DAB-DETR：动态锚框作为DETR查询更优

刘世龙[1,2*]，李锋[2,3]，张浩[2,3]，杨晓[1]，齐先标[2]，苏航[1,4]，朱军[1,4†]，张磊[2†]
1清华大学计算机科学与技术系，北京信息科学与技术国家研究中心，智能技术与系统国家重点实验室，人工智能研究院，清华-博世机器学习联合中心。[2]国际数字经济学院（IDEA）。[3]香港科技大学。[4]中国广东省深圳市鹏城实验室。{liusl20,yangxiao19}@mails.tsinghua.edu.cn {fliay,hzhangcx}@connect.ust.hk {qixianbiao,leizhang}@idea.edu.cn {suhangss,dcszj}@mail.tsinghua.edu.cn

摘要

本文提出了一种新颖的查询表述方法，采用动态锚框（dynamic anchor boxes）对DETR（DEtection TRansformer）进行改进，并深入探讨了查询在DETR中的作用机制。该新方法直接在Transformer解码器中使用边界框坐标作为查询{v*}，并逐层动态更新这些坐标。利用边界框坐标不仅能够通过显式位置先验提升查询与特征的匹配度，解决DETR训练收敛缓慢的问题，还能借助框的宽高信息调制位置注意力图。这一设计清晰地表明，DETR中的查询可视为以级联方式逐层执行软性ROI池化操作。实验表明，在相同设置下，该方法在MS-COCO基准测试中取得了类DETR检测模型的最佳性能（例如使用ResNet50-DC5主干网络训练50个周期时达到45.7%的AP指标）。我们通过大量实验验证了理论分析的正确性与方法的有效性。代码已开源：https://github.com/SlongLiu/DAB-DETR。

1 引言

目标检测是计算机视觉中一项基础且应用广泛的任务。大多数经典检测器基于卷积架构，在过去十年取得了显著进展（Ren等人，2017；Girshick，2015；Redmon等人，2016；Bochkovskiy等人，2020；Ge等人，2021）。最近，Carion等人（2020）提出了一种基于Transformer的端到端检测器DETR（DEtection TRansformer），它消除了对手工设计组件（如锚框）的需求，并与现代基于锚框的检测器（如Faster RCNN（Ren等人，2017））相比展现出优越性能。

与基于锚点的检测器不同，DETR将目标检测建模为一个集合预测问题，利用100个可学习的查询{v*}从图像中探查并汇聚特征，无需非极大值抑制即可直接作出预测。然而，由于查询{v*}的设计和使用效率不足，DETR存在训练收敛速度极慢的问题，通常需要500个训练周期才能达到良好性能。针对这一缺陷，后续研究（Zhu等，2021；Gao等，2021；Meng等，2021；Wang等，2021）纷纷尝试改进DETR查询{v*}的设计，以加速训练收敛并提升检测性能。

尽管取得了诸多进展，DETR中学习查询的作用仍未得到充分理解或利用。虽然先前大多数尝试让DETR中的每个查询更明确地与一个{v*}相关联

*This work was done when Shilong Liu was intern at IDEA.
†Corresponding author.

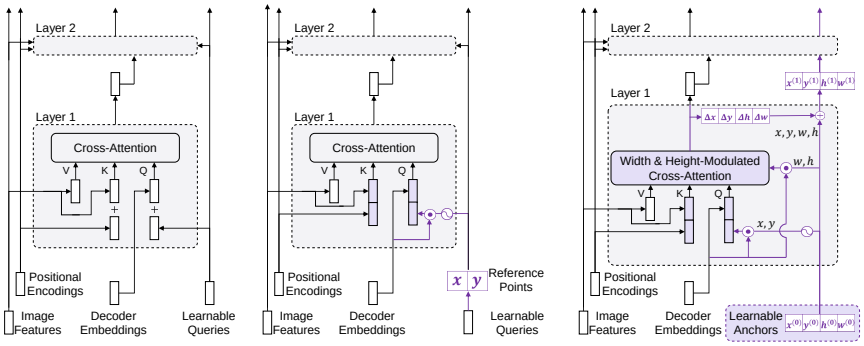


图1：DETR、Conditional DETR与我们提出的DAB-DETR对比。为清晰起见，我们仅展示Transformer解码器中的交叉注意力部分。(a) DETR在所有层中均使用未经调整的可学习查询向量，这导致其训练收敛速度较慢。(b) Conditional DETR为每层适配可学习查询向量，主要是为了提供更好的参考查询点以从图像特征图中汇聚特征。相比之下，(c) DAB-DETR直接使用动态更新的锚框，同时提供参考查询点(x, y)和参考锚框尺寸(w, h)来优化交叉注意力计算。差异模块已用紫色标出。

特定空间位置而非多个位置，技术方案存在显著差异。例如，Conditional DETR通过学习条件空间查询，根据内容特征调整查询以更好地匹配图像特征（Meng等人，2021）。Efficient DETR引入密集预测模块筛选Top-K目标查询（Yao等人，2021），而Anchor DETR将查询建模为二维锚点（Wang等人，2021），两者均将每个查询与特定空间位置关联。类似地，Deformable DETR直接将二维参考点作为查询，并在每个参考点执行可变形交叉注意力操作（Zhu等人，2021）。但上述工作仅利用二维位置作为锚点，未考虑目标尺度因素。

受这些研究的启发，我们深入探究了Transformer解码器中的交叉注意力模块，并提出使用锚框（即4D框坐标x, y, w, h）作为DETR中的查询项，逐层更新它们。这一新的查询构建方式通过同时考虑每个锚框的位置和大小，为交叉注意力模块引入了更优的空间先验，这不仅使实现更为简洁，也深化了对DETR中查询项作用的理解。

这一表述背后的关键洞见在于，DETR中的每个查询由两部分构成：内容部分（解码器自注意力输出）和位置部分（如DETR中的可学习查询）[1]。交叉注意力权重的计算通过将查询与一组键进行比较实现，这组键同样包含内容部分（编码后的图像特征）和位置部分（位置嵌入）。因此，Transformer解码器中的查询可解释为基于查询与特征的相似度度量从特征图中池化特征，该度量同时考虑了内容与位置信息。内容相似性用于池化语义相关的特征，而位置相似性则为在查询位置周围池化特征提供位置约束。这一注意力计算机制启发我们将查询建模为锚框，如图1(c)所示——利用锚框中心位置(x, y)来池化中心区域特征，并通过锚框尺寸(w, h)调节交叉注意力图，使其适应锚框大小。此外，由于将坐标作为查询使用，锚框能够逐层动态更新。如此一来，DETR中的查询可视为以级联方式逐层执行软性ROI池化操作。

我们通过利用锚框尺寸来调节交叉注意力，为特征池化提供了更优的位置先验。由于交叉注意力能够从整个特征图中汇聚特征，因此为每个查询提供恰当的位置先验至关重要，这能让交叉注意力模块聚焦于特定区域。

[1]See the DETR implementation at `https://github.com/facebookresearch/detr`. The components of queries and keys are also shown in each subplot of Fig. 1. Note that the learnable queries in DETR are only for the positional part. Related discussion can also be found in Conditional DETR (Meng et al., 2021).

