Volume Transformer: Revisiting Vanilla Transformers for 3D Scene Understanding

Kadir Yilmaz[1,*], Adrian Kruse[1,*], Tristan Höfer[1]
Daan de Geus[2], and Bastian Leibe[1]

[1] RWTH Aachen University, Germany
[2] Eindhoven University of Technology, Netherlands

[*]Equal contribution.

https://vision.rwth-aachen.de/Volt

Abstract. Transformers have become a common foundation across deep learning, yet 3D scene understanding still relies on specialized backbones with strong domain priors. This keeps the field isolated from the broader Transformer ecosystem, limiting the transfer of research advances from other domains and the benefits of increasingly optimized software and hardware stacks. To bridge this gap, we propose the Volume Transformer (Volt), which adapts the vanilla Transformer encoder to 3D scenes with minimal modifications. Specifically, Volt partitions 3D scenes into volumetric patch tokens, processes them with full global self-attention, and injects positional information through 3D rotary positional embeddings (RoPE). Our initial experiments reveal that naively training Volt on standard 3D benchmarks leads to poor generalization, highlighting the limited scale of current 3D supervision. To overcome this, we introduce a data-efficient training recipe based on strong 3D augmentations, regularization, and distillation from a convolutional teacher, making Volt competitive with state-of-the-art methods. We then scale supervision through joint training on multiple datasets and show that Volt benefits more from increased scale than domain-specific 3D backbones, achieving state-of-the-art results across indoor and outdoor semantic segmentation benchmarks. Finally, we use Volt as a drop-in backbone in a standard 3D instance segmentation pipeline, where it also achieves new state-of-the-art results, highlighting its potential as a simple, scalable, and general-purpose backbone for 3D scene understanding.

Keywords: 3D Scene Understanding · Transformers · Scalability

1 Introduction

The Transformer architecture [75] has emerged as a unifying building block in deep learning. Originally introduced for natural language processing, it has since become the standard architecture across a wide range of domains [7, 19, 50, 51]. This widespread adoption has created a shared research ecosystem, where advancements in one domain often transfer to others with minimal changes [26,

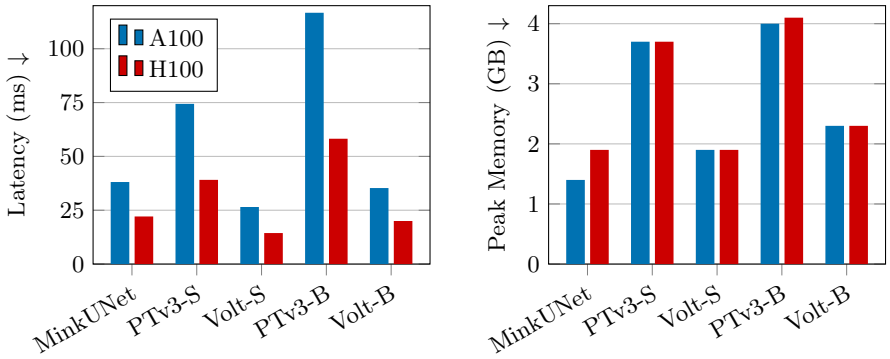
27, 65]. At the same time, a mature hardware and software stack has emerged: modern accelerators are explicitly optimized for Transformer workloads [2], and efficient implementations such as FlashAttention [16, 17, 59] leverage these specialized hardware capabilities to maximize throughput. Altogether, these properties make Transformers a compelling design choice for building scalable models over specialized, domain-specific architectures.

In contrast to the cross-domain convergence around Transformers, 3D scene understanding remains dominated by specialized architectures. Fully convolutional 3D U-Nets [13,92] have long served as the backbone of choice and continue to be highly competitive, particularly for instance-level tasks [35,58,86]. More recent works, motivated by the success of the Vision Transformer (ViT) [19], have incorporated attention mechanisms into 3D backbones while retaining the U-Net design and restricting the attention to local windows [36,37,79,80,83,88,90]. For instance, the current state of the art, PTv3 [79], utilizes space-filling curves to group points for local attention and uses 3D convolutions for positional encoding and feature aggregation [88]. As a result, both the earlier convolutional and recent attention-based 3D backbones use domain-specific local operations and depend on multi-resolution architectures to model long-range dependencies. These design choices provide strong inductive biases that are effective in low-data regimes, but can become a limitation as supervision scales, as demonstrated in 2D vision [19]. Furthermore, the resulting specialized architectures isolate 3D scene understanding from the broader Transformer ecosystem, limiting both the transfer of research advances from other domains and the adoption of Transformer-specific hardware and software optimizations.

To bridge this gap, we revisit the vanilla Transformer architecture and apply it to 3D scene understanding with minimal modifications. Given an input point cloud, we first impose a grid structure by subsampling it on a 3D voxel grid. Then, analogous to ViT patch embedding, we partition the voxelized space into non-overlapping cubic patches, flatten each patch, and linearly project it to the Transformer's embedding dimension, yielding a sequence of tokens. These tokens are processed by a standard Transformer encoder with full global self-attention. To inject positional information, we extend rotary positional embeddings [65] (RoPE), which have proven effective in 1D language [23,71] and 2D vision [9,61] data, to 3D volumes. We call the resulting architecture the *Volume Transformer* (Volt) and apply it to 3D segmentation tasks using a lightweight decoder to produce dense predictions.

A common concern is that full global self-attention may be prohibitively expensive for 3D scenes. In practice, this is not the case. With cubic patches of side length 10 cm for indoor environments and 25 cm for outdoor environments, typical scenes contain on the order of 5,000 tokens on datasets such as ScanNet [15] and nuScenes [8]. At this scale, fused attention implementations [16, 59] make full global attention practical for both training and inference. In fact, Volt-B, which uses the same Transformer encoder configuration as ViT-B, is 2× faster than the prior state-of-the-art PTv3 while using 48% less memory, despite PTv3 relying only on local attention and having fewer parameters (see Fig. 1).

Fig. 1: Inference efficiency on ScanNet. Despite using full global attention, Volt-S is more efficient than the commonly used MinkUNet [13] and state-of-the-art PTv3 [79]. Even the larger Volt-B model is 2× faster than PTv3-S while using less memory.



While Volt is both simple and efficient, we find that naive training on standard 3D datasets leads to pronounced overfitting. This is not surprising: unlike domain-specific architectures with strong inductive biases, a plain Transformer with global attention has a larger hypothesis space [14,52]. This can lead to poor generalization in data-scarce regimes, but it also allows the model to identify complex patterns when enough supervision is provided. For instance, the original Vision Transformer [19] underperforms in low-data regimes, but improves significantly as data and compute increase, eventually surpassing architectures with stronger inductive biases [32]. We expect a similar trend to hold in 3D, but current benchmarks do not yet provide comparable scale. For reference, widely used 3D datasets such as ScanNet [15] and nuScenes [8] are orders of magnitude smaller than even modest image datasets like ImageNet [18], let alone web-scale datasets such as LAION [56, 57] that power recent foundation models [12, 53].

To address the limited scale of current 3D datasets, we refrain from incorporating stronger priors into the architecture and instead tackle the problem on the training side by adapting data-efficient Transformer recipes from 2D vision [64, 74] to 3D. In particular, we combine strong 3D data augmentations [42] and regularization [30, 68] with knowledge distillation from a convolutional teacher [74]. These training techniques make Volt competitive with domain-specific 3D backbones under standard single-dataset training. To further increase supervision scale, we train Volt jointly on multiple datasets [4, 15, 85]. With increased data, we observe consistent gains and stronger scaling behavior compared to prior backbones, achieving new best results of 80.5 mIoU on the ScanNet test set, 41.6 mIoU on the ScanNet200 test set, as well as 82.2 mIoU on the nuScenes validation set. Finally, we demonstrate the generality of Volt beyond semantic segmentation by simply replacing the commonly used U-Net [13] backbone in a standard 3D instance segmentation pipeline [35] with Volt. In this setting, Volt again attains state-of-the-art performance, reaching 82.7 mAP50 on ScanNet, 47.5 mAP50 on ScanNet200, and 54.9 mAP50 on ScanNet++ test sets, indicating that it can serve as a general-purpose 3D backbone.

