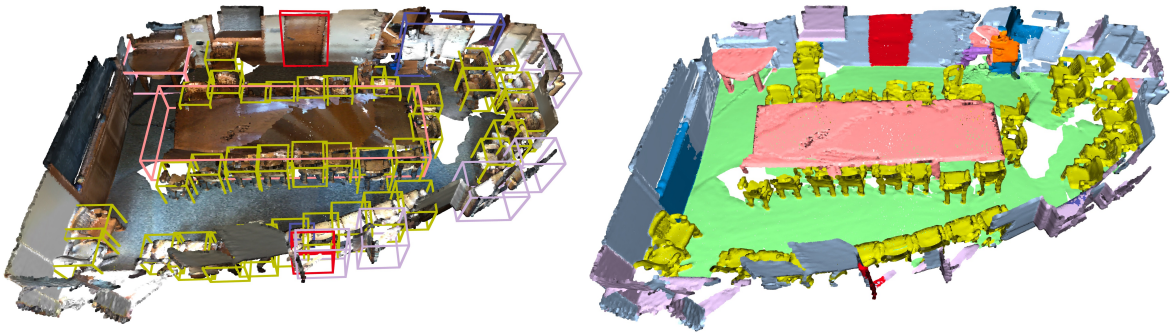
Fast Point Transformer

Chunghyun Park Yoonwoo Jeong Minsu Cho Jaesik Park
POSTECH GSAI & CSE
http://cvlab.postech.ac.kr/research/FPT



(a) 3D object detection (b) 3D semantic segmentation

Figure 1. **Fast Point Transformer** can process large-scale scenes using a local self-attention mechanism. Unlike Point Transformer [49], our approach can infer the scene at one shot without searching for point-wise neighbors. The average inference time of our network is 0.14 seconds per scene, resulting in 129 times faster than Point Transformer in 3D semantic segmentation on S3DIS dataset [2].

Abstract

The recent success of neural networks enables a better interpretation of 3D point clouds, but processing a large-scale 3D scene remains a challenging problem. Most current approaches divide a large-scale scene into small regions and combine the local predictions together. However, this scheme inevitably involves additional stages for pre- and post-processing and may also degrade the final output due to predictions in a local perspective. This paper introduces ***Fast Point Transformer*** *that consists of a new lightweight self-attention layer. Our approach encodes continuous 3D coordinates, and the voxel hashing-based architecture boosts computational efficiency. The proposed method is demonstrated with 3D semantic segmentation and 3D detection. The accuracy of our approach is competitive to the best voxel-based method, and our network achieves 129 times faster inference time than the state-of-the-art, Point Transformer, with a reasonable accuracy trade-off in 3D semantic segmentation on S3DIS dataset.*

1. Introduction

3D scene understanding is a fundamental task due to its importance to various fields, such as robotics, intelligent agents, and AR/VR. Recent approaches [6, 10, 22, 26, 27, 34, 37] utilize the deep learning frameworks, but processing a large-scale 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 [18, 26, 27, 34, 35, 41], or others approximate point coordinates for efficiency [6, 10, 23, 50]. Such techniques, however, typically lead to a substantial increase of inference time and/or degrade the final output due to the local or approximate predictions. Achieving both fast inference time and high accuracy is thus one of the primary challenges in the 3D scene understanding tasks.

The pioneering 3D understanding approaches, PointNet [26] and PointNet++ [27] process point clouds with multi-layer perceptrons (MLPs), which preserve permutation-invariance of the point clouds. Such *point-based methods* introduce impressive results [22, 37] recently, and Point Transformer [49] shows superior accuracy based on the local self-attention mechanism. However, it involves manual grouping of point clouds using k nearest neighbor search. Furthermore, scene-level inference with the point-based methods typically requires dividing a large-scale scene into smaller regions and stitching the predictions on them. While *Voxel-based methods* [1, 6, 10, 13, 19, 23, 24, 36, 50] are alternatives for a large-scale 3D scene understanding due

to their effectiveness of the network design, they may lose fine geometric patterns due to quantization artifacts. *Hybrid methods* [21, 33, 34] reduce the quantization artifacts by utilizing *both* point-level and voxel-level features. However, approaches in this category require additional memory space to cache both features.

We propose Fast Point Transformer, which effectively encodes continuous positional information of large-scale point clouds. Our approach leverages local self-attention [29, 38] of point clouds with voxel hashing architecture. To achieve higher accuracy, we present centroid-aware voxelization and devoxelization techniques that preserve the embedding of continuous coordinates. The proposed approach reduces quantization artifacts and allows the 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. Based on this, we introduce a local self-attention based U-shaped network, which naturally builds a feature hierarchy without manual grouping of point clouds. As the result, Fast Point Transformer 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: S3DIS [2] and ScanNet [7]. Our method shows competitive accuracy in the semantic segmentation task on various voxel hashing configurations. We also apply the Fast Point Transformer network as a backbone of VoteNet [25] to show the applicability in the 3D object detection task. We use ScanNet [7] dataset for the 3D detection, 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, named CScore, and demonstrate that our model outputs more coherent predictions under rigid transformations.

In summary, our contributions are as follows:

1. We propose a novel local self-attention-based network, called Fast Point Transformer that can handle large-scale 3D scenes quickly.

2. We introduce a lightweight local self-attention module that effectively learns continuous positional information of 3D point clouds while reducing space complexity.

3. We show that our model produces significantly more coherent predictions than the previous voxel-based approaches using the proposed evaluation metric.

