LitePT: Lighter Yet Stronger Point Transformer

Yuanwen Yue[1,2] Damien Robert[3] Jianyuan Wang[2] Sunghwan Hong[1] Jan Dirk Wegner[3]
Christian Rupprecht[2] Konrad Schindler[1]

[1]ETH Zurich [2]University of Oxford [3]University of Zurich

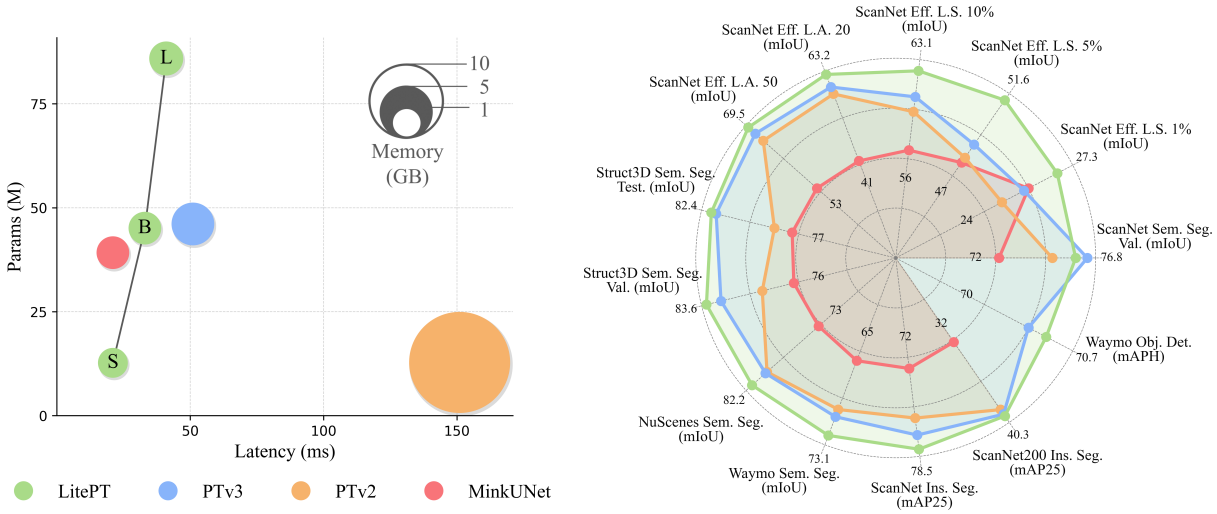


Figure 1. **LitePT is a lightweight, high-performance 3D point cloud architecture.** Left: LitePT-S has 3.6× fewer parameters, 2× faster runtime and 2× lower memory footprint than the state-of-the-art Point Transformer V3, and is even more memory-efficient than classical convolutional backbones. Moreover, it remains fast and memory-efficient even when scaled up to 86M parameters (LitePT-L). Right: Already the smallest variant, LitePT-S, matches or outperforms state-of-the-art point cloud backbones across a range of benchmarks.

Abstract

Modern neural architectures for 3D point cloud processing contain both convolutional layers and attention blocks, but the best way to assemble them remains unclear. We analyse the role of different computational blocks in 3D point cloud networks and find an intuitive behaviour: convolution is adequate to extract low-level geometry at high-resolution in early layers, where attention is expensive without bringing any benefits; attention captures high-level semantics and context in low-resolution, deep layers more efficiently. Guided by this design principle, we propose a new, improved 3D point cloud backbone that employs convolutions in early stages and switches to attention for deeper layers. To avoid the loss of spatial layout information when discarding redundant convolution layers, we introduce a novel, parameter-free 3D positional encoding, PointROPE. The resulting LitePT model has 3.6× fewer parameters, runs 2× faster, and uses 2× less memory than the state-of-the-art Point Transformer V3, but nonetheless matches or outperforms it on a range of tasks and datasets. Code and models are available at: https://github.com/prs-eth/LitePT.

1. Introduction

Visual understanding of 3D point clouds is central to a wide range of applications, including robotics [3, 77, 79, 89], autonomous driving [18, 59], localisation [39], mapping [45, 66, 68], and environmental monitoring [29, 55]. A variety of deep learning architectures and neural processing layers for unstructured point clouds have been proposed, yet the field still lacks a detailed understanding of their relative strengths and weaknesses, and principled guidelines on how to most efficiently combine them into versatile, high-performance architectures.

Lately, Transformer-based models have dominated 3D benchmarks. In particular, their most recent incarnation *Point Transformer V3* (PTv3) [75] has been shown to out-

perform earlier sparse convolutional [10, 19] and attention-based models [22, 74, 90], and is considered the state of the art. Importantly, PTv3 is in fact *not* a pure Transformer architecture: 67% of its parameters are allocated to (residual) sparse convolution layers. These are interleaved with the Transformer-style attention+MLP blocks and, among others, serve as a form of positional encoding. That design, with both convolution and attention operations at all hierarchy levels (resp., depths) of a U-net-like encoder-decoder scheme [51], is common in modern 3D point cloud architectures, which naturally leads to the question: *what are the respective roles of convolution and attention?*

Here, we analyse the contribution and interplay of these layers in more detail. We find a clear division of labour along the feature hierarchy. Early, high-resolution stages are dominated by the encoding of *local* geometry. Convolution or attention perform similarly well for that purpose, as the locality of convolutions is the right inductive bias. However, attention is substantially more expensive for early layers with high spatial resolutions (*i.e.*, a large number of tokens). Later, at lower-resolution stages, semantics and global context emerge. To capture the associated long-range interactions, the highly expressive attention mechanism is more suitable and also more parameter-efficient. As mentioned, in PTv3 and related architectures, the SparseConv [19] layer was primarily included to encode positional information. It turns out that, for that particular purpose, convolution is a possible solution, but not a necessity. We find that a ROPE-inspired [58] query-key positional encoding, which we call PointROPE, fulfills the role more effectively, while being more efficient and introducing no learnable parameters. Overall, our analysis points to a clear design principle: apply convolution when the focus is on local geometry, and attention when reasoning about semantics and global layout.

Building on these insights, we design LitePT, a hybrid network architecture for 3D point cloud analysis that leverages the computational tools in the most efficient manner; *i.e.*, sparse convolutions in the early stages and PointROPE-enhanced attention in the later stages. By tailoring the information processing to the level of abstraction, LitePT requires $3.6\times$ fewer parameters than PTv3. Our architecture cuts memory consumption by 60.3% during training and by 51.2% during inference, and reduces latency by 34.5% during training and by 58.8% during inference. Remarkably, LitePT also improves performance compared to PTv3 across a range of benchmarks on 3D semantic segmentation, 3D instance segmentation, and 3D object detection.

2. Related Work

In line with the purpose of LitePT, we review deep learning-based point cloud representations, with a specific focus on Transformer architectures and hybrid approaches.

Deep Point Cloud Understanding. To take advantage of mature image-based networks, early approaches used to project 3D point clouds into 2D image planes and then leverage standard 2D CNNs to extract features [2, 7, 31, 35, 57, 70]. These projection-based methods tend to work well only when several implicit assumptions are met, *e.g.*, relatively uniform point density, sufficient coverage, opaque surfaces, *etc.* Voxel-based methods transform irregular point clouds to regular voxel grids and then apply 3D convolution operations [23, 28, 37, 41, 56]. However, voxel representations are both computationally expensive and memory-intensive, motivating follow-up works to develop efficient sparse convolution frameworks [8, 10, 19, 44, 61]. Instead of projecting or quantising irregular point clouds into regular grids in 2D or 3D, point-based methods design operators that work directly on raw point coordinates, better preserving geometric information. Point operators have progressed from early MLP-based designs [14, 40, 46–48, 87] to point convolutions [1, 20, 27, 36, 62, 72, 80], graph-based networks [34, 69], and, more recently, attention-based mechanisms [5, 22, 49, 50, 64, 74, 75, 90]. Among modern point cloud backbones, Transformer-based architectures represent the state of the art.