与目标物体对应的局部区域。它还能加速DETR的训练收敛。大多数先前的工作通过将每个查询与特定位置关联来改进DETR，但它们假设了一个固定大小的各向同性高斯位置先验，这对于不同尺度的物体并不适用。利用每个查询锚框中的尺寸信息(w, h)，我们可以将高斯位置先验调制为椭圆形。更具体地说，我们将交叉注意力权重（在softmax之前）的宽度和高度分别除以其x部分和y部分，这有助于高斯先验更好地匹配不同尺度的物体。为了进一步提升位置先验，我们还引入了一个温度参数来调节位置注意力的平坦度，这一点在之前的所有工作中都被忽视了。

总之，我们提出的DAB-DETR（动态锚框DETR）通过直接学习锚点作为查询，提出了一种新颖的查询构建方式。这一构建方式深化了对查询作用的理解，使我们能够利用锚框尺寸来调节Transformer解码器中的位置交叉注意力图，并逐层执行动态锚框更新。实验结果表明，在相同设置下，DAB-DETR在COCO目标检测基准上达到了类DETR架构中的最佳性能。当使用单ResNet-50（He等人，2016）模型作为主干训练50个周期时，所提方法可实现45{v*}7%的AP。我们还进行了大量实验以验证分析结论，并证实了方法的有效性。

2 相关工作

大多数经典检测器都是基于锚点的，它们要么使用锚框（Ren等人，2017；Girshick，2015；Sun等人，2021），要么采用锚点（Tian等人，2019；Zhou等人，2019）。相比之下，DETR（Carion等人，2020）则是一种完全无锚的检测器，它利用一组可学习的向量作为查询。许多后续研究尝试从不同角度解决DETR收敛速度慢的问题。Sun等人（2020）指出，DETR训练缓慢的原因在于解码器中的交叉注意力机制，因此提出了一种仅包含编码器的模型。Gao等人（2021）则引入了高斯先验来规范交叉注意力。尽管这些方法提升了性能，但它们并未对训练速度慢的原因以及DETR中查询的作用给出合理解释。

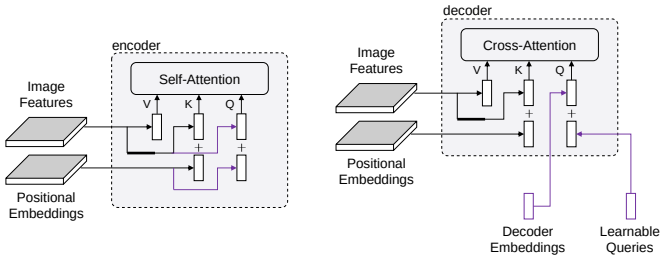
改进DETR的另一个方向——这与我们的工作更为相关——是更深入地理解查询在DETR中的作用。由于DETR中的可学习查询用于为特征池化提供位置约束，大多数相关研究试图让DETR中的每个查询更明确地与特定空间位置相关联，而非像原始DETR那样对应多个位置模式。例如，可变形DETR（Zhu等人，2021）直接将二维参考点作为查询，并为每个参考点预测可变形采样点以执行可变形交叉注意力操作。条件DETR（Meng等人，2021）解耦了注意力公式，基于参考坐标生成位置查询。高效DETR（Yao等人，2021）引入密集预测模块来选择前K个位置作为对象查询。尽管这些工作将查询与位置信息联系起来，但它们并未明确采用锚点的表述方式。

与先前工作中可学习查询向量包含框坐标信息的假设不同，我们的方法基于一个全新视角：查询中包含的所有信息皆为框坐标。即*anchor boxes are better queries for DETR*。同期工作Anchor DETR（Wang等人，2021）也提出直接学习锚点，但与其他先前工作一样忽略了锚框宽高信息。除DETR外，Sun等人（2021）通过直接学习框坐标提出稀疏检测器，其锚框公式与我们相似，但舍弃了Transformer结构而采用硬ROI对齐进行特征提取。表1总结了相关工作与我们提出的DAB-DETR的关键差异。我们从五个维度比较模型：是否直接学习锚框、是否预测中间阶段参考坐标、是否逐层更新参考锚框、是否使用标准稠密交叉注意力、注意力是否经调制以更好匹配不同尺度目标，以及是否逐层更新学习到的查询向量。附录B节提供了类DETR模型的详细对比，建议存在表格疑问的读者参阅该部分。

Models	Learn Anchors?	Reference Anchors	Dynamic Anchors	Standard Attention	Size-Modulated Attention	Update Learned Spatial Queries?
DETR	No	No		✓		
Deformable DETR	No	4D	✓		✓	
SMCA	No	4D		✓	✓	
Conditional DETR	No	2D		✓		
Anchor DETR	2D	2D	✓			
Sparse RCNN	4D	4D	✓			
DAB-DETR	4D	4D	✓	✓	✓	✓

表1：代表性相关模型与我们的DAB-DETR对比。"学习锚点？"指模型是否直接将2D点或4D锚点作为可学习参数。"参考锚点"表示模型是否基于参考点/锚点预测相对坐标。"动态锚点"指模型是否逐层更新锚点。"标准注意力"展示模型在交叉注意力模块中是否采用标准密集注意力。"物体尺度调制注意力"表示注意力机制是否经过调整以更好地匹配物体尺度。"尺寸调制注意力"指注意力是否根据物体尺度进行调制。"更新空间学习查询？"表示学习到的查询是否逐层更新。注意Sparse RCNN并非类DETR架构，因其锚点表述方式与我们相似而列入此表。更详细的模型对比参见附录B节。

3 为什么位置先验可以加速训练?



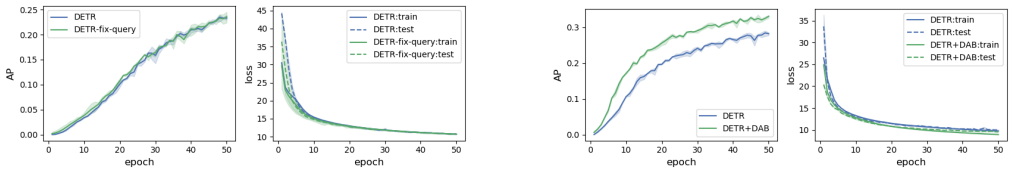
(a) Self-attention in encoder of DETR (b) Cross-attention in decoder of DETR

图2：DETR中编码器自注意力与解码器交叉注意力的对比。由于两者共享相同的键和值组件，唯一差异源于查询部分。编码器中的每个查询由图像特征（内容信息）和位置嵌入（位置信息）构成，而解码器中的每个查询则由解码器嵌入（内容信息）和可学习查询（位置信息）组成。两个模块间的差异已用紫色标出。

已有大量工作致力于加速DETR的训练收敛速度，但对其方法为何有效缺乏统一认识。Sun等人（2020年）指出交叉注意力模块是导致收敛缓慢的主因，但他们仅通过移除解码器来加速训练。我们沿袭其分析思路，探究交叉注意力中具体哪个子模块影响性能。通过比较编码器中的自注意力模块与解码器中的交叉注意力模块，发现两者输入的关键差异在于查询向量——如图2所示。由于解码器嵌入初始化为零值，它们会在首个交叉注意力模块后被映射至与图像特征相同的空间。此后，解码器层中的处理过程便与编码器层中图像特征的经历类似。因此根本原因很可能在于可学习的查询向量{v*}。

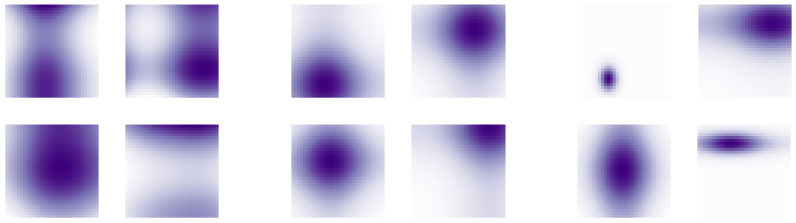
交叉注意力中导致模型训练收敛缓慢的两个可能原因：1）由于优化挑战，查询难以学习；2）所学查询中的位置信息编码方式与图像特征所用的正弦位置编码不一致。为验证是否为第一个原因，我们复用了DETR中已学习良好的查询（保持其固定），仅训练其他模块。图3(a)中的训练曲线表明，固定查询仅在极早期（如前25个周期）略微改善了收敛性。因此查询学习（或优化）很可能并非关键问题所在。

于是我们转向第二种可能性，试图探究学习到的查询是否具有某些不良特性。由于这些学习到的查询用于筛选特定区域内的对象，我们可视化了几组学习查询与图像位置嵌入之间的位置注意力图。



(a) (b)

图3：a) 原始DETR与固定查询DETR的训练曲线对比。b) 原始DETR与DETR+DAB的训练曲线对比。每组实验重复3次，绘制各项指标的均值及95%置信区间。



(a) DETR (b) Conditional DETR (c) DAB-DETR

图4：我们可视化了DETR、Conditional DETR以及我们提出的DAB-DETR中位置查询与位置键之间的位置注意力。(a)中随机采样了四张注意力图，(b)和(c)则选取了与(a)中查询位置相似的示例进行展示。颜色越深表示注意力权重越大，反之亦然。(a) DETR中的每张注意力图通过学习的查询与特征图位置嵌入进行点积计算得到，可能呈现多模态且注意力分散的特点。(b) Conditional DETR的位置查询采用与图像位置嵌入相同的编码方式，生成类高斯分布的注意力图，但无法适应不同尺度的目标。(c) DAB-DETR通过显式利用锚框的宽高信息调制注意力图，使其更适应目标尺寸和形状的变化。这种调制后的注意力可视为实现了软性ROI池化的效果。

