LW-DETR: A Transformer Replacement to YOLO for Real-Time Detection

Qiang Chen[1*], Xiangbo Su[1*], Xinyu Zhang[2,1*], Jian Wang[1], Jiahui Chen[3,1], Yunpeng Shen[1], Chuchu Han[1], Ziliang Chen[1], Weixiang Xu[1], Fanrong Li[4], Shan Zhang[5], Kun Yao[1], Errui Ding[1], Gang Zhang[1], and Jingdong Wang[1†]

[1] Baidu Inc., China
[2] The University of Adelaide, Australia
[3] Beihang University, China
[4] Institute of Automation, Chinese Academy of Sciences, China
[5] Australian National University, Australia
{chenqiang13,suxiangbo,zhangxinyu14,wangjingdong}@baidu.com

Abstract. In this paper, we present a light-weight detection transformer, LW-DETR, which outperforms YOLOs for real-time object detection. The architecture is a simple stack of a ViT encoder, a projector, and a shallow DETR decoder. Our approach leverages recent advanced techniques, such as training-effective techniques, e.g., improved loss and pretraining, and interleaved window and global attentions for reducing the ViT encoder complexity. We improve the ViT encoder by aggregating multi-level feature maps, and the intermediate and final feature maps in the ViT encoder, forming richer feature maps, and introduce window-major feature map organization for improving the efficiency of interleaved attention computation. Experimental results demonstrate that the proposed approach is superior over existing real-time detectors, e.g., YOLO and its variants, on COCO and other benchmark datasets. Code and models are available at https://github.com/Atten4Vis/LW-DETR.

Keywords: Object Detection · Real-Time · Detection Transformer

1 Introduction

Real-time object detection is an important problem in visual recognition and has wide real-world applications. The current dominant solutions are based on convolutional networks, such as the YOLO series [1, 2, 20, 29, 32, 33, 46, 59, 65]. Recently, transformer methods, e.g., detection transformer (DETR) [4], have witnessed significant progress [7, 19, 41, 47, 63, 67, 74, 75]. Unfortunately, DETR for real-time detection remains not fully explored, and it is unclear if the performance is comparable to the state-of-the-art convolutional methods.

In this paper, we build a light-weight DETR approach for real-time object detection. The architecture is very simple: a plain ViT encoder [17] and a DETR

* Equal contribution. †Corresponding author.

LW-DETR：替代YOLO的实时检测Transformer方案

强陈[1⋆]，苏相波[1⋆]，张新宇[2,1⋆]，王建[1]，陈佳慧[3,1]，沈云鹏[1]，韩楚楚[1]，陈子良[1]，徐伟翔[1]，李凡荣[4]，张珊[5]，姚坤[1]，丁二锐[1]，张刚[1]，王京东[1†]

1 百度公司，中国 [2] 阿德莱德大学，澳大利亚 [3] 北京航空航天大学，中国 [4] 中国科学院自动化研究所，中国 [5] 澳大利亚国立大学，澳大利亚 {chenqiang13,suxiangbo,zhangxinyu14,wangjingdong}@baidu.com

摘要。本文提出了一种轻量级检测变换器LW-DETR，在实时目标检测任务中性能超越YOLO系列。该架构由ViT编码器、投影器和浅层DETR解码器简单堆叠而成。我们的方法融合了多项前沿技术：采用提升训练效率的策略（如改进损失函数与预训练方法），通过交错局部窗口与全局注意力机制降低ViT编码器复杂度。我们通过聚合ViT编码器中多层级特征图及其中间与最终特征图来构建更丰富的特征表示，并创新性地采用窗口主导的特征图组织方式以提升交错注意力计算效率。实验结果表明，在COCO等基准数据集上，该方法优于YOLO系列等现有实时检测器。代码与模型已开源：https://github.com/Atten4Vis/LW-DETR。

关键词：目标检测 · 实时 · 检测变换器 {v*}

1 引言

实时目标检测是视觉识别中的一个重要问题，具有广泛的实际应用。当前主流的解决方案基于卷积网络，如YOLO系列[1, 2, 20, 29, 32, 33, 46, 59, 65]。近年来，基于Transformer的方法，例如检测变换器（DETR）[4]，取得了显著进展[7, 19, 41, 47, 63, 67, 74, 75]。然而，针对实时检测的DETR仍未得到充分探索，其性能是否可与最先进的卷积方法相媲美尚不明确。

本文提出了一种轻量级DETR方法用于实时目标检测。该架构极为简洁：仅包含一个普通ViT编码器[17]与DETR解码器，其中{v*}保持原公式表示不变。

⋆ Equal contribution. †Corresponding author.

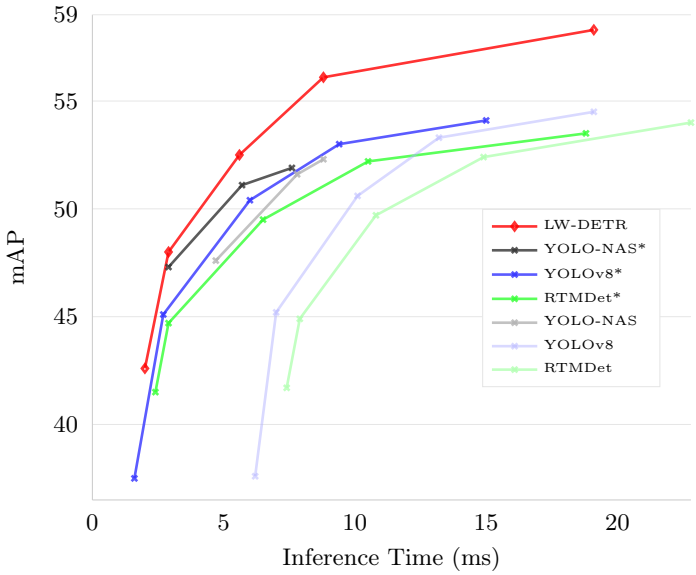


Fig. 1: Our approach outperforms previous SoTA real-time detectors. The x-axis corresponds to the inference time. The y-axis corresponds to the mAP score on COCO `val2017`. All the models are trained with pretraining on Objects365. The NMS post-processing times are included for other models and measured on the COCO `val2017` with the setting from the official implementation [1,29,46], and the well-tuned NMS postprocessing setting (labeled as "*").

decoder that are connected by a convolutional projector [29]. We propose to aggregate the multi-level feature maps, the intermediate and final feature maps in the encoder, forming stronger encoded feature maps. Our approach takes advantage of effective training techniques. For example, we use the deformable cross-attention forming the decoder [74], the IoU-aware classification loss [3], and the encoder-decoder pretraining strategy [7,67,71].

On the other hand, our approach exploits inference-efficient techniques. For example, we adopt interleaved window and global attentions [36,37], replacing some global attentions with window attentions in the plain ViT encoder to reduce the complexity. We use an efficient implementation for the interleaved attentions through a window-major feature map organization method, effectively reducing the costly memory permutation operations.

Figure 1 shows that the proposed simple baseline surprisingly outperforms the previous real-time detectors on COCO [39], *e.g.*, YOLO-NAS [1], YOLOv8 [29], and RTMDet [46]. These models are improved with pretraining on Objects365 [54], and the end-to-end time cost, including the NMS time, is measured using the setting from the official implementations[6].

[6] YOLO-NAS: https://github.com/Deci-AI/super-gradients
YOLOv8: https://docs.ultralytics.com
RTMDet: https://github.com/open-mmlab/mmyolo/tree/main/configs/rtmdet

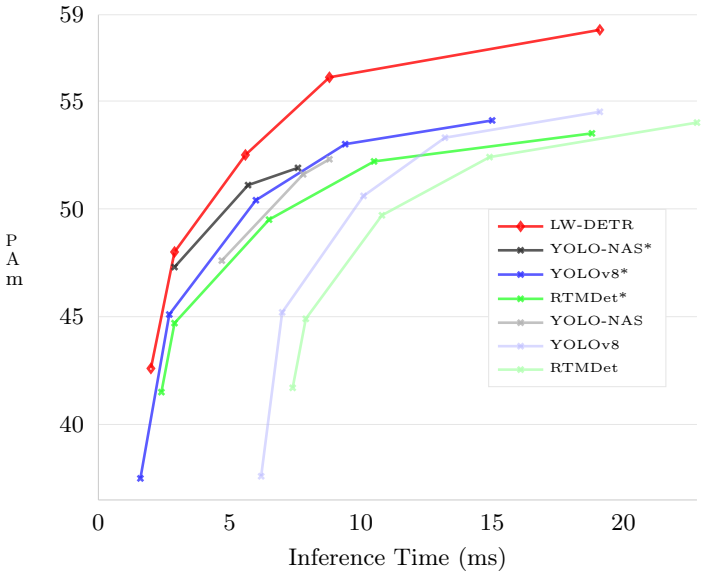


图1：我们的方法超越了之前的SoTA实时检测器。x轴对应推理时间，y轴对应COCO val2017数据集上的mAP分数。所有模型均在Objects365上进行预训练。其他模型的NMS后处理时间已包含在内，并在COCO val2017上按照官方实现[1,29,46]的设置进行测量，以及经过精心调优的NMS后处理设置（标记为“*”）。

由卷积投影器[29]连接的解码器。我们提出聚合编码器中多层级特征图、中间及最终特征图，以形成更强的编码特征图。我们的方法利用了多种高效训练技术，例如采用可变形交叉注意力构建解码器[74]、IoU感知分类损失[3]以及编码器-解码器预训练策略[7,67,71]。

另一方面，我们的方法采用了推理高效的技术。例如，我们采用交错窗口与全局注意力机制[36,37]，在朴素ViT编码器中用窗口注意力替代部分全局注意力以降低复杂度。通过窗口主导的特征图组织方法，我们实现了交错注意力机制的高效实现，有效减少了昂贵的内存置换操作。

图1显示，所提出的简单基线方法在COCO数据集[39]上出人意料地超越了以往的实时检测器，如YOLO-NAS[1]、YOLOv8[29]和RTMDet[46]。这些模型通过在Objects365[54]上进行预训练得到改进，端到端时间成本（包括NMS时间）的测量采用了官方实现[6]中的设置。

[6] YOLO-NAS: https://github.com/Deci-AI/super-gradients
YOLOv8: https://docs.ultralytics.com
RTMDet: https://github.com/open-mmlab/mmyolo/tree/main/configs/rtmdet

We conduct extensive experiments for the comparisons with existing real-time detection algorithms [1,12,29,46,58]. We further optimize the NMS setting and obtain improved performance for existing algorithms. The proposed baseline still outperforms these algorithms (labeled as "*" in Figure 1). In addition, we demonstrate the proposed approach with experimental results on more detection benchmarks.

The proposed baseline merely explores simple and easily implemented techniques and shows promising performance. We believe that our approach potentially benefits from other designs, such as efficient multi-scale feature fusion [34], token sparsification [36,72], distillation [5,5,9,64], as well as other training techniques, such as the techniques used in YOLO-NAS [1]. We also show that the proposed approach is applicable to the DETR approach with the convolutional encoder, such as ResNet-18 and ResNet-50 [23], and achieves good performance.

2 Related Work

Real-time object detection. Real-time object detection has wide real-world applications [18,25,30,31,49]. Existing state-of-the-art real-time detectors, such as YOLO-NAS [1], YOLOv8 [29], and RTMDet [46], have been largely improved compared with the first version of YOLO [50] through detection frameworks [20, 56], architecture designs [2,16,32,33,59,65], data augmentations [2,20,69], training techniques [1,20,29], and loss functions [38,68,73]. These detectors are based on convolutions. In this paper, we study transformer-based solutions to real-time detection that remains little explored.

ViT for object detection. Vision Transformer (ViT) [17,66] shows promising performance in image classification. Applying ViT to object detection usually exploits window attentions [17,43] or hierarchical architectures [21,43,62,70] to reduce the memory and computation cost. UViT [8] uses progressive window attention. ViTDet [36] implements the pre-trained plain ViT with interleaved window and global attentions [37]. Our approach follows ViTDet to use interleaved window and global attentions, and additionally uses window-major order feature map organization for reducing the memory permutation cost.

DETR and its variants. Detection Transformer (DETR) is an end-to-end detection method, with removing the necessity of many hand-crafted components, such as anchor generation [51] and non-maximum suppression (NMS) [24]. There are many followup methods for DETR improvement, such as architecture design [19,47,67,74], object query design [11,41,63], training techniques [6,7,27,35, 48,67,75], and loss function improvement [3,42]. Besides, various works have been done for reducing the computational complexity, by architecture design [34,40], computational optimization [74], pruning [53,72], and distillation [5,5,9,64]. The interest of this paper is to build a simple DETR baseline for real-time detection that are not explored by these methods.

Concurrent with our work, RT-DETR [45] also applies the DETR framework to construct real-time detectors with a focus on the CNN backbone forming the encoder. There are studies about relatively large models and a lack of tiny

我们进行了大量实验，与现有实时检测算法[1,12,29,46,58]进行对比。通过进一步优化NMS设置，提升了现有算法的性能。所提出的基线模型仍优于这些算法（在图1中以“*”标注）。此外，我们在更多检测基准测试上通过实验结果验证了所提方法的有效性。

所提出的基线仅探索了简单且易于实现的技术，并展现出令人期待的性能。我们相信，该方法有望从其他设计中获益，例如高效的多尺度特征融合[34]、令牌稀疏化[36,72]、蒸馏技术[5,5,9,64]，以及YOLO-NAS[1]等训练技巧。我们还表明，该方法适用于带有卷积编码器（如ResNet-18和ResNet-50[23]）的DETR方法，并取得了良好的性能表现。

2 相关工作

实时目标检测。实时目标检测在现实世界中有着广泛的应用[18,25,30,31,49]。现有的最先进实时检测器，如YOLO-NAS[1]、YOLOv8[29]和RTMDet[46]，通过检测框架[20,56]、架构设计[2,16,32,33,59,65]、数据增强[2,20,69]、训练技术[1,20,29]以及损失函数[38,68,73]等方面的改进，相较于最初的YOLO版本[50]已取得显著提升。这些检测器均基于卷积神经网络。本文中，我们研究基于transformer的实时检测解决方案，这一领域目前仍鲜有探索。

ViT在目标检测中的应用。视觉Transformer（ViT）[17, 66]在图像分类任务中展现出优异性能。将ViT应用于目标检测时，通常采用窗口注意力机制[17, 43]或分层架构[21, 43, 62, 70]以降低内存与计算开销。UViT[8]采用渐进式窗口注意力，ViTDet[36]则在预训练的基础ViT中交替使用窗口注意力与全局注意力[37]。我们的方法延续ViTDet的交替注意力机制，并额外引入窗口主导序的特征图组织方式以降低内存置换成本。

DETR及其变体。Detection Transformer（DETR）是一种端到端的检测方法，它消除了对许多手工设计组件的需求，例如锚框生成[51]和非极大值抑制（NMS）[24]。针对DETR的改进已有大量后续方法，包括架构设计[19,47,67,74]、目标查询设计[11,41,63]、训练技术[6,7,27,35,48,67,75]以及损失函数优化[3,42]。此外，为降低计算复杂度，研究者们通过架构设计[34,40]、计算优化[74]、剪枝[53,72]和蒸馏[5,5,9,64]等多种途径进行了探索。本文的关注点在于构建一个未被这些方法涉及的、面向实时检测的简单DETR基线。

与我们的工作同时，RT-DETR[45]同样应用了DETR框架来构建实时检测器，其重点在于构成编码器的CNN主干网络。现有研究多关注相对较大的模型，而微小模型的研究则较为缺乏。

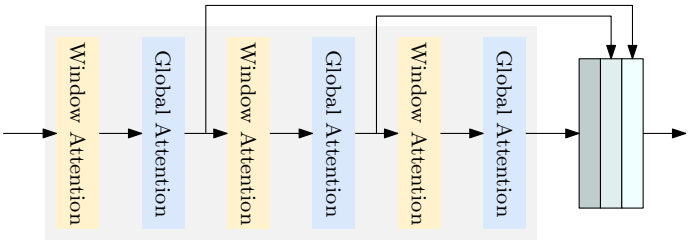


Fig. 2: An example of transformer encoder with multi-level feature map aggregation and interleaved window and global attentions. The FFN and LayerNorm layers are not depicted for clarification.

models. Our LW-DETR explores the feasibility of plain ViT backbones and DETR framework for real-time detection .

3 LW-DETR

3.1 Architecture

LW-DETR consists of a ViT encoder, a projector, and a DETR decoder.

Encoder. We adopt the ViT for the detection encoder. A plain ViT [17] consists of a patchification layer and transformer encoder layers. A transformer encoder layer in the initial ViT contains a global self-attention layer over all the tokens and an FFN layer. The global self-attention is computationally costly, and its time complexity is quadratic with respect to the number of tokens (patches). We implement some transformer encoder layers with window self-attention to reduce the computational complexity (detailed in Sec. 3.4). We propose to aggregate the multi-level feature maps, the intermediate and final feature maps in the encoder, forming stronger encoded feature maps. An example of the encoder is illustrated in Figure 2.