4. We demonstrate fast inference of our voxel-hashing-based architecture; our network performs a 129 times faster inference than Point Transformer does, obtaining a reasonable accuracy trade-off in 3D semantic segmentation on S3DIS dataset [2].

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. PointNet [26] introduces a multi-layer perceptrons (MLP) based approach for understanding 3D scenes. PointNet++ [27] advances the PointNet [26] by adding hierarchical sampling strategies. Recent studies attempt to apply convolution on point clouds since the heuristic local sampling and grouping mechanisms used in PointNet++ [27] can be represented by the convolution. However, applying convolution on point clouds is challenging since 3D points are sparse and unordered. KPConv [37] mimics 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. [22] 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. More recently, Guo et al. [11] and Zhao et al. [49] utilize local self-attention operations to learn richer feature representations than the fixed kernel-based methods [22, 37]. In fact, most point-based methods [11, 22, 26, 27, 37, 49] 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 [6, 10] constructs fully convolutional neural networks using discrete sparse tensors for fast processing of voxel data. The sparse convolution performs convolution on all valid neighbor voxels that are efficiently found using a hash table with constant time complexity, *i.e.*, $\mathcal{O}(1)$. Mao et al. [23] propose a voxel-based transformer architecture that adopts both local and dilated attention to enlarge receptive fields of the model. Despite the 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 with respect to the voxel size [46].

Hybrid methods. Another approach to handle point clouds is to extract both point- and voxel-level features. Recent work [21, 33, 44, 45] 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 advantage of 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 these approaches store both point- and voxel-level features.

Attention-based networks. Discussions regarding the attention operation have dominated research in recent years

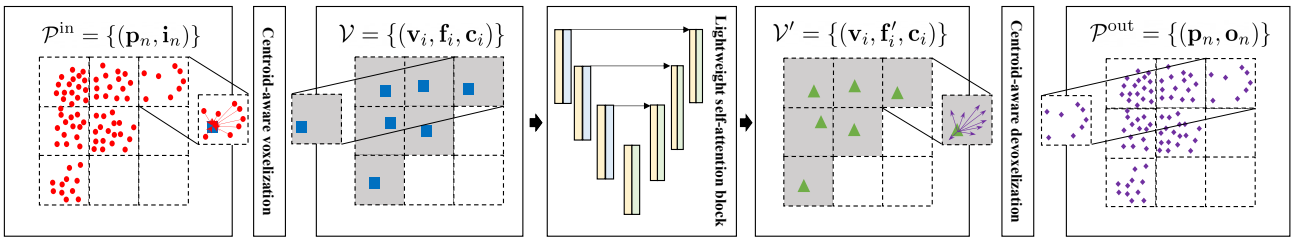


Figure 2. **Overall architecture.** We illustrate the overall architecture of the proposed Fast Point Transformer. The red points are input points and their features, and the purple points are output points and their features. The colored squares are non-empty voxels produced by voxelization. The blue and green points are centroids of non-empty voxels with their features.

in natural language processing [8, 28, 39]. Moreover, recent vision work [3, 9, 12, 43] has attempted to exploit the 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 [11, 23, 49] widely utilizes local self-attention [3, 29, 38] to process 3D point clouds. Guo et al. [11] and Zhao et al. [49] handle irregularity of point clouds with *k* nearest neighbor search, resulting in a remarkable performance gain.

3. Fast Point Transformer

3.1. Overview

Fast Point Transformer processes the point cloud through three steps: (Step 1) Centroid-aware voxelization, (Step 2) Lightweight self-attention, and (Step 3) Centroid-aware devoxelization. Figure 2 shows the overall architecture.

(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. For the computational 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 a 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 lightweight self-attention (LSA) block takes $\mathcal{V} = \{(\mathbf{v}_i, \mathbf{f}_i, \mathbf{c}_i)\}_{i=1}^{I}$ 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 done with voxel hashing having $\mathcal{O}(1)$ complexity for a single query.

(Step 3) The output voxels $\mathcal{V}' = \{(\mathbf{v}_i, \mathbf{f}'_i, \mathbf{c}_i)\}_{i=1}^{I}$ from the attention 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 for accurate point-level features.

3.2. Centroid-aware Voxel & 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 fast 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 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 $\mathbf{e}_n$:

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

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

We state that some voxel-based methods [31, 32, 45] introduce barycentric interpolation to embed $\mathbf{f}_i$ 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 [6, 10, 23] 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 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.

3.3. Lightweight Self-Attention

Local self-attention on centroids. Once an input point cloud $\mathcal{P}^{\text{in}} = \{(\mathbf{p}_n, \mathbf{i}_n)\}_{n=1}^{N}$ is transformed into a set of voxels $\mathcal{V} = \{(\mathbf{v}_i, \mathbf{f}_i, \mathbf{c}_i)\}_{i=1}^{I}$, we can apply local self-attention mechanism [29, 47, 51] with $\mathcal{V}$. In this procedure, we can query neighboring voxels quickly via voxel-hashing, which requires $\mathcal{O}(N)$ complexity. Note that point-based methods [41, 49] need to build neighbors using k nearest neighbor search having the complexity of $\mathcal{O}(N \log N)$, 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 $\delta(\mathbf{c}_i, \mathbf{c}_j) \in \mathbb{R}^D$ and ψ is the value projection layer.

