LitePT: Lighter Yet Stronger Point Transformer
Supplementary Material

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

In this Appendix, we provide detailed architecture of LitePT (Sec. A), detailed experimental settings (Sec. B), additional experiments (Sec. C), and visualization of LitePT's predictions for 3D semantic segmentation, 3D instance segmentation, and 3D object detection (Sec. D).

A. Detailed Architecture

Our full architecture is shown in Fig. 1. It follows U-Net-style [18] encoder-decoder design with skip connections, and is organized into five stages. Adjacent encoder (or decoder) stages are connected via pooling (or unpooling) blocks. We apply our stage-tailored design on the encoder: the first three stages use convolution blocks, while the final two use attention blocks. For LitePT-S/B/L, each stage in the decoder contains only an unpooling block. For LitePT-S*, we mirror the stage-tailored design in the decoder as well. Detailed architecture specifications can be found in Tab. 1. Below, we describe each block type in detail.

Attention block. Each attention block consists of a PointROPE attention module and a feed-forward network (FFN) module. Following the pre-norm [27] convention, a LayerNorm [1] is placed before both the attention and FFN modules. The FFN uses a hidden dimension four times larger than the channel dimension of its stage. We observe that adding an extra LayerNorm before the attention block further stabilizes the training. In the PointROPE attention module (Fig. 2), input point features are projected to query (Q), key (K), and value (V) representations. PointROPE is computed from point coordinates P and applied to Q and K, leaving V unchanged. The resulting "rotated" Q′ and K′ are fed into a standard scaled dot-product multi-head attention together with V, followed by a linear projection to produce the final output embeddings. Our PointROPE implementation is compatible with FlashAttention [6, 7, 20], which we use in our model. We apply PointROPE to locally-aggregated groups of 1024 points, formed using the same serialization sorting strategy as [26].

Convolution block. The convolution block includes of a single sparse convolution layer [4, 8] with a kernel size of $3 \times 3 \times 3$, followed by a linear projection layer and a LayerNorm [1] layer. A residual connection [9] links the block's input and output.

Pooling and unpooling blocks. We adopt the grid pooling and unpooling operation from [25]. During pooling, points are divided into non-overlapping partitions. Point features are first projected by a linear layer, then points within the same partition are max-pooled, followed by a GELU [10] activation and a BatchNorm layer [11]. The pooling stride is set to 2 at each stage, reducing the spatial resolution by a factor of 2 each time. During unpooling, point features from the current decoder stage and the corresponding encoder stage are each passed through their own linear layer, GELU activation, and BatchNorm. The resulting features are then merged through a skip connection via summation.

B. Detailed Experimental Settings

For indoor datasets, we use RGB and surface normals as input features. For outdoor datasets, where RGB and normal information are unavailable, we use xyz coordinates and intensity (plus elongation for object detection). Following common practice [4, 25, 26], we first downsample the point cloud on a grid. For 3D segmentation tasks, we set the grid size to 0.02m for indoor scenes and 0.05m for outdoor scenes. For 3D object detection, we adopt grid sizes of 0.32m in the xy plane and 6m along the z axis, consistent with [15, 26]. Detailed training configurations for semantic segmentation, instance segmentation and object detection are provided in Tab. 2, Tab. 3, and Tab. 4, respectively.

C. Additional Experiments

C.1. Further Ablation on PointROPE

Spherical *vs.* Cartesian coordinates. In PointROPE, we divide each point's feature embedding into three equal subspaces and then apply the standard 1D ROPE [23] embedding to each subspace using the respective Cartesian coordinates. Here, we investigate an alternative design that uses spherical coordinates. Specifically, we transform each

