

# D-FINE: 将DETRS中的回归任务重新定义为细粒度分布优化

彭彦松<sup>1</sup>、李河北<sup>1</sup>、吴培玺<sup>1</sup>、张悦怡<sup>1,\*</sup>、孙晓燕<sup>1,2\*</sup>、吴枫<sup>1,2</sup> <sup>1</sup>中国科学技术大学  
<sup>2</sup>合肥综合性国家科学中心人工智能研究院 {pengyansong, lihebei, wupeixi}@mail.u  
stc.edu.cn {zhyuey, sunxiaoyan, fengwu}@ustc.edu.cn

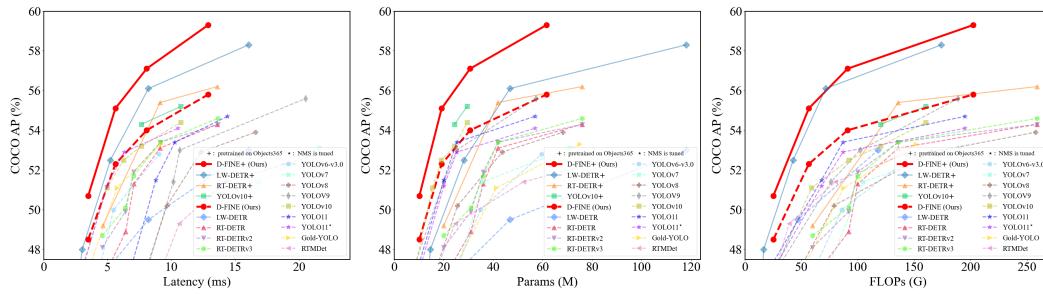


图1：与其他检测器在延迟（左）、模型大小（中）和计算成本（右）方面的比较。我们使用NVIDIA T4 GPU上的TensorRT FP16测量端到端延迟。

## 摘要

我们推出了D-FINE，一种强大的实时目标检测器，通过重新定义DETR模型中的边界框回归任务，实现了卓越的定位精度。D-FINE包含两个关键组件：细粒度分布优化（FDR）和全局最优定位自蒸馏（GO-LSD）。FDR将回归过程从预测固定坐标转变为迭代优化概率分布，提供了显著提升定位准确性的细粒度中间表示。GO-LSD是一种双向优化策略，通过自蒸馏将定位知识从优化后的分布传递至浅层网络，同时简化深层网络的残差预测任务。此外，D-FINE在计算密集型模块和操作中引入了轻量级优化，实现了速度与精度间更优的平衡。具体而言，D-FINE-L/X在NVIDIA T4 GPU上以124/78 FPS的速度，在COCO数据集上达到了54.0%/55.8% AP。当在Objects365上进行预训练时，D-FINE-L/X分别获得57.1%/59.3% AP，超越了所有现有实时检测器。进一步地，我们的方法仅需极少额外参数和训练成本，就能将各类DETR模型的性能最高提升5.3% AP。代码与预训练模型详见：<https://github.com/Peterande/D-FINE>。

## 1 引言

实时目标检测的需求在各种应用中持续增长（Arani等人，2022年）。在最具影响力实时检测器中，YOLO系列（Redmon等人，2016a；Wang等人，2023a, b；Glenn，2023；Wang与Liao，2024；Wang等人，2024a；Glenn，2024）因其高效性和强大的社区生态系统而广受认可。作为强有力的竞争者，Detection Transformer（DETR）（Carion等人，2020；Zhu等人，2020；Liu等人，2021；Li等人，2022；Zhang等人，2022）凭借其基于transformer的架构提供了独特优势，该架构允许

\*Corresponding authors

---

用于全局上下文建模和直接集合预测，无需依赖非极大值抑制（NMS）和锚框。然而，它们往往受限于高延迟和高计算需求（Zhu等, 2020; Liu等, 2021; Li等, 2022; Zhang等, 2022）。RT-DETR（Zhao等, 2024）通过开发实时变体解决了这些限制，为YOLO检测器提供了一种端到端的替代方案。此外，LW-DETR（Chen等, 2024）已证明，DETR能够达到比YOLO更高的性能上限，尤其是在像Objects365（Shao等, 2019）这样的大规模数据集上训练时。

尽管实时目标检测已取得显著进展，但若干未解难题仍制约着检测器的性能表现。其中核心挑战之一在于边界框回归的建模方式。当前大多数检测器通过回归固定坐标来预测边界框，将边缘视为狄拉克δ分布（Dirac delta distributions）建模的精确值（Liu et al., 2016; Ren et al., 2015; Tian et al., 2019; Lyu et al., 2022）。这种方法虽直观，却无法刻画定位不确定性，导致模型只能采用L1损失和IoU损失——这些损失函数难以对各边缘的独立调整提供充分指导（Girshick, 2015），使得优化过程对微小坐标变化异常敏感，造成收敛缓慢和次优性能。尽管GFocal等方法（Li et al., 2020; 2021）通过概率分布建模不确定性，但仍受限于锚框依赖性、粗粒度定位以及缺乏迭代优化机制。另一项挑战在于如何最大化实时检测器的效率——这类模型受限于有限的计算和参数量预算以维持速度。知识蒸馏（KD）是颇具前景的解决方案，其通过将大型教师模型的知识迁移至小型学生模型来提升性能而不增加成本（Hinton et al., 2015）。然而传统KD方法（如Logit Mimicking和Feature Imitation）在检测任务中效率低下，甚至会导致前沿模型性能下降（Zheng et al., 2022）。相比之下，定位蒸馏（LD）在检测任务中展现出更优效果，但由于其巨大的训练开销及与无锚检测器的兼容性问题，集成LD仍面临挑战。

为解决这些问题，我们提出了D-FINE——一种创新的实时目标检测器，它重新定义了边界框回归方法并引入高效的自蒸馏策略。该方法攻克了固定坐标回归优化困难、无法建模定位不确定性以及需要低成本高效蒸馏的难题。我们提出细粒度分布优化（FDR）技术，将边界框回归从预测固定坐标转变为建模概率分布，提供更精细的中间表征。FDR以残差方式迭代优化这些分布，实现渐进式微调并提升定位精度。基于深层网络能通过概率分布捕获更丰富定位信息从而产生更准确预测的认知，我们提出全局最优定位自蒸馏（GO-LSD）机制。GO-LSD以可忽略的额外训练成本，将深层定位知识迁移至浅层。通过对齐浅层预测与深层精修输出，模型能学习生成更优的早期调整，加速收敛并提升整体性能。此外，我们对现有实时DETR架构（Zhao et al., 2024; Chen et al., 2024）中计算密集型模块进行精简，使D-FINE更快速轻量。尽管这类修改通常会导致性能下降，但FDR与GO-LSD有效缓解了性能衰减，实现了速度与精度的更优平衡。

在COCO数据集（Lin等人, 2014a）上的实验结果表明，D-FINE在实时目标检测中实现了最先进的性能，在准确性和效率上超越了现有模型。D-FINE-L和D-FINE-X在COCO val2017上分别达到了54.0%和55.8%的AP，在NVIDIA T4 GPU上运行时分别达到124 FPS和78 FPS。通过在Objects365（Shao等人, 2019）等更大数据集上进行预训练后，D-FINE系列模型的AP最高可达59.3%，超越了所有现有的实时检测器，展现了其可扩展性和鲁棒性。此外，我们的方法以可忽略的额外参数和训练成本，将多种DETR模型的AP提升了高达5.3%，证明了其灵活性和泛化能力。总之，D-FINE推动了实时检测器的性能边界。通过FDR和GO-LSD解决边界框回归和蒸馏效率中的关键挑战，我们在目标检测领域迈出了有意义的一步，为该领域的进一步探索提供了启发。

---

## 2 相关工作

实时/端到端目标检测器。YOLO系列在实时目标检测领域引领潮流，通过架构创新、数据增强及训练技术的演进不断突破（Redmon等人，2016a；Wang等人，2023a；b；Glenn，2023；Wang与Liao，2024；Glenn，2024）。尽管高效，YOLO通常依赖非极大值抑制（NMS），这会在速度与精度间引入延迟和不稳定性。DETR（Carion等人，2020）通过摒弃NMS和锚框等手工设计组件，彻底革新了目标检测。传统DETR模型（Zhu等人，2020；Meng等人，2021；Zhang等人，2022；Wang等人，2022；Liu等人，2021；Li等人，2022；Chen等人，2022a；c）虽取得卓越性能，却以高计算需求为代价，难以应用于实时场景。近期，RT-DETR（Zhao等人，2024）与LW-DETR（Chen等人，2024）成功实现了DETR的实时化适配。与此同时，YOLOv10（Wang等人，2024a）也摒弃了NMS需求，标志着该系列向端到端检测的重要转型。

