LSK3DNet: Towards Effective and Efficient 3D Perception with Large Sparse Kernels
Supplemental Material

Tuo Feng[1], Wenguan Wang[2], Fan Ma[2]*, Yi Yang[2]
[1] ReLER, AAII, University of Technology Sydney [2] ReLER, CCAI, Zhejiang University
https://github.com/FengZicai/LSK3DNet

To enhance the clarity of the main paper, we include the following sections in this supplementary material. We first introduce the analysis and comparison with previous methods in Sec. A. Ablation studies on spatial group size, Effective Receptive Fields (ERFs), inference speed and FLOPs, and Convolutional Schemes are elaborated in Sec. B. More qualitative and quantitative results are further presented and analyzed in Sec. C. Finally, limitations and societal impact are discussed in Sec. D.

A. Comparison and Discussion

Comparison with LinK [1] in s and r Settings. Block Based Aggregation in LinK [1] divides the entire input space into several non-overlapping blocks. Then, it utilizes block-wise proxy aggregation to extract the corresponding features of proxy nodes. Furthermore, the corresponding features of the center are obtained by aggregating the features of the neighboring blocks closest to the center. In LinK, the variable s represents the scale of the block, while r denotes the scale of the block query. Actually, this paradigm is not novel; for example, a similar paradigm has already been employed in 2DPASS [2]. It first divides the entire input space into non-overlapping voxels (blocks) at four scales: 2, 4, 8, and 16. The features of these voxels are obtained by averaging the features within each voxel. Then, 3D sparse convolution is applied to extract features from adjacent voxels. Here, the kernel size of sparse convolution plays the same role as r. In addition, we also use s to represent the above four scales. In Tab. S1, we have fairly compared the receptive field sizes of each module for three methods, based on the default hayaparameters of models on the SemanticKITTI validation dataset [3]. Specifically, three modules are ELKBlock[1] in Link, SPVBlock[2] in 2DPASS, and LSK Block in our LSK3DNet. LinK achieves an equivalent receptive field size of $21 \times 21 \times 21$ for the four ELKBlocks in LinK. However, the second, third, and fourth LSK Blocks in LSK3DNet clearly exhibit larger equivalent receptive fields.

Table S1. **Receptive field size**. RF represents the receptive field size, with subscripts denoting the index of the scale.

Size	s_1/r_1	RF_1
LinK [1]	3/7	$21 \times 21 \times 21$
2DPASS [2]	2/3	$6 \times 6 \times 6$
LSK3DNet	2/9	$18 \times 18 \times 18$
Size	s_2/r_2	RF_2
LinK [1]	3/7	$21 \times 21 \times 21$
2DPASS [2]	4/3	$12 \times 12 \times 12$
LSK3DNet	4/9	$36 \times 36 \times 36$
Size	s_3/r_3	RF_3
LinK [1]	3/7	$21 \times 21 \times 21$
2DPASS [2]	8/3	$24 \times 24 \times 24$
LSK3DNet	8/9	$72 \times 72 \times 72$
Size	s_4/r_4	RF_4
LinK [1]	3/7	$21 \times 21 \times 21$
2DPASS [2]	16/3	$48 \times 48 \times 48$
LSK3DNet	16/9	$144 \times 144 \times 144$

The scales mentioned in section 3.5 and the convolution kernel size have similar spatial meanings to the s and r settings in LinK. Please note that we only adopted kernel size (i.e., r in LinK [1]) as a metric for large kernel design in the main paper, following LargeKernel3D [4]. Please do not confuse these two settings.

Performance of LinK. Upon comparing all the results reported in Link [1] and Model Zoo[1], we notice that the highest performance of 67.7% is achieved with a $(2 \times 3)^3$ receptive field size, surpassing other sizes such as $(3 \times 7)^3$ and $(3 \times 5)^3$. Interestingly, the $(2 \times 3)^3$ receptive field size appears noticeably smaller than $(3 \times 7)^3$. It turns out that Link [1] achieves its stronger performance with a smaller receptive field.

Kernel Size of LargeKernel3D on ScanNet v2 [5]. The claim that "the performance of LargeKernel3D [4] drops when scaling up the kernel size over $7 \times 7 \times 7$" is concluded from Table S - 9[3].

*Corresponding author.
[1] https://github.com/MCG-NJU/LinK
[2] https://github.com/yanx27/2DPASS
[3] https://arxiv.org/pdf/2206.10555v1.pdf

Table S2. **Ablation studies** on spatial group size.

kernel size	*Group*	mIoU (%)
9×9×9	3×3×3	70.2
	5×5×5	69.1
13 × 13 × 3	3×3×3	69.9
	5×5×3	69.5
	7×7×3	68.9

Table S3. **Ablation studies** on *Depth-wise Group Convolution* and *Spatial-wise Group Convolution* on ScanNet v2 [5] and SemanticKITTI [3].