图4(a)中的特征。每个查询可视为一种位置先验，使解码器专注于感兴趣区域。尽管它们作为位置约束发挥作用，但也携带了不良特性：多模态和近乎均匀的注意力权重。例如，图4(a)顶部的两个注意力映射存在两个或更多集中中心，当图像中存在多个物体时难以准确定位。图4(a)底部的映射则聚焦于过大或过小的区域，因而无法在特征提取过程中注入有效的定位信息。我们推测DETR中查询的多模态特性可能是其训练缓慢的根源，并认为引入显式位置先验将查询约束在局部区域有利于训练。为验证这一假设，我们用动态锚框取代DETR的查询公式，强制每个查询聚焦于特定区域，将该模型命名为DETR+DAB。图3(b)的训练曲线表明，在检测AP值和训练/测试损失方面，DETR+DAB相比DETR实现了显著提升。需注意DETR与DETR+DAB的唯一区别在于查询公式的设定，并未引入300个查询或焦点损失等技术。这表明在解决DETR查询的多模态问题后，我们既能实现更快的训练收敛，又能获得更高的检测精度。

先前的一些研究也进行了类似的分析并证实了这一点。例如，SMCA（Gao等人，2021年）通过在参考点周围应用预定义的高斯映射来加速训练。条件DETR（Meng等人，2021年）则采用显式位置嵌入作为训练时的位置查询，产生的注意力图类似于高斯核，如图4(b)所示。尽管显式位置先验在训练中带来了良好的性能，但它们忽略了物体的尺度信息。相比之下，我们提出的DAB-DETR明确考虑了物体的尺度信息，以自适应地调整注意力权重，如图4(c)所示。

4 DAB-DETR

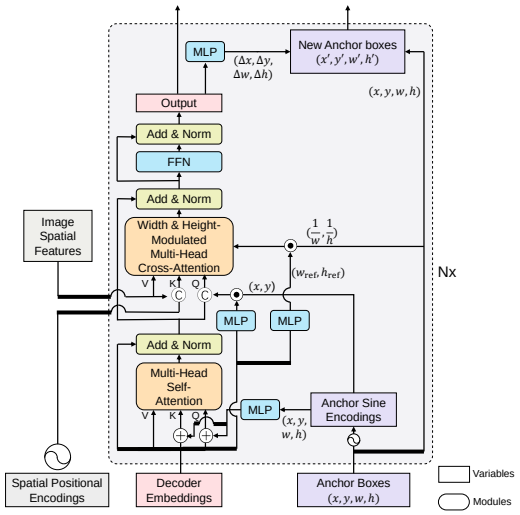


图5：我们提出的DAB-DETR框架。

4.1 概述

遵循DETR（Carion等人，2020年）的设计，我们的模型是一个端到端的目标检测器，包含CNN主干网络、Transformer（Vaswani等人，2017年）编码器与解码器，以及用于边界框和标签预测的头部结构。我们主要改进了解码器部分，如图5所示。

给定一张图像，我们首先利用CNN主干网络提取空间特征，随后通过Transformer编码器对CNN特征进行精炼。接着，将双重查询——包括位置查询（锚框）和内容查询（解码器嵌入向量）——输入解码器，以探测与锚框对应且与内容查询具有相似模式的目标物体。双重查询通过逐层更新逐步逼近真实标注目标。最终解码器层的输出被送入预测头，用于预测带有类别标签和边界框的物体，随后按照DETR的方式执行二分图匹配以计算损失。

为了展示我们动态锚框的通用性，我们还设计了一个更强的DAB-Deformable-DETR模型，具体内容见附录。

4.2 直接学习锚框

如第1节所述，关于查询在DETR中的作用，我们提出直接学习查询框（或称锚框），并从这些锚框派生出位置查询。每个解码器层包含两个注意力模块：自注意力模块和交叉注意力模块，分别用于查询更新和特征探测。每个模块需要查询、键和值来执行基于注意力的值聚合，但这些三元组的输入各有不同。

我们将$A_q = (x_q, y_q, w_q, h_q)$表示为$q$号锚点$x_q, y_q, w_q, h_q \in \mathbb{R}$，并将$C_q \in \mathbb{R}^D$和$P_q \in \mathbb{R}^D$定义为其对应的内容查询与位置查询，其中$D$为解码器嵌入及位置查询的维度。

给定一个锚点A_q，其位置查询P_q由以下方式生成：

$$P_q = \text{MLP}(\text{PE}(A_q)), \tag{1}$$

其中，PE表示位置编码，用于从浮点数生成正弦嵌入，且MLP的参数在所有层间共享。由于A_q是一个四元数，我们在此重载了PE运算符：

$$\text{PE}(A_q) = \text{PE}(x_q, y_q, w_q, h_q) = \text{Cat}(\text{PE}(x_q), \text{PE}(y_q), \text{PE}(w_q), \text{PE}(h_q)). \tag{2}$$

Cat的概念指的是拼接函数。在我们的实现中，位置编码函数PE将一个浮点数映射为一个$D/2$维向量，表示为：PE: $\mathbb{R} \to \mathbb{R}^{D/2}$。因此，函数MLP将2$D$维向量投影到$D$维：MLP: $\mathbb{R}^{2D} \to \mathbb{R}^{D}$。该MLP模块包含两个子模块，每个子模块由一个线性层和ReLU激活函数组成，特征降维在第一个线性层完成。

在自注意力模块中，查询、键和值三者均包含相同的内容项，而查询与键还额外包含了位置项：

$$\text{Self-Attn:} \quad Q_q = C_q + P_q, \quad K_q = C_q + P_q, \quad V_q = C_q, \tag{3}$$

受条件DETR（Meng等人，2021）的启发，我们将位置信息与内容信息在交叉注意力模块中拼接作为查询和键，从而能够解耦内容和位置对查询到特征相似度的贡献，该相似度通过查询与键的点积计算得出。为了重新缩放位置嵌入，我们也采用了条件空间查询（Meng等人，2021）。更具体地说，我们学习一个$\text{MLP}^{(\text{csq})} : \mathbb{R}^D \to \mathbb{R}^D$来获取基于内容信息的缩放向量，并用它与位置嵌入进行逐元素相乘：

$$\begin{aligned} \text{Cross-Attn:} \quad Q_q &= \text{Cat}(C_q, \text{PE}(x_q, y_q) \cdot \text{MLP}^{(\text{csq})}(C_q)), \\ K_{x,y} &= \text{Cat}(F_{x,y}, \text{PE}(x, y)), \quad V_{x,y} = F_{x,y}, \end{aligned} \tag{4}$$

其中$F_{x,y} \in \mathbb{R}^D$表示位置(x, y)处的图像特征，$\cdot$代表逐元素乘法运算。查询与键中的位置嵌入均基于二维坐标生成，这使得位置相似性的比较更为一致，如先前研究（Meng等人，2021；Wang等人，2021）所述。

4.3 锚点更新

使用坐标作为查询进行学习，使得逐层更新成为可能。相比之下，对于高维嵌入的查询，如DETR（Carion等人，2020）和Conditional DETR（Meng等人，2021）中所采用的，由于不清楚如何将更新后的锚点转换回高维查询嵌入，因此难以实现逐层查询优化。

遵循先前的研究实践（Zhu等，2021；Wang等，2021），我们在通过预测头预测相对位置（{v*}）后，逐层更新锚点，如图5所示。需注意的是，不同层中的所有预测头共享相同的参数。

4.4 宽度与高度调制的高斯核

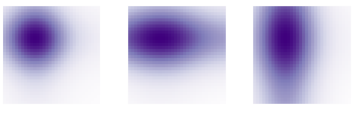


图6：由宽度和高度调制的空间注意力图。

图7：不同温度下的位置注意力图。

传统的注意力位置图被用作类似高斯的先验，如图6左侧所示。但这种先验简单地假设为各向同性且对所有物体固定大小，忽略了它们的尺度信息。

(宽度和高度)被忽略。为了改进位置先验，我们提出将尺度信息注入注意力图中。

原始位置注意力图中查询到键的相似度计算为两个坐标编码的点积之和:

$$\text{Attn}((x,y),(x_{\text{ref}},y_{\text{ref}})) = (\text{PE}(x)\cdot\text{PE}(x_{\text{ref}}) + \text{PE}(y)\cdot\text{PE}(y_{\text{ref}}))/\sqrt{D}, \tag{5}$$

