EFFICIENT POINT TRANSFORMER FOR LARGE-SCALE 3D SCENE UNDERSTANDING

Anonymous authors
Paper under double-blind review

ABSTRACT

The recent success of neural networks has enabled a better interpretation of 3D point clouds, but processing a large-scale 3D scene remains a challenging problem. Most approaches divide a large-scale 3D scene into multiple regions and combine the local predictions, but this inevitably increases inference time and involves preprocessing stages, such as k-nearest neighbor search. An alternative is to quantize the point cloud to voxels and process them with sparse convolution. Although sparse convolution is efficient and scalable for large 3D scenes, the quantization artifacts impair geometric details and degrade prediction accuracy. This paper proposes an *Efficient Point Transformer (EPT)* that effectively relieves the quantization artifacts and avoids expensive resource requirements. Each layer of EPT implements the local self-attention mechanism for analyzing continuous 3D coordinates and offers fast inference time using a voxel hashing-based architecture. The proposed method can be adopted for various 3D vision applications, such as 3D semantic segmentation and 3D detection. In experiments, the proposed EPT model outperforms the state-of-the-art on large-scale 3D semantic segmentation benchmarks and also shows better performance on 3D detection benchmarks than point-based or voxel-based baseline methods.

1 INTRODUCTION

3D scene understanding is fundamental due to its importance to various fields, such as robotics, intelligent agents, and AR/VR. Recent approaches (Choy et al., 2019; Graham et al., 2018; Mao et al., 2019; Qi et al., 2017a;b; Tatarchenko et al., 2018; Thomas et al., 2019) utilize the deep learning frameworks for 3D scene understanding, but handling a large 3D scene as a whole remains a challenging problem because it involves extensive computation and memory budgets. As an alternative, some methods crop 3D scenes and stitch predictions, or others approximate point coordinates for efficiency. However, such techniques either require a considerable inference time or show the lowered accuracies. As a result, ensuring fast inference time and accuracy is one of the primary considerations in the 3D scene understanding tasks.

The pioneering 3D understanding approaches, PointNet (Qi et al., 2017a) and PointNet++ (Qi et al., 2017b) process point clouds with multi-layer perceptrons (MLPs) that preserve permutation-invariance of the point clouds. Such *point-based methods* introduce impressive results (Mao et al., 2019; Thomas et al., 2019) recently. However, it involves grouping of point clouds using k-nearest neighbor search. In addition, applying scene-level inference with the point-based methods require cropping and stitching scenes. *Voxel-based methods* (Choy et al., 2019; Graham et al., 2018) are alternatives for a large-scale 3D scene understanding due to their efficiency of the network design. However, they may lose fine geometric patterns due to quantization artifacts. *Hybrid methods* (Tatarchenko et al., 2018; Liu et al., 2019; Tang et al., 2020) reduce the quantization artifacts by utilizing *both* point-level and voxel-level features. However, the hybrid models require additional memory space to cache both features.

This work proposes an efficient point transformer (EPT) that efficiently encodes continuous positional information of large-scale point clouds. Our approach leverages local self-attention (Vaswani et al., 2021; Ramachandran et al., 2019b) of point clouds with voxel hashing architecture. To achieve higher accuracy, we present centroid-aware voxelization and devoxelization techniques that effectively preserve the embedding of continuous coordinates. The proposed approach reduces quantization artifacts, and it allows coherency of dense predictions regardless of rigid transformations. We also introduce a reformulation of the standard local self-attention equation to reduce space complexity further. The proposed local self-attention module can replace the convolutional layers for 3D scene understanding. In this manner, we introduce a U-shaped EPT network, which naturally builds a hierarchical network structure without using manual grouping of point clouds. As a result, EPT collects rich geometric representations and exhibits a fast inference time even for large-scale scenes.

We conduct experiments using two datasets of large-scale scenes: SCANNET (Dai et al., 2017) and S3DIS (Armeni et al., 2016). Our method shows a consistent improvement in the semantic segmentation task on various voxel hashing configurations. We also apply the EPT network as a backbone of VoteNet (Qi et al., 2019) to show the applicability of EPT in the 3D object detection task. We use SCANNET (Dai et al., 2017) dataset for the 3D detection task, and our model shows better accuracy (mAP) than other baselines that use point- or voxel-based network backbones. Besides, we introduce a novel consistency score metric and demonstrate that our model outputs more coherent predictions under rigid transformations, i.e., rotation and translation.

In summary, our contributions are as follows:

1. We propose a novel local self-attention-based network, called Efficient Point Transformer (EPT), for large-scale 3D scene understanding. Our proposed method effectively learns fine geometric details by preserving positional information of points in 3D point clouds.
2. Our model benefits from the fast inference time of voxel hashing-based architecture, resulting in faster inference than point-based work while achieving better performance than prior voxel-based work.
3. We apply EPT on two 3D vision tasks: large-scale 3D semantic segmentation and detection. Our model achieves improved performance than the point- and voxel-based approaches.
4. We demonstrate our model produces much more consistent predictions than the previous voxel-based approach via controlled experiments. The results contend that our model makes coherent predictions when rigid transformations are applied to 3D scenes.

2 RELATED WORK

In this section, we review point-based, voxel-based, and hybrid methods for 3D scene understanding and then revisit the attention-based models.

Point-based methods. Qi et al. (2017a) introduce a multi-layer perceptrons (MLP) based approach for understanding 3D scenes. Qi et al. (2017b) advance PointNet (Qi et al., 2017a) by adding hierarchical sampling strategies on top of the PointNet architecture. Recent studies attempt to apply convolution on point clouds since the heuristic local sampling and grouping mechanisms used in Qi et al. (2017b) can be represented by the convolution. However, applying convolution on point clouds is challenging since 3D points are sparse and unordered. Thomas et al. (2019) mimic convolution using kernel points defined in the continuous space. They construct a k-d tree to perform point-wise convolution on the query points within a certain radius at the inference stage in exchange for inefficiency at the data preprocessing stage. Mao et al. (2019) adopt discretized convolution kernels instead of continuous kernels for efficiency and perform convolution on every point in a point cloud, which poses a bottleneck when processing large-scale 3D scene point clouds. In order to address inefficiency of point-based methods, Zhao et al. (2021) and Guo et al. (2020) utilize attention operations. Their attention operations enable them to learn richer feature representations than the fixed kernel-based methods (Mao et al., 2019; Thomas et al., 2019) by preserving permutation-invariance of point clouds. In fact, most point-based methods (Mao et al., 2019; Qi et al., 2017a;b; Thomas et al., 2019; Zhao et al., 2021; Guo et al., 2020) adopt expensive operations, such as k nearest neighbor search or k-d tree construction, resulting in heavy computational overhead when processing large-scale 3D scenes.

Voxel-based methods. Sparse convolution (Choy et al., 2019; Graham et al., 2018) constructs fully convolutional neural networks using discrete sparse tensors, enabling fast processing of voxel data. The sparse convolution performs convolution to all valid neighbor voxels that are efficiently found using a hash table with constant time complexity, i.e., $\mathcal{O}(1)$. Mao et al. (2021) propose a voxel-based transformer architecture that adopts both local and dilated attention to enlarge receptive fields of the model. In spite of effectiveness of voxel-based work on large-scale point clouds, they often fail to capture fine patterns of point clouds due to the quantization artifacts produced during voxelization. In other words, the features extracted by voxel-based methods are inconsistent regarding the voxel size (Zhang and Richard, 2019).

Hybrid methods. One another approach to handle point clouds is to extract both point- and voxel-level features. Recent work (Liu et al., 2019; Tang et al., 2020; Zhang et al., 2020; Zhang et al., 2021) attaches point-based layers, e.g. *mini*-PointNet, on top of the voxel-based methods to relieve the quantization artifacts produced during voxelization. They take advantages from fast neighbor search of voxel-based methods and high capability of capturing fine-geometries of point-based methods. However, the hybrid methods suffer from larger computation and memory budgets since the approaches in this category store both point- and voxel-level features.