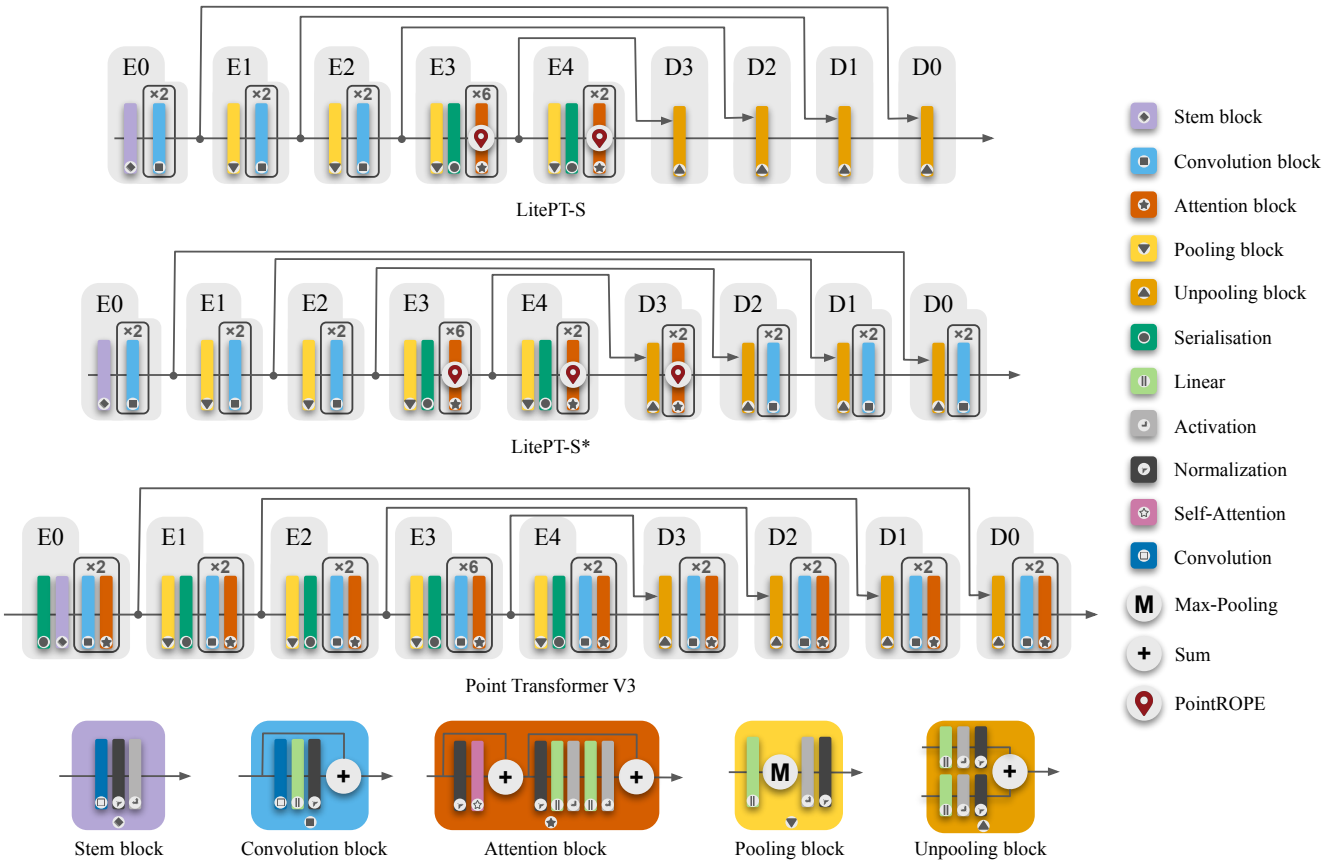


Figure 1. **Detailed architectures.** We illustrate the full pipelines of LitePT-S, LitePT-S*, Point Transformer V3 [26], and the building blocks of each architecture.

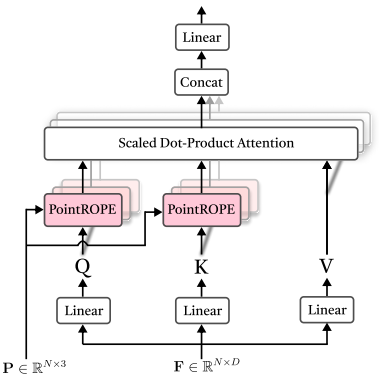


Figure 2. **PointROPE attention.** We apply PointROPE to query and key before standard scaled dot-product attention.

point (x_i, y_i, z_i) into spherical coordinates (r_i, θ_i, ϕ_i), using the mean of all points as the origin. We then apply 1D ROPE using r_i, θ_i and ϕ_i separately and concatenate the resulting embeddings. The motivation is that spherical coordinates decouple radial distance and angular structure, which could potentially make positional relationships easier to learn. However, as shown in Tab. 5, we empirically find that PointROPE in spherical coordinates is effective but offers no improvement over Cartesian coordinates, while adding additional computational overhead. Therefore, we retain our simpler per-axis Cartesian design.

Subdivision of the input space. For each attention head (with head dimension 18), we split the embedding evenly across three axes (x_i, y_i, z_i). Here we explore the impact of different subdivisions on each axis. In addition to equal split $(6:6:6)$, we try emphasizing the z axis $(4:4:10)$ and emphasizing the xy axes $(8:8:2)$. As shown in Tab. 5, uneven splits lead to suboptimal performance compared with equal weighting. This suggests that positional information along all three axes is similarly important, and manual reweighting is unnecessary.

Other positional encodings. We additionally compare PointROPE with absolute positional encoding (APE) using sinusoidal functions, and with relative positional encoding (RPE) on nuScenes, in Tab. 6. PointROPE achieves the clearly highest mIoU while introducing the smallest increase in latency.

C.2. Cross-dataset Transfer

To evaluate cross-dataset transfer performance, we first pretrain PTv3 and LitePT-S for semantic segmentation on

	LitePT-S	LitePT-S*	LitePT-B	LitePT-L
stem	C=36, K=5×5×5	C=36, K=5×5×5	C=36, K=5×5×5	C=36, K=5×5×5
E0	[C=36, K=3×3×3] ×2	[C=36, K=3×3×3] ×2	[C=54, K=3×3×3] ×3	[C=72, K=3×3×3] ×3
E1	pool stride 2 [C=72, K=3×3×3] ×2	pool stride 2 [C=72, K=3×3×3] ×2	pool stride 2 [C=108, K=3×3×3] ×3	pool stride 2 [C=144, K=3×3×3] ×3
E2	pool stride 2 [C=144, K=3×3×3] ×2	pool stride 2 [C=144, K=3×3×3] ×2	pool stride 2 [C=216, K=3×3×3] ×3	pool stride 2 [C=288, K=3×3×3] ×3
E3	pool stride 2 [C=252, H=14, b=100, F=4, N=1024] ×6	pool stride 2 [C=252, H=14, b=100, F=4, N=1024] ×6	pool stride 2 [C=432, H=24, b=100, F=4, N=1024] ×12	pool stride 2 [C=576, H=32, b=100, F=4, N=1024] ×12
E4	pool stride 2 [C=504, H=28, b=100, F=4, N=1024] ×2	pool stride 2 [C=504, H=28, b=100, F=4, N=1024] ×2	pool stride 2 [C=576, H=32, b=100, F=4, N=1024] ×3	pool stride 2 [C=864, H=48, b=100, F=4, N=1024] ×3
D3	unpool C=252	unpool C=252 [C=252, H=14, b=100, F=4, N=1024] ×2	unpool C=432	unpool C=576
D2	unpool C=144	unpool C=144 [C=144, K=3×3×3] ×2	unpool C=216	unpool C=288
D1	unpool C=72	unpool C=72 [C=72, K=3×3×3] ×2	unpool C=108	unpool C=144
D0	unpool C=72	unpool C=72 [C=72, K=3×3×3] ×2	unpool C=72	unpool C=72
#Params	12.7M	16.0M	45.1M	85.9M

Table 1. **Detailed architecture specifications.** C: channel dimension, K: kernel size in the convolution block, H: number of head, b: base frequency of PointROPE, F: MLP ratio in the FFN module, N: number of points in local group.

