# 代理注意力：Softmax与线性注意力的融合

Dongchen Han\* Tianzhu Ye\* Yizeng Han Zhuofan Xia Shiji Song Gao Huang

自动化系，清华大学，BNRist

摘要

注意力模块是Transformer中的关键组成部分。虽然全局注意力机制提供了高度的表示性，但其过高的计算成本限制了其在各种场景下的适用性。在本文中，我们提出了一种新颖的注意力范式，代理注意力，以在计算效率和表示能力之间取得有利的平衡。具体来说，代理注意力，表示为一个四元组 ，在传统的注意力模块中引入了一组额外的代理标记 。代理标记首先作为查询标记 的代理，从 和 聚合信息，然后将信息广播回 。由于代理标记的数量可以设计得远小于查询标记的数量，代理注意力比广泛采用的Soft-max注意力显著更高效，同时保留了全局上下文建模能力。有趣的是，我们证明了所提出的代理注意力等价于线性注意力的广义形式。因此，代理注意力无缝集成了强大的Softmax注意力和高效率的线性注意力。广泛的实验证明了代理注意力在各种视觉Transformer和多样化的视觉任务中的有效性，包括图像分类、目标检测、语义分割和图像生成。值得注意的是，由于代理注意力具有线性注意力的特性，在高分辨率场景中表现出了显著的性能。例如，当应用于Stable Diffusion时，我们的代理注意力加速了生成过程，并且在不需要额外训练的情况下显著提高了图像生成质量。代码可在 https://github.com/LeapLabTHU/Agent-Attention 获取。

# 1. 引言

源于自然语言处理，Transformer模型近年来在计算机视觉领域迅速崭露头角，并在图像分类 、目标检测 、语义分割 以及多模态任务 [29] 上取得了显著成功。

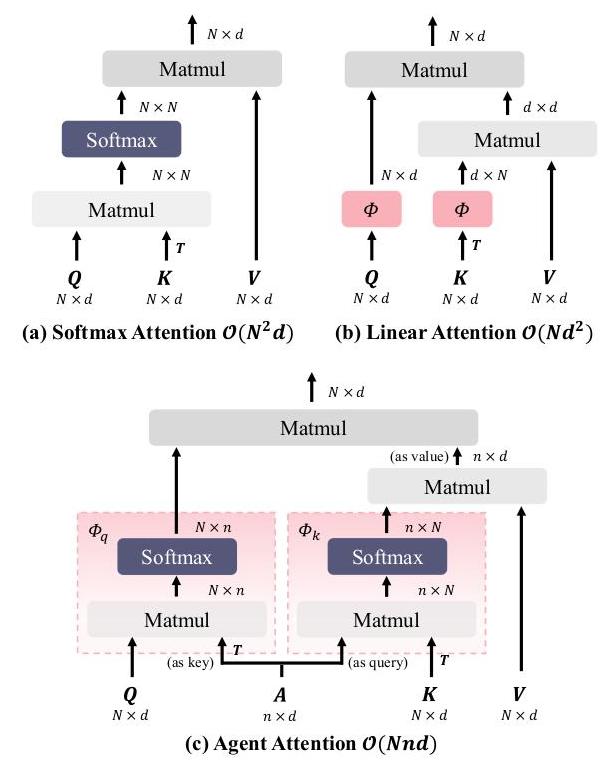


图1. Softmax注意力、线性注意力与代理注意力的区别。Softmax注意力计算所有查询-键对的相似度，导致二次复杂度。线性注意力分别对 和 应用映射函数 ，改变计算顺序，降低复杂度但牺牲了表达性。我们的代理注意力采用一组小型代理标记来聚合和广播全局信息，实现了Softmax和线性注意力的优雅融合，并自然地兼具高表达性和低计算复杂度的优点。

然而，将Transformers和自注意力机制融入视觉领域面临着巨大的挑战。现代Transformer模型通常采用Softmax注意力 [38]，它计算每一对查询-键的相似度，导致计算复杂度随标记数量的增加而呈二次增长。因此，直接在视觉任务中使用具有全局感受野的Softmax注意力会导致难以管理的计算需求。为了解决这个问题，现有工作 通过设计高效的注意力模式来尝试降低计算复杂度。作为两个代表，Swin Transformer [25] 减小了感受野，将自注意力计算限制在局部窗口内。PVT [39] 采用稀疏注意力模式，通过减少键和值的数量来减轻计算负担。尽管这些方法有效，但它们不可避免地牺牲了建模长距离关系的能力，并且在全局自注意力方面仍然处于劣势。

\*同等贡献。

对应作者。

在本文中，我们创新性地在注意力三元组 中引入了一组额外的标记 ，形成了一个四元组注意力范式 ，我们称之为代理注意力（Agent Attention）。如图1(c)所示，所得到的代理注意力模块由两个传统的Softmax注意力操作组成。第一个Softmax注意力应用于三元组 ，其中代理标记 作为查询，用于从值标记 中聚合信息，计算 和 之间的注意力矩阵。第二个Softmax注意力在三元组 上执行，其中 是前一步的结果，形成了所提出代理注意力的最终输出。直观上，新引入的标记 可以被视为查询标记 的“代理”，因为它们直接从 和 收集信息，然后将结果传递给 。查询标记 不再需要直接与原始的键 和值 通信。因此，我们将这些标记 称为代理标记。

由于全局自注意力固有的冗余性，可以设计代理标记的数量远小于查询标记的数量。例如，我们发现简单地汇总原始查询标记以形成代理标记的效果出奇地好。这一特性赋予了代理注意力高效率，将Soft-max注意力的二次复杂度（标记数量）降低到线性复杂度。同时，保留了全局上下文建模能力。有趣的是，如图1所示，所提出的代理注意力可以被视为线性注意力的广义形式，这从新的角度解释了代理注意力是如何解决效率与表达性之间的困境的。换句话说，代理注意力无缝集成了Softmax和线性注意力，并从两个世界中都获得了好处。

我们通过广泛的视觉任务，包括图像分类、目标检测、语义分割和图像生成，实证验证了我们的模型的有效性。我们的方法在各种任务中都有显著改进，特别是在高分辨率场景下。值得注意的是，我们的代理注意力可以直接插入到预训练的大型扩散模型中，而且无需任何额外训练，它不仅加速了生成过程，还显著提高了生成质量。

# 2. 相关工作

# 2.1. 视觉变换器

自从 Vision Transformer [13] 的创立以来，自注意力机制在计算机视觉领域取得了显著的进展。然而，普遍存在的 Softmax 注意力 [38] 的二次复杂度在将自注意力应用于视觉任务时构成了挑战。之前的工作提出了各种方法来解决这一计算挑战。PVT [39] 引入了稀疏全局注意力，通过降低 和 的分辨率来控制计算成本。Swin Transformer [25] 将自注意力计算限制在局部窗口内，并使用移位的窗口来建模整个图像。NAT [16] 模拟卷积操作，并在每个特征的邻域内计算注意力。DAT [41] 设计了一个可变形注意力模块，以实现依赖于数据的注意力模式。BiFormer [50] 使用双层路由注意力动态确定每个查询的兴趣区域。GRL [21] 采用锚定条纹注意力、窗口注意力和通道注意力的混合，以实现高效图像恢复。

然而，这些方法本质上限制了自注意力的全局感受野，或者容易受到特定设计的注意力模式的影响，这阻碍了它们在通用目的中的即插即用适应性。

# 2.2. 线性注意力

与限制感受野的想法相比，线性注意力通过降低计算复杂度直接应对计算挑战。先驱工作 [19] 弃用了 Softmax 函数，并用映射函数 作用于 和 ，从而将计算复杂度降低至 。然而，此类近似导致了性能显著下降。为了解决这个问题，Efficient Attention [34] 对 和 同时应用 Softmax 函数。SOFT [27] 和 Nyströmformer [44] 采用矩阵分解进一步近似 Softmax 操作。Castling-ViT [45] 使用 Softmax 注意力作为辅助训练工具，并在推理过程中完全采用线性注意力。FLatten Transformer [14] 提出聚焦函数并采用深度卷积以保持特征多样性。

虽然这些方法有效，但它们仍然在与线性注意力的表达力有限问题作斗争。在本文中，我们不是增强 Softmax 或线性注意力，而是提出了一种新颖的注意力范式，该范式整合了这两种注意力类型，在各种任务中实现了更优的性能。

# 3.预备知识

在本节中，我们首先回顾了现代视觉变换器中自注意力的通用形式，并简要分析了 Softmax 和线性注意力的优缺点。

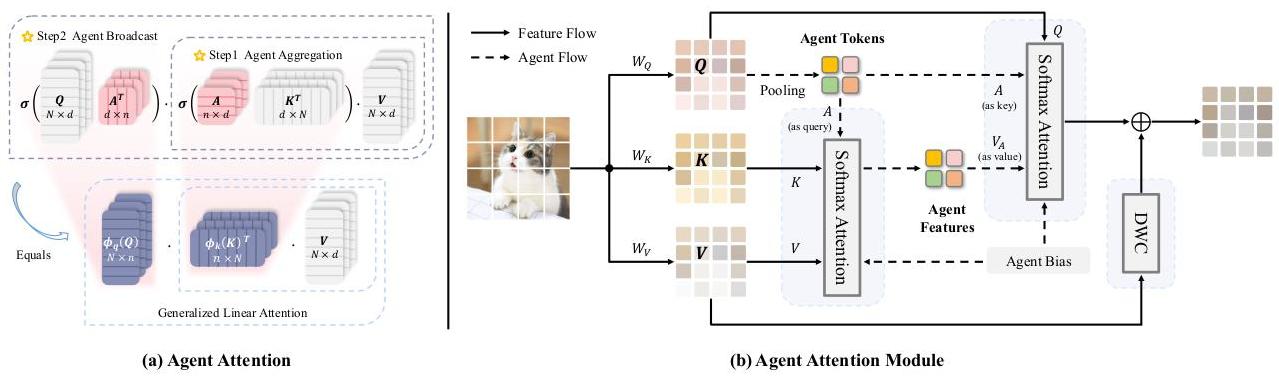


图 2. 我们代理注意力和代理注意力模块的示意图。（a）代理注意力使用代理标记来聚合全局信息并将其分配给单个图像标记，实现了 Softmax 和线性注意力的实用整合。 表示 Softmax 函数。在（b）中，我们描述了代理注意力模块的信息流。作为一个示例，我们通过池化获取代理标记。随后，代理标记被用来从 聚合信息，并且 查询从代理特征中获取特征。此外，采用了代理偏差和 DWC 来添加位置信息并保持特征多样性。

