP$^{val}_{S}$**	**AP$^{val}_{M}$**	**AP$^{val}_{L}$**
S and M models of YOLO Detectors										
YOLOv5-S[3]	300	7.2	16.5	74	37.4	56.8	-	-	-	-
YOLOv5-M[3]	300	21.2	49.0	64	45.4	64.1	-	-	-	-
PPYOLOE-S[12]	300	7.9	17.4	218	43.0	59.6	47.1	25.9	47.4	58.6
PPYOLOE-M[12]	300	23.4	49.9	131	48.9	65.8	53.7	30.8	53.4	65.3
YOLOv6-S[7]	300	18.5	45.3	201	45.0	61.8	48.9	24.3	50.2	62.7
YOLOv6-M[7]	300	34.9	85.8	121	50.0	66.9	54.6	30.6	55.4	67.3
YOLOv8-S[4]	-	11.2	28.6	136	44.9	61.8	48.6	25.7	49.9	61.0
YOLOv8-M[4]	-	25.9	78.9	97	50.2	67.2	54.6	32.0	55.7	66.4
Scaled RT-DETRs										
Scaled RT-DETR-R50-Dec^2	72	36†	98.4	**154**	**50.3**	**68.4**	54.5	**32.2**	55.2	**67.5**
Scaled RT-DETR-R50-Dec^3	72	36†	100.1	**145**	**51.3**	**69.6**	**55.4**	**33.6**	**56.1**	**68.6**
Scaled RT-DETR-R50-Dec^4	72	36†	101.8	**137**	**51.8**	**70.0**	**55.9**	**33.7**	**56.4**	**69.4**
Scaled RT-DETR-R50-Dec^5	72	36†	103.5	**132**	**52.1**	**70.5**	**56.2**	**34.3**	**56.9**	**69.9**
Scaled RT-DETR-R50-Dec^6	72	36	105.2	125	**52.2**	**70.6**	**56.4**	**34.4**	**57.0**	**70.0**
Scaled RT-DETR-R34-Dec^2	72	31†	89.3	185	**47.4**	**64.7**	**51.3**	**28.9**	**51.0**	**64.2**
Scaled RT-DETR-R34-Dec^3	72	31†	91.0	172	**48.5**	**66.2**	**52.3**	**30.2**	**51.9**	**66.2**
Scaled RT-DETR-R34-Dec^4	72	31	92.7	161	**48.9**	**66.8**	**52.9**	**30.6**	**52.4**	**66.3**
Scaled RT-DETR-R18-Dec^2	72	20†	59.0	**238**	**45.5**	**62.5**	**49.4**	**27.8**	48.7	61.7
Scaled RT-DETR-R18-Dec^3	72	20	60.7	217	**46.5**	**63.8**	**50.4**	**28.4**	49.8	**63.0**

Table B. Comparison with S and M models of YOLO detectors. The FPS of YOLO detectors are reported on T4 GPU with TensorRT FP16 using official pre-trained models according to the proposed end-to-end speed benchmark. † denotes the number of parameters during the training, not inference.

Model	**#Epochs**	**#Params (M)**	**GFLOPs**	**FPS$_{bs=1}$**	**AP^{val}**	**AP$^{val}_{50}$**	**AP$^{val}_{75}$**	**AP$^{val}_{S}$**	**AP$^{val}_{M}$**	**AP$^{val}_{L}$**
RT-DETR-R18	60	20	61	217	49.2 (↑ **2.7**)	66.6	53.5	33.2	52.3	64.8
RT-DETR-R50	24	42	136	108	55.3 (↑ **2.2**)	73.4	60.1	37.9	59.9	71.8
RT-DETR-R101	24	76	259	74	56.2 (↑ **1.9**)	74.6	61.3	38.3	60.5	73.5

Table C. Fine-tuning results on COCO `val2017` with pre-training on Objects365.

by 2.7%/2.2%/1.9% AP on COCO `val2017`. The surprising improvement further demonstrates the potential of RT-DETR and provides the strongest real-time object detector for various real-time scenarios in the industry.

4. Visualization of Predictions with Different Post-processing Thresholds

To intuitively demonstrate the impact of post-processing on the detector, we visualize the predictions produced by YOLOv8 [4] and RT-DETR using different post-processing thresholds, as shown in Figure A and Figure B, respectively. We show the predictions for two randomly selected samples from COCO `val2017` by setting different NMS thresholds for YOLOv8-L and score thresholds for RT-DETR-R50.