	nuScenes [3]	Waymo [24]	ScanNet [5]	Structured3D [28]
Input feature	XYZ+Intensity		RGB+Normal	
Grid size	0.05m		0.02m	
Head (framework)	Linear segmentor		Linear segmentor	
Loss	CrossEntropy+Lovasz [2]		CrossEntropy+Lovasz [2]	
Optimizer	AdamW [16]		AdamW [16]	
Weight decay	0.005		0.05	
Scheduler	OneCycleLR [22]		OneCycleLR [22]	
Learning rate	0.002	0.006	0.012	
Block lr rate	0.0002	0.0006	0.0012	
Batch size	12	12	48	
Epochs	50	1200 (800)	200	
Num GPUs	4	4	16	
Data augmentation	Random rotate, random scale random flip, random jitter		Random shift, random dropout random rotate, random scale random flip, random jitter elastic distortion, color auto contrast color jitter, sphere crop color normalization	

Table 2. **Detailed training settings for semantic segmentation.**

	ScanNet [5]	ScanNet200 [19]
Input feature	RGB+Normal	
Grid size	0.02m	
Head (framework)	PointGroup [12]	
Loss	check PointGroup [12]	
Optimizer	AdamW [16]	
Weight decay	0.05	
Scheduler	OneCycleLR [22]	
Learning rate	0.006	
Block lr rate	0.0006	
Batch size	12	
Epochs	800	
Num GPUs	4	
Data augmentation	Random shift, random dropout random rotate, random scale random flip, random jitter elastic distortion, color auto contrast color jitter sphere crop, color normalization	

Table 3. **Detailed training settings for instance segmentation.**

	Object Detection Waymo [24]
Input feature	XYZ+Intensity+Elongation
Grid size	(0.32m, 0.32m, 6.0m)
Head (framework)	CenterPoint-Pillar [13]
Loss	check CenterPoint-Pillar [13]
Optimizer	Adam [17]
Weight decay	0.01
Scheduler	OneCycleLR [22]
Learning rate	0.006
Block lr rate	0.006
Batch size	64
Epochs	40
Num GPUs	16
Data augmentation	Random flip, random rotate random scale

Table 4. **Detailed training settings for object detection.**

Structured3D, the largest dataset in our benchmark suite. We then freeze the pretrained encoders and test on ScanNet

	mIoU	mAcc
w/o PointROPE	79.6	86.5
Spherical	80.7	87.1
Cartesian	**82.2**	**88.1**
$x:y:z=6:6:6$	**82.2**	**88.1**
$x:y:z=4:4:10$	80.3	86.8
$x:y:z=8:8:2$	80.3	86.7

Table 5. **Additional ablation on PointROPE on nuScenes.**

	w/o PosEnc	APE (sinusoidal)	RPE	PointROPE
mIoU	79.6	80.2	80.6	**82.2**
Latency	20.7 ms	23.3 ms	27.6 ms	**21.5 ms**

Table 6. **Comparison with other positional encoding schemes on nuScenes.**

Method	#Param	mIoU	mAcc
PTv3 [26]	46.1M	57.1	69.5
LitePT-S	12.7M	**65.0**	**77.2**

Table 7. **Cross-dataset transfer.** Models are pretrained on Structured3D and evaluated on ScanNet validation set using linear probing.

with a linear probe on the multi-scale features. As shown in Tab. 7, LitePT outperforms PTv3 by 7.9 pp in mIoU, indicating better generalisation. We attribute the cross-dateset generalization to PointROPE, which preserves relative spatial relationships across scenes with differing layouts and sensor characteristics. This property enables the model to learn more transferable representations, reducing overfitting to dataset-specific structures and improving robustness under domain shift.

C.3. Comparison with PointMamba

We additionally compare LitePT with the Mamba-based point cloud backbone PointMamba (PM) [14]. For small, object-centric datasets, PM is indeed a viable alternative, see shape classification results for ModelNet40 in Tab. 8 ①. Both models are trained from scratch and evaluated without voting.

To adopt PM to scene-level semantic segmentation, we fix the $\frac{\#\texttt{group\_tokens}}{\#\texttt{input\_points}}$ ratio of the best-reported model and progressively increase the number of input points (Fig. 3). We observe that PM's reliance on farthest point sampling (FPS), k-NN, and PointNet-style feature propagation leads to rapid growth in compute and memory, limiting its scalability to larger scenes. In Tab. 8 ②, we train PM on ScanNet with 640 tokens per scene, the maximum feasible on an RTX 4090, resulting in a 16 pp mIoU drop compared to LitePT.

	① ModelNet40				② ScanNet
	#Params	Latency	Memory	OA (%)	mIoU
PointMamba	12.3 M	**10.5 ms**	0.20 G	92.4	60.5
LitePT-S (Ours)	12.7 M	11.7 ms	**0.13 G**	**92.5**	**76.5**

Table 8. **Comparison with PointMamba.**

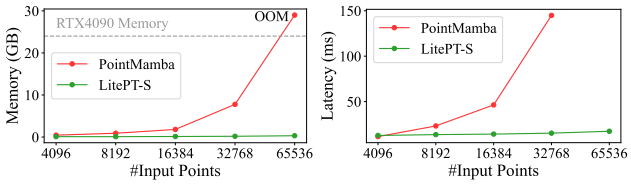


Figure 3. **LitePT-S vs. PointMamba efficiency on ScanNet.**

C.4. Additional Results for LitePT-L

In Tab. 9, we show LitePT-L results on key benchmarks. We note that, in accordance with established scaling laws, model scaling should be considered in conjunction with data volume. Upgrading from LitePT-S to LitePT-L consistently improves performance. As expected, the gain strongly depends on the dataset size (as denoted by #Points). For instance, scaling up to LitePT-L yields a +1.8 pp gain in mIoU for Structured3D (9.5 billion points) whereas the improvement is only +0.4 pp for nuScenes (1 billion points), as there is not enough data to fully exploit the higher capacity.