Attention-based Networks. Discussions regarding attention operation have dominated research in recent years in Natural Language Processing (Vaswani et al., 2017; Devlin et al., 2018; Radford et al., 2018). Moreover, recent vision work (Irwan Bello, 2021; Dosovitskiy et al., 2020; Han et al., 2021; Yuan et al., 2021) has attempted to exploit advantages of attention-based models. Prior research generally confirms that global self-attention is infeasible to be adopted in 3D vision tasks due to its costly operations. Thus, recent work (Guo et al., 2020; Zhao et al., 2021; Mao et al., 2021) widely utilizes local self-attention (Ramachandran et al., 2019b; Vaswani et al., 2021; Irwan Bello, 2021) to process 3D point clouds. Guo et al. (2020) and Zhao et al. (2021) handle irregularity of point clouds with *k* nearest neighbor search, resulting in remarkable performance gain.

3 Efficient Point Transformer

3.1 Overview

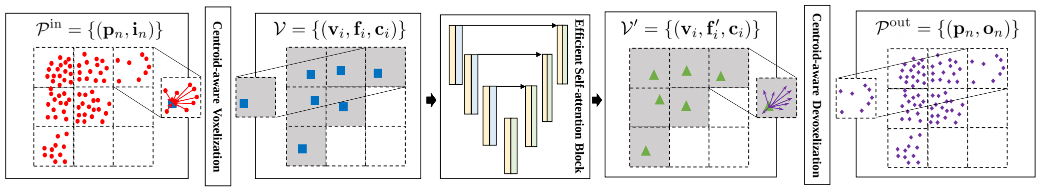
$\mathcal{P}^{\text{in}} = \{(\mathbf{p}_n, \mathbf{i}_n)\}$
Centroid-aware Voxelization
$\mathcal{V} = \{(\mathbf{v}_i, \mathbf{f}_i, \mathbf{c}_i)\}$
Efficient Self-attention Block
$\mathcal{V}' = \{(\mathbf{v}_i, \mathbf{f}'_i, \mathbf{c}_i)\}$
Centroid-aware Devoxelization
$\mathcal{P}^{\text{out}} = \{(\mathbf{p}_n, \mathbf{o}_n)\}$

Figure 1: Overview of the proposed Efficient Point Transformer (EPT).

Efficient Point Transformer (EPT) processes the input point cloud through three steps: (Step 1) *centroid-aware* voxelization, (Step 2) *Efficient* self-attention (ESA), and (Step 3) *centroid-aware* devoxelization.

(Step 1) Let $\mathcal{P}^{\text{in}} = \{(\mathbf{p}_n, \mathbf{i}_n)\}_{n=1}^{N}$ be an input point cloud, where $\mathbf{p}_n$ is the n-th point coordinate and $\mathbf{i}_n$ is any raw input feature of $\mathbf{p}_n$, *e.g.*, color of point clouds. For the efficiency, our approach voxelizes $\mathcal{P}^{\text{in}}$ into $\mathcal{V} = \{(\mathbf{v}_i, \mathbf{f}_i, \mathbf{c}_i)\}_{i=1}^{I}$, a set of tuples. Each tuple contains i-th voxel coordinate $\mathbf{v}_i$, voxel feature $\mathbf{f}_i$, and voxel centroid coordinate $\mathbf{c}_i$. We introduce centroid-aware voxelization process that utilizes learnable positional embedding $\mathbf{e}_n$ between n-th point and its voxel centroid to minimize the loss from the quantization procedure.

(Step 2) The Efficient Self-Attention (ESA) block takes $\mathcal{V} = \{(\mathbf{v}_i, \mathbf{f}_i, \mathbf{c}_i)\}$ and updates the feature $\mathbf{f}_i$ to the output feature $\mathbf{f}'_i$ using local self-attention. In this procedure, querying neighbor voxels can be efficiently done with voxel hashing having $O(N)$ complexity.

(Step 3) The output voxels $\mathcal{V}' = \{(\mathbf{v}_i, \mathbf{f}'_i, \mathbf{c}_i)\}$ from the ESA block are devoxelized into the output point cloud $\mathcal{P}^{\text{out}} = \{(\mathbf{p}_n, \mathbf{o}_n)\}_{n=1}^{N}$, where $\mathbf{o}_n$ is the output point feature. We propose to use learnable positional embedding $\mathbf{e}_n$ to properly assign voxel-wise features to the continuous 3D points.

3.2 Centroid-aware Voxelization and Devoxelization

Centroid-aware Voxelization. Let us consider an input point cloud $\mathcal{P}^{\text{in}} = \{(\mathbf{p}_n, \mathbf{i}_n)\}$. We voxelize input points for efficient and scalable querying. The output voxels are denoted by $\mathcal{V} = \{(\mathbf{v}_i, \mathbf{f}_i, \mathbf{c}_i)\}$. We introduce a novel *centroid-to-point* positional encoding $\mathbf{e}_n \in \mathbb{R}^{D_{\text{enc}}}$ to mitigate the geometric information loss during voxelization. With an encoding layer $\delta_{\text{enc}} : \mathbb{R}^3 \mapsto \mathbb{R}^{D_{\text{enc}}}$, the *centroid-to-point* positional encoding $\mathbf{e}_n$ is defined as follows:

$$\mathbf{e}_n = \delta_{\text{enc}}(\mathbf{p}_n - \mathbf{c}_{i=\mu(n)}), \tag{1}$$

where centroid $\mathbf{c}_i$ is $\mathbf{c}_i = \frac{1}{|\mathcal{M}(i)|} \sum_{n \in \mathcal{M}(i)} \mathbf{p}_n$, $\mathcal{M}(i)$ is a set of point indices within the i-th voxel, and $\mu : \mathbb{N} \mapsto \mathbb{N}$ is an index mapping from a point index n to its corresponding voxel index i. We define the output voxel feature $\mathbf{f}_i \in \mathbb{R}^{D_{\text{in}}+D_{\text{enc}}}$ with the input point feature $\mathbf{i}_n \in \mathbb{R}^{D_{\text{in}}}$ and the encoding $\mathbf{e}_n$ as follows:

$$\mathbf{f}_i = \Omega_{n \in \mathcal{M}(i)}(\mathbf{i}_n \oplus \mathbf{e}_n), \tag{2}$$

where $\oplus$ denotes vector concatenation and Ω is a permutation-invariant operator, *e.g.*, $\text{average}(\cdot)$.

We state that some voxel-based methods (Su et al., 2018; Rosu et al., 2019; Zhang et al., 2020) introduce barycentric interpolation to embded $\mathbf{f}_i$ into *regular* grids $\mathbf{v}_i$ for voxelization. The proposed centroid-aware voxelization is different from those methods in that it encodes the *centroid-to-point* position into $\mathbf{f}_i$ at *continuous* centroid coordinate $\mathbf{c}_i$. The proposed centroid-aware voxeliztion is also different from other class of voxel-based methods (Choy et al., 2019; Mao et al., 2021; Graham et al., 2018) that apply average- or max-pool voxel features without using intra-voxel coordinates of points.

Centroid-aware Devoxelization. Since the *centroid-to-point* positional encoding $\mathbf{e}_n$ has an useful information about the relative position between $\mathbf{p}_n$ and $\mathbf{c}_i$, we can propose a centroid-aware devoxelization process. Given an output voxels $\mathcal{V}' = \{(\mathbf{v}_i, \mathbf{f}'_i, \mathbf{c}_i)\}$ with the output voxel feature $\mathbf{f}'_i \in \mathbb{R}^{D_{\text{out}}}$, the proposed centroid-aware devoxelization process is formulated as follows:

$$\mathbf{o}_n = \text{MLP}(\mathbf{f}'_{i=\mu(n)} \oplus \mathbf{e}_n), \tag{3}$$

where $\mathbf{o}_n \in \mathbb{R}^{D_{\text{out}}}$ is the n-th output point feature of the output point cloud $\mathcal{P}^{\text{out}} = \{(\mathbf{p}_n, \mathbf{o}_n)\}$ and $\text{MLP}(\cdot) : \mathbb{R}^{D_{\text{out}}+D_{\text{enc}}} \mapsto \mathbb{R}^{D_{\text{out}}}$ denotes a multilayer perceptron. We experimentally show that the proposed centroid-aware voxelization and devoxelization process relieves quantization artifacts of voxelization more effectively than the results of barycentric interpolation based methods (Su et al., 2018; Liu et al., 2019; Zhang et al., 2020).

