Stratified Transformer for 3D Point Cloud Segmentation

Xin Lai Jianhui Liu Li Jiang Liwei Wang Hengshuang Zhao Shu Liu Xiaojuan Qi Jiaya Jia

摘要

近年来，3D 点云分割取得了巨大的进步。当前的大多数方法都专注于聚合本地特征，但无法直接对远程依赖项进行建模。在本文中，论文提出了能够捕获远程上下文并展示出强大的泛化能力和高性能的 Stratified Transformer。具体来说，论文首先提出了一种新颖的关键采样策略。对于每个查询点，论文以分层的方式密集地采样附近的点和稀疏的远程点作为其键，这使模型能够扩大有效感受野并以较低的计算成本享受远程上下文。此外，为了应对不规则点排列带来的挑战，论文提出了第一层点嵌入来聚合局部信息，这有助于收敛并提高性能。此外，论文采用上下文相对位置编码来自适应地捕获位置信息。最后，引入了一种内存高效的实现来克服每个窗口中不同点数的问题。大量实验证明了论文的方法在 S3DIS、ScanNetv2 和 ShapeNetPart 数据集上的有效性和优越性。

1 引言

3D 点云数据多可以方便收集，点云数据结构特殊，为点的数据集合，其排列不规则且大多无序，诸如自动驾驶、机器人、增强现实等等应用所应用的点云数据的需要更为高效可行的处理。

Transformer[1]，起初应用于序列的上下文建模处理，其捕捉位置信息的特性使得它呗广泛应用于广大领域，Transformer 系列模型特别适用于点云处理，因为作为 Transformer 网络核心的自注意算子本质上是一个集合算子：它对输入元素的排列和基数是不变的。因此，自注意力对 3D 点云的应用非常自然，因为点云本质上是嵌入在 3D 空间中的集合。

论文工作回答了两个问题。首先，可以以低计算成本建立直接的远程依赖关系并产生更高的性能。其次，标准的 Transformer 可以应用于 3D 点云，泛化能力强，性能强大；

2 相关工作

2.1 Transformer

最近，视觉转换器在二维图像理解中变得流行。ViT[2]将每个补丁视为一个令牌，并直接使用 Transformer 编码器提取特征进行图像分类。此外，PVT[3]提出了一种层次结构来获得用于语义分割的特征金字塔，并且还提出了 Spatial Reduction Attention 以节省内存。或者，Swin Transformer[4] 使用基于窗口的注意力，并在连续的 Transformer 块中提出移位窗口操作。Transformer 在 2D 中已经很流行，但在点云上仍未得到充分探索。受 Swin Transformer 的启发，论文对 3D 点云采用分层结构和移位窗口操作。最重要的是，提出了一种分层策略来采样键以获取远程上下文，并提出了几个基本设计来应对不规则点排列带来的挑战。

2.2 3D 点云分割

点云分割的方法可以分为两类，即基于体素的方法和基于点的方法。基于体素的解决方案[5] 首先将 3D 空间划分为规则体素，然后对其应用稀疏卷积。它们产生了不错的性能，但由于体素化而遭受不准确的位置信息。基于点的方法直接采用点特征和位置作为输入，从而保持位置信息完整。按照这一研究方向，设计了不同的特征聚合方法来学习高级语义特征。PointNet 及其变体[6-7] 使用最大池化来聚

合特征。PointConv[8] 和 KPConv[9] 尝试使用 MLP 或离散核点来模拟连续卷积核。Point Transformer[10] 使用“向量自注意力”算子来聚合局部特征，并使用“减法关系”来生成注意力权重，但它缺乏远程上下文，并且在测试中对各种扰动的鲁棒性不足。

本文工作是基于点且密切相关的 Point Transfomer，但有一个根本区别：本文工作克服了有限的有效感受野问题，并充分利用 Transformer 来建模远程上下文依赖关系，而不仅仅是局部聚合。

3 本文方法

3.1 本文方法概述

论文模型概述如图 1所示。论文的框架是基于点的，论文使用 xyz 坐标和 rgb 颜色作为输入。采用编码器-解码器结构，其中编码器由通过下采样层连接的多个阶段组成。在编码器的开始，第一层点嵌入模块用于局部聚合。然后，每个阶段都有几个 Transformer 块。至于解码器，编码器的特征被上采样，以类似于 U-Net 的方式逐层变得更密集。

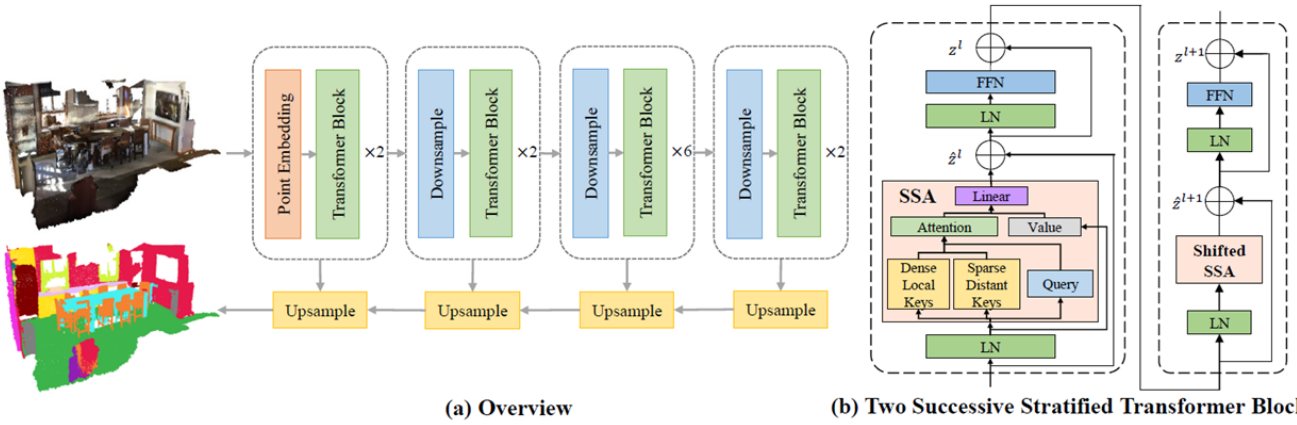

图 1: (a) 框架概述。(b) 分层变压器块的结构。采用层次结构来获得多级特征。输入点云首先经过 Point Embedding 模块聚合局部结构信息。在几个下采样层和转换器块之后，对特征进行上采样以进行分割。SSA：分层自注意力。移位 SSA：具有移位窗口的 SSA。