	Semantic Segmentation (mIoU)			Instance Segmentation (mAP)
	Structured3D	Waymo	nuScenes	ScanNet
#Points (Billion)	9.55	4.15	0.98	0.17
PTv3	82.4	71.3	80.4	40.9
LitePT-S	83.6	73.1	82.2	41.7
LitePT-L	**85.4** (+1.8)	**74.0** (+0.9)	**82.6** (+0.4)	**42.3** (+0.6)

Table 9. **More results for LitePT-L.**

C.5. Benchmarking Outdoor Efficiency

We evaluate the efficiency of semantic segmentation on Waymo, a large-scale outdoor LiDAR dataset, in Tab. 10. Similar to our observation on indoor datasets, LitePT-S remains more efficient than PTv3, with $> 2\times$ faster runtime at inference.

		Training		Inference		
Method	#Params	Latency	Memory	Latency	Memory	mIoU
PTv3	46.1 M	153 ms	14.1 G	100 ms	7.1 G	71.3
LitePT-S (Ours)	12.7 M	**86 ms**	**9.4 G**	**47 ms**	**5.4 G**	**73.1**

Table 10. **Outdoor Efficiency comparison on Waymo.**

C.6. Chunking and Test-Time Augmentation

In the main paper, we report semantic segmentation results following the same evaluation protocol as prior works [25, 26] to ensure a fair comparison. The testing pipeline applies chunking and test-time augmentations (TTA). Specifically, each augmented sample is partitioned into overlapping chunks, ensuring that every point is assigned to at least one chunk during grid sampling. The model is then run on each chunk individually, and the final label of each point is aggregated by voting across the predictions from all chunks it appears in. Although this multi-run and TTA protocol is common practice and is known to boost performance [21], it obscures the intrinsic merits of the underlying backbone. To communicate performance in a simpler single-pass setting useful for downstream users, we additionally report results for PTv3 and LitePT-S without TTA or chunking in Tab. 11. Overall, removing chunking and TTA reduces performance by roughly 2% mIoU for both methods.

Method	#Param	mIoU	mAcc
PTv3 [26]	46.1M	80.4	87.2
LitePT-S	12.7M	**82.2**	**88.1**
PTv3 [26] (w/o chunking and TTA)	46.1M	78.3	86.0
LitePT-S (w/o chunking and TTA)	12.7M	**80.4**	**86.9**

Table 11. **Semantic segmentation on nuScenes without chunking and TTA.**

D. Visualization

We visualize sample predictions of LitePT on three tasks: 3D semantic segmentation (Figs. 4 to 7), 3D instance segmentation (Fig. 8), and 3D object detection (Fig. 9).

References

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer Normalization. *arXiv preprint arXiv:1607.06450*, 2016. 1

[2] Maxim Berman, Amal Rannen Triki, and Matthew B Blaschko. The Lovász-Softmax Loss: A Tractable Surrogate for the Optimization of the Intersection-Over-Union Measure in Neural Networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 4

[3] 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. 4

[4] 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. 1

[5] 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. 4

[6] Tri Dao. FlashAttention-2: Faster Attention with Better Parallelism and Work Partitioning . *arXiv preprint arXiv:2307.08691*, 2023. 1

[7] Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness. *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. 1

[8] 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. 1

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1

[10] Dan Hendrycks and Kevin Gimpel. Gaussian Error Linear Units (GELUs). *arXiv preprint arXiv:1606.08415*, 2016. 1

[11] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *International Conference on Machine Learning (ICML)*, 2015. 1

[12] 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. 4

[13] 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. 4

[14] Dingkang Liang, Xin Zhou, Wei Xu, Xingkui Zhu, Zhikang Zou, Xiaoqing Ye, Xiao Tan, and Xiang Bai. PointMamba: A Simple State Space Model for Point Cloud Analysis. *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. 4

[15] 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. 1

[16] Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization. *International Conference on Learning Representations (ICLR)*, 2019. 4

[17] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations (ICLR)*, 2015. 4

[18] 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. 1

[19] David Rozenberszki, Or Litany, and Angela Dai. Language-Grounded Indoor 3D Semantic Segmentation in the Wild. In *European Conference on Computer Vision (ECCV)*, 2022. 4

[20] Jay Shah, Ganesh Bikshandi, Ying Zhang, Vijay Thakkar, Pradeep Ramani, and Tri Dao. FlashAttention-3: Fast and Accurate Attention with Asynchrony and Low-precision. *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. 1

[21] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv preprint arXiv:1409.1556*, 2014. 5

[22] Leslie N Smith and Nicholay Topin. Super-Convergence: Very Fast Training of Neural Networks Using Large Learning Rates. In *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, 2019. 4

[23] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. RoFormer: Enhanced Transformer with Rotary Position Embedding. *Neurocomputing*, 2024. 1