Decoder. The decoder is a stack of transformer decoder layers. Each layer consists of a self-attention, a cross-attention, and an FFN. We adopt deformable cross-attention [74] for computational efficiency. DETR and its variants usually adopt 6 decoder layers. In our implementation, we use 3 transformer decoder layers. This leads to a time reduction from 1.4 ms to 0.7 ms, which is significant compared to the time cost 1.3 ms of the remaining part for the tiny version in our approach.

We adopt a mixed-query selection scheme [67] to form the object queries as an addition of content queries and spatial queries. The content queries are learnable embeddings, which is similar to DETR. The spatial queries are based on a two-stage scheme: selecting top-K features from the last layer in the Projector, predicting the bounding boxes, and transforming the corresponding boxes into embeddings as spatial queries.

Projector. We use a projector to connect the encoder and the decoder. The projector takes the aggregated encoded feature maps from the encoder as the input.

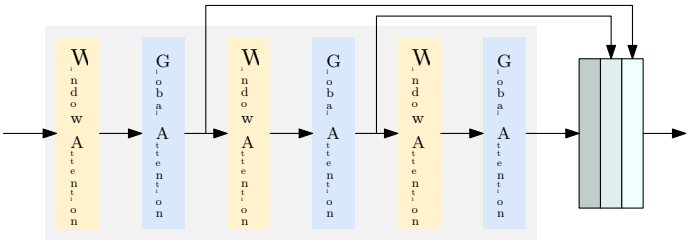


图2：一个具有多级特征图聚合及交错窗口与全局注意力机制的Transformer编码器示例。为清晰起见，未展示FFN层和LayerNorm层。

模型。我们的LW-DETR探索了使用普通ViT主干网络与DETR框架实现实时检测的可行性。

3 LW-DETR

3.1 架构

LW-DETR由ViT编码器、投影器和DETR解码器组成。

编码器。我们采用ViT作为检测编码器。基础的ViT[17]由图像块化层和Transformer编码器层构成。初始ViT中的每个Transformer编码器层包含一个作用于所有令牌的全局自注意力层和一个前馈网络层。全局自注意力计算成本高昂，其时间复杂度与令牌（图像块）数量呈平方关系。为降低计算复杂度，我们实现了采用窗口自注意力的部分Transformer编码器层（详见第3.4节）。我们提出聚合编码器中多层级特征图——包括中间特征图与最终特征图，以构建更强的编码特征表示。图2展示了该编码器的结构示例。

解码器。解码器由一系列Transformer解码层堆叠而成，每层包含自注意力机制、交叉注意力机制和前馈网络（FFN）。为提升计算效率，我们采用可变形交叉注意力[74]。DETR及其变体通常使用6层解码结构，而在我们的实现中仅采用3层Transformer解码层。这一改动使得处理时间从1.4毫秒降至0.7毫秒，与我们方法中微小版本剩余部分的耗时1.3毫秒相比，优化效果显著。

我们采用一种混合查询选择方案[67]来构建对象查询，即内容查询与空间查询的相加。内容查询是可学习的嵌入向量，类似于DETR的做法。空间查询则基于两阶段方案：首先从Projector最后一层筛选出前K个特征，预测其边界框，随后将这些对应框转换为嵌入向量作为空间查询。

投影器。我们使用一个投影器来连接编码器和解码器。该投影器以编码器输出的聚合特征图作为输入。

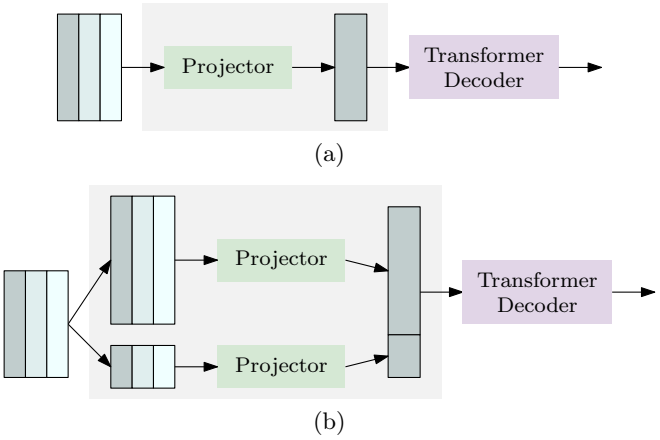


Fig. 3: Single-scale projector and multi-scale projector for (a) the `tiny`, `small`, and `medium` models, and (b) the `large` and `xlarge` models.

The projector is a C2f block (an extension of cross-stage partial DenseNet [26, 60]) that is implemented in YOLOv8 [29].

When forming the `large` and `xlarge` version of LW-DETR, we modify the projector to output two-scale ($\frac{1}{8}$ and $\frac{1}{32}$) feature maps and accordingly use the multi-scale decoder [74]. The projector contains two parallel C2f blocks. One processes $\frac{1}{8}$ feature maps, which are obtained by upsampling the input through a deconvolution, and the other processes $\frac{1}{32}$ maps that are obtained by downsampling the input through a stride convolution. Figure 3 shows the pipelines of the single-scale projector and the multi-scale projector.

Objective function. We adopt an IoU-aware classification loss, IA-BCE loss [3],

$$\ell_{\texttt{cls}} = \sum_{i=1}^{N_{pos}} \text{BCE}(s_i, t_i) + \sum_{j=1}^{N_{neg}} s_j^2 \,\text{BCE}(s_j, 0), \tag{1}$$

where N_{pos} and N_{neg} are the number of positive and negative samples. s is the predicted classification score. t is the target score absorbing the IoU score u (with the ground truth): $t = s^{\alpha} u^{1-\alpha}$, and α is empirically set as 0.25 [3].

The overall loss is a combination of the classification loss and the bounding box loss that is the same as in the DETR frameworks [4, 67, 74], which is formulated as follows:

$$\ell_{\texttt{cls}} + \lambda_{\texttt{iou}} \ell_{\texttt{iou}} + \lambda_{\ell_1} \ell_1. \tag{2}$$

where $\lambda_{\texttt{iou}}$ and λ_{ℓ_1} are set as 2.0 and 5.0 similar to [4, 67, 74]. $\ell_{\texttt{iou}}$ and ℓ_1 are the generalized IoU (GIoU) loss [52] and the L1 loss for the box regression.

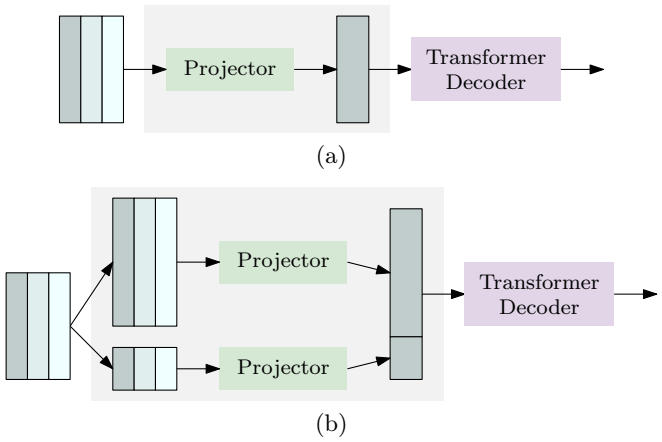


图3：(a) 微型、小型和中型模型及(b) 大型和超大型模型的单尺度投影仪与多尺度投影仪。

投影器是一个C2f模块（跨阶段部分DenseNet的扩展版本[26, 60]），在YOLOv8[29]中实现。

在构建LW-DETR的大规模及超大规模版本时，我们调整了投影器结构以输出双尺度（$\frac{1}{8}$和$\frac{1}{32}$）特征图，并相应采用了多尺度解码器[74]。该投影器包含两个并行的C2f模块：一个模块处理通过反卷积上采样输入得到的$\frac{1}{8}$特征图，另一个模块处理通过步进卷积下采样输入获得的$\frac{1}{32}$特征图。图3展示了单尺度投影器与多尺度投影器的处理流程。

目标函数。我们采用了一种IoU感知的分类损失，即IA-BCE损失[3],

$$\ell_{\texttt{cls}} = \sum_{i=1}^{N_{pos}} \text{BCE}(s_i, t_i) + \sum_{j=1}^{N_{neg}} s_j^2 \,\text{BCE}(s_j, 0), \tag{1}$$

其中N_{pos}和N_{neg}分别代表正负样本的数量。s为预测分类得分，t则是融合了IoU得分u(与真实标注)的目标分数：$t = s^{\alpha}u^{1-\alpha}$，而$\alpha$根据经验设定为0.25[3]。

总体损失是分类损失与边界框损失的组合，这与DETR框架[4, 67, 74]中的设定一致，其公式化表达如下：

$$\ell_{\texttt{cls}} + \lambda_{\texttt{iou}}\ell_{\texttt{iou}} + \lambda_{\ell_1}\ell_1. \tag{2}$$

其中$\lambda_{\texttt{iou}}$和λ_{ℓ_1}设置为2.0和5.0，类似于[4,67,74]中的做法。$\ell_{\texttt{iou}}$和ℓ_1分别表示边界框回归中的广义交并比损失（GIoU）[52]和L1损失。

Table 1: Architectures of five LW-DETR instances.

LW-DETR	ViT encoder				Projector			DETR decoder	
	#layers	dim	#global attention	#window attention	#blocks	dim	scales	#layers	#object queries
`tiny`	6	192	3	3	1	256	$\frac{1}{16}$	3	100
`small`	10	192	4	6	1	256	$\frac{1}{16}$	3	300
`medium`	10	384	4	6	1	256	$\frac{1}{16}$	3	300
`large`	10	384	4	6	1	384	$\frac{1}{8}, \frac{1}{32}$	3	300
`xlarge`	10	768	4	6	1	384	$\frac{1}{8}, \frac{1}{32}$	3	300

3.2 Instantiation

We instantiate 5 real-time detectors: `tiny`, `small`, `medium`, `large`, and `xlarge`. The detailed settings are given in Table 1.

The `tiny` detector consists of a transformer encoder with 6 layers. Each layer consists of a multi-head self-attention module and a feed-forward network (FFN). Each image patch is linear-mapped to a 192-dimensional representation vector. The projector outputs single-scale feature maps with 256 channels. There are 100 object queries for the decoder.

The `small` detector contains 10 encoder layers, and 300 object queries. Same as the `tiny` detector, the dimensions for the input patch representation and the output of the projector are 192 and 256. The `medium` detector is similar to `small`, and the differences include that the dimension of the input patch representation is 384, and accordingly the dimension for the encoder is 384.

The `large` detector consists of a 10-layer encoder and uses two-scale feature maps (see the part **Projector** in Section 3.1). The dimensions for the input patch representation and the output of the projector are 384 and 384. The `xlarge` detector is similar to `large`, and the difference is that the dimension of the input patch representation is 768.

3.3 Effective Training

More supervision. Various techniques have been developed to introduce more supervision for accelerating the DETR training, e.g., [6,27,75]. We adopt Group DETR [6] that is easily implemented and does not change the inference process. Following [6], we use 13 parallel weight-sharing decoders for training. For each decoder, we generate the object queries for each group from the output features of the projector. Following [6], we use the primary decoder for the inference.

Pretraining on Objects365. The pretraining process consists of two stages. First, we pretrain the ViT on the dataset Objects365 using a MIM method, CAEv2 [71], based on the pretrained models. This leads to a 0.7 mAP gain on COCO.

Second, we follow [7, 67] to retrain the encoder and train the projector and the decoder on Objects365 in a supervision manner.

3.4 Efficient Inference

We make a simple modification [36,37] and adopt interleaved window and global attentions: replacing some global self-attention layers with window self-attention

表1：五种LW-DETR实例的架构。

LW-DETR	ViT encoder				Projector			DETR decoder	
	#layers	dim	#global attention	#window attention	#blocks	dim	scales	#layers	#object queries
tiny	6	192	3	3	1	256	$\frac{1}{16}$	3	100
small	10	192	4	6	1	256	$\frac{1}{16}$	3	300
medium	10	384	4	6	1	256	$\frac{1}{16}$	3	300
large	10	384	4	6	1	384	$\frac{1}{8}$, $\frac{1}{32}$	3	300
xlarge	10	768	4	6	1	384	$\frac{1}{8}$, $\frac{1}{32}$	3	300

3.2 实例化

我们实例化了5个实时检测器：tiny、small、medium、large和xlarge。具体设置详见表1。

微型检测器由一个6层的Transformer编码器构成。每层包含一个多头自注意力模块和一个前馈网络(FFN)。每个图像块通过线性映射转换为192维表征向量。投影器输出256通道的单尺度特征图。解码器配备了100个目标查询向量。

小型检测器包含10个编码层和300个目标查询。与微型检测器相同，其输入补丁表示和投影器输出的维度分别为192和256。中型检测器与小型类似，不同之处在于输入补丁表示的维度为384，相应地编码器的维度也为384。

大型检测器由10层编码器构成，采用双尺度特征图（参见3.1节中的投影器部分）。输入补丁表示与投影器输出的维度均为384。超大型检测器与大型类似，区别在于其输入补丁表示的维度为768。

3.3 有效训练

更多监督。为了加速DETR训练，已发展出多种技术引入额外监督，例如[6,27,75]。我们采用易于实现且不改变推理过程的Group DETR[6]方案。按照[6]的方法，训练时使用13个权重共享的并行解码器。每个解码器的物体查询由投影器输出特征按组生成。推理阶段则如[6]所述，仅使用主解码器进行预测。

在Objects365上进行预训练。预训练过程分为两个阶段。首先，我们基于预训练模型，采用CAEv2[71]这一MIM方法在Objects365数据集上对ViT进行预训练。这一步骤使COCO数据集上的mAP提升了0.7。

其次，我们遵循[7, 67]的方法，以监督方式在Objects365上重新训练编码器，并训练投影器和解码器。

3.4 高效推理

我们做了一个简单的修改[36,37]，采用交错式窗口与全局注意力机制：将部分全局自注意力层替换为窗口自注意力层。

Table 2: The influence of effective training and efficient inference techniques. We show empirical results from an initial detector with global attention layers to the final LW-DETR-`small` model. '†' means we use the ViTDet implementation. The results except the last row are obtained under 45K iterations (equal to 12 epochs). The last row corresponds to the result of the final model with 180K training iterations.

model settings	#Params (M)	FLOPs (G)	Latency (ms)	mAP
initial detector	10.8	22.8	3.6	35.3
+ multi-level feature aggregation	11.0	23.0	3.7	36.0
+ interleaved window and global attention†	11.0	16.6	3.9	34.7
+ window-major feature map organization	11.0	16.6	2.9	34.7
+ iou-aware classification loss	11.0	16.6	2.9	35.4
+ more supervision	14.6	16.6	2.9	38.4
+ bounding box reparameterization	14.6	16.6	2.9	38.6
+ pretraining on Objects365	14.6	16.6	2.9	47.3
LW-DETR-`small`	14.6	16.6	2.9	48.0

layers. For example, in a 6-layer ViT, the first, third, and fifth layers are implemented with window attentions. The window attention is implemented by partitioning the feature map into non-overlapping windows and performing self-attention over each window separately.

We adopt a *window-major feature map organization* scheme for efficient interleaved attention, which organizes the feature maps window by window. The ViTDet implementation [36], where feature maps are organized row by row (row-major organization), requires costly permutation operations to transition feature maps from a row-major to a window-major organization for window attention. Our implementation removes these operations and thus reduces model latency.

We illustrate the window-major way using a toy example. Given a 4×4 feature map

$$\begin{bmatrix} f_{11} & f_{12} & f_{13} & f_{14} \\ f_{21} & f_{22} & f_{23} & f_{24} \\ f_{31} & f_{32} & f_{33} & f_{34} \\ f_{41} & f_{42} & f_{43} & f_{44} \end{bmatrix}, \tag{3}$$

the window-major organization for a window size 2×2 is as below:

$$\begin{aligned} &f_{11}, f_{12}, f_{21}, f_{22}; f_{13}, f_{14}, f_{23}, f_{24}; \\ &f_{31}, f_{32}, f_{41}, f_{42}; f_{33}, f_{34}, f_{43}, f_{44}. \end{aligned} \tag{4}$$

This organization is applicable to both window attention and global attention without rearranging the features. The row-major organization,

$$\begin{aligned} &f_{11}, f_{12}, f_{13}, f_{14}; f_{21}, f_{22}, f_{23}, f_{24}; \\ &f_{31}, f_{32}, f_{33}, f_{34}; f_{41}, f_{42}, f_{43}, f_{44}, \end{aligned} \tag{5}$$

is fine for global attention, and needs to be processed with the costly permutation operation for performing window attention.