There are two NMS thresholds: confidence threshold and IoU threshold, both of which affect the detection results. The higher the confidence threshold, the more prediction boxes are filtered out and the number of false negatives increases. However, using a lower confidence threshold, *e.g.*, 0.001, results in a large number of redundant boxes and increases the number of false positives. The higher the IoU threshold, the fewer overlapping boxes are filtered out in each round of screening, and the number of false positives increases (the position marked by the red circle in Figure A). Nevertheless, adopting a lower IoU threshold will result in true positives being deleted if there are overlapping or mutually occluding objects in the input. The confidence threshold is relatively straightforward to process predicted boxes and therefore easy to set, whereas the IoU threshold is difficult to set accurately. Considering that different scenarios place different emphasis on recall and accuracy, *e.g.*, the general detection scenario requires the lower confidence threshold and the higher IoU threshold to increase the recall, while the dedicated detection scenario requires the higher confidence threshold and the lower IoU threshold to increase the accuracy, it is neces-

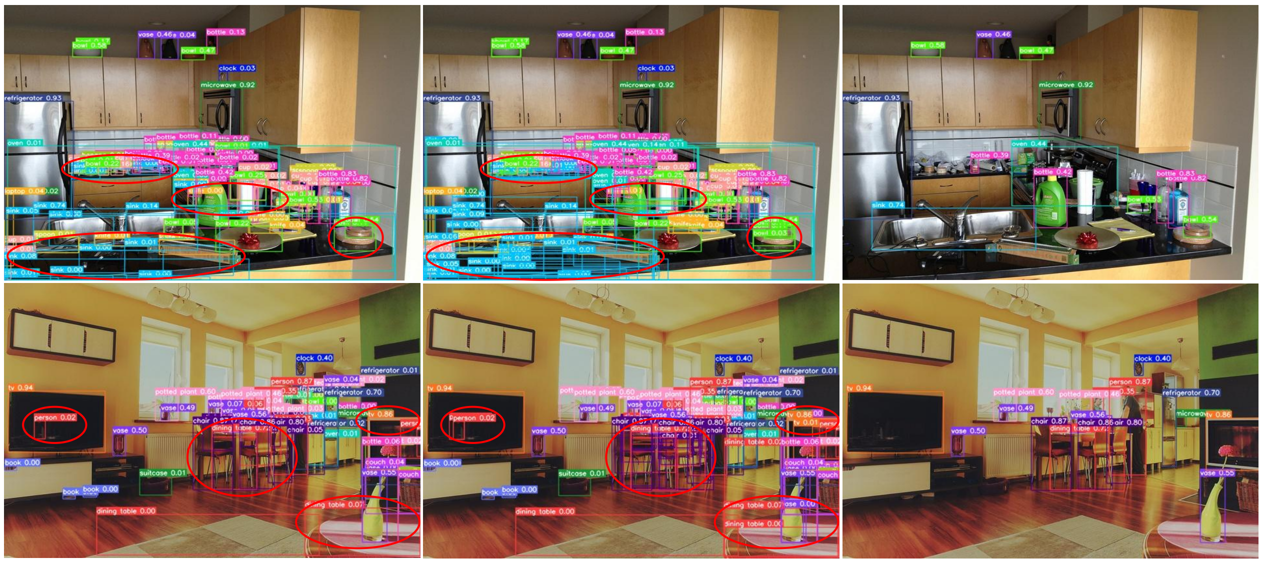

Figure A. Visualization of YOLOv8-L [4] predictions with different NMS thresholds.