其中$1/\sqrt{D}$用于按Vaswani等人(2017)的建议对数值进行重新缩放。我们通过分别从其x部分和y部分除以相对锚框宽度与高度，来调制位置注意力图（softmax前），从而平滑高斯先验，使其更好地匹配不同尺度的物体:

$$\text{ModulateAttn}((x,y),(x_{\text{ref}},y_{\text{ref}})) = (\text{PE}(x)\cdot\text{PE}(x_{\text{ref}})\frac{w_{q,\text{ref}}}{w_q} + \text{PE}(y)\cdot\text{PE}(y_{\text{ref}})\frac{h_{q,\text{ref}}}{h_q})/\sqrt{D}, \tag{6}$$

其中w_q和h_q是锚点A_q的宽度和高度，$w_{q,\text{ref}}$和$h_{q,\text{ref}}$则是通过以下公式计算得出的参考宽度和高度:

$$w_{q,\text{ref}}, h_{q,\text{ref}} = \sigma(\text{MLP}(C_q)). \tag{7}$$

这种调制的位置注意力机制帮助我们提取不同宽度和高度的物体特征，调制注意力的可视化效果如图6所示。

4.5 温度调谐

对于位置编码，我们采用正弦函数（Vaswani等人，2017年提出），其定义为:

$$\text{PE}(x)_{2i} = \sin(\frac{x}{T^{2i/D}}), \quad \text{PE}(x)_{2i+1} = \cos(\frac{x}{T^{2i/D}}), \tag{8}$$

其中T为人工设定的温度参数，上标$2i$和$2i+1$表示编码向量中的索引。如公式(8)中的温度T会影响位置先验的尺度，如图7所示。较大的T会导致注意力图更加平坦，反之亦然。值得注意的是，在自然语言处理中(Vaswani等人，2017)将温度T硬编码为10000，此时x的值为整数，代表单词在句子中的位置。而在DETR中，x的值为0到1之间的浮点数，表示边界框坐标。因此视觉任务亟需不同的温度设定。本工作中，我们通过实验在所有模型中选择$T=20$作为温度值。

5 实验

我们在附录A中提供了训练细节。

5.1 主要结果

表2展示了我们在COCO 2017验证集上的主要结果。我们将提出的DAB-DETR与DETR（Carion等人，2020）、Faster RCNN（Ren等人，2017）、Anchor DETR（Wang等人，2021）、SMCA（Gao等人，2021）、Deformable DETR（Zhu等人，2021）、TSP（Sun等人，2020）以及Conditional DETR（Meng等人，2021）进行了比较。我们展示了模型的两种变体：标准模型和标有上标*的模型（采用3种模式嵌入，Wang等人，2021）。我们的标准模型以显著优势超越了Conditional DETR。值得注意的是，我们的模型在GFLOPs上略有增加。由于计算脚本差异可能导致GFLOPs数值不同，表2中我们采用了原作者报告的结果。实际测试中发现，基于我们的GFLOPs计算脚本，标准模型的GFLOPs与对应Conditional DETR模型几乎相同，因此在相同配置下我们的模型仍具优势。当采用模式嵌入时，配备*的DAB-DETR在所有四种骨干网络上均大幅超越先前类DETR方法，甚至优于多尺度架构。这验证了我们分析的正确性及设计方案的有效性。

Model	MultiScale	#epochs	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	GFLOPs	Params
DETR-R50		500	42.0	62.4	44.2	20.5	45.8	61.1	86	41M
Faster RCNN-FPN-R50		108	42.0	62.1	45.5	26.6	45.5	53.4	180	42M
Anchor DETR-R50*		50	42.1	63.1	44.9	22.3	46.2	60.0	–	39M
Conditional DETR-R50		50	40.9	61.8	43.3	20.8	44.6	59.2	90	44M
DAB-DETR-R50		50	42.2	63.1	44.7	21.5	45.7	60.3	94	44M
DAB-DETR-R50*		50	**42.6**	63.2	45.6	21.8	46.2	61.1	100	44M
DETR-DC5-R50		500	43.3	63.1	45.9	22.5	47.3	61.1	187	41M
Deformable DETR-R50	✓	50	43.8	62.6	47.7	26.4	47.1	58.0	173	40M
SMCA-R50	✓	50	43.7	63.6	47.2	24.2	47.0	60.4	152	40M
TSP-RCNN-R50	✓	96	45.0	64.5	49.6	29.7	47.7	58.0	188	–
Anchor DETR-DC5-R50*		50	44.2	64.7	47.5	24.7	48.2	60.6	151	39M
Conditional DETR-DC5-R50		50	43.8	64.4	46.7	24.0	47.6	60.7	195	44M
DAB-DETR-DC5-R50		50	44.5	65.1	47.7	25.3	48.2	62.3	202	44M
DAB-DETR-DC5-R50*		50	**45.7**	66.2	49.0	26.1	49.4	63.1	216	44M
DETR-R101		500	43.5	63.8	46.4	21.9	48.0	61.8	152	60M
Faster RCNN-FPN-R101		108	44.0	63.9	47.8	27.2	48.1	56.0	246	60M
Anchor DETR-R101*		50	43.5	64.3	46.6	23.2	47.7	61.4	–	58M
Conditional DETR-R101		50	42.8	63.7	46.0	21.7	46.6	60.9	156	63M
DAB-DETR-R101		50	43.5	63.9	46.6	23.6	47.3	61.5	174	63M
DAB-DETR-R101*		50	**44.1**	64.7	47.2	24.1	48.2	62.9	179	63M
DETR-DC5-R101		500	44.9	64.7	47.7	23.7	49.5	62.3	253	60M
TSP-RCNN-R101	✓	96	46.5	66.0	51.2	29.9	49.7	59.2	254	–
SMCA-R101	✓	50	44.4	65.2	48.0	24.3	48.5	61.0	218	50M
Anchor DETR-R101*		50	45.1	65.7	48.8	25.8	49.4	61.6	–	58M
Conditional DETR-DC5-R101		50	45.0	65.5	48.4	26.1	48.9	62.8	262	63M
DAB-DETR-DC5-R101		50	45.8	65.9	49.3	27.0	49.8	63.8	282	63M
DAB-DETR-DC5-R101*		50	**46.6**	67.0	50.2	28.1	50.5	64.1	296	63M

表2：我们的DAB-DETR与其他检测模型的结果对比。除DETR外，所有类DETR模型均使用300个查询，而DETR使用100个。带有上标*的模型采用了Anchor DETR（Wang等人，2021）中的3种模式嵌入。我们还在附录G和附录C中提供了DAB-DETR更强劲的结果。

#Row	Anchor Box (4D) vs. Point (2D)	Anchor Update	wh-Modulated Attention	Temperature Tuning	AP
1	4D	✓	✓	✓	45.7
2	4D		✓	✓	44.0
3	4D	✓		✓	45.0
4	2D	✓		✓	44.0
5	4D	✓	✓		44.4

表3：我们的DAB-DETR消融实验结果。所有模型均在ResNet-50-DC5骨干网络上测试，其余参数与默认设置保持一致。

5.2 消融实验

表3展示了我们模型中各组成部分的有效性。研究发现，我们提出的所有模块对最终结果均有显著贡献。与锚点表示法相比（对比第3行与第4行数据），锚框表示法将性能从44.0% AP提升至45.0% AP；而锚框动态更新机制则带来了1.7% AP的性能提升（对比第1行与第2行数据），这验证了动态锚框设计的有效性。

移除调制注意力和温度调节后，模型性能分别降至45.0%（对比第1行与第3行）和44.4%（对比第1行与第5行）。因此，位置注意力的细粒度调谐对于提升检测性能同样至关重要。

6 结论

本文提出了一种新颖的查询构建方法，采用动态锚框作为DETR的查询机制，并深入探讨了查询在DETR中的作用。使用锚框作为查询带来多项优势，包括通过温度调节获得更优的位置先验、尺寸调制

关注不同尺度物体的处理，并通过迭代锚点更新逐步优化锚点估计。这一设计明确表明，DETR中的查询可以以级联方式逐层执行软性ROI池化操作。大量实验不仅有效支持了我们的分析，也验证了算法设计的合理性。