基于分布的目标检测。传统的边界框回归方法（Redmon等人，2016b；Liu等人，2016；Ren等人，2015）依赖于狄拉克 $\delta$ 分布，将边界框边缘视为精确且固定的，这使得建模定位不确定性具有挑战性。为解决这一问题，近期模型采用高斯分布或离散分布来表示边界框（Choi等人，2019；Li等人，2020；Qiu等人，2020；Li等人，2021），增强了对不确定性的建模能力。然而，这些方法均基于锚框框架，限制了其与现代无锚检测器如YOLOX（Ge等人，2021）和DETR（Carion等人，2020）的兼容性。此外，它们的分布表示通常以粗粒度方式构建，缺乏有效细化，阻碍了实现更精确预测的能力。

知识蒸馏。知识蒸馏（KD）（Hinton等人，2015）是一种强大的模型压缩技术。传统KD通常专注于通过Logit模仿（Zagoruyko & Komodakis，2017；Mirzadeh等人，2020；Son等人，2021）传递知识。FitNets（Romero等人，2015）首次提出特征模仿，启发了后续一系列基于此思想的扩展工作（Chen等人，2017；Dai等人，2021；Guo等人，2021；Li等人，2017；Wang等人，2019）。针对DETR（Chang等人，2023；Wang等人，2024b）的方法大多结合了logit与多种中间表示的混合蒸馏。最近，定位蒸馏（LD）（Zheng等人，2022）证明传递定位知识对检测任务更为有效。自蒸馏（Zhang等人，2019；2021）是KD的一种特殊形式，它使浅层能从模型自身精炼输出中学习，由于无需单独训练教师模型，其额外训练成本极低。

## 3 预备知识

目标检测中的边界框回归传统上依赖于对狄拉克 $\delta$ 分布的建模，通常采用基于中心点的 $\{x, y, w, h\}$ 或边缘距离 $\{\mathbf{c}, \mathbf{d}\}$ 的形式，其中距离 $\mathbf{d} = \{t, b, l, r\}$ 是从锚点 $\mathbf{c} = \{x_c, y_c\}$ 开始测量的。然而，狄拉克 $\delta$ 假设将边界框边缘视为精确且固定的，这使得难以建模定位不确定性，特别是在模糊情况下。这种僵化的表示不仅限制了优化空间，还会因微小的预测偏移导致显著的定位误差。

为了解决这些问题，GFocal（Li等人，2020；2021）通过离散化的概率分布回归锚点到四条边的距离，为边界框提供了更灵活的建模方式。实际应用中，边界框距离 $\mathbf{d} = \{t, b, l, r\}$ 被建模为：

$$\mathbf{d} = d_{\max} \sum_{n=0}^N \frac{n}{N} \mathbf{P}(n), \quad (1)$$

其中 $d_{\max}$ 是一个标量，用于限制与锚点中心的最大距离，而 $\mathbf{P}(n)$ 表示四条边各候选距离的概率。尽管GFocal通过概率分布在处理模糊性和不确定性方面迈出了一步，但其回归方法仍存在特定挑战：(1) *Anchor Dependency*：回归与锚框中心绑定，限制了预测的多样性及与无锚框架的兼容性。(2) *No Iterative Refinement*：预测一次性完成，缺乏迭代优化，降低了回归的鲁棒性。

(3) *Coarse Localization*: 固定的距离范围和均匀的区间划分可能导致定位粗糙，特别是对于小物体而言，因为每个区间代表了一个较宽的可能值范围。

定位蒸馏（LD）是一种前景广阔的方法，研究表明传递定位知识对检测任务更为有效（Zhang等人，2022）。该方法基于GFocal框架，通过从教师模型中蒸馏出有价值的定位知识来增强学生模型，而非简单地模仿分类逻辑或特征图。尽管具有优势，该方法仍依赖于基于锚点的架构，并会产生额外的训练成本。

## 4 方法

我们提出D-FINE，一种在速度、体积、计算成本和精度上均表现卓越的实时目标检测器。D-FINE通过两个关键组件解决现有边界框回归方法的不足：细粒度分布优化（FDR）与全局最优定位自蒸馏（GO-LSD）。这两个组件协同工作，能以可忽略的额外参数量和训练时间成本显著提升性能。

(1) FDR通过迭代优化作为边界框预测校正的概率分布，提供了一种更细粒度的中间表示。该方法独立地捕捉并优化每条边的不确定性。通过利用非均匀加权函数，FDR能够在每个解码器层实现更精确、渐进的调整，从而提高定位精度并减少预测误差。FDR在无需锚框的端到端框架内运行，使得优化过程更加灵活且鲁棒。

(2) GO-LSD从精炼的分布中提取定位知识，并将其蒸馏至较浅层。随着训练推进，最终层生成的软标签愈发精确。较浅层通过GO-LSD使其预测与这些标签对齐，从而获得更准确的预测结果。早期阶段预测能力的提升，使得后续层能专注于细化更小的残差。这种相互促进形成协同效应，逐步实现更精准的定位。

为了进一步提升D-FINE的效率，我们对现有实时DETR架构（Zhao等人，2024）中计算密集的模块和操作进行了精简，使D-FINE更快、更轻量化。尽管这些修改通常会带来一定的性能损失，但FDR和GO-LSD有效缓解了这种性能下降。具体修改内容详见表3。

### 4.1 细粒度分布优化

细粒度分布优化（FDR）通过迭代方式优化由解码器层生成的细粒度分布，如图2所示。初始阶段，首个解码器层通过传统的边界框回归头和D-FINE头（两者均为MLP结构，仅输出维度不同）预测初步边界框及概率分布。每个边界框关联四个分布，分别对应四条边。初始边界框作为参考框，后续层则以残差方式专注于调整分布以细化结果。优化后的分布随即用于调整对应初始边界框的四条边，通过每次迭代逐步提升其精确度。

数学上，设 $\mathbf{b}^0 = \{x, y, W, H\}$ 表示初始的边界框预测，其中 $\{x, y\}$ 代表预测的边界框中心， $\{W, H\}$ 则表示框的宽度和高度。随后，我们可以将 $\mathbf{b}^0$ 转换为中心坐标 $\mathbf{c}^0 = \{x, y\}$ 及边缘距离 $\mathbf{d}^0 = \{t, b, l, r\}$ ，后者表示从中心到上、下、左、右各边的距离。对于第 $l$ 层，精炼后的边缘距离 $\mathbf{d}^l = \{t^l, b^l, l^l, r^l\}$ 计算如下：

$$\mathbf{d}^l = \mathbf{d}^0 + \{H, H, W, W\} \cdot \sum_{n=0}^N W(n) \mathbf{Pr}^l(n), \quad l \in \{1, 2, \dots, L\}, \quad (2)$$

其中 $\mathbf{Pr}^l(n) = \{\Pr_t^l(n), \Pr_b^l(n), \Pr_l^l(n), \Pr_r^l(n)\}$ 代表四个独立的分布，每个边缘对应一个。每个分布预测对应边缘候选偏移值的可能性。这些候选值由加权函数 $W(n)$ 决定，其中 $n$ 从 $N$ 个离散区间中索引，每个区间对应一个潜在的边缘偏移量。分布的加权和产生边缘偏移量。随后，这些边缘偏移量会根据初始边界框的高度 $H$ 和宽度 $W$ 进行缩放，确保调整与框尺寸成比例。

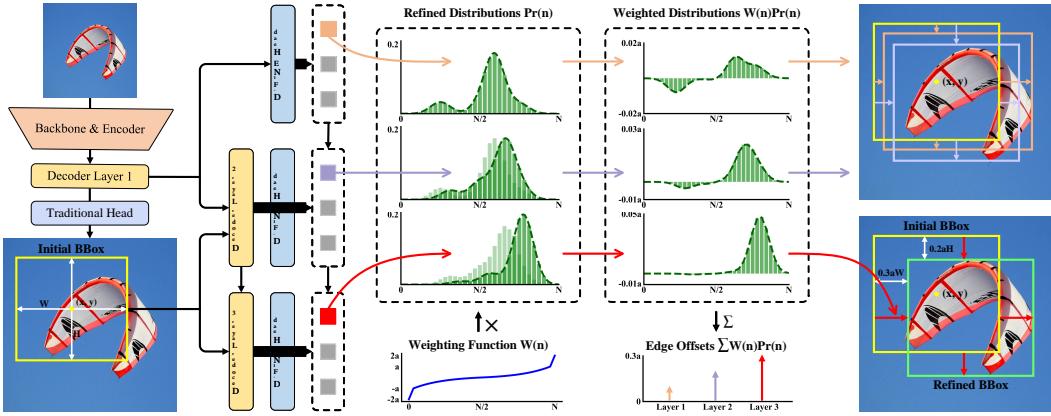


图2：采用FDR的D-FINE框架概览。作为更细粒度中间表征的概率分布通过解码器层以残差方式迭代优化。应用非均匀加权函数以实现更精细的定位。

通过残差调整来更新精炼后的分布，定义如下：

$$\text{Pr}^l(n) = \text{Softmax}(\logits^l(n)) = \text{Softmax}(\Delta\logits^l(n) + \logits^{l-1}(n)), \quad (3)$$