In summary, our contributions are threefold: (i) We introduce the Volume Transformer (Volt), a simple and efficient 3D backbone that represents 3D scenes as sequences of volumetric patch tokens, applies a vanilla Transformer encoder with full global self-attention, and injects positional information via a 3D extension of RoPE. (ii) We propose a data-efficient training recipe for Volt that combines strong 3D augmentations, regularization, and distillation from a convolutional teacher, enabling effective training in today's limited-data regime. (iii) We demonstrate that this design scales favorably with increased supervision, achieving strong results on indoor and outdoor semantic and instance segmentation benchmarks, while being fast and memory-efficient.

2 Related Work

Unlike images, which lie on a regular 2D grid, 3D point clouds are unordered and irregular. Early methods, such as PointNet and PointNet++ [47,48], address these challenges by treating point clouds as sets and enforcing permutation invariance through shared MLPs and symmetric pooling operations. To better exploit local geometric structure, subsequent works introduce continuous convolution operators defined directly in 3D space [72, 73, 78]. By learning kernel functions over continuous spatial regions, these methods avoid discretization artifacts and preserve fine-grained geometric details. However, they rely on neighbor search and kernel evaluation algorithms, resulting in high computational cost and sensitivity to point densities, limiting their applicability to large-scale scenes [25].

An alternative line of work discretizes 3D space into voxels, enabling the use of 3D convolutional layers operating on a grid. However, representing point clouds as dense voxel grids is memory-intensive, since the number of voxels scales cubically with spatial resolution. To address this, sparse convolution engines [13, 22, 63, 69] exploit the inherent sparsity of 3D data by using hash-based data structures and perform convolutions only on active voxels, achieving substantial gains in efficiency. Building on these engines, fully convolutional 3D UNet architectures [13, 22, 54, 69] have been widely adopted for 3D scene understanding tasks, spanning indoor and outdoor environments as well as semantic and instance-level tasks [35, 36, 58, 86, 92]. Despite their efficiency, they are ultimately built on local convolutional operators, so global context is typically aggregated only through network depth, pooling, and multi-scale feature fusion.

Motivated by the success of Vision Transformers [19], recent works explore attention-based architectures for 3D understanding. While global self-attention offers a flexible mechanism for modeling long-range interactions, most methods constrain attention to local neighborhoods to avoid the quadratic cost on large-scale point clouds. Examples include octree- and window-based variants such as OctFormer [76], Stratified Transformer [37], and Swin3D [83], which adapt core principles from 2D Transformers to 3D data. Finally, the Point Transformer family [79,80,90] proposes attention operators with improved efficiency and scalability. Point Transformer v3 (PTv3), the current state-of-the-art 3D backbone,

introduces space-filling curves to serialize 3D points into 1D sequences and partitions scenes into smaller chunks according to this ordering. This design largely preserves spatial locality while enabling efficient local self-attention. Overall, however, despite having attention layers, current 3D backbone architectures remain a hybrid design rather than a direct instantiation of the Transformer encoder. They depend on U-Net-style multi-resolution hierarchies and use local operators with hand-designed neighborhoods. While such inductive biases are effective in current data-limited 3D benchmarks, they may constrain scalability as supervision and compute increase.

In this work, we take a complementary perspective by adopting a vanilla Transformer architecture that closely follows the ViT design and by leveraging training strategies shown to be effective in 2D. Our results show that architectural simplicity, when paired with appropriate training recipes, can offer a scalable and highly competitive alternative for 3D scene understanding.

In parallel to scene-scale 3D perception, Transformers have also been studied for object-centric point clouds [24, 33, 44, 87], including shape classification and part segmentation, most commonly on synthetic benchmarks such as ModelNet40 [82] and ShapeNet [10]. However, these single-object settings differ fundamentally from large-scale scene understanding. Object-level benchmarks provide isolated, canonicalized shapes with a fixed and relatively small number of points (e.g., 1,024 or 2,048), and the geometry is typically clean and uniformly sampled. In contrast, 3D scene understanding requires handling point clouds that are orders of magnitude larger, with substantial density variation, clutter, and occlusions, and semantic reasoning that spans entire rooms or outdoor environments. As a result, object-level Transformer architectures and training regimes do not directly transfer to full-scene 3D understanding.

3 Method

We present the *Volume Transformer* (Volt), a plain Transformer backbone for 3D scene understanding. In contrast to current 3D architectures that rely on domain-specific modules [13, 76, 79], Volt closely follows ViT [19] and makes a few targeted modifications for 3D data. An overview is shown in Fig. 2. Given a 3D scene, Volt partitions the scene into non-overlapping cubic patches, tokenizes them, and processes the resulting token sequence with a standard Transformer encoder using full global self-attention. The encoder outputs a set of latent tokens forming a volumetric feature map that can be used across downstream 3D tasks. In this work, we attach a lightweight decoder to predict per-point logits for semantic segmentation and a Transformer decoder for instance segmentation. Since Volt relies only on a vanilla Transformer encoder and linear layers, it can be *implemented purely in PyTorch* [45] without specialized sparse-convolution engines. Moreover, it is fully compatible with highly optimized attention kernels such as FlashAttention [16,17,59] and can directly benefit from ongoing advances in the broader Transformer ecosystem.

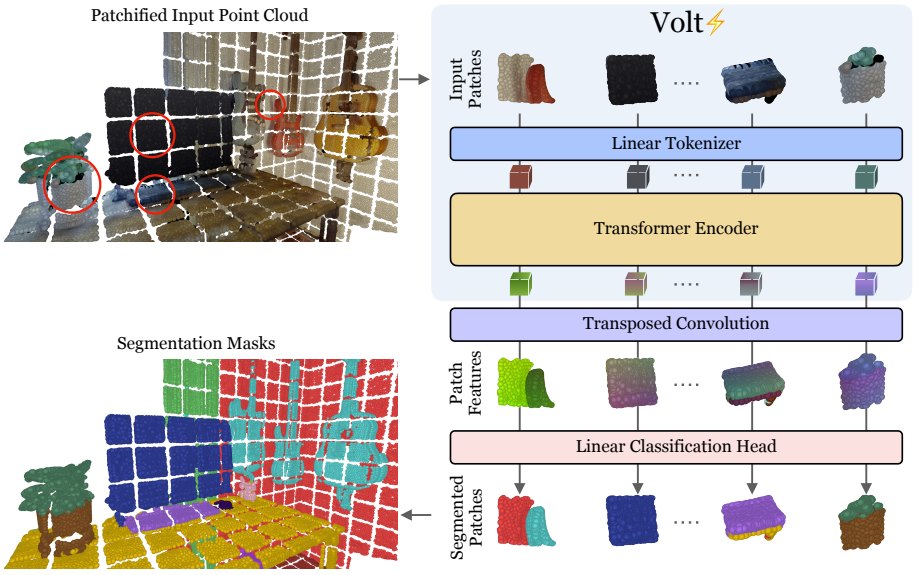


Fig. 2: Volume Transformer (Volt) architecture. The 3D scene is partitioned into non-overlapping volumetric patches, each patch is embedded into a token with a linear tokenizer, and the tokens are processed by a Transformer encoder with global attention. A transposed convolution maps the encoded tokens back to dense features, which are finally projected to per-point semantic predictions using a linear classification head.

