

# 基于混合像素编码器和代理注意力的实时实例分割模型

段淇超, 蒋三新

(上海电力大学 电子与信息工程学院, 上海 201306)

**摘要:**最近基于 Transformer 的模型在视觉任务上表现出色,其推理精度大大超越了传统卷积神经网络(convolutional neural network, CNN)模型,然而在推理速度方面却表现欠佳。提出一种基于 Transformer 的轻量级实时实例分割模型 FastInst-HA。首先模型采用混合像素编码器来融合多个特征层之间的图像信息,以增强网络对于不同尺度图像的特征提取能力,并为后续初始查询提供更多语义信息。在 Transformer 解码器中引入代理注意力(Agent-attention)来替换传统的自注意力(Self-attention)预测每个实例的类别。代理注意力避免  $q$  和  $k$  之间进行成对相似性计算,将自注意力的复杂性从二次方降低到线性,从而大大提升了模型的推理速度。实验结果表明, FastInst-HA 在 COCO 数据集上能够在 38.9 FPS 的实时速度下实现 38.2 的平均精度(average precision, AP),在推理速度和精度上超过了大多数实时实例分割模型。

**关键词:**实例分割; Transformer; 轻量级; 注意力机制; 编码器; 解码器

**中图分类号:** TP391.4; TP18; TP751

**文献标志码:** A

**文章编号:** 1003-7241(2026)10-0109-06

## Real-time instance segmentation model based on hybrid pixel encoder and Agent-attention

Duan Qichao, Jiang Sanxin

(School of Electronic and Information Engineering, Shanghai Electric Power University, Shanghai 201306, China)

**Abstract:** Recently, Transformer based models show excellent performance in visual tasks, with inference accuracy significantly surpassing traditional convolutional neural networks (CNN). However, they perform poorly in terms of inference speed. This paper proposes a lightweight real-time instance segmentation model based on Transformer (FastInst-HA). The model uses a hybrid pixel encoder to fuse image information between multiple feature layers, providing more semantic information for subsequent initial queries. Agent-attention is introduced in the Transformer decoder to predict the category of each instance. Agent-attention avoids pairwise similarity calculations between  $q$  and  $k$ , reducing the complexity of self-attention from quadratic to linear, thereby greatly improving the inference speed of the model. The experimental results show that FastInst-HA can achieve an average precision (AP) of 38.2 on the COCO dataset at a real-time speed of 38.9 FPS, surpassing most state-of-the-art real-time instance segmentation models in both speed and accuracy.

**Keywords:** instance segmentation; Transformer; lightweight; attention mechanism; encoder; decoder

实例分割是计算机视觉领域的一个重要研究方向,目的是分割图像中感兴趣的对象。在自动驾驶、智能监控、医学图像分析、人脸识别、工业自动化和遥感图像分析等领域,传统分割算法通常受限于其处理速度和空间占用而无法快速响应,此时则需要更加轻量高效的实时实例分割算法。YOLACT<sup>[1]</sup>将实例分割任务分解为两个并行的子任务:生成原型掩码(Prototype masks)和预测掩码系数(Mask coefficients)。通过这种方式,YOLACT能够在保持较高精度的同时实现实时处理。SOLOv2<sup>[2]</sup>通过将图像划分为网格,并在每个网格位置预测实例掩码,从而实现高效的实例分割。SOLOv2还使用了Matrix NMS来加速非

极大值抑制过程。这些实时分割算法在计算机视觉的多个领域有着广泛的应用。

最近几年,许多端到端的模型<sup>[3]</sup>尝试将可学习的查询应用在图像分割领域。对象查询为每一个基础真值分配一个相应的查询用于学习。在推理过程中,模型选择得分高的查询作为输出来推断对象类别以及分割掩码。由于其精简架构和不依赖于非极大值抑制(non-maximum suppression, NMS)的操作,简化了检测和分割模型的设计。SOTR<sup>[4]</sup>分别通过特征金字塔网络(feature pyramid network, FPN)<sup>[5]</sup>和孪生注意力(Twin Transformer)有效地提取底层特征表示并捕获远程上下文依赖关系。SOLO<sup>[6]</sup>

收稿日期:2025-09-21;录用日期:2025-10-20

基金项目:上海市科技计划项目(25ZR1401144)

作者简介:段淇超(1999—),男,硕士研究生,研究方向为机器视觉。

通信作者:蒋三新(1975—),男,博士研究生,讲师,研究方向为图像处理、信号处理等。

引用本文:段淇超, 蒋三新. 基于混合像素编码器和代理注意力的实时实例分割模型[J]. 自动化技术与应用, 2026, 45(10): 109-114. (Duan Qichao, Jiang Sanxin. Real-time instance segmentation model based on hybrid pixel encoder and Agent-attention[J]. Techniques of Automation and Applications, 2026, 45(10): 109-114.)