3.3 Efficient Self-Attention

Local self-attention on centroids. Once an input point cloud $\mathcal{P}^{\text{in}} = \{(\mathbf{p}_n, \mathbf{i}_n)\}$ is transformed into a set of voxels $\mathcal{V} = \{(\mathbf{v}_i, \mathbf{f}_i, \mathbf{c}_i)\}$, we can apply local self-attention mechanism (Ramachandran et al., 2019a; Zhao et al., 2020; Xizhou et al., 2021) with $\mathcal{V}$. In this procedure, we can query neighboring voxels quickly via voxel-hashing, that requires $O(N)$ complexity. Note that point-based methods (Xu et al., 2021; Zhao et al., 2021) need to build neighbors using k-nearest neighbor search having complexity of $O(N \log(N))$, which become burdensome for processing large-scale point clouds. Given local neighbor indices of $\mathbf{c}_i$ denoted by $\mathcal{N}(i)$, local self-attention on $\mathbf{c}_i$ can be formulated as follows:

$$\mathbf{f}'_i = \sum_{j \in \mathcal{N}(i)} a(\mathbf{f}_i, \delta(\mathbf{c}_i, \mathbf{c}_j)) \psi(\mathbf{f}_j), \tag{4}$$

where $\mathbf{f}'_i$ is output feature, $a(\mathbf{f}_i, \delta(\mathbf{c}_i, \mathbf{c}_j))$ is a function of attention weights using positional encoding $\delta(\mathbf{c}_i, \mathbf{c}_j)$ and ψ is the value projection layer.

Although the voxel hashing enables an efficient neighbor search with time complexity of $\mathcal{O}(1)$ for a single query, designing an efficient form of continuous positional encoding $\delta(\mathbf{c}_i, \mathbf{c}_j)$ still remains challenging problem. Specifically, Zhao et al. (2021) use $\text{MLP}(\mathbf{c}_i - \mathbf{c}_j)$ for implementing $\delta(\mathbf{c}_i, \mathbf{c}_j) \in \mathbb{R}^D$, but it requires $\mathcal{O}(NKD)$ space complexity, where N is the number of voxels and K is the cardinality of neighboring voxels. This is because there can be total NK different relative positions of $(\mathbf{c}_i - \mathbf{c}_j)$ for possible (i, j) pairs due to the continuity of $\mathbf{c}$.

Reducing space complexity. We introduce a coordinate decomposition technique to reduce space complexity. Given a query voxel $(\mathbf{v}_i, \mathbf{f}_i, \mathbf{c}_i)$ and a key voxel $(\mathbf{v}_j, \mathbf{f}_j, \mathbf{c}_j)$, the relative position of centroids $\mathbf{c}_i - \mathbf{c}_j$ can be decomposed as follows:

$$\mathbf{c}_i - \mathbf{c}_j = (\mathbf{c}_i - \mathbf{v}_i) - (\mathbf{c}_j - \mathbf{v}_j) + (\mathbf{v}_i - \mathbf{v}_j). \tag{5}$$

With Eq. (5), we can decompose the memory-consuming $\delta(\mathbf{c}_i, \mathbf{c}_j)$ into two kinds of positional encodings: (1) a continuous positional encoding $\delta_{\text{abs}}(\mathbf{c}_i - \mathbf{v}_i)$ whose space complexity is $\mathcal{O}(ND)$

due to continuity of $\mathbf{c}$, and (2) a discretized positional encoding $\delta_{\text{rel}}(\mathbf{v}_i - \mathbf{v}_j)$ whose space complexity is $\mathcal{O}(KD)$. $\delta_{\text{rel}}(\mathbf{v}_i - \mathbf{v}_j)$ is memory-efficient because there can be only K different discretized relative positions of $(\mathbf{v}_i - \mathbf{v}_j) \in \mathbb{R}^3$ for all possible (i, j) pairs. In addition, it is due to the fact that the K is a way smaller than number of points N. $\delta_{\text{abs}}(\mathbf{c}_j - \mathbf{v}_j)$ in Eq. (5) does not add any additional space complexity because we already have $\delta_{\text{abs}}(\mathbf{c}_i - \mathbf{v}_i)$ for every voxel. As a result, space complexity of $\delta(\mathbf{c}_i, \mathbf{c}_j)$ become $\mathcal{O}(NKD)$ to $\mathcal{O}(ND + KD)$.

Given, Eq. (4) and (5), we see that local self-attention use continuous positional encoding $\delta_{\text{abs}}(\mathbf{c}_i - \mathbf{v}_i)$ and input voxel feature $\mathbf{f}_i$. Therefore, the local self-attention pipeline has *centroid-aware* property that can reduce quantization artifacts. Based on these insights, we propose to use an aggregated feature $\mathbf{g}_i = \mathbf{f}_i + \delta_{\text{abs}}(\mathbf{c}_i - \mathbf{v}_i)$ and name it as *centroid-aware* voxel feature. We compute attention weights with $\delta_{\text{rel}}(\mathbf{v}_i - \mathbf{v}_j)$ as follows:

$$\mathbf{f}'_i = \sum_{j \in \mathcal{N}(i)} a(\mathbf{g}_i, \delta_{\text{rel}}(\mathbf{v}_i - \mathbf{v}_j))\psi(\mathbf{g}_j). \tag{6}$$

Efficient Self-Attention (ESA) layer. Now, we propose the ESA layer by defining attention function $a(\cdot)$ in Eq. (6) as follows:

$$\mathbf{f}'_i = \sum_{j \in \mathcal{N}(i)} \frac{\psi(\mathbf{g}_i) \cdot \delta_{\text{rel}}(\mathbf{v}_i - \mathbf{v}_j)}{\|\psi(\mathbf{g}_i)\| \|\delta_{\text{rel}}(\mathbf{v}_i - \mathbf{v}_j)\|} \phi(\mathbf{g}_j). \tag{7}$$

It is worth noting that ESA layer use the *cosine similarity* between $\psi(\mathbf{g}_i)$ and $\delta_{\text{rel}}(\mathbf{v}_i - \mathbf{v}_j)$. Instead of using $\text{softmax}(\psi(\mathbf{g}_i)^\top \delta_{\text{rel}}(\mathbf{v}_i - \mathbf{v}_j))$, cosine similarity can effectively handle the sparsity issue of input voxels $\mathcal{V}$ properly. For an example, an issue arises if we use $\text{softmax}(\cdot)$ and $|\mathcal{N}(i)|$ is 1. In this case, $\text{softmax}(\cdot)$ normalizes the attention weights into 1.0, and it can make the ESA layer to be a simple linear layer ϕ. In addition, as ESA layer queries local neighbor indices, $|\mathcal{N}(i)|$ varies from 1 to the number of neighboring voxels. Therefore, *cosine similarity* is more natural choice for handling varying number of voxels than $\text{softmax}(\cdot)$.

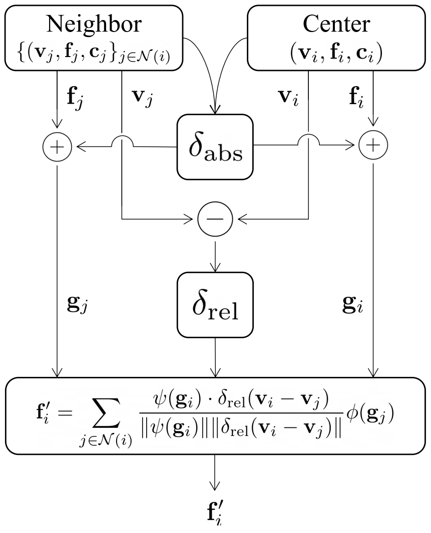
Neighbor
$\{(\mathbf{v}_j, \mathbf{f}_j, \mathbf{c}_j\}_{j \in \mathcal{N}(i)}$
Center
$(\mathbf{v}_i, \mathbf{f}_i, \mathbf{c}_i)$
$\mathbf{f}_j$
$\mathbf{v}_j$
$\mathbf{v}_i$
$\mathbf{f}_i$
δ_{abs}
δ_{rel}
$\mathbf{g}_j$
$\mathbf{g}_i$
$\mathbf{f}'_i = \sum_{j \in \mathcal{N}(i)} \frac{\psi(\mathbf{g}_i) \cdot \delta_{\text{rel}}(\mathbf{v}_i - \mathbf{v}_j)}{\|\psi(\mathbf{g}_i)\| \|\delta_{\text{rel}}(\mathbf{v}_i - \mathbf{v}_j)\|} \phi(\mathbf{g}_j)$
$\mathbf{f}'_i$

Figure 2: Computational graph of ESA.