# 3.1. 自注意力的通用形式

在每个头中，输入为 个标记，表示为 ，自注意力可以如下公式化：

其中 表示投影矩阵， 和 分别是模块和每个头的通道维度， 表示相似度函数。

# 3.2. Softmax 注意力和线性注意力

当在公式 (1) 中使用 时，它变成了 Softmax 注意力 [38]，这在现代视觉 Transformer 设计中取得了巨大成功。然而，Softmax 注意力需要计算所有查询-键对的相似度，导致 复杂度。因此，在全局感受野中使用 Softmax 注意力会导致计算复杂度过高。为了解决这个问题，之前的工作尝试通过设计稀疏全局注意力 或窗口注意力 模式来减少标记 的数量。虽然有效，但这些策略不可避免地牺牲了自注意力在长距离建模方面的能力。

相比之下，线性注意力 [19] 以线性复杂度 高效地解决了计算挑战。具体来说，精心设计的映射函数分别应用于 和 ，即 。这使我们有机会根据矩阵乘法的结合律改变计算顺序，从 到 。如图 1 所示，这样做将关于标记数量的计算复杂度降低到 。

然而，设计有效的映射函数 被证明是一个非平凡的任务。简单的函数 [34] 如 ReLU 会导致性能显著下降，而更复杂的设计 [7] 或矩阵分解方法 可能会引入额外的计算开销。总的来说，当前的线性注意力方法仍然不如 Softmax 注意力，这限制了它们的实际应用。

# 4. 代理变换器（Agent Transformer）

如第3节所述，Softmax 和线性注意力要么受到计算复杂度过高的困扰，要么模型表现力不足。之前的研究通常将这两种注意力范式视为不同的方法，并尝试减少 Softmax 注意力的计算成本或提高线性注意力的性能。在本节中，我们提出了一个新的注意力范式，名为代理注意力（Agent Attention），它实际上形成了 Softmax 和线性注意力的优雅整合，同时享有线性复杂度和高表现力的好处。

# 4.1. 代理注意力（Agent Attention）

为了简化，我们将 Softmax 和线性注意力简写为：

其中 表示查询、键和值矩阵， 代表 Softmax 函数。然后我们的代理注意力可以写成：

它等同于：

其中 是我们新定义的代理令牌。

如方程（3）和图2(a)所示，我们的代理注意力机制包括两个Softmax注意力操作，即代理聚合和代理广播。具体来说，我们最初将代理标记 视为查询，并在 与 之间进行注意力计算，以从所有值中聚合代理特征 。随后，我们在第二次注意力计算中将 作为键， 作为值，与查询矩阵 配对，将全局信息从代理特征广播到每个查询标记，并获得最终输出 。这样，我们避免了 与 之间成对相似度的计算，同时通过代理标记保持了每个查询-键对之间的信息交换。

新定义的代理标记 本质上充当了 的代理，从 和 中聚合全局信息，然后将其广播回 。实际上，我们将代理标记的数量 设置为一个小的超参数，从而相对于输入特征的数量 实现了 的线性计算复杂度，同时保持了全局上下文建模能力。

有趣的是，如方程（4）和图2(a)所示，我们在实践中整合了强大的Softmax注意力和高效的线性注意力，通过使用两个Softmax注意力操作，建立了一个广义的线性注意力范式，其等价映射函数定义为 。

在实际应用中，可以通过不同的方法获取代理标记，例如简单地将其设置为一组可学习的参数，或通过池化从输入特征中提取。值得注意的是，还可以使用更先进的技术，如变形点 [41] 或标记合并 [3] 来获取代理标记。在本文中，我们采用了简单的池化策略来获取代理标记，效果已经非常好。

# 4.2. 代理注意力模块

代理注意力继承了Softmax和线性注意的优点。在实际应用中，我们对代理注意力进行了两项改进，以最大化其潜力。

代理偏置。为了更好地利用位置信息，我们为我们的代理注意力提出了一种精心设计的代理偏置。具体来说，受到RPE [33]的启发，我们在注意力计算中引入了代理偏置，即，

其中 是我们的代理偏置。为了参数效率，我们使用三个偏置组件构建每个代理偏置，而不是直接将 设置为可学习参数（见附录）。代理偏置通过增加空间信息增强了原始的代理注意力，帮助不同的代理标记专注于不同的区域。如表6所示，在引入我们的代理偏置项后，可以观察到显著的改进。

多样性恢复模块。尽管代理注意力从低计算复杂度和高模型表现力中受益，但作为广义线性注意力，它也受到特征多样性不足的影响 [14]。作为一种补救措施，我们遵循 [14] 并采用深度卷积（DWC）模块来保持特征多样性。

代理注意力模块。基于这些设计，我们提出了一个名为代理注意力模块的新型注意力模块。如图2(b)所示，我们的模块由三部分组成，分别是纯代理注意力、代理偏置和DWC模块。我们的模块可以表示为：

其中 和 。

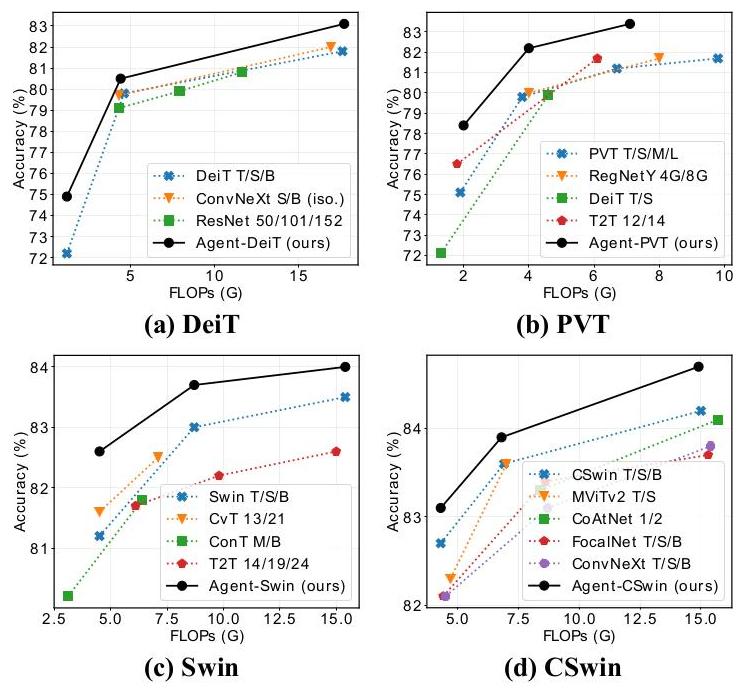
结合了Softmax和线性注意的优点，我们的模块具有以下优势：

（1）高效计算和高度表现力。之前的工作通常将Softmax注意力和线性注意力视为两种不同的注意力范式，旨在解决它们各自的局限性。作为这两种注意力形式的无缝集成，我们的代理注意力自然地继承了这两者的优点，同时享受低计算复杂度和高模型表达能力。

(2) 大感受野。我们的模块在保持相同计算量的同时，可以采用大感受野。现代视觉Transformer模型通常采用稀疏注意力 或窗口注意力 来减轻Softmax注意力的计算负担。得益于线性复杂度，我们的模型在保持相同计算量的同时，可以享受大甚至全局感受野的优势。

# 4.3. 实现

我们的代理注意力模块可以作为插件模块，并且可以轻松地应用于各种现代视觉Transformer架构。作为展示，我们实证地将我们的方法应用于四种先进且具有代表性的Transformer模型，包括DeiT [37]、PVT [39]、Swin [25] 和 CSwin [12]。我们还把代理注意力应用于Stable Diffusion [30] 以加速图像生成。详细的模型架构展示在附录中。



| Method | Reso | #Params | FLOPs | Top-1 |
| --- | --- | --- | --- | --- |
| DeiT-T [37] |  | 5.7M | 1.2G | 72.2 |
| Agent-DeiT-T |  | 6.0M | 1.2G | 74.9 |
| DeiT-S |  | 22.1M | 4.6G | 79.8 |
| Agent-DeiT-S |  |  | 4.4G | 80.5 |
| PVT-T [39] |  | 13.2M | 1.9G | 75.1 |
| Agent-PVT-T |  | 11.6M | 2.0G | 78.4 |
| PVT-S |  | 24.5M | 3.8G | 79.8 |
| Agent-PVT-S |  | 20.6M | 4.0G | 82.2 |
| PVT-M |  | 44.2M | 6.7G | 81.2 |
| Agent-PVT-M |  | 35.9M | 7.0G | 83.4 |
| PVT-L |  | 61.4M | 9.8G | 81.7 |
| Agent-PVT-L |  | 48.7M | 10.4G | 83.7 |
| Swin-T [25] |  | 29M | 4.5G | 81.3 |
| Agent-Swin-T |  | 29M | 4.5G |  |
| Swin-S |  | 50M | 8.7G | 83.0 |
| Agent-Swin-S |  | 50M | 8.7G | 83.7 |
| Swin-B |  | 88M | 15.4G | 83.5 |
| Agent-Swin-B |  | 88M | 15.4G | 84.0 (+0.5) |
| Swin-B |  | 88M | 47.0G | 84.5 |
| Agent-Swin-B |  | 88M | 46.3G | 84.9 |
| CSwin-B [12] |  | 78M | 15.0G | 84.2 |
| Agent-CSwin-B |  | 73M | 14.9G | 84.7 (+0.5) |
| CSwin-B |  | 78M | 47.0G | 85.4 |
| Agent-CSwin-B |  | 73M | 46.3G | 85.8 |