Although the voxel hashing enables a fast neighbor search with time complexity of $\mathcal{O}(1)$ for a single query, designing a memory-efficient form of continuous positional encoding $\delta(\mathbf{c}_i, \mathbf{c}_j)$ still remains a challenging problem. Specifically, inspired by $\text{MLP}(\mathbf{p}_i - \mathbf{p}_j)$ in Point Transformer [49], implementing $\delta(\mathbf{c}_i, \mathbf{c}_j)$ as $\text{MLP}(\mathbf{c}_i - \mathbf{c}_j)$ requires $\mathcal{O}(IKD)$ space complexity, where K is the cardinality of neighboring voxels. This is because there can be $\mathcal{O}(IK)$ different relative positions of $(\mathbf{c}_i - \mathbf{c}_j)$ for possible (i, j) pairs due to the continuity of $\mathbf{c}$ as shown in Figure 3.

Reducing space complexity. We introduce a coordinate decomposition approach 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

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

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 $\mathcal{O}(ID)$ 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 significantly smaller than number of voxels I. $\delta_{\text{abs}}(\mathbf{c}_j - \mathbf{v}_j)$ 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 $\delta(\mathbf{c}_i, \mathbf{c}_j)$ goes down from $\mathcal{O}(IKD)$ to $\mathcal{O}(ID + KD)$ as illustrated in Figure 3.

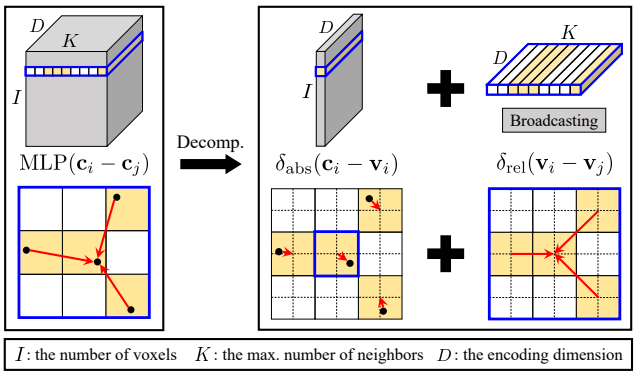


Figure 3. **Decomposition of relative position.** Note that we use the continuous positional encoding $\delta_{\text{abs}}(\mathbf{c}_i - \mathbf{v}_i)$ to transform the input voxel feature $\mathbf{f}_i$ to the *centroid-aware* voxel feature $\mathbf{g}_i$.

Given, Eq. (4) and (5), we see that local self-attention uses 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 a *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

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

We illustrate the reduction of the space complexity in Figure 3, and evaluate the effectiveness of the decomposition in Table A4 and Table A5 of the supplementary material.

Lightweight self-attention layer. Now, we propose the new local self-attention layer, named LSA layer, by defining attention function $a(\cdot)$ in Eq. (6) as

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

It is worth noting that the LSA layer uses the *cosine similarity* between $\phi(\mathbf{g}_i)$ and $\delta_{\text{rel}}(\mathbf{v}_i - \mathbf{v}_j)$. Instead of using $\text{softmax}(\phi(\mathbf{g}_i)^\top \delta_{\text{rel}}(\mathbf{v}_i - \mathbf{v}_j))$, cosine similarity can effectively handle the sparsity issue of input voxels $\mathcal{V}$ properly. For 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 LSA layer to be a simple linear layer ψ. In addition, as the LSA 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 $\text{softmax}(\cdot)$ as shown in Table 6.

The dynamics of the LSA layer (Eq. (7)) generates weights using the *centroid-aware* features $\phi(\mathbf{g}_i)$ and relative voxel features $\delta_{\text{rel}}(\mathbf{v}_i - \mathbf{v}_j)$. This design enables LSA

layer to learn more coherent representation under the rigid transformations than sparse convolution based approach [6], 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, Table 3, and Table 8. We also experimentally show that the reformulation from Eq. (4) to Eq. (6) works reasonably (as shown in Table 5 and Table 6) and introduces extra efficiency (as shown in Table 2).

3.4. Network Architecture

We develop Fast Point Transformer for dense prediction on point cloud based on the modules introduced above. Using coordinate hashing (Sec. 3.2) and decomposed positional encodings (Sec. 3.3), Fast Point Transformer is less prone to quantization errors than previous voxel-based methods [6, 10, 23], while also being significantly faster than point-based methods [41, 49] in terms of both space and time. Furthermore, the proposed local self-attention layer can be easily be *integrated to voxel-based downsampling and upsampling layer* without introducing heuristic sampling and grouping mechanisms that are often used in the point-based methods [27, 41, 49]. Note that we can build local self-attention networks by substituting convolution layers with LSA layers. Therefore, any sparse CNN architecture can be modified to faciliate local self-attention, *e.g.*, ResNet [14] and U-Net [30]. We implement our model for semantic segmentation using the U-Net [30] architecture. Further details are described in the supplementary material.

4. Experiments

In this section, we evaluate our model on two popular large-scale 3D scene datasets: S3DIS [2] and ScanNet [7]. We have selected the two datasets due to their rich diversity and densely annotated labels. We first validate the 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. 4.5. Specifically, we provide stochastic numbers averaged from three different experiments with the same training configuration except random seed numbers for the comparison tables: Table 1, Table 2, Table 3, Table 4, and Table 8.

4.1. Datasets

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 [6], we do not use any preprocessing methods, *e.g.*, *cropping into small blocks*, that are widely used in point-based methods [17, 18, 26, 34, 35, 41].

ScanNet. We use the second official release of ScanNet [7], which consists of 1.5k room scenes with some rooms captured repeatedly with different sensors. Following the experimental settings of prior work [4, 25], our model uses point-wise RGB colors as input point features $\{\mathbf{i}_n\}$ both for 3D semantic segmentation task and 3D objection detection.