前一层的逻辑值 $\logits^{l-1}(n)$ 反映了对四条边各区间偏移值的置信度。当前层预测残差逻辑值 $\Delta\logits^l(n)$ ，将其与前一层的逻辑值相加，形成更新后的逻辑值 $\logits^l(n)$ 。随后通过softmax函数对这些更新后的逻辑值进行归一化处理，生成优化后的概率分布。

为了便于进行精确而灵活的调整，权重函数 $W(n)$ 定义为：

$$W(n) = \begin{cases} 2 \cdot W(1) = -2a & n = 0 \\ c - c \left( \frac{a}{c} + 1 \right)^{\frac{N-2n}{N-2}} & 1 \leq n < \frac{N}{2} \\ -c + c \left( \frac{a}{c} + 1 \right)^{\frac{-N+2n}{N-2}} & \frac{N}{2} \leq n \leq N-1 \\ 2 \cdot W(N-1) = 2a & n = N, \end{cases} \quad (4)$$

其中 $a$ 和 $c$ 是控制函数上界和曲率的超参数。如图2所示， $W(n)$ 的形状确保了当边界框预测接近准确时， $W(n)$ 中的小曲率允许进行更精细的调整。相反，若边界框预测远不准确， $W(n)$ 边缘处较大的曲率及边界上的急剧变化则为大幅修正提供了足够的灵活性。

为了进一步提升分布预测的准确性并与真实值对齐，受分布焦点损失（DFL）（Li等人，2020）的启发，我们提出了一种新的损失函数——细粒度定位（FGL）损失，其计算方式如下：

$$\begin{aligned} \mathcal{L}_{\text{FGL}} &= \sum_{l=1}^L \left( \sum_{k=1}^K \text{IoU}_k \left( \omega_{\leftarrow} \cdot \text{CE} \left( \text{Pr}^l(n)_k, n_{\leftarrow} \right) + \omega_{\rightarrow} \cdot \text{CE} \left( \text{Pr}^l(n)_k, n_{\rightarrow} \right) \right) \right) \\ \omega_{\leftarrow} &= \frac{|\phi - W(n_{\rightarrow})|}{|W(n_{\leftarrow}) - W(n_{\rightarrow})|}, \quad \omega_{\rightarrow} = \frac{|\phi - W(n_{\leftarrow})|}{|W(n_{\leftarrow}) - W(n_{\rightarrow})|}, \end{aligned} \quad (5)$$

其中 $\text{Pr}^l(n)_k$ 表示对应于第 $k$ 个预测的概率分布。 $\phi$ 是计算为 $\phi = (\mathbf{d}^{GT} - \mathbf{d}^0) / \{H, H, W, W\}$ 的相对偏移量。 $\mathbf{d}^{GT}$ 代表真实边缘距离， $n_{\leftarrow}, n_{\rightarrow}$ 是与 $\phi$ 相邻的区间索引。带有权重 $\omega_{\leftarrow}$ 和 $\omega_{\rightarrow}$ 的交叉熵（CE）损失确保区间之间的插值与真实偏移量精确对齐。通过引入基于IoU的加权，FGL损失促使不确定性较低的分布更加集中，从而实现更精确可靠的边界框回归。

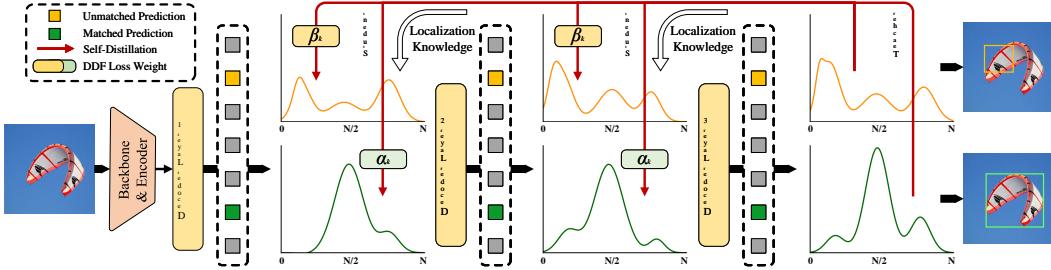


图3：GO-LSD流程概览。通过采用解耦加权策略的DDF损失，将最终层精炼分布中的定位知识蒸馏至较浅层。

#### 4.2 全局最优定位自蒸馏

全局最优定位自蒸馏（GO-LSD）利用最终层精炼的分布预测 $\{v^*\}$ 将定位知识蒸馏至浅层网络，如图3所示。该过程首先通过匈牙利匹配算法（Carion等人，2020）对各层预测进行匹配，识别模型每个阶段的局部边界框对应关系。为实现全局优化，GO-LSD将所有层的匹配索引聚合为统一联合集，该集合整合了各层最精确的候选预测，确保所有预测都能受益于蒸馏过程。除优化全局匹配外，GO-LSD还会在训练过程中优化未匹配预测，从而提升整体稳定性并改善综合性能。虽然定位任务通过该联合集进行优化，但分类任务仍遵循一对一匹配原则，避免产生冗余检测框。这种严格匹配机制意味着联合集中部分预测虽定位精准但置信度较低——这些低置信预测往往代表具有精确定位能力的候选目标，仍需通过蒸馏有效提取其知识。

为此，我们提出了解耦蒸馏焦点（DDF）损失函数，它采用解耦加权策略，确保对高IoU但低置信度的预测赋予适当权重。DDF损失还根据匹配与非匹配预测的数量进行加权，平衡它们的整体贡献与个体损失。这种方法使得蒸馏过程更加稳定高效。解耦蒸馏焦点损失 $\mathcal{L}_{\text{DDF}}$ 的公式表达如下：

$$\begin{aligned} \mathcal{L}_{\text{DDF}} = T^2 \sum_{l=1}^{L-1} & \left( \sum_{k=1}^{K_m} \alpha_k \cdot \text{KL} \left( \mathbf{Pr}^l(n)_k, \mathbf{Pr}^L(n)_k \right) + \sum_{k=1}^{K_u} \beta_k \cdot \text{KL} \left( \mathbf{Pr}^l(n)_k, \mathbf{Pr}^L(n)_k \right) \right) \\ \alpha_k = \text{IoU}_k \cdot \frac{\sqrt{K_m}}{\sqrt{K_m} + \sqrt{K_u}}, \quad \beta_k = \text{Conf}_k \cdot \frac{\sqrt{K_u}}{\sqrt{K_m} + \sqrt{K_u}}, \end{aligned} \quad (6)$$

其中KL表示Kullback-Leibler散度（Hinton等人，2015年）， $T$ 是用于平滑逻辑值的温度参数。第 $k$ 个匹配预测的蒸馏损失由 $\alpha_k$ 加权，其中 $K_m$ 和 $K_u$ 分别是匹配与未匹配预测的数量。对于第 $k$ 个未匹配预测，其权重为 $\beta_k$ ， $\text{Conf}_k$ 表示分类置信度。

## 5 实验

### 5.1 实验设置

为验证所提方法的有效性，我们在COCO（Lin等人，2014a）和Objects365（Shao等人，2019）数据集上进行了实验。采用标准COCO指标评估D-FINE性能，包括IoU阈值从0.50到0.95区间平均的平均精度（AP），特定阈值下的AP（ $\text{AP}_{50}$ 与 $\text{AP}_{75}$ ）以及不同物体尺度下的AP：小物体（ $\text{AP}_S$ ）、中等物体（ $\text{AP}_M$ ）和大物体（ $\text{AP}_L$ ）。此外，我们还通过参数量（#Params.）、计算成本（GFLOPs）及端到端延迟来提供模型效率指标，其中延迟数据基于NVIDIA T4 GPU使用TensorRT FP16测得。