Point Cloud Transformers. Transformer-based architectures employ the attention mechanism as their core feature extractor. To mitigate the quadratic complexity of global self-attention, most approaches adopt some form of windowed attention, restricted to a local spatial neighbourhood. Point cloud Transformers mainly differ in how these localised attention patches are constructed to best balance performance and efficiency. Common strategies include k-nearest neighbour search [74, 85, 90], window or voxel partitioning [17, 38, 43, 60, 67, 83, 84, 88], superpoints [49, 50], and 1D sorting with space-filling curves [6, 75]. Such local attention mechanisms are often integrated with shifted patch grouping [84] and hierarchical architectures in the spirit of U-Net [51], so as to aggregate global context. Existing works typically apply attention at all stages of the hierarchical network. We argue that attention in shallow stages, where the number of tokens is large and local patterns dominate, is computationally inefficient and unnecessary, as seen in Secs. 3.1 and 4.1.

Positional Encoding in Point Cloud Transformers. Attention does not take into account spatial layout; therefore, positional encoding plays an important role in Transformers. PTv1 [90] and PTv2 [74] employ relative positional encoding (RPE), where an MLP encodes relative positions between points. Stratified Transformer [32] and Swin3D [84] use contextual relative positional encoding (cRPE), which maintains three learnable look-up tables for the (x, y, z) axes that are computationally rather inefficient. OctFormer [67] and PTv3 [75] employ conditional positional encoding (CPE) [11], which is implemented via a

convolutional layer preceding each attention module. CPE improves efficiency, but introduces a substantial number of learnable parameters. Here, we adapt rotary positional embedding (RoPE) [58] to point cloud learning, a parameter-free module that offers both efficiency and strong empirical performance.

Hybrid Models. Convolution is by design capable of capturing local features, whereas Transformers excel at modelling long-range dependencies. In the vision domain, since the introduction of the Vision Transformer [15], numerous studies have explored the integration of convolutional operators with attention for efficient image analysis [13, 21, 42, 65, 71, 81]. Similarly, in the 3D point cloud field, several works have investigated hybrid architectures that combine the strengths of convolution and attention. DyCo3D [25] augments Sparse U-Net with a bottleneck Transformer to capture long-range context. Stratified Transformer [32] reports that a KPConv [62] block provides substantially stronger local features than attention. Superpoint Transformer [49] leverages a lightweight PointNet [46] to encode geometrically-homogeneous superpoints. PointConvFormer [73] and KPConvX [63] augment convolution kernels with attention to improve feature modelling. ConDaFormer [16] adds two sparse convolution blocks before and after each attention module to better capture local structure. We note that PTv3 [75] is also arguably a hybrid model, as it utilizes sparse convolutions as positional encoding, which account for the majority of its trainable parameters. Another common design paradigm applies a sparse U-Net for feature extraction followed by a task-specific Transformer decoder [53, 82]. In contrast to prior works, which typically employ a uniform hybrid block repeated across the hierarchy, we rethink hybrid design from a multi-scale perspective and decouple convolution and attention, allowing for the selective use of each at different hierarchy levels to exploit their complementary advantages.

3. Methodology

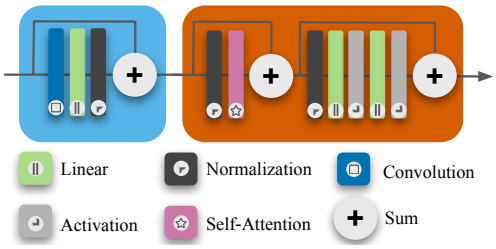


Figure 2. **PTv3 block.** The block is composed of a convolutional conditional positional encoding module followed by an attention module.

To motivate our network design, we begin with an empirical study that investigates the respective roles of convolution and attention in PTv3 [75]. We then introduce the components of LitePT: computational blocks that are reduced to the essentials and tailored to different processing stages (Sec. 3.2); and an alternative, learning-free positional encoding for the simplified blocks (Sec. 3.3). Finally, we describe the overall architecture in Sec. 3.4.

3.1. Revisiting PTv3: Convolution vs. Attention

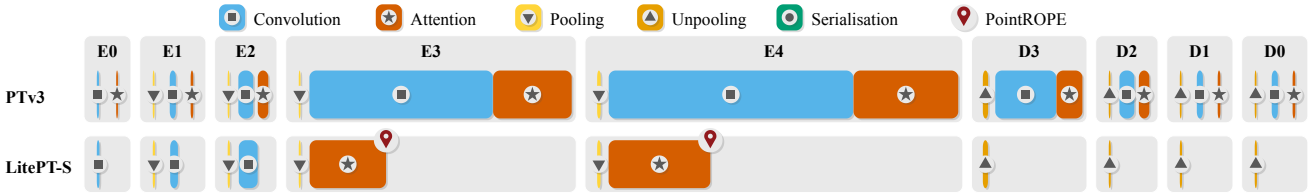
Preliminaries. PTv3 [75] represents the current state-of-the-art architecture for point cloud understanding. Similar to earlier point cloud backbones [10, 49, 74, 90], it adopts a U-Net architecture [51] composed of multiple encoder and decoder stages with skip connections. Between consecutive encoding (or decoding) stages, pooling (or unpooling) operations are applied to downsample (or upsample) the point cloud and its associated features. Each encoder and decoder stage consists of several blocks. Fig. 2 depicts a single block as used in PTv3, consisting of a convolutional positional encoding module ■ and an attention module ■. Inspired by [11], PTv3 adopts conditional positional encoding, implemented by prepending a sparse convolution layer, a linear projection, and a LayerNorm, with a skip connection, before each attention module. The attention module follows a standard pre-norm structure [78], where self-attention is applied between local groups of points obtained via serialisation sorting, followed by a multilayer perceptron (MLP).

Conditional positional encoding, and in particular its sparse convolution layer, has proved to be an important part of the overall architecture, but its precise role remains somewhat unclear. Does it indeed just serve to encode the spatial layout of the tokens that flow through the attention layer, or does it actually act as a local feature extractor in the spirit of classical convolutional networks? In the following, we analyse the parameter efficiency and the computational cost of different components along the U-Net hierarchy, revealing striking differences between the stages.

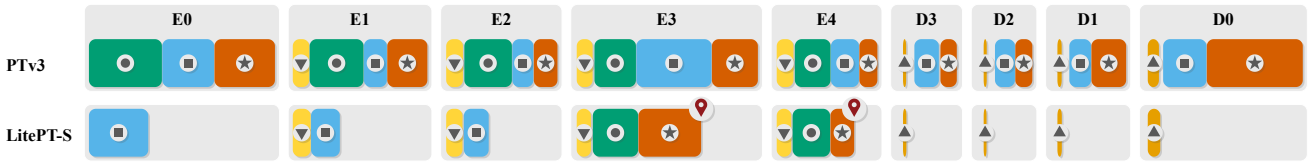
	Model	#Params	ScanNet [12] mIoU	nuScenes [4] mIoU
	PTv3 [75]	46.1M	77.5	80.4
①	PTv3 *w/o* Transformer	32.4M	73.4	76.1
②	PTv3 *w/o* SPConv	15.4M	70.7	74.9

Table 1. **Revisiting PTv3.** We evaluate two PTv3 variants: in ①, the attention and MLP modules are removed, and in ②, only the sparse convolution layers are removed.

