DeepGCNs: Can GCNs Go as Deep as CNNs?
– Supplementary Material –

Guohao Li* Matthias Müller* Ali Thabet Bernard Ghanem
Visual Computing Center, KAUST, Thuwal, Saudi Arabia
{guohao.li, matthias.mueller.2, ali.thabet, bernard.ghanem}@kaust.edu.sa

1. Deep GCN Variants

In our experiments in the paper, we work with a GCN based on EdgeConv [2] to show how very deep GCNs can be trained. However, it is straightforward to build other deep GCNs with the same concepts we proposed (*e.g. residual/dense graph connections*, *dilated graph convolutions*). To show that these concepts are universal operators and can be used for general GCNs, we perform additional experiments. In particular, we build ResGCNs based on GraphSAGE [1], Graph Isomorphism Network (GIN) [3] and *MRGCN* (Max-Relative GCN) which is a new GCN operation we proposed. In practice, we find that EdgeConv learns a better representation than the other implementations. However, it is less memory and computation efficient. Therefore, we propose a simple GCN combining the advantages of them all.

All of the ResGCNs have the same components (*e.g.* dynamic $k-NN$, residual connections, stochastic dilation) and parameters (*e.g.* #NNs, #filters and #layers) as *ResGCN-28* in Table **Ablation Study** of the paper except for the internal GCN operations. To simplify, we refer to these models as *ResEdgeConv*, *ResGraphSAGE*, *ResGIN* and *NewResGCN* respectively. Note that *ResEdgeConv* is an alias for *ResGCN* in our paper. We refer to it as *ResEdgeConv* to distinguish it from the other GCN operations.

ResEdgeConv. Instead of aggregating neighborhood features directly, EdgeConv [2] proposes to first get local neighborhood information for each neighbor by subtracting the feature of the central vertex from its own feature. In order to train deeper GCNs, we add *residual/dense graph connections* and *dilated graph convolutions* to EdgeConv:

$$\begin{aligned}\mathbf{h}^{res}_{v_{l+1}} &= max\left(\{mlp(concat(\mathbf{h}_{v_l}, \mathbf{h}_{u_l}-\mathbf{h}_{v_l}))|u_l \in \mathcal{N}^{(d)}(v_l)\}\right),\\ \mathbf{h}_{v_{l+1}} &= \mathbf{h}^{res}_{v_{l+1}} + \mathbf{h}_{v_l}.\end{aligned} \tag{1}$$

ResGraphSAGE. GraphSAGE [1] proposes different types of aggregator functions including a *Mean aggregator*, *LSTM aggregator* and *Pooling aggregator*. Their experiments show that the *Pooling aggregator* outperforms the others. We adapt GraphSAGE with the max-pooling aggregator to obtain *ResGraphSAGE*:

$$\begin{aligned}\mathbf{h}^{res}_{\mathcal{N}^{(d)}(v_l)} &= max\left(\{mlp(\mathbf{h}_{u_l})|u_l \in \mathcal{N}^{(d)}(v_l)\}\right),\\ \mathbf{h}^{res}_{v_{l+1}} &= mlp\left(concat\left(\mathbf{h}_{v_l}, \mathbf{h}^{res}_{\mathcal{N}^{(d)}(v_l)}\right)\right),\\ \mathbf{h}_{v_{l+1}} &= \mathbf{h}^{res}_{v_{l+1}} + \mathbf{h}_{v_l},\end{aligned} \tag{2}$$

In the original GraphSAGE paper, the vertex features are normalized after aggregation. We implement two variants, one without normalization (see Equation (2)), the other one with normalization $\mathbf{h}^{res}_{v_{l+1}} = \mathbf{h}^{res}_{v_{l+1}} / \left\|\mathbf{h}^{res}_{v_{l+1}}\right\|_2$.

