Supplementary Material

A. Backward propagation analysis

In the main paper, we compare the forward computation of the hard spherical kernel in Eq. (5) with the forward pass of the proposed fuzzy spherical kernel in Eq. (13). In this section, we correspondingly analyze the backward propagations of the two kernels in Eq. (A.1) and (A.2). We also provide the forward computations here as references. It can be noticed that compared to the computation of gradient for only one $w_{\kappa c}$ in the hard kernel, the fuzzy kernel requires 3 more multiplications to fulfill the gradient computation for three learnable parameters $w_{0c}, w_{\kappa' c}, w_{\tilde{\kappa}' c}$ during its backward propagation. To compute gradient for the input feature f_c^l, the fuzzy kernel takes 3 more multiplications and 2 more additions, identical to the forward propagation. Eventually, the fuzzy kernel performs 6 more multiplications and 2 more additions compared to the hard kernel to achieve the gradient computation. Note that, to save the GPU memory, we compute $(\xi_0 w_{0c} + \xi_{\kappa'} w_{\kappa' c} + \xi_{\tilde{\kappa}'} w_{\tilde{\kappa}' c})$ again in the backward propagation instead of using the result initially computed in the forward pass, whose storage consumes considerable memory. The additional computations of the fuzzy kernel make it slightly slower in training and inference. However, its performance superiority and robustness to point density variations makes it an attractive choice.

$$\text{Hard kernel:} \begin{cases} f_{ic}^{l+1} = \frac{1}{\mathcal{N}(\mathbf{x}_i)} \sum_{\mathbf{x} \in \mathcal{N}(\mathbf{x}_i)} w_{\kappa c} f_c^l. \\ \frac{\partial f_{ic}^{l+1}}{\partial f_c^l} = \frac{1}{\mathcal{N}(\mathbf{x}_i)} w_{\kappa c}. \\ \frac{\partial f_{ic}^{l+1}}{\partial w_{\kappa c}} = \frac{1}{\mathcal{N}(\mathbf{x}_i)} f_c^l. \end{cases} \tag{A.1}$$

$$\text{Fuzzy kernel:} \begin{cases} f_{ic}^{l+1} = \frac{1}{\mathcal{N}(\mathbf{x}_i)} \sum_{\mathbf{x} \in \mathcal{N}(\mathbf{x}_i)} (\xi_0 w_{0c} + \xi_{\kappa'} w_{\kappa' c} + \xi_{\tilde{\kappa}'} w_{\tilde{\kappa}' c}) f_c^l. \\ \frac{\partial f_{ic}^{l+1}}{\partial f_c^l} = \frac{1}{\mathcal{N}(\mathbf{x}_i)} (\xi_0 w_{0c} + \xi_{\kappa'} w_{\kappa' c} + \xi_{\tilde{\kappa}'} w_{\tilde{\kappa}' c}). \\ \frac{\partial f_{ic}^{l+1}}{\partial w_{0c}} = \frac{1}{\mathcal{N}(\mathbf{x}_i)} \xi_0 f_c^l, \quad \frac{\partial f_{ic}^{l+1}}{\partial w_{\kappa' c}} = \frac{1}{\mathcal{N}(\mathbf{x}_i)} \xi_{\kappa'} f_c^l, \quad \frac{\partial f_{ic}^{l+1}}{\partial w_{\tilde{\kappa}' c}} = \frac{1}{\mathcal{N}(\mathbf{x}_i)} \xi_{\tilde{\kappa}'} f_c^l. \end{cases} \tag{A.2}$$

B. Different decay rate of 'γ'

We experimented with three different settings of 'γ' to control the influence of distance $\|\mathbf{x} - \mathbf{x}_i\|$ to the self-convolution bin. Details of the experimented functions are provided in Eq. (B.1). Among them, 'γ_a' decreases linearly *w.r.t.* distance