3.5 Empirical Study

We empirically show how effective training and efficient inference techniques improve the DETR. We use the `small` detector as the example. The study is

表2：有效训练与高效推理技术的影响。我们展示了从初始带有全局注意力层的检测器到最终LW-DETR-small模型的实证结果。‘†’表示我们采用了ViTDet的实现方式。除最后一行外，其余结果均在45K次迭代（相当于12个训练周期）下获得。最后一行对应的是最终模型经过180K次训练迭代后的结果。

model settings	#Params (M)	FLOPs (G)	Latency (ms)	mAP
initial detector	10.8	22.8	3.6	35.3
+ multi-level feature aggregation	11.0	23.0	3.7	36.0
+ interleaved window and global attention†	11.0	16.6	3.9	34.7
+ window-major feature map organization	11.0	16.6	2.9	34.7
+ iou-aware classification loss	11.0	16.6	2.9	35.4
+ more supervision	14.6	16.6	2.9	38.4
+ bounding box reparameterization	14.6	16.6	2.9	38.6
+ pretraining on Objects365	14.6	16.6	2.9	47.3
LW-DETR-small	14.6	16.6	2.9	48.0

层。例如，在一个6层的ViT中，第一、第三和第五层采用了窗口注意力机制实现。窗口注意力的具体做法是将特征图分割成不重叠的窗口，并在每个窗口内分别执行自注意力计算。

我们采用了一种窗口主导的特征图组织方案，以实现高效的交叉注意力机制，该方案按窗口逐一对特征图进行组织。相比之下，ViTDet的实现[36]采用逐行（行主导组织）排列特征图，为了在窗口注意力中从行主导转为窗口主导组织，需进行昂贵的置换操作。我们的实现省去了这些操作，从而降低了模型延迟。

我们通过一个简单的示例来说明窗口主导的方式。给定一个4×4的特征图

$$\begin{bmatrix} f_{11} & f_{12} & f_{13} & f_{14} \\ f_{21} & f_{22} & f_{23} & f_{24} \\ f_{31} & f_{32} & f_{33} & f_{34} \\ f_{41} & f_{42} & f_{43} & f_{44} \end{bmatrix}, \tag{3}$$

窗口大小为2 × 2的窗口主导组织结构如下：

$$\begin{aligned} &f_{11}, f_{12}, f_{21}, f_{22}; f_{13}, f_{14}, f_{23}, f_{24}; \\ &f_{31}, f_{32}, f_{41}, f_{42}; f_{33}, f_{34}, f_{43}, f_{44}. \end{aligned} \tag{4}$$

该组织方式适用于窗口注意力和全局注意力，无需重新排列特征。行主序组织方式，

$$\begin{aligned} &f_{11}, f_{12}, f_{13}, f_{14}; f_{21}, f_{22}, f_{23}, f_{24}; \\ &f_{31}, f_{32}, f_{33}, f_{34}; f_{41}, f_{42}, f_{43}, f_{44}, \end{aligned} \tag{5}$$

全局注意力尚可，但需要进行代价高昂的排列操作以执行窗口注意力。

3.5 实证研究

我们通过实证展示了有效的训练和高效的推理技术如何提升DETR性能。以小型检测器为例，该研究

based on an initial detector: the encoder is formed with global attention in all the layers and outputs the feature map of the last layer. The results are shown in Table 2.

Latency improvements. The interleaved window and global attention, adopted by ViTDet, reduces the computational complexity from 23.0 GFlops to 16.6 GFlops, validating the benefit of replacing expensive global attention with cheaper window attention. The latency is not reduced and even increased by 0.2 ms. This is because extra costly permutation operations are needed in the row-major feature map organization. Window-major feature map organization alleviates the side effects and leads to a larger latency reduction of 0.8 ms, from 3.7 ms to 2.9 ms.

Performance improvements. Multi-level feature aggregation brings a 0.7 mAP gain. Iou-aware classification loss and more supervision improve the mAP scores from 34.7 to 35.4 and 38.4. Bounding box reparameterization for box regression target [40] (details in the Supplementary Material) makes a slight performance improvement. The significant improvement comes from pretraining on Objects365 and reaches 8.7 mAP, implying that the transformer indeed benefits from large data. A longer training schedule can give further improvements, forming our LW-DETR-`small` model.

4 Experiments

4.1 Settings

Datasets. The dataset for pretraining is Objects365 [54]. We follow [7, 67] to combine the images of the `train` set and the images in the `validate` set except the first $5k$ images for detection pretraining. We use the standard COCO2017 [39] data splitting policy and perform the evaluation on COCO `val2017`.

Data augmentations. We adopt the data augmentations in the DETR and its variants [4, 74]. We follow the real-time detection algorithms [1, 29, 46] and randomly resize the images into squares for training. For evaluating the performance and the inference time, we follow the evaluation scheme used in the real-time detection algorithms [1, 29, 46] to resize the images to 640×640. We use a window size of 10×10 to make sure that the image size can be divisible by the window size.

Implementation details. We pretrain the detection model on Objects365 [54] for 30 epochs and finetune the model on COCO [39] for a total number of 180K training iterations. We adopt the exponential moving average (EMA) technique [55] with a decay of 0.9997. We use the AdamW optimizer [44] for training.

For pretraining, we set the initial learning rate of the projector and the DETR decoder as $4 \times e^{-4}$, the initial learning rate of the ViT backbone is $6 \times e^{-4}$, and the batch size is 128. For fine-tuning, we set the initial learning rate of the projector and the DETR decoder as $1 \times e^{-4}$, and the initial learning rate of the ViT backbone as $1.5 \times e^{-4}$. We set the batch size as 32 in the `tiny`, `small`, and `medium` models, and the batch size as 16 in the `large` and `xlarge` models.

基于初始检测器：编码器在所有层中采用全局注意力机制，并输出最后一层的特征图。结果如表2所示。

延迟改进。ViTDet采用的交错窗口与全局注意力机制，将计算复杂度从23.0 GFlops降至16.6 GFlops，验证了用成本更低的窗口注意力替代昂贵全局注意力的优势。延迟并未降低，反而增加了0.2毫秒，这是由于在行优先特征图组织中需要额外的昂贵置换操作。采用窗口优先特征图组织缓解了副作用，实现了更大的延迟降幅0.8毫秒，从3.7毫秒减少至2.9毫秒。

性能提升。多级特征聚合带来了0.7 mAP的提升。IoU感知分类损失及额外监督将mAP分数从34.7提高到35.4和38.4。边界框回归目标的重新参数化[40]（详见补充材料）带来了轻微的性能改进。显著提升源自Objects365数据集上的预训练，达到了8.7 mAP，这表明Transformer确实受益于大数据量。更长的训练周期可带来进一步改进，由此形成了我们的LW-DETR-small模型。

4 实验

4.1 设置

数据集。预训练所用的数据集为Objects365 [54]。我们遵循[7, 67]的做法，将训练集的图像与验证集中除前5k张图像外的其余图像合并，用于检测预训练。我们采用标准的COCO2017 [39]数据划分策略，并在COCO val2017上进行评估。

数据增强。我们采用了DETR及其变体[4,74]中的数据增强方法。遵循实时检测算法[1,29,46]的做法，在训练时随机将图像调整为正方形尺寸。为评估性能和推理时间，我们沿用实时检测算法[1,29,46]的评估方案，将图像尺寸统一缩放至640×640。为确保图像尺寸可被窗口大小整除，我们采用10×10的窗口尺寸。

实现细节。我们首先在Objects365 [54]数据集上对检测模型进行了30个epoch的预训练，随后在COCO [39]数据集上进行了总计180K次训练迭代的微调。训练过程中采用了衰减率为0.9997的指数移动平均(EMA)技术[55]，并使用AdamW优化器[44]进行模型优化。

在预训练阶段，我们将投影器和DETR解码器的初始学习率设为4$\times e^{-4}$，ViT骨干网络的初始学习率设为6$\times e^{-4}$，批量大小设为128。在微调阶段，投影器和DETR解码器的初始学习率设为1$\times e^{-4}$，ViT骨干网络的初始学习率设为1.5 $\times e^{-4}$。对于tiny、small和medium模型，批量大小设为32；对于large和xlarge模型，批量大小设为16。

Table 3: Comparisons with state-of-the-art real-time detectors, including RTMDet [46], YOLOv8 [29], and YOLO-NAS [1]. The total latency is evaluated in an end-to-end manner on COCO `val2017` and includes the model latency and the postprocessing procedure NMS for non-DETR methods. We measure the total latency in two settings for NMS: official implementation and tuned score threshold. Our LW-DETR does not need NMS and the total latency is equal to the model latency. 'pretraining' means the result is based on pretraining on Objects365.

Method	pretraining	#Params (M)	FLOPs (G)	Model Latency (ms)	official implementation		tuned score threshold	
					Total Latency (ms)	mAP	Total Latency (ms)	mAP
RTMDet-tiny		4.9	8.1	2.1	7.4	41.0	2.4	40.8
RTMDet-tiny	✓	4.9	8.1	2.1	7.4	41.7	2.4	41.5
YOLOv8n		3.2	4.4	1.5	6.2	37.4	1.6	37.3
YOLOv8n	✓	3.2	4.4	1.5	6.2	37.6	1.6	37.5
LW-DETR-tiny	✓	12.1	11.2	2.0	2.0	42.6	-	-
RTMDet-s		8.9	14.8	2.8	7.9	44.6	2.9	44.4
RTMDet-s	✓	8.9	14.8	2.8	7.9	44.9	2.9	44.7
YOLOv8s		11.2	14.4	2.6	7.0	45.0	2.7	44.8
YOLOv8s	✓	11.2	14.4	2.6	7.0	45.2	2.7	45.1
YOLO-NAS-s	✓	19.0	17.6	2.8	4.7	47.6	2.9	47.3
LW-DETR-small	✓	14.6	16.6	2.9	2.9	48.0	-	-
RTMDet-m		24.7	39.2	6.2	10.8	49.3	6.5	49.1
RTMDet-m	✓	24.7	39.2	6.2	10.8	49.7	6.5	49.5
YOLOv8m		25.6	39.7	5.9	10.1	50.3	6.0	50.0
YOLOv8m	✓	25.6	39.7	5.9	10.1	50.6	6.0	50.4
YOLO-NAS-m	✓	51.1	48.0	5.5	7.8	51.6	5.7	51.1
LW-DETR-medium	✓	28.2	42.8	5.6	5.6	52.5	-	-
RTMDet-l		52.3	80.1	10.3	14.9	51.4	10.5	51.2
RTMDet-l	✓	52.3	80.1	10.3	14.9	52.4	10.5	52.2
YOLOv8l		43.7	82.7	9.3	13.2	53.0	9.4	52.5
YOLOv8l	✓	43.7	82.7	9.3	13.2	53.3	9.4	53.0
YOLO-NAS-l	✓	66.9	65.5	7.5	8.8	52.3	7.6	51.9
LW-DETR-large	✓	46.8	71.6	8.8	8.8	56.1	-	-
RTMDet-x		94.9	141.7	18.4	22.8	52.8	18.8	52.5
RTMDet-x	✓	94.9	141.7	18.4	22.8	54.0	18.8	53.5
YOLOv8x		68.2	129.3	14.8	19.1	54.0	15.0	53.5
YOLOv8x	✓	68.2	129.3	14.8	19.1	54.5	15.0	54.1
LW-DETR-xlarge	✓	118.0	174.2	19.1	19.1	58.3	-	-

The number of training iterations 180K is 50 epochs for the `tiny`, `small`, and `medium` models, and 25 epochs for the `large` and `xlarge` models. More details, such as weight decay, layer-wise decay in the ViT encoder, and component-wise decay [7] in the fine-tuning process, are given in the Supplementary Material.

We measure the averaged inference latency in an end-to-end manner with `fp16` precision and a batch size of 1 on COCO `val2017` with a T4 GPU, where the environment settings are with `TensorRT-8.6.1`, `CUDA-11.6`, and `CuDNN-8.7.0`. The `efficientNMSPlugin` in TensorRT is adopted for efficient NMS implementation. The performance and the end-to-end latency are measured for all real-time detectors using the official implementations.

4.2 Results

The results of our five LW-DETR models are reported in Table 3. LW-DETR-`tiny` achieves 42.6 mAP with 500 FPS on a T4 GPU. LW-DETR-`small` and LW-DETR-`medium` get 48.0 mAP with over 340 FPS and 52.5 mAP with a

表3：与最先进的实时检测器进行比较，包括RT-MDet [46]、YOLOv8 [29]和YOLO-NAS [1]。总延迟是在COCO val2017上以端到端方式评估的，包含模型延迟以及非DETR方法的后处理步骤NMS。我们在两种NMS设置下测量总延迟：官方实现和调整得分阈值。我们的LW-DETR不需要NMS，总延迟等同于模型延迟。“pretraining”表示结果基于在Objects365上的预训练。

Method	pretraining	#Params (M)	FLOPs (G)	Model Latency (ms)	official implementation		tuned score threshold	
					Total Latency (ms)	mAP	Total Latency (ms)	mAP
RTMDet-tiny		4.9	8.1	2.1	7.4	41.0	2.4	40.8
RTMDet-tiny	✓	4.9	8.1	2.1	7.4	41.7	2.4	41.5
YOLOv8n		3.2	4.4	1.5	6.2	37.4	1.6	37.3
YOLOv8n	✓	3.2	4.4	1.5	6.2	37.6	1.6	37.5
LW-DETR-tiny	✓	12.1	11.2	2.0	2.0	42.6	-	-
RTMDet-s		8.9	14.8	2.8	7.9	44.6	2.9	44.4
RTMDet-s	✓	8.9	14.8	2.8	7.9	44.9	2.9	44.7
YOLOv8s		11.2	14.4	2.6	7.0	45.0	2.7	44.8
YOLOv8s	✓	11.2	14.4	2.6	7.0	45.2	2.7	45.1
YOLO-NAS-s	✓	19.0	17.6	2.8	4.7	47.6	2.9	47.3
LW-DETR-small	✓	14.6	16.6	2.9	2.9	48.0	-	-
RTMDet-m		24.7	39.2	6.2	10.8	49.3	6.5	49.1
RTMDet-m	✓	24.7	39.2	6.2	10.8	49.7	6.5	49.5
YOLOv8m		25.6	39.7	5.9	10.1	50.3	6.0	50.0
YOLOv8m	✓	25.6	39.7	5.9	10.1	50.6	6.0	50.4
YOLO-NAS-m	✓	51.1	48.0	5.5	7.8	51.6	5.7	51.1
LW-DETR-medium	✓	28.2	42.8	5.6	5.6	52.5	-	-
RTMDet-l		52.3	80.1	10.3	14.9	51.4	10.5	51.2
RTMDet-l	✓	52.3	80.1	10.3	14.9	52.4	10.5	52.2
YOLOv8l		43.7	82.7	9.3	13.2	53.0	9.4	52.5
YOLOv8l	✓	43.7	82.7	9.3	13.2	53.3	9.4	53.0
YOLO-NAS-l	✓	66.9	65.5	7.5	8.8	52.3	7.6	51.9
LW-DETR-large	✓	46.8	71.6	8.8	8.8	56.1	-	-
RTMDet-x		94.9	141.7	18.4	22.8	52.8	18.8	52.5
RTMDet-x	✓	94.9	141.7	18.4	22.8	54.0	18.8	53.5
YOLOv8x		68.2	129.3	14.8	19.1	54.0	15.0	53.5
YOLOv8x	✓	68.2	129.3	14.8	19.1	54.5	15.0	54.1
LW-DETR-xlarge	✓	118.0	174.2	19.1	19.1	58.3	-	-

训练迭代次数180K对应tiny、small和medium模型为50个epoch，large和xlarge模型则为25个epoch。更多细节（如权重衰减、ViT编码器中的分层衰减以及微调过程中的组件级衰减[7]）详见补充材料。

我们采用端到端的方式测量平均推理延迟，使用fp16精度和批大小为1的设置在COCO val2017数据集上，搭配T4 GPU进行测试。环境配置为TensorRT-8.6.1、CUDA-11.6和CuDNN-8.7.0。采用TensorRT中的efficientNMSPlugin实现高效非极大值抑制。所有实时检测器的性能及端到端延迟均通过官方实现进行测量。

4.2 结果

我们五个LW-DETR模型的结果如表3所示。LW-DETR-tiny在T4 GPU上以500 FPS的速度实现了42.6 mAP。LW-DETR-small和LW-DETR-medium分别以超过340 FPS的速度获得48.0 mAP，以及52.5 mAP。

speed of over 178 FPS respectively. The `large` and `xlarge` models achieve 56.1 mAP with 113 FPS, and 58.3 mAP with 52 FPS.

Comparisons with state-of-the-art real-time detectors. In Table 3, we report the comparison of the LW-DETR models against representative real-time detectors, including YOLO-NAS [1], YOLOv8 [29], and RTMDet [46]. One can see that LW-DETR consistently outperforms previous SoTA real-time detectors with and without using pretraining. Our LW-DETR shows clear superiority over YOLOv8 and RTMDet in terms of latency and detection performance for the five scales from `tiny` to `xlarge`.