3.2 Transformer 模块

Transformer 模块由标准的多头自注意力模块和前馈网络 (FFN) 组成。以数万个点作为输入，直接应用全局自注意力会导致不可接受的 $O(N^2)$ 内存消耗，其中 N 是输入点数。

3.2.1 基础版本

采用基于窗口的自我注意。3D 空间首先被划分为不重叠的立方窗口，其中的点分散在不同的窗口中。不像全局自注意力那样关注所有点，每个查询点只需要考虑同一窗口中的邻居。多头自注意力在每个窗口中独立执行。由于不同的窗口可能包含不同数量的点，论文将 k_t 表示为第 t 个窗口内的点数。形式上，给定 N_h 是 head 的数量，N_d 是每个 head 的维度，$N_c = N_h \times N_d$ 是特征维度，对于第 t

个窗口中的输入点 $x \in \mathbb{R}^{k_t \times (N_h \times N_d)}$，第 t 个窗口中的多头自注意力可表示为

$$
\begin{aligned}
\mathbf{q} = \text{Linear}_q(\mathbf{x}), \quad \mathbf{k} = \text{Linear}_k(\mathbf{x}), \quad \mathbf{v} = \text{Linear}_v(\mathbf{x}), \\
\boldsymbol{attn}_{i,j,h} &= \mathbf{q}_{i,h} \cdot \mathbf{k}_{j,h}, \\
\text{attn}_{i,j,h} &= \text{softmax}\left(\text{attn}_{i,j,h}\right), \\
\mathbf{y}_{i,h} &= \sum_{j=1}^{k_t} \text{attn}_{i,j,h} \times \mathbf{v}_{j,h}, \\
\hat{\mathbf{z}} &= \text{Linear}(\mathbf{y}),
\end{aligned}
\tag{1}
$$

其中 $q, k, v \in \mathbb{R}^{k_t \times N_h \times N_d}$ 由 x 经过三个线性层得到，$\cdot$ 表示向量 $q_{i,h}$ 和 $k_{j,h}$ 之间的点积。$attn \in \mathbb{R}^{k_t \times N_h \times N_d}$ 是注意力图，$y \in \mathbb{R}^{k_t \times N_h \times N_d}$ 是聚合特征，进一步投影到输出特征 $\hat{z} \in \mathbb{R}^{k_t \times (N_h \times N_d)}$。请注意，上述等式仅显示了单个窗口中的计算，不同的窗口以相同的方式独立工作。通过这种方式，内存复杂度显着降低到 $O(\frac{N}{k} \times k^2) = O(N \times k)$，其中 k 是分散在每个窗口中的平均点数。为了促进跨窗口通信，论文还在两个连续的 Transformer 块之间将窗口大小移动了一半，类似于[4]。移动窗口的示意图如图2所示

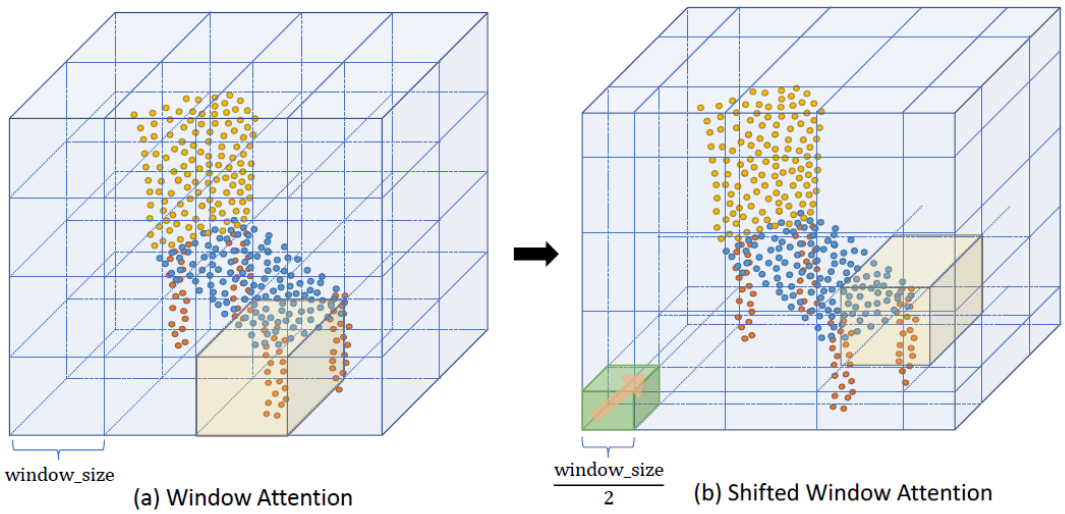

图 2: 3D 中移动窗口操作的图示。(a) 执行窗口注意力机制。(b) 窗口在连续的 Transformer 块中移动。

3.2.2 分层键采样的优化版本

由于每个查询点只关注其自己窗口中的局部点，因此即使窗口移位，基础版本的 Transformer 块也会受到有限的有效感受野的影响，因此它无法捕获远距离的上下文依赖关系对象，导致错误的预测。

一个简单的解决方案是扩大立方窗口的大小。但是，内存会随着窗口大小的增加而增加。为了以较低的内存成本有效地聚合远程上下文，论文提出了一种用于采样键的分层策略，如图 3 所示