和 ISTR<sup>[7]</sup> 利用可学习令牌作为初始查询来推断掩码嵌入以进行实例分割。Panoptic SegFormer<sup>[8]</sup> 增加了一个位置解码器来提供物体的位置信息。Mask DINO<sup>[9]</sup> 使用从 DINO 查询嵌入到高分辨率像素嵌入图的点积来预测一组二值掩码,在实例分割上取得了很好的效果。

为了提高 Transformer 解码器的推理速度,一些研究也提出一些融合了高效注意力模块的模型。PVT<sup>[10]</sup> 通过使用稀疏全局注意力和渐进式收缩金字塔来降低高分辨率特征图的计算量。Swin Transformer<sup>[11]</sup> 使用可移动的窗口将自注意力计算限制在不重叠的局部窗口,同时允许跨窗口连接,从而提高了效率。NAT<sup>[12]</sup> 使用可扩展的滑动窗口注意力机制,通过计算每个像素邻域内的注意力来模拟卷积操作。DAT<sup>[13]</sup> 提出了一种新颖的可变形自注意模块。在这种模块中,自注意力根据数据选择 Key 和 Value 的位置。这使得自注意力能集中在对应领域,捕捉到更多信息特征。BiFormer<sup>[14]</sup> 使用双层路由注意力动态确定每个查询的兴趣区域。Mask2Former<sup>[15]</sup> 引入了掩码注意力 (Masked-attention),以提高解码器的收敛速度和性能。然而,这些方法在本质上限制了自注意力的全局感受野。

尽管 Transformer 模型取得了很多出色的性能,但仍然存在一些问题:1) 引入多尺度特征有利于加速训练收敛,但它会导致输入编码器的序列长度显著增加。多尺度特征交互会导致高额的计算成本使 Transformer 解码器难以应用在实时场景。2) 由于缺乏先验知识,可变形注意力需要大量的解码器层<sup>[16]</sup> 来解码对象查询。且自注意力的高计算复杂度限制了模型在高分辨率图像特征上的应用。为了更好地将 Transformer 架构扩展到边缘、嵌入式设备等实时场景中去,本文提出了一种新的轻量级实时实

例分割框架:FastInst-HA。本文对关键结构进行详细分析,以减少不必要的计算冗余,进一步提高训练速度与精度。

1) 在特征金字塔<sup>[17]</sup> (pyramid pooling module, PPM) 后加入跨尺度特征融合 (cross stage partial block, CSPBlock) 模块。它通过一系列卷积操作来提取图像的纹理特征,再融合相邻两个特征层之间的图像信息,为后续初始查询提供更多语义信息。

2) 对颈部混合像素编码器进行多层特征融合。FastInst-HA 能够有效地解耦不同尺度内的特征交互,这对于处理多尺度的图像和复杂场景至关重要,有助于提升模型的表现。

3) 将代理注意力 (Agent-attention)<sup>[18]</sup> 与 Transformer 解码器相结合,并将目标查询作为输入进行更新<sup>[15]</sup>。代理注意力极大地降低了全局自注意力的内存消耗和二次复杂度。代理令牌 (agent token) 的引入避免了查询令牌 Query 与原始 Key 和 Value 值直接计算,提高了目标查询的收敛速度。

## 1 算法介绍

### 1.1 整体架构

整体架构如图 1。FastInst-HA 由一个主干提取网络,一个混合像素编码器和一个 Transformer 解码器组成。输入图像通过骨干提取网络得到 3 个特征图  $C_3$ 、 $C_4$  和  $C_5$ , 它们的分辨率分别为输入图像的 1/8、1/16 和 1/32。然后通过  $1 \times 1$  卷积层将这 3 个特征映射投影到具有 256 个通道的特征映射,并将它们输入混合像素编码器中。编码器聚合上下文信息并输出增强后的多尺度特征图  $S_3$ 、 $S_4$  和  $S_5$ 。