In comparison to one of previous best methods YOLO-NAS, that is obtained with neural architecture search, our LW-DETR model outperforms it by 0.4 mAP and 0.9 mAP, and runs 1.6× and ∼1.4× faster at the `small` and `medium` scales. When the model gets larger, the improvement becomes more significant: a 3.8 mAP improvement when running at the same speed at the `large` scale.

We further improve other methods by well-tuning the classification score threshold in the NMS procedure, and report the results in the right two columns. The results are greatly improved, and still lower than our LW-DETR. We expect that our approach potentially benefits from other improvements, such as neural architecture search (NAS), data augmentation, pseudo-labeled data, and knowledge distillation that are exploited by previous real-time detectors [1, 29, 46].

Comparison with concurrent works. We compare our LW-DETR with concurrent works in real-time detection, YOLO-MS [12], Gold-YOLO [58], RT-DETR [45], and YOLOv10 [57]. YOLO-MS improves the performance by enhancing the multi-scale feature representations. Gold-YOLO boosts the multi-scale feature fusion and applies MAE-style pretraining [22] to improve the YOLO performance. YOLOv10 designs several efficiency and accuracy driven modules to improve the performance. RT-DETR [45], closely related to LW-DETR, is also built on the DETR framework, with many differences from our approach in the backbone, the projector, the decoder, and the training schemes.

The comparisons are given in Table 4 and Figure 4. Our LW-DETR consistently achieves a better balance between the detection performance and the latency. YOLO-MS and Gold-YOLO clearly show worse results than our LW-DETR for all the model scales. LW-DETR-`large` outperforms the closely related RT-DETR-R50 by 0.8 mAP and shows faster speed (8.8 ms vs. 9.9 ms). LW-DETR with other scales also shows better results than RT-DETR. Compared to the latest work, YOLOv10-X [57], our LW-DETR-`large` achieves higher performance (56.1 mAP vs. 54.4 mAP) with lower latency (8.8 ms vs. 10.70 ms).

4.3 Discussions

NMS post-processing. The DETR method is an end-to-end algorithm that does not need the NMS post-processing procedure. In contrast, existing real-time detectors, such as YOLO-NAS [1], YOLOv8 [29], and RTMDet [46], needs NMS [24] post-processing. The NMS procedure takes extra time. We include the extra time for measuring the end-to-end inference cost, which is counted in

速度分别超过178 FPS。大型和超大型模型分别以113 FPS实现56.1 mAP，以及以52 FPS实现58.3 mAP。

与最先进的实时检测器对比。在表3中，我们报告了LW-DETR模型与代表性实时检测器（包括YOLO-NAS[1]、YOLOv8[29]和RTMDet[46]）的对比结果。可以看出，无论是否使用预训练，LW-DETR始终优于之前的SoTA实时检测器。从微小到超大的五个规模级别来看，我们的LW-DETR在延迟和检测性能方面均明显优于YOLOv8和RTMDet。

与之前最佳方法之一YOLO-NAS（通过神经架构搜索获得）相比，我们的LW-DETR模型在mAP指标上分别超出0.4和0.9，并在小规模和中等规模下运行速度提升1.6×倍和∼1.4×倍。当模型规模增大时，改进更为显著：在大规模下保持相同速度运行时，mAP提升达3.8。

我们通过精细调整NMS过程中的分类分数阈值，进一步改进了其他方法，并在右侧两列中报告了结果。这些结果得到了显著提升，但仍低于我们的LW-DETR。我们预期，我们的方法还可能受益于其他改进措施，如神经架构搜索（NAS）、数据增强、伪标签数据以及知识蒸馏等，这些技术已被先前的实时检测器所采用[1, 29, 46]。

与同期工作的比较。我们将LW-DETR与实时检测领域的同期工作进行了对比，包括YOLO-MS [12]、Gold-YOLO [58]、RT-DETR [45]和YOLOv10 [57]。YOLO-MS通过增强多尺度特征表示来提升性能。Gold-YOLO优化了多尺度特征融合，并采用MAE式预训练[22]以提高YOLO的性能。YOLOv10设计了多个以效率和精度为导向的模块来改进性能。RT-DETR [45]与LW-DETR密切相关，同样基于DETR框架，但在骨干网络、投影器、解码器及训练方案等方面与我们的方法存在诸多差异。

对比结果如表4和图4所示。我们的LW-DETR始终在检测性能和延迟之间实现了更好的平衡。YOLO-MS和Gold-YOLO在所有模型规模上均明显逊色于LW-DETR。LW-DETR-large以0.8 mAP的优势超越密切相关的RT-DETR-R50，并展现出更快的速度（8.8毫秒 vs. 9.9毫秒）。其他规模的LW-DETR同样优于RT-DETR。与最新工作YOLOv10-X[57]相比，我们的LW-DETR-large以更低延迟（8.8毫秒 vs. 10.70毫秒）实现了更高性能（56.1 mAP vs. 54.4 mAP）。

4.3 讨论

NMS后处理。DETR方法是一种端到端算法，无需NMS后处理步骤。相比之下，现有的实时检测器如YOLO-NAS[1]、YOLOv8[29]和RTMDet[46]均需依赖NMS[24]后处理。该后处理过程会消耗额外时间。我们在测量端到端推理成本时已计入这部分额外耗时。

Table 4: Comparisons with concurrent works, including YOLO-MS [12], Gold-YOLO [58], YOLOv10 [57], and RT-DETR [45] on COCO. For YOLO-MS and Gold-YOLO, we measure the total latency in two settings for NMS: official implementation and tuned score threshold. For YOLOv10, we report the results in the official paper [57]. RT-DETR is based on DETR, and the total latency is equal to the model latency. We provide the best latency among the reported inference time in the paper [45] and the measured time in our environment for RT-DETR. LW-DETR consistently gets superior results. 'pretraining' means that the results are based on pretraining on Objects365.

Method	pretraining	#Params (M)	FLOPs (G)	Model Latency (ms)	official implementation Total Latency (ms)	official implementation mAP	tuned score threshold Total Latency (ms)	tuned score threshold mAP
YOLO-MS-XS		4.5	8.7	3.0	6.9	43.4	3.2	43.3
YOLO-MS-XS	✓	4.5	8.7	3.0	6.9	43.9	3.2	43.8
YOLO-MS-S		8.1	15.6	5.4	9.2	46.2	5.6	46.1
YOLO-MS-S	✓	8.1	15.6	5.4	9.2	46.8	5.6	46.7
YOLO-MS		22.0	40.1	8.6	12.3	51.0	9.0	50.8
Gold-YOLO-S		21.5	23.0	2.9	3.6	45.5	3.4	45.4
Gold-YOLO-S	✓	21.5	23.0	2.9	3.6	46.1	3.4	46.0
Gold-YOLO-M		41.3	43.8	5.8	6.3	50.2	6.1	50.2
Gold-YOLO-M	✓	41.3	43.8	5.8	6.3	50.4	6.1	50.3
Gold-YOLO-L		75.1	75.9	10.2	10.6	52.3	10.5	52.2
YOLOv10-N		2.3	6.7	-	1.84	38.5	-	-
YOLOv10-S		7.2	21.6	-	2.49	46.3	-	-
YOLOv10-M		15.4	59.1	-	4.74	51.1	-	-
YOLOv10-B		19.1	92.0	-	5.74	52.5	-	-
YOLOv10-L		24.4	120.3	-	7.28	53.2	-	-
YOLOv10-X		29.5	160.4	-	10.70	54.4	-	-
RT-DETR-R18		20	30.0	4.6	4.6	46.5	-	-
RT-DETR-R18	✓	20	30.0	4.6	4.6	49.2	-	-
RT-DETR-R50		42	69.4	9.3	9.3	53.1	-	-
RT-DETR-R50	✓	42	69.4	9.3	9.3	55.3	-	-
RT-DETR-R101		76	131.0	13.5	13.5	54.3	-	-
RT-DETR-R101	✓	76	131.0	13.5	13.5	56.2	-	-
LW-DETR-tiny	✓	12.1	11.2	2.0	2.0	42.6	-	-
LW-DETR-small	✓	14.6	16.6	2.9	2.9	48.0	-	-
LW-DETR-medium	✓	28.2	42.8	5.6	5.6	52.5	-	-
LW-DETR-large	✓	46.8	71.6	8.8	8.8	56.1	-	-
LW-DETR-xlarge	✓	118.0	174.2	19.1	19.1	58.3	-	-

real-world application. The results using the NMS setting in the official implementations are shown in Figure 1 and Table 3.

We further make improvements for the methods with NMS by tuning the classification score threshold for the NMS post-processing. We observe that the default score threshold, 0.001, in YOLO-NAS, YOLOv8, and RTMDet, results in a high mAP, but a large number of boxes and thus high latency. In particular, when the model is small, the end-to-end latency is dominated by the NMS latency. We tune the threshold, obtaining a good balance between the mAP score and the latency. It is observed that the mAP scores are slightly dropped, e.g., by -0.1 mAP to -0.5 mAP, and the running times are largely reduced, e.g., with a reduction of $4\sim5$ ms for RTMDet and YOLOv8, a reduction of $1\sim2$ ms for YOLO-NAS. These reductions are from that fewer predicted boxes are fed into NMS after tuning the score threshold. Detailed results with different score thresholds, along with the distribution of the number of remaining boxes across the COCO `val2017` are given in the Supplementary Material.

表4：与同期工作的比较，包括YOLO-MS [12]、Gold-YOLO [58]、YOLOv10 [57]和RT-DETR [45]在COCO数据集上的表现。对于YOLO-MS和Gold-YOLO，我们测量了两种NMS设置下的总延迟：官方实现和调整得分阈值。对于YOLOv10，我们报告了官方论文[57]中的结果。RT-DETR基于DETR架构，其总延迟等同于模型延迟。我们提供了论文[45]中报告的最佳推理时间及在我们环境中测量的RT-DETR时间。LW-DETR始终取得更优结果。“pretraining”表示结果基于Objects365数据集上的预训练。

Method	pretraining	#Params (M)	FLOPs (G)	Model Latency (ms)	official implementation		tuned score threshold	
					Total Latency (ms)	mAP	Total Latency (ms)	mAP
YOLO-MS-XS		4.5	8.7	3.0	6.9	43.4	3.2	43.3
YOLO-MS-XS	✓	4.5	8.7	3.0	6.9	43.9	3.2	43.8
YOLO-MS-S		8.1	15.6	5.4	9.2	46.2	5.6	46.1
YOLO-MS-S	✓	8.1	15.6	5.4	9.2	46.8	5.6	46.7
YOLO-MS		22.0	40.1	8.6	12.3	51.0	9.0	50.8
Gold-YOLO-S		21.5	23.0	2.9	3.6	45.5	3.4	45.4
Gold-YOLO-S	✓	21.5	23.0	2.9	3.6	46.1	3.4	46.0
Gold-YOLO-M		41.3	43.8	5.8	6.3	50.2	6.1	50.2
Gold-YOLO-M	✓	41.3	43.8	5.8	6.3	50.4	6.1	50.3
Gold-YOLO-L		75.1	75.9	10.2	10.6	52.3	10.5	52.2
YOLOv10-N		2.3	6.7	-	1.84	38.5	-	-
YOLOv10-S		7.2	21.6	-	2.49	46.3	-	-
YOLOv10-M		15.4	59.1	-	4.74	51.1	-	-
YOLOv10-B		19.1	92.0	-	5.74	52.5	-	-
YOLOv10-L		24.4	120.3	-	7.28	53.2	-	-
YOLOv10-X		29.5	160.4	-	10.70	54.4	-	-
RT-DETR-R18		20	30.0	4.6	4.6	46.5	-	-
RT-DETR-R18	✓	20	30.0	4.6	4.6	49.2	-	-
RT-DETR-R50		42	69.4	9.3	9.3	53.1	-	-
RT-DETR-R50	✓	42	69.4	9.3	9.3	55.3	-	-
RT-DETR-R101		76	131.0	13.5	13.5	54.3	-	-
RT-DETR-R101	✓	76	131.0	13.5	13.5	56.2	-	-
LW-DETR-tiny	✓	12.1	11.2	2.0	2.0	42.6	-	-
LW-DETR-small	✓	14.6	16.6	2.9	2.9	48.0	-	-
LW-DETR-medium	✓	28.2	42.8	5.6	5.6	52.5	-	-
LW-DETR-large	✓	46.8	71.6	8.8	8.8	56.1	-	-
LW-DETR-xlarge	✓	118.0	174.2	19.1	19.1	58.3	-	-

实际应用。采用官方实现中的非极大值抑制（NMS）设置所得结果如图1和表3所示。

我们进一步对采用非极大值抑制（NMS）的方法进行优化，通过调整NMS后处理中的分类得分阈值。观察到YOLO-NAS、YOLOv8和RTMDet中默认的得分阈值0.001虽然能带来较高的mAP值，但会导致预测框数量过多，从而显著增加延迟。特别是当模型较小时，端到端延迟主要由NMS处理时间决定。通过调整该阈值，我们在mAP分数与延迟之间取得了良好平衡。实验表明，mAP分数略有下降（例如降低−0.1至−0.5 mAP），而运行时间大幅缩短（如RTMDet和YOLOv8减少4∼5毫秒，YOLO-NAS减少1∼2毫秒）。这些改进源于调整得分阈值后，输入NMS的预测框数量减少。补充材料中提供了不同得分阈值的详细实验结果，以及COCO val2017数据集上剩余框数量的分布情况。

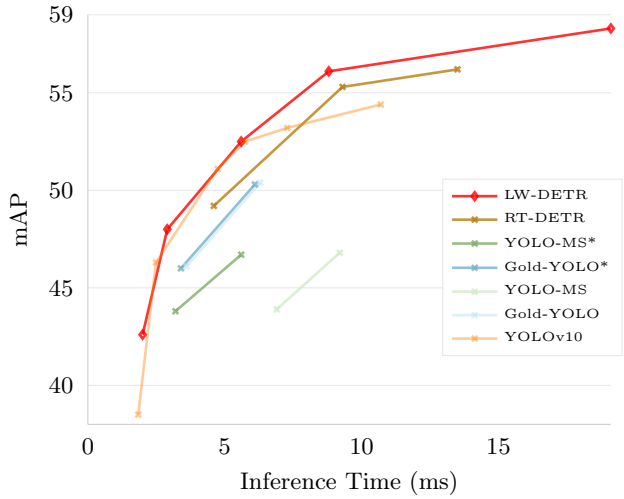


Fig. 4: Our approach outperforms concurrent works. The x-axis corresponds to the inference time. The y-axis corresponds to the mAP score on COCO `val2017`. Our LW-DETR, RT-DETR [45], YOLO-MS [12], and Gold-YOLO [58] are trained with pretraining on Objects365, while YOLOv10 [57] is not. The NMS post-processing times are included for YOLO-MS and Gold-YOLO, and measured on the COCO `val2017` with the setting from the official implementation, and the well-tuned NMS postprocessing setting (labeled as "*").

Table 5: The effect of pertaining in our LW-DETR. Pretraining on Objects365 improves our approach a lot. This observation is consistent with the observations from methods with large models [7, 67, 75].

LW-DETR	#Params (M)	FLOPs (G)	Latency (ms)	mAP w/o pretraining	mAP w/ pretraining
`tiny`	12.1	11.2	2.0	36.5	42.6
`small`	14.6	16.6	2.9	43.6	48.0
`medium`	28.2	42.8	5.6	47.2	52.5
`large`	46.8	71.6	8.8	49.5	56.1
`xlarge`	118.0	174.2	19.1	53.0	58.3
R18	21.2	21.4	2.5	40.9	44.4
R50	54.6	67.7	8.7	49.7	54.4

Figure 1 shows the comparison against other methods with well-tuned NMS procedures. The methods with NMS are improved. Our approach still outperforms other methods. The second-best approach, YOLO-NAS, is a network architecture search algorithm and performs very closely to the proposed baseline. We believe that the complicated network architecture search procedure, like the one used in YOLO-NAS, potentially benefits the DETR approach, and further improvement is expected.

Pretraining. We empirically study the effect of pretraining. The results, shown in Table 5, indicate that pretraining leads to significant improvements for our approaches, with an average improvement of 5.5 mAP. The `tiny` model gets an mAP gain of 6.1, and the `xlarge` model gets an mAP gain of 5.3. This implies that pretraining on a large dataset is highly beneficial for DETR-based models.