4.2. Baselines

We have selected PointNet [26], PointWeb [48], SPGraph [17], PointConv [40], PointASNL [42], KPConv [37], PAConv [41], Point Transformer [49], SparseConvNet [10], and MinkowskiNet [6] as the baseline approaches. MinkowskiNet32 and MinkowskiNet42 [6] are compared as representative voxel-based methods that comprise 32 and 42 U-Net layers, respectively. We reproduce MinkowskiNet42 [6] with the official source code and denote it as MinkowskiNet42†, with different voxel sizes. PointNet [26], SPGraph [17], PointWeb [48], KPConv [37], PAConv [41] and Point Transformer [49] are selected since they are representative point-based methods. The main difference between KPConv [37] and the others is that KPConv [37] uses a k-d tree to boost its inference time while the others do not. We follow the official guideline of the methods and reproduce the results. A more recent method, Point Transformer [49] has also been selected due to its superiority on several datasets. Unlike our method and selected baselines, other approaches [5, 15, 16] 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 ($\text{CScore}(f; \mathcal{S}, \mathcal{T})$) as follows:

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

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 $\mathcal{P}$ are vulnerable to $\mathcal{T}$. We apply 41 different rigid transformations that consist of 26 translations and 15 rotations around the gravity axis. For the voxel size L, 26 translations are set to $[0, L/3, 2L/3]^3$ except zero translation $[0, 0, 0]$. Fifteen rotation angles along gravity axis is set to $[0.125\pi, 0.25\pi, \cdots, 1.875\pi]$. We evaluate CScore of MinkowskiNet42 and Fast Point Transformer on the ScanNet validation split. The evaluation results (Table 1) and

Table 1. **Comparison of consistency score (CScore) and mIoU.** We compare the consistency scores of Fast Point Transformer and MinkowskiNet42†, which is the reproduced model, on different transformation sets. The transformation sets are 1) rotation only ($\mathbf{R}$), 2) translation only ($\mathbf{t}$), and 3) both ($\mathbf{R}$ and $\mathbf{t}$). The size of voxel is set to 10cm, 5cm, and 2cm for 3D semantic segmentation on the ScanNet validation dataset [7]. Fast Point Transformer reduces the prediction inconsistency that occurred by voxelization artifact.

Method	CScore (%)			mIoU (%)
	$\mathbf{R}$	$\mathbf{t}$	$\mathbf{R}$ and $\mathbf{t}$	
*Voxel size: 10*cm				
MinkowskiNet42†	92.2±0.1	92.0±0.1	92.0±0.1	60.5±0.2
FastPointTr. (ours)	94.7±0.3	94.6±0.1	94.6±0.1	65.9±0.6
*Voxel size: 5*cm				
MinkowskiNet42†	94.2±0.1	95.1±0.1	94.8±0.1	66.7±0.2
FastPointTr. (ours)	95.9±0.4	96.4±0.1	96.2±0.2	70.0±0.1
*Voxel size: 2*cm				
MinkowskiNet42†	95.9±0.6	96.9±0.3	96.6±0.1	71.9±0.2
FastPointTr. (ours)	**96.9**±0.3	**97.4**±0.4	**97.2**±0.1	**72.1**±0.3

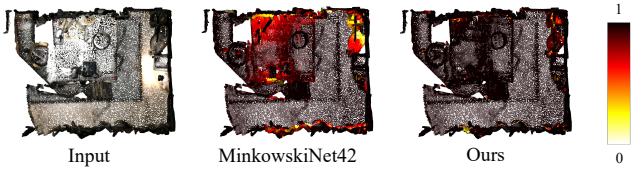


Figure 4. **Heatmap visualization of consistency score (CScore).** We visualize consistency scores of MinkowskiNet [6] and the proposed Fast Point Transformer with the hot heatmap. Points with high CScore (consistently predicted with the same class) are colored black, and points with low CScore (the predicted class is not consistent with arbitrary rigid transformations) are colored white. Table 1 shows the quantitative evaluation.

the qualitative results (Figure 4) show that Fast Point Transformer outputs more coherent feature representations than MinkowskiNet42 [6]. Moreover, the coherent predictions indicate that the Fast Point Transformer successfully relieves quantization artifacts.

4.4. 3D Semantic Segmentation

We compare our approach with the state of the art in 3D semantic segmentation on S3DIS [2] and ScanNet [7]. We use the mean of class-wise IoU scores as the primary evaluation metric for both datasets.

S3DIS. We compare the computational complexity, the mean accuracy, and the mean IoU of Fast Point Transformer with the state of the arts on the S3DIS Area 5 test split. Since Choy et al. [6] reported results with a lightweight network (MinkowskiNet32), we utilize the official code of MinkowskiNet42 and reproduce the results denoted by MinkowskiNet42† with voxel size 4cm. We also provide the performance of MinkowskiNet42† and Fast Point Transformer with voxel size 5cm in the supplementary material.

Table 2 theoretically analyzes the time complexity and reports the average wall-time latency of each method when processing S3DIS Area 5 scenes. We measure the inference time of MinkowskiNet42†, PointNet [26], SPGraph [17], PointWeb [48], KPConv [37], PAConv [41], and Point Transformer [49] using the official codes. We use the same machine with Intel(R) Core(TM) i7-5930K CPU and a single NVIDIA Geforce RTX 3090 GPU to measure the latency of methods. Detailed information about the time complexity analysis is included in the supplementary material.

