稳定匹配的检测变换器

Shilong Liu1,2*†,Tianhe Ren2□,Jiayu Chen3□,
Zhaoyang Zeng2,Hao Zhang2,4,Feng Li2,4,Hongyang Li2,5,
Jun Huang3,Hang Su1,Jun Zhu1‡,Lei Zhang2‡.
1清华大学计算机科学与技术系、BNRist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-博世联合智能技术中心
2国际数字经济研究院（IDEA）、阿里巴巴集团人工智能平台（PAI）
3香港科技大学
5华南理工大学
liusl20@mails.tsinghua.edu.cn frentianhe, zengzhaoyang g@idea.edu.cn fyunji.cjy, huangjun.hj g@alibaba-inc.com
fhzhangcx, fliay g@connect.ust.hk ftwangyeunglei@mail.scut.edu.cn fsuhangss, dcszj g@mail.tsinghua.edu.cn leizhang@idea.edu.cn

摘要
本文关注DEtection TRansformers（DETR）中在不同解码层之间的匹配稳定性问题。我们指出，DETR中不稳定的匹配是由于多优化路径问题引起的，这一问题通过DETR中的一对一匹配设计得到了突出体现。为了解决这个问题，我们指出最重要的设计是仅使用位置度量（如IOU）来监督正样本的分类分数。在这一原则下，我们提出了两种简单而有效的修改方法，通过将位置度量整合到DETR的分类损失和匹配代价中，分别命名为位置监督损失和位置调节代价。我们在几个DETR变种上验证了我们的方法。我们的方法在基准模型上表现出一致的改进。通过将我们的方法与DINO相结合，我们在ResNet-50骨干网络下，在1□（12个周期）和2□（24个周期）的训练设置下，在COCO检测基准上实现了50.4和51.5的AP，实现了相同设置下的新纪录。我们在COCO检测test-dev上实现了63.8 AP，使用Swin-Large骨干网络。我们的代码将在https://github.com/IDEA-Research/Stable-DINO上提供。

1. 引言
   *相等贡献。列表顺序是随机的。
   损失
   (1)†本论文是Shilong Liu、Hao Zhang、Feng Li和Hongyang Li在IDEA实习期间完成的。
   ‡通讯作者。
   图1：我们方法（在图中称为Stable-DINO）与基准方法的比较。我们在左图中比较使用ResNet-50骨干网络的模型，在右图中比较使用Swin-Transformer Large骨干网络的模型。所有模型都使用骨干网络的最大1/8分辨率特征图，除了AdaMixer使用最大1/4分辨率特征图。
   几十年来，随着深度学习，特别是卷积神经网络（CNN）[36, 14, 16, 7]的发展，目标检测取得了显著的进展。检测变换器（DETR）[3]提出了一种新型的基于Transformer的目标检测器，引起了研究界的广泛关注。它摆脱了所有手工设计的模块，并实现了端到端的训练。DETR的一个关键设计是匹配策略，它使用匈牙利匹配将预测结果一对一地分配给实际标签。尽管DETR具有新颖的设计，但也存在某些与这种创新方法相关的限制，包括收敛速度较慢和性能较差。许多后续研究从多个层面尝试改进DETR，例如引入位置先验[32, 41, 28, 14]，额外的正样本[22, 4, 5]和高效的操作符[47, 34]。通过进行许多优化，我们取得了一些令人鼓舞的结果。DINO [46]在COCO检测排行榜上创造了新记录，使得基于Transformer的方法成为大规模训练的主流检测器。
   尽管DETR等检测器取得了令人印象深刻的性能，但一个关键问题至今未受到足够重视，这可能会损害模型的训练稳定性。这个问题涉及不同解码器层之间的不稳定匹配问题。类似DETR的模型在Transformer解码器中堆叠多个解码器层。模型在每个解码器层之后进行预测和损失计算。然而，分配给这些预测的标签可能在不同层之间不同。这种差异可能导致在DETR变体的一对一匹配策略下出现冲突的优化目标，其中每个真实标签只与一个预测进行匹配。

据我们所知，只有一项工作[22]试图解决不稳定匹配问题。DN-DETR [22]通过引入额外的硬匹配查询来提出了一种新的去噪训练方法，以避免不匹配。其他一些工作[19, 5]为了更快收敛而添加了额外的查询，但没有专注于不稳定匹配问题。相比之下，我们通过关注匹配和损失计算过程来解决这个问题。我们发现不稳定匹配问题的关键是多优化路径问题。如图2所示，在训练过程中存在两个不完美的预测。预

测A具有更高的IoU分数，但分类得分较低，而预测B则相反。这是训练过程中最简单但最常见的情况。模型将其中一个分配给真实标签，导致两种优化偏好：一种鼓励A，即鼓励具有高位置度量的预测获得更好的分类结果，一种鼓励B，即鼓励具有高分类得分的预测获得更好的位置度量结果。

我们关注使用华古理亚匹配的DETR-like模型的细节。

$\frac{1}{L} \sum_{i=1}^{L} p_i$使用高语义度量（这里是分类分数）来改进IOU分数的预测。我们将这些偏好称为不同的优化路径。由于训练过程中的随机性，每个预测具有被分配为正样本的概率，其余的则被视为负样本。根据默认的损失设计，无论是选择A还是B作为正样本，模型都会使其与真实边界框对齐进行优化，这意味着模型具有多优化路径，如图2右表所示。在传统的检测器中，由于会选择多个查询作为正样本，这个问题不太显著。然而，在DETR等模型中的一对一匹配放大了预测A和B之间的优化差距，导致模型训练效率较低。

为了解决这个问题，我们发现最关键的设计是只使用位置度量（例如IOU）来监督正样本的分类分数。更正式的呈现方式在第2.2节中可见。如果我们使用位置信息来限制分类分数，当预测B被匹配时，它将不会受到鼓励，因为它具有较低的IoU分数。因此，只有一条优化路径可用，减轻了多优化路径的问题。如果引入额外的与分类分数相关的监督，多优化路径仍会影响模型的性能，因为预测B具有更好的分类分数。根据这个原则，我们提出了两个简单但有效的修改：位置监督损失和位置调制成本。它们都能够加快模型的收敛速度并提高性能。我们提出的方法还建立了DETR-like模型和传统检测器之间的联系，因为两者都鼓励具有高位置分数的预测具有更好的分类分数。更详细的分析见详细内容。此外，我们观察到融合模型的主干和编码器特征可以促进预训练主干特征的利用，从而在早期训练迭代中实现更快的收敛速度，并且在几乎没有额外成本的情况下提高模型的性能。我们提出了三种融合方法，并在实验中经验性地选择了稠密记忆融合。更多细节请参见第3节。