Methods	Kernel	mIoU (%)
ScanNet v2		
MinkowskiNet-14 [4]	7×7×7	68.6
+ *depth-wise* conv	7×7×7	56.4
+ *spatial-wise* conv	7×7×7 → 3×3×3	69.7
+ *spatial-wise* conv	9×9×9 → 7×7×7	69.6
MinkowskiNet-34 [4]	7×7×7	68.6
+ *depth-wise* conv	7×7×7	68.7
+ *spatial-wise* conv	7×7×7 → 3×3×3	73.2
Modified SPVCNN	9×9×9	72.4
+ *depth-wise* conv (SDS and CWS)	9×9×9	67.0
+ *spatial-wise* conv (SDS and CWS)	9×9×9	75.7
SemanticKITTI		
Modified SPVCNN	9×9×9	67.5
+ *depth-wise* conv (SDS and CWS)	9×9×9	65.4
+ *spatial-wise* conv (SDS and CWS)	9×9×9	70.2

B. Ablation Studies

Spatial Group Size. We report the results of the ablation study on the kernel size selection in Tab. S2. *kernel size* means the absolute kernel size. *group* means the number of groups to split kernels. We conduct the experiment on LSK3DNet on the SemanticKITTI dataset, with the same training hyper-parameters in the main paper. For $3\times3\times3$ *group* and $9\times9\times9$ *kernel size*, every group has $\{3,3,3\}$ divisions in each dimension. Moreover, $9\times9\times9$ is split into $5\times5\times5$ with $\{2,2,1,2,2\}$ divisions. For $13\times13\times3$ *kernel size*, $3\times3\times3$ *group* has $\{4,5,4\}$ divisions in 13-dimensional axis, and similarly, $\{2,3,3,3,2\}$ divisions for $5\times5\times3$ *group*, $\{2,2,2,1,2,2,2\}$ divisions for $7\times7\times3$ *group*. We finally chose the $9\times9\times9$ *kernel size* and $3\times3\times3$ *group* based on our empirical results. We speculate that this is because each group has a $3\times3\times3$ size, showing more parameter space to explore than other *group* settings. Previous studies have shown that exploring a large parameter space is important for dynamic sparse training [6–8]. In addition, the kernel size of $13\times13\times3$ does not achieve better performance, being affected by overparameterization and overfitting issues.

Inference Speed and FLOPs. As described in §3.3 and Sec. D, SDS involves unstructured sparsity, whose GPU acceleration is still a challenging issue. This is why SDS could in theory save computation, but in practice that doesn't happen (see previous studies of Dynamic Sparse Training [6, 8–15] for related discussion). Thus, only small speed change (4ms) is observed with SDS in Tab. 6. Our speed up mainly comes from CWS. CWS can significantly reduce the model size at the channel level, leading to a primary acceleration of **84** ms (177ms→93ms). Moreover, for a similar reason, we present the theoretical inference FLOPs of the models under different settings.

CWS. Our CWS is of great importance for building 3D large kernels. Previous 2D large sparse kernels like [16] follows a regime of "*use sparse groups, expand more*"; they attain better performance with enlarged network width, inevitably causing increased model size. This becomes more pronounced in 3D. For a naive dense kernel, lifting (64-d, 9×9×9) to (128-d, 9×9×9) will cause a significant (somewhat unacceptable) increase of parameter number: 47.94M → 191.69M. As shown in Tab. 7 of the main paper, LSK3DNet (with CWS) successfully achieves better performance while using fewer parameters.

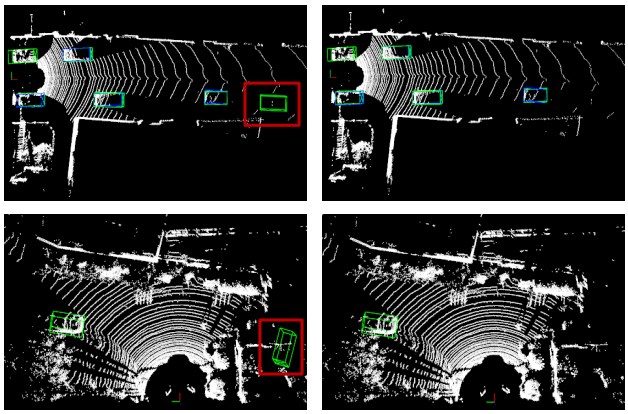
Kernel Size. Due to the increased dimensionality, enlarging 3D kernel size brings much heavier computational burden, compared with 2D kernel. As shown in Tab. 5 of the main paper, comparing Dense ($7\times7\times7$, $1\times D$) and Dense ($9\times9\times9$, $1\times D$), computational burden has **5**× rise: 381.2G→1916.3G. Though large kernel gains larger receptive fields, it brings risk of overparameterization and overfitting, and is hard to train. [4, 16–18] encounter a similar issue, *e.g.*, increasing $3\times3\times3$ to $7\times7\times7$ leads to inferior results [4], 31×31 performs worse than 7×7 [16].