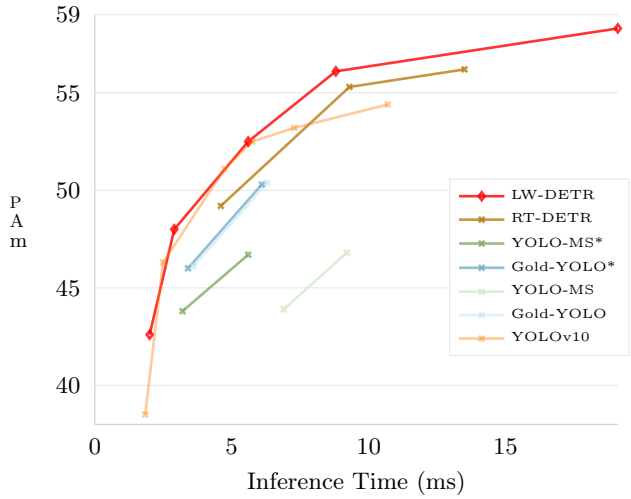


图4：我们的方法优于同期工作。x轴对应推理时间，y轴对应COCO val2017数据集的mAP分数。我们的LW-DETR、RT-DETR[45]、YOLO-MS[12]和Gold-YOLO[58]均在Objects365上进行了预训练，而YOLOv10[57]则未进行。YOLO-MS和Gold-YOLO的NMS后处理时间包含在内，该时间基于官方实现设置及经调优的NMS后处理设置（标注为“*”）在COCO val2017上测得。

表5：预训练对我们LW-DETR的影响。在Objects365上进行预训练显著提升了我们的方法。这一观察结果与使用大型模型的方法[7, 67, 75]中的观察一致。

LW-DETR	#Params (M)	FLOPs (G)	Latency (ms)	mAP w/o pretraining	mAP w/ pretraining
tiny	12.1	11.2	2.0	36.5	42.6
small	14.6	16.6	2.9	43.6	48.0
medium	28.2	42.8	5.6	47.2	52.5
large	46.8	71.6	8.8	49.5	56.1
xlarge	118.0	174.2	19.1	53.0	58.3
R18	21.2	21.4	2.5	40.9	44.4
R50	54.6	67.7	8.7	49.7	54.4

图1展示了与其他采用精心调校NMS流程的方法的对比。采用NMS的方法均有所改进。我们的方法依然优于其他方法。表现次优的YOLO-NAS是一种网络架构搜索算法，其性能与所提基线非常接近。我们认为，类似YOLO-NAS采用的复杂网络架构搜索流程，可能对DETR方法有所裨益，预期会带来进一步的性能提升。

预训练。我们通过实证研究预训练的效果。如表5所示，结果表明预训练为我们的方法带来了显著提升，平均提升幅度达5.5 mAP。其中，微型模型的mAP增益为6.1，超大型模型的mAP增益为5.3。这说明在大规模数据集上进行预训练对基于DETR的模型极为有利。

Table 6: The effect of pertaining along with training epochs for non-end-to-end detectors.

Model	pretraining	20 epochs	60 epochs	100 epochs	200 epochs	300 epochs	500 epochs
YOLOv8n		26.2	30.1	32.3	34.0	35.0	37.4
YOLOv8n	✓	31.5	32.8	33.2	34.3	35.2	37.6
RTMDet-t		30.4	34.2	35.2	36.8	41.0	-
RTMDet-t	✓	33.4	36.5	36.7	37.5	41.7	-
YOLO-MS-XS		24.7	34.3	36.4	39.2	43.4	-
YOLO-MS-XS	✓	37.5	38.6	39.1	40.0	43.9	-
Gold-YOLO-S		33.4	37.1	38.2	43.6	45.5	-
Gold-YOLO-S	✓	39.3	41.3	42.1	44.9	46.1	-

We further show that the training procedure applies to the DETR approach with convolutional encoders. We replace transformer encoders with ResNet-18 and ResNet-50. One can see that in Table 5, the results of these LW-DETR variants are close to LW-DETR with transformer encoders in terms of latency and mAP, and the pretraining brings benefits that are similar to and a little lower than LW-DETR with transformer encoders.

Meanwhile, we investigate the pretraining improvements on non-end-to-end detectors. According to the results in Table 3, Table 4 and Table 6, it seems that pretraining on Objects365 only show limited gains for non-end-to-end detectors [12, 29, 46, 58], which is different from the phenomenon in DETR-based detectors, where pretraining give large improvements. As the non-end-to-end detectors train 300 epochs even 500 epochs in YOLOv8, we wonder if the limited gain is related to the training epochs. We compare the improvements brought by the pretrained weights along with the training epochs. Table 6 shows that the improvements diminished along with the training epochs, which partly supports the above hypothesis. The above illustration is a preliminary step. We believe that more investigations are needed to figure out the underlying reasons for the difference in benefits of pretraining.

4.4 Experiments on more datasets

We test the generalizability of our LW-DETR on more detection datasets. We consider two types of evaluation methods, cross-domain evaluation and multi-domain finetuning. For cross-domain evaluation, we directly evaluate the real-time detectors trained on COCO on the Unidentified Video Objects (UVO) [61]. For multi-domain fine-tuning, we finetune the pretrained real-time detectors on the multi-domain detection dataset Roboflow 100 (RF100) [13]. We do a coarse search on the hyperparameters on each dataset for all models, such as the learning rate. Please refer to the Supplementary Material for more details.

Cross-domain evaluation. One possible way to evaluate the generalizability of the models is to directly evaluate them on the datasets with different domains. We adopt a class-agnostic object detection benchmark, UVO [61], where 57% object instances do not belong to any of the 80 COCO classes. UVO is based on YouTube videos, whose appearance is very different from COCO, e.g., some

表6：非端到端检测器中预训练与训练轮次的影响。

Model	pretraining	20 epochs	60 epochs	100 epochs	200 epochs	300 epochs	500 epochs
YOLOv8n		26.2	30.1	32.3	34.0	35.0	37.4
YOLOv8n	✓	31.5	32.8	33.2	34.3	35.2	37.6
RTMDet-t		30.4	34.2	35.2	36.8	41.0	-
RTMDet-t	✓	33.4	36.5	36.7	37.5	41.7	-
YOLO-MS-XS		24.7	34.3	36.4	39.2	43.4	-
YOLO-MS-XS	✓	37.5	38.6	39.1	40.0	43.9	-
Gold-YOLO-S		33.4	37.1	38.2	43.6	45.5	-
Gold-YOLO-S	✓	39.3	41.3	42.1	44.9	46.1	-

我们进一步证明，该训练流程同样适用于采用卷积编码器的DETR方法。我们将transformer编码器替换为ResNet-18和ResNet-50。从表5可见，这些LW-DETR变体在延迟和mAP指标上与使用transformer编码器的LW-DETR表现接近，且预训练带来的收益与transformer编码器版本相似，仅略低一些。

与此同时，我们研究了预训练对非端到端检测器的改进效果。根据表3、表4和表6的结果显示，在Objects365上的预训练仅能为非端到端检测器[12,29,46,58]带来有限提升，这与基于DETR的检测器中预训练带来显著改进的现象形成对比。考虑到非端到端检测器（如YOLOv8）通常需要训练300甚至500个周期，我们推测这种有限增益可能与训练周期数相关。通过对比预训练权重随训练周期增加带来的改进幅度，表6数据表明：随着训练周期延长，改进效果逐渐减弱，这部分验证了上述假设。当前分析仅为初步探索，我们认为仍需进一步研究以揭示预训练效益差异的内在原因。

4.4 更多数据集上的实验

我们在更多检测数据集上测试了LW-DETR的泛化能力。我们考虑两种评估方法：跨域评估和多域微调。对于跨域评估，我们直接将基于COCO训练的实时检测器应用于未识别视频对象(UVO)[61]进行测试。在多域微调方面，我们在多域检测数据集Roboflow 100(RF100)[13]上对预训练的实时检测器进行微调。针对所有模型，我们在每个数据集上对超参数（如学习率）进行了粗粒度搜索。更多细节请参阅补充材料。

跨领域评估。评估模型泛化能力的一种可能方式是直接在具有不同领域的数据集上进行测试。我们采用了一个类别无关的目标检测基准——UVO[61]，其中57%的目标实例不属于COCO数据集的80个类别中的任何一个。UVO基于YouTube视频，其外观与COCO存在显著差异，例如，一些

Table 7: Cross-domain evaluation on UVO. We evaluate the performance in a class-agnostic way as UVO is class-agnostic. LW-DETR demonstrates higher AP and AR than other detectors.

Method	mAP	AP50	AR@100	AR_s	AR_m	AR_l
RTMDet-s	29.7	43.3	55.7	26.5	49.4	71.5
YOLOv8-s	29.1	42.4	54.3	27.4	48.6	68.8
YOLO-NAS-s	31.0	44.5	55.1	25.8	48.1	71.6
LW-DETR-small	**32.3**	**45.1**	**59.8**	**29.4**	**52.4**	**77.1**

Table 8: Multi-domain finetuning on RF100. We compare our LW-DETR with previous real-time detectors, including YOLOv5, YOLOv7, RTMDet, YOLOv8, and YOLO-NAS on all data domains of RF100. Gray entries are the results from the RF100 paper [13]. The data domains are aerial, videogames, microscopic, underwater, documents, electromagnetic, and real world. The AP50 metric is used. '-' means that YOLO-NAS [1] does not report their detailed results.

Method	Average	Domains in Roboflow 100						
		aerial	videogames	microscopic	underwater	documents	electromagnetic	real world
YOLOv5-s	73.4	63.6	85.9	65.0	56.0	71.6	74.2	76.9
YOLOv7-s	67.4	50.4	79.6	59.1	66.2	72.2	63.9	70.5
RTMDet-s	79.2	70.1	88.0	68.1	68.0	81.0	77.4	83.3
YOLOv8-s	80.1	70.7	87.9	74.7	70.2	79.8	79.0	82.9
YOLO-NAS-s	81.5	-	-	-	-	-	-	-
YOLO-NAS-m	81.8	-	-	-	-	-	-	-
LW-DETR-small	**82.5**	71.8	88.9	74.5	69.6	86.7	84.6	85.3
LW-DETR-medium	**83.5**	72.9	90.8	75.4	70.5	86.1	86.2	86.4

videos are in egocentric views and have significant motion blur. We evaluate the models trained with COCO (taken from Table 3) on the validation split of UVO.

Table 7 provides the results. LW-DETR excels over competing SoTA real-time detectors. Specifically, LW-DETR-`small` is 1.3 mAP and 4.1 AR higher than the best result among RTMDet-s, YOLOv8-s, and YOLO-NAS-s. In terms of recall, it also shows enhanced abilities to detect more objects across different scales: small, medium, and large. The above findings imply that the superiority of our LW-DETR over previous real-time detectors is attributed not to specific tuning for COCO, but to its capacity for producing more generalizable models.

Multi-domain finetuning. Another way is to finetune the pretrained detectors on small datasets across different domains. RF100 consists of 100 small datasets, 7 imagery domains, 224k images, and 829 class labels. It can help researchers test the model's generalizability with real-life data. We finetune the real-time detectors on each small dataset of RF100.

The results are given in Table 8. LW-DETR-`small` shows superiority over current state-of-the-art real-time detectors across different domains. In particular, for the 'documents' and the 'electromagnetic' domains, our LW-DETR is significantly better than YOLOv5, YOLOv7, RTMDet, and YOLOv8 (5.7 AP and 5.6 AP higher than the best among the four). LW-DETR-`medium` can give further improvements overall. These findings highlight the versatility of our LW-DETR, positioning it as a strong baseline in a range of closed-domain tasks.

表7：UVO上的跨领域评估。由于UVO是类别无关的，我们以类别无关的方式评估性能。LW-DETR显示出比其他检测器更高的AP和AR。

Method	mAP	AP50	AR@100	AR_s	AR_m	AR_l
RTMDet-s	29.7	43.3	55.7	26.5	49.4	71.5
YOLOv8-s	29.1	42.4	54.3	27.4	48.6	68.8
YOLO-NAS-s	31.0	44.5	55.1	25.8	48.1	71.6
LW-DETR-small	**32.3**	**45.1**	**59.8**	**29.4**	**52.4**	**77.1**

表8：RF100上的多领域微调。我们将LW-DETR与之前的实时检测器进行比较，包括YOLOv5、YOLOv7、RTMDet、YOLOv8和YOLO-NAS在RF100所有数据领域上的表现。灰色条目为RF100论文[13]中的结果。数据领域涵盖航空、视频游戏、显微、水下、文档、电磁及现实世界。采用AP50指标。“-”表示YOLO-NAS[1]未报告其详细结果。

Method	Average	Domains in Roboflow 100						
		aerial	videogames	microscopic	underwater	documents	electromagnetic	real world
YOLOv5-s	73.4	63.6	85.9	65.0	56.0	71.6	74.2	76.9
YOLOv7-s	67.4	50.4	79.6	59.1	66.2	72.2	63.9	70.5
RTMDet-s	79.2	70.1	88.0	68.1	68.0	81.0	77.4	83.3
YOLOv8-s	80.1	70.7	87.9	74.7	70.2	79.8	79.0	82.9
YOLO-NAS-s	81.5	-	-	-	-	-	-	-
YOLO-NAS-m	81.8	-	-	-	-	-	-	-
LW-DETR-small	**82.5**	71.8	88.9	74.5	69.6	86.7	84.6	85.3
LW-DETR-medium	**83.5**	72.9	90.8	75.4	70.5	86.1	86.2	86.4

视频采用第一人称视角拍摄，存在显著的运动模糊现象。我们将在UVO验证集上评估基于COCO数据训练的模型（数据引自表3）。

表7展示了相关结果。LW-DETR在实时检测器中表现卓越，超越了当前最先进的竞争对手。具体而言，LW-DETR-small相较于RTMDet-s、YOLOv8-s和YOLO-NAS-s中的最佳结果，mAP提升了1.3，AR提高了4.1。在召回率方面，该模型同样展现出更强的能力，能够检测到更多不同尺度的目标——小、中、大尺寸均有覆盖。上述发现表明，我们的LW-DETR之所以优于以往的实时检测器，并非因为针对COCO数据集的特定调优，而是源于其构建更具泛化能力模型的本质优势。

多领域微调。另一种方法是在不同领域的小型数据集上对预训练的检测器进行微调。RF100包含100个小型数据集、7个图像领域、224k张图片和829个类别标签，能帮助研究者用真实数据测试模型的泛化能力。我们在RF100的每个小型数据集上对实时检测器进行了微调。

结果如表8所示。LW-DETR-small在不同领域均展现出优于当前最先进实时检测器的性能。特别是在“文档”和“电磁”领域，我们的LW-DETR显著优于YOLOv5、YOLOv7、RTMDet和YOLOv8（分别比四者中的最佳表现高出5.7 AP和5.6 AP）。LW-DETR-medium则能带来更全面的性能提升。这些发现凸显了LW-DETR的多功能性，使其成为一系列封闭领域任务中的强有力基准。

5 Limitation and future works

Currently, we only demonstrate the effectiveness of LW-DETR in real-time detection. This is the first step. Extending LW-DETR for open-world detection and applying LW-DETR to more vision tasks, such as multi-person pose estimation and multi-view 3D object detection, needs more investigation. We leave them for future work.

6 Conclusion

This paper shows that detection transformers achieve competitive and even superior results over existing real-time detectors. Our method is simple and efficient. The success stems from multi-level feature aggregation and training-effective and inference-efficient techniques. We hope our experience can provide insights for building real-time models with transformers in vision tasks.