Number of parameters. An often overlooked, yet important fact is that 67% of the total parameter budget in PTv3 is spent on the sparse convolution layers of the positional encoding, while the Transformer part (*i.e.*, attention and MLP) only accounts for 30% of the learnable parameters. Furthermore, the parameter count of the sparse convolution layers increases substantially with depth and is largest near the bottleneck, due to the high feature dimension of the late encoder and early decoder stages. See Fig. 3a.



(a) Breakdown of trainable parameters



(b) Breakdown of latencies

Figure 3. **Parameter count and latency.** E0-E4 denote encoder stages from shallow to deep, and D3-D0 denote decoder stages from deep to shallow. The length of each bar reflects the relative parameter count or latency of the corresponding module. Top: In PTv3, the positional encoding implemented via a convolution block accounts for the majority of its parameters, particularly in the later stages. In contrast, our PointROPE is parameter-free. Bottom: The PTv3 latency map reveals the significant cost of early-stage attention. LitePT restricts attention to late stages, where it is most effective and less costly.

Latency. Fig. 3b graphically depicts the computational latency of attention and convolution across different network stages. Attention, with its quadratic computational complexity, accounts for the majority of the computational cost. Importantly, that cost decreases as one progresses towards deeper stages near the bottleneck, because hierarchical downsampling quadratically reduces the number of point tokens.

Convolution vs. attention. So far, we have clarified that convolution accounts for the majority of trainable parameters, whereas attention dominates the computational cost, and that both vary strongly along the U-Net hierarchy. To separate the contributions of the two modules, we design two reduced variants of the PTv3 block. In the first one, we remove the attention modules. Using exclusively this variant degenerates to a classical sparse U-Net structure [10, 19]. In the second variant, we remove only the sparse convolution layer to obtain a "pure" Transformer. Table 1 contrasts the semantic segmentation performance of the two variants for ScanNet [12] and nuScenes [4]. It turns out that removing convolutions causes a larger performance drop than removing the attention modules, suggesting that the "positional encoding" actually does much of the heavy lifting. We visualise the learnt embeddings at each encoding stage for the three variants using PCA (Fig. 4) and find that a distinct division of labour emerges along the hierarchy, regardless of whether convolution, attention, or both are used. Early stages primarily encode local geometry, later stages capture high-level semantics.

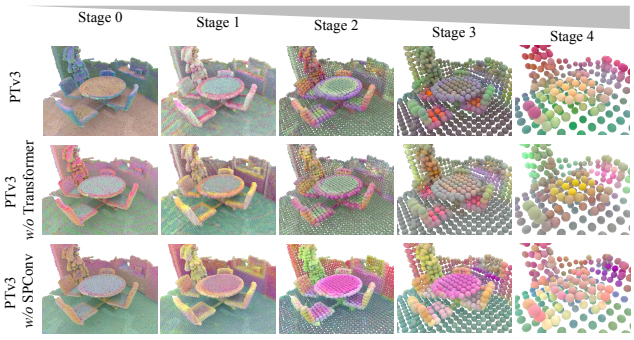


Figure 4. **Representations learnt by the hierarchical U-Net encoder.** The hierarchical U-Net encoder exhibits an operator-agnostic feature hierarchy: shallow stages consistently encode local geometric structure, while semantics emerge in deeper stages.

Discussion. The above analysis leads us to the following hypotheses:

1. It may not be necessary to use both convolution *and* attention at every stage. In the early stages, which prioritise local feature extraction, convolution is adequate. In deep stages, where the focus is on long-range context and semantic concepts, attention is key.
2. It would be a sweet spot in terms of efficiency if one could indeed avoid attention at early stages, where it is most expensive, and convolution at late stages, where it inflates the parameter count.
3. Pure attention blocks will require an alternative positional encoding—but storing spatial layout is apparently *not* the main function of the convolution, so a more parameter-efficient replacement should be possible.

3.2. Tailored Blocks for Different Network Stages

Driven by the insights from the study described above, we propose a simple yet effective design that retains only the essential operations in each stage. Convolutions are allocated to earlier stages with high spatial resolution and low

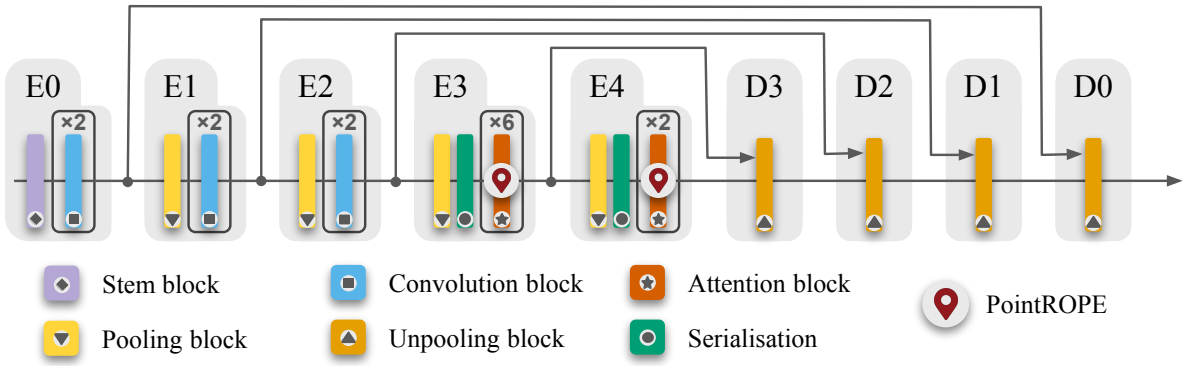


Figure 5. **LitePT-S architecture.** Our model comprises five stages, employing convolution blocks in the early stages and PointROPE augmented attention blocks in the later ones. LitePT-S uses a lightweight decoder. Alternatively, adding convolution or attention blocks symmetrically in the decoder produces LitePT-S*.

channel depth, and attention is reserved for deep stages with only few, but high-dimensional tokens.

Formally, let the hierarchical encoder consist of L stages, where the i-th stage transforms the feature representation f_{i-1} into f_i via a function $\mathcal{B}_i(\cdot)$:

$$f_i = \mathcal{B}_i(f_{i-1}), \quad i = 1, ..., L \tag{1}$$

Depending on the stage index, each block $\mathcal{B}_i$ is instantiated as either pure convolution or pure attention:

$$\mathcal{B}_i = \begin{cases} \text{ConvBlock}_i, & \text{if } i \leq L_c \\ \text{AttnBlock}_i, & \text{if } i > L_c \end{cases} \tag{2}$$

Early stages ($i \leq L_c$) operate on point sets with high spatial resolution and density, where local geometric reasoning is critical. Employing convolution layers in these stages efficiently aggregates information over local receptive fields, with minimal parameter overhead. As one progresses to deeper stages ($i > L_c$), the number of point tokens is greatly reduced and semantic abstraction becomes more important, hence one switches to attention-based blocks. Optionally, one can also include a "hand-over" stage i with both ConvBlock_i and AttnBlock_i. See ablation studies in Sec. 4.1. More gradual transitions between the two mechanisms are, in principle, possible, but unnecessarily complicate the design.

Our LitePT follows a different philosophy than PTv3 and other hybrid point cloud Transformers: [16, 63, 73] all uniformly repeat the same computational block at all stages; as a consequence, that unit must include both attention and convolution. In contrast, we prefer to simplify individual blocks as much as possible, which then requires different forms of simplification depending on the network stage. Empirically, we find that strategically distributing custom blocks along the hierarchy yields higher performance with significantly lower memory footprint and computational cost.