致谢

本研究得到了国家重点研发计划（2020AAA0104304、2020AAA0106000、2020AAA0106302）、国家自然科学基金项目（61620106010、62061136001、61621136008、62076147、U19B2034、U1811461、U19A2081）、北京市自然科学基金项目（JQ19016）、北京智源人工智能研究院（BAAI）、清华大学-阿里巴巴联合研究计划、清华大学国强研究院、清华大学-OPPO未来终端技术联合研究中心的资助。

道德声明

目标检测是计算机视觉中的一项基础任务，应用广泛。因此，该领域的任何改进都将产生深远影响。自动驾驶车辆高度依赖这一技术来实现对环境的视觉感知与交互，其性能提升将使这类系统直接受益。该技术还推动了医学影像分析、文字识别、自然图像实例分割等领域的进步。一旦模型失效，便可能波及众多下游任务。本研究深入探讨了DETR中查询机制的作用，从而提升了这一端到端基于Transformer的检测框架重要子模块的可解释性。

由于我们的模型依赖于深度神经网络，它可能受到对抗样本的攻击。同样，由于依赖训练数据，模型可能会因训练样本而产生有偏差的结果。这些都是深度学习中常见的问题，我们的研究社区正共同努力改进它们。最后值得注意的是，检测模型，尤其是人脸或人体检测模型，若被心怀不轨之人利用，可能会对人们的隐私和安全构成威胁。

可复现性声明

我们确认了结果的可复现性。复现我们结果所需的所有材料将在盲审后发布。我们也将开源代码。

参考文献

Alexey Bochkovskiy、Chien-Yao Wang与洪元马克·廖。YOLOv4：目标检测的最佳速度与精度。*arXiv preprint arXiv:2004.10934*，2020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 Conferenceon Computer Vision*，第213–229页。Springer，2020年。 戴熙阳、陈寅鹏、杨建伟、张鹏川、袁璐与张磊。动态DETR：基于动态注意力的端到端目标检测。载于*Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*，第2988–2997页，2021年10月。 高鹏、郑明航、王晓刚、代继峰与李洪胜。空间调制协同注意力加速DETR收敛。*arXiv preprint arXiv:2101.07448*，2021年。 葛政、刘松涛、王峰、李泽明与孙剑。YOLOX：2021年超越YOLO系列。*arXiv preprint arXiv:2107.08430*，2021年。 Ross Girshick。Fast R-CNN。载于*2015 IEEE International Conference on Computer Vision (ICCV)*，第1440–1448页，2015年。 何恺明、张翔宇、任少卿与孙剑。深入研究整流器：超越ImageNet分类的人类水平性能。载于*2015 IEEE International Conference onComputer Vision (ICCV)*，第1026–1034页，2015年。

何恺明、张翔宇、任少卿、孙剑。深度残差学习在图像识别中的应用。于
2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)，第770–778页，2016年。

林宗仪、Michael Maire、Serge Belongie、James Hays、Pietro Perona、Deva Ramanan、Piotr Dollár与C Lawrence Zitnick。微软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，2014年。

林宗仪、普里亚·戈亚尔、罗斯·吉斯克、何恺明与皮奥特·多拉尔。密集目标检测中的焦点损失。《*IEEE Transactions on Pattern Analysis and Machine Intelligence*》，第42卷第2期，第318至327页，2020年。

伊利亚·洛什奇洛夫和弗兰克·胡特。解耦权重衰减正则化。发表于*International Conference on Learning Representations*，2018年。

孟德普、陈晓康、范泽佳、曾刚、李厚强、袁玉辉、孙磊、王井东。条件式DETR实现快速训练收敛。*arXiv preprint arXiv:2108.06152*，2021年。

约瑟夫·雷德蒙、桑托什·迪瓦拉、罗斯·吉尔希克和阿里·法哈迪。你只需看一次：统一的实时目标检测。载于*2016 IEEE Conference on Computer Vision and Pattern Recognition(CVPR)*，第779–788页，2016年。

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。*IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149, 2017。

哈米德·雷扎托菲吉、内森·崔、郭俊英、阿米尔·萨德吉安、伊恩·里德和西尔维奥·萨瓦雷斯。广义交并比：一种用于边界框回归的度量与损失函数。载于*Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*，第658–666页，2019年。

孙培泽，张如峰，姜毅，孔涛，徐晨风，战伟，富塚雅义，李磊，袁泽寰，王长虎，罗平。稀疏R-CNN：基于可学习建议框的端到端目标检测。载于
Proceedings of the IEEE/CVF Conference on Computer Vision andPattern Recognition，第14454–14463页，2021年。

孙志清、曹圣操、杨一鸣与Kris Kitani。重新思考基于Transformer的集合预测在目标检测中的应用。*arXiv preprint arXiv:2011.10881*，2020年。

支天，沈春华，陈浩，何通。FCOS：全卷积一阶段目标检测。载于
2019 IEEE/CVF International Conference on Computer Vision (ICCV)，第9627–9636页，2019年。

阿希什·瓦斯瓦尼（Ashish Vaswani）、诺姆·沙泽尔（Noam Shazeer）、尼基·帕尔马（Niki Parmar）、雅各布·乌兹科雷特（Jakob Uszkoreit）、利昂·琼斯（Llion Jones）、艾丹·N·戈麦斯（Aidan N Gomez）、卢卡什·凯泽（ ukasz Kaiser）和伊利亚·波洛苏金（Illia Polosukhin）。注意力就是你所需要的一切。载于*Advances in neural informationprocessing systems*，第5998–6008页，2017年。

王烽、徐斌、张翔宇、杨桐和孙健。Anchor DETR：基于Transformer的检测器查询设计。*arXiv preprint arXiv:2109.07107*，2021年。

姚竹雨、艾江波、李伯勋和张驰。高效DETR：利用密集先验改进端到端目标检测器。*arXiv preprint arXiv:2104.01318*，2021年。

周星熠、王德全与Philipp Krähenbühl。目标即点。*arXiv preprintarXiv:1904.07850*，2019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*ICLR 2021: The Ninth International
Conference on Learning Representations*，2021年。

DAB-DETR 附录

训练详情

架构。我们的模型与DETR几乎相同，包含一个CNN主干网络、多个Transformer（Vaswani等人，2017）编码器和解码器，以及用于边界框和标签预测的两个预测头。我们采用ImageNet预训练的ResNet（He等人，2016）作为主干网络，并在实现中使用了6个Transformer编码器和6个Transformer解码器。我们遵循先前工作，报告了四种主干网络的结果：ResNet-50、ResNet-101，以及它们的16×分辨率扩展版本ResNet-50-DC5和ResNet-101-DC5。由于需要在每个解码器层预测边界框和标签，用于边界框和标签预测的MLP网络在不同解码器层间共享相同参数。受Anchor DETR启发，我们还利用多模式嵌入在单一位置进行多重预测，模式数量设置为3，与Anchor DETR保持一致。此外，我们采用PReLU（He等人，2015）作为激活函数。

遵循Deformable DETR和Conditional DETR的做法，我们采用300个锚点作为查询。同样选取分类逻辑值最大的300个预测框及对应标签用于评估。分类任务采用Focal Loss（Lin等人，2020），参数设定为$\alpha = 0.25, \gamma = 2$。双向匹配与最终损失计算采用相同损失项但系数不同：分类损失系数在双向匹配中为2.0，最终损失中为1.0；L1损失系数5.0与GIOU损失（Rezatofighi等人，2019）系数2.0在匹配与最终计算中保持一致。所有模型均在16块GPU上训练，每GPU处理1张图像，使用AdamW优化器（Loshchilov & Hutter，2018），权重衰减设为10^{-4}。骨干网络与其他模块学习率分别设置为10^{-5}和10^{-4}。训练50个周期，40周期后学习率下降0.1。实验均在NVIDIA A100 GPU完成，超参数搜索采用批量64，论文所有结果基于批量16报告。为便于复现，表4提供了显存需求及每GPU批量配置信息。

数据集。我们在COCO（Lin等人，2014）目标检测数据集上进行实验。所有模型均在train2017分割上训练，并在val2017分割上评估。

Model	Batch Size/GPU	GPU Memory (MB)
DAB-DETR-R50	2	6527
DAB-DETR-R50*	1	3573
DAB-DETR-R50-DC5	1	13745
DAB-DETR-R50-DC5*	1	15475
DAB-DETR-R101	2	6913
DAB-DETR-R101*	1	4369
DAB-DETR-R101-DC5	1	13148
DAB-DETR-R101-DC5*	1	16744