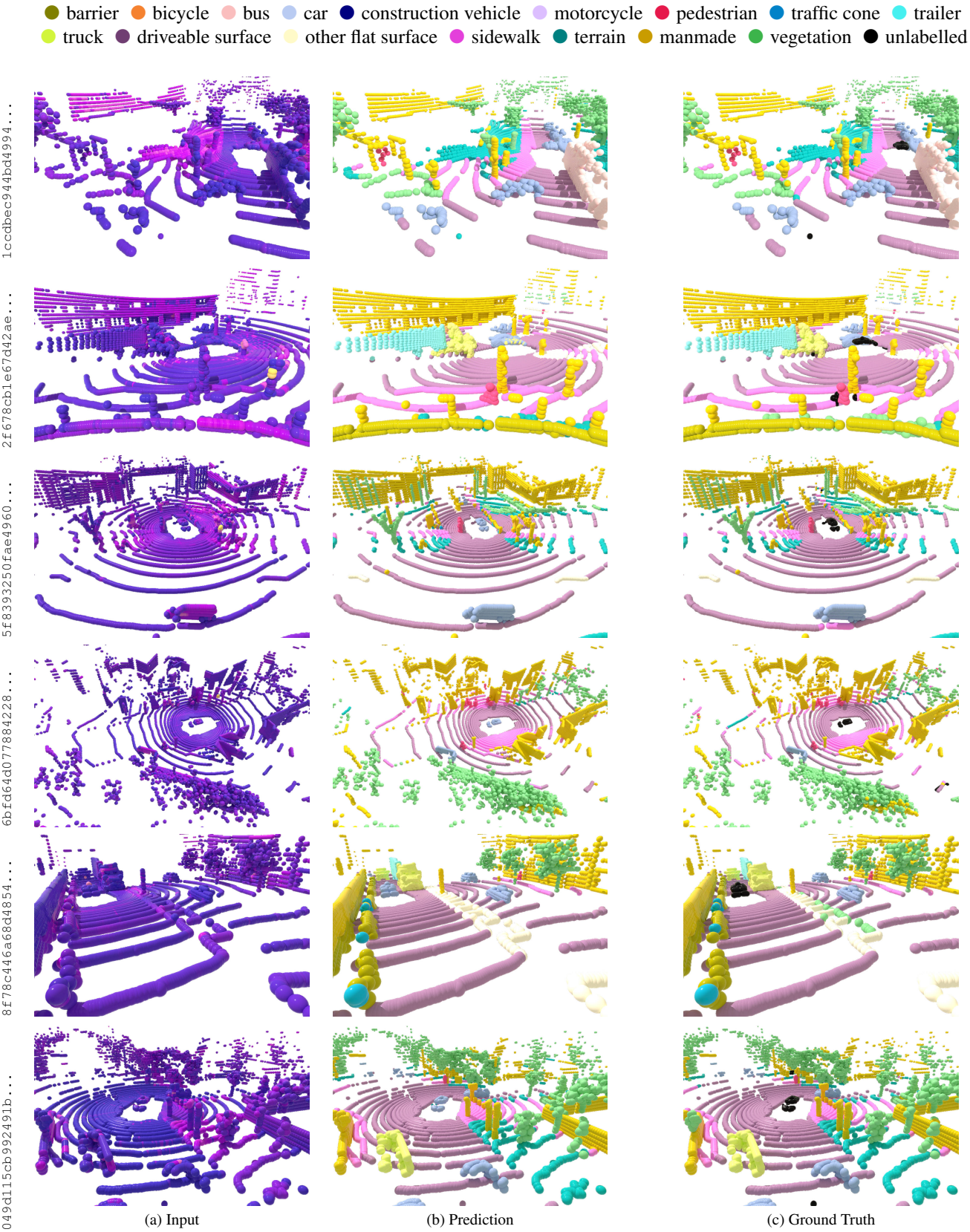
[24] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in Perception for Autonomous Driving: Waymo Open Dataset. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 4

[25] 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. 1, 5

[26] 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, 4, 5

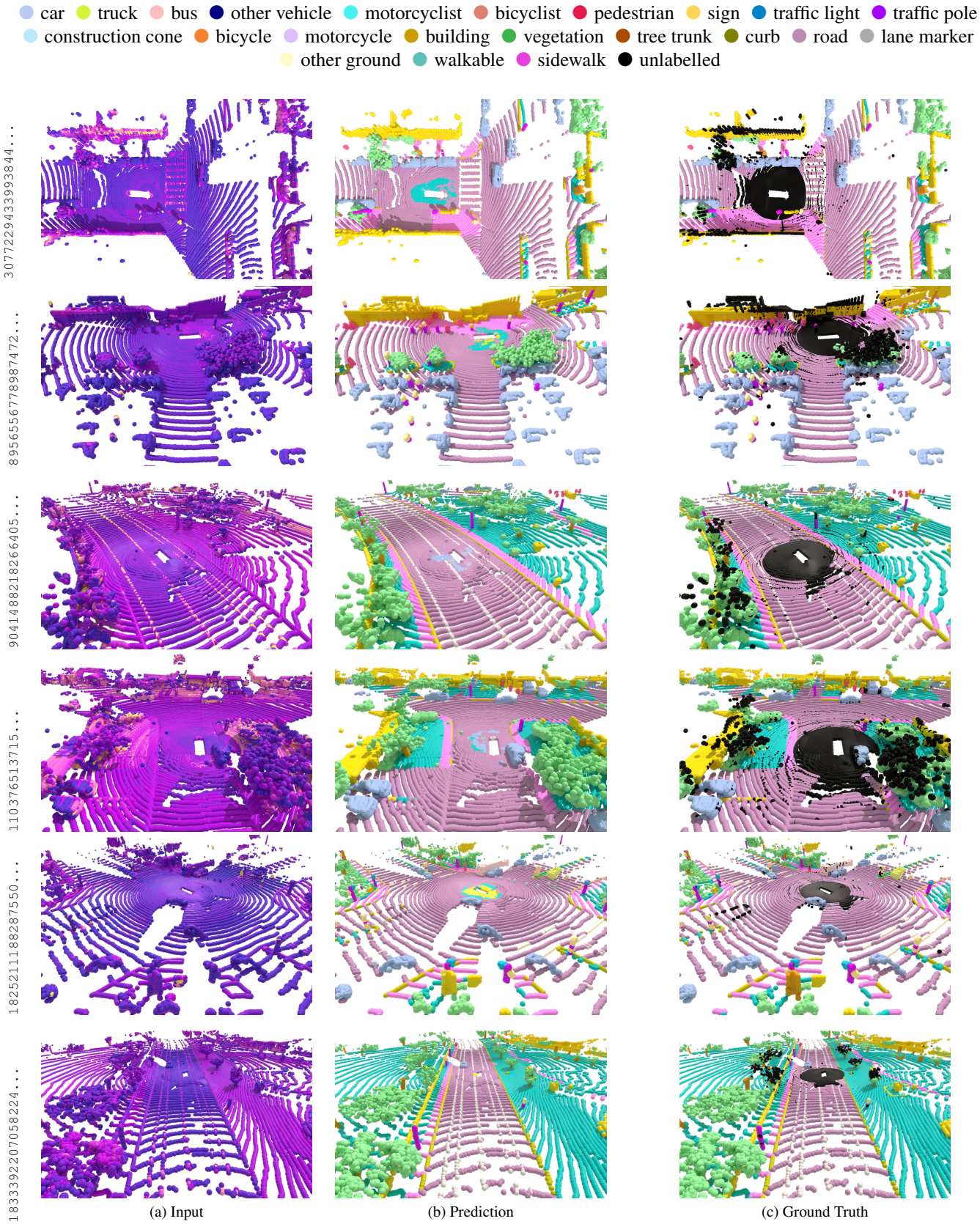
[27] Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tieyan Liu. On Layer Normalization in the Transformer Architecture. In *International Conference on Machine Learning (ICML)*, 2020. 1