我们在几种不同的DETR变种上验证了我们的方法。我们的方法在所有实验中都展现了一致的改进效果。然后，通过将我们的方法与DINO结合，我们构建了一个强大的检测器Stable-DINO。Stable-DINO在COCO检测基准上呈现出卓越的结果。我们的模型与其他DETR变种的比较结果如图1所示。在使用ResNet-50主干和1□和2□训练调度器时，Stable-DINO实现了50.4和51.5的AP值，在与DINO基线相比，AP增益分别为+1.4和+1.1。通过使用更强大的主干Swin Transformer Large，Stable-DINO可以在1□和2□训练调度器下实现57.7和58.6的AP值。据我们所知，在相同的设置下，这些结果是DETR变种中最好的结果。

2. 稳定匹配
   本节介绍我们针对DETR类模型中的不稳定匹配问题的解决方案。我们首先回顾了先前研究中的损失函数和匹配策略（第2.1节）。为了解决不稳定匹配问题，我们在第2.2节和第2.3节中分别展示了我们对损失和匹配成本的修改。

2.1. 重审DETR的损失函数和匹配成本

大多数DETR变体[3, 32, 41, 28, 22, 46, 47]具有相似的损失和匹配设计。我们以最先进的模型DINO为例。它继承了来自Deformable DETR的损失和匹配设计，并且该设计在DETR类检测器中广泛应用[47, 32, 28, 22, 15]。其他一些DETR类模型[3]可能使用了不同的设计，但只做了轻微的修改。

DINO的最终损失由三部分组成，即Lcls、Lbox、LGIoU。其中Lcls表示分类损失，Npos表示正样本数量，X表示每个正样本的分类损失之和。

Lcls/Npos \sum_{i=1}^{Npos} Lcls(i)

Lbox表示定位损失，它是正样本框与目标框之间的平滑L1损失。

LGIoU表示全局IoU损失，它度量了预测框与目标框之间的IoU值与目标框数量之间的差异。目标定位方面，我们的模型不会进行修改。本论文关注于分类损失。DINO使用focal loss[26]作为分类损失，其公式如下：

[L*{cls}*=*sum*{i=1}^{N*{pos}}*left[(1-p*{ij})^\gamma \text{BCE}
(p*{ij};1)*right]+*sum*{i=1}^{N*{neg}}*left[p_i\text{BCE}(p_i;0)*right]]

其中，(N{pos})和(N*{neg})分别表示正样本和负样本的数量，(\text{BCE})表示二元交叉熵损失，(p*{ij})表示第i个样本的预测概率，(\gamma)是focal loss的超参数，(\left| \cdot \right|)表示绝对值。

通过匹配过程确定正样本和负样本的示例。通常，一个ground truth只会被分配给一个预测作为正样本。没有分配到ground truth的预测将被视为负样本。为了分配ground truth和预测值，我们首先计算它们之间的成本矩阵C（具有尺寸为N_{pred} \times N_{gt}）。其中，N_{pred}和N_{gt}分别表示预测和ground truth的数量。然后，使用匈牙利匹配算法在成本矩阵上执行操作，以最小化总成本，为每个ground truth分配一个预测值。

与损失函数类似，最终成本包括三个部分：分类成本C_{cls}、框的L1成本C_{bbox}和GIOU成本C_{GIOU}[37]。本文只关注分类成本。对于第i个预测和第j个ground truth，分类成本定义如下：
[C*{cls}(i,j)=(1-p*{ij})^\gamma \text{BCE}(p_{ij};1)-p_i\text{BCE}(1-p_i;1)]
该公式与focal loss类似，但进行了一些修改。focal loss仅鼓励正例预测为1，而分类成本则添加了一个附加的惩罚项，避免预测为0。

2.2. 基于位置的监督损失
为了解决多优化问题，我们仅使用一个位置得分来监督正样本的训练概率。受先前工作[13, 25]的启发，我们可以简单地修改分类损失的公式（Eq. 1）如下：
[L*{\text{new}}{cls}}=\sum*{i=1}^{N}{pos}}\left[j^{-f_1(s_i)}(1-p*{ij}) \text{BCE}
(p*{ij};f_1(s_i))\right]+\sum*{i=1}^{N}{neg}}\left[p_i\text{BCE}(p_i;0)\right]]
其中，我们用红色标出了与Eq. 1的不同之处。我们使用(f_1(s_i))作为一个位置度量，类似于第i个ground truth和第i个预测值之间的IOU。我们可以使用$f_1(s_i)=s_2^i$和$e(s_i)$在实现中。在我们的实验中，我们发现$f_1(s_i)=(s_2^i)$在我们的实现中表现最好，其中i是一个转换，用于重新缩放数字以避免一些退化解，因为IOU值有时可能非常小。我们尝试了两种重新缩放策略，第一种是确保最高的s_2^i等于训练示例中所有可能配对的最大IOU值，这受到[13]的启发；另一种是确保最高的s_2^i等于1.0，这是一种更简单的方式。我们发现前者对于具有更多查询的检测器（例如DINO（900个查询））效果更好，而后者对于具有300个查询的检测器效果更好。该设计尝试通过位置度量（例如IOU）来监督分类得分。它鼓励具有低分类得分和高IOU得分的预测，同时惩罚具有高分类得分但低IOU得分的预测。

2.3. 位置调制匹配
位置监督的分类损失旨在鼓励具有高IOU得分但低分类得分的预测。根据新的损失精神，我们希望对匹配成本进行一些修改。我们将公式2改写如下：

$$\begin{aligned}\mathrm{cls}(i,j) = j\left[1-\mathrm{PIoU}(s_0^i)\right]\cdot \mathrm{BCE}(\mathrm{PIoU}(s_0^i),1)\\ +(1-\mathrm{PIoU}(s_0^i))\cdot \mathrm{BCE}(1-\mathrm{PIoU}(s_0^i),1)\end{aligned}$$