ResGIN. The main difference between GIN [3] and other GCNs is that an ϵ is learned at each GCN layer to give the central vertex and aggregated neighborhood features different weights. Hence *ResGIN* is formulated as follows:

$$\begin{aligned}\mathbf{h}^{res}_{v_{l+1}} &= mlp\left((1+\epsilon)\cdot\mathbf{h}_{v_l} + sum(\{\mathbf{h}_{u_l}|u_l \in \mathcal{N}^{(d)}(v_l)\})\right),\\ \mathbf{h}_{v_{l+1}} &= \mathbf{h}^{res}_{v_{l+1}} + \mathbf{h}_{v_l}.\end{aligned} \tag{3}$$

ResMRGCN. We find that first using a max aggregator to aggregate neighborhood relative features $(\mathbf{h}_{u_l}-\mathbf{h}_{v_l})$, $u_l \in \mathcal{N}(v_l)$ is more efficient than aggregating raw neighborhood features $\mathbf{h}_{v_l}$, $u_l \in \mathcal{N}(v_l)$ or aggregating features after nonlinear transforms. We refer to this simple GCN as *MRGCN* (Max-Relative GCN). The residual version of *MRGCN* is as such:

$$\begin{aligned}\mathbf{h}^{res}_{\mathcal{N}^{(d)}(v_l)} &= max\left(\{\mathbf{h}_{u_l}-\mathbf{h}_{v_l}|u_l \in \mathcal{N}^{(d)}(v_l)\}\right),\\ \mathbf{h}^{res}_{v_{l+1}} &= mlp\left(concat\left(\mathbf{h}_{v_l}, \mathbf{h}^{res}_{\mathcal{N}^{(d)}(v_l)}\right)\right),\\ \mathbf{h}_{v_{l+1}} &= \mathbf{h}^{res}_{v_{l+1}} + \mathbf{h}_{v_l}.\end{aligned} \tag{4}$$

Where $\mathbf{h}_{v_{l+1}}$ and $\mathbf{h}_{v_l}$ are the hidden state of vertex v at $l+1$; $\mathbf{h}^{res}_{v_{l+1}}$ is the hidden state of the residual graph. All the *mlp*

*equal contribution

Model	mIoU	ΔmIoU	dynamic	connection	dilation	stochastic	# NNs	# filters	# layers
ResEdgeConv-28	**52.49**	0.00	✓	⊕	✓	✓	16	64	28
PlainGCN-28	**40.31**	-12.18	✓				16	64	28
ResGraphSAGE-28	**49.20**	-3.29	✓	⊕	✓	✓	16	64	28
ResGraphSAGE-N-28	**49.02**	-3.47	✓	⊕	✓	✓	16	64	28
ResGIN-ϵ-28	**42.81**	-9.68	✓	⊕	✓	✓	16	64	28
ResMRGCN-28	**51.17**	-1.32	✓	⊕	✓	✓	16	64	28

Table 1. **Comparisons of Deep GCNs variants on area 5 of S3DIS**. We compare our different types of ResGCN (*ResEdgeConv*, *ResGraphSAGE*, *ResGIN* and *ResMRGCN*) with 28 layers. *Residual graph connections* and *Dilated graph convolutions* are added to all the GCN variants. All models were trained with the same hyper-parameters for 100 epochs on all areas except for area 5 which is used for evaluation. We denote residual with the ⊕ symbols.

(multilayer perceptron) functions use a ReLU as activation function; all the *max* and *sum* functions above are vertex-wise feature operators; *concat* functions concatenate features of two vertices into one feature vector. $\mathcal{N}^{(d)}(v_l)$ denotes the neighborhood of vertex v_l obtained from *Dilated k-NN*.

2. Results for Deep GCN Variants

Table 1 shows a comparison of different deep residual GCNs variants on the task of semantic segmentation; we report the mIOU for area 5 of S3DIS. All deep GCN variants are 28 layers deep and we denote them as *ResEdgeConv-28*, *ResGraphSAGE-28*, *ResGraphSAGE-N-28*, *ResGIN-ϵ-28* and *ResMRGCN-28*; *ResGraphSAGE-28* is GraphSAGE without normalization, *ResGraphSAGE-N-28* is the version with normalization. The results clearly show that different deep GCN variants with *residual graph connections* and *dilated graph convolutions* converge better than the *PlainGCN*. *ResMRGCN-28* achieves almost the same performance as *ResEdgeConv-28* while only using half of the GPU memory. *ResGraphSAGE-28* and *ResGraphSAGE-N-28* are slightly worse than *ResEdgeConv-28* and *ResMRGCN-28*. The results also show that using normalization for *ResGraphSAGE* is not essential. Interestingly, we find that *ResGIN-ϵ-28* converges well during the training phase and has a high training accuracy. However, it fails to generalize to the test set. This phenomenon is also observed in the original paper [3] in which they find setting ϵ to 0 can get the best performance. Therefore, we can draw the conclusion that the concepts we proposed (*e.g. residual/dense graph connections* and *dilated graph convolutions*) generalize well to different types of GCNs and enable training very deep GCNs.