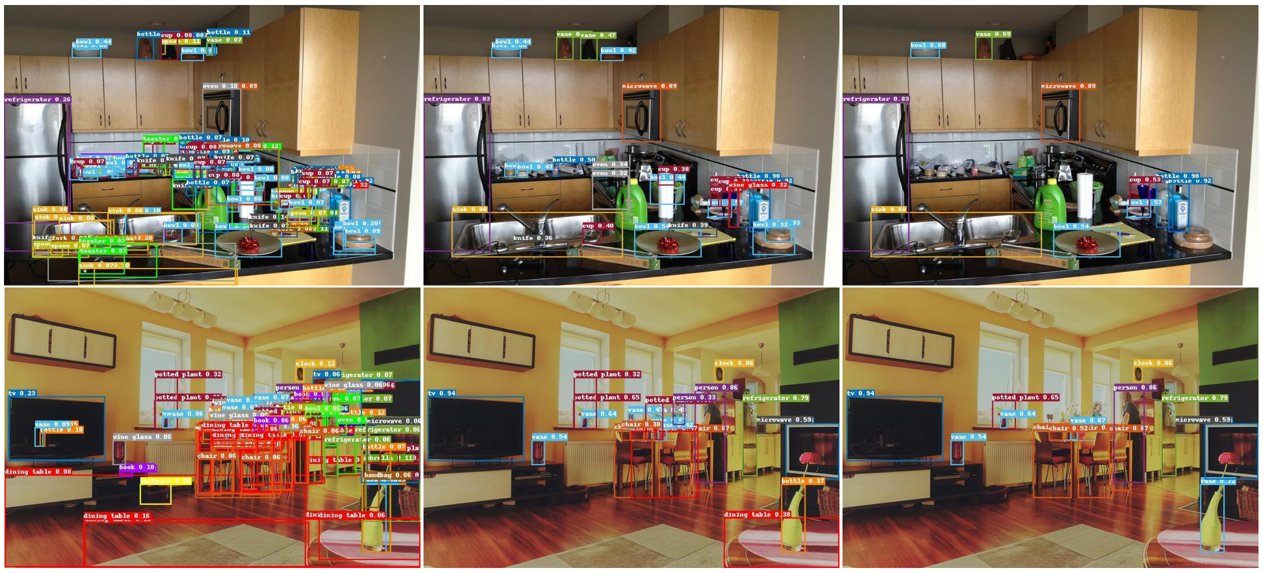

Figure B. Visualization of RT-DETR-R50 predictions with different score thresholds.

sary to carefully select the appropriate NMS thresholds for different scenarios.

RT-DETR utilizes bipartite matching to predict the one-to-one object set, eliminating the need for suppressing overlapping boxes. Instead, it directly filters out low-confidence boxes with a score threshold. Similar to the confidence threshold used in NMS, the score threshold can be adjusted in different scenarios based on the specific emphasis to achieve optimal detection performance. Thus, setting the post-processing threshold in RT-DETR is straightforward and does not affect the inference speed, enhancing the adaptability of real-time detectors across various scenarios.

5. Visualization of RT-DETR Predictions

We select several samples from the COCO `val2017` to showcase the detection performance of RT-DETR in complex scenarios and challenging conditions (refer to Figure C

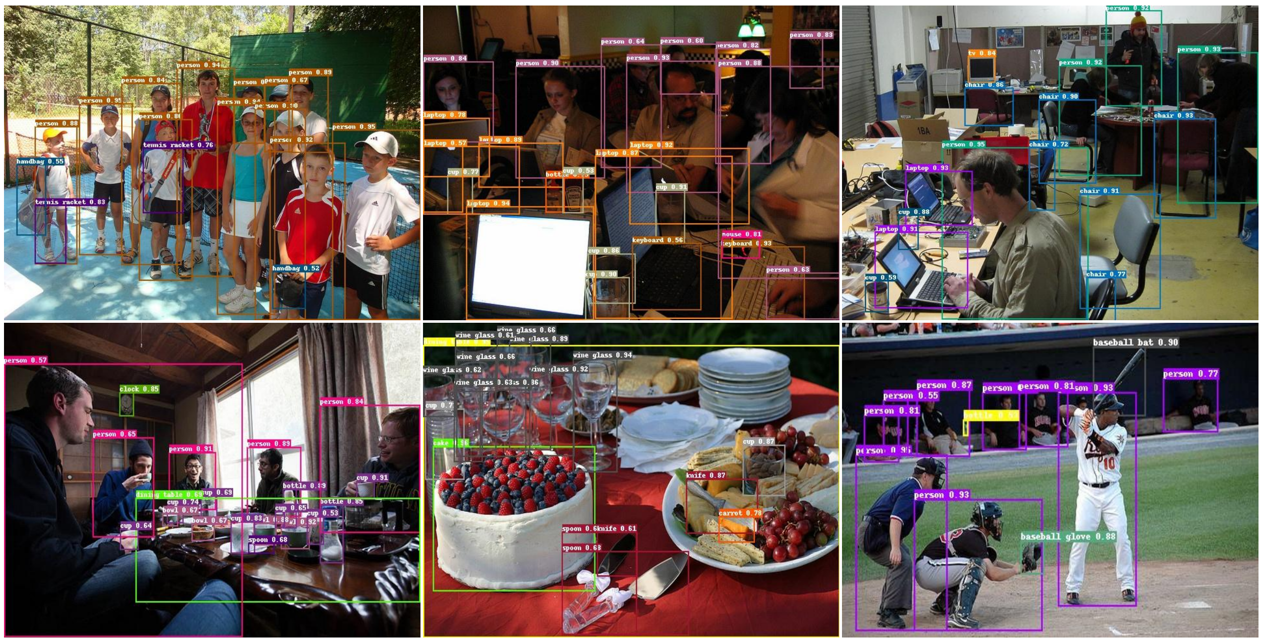

Figure C. Visualization of RT-DETR-R101 predictions in complex scenarios (score threshold=0.5).