The dynamics of ESA layer shown in Eq. (7) generates weights using the *centroid-aware* features $\psi(\mathbf{g}_i)$ and relative voxel features $\delta_{\text{rel}}(\mathbf{v}_i - \mathbf{v}_j)$). This design enables ESA layer to learn more coherent representation under the rigid transformations than sparse convolution based approach (Choy et al., 2019), as shown in Table. 1 and to outperform sparse convolution on various tasks (*e.g.*, 3D semantic segmentaion, 3D object detection) as shown in Table. 2, and Table. 5. We also experimentally show that the reformulation from Eq. (4) to Eq. (6) works reasonably. (as shown in Table. 3) and introduces extra efficiency (as shown in Table. 4).

3.4 Network Architecture

Based on the modules introduced above, we develop an efficient transformer for dense prediction on point cloud, dubbed *Efficient Point Transformer (EPT)*. Using coordinate hashing (Sec. 3.2) and decomposed positional encodings (Sec. 3.3), EPT is less prone to quantization errors than previous voxel-based methods (Choy et al., 2019; Graham et al., 2018; Mao et al., 2021), while also being significantly more efficient than point-based methods (Xu et al., 2021; Zhao et al., 2021) in terms of both space and time. Furthermore, the proposed ESA layer can be easily be *integrated to voxel-based downsampling and upsampling layer* without introducing heuristic sampling and grouping mechanism like most of the point-based methods (Qi et al., 2017b; Thomas et al., 2019; Xu et al., 2021; Zhao et al., 2021). Note that we can build local self-attention networks by substituting convolution layers with ESA layers. Therefore, any successful sparse CNN architectures can be modified to faciliate local self-attention, *e.g.*, U-Net (Ronneberger et al., 2015) and ResNet (He et al., 2016). We implement our model for semantic segmentation using the U-Net (Ronneberger et al., 2015) architecture. Further details are described in Appendix A.1.

4 Experiment

In this section, we evaluate our model on popular large-scale 3D scene datasets: ScanNet (Dai et al., 2017) and S3DIS (Armeni et al., 2016) because they provide rich diversity and dense 3D annotations. We first validate robustness of our approach to voxel hashing configurations described in Sec. 4.3. Then, we compare the proposed method with the state-of-the-art and discuss the results in Sec. 4.4 and Sec. 4.6. We have conducted all experiments with a fixed random seed for reproducibility. We have described details (*e.g.*, hyperparameters, learning rate) of all the experiments in Appendix A.1.

4.1 Dataset

ScanNet. We use the second official release of ScanNet (Dai et al., 2017), which consists of 1.5k room scenes with some rooms captured repeatedly with different sensors. Following the experimental settings of prior work(Qi et al., 2019; Chaton et al., 2020), our model uses point-wise RGB colors and coordinates as input point features $\{\mathbf{i}_n\}$ for 3D semantic segmentation task and 3D objection detection, respectively.

S3DIS is a large-scale indoor dataset which consists of six large-scale areas with 271 room scenes. We test on Area 5 and utilize the other splits during training. Following Choy et al. (2019), we do not use any preprocessing methods *e.g.*, *cropping into small blocks* that are widely used in point-based methods (Qi et al., 2017a; Tchapmi et al., 2017; Tatarchenko et al., 2018; Li et al., 2018; Landrieu et al., 2018; Xu et al., 2021; Zhao et al., 2021).

4.2 Baselines

We have selected PointNet (Qi et al., 2017a), PointNet++ (Qi et al., 2017b), SegCloud (Tchapmi et al., 2017), TangentConv (Tatarchenko et al., 2018), PointCNN (Li et al., 2018), SPGraph (Landrieu et al., 2018), FPConv (Lin et al., 2020), PointConv (Wu et al., 2019), PointASNL (Yan et al., 2020), and SparseConvNet (Graham et al., 2018) as the baseline approaches. MinkowskiNet32 and MinkowskiNet42 (Choy et al., 2019) are compared as representative voxel-based methods that comprise 32 and 42 U-Net layers, respectively.

We reproduce MinkowskiNet42 with official source code and denote it as MinkowskiNet42† with different voxel sizes. KPConv (Thomas et al., 2019) and PAConv (Xu et al., 2021) are selected since they are representative point-based methods. The main difference between the two methods is that KPConv (Thomas et al., 2019) uses a k-d tree to boost its inference time while PAConv (Xu et al., 2021) does not. We follow the official guideline of the two methods and reproduce the results. A most recent method, Point Transformer (Zhao et al., 2021) has also been selected due to its superiority on several datasets. However, since there is no official release of the Point Transformer, we use the reported performance in the paper. Unlike our method and selected baselines, other approaches (Kundu et al., 2020; Chiang et al., 2019; Hu et al., 2021) use additional inputs *e.g.*, 2D images or meshes. Accordingly, we have excluded these methods from the comparison.

4.3 Consistency Test

We introduce a new evaluation metric to measure the coherency of predictions under various rigid transformations, such as translation and rotation. Let us consider a set of point clouds $\mathcal{S} = \{\mathcal{P}^{\text{in}}\}$ and a 3D semantic segmentation model $f : \mathcal{P}^{\text{in}} \mapsto \mathbb{C}$ which predicts a semantic class of each point in $\mathcal{P}^{\text{in}} = \{(\mathbf{p}_n, \mathbf{i}_n)\}$. Given $\mathcal{S}$ and a set of rigid transformations $\mathcal{T} = \{\mathbf{T}_m\}$, we introduce the consistency score (CScore) of f on $\mathcal{S}$ as follows:

$$\text{CScore}(f; \mathcal{S}, \mathcal{T}) = \frac{1}{|\mathcal{S}|} \sum_{\mathcal{P}^{\text{in}} \in \mathcal{S}} \frac{1}{|\mathcal{P}^{\text{in}}||\mathcal{T}|} \sum_{n}^{|\mathcal{P}^{\text{in}}|} \sum_{m}^{|\mathcal{T}|} \mathbb{I}[f(\mathbf{p}_n, \mathbf{i}_n) = f(\mathbf{T}_m \mathbf{p}_n, \mathbf{i}_n)], \tag{8}$$

where $\mathbb{I}[\cdot]$ is the indicator function, and it checks whether class predictions of the original point and the transformed point are the same. CScore is an averaged accuracy over $\mathcal{S}$, $\mathcal{P}$, and $\mathcal{T}$. Similarly, we use the point-wise CScore of f on $\mathcal{P}$ to show which points in $\mathcal{P}$ is vulnerable to $\mathcal{T}$. We apply 14 different rigid transformations that consist of seven translations and seven rotations around the gravity axis. For the voxel size L, seven translations are set to $[0, 0.5L]^3$ except zero translation $[0, 0, 0]$.

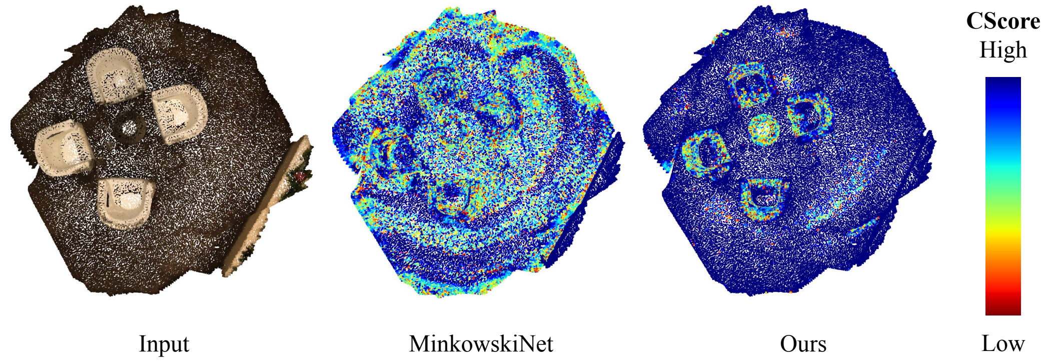

Figure 3: Heatmap visualization of consistency score (CScore) of MinkowskiNet (Choy et al., 2019) and the proposed efficient point transformer. Points with high CScore (consistently predicts the same class) are colored blue and points with low CScore (the predicted class is not consistent with arbitrary rigid transformations) are colored red. Table 1 shows quantitative evaluation.