Due to the preprocessing stage and stitching the multiple local predictions [17, 26, 41, 48] or multiple inferences [37, 49], the point-based methods take much more time to inference a single scene than our approach. Note that KPConv [37] constructs k-d tree, but we do not include this process into inference time. Our Fast Point Transformer processes a large-scale scene at least 83 times faster than point-based methods [17, 26, 37, 41, 48, 49] as shown in Table 2. Specifically, PointNet [26] takes 18.16 seconds for processing a scene on average because it crops the scene into $1m \times 1m \times 1m$ blocks, predicts on the blocks, and stitches the predictions for the scene-level prediction (denoted by 'Crop-and-stitch' in Table 2). Moreover, Fast Point Transformer outperforms MinkowskiNet42† by 1.4 absolute percentage score in mean IoU (%) with a comparable speed. Given the reported results by Zhao et al. [49], Point Transformer shows the best accuracy. However, Point Transformer [49] shows 129 times slower inference speed than our approach. This is because it grid-subsamples points and inferences the sampled points multiple times with the expensive k nearest neighbor search to cover the whole scene (denoted by 'Multi-shot' in Table 2), while our approach can handle the whole scene with a single feed-forward operation (denoted by 'Single-shot' in Table 2).

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 MinkowskiNet42† at voxel sizes of 2cm, 5cm, and 10cm by 0.2, 3.3, and 5.4 absolute percentage point gain in mean IoU (%) respectively. The experimental results in Table 1 and Table 3 indicate that the proposed method can represent a large-scale point cloud as features that are more robust to quantization error.

mIoU vs. model size. We compare the accuracy of both Fast Point Transformer and MinkowskiNet with the different number of parameters. We build small network models by reducing the number of building blocks as MinkowskiNet [6] does and maintaining the number of channels. Detailed illustration about network architecture is shown in the supplementary material. Table 4 shows the evaluation results.

Interestingly, we observe that Fast Point Transformer is more resilient to the network parameter reduction, and

Table 2. **3D semantic segmentation on S3DIS [2] Area 5 test.** We mark the reproduced models using the official source codes with †. We analyze the theoretical time complexity of neighbor search algorithms and evaluate the per-scene wall-time latency of each network. We denote N as the number of dataset points, M as the number of query points (or voxel centroids), and K as the number of neighbors to search. Both M and N are much larger than K in a large-scale point cloud.

Method	Neighbor Search		Large-scale Inference	Latency (Seconds)	Latency (Normalized)	mAcc (%)	mIoU (%)
	Preparation	Inference					
PointNet [26]	✗	✗	Crop-and-stitch	18.16	129.71	49.0	41.1
SPGraph [17]	✗	✗	Crop-and-stitch	18.28	130.57	66.5	58.0
PointWeb [48]	$\mathcal{O}(1)$	$\mathcal{O}(MNK)$	Crop-and-stitch	11.62	83.00	66.6	60.3
MinkowskiNet32 [6] (5cm)	$\mathcal{O}(N)$	$\mathcal{O}(M)$	Single-shot	**0.08**	**0.57**	71.7	65.4
KPConv *deform* [37]	$\mathcal{O}(N \log N)$	$\mathcal{O}(KM \log N)$	Multi-shot	105.15	751.07	72.8	67.1
PAConv [41]	$\mathcal{O}(1)$	$\mathcal{O}(MN \log K)$	Crop-and-stitch	28.13	200.93	73.0	66.6
PointTransformer [49]	$\mathcal{O}(1)$	$\mathcal{O}(MN \log K)$	Multi-shot	18.07	129.07	76.5	**70.4**
*Voxel size: 4*cm							
MinkowskiNet42†	$\mathcal{O}(N)$	$\mathcal{O}(M)$	Single-shot	**0.08**	**0.57**	74.4±0.8	67.1±0.1
+ rotation average	$\mathcal{O}(N)$	$\mathcal{O}(M)$	Multi-shot	0.66	4.71	75.0±0.7	68.4±0.1
FastPointTransformer (ours)	$\mathcal{O}(N)$	$\mathcal{O}(M)$	Single-shot	0.14	1.00	76.5±0.6	68.5±0.2
+ rotation average	$\mathcal{O}(N)$	$\mathcal{O}(M)$	Multi-shot	1.13	8.07	**77.3**±0.7	70.1±0.3

Table 3. **3D semantic segmentation on ScanNet [7] validation.** We make the reproduced models using the official codes with †.

Method	mIoU (%)
PointNet [26]	53.5
PointConv [40]	61.0
PointASNL [42]	63.5
KPConv *deform* [37]	69.2
*Voxel size: 2*cm	
SparseConvNet [10]	69.3
MinkowskiNet42 [6]	**72.2**
MinkowskiNet42†	71.9±0.2
FastPointTransformer (ours)	72.1±0.3

Table 4. **mIoU vs. model size.** Under reduced number of network parameters, Fast Point Transformer shows little performance drop while MinkowskiNet [6] gradually degrades. We color green for the positive changes and red for the negative changes w.r.t. the base model. We use ScanNet [7] validation set for the experiment.