3.1 Volt Architecture

Tokenization. Let $\mathcal{X} = \{(\mathbf{p}_n, \mathbf{f}_n)\}_{n=1}^{N}$ denote an input point cloud with N points, where each point has a 3D coordinate $\mathbf{p}_n \in \mathbb{R}^3$ and a C-dimensional feature vector $\mathbf{f}_n \in \mathbb{R}^C$ (e.g., color). A naive application of a Transformer would treat each point as a token, yielding a sequence of length N and a quadratic attention cost of $\mathcal{O}(N^2)$. This is infeasible for typical scene-scale inputs where $N > 100{,}000$. Instead, Volt first voxelizes the point cloud $\mathcal{X}$ with voxel size δ to impose a regular grid structure. For each occupied voxel, a single representative point is selected, resulting in a sparse voxel set $\mathcal{X}_v = \{(\mathbf{z}_m, \tilde{\mathbf{f}}_m)\}_{m=1}^{M}$ where $M < N$, $\mathbf{z}_m \in \mathbb{Z}^3$ is the integer voxel coordinate, and $\tilde{\mathbf{f}}_m \in \mathbb{R}^C$ is the feature vector of the representative point in that voxel. The sparse voxel grid is then partitioned into non-overlapping cubic patches of $P \times P \times P$ voxels. Each patch with at least one active voxel, i.e. each non-empty patch, is densified by filling empty voxels with zero features while completely empty patches are discarded. Then, each densified patch of shape $P \times P \times P$ is flattened, yielding a patch vector $\mathbf{u}_t \in \mathbb{R}^{P^3 C}$, where $t \in \{1, \ldots, T\}$ are patches indices. Finally, each patch vector $\mathbf{u}_t$ is projected to the Transformer embedding dimension D using a shared learnable linear mapping, producing a token $\mathbf{x}_t \in \mathbb{R}^D$. Together, these form the token sequence $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_T]$ processed by the Transformer encoder. Since Volt only computes a single token for each non-empty patch, the number of

tokens T depends on the scene size, point density, and spatial distribution of occupied voxels. Thus, unlike in ViTs with fixed-size images, the token sequence length T varies across inputs.

Overall, the tokenization step mirrors the patch embedding step in ViT: the input is split into regular patches, each patch is flattened, and the resulting vector is projected to the Transformer dimension. In essence, it aggregates neighboring points and their low-dimensional features into a token, while simultaneously downsampling the scene to keep the sequence length manageable. Compared to kNN-based grouping, which is density-dependent and can become a bottleneck at the scene scale [79, 80], grid-based patchification followed by a linear layer provides a simple and efficient way to form tokens.

Transformer Encoder. The token sequence $\mathbf{X}$ is processed by a *standard* Transformer encoder consisting of a stack of L identical Transformer blocks. Each block contains a multi-head self-attention and an MLP layer with residual skip connections. Following common practice, we apply LayerNorm [3] before both the attention and MLP layers. There are no hierarchical stages, no local windows, and no domain-specific grouping strategies, but only global attention and MLPs. To make full global self-attention practical, we use FlashAttention [16, 17, 59], which provides a fused and memory-efficient attention kernel for variable-length sequences. This allows Volt to retain full-scene interactions while directly leveraging highly optimized, general-purpose attention implementations, in contrast to other structured attention variants that typically rely on dedicated kernels and custom indexing logic [76, 80].

Positional Encodings. Injecting positional information is essential for a permutation equivariant Transformer, and it is particularly critical in 3D, where semantics are largely determined by geometry and spatial occupancy. Absolute learnable positional embeddings, while standard in early ViTs, are not suitable for point clouds. They assume a fixed, dense grid, which does not hold for sparse, unbounded 3D scenes, where the number of possible token positions can reach into the millions even at modest scales. In contrast, RoPE [65], which has become a standard choice in modern LLMs [23, 71] and ViTs [61], is well-suited for 3D data. RoPE injects position information by rotating the query and key vectors before scaled dot-product attention, so attention scores depend on *relative* positional offsets. It supports variable sequence lengths, generalizes beyond positions seen during training, and is compatible with fused attention kernels.

Inspired by 2D extensions of RoPE for ViTs [20, 28], we generalize RoPE to 3D, similar to concurrent work LitePT [88], by factorizing each query and key vector across the three spatial axes. Let $\mathbf{q}, \mathbf{k} \in \mathbb{R}^D$ denote the query and key vectors of a token at position $\mathbf{p} = (p_x, p_y, p_z)$. We decompose $\mathbf{q}$ and $\mathbf{k}$ into axis-specific components, i.e., $\mathbf{q} = [\mathbf{q}_x, \mathbf{q}_y, \mathbf{q}_z]$ and $\mathbf{k} = [\mathbf{k}_x, \mathbf{k}_y, \mathbf{k}_z]$, and apply RoPE independently along each axis:

$$\begin{aligned} \tilde{\mathbf{q}} &= \mathrm{concat}(\mathcal{R}_\Theta(p_x)\mathbf{q}_x,\ \mathcal{R}_\Theta(p_y)\mathbf{q}_y,\ \mathcal{R}_\Theta(p_z)\mathbf{q}_z)\,, \\ \tilde{\mathbf{k}} &= \mathrm{concat}(\mathcal{R}_\Theta(p_x)\mathbf{k}_x,\ \mathcal{R}_\Theta(p_y)\mathbf{k}_y,\ \mathcal{R}_\Theta(p_z)\mathbf{k}_z)\,. \end{aligned} \tag{1}$$

where $\mathcal{R}_\Theta(\cdot)$ denotes the standard block-diagonal rotary transformation [65] parameterized by frequencies Θ. We use the discrete token indices obtained after patchification as positions $\mathbf{p}$. These indices are proportional to the underlying 3D coordinates up to the constant scale factor δP, preserving metric consistency across scenes and enabling the model to exploit globally meaningful spatial relationships. In addition, we allocate the query and key dimensions asymmetrically across axes, assigning more capacity to x and y than to the gravity-aligned z axis, reflecting the larger spatial extent of typical scenes in the horizontal plane.

3.2 Lightweight Decoder

3D scene understanding tasks require per-point predictions, whereas Volt operates on patch tokens. Therefore, after processing the tokens through Transformer layers, we upsample them back to the original voxel resolution using a *single transposed convolution*. We deliberately avoid deeper convolutional decoders to minimize overhead and reduce the risk of overfitting, since the Transformer backbone already provides strong representation capacity. With kernel size equal to the patch size P, this layer serves as the natural inverse of tokenization and produces per-voxel features $\mathbf{F} \in \mathbb{R}^{M \times D'}$. In practice, we implement the transposed convolution as a linear layer followed by a voxel shuffling operation to remove any dependency on sparse convolution engines [13, 63]. For semantic segmentation, we map $\mathbf{F}$ to per-voxel class logits with a linear classifier. For instance segmentation, we follow the common paradigm and use a Transformer decoder [66]. In both cases, during evaluation, we obtain per-point predictions by assigning each point the prediction of its containing voxel.

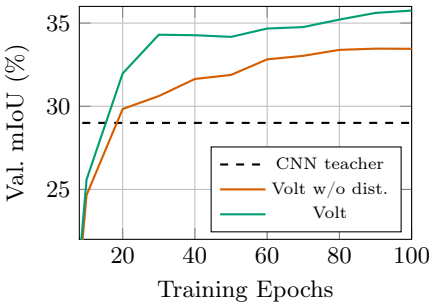
4 Training Volt

Full global self-attention gives Transformers the capacity to model long-range interactions across an entire 3D scene, but it also makes them data-hungry. In 3D, supervision is comparatively scarce, and we find that naively training Volt on standard 3D benchmarks leads to poor generalization. To address this, we propose a rigorous training recipe inspired by data-efficient training strategies [64, 74] in 2D vision, combining strong data augmentation, model regularization, and distillation to enable effective training of Volt from scratch. Finally, we scale supervision via multi-dataset training to study the scaling behavior of Volt. Fig. 3 (left) summarizes the cumulative effect of our training recipe on Volt and PTv3 [79].