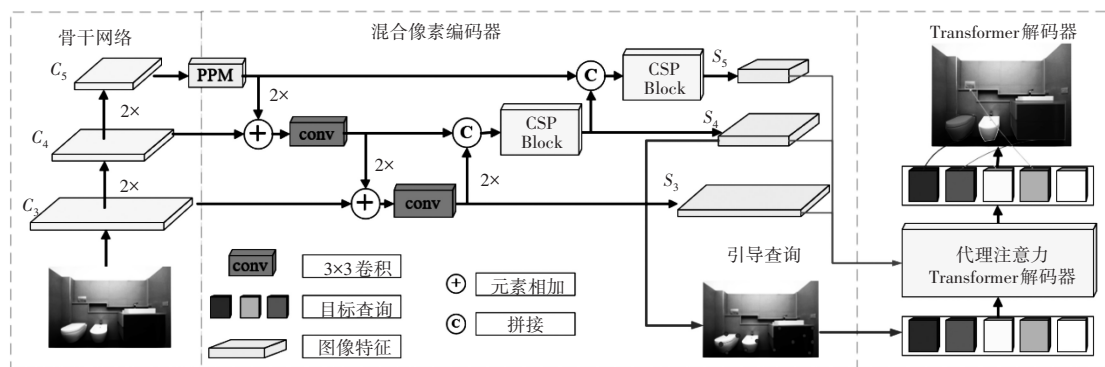


图 1 整体框架

Fig. 1 Overall framework

为了得到查询  $q$ , FastInst-HA 从混合像素编码器输出的特征图  $S_4$  中选择 100 个实例引导激活查询 (IA-guided queries) 作为初始查询。接着, Transformer 解码器将初始查询  $q$  分别和像素编码器输出的 3 个特征层  $S_3$ 、 $S_4$ 、 $S_5$  做计算并动态更新。最终和  $S_3$  特征层输出分割掩码图。

### 1.2 混合像素编码器

对于图像分割而言,过去的模型通常使用更大的骨干

网络或参数量更多的颈部来增强多尺度特征。高层特征层具有丰富的语义信息,能够捕捉更多实例类别之间的联系,便于后续模块对目标的定位和识别,因此如何聚合多尺度上下文信息十分重要。本文重新思考了编码器的结构,提出了一种高效的混合像素编码器。该编码器由融合了 PPM 池化模块的特征金字塔和基于 CNN 的跨尺度特征融合模块 (CSPBlock) 组成。如图 2 所示。

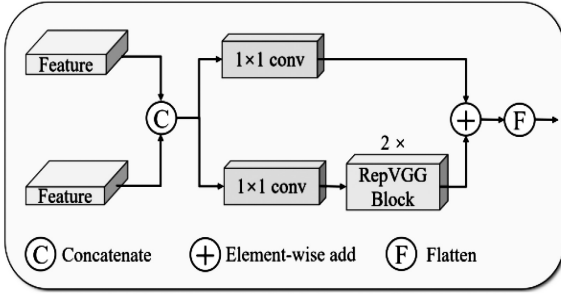


图 2 特征融合模块 (CSPBlock)

Fig. 2 Feature fusion module (CSPBlock)

图 2 中, CSPBlock 由 2 个特征融合通道以及多个卷积层组成。不同尺度特征层经过 2 个  $1 \times 1$  的卷积来调整通道数, 其中高维通道经过 2 个 Rep Vggblock<sup>[19]</sup> 用于特征融合, 再通过元素添加将 2 个相邻的尺度特征融合为一个新特征, 送入后续的 Transformer 解码器中用于计算。

不仅如此, 本文采用多级预测, 在不同的特征层次上共享解码器, 进一步提高模型在不同尺度上的性能和效率。高分辨率特征层更容易识别图像中小目标类别, 而低分辨率特征层包含更多的语义信息, 使用由低分辨率和高分辨率特征组成的特征金字塔, 并一次向一个 Transformer 解码器层提供一个多尺度特征的分辨率。而且模型在每一层 Transformer 解码器之前对图像嵌入进一个可学习的正弦位置信息, 来更好地更新查询  $q$ , 生产掩码图。

### 1.3 引导查询

目标查询 (object query) 是 Transformer 中一个输入向

量的集合, 它通过与像素编码器的输出特征进行交互, 从特征映射中提取相关信息, 来为图像中的每个像素块分配一个对象查询。通过这些查询向量, 模型可以生成对每个物体类别或物体实例的预测结果。然而, 由于它们的对象查询是零初始化的, 并不包含任何目标初始信息, 使得 Transformer 解码器收敛速度慢且推理精度低。本文使用 FastInst<sup>[20]</sup> 中的实例激活引导查询, 从像素编码器的  $S_4$  特征映射中选择具有高语义信息的参考点来初始化对象查询  $q$ 。由于模型只关注参考点周围的一小部分关键采样点, 而不考虑特征映射的空间大小, 可以减轻收敛和特征空间的分辨率问题。而且对象查询在初始化时保存了关于潜在对象的丰富信息, 从而提高了 Transformer 解码器中查询迭代的效率。

### 1.4 代理注意力 Transformer 解码器

如图 3(a) 所示, 一个标准的 Transformer 解码器层由 3 个模块组成: 自注意力模块 (self-Attention)、交叉注意力模块 (cross-attention) 和前馈网络 (feed-forward network, FFN)。注意力机制是 Transformer 模型的关键组成部分, 它减少了对外部信息的依赖, 更擅长捕捉全局上下文信息并学习输入序列中每个元素之间的远程交互。

为了提高查询  $q$  对图像特征的提取能力, 本文将 Agent-attention<sup>[17]</sup> 融入进解码器中, 如图 3(b)。Agent-attention 可以高效地将 Softmax 关注的二次复杂度 (在令牌数量上) 降低为线性复杂度。本文将 IA 引导查询  $q$  与不同分辨率像素特征  $x$  一起馈送到解码器, 然后通过连续的 Transformer 解码器层来更新它们。