Method	# Param. (M)		mIoU (%)	
	Rel. (%)			Δ
*Voxel size: 10*cm				
MinkowskiNet42†	37.9	±0.0	60.5±0.2	±0.0
MinkowskiNet (small)	21.7	↓ 42.7	59.9±0.6	↓ 0.6
MinkowskiNet (smaller)	11.6	↓ 69.4	58.2±0.9	↓ 2.3
FastPointTrans. (ours)	37.9	±0.0	65.9±0.6	±0.0
FastPointTrans. (small)	20.2	↓ 46.7	66.0±0.3	↑ 0.1
FastPointTrans. (smaller)	10.8	↓ 71.5	65.7±0.1	↓ 0.2
*Voxel size: 5*cm				
MinkowskiNet42†	37.9	±0.0	66.7±0.3	±0.0
MinkowskiNet (small)	21.7	↓ 42.7	66.0±0.1	↓ 0.7
MinkowskiNet (smaller)	11.6	↓ 69.4	64.2±0.4	↓ 2.5
FastPointTrans. (ours)	37.9	±0.0	70.0±0.1	±0.0
FastPointTrans. (small)	20.2	↓ 46.7	70.3±0.2	↑ 0.3
FastPointTrans. (smaller)	10.8	↓ 71.5	69.7±0.2	↓ 0.3

Fast Point Transformer models outperform their counterpart models of MinkowskiNet. We can observe that the most lightweight Fast Point Transformer with voxel size 10cm outperforms the most lightweight MinkowskiNet [6] with voxel size 5cm. MinkowskiNet [6] requires lots of parameters to overcome voxelization artifacts, whereas Fast Point Transformer shows a consistent accuracy even with 71.5% fewer network parameters.

These results imply that the proposed lightweight self-attention (LSA) layer can learn a 3D geometry more effectively than an over-parameterized sparse convolutional layer thanks to its dynamic kernel weights.

Ablation study. We conduct ablation studies on (1) the proposed positional encodings, (2) attention types, and (3) the local window size. We have followed the same setup with the main experiments with a voxel size of 10cm using a fixed random seed on ScanNet [7] validation dataset.

Table 5 shows ablation results on the proposed positional encodings, *i.e.*, δ_{enc} and δ_{abs}. 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 our network also degrades the performance. These results indicate that the two proposed voxelization and devoxelization effectively maintain continuous geometric information of the input point cloud. Moreover, the proposed positional encodings also improve the performance of MinkowskiNet42† although the total number of parameters becomes much bigger than Fast Point Transformer. However, additional usage of δ_{abs} does not improve the performance of MinkowskiNet42 [6], which

Table 5. **Ablation study on the proposed positional encodings.** Note that Mink42† and FastPointTrans. denote MinkowskiNet42† and Fast Point Transformer, respectively. We use ScanNet validation dataset [7] with voxel size 10cm.

	# Param. (M)	δ_{enc}		δ_{abs}	mIoU (%)
		Vox	Devox		
Mink42†	37.9				60.4
	38.0	✓			63.2
	38.0	✓	✓		65.1
	51.6	✓	✓	✓	65.0
FastPointTrans.	27.3				59.1
	27.3	✓			61.3
	37.8			✓	62.1
	27.3	✓	✓		62.7
	37.8	✓		✓	63.4
	37.9	✓	✓	✓	65.3

Table 6. **Ablation study on attention types.** Note that $\phi(\mathbf{g}_i)$ and $\xi(\mathbf{g}_j)$ denote a query and its neighboring key, respectively. We use ScanNet validation dataset [7] with voxel size 10cm.

$a(\cdot)$ in Eq. (6)	mIoU (%)
$\text{softmax}(\phi(\mathbf{g}_i), \delta_{\text{rel}}(\mathbf{v}_i - \mathbf{v}_j))$	61.0
$\text{cosine}(\phi(\mathbf{g}_i), \xi(\mathbf{g}_j) + \delta_{\text{rel}}(\mathbf{v}_i - \mathbf{v}_j))$	62.1
$\text{cosine}(\phi(\mathbf{g}_i), \delta_{\text{rel}}(\mathbf{v}_i - \mathbf{v}_j))$	65.3

Table 7. **Ablation study on the local window size.** Note that k is the local window size used to find the neighbors, $\mathcal{N}(i)$, in Eq. (7). We use ScanNet validation dataset [7] with voxel size 10cm.

k	Latency (sec)	mIoU (%)
3	0.106	65.3
5	0.127	62.4
7	0.168	61.9

means that the self-attention mechanism is a more proper way to use δ_{abs} than sparse convolution.

Table 6 shows the effects of attention types used in the proposed LSA layer. $\text{cosine}(\cdot)$ handles the varying number of neighbors more effectively than $\text{softmax}(\cdot)$ as shown in Table 6. However, as reported in local self-attention literature [3, 29], additional usage of the similarity between query $\phi(\mathbf{g}_i)$ and key $\xi(\mathbf{g}_j)$ does not enhance the LSA layer.

In Table 7, we show the effect of the local window size in the proposed LSA layer. Since we currently use learnable tokens for $\delta_{\text{rel}}(\mathbf{v}_i - \mathbf{v}_j)$, increasing the local window size degrades the performance due to the sparsity of 3D data. Introducing an inductive bias, such as concatenating the positional encodings [29] or a shared mapping layer [49] can be one of the possible solutions.

4.5. 3D Object Detection

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

Setups. For a fair comparison of Fast Point Transformer with previous methods [6, 27], we use Torch-Points3D, an open-source library implemented by Chaton et al. [4] for reproducible deep learning on 3D point clouds. Torch-Points3D sub-samples a fixed number of points from an input point cloud, which is widely used for PointNet++ [27] to process a scene-level point cloud-like ScanNet. We notice that the library also sub-samples points for the voxel-based methods, such as MinkowskiNet [6], which is not a suitable experimental configuration. Therefore, we reproduce VoteNet with the MinkowskiNet backbone, which is denoted by MinkowskiNet† in Table 8, without input point sub-sampling, and we use the original experimental configurations. Additionally, we train a new VoteNet [25] with the Fast Point Transformer backbone without any change of detection network (*e.g.*, voting module).