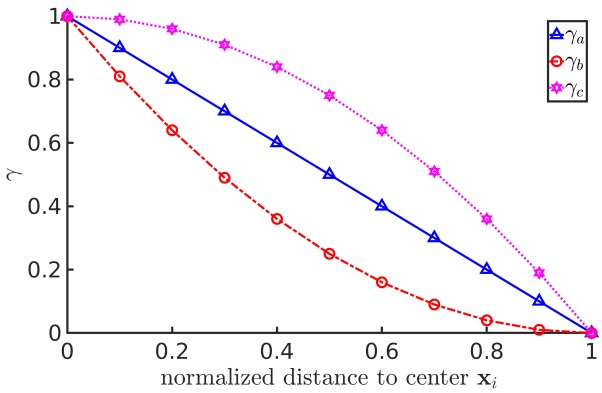

Figure A.1. Different decay behaviors of 'γ' *w.r.t.* normalized distance of each neighbor $\mathbf{x}$ to the center point $\mathbf{x}_i$, i.e. $\frac{\|\mathbf{x}-\mathbf{x}_i\|}{\rho}$. '$\gamma_a$' shows a linear decay. 'γ_b' accelerates the decay with increasing distance. Thus, the farther points contribute less to the self-convolution bin. 'γ_c' has a converse behavior as compared to 'γ_b'.

$\|\mathbf{x}-\mathbf{x}_i\|$, which is the setting we use in our fuzzy kernel design. For the other two settings, 'γ_b' increases the decay rate and makes farther points contribute less to the self-convolution bin w_{0c}, while 'γ_c' slows down the decay which makes farther points contribute more to w_{0c}. In our evaluation, applying γ_a, γ_b, γ_c to the kernel design resulted in mIoU of 63.6%, 62.7%, 62.2% respectively on the Area 5 of the S3DIS dataset. We employ 'γ_a' in our design for its intuitive nature and effectiveness.

$$\begin{cases} \gamma_a = 1 - \frac{\|\mathbf{x}-\mathbf{x}_i\|}{\rho}. \\ \gamma_b = \left(1 - \frac{\|\mathbf{x}-\mathbf{x}_i\|}{\rho}\right)^2. \\ \gamma_c = 1 - \left(\frac{\|\mathbf{x}-\mathbf{x}_i\|}{\rho}\right)^2. \end{cases} \tag{B.1}$$

C. Fuzzy assignment in the elevation direction

To further explain the fuzzy assignment used in the proposed kernel, we provide a 2D slice example of division in the elevation direction in Fig. C.1. In the shown example, the elevation is divided into 2 bins (i.e. $q = 2$) that generally partition the points above or below the xy plane. We show the results of k_ϕ and $\tilde{k}_\phi$ when the neighbor point 's' is in different positions: (a) $\phi_s \in (-\frac{\pi}{2}, -\frac{\pi}{4})$; (b) $\phi_s \in (-\frac{\pi}{4}, 0)$; (c) $\phi_s \in (0, \frac{\pi}{4})$; (d) $\phi_s \in (\frac{\pi}{4}, \frac{\pi}{2})$.

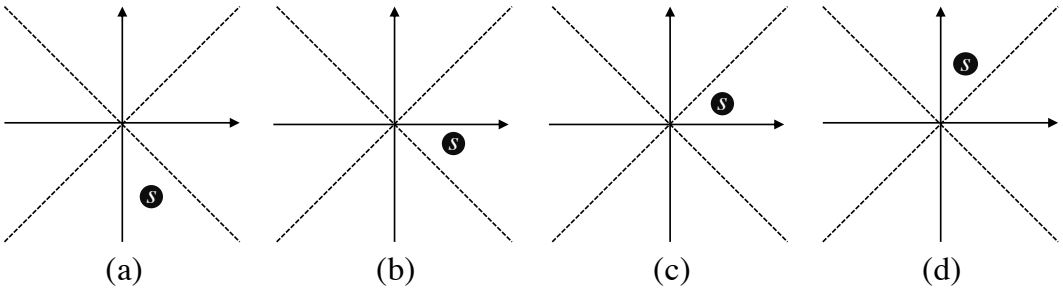

Figure C.1. Four examples of fuzzy results along the elevation direction. In this example $q = 2$. The dotted line shows the center of the bin. (a) $k_{\phi_s} = \tilde{k}_{\phi_s} = 0$; (b) $k_{\phi_s} = 0, \tilde{k}_{\phi_s} = 1$; (c) $k_{\phi_s} = 1, \tilde{k}_{\phi_s} = 0$; (d) $k_{\phi_s} = \tilde{k}_{\phi_s} = 1$.