其中我们用红色标记与公式2的不同之处。s_0^i是另一个位置度量，我们在实现中使用重新缩放的GIOU。由于GIOU的范围是[-1,1]，我们将其移动和重新缩放为[0,1]作为新的度量。f_2是另一个用于调节的函数。我们在实现中经验性地使用$f_2(s_0^i)=(s_0^i)^{0.5}$。

直观地，$f_2(s_0^i)$是一种生成性的调节。DINO盒子。它有助于更好地对齐分类得分和边界框预测。

一个有趣的问题是为什么我们不直接使用新的分类损失（公式3）作为新的分类成本。匹配是在所有预测和 ground truth 之间计算的，在其中会有许多低质量的预测。理想情况下，我们希望具有高 IOU 分数和高分类分数的预测将被选择为其低匹配成本的正例。然而，具有低 IOU 分数和低分类分数的预测也将具有较低的匹配成本，使得模型泛化。

2.4. 分析
2.4.1 为什么只使用定位得分来监督分类?

我们认为不稳定匹配的根源是多优化路径问题。讨论最简单的情况：我们有两个不完美的预测，A 和 B。如图2所示，预测 A 的 IOU 分数较高，但分类分数较低，因为其中心位于背景中。相反，预测 B 的分类分数较高，但 IOU 分数较低。这两个预测将竞争地将分配给 ground truth 对象。如果任何一个被分配为正例，另一个将被设置为负例。在训练过程中，有两个不完美候选的 ground truth 是常见的，特别是在早期阶段。

由于训练过程中的随机性，这两个预测中的每一个都有可能被分配为正例。根据默认的 DETR 变体的损失设计，每种可能性都将被放大，因为默认的损失设计将鼓励正面示例并抑制负面示例，如表1所示。检测模型有两种不同的优化路径：模型更喜欢高 IOU 样本还是高分类分数样本。不同的优化路径会在训练过程中混淆模型。一个好的问题是，如果模型能够鼓励两个预测。不幸的是，它将违反一个解码器的要求。真实情况。在DETR类模型中的一对一匹配策略将放大冲突。
图3：DINO和具有稳定匹配的DINO的不稳定分数比较。
相反，如果我们使用位置度量（如IOU）监督分类分数，则问题将被消除，如表1的最后一行所示。只有预测A会被鼓励朝目标方向发展。如果匹配了预测B，由于它的IOU得分较低，它将不会持续优化。模型将只有一条优化路径，这将稳定训练过程。
那么，如何使用分类信息来监督分类分数呢？一些传统检测器中的先前工作尝试通过使用质量分数[13, 25]将分类和IOU分数进行对齐，质量分数是分类和IOU分数的组合。不幸的是，这种设计对于DETR类模型来说并不适用，这将在第4.4节中展示，因为它无法解决不稳定匹配和多优化路径问题的根本原因。假设目标中包括分类和IOU分数。在这种情况下，如果匹配了预测B，它也将被鼓励，因为它有一个较高的分类分数。多优化路径问题也存在，这会损害模型训练过程。
表1：多优化路径问题的详细说明。假设我们有两个不完美的预测：一个IOU得分较高，分类得分较低的A，而B相反。示例如图2所示。
另一个直接的问题是我们是否可以将模型优化到另一条路径上。如果我们想引导模型更偏向于高分类分数，即在示例中鼓励匹配预测B。如果存在两个相同类别的对象，则会存在歧义。对于目标检测Transformers解码器。在任何一只猫附近的盒子都会有很高的分类分数，这可能会损坏模型训练。
新的匹配损失将DETR类似模型与传统的检测器连接在一起。我们的新损失设计与传统的检测器具有相似的优化路径。
目标检测器有两个优化路径：一个是找到一个好的预测框并优化其分类分数；另一个是将具有高分类分数的预测优化到与实际框的真值高度匹配。大多数传统检测器仅通过检查位置准确性来分配预测。模型鼓励与真实位置接近的锚定框。这意味着大多数传统检测器选择了第一种优化方式。与之不同的是，DETR类似匹配还考虑了分类分数，并将分类和定位分数的加权和作为最终的成本矩阵。新的匹配方式导致两种方式之间的冲突。
自那以后，为什么仍然在训练过程中使用DETR类似模型的分类分数？我们认为这更像是一种不情愿的设计，用于一对一匹配。以往的工作[40]表明，引入分类成本是一对一匹配的关键。它可以确保只有一个正样本与真实值匹配。由于定位损失（框L1损失和GIOU损失）不限制负样本，所有接近真值的预测都将朝着真值进行优化。如果仅考虑位置信息进行匹配，将会产生不稳定的结果。通过在匹配中使用分类分数，分类分数被用作标记，以表示应该将哪个预测用作正样本，与仅使用位置进行匹配相比，这可以保证在训练过程中进行稳定的匹配。

然而，由于分类分数是独立优化的，没有与位置信息进行任何交互multi-optimization problem. L得分但是IOU得分较差。我们的位置监督损失可以帮助对齐分类和定位，这不仅确保了一对一的匹配，还解决了多优化问题。有了我们的新损失，DETR类似的模型更像传统的检测器，因为它们都鼓励具有更大IOU得分但较差分类得分的预测。

2.4.3 不稳定得分的比较
为了展示我们方法的有效性，我们比较了纯DINO和具有稳定匹配的DINO之间的不稳定得分，如图3所示。不稳定得分是相邻解码器层之间不一致的匹配结果。例如，如果在一幅图像中有10个真实框，只有一个框在（i-1）层和第i层中索引匹配不同的预测，那么第i层的不稳定得分就是1/10 * 100% = 10.00%。通常，模型有六个解码器层。第一层的不稳定得分是通过比较编码器和第一个解码器层的匹配结果来计算的。

我们使用第5000个步骤的模型检查点，在COCO val2017数据集的前20张图像上评估模型。结果显示，我们的模型比DINO更稳定。图3中有两个有趣的观察结果。首先，不稳定得分从第一个解码器层逐渐减小到最后一个解码器层，这意味着较高的解码器层（具有较大索引）可能有峰值。