Seven rotation angles along gravity axis is set to $[0.25\pi, 0.5\pi, \cdots, 1.75\pi]$. We evaluate CScore of MinkowskiNet42 and EPT on SCANNET validation split. The evaluation results (Table. 1) and the qualitative results (Figure. 3) show that EPT outputs more coherent feature representations than the MinkowskiNet42.

4.4 3D SEMANTIC SEGMENTATION

We compare our approach with the state of the arts on SCANNET and S3DIS. We use the mean of class-wise IoU scores as the primary evaluation metric for both datasets.

SCANNET. We evaluate the models on the ScanNet validation split due to strict submission policies of SCANNET online test benchmark, where one method can be tested at most once. Our proposed method outperforms the MinkowskiNet42† at voxel sizes of 2cm, 5cm, and 10cm by 0.3, 4.1, and 4.9 absolute percentage point gain in Mean IoU (%) respectively[1]. The experimental results in Table. 1 and Table. 2 indicate that the proposed method can represent a large-scale point cloud as features that are more robust to quantization error. We also visualize the semantic segmentation results of EPT (10cm) and MinkowskiNet42 (Choy et al., 2019) (10cm) in Figure. 4.

Table 1: Consistency scores of the proposed efficient point transformer (EPT) and MinkowskiNet42† on different transformation sets, namely, rotation only ($\mathbf{R}$), translation only ($\mathbf{t}$), and both ($\mathbf{R}$ and $\mathbf{t}$). The size of voxel is set to 10cm, 5cm, and 2cm in a SCANNET dataset. EPT relieves the prediction inconsistency occurred by voxelization artifact.

Voxel	Methods	$\mathbf{R}$ (%)	$\mathbf{t}$ (%)	$\mathbf{R}$ and $\mathbf{t}$ (%)
10cm	MinkowskiNet42†	73.88	76.28	75.08
	EPT (ours)	**75.63**	**78.63**	**77.13**
5cm	MinkowskiNet42†	74.37	84.43	79.40
	EPT (ours)	**90.61**	**84.90**	**87.75**
2cm	MinkowskiNet42†	96.23	97.99	97.11
	EPT (ours)	**99.67**	**99.60**	**99.63**

S3DIS. We compare the mean accuracy and mean IoU of EPT with the state of the arts on the S3DIS Area 5 test split. Since Choy et al. (2019) reported results with a lightweight network (MinkowskiNet32), we utilize the official code of MinkowskiNet42 and reproduce the results with voxel sizes of 5cm and 10cm. We denote the results of reproduction as MinkowskiNet42†. EPT outperforms the MinkowskiNet42† at a 10cm and 5cm voxel size by 4.0 and 1.5 absolute percentage score in mean IoU (%), respectively. Given the reported performance by Zhao et al. (2021), PointTransformer shows the best performance. However, PointTransformer involves k-nearest neighbor search and requires multiple inferences of parts of the scene (as denoted in Handling Subregions column of Table 2), whereas our approach can handle the whole scene with a single feed-forward operation. As shown in Table 4, PAConv having similar computational complexity of PointTransformer shows 108 times slower inference speed than our approach.

Ablation study. We conduct ablation studies w.r.t. positional encodings, i.e., δ_{enc} and δ_{abs}, to check effects of each positional encoding in ESA layer on the SCANNET dataset. We have followed the same setup with the main experiments with a voxel size of 10cm. Table. 3 shows the results of

[1]We state that the test accuracy of MinkowskiNet42 on ScanNet leaderboard 73.6, which is better than ours. However, the reported result uses test-time augmentation that aggregates multiple inferences of rotationally augmented 3D scenes, and the number of augmentation is not reported. Our results and MinkowskiNet42† show the results without using the test-time augmentation.

Table 2: Results of semantic segmentation on the ScanNet (Dai et al., 2017) dataset (left) and S3DIS (Armeni et al., 2016) dataset (right). Note that the score denoted by † is the reproduced performance of MinkowskiNet42 (Choy et al., 2019) with the official source code.

Methods	mIoU (%) (val)	mIoU (%) (test)
PointNet++	-	33.9
FPConv	-	63.0
PointConv	61.0	66.6
PointASNL	63.5	66.6
KPConv *deform*	69.2	68.4
SparseConvNet (2cm)	-	72.5
MinkowskiNet42† (10cm)	60.4	-
EPT (ours - 10cm)	65.3	-
MinkowskiNet42 (5cm)	-	67.9
MinkowskiNet42† (5cm)	66.6	-
EPT (ours - 5cm)	70.1	-
MinkowskiNet42† (2cm)	71.7	-
EPT (ours - 2cm)	**72.0**	

Methods	Handling Subregions	mAcc (%)	mIoU (%)
PointNet	Block	49.0	41.1
SegCloud	Block	57.4	48.9
TangentConv	Block	62.2	52.6
PointCNN	Block	63.9	57.3
SPGraph	Block	66.5	58.0
PAConv	Block	-	66.6
KPConv	Grid	72.8	67.1
PointTransformer	Block	**76.5**	**70.4**
MinkowskiNet42† (10cm)	Not Needed	69.2	61.3
EPT (ours - 10cm)	Not Needed	72.6	64.0
MinkowskiNet32 (5cm)	Not Needed	71.7	65.4
MinkowskiNet42† (5cm)	Not Needed	73.3	66.0
EPT (ours - 5cm)	Not Needed	74.7	67.5

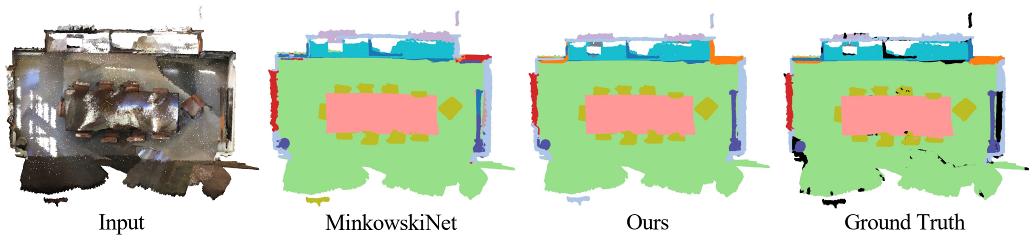

Figure 4: Visualization of semantic segmentation results on ScanNet (Dai et al., 2017).

the ablation studies. Models with full positional encodings achieved the best mIoU score. When removing δ_{abs} from our model, we have observed a large performance drop since the model does not adopt continuous position information. Removing either positional encodings of *centroid-aware* voxelization or devoxelization from EPT also degrades the performance. These results indicate that both proposed voxelization and devoxelization effectively maintain continuous geometric information of the input point cloud.

Table 3: Ablation studies on the proposed positional encoding layers.

Method	δ_{enc} - Voxelization	δ_{enc} - Devoxelization	δ_{abs}	mIoU (%)
EPT (ours - 10cm)	✗	✗	✓	62.1
EPT (ours - 10cm)	✓	✓	✗	62.7
EPT (ours - 10cm)	✓	✗	✓	63.4
EPT (ours - 10cm)	✗	Barycentric interp.	✓	63.4
EPT (ours - 10cm)	✓	✓	✓	65.3

4.5 Computational Complexity

Table. 4 theoretically analyzes the time complexity and reports the average wall-time latency of each method when processing S3DIS scenes Area 5. We measure the inference time of MinkowskiNet42 (Choy et al., 2019), KPConv (Thomas et al., 2019) and PAConv (Xu et al., 2021) using the official codes. Further details are explained in Appendix. A.3.

Due to the preprocessing stages and stitching the predictions of multiple subregions, the point-based methods (Landrieu et al., 2018; Li et al., 2018; Qi et al., 2017a; Tchapmi et al., 2017; Thomas et al., 2019; Xu et al., 2021) take much more time to inference a single scene than our approach. Note that KPConv (Thomas et al., 2019) constructs k-d tree, but we do not include this process into inference time. Detailed information about the time complexity analysis is presented in Appendix A.2.

4.6 3D Object Detection

We have conducted experiments on ScanNet 3D object detection dataset, where fine-grained point cloud representation is essential to detect and localize 3D objects.