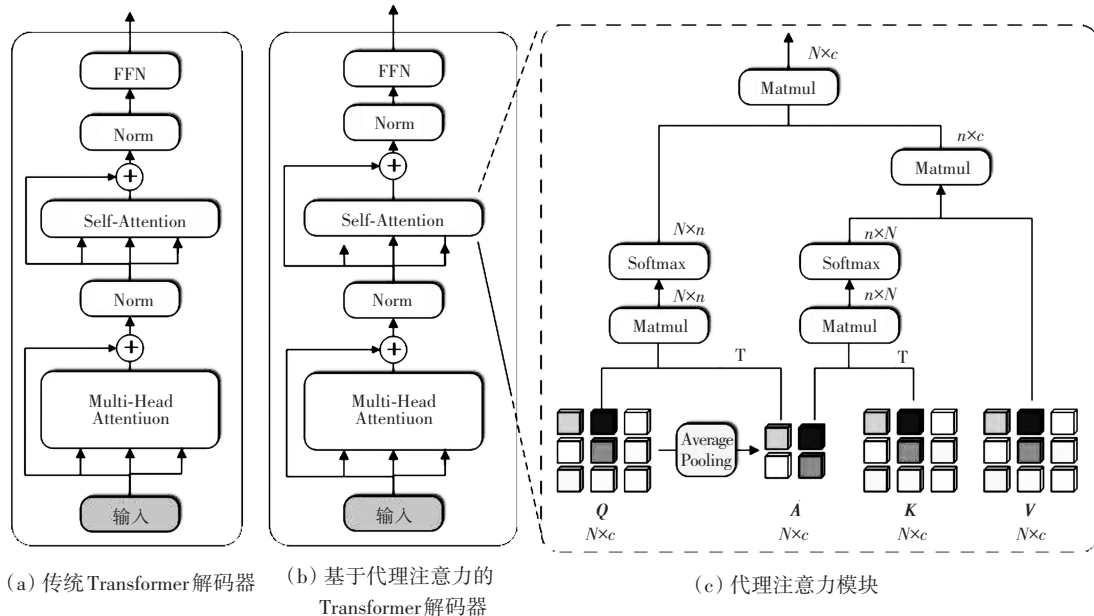


图 3 传统 Transformer 解码器与基于代理注意力的解码器的工作流程比较

Fig. 3 Comparison of workflows between traditional Transformer decoder and agent-attention based decoder

输入  $x$  可以表示为  $x \in \mathbb{R}^{N \times c}$ , 经过权重矩阵得到三个投影矩阵  $Q, K, V \in \mathbb{R}^{N \times c}$ 。其中  $Q, K, V$  为查询 (Query) 矩阵、键 (Key) 矩阵和值 (Value) 矩阵,  $N$  为查询令牌数,  $c$  为通道数, 缩放因子  $\sqrt{d}$  用于缓解高维空间点积值过大的

问题。Self-Attention 计算式如下。

$$\text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{QK^T}{\sqrt{d}}\right)V \quad (1)$$

由于自注意力需要计算所有查询键对之间的相似度,

导致时间复杂度和内存占用复杂度  $O(N^2c)$  都是二次的,这在图像等高维序列上会产生更高的计算成本,阻碍了模型在不同环境下的可扩展性。

代理注意力模块如图 3(c) 所示,它在注意力三元组  $(Q, K, V)$  进行 Softmax 计算之前引入了一组额外的代理令牌  $A$ ,从而组成注意力四元组  $(Q, A, K, V)$ 。本文  $\mathbb{R}$  对查询  $Q$  进行一次平均池化得到代理令牌  $A$ ,表示为  $A \in \mathbb{R}^{n \times c}$ 。代理令牌  $A$  不仅减少了引导查询矩阵的尺寸,而且保留了最重要的语义信息。于是代理注意力  $O_A$  可以表示为

$$O_A = \text{Attention}[Q, A, \text{Attention}(A, K, V)] \quad (2)$$

具体来说,在第一个 Softmax 中将代理令牌  $A$  视为查询,并在  $A, K$  和  $V$  之间执行注意力计算,以从所有值中聚合代理特征  $V_A$ 。

$$V_A = \text{Softmax}\left(\frac{AK^T}{\sqrt{d}}\right)V \quad (3)$$

随后,在三元组  $(Q, A, V_A)$  上执行第二次 Softmax,其中  $V_A$  是上一步计算的代理特征。利用  $A$  作为 key,  $V_A$  作为 value,利用查询矩阵  $Q$  进行第二次 Attention 计算。将特征映射的全局信息扩散到每个查询令牌上,得到最终输出  $O_A$ 。

$$O_A = \text{SoftMax}\left(\frac{QA^T}{\sqrt{d}}\right)V_A \quad (4)$$

通过 2 次 Softmax 操作将计算复杂度从原先的  $O(N^2c)$  降低至  $O(Nnc)$ 。这样既避免了  $Q$  和  $K$  的成对相似度计算,又保留了每个  $Q, K$  对之间的信息交换,实现了线性复杂性和高表达性的优点。