Convolutional Schemes. The results of MinkowskiNet-14 and MinkowskiNet-34 are directly taken from LargeKernel3D [4]. MinkowskiNet-14 shows stagnation when attempting to expand the convolution kernel to 9×9×9. As shown in Tab. S3, we use Modified SPVCNN as the baseline and conduct experiments on ScanNet v2 [5] and SemanticKITTI [3]. Our findings indicate that *Spatial-wise Group Convolution* is more suitable for 3D large kernel designs, aligning with the results observed in LargeKernel3D [4]. Therefore, we carry out dynamic sparse training within the spatial-wise groups.

C. More Qualitative and Quantitative Results

Quantitative Results on NuScenes. In Tab. S4, we present the results of semantic segmentation on the nuScenes validation set [19]. Our approach consistently surpasses others by a significant margin, establishing itself as the state-of-the-art (SOTA) performer on this benchmark. What's particularly intriguing is that our method relies solely on LiDAR data, yet it outperforms multi-modal techniques [2, 20], which incorporate supplementary 2D information.

Concrete Results on SemanticKITTI Multi-scan Test. The full results of our LSK3DNet on the SemanticKITTI [3] multi-scan `test` challenge are shown in Tab. S5 and S6. Our method achieves a state-of-the-art performance on both mIoU and Acc metrics, surpassing other methods in 12 out of the 25 categories. Unlike the previous method 2DPASS, which relies on 2D images as an auxiliary input, our algo-



Voxel R-CNN LSK3DNet

Figure S1. **Qualitative detection results** of LSK3DNet and Voxel R-CNN on the KITTI *val* split (Sec. 4.3). Results in the red box are false positives.

rithm is purely based on point cloud data, which is a valuable advantage.

Qualitative Detection Results for KITTI. We compare the visualization results between LSK3DNet and Voxel R-CNN for the car category on the KITTI *val* split [21], as illustrated in Fig. S1. The blue-bordered boxes depict ground truth bounding boxes, while the green-bordered boxes represent predicted bounding boxes. In comparison to Voxel R-CNN, LSK3DNet demonstrates more accurate prediction results.

Qualitative Results for SemanticKITTI. We provide visual examples of our algorithm on two challenges for 3D semantic segmentation: SemanticKITTI [3] single-scan `val` challenge and SemanticKITTI [3] multi-scan `test` challenge. The corresponding figures are Fig. S2 and Fig. S3, respectively. Moreover, we display the predicted labels of three successive frames in one single frame in Fig. S3. It can be seen that our LSK3DNet produces more accurate predictions than the baseline (Modified SPVCNN [2]). By a larger kernel size, our LSK3DNet expands the receptive field of submanifold convolution and enhances the flow of discrete spatial information. This results in better capturing the object boundaries and distinguishing between different semantic classes. As shown, our LSK3DNet can segment the ground classes and natural objects more effectively.

D. Limitation and Societal Impact

Limitation. Dense matrix multiplications on graphics processing units (GPUs) are the foundation of the current state-of-the-art deep learning methods. However, sparse matrix multiplications, which are crucial for Dynamic Sparse Training operations, cannot be efficiently performed [10, 22, 23]. Hence, the FLOPs for inference that we present in the main paper are based on theoretical calculations. This creates a need for optimized hardware that can handle such operations. However, there is a good news that sparsity support is becoming a more common feature in hardware design for many companies and researchers [12, 24, 25]. Our work aims to inspire more progress in this direction, especially for 3D tasks.

Societal Impact. Self-driving cars need to efficiently and accurately understand 3D scenes to ensure safe driving. Since passenger safety is the top priority for autonomous vehicles, 3D perception models are required to achieve both high accuracy and low latency. Autonomous vehicles have limited hardware resources due to the physical limitations of size and heat dissipation. Therefore, it is important to design 3D neural networks that are both efficient and effective for constrained computing resources. We use the method of Spatial Sparse Group to further expand the 3D convolution kernel size, which overcomes the performance drop of previous 3D large kernel methods. Moreover, the model is very efficient due to dynamic sparse and *Channel-wise Weight Selection*.