D. Different BN-ELU order in ResNet block

Empirically, we experimented with different arrangements of batch normalization (BN) and non-linear activation (ELU) in the used ResNet blocks. We illustrate three representative arrangements tested in this work in Fig. D.1. The block B applies BN before ELU, which is the same as its standard applications [12, 13]. The blocks A and C apply BN after ELU, where C differs from A in the shortcut connection. Based on our evaluation on S3DIS, the results of using ResNet Block A, B & C in SegGCN architecture are respectively 63.6%, 63.2%, 62.4%. The Block A performs slightly better than Block B. The significant gap between Block A and C suggests that separate BN application to the two branches before shortcut addition '$\bigoplus$' is more effective, which is eventually done in the proposed network.

E. Robustness to point density (adding data)

We compared the performance of fuzzy kernel and hard kernel for different missing data ratios in Fig. 4 of the main paper. Here, we also study their performance when we increase the density of point clouds instead of decreasing it. The two SegGCN networks using fuzzy and hard spherical kernel that are trained with 8192 points, are tested on input clouds of 16384, 14336, 12288, 10240, 8192 points. Using the convention of the main paper, these points corresponds to data 'drop' ratios of -1, -0.75, -0.5, -0.25, 0, respectively. The performance of both kernels for these cases are shown in Fig. E.1 for both mIoU and mAcc metrics. For completeness, we also show the curve for the missing data. The performance with data addition can be seen on the left side of the dotted lines. The results demonstrate that our fuzzy kernel's performance is more stable as compared to its hard counterpart. We note that both kernels perform more stably in the case of adding data than the scenario of removing it.