Table 8. **3D object detection on ScanNet [7] validation.** We report two mAP scores of VoteNet [25] with different backbones on ScanNet [7] dataset. Numbers except that of MinkowskiNet† and Fast Point Transformer are taken from Chaton et al. [4].

Backbone	mAP@0.25	mAP@0.50
PointNet++ [27]	54.2	30.1
RS-CNN [20]	51.6	29.5
KPConv [37]	48.9	29.2
MinkowskiNet [6]	53.8	30.2
MinkowskiNet†	55.3±0.2	33.0±0.5
FastPointTransformer (ours)	**59.1**±0.1	**35.6**±0.4

Results. As shown in Table 8, the VoteNet [25] model with Fast Point Transformer as a backbone outperforms other baselines with a large margin. The results show that the proposed continuous positional encodings that Fast Point Transformer uses can effectively encode point cloud representation and help the 3D detection task.

5. Conclusion

We have introduced the Fast Point Transformer and demonstrated its speed and accuracy on 3D semantic segmentation and 3D detection tasks. The experimental results on large-scale 3D datasets [2, 7] show that our approach is competitive to the best voxel-based method [6], and our network achieves 129 times faster inference time than the state-of-the-art, Point Transformer, with a reasonable accuracy trade-off in 3D semantic segmentation [2]. However, there is room for improvement of the Fast Point Transformer at a small voxel size. In the future, we will explore architectures for Fast Point Transformer rather than U-shaped architectures [30] that are initially designed for convolutional layers. Our code and data are going to be publicly available.

Acknowledgement. This work was supported by Qualcomm and the IITP grant (2021-0-02068: AI Innovation Hub and 2019-0-01906: AI Grad. School Prog.) funded by the Korea government (MSIT) and the NRF grant (NRF-2020R1C1C1015260).

References

[1] Dan A Alcantara, Andrei Sharf, Fatemeh Abbasinejad, Shubhabrata Sengupta, Michael Mitzenmacher, John D Owens, and Nina Amenta. Real-time parallel hashing on the gpu. In *ACM SIGGRAPH Asia 2009 papers*, pages 1–9. 2009. 1

[2] Iro Armeni, Ozan Sener, Amir R Zamir, Helen Jiang, Ioannis Brilakis, Martin Fischer, and Silvio Savarese. 3d semantic parsing of large-scale indoor spaces. In *CVPR*, pages 1534–1543, 2016. 1, 2, 5, 6, 7, 8

[3] Irwan Bello. Lambdanetworks: Modeling long-range interactions without attention. In *ICLR*, 2021. 3, 8

[4] 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 *3DV*, pages 1–10. IEEE, 2020. 5, 8

[5] Hung-Yueh Chiang, Yen-Liang Lin, Yueh-Cheng Liu, and Winston H Hsu. A unified point-based framework for 3d segmentation. In *3DV*, pages 155–163. IEEE, 2019. 5

[6] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *CVPR*, pages 3075–3084, 2019. 1, 2, 3, 5, 6, 7, 8

[7] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *CVPR*, pages 5828–5839, 2017. 2, 5, 6, 7, 8

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. 3

[9] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021. 3

[10] Benjamin Graham, Martin Engelcke, and Laurens Van Der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks. In *CVPR*, pages 9224–9232, 2018. 1, 2, 3, 5, 7

[11] Meng-Hao Guo, Jun-Xiong Cai, Zheng-Ning Liu, Tai-Jiang Mu, Ralph R. Martin, and Shi-Min Hu. Pct: Point cloud transformer. *Computational Visual Media*, 7(2):187–199, Apr 2021. 2, 3

[12] Kai Han, An Xiao, Enhua Wu, Jianyuan Guo, Chunjing Xu, and Yunhe Wang. Transformer in transformer. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *NeurIPS*, 2021. 3

[13] Lei Han, Tian Zheng, Yinheng Zhu, Lan Xu, and Lu Fang. Live semantic 3d perception for immersive augmented reality. *IEEE TVCG*, 26(5):2012–2022, 2020. 1

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 5

[15] Wenbo Hu, Hengshuang Zhao, Li Jiang, Jiaya Jia, and Tien-Tsin Wong. Bidirectional projection network for cross dimension scene understanding. In *CVPR*, pages 14373–14382, June 2021. 5

[16] Abhijit Kundu, Xiaoqi Yin, Alireza Fathi, David Ross, Brian Brewington, Thomas Funkhouser, and Caroline Pantofaru. Virtual multi-view fusion for 3d semantic segmentation. In *ECCV*, pages 518–535. Springer, 2020. 5

[17] Loic Landrieu and Martin Simonovsky. Large-scale point cloud semantic segmentation with superpoint graphs. In *CVPR*, pages 4558–4567, 2018. 5, 6, 7

[18] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on χ-transformed points. In *NeurIPS*, pages 828–838, 2018. 1, 5

[19] Yuchen Li, Qiwei Zhu, Zheng Lyu, Zhongdong Huang, and Jianling Sun. Dycuckoo: dynamic hash tables on gpus. In *ICDE*, pages 744–755. IEEE, 2021. 1

[20] Yongcheng Liu, Bin Fan, Shiming Xiang, and Chunhong Pan. Relation-shape convolutional neural network for point cloud analysis. In *CVPR*, pages 8895–8904, 2019. 8

[21] Zhijian Liu, Haotian Tang, Yujun Lin, and Song Han. Point-voxel cnn for efficient 3d deep learning. In *NeurIPS*, 2019. 2

[22] Jiageng Mao, Xiaogang Wang, and Hongsheng Li. Interpolated convolutional networks for 3d point cloud understanding. In *ICCV*, October 2019. 1, 2