3. Qualitative Results for the Ablation Study

We summarize the most important insights of the ablation study in Figure 1. Figures 2, 3, 4, 5, 6 show qualitative results for the ablation study presented in the paper.

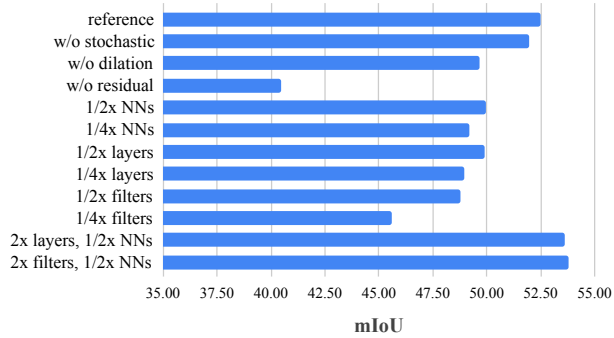

Figure 1. **Ablation study on area 5 of S3DIS**. We compare our reference network (*ResGCN-28*) with 28 layers, *residual graph connections* and *dilated graph convolutions* to several ablated variants. All models were trained for 100 epochs on all areas except for area 5 with the same hyper-parameters.

4. Run-time Overhead of Dynamic k-NN

We conduct a run-time experiment comparing the inference time of the reference model (28 layers, k=16) with dynamic k-NN and fixed k-NN. The inference time with fixed k-NN is 45.63ms. Computing the dynamic k-NN increases the inference time by 150.88ms. It is possible to reduce computation by updating the k-NN less frequently (*e.g.* computing the dynamic k-NN every 3 layers).

5. Comparison with DGCNN over All Classes

To showcase the consistent improvement of our framework over the baseline DGCNN [2], we reproduce the results of DGCNN* in Table 2 and find our method outperforms DGCNN in all classes.

*The results over all classes were not provided in the original DGCNN paper

Class	DGCNN [2]	ResGCN-28 (*Ours*)
ceiling	92.7	**93.1**
floor	93.6	**95.3**
wall	77.5	**78.2**
beam	32.0	**33.9**
column	36.3	**37.4**
window	52.5	**56.1**
door	63.7	**68.2**
table	61.1	**64.9**
chair	60.2	**61.0**
sofa	20.5	**34.6**
bookcase	47.7	**51.5**
board	42.7	**51.1**
clutter	51.5	**54.4**
mIOU	56.3	**60.0**

Table 2. **Comparison of *ResGCN-28* with DGCNN**. Average per-class results across all areas for our reference network with 28 layers, *residual graph connections* and *dilated graph convolutions* compared to DGCNN baseline. *ResGCN-28* outperforms DGCNN across all the classes. Metric shown is IoU.

References

[1] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pages 1024–1034, 2017.

[2] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *arXiv preprint arXiv:1801.07829*, 2018.