3.3. Point Rotary Positional Embedding

Discarding the expensive convolution layer at deep hierarchy levels has an undesired side effect: one loses the positional encoding. Hence, a more parameter-efficient replacement is needed.

Rotary Positional Embedding (RoPE) [58] has proven to be remarkably effective in natural language processing. In RoPE, relative positional awareness is introduced into the attention mechanism through rotations of the feature space. Originally, the method is designed for 1D sequence data. It does not have a direct generalisation to irregular point clouds in 3D point space.

We adapt RoPE to 3D in a straightforward manner to obtain Point Rotary Positional Embedding (PointROPE). Given a point feature vector $\mathbf{f}_i \in \mathbb{R}^d$ at position $\mathbf{p}_i = (x_i, y_i, z_i)$, we divide the embedding dimension d into three equal subspaces corresponding to the x, y, and z axes:

$$\mathbf{f}_i = [\mathbf{f}_i^x; \mathbf{f}_i^y; \mathbf{f}_i^z], \qquad \mathbf{f}_i^x, \mathbf{f}_i^y, \mathbf{f}_i^z \in \mathbb{R}^{d/3}\,. \tag{3}$$

We then independently apply the standard 1D RoPE embedding to each subspace, using the respective point coordinate, and concatenate the axis-wise embeddings to form the final point representation:

$$\tilde{\mathbf{f}}_i = \begin{bmatrix} \tilde{\mathbf{f}}_i^x \\ \tilde{\mathbf{f}}_i^y \\ \tilde{\mathbf{f}}_i^z \end{bmatrix} = \begin{bmatrix} \text{RoPE}_{1D}(\mathbf{f}_i^x, x_i) \\ \text{RoPE}_{1D}(\mathbf{f}_i^y, y_i) \\ \text{RoPE}_{1D}(\mathbf{f}_i^z, z_i) \end{bmatrix}\,. \tag{4}$$

For each point with coordinates (x_i, y_i, z_i), we directly use its grid coordinates as input, which are already correctly scaled during the pooling operation.

The embedding scheme preserves the directional separability of 3D points while jointly encoding the feature's positional phase rotation, effectively capturing relative geometry. Compared to the learned convolutional positional encoding of PTv3 [75], PointROPE is parameter-free, lightweight, and, by construction, rotation-friendly. As part

of our open source code, we provide an optimised CUDA implementation.

3.4. Architecture

Our model follows the conventional U-Net [51] structure, with five stages. We build three variants of the encoder, with varying number C of channels in each stage and B blocks per stage. Note that C must be divisible by 6 in stages that include PointROPE.

LitePT-S: $C = (36, 72, 144, 252, 504), B = (2, 2, 2, 6, 2)$
LitePT-B: $C = (54, 108, 216, 432, 576), B = (3, 3, 3, 12, 3)$
LitePT-L: $C = (72, 144, 288, 576, 864), B = (3, 3, 3, 12, 3)$

We use LitePT-S as the main variant for the experiments, since it already delivers excellent performance across all benchmarks. Model scaling is examined in Tab. 5. Per default, we set $L_c = 3$, meaning that stages 1, 2, 3 use ConvBlock$_i$, while stages 4, 5 use AttnBlock$_i$. Each ConvBlock$_i$ consists of a sparse convolution layer, a linear layer and LayerNorm, and has a residual connection. Each AttnBlock$_i$ consists of a PointROPE embedding followed by attention, where the latter is computed locally within groups of points, found with the same serialisation sorting as in PTv3 [75]. For semantic segmentation, we simplify the decoder to only the linear projection layer and LayerNorm in each stage. For instance segmentation, we apply the stage-specific design also in the decoder and symmetrically assign ConvBlock$_i$ and AttnBlock$_i$, in reverse order of the encoder.

4. Experiments

Method	#Params	Training		Inference	
		Latency	Memory	Latency	Memory
MinkUNet [10]	39.2M	60ms	1.9G	21ms	2.4G
PTv2 [74]	12.8M	188ms	22.8G	151ms	22.9G
PTv3 [75]	46.1M	110ms	5.8G	51ms	4.1G
LitePT-S (Ours)	12.7M	72ms	2.3G	21ms	2.0G
LitePT-S* (Ours)	16.0M	81ms	3.3G	26ms	2.0G
LitePT-B (Ours)	45.1M	93ms	5.5G	33ms	2.4G
LitePT-L (Ours)	85.9M	97ms	8.4G	41ms	2.6G

Table 2. **Efficiency comparison.** Results are reported as average over the full ScanNet dataset using a single RTX 4090 GPU. Automatic Mixed Precision (AMP) is enabled for all models during training and disabled during inference. * denotes our variant with a heavier decoder that includes attention or convolutional blocks.

We begin with a series of ablation studies to analyse different configurations of our hybrid design, the model's scaling behaviour, and PointROPE (Sec. 4.1). We then present comparisons with state-of-the-art methods for 3D semantic segmentation (Sec. 4.2), 3D instance segmentation (Sec. 4.3) and 3D object detection (Sec. 4.4).

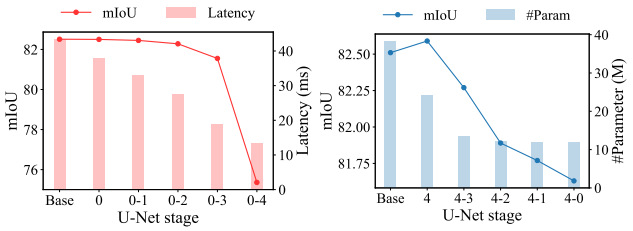


Figure 6. **Performance-efficiency trade off.** Left: Progressively dropping attention in more of the early stages. Right: Progressively dropping convolution in more of the late stages.

4.1. Ablation Studies and Analysis

Are both convolution and attention needed at every stage? To verify our first hypothesis from Sec. 3.1, we design two sets of experiments on nuScenes. We begin with a baseline model that incorporates both convolution and PointROPE attention at all stages. In Experiment 1, we progressively remove *attention*, first from stage 0, then from stages 0 and 1, *etc.* In Experiment 2, we progressively remove *convolution*, first from stage 4, then from stages 4 and 3, *etc.* We then plot the mIoU of those configurations against latency (resp. parameter count).

As shown in Fig. 6 (*left*), removing attention in early stages boosts efficiency with almost no drop in mIoU, whereas removing attention in later stages harms performance. On the other hand, Fig. 6 (*right*) shows that removing convolution in later stages greatly reduces the parameter count with a negligible change in mIoU, whereas removing convolution in early stages only marginally improves efficiency but adversely affects performance. The analysis confirms that one needs *not* include both convolution and attention at every stage. Their contribution and their cost highly depend on the hierarchy level.

Where is the sweet spot in terms of efficiency and performance? To determine the optimal transition point L_c between convolution and attention, we conduct an ablation study on nuScenes as shown in Tab. 3. Optionally, we include a "hand-over" stage, denoted by "X", that includes both convolution and attention. Setting $L_c = 3$, *i.e.*, convolution in the first three stages and attention in the last two, achieves the best trade-off between parameter count, latency, and mIoU. We adopt $L_c = 3$ as our default setting for all experiments.