图3. 不同模型在ImageNet-1K上的对比。完整的对比表格见附录。

# 5. 实验

为了验证我们方法的有效性，我们在ImageNet-1K分类 [9]、ADE20K语义分割 [49] 和COCO对象检测 [22] 上进行实验。此外，我们将代理注意力集成到最先进的生成模型Stable Diffusion [30] 中。我们还构建了具有大感受野的高分辨率模型，以最大化代理注意力的优势。另外，我们进行了足够的消融实验，以展示每个设计的效果。

# 5.1. ImageNet-1K 分类

ImageNet [9] 包含1000个类别，有120万张训练图像和5万张验证图像。我们在四个代表性的视觉Transformer上实现我们的模块，并与各种最先进的模型在验证分割上的top-1准确率进行比较。

训练设置展示在附录中。

结果。如图3所示，在多种模型中用代理注意力替换Softmax注意力会带来性能的显著提升。例如，Agent-PVT-S在仅使用 的参数和 的浮点运算次数的情况下超越了PVT-L。Agent-Swin-T/S在保持相似的浮点运算次数的同时，性能超过了Swin-T/S 和 。这些结果毫无疑问地证明了我们的方法具有稳健的优势，并且适用于不同的架构。

推断时间。我们进一步通过在多种设备上部署模型来进行实际速度测量。如

| (a) Mask R-CNN Object Detection on COCO | | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Method | FLOPs | Sch. |  |  |  |  |  |  |
| PVT-T | 240G | 1x | 36.7 | 59.2 | 39.3 | 35.1 | 56.7 | 37.3 |
| Agent-PVT-T | 230G | 1x | 41.4 | 64.1 | 45.2 | 38.7 | 61.3 | 41.6 |
| PVT-S | 305G | 1x | 40.4 | 62.9 | 43.8 | 37.8 | 60.1 | 40.3 |
| Agent-PVT-S | 293G | 1x | 44.5 | 67.0 | 49.1 | 41.2 | 64.4 | 44.5 |
| PVT-M | 392G | 1x | 42.0 | 64.4 | 45.6 | 39.0 | 61.6 | 42.1 |
| Agent-PVT-M | 400G | 1x | 45.9 | 67.8 | 50.4 | 42.0 | 65.0 | 45.4 |
| PVT-L | 494G | 1x | 42.9 | 65.0 | 46.6 | 39.5 | 61.9 | 42.5 |
| Agent-PVT-L | 510G | 1x | 46.9 | 69.2 | 51.4 | 42.8 | 66.2 | 46.2 |
| Swin-T | 267G | 1x | 43.7 | 66.6 | 47.7 | 39.8 | 63.3 | 42.7 |
| Agent-Swin-T | 276G | 1x | 44.6 | 67.5 | 48.7 | 40.7 | 64.4 | 43.4 |
| Swin-T | 267G | 3x | 46.0 | 68.1 | 50.3 | 41.6 | 65.1 | 44.9 |
| Agent-Swin-T | 276G | 3x | 47.3 | 69.5 | 51.9 | 42.7 | 66.4 | 46.2 |
| Swin-S | 358G | 1x | 45.7 | 67.9 | 50.4 | 41.1 | 64.9 | 44.2 |
| Agent-Swin-S | 364G | 1x | 47.2 | 69.6 | 52.3 | 42.7 | 66.6 | 45.8 |

(b) 在COCO上的级联Mask R-CNN目标检测

| Method | FLOPs | Sch. |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Swin-T | 745G | 1x | 48.1 | 67.1 | 52.2 | 41.7 | 64.4 | 45.0 |
| Agent-Swin-T | 755G | 1x | 49.2 | 68.6 | 53.2 | 42.7 | 65.6 | 45.9 |
| Swin-T | 745G | 3x | 50.4 | 69.2 | 54.7 | 43.7 | 66.6 | 47.3 |
| Agent-Swin-T |  | 3x | 51.4 | 70.2 | 55.9 | 44.5 | 67.6 | 48.4 |
| Swin-S | 837G | 3x | 51.9 | 70.7 | 56.3 | 45.0 | 68.2 | 48.8 |
| Agent-Swin-S | 843G | 3x | 52.6 | 71.3 | 57.1 | 45.5 | 68.9 | 49.2 |
| Swin-B | 981 G | 3x | 51.9 | 70.5 | 56.4 | 45.0 | 68.1 | 48.9 |
| Agent-Swin-B | 990G | 3x | 52.6 | 71.1 | 57.1 | 45.3 | 68.6 | 49.2 |

表1。在COCO数据集上的结果。FLOPs是在输入分辨率为 的骨干网络、FPN和检测头上计算的。我们在下游任务中适当地增加了代理标记的数量，以更好地建模高分辨率图像。

图4显示，我们的模型在CPU上的推断速度提高了1.7到2.1倍，同时性能也有所提升。在RTX3090 GPU和A100 GPU上，我们的模型也实现了 到 更快的推断速度。

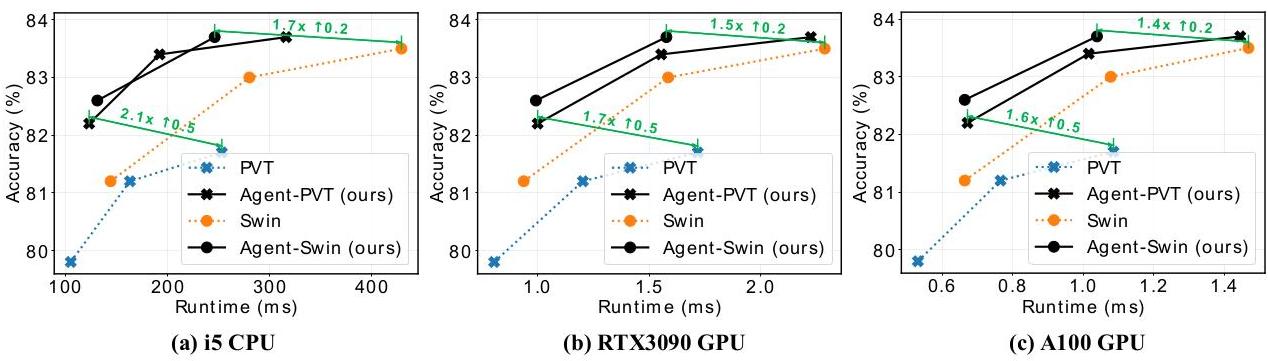


图4。ImageNet上的准确率-运行时间曲线。运行时间是在图像分辨率为224x224时测试的。

| Semantic Segmentation on ADE20K | | | | | |
| --- | --- | --- | --- | --- | --- |
| Backbone | Method | FLOPs | #Params | mIoU | mAcc |
| PVT-T | S-FPN | 158G | 17M | 36.57 | 46.72 |
| Agent-PVT-T | S-FPN | 147G | 15M | 40.18 | 51.76 |
| PVT-S | S-FPN | 225G | 28M | 41.95 | 53.02 |
| Agent-PVT-S | S-FPN | 211G | 24M | 44.18 | 56.17 |
| PVT-L | S-FPN | 420G | 65M | 43.49 | 54.62 |
| Agent-PVT-L | S-FPN | 434G | 52M | 46.52 | 58.50 |
| Swin-T | UperNet | 945G | 60M | 44.51 | 55.61 |
| Agent-Swin-T | UperNet | 954G | 61M | 46.68 | 58.53 |

表2。语义分割的结果。FLOPs是在输入图像分辨率为 的编码器和解码器上计算的。S-FPN指的是SemanticFPN [20]模型。

# 5.2. 目标检测

COCO [22] 目标检测和实例分割数据集包含 训练图像和 验证图像。我们将我们的模型应用于 RetinaNet [23]、Mask R-CNN [17] 和级联 Mask R-CNN [4] 框架，以评估我们的方法性能。通过使用 和 调度以及不同检测头的系列实验表明，我们的模型在所有配置下都表现出一致的改进。Agent-PVT 在 +3.9 到 +4.7 的框 AP 范围内超过了 PVT 模型，而 Agent-Swin 通过高达 +1.5 的框 AP 超越了 Swin 模型。这些显著的改进可以归因于我们设计带来的大感受野，证明了在高清场景中代理注意力的有效性。

# 5.3. 语义分割

ADE20K [49] 是语义分割领域广泛认可的基准，包括 训练图像和 验证图像。我们将我们的模型应用于两个示例分割模型，即 SemanticFPN [20] 和 UperNet [42]。结果展示在表 2 中。值得注意的是，我们的 Agent-PVT-T 和 Agent-Swin-T 分别比它们的对比模型高出 +3.61 和 +2.17 的 mIoU。结果表明，我们的模型与各种分割基础模型兼容，并且一致地实现了改进。

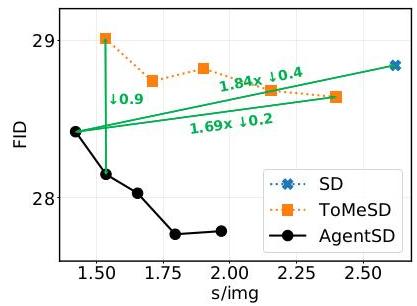


图 5. 稳定扩散、ToMeSD 和我们的 AgentSD 的定量结果。对于 ToMeSD，我们采用合并比例 来构建五个不同的模型。此外，我们将代理注意力应用于每个 ToMeSD 模型，以获得相应的 AgentSD 模型。

# 5.4. 用于稳定扩散的代理注意力

扩散模型的出现使得生成高分辨率和高质量图像成为可能。然而，当前的扩散模型主要使用具有全局感受野的原始Softmax注意力机制，导致计算成本巨大且生成速度缓慢。鉴于此，我们将我们的代理注意力机制应用于Stable Diffusion [30]，希望提高模型的生成速度。令人惊讶的是，经过简单的调整后，使用代理注意力的Stable Diffusion模型，称为AgentSD，显示出显著的生成速度提升，并且在不进行额外训练的情况下产生了更好的图像质量。