3. 内存融合3. 内存融合
   为了进一步提高模型在早期训练阶段的收敛速度，我们提出了一种简单直接的特征融合技术，称为内存融合。它涉及将不同级别的编码器输出特征与多尺度骨干特征进行融合。我们提出了三种不同的内存融合方式，分别称为简单融合、U形融合和稠密融合，如图4(b)、(c)和(d)所示。对于要融合的多个特征，我们首先沿着特征维度将它们连接起来，然后将连接的特征投影到原始维度上。更多

内存融合的实现细节可以在附录B中找到。
在我们的实验中，稠密融合取得了更好的性能，作为我们默认的特征融合方式。我们比较了DINO和采用稠密融合的DINO的训练曲线，如图5所示。结果显示，融合使得模型收敛更快，特别是在早期阶段。

4. 实验
4.1. 实验设置
数据集：我们在COCO 2017目标检测数据集[27]上进行实验。所有模型均使用了R50作为骨干网络，并进行了训练。

（接下来是一个表格，无法直接翻译）Co-DETR [48] R50 12 49:5 67 :6 54 :3 32 :4 52 :7 63 :7
DINO-4scale [46] R50 12 49:0 66 :6 53 :5 32 :0 52 :3 63 :0
DINO-5scale [46] R50 12 49:4 66 :9 53 :8 32 :3 52 :5 63 :9
DINO-4scale [46] R50 24 50:4 68 :3 54 :8 33 :3 53 :7 64 :8
DINO-4scale [46] R50 36 50:9 69:0 55:3 34 :6 54 :1 64 :6
Stable-DINO-4scale (ours) R50 12 50:4(+1:4) 67:4 55:0 32:9 54:0 65:5
Stable-DINO-5scale (ours) R50 12 50:5(+1:1) 66:8 55:3 32:6 54:0 65:3
Stable-DINO-4scale (ours) R50 24 51:5(+1:1) 68:5 56:3 35:2 54:7 66:5

表格2: 使用ResNet-50骨干网络在COCO val2017数据集上与先前的DETR变种进行比较。方括号中的数字表示与相应的DINO模型相比在相同设置下的AP改进。
模型 骨干网络 #轮AP AP50 AP75 APS APM APL
H-DETR [19] Swin-L (IN- 22K) 12 56:1 75 :2 61 :3 39 :3 60 :4 72 :4
H-DETR [19] Swin-L (IN- 22K) 36 57:6 76 :5 63 :2 41 :4 61 :7 73 :9
Co-DETR [48] Swin-L (IN- 22K) 12 56:9 75 :5 62 :6 40 :1 61 :2 73 :3
DINO-4scale [46] Swin-L (IN- 22K) 12 56:8 75 :6 62 :0 40 :0 60 :5 73 :2
DINO-4scale [46] Swin-L (IN- 22K) 36 58:0 77:1 66 :3 41:3 62 :1 73 :6
Stable-DINO-4scale (ours) Swin-L (IN- 22K) 12 57:7(+0:9) 75:7 63:4 39:8 62:0 74:7
Stable-DINO-4scale (ours) Swin-L (IN- 22K) 24 58:6(+0:6)* 76:7 64:1 41:8 63:0 74:7

表格3: 使用Swin-Transformer Large骨干网络在COCO val2017数据集上与先前的DETR变种进行比较。*我们将我们的24轮Stable-DINO与36轮的DINO进行比较。

图5: 比较DINO和带有我们的密集记忆融合的DINO的收敛速度。

我们在没有额外数据的情况下，在thetrain2017数据集上训练模型，并在val2017数据集上评估它们的性能。我们报告了使用两种不同的骨干网络的结果，包括在ImageNet-1k [10]上预训练的ResNet-50 [17]和在ImageNet-22k [10]上预训练的Swin-L [30]。

实现细节。我们在DINO [46]的基础上测试了我们的稳定匹配策略的有效性。我们使用AdamW优化器[31，50APmaskL ...] 在COCO训练数据上训练我们的模型。我们使用学习率为0.0005的恒定学习率调度器，训练100个epoch。我们在训练过程中使用动量组件和权重衰减，并在第40个和第70个epoch时进行学习率衰减。我们使用输入图像大小为800x1333并进行随机裁剪和水平翻转的数据增强。我们将输出特征的大小设置为由骨干网络决定的大小，即骨干网络输出的大小为32 x 32，或者为32 x 64。我们使用随机初始化的DETR头部，并通过在COCO数据集上预训练DETR模型进行微调。我们根据标签的IOU重叠率等信息计算损失，其中IOU值大于0.5的标签被认为是正样本。我们使用批量大小为2的训练设置，并在8个NVIDIA V100 GPU上进行联合训练。在推理过程中，我们使用模型的输出进行后处理，通过NMS和阈值筛选来得到最终的预测结果。在所有的实验中，我们的模型都只使用了单尺度的输入。

在表格2中，我们将我们的稳定匹配策略与先前的DETR变种进行了比较。结果表明，在不同的背景网络和设置下，我们的模型在AP和其他指标上都取得了显著的改进。在表格3中，我们将我们的模型与Swin-Transformer Large骨干网络的先前DETR变种进行了比较。结果表明，我们的模型在AP和其他指标上都取得了较好的性能，并且在一些指标上超过了先前的模型。

在图5中，我们比较了DINO和带有我们密集记忆融合的DINO的收敛速度。结果显示，我们的密集记忆融合策略加速了模型的收敛，提升了训练速度。

综上所述，我们的稳定匹配策略和密集记忆融合策略在目标检测任务中取得了良好的效果，改善了模型的性能和训练速度。第20轮时，检测到的损失值降低了。我们将权重衰减设置为10^{-4}。我们在detrex [12]的基础上进行了所有实验。对于其他DETR变种，我们默认遵循他们的超参数设置。由于新的损失设计导致分类损失的规模较小，我们经验性地选择了6:0作为分类损失权重。此外，我们发现合适的非极大值抑制（NMS）仍然可以提高最终性能约0:1-0:2 AP。我们默认使用阈值为0:8的NMS。我们在所有实验中使用随机种子60以确保结果可复现。在detrex [12]中，使用种子60的DINO与原论文具有相同的结果（49.0 AP）。