License. We study 3D semantic segmentation and object detection on four famous datasets. We use the SemanticKITTI dataset under the permission of its creators and authors by registering at `https://codalab.lisn.upsaclay.fr/competitions/6280`. Scannet v2 is released under the ScanNet Terms of Use (`https://kaldir.vc.in.tum.de/scannet/ScanNet_TOS.pdf`). KITTI is published under the Creative Commons Attribution-NonCommercial-ShareAlike 3.0 License. nuScenes is available for non-commercial use subject to the Terms of Use (See `https://www.nuscenes.org/terms-of-use`).

Computing Infrastructure. The hardware configuration consists of an Intel(R) Xeon(R) Gold 6132 CPU @ 2.60GHz and 18GB of memory. The operating system is Ubuntu 18.04. All the experiments use Tesla V100 GPUs with 32GB of VRAM. Moreover, our method is executed on top of the PyTorch framework.

Table S4. **Semantic segmentation results** on NuScenes `validation` set [19]. Regarding input data format, P denotes points, V represents voxelizations, R signifies range images, 2D3DNet and 2DPASS incorporate additional 2D data. mIoU (%) and IoUs (%) are reported.

Method	Input	**mIoU**	barrier	bicycle	bus	car	construction	motorcycle	pedestrian	traffic cone	trailer	truck	driveable	other flat	sidewalk	terrain	manmade	vegetation
$(AF)^2$-S3Net[26]	V	62.2	60.3	12.6	82.3	80.0	20.1	62.0	59.0	49.0	42.2	67.4	94.2	68.0	64.1	68.6	82.9	82.4
RangeNet++[27]	R	65.5	66.0	21.3	77.2	80.9	30.2	66.8	69.6	52.1	54.2	72.3	94.1	66.6	63.5	70.1	83.1	79.8
PolarNet[28]	R	71.0	74.7	28.2	85.3	90.9	35.1	77.5	71.3	58.8	57.4	76.1	96.5	71.1	74.7	74.0	87.3	85.7
Salsanext[29]	R	72.2	74.8	34.1	85.9	88.4	42.2	72.4	72.2	63.1	61.3	76.5	96.0	70.8	71.2	71.5	86.7	84.4
AMVNet[30]	P	76.1	79.8	32.4	82.2	86.4	62.5	81.9	75.3	72.3	83.5	65.1	97.4	67.0	78.8	74.6	90.8	87.9
Cylinder3D[31]	V	76.1	76.4	40.3	91.2	93.8	51.3	78.0	78.9	64.9	62.1	84.4	96.8	71.6	76.4	75.4	90.5	87.4
RPVNet[32]	RPV	77.6	78.2	43.4	92.7	93.2	49.0	85.7	80.5	66.0	66.9	84.0	96.9	73.5	75.9	76.0	90.6	88.9
PVKD[33]	V	76.0	76.2	40.0	90.2	94.0	50.9	77.4	78.8	64.7	62.0	84.1	96.6	71.4	76.4	76.3	90.3	86.9
2D3DNet[20]	V	79.0	78.3	55.1	95.4	87.7	59.4	79.3	80.7	70.2	68.2	86.6	96.1	74.9	75.7	75.1	91.4	89.9
2DPASS[2]	PV	79.4	78.8	49.6	95.6	93.6	60.0	84.1	82.2	67.5	72.6	88.1	96.8	72.8	76.2	76.5	89.4	87.2
SphereFormer[34]	V	79.5	78.7	46.7	95.2	93.7	54.0	88.9	81.1	68.0	74.2	86.2	97.2	74.3	76.3	75.8	91.4	89.7
LSK3DNet	PV	**80.1**	80.0	52.5	94.5	91.8	58.8	85.8	84.4	71.2	73.8	88.3	96.9	74.8	75.9	75.9	89.3	87.5

Table S5. **Quantitative results** on SemanticKITTI [3] multi-scan challenge `test` (Sec. 4.2) - Part I. mIoU (%) and IoUs (%) are reported.