Decoder design. The mixed design with blocks tailored to the layer depth is always used in the U-Net **en**coder. On the contrary, we propose two design variants for the U-Net **de**coder. In LitePT-S*, the same mixed design is used in the decoder, in reverse order. In LitePT-S, we further strip down the architecture and keep only a linear projection layer per stage (as needed to integrate skip connections), making the method even more efficient. We find empirically that the optimal choice is task-dependent, as shown in

L_c	Setting	#Params	Latency	mIoU
0	A-A-A-A-A	11.8M	35.1ms	82.1
1	C-A-A-A-A	11.9M	30.4ms	81.7
2	C-C-A-A-A	12.0M	25.8ms	82.0
3	C-C-C-A-A	12.7M	21.5ms	82.2
4	C-C-C-C-A	18.8M	16.2ms	80.9
5	C-C-C-C-C	26.9M	13.5ms	75.4
	C-X-A-A-A	12.2M	30.9ms	81.9
	C-C-X-A-A	13.2M	26.7ms	82.3
	C-C-C-X-A	23.4M	24.9ms	82.4

Table 3. **Effect of L_c and "hand-over" stage.** C: convolutional block; A: attention block; X: both convolution and attention are used at that stage. We compare model variants and report latency, memory usage, and validation mIoU on the nuScenes dataset. The grey-shaded row is our recommended setting.

Tab. 4. For semantic segmentation, the simple decoder is the best choice. For instance segmentation, the variant with convolution and attention blocks has a noticeable edge. We point out that even the slightly heavier LitePT-S* is still a lot more efficient than other Point Transformers (see Tab. 2), and leave the choice of decoder to the user.

	Semantic Segmentation (mIoU)				Instance Segmentation (mAP_{50})
Decoder	ScanNet [12]	Structured3D [91]	nuScenes [4]	Waymo [59]	ScanNet [12]
LitePT-S	76.5	**83.7**	**82.2**	**73.1**	62.2
LitePT-S*	**76.8**	83.0	81.8	72.7	**64.9**

Table 4. **Decoder design.** We compare two decoder variants: in LitePT-S*, we apply our stage-tailored design symmetrically to the decoder stages, while in LitePT-S, we retain only linear projection layers in all decoder stages.

Model scaling. Due to the parameter-free PointROPE encoding, our model has substantially fewer trainable weights. This offers the possibility to repurpose the saved capacity and scale up LitePT. We assess scaling behaviour on Structured3D, the largest dataset in our evaluation suite. As shown in Tab. 5, the model scales favourably: increasing the model size from LitePT-S to LitePT-L continuously improves performance, with only a modest increase in test-time latency and memory usage. Notably, even LitePT-L, with a parameter count twice that of PTv3, still runs faster than PTv3 and has a lower memory footprint.

Method	#Params	Latency	Memory	mIoU
PTv3 [75]	46.1M	57ms	5.83G	82.4
LitePT-S (Ours)	12.7M	23ms	2.56G	83.6
LitePT-B (Ours)	45.1M	36ms	2.60G	85.1
LitePT-L (Ours)	85.9M	44ms	3.58G	85.4

Table 5. **Model scaling on Structured3D dataset.** Our model scales efficiently, achieving consistent performance gains from small to large variants with modest increases in latency and memory. Even when scaled to twice the parameters of PTv3, LitePT-L remains more efficient.

PointROPE. In Tab. 6 we ablate the effectiveness of the proposed PointROPE, on nuScenes. Removing PointROPE leads to a significant performance drop of 2.6 percentage points in mIoU. We additionally ablate the base frequency d, which controls how *fast* each embedding dimension "rotates" as the position increases (uniformly for the three axes). PointROPE is fairly robust to the choice of frequency. Setting $b = 100$ yields the best score; we fix that value for all datasets to avoid excessive hyperparameter tuning.

	w/o PointROPE	$b = 10$	$b = 100$	$b = 1000$	$b = 10000$
mIoU	79.6	81.7	82.2	81.8	81.3

Table 6. **PointROPE.** Dedicated positional encoding is needed—dropping PointROPE leads to a significant performance drop. PointROPE works similarly well with a wide range of base frequencies, the grey-shaded column is our recommended setting.

4.2. Semantic Segmentation

		nuScenes [4]		Waymo [59]	
Method	#Param	mIoU	mAcc	mIoU	mAcc
MinkUNet [10]	39.2M	73.3	-	65.9	76.6
SPVNAS [61]	-	77.4	-	-	-
Cylinder3D [92]	-	76.1	-	-	-
AF2S3Net [9]	-	62.2	-	-	-
SphereFormer [33]	-	78.4	-	69.9	-
PTv2 [74]	12.8M	80.2	-	70.6	80.2
PTv3 [75]	46.1M	80.4	87.2	71.3	80.5
LitePT-S (Ours)	12.7M	**82.2**	**88.1**	**73.1**	**83.8**

Table 7. **Outdoor semantic segmentation on nuScenes and Waymo validation set.** Scores of prior work courtesy of [75, 76].

			Limited Scenes (Pct.)				Limited Annotations (Pts.)			
Method	#Params	Full	1%	5%	10%	20%	20	50	100	200
MinkUNet [10]	39.2M	72.2	26.0	47.8	56.7	62.9	41.9	53.9	62.2	65.5
PTv2 [74]	12.8M	75.4	24.8	48.1	59.8	66.3	58.4	66.1	70.3	71.2
PTv3 [75]	46.1M	**77.5**	25.8	48.9	61.0	67.0	60.1	67.9	71.4	72.7
LitePT-S (Ours)	12.7M	76.5	**27.3**	50.6	**63.1**	**67.3**	62.5	68.4	70.9	72.8
LitePT-S* (Ours)	16.0M	76.8	27.2	**51.6**	63.0	67.1	**63.2**	**69.5**	**72.0**	**74.2**

Table 8. **Indoor semantic segmentation on ScanNet validation set.** In mean IoU. Scores of prior work courtesy of [75].

		Val		Test	
Method	#Params	mIoU	mAcc	mIoU	mAcc
MinkUNet [10]	39.2M	76.4	84.3	77.4	85.5
PTv2 [74]	12.8M	79.0	86.8	78.5	86.6
PTv3 [75]	46.1M	82.4	90.3	82.1	90.3
LitePT-S (Ours)	12.7M	**83.6**	**90.7**	**82.4**	**90.3**

Table 9. **Indoor semantic segmentation on Structured3D.**

Setup. We perform semantic segmentation for four different datasets. nuScenes [4] and Waymo [59] are two outdoor datasets of first-person driving scenes, captured with vehicle-mounted LiDAR. ScanNet [12] and Structured3D [91] show indoor settings. The former was captured using an RGB-D camera. It is relatively small by today's standards, comprising 1,201 training scenes. Structured3D is a synthetic dataset and the largest public collection of 3D scenes with semantic annotations, and contains 18,348 training scenes. We follow PTv3 and use test time augmentation (TTA). Results without TTA can be found in the appendix.

Results. Tab. 7 reports semantic segmentation results on the nuScenes and Waymo validation sets. LitePT achieves marked improvements over competing architectures, in both cases +1.8 mIoU. We note that automotive LiDAR has different, more challenging properties compared with indoor datasets: the model must learn to handle massive differences in point density due to the large range, and highly anisotropic point distributions due to the scan line pattern and frequent specular reflections and ray drops.

Table 8 shows IoU scores for the ScanNet validation set. Following the literature [26], we also report results with limited training, obtained either by restricting the number of available training scenes or by reducing the number of annotated points per scene. The performance of LitePT is comparable to PTv3, which has ≈4× more parameters—in data-constrained settings, even slightly better—and clearly superior to PTv2, which has a similar parameter count. On the more than 10× larger Structured3D dataset, LitePT consistently outperforms all competing methods, including the much larger state-of-the-art PTv3.