表1：各类实时目标检测器在COCO val2017数据集上的性能对比。

Model	#Params.	GFLOPs	Latency (ms)	AP <sup>val</sup>	AP <sub>50</sub> <sup>val</sup>	AP <sub>75</sub> <sup>val</sup>	AP <sub>S</sub> <sup>val</sup>	AP <sub>M</sub> <sup>val</sup>	AP <sub>L</sub> <sup>val</sup>
<i>Non-end-to-end Real-time Object Detectors</i>									
YOLOv6-L	59M	150	9.04	52.8	70.3	57.7	34.4	58.1	70.1
YOLOv7-L	36M	104	16.81	51.2	69.7	55.5	35.2	55.9	66.7
YOLOv7-X	71M	189	21.57	52.9	71.1	57.4	36.9	57.7	68.6
YOLOv8-L	43M	165	12.31	52.9	69.8	57.5	35.3	58.3	69.8
YOLOv8-X	68M	257	16.59	53.9	71.0	58.7	35.7	59.3	70.7
YOLOv9-C	25M	102	10.66	53.0	70.2	57.8	36.2	58.5	69.3
YOLOv9-E	57M	189	20.53	55.6	72.8	60.6	40.2	61.0	71.4
Gold-YOLO-L	75M	152	9.21	53.3	70.9	-	33.8	58.9	69.9
RTMDet-L	52M	80	14.23	51.3	68.9	55.9	33.0	55.9	68.4
RTMDet-X	95M	142	21.59	52.8	70.4	57.2	35.9	57.3	69.1
YOLO11-L	25M	87	10.28	53.4	70.1	58.2	35.6	59.1	69.2
YOLO11-X	57M	195	14.39	54.7	71.6	59.5	37.7	59.7	70.2
YOLO11-L*	25M	87	6.31	52.9	69.4	57.7	35.2	58.7	68.8
YOLO11-X*	57M	195	10.52	54.1	70.8	58.9	37.0	59.2	69.7
<i>End-to-end Real-time Object Detectors</i>									
YOLOv10-L	24M	120	7.66	53.2	70.1	58.1	35.8	58.5	69.4
YOLOv10-X	30M	160	10.74	54.4	71.3	59.3	37.0	59.8	70.9
RT-DETR-R50	42M	136	9.12	53.1	71.3	57.7	34.8	58.0	70.0
RT-DETR-R101	76M	259	13.61	54.3	72.7	58.6	36.0	58.8	72.1
RT-DETR-HG-L	32M	107	9.25	53.0	71.7	57.3	34.6	57.4	71.2
RT-DETR-HG-X	67M	234	14.01	54.8	73.1	59.4	35.7	59.6	72.9
RT-DETRv2-L	42M	136	9.15	53.4	71.6	57.4	36.1	57.9	70.8
RT-DETRv2-X	76M	259	13.66	54.3	72.8	58.8	35.8	58.8	72.1
RT-DETRv3-L	42M	136	9.12	53.4	-	-	-	-	-
RT-DETRv3-X	76M	259	13.61	54.6	-	-	-	-	-
LW-DETR-L	47M	72	8.21	49.5	-	-	-	-	-
LW-DETR-X	118M	174	16.06	53.0	-	-	-	-	-
<b>D-FINE-L (Ours)</b>	31M	91	8.07	<b>54.0</b>	71.6	58.4	36.5	58.0	71.9
<b>D-FINE-X (Ours)</b>	62M	202	12.89	<b>55.8</b>	73.7	60.2	37.3	60.5	73.4
<i>End-to-end Real-time Object Detectors (Pretrained on Objects365)</i>									
YOLOv10-L	24M	120	7.66	54.0	71.0	58.9	36.5	59.2	70.5
YOLOv10-X	30M	160	10.74	54.9	71.9	59.8	37.6	60.2	71.7
RT-DETR-R50	42M	136	9.12	55.3	73.4	60.1	37.9	59.9	71.8
RT-DETR-R101	76M	259	13.61	56.2	74.6	61.3	38.3	60.5	73.5
LW-DETR-L	47M	72	8.21	56.1	74.6	60.9	37.2	60.4	73.0
LW-DETR-X	118M	174	16.06	58.3	76.9	63.3	40.9	63.3	74.8
<b>D-FINE-L (Ours)</b>	31M	91	8.07	<b>57.1</b>	74.7	62.0	40.0	61.5	74.2
<b>D-FINE-X (Ours)</b>	62M	202	12.89	<b>59.3</b>	76.8	64.6	42.3	64.2	76.4

\* NMS的置信度阈值设置为0.01。

## 5.2 与实时检测器的比较

表1提供了D-FINE与多种实时目标检测器在COCO val2017数据集上的全面对比。D-FINE在多项指标上实现了效率与精度的出色平衡。具体而言，D-FINE-L以3100万参数和91 GFLOPs的计算量，达到了54.0%的AP，同时保持8.07毫秒的低延迟；而D-FINE-X则以6200万参数和202 GFLOPs的计算量，实现了55.8%的AP，运行延迟为12.89毫秒。

如图1所示，图中展示了延迟与AP、参数量与AP、以及浮点运算次数(FLOPs)与AP的散点关系，D-FINE在所有关键维度上均持续优于其他最先进模型。D-FINE-L以更少的计算资源需求(91 GFLOPs vs. 120/136/174)，实现了比YOLOv10-L(53.2%)、RT-DETR-R50(53.1%)和LW-DETR-X(53.0%)更高的AP(54.0%)。同样地，D-FINE-X以更优性能(55.8% AP vs. 54.4%/54.3%)和更高效的参数数量、GFLOPs及延迟表现，超越了YOLOv10-X和RT-DETR-R101。

我们进一步在Objects365数据集(Shao等人, 2019)上对D-FINE和YOLOv10进行预训练，随后在COCO上进行微调。预训练后，D-FINE-L和D-FINE-X均展现出显著的

表2：FDR与GO-LSD在不同DETR模型上于COCO val2017数据集的有效性对比。

Model	#Params.	#Epochs	AP <sup>val</sup>	AP <sup>val</sup> <sub>50</sub>	AP <sup>val</sup> <sub>75</sub>	AP <sup>val</sup> <sub>S</sub>	AP <sup>val</sup> <sub>M</sub>	AP <sup>val</sup> <sub>L</sub>
Deformable-DETR + FDR & GO-LSD	40M	12	43.7	62.2	46.9	26.4	46.4	57.9
	40M	12	47.1 (+3.4)	64.7	50.8	29.0	50.3	62.8
DAB-DETR + FDR & GO-LSD	48M	12	44.2	62.5	47.3	27.5	47.1	58.6
	48M	12	49.5 (+5.3)	67.2	54.1	31.8	53.2	63.3
DN-DETR + FDR & GO-LSD	48M	12	46.0	64.8	49.9	27.7	49.1	62.3
	48M	12	49.7 (+3.7)	67.5	54.4	31.8	53.4	63.8
DINO + FDR & GO-LSD	47M	12	49.0	66.6	53.5	32.0	52.3	63.0
	47M	12	51.6 (+2.6)	68.6	56.3	33.8	55.6	65.3
DINO + FDR & GO-LSD	47M	24	50.4	68.3	54.8	33.3	53.7	64.8
	47M	24	52.4 (+2.0)	69.5	56.9	34.6	55.7	66.2

性能提升显著，分别达到了57.1%和59.3%的平均精度（AP）。这些改进使它们的AP值超越了YOLOv10-L和YOLOv10-X，分别高出3.1%和4.4%，从而确立了在此次比较中的顶尖模型地位。更重要的是，遵循YOLOv8（Glenn等人，2023年）的预训练协议，YOLOv10需在Objects365数据集上进行300轮预训练。相比之下，D-FINE仅需21轮训练即可实现显著的性能提升。这些发现与LW-DETR（Chen等人，2024年）的结论相互印证，表明基于DETR的模型相比YOLO等其他检测器，能从预训练中获得更为显著的收益。

### 5.3 各类DETR模型的有效性

表2展示了我们提出的FDR和GO-LSD方法在COCO val2017数据集上对多种基于DETR的目标检测器的有效性。我们的方法设计灵活，可无缝集成到任何DETR架构中，在不增加参数数量和计算负担的情况下显著提升性能。将FDR和GO-LSD应用于Deformable DETR、DAD-DETR、DN-DETR和DINO后，检测精度持续提高，提升幅度从2.0%到5.3%不等。这些结果凸显了FDR和GO-LSD在提升定位精度和最大化效率方面的有效性，证明了它们在不同端到端检测框架中的适应性和显著影响。

### 5.4 消融研究

#### 5.4.1 D-FINE的路线图

表3展示了从基线模型（RT-DETR-HGNetv2-L（Zhao等人，2024））逐步演进至我们提出的D-FINE框架的过程。初始基线指标为53.0% AP、32M参数量、110 GFLOPs运算量和9.25 ms延迟。我们首先移除了所有解码器投影层，这一改动将GFLOPs降至97，延迟缩短至8.02 ms，但AP也略微下降至52.4%。为弥补这一下降，我们引入了目标门控层，仅以计算成本微增的代价将AP恢复至52.8%。

目标门控层被策略性地放置在解码器的交叉注意力模块之后，取代了原有的残差连接。它使得查询能够动态地在不同层间切换对不同目标的关注焦点，有效防止信息纠缠。该机制的运作方式如下：

$$\mathbf{x} = \sigma([\mathbf{x}_1, \mathbf{x}_2] \mathbf{W}^T + \mathbf{b})_1 \cdot \mathbf{x}_1 + \sigma([\mathbf{x}_1, \mathbf{x}_2] \mathbf{W}^T + \mathbf{b})_2 \cdot \mathbf{x}_2 \quad (7)$$