Method	mIoU	Acc	road	sidewalk	parking	other-ground	building	car	moving car	truck	moving truck	bicycle	motorcycle	other-vehicle	moving other-vehicle
TangentConv[35]	34.1	-	83.9	64.0	38.3	15.3	85.8	84.9	40.3	21.1	1.1	2.0	18.2	18.5	6.4
DarkNet53[3]	41.6	-	91.6	75.3	64.9	27.5	85.2	84.1	61.5	20.0	14.1	30.4	32.9	20.7	15.2
TemporalLidarSeg[36]	47.0	89.6	91.8	75.8	59.6	23.2	89.8	92.1	68.2	39.2	2.1	47.7	40.9	35.0	12.4
SpSeqnet[37]	43.1	-	90.1	73.9	57.6	27.1	91.2	88.5	53.2	29.2	41.2	24.0	26.2	22.7	26.2
KPConv[38]	51.2	89.3	86.5	70.5	58.4	26.7	90.8	93.7	69.4	42.5	5.8	44.9	47.2	38.6	4.7
Cylinder3D[31]	52.5	91.0	90.7	74.5	65.0	32.3	92.6	94.6	74.9	41.3	0.0	67.6	63.8	38.8	0.1
$(AF)^2$-S3Net[26]	56.9	88.1	91.3	72.5	68.8	53.5	87.9	91.8	65.3	15.7	5.6	65.4	86.8	27.5	3.9
PV-KD[33]	58.2	91.9	92.4	77.4	69.9	31.5	92.7	96.2	84.3	50.0	20.9	64.9	64.8	46.4	19.0
2DPASS[2]	62.4	91.4	89.7	74.7	67.4	40.0	93.6	96.2	82.1	48.2	16.1	63.6	63.7	52.7	3.8
LSK3DNet	**63.4**	**92.2**	92.1	79.0	67.4	41.0	92.9	96.4	84.4	58.1	7.2	71.2	73.9	61.8	40.9

Table S6. **Quantitative results** on SemanticKITTI [3] multi-scan challenge `test` (Sec. 4.2) - Part II. mIoU (%) and IoUs (%) are reported.

Method	mIoU	Acc	vegetation	trunk	terrain	person	moving person	bicyclist	moving bicyclist	motorcyclist	moving motorcyclist	fence	pole	traffic-sign
TangentConv[35]	34.1	-	79.5	43.2	56.7	1.6	1.9	0.0	30.1	0.0	42.2	49.1	36.4	31.2
DarkNet53[3]	41.6	-	78.4	50.7	64.8	7.5	0.2	0.0	28.9	0.0	37.8	56.5	38.1	53.3
TemporalLidarSeg[36]	47.0	89.6	82.3	62.5	64.7	14.4	40.4	0.0	42.8	0.0	12.9	63.8	52.6	60.4
SpSeqnet[37]	43.1	-	84.0	66.0	65.7	6.3	36.2	0.0	2.3	0.0	0.1	66.8	50.8	48.7
KPConv[38]	51.2	89.3	84.6	70.3	66.0	21.6	67.5	0.0	67.4	0.0	47.2	64.5	57.0	53.9
Cylinder3D[31]	52.5	91.0	85.8	72.0	68.9	12.5	65.7	1.7	68.3	0.2	11.9	66.0	63.1	61.4
$(AF)^2$-S3Net[26]	56.9	88.1	75.1	64.6	57.4	16.4	67.6	15.1	66.4	67.1	59.6	63.2	62.6	71.0
PV-KD[33]	58.2	91.9	86.4	74.1	70.2	16.6	68.5	0.0	69.2	2.0	50.5	70.3	66.9	70.6
2DPASS[2]	62.4	91.4	86.2	73.9	71.0	35.4	80.3	7.9	71.2	62.0	73.1	72.9	65.0	70.5
LSK3DNet	**63.4**	**92.2**	86.9	74.2	72.6	29.3	77.4	0.1	69.9	22.8	72.1	72.3	66.7	74.3