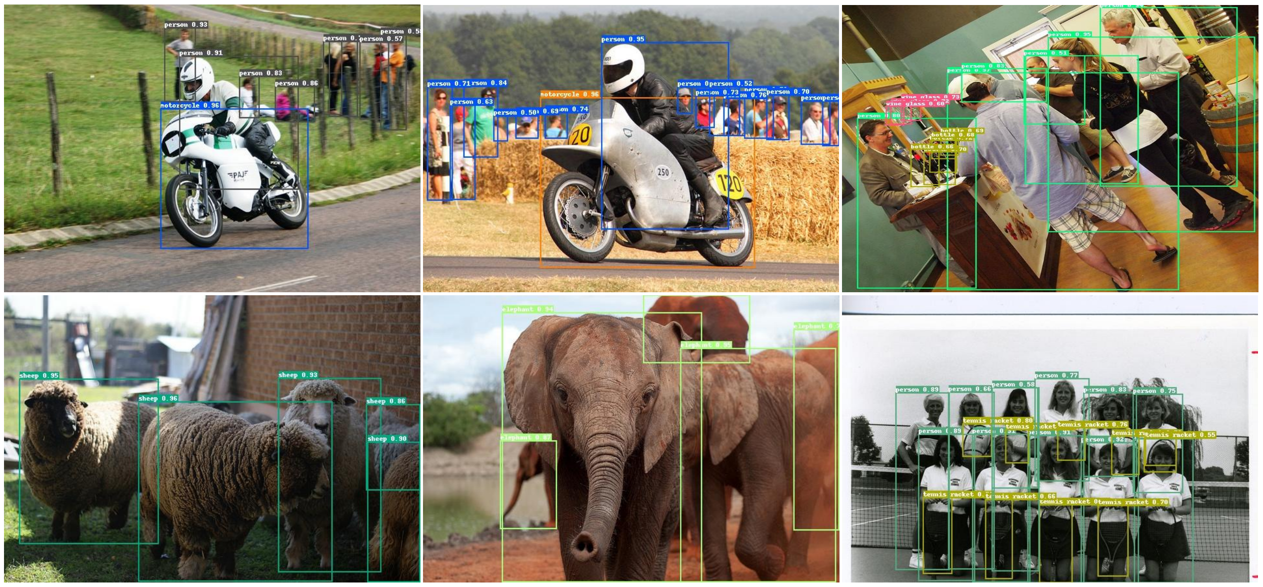

Figure D. Visualization of RT-DETR-R101 predictions under difficult conditions, including motion blur, rotation, and occlusion (score threshold=0.5).

and Figure D). In complex scenarios, RT-DETR demonstrates its capability to detect diverse objects, even when they are small or densely packed, *e.g.*, cups, wine glasses, and individuals. Moreover, RT-DETR successfully detects objects under various difficult conditions, including motion blur, rotation, and occlusion. These predictions substantiate the excellent detection performance of RT-DETR.

References

[1] 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. 1

[2] Cheng Cui, Ruoyu Guo, Yuning Du, Dongliang He, Fu Li, Zewu Wu, Qiwen Liu, Shilei Wen, Jizhou Huang, Xiaoguang Hu, Dianhai Yu, Errui Ding, and Yanjun Ma. Beyond self-

supervision: A simple yet effective network distillation alternative to improve backbones. *CoRR*, abs/2103.05959, 2021. 1

[3] Jocher Glenn. Yolov5 release v7.0. https://github.com/ultralytics/yolov5/tree/v7.0, 2022. 2

[4] Jocher Glenn. Yolov8. https://github.com/ultralytics/ultralytics/tree/main, 2023. 1, 2, 3

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016. 1

[6] Tong He, Zhi Zhang, Hang Zhang, Zhongyue Zhang, Junyuan Xie, and Mu Li. Bag of tricks for image classification with convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 558–567, 2019. 1

[7] 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

[8] 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 *European Conference on Computer Vision*, pages 740–755. Springer, 2014. 1

[9] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2018. 1

[10] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115:211–252, 2015. 1

[11] Shuai Shao, Zeming Li, Tianyuan Zhang, Chao Peng, Gang Yu, Xiangyu Zhang, Jing Li, and Jian Sun. Objects365: A large-scale, high-quality dataset for object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8430–8439, 2019. 1

[12] Shangliang Xu, Xinxin Wang, Wenyu Lv, Qinyao Chang, Cheng Cui, Kaipeng Deng, Guanzhong Wang, Qingqing Dang, Shengyu Wei, Yuning Du, et al. Pp-yoloe: An evolved version of yolo. *arXiv preprint arXiv:2203.16250*, 2022. 1, 2

[13] Hao Zhang, Feng Li, Shilong Liu, Lei Zhang, Hang Su, Jun Zhu, Lionel Ni, and Heung-Yeung Shum. Dino: Detr with improved denoising anchor boxes for end-to-end object detection. In *International Conference on Learning Representations*, 2022. 1