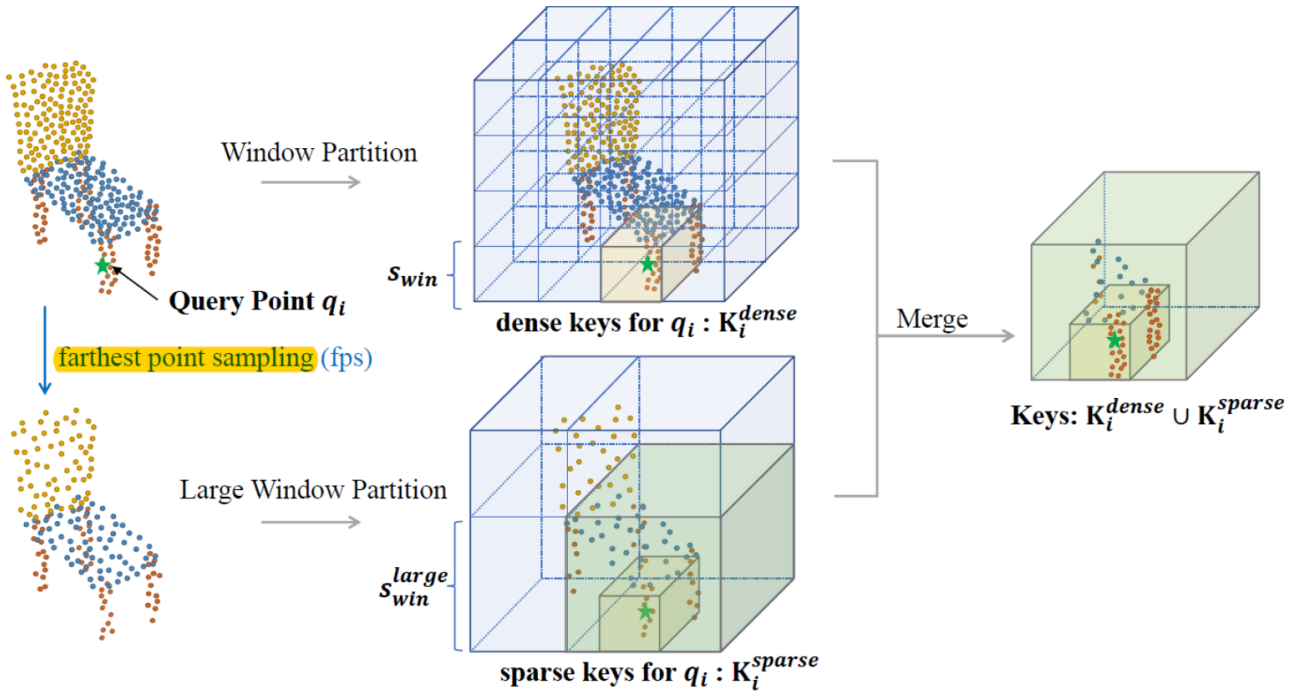

图 3: 键采样的分层策略示意图。绿色星号表示给定的查询点。

论文在其窗口中找到点 $K^{dense}i$，与基础版本相同。此外，论文在 s 的尺度上通过最远点采样 (fps) 对输入点进行下采样，并找到具有较大窗口大小 s_{win}^{large} 的点 K_i^{sparse}。最后，密集 key 和稀疏 key 都形成了最终 key，即 $K_i = K_i^{dense} \square K_i^{sparse}$ 。请注意，重复的关键点只计算一次。

Transformer 块的完整结构如图 1（b）所示。按照惯例，论文在每个自注意力模块或前馈网络之前使用 LayerNorm[11]。为了进一步补充跨窗口的信息交互，在后续的 Transformer 块中，原始窗口移动了 $\frac{1}{2}S_{win}$，而大窗口移动了 $\frac{1}{2}s_{win}^{large}$。

由于键抽样的分层策略，有效感受野显着扩大，查询特征能够有效地聚合远程上下文。与原版相比，论文只是在稀疏的远距离键上产生了额外的计算，仅占最终键 K_i 的 10 % 左右。

3.3 第一层点嵌入

在第一层，论文构建了一个点嵌入模块。一个直观的选择是使用线性层或 MLP 将输入特征投影到高维。然而，通过在第一层中使用线性层，论文凭经验观察到收敛速度相对较慢且性能较差，如图 4 所示。注意到来自线性层或 MLP 的点特征仅包含其自身 xyz 位置的原始信息和 rgb 颜色，但它缺乏局部几何和上下文信息。结果，在第一个 Transformer 块中，注意力图无法捕获 query 和仅包含原始 xyz 和 rgb 信息的 key 之间的高级相关性。这会对模型的表示能力和泛化能力产生负面影响。相反，论文建议在点嵌入模块中聚合每个点的局部邻居的特征。