[3] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

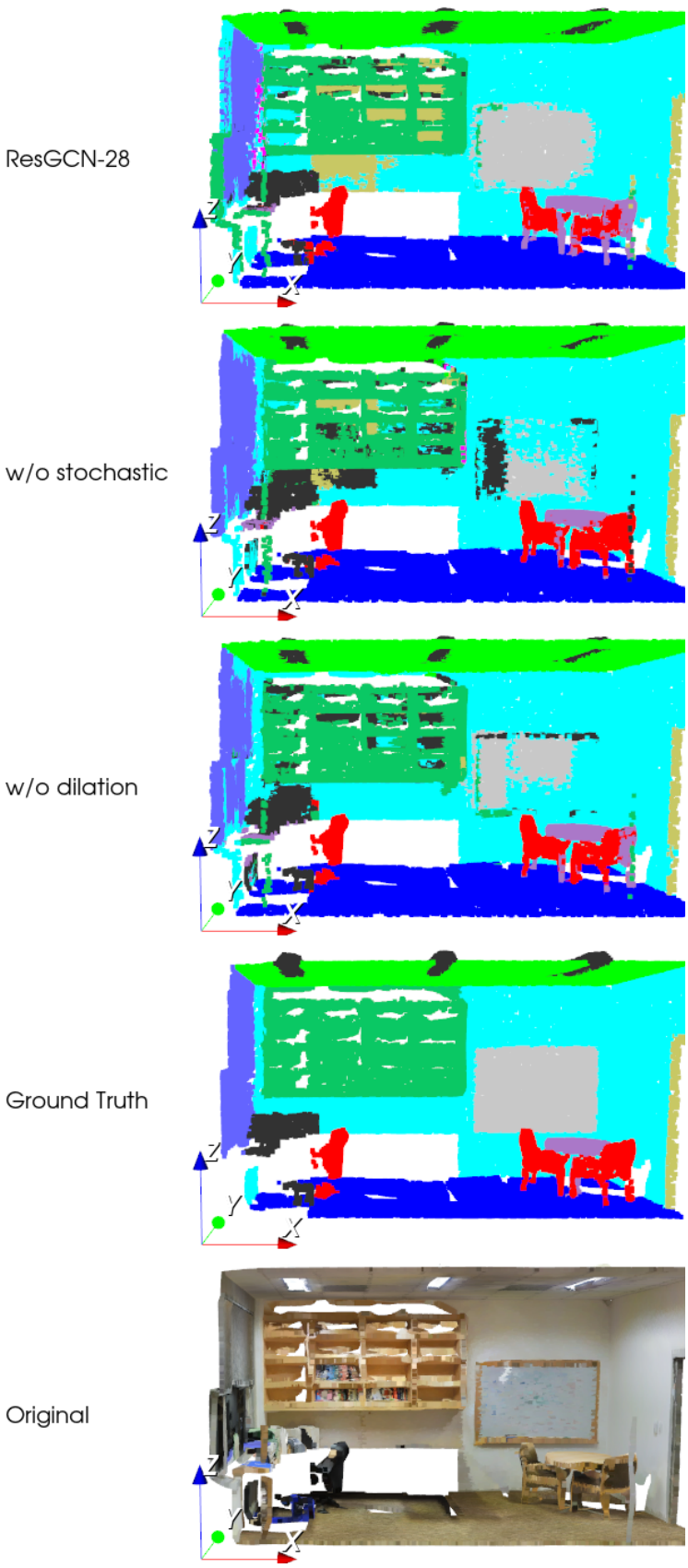

Figure 2. **Qualitative Results for S3DIS Semantic Segmentation**. We show the importance of stochastic dilated convolutions.