References

1. Aharon, S., Louis-Dupont, Ofri Masad, Yurkova, K., Lotem Fridman, Lkdci, Khvedchenya, E., Rubin, R., Bagrov, N., Tymchenko, B., Keren, T., Zhilko, A., Eran-Deci: Super-gradients (2021). https://doi.org/10.5281/ZENODO.7789328, https://zenodo.org/record/7789328
2. Bochkovskiy, A., Wang, C.Y., Liao, H.Y.M.: Yolov4: Optimal speed and accuracy of object detection. arXiv preprint arXiv:2004.10934 (2020)
3. Cai, Z., Liu, S., Wang, G., Ge, Z., Zhang, X., Huang, D.: Align-detr: Improving detr with simple iou-aware bce loss. arXiv preprint arXiv:2304.07527 (2023)
4. Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S.: End-to-end object detection with transformers. In: European conference on computer vision. pp. 213–229. Springer (2020)
5. Chang, J., Wang, S., Xu, H.M., Chen, Z., Yang, C., Zhao, F.: Detrdistill: A universal knowledge distillation framework for detr-families. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 6898–6908 (2023)
6. Chen, Q., Chen, X., Wang, J., Zhang, S., Yao, K., Feng, H., Han, J., Ding, E., Zeng, G., Wang, J.: Group detr: Fast detr training with group-wise one-to-many assignment. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV) (2023)
7. Chen, Q., Wang, J., Han, C., Zhang, S., Li, Z., Chen, X., Chen, J., Wang, X., Han, S., Zhang, G., et al.: Group detr v2: Strong object detector with encoder-decoder pretraining. arXiv preprint arXiv:2211.03594 (2022)
8. Chen, W., Du, X., Yang, F., Beyer, L., Zhai, X., Lin, T.Y., Chen, H., Li, J., Song, X., Wang, Z., et al.: A simple single-scale vision transformer for object localization and instance segmentation. arXiv preprint arXiv:2112.09747 (2021)
9. Chen, X., Chen, J., Liu, Y., Zeng, G.: D^3 etr: Decoder distillation for detection transformer. arXiv preprint arXiv:2211.09768 (2022)
10. Chen, X., Ding, M., Wang, X., Xin, Y., Mo, S., Wang, Y., Han, S., Luo, P., Zeng, G., Wang, J.: Context autoencoder for self-supervised representation learning. arXiv preprint arXiv:2202.03026 (2022)

5 局限性与未来工作

目前，我们仅展示了LW-DETR在实时检测中的有效性。这是第一步。将LW-DETR扩展至开放世界检测，并将其应用于更多视觉任务——如多人姿态估计与多视角3D目标检测——仍需进一步研究。这些工作将留待未来展开。

6 结论

本文表明，检测变换器在现有实时检测器基础上取得了具有竞争力甚至更优的结果。我们的方法简洁高效，其成功源于多层次特征聚合以及训练高效、推理迅捷的技术。我们期望这些经验能为视觉任务中利用变换器构建实时模型提供启示。

参考文献

1. Aharon, S., Louis-Dupont, Ofri Masad, Yurkova, K., Lotem Fridman, Lkdci, Khvedchenya, E., Rubin, R., Bagrov, N., Tymchenko, B., Keren, T., Zhilko, A., Eran-Deci: 超级梯度 (2021). https://doi.org/10.5281/ZENODO.7789328, https://zenodo.org/record/7789328 2. Bochkovskiy, A., Wang, C.Y., Liao, H.Y.M.: YOLOv4：目标检测的最佳速度与精度. arXiv预印本 arXiv:2004.10934 (2020) 3. Cai, Z., Liu, S., Wang, G., Ge, Z., Zhang, X., Huang, D.: Align-DETR：通过简单IOU感知BCE损失改进DETR. arXiv预印本 arXiv:2304.07527 (2023) 4. Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S.: 基于Transformer的端到端目标检测. 载于：欧洲计算机视觉会议. 第213–229页. Springer (2020) 5. Chang, J., Wang, S., Xu, H.M., Chen, Z., Yang, C., Zhao, F.: DETRDistill：面向DETR系列模型的通用知识蒸馏框架. 载于：IEEE/CVF国际计算机视觉会议论文集. 第6898–6908页 (2023) 6. Chen, Q., Chen, X., Wang, J., Zhang, S., Yao, K., Feng, H., Han, J., Ding, E., Zeng, G., Wang, J.: Group DETR：通过分组一对多分配实现快速DETR训练. 载于：IEEE国际计算机视觉会议(ICCV)论文集 (2023) 7. Chen, Q., Wang, J., Han, C., Zhang, S., Li, Z., Chen, X., Chen, J., Wang, X., Han, S., Zhang, G., 等: Group DETR v2：基于编码器-解码器预训练的强目标检测器. arXiv预印本 arXiv:2211.03594 (2022) 8. Chen, W., Du, X., Yang, F., Beyer, L., Zhai, X., Lin, T.Y., Chen, H., Li, J., Song, X., Wang, Z., 等: 用于目标定位与实例分割的简单单尺度视觉Transformer. arXiv预印本 arXiv:2112.09747 (2021) 9. Chen, X., Chen, J., Liu, Y., Zeng, G.: D^3ETR：检测Transformer的解码器蒸馏. arXiv预印本 arXiv:2211.09768 (2022)

10. 陈旭、丁明、王欣、辛宇、莫山、王洋、韩硕、罗平、曾刚、王杰：面向自监督表征学习的上下文自编码器。arXiv预印本 arXiv:2202.03026 (2022)

11. Chen, X., Wei, F., Zeng, G., Wang, J.: Conditional detr v2: Efficient detection transformer with box queries. arXiv preprint arXiv:2207.08914 (2022)
12. Chen, Y., Yuan, X., Wu, R., Wang, J., Hou, Q., Cheng, M.M.: Yolo-ms: Rethinking multi-scale representation learning for real-time object detection. arXiv preprint arXiv:2308.05480 (2023)
13. Ciaglia, F., Zuppichini, F.S., Guerrie, P., McQuade, M., Solawetz, J.: Roboflow 100: A rich, multi-domain object detection benchmark. arXiv preprint arXiv:2211.13523 (2022)
14. Clark, K., Luong, M.T., Le, Q.V., Manning, C.D.: Electra: Pre-training text encoders as discriminators rather than generators. arXiv preprint arXiv:2003.10555 (2020)
15. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition. pp. 248–255. Ieee (2009)
16. Ding, X., Zhang, X., Ma, N., Han, J., Ding, G., Sun, J.: Repvgg: Making vgg-style convnets great again. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 13733–13742 (2021)
17. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al.: An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929 (2020)
18. Feng, D., Haase-Schütz, C., Rosenbaum, L., Hertlein, H., Glaeser, C., Timm, F., Wiesbeck, W., Dietmayer, K.: Deep multi-modal object detection and semantic segmentation for autonomous driving: Datasets, methods, and challenges. IEEE Transactions on Intelligent Transportation Systems **22**(3), 1341–1360 (2020)
19. Gao, Z., Wang, L., Han, B., Guo, S.: Adamixer: A fast-converging query-based object detector. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5364–5373 (2022)
20. Ge, Z., Liu, S., Wang, F., Li, Z., Sun, J.: Yolox: Exceeding yolo series in 2021. arXiv preprint arXiv:2107.08430 (2021)
21. Hatamizadeh, A., Heinrich, G., Yin, H., Tao, A., Alvarez, J.M., Kautz, J., Molchanov, P.: Fastervit: Fast vision transformers with hierarchical attention. arXiv preprint arXiv:2306.06189 (2023)
22. He, K., Chen, X., Xie, S., Li, Y., Dollár, P., Girshick, R.: Masked autoencoders are scalable vision learners. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 16000–16009 (2022)
23. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
24. Hosang, J., Benenson, R., Schiele, B.: Learning non-maximum suppression. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4507–4515 (2017)
25. Hu, Y., Yang, J., Chen, L., Li, K., Sima, C., Zhu, X., Chai, S., Du, S., Lin, T., Wang, W., et al.: Planning-oriented autonomous driving. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 17853–17862 (2023)
26. Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4700–4708 (2017)

11. 陈旭、魏峰、曾光、王军：《条件DETR v2：基于框查询的高效检测Transformer》。arXiv预印本 arXiv:2207.08914（2022） 12. 陈阳、袁鑫、吴睿、王军、侯强、程明明：《YOLO-MS：重新思考实时目标检测中的多尺度表示学习》。arXiv预印本 arXiv:2308.05480（2023） 13. 西亚利亚、祖皮奇尼、盖里、麦奎德、索拉维茨：《Roboflow 100：一个丰富的多领域目标检测基准》。arXiv预印本 arXiv:2211.13523（2022） 14. 克拉克、梁明德、黎维、曼宁：《ELECTRA：以判别器而非生成器方式预训练文本编码器》。arXiv预印本 arXiv:2003.10555（2020） 15. 邓嘉、董伟、索彻、李飞飞等：《ImageNet：一个大规模分层图像数据库》。载于2009年IEEE计算机视觉与模式识别会议，第248-255页。IEEE（2009） 16. 丁霄、张翔、马宁、韩军、丁国、孙剑：《RepVGG：让VGG风格卷积网络重焕光彩》。载于IEEE/CVF计算机视觉与模式识别会议，第13733-13742页（2021） 17. 多索维茨基、拜尔等：《一幅图像值16x16个词：大规模图像识别的Transformer》。arXiv预印本 arXiv:2010.11929（2020） 18. 冯丹等：《自动驾驶中的深度多模态目标检测与语义分割：数据集、方法与挑战》。IEEE智能交通系统汇刊22(3):1341-1360（2020） 19. 高哲、王磊、韩冰、郭松：《Adamixer：快速收敛的基于查询目标检测器》。载于IEEE/CVF计算机视觉与模式识别会议，第5364-5373页（2022） 20. 葛政、刘松、王飞、李志、孙剑：《YOLOX：2021年超越YOLO系列》。arXiv预印本 arXiv:2107.08430（2021） 21. 哈塔米扎德等：《FasterViT：具有分层注意力的快速视觉Transformer》。arXiv预印本 arXiv:2306.06189（2023） 22. 何恺明、陈熙、谢赛宁、李远清、多拉尔、吉什克：《掩码自编码器是可扩展的视觉学习器》。载于IEEE/CVF计算机视觉与模式识别会议，第16000-16009页（2022） 23. 何恺明、张翔、任少卿、孙剑：《深度残差学习用于图像识别》。载于IEEE计算机视觉与模式识别会议，第770-778页（2016） 24. 霍桑、本南森、席勒：《学习非极大值抑制》。载于IEEE计算机视觉与模式识别会议，第4507-4515页（2017） 25. 胡渊等：《面向规划的自适应驾驶》。载于IEEE/CVF计算机视觉与模式识别会议，第17853-17862页（2023） 26. 黄高、刘子、范德马滕、温伯格：《密集连接卷积网络》。载于IEEE计算机视觉与模式识别会议，第4700-4708页（2017）

27. Jia, D., Yuan, Y., He, H., Wu, X., Yu, H., Lin, W., Sun, L., Zhang, C., Hu, H.: Detrs with hybrid matching. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 19702–19712 (2023)
28. Jocher, G.: Ultralytics yolov5 (2020), https://github.com/ultralytics/yolov5
29. Jocher, G., Chaurasia, A., Qiu, J.: Ultralytics yolov8 (2023), https://github.com/ultralytics/ultralytics
30. Karaoguz, H., Jensfelt, P.: Object detection approach for robot grasp detection. In: 2019 International Conference on Robotics and Automation (ICRA). pp. 4953–4959. IEEE (2019)
31. Li, B., Ouyang, W., Sheng, L., Zeng, X., Wang, X.: Gs3d: An efficient 3d object detection framework for autonomous driving. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 1019–1028 (2019)
32. Li, C., Li, L., Geng, Y., Jiang, H., Cheng, M., Zhang, B., Ke, Z., Xu, X., Chu, X.: Yolov6 v3. 0: A full-scale reloading. arXiv preprint arXiv:2301.05586 (2023)
33. Li, C., Li, L., Jiang, H., Weng, K., Geng, Y., Li, L., Ke, Z., Li, Q., Cheng, M., Nie, W., et al.: Yolov6: A single-stage object detection framework for industrial applications. arXiv preprint arXiv:2209.02976 (2022)
34. Li, F., Zeng, A., Liu, S., Zhang, H., Li, H., Zhang, L., Ni, L.M.: Lite detr: An interleaved multi-scale encoder for efficient detr. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 18558–18567 (2023)
35. Li, F., Zhang, H., Liu, S., Guo, J., Ni, L.M., Zhang, L.: Dn-detr: Accelerate detr training by introducing query denoising. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 13619–13627 (2022)
36. Li, Y., Mao, H., Girshick, R., He, K.: Exploring plain vision transformer backbones for object detection. In: European Conference on Computer Vision. pp. 280–296. Springer (2022)
37. Li, Y., Xie, S., Chen, X., Dollar, P., He, K., Girshick, R.: Benchmarking detection transfer learning with vision transformers. arXiv preprint arXiv:2111.11429 (2021)
38. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. In: Proceedings of the IEEE international conference on computer vision. pp. 2980–2988 (2017)
39. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13. pp. 740–755. Springer (2014)
40. Lin, Y., Yuan, Y., Zhang, Z., Li, C., Zheng, N., Hu, H.: Detr doesn't need multi-scale or locality design. arXiv preprint arXiv:2308.01904 (2023)
41. Liu, S., Li, F., Zhang, H., Yang, X., Qi, X., Su, H., Zhu, J., Zhang, L.: Dab-detr: Dynamic anchor boxes are better queries for detr. arXiv preprint arXiv:2201.12329 (2022)
42. Liu, S., Ren, T., Chen, J., Zeng, Z., Zhang, H., Li, F., Li, H., Huang, J., Su, H., Zhu, J., et al.: Detection transformer with stable matching. arXiv preprint arXiv:2304.04742 (2023)
43. Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B.: Swin transformer: Hierarchical vision transformer using shifted windows. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 10012–10022 (2021)
44. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101 (2017)

27. Jia, D., Yuan, Y., He, H., Wu, X., Yu, H., Lin, W., Sun, L., Zhang, C., Hu, H.: 基于混合匹配的DETR模型。见：IEEE/CVF计算机视觉与模式识别会议论文集。第19702–19712页 (2023) 28. Jocher, G.: Ultralytics YOLOv5 (2020), https://github.com/ultralytics/yolov5 29. Jocher, G., Chaurasia, A., Qiu, J.: Ultralytics YOLOv8 (2023), https://github.com/ultralytics/ultralytics 30. Karaoguz, H., Jensfelt, P.: 机器人抓取检测的物体检测方法。见：2019年机器人与自动化国际会议(ICRA)。第4953–4959页。IEEE (2019) 31. Li, B., Ouyang, W., Sheng, L., Zeng, X., Wang, X.: GS3D：面向自动驾驶的高效3D物体检测框架。见：IEEE/CVF计算机视觉与模式识别会议论文集。第1019–1028页 (2019) 32. Li, C., Li, L., Geng, Y., Jiang, H., Cheng, M., Zhang, B., Ke, Z., Xu, X., Chu, X.: YOLOv6 v3.0：全面升级版。arXiv预印本 arXiv:2301.05586 (2023) 33. Li, C., Li, L., Jiang, H., Weng, K., Geng, Y., Li, L., Ke, Z., Li, Q., Cheng, M., Nie, W., 等：YOLOv6：面向工业应用的单阶段物体检测框架。arXiv预印本 arXiv:2209.02976 (2022) 34. Li, F., Zeng, A., Liu, S., Zhang, H., Li, H., Zhang, L., Ni, L.M.: Lite DETR：交错多尺度编码器实现高效DETR。见：IEEE/CVF计算机视觉与模式识别会议论文集。第18558–18567页 (2023) 35. Li, F., Zhang, H., Liu, S., Guo, J., Ni, L.M., Zhang, L.: DN-DETR：通过查询去噪加速DETR训练。见：IEEE/CVF计算机视觉与模式识别会议论文集。第13619–13627页 (2022) 36. Li, Y., Mao, H., Girshick, R., He, K.: 探索朴素视觉Transformer骨干网络在物体检测中的应用。见：欧洲计算机视觉会议。第280–296页。Springer (2022) 37. Li, Y., Xie, S., Chen, X., Dollar, P., He, K., Girshick, R.: 基于视觉Transformer的检测迁移学习基准测试。arXiv预印本 arXiv:2111.11429 (2021) 38. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P.: 密集物体检测的焦点损失函数。见：IEEE国际计算机视觉会议论文集。第2980–2988页 (2017) 39. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft COCO：上下文中的常见物体。见：计算机视觉–ECCV 2014：第13届欧洲计算机视觉会议，瑞士苏黎世，2014年9月6-12日，第五部分13。第740–755页。Springer (2014) 40. Lin, Y., Yuan, Y., Zhang, Z., Li, C., Zheng, N., Hu, H.: DETR无需多尺度或局部性设计。arXiv预印本 arXiv:2308.01904 (2023) 41. Liu, S., Li, F., Zhang, H., Yang, X., Qi, X., Su, H., Zhu, J., Zhang, L.: DAB-DETR：动态锚框作为DETR的更优查询。arXiv预印本 arXiv:2201.12329 (2022) 42. Liu, S., Ren, T., Chen, J., Zeng, Z., Zhang, H., Li, F., Li, H., Huang, J., Su, H., Zhu, J., 等：具有稳定匹配的检测Transformer。arXiv预印本 arXiv:2304.04742 (2023) 43. Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B.: Swin Transformer：基于移位窗口的分层视觉Transformer。见：IEEE/CVF国际计算机视觉会议论文集。第10012–10022页 (2021) 44. Loshchilov, I., Hutter, F.: 解耦权重衰减正则化。arXiv预印本 arXiv:1711.05101 (2017)