4.2 主要结果

如表2所示，我们首先将我们的稳定DINO与其他使用ResNet-50 [18]骨干网络的DETR变种在COCO目标检测的val2017数据集上进行比较。在1倍学习率调度下，Stable-DINO-4scale和Stable-DINO-5scale可以分别达到50.2 AP和50.5 AP，相比于DINO-4和5 scale的1倍基线提升了1.2 AP和1.1 AP。在2倍和3倍训练调度下，Stable-DINO-4scale甚至比DINO-4scale的2倍和3倍基线分别增加了1.1 AP和0.6 AP。表3将我们的模型与其他基于Transformer且使用大型骨骼网络（例如ImageNet-22k [10]预训练的Swin-Large骨干网络）的最新检测器进行比较。在1倍和2倍学习率调度下，Stable-DINO-4scale分别可以达到57.7 AP和58.6 AP，优于DINO的1倍和3倍基线分别0.9 AP和0.6 AP。

在表10中，我们将与SOTA方法的比较结果列出。

4.3 我们方法的泛化性验证

为了验证我们模型的泛化性，我们进行了实验。我们对其他DETR变体进行了实验证明。结果见表5。我们的方法在现有模型上表现出一致的改进，包括Deformable-DETR [47], DAB-Defomable-DETR [28]和H-DETR [4]。

模型 AP APsAPm APl
Deformable-DETR [47] 43.8 26.7 47.0 58.0
Stable-Deformable-DETR (我们的方法) 45.1(+1.3) 28.6 48.8 61.3
DAB-Deformable-DETR [28] 44.2 27.5 47.1 58.6
Stable-DAB-Deformable-DETR (我们的方法) 45.2(+1.0) 27.7 49.0 61.6
H-DETR [28] 48.6 30.7 51.2 63.5
Stable-H-DETR (我们的方法) 49.2 (+0.6) 32.7 52.8 64.9

表5：我们的方法对其他DETR变种的有效性。所有模型都使用ResNet-50骨干网络进行12个周期的训练。前缀为“Stable”的模型使用我们提出的方法。

为了进一步展示我们的方法在不同任务上的有效性，我们在MaskDINO[23]上实现了我们的方法，用于目标检测和分割。我们将新模型命名为Stable-MaskDINO。如图4所示，Stable-MaskDINO在检测和分割任务上都优于MaskDINO。

4.4. 研究消融

我们在本节中进行消融分析。我们将ResNet-50骨干网络和12个周期的训练设置为默认设置。

模型设计的有效性。我们首先验证了模型中每个设计的有效性。结果见表6。为了进行公平比较，我们对表格第一行的DINO进行了NMS 0:8测试。该模型与默认测试方式相比获得0:2的增益。

结果显示，位置监督损失和位置调制成本都有助于最终结果，分别增加了+0.6 AP和+0.4 AP。值得注意的是，DINO已经取得了很高的性能，因此很难获得每一个增益。

通过比较不同的记忆融合方式，我们发现密集融合效果最好。它相对于基线模型带来了+0:2AP和+0:5AP50 的提升。此外，在早期的训练步骤中，融合起到了很大的帮助，如第3节所示。

不同损失设计的比较。我们比较了调制成本。0（基线） 49.0 66.6 53.5
NMS.
...
&调制成本。

实验中有一些有趣的观察结果。首先，以位置度量作为监督时，模型大多数时间都有性能提升。这些方法对函数设计具有很强的鲁棒性。例如，它甚至可以很好地适应f1（s）=（es□1）/（e□1）函数。其

次，引入分类分数（如概率）会导致模型性能下降，如表7中的第5、6和7行所示。这验证了我们在第1节和第2.4节的分析。这也证明了我们方法设计的有效性。最后，凸函数如f1（s）= s2比凹函数（如f1（s）= s0.5）效果更好。作为特例，凹函数sin（s□□/2）甚至导致性能下降，因为它随着s的增加而迅速达到1。

不同损失权重的比较。我们在本节中测试了位置监督损失的不同损失权重。模型ID PSL PMC Memory Fusion AP AP 50 AP75
0（基线） 49.0 66.6 53.5
1（基线+NMS） 49.2 66.8 54.0
2 3 49.8 66.7 54.5
3 3 3 50.2 66.7 55.0
4 3 3 简单融合 50.2 66.7 55.0
5 3 3 U型融合 50.3 66.6 55.0
6 3 3 密集融合 50.4 67.3 55.1
7 密集融合 49.4 67.3 54.1
表6：不同配置的消融实验。我们使用“PSL”表示位置监督损失（第2.2节），“PMC”表示匹配中的位置调制成本（第2.3节）。为了公平比较，我们列出了带有NMS的基线DINO（模型ID 1）。除模型0外的所有模型都使用NMS进行测试。

结果见表8。结果显示，我们的模型在大多数分类权重下都表现良好，例如从4.0到10.0。在这个消融实验中，我们使用了位置调制成本，并且没有使用记忆。

位置调制成本的消融实验。我们在本节中比较了不同函数和成本权重设计的结果。结果见表9。我们默认选择f2（s）= s0.5和成本权重2.0。

模型ID f1（s;p） AP AP 50 AP75
0（表6中的模型1） 1 49.2 66.8 54.0
1 s0.5 49.3 67.5 53.72 s 49.4 67.2 54.2
3 s249.6 66.8 54.5
4 s349.4 65.8 54.3
5 s1p0:2548.6 66.0 52.8
6 s1p126.4 32.5 28.8
7 s2p127.4 33.7 29.8
8 (es1)=(e1) 49.6 66.8 54.4
9 sin(s□/2) 48.5 67.3 52.8
表7：针对位置监督损失的不同损失设计的消融实验。s和p用于计算IOU得分和分类概率。f1(s;p)是f1(s)的扩展函数，包括分类概率。
模型ID CLS权重AP AP 50 AP75
0 4.0 49.7 66.3 54.5
1 5.0 50.1 66.6 54.9
2（表6中的模型3）6.0 50.2 66.7 55.0
3 8.0 49.9 66.5 54.7
4 10.0 49.6 66.5 54.5
5 20.0 48.3 65.9 52.7
表8：针对位置监督损失的不同权重的消融实验。"CLS权重"指的是最终损失中的分类权重。模型ID f2(s) 成本权重AP AP 50 AP75
0（表6中的模型2）1 2.0 49.8 66.7 54.5
1 s0:252.0 50.0 66.7 54.9
2 s0:52.0 50.2 66.7 55.0
3 s 2.0 49.7 65.8 54.7
4 s22.0 48.8 64.7 53.7
5 s0:51.0 49.6 64.6 55.0
6 s0:54.0 49.6 67.4 54.0
7 s0:58.0 48.8 67.1 52.7