其中 $\mathbf{x}_1$ 代表先前的查询， $\mathbf{x}_2$ 是交叉注意力结果。 $\sigma$ 为应用于拼接输出的sigmoid激活函数， $[.]$ 则表示拼接操作。

接下来，我们将编码器的CSP层替换为GELAN层（Wang & Liao, 2024）。这一替换使AP提升至53.5%，但同时也增加了参数量、GFLOPs和延迟。为了缓解复杂度上升的问题，我们降低了GELAN的隐藏维度，从而平衡模型的

表3：从基线模型到D-FINE的逐步改进步骤。每一步均展示了AP、参数量、延迟及FLOPs的变化情况。

Model	AP <sup>val</sup>	#Params.	Latency (ms)	GFLOPs
baseline: RT-DETR-HGNetv2-L (Zhao et al., 2024)	53.0	32M	9.25	110
Remove Decoder Projection Layers	52.4	32M	8.02	97
+ Target Gating Layers	52.8	33M	8.15	98
Encoder CSP layers → GELAN (Wang & Liao, 2024)	53.5	46M	10.69	167
Reduce Hidden Dimension in GELAN by half	52.8	31M	8.01	91
Uneven Sampling Points (S: 3, M: 6, L: 3)	52.9	31M	7.90	91
RT-DETRv2 Training Strategy (Lv et al., 2024)	53.0	31M	7.90	91
+ FDR	53.5	31M	8.07	91
+ GO-LSD	<b>54.0</b> (+1.0)	<b>31M</b> (-3%)	<b>8.07</b> (-13%)	<b>91</b> (-17%)

表4：D-FINE-L上的超参数消融研究。 $\epsilon$ 是一个非常小的值。 $\tilde{a}$ 、 $\tilde{c}$ 表明 $a$ 和 $c$ 是可学习参数。

$a, c$	$\frac{1}{4}, \frac{1}{4}$	$\frac{1}{2}, \frac{1}{\epsilon}$	$\frac{1}{2}, \frac{1}{2}, \frac{1}{4}$	$\frac{1}{2}, \frac{1}{8}$	$1, \frac{1}{4}$	$\tilde{a}, \tilde{c}$
AP <sup>val</sup>	52.7	53.0	<b>53.3</b>	53.2	53.2	53.1
$N$	4	8	16	<b>32</b>	64	128
AP <sup>val</sup>	53.3	53.4	53.5	<b>53.7</b>	53.6	53.6
$T$	1	2.5	<b>5</b>	7.5	10	20
AP <sup>val</sup>	53.2	53.7	<b>54.0</b>	53.8	53.7	53.5

表5：蒸馏方法在性能、训练时间和GPU内存使用方面的比较。GO-LSD以最小的额外训练成本实现了最高的AP<sup>val</sup>。

Methods	AP <sup>val</sup>	Time/Epoch	Memory
baseline	53.0	29min	8552M
Logit Mimicking	52.6	31min	8554M
Feature Imitation	52.9	31min	8554M
baseline + FDR	53.8	30min	8730M
Localization Distill.	53.7	31min	8734M
GO-LSD	<b>54.5</b>	31min	8734M

复杂度并保持AP在52.8%的同时提升效率。我们通过在不同尺度上实施非均匀采样（小尺度：3，中尺度：6，大尺度：3）进一步优化采样点，使AP小幅提升至52.9%。然而，其他采样组合如（小尺度：6，中尺度：3，大尺度：3）和（小尺度：3，中尺度：3，大尺度：6）会导致AP性能轻微下降0.1%。采用RT-DETRv2训练策略（Lv等人，2024）（详见附录A.1.1）在不影响参数量或延迟的情况下，将AP提升至53.0%。最终，结合FDR与GO-LSD模块将AP推升至54.0%，相比基线模型实现了延迟降低13%和GFLOPs减少17%。这些渐进式改进证明了我们D-FINE框架的稳健性与有效性。

#### 5.4.2 超参数敏感性分析

5.4.1节展示了一部分超参数消融研究，评估了我们的模型对FDR和GO-LSD模块中关键参数的敏感性。我们考察了加权函数参数 $a$ 和 $c$ 、分布区间数量 $N$ ，以及用于平滑KL散度中逻辑值的温度参数 $T$ 。

(1) 设置 $a = \frac{1}{2}$ 和 $c = \frac{1}{4}$ 可获得53.3%的最高平均精度（AP）。值得注意的是，将 $a$ 和 $c$ 作为可学习参数( $\tilde{a}, \tilde{c}$ )会使AP略微下降至53.1%，这表明固定值能简化优化过程。当 $c$ 取值极大时，加权函数会逼近等间隔线性函数，导致AP次优值为53.0%。此外， $a$ 取值过大或过小会降低精细度或限制灵活性，从而对定位精度产生不利影响。(2) 增加分布区间的数量可提升性能，在 $N = 32$ 时达到53.7%的最大AP值。超过 $N = 32$ 后未观察到显著增益。(3) 温度参数 $T$ 控制蒸馏过程中逻辑值的平滑程度，在 $T = 5$ 时取得54.0%的最优AP值，这表明在分布软化和有效知识迁移之间达到了平衡。

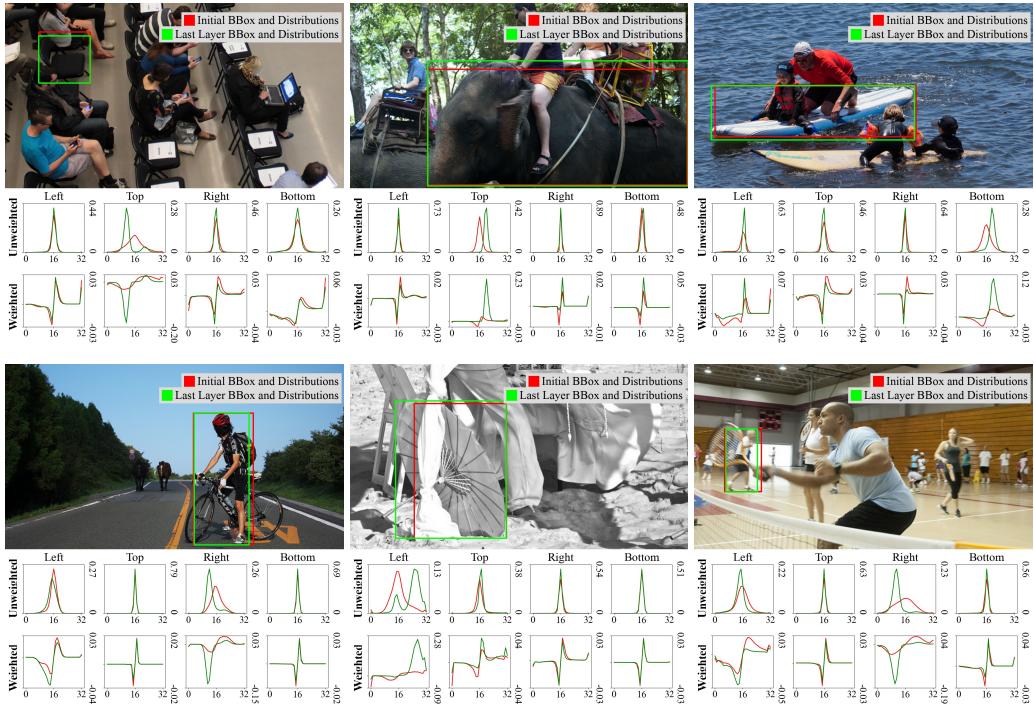


图4：不同检测场景下初始与优化边界框的FDR可视化，包括未加权与加权分布，突显定位精度的提升。

#### 5.4.3 蒸馏方法比较

5.4.1节从性能、训练时间和GPU内存占用三个维度对比了不同蒸馏方法。基线模型在四块NVIDIA RTX 4090 GPU上实现了53.0%的AP，每轮训练耗时29分钟，内存占用8552 MB。由于DETR中一对一匹配的不稳定性，传统蒸馏技术如Logit Mimicking和Feature Imitation未能提升性能——Logit Mimicking使AP降至52.6%，而Feature Imitation达到52.9%。引入我们的FDR模块后，AP提升至53.8%，且训练成本增幅极小。应用基础版定位蒸馏（Zheng等人，2022）可进一步将AP提高至53.7%。我们的GO-LSD方法实现了54.5%的最高AP，相较基线仅增加6%训练时间和2%内存占用。值得注意的是，本比较未采用任何轻量化优化，纯粹聚焦蒸馏性能。

#### 5.5 可视化分析

图4展示了FDR在不同检测场景下的处理过程。我们在图像上叠加显示两个边界框的筛选检测结果。红色框代表初始来自第一解码器层的预测，绿色框则表示最终解码器层优化后的预测。最终预测与目标物体的贴合度更高。图像下方的第一行展示了四条边（左、上、右、下）未加权的概率分布。第二行显示经过加权函数 $\{v^*\}$ 处理后的分布情况。红色曲线对应初始分布，绿色曲线呈现最终优化后的分布。加权分布能在准确预测附近实现精细调整，同时支持大幅偏移的快速修正，进一步阐明FDR如何通过优化初始边界框的偏移量来实现逐级精准定位。