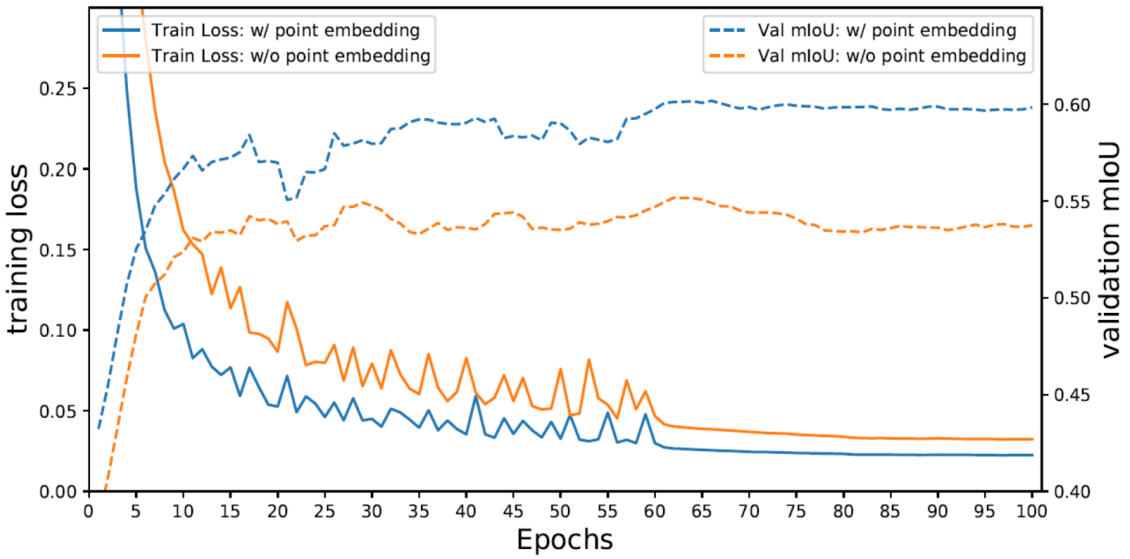

图 4: 训练过程中的训练 loss（实线）和验证 mIoU（虚线）图。比较了有（蓝色曲线）和没有（橙色曲线）第一层点嵌入的模型。

尝试了各种局部聚合方法，例如最大池化和平均池化，发现 KPConv 表现最好。令人惊讶的是，对架构的这种微小修改带来了相当大的改进。它证明了初始本地聚合在基于 Transformer 的网络中的重要性。值得注意的是，与整个网络相比，单个 KPConv 产生的额外计算可以忽略不计（仅 2% 的 FLOPs）

3.4 上下文相对位置编码

与 2D 空间规则像素相比，3D 点处于更复杂的连续空间中，对利用 xyz 位置提出了挑战。[12] 声称位置编码对于基于 3D Transformer 的网络来说是不必要的，因为 xyz 坐标已经被用作输入特征。然而，虽然 Transformer 块的输入已经包含 xyz 位置，但是当深入网络时，细粒度的位置信息可能会丢失在高级特征中。为了更好地利用位置信息，论文采用受[13] 启发的基于上下文的自适应相对位置编码方案。

特别是，对于第 t 个窗口中的点特征 $x \in \mathbb{R}^{k_t \times (N_h \times N_d)}$，论文将 xyz 坐标表示为 $\mathbf{p} \in \mathbb{R}^{k_t \times 3}$。因此，查询和键之间的相对 xyz 坐标 $\mathbf{r} \in \mathbb{R}^{k_t \times k_h \times 3}$ 表示为

$$\mathbf{r}_{i,j,m} = \mathbf{p}_{i,m} - \mathbf{p}_{j,m}, \quad 1 \leqslant i, j \leqslant k_t, m \in \{1, 2, 3\}. \tag{2}$$

为了将相对坐标映射到相应的位置编码，论文中维护了三个可学习的查找表 $\mathbf{t}_x, \mathbf{t}_y, \mathbf{t}_z \in \mathbb{R}^{L \times (N_h \times N_d)}$，分别对应于 x、y 和 z 轴。由于相对坐标是连续的浮点数，论文将 $r_{i,j,m}$ 的范围，即 $(-s_{win}, s_{win})$ 统一量化为 L 个离散部分，并将相对坐标 $r_{i,j,m}$ 映射到表：

$$\mathbf{idx}_{i,j,m} = \left\lfloor \frac{\mathbf{r}_{i,j,m} + s_{win}}{s_{quant}} \right\rfloor, \tag{3}$$

其中 s_{win} 是窗口大小，$s_{quant} = \frac{2 \cdot s_{win}}{L}$ 是量化大小，$\lfloor \cdot \rfloor$ 表示底边舍入。

论文查找表以检索与索引对应的嵌入并将它们相加得到位置编码:

$$\mathbf{e}_{i,j} = \mathbf{t}_x \left[\mathbf{idx}_{i,j,1}\right] + \mathbf{t}_y \left[\mathbf{idx}_{i,j,2}\right] + \mathbf{t}_z \left[\mathbf{idx}_{i,j,3}\right], \tag{4}$$

其中 $\mathbf{t}\left[idx\right] \in \mathbb{R}^{N_h \times N_d}$ 表示表 $\mathbf{t}$ 的第 idx 项，$\mathbf{e} \in \mathbb{R}^{k_t \times k_t \times N_h \times N_d}$ 是位置编码。

实际上，query、key、value 的表是不共享的。所以论文通过添加上标来区分它们，其中 $\mathbf{t}_x^q$ 表示查询的 x 轴表。同理，query、key、value 对应的位置编码分别用 e^q, e^k, e^v 表示

然后位置编码与查询和关键特征进行点积以获得位置偏差 $\mathbf{pos_bias} \in \mathbb{R}^{k_t \times k_t \times N_h}$，然后将其添加到

注意力图中。此外，论文添加了具有相应位置编码的值特征，然后是加权和聚合。最后，基础版本方程 (1) 更新为上下文相对位置编码 (cRPE) 版本:

$$
\begin{aligned}
\mathbf{pos\_bias}_{\mathrm{i,j,h}}^{\mathrm{cRPE}} &= \mathbf{q}_{i,h} \cdot \mathbf{e}_{i,j,h}^{q} + \mathbf{k}_{j,h} \cdot \mathbf{e}_{i,j,h}^{k}, \\
\mathrm{attn}_{i,j,h}^{cRPE} &= \mathbf{q}_{i,h} \cdot \mathbf{k}_{j,h} + \mathbf{pos\_bias}_{i,j,h}^{cRP}, \\
\hat{\mathbf{attn}}_{i,.,h}^{cRPE} &= \mathrm{softmax}\left(\mathbf{attn}_{i,j,h}^{cRPE}\right), \\
\mathbf{y}_{i,h}^{cRPE} &= \sum_{j=1}^{k_t} \hat{\mathbf{attn}}_{i,j,h}^{cRPE} \times \left(\mathbf{v}_{j,h} + \mathbf{e}_{i,j,h}^{v}\right).
\end{aligned}
\tag{5}
$$