Data Augmentation. We apply a diverse set of augmentations tailored to 3D point clouds. Specifically, scene mixing [42], random cropping, and instance-level transformations alter the global scene context, whereas elastic distortions as well as random scaling, translation, rotation, and flipping perturb local geometry. Together, these augmentations increase data diversity and encourage invariance to local geometric perturbations.

Fig. 3: Volt training recipe analysis on ScanNet200. Left: incremental training recipe analysis. Right: training curves with and without CNN distillation.

	Volt	PTv3 [79]
Naive (baseline)	31.0	35.2
+ stronger aug.	32.9 (+1.9)	35.5 (+0.3)
+ stronger reg.	34.1 (+1.2)	35.5 (+0.0)
+ CNN distillation	36.2 (+2.1)	35.8 (+0.3)
+ scaling data	38.1 (+1.9)	36.7 (+0.9)
+ scaling model	40.0 (+1.9)	37.5 (+0.8)



Regularization. To mitigate overfitting, we apply established regularization techniques used for training Transformers [43, 61, 74]. We apply *DropPath* [30], also known as stochastic depth, with a relatively high drop rate, randomly dropping the residual branches of the attention and MLP sublayers during training. The drop rate is increased linearly with depth, so earlier layers are retained more frequently, which stabilizes optimization in deep encoders. We further use AdamW with relatively strong *weight decay* to discourage overly large weights. At the loss level, we apply *label smoothing* [68] to prevent over-confident predictions. Together, these choices form a simple but effective regularization recipe, consistent with best practices for ViT training.

Distillation from a Convolutional Teacher. Prior work in 2D vision shows that distillation from CNNs transfers useful inductive biases into Transformers, improving performance especially in data-scarce regimes [1, 11, 89]. We adopt this idea in 3D by using a 3D U-Net [13] that is trained on the same data as a teacher while training Volt. Note that *this is not the typical distillation setting [29]* where a stronger teacher guides a weaker student. In fact, the teacher consistently underperforms Volt across different settings on the validation set, while still benefiting Volt through the distillation objective (see Fig. 3, right).

Following DeiT [74], we use the teacher's hard segmentation predictions y_{teacher} as distillation targets. Accordingly, Volt uses two linear classification heads on the voxel features $\mathbf{F}$: a segmentation head with weights $\mathbf{W}_{\text{seg}}$ supervised by the ground truth y_{gt}, and a distillation head with weights $\mathbf{W}_{\text{distill}}$ supervised by the teacher predictions y_{teacher}. The loss function is defined as:

$$\mathcal{L} = 0.5\,\mathcal{L}_{\text{seg}}\big(\mathbf{W}_{\text{seg}}^{T}\mathbf{F},\, y_{\text{gt}}\big) + 0.5\,\mathcal{L}_{\text{seg}}\big(\mathbf{W}_{\text{distill}}^{T}\mathbf{F},\, y_{\text{teacher}}\big)\,. \tag{2}$$

This joint loss function allows the model to benefit from the teacher's inductive biases while remaining directly optimized for the semantic segmentation objective. Moreover, because the teacher receives the same augmented inputs as the student, its predictions vary across different augmentations of the same scene, providing an additional source of regularization. After training, the teacher is discarded, so distillation introduces no additional cost during inference.

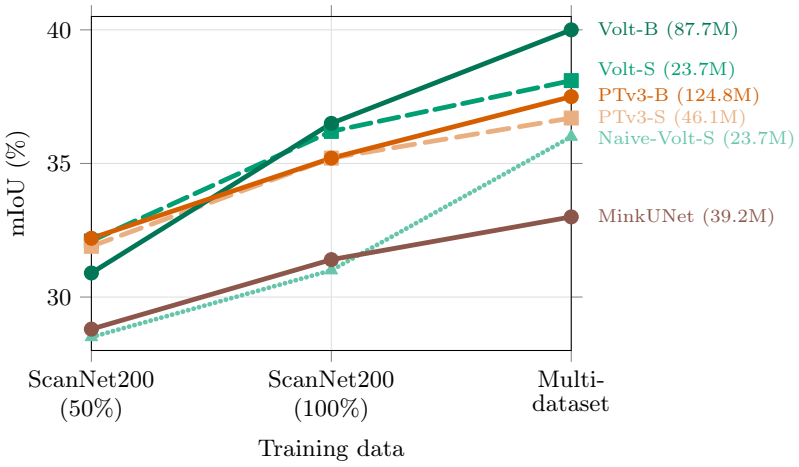


Fig. 4: Scaling behaviour comparison of 3D backbones. Semantic segmentation mIoU on ScanNet200 when training with 50% and 100% of ScanNet200, and under multi-dataset joint training, also using ARKitScenes [31] and ScanNet++ [85]. As supervision increases, Volt scales more favorably than prior domain-specific backbones.

Scaling Data. As a standard Transformer with minimal inductive biases, Volt is expected to benefit more from additional data than domain-specific architectures [13, 79], mirroring the strong scaling behavior observed in ViTs [19]. To investigate this and better assess the potential of Volt on currently available 3D data, we train it jointly across multiple 3D datasets. We group data by domain, performing joint training on indoor RGB-D datasets and, separately, on outdoor LiDAR datasets following standard practice [31,34,81]. Notably, each dataset is collected with a different sensor and reconstructed via different pipelines, leading to substantial variation in point density, coverage, and noise, making cross-dataset transfer challenging [81]. Nonetheless, joint training increases both the amount of supervision and the diversity of the training data, offering a glimpse of how Volt scales with data size. To account for the different semantic label spaces across datasets, we use a per-dataset linear classification head while sharing the rest of the backbone.

Training Recipe Results. When evaluating the impact of this training recipe in Fig. 3, we find that PTv3 is stronger under naive training (35.2 vs. 31.0 mIoU), but Volt benefits substantially more from the data-efficient training recipe and increased supervision. Across the same sequence of upgrades, Volt improves by +9.0 mIoU (31.0→40.0), whereas PTv3 gains only +2.3 mIoU (35.2→37.5). Notably, stronger regularization and CNN distillation improve Volt significantly more than PTv3, suggesting that they mitigate the effects of weaker inductive biases and the resulting overfitting under limited supervision. This trend is further supported by Fig. 4, which evaluates segmentation accuracy as the amount of training data increases. Under naive training, Volt performs poorly in the low-

data regime but improves rapidly as the training set grows, revealing substantial headroom with scale. Applying data-efficient training recipe significantly improves performance, making Volt-S consistently better than the 5× larger PTv3-B model across all data scales. At the largest supervision scale, improvements for Volt-S begin to saturate, suggesting performance becomes capacity-limited with 23.7M parameters. Scaling the backbone to Volt-B alleviates this saturation: Volt-B continues to benefit from additional supervision and opens a clear gap under multi-dataset training. Overall, these results suggest that, as supervision increases, hand-crafted architectural priors become less critical and may even introduce unnecessary overhead, while Volt can instead learn domain-specific 3D patterns directly from data.