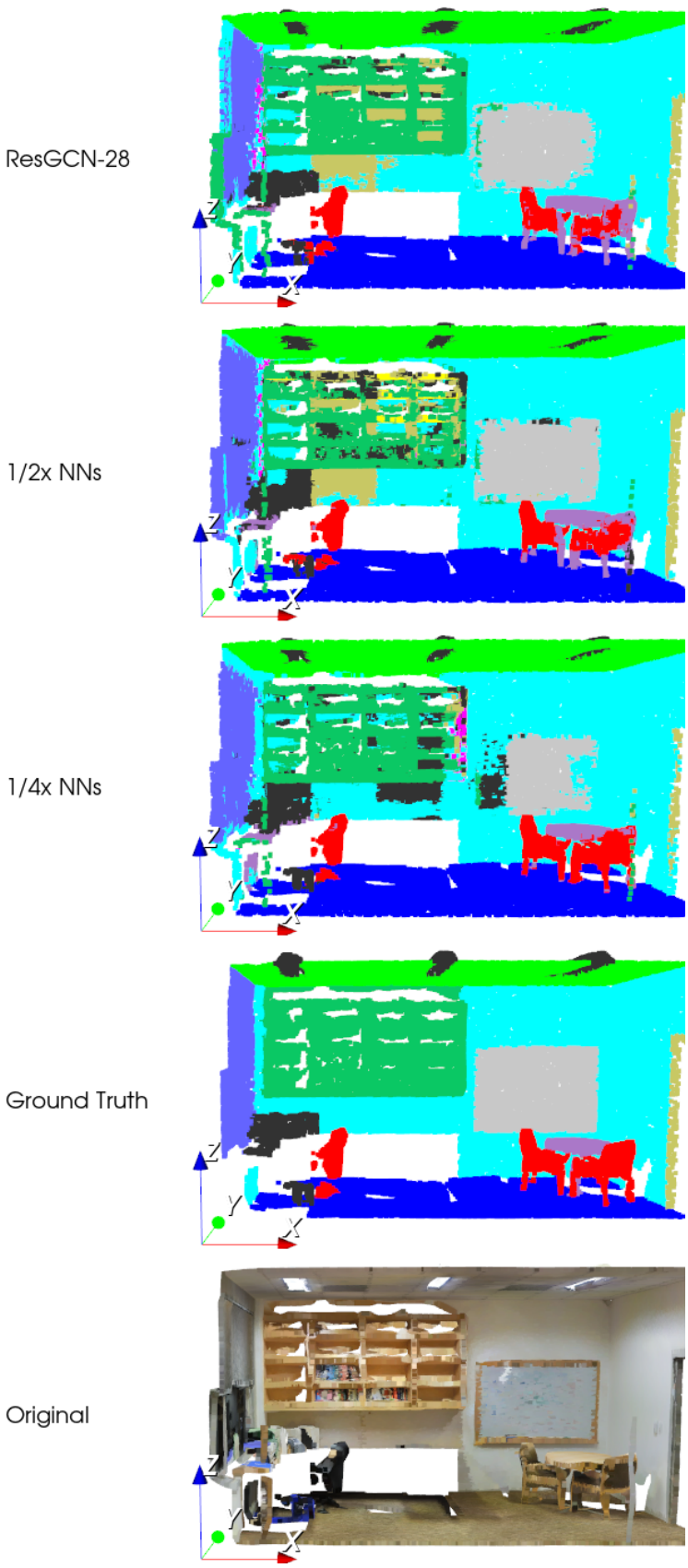

Figure 3. **Qualitative Results for S3DIS Semantic Segmentation**. We show the importance of the number of nearest neighbors used in the convolutions.