将代理注意力应用于Stable Diffusion。我们实际上将代理注意力应用于ToMeSD模型 [1]。ToMeSD通过在注意力计算之前减少Stable Diffusion中的标记数量，提高了生成速度。尽管如此，合并后的标记数量仍然可观，导致持续的复杂性和延迟。因此，我们用我们的代理注意力替换了ToMeSD模型中使用的Softmax注意力，以进一步加快速度。我们通过实验发现，在通过标记合并 [3] 生成代理标记时，我们的代理注意力可以直接应用于Stable Diffusion和ToMeSD模型，而无需额外训练。然而，我们无法应用代理偏置和DWC。

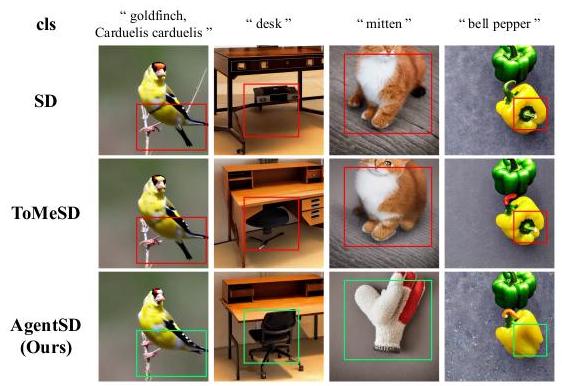


图6. 由Stable Diffusion、ToMeSD ) 和AgentSD 生成的样本。提示是“一张高质量的{cls}照片。”。

以这种方式。作为一种补救措施，我们对代理注意力进行了两项简单的调整，这些调整在附录中详细描述。此外，我们在早期扩散生成步骤中应用代理注意力，并保持后续步骤不变，从而获得了显著的提升。

定量结果。我们遵循 [1]，并定量比较了 AgentSD 与 Stable Diffusion 和 ToMeSD。如图 5 所示，ToMeSD 加速了 Stable Diffusion 的运行，同时保持了相似的图像质量。AgentSD 不仅进一步加速了 ToMeSD，而且显著提高了图像生成质量。具体来说，在保持卓越的图像生成质量的同时，AgentSD 相较于 Stable Diffusion 和 ToMeSD 分别实现了 和 更快的生成速度。在相同的生成速度下，AgentSD 生成的样本的 FID 分数比 ToMeSD 低 0.9。有关实验细节和完整的比较表格，请参见附录。

可视化。我们在图 6 中展示了一些可视化结果。与 Stable Diffusion 和 ToMeSD 相比，AgentSD 显著减少了模糊性和生成错误。例如，在第一列中，Stable Diffusion 和 ToMeSD 生成的鸟有一只腿和两条尾巴，而 AgentSD 的样本没有这个问题。在第三列中，当给定提示 "一张高质量的手套照片。" 时，Stable Diffusion 和 ToMeSD 错误地生成了猫，而 AgentSD 生成了正确的图像。

AgentSD 用于微调。我们将代理注意力应用于基于 SD 的 Dreambooth [31]，以验证其在微调情况下的性能。在微调后，代理注意力可以被整合到所有扩散生成步骤中，相较于原始 Dreambooth 实现 的生成速度加速。具体细节请参见附录。

# 5.5. 大感受野与高分辨率

大感受野。现代视觉变换器通常将自注意力计算限制在局部窗口内以降低计算复杂度，例如Swin [25]。在表3中，我们逐渐增大了Agent-Swin-T的窗口大小，从 到 。显然，随着感受野的扩大，模型的性能持续提高。这表明虽然窗口注意力模式有效，但它不可避免地妥协了自注意力的长距离建模能力，并且仍然不如全局注意力。由于代理注意力的线性复杂度，我们可以在保持相同计算复杂度的同时，从全局感受野中获益。

|  | Window | FLOPs | #Param | Acc. | Diff. |
| --- | --- | --- | --- | --- | --- |
| Agent-Swin-T |  | 4.5G | 29M | 82.0 | -0.6 |
|  | 4.5G | 29M | 82.2 | -0.4 |
|  | 4.5G | 29M | 82.4 | -0.2 |
|  | 4.5G | 29M | 82.6 | Ours |
| Swin-T |  | 4.5G | 29M | 81.3 | -1.3 |

表3. 基于Agent-Swin-T的窗口大小消融研究。

| Method | Reso | #Params | Flops | Top-1 |
| --- | --- | --- | --- | --- |
| DeiT-B [37] |  | 86.6M | 17.6G | 81.8 |
| DeiT-S |  | 22.2M | 18.8G | 82.9 (+1.1) |
| Agent-DeiT-B |  | 87.2M | 17.6G | 82.0 (+0.2) |
| Agent-DeiT-S |  | 23.1M | 17.7G | 83.1 (+1.3) |
| PVT-L [39] |  | 61.4M | 9.8G | 81.7 |
| PVT-M |  | 44.3M | 8.8G |  |
| Agent-PVT-L |  | 48.7M | 10.4G | 83.7 |
| Agent-PVT-M |  | 36.1M | 9.2G | 83.8 (+2.1) |
| Swin-B [25] |  | 88M | 15.4G | 83.5 |
| Swin-S |  | 50M | 14.7G | 83.7 (+0.2) |
| Agent-Swin-B |  | 88M | 15.4G | 84.0 (+0.5) |
| Agent-Swin-S |  | 50M | 14.6G | 84.1 (+0.6) |

表4. 通过增加分辨率进行扩展。所有这些模型都是从零开始训练了300个周期。

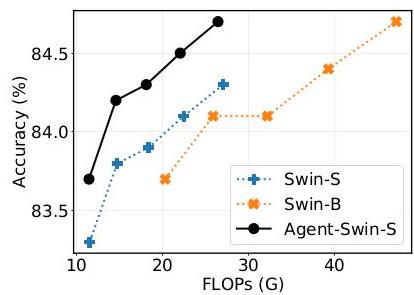


图7. 将分辨率增加到 ， 。所有这些模型都是从相应的 分辨率模型微调了30个周期。

高分辨率。由于Softmax注意力的二次复杂度限制，当前的视觉变换器模型通常通过增加模型深度和宽度来扩展。基于[36]的见解，我们发现提高分辨率可能是扩展视觉变换器的一种更有效的方法，特别是对于那些采用具有全局感受野的代理注意力的模型。如表4所示，Agent-DeiT-B与DeiT-B相比，准确度提高了0.2，而Agent-DeiT-S在 分辨率下仅用四分之一的参数就达到了83.1的准确度。我们在扩展Agent-PVT-M的分辨率时也观察到了类似的趋势。

| (a) Comparison on DeiT-T Setting | | | |
| --- | --- | --- | --- |
| Linear Attention | FLOPs | #Param | Acc. |
| Hydra Attn [2] | 1.1G | 5.7M | 68.3 |
| Efficient Attn [34] | 1.1G | 5.7M | 70.2 |
| Linear Angular Attn [45] | 1.1G | 5.7M | 70.8 |
| Focused Linear Attn [14] | 1.1G | 6.1M | 74.1 |
| Ours | 1.2G |  | 74.9 |

(b) 在Swin-T设置上的比较

| Linear Attention | FLOPs | #Param | Acc. |
| --- | --- | --- | --- |
| Hydra Attn [2] | 4.5G | 29M | 80.7 |
| Efficient Attn [34] | 4.5G | 29M | 81.0 |
| Linear Angular Attn [45] | 4.5G | 29M | 79.4 |
| Focused Linear Attn [14] | 4.5G | 29M | 82.1 |
| Ours | 4.5G | 29M | 82.6 |

表5. 在DeiT-Tiny和Swin-Tiny结构上不同线性注意力设计的比较。

|  | FLOPs | #Param | Acc. | Diff. |
| --- | --- | --- | --- | --- |
| Vanilla Linear Attention | 4.5G | 29M | 77.8 | -4.8 |
| Agent Attention | 4.5G | 29M | 79.0 | -3.6 |
| + Agent Bias | 4.5G | 29M | 81.1 | -1.5 |
| + DWC | 4.5G | 29M | 82.6 | Ours |
| Swin-T | 4.5G | 29M | 81.3 |  |

表6. 对代理注意力的每个模块以及Agent-Swin-S的消融研究。在图7中，我们逐步提高了Agent-Swin-S、Swin-S和Swin-B的分辨率。很明显，在高分辨率场景中，我们的模型始终提供显著优越的结果。

|  | FLOPs | #Param | Acc. | Diff. |
| --- | --- | --- | --- | --- |
| Static Agent | 4.5G | 29M | 82.2 | -0.4 |
| Dynamic Agent | 4.5G | 29M | 82.6 | Ours |
| Swin-T | 4.5G | 29M | 81.3 | -1.3 |

表7. 对代理标记类型的消融研究。

# 5.6. 与其他线性注意力的比较

我们使用DeiT-T和Swin-T进行了我们的代理注意力与其他线性注意力方法的比较。如表5所示，将DeiT-T和Swin-T中使用的Softmax注意力替换为各种线性注意力方法通常会导致性能显著下降。值得注意的是，我们的模型不仅超过了所有其他方法，也超过了Softmax基线。

# 5.7. 消融研究

在本节中，我们对我们的代理注意力模块中的关键组成部分进行消融，以验证这些设计的效果。我们基于Agent-Swin-T在ImageNet-1K分类上报告了结果。

代理注意力、代理偏置和DWC。我们首先评估我们的代理注意力三个关键设计的有效性。我们将Swin-T中的Softmax注意力替换为普通的线性注意力，然后逐步引入代理注意力、代理偏置和DWC来创建Agent-Swin-T。如表6所示，这三个设计的加入分别带来了 和1.5的准确度提升。