4.3. Instance Segmentation

		ScanNet [12]			ScanNet200 [52]		
PointGroup [30]	#Params	mAP_{25}	mAP_{50}	mAP	mAP_{25}	mAP_{50}	mAP
MinkUNet [10]	39.2M	72.8	56.9	36.0	32.2	24.5	15.8
PTv2 [74]	12.8M	76.3	60.0	38.3	39.6	31.9	21.4
PTv3 [75]	46.2M	77.5	61.7	40.9	40.1	**33.2**	**23.1**
LitePT-S* (Ours)	16.0M	**78.5**	**64.9**	**41.7**	**40.3**	33.1	22.2

Table 10. **Indoor instance segmentation on ScanNet and ScanNet200 validation set.** Scores of prior work courtesy of [75].

Setup. We evaluate our method for instance segmentation on ScanNet [12] and ScanNet200 [52]. Following the protocol of prior work, we employ PointGroup [30] as instance segmentation head on top of the decoder to achieve a fair comparison.

Results. Tab. 10 summarise the results. On ScanNet, LitePT again outperforms all prior backbones and sets a new state of the art, with 64.9 mAP_{50}, a +3.2 percentage point improvement over PTv3. On ScanNet200, which includes a long tail of rare categories, the results are comparable to PTv3 and significantly better than all previous methods. For example, our method achieves 1.2% higher mAP_{50} than PTv2, which has a similar parameter count, but 11× larger memory footprint and 6× longer runtime.

4.4. Object Detection

Setup. We evaluate 3D object detection on Waymo. For a fair comparison with prior work [38, 75], we employ the same 3D object detection framework, CenterPoint-Pillar [86]. Consistent with [17, 38, 75], we avoid spatial downsampling, thus turning LitePT into a single-stage network with 8 blocks, to allow detection of small objects. Objects are divided into two difficulty levels, and we report level-2 metrics.

	Vehicle L2		Pedestrian L2		Cyclist L2		Mean L2
Method	mAP	APH	mAP	APH	mAP	APH	mAPH
PointPillars [35]	63.6	63.1	62.8	50.3	61.9	59.9	57.8
CenterPoint [86]	66.7	66.2	68.3	62.6	68.7	67.6	65.5
SST [17]	64.8	64.4	71.7	63.0	68.0	66.9	64.8
SST-Center [17]	66.6	66.2	72.4	65.0	68.9	67.6	66.3
VoxSet [24]	66.0	65.6	72.5	65.4	69.0	67.7	66.2
PillarNet [54]	70.4	69.9	71.6	64.9	67.8	66.7	67.2
FlatFormer [38]	69.0	68.6	71.5	65.3	68.6	67.5	67.2
PTv3 [75]	71.2	70.8	**76.3**	**70.4**	71.5	70.4	70.5
LitePT (Ours)	**71.6**	**71.2**	76.1	70.1	**71.8**	**70.7**	**70.7**

Table 11. **Outdoor object detection on Waymo with single frames input.** Scores of prior work courtesy of [75].

Results. Tab. 11 reports scores based on single-scan LiDAR inputs. Also in this application, LitePT reaches the highest score overall and on two out of three object categories, and comfortably matches the performance of the closest competitor, PTv3.

5. Conclusion and Discussion

We have introduced LitePT, a lighter yet stronger point Transformer for various point cloud processing tasks. Our starting point was the question, which distinct roles and impacts different operators have along the processing hierarchy. Experiments confirm that (sparse) convolutions are adequate, and more efficient, at early hierarchy levels, whereas attention comes into its own at higher levels, where semantic abstraction and global context over a comparatively small token set are key. In itself, these observations are not unexpected, but surprisingly, they have not been leveraged in contemporary point cloud architectures. LitePT embodies the simple principle "convolutions for low-level geometry, attention for high-level relations" and strategically places only the required operations at each hierarchy level, avoiding wasted computations. To achieve this, we equip our method with parameter-free PointROPE positional encoding to compensate for the loss of spatial layout information that occurs when discarding convolutional layers. We hope that LitePT will be useful as a generic high-performance backbone for 3D point cloud processing, and that our analysis can serve as practical guidance for architecture design beyond our current version.

In our architecture, attention is applied only in the later stages, where the reduced token count is small. It would therefore be affordable to compute self-attention globally across all tokens, rather than locally. In future work, it may be interesting to eliminate the local grouping operation, which could on the one hand strengthen long-range context modelling, and on the other hand further reduce the computation time at inference.

Acknowledgments. Part of the compute is supported by the Swiss AI Initiative under project a144 and a154 on Alps. We thank Xiaoyang Wu, Liyan Chen and Liyuan Zhu for their help with the comparison to PTv3. The project is supported by the Circular Bio-based Europe Joint Undertaking and its members under Grant Agreement No 101157488. Embed2Scale is co-funded by the EU Horizon Europe program under Grant Agreement No 101131841. Additional funding for this project has been provided by the Swiss State Secretariat for Education, Research and Innovation (SERI) and UK Research and Innovation (UKRI).

References

[1] Matan Atzmon, Haggai Maron, and Yaron Lipman. Point Convolutional Neural Networks by Extension Operators. *ACM Transactions on Graphics (TOG)*, 2018. 2

[2] Alexandre Boulch, Joris Guerry, Bertrand Le Saux, and Nicolas Audebert. Snapnet: 3D Point Cloud Semantic Labeling with 2D Deep Segmentation Networks. *Computers & Graphics*, 2018. 2

[3] Finn Lukas Busch, Timon Homberger, Jesús Ortega-Peimbert, Quantao Yang, and Olov Andersson. One Map to Find them All: Real-time Open-vocabulary Mapping for Zero-shot Multi-object Navigation. In *International Conference on Robotics and Automation (ICRA)*, 2025. 1

[4] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuScenes: A Multimodal Dataset for Autonomous Driving. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 3, 4, 7

[5] Liyan Chen, Gregory P Meyer, Zaiwei Zhang, Eric M Wolff, and Paul Vernaza. Flash3D: Super-scaling Point Transformers through Joint Hardware-Geometry Locality. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025. 2

[6] Wanli Chen, Xinge Zhu, Guojin Chen, and Bei Yu. Efficient Point Cloud Analysis Using Hilbert Curve. In *European Conference on Computer Vision (ECCV)*, 2022. 2

[7] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-View 3D Object Detection Network for Autonomous Driving. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2

[8] Yukang Chen, Jianhui Liu, Xiangyu Zhang, Xiaojuan Qi, and Jiaya Jia. LargeKernel3D: Scaling Up Kernels in 3D Sparse CNNs. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 2

[9] Ran Cheng, Ryan Razani, Ehsan Taghavi, Enxu Li, and Bingbing Liu. (AF)2-S3Net: Attentive Feature Fusion with Adaptive Feature Selection for Sparse Semantic Segmentation Network. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 7

[10] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4D Spatio-Temporal ConvNets: Minkowski Convolutional Neural Networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2, 3, 4, 6, 7, 8

[11] Xiangxiang Chu, Zhi Tian, Bo Zhang, Xinlong Wang, and Chunhua Shen. Conditional Positional Encodings for Vision Transformers. *International Conference on Learning Representations (ICLR)*, 2023. 2, 3

[12] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. ScanNet: Richly-Annotated 3D Reconstructions of Indoor Scenes. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 3, 4, 7, 8

[13] Zihang Dai, Hanxiao Liu, Quoc V Le, and Mingxing Tan. CoAtNet: Marrying Convolution and Attention for All Data Sizes. *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. 3

[14] Hao Deng, Kunlei Jing, Shengmei Cheng, Cheng Liu, Jiawei Ru, Jiang Bo, and Lin Wang. LinNet: Linear Network for Efficient Point Cloud Representation Learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. 2