Table 4: The theoretical time complexity of neighbor search and per-scene wall-time latency of each network on S3DIS Area 5. N is the number of dataset points, M is the number of query points, and K is the number of neighbors to search. Note that both M and N are much larger than K in a large-scale point cloud. The mIoU score denoted by * is the performance of PAConv that uses an efficient implementation of k-NN search algorithm, and PointTransformer also uses the same.

Methods	Neighbor Search		Latency	Latency	mIoU (%)
	Preparation	Inference	(sec)	(norm)	
KPConv *deform*† (100 votes)	$\mathcal{O}(N \log N)$	$\mathcal{O}(KM \log N)$	105.15	404.42	67.4
KPConv *deform*†	$\mathcal{O}(N \log N)$	$\mathcal{O}(KM \log N)$	1.18	4.54	65.5
PAConv	$\mathcal{O}(1)$	$\mathcal{O}(MN \log K)$	28.13	108.19	66.0*
PointTransformer	$\mathcal{O}(1)$	$\mathcal{O}(MN \log K)$	-	-	70.4
MinkowskiNet42† (5cm)	$\mathcal{O}(N)$	$\mathcal{O}(M)$	0.23	0.88	66.0
EPT (ours - 5cm)	$\mathcal{O}(N)$	$\mathcal{O}(M)$	0.26	1.00	67.5

Table 5: 3D object detection results of VoteNet (Qi et al., 2019) with various backbones on ScanNet dataset (Dai et al., 2017). Numbers except that of MinkowskiNet† and EPT are taken from Chaton et al. (2020).

Backbones	mAP@0.25	mAP@0.50
PointNet++	54.2	30.1
RS-CNN	51.6	29.5
KPConv	48.9	29.2
MinkowskiNet	53.8	30.2
MinkowskiNet†	55.3	32.8
EPT (ours)	**59.1**	**35.3**

Setups. For a fair comparison of EPT with previously proposed point-based methods (Qi et al., 2017b; Liu et al., 2019; Thomas et al., 2019), we use Torch-Points3D, an open-source library implemented by Chaton et al. (2020) for reproducible deep learning on 3D point clouds. Torch-Points3D sub-sample a fixed number of points from an input point cloud which is widely used for point-based methods (Qi et al., 2017b; Liu et al., 2019; Thomas et al., 2019) to process a scene-level point cloud-like ScanNet. We notice that the library also sub-sample points for the voxel-based methods, such as MinkowskiNet (Choy et al., 2019), which is not a suitable experimental configuration. Therefore, we reproduce VoteNet with the MinkowskiNet backbone, which is denoted by MinkowskiNet† in Table. 5 without input point sub-sampling, and we do not change any experimental configurations, *e.g.*, the number of voting points. We train a new VoteNet (Qi et al., 2019) with EPT backbone with the same training configuration of the reproduced VoteNet using MinkowskiNet†. We implement the EPT backbone for VoteNet (Qi et al., 2017a) by replacing sparse convolution layers in MinkowskiNet (Choy et al., 2019) with ESA layers without any change of detection network architecture (*e.g.*, voting module).

Results. As shown in Table. 5, the VoteNet (Qi et al., 2019) model with EPT as a backbone outperforms other baselines with a large margin. The results show that the proposed continuous positional encodings that EPT uses can effectively encode point cloud representation and helps the 3D detection task.

5 Conclusion

We have introduced the Efficient Point Transformer (EPT) and demonstrated its performance on 3D semantic segmentation and 3D detection tasks. The experimental results on large-scale 3D datasets show that EPT outperforms sparse convolutional neural networks (Choy et al., 2019) and point-based approaches (Qi et al., 2017b; Thomas et al., 2019; Xu et al., 2021) while having fast inference time and small resource consumption. Our approach is based on a voxel hashing structure, and the proposed centroid-aware voxelization and devoxlization relieve quantization artifacts. In the future, we will explore effective architectures for EPT rather than U-shaped architectures (Ronneberger et al., 2015) as the U-shaped network is initially designed for convolutional layers. Our code and data are going to be publicly available upon acceptance.

Concerning Reproducibility and Ethical Issues

Reproducibility. We use publicly available popular datasets: ScanNet and S3DIS. In addition, to provide clear details for inference time evaluation, details for hardware and software are explained in Appendix A.1. Our code strongly relies on external libraries, i.e., MinkowskiEngine. Thus, we have also reported version details of libraries used for implementation.

Ethical Issues. Recognizing scenes without strong supervision could cause serious privacy invasion issues. Such invasions are detrimental for locations where securities are crucial, e.g., military and bank. Besides, the real applications where each decision is critical for users, e.g., autonomous driving, would have responsibility issues if decisions of the 3D understanding network caused accidents.

References

Armeni et al. 3d semantic parsing of large-scale indoor spaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1534–1543, 2016.

Thomas Chaton, Nicolas Chaulet, Sofiane Horache, and Loic Landrieu. Torch-points3d: A modular multi-task framework for reproducible deep learning on 3d point clouds. In *2020 International Conference on 3D Vision (3DV)*, pp. 1–10. IEEE, 2020.

Chiang et al. A unified point-based framework for 3d segmentation. In *2019 International Conference on 3D Vision (3DV)*, pp. 155–163. IEEE, 2019.

Choy et al. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pp. 3075–3084, 2019.

Dai et al. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pp. 5828–5839, 2017.

Devlin et al. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Dosovitskiy et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Graham et al. 3d semantic segmentation with submanifold sparse convolutional networks. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pp. 9224–9232, 2018.

Guo et al. Pct: Point cloud transformer. *arXiv preprint arXiv:2012.09688*, 2020.

Han et al. Transformer in transformer. *arXiv preprint arXiv:2103.00112*, 2021.

He et al. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Hu et al. Bidirectional projection network for cross dimension scene understanding. *arXiv preprint arXiv:2103.14326*, 2021.

Irwan Bello. Lambdanetworks: Modeling long-range interactions without attention. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=xTJEN-ggl1b`.

Kundu et al. Virtual multi-view fusion for 3d semantic segmentation. In *European Conference on Computer Vision*, pp. 518–535. Springer, 2020.

Landrieu et al. Large-scale point cloud semantic segmentation with superpoint graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4558–4567, 2018.

Li et al. Pointcnn: Convolution on χ-transformed points. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pp. 828–838, 2018.

Lin et al. Fpconv: Learning local flattening for point convolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4293–4302, 2020.

Yongcheng Liu, Bin Fan, Shiming Xiang, and Chunhong Pan. Relation-shape convolutional neural network for point cloud analysis. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pp. 8895–8904, 2019.

Liu et al. Point-Voxel CNN for Efficient 3D Deep Learning. In *Adv. Neural Inform. Process. Syst.*, 2019.

Jiageng Mao, Yujing Xue, Minzhe Niu, Haoyue Bai, Jiashi Feng, Xiaodan Liang, Hang Xu, and Chunjing Xu. Voxel transformer for 3d object detection. *arXiv preprint arXiv:2109.02497*, 2021.

Mao et al. Interpolated Convolutional Networks for 3D Point Cloud Understanding. In *Int. Conf. Comput. Vis.*, October 2019.

Charles R Qi, Or Litany, Kaiming He, and Leonidas J Guibas. Deep hough voting for 3d object detection in point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9277–9286, 2019.

Qi et al. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pp. 652–660, 2017a.

Qi et al. PointNet++ deep hierarchical feature learning on point sets in a metric space. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pp. 5105–5114, 2017b.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018.

Ramachandran et al. Stand-alone self-attention in vision models. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019a. URL `https://proceedings.neurips.cc/paper/2019/file/3416a75f4cea9109507cacd8e2f2aefc-Paper.pdf`.

Ramachandran et al. Stand-alone self-attention in vision models. *arXiv preprint arXiv:1906.05909*, 2019b.

Ronneberger et al. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241. Springer, 2015.

Radu Alexandru Rosu, Peer Schütt, Jan Quenzel, and Sven Behnke. Latticenet: Fast point cloud segmentation using permutohedral lattices. *arXiv preprint arXiv:1912.05905*, 2019.

Su et al. Splatnet: Sparse lattice networks for point cloud processing. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2530–2539, 2018.

Tang et al. Searching efficient 3d architectures with sparse point-voxel convolution. In *Eur. Conf. Comput. Vis.*, pp. 685–702. Springer, 2020.