---

## 6 结论

本文介绍了D-FINE，一种强大的实时目标检测器，它通过细粒度分布优化（FDR）和全局最优定位自蒸馏（GO-LSD），重新定义了DETR模型中的边界框回归任务。在COCO数据集上的实验结果表明，D-FINE实现了最先进的准确性和效率，超越了所有现有的实时检测器。局限性与未来工作：然而，轻量级D-FINE模型与其他紧凑模型之间的性能差距仍然较小。一个可能的原因是浅层解码器层可能产生不够精确的最终层预测，限制了将定位知识蒸馏到早期层的效果。解决这一挑战需要在不增加推理延迟的情况下增强轻量级模型的定位能力。未来的研究可以探索先进的架构设计或新颖的训练范式，这些方法允许在训练时加入更多复杂的解码器层，同时在测试时简单地丢弃它们以保持轻量级推理。我们希望D-FINE能激励这一领域的进一步进展。

## 参考文献

Elahe Arani、Shruthi Gowda、Ratnajit Mukherjee、Omar Magdy、Senthilkumar Kathiresan与B ahram Zonooz。跨多领域的实时目标检测网络综合研究：综述。{v\*}，2022年。 Nicolas Carion、Francisco Massa、Gabriel Synnaeve、Nicolas Usunier、Alexander Kirillov及Sergey Zagoruyko。基于Transformer的端到端目标检测。载于*European Conference on Computer Vision*, 第213–229页。Springer, 2020年。 Jiahao Chang、Shuo Wang、Hai-Ming Xu、Zehui Chen、Chenhongyi Yang与Feng Zhao。DETRDistill：面向DETR系列模型的通用知识蒸馏框架。载于*Proceedings of the IEEE/CVF International Conference on Computer Vision*, 第6898–6908页，2023年。 Guobin Chen、Wongun Choi、Xiang Yu、Tony Han及Manmohan Chandraker。基于知识蒸馏的高效目标检测模型学习。载于*NeurIPS*, 2017年。 Qiang Chen、Xiaokang Chen、Gang Zeng与Jingdong Wang。Group DETR：通过解耦一对多标签分配实现快速训练收敛。{v\*}，2022a。 Qiang Chen、Jian Wang、Chuchu Han、Shan Zhang、Zexian Li、Xiaokang Chen、Jiahui Chen、Xiaodi Wang、Shuming Han、Gang Zhang、Haocheng Feng、Kun Yao、Junyu Han、Errui Ding及Jingdong Wang。Group DETR v2：基于编码器-解码器预训练的强目标检测器，2022b。 Qiang Chen、Jian Wang、Chuchu Han、Shan Zhang、Zexian Li、Xiaokang Chen、Jiahui Chen、Xiaodi Wang、Shuming Han、Gang Zhang等。Group DETR v2：基于编码器-解码器预训练的强目标检测器。{v\*}，2022c。 Qiang Chen、Xiangbo Su、Xinyu Zhang、Jian Wang、Jiahui Chen、Yunpeng Shen、Chuchu Han、Ziliang Chen、Weixiang Xu、Fanrong Li等。LW-DETR：替代YOLO的实时检测Transformer方案。{v\*}，2024年。 Jiwoong Choi、Dayoung Chun、Hyun Kim与Hyuk-Jae Lee。Gaussian YOLOv3：利用定位不确定性的自动驾驶高精度快速目标检测器。载于*ICCV*, 2019年。 Cheng Cui、Ruoyu Guo、Yuning Du、Dongliang He、Fu Li、Zewu Wu、Qiwen Liu、Shilei Wen、Jizhou Huang、Xiaoguang Hu、Dianhai Yu、Errui Ding及Yanjun Ma。超越自监督：一种简单有效的骨干网络改进蒸馏方案，2021年。 Xing Dai、Zeren Jiang、Zhao Wu、Yiping Bao、Zhicheng Wang、Si Li与Erjin Zhou。通用实例蒸馏在目标检测中的应用。载于*CVPR*, 2021年。 Zheng Ge、Songtao Liu、Feng Wang、Zeming Li及Jian Sun。YOLOX：2021年超越YOLO系列。{v\*}，2021年。 Ross Girshick。Fast R-CNN。载于*ICCV*, 2015年。 Jocher Glenn。YOLOv8。<https://docs.ultralytics.com/models/yolov8/>, 2023年。

---

Jocher Glenn. Yolo11. <https://docs.ultralytics.com/models/yolo11/>, 2024.

郭建元、韩锴、王云鹤、吴晗、陈星昊、徐春景、徐畅。基于解耦特征的物体检测器蒸馏。载于*CVPR*, 2021年。

杰弗里·辛顿、奥里奥尔·维尼利亚尔斯和杰夫·迪恩。神经网络中的知识蒸馏。*arXiv preprint arXiv:1503.02531*, 2015年。

李锋, 张浩, 刘世龙, 郭健, 倪明选, 张磊. DN-DETR: 通过引入查询去噪加速DETR训练. 载于 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 第13619-13627页, 2022年.

李全全, 金胜英, 严俊杰。模仿高效网络进行目标检测。于*CVPR*, 2017年。

李翔、王文海、吴立军、陈硕、胡晓林、李军、唐金辉和杨健。广义焦点损失：学习高质量与分布式的边界框用于密集目标检测。发表于*NeurIPS*, 2020年。

李翔、王文海、胡晓林、李军、唐金辉和杨健。广义焦点损失v2：学习可靠定位质量估计用于密集目标检测。发表于*CVPR*, 2021年。

Tsung-Yi Lin、Michael Maire、Serge Belongie、James Hays、Pietro Perona、Deva Ramanan、Piotr Dollár 和 C. Lawrence Zitnick。Microsoft COCO: 上下文中的常见物体。收录于*ECCV*, 2014a。  
林宗仪、Michael Maire、Serge Belongie、James Hays、Pietro Perona、Deva Ramanan、Piotr Dollár与C Lawrence Zitnick。Microsoft COCO: 上下文中的常见物体。载于*European Conference on Computer Vision*, 第740-755页。Springer出版社, 2014b年。

刘世龙, 李峰, 张浩, 杨晓, 齐贤标, 苏航, 朱军, 张磊. Dab-detr: 动态锚框作为更优查询在detr中的应用. 发表于*International Conference on Learning Representations*, 2021.

Wei Liu、Dragomir Anguelov、Dumitru Erhan、Christian Szegedy、Scott Reed、Cheng-Yang Fu和Alexander C. Berg。SSD: 单次多框检测器。收录于*ECCV*, 2016年。

吕文字, 赵一安, 常钦尧, 黄奎, 王冠中, 刘毅。Rt-detrv2: 基于免费技巧改进的实时检测Transformer基线模型, 2024。

程琪吕、张文威、黄海安、周越、王宇东、刘燕怡、张世龙和陈凯。RTMDet: 实时目标检测器设计的实证研究。*ArXiv*, abs/2212.07784, 2022年。

孟德普、陈晓康、范泽佳、曾刚、李厚强、袁钰辉、孙磊、王井东。条件DETR实现快速训练收敛。载于 *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 第36 51–3660页, 2021年。

赛义德·伊曼·米尔扎德 (Seyed Iman Mirzadeh)、梅尔达德·法拉杰塔巴尔 (Mehrdad Farajtabar)、李昂 (Ang Li)、尼尔·莱文 (Nir Levine)、松川明博 (Akihiro Matsukawa) 与哈桑·加塞姆扎德 (Hassan Ghasemzadeh)。通过教师助理改进知识蒸馏。载于*AAAI*, 2020年。  
仇贺前, 李洪亮, 吴庆波, 石恒灿。基于偏移量分箱分类网络的精确目标检测。载于*CVPR*, 2020年。

约瑟夫·雷德蒙、桑托什·迪瓦拉、罗斯·吉斯克与阿里·法哈迪。你只需看一次：统一、实时的目标检测。载于 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 第779-788页, 2016年a。

约瑟夫·雷德蒙、桑托什·迪瓦拉、罗斯·吉斯克与阿里·法哈迪。《只需一眼：统一、实时的目标检测》。刊于*CVPR*, 2016b。

任少卿, 何恺明, Ross Girshick, 孙剑。Faster R-CNN: 利用区域提议网络实现实时目标检测。发表于*NeurIPS*, 2015年。

Adriana Romero、Nicolas Ballas、Samira Ebrahimi Kahou、Antoine Chassang、Carlo Gatta与Yoshua Bengio。Fitnets: 瘦深度网络的提示。载于*ICLR*, 2015年。