表4：各模型的GPU内存使用情况。

B DETR类模型的比较

在本节中，我们将对DETR类模型进行更详细的比较，包括DETR（Carion等人，2020）、Conditional DETR（Meng等人，2021）、Anchor DETR（Wang等人，2021）、Deformable DETR（Zhu等人，2021）、我们提出的DAB-DETR以及DAB-Deformable-DETR。这些模型的设计如图8所示。我们将讨论先前模型与我们模型之间的差异。

Anchor DETR（Wang等人，2021年）通过引入逐层更新的2D锚点改进了DETR，其动机与我们的工作相似。但该方法未考虑物体尺度信息，因此无法调节交叉注意力以适应不同尺度的物体。此外，其框架中的位置查询维度较高，且未经任何调整就传递至所有层的自注意力模块中（详见图8(d)棕色部分）。由于自注意力模块无法利用不同层中精炼的锚点信息，这一设计可能并非最优。

可变形DETR（Zhu等人，2021年）引入了4D锚框并逐层更新，其论文中称之为*iterative bounding box refinement*。该算法主要基于可变形注意力机制开发，需要参考点来采样注意力点，同时利用框的宽度和高度调节注意力区域。然而，由于*iterative bounding box refinement*与可变形注意力的特殊设计紧密耦合，将其应用于基于通用Transformer解码器的DETR模型并非易事。这可能是为何在可变形DETR之后鲜有工作采纳这一理念的原因。此外，可变形DETR中的位置查询在所有层的自注意力模块和交叉注意力模块间直接传递，未作任何适配处理。

详见图8(e)中棕色标注部分。因此，其自注意力模块和交叉注意力模块均无法充分利用不同层级中优化后的锚框{v*}。

为了验证我们的分析，我们开发了Deformable-DETR的一个变体，通过将其查询项建模为动态锚框，如DAB-DETR中所示。我们将此变体称为DAB-Deformable-DETR，其结构如图8(f)所示。在使用R50作为骨干网络的完全相同设置下，DAB-Deformable-DETR在COCO数据集上将Deformable-DETR的性能提升了0.5 AP（从46.3提高到46.8）。具体性能对比见表5，更多实现细节参见附录C。

动态DETR（Dai等人，2021）是DETR另一项引人注目的改进。该方法同样利用锚框进行特征池化，但采用ROI池化进行特征提取，这使得相较于我们提出的动态锚框，其通用性在类DETR模型中有所降低。此外，与Transformer解码器中基于注意力图进行全局软性特征池化的交叉注意力机制相比，ROI池化操作仅能在ROI窗口内执行局部特征池化。我们认为，ROI池化通过强制每个查询与特定空间位置关联，有助于加速模型收敛。但由于其忽略了ROI窗口外的全局上下文信息，可能导致次优的检测结果。

C DAB-可变形DETR

为进一步验证我们动态锚框的有效性，我们在可变形DETR（Zhu等人，2021）[2]的基础上引入动态锚框设计，开发了DAB-Deformable-DETR。图8(e)与(f)展示了可变形DETR与DAB-Deformable-DETR的架构差异，表5则呈现了两者的性能对比。仅需修改不超过10行代码，我们的DAB-Deformable-DETR（第4行）相较原始可变形DETR（第3行）实现了显著性能提升（+0.5 AP）。本实验中，除查询表述方式外，其余参数设置均保持完全一致。

我们还在图9中比较了收敛速度。结果显示，我们提出的动态锚框同样加速了训练过程（图9左）。我们认为性能提升的原因之一在于学习查询的更新。在图9中间的图表中，我们绘制了训练期间总损失的变化情况，即所有解码器层损失的总和。有趣的是，DAB-Deformable-DETR的总损失高于Deformable DETR。然而，DAB-Deformable-DETR最后一层的损失低于Deformable DETR（图9右），这是DAB-Deformable-DETR性能更优的良好指标，因为推理结果仅取自最后一层。

D 锚点可视化

我们在图10中可视化了学习到的锚框。当将锚点作为查询学习时，学习到的点在图像周围均匀分布，而直接学习锚框时，中心点似乎随机分布。这可能是因为中心点与锚框尺寸存在耦合关系。最右侧的图展示了学习到的锚框可视化结果。为了图像清晰，我们仅展示了部分集合。大多数锚框为中等尺寸，且未发现其分布存在特定规律。

[2]We used the open-source implementation from `https://github.com/fundamentalvision/Deformable-DETR`

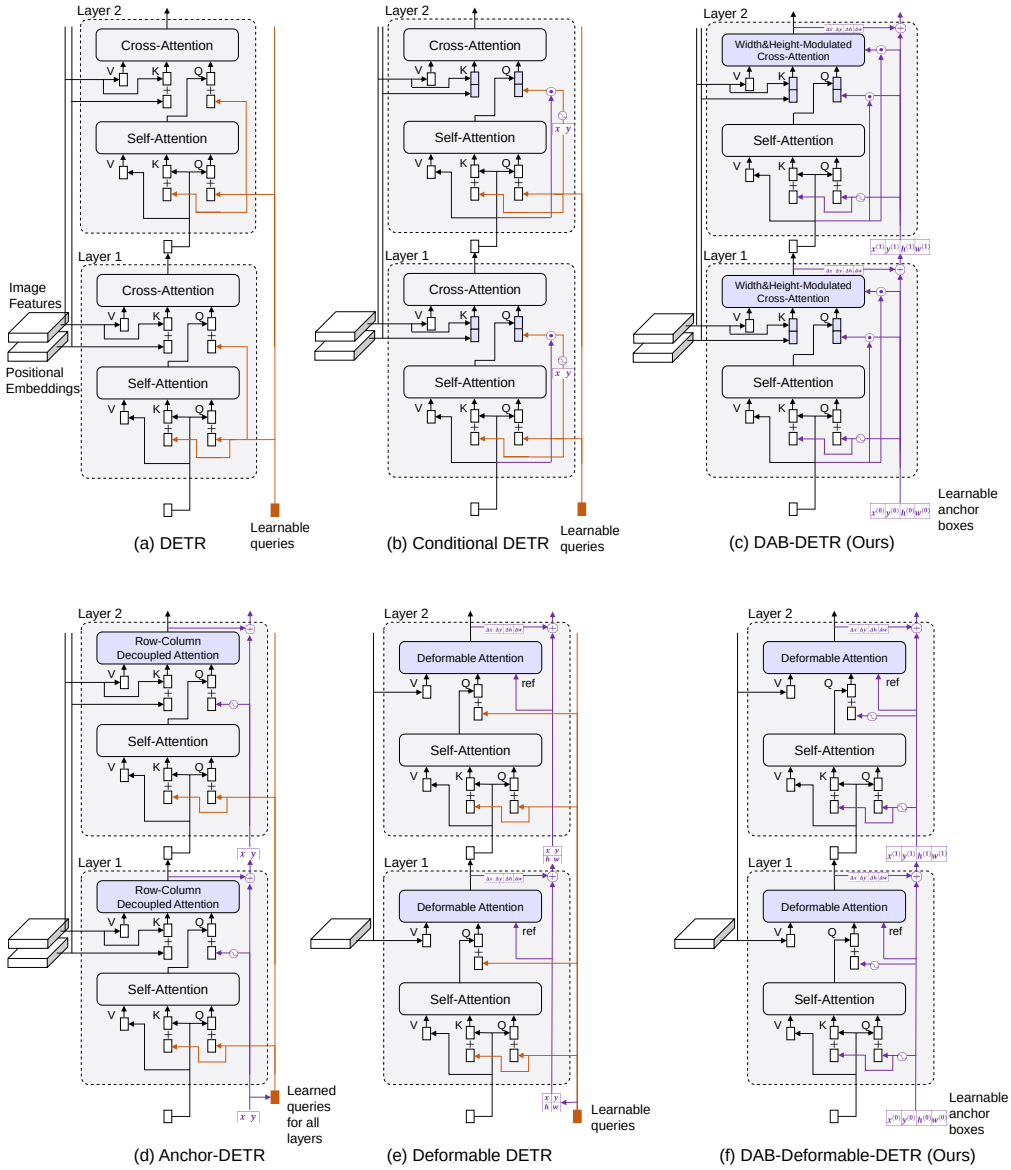


图8：DETR类模型对比。为清晰起见，我们仅展示两层Transformer解码器并省略了FFN模块。差异模块用紫色标注，学习到的高维查询用棕色标注。DAB-DETR（c）是我们论文提出的方法，DAB-Deformable-DETR（f）是通过引入动态锚框改进的Deformable DETR变体。先前所有模型（a,b,d,e）均采用高维查询（棕色阴影部分）向各层传递位置信息，这些查询语义模糊且不逐层更新。相比之下，DAB-DETR（c）直接使用动态更新的锚框同时提供参考查询点（x, y）和参考锚框尺寸（w, h）以改进交叉注意力计算。DAB-Deformable-DETR（f）同样采用动态更新的锚框来构建其查询。