代理标记的类型。如第4.1节所述，代理标记可以通过多种方法获得。作为一种

| Num of Agent Tokens | | | | FLOPS | #Param | Acc. | Diff. |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Stage1 | Stage2 | Stage3 | Stage4 |  |  |  |  |
| 49 | 49 | 49 | 49 | 4.7G | 29M | 82.6 | -0.0 |
| 9 | 16 | 49 | 49 | 4.5G | 29M | 82.6 | Ours |
| 9 | 16 | 25 | 49 | 4.5G | 29M | 82.2 | -0.4 |
| 4 | 9 | 49 | 49 | 4.5G | 29M | 82.4 | -0.2 |
| Swin-T | | | | 4.5G | 29M | 81.3 | -1.3 |

表8. 对代理标记数量的消融研究。

| Stages w/ Agent Attn | | | | FLOPS | #Param | Acc. | Diff. |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Stage1 | Stage2 | Stage3 | Stage4 |  |  |  |  |
|  |  |  |  | 4.5G | 29M | 81.7 | -0.9 |
|  |  |  |  | 4.5G | 29M | 81.8 | -0.8 |
|  |  |  |  | 4.5G | 29M | 82.6 | Ours |
|  |  |  |  | 4.5G | 29M | 82.5 | -0.1 |
| Swin-T | | | | 4.5G | 29M | 81.3 | -1.3 |

表9. 对在Swin-T结构的不同阶段应用代理注意力模块的消融研究。

展示中，我们将代理标记大致分为两种类型：静态和动态。前者将代理标记设置为可学习的参数，而后者使用池化来获取代理标记。如表7所示，动态代理标记产生了更好的结果。

对代理标记数量的消融研究。模型的计算复杂度可以通过改变代理标记的数量来调节。如表8所示，我们观察到在模型的较浅层中适当减少代理标记的数量对性能没有不利影响。然而，在更深层减少代理标记会导致性能下降。

不同阶段的代理注意力。我们在不同阶段将Softmax注意力替换为我们的代理注意力。如表9所示，替换前三个阶段会导致性能提升1.3%，而替换最后一个阶段会略微降低整体准确性。我们将这一结果归因于前三个阶段较高的分辨率，这对于具有全局感受野的代理注意力模块更为有利。

# 6. 结论

本文提出了一种新的注意力范式，名为代理注意力，它适用于各种视觉Transformer模型。作为Softmax和线性注意力的优雅集成，代理注意力既具有高表达力，又具有低计算复杂度。在图像分类、语义分割和目标检测上的大量实验明确地确认了我们的方法的有效性，特别是在高分辨率场景下。当与稳定扩散集成时，我们的代理注意力加速了图像生成，并在不需要额外训练的情况下显著提高了图像质量。由于其与标记数量线性相关的复杂度及其强大的表示力，代理注意力可能为处理超长标记序列的挑战性任务，如视频建模和多模态基础模型，铺平道路。

# 参考文献

[1] Daniel Bolya and Judy Hoffman. Token merging for fast stable diffusion. In CVPRW, 2023. 6, 7, 11

[2] Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, and Judy Hoffman. Hydra attention: Efficient attention with many heads. In ECCVW, 2022. 8

[3] Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your ViT but faster. In ICLR, 2023. 4, 6

[4] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: Delving into high quality object detection. In CVPR, 2018. 6, 15

[5] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In ECCV, 2020. 1

[6] Bowen Cheng, Alex Schwing, and Alexander Kirillov. Per-pixel classification is not all you need for semantic segmentation. In NeurIPS, 2021. 1

[7] Krzysztof Choromanski, Valerii Likhosherstov, David Do-han, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. Rethinking attention with performers. In ICLR, 2021. 3

[8] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In CVPRW, 2020. 13

[9] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Fei-Fei. Imagenet: A large-scale hierarchical image database. In CVPR, 2009. 5, 11

[10] Tim Dettmers, Mike Lewis, Sam Shleifer, and Luke Zettle-moyer. 8-bit optimizers via block-wise quantization. In ICLR, 2022. 13

[11] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. In NeurIPS, 2021. 11

[12] Xiaoyi Dong, Jianmin Bao, Dongdong Chen, Weiming Zhang, Nenghai Yu, Lu Yuan, Dong Chen, and Baining Guo. Cswin transformer: A general vision transformer backbone with cross-shaped windows. In CVPR, 2022. 3, 4, 5, 13

[13] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Syl-vain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In ICLR, 2021. 1, 2

[14] Dongchen Han, Xuran Pan, Yizeng Han, Shiji Song, and Gao Huang. Flatten transformer: Vision transformer using focused linear attention. In

[15] Yizeng Han, Dongchen Han, Zeyu Liu, Yulin Wang, Xuran Pan, Yifan Pu, Chao Deng, Junlan Feng, Shiji Song, and Gao Huang. Dynamic perceiver for efficient visual recognition. In

[16] Ali Hassani, Steven Walton, Jiachen Li, Shen Li, and Humphrey Shi. Neighborhood attention transformer. In CVPR, 2023. 2

[17] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Gir-shick. Mask r-cnn. In ICCV, 2017. 6, 15

[18] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a

two time-scale update rule converge to a local nash equilibrium. In NeurIPS, 2017. 11

[19] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In ICML, 2020. 2, 3

[20] Alexander Kirillov, Ross Girshick, Kaiming He, and Piotr Dollár. Panoptic feature pyramid networks. In CVPR, 2019.

[21] Yawei Li, Yuchen Fan, Xiaoyu Xiang, Denis Demandolx, Rakesh Ranjan, Radu Timofte, and Luc Van Gool. Efficient and explicit modelling of image hierarchies for image restoration. In CVPR, 2023. 2

[22] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In ECCV, 2014. 5, 6, 13

[23] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In ICCV, 2017. 6, 14, 15

[24] Luping Liu, Yi Ren, Zhijie Lin, and Zhou Zhao. Pseudo numerical methods for diffusion models on manifolds. In ICLR, 2022. 11

[25] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In ICCV, 2021. 2, 3, 4, 5, 7, 13

[26] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In ICLR, 2018. 13

[27] Jiachen Lu, Jinghan Yao, Junge Zhang, Xiatian Zhu, Hang , Weiguo Gao, Chunjing , Tao Xiang, and Zhang. Soft: Softmax-free transformer with linear complexity. In NeurIPS, 2021. 2, 3

[28] Boris T Polyak and Anatoli B Juditsky. Acceleration of stochastic approximation by averaging. SIAM Journal on Control and Optimization, 1992. 13

[29] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In ICML, 2021. 1

[30] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In CVPR, 2022. 4, 5, 6, 11

[31] Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. In CVPR, 2023. 7, 12, 13

[32] Maximilian Seitzer. pytorch-fid: FID Score for PyTorch. https://github.com/mseitzer/pytorch-fid, 2020. Version 0.3.0.11

[33] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. In , 2018. 4

[34] Zhuoran Shen, Mingyuan Zhang, Haiyu Zhao, Shuai Yi, and Hongsheng Li. Efficient attention: Attention with linear complexities. In WACV, 2021. 2, 3, 8

[35] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denois-ing diffusion implicit models. In ICLR, 2021. 13

[36] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In ICML, 2019. 7

[37] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In

[38] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszko-reit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In NeurIPS, 2017. 1, 2, 3

[39] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In , 13

[40] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pvt v2: Improved baselines with pyramid vision transformer. Computational Visual Media, 2022. 3, 4

[41] Zhuofan Xia, Xuran Pan, Shiji Song, Li Erran Li, and Gao Huang. Vision transformer with deformable attention. In CVPR, 2022. 2, 4

[42] Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified perceptual parsing for scene understanding. In ECCV, 2018. 6, 14, 15

[43] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M Alvarez, and Ping Luo. Segformer: Simple and efficient design for semantic segmentation with transformers. In NeurIPS, 2021. 1

[44] Yunyang Xiong, Zhanpeng Zeng, Rudrasis Chakraborty, Mingxing Tan, Glenn Fung, Yin Li, and Vikas Singh. Nyströmformer: A nyström-based algorithm for approximating self-attention. In AAAI, 2021. 2, 3

[45] Haoran You, Yunyang Xiong, Xiaoliang Dai, Bichen Wu, Peizhao Zhang, Haoqi Fan, Peter Vajda, and Yingyan Celine Lin. Castling-vit: Compressing self-attention via switching towards linear-angular attention at vision transformer inference. In CVPR, 2023. 2, 8

[46] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In ICCV, 2019. 13

[47] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In ICLR, 2018. 13

[48] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In AAAI, 2020. 13

[49] Bolei Zhou, Hang Zhao, Xavier Puig, Tete Xiao, Sanja Fi-dler, Adela Barriuso, and Antonio Torralba. Semantic understanding of scenes through the ade20k dataset. IJCV, 2019.