[28] 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. 4



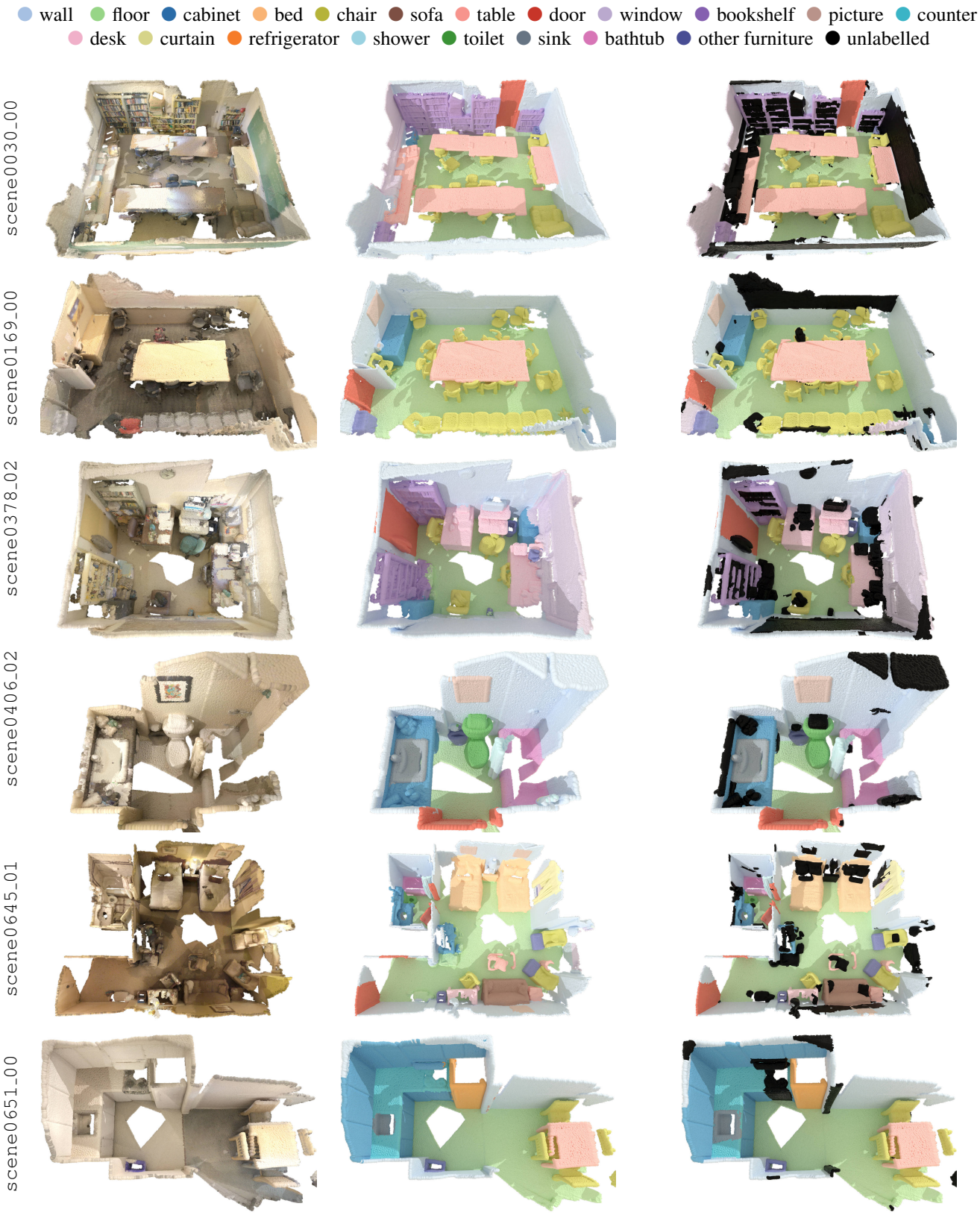
(a) Input (b) Prediction (c) Ground Truth

Figure 4. **nuScenes semantic segmentation.** We present various scenes of the nuScenes validation set: the input point cloud colored by LiDAR intensity, the semantic segmentation from LitePT-S, and the corresponding ground truth.