## 2 实验

本文对模型进行详细的消融研究,以验证每个模块的有效性。在 COCO 数据集上评估 FastInst-HA,并将其与几

种最先进的方法进行比较。

### 2.1 实验细节

本文在具有挑战性的 MS COCO 2017 数据集上进行了实验,该数据集包含 12.3 万张图像,其中训练集 11.8 万张,验证集 5 000 张,具有 80 个类别的实例标签。所有模型都在 COCO train2017 子集上进行训练。本文使用了标准的 COCO 评估指标,包括平均精度 (AP),IoU0.5 ( $AP_{50}$ ),0.75 ( $AP_{75}$ ) 的 AP 和不同尺寸  $AP_s, AP_m$  和  $AP_L$  对象的 AP,并在 COCO-val 验证集上进行测试。

本文使用 5 000 张 COCO 测试集图像在 RTX 3090 GPU 上验证平均 FLOPs 与 FPS,批处理大小为 1。本文将损失权重  $\lambda_{cls}, \lambda_{ce}$  和  $\lambda_{dice}$  分别设置为 2.0、5.0 和 5.0。 $\lambda_{cls-q}$  和  $\lambda_{loc}$  分别设为 20.0 和 1 000.0。使用 AdamW 优化器进行优化,学习率为 0.000 1,设置动量为 0.9,最后得出平均精度、GFLOPs 和 FPS。使用的编程语言为 Python,深度学习框架是 PyTorch 和 Detectron2。所有实验模型都在 24 GB RAM 的 RTX3090 GPU 上进行训练,批量大小为 16。

### 2.2 实验结果及分析

本文与 9 种先进的实时实例分割方法进行了比较,如表 1 所示。实验结果表明,模型在 COCO 数据集上实现了 38.2% 的 AP,在 RTX 3090 GPU 上达到 38.9 FPS,优于 COCO 基准上大多数先前最先进的方法。对于轻量级实例分割模型,计算复杂度和 AP 是两个需要权衡的重要指标。如表 1 所示,本文提出的模型的计算复杂度并不是最低的,GFLOPs 值为 66.5,略高于 OrienMask<sup>[21]</sup> 的 41.9 和 FastInst<sup>[20]</sup> 的 60.1,但低于所有其他模型。然而,FastInst-HA 模型实现了 38.2 的最高 AP,明显优于 YOLACT<sup>[1]</sup> 的 28.9 和 MEInst<sup>[22]</sup> 的 32.2。这表明模型在计算复杂度和平均精度之间取得了更好的平衡。

表 1 FastInst-HA 与最新轻量级实时实例分割方法的性能比较

Tab. 1 Performance comparison between FastInst-HA and state-of-the-art lightweight real-time instance segmentation methods