[50] Lei Zhu, Xinjiang Wang, Zhanghan Ke, Wayne Zhang, and Rynson WH Lau. Biformer: Vision transformer with bi-level routing attention. In CVPR, 2023. 2

# 附录

# A. 代理偏置的构成

如主论文中所述，为了更好地利用位置信息，我们为我们的代理注意力提出了一种精心设计的代理偏置，即

其中 是我们的代理偏差。为了参数效率，我们使用三个偏差组件来构建每个代理偏差，而不是直接将 设置为可学习的参数。例如， 中的值是从列偏差 、行偏差 和块偏差 演变而来的，其中 是特征图或注意力窗口的高度和宽度，而 是预定义的超参数，远小于 。在计算注意力权重期间， 通过重复或插值被调整到 的尺寸，并且 。reshape 被用作完整的代理偏差。

# B. 代理注意力用于稳定扩散

# B.1. 调整

如主论文中所述，在通过代币合并生成代理代币时，我们的代理注意力可以直接应用于稳定扩散模型，而无需额外训练。然而，未经训练，我们无法应用代理偏差和DWC。作为一种补救措施，我们对代理注意力进行了两项简单调整。一方面，我们将代理注意力模块从

改为

其中 是预定义的超参数。另一方面，与原始的softmax注意力相比，代理注意力的两个softmax注意力操作可能在未经训练的情况下导致更平滑的特征分布。基于这一点，我们略微增加了用于第二次Softmax注意力的尺度，即代理广播。

# B.2. 实验细节

为了定量比较 AgentSD 与 Stable Diffusion 和 ToMeSD，我们遵循 [1]，并使用 Stable Diffusion v1.5 生成 ImageNet-1k [9] 类别的图像，每类两个图像，采用 50 个 PLMS [24] 扩散步骤，cfg 缩放 [11] 为 7.5。随后，我们计算了这些 2,000 个样本与 AgentSD 之间的 FID [18] 分数。GB/img 被测量为增加批处理大小 1 时的总内存使用变化。并使用 [32] 处理 50,000 个 ImageNet-1k 验证示例。为了评估速度，我们计算了在单个 RTX4090 GPU 上所有 2,000 个样本的平均生成时间。

| Method |  | FID | s/img | GB/img |
| --- | --- | --- | --- | --- |
| SD [30] | 0 | 28.84 | 2.62 | 3.13 |
| ToMeSD [1] | 0.1 | 28.64 | 2.40 | 2.55 |
| 0.2 | 28.68 | 2.15 | 2.03 |
| 0.3 | 28.82 | 1.90 | 2.09 |
| 0.4 | 28.74 | 1.71 | 1.69 |
| 0.5 | 29.01 | 1.53 | 1.47 |
| AgentSD | 0.1 | 27.79 | 1.97 | 1.77 |
| 0.2 | 27.77 | 1.80 | 1.60 |
| 0.3 | 28.03 | 1.65 | 2.05 |
| 0.4 | 28.15 | 1.54 | 1.55 |
| 0.5 | 28.42 | 1.42 | 1.21 |

表 10。Stable Diffusion、ToMeSD 的定量结果

|  | 0 | 0.025 | 0.075 | 0.15 |
| --- | --- | --- | --- | --- |
| FID | 28.80 | 28.67 | 28.42 | 28.61 |

表 11。对式 (9) 中因子 的消融研究。

| Scale |  |  |  |  |
| --- | --- | --- | --- | --- |
| FID | 28.86 | 28.64 | 28.42 | 28.60 |

表 12。对第二个 Softmax 注意力使用的尺度的消融研究。

完整的定量结果呈现在表 10 中。与 SD 和 ToMeSD 相比，我们的 AgentSD 不仅加快了生成速度和减少了内存使用，而且显著提高了图像生成质量。

# B.3. 消融研究

我们进一步研究了在将代理注意力应用于 Stable Diffusion 时所做的调整。如表 11 和表 12 所示，对代理注意力的两项调整都提高了 AgentSD 生成的质量。表 13 证明了在早期阶段应用代理注意力可以带来显著的性能提升。

# B.4. AgentSD 用于微调

我们的代理注意力模块同样适用于微调场景。为了验证这一点，我们选择以主体驱动的任务作为示例，并将代理注意力应用于基于SD的Dream-booth [31]。我们通过实验发现，微调使得代理注意力模块能够整合到所有的扩散生成步骤中，与原始的Dreambooth相比，在生成速度上达到了 的加速，同时不牺牲图像质量。此外，微调过程中的时间和内存消耗也可以降低。

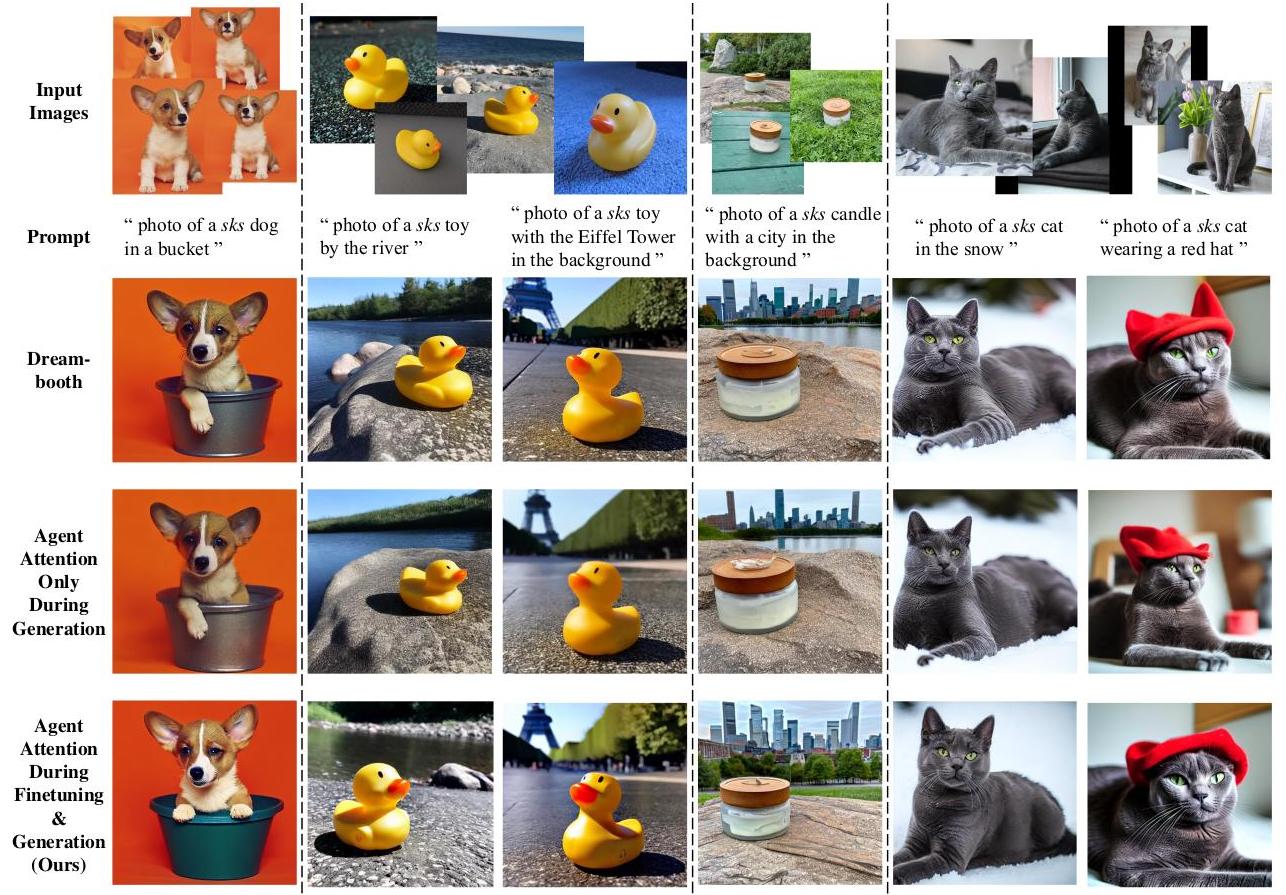


图8. 使用相同种子生成的Dreambooth和我们的Agent Dreambooth样本。在倒数第二行中，我们仅在生成过程中将代理注意力应用于所有扩散步骤，如预期的那样，导致图像质量略有下降。在最后一行中，代理注意力被整合到微调和生成的所有步骤中，从而在不牺牲图像质量的情况下实现了 的速度提升。放大查看最佳效果。

| Steps | early 20% | early 40% | early 60% | early 80% |
| --- | --- | --- | --- | --- |
| FID | 28.58 | 28.42 | 28.83 | 29.77 |
| s/img | 1.50 | 1.42 | 1.39 | 1.34 |

表13. 对应用代理注意力的扩散步骤数量的消融研究。

任务和基线。扩散主体驱动生成任务包括在生成不同背景下的新渲染时保持给定主体的外观，例如，生成一张你的宠物狗跳舞的照片。Dream-booth [31] 通过微调预训练的文本到图像扩散模型，将唯一标识符与给定主体绑定，有效地解决了这一任务。然后可以使用唯一标识符合成主体的新图像。

将代理注意力应用于 Dreambooth。如前所述，Dreambooth[31] 需要一个额外的微调过程。我们探索了两种将代理注意力应用于 Dreambooth 的方法：（1）仅在生成过程中应用；（2）在微调和生成过程中都应用。第一种方法与主论文中详述的 AgentSD 相同，我们通常在生成的早期步骤 应用代理注意力，实现了大约 的速度提升（合并比例 ）。然而，如表格 13 和图 8 倒数第二行所示，将代理注意力应用于更多的扩散步骤以进一步加速会导致图像细节和质量下降。相反，采用第二种方法，即在微调和生成过程中的所有步骤都应用代理注意力模块，可以在不牺牲性能的情况下实现生成过程的 速度提升。此外，微调过程中的时间和内存成本都减少了大约 ，使得可以在单个 RTX4090 GPU 上用大约 7 分钟时间，使用少于 的 GPU 内存进行模型微调。图 8 的最后一行显示了这种设置的结果。

数据集和实验细节。我们采用了由Dreambooth [31] 提供的数据集，该数据集包含15个不同类别的30个主题。它以各种条件、环境和角度捕捉实时的主题和物体。我们使用预训练的 Stable Diffusion v1.5 并在所有扩散生成步骤中应用代理注意力。合并比例 设置为 设置为 0.075，第二个softmax注意力的规模设置为 。我们对所有模型进行800次迭代微调，学习率为1e-6，使用8位AdamW [10] 作为优化器。我们遵循 [31] 并选择sks作为所有设置的唯一标识符。使用DDIM [35] 采样器在单个RTX4090 GPU上进行100步生成，来采样新的合成图像。

可视化和讨论。合成的主题驱动图像显示在图8中。我们得出两个关键观察：1）在微调和生成过程中应用代理注意力的Dreambooth，其保真度和可编辑性与基线Dreambooth相当或超过，并且2）在微调过程中应用代理注意力进一步增强了合成图像的保真度和细节质量，使我们能够将代理注意力应用到所有扩散步骤以获得更多加速。对于第一个观察，第一列显示我们的方法确保合成的狗的颜色与输入图像相比更加一致，并且保持了与原始Dreambooth相当的可编辑性。对于第二个观察，比较第三列的最后两行揭示了在没有微调的情况下将代理注意力应用到所有扩散步骤会产生模糊的图像，而我们的方法产生了更清晰、更锐化的鸭子玩具图像。此外，在第五列中，我们的方法准确地生成了猫的眼睛，而没有微调的代理注意力在这方面失败了。