# row	Model	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	Params
1	Deformable DETR	43.8	62.6	47.7	26.4	47.1	58.0	40M
2	Deformable DETR+	45.4	64.7	49.0	26.8	48.3	61.7	40M
3	Deformable DETR+ (open source)	46.3	65.3	50.2	28.6	49.3	62.1	47M
4	DAB-Deformable-DETR(Ours)	**46.8**	**66.0**	**50.4**	**29.1**	**49.8**	**62.3**	47M

表5：Deformable DETR与DAB-Deformable-DETR的结果对比。第1行和第2行的模型复制自原论文，第3行和第4行的模型在相同的标准R50多尺度设置下进行测试。Deformable DETR +表示带有迭代边界框优化的Deformable DETR模型，而Deformable DETR+ (开源)的结果是我们使用开源代码复现得出的。第3行与第4行的唯一区别在于查询的表述方式。

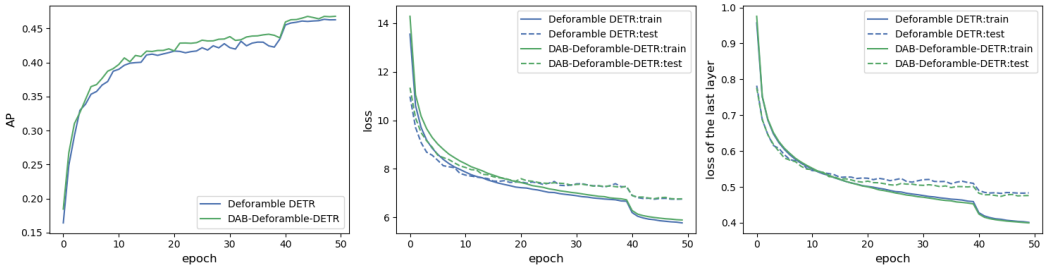


图9：Deformable DETR与DAB-Deformable-DETR模型的训练对比。我们分别绘制了训练过程中AP值的变化（左图）、所有层的损失（中图）以及最后一层损失（右图）的变化情况。仅修改不超过10行代码，DAB-Deformable-DETR就实现了比原始Deformable DETR模型更优的性能（见左图）。虽然DAB-Deformable-DETR所有层的损失值大于Deformable DETR（见中图），但我们的模型在最重要的最后一层表现出更低的损失（见右图），因为推理结果仅取决于最后一层输出。两个模型均在相同的标准R50多尺度设置下进行测试。

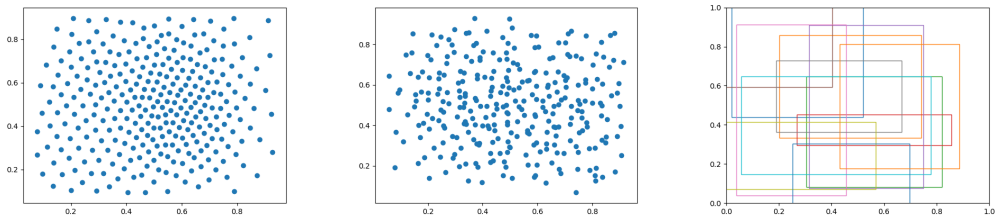


图10：仅学习2D坐标时的学习锚点（左），以及直接学习锚框时的锚中心点（中）和部分锚框（右）。

不同温度下的E结果

表6展示了在位置编码函数中使用不同温度值的模型结果。由于较大的温度会生成更为平坦的注意力图，这使得模型在检测较大物体时表现更优。例如，采用$T = 2$的模型与采用$T = 10000$的模型在AP指标上结果相近，但前者在AP_S和AP_M上表现更好，而后者则在AP_L上更胜一筹，这也验证了位置先验在DETR中的作用。

Temperature	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
2	39.6	60.7	41.9	19.3	43.3	58.0
5	40.0	61.1	42.1	19.5	43.4	58.9
10	40.0	61.1	42.3	19.7	43.5	59.3
20	40.1	61.1	42.8	19.8	43.7	58.6
50	39.8	61.0	42.2	19.7	43.2	58.8
100	39.8	60.8	42.1	19.3	43.3	58.4
10000	39.5	60.7	41.7	18.9	42.6	58.9

表6：不同温度下模型的比较。所有模型均采用ResNet-50主干网络，批量大小为64，未使用多重模式嵌入，也未采用调制注意力机制。其余参数均采用默认设置。

F 使用较少解码器层的结果

表7展示了不同解码器层数模型的结果。除解码器层数外，所有模型均在我们的标准ResNet-50-DC设置下进行训练。

decoder layers	GFLOPs	Parmas	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
2	202	36M	40.2	59.0	42.9	22.2	43.5	55.4
3	206	38M	43.9	63.4	47.4	24.6	47.8	60.5
4	210	40M	44.9	64.5	48.2	25.9	48.5	61.0
5	213	42M	45.2	65.5	48.6	26.6	48.9	62.3
6	216	44M	45.7	66.2	49.0	26.1	49.4	63.1

表7：不同解码器层数模型的比较。除解码器层数外，所有模型均在我们的标准ResNet-50-DC设置下训练。

G 固定 x, y 以获得更好的性能

在本节中，我们提供了一个有趣的实验。众所周知，所有边界框坐标x, y, h, w都是从数据中学习得到的。当我们用随机初始化的锚框x, y进行固定时，模型的性能持续提升。标准DAB-DETR与固定x, y坐标的DAB-DETR对比结果如表8所示。需要注意的是，我们仅在首层固定x, y以防止其从数据中学习信息，但x, y会在其他层进行更新。我们推测，随机初始化并固定的x, y坐标有助于避免过拟合，这可能是导致该现象的原因。

H 框更新比较

为了进一步证明我们动态锚框设计的有效性，我们在图11中绘制了DAB-DETR与Conditional DETR逐层更新框的结果。所有类DETR模型都具有堆叠层结构，因此每一层的输出可视为一个精炼过程。然而，由于所有层共享高维查询，层间查询的更新并不稳定。如图11(b)黄色阴影部分所示，某些后续层预测的框质量反而低于前序层。

I 故障案例分析

图12展示了一些我们的模型预测效果不佳的样本。我们发现，当图像中存在密集物体、非常小的物体或非常大的物体时，我们的模型可能会遇到一些困难。为了

Model	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
DAB-DETR-R50*	42.6	63.2	45.6	21.8	46.2	61.1
DAB-DETR-R50*-fixed$x$&y	**42.9**(+0.3)	63.7	45.3	22.0	46.8	60.9
DAB-DETR-DC5-R50	44.5	65.1	47.7	25.3	48.2	62.3
DAB-DETR-DC5-R50-fixed$x$&y	**44.7**(+0.2)	65.3	47.9	24.9	48.2	62.0
DAB-DETR-DC5-R50*	45.7	66.2	49.0	26.1	49.4	63.1
DAB-DETR-DC5-R50*-fixed$x$&y	**45.8**(+0.1)	66.5	48.9	26.4	49.6	62.7
DAB-DETR-R101*	44.1	64.7	47.2	24.1	48.2	62.9
DAB-DETR-R101*-fixed$x$&y	**44.8**(+0.7)	65.4	48.2	25.1	48.9	63.1
DAB-DETR-DC5-R101*	46.6	67.0	50.2	28.1	50.5	64.1
DAB-DETR-DC5-R101*-fixed$x$&y	**46.7**(+0.1)	67.3	50.7	27.3	50.9	64.1

表8：DAB-DETR与固定锚点中心x, y的DAB-DETR对比。当使用随机值固定查询的x, y时，模型性能得到一致提升。带有上标*的模型采用与Anchor DETR相同的3种模式嵌入。