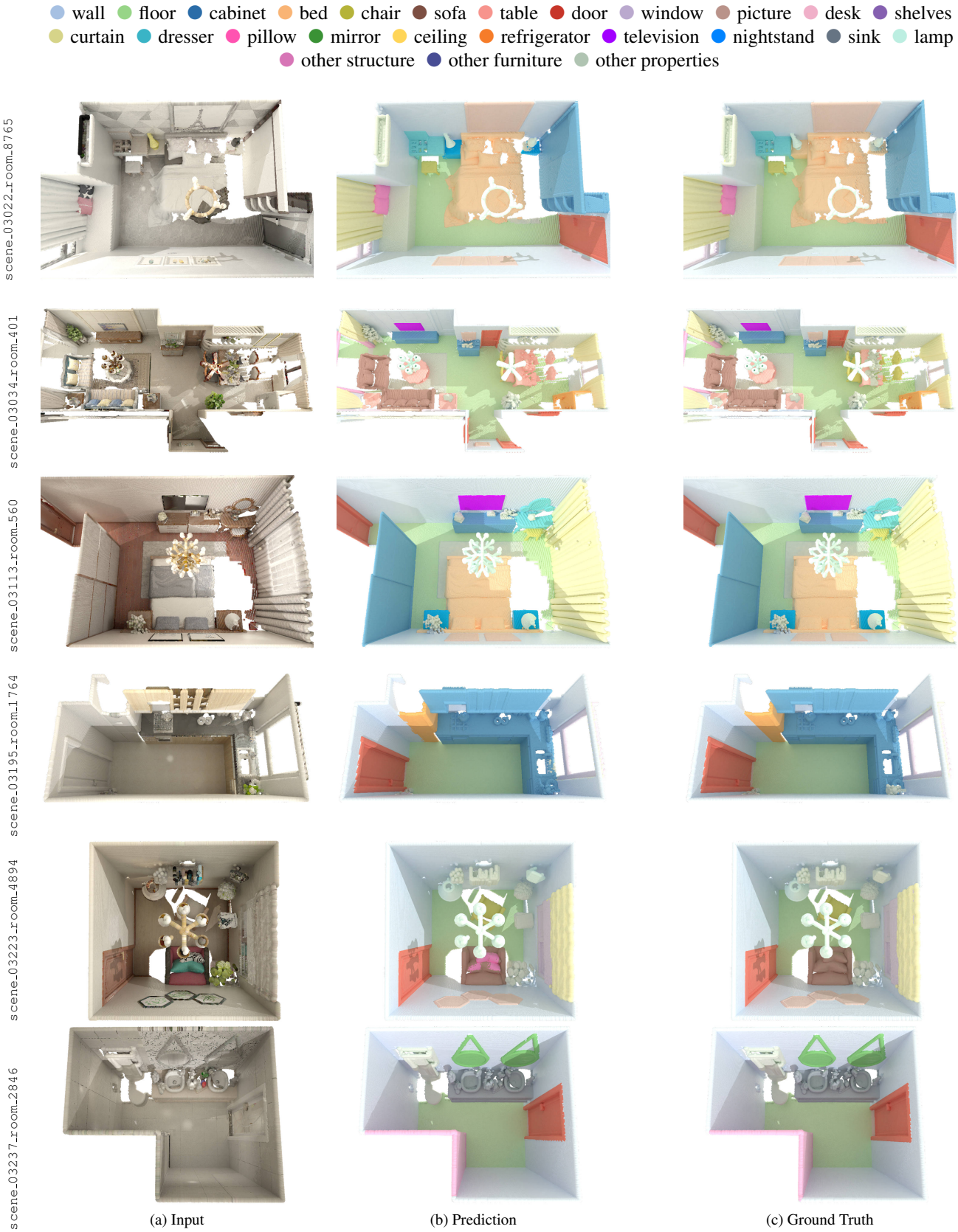
(a) Input (b) Prediction (c) Ground Truth

Figure 5. **Waymo semantic segmentation.** We present various scenes of the Waymo validation set: the input point cloud colored by LiDAR intensity, the semantic segmentation from LitePT-S, and the corresponding ground truth.



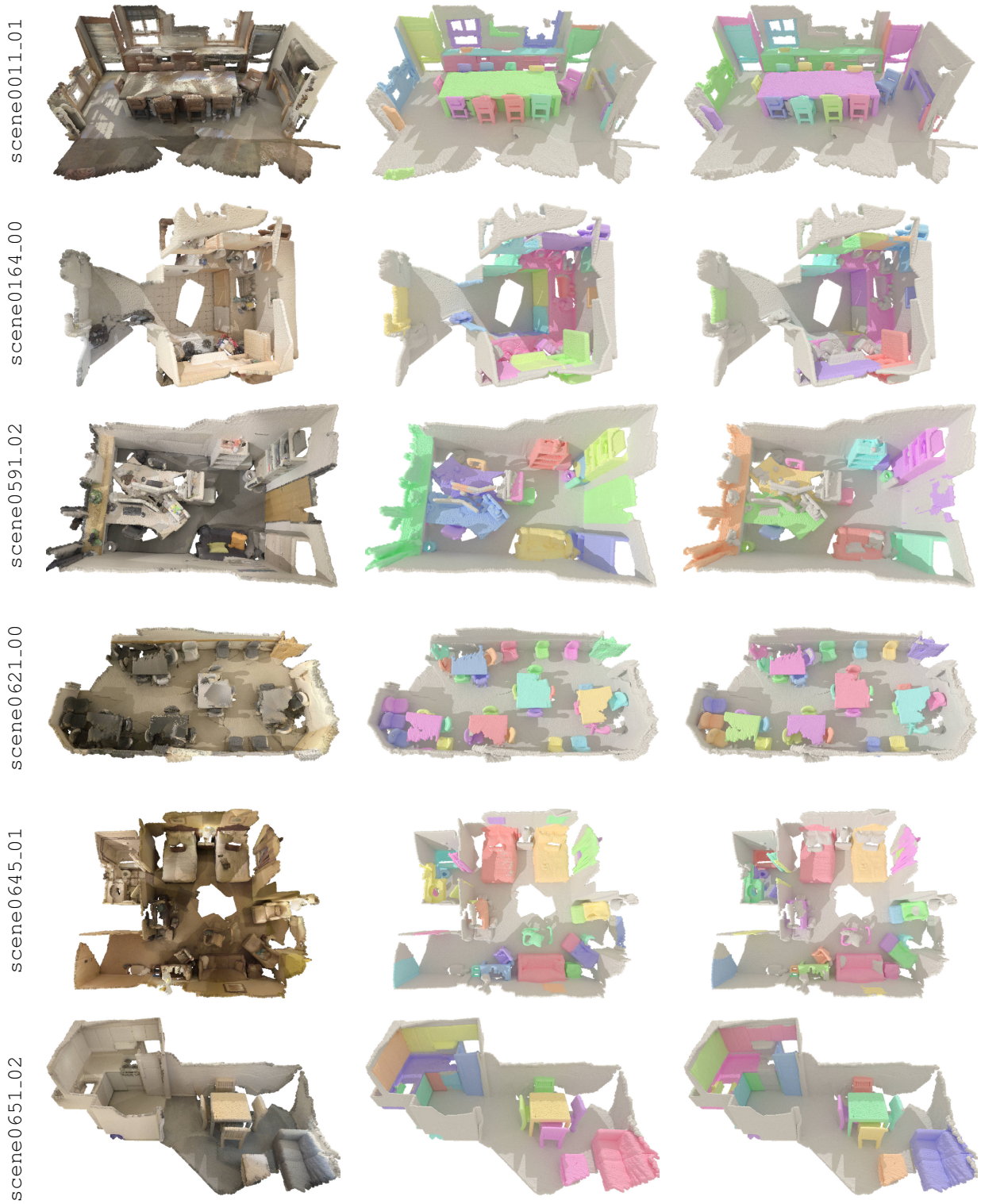
(a) Input (b) Prediction (c) Ground Truth