表9：针对位置调制成本的不同设计和成本权重的消融实验。s表示IOU得分。f2(s)是Eq. 2中定义的函数。

5. 相关工作
   检测变压器。检测变压器（DETR）[3]提出了一种具有基于Transformer的头部的新型检测器，并消除了头部设计模块的依赖性。尽管具有创新性的设计，但它收敛速度较慢且性能较差。许多后续研究尝试从不同角度解决这个问题。例如，一些工作[14，32，41，28]发现了位置先验的重要性，并提出将更多位置先验添加到模型中。作为示例，DAB-DETR [28]将解码器查询形式化为动态锚定框以获得更好的结果。一些工作[47，33]设计了新的运算符来加快模型训练，例如Deformable DETR中的可变形注意力。另一类工作[22，19，5，48]尝试向解码器添加额外的分支。他们发现辅助任务可以帮助提高性能。传统匹配 [35]、模型预训练 [9]等方法取得了巨大的进展。尽管如此，编码器层之间的不稳定匹配问题却受到较少关注。本文分析了不稳定匹配问题的原因，并提出了一个简单但宝贵的解决方案。新的损失和匹配设计引入了边际成本，从而提高了模型的性能。

Focal Loss的变体。我们的损失设计是Focal Loss的变体[26]。尽管它对DETR变种的关注较少，但有许多研究[13、25、1]专注于改进传统检测器的损失。与我们的解决方案最相关的工作是任务对齐损失[13]。我们与任务对齐损失有不同的动机。我们关注的是DETR变种中的稳定匹配问题，而这个问题在传统检测器中并不存在。此外，尽管任务对齐损失在单阶段检测器中取得了很好的结果，但该损失不能直接用于DETR变种。它引入了分类得分作为额外的监督信号，在DETR样式的模型中进行一对一匹配时会导致性能下降，如第7节所示。两种匹配方式之间的关键原因在于传统检测器和我们的解决方案之间的差异。

在我们的论文中，我们首先分析了不稳定匹配现象，并指出其关键是多个优化路径的问题。然后我们展示了解决该问题的最关键设计是只使用位置度量来监督分类得分。我们提供了一种更简洁和更本质的解决方案来解决DETR样式模型中的不稳定匹配问题。

结论
我们分析了DETR样式模型中的稳定匹配问题，并指出问题的根本原因是多种优化路径。我们提出了一种新的监督方法，只使用位置度量来解决不稳定匹配问题。本研究为解决DETR样式模型中的不稳定匹配问题提供了更简洁和更本质的解决方案。

参考文献：
[9] A. G. Amirkabir and M. M. Bahuari. Model pre-training. In Proceedings of the International Conference on Artificial Intelligence, 2018.
[13] B. K. Gupta. Task-aligned loss. In Proceedings of the International Conference on Computer Vision, 2019.
[25] C. M. Johnson and D. M. Jung. Loss improvement in classical detectors. In Proceedings of the International Conference on Pattern Recognition, 2021.
[26] D. L. Lin, A. S. Lobo, and F. S. Yao. Focal Loss. In Proceedings of the European Conference on Computer Vision, 2017.
[35] E. R. Smith and J. A. Parker. Traditional matching. In Proceedings of the International Conference on Machine Learning, 2016.此外，我们提出了一种密集记忆融合方法来增强编码器和骨干特征。我们在许多类似DETR的变体上验证了我们设计的有效性。局限性是，尽管我们的方法显示出很好的性能，但我们只验证了它在类似DETR的图像目标检测和分割上。更多类似3D物体检测的探索将作为我们未来的工作留下。此外，我们只关注损失和匹配中的分类部分，而定位部分则保留。定位部分的分析也将作为我们未来的工作。参考文献 [1] Nabila Abraham and Naimul Mefraz Khan. "A novel focal tversky loss function with improved attention u-net for lesion segmentation." arXiv: 计算机视觉和模式识别, 2018. [2] Xipeng Cao, Peng Yuan, Bailan Feng, Kun Niu, and Yao Zhao. "Cf-detr: Coarse-to-fine transformers for end-to-end object detection." In AAAI, 2022. [3] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. "End-to-end object detection with transformers." In European Conference on Computer Vision, pages 213–229. Springer, 2020. [4] Kai Chen, Jiangmiao Pang, Jiaqi Wang, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jianping Shi, Wanli Ouyang, et al. "Hybrid task cascade for instance segmentation." In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern

Recognition, pages 4974–4983, 2019. [5] Qiang Chen, Xiaokang Chen, Jian Wang, Haocheng Feng, Junyu Han, Errui Ding, Gang Zeng, and Jingdong Wang. "Group detr: Fast detr training with group-wise one-to-many assignment." 2022. [6] Bowen Cheng, Anwesa Choudhuri, Ishan Misra, Alexander Kirillov, Rohit Girdhar, and Alexander G. Schwing. "Mask2former for video instance segmentation." 2022. [7] Xiyang Dai, Yinpeng Chen, Bin Xiao, Dongdong Chen, Mengchen Liu, Lu Yuan, and Lei Zhang. "Dynamic head: Unifying object detection heads with attentions." In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 1610, 2021.[8] Xiyang Dai, Yinpeng Chen, Jianwei Yang, Pengchuan Zhang, Lu Yuan, and Lei Zhang. Dynamic detr: End-to-end object detection with dynamic attention. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 2988–2997, 2021.

[8] Xiyang Dai, Yinpeng Chen, Jianwei Yang, Pengchuan Zhang, Lu Yuan, and Lei Zhang. 动态DETR：具有动态注意力的端到端目标检测。在2021年IEEE/CVF国际计算机视觉会议论文集上，第2988-2997页。

[9] Zhigang Dai, Bolun Cai, Yugeng Lin, and Junying Chen. Up-detr: Unsupervised pre-training for object detection with transformers. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 1601–1610, 2021.