---

奥尔加·鲁萨科夫斯基、贾登、苏浩、乔纳森·克劳斯、桑吉夫·萨提什、马思恩、黄志恒、安德烈·卡帕西、阿迪亚·科斯拉、迈克尔·伯恩斯坦等。ImageNet大规模视觉识别挑战赛。*International Journal of Computer Vision*, 115: 211–252, 2015年。

邵帅、李泽明、张天元、彭超、余刚、张翔宇、李静、孙剑。Objects365: 一个面向目标检测的大规模高质量数据集。发表于*Proceedings of the IEEE/CVF International Conference on Computer Vision*, 第8430–8439页, 2019年。

孙元哲、罗在民、崔俊勇、黄元俊。基于多重教师助理的密集引导知识蒸馏。发表于*ICCV*, 2021年。

田志、沈春华、陈浩与何通。FCOS: 全卷积一阶段目标检测。发表于*ICCV*, 2019年。

王傲、陈辉、刘立浩、陈凯、林子佳、韩军功和丁贵广。Yolov10: 实时端到端目标检测。收录于*The European Conference on Computer Vision (ECCV)*, 2024a。

程成王、贺伟、聂颖、郭建元、刘传建、王云鹤与韩凯。Gold-yolo: 基于聚集-分发机制的高效目标检测器。载于*Advances in Neural Information Processing Systems*。Curran Associates出版社, 2023a。

王建尧与廖弘源。YOLOv9: 利用可编程梯度信息学习你想学的内容。*arXiv preprint arXiv:2402.13616*, 2024年。

王建尧、Alexey Bochkovskiy与廖弘源。YOLOv7: 可训练免费技巧集为实时目标检测器树立新标杆。载于*Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 第7464–7475页, 2023b。

陶王、李元、张晓鹏和冯家石。通过细粒度特征模仿蒸馏目标检测器。发表于*CVPR*, 2019年。

王英明、张翔宇、杨桐与孙健。Anchor DETR: 基于Transformer检测器的查询设计。载于*Proceedings of the AAAI Conference on Artificial Intelligence*, 第2567–2575页, 2022年。

王宇, 李鑫, 翁盛钊, 张刚, 岳海啸, 冯浩成, 韩俊宇, 丁二锐。Kd-detr: 基于一致蒸馏点采样的检测Transformer知识蒸馏。载于*Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 第16016–16025页, 2024b。

谢尔盖·扎戈鲁伊科与尼科斯·科莫达基斯。更加关注注意力机制: 通过注意力迁移提升卷积神经网络性能。发表于*ICLR*, 2017年。

郝张、李峰、刘世龙、张磊、苏杭、朱军、倪亦宏和沈向洋。Dino: 采用改进去噪锚框的端到端目标检测DETR。载于*International Conference on Learning Representations*, 2022年。

张林峰, 宋杰波, 高安妮, 陈经纬, 包成龙, 马凯声。做自己的老师: 通过自蒸馏提升卷积神经网络性能。载于*ICCV*, 第3713–3722页, 2019年。

张林峰、包成龙与马凯盛。自蒸馏: 迈向高效与紧凑的神经网络。

*IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(8):4388–4403, 2021年。

赵毅安、吕文字、徐尚亮、魏金满、王冠中、党青青、刘毅和陈杰。DETR在实时目标检测上超越YOLO。发表于*The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024年。

郑兆辉、叶荣光、王平、任东伟、左旺孟、侯启斌、程明明。密集目标检测中的定位蒸馏。载于*Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 第9407–9416页, 2022年。

---

朱熙洲，苏伟杰，卢乐为，李斌，王晓刚，戴继峰。可变形DETR：端到端目标检测中的可变形Transformer。发表于*International Conference on Learning Representations*, 2020年。

## 附录

### A.1 实现细节

#### A.1.1 超参数配置

表6总结了D-FINE模型的超参数配置。所有变体均采用在ImageNet上预训练的HGNetV2骨干网络（Cui等人，2021；Russakovsky等人，2015）及AdamW优化器。D-FINE-X的嵌入维度设为384，前馈维度为2048，其他模型则分别使用256和1024。D-FINE-X与D-FINE-L配备6层解码器，而D-FINE-M和D-FINE-S分别为4层和3层。GELAN模块从D-FINE-X的192维3层结构逐步缩减至D-FINE-S的64维单层结构。D-FINE-X和D-FINE-L的基础学习率与权重衰减分别为 $2.5 \times 10^{-4}$ 和 $1.25 \times 10^{-4}$ ，而D-FINE-M和D-FINE-S则采用 $2 \times 10^{-4}$ 与 $1 \times 10^{-4}$ 。较小模型还配置了较大模型更高的骨干网络学习率。所有变体的总批次大小统一为32。训练方案包含：D-FINE-X与D-FINE-L先进行72轮增强训练（随机光度畸变、随机缩放输出、随机IoU裁剪及多尺度输入），再执行2轮无增强训练；D-FINE-M和D-FINE-S则进行120轮增强训练后接4轮无增强训练（参见表3中RT-DETRv2训练策略（Lv等人，2024））。预训练轮数方面，D-FINE-X与D-FINE-L为21轮，D-FINE-M和D-FINE-S则在28至29轮之间。

表6：不同D-FINE模型的超参数配置。

Setting	D-FINE-X	D-FINE-L	D-FINE-M	D-FINE-S
Backbone Name	HGNetv2-B5	HGNetv2-B4	HGNetv2-B2	HGNetv2-B0
Optimizer	AdamW	AdamW	AdamW	AdamW
Embedding Dimension	384	256	256	256
Feedforward Dimension	2048	1024	1024	1024
GELAN Hidden Dimension	192	128	128	64
GELAN Depth	3	3	2	1
Decoder Layers	6	6	4	3
Queries	300	300	300	300
$a, c$ in $W(n)$	0.5, 0.125	0.5, 0.25	0.5, 0.25	0.5, 0.25
Bin Number $N$	32	32	32	32
Sampling Point Number	(S: 3, M: 6, L: 3)			
Temperature $T$	5	5	5	5
Base LR	2.5e-4	2.5e-4	2e-4	2e-4
Backbone LR	2.5e-6	1.25e-5	2e-5	1e-4
Weight Decay	1.25e-4	1.25e-4	1e-4	1e-4
Weight of $\mathcal{L}_{VFL}$	1	1	1	1
Weight of $\mathcal{L}_{BBox}$	5	5	5	5
Weight of $\mathcal{L}_{GIoU}$	2	2	2	2
Weight of $\mathcal{L}_{FGL}$	0.15	0.15	0.15	0.15
Weight of $\mathcal{L}_{DDF}$	1.5	1.5	1.5	1.5
Total Batch Size	32	32	32	32
EMA Decay	0.9999	0.9999	0.9999	0.9999
Epochs (w/ + w/o Adv. Aug.)	72 + 2	72 + 2	120 + 4	120 + 4
Epochs (Pretrain + Finetune)	21 + 31	21 + 32	29 + 49	28 + 58

#### A.1.2 数据集设置

在预训练阶段，我们遵循(Chen et al., 2022b; Zhang et al., 2022; Chen et al., 2024)的方法，将Objects365 (Shao et al., 2019)训练集与验证集合并，但排除了前5千张图像。为进一步提升训练效率，我们预先将所有分辨率超过 $640 \times 640$ 的图像调整至 $640 \times 640$ 。采用标准COCO2017 (Lin et al., 2014b)数据划分策略，即在COCO train2017上进行训练，在COCO val2017上进行评估。

## A.2 VISUALIZATION OF D-FINE PREDICTIONS

图5展示了D-FINE-X模型的鲁棒性，可视化了其在多种挑战性场景下的预测表现。这些场景包括遮挡、低光照条件、运动模糊、景深效果、旋转以及密集物体近距离分布的复杂场景。尽管面临这些困难，该模型仍能准确识别并定位诸如动物、车辆和人物等对象。这一可视化结果凸显了模型在保持稳健检测性能的同时，处理复杂现实条件的能力。