5 Experiments

Implementation Details. We introduce two variants of Volt, Volt-S and Volt-B, which use the same Transformer encoder configuration as ViT-S and ViT-B, respectively, and apply QKNorm [27] in all attention layers. Following common practice, we use a voxel size δ of 2 cm for indoor datasets and 5 cm for outdoor datasets. With a patch size $P = 5$, this results in cubic patches of side length 10 cm for indoor datasets and 25 cm for outdoor datasets. For naive training, we use the same data augmentations as PTv3 [79]. For our data-efficient training recipe, we additionally apply random point dropout, per-instance rotation, shift, and scaling using the provided instance masks, as well as aggressive scene mixing [42] with probability 0.85. For CNN distillation, we adopt the commonly used MinkUNet Res16UNet34C [13] as the convolutional teacher. Following prior work [66, 79], we use a combination of cross-entropy and Lovász loss [6] for semantic segmentation, and a combination of Dice loss and cross-entropy loss for instance segmentation. We optimize all models with AdamW [39] using a 1-cycle [62] learning-rate schedule, a weight decay of 0.05, and label smoothing of 0.1. We maintain an exponential moving average of the model weights with decay 0.999 for more stable evaluation. All models are trained on 8 A100 GPUs with a global batch size of 16. Unless stated otherwise, we use FP16 mixed precision and FlashAttention-2 for improved memory efficiency and training speed. Following the timm library [77], we initialize all linear layers with a truncated normal distribution (σ=0.02) and set all biases to zero.

Datasets. We evaluate Volt on a diverse suite of indoor and outdoor 3D benchmarks, following the official splits and evaluation protocols for all datasets.

For indoor scene understanding, we train and evaluate separately on the ScanNet, ScanNet200, and ScanNet++ datasets. ScanNet [15] contains 1,613 scenes with hundreds of raw semantic annotation categories, but its standard benchmark is limited to a subset of 20 classes. ScanNet200 [55] proposes to use a finer-grained label space of 200 semantic classes for the same underlying ScanNet scans, better capturing the long-tail diversity of real indoor environments. ScanNet++ [85] provides higher-quality 3D reconstructions and comprises 956 scenes, annotated with 100 semantic classes. For joint training, we combine these

Table 1: Indoor semantic segmentation results (mIoU).

Method	#Params	ScanNet		ScanNet200		ScanNet++
		Val	Test	Val	Test	Test
ST [37]	18.8M	74.3	73.7	–	–	–
PTv1 [90]	11.4M	70.6	–	27.8	–	–
PointNeXt [49]	41.6M	71.5	71.2	–	–	–
MinkUNet [13]	39.2M	72.2	73.6	25.0	25.3	45.6
OctFormer [76]	44.0M	75.7	76.6	32.6	32.6	46.0
Swin3D [83]	23.6M	76.4	–	–	–	–
PTv2 [80]	12.8M	75.4	75.2	30.2	–	44.5
OA-CNN [46]	51.5M	76.1	75.6	32.3	33.3	47.0
PTv3 [79]	46.1M	77.5	77.9	35.2	37.8	48.8
Volt-S	23.7M	77.2	–	36.2	–	49.3
↓ Jointly trained on multiple datasets						
PTv3/PPT [31, 81]	124.8M	79.1	79.8	37.5	41.4	–
Volt-S	23.7M	80.2	–	38.1	–	–
Volt-B	87.7M	**80.5**	**80.5**	**40.0**	**41.6**	**49.5**

three datasets and additionally include ARKit LabelMaker [31], which augments ARKitScenes [4] with automatically generated semantic labels, providing 5,019 real scans annotated with 184 classes. While these labels are imperfect, their scale substantially increases supervision and enables us to study the data-scaling behavior of Volt beyond what is possible with manually annotated data.

For outdoor evaluation, we use the nuScenes [8], Waymo Open Dataset [67], and SemanticKITTI [5] datasets, which differ in sensor configurations and scene dynamics. nuScenes contains 1,000 sequences of duration 20 s captured with a 32-beam LiDAR and provides 16 semantic classes. The Waymo Open Dataset is of comparable scale but uses a denser 64-beam LiDAR and contains 22 classes. SemanticKITTI is annotated with 19 semantic classes and uses a LiDAR sensor similar to Waymo. However, it is less diverse, containing only 9 training sequences and 1 validation sequence collected in the Karlsruhe suburbs, which results in limited geographic diversity.

Indoor Semantic Segmentation. Table 1 compares Volt to prior work on indoor semantic segmentation. Under single-dataset training, Volt-S is highly competitive and even surpasses previous approaches while being 2× smaller and faster. Next, we scale up supervision via joint training on ScanNet, ScanNet200, ScanNet++, and ARKitScenes, and compare to PPT [31, 81], which builds on the PTv3-B backbone and augments it with dataset-specific BatchNorm/LayerNorm for joint training. We find that increasing supervision yields strong performance improvements for Volt-S, reaching 80.2 mIoU on ScanNet and 38.1 mIoU on ScanNet200, surpassing even the 6× larger PPT trained additionally on the Structured3D dataset [91] (5 datasets in total). We then scale up

Table 2: Outdoor semantic segmentation results (mIoU).

Method	#Params	nuScenes		Sem.KITTI		Waymo
		Val	Test	Val	Test	Val
MinkUNet [13]	39.2M	73.3	–	63.8	–	65.9
SPVNAS [70]	10.8M	77.4	–	64.7	66.4	–
Cylinder3D [92]	26.1M	76.1	77.2	64.3	67.8	–
SphereFormer [36]	32.3M	78.4	81.9	67.8	74.8	69.9
LSK3DNet [21]	28.8M	80.1	–	70.2	**75.6**	–
PTv2 [80]	12.8M	80.2	82.6	70.3	72.6	70.6
PTv3 [79]	46.1M	80.4	82.7	69.1	74.2	71.3
Volt-S	23.7M	81.0	–	70.5	–	71.5
↓ Jointly trained on multiple datasets						
PTv3/PPT [81]	46.3M	81.2	**83.0**	72.3	75.5	72.1
Volt-S	23.7M	81.8	–	72.2	–	72.4
Volt-B	87.7M	**82.2**	82.2	**72.5**	75.2	**73.4**

to Volt-B, achieving new best results across all datasets in all settings, with test-set scores of 80.5 mIoU on ScanNet, 41.6 mIoU on ScanNet200, and 49.5 mIoU on ScanNet++. Overall, these results show that Volt benefits substantially from increased and more diverse supervision, and they highlight the effectiveness of a plain Transformer backbone with full scene-level self-attention for indoor 3D understanding.

Beyond segmentation accuracy, Fig. 1 shows that Volt is also efficient: using FlashAttention-2, it achieves lower inference latency on both A100 and H100 GPUs compared to MinkUNet and PTv3, despite using full global self-attention. Even Volt-B remains faster than the fully convolutional MinkUNet with only a modest increase in peak GPU memory, while delivering substantially stronger segmentation accuracy. Together, these results demonstrate that Volt is not only effective, but also an efficient and scalable backbone for indoor 3D understanding.

Outdoor Semantic Segmentation. As summarized in Table 2, Volt transfers cleanly to large-scale outdoor scenes, despite the sparser and irregular sampling of LiDAR compared to indoor RGB-D reconstructions. Under single-dataset training, Volt-S already achieves the best performance on SemanticKITTI, nuScenes, and Waymo validation sets. Joint training across outdoor datasets yields consistent gains, and Volt-S reaches state-of-the-art performance, outperforming PPT on one benchmark and matching it on the two. Again, scaling to Volt-B further improves results. Notably, these gains are achieved without introducing LiDAR-specific design choices [21, 36, 92], underscoring that the same tokenization and backbone design remains effective under substantial shifts in point density, range, and motion patterns. Overall, Volt is not tied to indoor-specific priors and generalizes well to LiDAR-based perception at scale.

Table 3: Indoor instance segmentation results (mAP50).