# C. 数据集和训练设置

# C.1. ImageNet

训练设置。为确保公平比较，我们使用与相应基线模型相同的设置来训练我们的代理注意力模型。具体来说，我们采用AdamW [26]优化器从头开始训练所有模型300个周期，使用余弦学习率衰减和20个周期的线性预热。我们为批次大小1024设置初始学习率为 ，并相对于批次大小线性缩放。遵循DeiT [37]，我们使用Ran-dAugment [8]、Mixup [47]、CutMix [46]和随机擦除[48]来防止过拟合。我们还应用了0.05的权重衰减。为了与[12]保持一致，我们将EMA [28]融入我们的Agent-CSwin模型训练中。对于在较大分辨率下的微调，我们遵循[12, 25]中的设置，并将模型微调30个周期。

| Method | Reso | #Params | Flops | Top-1 |
| --- | --- | --- | --- | --- |
| DeiT-T [37] |  | 5.7M | 1.2G | 72.2 |
| Agent-DeiT-T |  | 6.0M | 1.2G | 74.9 |
| DeiT-S |  | 22.1M | 4.6G | 79.8 |
| Agent-DeiT-S |  | 22.7M | 4.4G | 80.5 |
| DeiT-B [37] |  | 86.6M | 17.6G | 81.8 |
| Agent-DeiT-B |  | 87.2M | 17.6G | 82.0 |
| Agent-DeiT-S |  | 23.1M | 17.7G | 83.1 |
| PVT-T [39] |  | 13.2M | 1.9G | 75.1 |
| Agent-PVT-T |  | 11.6M | 2.0G | 78.4 |
| PVT-S |  | 24.5M | 3.8G | 79.8 |
| Agent-PVT-S |  | 20.6M | 4.0G | 82.2 |
| PVT-M |  | 44.2M | 6.7G | 81.2 |
| Agent-PVT-M |  | 35.9M | 7.0G | 83.4 |
| PVT-L |  | 61.4M | 9.8G | 81.7 |
| Agent-PVT-L |  | 48.7M | 10.4G | 83.7 |
| Agent-PVT-M |  | 36.1M | 9.2G | 83.8 |
| Swin-T [25] |  | 29M | 4.5G | 81.3 |
| Agent-Swin-T |  | 29M | 4.5G | 82.6 |
| Swin-S |  | 50M | 8.7G | 83.0 |
| Agent-Swin-S |  | 50M | 8.7G | 83.7 |
| Swin-B |  | 88M | 15.4G | 83.5 |
| Agent-Swin-B |  | 88M | 15.4G | 84.0 |
| Agent-Swin-S |  | 50M | 14.6G | 84.1 |
| Swin-B |  | 88M | 47.0G | 84.5 |
| Agent-Swin-B |  | 88M | 46.3G | 84.9 |
| CSwin-T [12] |  | 23M | 4.3G | 82.7 |
| Agent-CSwin-T |  | 21M | 4.3G | 83.1 |
| CSwin-S |  | 35M | 6.9G | 83.6 |
| Agent-CSwin-S |  | 33M | 6.8G | 83.9 |
| CSwin-B [12] |  | 78M | 15.0G | 84.2 |
| Agent-CSwin-B |  | 73M | 14.9G | 84.7 |
| CSwin-B |  | 78M | 47.0G | 85.4 |
| Agent-CSwin-B |  | 73M | 46.3G | 85.8 |

表14。代理注意力与其他视觉变换器骨干网络在ImageNet-1K分类任务上的比较。

# C.2. COCO

训练设置。COCO [22] 目标检测和实例分割数据集包含 训练图像和 验证图像。我们使用 样本的子集作为训练集， 用于验证。主干网络在 ImageNet 数据集上使用 AdamW 预训练，遵循原论文中提到的训练配置。应用了标准的数据增强，包括调整大小、随机翻转和归一化。我们将学习率设置为 并遵循 学习计划：整个网络训练 12 个周期，学习率分别在第 8 和第 11 个周期时除以 10。对于某些模型，我们使用 计划：网络训练 36 个周期，学习率分别在第 27 和第 33 个周期时除以 10。主论文中的所有 mAP 结果都是使用输入图像大小 测试的。

(a) COCO 上的 Mask R-CNN 目标检测

| Method | FLOPs | Sch. |  |  |
| --- | --- | --- | --- | --- |
| PVT-T | 240G | 1x | 36.7 59.2 39.3 | 35.1 56.7 37.3 |
| Agent-PVT-T | 230G | 1x | 41.4 64.1 45.2 | 38.7 61.3 41.6 |
| PVT-S | 305G | 1x | 40.4 62.9 43.8 | 37.860.140.3 |
| Agent-PVT-S | 293G | 1x | 44.5 67.0 49.1 | 41.2 64.4 44.5 |
| PVT-M | 392G | 1x | 42.0 64.4 45.6 | 39.0 61.6 42.1 |
| Agent-PVT-M | 400G | 1x | 45.9 67.8 50.4 | 42.0 65.0 45.4 |
| PVT-L | 494G | 1x | 42.9 65.0 46.6 | 39.5 61.9 42.5 |
| Agent-PVT-L | 510G | 1x | 46.9 69.2 51.4 | 42.866.246.2 |
| Swin-T | 267G | 1x | 43.766.647.7 | 39.863.342.7 |
| Agent-Swin-T | 276G | 1x | 44.667.548.7 | 40.764.443.4 |
| Swin-T | 267G | 3x | 46.068.150.3 | 41.665.144.9 |
| Agent-Swin-T | 276G | 3x | 47.369.551.9 | 42.766.446.2 |
| Swin-S | 358G | 1x | 45.767.950.4 | 41.164.944.2 |
| Agent-Swin-S | 364G | 1x | 47.269.652.3 | 42.766.645.8 |
| Swin-S | 358G | 3x | 48.570.253.5 | 43.367.346.6 |
| Agent-Swin-S | 364G | 3x | 48.970.953.6 | 43.867.947.3 |

(b) COCO 上的级联 Mask R-CNN 目标检测

| Method | FLOPs | Sch. |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Swin-T | 745G | 1x | 48.167.152.2 | 41.7 | 64.445.0 |
| Agent-Swin-T | 755G | 1x | 49.268.653.2 | 42.7 | 65.645.9 |
| Swin-T | 745G | 3x | 50.469.254.7 | 43.7 | 66.647.3 |
| Agent-Swin-T | 755G | 3x | 51.470.255.9 | 44.5 | 67.648.4 |
| Swin-S | 837G | 3x | 51.970.756.3 | 45.0 | 68.248.8 |
| Agent-Swin-S | 843G | 3x | 52.671.357.1 | 45.5 | 68.949.2 |
| Swin-B | 981G | 3x | 51.970.556.4 | 45.0 | 68.148.9 |
| Agent-Swin-B | 990G | 3x | 52.671.157.1 | 45.3 | 68.649.2 |

表 15。COCO 数据集上的结果。FLOPs 是在主干网络、FPN 和检测头具有 输入分辨率的情况下计算的。

代理标记数量。我们使用 ImageNet 预训练模型作为主干网络，该模型在四个阶段分别训练时设置的代理标记数量 为 。由于密集预测任务涉及到的图像分辨率比 ImageNet 高，我们适当地增加了代理标记数量以更好地保留丰富的信息。具体来说，对于所有 Agent-PVT 模型，我们为四个阶段分配的代理标记数量为 ，而所有 Agent-Swin 模型，我们分配 。我们使用双线性插值来适应增加的代理标记数量 的代理偏置。同样的策略也应用于 ADE20k 实验中。

RetinaNet在COCO数据集上的目标检测（Sch. 1x）

| Method | FLOPs | AP |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| PVT-T | 221G | 36.7 | 56.9 | 38.9 | 22.6 | 38.8 | 50.0 |
| Agent-PVT-T | 211G | 40.3 | 61.2 | 42.9 | 25.5 | 43.4 | 54.3 |
| PVT-S | 286G | 38.7 | 59.3 | 40.8 | 21.2 | 41.6 | 54.4 |
| Agent-PVT-S | 274G | 44.1 | 65.3 | 47.3 | 29.2 | 47.5 | 59.8 |
| PVT-M | 373G | 41.9 | 63.1 | 44.3 | 25.0 | 44.9 | 57.6 |
| Agent-PVT-M | 382G | 45.8 | 66.9 | 49.1 | 28.8 | 49.2 | 61.7 |
| PVT-L | 475G | 42.6 | 63.7 | 45.4 | 25.8 | 46.0 | 58.4 |
| Agent-PVT-L | 492G | 46.8 | 68.2 | 50.7 | 30.9 | 50.8 | 62.9 |

表16. 在COCO数据集上使用RetinaNet [23]的检测结果。FLOPs是在输入分辨率为 的骨干网络、FPN和检测头上计算的。

| Semantic Segmentation on ADE20K | | | | | |
| --- | --- | --- | --- | --- | --- |
| Backbone | Method | FLOPs | #Params | mIoU | mAcc |
| PVT-T | S-FPN | 158G | 17M | 36.57 | 46.72 |
| Agent-PVT-T | S-FPN | 147G | 15M | 40.18 | 51.76 |
| PVT-S | S-FPN | 225G | 28M | 41.95 | 53.02 |
| Agent-PVT-S | S-FPN | 211G | 24M | 44.18 | 56.17 |
| PVT-M | S-FPN | 315G | 48M | 42.91 | 53.80 |
| Agent-PVT-M | S-FPN | 321G | 40M | 44.30 | 56.42 |
| PVT-L | S-FPN | 420G | 65M | 43.49 | 54.62 |
| Agent-PVT-L | S-FPN | 434G | 52M | 46.52 | 58.50 |
| Swin-T | UperNet | 945G | 60M | 44.51 | 55.61 |
| Agent-Swin-T | UperNet | 954G | 61M | 46.68 | 58.53 |
| Swin-S | UperNet | 1038G | 81M | 47.64 | 58.78 |
| Agent-Swin-S | UperNet | 1043G | 81M | 48.08 | 59.78 |
| Swin-B | UperNet | 1188G | 121M | 48.13 | 59.13 |
| Agent-Swin-B | UperNet | 1196G | 121M | 48.73 | 60.01 |