与基于 MLP 的位置编码相比，相对 xyz 坐标 $\mathbf{r} \in \mathbb{R}^{k_t \times k_t \times 3}$ 通过 MLP 直接投影到位置偏差 $\mathbf{pe\_bias} \in \mathbb{R}^{k_t \times k_t \times N_h}$，cRPE 通过点积自适应地生成位置偏差带有 query 和 key，从而提供语义信息。基于 MLP 和 cRPE 的位置偏差在图中可视化。它揭示了一个事实，即基于 MLP 的模型生成的位置偏差在键之间是相似的。因此，注意力权重几乎没有区别。但是对于 cRPE，位置偏差对于不同的键有很大的不同。

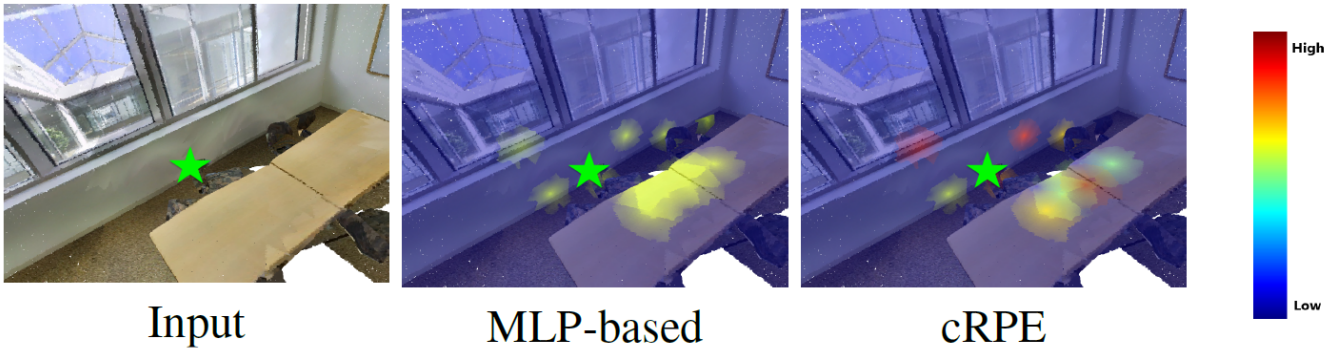

图 5: 给定查询点（以绿色星号显示）的最后一个 Transformer Block 的第一个头部的每个键的位置偏差的可视化。彩色地图显示在右侧。

上下文相对位置编码的总览如图 6 所示

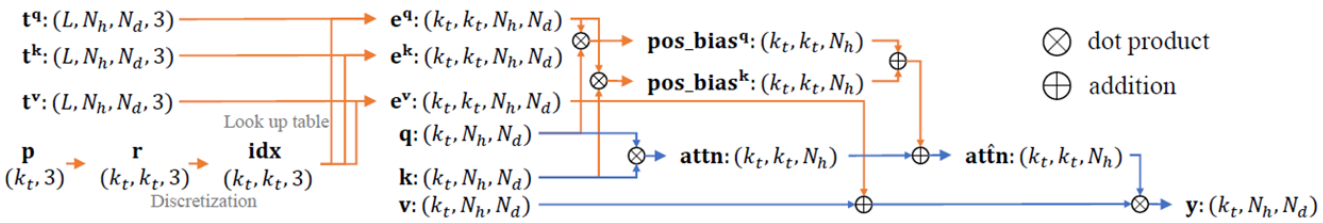

图 6: 上下文相对位置编码的总览。

3.5 下采样和上采样层

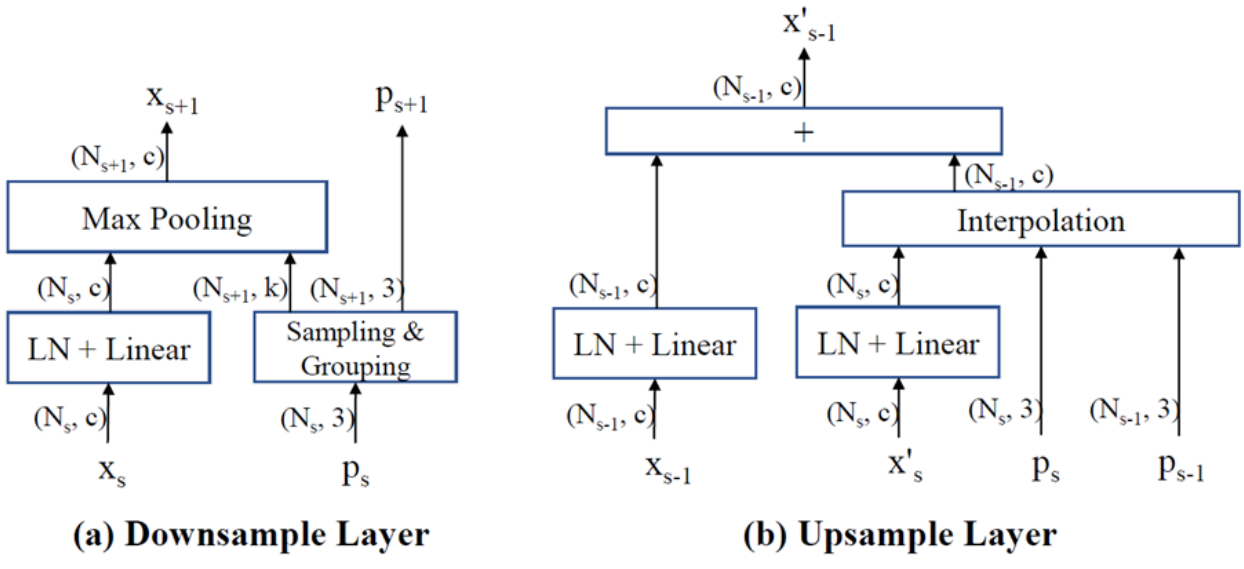

图 7: (a) 下采样层和 (b) 上采样层的结构图。

下采样层如图 7(a) 所示。首先，xyz 坐标 $\mathbf{p}_s$ 通过 Sampling & Grouping 模块，论文首先通过最远点采样 (fps)[7] 对质心点 $\mathbf{p}_{s+1}$ 进行采样，然后使用 kNN 查询原始点得到分组索引 $\mathbf{idx}_{\mathbf{group}} \in \mathbb{R}^{N_{s+1} \times k}$。质心点数为原点数的 $\frac{1}{4}$，即 $N_{s+1} = \left\lceil \frac{1}{4} N_s \right\rceil$。同时，点特征 $\mathbf{x}_s$ 被输入到 Pre-LN[14] 线性投影层。此外，论文利用最大池化来使用分组索引聚合投影特征，产生输出特征 $\mathbf{x}_{s+1}$。