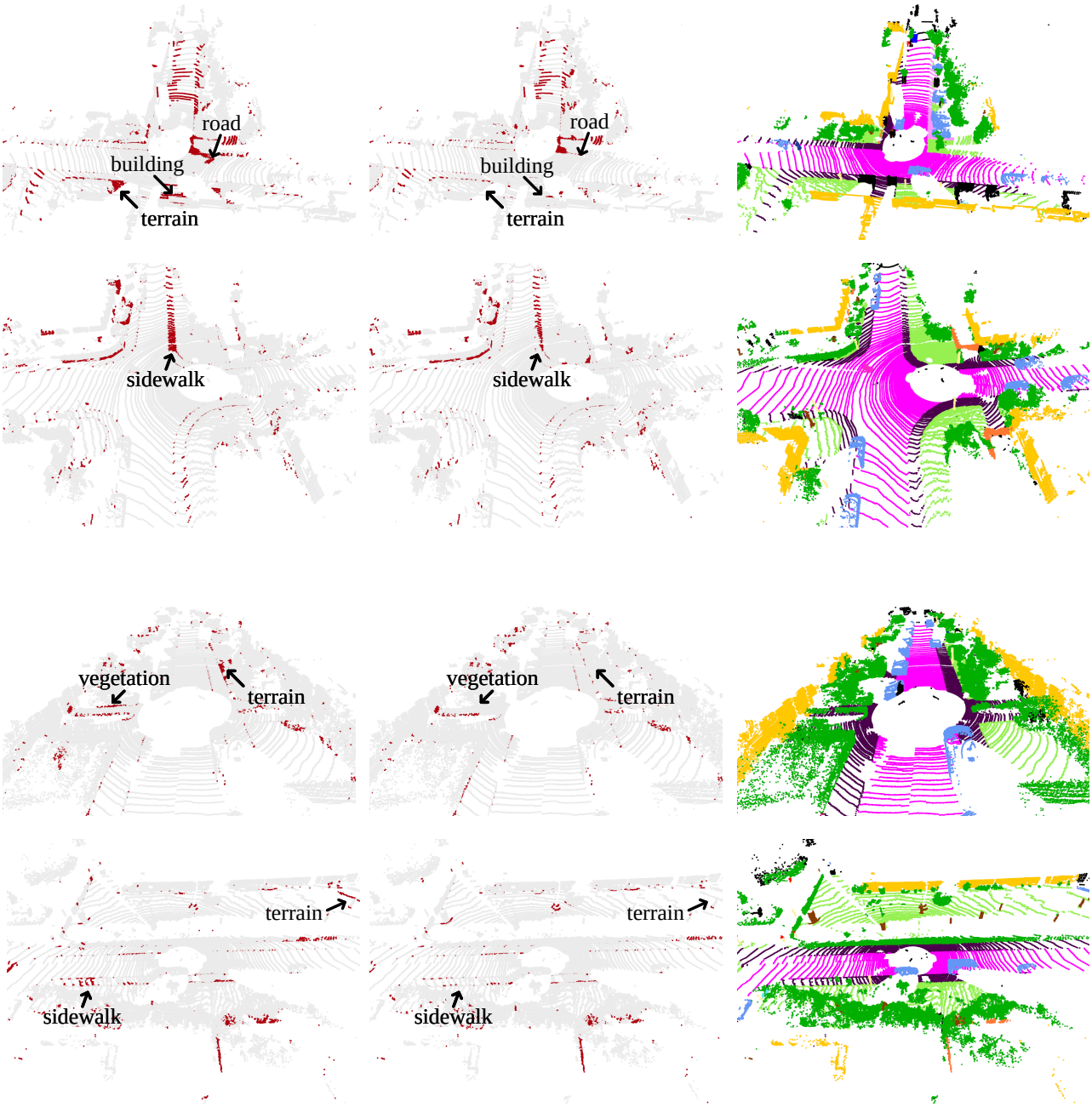
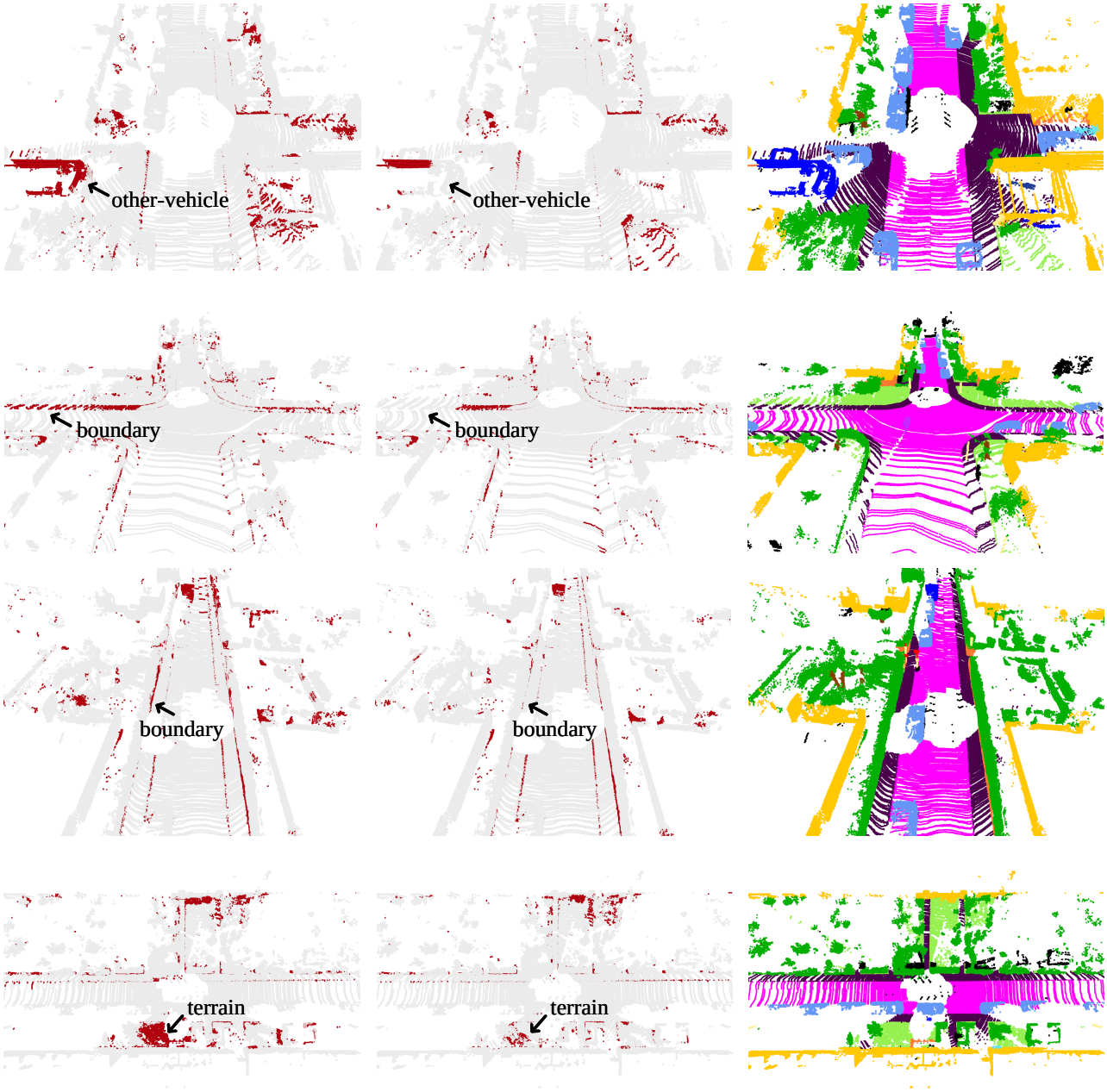