| method                     | 骨干网络       | epochs | FPS         | 参数/M        | GFLOPs      | AP          | AP <sub>50</sub> | AP <sub>75</sub> | AP <sub>s</sub> | AP <sub>m</sub> | AP <sub>L</sub> |
|----------------------------|------------|--------|-------------|-------------|-------------|-------------|------------------|------------------|-----------------|-----------------|-----------------|
| MEInst <sup>[22]</sup>     | R50        | 36     | 26.4        | —           | —           | 32.2        | 53.9             | 33.0             | 13.9            | 34.4            | 48.7            |
| CenterMask <sup>[23]</sup> | R50        | 48     | 32.6        | 51.2        | —           | 34.7        | —                | —                | 15.5            | 37.6            | 51.5            |
| SOLOv2 <sup>[2]</sup>      | R50        | 36     | 41.3        | 46.4        | 253.5       | 33.7        | 53.3             | 56.6             | 11.3            | 36.9            | 55.4            |
| OrienMask <sup>[21]</sup>  | Darknet-53 | 100    | 41.9        | —           | <b>41.9</b> | 34.5        | 56.0             | 35.8             | 16.8            | 38.5            | 49.1            |
| SparseInst <sup>[17]</sup> | R50        | 144    | 51.2        | 31.6        | 99.1        | 32.9        | 52.8             | 34.5             | 13.7            | 35.1            | 49.4            |
| YOLACT <sup>[1]</sup>      | R50        | 54     | <b>53.5</b> | 140.23      | 61.6        | 28.9        | 46.9             | 30.3             | 9.8             | 30.9            | 47.3            |
| YOLACT++ <sup>[24]</sup>   | R50        | 50     | 33.5        | —           | —           | 33.7        | 52.7             | 35.5             | 11.9            | 36.6            | 54.6            |
| FastInst <sup>[20]</sup>   | R50        | 50     | 35.5        | 31.0        | 60.1        | 37.9        | 59.4             | 39.5             | 16.0            | 40.7            | 60.1            |
| CondInst <sup>[25]</sup>   | R50        | 36     | 20.0        | 33.9        | 240.8       | 36.0        | 56.6             | 38.5             | <b>18.0</b>     | 39.4            | 50.4            |
| FastInst-HA (ours)         | R50        | 50     | 38.9        | 35.3        | 66.5        | <b>38.2</b> | <b>59.9</b>      | <b>40.0</b>      | 15.9            | <b>41.1</b>     | <b>60.4</b>     |
| YOLACT <sup>[1]</sup>      | R101       | 54     | 33.5        | 155.8       | 159.1       | 31.2        | 50.6             | 32.8             | 12.1            | 33.3            | 47.1            |
| CenterMask <sup>[23]</sup> | R101       | 50     | —           | 70.1        | —           | 36.0        | —                | —                | 16.5            | 39.2            | 54.4            |
| FastInst <sup>[20]</sup>   | R101       | 50     | <b>35.3</b> | <b>50.5</b> | <b>85.2</b> | 38.9        | 60.3             | 40.9             | 16.4            | 42.3            | 62.1            |
| SOLOv2 <sup>[2]</sup>      | R101       | 36     | 15.7        | 62.0        | 286.9       | 39.1        | 60.1             | <b>41.9</b>      | <b>17.0</b>     | <b>43.1</b>     | 58.4            |
| FastInst-HA (ours)         | R101       | 50     | 32.3        | 50.9        | 94.7        | <b>39.3</b> | <b>60.8</b>      | 41.3             | 16.7            | 42.8            | <b>62.6</b>     |

例如同样使用 ResNet50 作为主干, FastInst-HA 模型比 SparseInst<sup>[17]</sup> 性能高 5.3AP、比 OrienMask<sup>[21]</sup> 性能高 3.7AP。而在模型大小和推理速度方面, 模型参数量同 CondInst<sup>[25]</sup> 相近, 而 FPS 提升近一倍。不管是参数量还是运算速度上, 本文的模型都展现出惊人的优势。

骨干网的选择也对模型的性能有重大影响。当 FastInst-HA 使用 Resnet-50 和 Resnet-101 作为骨干网络时, 其 AP 值分别为 38.2 和 39.3, 这表明其性能稳步提升。此外, 虽然更强的骨干网络可以有效地改善模型的 AP, 但它们通常包含更多的参数, 导致计算复杂性增加。例如, 当 FastInst-HA 采用 R101 骨干网时, 其 AP 高达 39.3, 但其 GFLOPs 也增加到 94.7, FPS 从 38.9 下降到 32.3。在其他模型中也观察到类似的模式。这表明, 有必要根据实例分割的实际应用需求选择合适的骨干网。对于多数模型, FastInst-HA 在模型参数更少的情况下实现了更高的精度和更快的运行速度, 这也证明了本文方法的优

越性。

2.3 消融实验

本文通过一系列消融实验来分析模型的有效性。首先对提出的 3 个关键创新模块进行实验, 即混合像素编码器、多尺度融合架构和代理注意力, 不仅如此, 还对比研究了一些关于 FastInst-HA 的模块效果。在 COCO-val2017 数据集上进行评估消融结果, 除非特别说明, 所有对比实验都是在 ResNet-50 骨干网络上进行。

混合像素编码器。模型架构是本文的重点关注对象, 它会对模型参数、计算速度以及资源占用产生重大影响, 如何更好地聚合多尺度上下文信息会直接影响后续的分割结果。根据表 2, 本文通过替换不同 PPM 特征层传入 Transformer 的位置以及数量来验证融合的有效性。结果表明, 本文的多尺度混合像素编码器模块在不牺牲特别多推理速度的前提下极大地提升了平均精度, 增长了 3 个百分点。