Tatarchenko et al. Tangent convolutions for dense prediction in 3d. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3887–3896, 2018.

Tchapmi et al. Segcloud: Semantic segmentation of 3d point clouds. In *2017 international conference on 3D vision (3DV)*, pp. 537–547. IEEE, 2017.

Thomas et al. Kpconv: Flexible and deformable convolution for point clouds. In *Int. Conf. Comput. Vis.*, pp. 6411–6420, 2019.

Vaswani et al. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

Vaswani et al. Scaling local self-attention for parameter efficient visual backbones. *arXiv preprint arXiv:2103.12731*, 2021.

Wu et al. Pointconv: Deep convolutional networks on 3d point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9621–9630, 2019.

Xizhou et al. Deformable {detr}: Deformable transformers for end-to-end object detection. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=gZ9hCDWe6ke`.

Xu et al. Paconv: Position adaptive convolution with dynamic kernel assembling on point clouds. *arXiv preprint arXiv:2103.14635*, 2021.

Yan et al. Pointasnl: Robust point clouds processing using nonlocal neural networks with adaptive sampling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5589–5598, 2020.

Yuan et al. Tokens-to-token vit: Training vision transformers from scratch on imagenet. *arXiv preprint arXiv:2101.11986*, 2021.

Cheng Zhang, Haocheng Wan, Shengqiang Liu, Xinyi Shen, and Zizhao Wu. Pvt: Point-voxel transformer for 3d deep learning, 2021.

Zhang and Richard. Making convolutional networks shift-invariant again. In *International Conference on Machine Learning*, pp. 7324–7334. PMLR, 2019.

Zhang et al. Deep fusionnet for point cloud semantic segmentation. In *ECCV*, volume 2, pp. 6, 2020.

Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip Torr, and Vladlen Koltun. Point transformer, 2021.

Zhao et al. Exploring self-attention for image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

A Appendix

A.1 Detailed experimental settings

Figure 5 illustrates detailed model designs of MinkowskiNet Choy et al. (2019) and our EPT model. To set the total parameter numbers to be similar, we adjust the channel dimensions, resulting in similar parameter numbers; 37.9M for both networks.

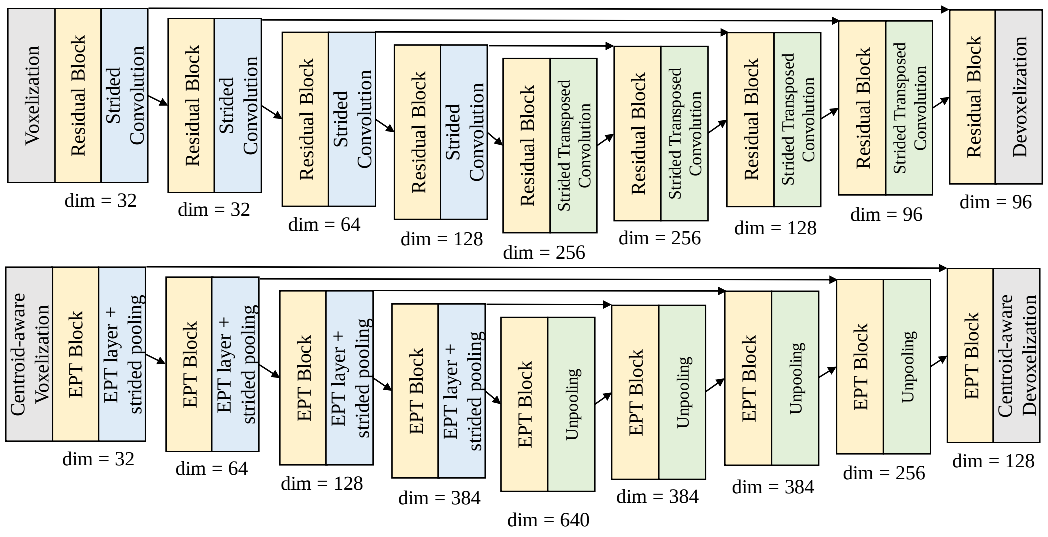

Figure 5: An illustration of network architecture that has been used in our experiments. The network figure above illustrates MinkowskiNet and the figure below illustrates our EPT model.

A.2 Time Complexity Analysis: Nearest Neighbor Search

In this section, we explain step-wise analysis of the time complexity of each method.

KPConv constructs k-d tree before inference. Based on the official implementation of KPConv, we analyze time complexity of preparation time and inference time in Algorithm 1 and Algorithm 2.

Algorithm 1 KPConv Preparation Time - k-d Tree Construction. Total: $\mathcal{O}(N \log N)$

```
Number of training points: $N$
An empty tree $T$
for point=1, 2, $\cdots$, $N$ do                          $\mathcal{O}(N \log N)$
    insert($T$, point)                                    $\mathcal{O}(\log N)$
end for
```

PAConv and PointTransformer do not require preparation steps for neighbor searching. Thus we set the preparation time to constant time. For analyzing inference time, we have followed the official implementation. Details are explained in Algorithm 3

MinkowskiNet and ours require the same process for neighbor searching since both methods benefit from voxel hashing tables. We analyze preparation and inference time complexity on Algorithm 4 and Algorithm 5, respectively.

A.3 Inference Time Evaluation

In this section, we describe the detailed setups that have been used during the inference time evaluation on the main paper. For fair comparison, no other program was run during the experiments.

Algorithm 2 KPConv Inference Time. Total: $\mathcal{O}KM \log N)$

```
Number of training points: $B$
Number of query points: $N$
Number of neighbors to search: $K$
Constructed $k$-d tree: $\bar{T}$
$k$-th nearest neighbors dictionary: $S = \{\}$
for query = 1, 2, $\cdots$, $M$ do                                    $\mathcal{O}(KM \log N)$
    arr = []
    for i = 1, 2, $\cdots$, $K$ do                                    $\mathcal{O}(K \log N)$
        point = $\mathrm{SearchClosest}(\bar{T}$, query)                $\mathcal{O}(\log N)$
        $\bar{T}$ = $\mathrm{pop}(\bar{T}$, point)                        $\mathcal{O}(\log N)$
        arr.append(point)
    end for
    S[query] = arr
end for
```

Algorithm 3 PAConv and PointTransformer Inference Time - Heap Construction. Total: $\mathcal{O}(MN \log K)$

```
Number of training points: $N$
Number of query points: $M$
Number of neighbors to search: $K$
for query = 1, 2, $\cdots$, m do                                      $\mathcal{O}(MN \log K)$
    $H = \mathrm{InitHeap}()$                                          $\mathcal{O}(K)$
    ShortestDist = 1e10
    ShortestIdx = 0
    for point = 1,2,$\cdots$,$N$ do                                    $\mathcal{O}(N \log K)$
        if $d$(point, query) $<$ ShortestDist then
            $\mathrm{reheap}$(H, ShortestDist, ShortestIdx, $K$)         $\mathcal{O}(\log K)$
            ShortestDist = $d$(point, query)
            ShortestIdx = point
        end if
    end for
    $\mathrm{Heapsort}$(H, ShortestIdx, ShortestDist, $k$)               $\mathcal{O}(K \log K)$
end for
```

1. CUDA version: 11.1
2. CUDNN version: 8.2.1
3. PyTorch version: 1.7.1
4. MinkowskiEngine version: 0.5.4
5. Hardware: single NVIDIA Geforce RTX 3090 GPU
6. Batch size: 1

A.4 More Segementation Results

We report class-wise scores in semantic segmentation task on S3DIS dataset.

	mIoU	mAcc	ceil.	floor	wall	beam	col.	wind.	door	chair	table	book.	sofa	board	clut.
PointNet	41.09	48.98	88.80	97.33	69.80	0.05	3.92	46.26	10.76	52.61	58.93	40.28	5.85	26.38	33.22
SegCloud	48.92	57.35	90.06	96.05	69.86	0.00	18.37	38.35	23.12	75.89	70.40	58.42	40.88	12.96	41.60
TangentConv	52.6	62.2	90.5	97.7	74.0	0.0	20.7	39.0	31.3	69.4	77.5	38.5	57.3	48.4	39.8
PointCNN	57.26	63.86	92.31	98.24	79.41	0.00	17.60	22.77	62.09	80.59	74.39	66.67	31.67	62.05	56.74
KPConv	67.1	72.8	92.8	97.3	82.4	0.0	23.9	58.0	69.0	91.0	81.5	75.3	75.4	66.7	58.9
PAConv	66.58	73.00	94.55	98.59	82.37	0.00	26.43	57.96	59.96	89.73	80.44	74.32	69.80	73.50	57.72
Point Transformer	70.4	76.5	94.0	98.5	86.3	0.0	38.0	63.4	75.3	89.1	82.4	74.3	80.2	76.0	59.3
EPT (ours -5cm)	67.5	74.7	91.5	97.4	86.0	0.2	40.4	60.8	66.7	87.7	79.6	73.7	58.6	77.2	57.3