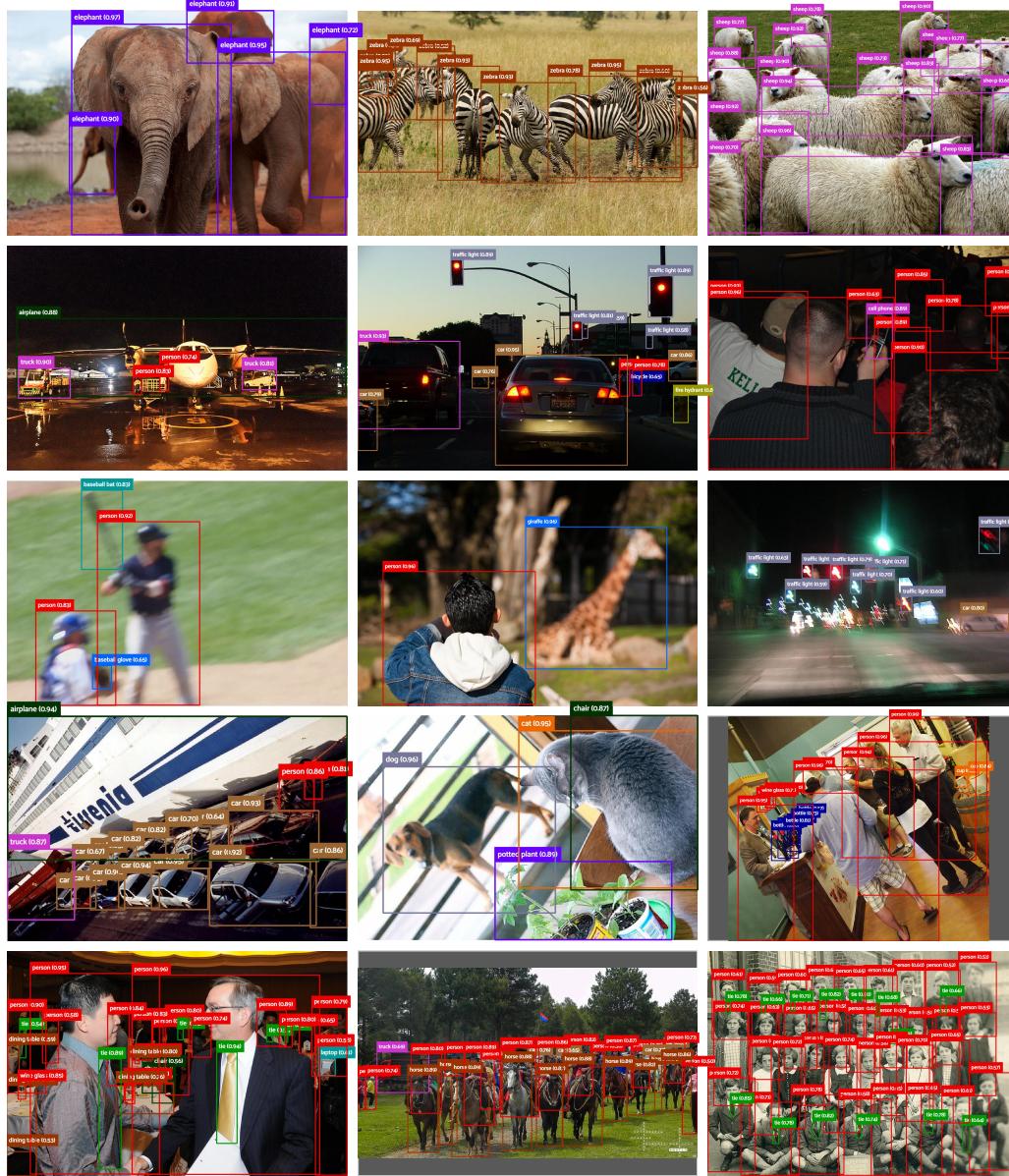


图5：D-FINE-X（未在Objects365上预训练）在挑战性条件下的预测可视化，包括遮挡、低光照、运动模糊、景深效果、旋转以及密集人群场景（置信度阈值=0.5）。

### A.3 与轻量级检测器的比较

表7全面对比了D-FINE模型与多种轻量级实时目标检测器在COCO val2017数据集上S和M尺寸模型的表现。D-FINE-S以48.5%的AP值取得令人瞩目的成绩，超越了Gold-YOLO-S（46.4%）和RT-DETRv2-S（48.1%）等其他轻量级模型，同时仅以10.2M参数和25.2 GFLOPs的计算量保持了3.49毫秒的低延迟。在Objects365上的预训练进一步将D-FINE-S提升至50.7%，实现了+2.2%的进步。同样地，D-FINE-M以19.2M参数和56.6 GFLOPs在5.62毫秒内达到52.3%的AP值，优于YOLOv10-M（51.1%）和RT-DETRv2-M（49.9%）。Objects365预训练持续增强D-FINE-M性能，带来+2.8%的提升。这些结果表明，D-FINE模型在精度与效率间实现了卓越平衡，在保持实时性能的同时持续超越其他最先进的轻量级检测器。

表7：S和M尺寸实时目标检测器在COCO val2017上的性能对比。

Model	#Params.	GFLOPs	Latency (ms)	$AP^{val}$	$AP_{50}^{val}$	$AP_{75}^{val}$	$AP_S^{val}$	$AP_M^{val}$	$AP_L^{val}$
<i>Non-end-to-end Real-time Object Detectors</i>									
YOLOv6-S	7M	17	3.62	45.0	61.8	48.9	24.3	50.2	62.7
YOLOv6-M	35M	86	5.48	50.0	66.9	54.6	30.6	55.4	67.3
YOLOv8-S	11M	29	6.96	44.9	61.8	48.6	25.7	49.9	61.0
YOLOv8-M	26M	79	9.66	50.2	67.2	54.6	32.0	55.7	66.4
YOLOv9-S	7M	26	8.02	44.9	61.8	48.6	25.7	49.9	61.0
YOLOv9-M	20M	76	10.15	50.2	67.2	54.6	32.0	55.7	66.4
Gold-YOLO-S	22M	46	2.01	46.4	63.4	-	25.3	51.3	63.6
Gold-YOLO-M	41M	88	3.21	51.1	68.5	-	32.3	56.1	68.6
RTMDet-S	9M	15	7.77	44.6	61.9	48.1	24.9	48.5	62.5
RTMDet-M	25M	39	10.62	49.4	66.8	53.7	30.3	53.9	66.2
YOLO11-S	9M	22	6.81	46.6	63.4	50.3	28.7	51.3	64.1
YOLO11-M	20M	68	8.79	51.2	67.9	55.3	33.0	56.7	67.5
YOLO11-S*	9M	22	2.86	47.0	63.9	50.7	29.0	51.7	64.4
YOLO11-M*	20M	68	4.95	51.5	68.5	55.7	33.4	57.1	67.9
<i>End-to-end Real-time Object Detectors</i>									
YOLOv10-S	7M	22	2.65	46.3	63.0	50.4	26.8	51.0	63.8
YOLOv10-M	15M	59	4.97	51.1	68.1	55.8	33.8	56.5	67.0
RT-DETR-R18	20M	61	4.63	46.5	63.8	50.4	28.4	49.8	63.0
RT-DETR-R34	31M	93	6.43	48.9	66.8	52.9	30.6	52.4	66.3
RT-DETRv2-S	20M	60	4.59	48.1	65.1	57.4	36.1	57.9	70.8
RT-DETRv2-M	31M	92	6.40	49.9	67.5	58.6	35.8	58.6	72.1
RT-DETRv3-R18	20M	61	4.63	48.7	-	-	-	-	-
RT-DETRv3-R34	31M	93	6.43	50.1	-	-	-	-	-
LW-DETR-S	15M	17	3.02	43.6	-	-	-	-	-
LW-DETR-M	28M	43	5.23	47.2	-	-	-	-	-
<b>D-FINE-S (Ours)</b>	10.2	25.2	3.49	<b>48.5</b>	65.6	52.6	29.1	52.2	65.4
<b>D-FINE-M (Ours)</b>	19.2	56.6	5.55	<b>52.3</b>	69.8	56.4	33.2	56.5	70.2
<i>End-to-end Real-time Object Detectors (Pretrained on Objects365)</i>									
RT-DETR-R18	20M	61	4.63	49.2	66.6	53.5	33.2	52.3	64.8
LW-DETR-S	15M	17	3.02	48.0	66.9	51.7	26.8	52.5	65.5
LW-DETR-M	28M	43	5.23	52.6	72.0	56.7	32.6	57.7	70.7
<b>D-FINE-S (Ours)</b>	10.2	25.2	3.49	<b>50.7</b>	67.6	55.1	32.7	54.6	66.5
<b>D-FINE-M (Ours)</b>	19.2	56.6	5.62	<b>55.1</b>	72.6	59.7	37.9	59.4	71.7

\* NMS的置信度阈值设置为0.01。

### A.4 初始层细化说明

在正文中，我们将层 $l$ 的细化分布定义为：

$$\text{Pr}^l(n) = \text{Softmax} \left( \Delta \text{logits}^l(n) + \text{logits}^{l-1}(n) \right), \quad (8)$$

其中 $\Delta \text{logits}^l(n)$ 是由层 $l$ 预测的残差logits，而 $\text{logits}^{l-1}(n)$ 来自前一层的logits。

---

对于初始层 ( $l = 1$ ) , 由于没有前一层, 公式简化为:

$$\mathbf{Pr}^1(n) = \text{Softmax}(\text{logits}^1(n)). \quad (9)$$

这里,  $\text{logits}^1(n)$ 是由第一层预测的逻辑值。

这一澄清确保了公式在所有层面上保持一致且数学严谨。