对于上采样层，如图 7（b）所示，解码器特征 $\mathbf{x}'_s$ 首先由 Pre-LN 线性层投影。论文在当前 xyz 坐标 $\mathbf{p}_s$ 和之前的坐标 $\mathbf{p}_{s-1}$ 之间执行插值[7]。前一阶段中的编码器点特征 $\mathbf{x}_{s-1}$ 经过一个 Pre-LN 线性层。最后，论文将它们相加得到下一个解码器特征 $\mathbf{x}'_{s-1}$。

3.6 内存高效的实现

在 2D Swin Transformer 中，很容易实现基于窗口的注意力，因为每个窗口中令牌的数量是固定的。然而，由于 3D 中的点排列不规则，每个窗口中的令牌数量变化很大。一个简单的解决方案是使用虚拟令牌将每个窗口中的令牌填充到最大令牌数 kmax，然后应用掩码自注意。但是这种解决方案浪费了大量的内存和计算。

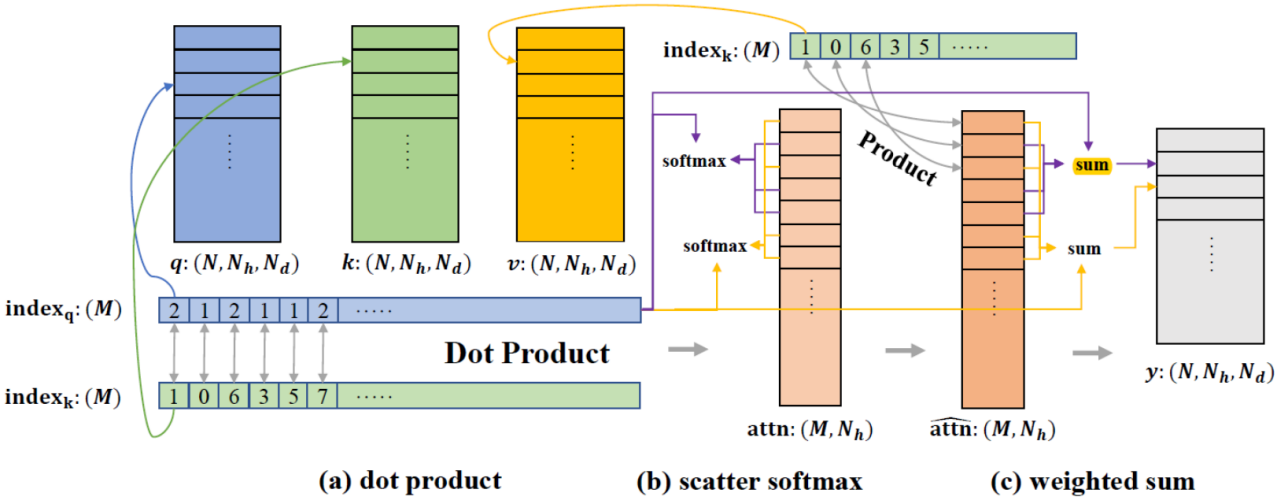

图 8: 内存高效实现包括三个步骤：(a) 点积；(b) 分散 softmax；(c) 加权和。

相反，论文首先预先计算所有需要执行点积的查询和键对。如图 8(a) 所示，论文使用两个索引

index$_q, index$**index**$_k \in \mathbb{R}^M$ 分别对形状为 (N, N_h, N_d) 的 **q** 和 **k** 进行索引，其中 N 表示输入点的总数。然后，论文在 **index**$_q$ 和 **index**$_k$ 索引的条目之间执行点积，产生形状 (M, N_h) 的注意力图 **attn**。之后，如图 8 (b) 所示，论文使用查询索引 **q** 直接在 **attn** 上执行 scatter softmax，其中 softmax 函数应用于 **attn** 中与 **index**$_q$ 中具有相同索引的条目。此外，如图 8 (c) 所示，论文使用 **index**$_k$ 来索引值 **v** 并将它们与注意力图 **a** 相乘。论文最终将 indexq 中具有相同索引的条目求和，并将结果保存到输出特征 **y** 中。请注意，每个步骤都由单个 CUDA 内核实现。所以每一步里面的中间变量几乎不会占用内存。通过这种方式，论文达到了 $O(M{\cdot}Nh)$ 的内存复杂度，远低于基础版本实现中使用的内存复杂度。论文中补充材料中提及与普通版本相比，论文的实现节省了 57% 的内存。

4 复现细节

4.1 与已有开源代码对比

论文工作将上述工作流程执行得很完美，对照作者提供部分源代码进行学习复现。主要的复现工作为将原始处理数据进行预备整理。其次为在不同实验环境设置条件下进行模型的训练和调试，由于模型的较高复杂度和数据本身的大体积和高容量以及实验环境的限制，每次训练时间将近 4-5 天，所以实验过程也是相对较少，实验结果仅部分达成，复现过程实现并不是很完美，重点过程在于论文中闪光点的学习吸收和项目中已有代码的关于闪光点的思路借鉴。