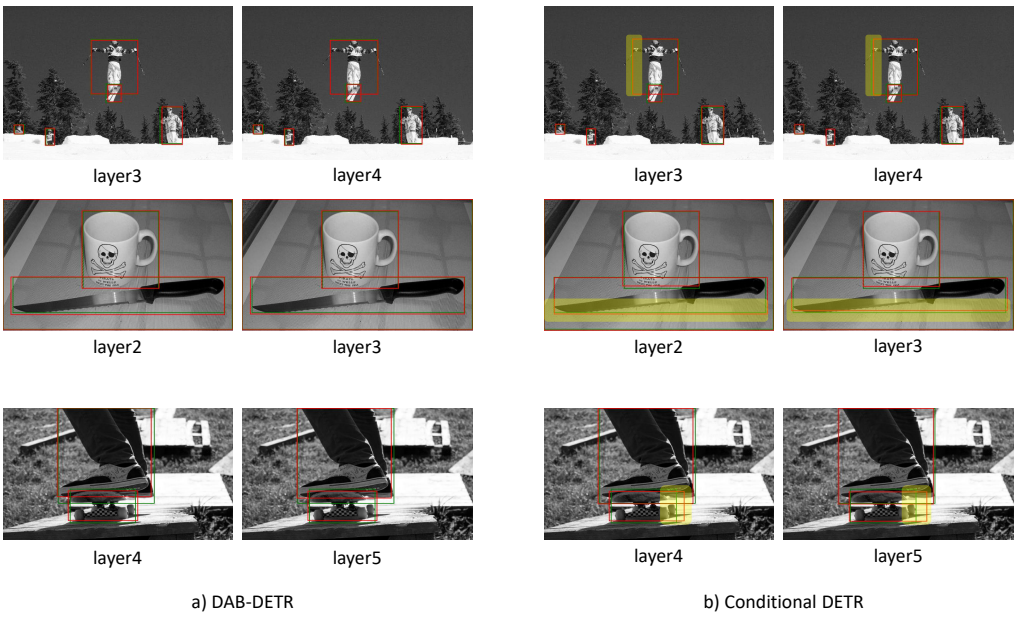


图11：我们对比了DAB-DETR（a）和条件式DETR（b）逐层更新的边界框。绿色框代表真实标注，红色框为模型预测结果。条件式DETR的边界框变化幅度较大，我们将部分变化显著的框边界用黄色标出。

为了提高我们模型的性能，我们将在模型中引入多尺度技术，以提升对小目标和大型目标的检测效果。

J 运行时间比较

我们在表9中比较了DETR、Conditional DETR和我们提出的DAB-DETR的运行时间。它们的运行速度是在单个Nvidia A100 GPU上报告的。与我们的直接竞争对手Conditional DETR相比，我们的DAB-DETR具有相似的推理速度，但性能更优。

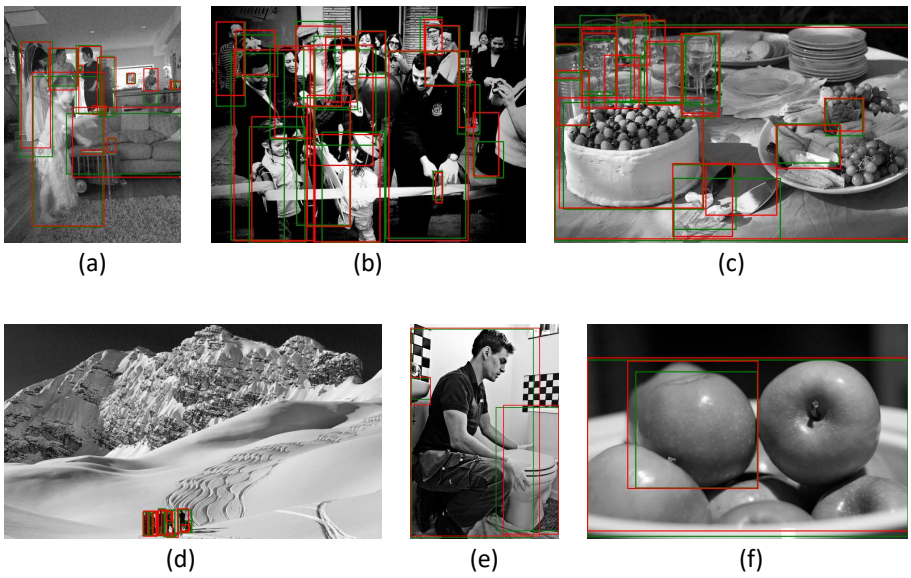


图12：我们可视化了一些模型预测效果不佳的图像，包括密集物体（a,b,c）、非常小的物体（d）以及非常大的物体（e,f）。绿色框为真实标注，红色框为模型的预测结果。

Model	time(s/img)	epoches	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	Parmas
DETR-R50	0.048	500	42.0	62.4	44.2	20.5	45.8	61.1	41M
Conditional DETR-R50	0.057	50	40.9	61.8	43.3	20.8	44.6	59.2	44M
DAB-DETR-R50	0.059	50	42.2	63.1	44.7	21.5	45.7	60.3	44M
DETR-R101	0.074	500	43.5	63.8	46.4	21.9	48.0	61.8	60M
Conditional DETR-R101	0.082	50	42.8	63.7	46.0	21.7	46.6	60.9	63M
DAB-DETR-R101	0.085	50	43.5	63.9	46.6	23.6	47.3	61.5	63M

表9：DETR、Conditional DETR与我们提出的DAB-DETR运行时间对比。所有速度数据均在单块Nvidia A100 GPU上测得。

K 模型收敛性比较

我们在图13中展示了DETR、Conditional DETR以及我们的DAB-DETR的收敛曲线。所有模型均在标准R50(DC5)配置下训练。结果证明了我们模型的有效性。我们的DAB-DETR采用了$fix\ x$&y变体进行训练。关于$fix\ x$&y结果的更多细节，请参阅附录G。Conditional DETR和我们的DAB-DETR均使用300个查询，而DETR则采用100个查询。

我们的DAB-DETR收敛速度比Conditional DETR更快，尤其在早期训练周期，如图13所示。

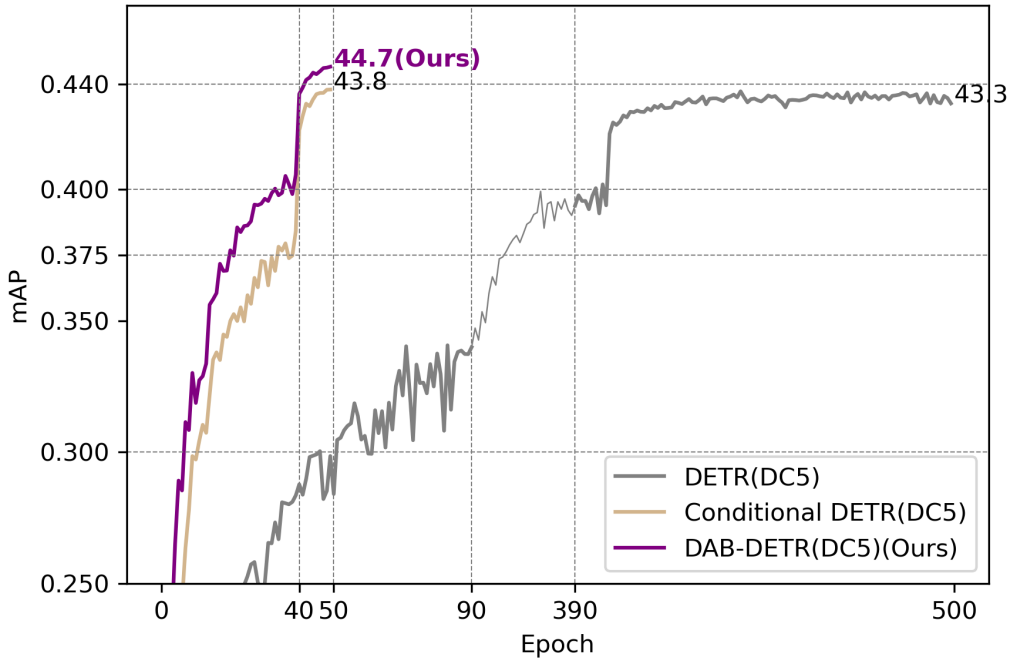


图13：DETR、Conditional DETR以及我们的DAB-DETR的收敛曲线。所有模型均在R50(DC5)设置下训练。