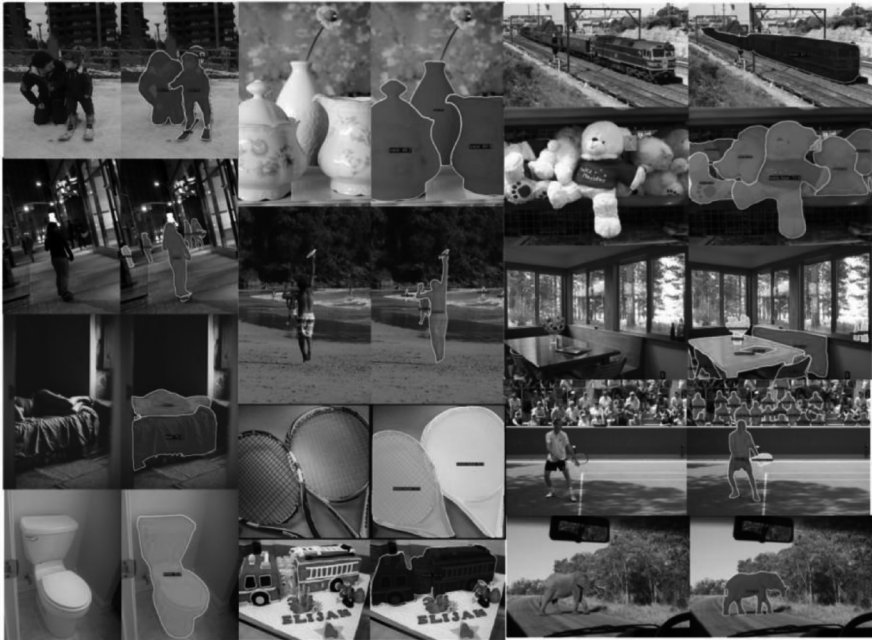


图 4 FastInst-HA 在 COCO 数据集上的实例分割效果图 (左侧为 COCO 数据集中原图, 右侧为分割掩码图)

Fig. 4 Instance segmentation results of FastInst-HA on the COCO dataset (Left: original images from the COCO dataset; Right: predicted masks)

表 2 混合像素编码器

Tab. 2 Hybrid pixel encoder

| 编码器层              | CSPBlock | AP   | GFLOPs | FPS  |
|-------------------|----------|------|--------|------|
| $S_3$             |          | 34.9 | 59.4   | 45.1 |
| $S_3$             | ✓        | 35.2 | 61.2   | 42.3 |
| $S_3 + S_4 + S_5$ |          | 37.9 | 63.4   | 40.9 |
| $S_3 + S_4 + S_5$ | ✓        | 38.2 | 66.5   | 38.9 |

注意力模块。本文使用自注意力、交叉注意力和代理注意力测试了 FastInst-HA 的性能, 结果如表 3 所示。从表 3 中可以看出, 当使用代理注意力时, 快速启动辅助系统的性能并不理想。从表中可以看出, 在使用代理注意力时, FastInst-HA 不仅获得了最高的准确率, 而且还获得了

更快的推理速度。这证明了 Agent-Attention 在实例分割的有效性。

表 3 注意力模块

Tab. 3 Attention module

| 注意力模块 | AP   | AP <sub>S</sub> | AP <sub>M</sub> | AP <sub>L</sub> | FPS  |
|-------|------|-----------------|-----------------|-----------------|------|
| 自注意力  | 38.1 | 15.7            | 40.8            | 60.3            | 38.7 |
| 交叉注意力 | 37.9 | 15.3            | 40.6            | 60.0            | 37.0 |
| 代理注意力 | 38.2 | 15.9            | 41.1            | 60.4            | 38.9 |

此外, 还可以通过改变初始查询  $q$  和代理令牌  $a$  的数量来调整 FastInst-HA 的性能。如表 4 所示, 查询  $q$  经过平均池化得到代理令牌  $a$ 。查询和代理令牌数量过多会增加注意力机制的计算复杂度, 过少则会导致性能下降。

在进行代理注意力计算时 Token 的值会影响目标实例的最终分类结果。经过多次参数选择和消融实验,本文最终确定了查询  $q$ (100) 和代理令牌  $a$ (25) 的最佳参数值。

表4 代理令牌和查询数量

Tab.4 Agent tokens and query numbers

| $q$        | Agent tokens | AP   | Param/M | GFLOPs |
|------------|--------------|------|---------|--------|
| 81         | 16           | 38.1 | 5.9     | 1.05   |
| 108        | 18           | 37.9 | 6.1     | 1.67   |
| <b>100</b> | 25           | 38.2 | 6.0     | 1.39   |

### 3 结论

本文提出的轻量级 FastInst-HA 用于实时实例分割。FastInst-HA 基于查询的分割框架和高效组件,即多尺度特征融合和代理注意力模块。在保持快速推理速度的同时,在 COCO 数据集上实现了出色的性能。大量的实验证明了本文的模型优于目前大多数分割模型和核心思想的有效性。希望这项工作能够促进基于查询的图像分割算法的发展。本文发现模型在处理小目标上并不那么令人满意,而其他很多基于查询的模型也暴露类似问题。尽管本文可以替换为更强的像素编码器或更大的骨干提取网络来改善它,但它带来了更多的计算负担,且效果并不明显。期待着未来能够解决这个问题。