Figure S2. Error maps of Baseline and Ours on SemanticKITTI [3] single-scan `val` (Sec.4.2).



Error by Baseline | Error by LSK3DNet | Ground Truth

Figure S3. Error maps of Baseline and Ours on SemanticKITTI [3] multi-scan `val` (Sec.4.2).

References

[1] Tao Lu, Xiang Ding, Haisong Liu, Gangshan Wu, and Limin Wang. Link: Linear kernel for lidar-based 3d perception. In *CVPR*, 2023. S1

[2] Xu Yan, Jiantao Gao, Chaoda Zheng, Chao Zheng, Ruimao Zhang, Shuguang Cui, and Zhen Li. 2dpass: 2d priors assisted semantic segmentation on lidar point clouds. In *ECCV*, 2022. S1, S2, S3, S4

[3] Jens Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Cyrill Stachniss, and Jurgen Gall. Semantickitti: A dataset for semantic scene understanding of lidar sequences. In *ICCV*, 2019. S1, S2, S3, S4, S5, S6

[4] Yukang Chen, Jianhui Liu, Xiaojuan Qi, Xiangyu Zhang, Jian Sun, and Jiaya Jia. Scaling up kernels in 3d cnns. *CVPR*, 2023. S1, S2

[5] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *CVPR*, 2017. S1, S2

[6] Siddhant Jayakumar, Razvan Pascanu, Jack Rae, Simon Osindero, and Erich Elsen. Top-kast: Top-k always sparse training. In *NeurIPS*, 2020. S2

[7] Md Aamir Raihan and Tor Aamodt. Sparse weight activation training. In *NeurIPS*, 2020.

[8] Shiwei Liu, Decebal Constantin Mocanu, Yulong Pei, and Mykola Pechenizkiy. Selfish sparse rnn training. In *ICML*, 2021. S2

[9] Guillaume Bellec, David Kappel, Wolfgang Maass, and Robert Legenstein. Deep rewiring: Training very sparse deep networks. *arXiv preprint arXiv:1711.05136*, 2017.

[10] Decebal Constantin Mocanu, Elena Mocanu, Peter Stone, Phuong H Nguyen, Madeleine Gibescu, and Antonio Liotta. Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. *Nature communications*, 9(1):2383, 2018. S3

[11] Tim Dettmers and Luke Zettlemoyer. Sparse networks from scratch: Faster training without losing performance. *arXiv preprint arXiv:1907.04840*, 2019.

[12] Shiwei Liu, Decebal Constantin Mocanu, Amarsagar Reddy Ramapuram Matavalam, Yulong Pei, and Mykola Pechenizkiy. Sparse evolutionary deep learning with over one million artificial neurons on commodity hardware. *Neural Computing and Applications*, 33:2589–2604, 2021. S3

[13] Utku Evci, Trevor Gale, Jacob Menick, Pablo Samuel Castro, and Erich Elsen. Rigging the lottery: Making all tickets winners. In *ICML*, 2020.