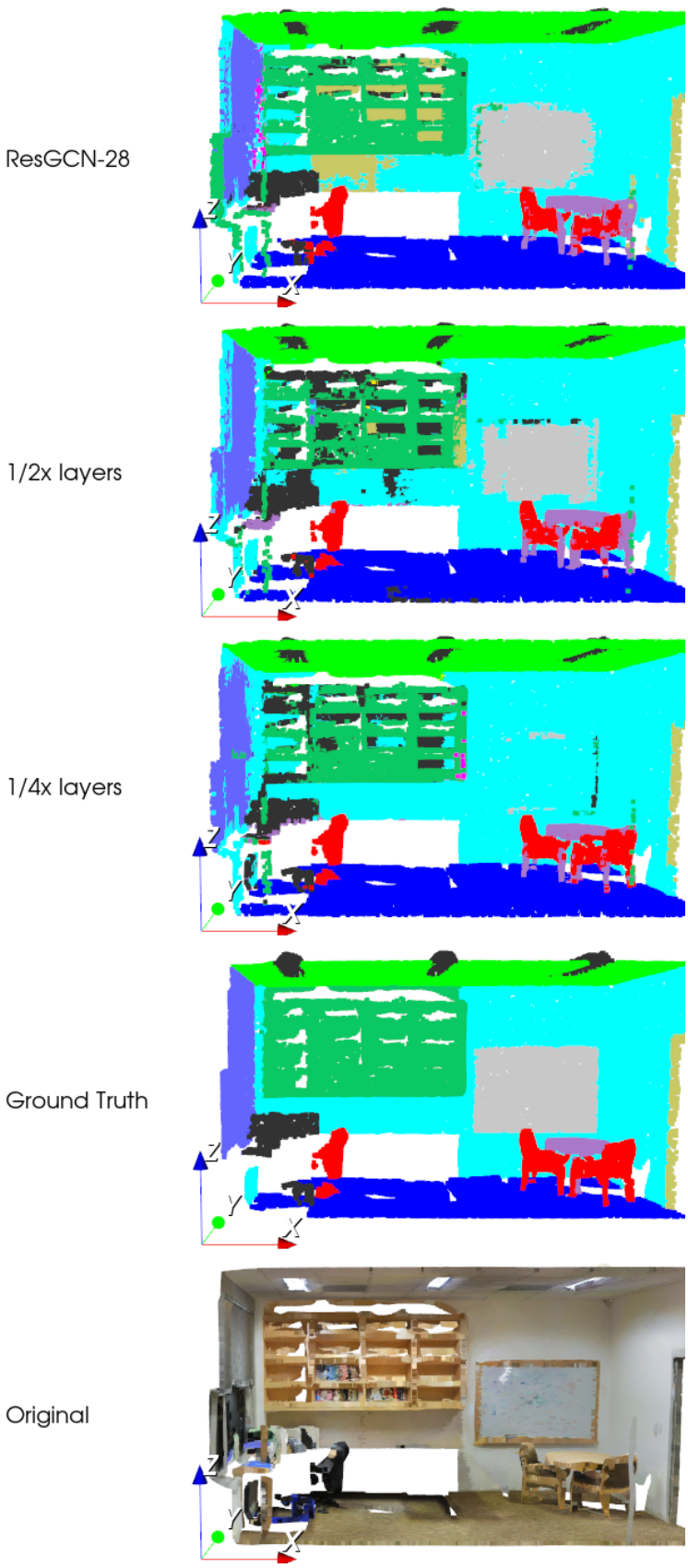

Figure 4. **Qualitative Results for S3DIS Semantic Segmentation**. We show the importance of network depth (number of layers).

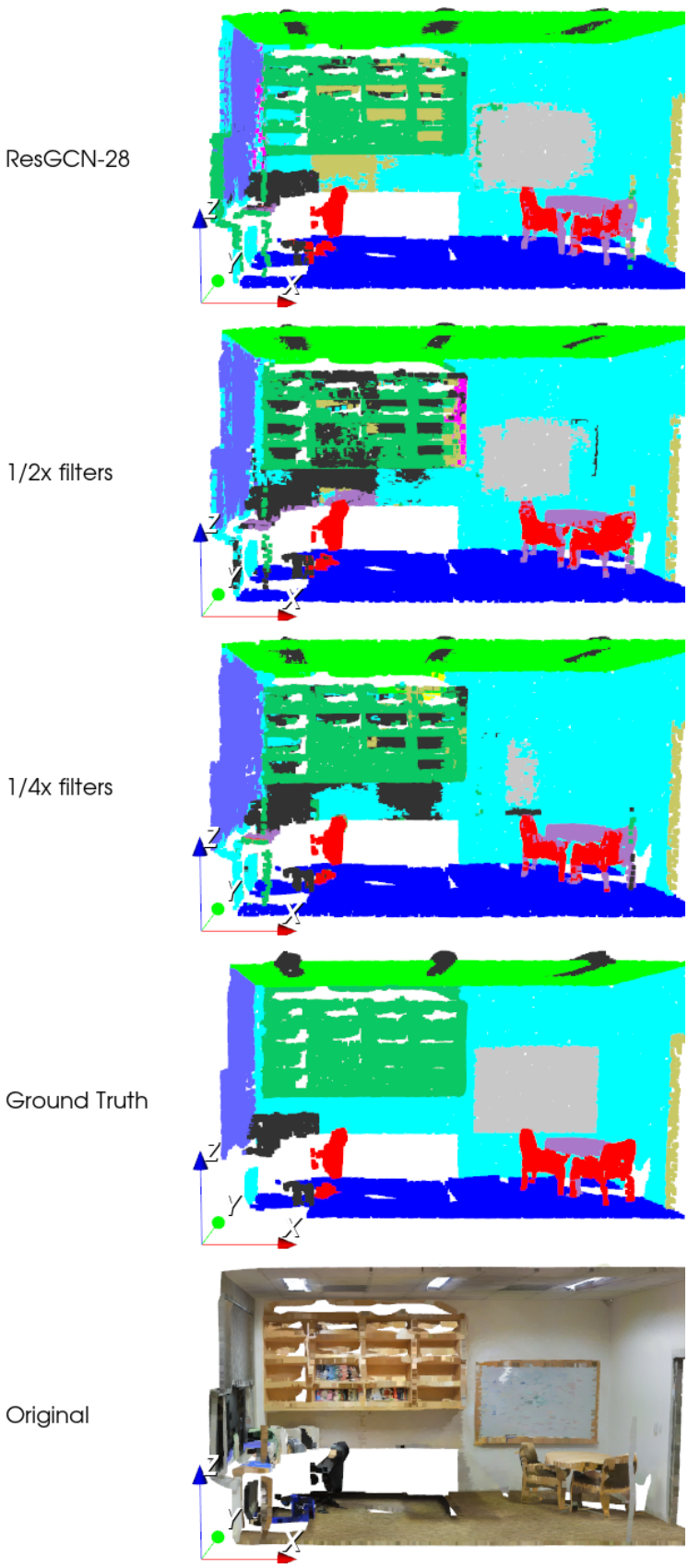

Figure 5. **Qualitative Results for S3DIS Semantic Segmentation**. We show the importance of network width (number of filters per layer).