### 参考文献

- [1] Bolya D, Zhou C, Xiao F, et al. YOLACT: real-time instance segmentation [C]//Proceedings of the IEEE/CVF International Conference on Computer Vision. Piscataway, NJ: IEEE, 2019: 9157–9166.
- [2] Wang X, Zhang R, Kong T, et al. SOLOv2: dynamic and fast instance segmentation[J]. Advances in Neural Information Processing Systems, 2020, 33: 17721–17732.
- [3] Dai X, Chen Y, Yang J, et al. Dynamic detr: End-to-end object detection with dynamic attention[C]//Proceedings of the IEEE/CVF International Conference on Computer vision. Piscataway, NJ: IEEE, 2021: 2988–2997.
- [4] Guo R, Niu D, Qu L, et al. SOTR: segmenting objects with Transformers [C]//Proceedings of the IEEE/CVF International Conference on Computer Vision. Piscataway, NJ: IEEE, 2021: 7157–7166.
- [5] Lin T Y, Dollár P, Girshick R, et al. Feature pyramid networks for object detection [C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Piscataway, NJ: IEEE, 2017: 2117–2125.
- [6] Dong B, Zeng F, Wang T, et al. SOLQ: segmenting objects by learning queries[J]. Advances in Neural Information Processing Systems, 2021, 34: 21898–21909.
- [7] Hu J, Lu Y, Zhang S, et al. ISTR: mask-embedding-based instance segmentation transformer[J]. IEEE Transactions on Image Processing, 2024, 33: 2895–2907.
- [8] Li Z, Wang W, Xie E, et al. Panoptic SegFormer: delving deeper into panoptic segmentation with Transformers [C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. Piscataway, NJ: IEEE, 2022: 1280–1289.
- [9] Li F, Zhang H, Xu H, et al. Mask DINO: towards a unified Transformer-based framework for object detection and segmentation[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. Piscataway, NJ: IEEE, 2023: 3041–3050.
- [10] Wang W, Xie E, Li X, et al. Pyramid vision Transformer: a versatile backbone for dense prediction without convolutions [C]//Proceedings of the IEEE/CVF International Conference on Computer Vision. Piscataway, NJ: IEEE, 2021: 568–578.
- [11] Liu Z, Lin Y, Cao Y, et al. Swin Transformer: hierarchical vision transformer using shifted windows [C]//Proceedings of the IEEE/CVF International Conference on Computer Vision. Piscataway, NJ: IEEE, 2021: 10012–10022.
- [12] Hassani A, Walton S, Li J, et al. Neighborhood attention Transformer [C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. Piscataway, NJ: IEEE, 2023: 6185–6194.
- [13] Xia Z, Pan X, Song S, et al. Vision Transformer with deformable attention [C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. Piscataway, NJ: IEEE, 2022: 4794–4803.
- [14] Zhu L, Wang X, Ke Z, et al. BiFormer: vision Transformer with bi-level routing attention [C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. Piscataway, NJ: IEEE, 2023: 10323–10333.
- [15] Cheng B, Misra I, Schwing A G, et al. Masked-attention mask Transformer for universal image segmentation [C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. Piscataway, NJ: IEEE, 2022: 1290–1299.
- [16] Zhu X, Su W, Lu L, et al. Deformable DETR: deformable Transformers for End-to-End object detection [C]//International Conference on Learning Representations. Austria: OpenReview, 2021: 1–16.
- [17] Cheng T, Wang X, Chen S, et al. Sparse instance activation for real-time instance segmentation [C]//Proceedings of the IEEE/CVF Conference on Computer vision and Pattern Recognition. Piscataway, NJ: IEEE, 2022: 4433–4442.
- [18] Han D, Ye T, Han Y, et al. Agent attention: on the integration of softmax and linear attention [C]//European Conference on Computer Vision. Cham: Springer Nature Switzerland, 2024: 124–140.
- [19] Ding X, Zhang X, Ma N, et al. Repvgg: making vgg-style convnets great again [C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. Piscataway, NJ: IEEE, 2021: 13733–13742.
- [20] He J, Li P, Geng Y, et al. Fastinst: a simple query-based model for real-time instance segmentation [C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. Piscataway, NJ: IEEE, 2023: 23663–23672.
- [21] Du W, Xiang Z, Chen S, et al. Real-time instance segmentation with discriminative orientation maps [C]//Proceedings of the IEEE/CVF International Conference on Computer Vision. Piscataway, NJ: IEEE, 2021: 7314–7323.
- [22] Zhang R, Tian Z, Shen C, et al. Mask encoding for single shot instance segmentation [C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. Piscataway, NJ: IEEE, 2020: 10226–10235.
- [23] Lee Y, Park J. Centermask: real-time anchor-free instance segmentation [C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. Piscataway, NJ: IEEE, 2020: 13906–13915.
- [24] D. Bolya, C. Zhou, F. Xiao, et al. YOLACT++ better real-time instance segmentation [J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2022, 44: 1108–1121.
- [25] Tian Z, Shen C, Chen H. Conditional convolutions for instance segmentation [C]//European Conference on Computer Vision. Cham: Springer International Publishing, Berlin: Springer, 2020: 282–298.