表17. 语义分割的结果。FLOPs是在输入分辨率为 的编码器和解码器上计算的。S-FPN是SemanticFPN [20]模型的简称。

# C.3. ADE20K

训练设置。ADE20K [49] 是一个公认的语义分割基准，包含 个训练图像和 个验证图像。骨干网络在ImageNet数据集上预训练，使用AdamW优化，遵循原论文中提到的训练配置。对于UperNet [42]，我们使用AdamW进行优化，并将初始学习率设置为 ，进行1,500次迭代的线性预热。模型总共训练 次迭代。对于Semantic FPN [20]，我们使用AdamW对模型进行 次迭代的优化，初始学习率为 。训练时，我们随机调整图像大小并裁剪到 ，测试时将短边缩放到512像素。

代理注意力分布

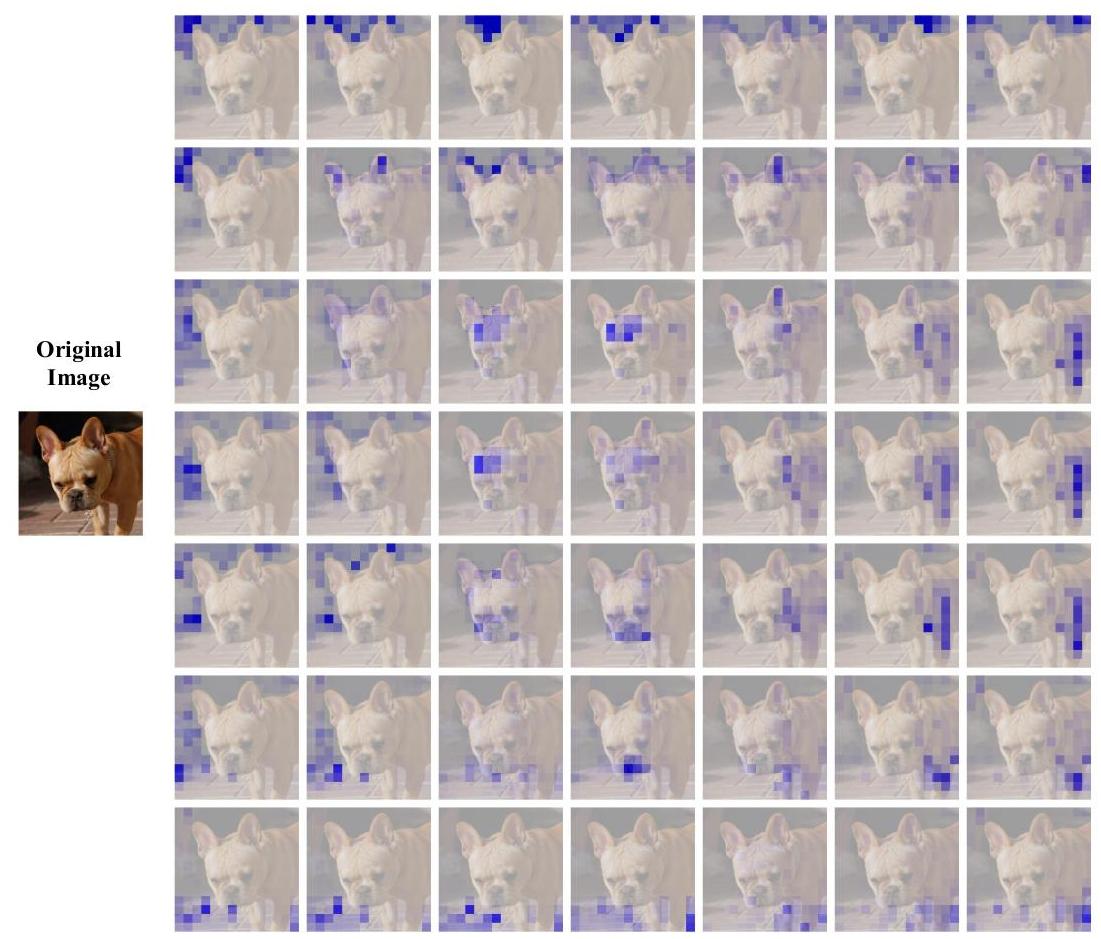


图9. 与Agent-Swin-T的第三块中49个代理标记相对应的注意力权重分布。

# D. 完整的实验结果

完整的分类结果。我们在表14中提供了完整的ImageNet- 分类结果。很明显，在各种模型中用我们的代理注意力替换Softmax注意力会导致性能的一致提升。

额外的下游实验。我们在表16、表15和表17中提供了目标检测和语义分割的额外实验结果。对于目标检测，我们展示了在RetinaNet [23]、Mask R-CNN [17] 和 Cascade Mask R-CNN [4] 框架上的结果，而对于语义分割，我们展示了 Se-manticFPN [20] 和 UperNet [42] 的结果。可以看出，我们的模型在各种设置下都对其基线对比模型实现了持续改进。

# E. 代理注意力可视化

我们在图9中可视化了代理注意力的分布。可以看出，不同的代理标记专注于不同的区域，例如耳朵（第二行第二个）和鼻子/嘴巴（第六行第四个）。这种多样性确保了在代理广播过程中，不同的查询可以专注于它们的兴趣区域。

# F. 模型架构

我们展示了本文中使用到的四个Transformer模型的架构，包括 Agent-DeiT、Agent-PVT、Agent-Swin 和 Agent-CSwin，在表18-22中。考虑到扩大感受野的优势，我们主要在视觉Transformer模型的早期阶段将Softmax注意力块替换为我们的代理注意力模块。

| stage | output | Agent-DeiT-T | | Agent-DeiT-S | | Agent-DeiT-B | |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Agent | DeiT Block | Agent | DeiT Block | Agent | DeiT Block |
| res1 |  | win dim 192 head 3 | None | win dim 384 head 6 | None |  |  |

| stage | output | Agent-PVT-T | | | | Agent-PVT-S | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | Agent |  | PVT Block | Agent |  | PVT Block |
| res1 |  | Conv1×1, stride=4, 64, LN | | | | | | |
|  |  |  | None | win dim 64 head 1 agent 9 |  | None |
| res2 |  | Conv1×1, stride=2, 128, LN | | | | | | |
|  |  |  | None |  |  | None |
| res3 |  | Conv1×1, stride=2, 320, LN | | | | | | |
|  |  |  | None |  |  | None |
| res4 |  | Conv1×1, stride=2, 512, LN | | | | | | |
|  |  |  | None |  |  | None |

表19。Agent-PVT模型架构（第一部分）。

| stage | output | Agent-PVT-M | | | | Agent-PVT-L | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | Agent |  | PVT Block | Agent |  | PVT Block |
| res1 |  | Conv1×1, stride=4, 64, LN | | | | | | |
|  |  |  | None |  |  | None |
| res2 |  | Conv1×1, stride=2, 128, LN | | | | | | |
|  |  |  | None |  |  | None |
| res3 |  | Conv1×1, stride=2, 320, LN | | | | | | |
|  |  |  | None |  |  | None |
| res4 |  | Conv1×1, stride=2, 512, LN | | | | | | |
|  |  |  | None |  |  | None |

| stage | output | Agent-Swin-T | | | Agent-Swin-S | | | Agent-Swin-B | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Agent |  | Swin Block | Agent |  | Swin Block | Agent |  | Swin Block |
| res1 |  | concat , , | | | concat , , | | | concat , , LN | | |
| win dim 96 head 3 agent 9 |  | None | win dim 96 head 3 agent 9 |  | None | win dim 128 head 3 agent 9 |  | None |
| res2 |  | concat , , | | | concat | | | concat | | |
| win dim 192 head 6 agent 16 |  | None | win |  | None | win dim 256 head 6 agent 16 |  | None |
| res3 |  | concat , 384, LN | | | concat | | | concat | | |
| None | |  | None |  |  | win dim 512 head 12 |  |  |
| res4 |  | concat | | | concat | | | concat , , | | |
| None |  |  | None |  |  | None | | win dim 1024 head 24 |

| stage | output | Agent-CSwin-T | | | | Agent-CSwin-S | | | | Agent-CSwin-B | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Agent |  | CSwin Block |  | Agent |  | CSwin Block |  | Agent |  | CSwin Block |  |
| res1 |  | Conv7×7, stride=4, 64, LN | | | | | | | | Conv7x7, stride=4, 96, LN | | | |
|  |  | None |  | win dim 64 head 2 agent 9 |  | None |  | win dim 96 head 4 agent 9 |  | None |  |
| res2 |  | Conv7×7, stride=4, 128, LN | | | | | | | | Conv7×7, stride=4, 192, LN | | | |
| win dim 128 head 4 agent 16 |  | None | | win dim 128 head 4 agent 16 |  | None | | win dim 192 head 8 agent 16 |  | None | |
| res3 |  | Conv7x7, stride=4, 256, LN | | | | | | | | Conv7×7, stride=384, LN | | | |
| None |  | win dim 256 head 8 |  | None |  | win dim 256 head 8 |  | None | | win dim 384 head 16 |  |
| res4 |  | Conv7×7, stride=4, 512, LN | | | | | | | | Conv7×7, stride=4, 768, LN | | | |
| None |  | win dim 512 head 16 |  | None |  | win dim 512 head 16 |  | None |  | win dim 768 head 32 |  |

表22。Agent-CSwin模型架构。