4.2 实验数据准备

我们使用 S3DIS[15] 和 ScanNetv2[16] 数据集进行语义分割任务。他们都在挑战大规模的室内场景数据集。S3DIS 数据集包括来自三座建筑物的 6 个区域的 271 个房间，并标注了 13 个常见类别。按照惯例，我们使用 Area_5 中的场景进行测试，其他场景进行训练。ScanNetv2 数据集包含 1201、312 和 100 个室内 RGB-D 场景，分别用于训练、验证和测试。并标注了 20 个类别内的语义标签。

4.3 实验环境搭建

按照图 1中架构，xyz 坐标和 rgb 颜色都用作输入。我们将初始特征维度和头数分别设置为 48 和 3，并且它们在每个下采样层中都会翻倍。对于 S3DIS，构建了四个阶段，块深度为 [2, 2, 6, 2]。相比之下，对于 ScanNetv2，注意到点数更大。因此在第一层点嵌入模块的顶部添加了一个额外的下采样层。然后，添加块深度为 [3, 9, 3, 3] 的后面四个阶段。所以一共为 ScanNetv2 构建了五个阶段。

对于 S3DIS，按照惯例，首先对原始输入点进行网格采样，网格大小设置为 0.04m。在训练过程中，最大输入点数设置为 80000，如果点数达到此数量，则丢弃所有多余的点。窗口大小最初设置为 0.16m，在每个下采样层之后它会加倍。分层采样策略的下采样比例设置为 8。使用 z 轴旋转、比例、抖动和下降颜色作为数据增强。对于 ScanNetv2，权重衰减和批量大小分别设置为 0.1 和 8，网格采样的网格大小设置为 0.02m。在训练期间，最多将 120000 个点云点输入网络。初始窗口大小设置为 0.1m。分层采样的下采样比例设置为 4。除随机抖动外，数据增强与 S3DIS 上的数据增强相同。

4.4 创新点

1、分层 Transformer：分层策略能够扩大有效感受野并提高性能。

2、第一层点嵌入：证明了初始本地聚合在基于 Transformer 的网络中的重要性。

3、上下文相对位置编码：纯粹基于 xyz 坐标的相对位置信息是没有帮助的，因为网络的输入点

特征已经包含了 xyz 坐标。相比之下，cRPE 同时基于 xyz 坐标和上下文特征，因而特征提取效果更佳。

4、移位窗口机制：采用移位窗口来补充跨窗口的信息交互。

5、数据增强：在训练基于 Transformer 的网络中数据增强起着重要作用，使得训练的样本数量和质量都有提升。

5 实验结果分析

对于 S3DIS，训练和测试结果如 9所示；对于 ScanNetv2，训练和测试结果如 10所示。由于训练时长的限制以及部分模型组件的完成程度不高，模型对于这两个大体积数据集并没有很好地学习，在测试阶段的分类结果虽然大多比训练时结束时验证结果稍好，但是对比其他诸如 PointTransformer[10]、MinkowskiNet[17]的 mAcc、mIoU 参数指标仍然不够格，一方面是时长不够，另一方面是模型的参数并没有调整到最优，应当再结合论文中设置进行多次跑实验和调参数。