[23] Jiageng Mao, Yujing Xue, Minzhe Niu, Haoyue Bai, Jiashi Feng, Xiaodan Liang, Hang Xu, and Chunjing Xu. Voxel transformer for 3d object detection. In *ICCV*, pages 3164–3173, October 2021. 1, 2, 3, 5

[24] Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Marc Stamminger. Real-time 3d reconstruction at scale using voxel hashing. *ACM TOG*, 32(6):1–11, 2013. 1

[25] Charles R Qi, Or Litany, Kaiming He, and Leonidas J Guibas. Deep hough voting for 3d object detection in point clouds. In *ICCV*, pages 9277–9286, 2019. 2, 5, 8

[26] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, pages 652–660, 2017. 1, 2, 5, 6, 7

[27] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++ deep hierarchical feature learning on point sets in a metric space. In *CVPR*, pages 5105–5114, 2017. 1, 2, 5, 8

[28] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018. 3

[29] Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jonathon Shlens. Stand-alone self-attention in vision models. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *NeurIPS*, volume 32. Curran Associates, Inc., 2019. 2, 3, 4, 8

[30] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, pages 234–241. Springer, 2015. 5, 8

[31] Radu Alexandru Rosu, Peer Schütt, Jan Quenzel, and Sven Behnke. Latticenet: Fast point cloud segmentation using permutohedral lattices. In *RSS*, 2020. 3

[32] Hang Su, Varun Jampani, Deqing Sun, Subhransu Maji, Evangelos Kalogerakis, Ming-Hsuan Yang, and Jan Kautz. Splatnet: Sparse lattice networks for point cloud processing. In *CVPR*, pages 2530–2539, 2018. 3

[33] Haotian Tang, Zhijian Liu, Shengyu Zhao, Yujun Lin, Ji Lin, Hanrui Wang, and Song Han. Searching efficient 3d

architectures with sparse point-voxel convolution. In *ECCV*, pages 685–702. Springer, 2020. 2

[34] Maxim Tatarchenko, Jaesik Park, Vladlen Koltun, and Qian-Yi Zhou. Tangent convolutions for dense prediction in 3d. In *CVPR*, pages 3887–3896, 2018. 1, 2, 5

[35] Lyne Tchapmi, Christopher Choy, Iro Armeni, JunYoung Gwak, and Silvio Savarese. Segcloud: Semantic segmentation of 3d point clouds. In *3DV*, pages 537–547. IEEE, 2017. 1, 5

[36] Matthias Teschner, Bruno Heidelberger, Matthias Müller, Danat Pomerantes, and Markus H Gross. Optimized spatial hashing for collision detection of deformable objects. In *Vmv*, volume 3, pages 47–54, 2003. 1

[37] Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *ICCV*, pages 6411–6420, 2019. 1, 2, 5, 6, 7, 8

[38] Ashish Vaswani, Prajit Ramachandran, Aravind Srinivas, Niki Parmar, Blake Hechtman, and Jonathon Shlens. Scaling local self-attention for parameter efficient visual backbones. In *CVPR*, pages 12894–12904, June 2021. 2, 3

[39] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *NeurIPS*, volume 30. Curran Associates, Inc., 2017. 3

[40] Wenxuan Wu, Zhongang Qi, and Li Fuxin. Pointconv: Deep convolutional networks on 3d point clouds. In *CVPR*, pages 9621–9630, 2019. 5, 7

[41] Mutian Xu, Runyu Ding, Hengshuang Zhao, and Xiaojuan Qi. Paconv: Position adaptive convolution with dynamic kernel assembling on point clouds. In *CVPR*, pages 3173–3182, June 2021. 1, 4, 5, 6, 7

[42] Xu Yan, Chaoda Zheng, Zhen Li, Sheng Wang, and Shuguang Cui. Pointasnl: Robust point clouds processing using nonlocal neural networks with adaptive sampling. In *CVPR*, pages 5589–5598, 2020. 5, 7

[43] Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Zi-Hang Jiang, Francis E.H. Tay, Jiashi Feng, and Shuicheng Yan. Tokens-to-token vit: Training vision transformers from scratch on imagenet. In *ICCV*, pages 558–567, October 2021. 3

[44] Cheng Zhang, Haocheng Wan, Shengqiang Liu, Xinyi Shen, and Zizhao Wu. Pvt: Point-voxel transformer for 3d deep learning. *arXiv preprint arXiv:2108.06076*, 2021. 2

[45] Feihu Zhang, Jin Fang, Benjamin Wah, and Philip Torr. Deep fusionnet for point cloud semantic segmentation. In *ECCV*, volume 2, page 6, 2020. 2, 3

[46] Richard Zhang. Making convolutional networks shift-invariant again. In *ICML*, pages 7324–7334. PMLR, 2019. 2

[47] Hengshuang Zhao, Jiaya Jia, and Vladlen Koltun. Exploring self-attention for image recognition. In *CVPR*, June 2020. 4

[48] Hengshuang Zhao, Li Jiang, Chi-Wing Fu, and Jiaya Jia. Pointweb: Enhancing local neighborhood features for point cloud processing. In *CVPR*, pages 5565–5573, 2019. 5, 6, 7

[49] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip H.S. Torr, and Vladlen Koltun. Point transformer. In *ICCV*, pages 16259–16268, October 2021. 1, 2, 3, 4, 5, 6, 7, 8

[50] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *CVPR*, pages 4490–4499, 2018. 1

[51] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable {detr}: Deformable transformers for end-to-end object detection. In *ICLR*, 2021. 4