[14] Hesham Mostafa and Xin Wang. Parameter efficient training of deep convolutional neural networks by dynamic sparse reparameterization. In *ICML*, 2019.

[15] Tianlong Chen, Yu Cheng, Zhe Gan, Lu Yuan, Lei Zhang, and Zhangyang Wang. Chasing sparsity in vision transformers: An end-to-end exploration. In *NeurIPS*, 2021. S2

[16] Shiwei Liu, Tianlong Chen, Xiaohan Chen, Xuxi Chen, Qiao Xiao, Boqian Wu, Mykola Pechenizkiy, Decebal Mocanu, and Zhangyang Wang. More convnets in the 2020s: Scaling up kernels beyond 51x51 using sparsity. *arXiv preprint arXiv:2207.03620*, 2022. S2

[17] Tianyu Ma, Alan Q Wang, Adrian V Dalca, and Mert R Sabuncu. Hyper-convolutions via implicit kernels for medical imaging. *arXiv preprint arXiv:2202.02701*, 2022.

[18] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *CVPR*, 2019. S2

[19] 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 *CVPR*, 2020. S2, S4

[20] Kyle Genova, Xiaoqi Yin, Abhijit Kundu, Caroline Pantofaru, Forrester Cole, Avneesh Sud, Brian Brewington, Brian Shucker, and Thomas Funkhouser. Learning 3d semantic segmentation with only 2d image supervision. In *3DV*, 2021. S2, S4

[21] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013. S3

[22] Vadim Lebedev and Victor Lempitsky. Fast convnets using group-wise brain damage. In *CVPR*, 2016. S3

[23] Soravit Changpinyo, Mark Sandler, and Andrey Zhmoginov. The power of sparsity in convolutional neural networks.(2017). *arXiv preprint cs.CV/1702.06257*, 2017. S3

[24] Trevor Gale, Matei Zaharia, Cliff Young, and Erich Elsen. Sparse gpu kernels for deep learning. In *SC20*, 2020. S3

[25] Jack Choquette, Wishwesh Gandhi, Olivier Giroux, Nick Stam, and Ronny Krashinsky. Nvidia a100 tensor core gpu: Performance and innovation. *IEEE Micro*, 41(2):29–35, 2021. S3

[26] 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 *CVPR*, 2021. S4

[27] Andres Milioto, Ignacio Vizzo, Jens Behley, and Cyrill Stachniss. Rangenet++: Fast and accurate lidar semantic segmentation. In *IROS*, 2019. S4

[28] Yang Zhang, Zixiang Zhou, Philip David, Xiangyu Yue, Zerong Xi, Boqing Gong, and Hassan Foroosh. Polarnet: An improved grid representation for online lidar point clouds semantic segmentation. In *CVPR*, 2020. S4

[29] Tiago Cortinhal, George Tzelepis, and Eren Erdal Aksoy. Salsanext: fast, uncertainty-aware semantic segmentation of lidar point clouds for autonomous driving. *arXiv preprint arXiv:2003.03653*, 2020. S4

[30] Venice Erin Liong, Thi Ngoc Tho Nguyen, Sergi Widjaja, Dhananjai Sharma, and Zhuang Jie Chong. Amvnet: Assertion-based multi-view fusion network for lidar semantic segmentation. *arXiv preprint arXiv:2012.04934*, 2020. S4

[31] 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 *CVPR*, 2021. S4

[32] Jianyun Xu, Ruixiang Zhang, Jian Dou, Yushi Zhu, Jie Sun, and Shiliang Pu. Rpvnet: A deep and efficient range-point-voxel fusion network for lidar point cloud segmentation. In *ICCV*, 2021. S4

[33] Yuenan Hou, Xinge Zhu, Yuexin Ma, Chen Change Loy, and Yikang Li. Point-to-voxel knowledge distillation for lidar semantic segmentation. In *CVPR*, 2022. S4

[34] Xin Lai, Yukang Chen, Fanbin Lu, Jianhui Liu, and Jiaya Jia. Spherical transformer for lidar-based 3d recognition. In *CVPR*, 2023. S4

[35] Maxim Tatarchenko, Jaesik Park, Vladlen Koltun, and Qian-Yi Zhou. Tangent convolutions for dense prediction in 3d. In *CVPR*, 2018. S4

[36] Fabian Duerr, Mario Pfaller, Hendrik Weigel, and Jürgen Beyerer. Lidar-based recurrent 3d semantic segmentation with temporal memory alignment. In *3DV*, 2020. S4

[37] Hanyu Shi, Guosheng Lin, Hao Wang, Tzu-Yi Hung, and Zhenhua Wang. Spsequencenet: Semantic segmentation network on 4d point clouds. In *CVPR*, 2020. S4

[38] Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *ICCV*, 2019. S4