Method	Backbone	ScanNet		ScanNet200		ScanNet++
		Val	Test	Val	Test	Test
SPFormer [66]	MinkUNet	73.9	77.0	33.8	–	43.5
Mask3D [58]	MinkUNet	73.7	78.0	37.0	38.8	–
OneFormer3D [35]	MinkUNet	78.1	80.1	40.8	–	43.3
MAFT [38]	MinkUNet	76.5	78.6	38.2	–	–
QueryFormer [41]	MinkUNet	74.2	78.7	37.1	–	–
SphericalMask [60]	MinkUNet	79.9	81.2	–	–	–
Relation3D [40]	MinkUNet	80.2	81.6	41.2	–	–
SGIFormer [84]	MinkUNet	**81.2**	79.9	39.4	–	45.7
SPFormer [66]	**Volt-S**	78.1	–	**48.4**	–	–
SPFormer [66]	**Volt-B**	78.3	**82.7**	**48.4**	**47.5**	**54.9**

Indoor Instance Segmentation. We further evaluate Volt on indoor 3D instance segmentation to test whether the backbone improvements extend beyond semantic segmentation. For a controlled comparison, we follow the established SPFormer pipeline [66] and keep the decoder and the losses unchanged, replacing only the original MinkUNet [13] backbone with Volt. Following prior work [35, 38, 40, 60, 84], we initialize Volt from our semantic segmentation checkpoint and fine-tune for instance segmentation. Results are reported in Table 3. We find that swapping MinkUNet for Volt-S yields substantial gains over the SPFormer baseline, with an especially large improvement on ScanNet200 (+14.6 mAP50), setting a new state of the art on the validation set. Scaling the backbone further to Volt-B achieves new best results on all three test sets, reaching 82.7 mAP50 on ScanNet, 47.5 mAP_{50} on ScanNet200, and 54.9 mAP50 on ScanNet++. Overall, these results suggest that strengthening just the underlying 3D representation can deliver gains larger than those obtained from increasingly specialized decoder designs [35, 40, 84].

Convolutional decoder. Table 4 (left) ablates the decoder used to produce dense semantic predictions. Without a decoder, we predict directly from patch tokens. This variant is the fastest, but it discards fine-grained spatial detail. Consequently, performance drops by 0.6 mIoU on SemanticKITTI and 1.2 mIoU on ScanNet200. Notably, though, even without any decoder Volt remains competitive. We also evaluate a larger decoder that appends two residual convolution blocks at the highest resolution after the transposed convolution. This variant is slightly slower and even reduces accuracy, suggesting that the Transformer backbone already provides strong representations and that a heavier decoder can exacerbate overfitting under limited supervision. Overall, a minimal decoder captures most of the benefit with only modest overhead.

Patch size. Table 4 (right) ablates the cubic patch size P used for tokenization. Using smaller patches of size $3 \times 3 \times 3$ yields the highest accuracy on both

Table 4: Left: Conv. decoder ablation. Having a lightweight decoder consistently boosts mIoU with modest overhead. Conversely, scaling to a larger decoder yields lower segmentation accuracy due to increased overfitting. **Right: Patch size ablation.** Larger patches shorten the token sequence and improve latency but reduce segmentation accuracy. $5\times5\times5$ patches offer the best overall trade-off.

Setting	ScanNet200		Sem.KITTI	
	mIoU	Speed	mIoU	Speed
no decoder	35.0	**26ms**	69.9	**48ms**
Volt-S	**36.2**	27ms	**70.5**	49ms
big decoder	35.8	29ms	69.5	50ms

Patch size	ScanNet200		Sem.KITTI	
	mIoU	Speed	mIoU	Speed
$3\times3\times3$	**37.7**	34ms	**71.0**	113ms
$5\times5\times5$	36.2	14ms	70.5	49ms
$7\times7\times7$	33.5	**13ms**	69.6	**34ms**

ScanNet200 (37.7 mIoU) and SemanticKITTI (71.0 mIoU), but is substantially slower due to the longer token sequence (2.4× on ScanNet200 and 2.3× on SemanticKITTI). Conversely, a larger patch size of $7\times7\times7$ shortens the sequence and improves latency, but noticeably degrades performance, with a larger drop on ScanNet200, reflecting the sensitivity of small objects and thin structures to coarse tokenization. Overall, $5\times5\times5$ provides a good trade-off, retaining most of the accuracy while delivering a large speedup over smaller patches.

6 Conclusion

We revisit the vanilla Transformer encoder as a backbone for 3D scene understanding and show that, with few targeted adaptations, it can serve as a simple, scalable alternative to current specialized 3D architectures. A key challenge is that Transformers generalize poorly under the limited supervision of current 3D benchmarks. We address this with a data-efficient training recipe and show that this architecture benefits more from increased supervision, achieving new best results across 3D benchmarks while remaining more efficient than prior state-of-the-art methods. Looking ahead, Volt's architectural simplicity and strong scaling behavior suggest a promising path toward further unification of 3D scene understanding with the broader Transformer ecosystem. We expect this work to unlock a new potential for larger-scale 3D supervision and pretraining, seamless multi-modal extensions with other Transformers, and continued progress driven by rapidly improving Transformer software and hardware support.

Acknowledgements

We acknowledge funding by BMFTR project "WestAI" (grant no. 16IS22094D) and the EU project "JUPITER AI Factory" (grant no. 101250682). Computations were performed with computing resources granted by RWTH Aachen under projects `rwth1742`, `rwth1788` and `rwth1968` and by the Gauss Centre for Supercomputing e.V. through the John von Neumann Institute for Computing on the GCS Supercomputer JUWELS at Julich Supercomputing Centre.

References

1. Abnar, S., Dehghani, M., Zuidema, W.: Transferring Inductive Biases through Knowledge Distillation. arXiv preprint arXiv:2006.00555 (2020)
2. Andersch, M., Palmer, G., Krashinsky, R., Stam, N., Mehta, V., Brito, G., Ramaswamy, S.: NVIDIA Hopper Architecture In-Depth. `https://developer.nvidia.com/blog/nvidia-hopper-architecture-in-depth/` (2022), NVIDIA Technical Blog. Accessed 2026-02-23.
3. Ba, J.L., Kiros, J.R., Hinton, G.E.: Layer Normalization. arXiv preprint arXiv:1607.06450 (2016)
4. Baruch, G., Chen, Z., Dehghan, A., Dimry, T., Feigin, Y., Fu, P., Gebauer, T., Joffe, B., Kurz, D., Schwartz, A., Shulman, E.: ARKitScenes - a diverse real-world dataset for 3d indoor scene understanding using mobile rgb-d data. In: NeurIPS (2021)
5. Behley, J., Garbade, M., Milioto, A., Quenzel, J., Behnke, S., Stachniss, C., Gall, J.: SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences. In: ICCV (2019)
6. Berman, M., Triki, A.R., Blaschko, M.B.: The Lovász-Softmax loss: A tractable surrogate for the optimization of the intersection-over-union measure in neural networks. In: CVPR (2018)
7. Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al.: Language Models are Few-Shot Learners. In: NeurIPS (2020)
8. Caesar, H., Bankiti, V., Lang, A.H., Vora, S., Liong, V.E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., Beijbom, O.: nuScenes: A Multimodal Dataset for Autonomous Driving. In: CVPR (2020)
9. Carion, N., Gustafson, L., Hu, Y.T., Debnath, S., Hu, R., Suris, D., Ryali, C., Alwala, K.V., Khedr, H., Huang, A., et al.: SAM 3: Segment Anything with Concepts. arXiv preprint arXiv:2511.16719 (2025)
10. Chang, A.X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., et al.: ShapeNet: An Information-Rich 3D Model Repository. arXiv preprint arXiv:1512.03012 (2015)
11. Chen, X., Cao, Q., Zhong, Y., Zhang, J., Gao, S., Tao, D.: DearKD: Data-Efficient Early Knowledge Distillation for Vision Transformers. In: CVPR (2022)
12. Cherti, M., Beaumont, R., Wightman, R., Wortsman, M., Ilharco, G., Gordon, C., Schuhmann, C., Schmidt, L., Jitsev, J.: Reproducible Scaling Laws for Contrastive Language-Image Learning. In: CVPR (2023)
13. Choy, C., Gwak, J., Savarese, S.: 4D Spatio-Temporal ConvNets: Minkowski Convolutional Neural Networks. In: CVPR (2019)
14. Cordonnier, J.B., Loukas, A., Jaggi, M.: On the Relationship between Self-Attention and Convolutional Layers. In: ICLR (2020)
15. Dai, A., Chang, A.X., Savva, M., Halber, M., Funkhouser, T., Nießner, M.: ScanNet: Richly-annotated 3D Reconstructions of Indoor Scenes. In: CVPR (2017)
16. Dao, T.: FlashAttention-2: Faster Attention with Better Parallelism and Work Partitioning. In: ICLR (2024)
17. Dao, T., Fu, D.Y., Ermon, S., Rudra, A., Ré, C.: FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness. In: NeurIPS (2022)
18. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: ImageNet: A large-scale hierarchical image database. In: CVPR (2009)

19. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., et al.: An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In: ICLR (2021)
20. Fang, Y., Sun, Q., Wang, X., Huang, T., Wang, X., Cao, Y.: EVA-02: A Visual Representation for Neon Genesis. Image and Vision Computing **149**, 105171 (2024)
21. Feng, T., Wang, W., Ma, F., Yang, Y.: LSK3DNet: Towards Effective and Efficient 3D Perception with Large Sparse Kernels. In: CVPR (2024)
22. Graham, B., Engelcke, M., Van Der Maaten, L.: 3D Semantic Segmentation with Submanifold Sparse Convolutional Networks. In: CVPR (2018)
23. Grattafiori, A., Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Vaughan, A., et al.: The Llama 3 Herd of Models. arXiv preprint arXiv:2407.21783 (2024)
24. Guo, M.H., Cai, J.X., Liu, Z.N., Mu, T.J., Martin, R.R., Hu, S.M.: PCT: Point Cloud Transformer. Computational Visual Media (2021)
25. Guo, Y., Wang, H., Hu, Q., Liu, H., Liu, L., Bennamoun, M.: Deep Learning for 3D Point Clouds: A Survey. IEEE TPAMI (2020)
26. He, K., Chen, X., Xie, S., Li, Y., Dollár, P., Girshick, R.: Masked Autoencoders Are Scalable Vision Learners. In: CVPR (2022)
27. Henry, A., Dachapally, P.R., Pawar, S.S., Chen, Y.: Query-Key Normalization for Transformers. In: Findings of the Association for Computational Linguistics: EMNLP 2020. pp. 4246–4253 (2020)
28. Heo, B., Park, S., Han, D., Yun, S.: Rotary Position Embedding for Vision Transformer. In: ECCV (2024)
29. Hinton, G., Vinyals, O., Dean, J.: Distilling the Knowledge in a Neural Network. arXiv preprint arXiv:1503.02531 (2015)
30. Huang, G., Sun, Y., Liu, Z., Sedra, D., Weinberger, K.Q.: Deep Networks with Stochastic Depth. In: ECCV (2016)
31. Ji, G., Weder, S., Engelmann, F., Pollefeys, M., Blum, H.: ARKit LabelMaker: A New Scale for Indoor 3D Scene Understanding. In: CVPR (2025)
32. Kaplan, J., McCandlish, S., Henighan, T., Brown, T.B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., Amodei, D.: Scaling Laws for Neural Language Models. arXiv preprint arXiv:2001.08361 (2020)
33. Knaebel, K., Schult, J., Hermans, A., Leibe, B.: Point2Vec for Self-Supervised Representation Learning on Point Clouds. In: GCPR (2023)
34. Knaebel, K., Yilmaz, K., de Geus, D., Hermans, A., Adrian, D., Linder, T., Leibe, B.: DINO in the Room: Leveraging 2D Foundation Models for 3D Segmentation. In: 3DV (2026)
35. Kolodiazhnyi, M., Vorontsova, A., Konushin, A., Rukhovich, D.: OneFormer3D: One Transformer for Unified Point Cloud Segmentation. In: CVPR (2024)
36. Lai, X., Chen, Y., Lu, F., Liu, J., Jia, J.: Spherical Transformer for LiDAR-Based 3D Recognition. In: CVPR (2023)
37. Lai, X., Liu, J., Jiang, L., Wang, L., Zhao, H., Liu, S., Qi, X., Jia, J.: Stratified Transformer for 3D Point Cloud Segmentation. In: CVPR (2022)
38. Lai, X., Yuan, Y., Chu, R., Chen, Y., Hu, H., Jia, J.: Mask-Attention-Free Transformer for 3D Instance Segmentation. In: ICCV (2023)
39. Loshchilov, I., Hutter, F.: Decoupled Weight Decay Regularization. In: ICLR (2019)
40. Lu, J., Deng, J.: Relation3D: Enhancing Relation Modeling for Point Cloud Instance Segmentation. In: CVPR (2025)
41. Lu, J., Deng, J., Wang, C., He, J., Zhang, T.: Query Refinement Transformer for 3D Instance Segmentation. In: ICCV (2023)

42. Nekrasov, A., Schult, J., Litany, O., Leibe, B., Engelmann, F.: Mix3D: Out-of-Context Data Augmentation for 3D Scenes. In: 3DV (2021)
43. Oquab, M., Darcet, T., Moutakanni, T., Vo, H.V., Szafraniec, M., Khalidov, V., et al.: DINOv2: Learning Robust Visual Features without Supervision. TMLR (2024)
44. Pang, Y., Wang, W., Tay, F.E., Liu, W., Tian, Y., Yuan, L.: Masked Autoencoders for Point Cloud Self-supervised Learning. In: ECCV (2022)
45. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: PyTorch: An Imperative Style, High-Performance Deep Learning Library. In: NeurIPS (2019)
46. Peng, B., Wu, X., Jiang, L., Chen, Y., Zhao, H., Tian, Z., Jia, J.: OA-CNNs: Omni-Adaptive Sparse CNNs for 3D Semantic Segmentation. In: CVPR (2024)
47. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In: CVPR (2017)
48. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. In: NeurIPS (2017)
49. Qian, G., Li, Y., Peng, H., Mai, J., Hammoud, H., Elhoseiny, M., Ghanem, B.: PointNeXt: Revisiting PointNet++ with Improved Training and Scaling Strategies. In: NeurIPS (2022)
50. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al.: Learning Transferable Visual Models From Natural Language Supervision. In: ICML (2021)
51. Radford, A., Kim, J.W., Xu, T., Brockman, G., McLeavey, C., Sutskever, I.: Robust Speech Recognition via Large-Scale Weak Supervision. In: ICML (2023)
52. Raghu, M., Unterthiner, T., Kornblith, S., Zhang, C., Dosovitskiy, A.: Do Vision Transformers See Like Convolutional Neural Networks? In: NeurIPS (2021)
53. Rombach, R., Blattmann, A., Lorenz, D., Esser, P., Ommer, B.: High-Resolution Image Synthesis With Latent Diffusion Models. In: CVPR (2022)
54. Ronneberger, O., Fischer, P., Brox, T.: U-Net: Convolutional Networks for Biomedical Image Segmentation. In: MICCAI (2015)
55. Rozenberszki, D., Litany, O., Dai, A.: Language-Grounded Indoor 3D Semantic Segmentation in the Wild. In: ECCV (2022)
56. Schuhmann, C., Beaumont, R., Vencu, R., Gordon, C., Wightman, R., Cherti, M., Coombes, T., Katta, A., Mullis, C., Wortsman, M., et al.: LAION-5B: An Open Large-Scale Dataset for Training Next Generation Image-Text Models. NeurIPS (2022)
57. Schuhmann, C., Vencu, R., Beaumont, R., Kaczmarczyk, R., Mullis, C., Katta, A., Coombes, T., Jitsev, J., Komatsuzaki, A.: LAION-400M: Open Dataset of CLIP-Filtered 400 Million Image-Text Pairs. arXiv preprint arXiv:2111.02114 (2021)
58. Schult, J., Engelmann, F., Hermans, A., Litany, O., Tang, S., Leibe, B.: Mask3D: Mask Transformer for 3D Semantic Instance Segmentation. In: ICRA (2023)
59. Shah, J., Bikshandi, G., Zhang, Y., Thakkar, V., Ramani, P., Dao, T.: FlashAttention-3: Fast and Accurate Attention with Asynchrony and Low-precision. In: NeurIPS (2024)
60. Shin, S., Zhou, K., Vankadari, M., Markham, A., Trigoni, N.: Spherical Mask: Coarse-to-Fine 3D Point Cloud Instance Segmentation with Spherical Representation. In: CVPR (2024)
61. Siméoni, O., Vo, H.V., Seitzer, M., Baldassarre, F., Oquab, M., Jose, C., Khalidov, V., Szafraniec, M., Yi, S., Ramamonjisoa, M., Massa, F., Haziza, D., Wehrstedt, L., Wang, J., Darcet, T., Moutakanni, T., Sentana, L., Roberts, C., Vedaldi, A.,