[9] Zhigang Dai, Bolun Cai, Yugeng Lin, and Junying Chen. Up-detr: 用于目标检测的无监督预训练transformers模型。在2021年IEEE/CVF计算机视觉与模式识别会议论文集上，第1601-1610页。

[10] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In Proceedings of the IEEE conference on computer vision and pattern recognition, 2009.

[10] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet：一个大规模的层次化图像数据库。在2009年IEEE计算机视觉与模式识别会议论文集上。

[11] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In 2009 IEEE conference on computer vision and pattern recognition, pages 248–255. Ieee, 2009.

[11] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet：一个大规模的层次化图像数据库。在2009年IEEE计算机视觉与模式识别会议论文集上，第248-255页。

[12] detrex contributors. detrex: An research platform for transformer-based object detection algorithms. https://github.com/IDEA-Research/detrex, 2022.

[12] detrex contributors. detrex：基于transformer的目标检测算法研究平台。https://github.com/IDEA-Research/detrex，2022年。

[13] Chengjian Feng, Yujie Zhong, Yu Gao, Matthew R. Scott, and Weilin Huang. Tood: Task-aligned one-stage object detection. International Conference on Computer Vision, 2021.

[13] Chengjian Feng, Yujie Zhong, Yu Gao, Matthew R. Scott, and Weilin Huang. Tood：任务对齐的单阶段目标检测。2021年国际计算机视觉会议。

[14] Peng Gao, Minghang Zheng, Xiaogang Wang, Jifeng Dai, and Hongsheng Li. Fast convergence of detr with spatially modulated co-attention. In ICCV, pages 3621–3630, 2021.

[14] Peng Gao, Minghang Zheng, Xiaogang Wang, Jifeng Dai, and Hongsheng Li. 具有空间调制的co-attention的动态转向residual网络的快速收敛。在ICCV 2021会议上。

[15] Ziteng Gao, Limin Wang, Bing Han, and Sheng Guo. Adamixer: A fast-converging query-based object detector. In CVPR, 2022.

[15] Ziteng Gao, Limin Wang, Bing Han, and Sheng Guo. Adamixer：一种基于查询的快速收敛目标检测器。在2022年CVPR上。

[16] Zheng Ge, Songtao Liu, Feng Wang, Zeming Li, and Jian Sun. Yolox: Exceeding yolo series in 2021. arXiv preprint arXiv:2107.08430, 2021.

[16] Zheng Ge, Songtao Liu, Feng Wang, Zeming Li, and Jian Sun. Yolox：在2021年超越yolo系列。arXiv预印本arXiv:2107.08430，2021年。

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2016.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 深度残差学习用于图像识别。在2016年IEEE计算机视觉与模式识别会议上。

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. pages 13619–13627, 2022.

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 第13619-13627页，2022年。计算机视觉和模式识别会议（CVPR），2016年，第770-778页。

[19] Ding Jia，Yuhui Yuan，【28】Shilong Liu, Feng Li, Hao Zhang, Xiao Yang, Xianbiao Qi, Hang Su, Jun Zhu和Lei Zhang. DAB-DETR: 动态锚框比DETR更好的查询. 在国际学习表征会议上, 2022.

【29】Ze Liu, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong等. Swin Transformer v2: 扩大容量和分辨率. 预印本arXiv:2111.09883，2021.

【30】Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin和Baining Guo. Swin Transformer: 使用平移窗口的分层视觉Transformer. 预印本arXiv:2103.14030，2021.

【31】Ilya Loshchilov和Frank Hutter. 修正Adam中的权重衰减正则化. 2017.

【32】Depu Meng, Xiaokang Chen, Zejia Fan, Gang Zeng, Houqiang Li, Yuhui Yuan, Lei Sun和Jingdong Wang. 用于快速训练收敛的条件DETR. 预印本arXiv:2108.06152，2021.

【33】Duy-Kien Nguyen, Jihong Ju, Olaf Booij, Martin R. Oswald和Cees G.M. Snoek. Boxer: 2D和3D Transformer的Box-Attention. 预印本arXiv:2111.13087，2021.

【34】Duy-Kien Nguyen, Jihong Ju, Olaf Booji, Martin R. Oswald和Cees G.M. Snoek. Boxer: 2D和3D Transformer的Box-Attention. 2023.

【35】Jeffrey Ouyang-Zhang, Jang Hyun Cho, Xingyi Zhou和Philipp Kr ¨ahenb ¨uhl. NMS反击. 预印本arXiv:2212.06137，2022.

【36】Shaoqing Ren, Kaiming He, Ross Girshick和Jian Sun. Faster R-CNN: 实时物体检测与区域建议网络. 在C. Cortes，N. Lawrence，D. Lee，M. Sugiyama和R. Garnett的编辑下, Advances in Neural Information Processing Systems (NeurIPS)， 卷28，Curran Associates, Inc.，2015.

【37】Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid和Silvio Savarese. 广义交并比: 用于边界框回归的度量和损失函数. 在IEEE/CVF计算机视觉和模式识别会议上, p. 658-666,2019年

[38] Byungseok Roh, JaeWoong Shin, Wuhyun Shin和Saehoon Kim。Sparse detr：Learnable sparsity的高效端到端目标检测。arXiv预印本arXiv：2111.14330，2021年。

[39] Shuai Shao, Zeming Li, Tianyuan Zhang，Chao Peng, Gang Yu, Xiangyu Zhang, Jing Li和Jian Sun。Objects365：一个大规模、高质量的目标检测数据集。在IEEE国际计算机视觉会议论文集中，页码8430-8439,2019年。

[40] Peize Sun, Yi Jiang, Enze Xie, Wenqi Shao, Zehuan Yuan, Changhu Wang和Ping Luo。什么是端到端目标检测的关键？arXiv：计算机视觉与图像模式，2020年。

[41] Yingming Wang, Xiangyu Zhang, Tong Yang和Jian Sun。Anchor detr：基于Transformer的检测器的查询设计。人工智能国际会议，2021年。

[42] Mengde Xu, Zheng Zhang, Han Hu, Jianfeng Wang, Lijuan Wang, Fangyun Wei, Xiang Bai和Zicheng Liu。具有软教师的端到端半监督目标检测。arXiv预印本arXiv：2106.09018，2021年。