[15] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An Image is Worth 16×16 Words: Transformers for Image Recognition at Scale. *International Conference on Learning Representations (ICLR)*, 2021. 3

[16] Lunhao Duan, Shanshan Zhao, Nan Xue, Mingming Gong, Gui-Song Xia, and Dacheng Tao. ConDaFormer: Disassembled Transformer with Local Structure Enhancement for 3D Point Cloud Understanding. *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. 3, 5

[17] Lue Fan, Ziqi Pang, Tianyuan Zhang, Yu-Xiong Wang, Hang Zhao, Feng Wang, Naiyan Wang, and Zhaoxiang Zhang. Embracing Single Stride 3D Object Detector with Sparse Transformer. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2, 8

[18] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012. 1

[19] Benjamin Graham, Martin Engelcke, and Laurens Van Der Maaten. 3D Semantic Segmentation with Submanifold Sparse Convolutional Networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2, 4

[20] Fabian Groh, Patrick Wieschollek, and Hendrik PA Lensch. Flex-Convolution. In *Asian Conference on Computer Vision (ACCV)*, 2018. 2

[21] Jianyuan Guo, Kai Han, Han Wu, Yehui Tang, Xinghao Chen, Yunhe Wang, and Chang Xu. CMT: Convolutional Neural Networks Meet Vision Transformers . In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 3

[22] 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*, 2021. 2

[23] Lei Han, Tian Zheng, Lan Xu, and Lu Fang. OccuSeg: Occupancy-aware 3D Instance Segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2

[24] Chenhang He, Ruihuang Li, Shuai Li, and Lei Zhang. Voxel Set Transformer: A Set-to-Set Approach to 3D Object Detection From Point Clouds. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 8

[25] Tong He, Chunhua Shen, and Anton Van Den Hengel. DyCo3D: Robust Instance Segmentation of 3D Point Clouds Through Dynamic Convolution. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 3

[26] Ji Hou, Benjamin Graham, Matthias Nießner, and Saining Xie. Exploring Data-Efficient 3D Scene Understanding with Contrastive Scene Contexts. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 8

[27] Binh-Son Hua, Minh-Khoi Tran, and Sai-Kit Yeung. Pointwise Convolutional Neural Networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2

[28] Jing Huang and Suya You. Point Cloud Labeling Using 3D Convolutional Neural Network. In *International Conference on Pattern Recognition (ICPR)*, 2016. 2

[29] Jakob Iglhaut, Carlos Cabo, Stefano Puliti, Livia Piermattei, James O'Connor, and Jacqueline Rosette. Structure from Motion Photogrammetry in Forestry: A Review. *Current Forestry Reports*, 2019. 1

[30] Li Jiang, Hengshuang Zhao, Shaoshuai Shi, Shu Liu, Chi-Wing Fu, and Jiaya Jia. PointGroup: Dual-Set Point Grouping for 3D Instance Segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 8

[31] Evangelos Kalogerakis, Melinos Averkiou, Subhransu Maji, and Siddhartha Chaudhuri. 3D Shape Segmentation with Projective Convolutional Networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2

[32] Xin Lai, Jianhui Liu, Li Jiang, Liwei Wang, Hengshuang Zhao, Shu Liu, Xiaojuan Qi, and Jiaya Jia. Stratified Transformer for 3D Point Cloud Segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2, 3

[33] Xin Lai, Yukang Chen, Fanbin Lu, Jianhui Liu, and Jiaya Jia. Spherical Transformer for LiDAR-Based 3D Recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 7

[34] Loic Landrieu and Martin Simonovsky. Large-Scale Point Cloud Semantic Segmentation with Superpoint Graphs. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2

[35] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. PointPillars: Fast Encoders for Object Detection from Point Clouds. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2, 8

[36] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. PointCNN: Convolution On X-Transformed Points. *Advances in Neural Information Processing Systems (NeurIPS)*, 2018. 2

[37] Zhijian Liu, Haotian Tang, Yujun Lin, and Song Han. Point-Voxel CNN for Efficient 3D Deep Learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. 2

[38] Zhijian Liu, Xinyu Yang, Haotian Tang, Shang Yang, and Song Han. FlatFormer: Flattened Window Attention for Efficient Point Cloud Transformer. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 2, 8

[39] Kan Luo, Hongshan Yu, Xieyuanli Chen, Zhengeng Yang, Jingwen Wang, Panfei Cheng, and Ajmal Mian. 3D Point Cloud-based Place Recognition: A Survey. *Artificial Intelligence Review*, 2024. 1

[40] Xu Ma, Can Qin, Haoxuan You, Haoxi Ran, and Yun Fu. Rethinking Network Design and Local Geometry in Point Cloud: A Simple Residual MLP Framework. *International Conference on Learning Representations (ICLR)*, 2022. 2

[41] Daniel Maturana and Sebastian Scherer. VoxNet: A 3D Convolutional Neural Network for Real-Time Object Recognition. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015. 2

[42] Sachin Mehta and Mohammad Rastegari. MobileViT: Light-Weight, General-Purpose, and Mobile-Friendly Vision Transformer. *International Conference on Learning Representations (ICLR)*, 2022. 3

[43] Chunghyun Park, Yoonwoo Jeong, Minsu Cho, and Jaesik Park. Fast Point Transformer. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2

[44] Bohao Peng, Xiaoyang Wu, Li Jiang, Yukang Chen, Hengshuang Zhao, Zhuotao Tian, and Jiaya Jia. OA-CNNs: Omni-Adaptive Sparse CNNs for 3D Semantic Segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 2

[45] Patrick Pfaff, Rudolph Triebel, Cyrill Stachniss, Pierre Lamon, Wolfram Burgard, and Roland Siegwart. Towards Mapping of Cities. In *International Conference on Robotics and Automation (ICRA)*, 2007. 1

[46] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2, 3

[47] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[48] Guocheng Qian, Yuchen Li, Houwen Peng, Jinjie Mai, Hasan Hammoud, Mohamed Elhoseiny, and Bernard Ghanem. PointNeXt: Revisiting PointNet++ with Improved Training and Scaling Strategies. *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. 2

[49] Damien Robert, Hugo Raguet, and Loic Landrieu. Efficient 3D Semantic Segmentation with Superpoint Transformer. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 2, 3

[50] Damien Robert, Hugo Raguet, and Loic Landrieu. Scalable 3D Panoptic Segmentation As Superpoint Graph Clustering. In *International Conference on 3D Vision (3DV)*, 2024. 2

[51] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 2015. 2, 3, 6

[52] David Rozenberszki, Or Litany, and Angela Dai. Language-Grounded Indoor 3D Semantic Segmentation in the Wild. In *European Conference on Computer Vision (ECCV)*, 2022. 8

[53] Jonas Schult, Francis Engelmann, Alexander Hermans, Or Litany, Siyu Tang, and Bastian Leibe. Mask3D: Mask Transformer for 3D Semantic Instance Segmentation. In *International Conference on Robotics and Automation (ICRA)*, 2023. 3

[54] Guangsheng Shi, Ruifeng Li, and Chao Ma. PillarNet: Real-Time and High-Performance Pillar-Based 3D Object Detection. In *European Conference on Computer Vision (ECCV)*, 2022. 8

[55] Hongli Song, Weiliang Wen, Sheng Wu, and Xinyu Guo. Comprehensive Review on 3D Point Cloud Segmentation in Plants. *Artificial Intelligence in Agriculture*, 2025. 1

[56] Shuran Song, Fisher Yu, Andy Zeng, Angel X Chang, Manolis Savva, and Thomas Funkhouser. Semantic Scene Completion from a Single Depth Image. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2