Figure 6. **ScanNet semantic segmentation.** We present various scenes of the ScanNet validation set: the input point cloud, the semantic segmentation from LitePT-S, and the corresponding ground truth.



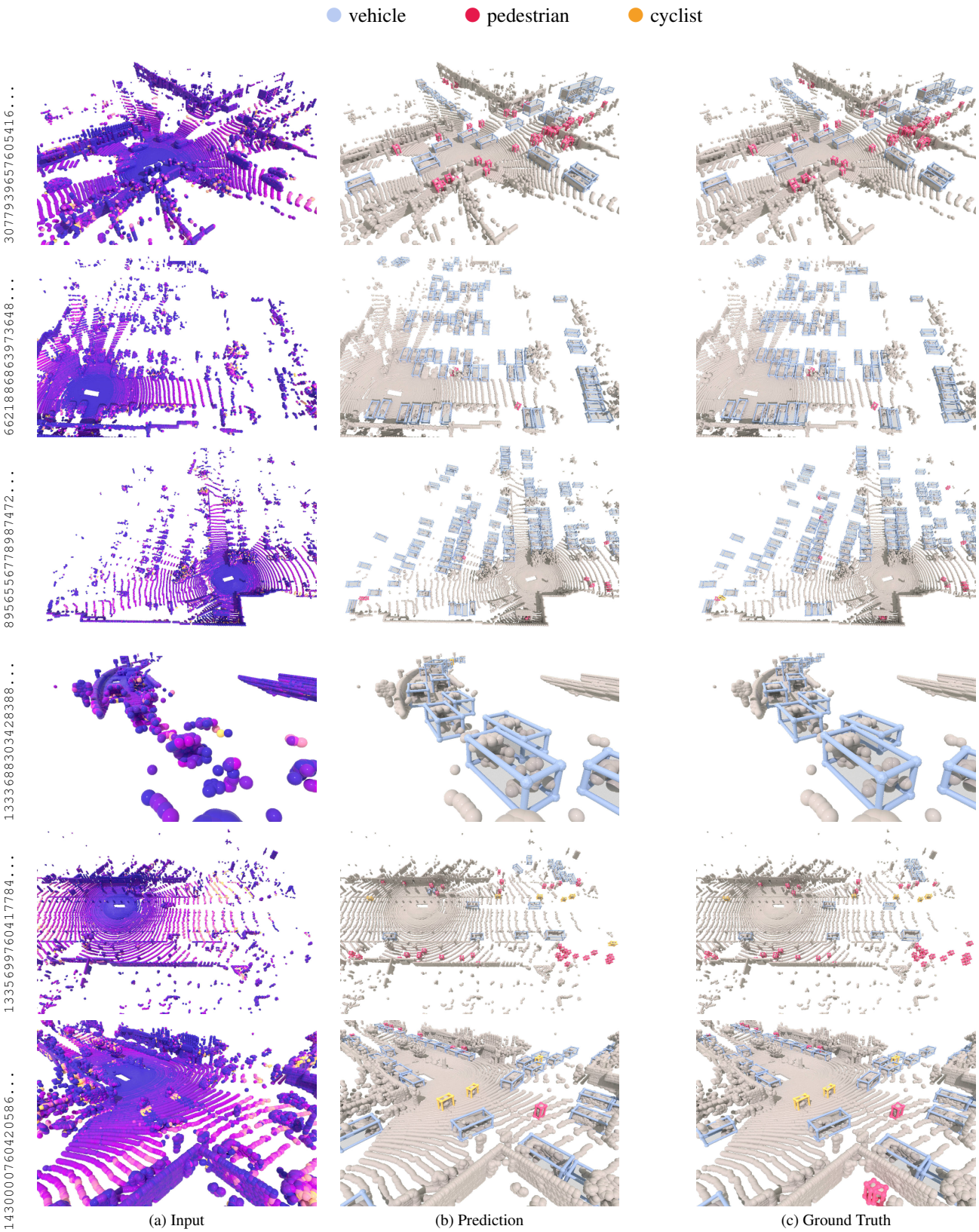
(a) Input (b) Prediction (c) Ground Truth

Figure 7. **Structured3D semantic segmentation.** We present various scenes of the Structured3D validation set: the input point cloud, the semantic segmentation from LitePT-S, and the corresponding ground truth.



(a) Input (b) Prediction (c) Ground Truth

Figure 8. **ScanNet instance segmentation.** We present various scenes of the ScanNet validation set: the input point cloud, the instance segmentation from LitePT-S*, and the corresponding ground truth. Colors for each instance are randomly assigned.



(a) Input (b) Prediction (c) Ground Truth

Figure 9. **Waymo object detection.** We present various scenes of the Waymo validation set: the input point cloud, the object detections from LitePT, and the corresponding ground truth.