45. Lv, W., Xu, S., Zhao, Y., Wang, G., Wei, J., Cui, C., Du, Y., Dang, Q., Liu, Y.: Detrs beat yolos on real-time object detection. arXiv preprint arXiv:2304.08069 (2023)
46. Lyu, C., Zhang, W., Huang, H., Zhou, Y., Wang, Y., Liu, Y., Zhang, S., Chen, K.: Rtmdet: An empirical study of designing real-time object detectors. arXiv preprint arXiv:2212.07784 (2022)
47. Meng, D., Chen, X., Fan, Z., Zeng, G., Li, H., Yuan, Y., Sun, L., Wang, J.: Conditional detr for fast training convergence. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 3651–3660 (2021)
48. Ouyang-Zhang, J., Cho, J.H., Zhou, X., Krähenbühl, P.: Nms strikes back. arXiv preprint arXiv:2212.06137 (2022)
49. Paul, S.K., Chowdhury, M.T., Nicolescu, M., Nicolescu, M., Feil-Seifer, D.: Object detection and pose estimation from rgb and depth data for real-time, adaptive robotic grasping. In: Advances in Computer Vision and Computational Biology: Proceedings from IPCV'20, HIMS'20, BIOCOMP'20, and BIOENG'20, pp. 121–142. Springer (2021)
50. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 779–788 (2016)
51. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. Advances in neural information processing systems **28** (2015)
52. Rezatofighi, H., Tsoi, N., Gwak, J., Sadeghian, A., Reid, I., Savarese, S.: 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. pp. 658–666 (2019)
53. Roh, B., Shin, J., Shin, W., Kim, S.: Sparse detr: Efficient end-to-end object detection with learnable sparsity. arXiv preprint arXiv:2111.14330 (2021)
54. Shao, S., Li, Z., Zhang, T., Peng, C., Yu, G., Zhang, X., Li, J., Sun, J.: Objects365: A large-scale, high-quality dataset for object detection. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 8430–8439 (2019)
55. Tarvainen, A., Valpola, H.: Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. Advances in neural information processing systems **30** (2017)
56. Tian, Z., Shen, C., Chen, H., He, T.: Fcos: Fully convolutional one-stage object detection. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 9627–9636 (2019)
57. Wang, A., Chen, H., Liu, L., Chen, K., Lin, Z., Han, J., Ding, G.: Yolov10: Real-time end-to-end object detection. arXiv preprint arXiv:2405.14458 (2024)
58. Wang, C., He, W., Nie, Y., Guo, J., Liu, C., Han, K., Wang, Y.: Gold-yolo: Efficient object detector via gather-and-distribute mechanism. arXiv preprint arXiv:2309.11331 (2023)
59. Wang, C.Y., Bochkovskiy, A., Liao, H.Y.M.: 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. pp. 7464–7475 (2023)
60. Wang, C.Y., Liao, H.Y.M., Wu, Y.H., Chen, P.Y., Hsieh, J.W., Yeh, I.H.: 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. pp. 390–391 (2020)

45. 吕文、徐松、赵阳、王刚、魏杰、崔晨、杜洋、党强、刘洋：DETR在实时目标检测上超越YOLO。arXiv预印本 arXiv:2304.08069 (2023) 46. 吕超、张伟、黄浩、周洋、王毅、刘洋、张帅、陈凯：RTMDet：实时目标检测器设计的实证研究。arXiv预印本 arXiv:2212.07784 (2022) 47. 孟迪、陈星、范哲、曾光、李航、袁野、孙磊、王军：Conditional DETR加速训练收敛。载于：IEEE/CVF国际计算机视觉会议论文集。第3651–3660页 (2021) 48. 欧阳-张杰、赵俊浩、周鑫、菲利普·克伦布尔：NMS强势回归。arXiv预印本 arXiv:2212.06137 (2022) 49. 保罗·S·K、乔杜里·M·T、尼科列斯库·M、尼科列斯库·M、费尔-赛弗·D：基于RGB与深度数据的实时自适应机器人抓取目标检测与姿态估计。载于：《计算机视觉与计算生物学进展：IPCV’20, HIMS’20, BIOCOMP’20, BIOENG’20会议论文集》，第121–142页。Springer (2021) 50. 雷德蒙·J、迪瓦拉·S、吉尔希克·R、法尔哈迪·A：YOLO：统一实时目标检测。载于：IEEE计算机视觉与模式识别会议论文集。第779–788页 (2016) 51. 任少卿、何恺明、吉尔希克·R、孙剑：Faster R-CNN：基于区域提议网络的实时目标检测。《神经信息处理系统进展》第28卷 (2015) 52. 雷扎托菲·H、蔡尼、郭宰杰、萨德吉安·A、里德·I、萨瓦雷斯·S：广义交并比：边界框回归的度量与损失函数。载于：IEEE/CVF计算机视觉与模式识别会议论文集。第658–666页 (2019) 53. 卢炳、申俊、申宇、金晟：Sparse DETR：可学习稀疏性的高效端到端目标检测。arXiv预印本 arXiv:2111.14330 (2021) 54. 邵帅、李哲、张涛、彭超、余刚、张旭、李军、孙剑：Objects365：大规模高质量目标检测数据集。载于：IEEE/CVF国际计算机视觉会议论文集。第8430–8439页 (2019) 55. 塔尔维宁·A、瓦尔波拉·H：均值教师是更好的榜样：权重平均一致性目标提升半监督深度学习效果。《神经信息处理系统进展》第30卷 (2017) 56. 田震、沈超、陈浩、何涛：FCOS：全卷积单阶段目标检测。载于：IEEE/CVF国际计算机视觉会议论文集。第9627–9636页 (2019) 57. 王安、陈浩、刘磊、陈凯、林哲、韩杰、丁刚：YOLOv10：实时端到端目标检测。arXiv预印本 arXiv:2405.14458 (2024) 58. 王超、何伟、聂阳、郭佳、刘畅、韩凯、王洋：Gold-YOLO：基于聚集-分发机制的高效目标检测器。arXiv预印本 arXiv:2309.11331 (2023) 59. 王昌宇、博奇科夫斯基·A、廖宏毅明：YOLOv7：可训练免费礼包为实时检测器设新标杆。载于：IEEE/CVF计算机视觉与模式识别会议论文集。第7464–7475页 (2023) 60. 王昌宇、廖宏毅明、吴宇航、陈培元、谢政文、叶奕宏：CSPNet：增强CNN学习能力的新骨干网络。载于：IEEE/CVF计算机视觉与模式识别研讨会论文集。第390–391页 (2020)

61. Wang, W., Feiszli, M., Wang, H., Tran, D.: Unidentified video objects: A benchmark for dense, open-world segmentation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 10776–10785 (2021)
62. Wang, W., Xie, E., Li, X., Fan, D.P., Song, K., Liang, D., Lu, T., Luo, P., Shao, L.: Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 568–578 (2021)
63. Wang, Y., Zhang, X., Yang, T., Sun, J.: Anchor detr: Query design for transformer-based detector. In: Proceedings of the AAAI conference on artificial intelligence. vol. 36, pp. 2567–2575 (2022)
64. Wang, Y., Li, X., Wen, S., Yang, F., Zhang, W., Zhang, G., Feng, H., Han, J., Ding, E.: Knowledge distillation for detection transformer with consistent distillation points sampling. arXiv preprint arXiv:2211.08071 (2022)
65. Xu, S., Wang, X., Lv, W., Chang, Q., Cui, C., Deng, K., Wang, G., Dang, Q., Wei, S., Du, Y., et al.: Pp-yoloe: An evolved version of yolo. arXiv preprint arXiv:2203.16250 (2022)
66. Zhai, X., Kolesnikov, A., Houlsby, N., Beyer, L.: Scaling vision transformers. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 12104–12113 (2022)
67. Zhang, H., Li, F., Liu, S., Zhang, L., Su, H., Zhu, J., Ni, L.M., Shum, H.Y.: Dino: Detr with improved denoising anchor boxes for end-to-end object detection. arXiv preprint arXiv:2203.03605 (2022)
68. Zhang, H., Wang, Y., Dayoub, F., Sunderhauf, N.: Varifocalnet: An iou-aware dense object detector. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 8514–8523 (2021)
69. Zhang, H., Cisse, M., Dauphin, Y.N., Lopez-Paz, D.: mixup: Beyond empirical risk minimization. arXiv preprint arXiv:1710.09412 (2017)
70. Zhang, X., Tian, Y., Huang, W., Ye, Q., Dai, Q., Xie, L., Tian, Q.: Hivit: Hierarchical vision transformer meets masked image modeling. arXiv preprint arXiv:2205.14949 (2022)
71. Zhang, X., Chen, J., Yuan, J., Chen, Q., Wang, J., Wang, X., Han, S., Chen, X., Pi, J., Yao, K., Han, J., Ding, E., Wang, J.: CAE v2: Context autoencoder with CLIP latent alignment. Transactions on Machine Learning Research (2023)
72. Zheng, D., Dong, W., Hu, H., Chen, X., Wang, Y.: Less is more: Focus attention for efficient detr. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 6674–6683 (2023)
73. Zheng, Z., Wang, P., Liu, W., Li, J., Ye, R., Ren, D.: Distance-iou loss: Faster and better learning for bounding box regression. In: Proceedings of the AAAI conference on artificial intelligence. vol. 34, pp. 12993–13000 (2020)
74. Zhu, X., Su, W., Lu, L., Li, B., Wang, X., Dai, J.: Deformable detr: Deformable transformers for end-to-end object detection. arXiv preprint arXiv:2010.04159 (2020)
75. Zong, Z., Song, G., Liu, Y.: Detrs with collaborative hybrid assignments training. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 6748–6758 (2023)

61. 王伟、Feiszli, M.、王浩、Tran, D.：《未识别视频对象：密集开放世界分割的基准》。载于：IEEE/CVF国际计算机视觉会议论文集。第10776–10785页（2021年） 62. 王伟、谢恩、李翔、范东平、宋凯、梁栋、卢涛、罗平、邵立：《金字塔视觉Transformer：无需卷积的密集预测通用骨干网络》。载于：IEEE/CVF国际计算机视觉会议论文集。第568–578页（2021年） 63. 王洋、张鑫、杨涛、孙剑：《Anchor DETR：基于Transformer检测器的查询设计》。载于：AAAI人工智能会议论文集。第36卷，第2567–2575页（2022年） 64. 王洋、李翔、温帅、杨帆、张伟、张刚、冯浩、韩军、丁二：《检测Transformer的知识蒸馏：基于一致蒸馏点采样的方法》。arXiv预印本 arXiv:2211.08071（2022年） 65. 徐松、王旭、吕文、常青、崔灿、邓凯、王刚、党强、魏松、杜阳等：《PP-YOLOE：YOLO的进化版本》。arXiv预印本 arXiv:2203.16250（2022年） 66. 翟翔、Kolesnikov, A.、Houlsby, N.、Beyer, L.：《视觉Transformer的规模化研究》。载于：IEEE/CVF计算机视觉与模式识别会议论文集。第12104–12113页（2022年） 67. 张浩、李飞、刘松、张林、苏航、朱军、倪明、沈向洋：《DINO：改进去噪锚框的端到端目标检测DETR》。arXiv预印本 arXiv:2203.03605（2022年） 68. 张浩、王洋、Dayoub, F.、Sunderhauf, N.：《VarifocalNet：基于IOU感知的密集目标检测器》。载于：IEEE/CVF计算机视觉与模式识别会议论文集。第8514–8523页（2021年） 69. 张浩、Cisse, M.、Dauphin, Y.N.、Lopez-Paz, D.：《mixup：超越经验风险最小化》。arXiv预印本 arXiv:1710.09412（2017年） 70. 张鑫、田雨、黄伟、叶强、戴强、谢乐、田琪：《HiViT：分层视觉Transformer与掩码图像建模的结合》。arXiv预印本 arXiv:2205.14949（2022年） 71. 张鑫、陈杰、袁健、陈强、王军、王旭、韩松、陈翔、皮杰、姚凯、韩军、丁二、王军：《CAE v2：基于CLIP潜在对齐的上下文自编码器》。载于：机器学习研究汇刊（2023年） 72. 郑东、董伟、胡浩、陈翔、王洋：《少即是多：高效DETR的焦点注意力机制》。载于：IEEE/CVF国际计算机视觉会议论文集。第6674–6683页（2023年） 73. 郑哲、王鹏、刘伟、李杰、叶荣、任栋：《Distance-IOU损失：更快更好的边界框回归学习》。载于：AAAI人工智能会议论文集。第34卷，第12993–13000页（2020年） 74. 朱翔、苏文、卢林、李波、王旭、戴杰：《Deformable DETR：端到端目标检测的可变形Transformer》。arXiv预印本 arXiv:2010.04159（2020年） 75. 宗哲、宋刚、刘洋：《协作混合分配训练的DETRs》。载于：IEEE/CVF国际计算机视觉会议论文集。第6748–6758页（2023年）

Supplementary Material

A Experimental Details

This section includes details on the hyper-parameters of pretraining on Objects365 [54], finetuning on COCO [39], and finetuning on Roboflow 100 [13], on the architectures of convolutional encoders, and on the modeling of box regression. We represent the `tiny`/`small`/`medium`/`large`/`xlarge` versions of our LW-DETR with T/S/M/L/X in the tables for neat representations.

A.1 Experimental settings

Pretraining settings. The default settings are in Table 9. We do not use the learning rate drop schedule and keep the initial learning rate along with the training process. When performing window attention in the ViT encoder, we fix the number of windows as 16 for different image resolutions for easy implementation. We use layer-wise lr decay [14] following previous MIM methods [10,22,71].

Table 9: Pretraining settings.

Setting	Value
optimizer	AdamW
base learning rate	$4.0 \times e^{-4}$
encoder learning rate	$6.0 \times e^{-4}$
weight decay	$1 \times e^{-4}$
batch size	128
epochs	30
training images resolutions	[448, 512, 576, 640, 704, 768, 832, 896]
encoder layer-wise lr decay	0.8 (T/S), 0.7 (M/L), 0.75 (X)
number of encoder layers	6 (T), 10(S/M/L/X)
drop path	0 (T/S/M), 0.05 (L/X)
window numbers	16
window attention indexes	[0, 2, 4] (T), [0, 1, 3, 6, 7, 9] (S/M/L/X)
output feature indexes	[0, 2, 4] (T), [2, 4, 5, 9] (S/M/L/X)
feature scales	$\frac{1}{16}$ (T/S/M), [$\frac{1}{8}$, $\frac{1}{32}$] (L/X)
number of object queries	100 (T), 300 (S/M/L/X)
number of decoder layers	3
hidden dimensions	256 (T/S/M), 384 (L/X)
decoder self-attention heads	8 (T/S/M), 12 (L/X)
decoder cross-attention heads	16 (T/S/M), 24 (L/X)
decoder sampling points	2 (T/S/M), 4 (L/X)
group detr	13
ema decay	0.997

COCO experimental settings. Most of the settings follow the ones in the pretraining stage. We share the modifications of settings in Table 10. When finetuning LW-DETR on COCO, we use component-wise lr decay [7], which gives different scale factors for the learning rate in the ViT encoder, the Projector, and the DETR decoder. For example, the component-wise lr decay is 0.7 means that we set the lr scale factor as 0.7^0 for the prediction heads, 0.7^1 for the transformer decoder layers in DETR decoder, 0.7^2 for the Projector, and 0.7^3 for the ViT encoder.

补充材料

A 实验细节

本节详细介绍了在Objects365 [54]上进行预训练、在COCO [39]上进行微调、在Roboflow 100 [13]上进行微调的超参数设置，卷积编码器的架构设计，以及边界框回归的建模方法。为简洁起见，我们在表格中用T/S/M/L/X分别代表LW-DETR的微型/小型/中型/大型/超大型版本。

A.1 实验设置

预训练设置。默认设置如表9所示。我们不采用学习率下降调度，而是保持初始学习率贯穿整个训练过程。在ViT编码器中执行窗口注意力时，为便于实现，我们将不同图像分辨率下的窗口数量固定为16。遵循先前MIM方法[10,22,71]的做法，我们采用了分层学习率衰减[14]。

表9：预训练设置。

Setting	Value
optimizer	AdamW
base learning rate	$4.0 \times e^{-4}$
encoder learning rate	$6.0 \times e^{-4}$
weight decay	$1 \times e^{-4}$
batch size	128
epochs	30
training images resolutions	[448, 512, 576, 640, 704, 768, 832, 896]
encoder layer-wise lr decay	0.8 (T/S), 0.7 (M/L), 0.75 (X)
number of encoder layers	6 (T), 10(S/M/L/X)
drop path	0 (T/S/M), 0.05 (L/X)
window numbers	16
window attention indexes	[0, 2, 4] (T), [0, 1, 3, 6, 7, 9] (S/M/L/X)
output feature indexes	[0, 2, 4] (T), [2, 4, 5, 9] (S/M/L/X)
feature scales	$\frac{1}{16}$ (T/S/M), [$\frac{1}{8}$, $\frac{1}{32}$] (L/X)
number of object queries	100 (T), 300 (S/M/L/X)
number of decoder layers	3
hidden dimensions	256 (T/S/M), 384 (L/X)
decoder self-attention heads	8 (T/S/M), 12 (L/X)
decoder cross-attention heads	16 (T/S/M), 24 (L/X)
decoder sampling points	2 (T/S/M), 4 (L/X)
group detr	13
ema decay	0.997

COCO实验设置。大部分设置遵循预训练阶段的配置。我们在表10中分享了设置的修改内容。在COCO上微调LW-DETR时，我们采用了组件级学习率衰减[7]，即为ViT编码器、投影器和DETR解码器中的学习率分配不同的比例因子。例如，组件级学习率衰减为0.7表示：我们将预测头的学习率比例因子设为0.7^0，DETR解码器中Transformer解码层设为0.7^1，投影器设为0.7^2，ViT编码器设为0.7^3。

Table 10: COCO experimental settings.