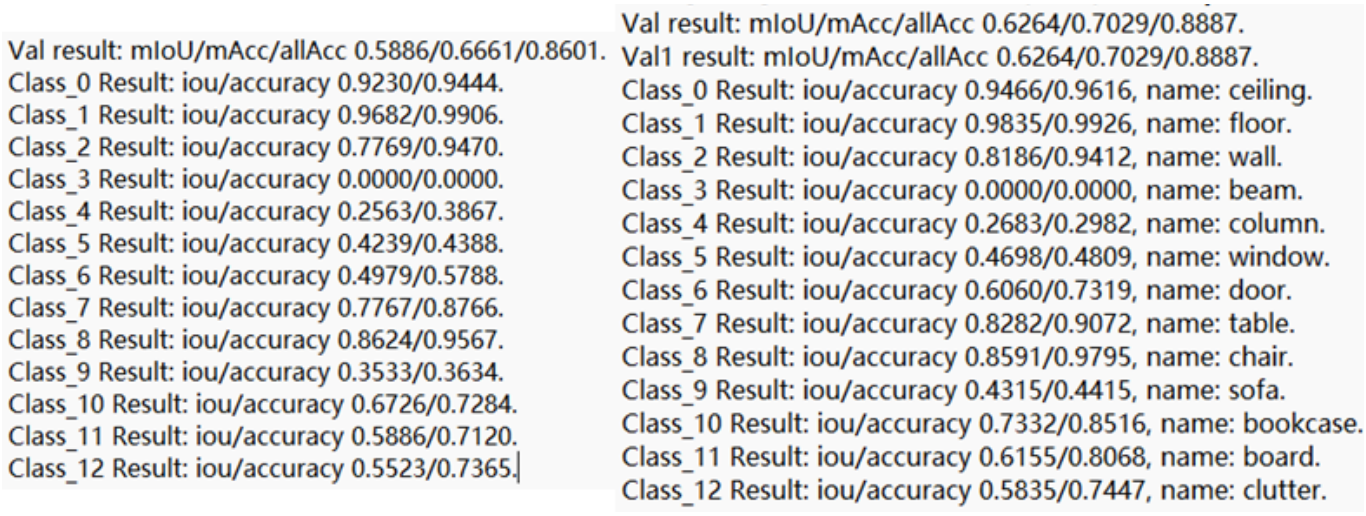

图 9: 在 S3DIS 上的训练和测试结果

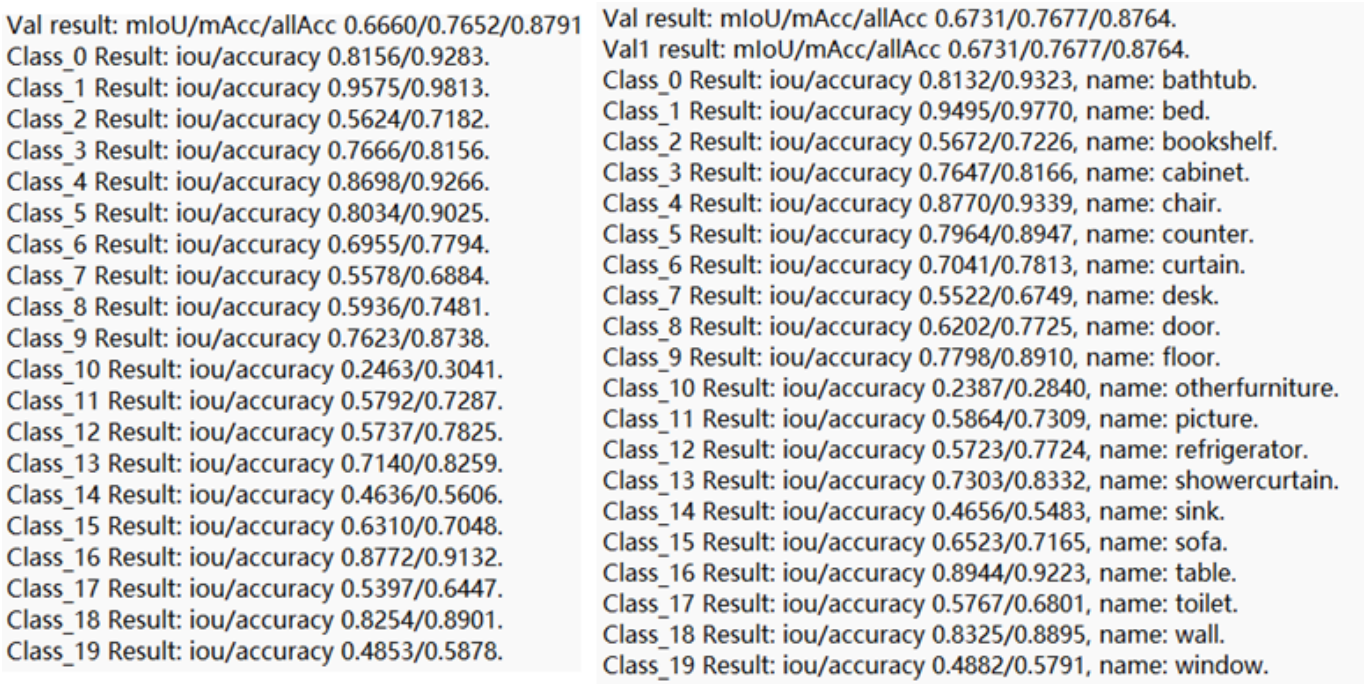

图 10: 在 ScanNetv2 上的训练和测试结果

6 总结与展望

6.1 局限性分析

所提出的方法能够在大规模数据集中产生强大的性能，但对于小型数据集可能不是最佳解决方案，因为在基于 Transformer 的网络中引入了较少的归纳偏差；在训练期间，模型没有应用更高级的数据增强，这可能会限制性能；由于缺乏使用共享内存等高级 CUDA 功能，与以前的方法相比，训练速度相对较慢。

6.2 未来的工作

将框架扩展到许多其他 3D 点云任务，3D 对象检测，实例分割；探索用于训练的高级数据增强，并将其融合进模型当中；采用更高级的 CUDA 功能或方法来降低高内存吞吐量。

参考文献

[1] VASWANI A, SHAZEER N, PARMAR N, et al. Attention is all you need[J]. Advances in neural information processing systems, 2017, 30.

[2] DOSOVITSKIY A, BEYER L, KOLESNIKOV A, et al. An image is worth 16x16 words: Transformers for image recognition at scale[J]. arXiv preprint arXiv:2010.11929, 2020.

[3] 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. 2021: 568-578.

[4] 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. 2021: 10012-10022.

[5] GRAHAM B, ENGELCKE M, VAN DER MAATEN L. 3d semantic segmentation with submanifold sparse convolutional networks[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2018: 9224-9232.

[6] QI C R, SU H, MO K, et al. Pointnet: Deep learning on point sets for 3d classification and segmentation [C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2017: 652-660.

[7] QI C R, YI L, SU H, et al. Pointnet++: Deep hierarchical feature learning on point sets in a metric space [J]. Advances in neural information processing systems, 2017, 30.

[8] WU W, QI Z, FUXIN L. Pointconv: Deep convolutional networks on 3d point clouds[C]//Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition. 2019: 9621-9630.

[9] THOMAS H, QI C R, DESCHAUD J E, et al. Kpconv: Flexible and deformable convolution for point clouds[C]//Proceedings of the IEEE/CVF international conference on computer vision. 2019: 6411-6420.

[10] ZHAO H, JIANG L, JIA J, et al. Point transformer[C]//Proceedings of the IEEE/CVF international conference on computer vision. 2021: 16259-16268.

[11] BA J L, KIROS J R, HINTON G E. Layer normalization[J]. arXiv preprint arXiv:1607.06450, 2016.

[12] MISRA I, GIRDHAR R, JOULIN A. An end-to-end transformer model for 3d object detection[C]//Proceedings of the IEEE/CVF International Conference on Computer Vision. 2021: 2906-2917.

[13] WU K, PENG H, CHEN M, et al. Rethinking and improving relative position encoding for vision transformer[C]//Proceedings of the IEEE/CVF International Conference on Computer Vision. 2021: 10033-10041.

[14] XIONG R, YANG Y, HE D, et al. On layer normalization in the transformer architecture[C]//International Conference on Machine Learning. 2020: 10524-10533.

[15] ARMENI I, SENER O, ZAMIR A R, et al. 3D Semantic Parsing of Large-Scale Indoor Spaces[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2016.

[16] DAI A, CHANG A X, SAVVA M, et al. Scannet: Richly-annotated 3d reconstructions of indoor scenes [C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2017: 5828-5839.

[17] CHOY C, GWAK J, SAVARESE S. 4d spatio-temporal convnets: Minkowski convolutional neural networks[C]//Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2019: 3075-3084.