Tolan, J., Brandt, J., Couprie, C., Mairal, J., Jégou, H., Labatut, P., Bojanowski, P.: DINOv3. arXiv preprint arXiv:2508.10104 (2025)
62. Smith, L.N., Topin, N.: Super-Convergence: Very Fast Training of Neural Networks Using Large Learning Rates. In: Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications (2019)
63. SpconvContributors: Spconv: Spatially Sparse Convolution Library. `https://github.com/traveller59/spconv` (2022)
64. Steiner, A.P., Kolesnikov, A., Zhai, X., Wightman, R., Uszkoreit, J., Beyer, L.: How to train your ViT? Data, Augmentation, and Regularization in Vision Transformers. TMLR (2022)
65. Su, J., Ahmed, M., Lu, Y., Pan, S., Bo, W., Liu, Y.: RoFormer: Enhanced Transformer with Rotary Position Embedding. Neurocomputing (2024)
66. Sun, J., Qing, C., Tan, J., Xu, X.: Superpoint Transformer for 3D Scene Instance Segmentation. In: AAAI (2023)
67. Sun, P., Kretzschmar, H., Dotiwalla, X., Chouard, A., Patnaik, V., Tsui, P., Guo, J., Zhou, Y., Chai, Y., Caine, B., Vasudevan, V., Han, W., Ngiam, J., Zhao, H., Timofeev, A., Ettinger, S., Krivokon, M., Gao, A., Joshi, A., Zhang, Y., Shlens, J., Chen, Z., Anguelov, D.: Scalability in Perception for Autonomous Driving: Waymo Open Dataset. In: CVPR (2020)
68. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the Inception Architecture for Computer Vision. In: CVPR (2016)
69. Tang, H., Liu, Z., Zhao, S., Lin, Y., Lin, J., Wang, H., Han, S.: Searching Efficient 3D Architectures with Sparse Point-Voxel Convolution. In: ECCV (2020)
70. Tang, H., Liu, Z., Zhao, S., Lin, Y., Lin, J., Wang, H., Han, S.: Searching Efficient 3D Architectures with Sparse Point-Voxel Convolution. In: ECCV (2020)
71. Team, G., Kamath, A., Ferret, J., Pathak, S., Vieillard, N., Merhej, R., Perrin, S., Matejovicova, T., Ramé, A., Rivière, M., et al.: Gemma 3 Technical Report. arXiv preprint arXiv:2503.19786 (2025)
72. Thomas, H., Qi, C.R., Deschaud, J.E., Marcotegui, B., Goulette, F., Guibas, L.J.: KPConv: Flexible and Deformable Convolution for Point Clouds. In: CVPR (2019)
73. Thomas, H., Tsai, Y.H.H., Barfoot, T.D., Zhang, J.: KPConvX: Modernizing Kernel Point Convolution with Kernel Attention. In: CVPR (2024)
74. Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., Jégou, H.: Training Data-Efficient Image Transformers & Distillation Through Attention. In: ICML (2021)
75. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is All You Need. In: NeurIPS (2017)
76. Wang, P.S.: OctFormer: Octree-based Transformers for 3D Point Clouds. ACM Transactions on Graphics (SIGGRAPH) **42**(4) (2023)
77. Wightman, R.: PyTorch Image Models. `https://github.com/rwightman/pytorch-image-models` (2019). `https://doi.org/10.5281/zenodo.4414861`
78. Wu, W., Qi, Z., Fuxin, L.: Pointconv: Deep Convolutional Networks on 3D Point Clouds. In: CVPR (2019)
79. Wu, X., Jiang, L., Wang, P.S., Liu, Z., Liu, X., Qiao, Y., Ouyang, W., He, T., Zhao, H.: Point Transformer V3: Simpler, Faster, Stronger. In: CVPR (2024)
80. Wu, X., Lao, Y., Jiang, L., Liu, X., Zhao, H.: Point Transformer V2: Grouped Vector Attention and Partition-based Pooling. In: NeurIPS (2022)
81. Wu, X., Tian, Z., Wen, X., Peng, B., Liu, X., Yu, K., Zhao, H.: Towards Large-scale 3D Representation Learning with Multi-dataset Point Prompt Training. In: CVPR (2024)

82. Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., Xiao, J.: 3D ShapeNets: A Deep Representation for Volumetric Shapes. In: CVPR (2015)
83. Yang, Y.Q., Guo, Y.X., Xiong, J.Y., Liu, Y., Pan, H., Wang, P.S., Tong, X., Guo, B.: Swin3D: A Pretrained Transformer Backbone for 3D Indoor Scene Understanding. arXiv preprint arXiv:2304.06906 (2023)
84. Yao, L., Wang, Y., Liu, M., Chau, L.P.: SGIFormer: Semantic-Guided and Geometric-Enhanced Interleaving Transformer for 3D Instance Segmentation. IEEE Transactions on Circuits and Systems for Video Technology (2024)
85. Yeshwanth, C., Liu, Y.C., Nießner, M., Dai, A.: ScanNet++: A High-Fidelity Dataset of 3D Indoor Scenes. In: ICCV (2023)
86. Yilmaz, K., Schult, J., Nekrasov, A., Leibe, B.: Mask4Former: Mask Transformer for 4D Panoptic Segmentation. In: ICRA (2024)
87. Yu, X., Tang, L., Rao, Y., Huang, T., Zhou, J., Lu, J.: Point-BERT: Pre-Training 3D Point Cloud Transformers With Masked Point Modeling. In: CVPR (2022)
88. Yue, Y., Robert, D., Wang, J., Hong, S., Wegner, J.D., Rupprecht, C., Schindler, K.: LitePT: Lighter Yet Stronger Point Transformer. arXiv preprint arXiv:2512.13689 (2025)
89. Zhao, B., Song, R., Liang, J.: Cumulative Spatial Knowledge Distillation for Vision Transformers. In: ICCV (2023)
90. Zhao, H., Jiang, L., Jia, J., Torr, P.H., Koltun, V.: Point Transformer. In: ICCV (2021)
91. Zheng, J., Zhang, J., Li, J., Tang, R., Gao, S., Zhou, Z.: Structured3D: A Large Photo-realistic Dataset for Structured 3D Modeling. In: ECCV (2020)
92. Zhu, X., Zhou, H., Wang, T., Hong, F., Ma, Y., Li, W., Li, H., Lin, D.: Cylindrical and Asymmetrical 3D Convolution Networks for LiDAR Segmentation. In: CVPR (2021)