[57] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. Multi-View Convolutional Neural Networks for 3D Shape Recognition. In *International Conference on Computer Vision (ICCV)*, 2015. 2

[58] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. RoFormer: Enhanced Transformer with Rotary Position Embedding. *Neurocomputing*, 2024. 2, 3, 5

[59] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in Perception for Autonomous Driving: Waymo Open Dataset. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 1, 7

[60] Pei Sun, Mingxing Tan, Weiyue Wang, Chenxi Liu, Fei Xia, Zhaoqi Leng, and Dragomir Anguelov. SWFormer: Sparse Window Transformer for 3D Object Detection in Point Clouds. In *European Conference on Computer Vision (ECCV)*, 2022. 2

[61] 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 *European Conference on Computer Vision (ECCV)*, 2020. 2, 7

[62] 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 *International Conference on Computer Vision (ICCV)*, 2019. 2, 3

[63] Hugues Thomas, Yao-Hung Hubert Tsai, Timothy D Barfoot, and Jian Zhang. KPConvX: Modernizing Kernel Point Convolution with Kernel Attention. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 3, 5

[64] Tuan Anh Tran, Duy Minh Ho Nguyen, Hoai-Chau Tran, Michael Barz, Khoa D. Doan, Roger Wattenhofer, Vien Anh Ngo, Mathias Niepert, Daniel Sonntag, and Paul Swoboda. How Many Tokens Do 3D Point Cloud Transformer Architectures Really Need? *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. 2

[65] Zhengzhong Tu, Hossein Talebi, Han Zhang, Feng Yang, Peyman Milanfar, Alan Bovik, and Yinxiao Li. MaxViT: Multi-axis Vision Transformer. In *European Conference on Computer Vision (ECCV)*, 2022. 3

[66] Nina Varney, Vijayan K Asari, and Quinn Graehling. DALES: A Large-scale Aerial LiDAR Data Set for Semantic Segmentation. *CVPR Workshops*, 2020. 1

[67] Peng-Shuai Wang. OctFormer: Octree-based Transformers for 3D Point Clouds. *ACM Transactions on Graphics (TOG)*, 2023. 2

[68] Ruisheng Wang, Jiju Peethambaran, and Dong Chen. Lidar Point Clouds to 3-D Urban Models: A Review. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 2018. 1

[69] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic Graph CNN for Learning on Point Clouds. *ACM Transactions on Graphics (TOG)*, 2019. 2

[70] Bichen Wu, Alvin Wan, Xiangyu Yue, and Kurt Keutzer. SqueezeSeg: Convolutional Neural Nets with Recurrent CRF for Real-Time Road-Object Segmentation from 3D Lidar Point Cloud. In *International Conference on Robotics and Automation (ICRA)*, 2018. 2

[71] Haiping Wu, Bin Xiao, Noel Codella, Mengchen Liu, Xiyang Dai, Lu Yuan, and Lei Zhang. CvT: Introducing Convolutions to Vision Transformers. In *International Conference on Computer Vision (ICCV)*, 2021. 3

[72] Wenxuan Wu, Zhongang Qi, and Li Fuxin. PointConv: Deep Convolutional Networks on 3D Point Clouds. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

[73] Wenxuan Wu, Li Fuxin, and Qi Shan. PointConvFormer: Revenge of the Point-based Convolution. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 3, 5

[74] Xiaoyang Wu, Yixing Lao, Li Jiang, Xihui Liu, and Hengshuang Zhao. Point Transformer V2: Grouped Vector Attention and Partition-Based Pooling. *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. 2, 3, 6, 7, 8

[75] Xiaoyang Wu, Li Jiang, Peng-Shuai Wang, Zhijian Liu, Xihui Liu, Yu Qiao, Wanli Ouyang, Tong He, and Hengshuang Zhao. Point Transformer V3: Simpler, Faster, Stronger. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 1, 2, 3, 5, 6, 7, 8

[76] Xiaoyang Wu, Daniel DeTone, Duncan Frost, Tianwei Shen, Chris Xie, Nan Yang, Jakob Engel, Richard Newcombe, Hengshuang Zhao, and Julian Straub. Sonata: Self-Supervised Learning of Reliable Point Representations. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025. 7

[77] Kai M Wurm, Armin Hornung, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. OctoMap: A Probabilistic, Flexible, and Compact 3D Map Representation for Robotic Systems. In *International Conference on Robotics and Automation (ICRA)*, 2010. 1

[78] Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tieyan Liu. On Layer Normalization in the Transformer Architecture. In *International Conference on Machine Learning (ICML)*, 2020. 3

[79] Shengdong Xu, Dominik Honegger, Marc Pollefeys, and Lionel Heng. Real-time 3D navigation for autonomous vision-guided MAVs. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015. 1

[80] Yifan Xu, Tianqi Fan, Mingye Xu, Long Zeng, and Yu Qiao. SpiderCNN: Deep Learning on Point Sets with Parameterized Convolutional Filter. In *European Conference on Computer Vision (ECCV)*, 2018. 2

[81] Yufei Xu, Qiming Zhang, Jing Zhang, and Dacheng Tao. ViTAE: Vision Transformer Advanced by Exploring Intrinsic Inductive Bias. *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. 3

[82] Honghui Yang, Wenxiao Wang, Minghao Chen, Binbin Lin, Tong He, Hua Chen, Xiaofei He, and Wanli Ouyang. PVT-SSD: Single-Stage 3D Object Detector With Point-Voxel Transformer. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 3

[83] Yu-Qi Yang, Yu-Xiao Guo, and Yang Liu. Swin3D++: Effective Multi-Source Pretraining for 3D Indoor Scene Understanding. *Computational Visual Media*, 2025. 2

[84] Yu-Qi Yang, Yu-Xiao Guo, Jian-Yu Xiong, Yang Liu, Hao Pan, Peng-Shuai Wang, Xin Tong, and Baining Guo. Swin3D: A Pretrained Transformer Backbone for 3D Indoor Scene Understanding. *Computational Visual Media*, 2025. 2

[85] Zetong Yang, Li Jiang, Yanan Sun, Bernt Schiele, and Jiaya Jia. A Unified Query-Based Paradigm for Point Cloud Understanding. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2

[86] Tianwei Yin, Xingyi Zhou, and Philipp Krahenbuhl. Center-Based 3D Object Detection and Tracking. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 8

[87] Ziyin Zeng, Mingyue Dong, Jian Zhou, Huan Qiu, Zhen Dong, Man Luo, and Bijun Li. DeepLA-Net: Very Deep Local Aggregation Networks for Point Cloud Analysis. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025. 2

[88] Cheng Zhang, Haocheng Wan, Xinyi Shen, and Zizhao Wu. PatchFormer: An Efficient Point Transformer with Patch Attention. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2

[89] Ji Zhang and Sanjiv Singh. LOAM: Lidar odometry and mapping in real-time. In *Robotics: Science and Systems*, 2014. 1

[90] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. Point Transformer. In *International Conference on Computer Vision (ICCV)*, 2021. 2, 3

[91] Jia Zheng, Junfei Zhang, Jing Li, Rui Tang, Shenghua Gao, and Zihan Zhou. Structured3D: A Large Photo-Realistic Dataset for Structured 3D Modeling. In *European Conference on Computer Vision (ECCV)*, 2020. 7

[92] Xinge Zhu, Hui Zhou, Tai Wang, Fangzhou Hong, Yuexin Ma, Wei Li, Hongsheng Li, and Dahua Lin. Cylindrical and Asymmetrical 3D Convolution Networks for LiDAR Segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 7