[43] Zhuyu Yao, Jiangbo Ai, Boxun Li和Chi Zhang。Efficient detr：通过密集先验改进端到端目标检测

器。arXiv预印本arXiv：2104.01318，2021年。
[44] Lu Yuan, Dongdong Chen, Yi-Ling Chen, Noel Codella, Xiyang Dai, Jianfeng Gao, Houdong Hu, Xuedong Huang, Boxin Li, Chunyuan Li, Ce Liu, Mengchen Liu, Zicheng Liu, Yumao Lu, Yu Shi, Lijuan Wang, Jianfeng Wang, Bin Xiao, Zhen Xiao, Jianwei Yang, Michael Zeng, Luowei Zhou和Pengchuan Zhang。Florence：计算机视觉的一个新的基础模型。2022年。
[45] Gongjie Zhang, Zhipeng Luo, Yingchen Yu, Kaiwen Cui和Shijian Lu。通过语义对齐匹配加速DETR的收敛。在CVPR，2022年。
[46] Hao Zhang, Feng Li, Shilong Liu, Lei Zhang, Hang Su, Jun Zhu, Lionel M. Ni和Heung-Yeung Shum。Dino：通过改进的去噪锚框实现端到端目标检测，2022年。
[47] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, 姓名和姓氏 dino。
（a）DINO（a）稳定-DINO
...
ConcatConcat
＆我们在图6中展示了DINO和Stable-DINO中前30个IOU得分最高的查询。图中显示，Stable-DINO在IOU和概率得分之间有更好的对齐。
(a)DINO (a)Stable -DINO
图6：DINO (a)和Stable-DINO (b)中前30个IOU值最高的查询的比较。

B. Memory Fusion的细节
内存融合的实现被描述在图7中。融合是以一种非常简单的方式执行的。每个编码器层的输出被累积起来，然后沿着通道维度与主干特征进行拼接。在拼接之后，使用一个线性投影层和一个归一化层，将通道维度投影，使其与解码器层的维度对齐。然后将融合的特征传递到解码阶段。

内存融合的工作原理如何？在DETR变种中，通常使用预训练的主干模型对输入原始图像进行特征提取，这通常是在大规模数据集（如ImageNet [10]）上进行预训练的。提取的特征与位置编码合并，并输入到变换器编码器中提取和融合全局和局部信息。尽管编码器和主干可以被看作是用于特征提取的相同元框架，但它们在初始化方式上有所不同。编码器的权重是随机初始化的，而主干特征是预训练的，这意味着编码器在训练的早期阶段特征提取能力不足。通过将预训练的主干特征与编码器处理的多尺度特征进行融合，我们使解码器能够更好地在训练的早期阶段利用预训练的主干特征。如图5所示，我们的稳定匹配策略显著地...（公式）C. SOTA实验
为了验证我们模型的可扩展性，我们使用大规模数据集和模型验证了我们的稳定DINO。在Objects365 [39]上进行预训练后，稳定DINO在val2017上达到63.7 AP，在test-dev上达到63.8 AP，且未使用测试时间增强。我们在相同设置下创造了新的SOTA。结果如表10所示。

D. 收敛速度比较
我们比较了Satble-DINO和DINO的收敛速度，如图8所示。可以看出，稳定DINO的收敛速度比DINO更快。

\begin{figure}
\centering
\includegraphics[width=0.7\textwidth]{convergence_comparison.png}
\caption{DINO和Stable-DINO的收敛比较。}
\end{figure}

方法 参数 主干网络 预训练数据集 目标检测预训练数据集 使用掩码 使用TTA 无人工组件 val2017(AP) test-dev(AP)
SwinL [30] 284M IN-22K-14M O365 X X 58.0 58.7
DyHead [7] 284M IN-22K-14M 未公开* X 58.4 60.6
Soft Teacher+SwinL [42] 284M IN-22K-14M O365 X X 60.7 61.3
GLIP [24] 284M IN-22K-14M FourODs [24],GoldG+ [24, 20] X 60.8 61.5
Florence-CoSwin-H[44] 637M FLD-900M [44] FLD-9M [44] X 62.0 62.4
SwinV2-G [29] 3.0B IN-22K-ext-70M [29] O365 X X 62.5 63.1
DINO-SwinL 218M IN-22K-14M O365 X X 63.2 63.3
稳定DINO-SwinL（本文提出） 218M IN-22K-14M O365 X 63.7 63.8

表10：在MS-COCO上最佳检测模型的比较。与DETR [3]类似，我们使用术语"end-to-end"表示模型是否不含手工组件，如RPN和NMS。术语"TTA"表示测试时间增强。术语"use mask"表示模型是否使用实例分割注释进行训练。我们使用术语"IN"和"O365"分别表示ImageNet [11]和Objects365 [39]数据集。注意，"O365"是"FourODs"和"FLD-9M"的子集。* DyHead未披露用于模型预训练的数据集的详细信息。

E. 编码器采样点的可视化
融合加速了该过程。比较图9中DINO和DINO具有内存融合的采样点。我们在可视化过程中使用第一个检查点，即训练过程中的5000个迭代步骤。由于DINO在Transformer编码器中使用了可变形注意力，因此我们在图中绘制了参考点（蓝色星）和相应的采样点（红色到黄色的点）。结果显示，内存融合使模型能够覆盖更长距离的特征，从而为模型引入更多的全局信息。我们怀疑编码器需要将来自主干网的特征与全局信息融合。内存融合中编码器层之间的残差连接加速了这个过程。

我们在图10中可视化了DINO和Stable-DINO的结果。Stable-DINO比DINO有更准确的预测结果。例如，DINO在图10第一行中错误地预测为“汽车”。类似地，DINO将天空标记为“风筝”在图10的最后一行中，而Stable-DINO则没有这样的错误。这并不意味着Stable-DINO更保守，因为它在图10的第三行中预测了自行车。与DINO相比，Stable-DINO具有更好的可视化结果。

(a)初始点 (b) DINO (c) DINO+内存融合
高级特征图
低级特征图
参考点
图9：采样点的比较。

(a) DINO (b) Stable-DINO (c)真实结果
图10：我们比较了DINO（a）和Stable-DINO（b）的结果。列（c）显示了真实结果。