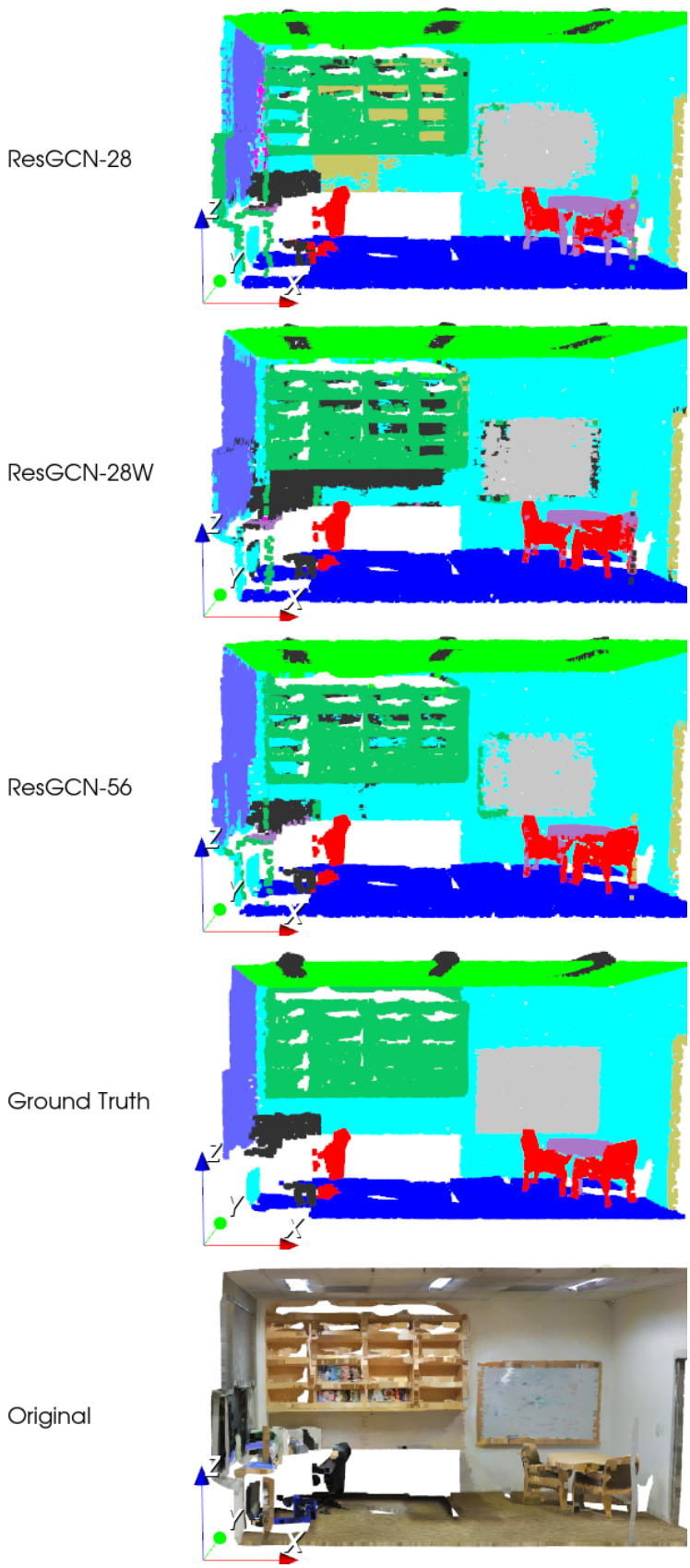

Figure 6. **Qualitative Results for S3DIS Semantic Segmentation**. We show the benefit of a wider and deeper network even with only half the number of nearest neighbors.