Setting	Value
base learning rate	$1.0 \times e^{-4}$
encoder learning rate	$1.5 \times e^{-4}$
weight decay	$1 \times e^{-4}$ (T/S/M/L), $1 \times e^{-3}$ (X)
batch size	32 (T/S/M), 16 (L/X)
epochs	50 (T/S/M), 25 (L/X)
drop path	0 (T/S/M), 0.1 (L/X)
component-wise lr decay	0.7 (T/S/M), 0.5 (L/X)

Table 11: Roboflow 100 experimental settings.

Setting	Value
base learning rate	$8.0 \times e^{-4}$ (S), $3.0 \times e^{-4}$ (M)
batch size	16 (S/M)
encoder learning rate	$1.2 \times e^{-3}$ (S), $4.5 \times e^{-4}$ (M)
encoder layer-wise lr decay	0.9 (S), 0.8 (M)
component-wise lr decay	0.7 (S), 0.9 (M)

Roboflow 100 experimental settings. Roboflow 100 [13] consists of 100 small datasets. We finetune our LW-DETR on these datasets based on the pretrained model on Objects365 [54]. As the training images are insufficient, we set the batch size as 16 and finetune the model for 100 epochs following [13] on all small datasets to make sure to have sufficient training iterations.

We tune the learning rate, encoder layer-wise lr decay, and component-wise lr decay (as shown in Table 11), we do a coarse search on the 'microscopic' domain and fix these hyper-parameters for other datasets. The other hyper-parameters are kept same with the finetuning experiments on COCO. We also perform the same processes for RTMDet [46] and YOLOv8 [29] for fair comparisons.

A.2 Settings for convolutional encoders

We also explore convolutional encoders, ResNet-18 and ResNet-50, in our LW-DETR. We load the ImageNet [15] pretrained encoder weights from the RT-DETR repo[7]. Instead of directly outputting multi-level feature maps with scales of $[\frac{1}{8}, \frac{1}{16}, \frac{1}{32}]$, we make a simple modification to only output a feature map in $\frac{1}{16}$. We first upsample the feature map in $\frac{1}{32}$ scale to $\frac{1}{16}$, downsample the feature map in $\frac{1}{8}$ to $\frac{1}{16}$, and then concatenates all the feature maps. We add additional convolution layers to reduce the feature dimension to prevent the final concatenated feature map from having extremely large feature dimensions.

A.3 Box regression target reparameterization

Box regression target parameterization is a widely used technique in two-stage and one-stage detectors [2, 28, 38, 51], which predicts the parameters for a box transformation that transforms an input proposal into a predicted box. We follow Plain DETR [40] to use this technique in our LW-DETR.

For box regression in the first stage and each decoder layer, we predict four parameters $[\delta_x, \delta_y, \delta_w, \delta_h]$ to a transformation, which transforms a proposal

[7] https://github.com/lyuwenyu/RT-DETR/issues/42#issue-1860463373

表10：COCO实验设置。

Setting	Value
base learning rate	$1.0 \times e^{-4}$
encoder learning rate	$1.5 \times e^{-4}$
weight decay	$1 \times e^{-4}$ (T/S/M/L), $1 \times e^{-3}$ (X)
batch size	32 (T/S/M), 16 (L/X)
epochs	50 (T/S/M), 25 (L/X)
drop path	0 (T/S/M), 0.1 (L/X)
component-wise lr decay	0.7 (T/S/M), 0.5 (L/X)

表11：Roboflow 100实验设置。

Setting	Value
base learning rate	$8.0 \times e^{-4}$ (S), $3.0 \times e^{-4}$ (M)
batch size	16 (S/M)
encoder learning rate	$1.2 \times e^{-3}$ (S), $4.5 \times e^{-4}$ (M)
encoder layer-wise lr decay	0.9 (S), 0.8 (M)
component-wise lr decay	0.7 (S), 0.9 (M)

Roboflow 100实验设置。Roboflow 100 [13]包含100个小型数据集。我们基于在Objects365 [54]上预训练的模型，在这些数据集上对LW-DETR进行微调。由于训练图像数量有限，我们按照[13]的方法，在所有小型数据集上将批量大小设为16，并微调模型100个周期，以确保有足够的训练迭代次数。

我们调整了学习率、编码器逐层学习率衰减以及组件级学习率衰减（如表11所示），在“微观”领域进行了粗粒度搜索，并为其他数据集固定这些超参数。其余超参数与在COCO上的微调实验保持一致。为了公平比较，我们对RTMDet[46]和YOLOv8[29]也执行了相同流程。

A.2 卷积编码器设置

我们还在LW-DETR中探索了卷积编码器、ResNet-18和ResNet-50的应用。我们从RT-DETR代码库[7]加载了ImageNet[15]预训练的编码器权重。不同于直接输出具有[$\frac{1}{8}, \frac{1}{16}, \frac{1}{32}$]尺度的多级特征图，我们进行了简单修改，仅输出$\frac{1}{16}$尺度的特征图。首先将$\frac{1}{32}$尺度的特征图上采样至$\frac{1}{16}$，将$\frac{1}{8}$尺度的特征图下采样至$\frac{1}{16}$，随后将所有特征图进行拼接。我们额外添加了卷积层以降低特征维度，避免最终拼接后的特征图维度过于庞大。

A.3 边界框回归目标重参数化

边界框回归目标参数化是两阶段和单阶段检测器中广泛采用的技术[2, 28, 38, 51]，其通过预测框变换参数，将输入提案转换为预测框。我们遵循Plain DETR[40]的做法，在LW-DETR中应用了该技术。

在第一阶段及每个解码器层的边界框回归中，我们预测四个参数[$\delta_x, \delta_y, \delta_w, \delta_h$]以构建一个变换，该变换作用于候选框

[7] https://github.com/lyuwenyu/RT-DETR/issues/42#issue-1860463373

Table 12: Tuning score threshold for non-end-to-end detectors. We show how the score threshold affects the time of NMS and the detection performance in YOLO-NAS, YOLOv8, RTMDet, YOLO-MS, and Gold-YOLO. We share the detection performance and total latency under three different score thresholds. The first score threshold is the default one in the official implementations. The score threshold in bold represents a good balance between the mAP score and the NMS latency.

Model	Model Latency (ms)	mAP	Total Latency (ms)	mAP	Total Latency (ms)	mAP	Total Latency (ms)
Thresholds		score = 0.01[8]		**score =0.1**		score = 0.15	
YOLO-NAS-s	2.75	47.6	4.68	47.3	2.88	46.7	2.82
YOLO-NAS-m	5.52	51.6	7.76	51.1	5.70	50.6	5.53
YOLO-NAS-l	7.49	52.3	8.84	51.9	7.64	51.2	7.52
Thresholds		score = 0.001		**score =0.01**		score = 0.05	
YOLOv8n	1.51	37.4	6.21	37.3	1.62	36.0	1.54
YOLOv8s	2.64	45.0	7.00	44.8	2.71	43.7	2.70
YOLOv8m	5.90	50.3	10.11	50.0	6.06	49.0	5.92
YOLOv8l	9.30	53.0	13.16	52.5	9.44	51.3	9.31
YOLOv8x	14.88	54.0	19.17	53.5	15.09	52.3	14.91
Thresholds		score = 0.001		**score =0.1**		score = 0.25	
RTMDet-t	2.16	41.0	7.41	40.8	2.45	39.1	2.37
RTMDet-s	2.88	44.6	7.88	44.4	2.94	42.6	2.93
RTMDet-m	6.27	49.3	10.82	49.1	6.52	47.2	6.31
RTMDet-l	10.37	51.4	14.84	51.2	10.54	49.2	10.48
RTMDet-x	18.44	52.8	22.81	52.5	18.88	50.5	18.69
Thresholds		score = 0.001		**score =0.1**		score = 0.25	
YOLO-MS-XS	3.02	43.4	6.99	43.3	3.26	41.9	3.19
YOLO-MS-S	5.40	46.2	9.18	46.1	5.62	44.2	5.56
YOLO-MS	8.56	51.0	12.38	50.8	9.09	48.8	8.87
Thresholds		score = 0.03		**score =0.05**		score = 0.25	
Gold-YOLO-S	2.94	45.5	3.63	45.4	3.35	43.1	3.08
Gold-YOLO-M	5.84	50.2	6.34	50.2	6.14	47.6	5.98
Gold-YOLO-L	10.15	52.3	10.58	52.2	10.46	50.5	10.22

$[p_{c_x}, p_{c_y}, p_w, p_h]$ to a predicted bounding box $[b_{c_x}, b_{c_y}, b_w, b_h]$ by applying:

$$\begin{aligned} b_{c_x} = \delta_x * p_w + p_{c_x}, b_{c_y} = \delta_y * p_h + p_{c_y}, \\ b_w = \exp(\delta_w) * p_w, b_h = \exp(\delta_h) * p_h. \end{aligned} \tag{6}$$

The predicted box $[b_{c_x}, b_{c_y}, b_w, b_h]$ is used for calculating the box regression losses and for output.

B Analysis on NMS

Tuning score threshold. The score threshold in non-end-to-end detectors, decides the number of predicted boxes that are passed to the NMS, which largely affects the NMS latency. Table 12 verifies it with YOLO-NAS, YOLOv8, RTMDet, YOLO-MS, and Gold-YOLO. A large score threshold can largely reduce the overhead of NMS, but bring negative results to detection performance. We optimize the NMS latency by carefully tuning the score thresholds for non-end-to-end detectors, achieving a balance between the detection performance and the

[8] We have corrected a typo in the main paper regarding YOLO-NAS: the default score threshold should be 0.01, not the value mentioned in L298.

表12：非端到端检测器的分数阈值调优。我们展示了在YOLO-NAS、YOLOv8、RTMDet、YOLO-MS和Gold-YOLO中，分数阈值如何影响NMS处理时间与检测性能。我们分享了三种不同分数阈值下的检测性能与总延迟数据。首个分数阈值采用各官方实现的默认设置，加粗标注的阈值则代表mAP分数与NMS延迟之间的较优平衡点。

Model	Model Latency (ms)	mAP	Total Latency (ms)	mAP	Total Latency (ms)	mAP	Total Latency (ms)
Thresholds		score = 0.01[8]		**score =0.1**		score = 0.15	
YOLO-NAS-s	2.75	47.6	4.68	47.3	2.88	46.7	2.82
YOLO-NAS-m	5.52	51.6	7.76	51.1	5.70	50.6	5.53
YOLO-NAS-l	7.49	52.3	8.84	51.9	7.64	51.2	7.52
Thresholds		score = 0.001		**score =0.01**		score = 0.05	
YOLOv8n	1.51	37.4	6.21	37.3	1.62	36.0	1.54
YOLOv8s	2.64	45.0	7.00	44.8	2.71	43.7	2.70
YOLOv8m	5.90	50.3	10.11	50.0	6.06	49.0	5.92
YOLOv8l	9.30	53.0	13.16	52.5	9.44	51.3	9.31
YOLOv8x	14.88	54.0	19.17	53.5	15.09	52.3	14.91
Thresholds		score = 0.001		**score =0.1**		score = 0.25	
RTMDet-t	2.16	41.0	7.41	40.8	2.45	39.1	2.37
RTMDet-s	2.88	44.6	7.88	44.4	2.94	42.6	2.93
RTMDet-m	6.27	49.3	10.82	49.1	6.52	47.2	6.31
RTMDet-l	10.37	51.4	14.84	51.2	10.54	49.2	10.48
RTMDet-x	18.44	52.8	22.81	52.5	18.88	50.5	18.69
Thresholds		score = 0.001		**score =0.1**		score = 0.25	
YOLO-MS-XS	3.02	43.4	6.99	43.3	3.26	41.9	3.19
YOLO-MS-S	5.40	46.2	9.18	46.1	5.62	44.2	5.56
YOLO-MS	8.56	51.0	12.38	50.8	9.09	48.8	8.87
Thresholds		score = 0.03		**score =0.05**		score = 0.25	
Gold-YOLO-S	2.94	45.5	3.63	45.4	3.35	43.1	3.08
Gold-YOLO-M	5.84	50.2	6.34	50.2	6.14	47.6	5.98
Gold-YOLO-L	10.15	52.3	10.58	52.2	10.46	50.5	10.22

通过应用以下方法，将[$p_{c_x}, p_{c_y}, p_w, p_h$]转换为预测边界框[$b_{c_x}, b_{c_y}, b_w, b_h$]:

$$\begin{aligned} b_{c_x} = \delta_x * p_w + p_{c_x}, b_{c_y} = \delta_y * p_h + p_{c_y}, \\ b_w = \exp(\delta_w) * p_w, b_h = \exp(\delta_h) * p_h. \end{aligned} \tag{6}$$

预测框[$b_{c_x}, b_{c_y}, b_w, b_h$]用于计算边界框回归损失以及输出。

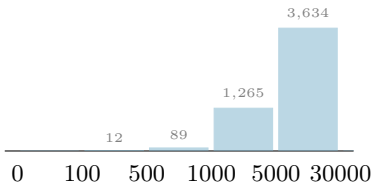
B NMS分析

调整得分阈值。非端到端检测器中的得分阈值决定了传递给非极大值抑制（NMS）的预测框数量，这极大影响了NMS的延迟。表12通过YOLO-NAS、YOLOv8、RT-MDet、YOLO-MS和Gold-YOLO验证了这一点。较高的得分阈值能大幅降低NMS开销，但会对检测性能产生负面影响。我们通过精细调节非端到端检测器的得分阈值来优化NMS延迟，在检测性能与{v*}之间取得平衡。

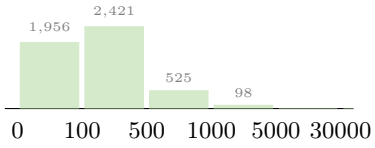
8 我们已更正主论文中关于YOLO-NAS的一个笔误：默认分数阈值应为0.01，而非L298中提到的数值。

total latency. The overhead brought by NMS is significantly reduced to 0.1∼0.5 ms with slight drops in detection performance.

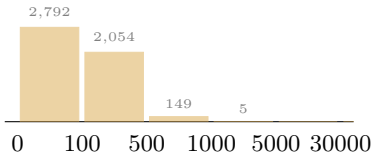
Distribution of the number of boxes for NMS. The latency is measured on the COCO `val2017`, which is an average of 5000 images. Figure 5 gives the distribution of the number of remaining boxes in NMS across the COCO `val2017` under different score thresholds in YOLO-NAS. The large overhead brought by the NMS is due to the large number of remaining boxes under the default score threshold. Tuning the score threshold effectively decreases the remaining boxes in NMS, thus providing optimization in total latency for non-end-to-end detectors.



(a) YOLO-NAS-s score threshold = 0.01



(b) YOLO-NAS-s score threshold = 0.1

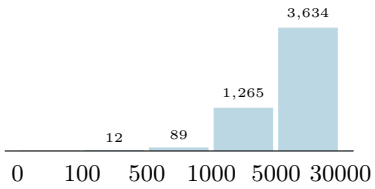


(c) YOLO-NAS-s score threshold = 0.15

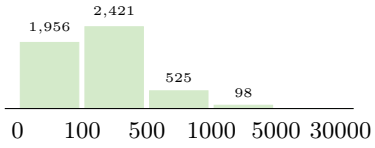
Fig. 5: Distribution of the number of boxes. The x-axis corresponds to the number of boxes that are fed into NMS. The y-axis corresponds to the number of images on COCO `val2017` whose remaining box numbers are in the corresponding interval. (a) is under the default score threshold. (b) is tuning the score threshold to get the balance between detection performance and latency. (c) is tuning a higher score threshold.

总延迟。NMS带来的开销显著降低至0.1∼0.5毫秒，同时检测性能仅有轻微下降。

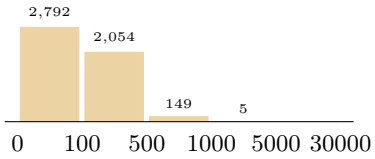
NMS处理中保留框数量的分布。延迟数据是在COCO val2017数据集上测量的，取5000张图像的平均值。图5展示了YOLO-NAS模型在不同分数阈值下，COCO val2017数据集中NMS处理后剩余框数量的分布情况。NMS带来的较大开销源于默认分数阈值下保留的框数量过多。通过调整分数阈值，可有效减少NMS中的保留框数量，从而为非端到端检测器的整体延迟提供优化空间。



(a) YOLO-NAS-s score threshold = 0.01



(b) YOLO-NAS-s score threshold = 0.1



(c) YOLO-NAS-s score threshold = 0.15

图5：框数量的分布。x轴对应输入至非极大值抑制（NMS）的框数量，y轴对应COCO val2017数据集中剩余框数量落在相应区间内的图像数量。(a)为默认分数阈值下的情况。(b)通过调整分数阈值以权衡检测性能与延迟。(c)则采用更高的分数阈值进行调整。