Algorithm 4 MinkowskiNet and Ours Preparation Time - Hash Table Construction. Total: $\mathcal{O}(N)$

```
Number of training points: $N$
An empty hash table: $h$
**for** point = 1, 2, $\cdots$, $N$ **do**                                $\mathcal{O}(N)$
    $\mathrm{Insert}(h$, point)                                        $\mathcal{O}(1)$
**end for**
```

Algorithm 5 MinkowskiNet and Ours Inference Time - Hash Table Construction. Total: $\mathcal{O}(M)$

```
Number of query points: $M$
A constructed hash table: $\bar{h}$
**for** query = 1, 2, $\cdots$, m **do**                                  $\mathcal{O}(M)$
    lookup($\bar{h}$, query)                                           $\mathcal{O}(1)$
**end for**
```

A.5 Qualitative Results

Semantic Segmentation on the ScanNet dataset. Figure 6 visualizes more qualitative results of semantic segmentation reported in the main paper.

Consistency Score on the ScanNet dataset. Figure 7 visualizes more qualitative results of consistency evaluation in the main paper.

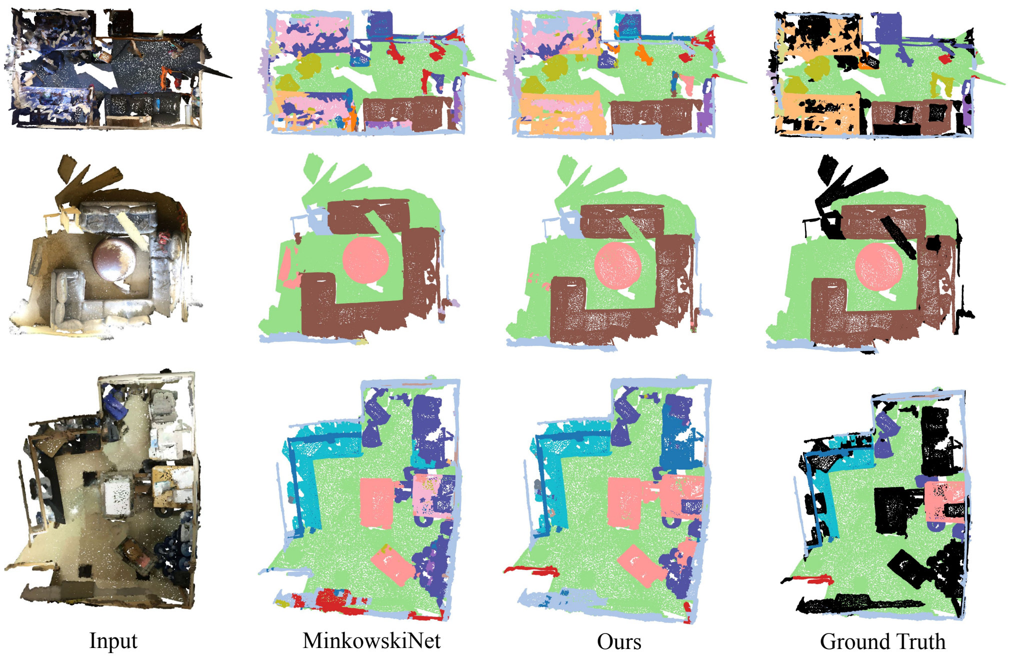

Figure 6: Qualitative semantic segmentation results of MinkowskiNet (Choy et al., 2019) and our EPT model on the SCANNET dataset.

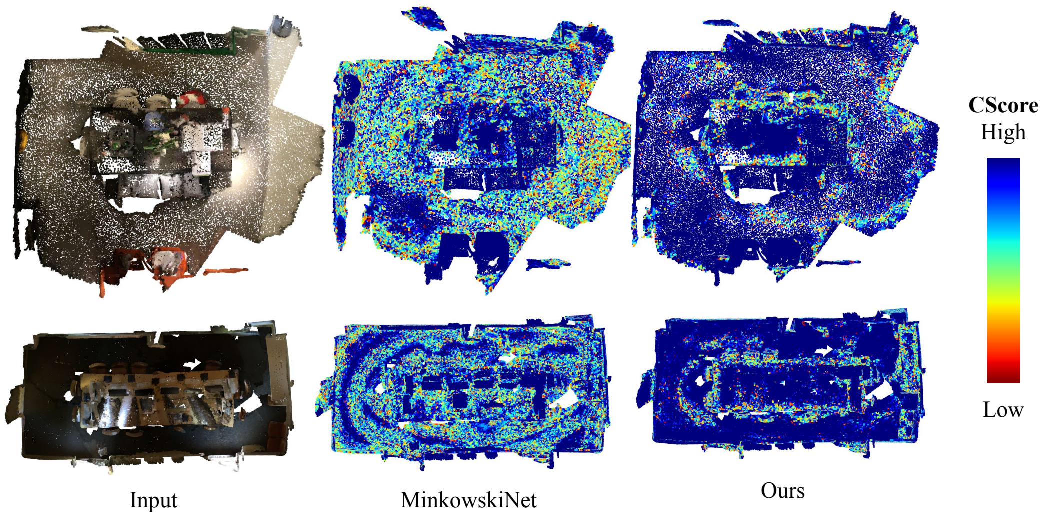

Figure 7: Heatmap visualization of consistency score (CScore) of MinkowskiNet (Choy et al., 2019) and the proposed efficient point transformer. Points with high CScore (consistently predicts the same class) are colored blue and points with low CScore (the predicted class is not consistent with arbitrary rigid transformations) are colored red. Table 1 shows quantitative evaluation.

A.6 Notations

$\mathcal{P}^{\text{in}} = \{(\mathbf{p}_n, \mathbf{i}_n)\}$	Input point cloud
$\mathbf{p}_n \in \mathbb{R}^3$	The n-th point coordinate
$\mathbf{i}_n \in \mathbb{R}^{D_{\text{in}}}$	The n-th input point feature
$\mathcal{P}^{\text{out}} = \{(\mathbf{p}_n, \mathbf{o}_n)\}$	Output point cloud
$\mathbf{o}_n \in \mathbb{R}^{D_{\text{out}}}$	The n-th point feature
$\mathcal{V} = \{(\mathbf{v}_i, \mathbf{f}_i, \mathbf{c}_i)\}$	Input voxels with centroids
$\mathbf{v}_i \in \mathbb{R}^3$	The i-th voxel center coordinate
$\mathbf{f}_i \in \mathbb{R}^{D_{\text{in}}}$	The i-th input voxel feature
$\mathbf{c}_i \in \mathbb{R}^3$	The i-th voxel centroid coordinate
$\mathcal{M}(i)$	A set of point indices within the i-th voxel
Ω	A permutation-invariant operator (*e.g.*, average)
$\mathcal{V}' = \{(\mathbf{v}_i, \mathbf{f}'_i, \mathbf{c}_i)\}$	Output voxels with centroids
$\mathbf{f}'_i \in \mathbb{R}^{D_{\text{out}}}$	The i-th output voxel feature
$\mathcal{N}(i)$	A set of neighbor voxel indices the i-th voxel
$\mathbf{e}_n$	The centroid-to-point positional encoding
δ_{enc}	An encoding layer used in centroid-to-point positional encoding
$\mathbf{o}_n$	The n-th output point feature of the output point cloud $\mathcal{P}^{\text{out}}$
$\oplus$	A vector concatenation operation
$a(\dot{)}$	An attention operation
ψ	A value projection layer in attention operations
$\mathbf{g}_i$	A centroid-aware voxel feature
δ_{rel}	A discretized positional encoding layer
δ_{abs}	A continuous positional encoding layer