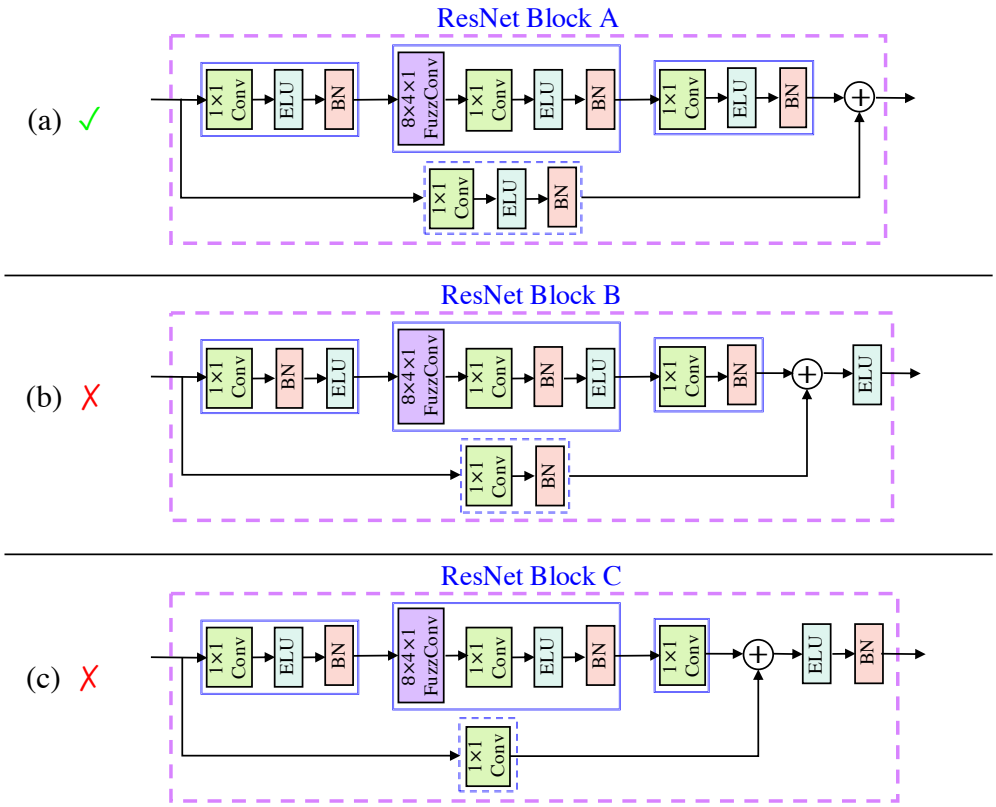

Figure D.1. Illustration of different tested arrangements of batch normalization (BN) and activation (ELU) in ResNet blocks A, B and C. Block A performs slightly better than block B, and much better than block C. Therefore, we exploit block A in our architecture.

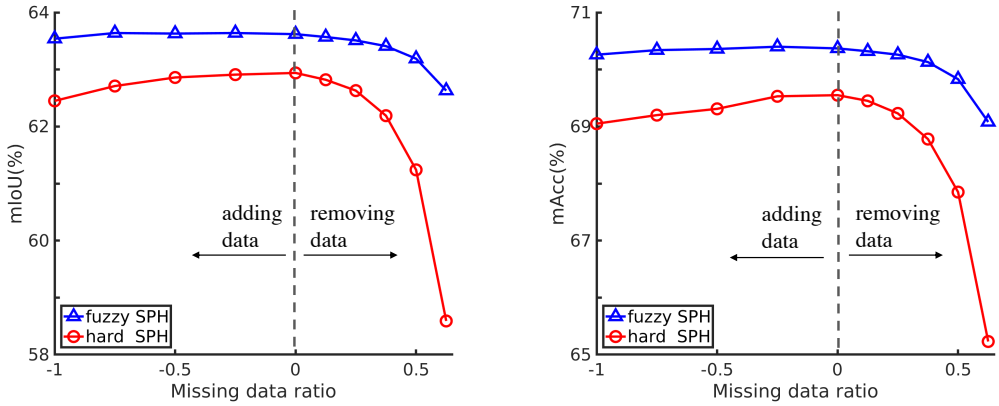

Figure E.1. Robustness of fuzzy kernel vs. hard kernel to adding and removing data. SegGCNs trained on 8129 points with fuzzy SPH kernel and hard SPH kernel are tested by adding and dropping a certain data percentage. The proposed fuzzy SPH kernel shows a clear relative stability over its hard counterpart for a considerable range of missing data. For the adding data curve, the standard deviations of mIoU/mAcc metrics of the fuzzy kernel are 0.04/0.05. Which are over 4 times better than 0.20/0.21 of the hard kernel. For the missing data, the standard deviations of mIoU/mAcc metrics of the fuzzy kernel are 0.37/0.49 - over 3 times better than 1.66/1.65 of the hard kernel.

F. S3DIS 6-fold results

We report the results of SegGCN on all 6 folds of S3DIS dataset in Table F.1. It can be noticed that SegGCN performs on-par with SSP+SPG results - technique that must additionally benefit from local labelling consistency cast by over-segmented point sets. We show representative segmentations generated by SegGCN in the Areas 1,2,3,4, 6 in Fig. F.1.

Table F.1. Performance of SegGCN on all 6 folds of S3DIS. SSP+SPG additionally uses the constraint of local labelling consistency.

	Methods	OA	mAcc	mIoU	ceiling	floor	wall	beam	column	window	door	table	chair	sofa	bookcase	board	clutter
All 6 Folds	PointNet [27]	78.5	66.2	47.6	88.0	88.7	69.3	42.4	23.1	47.5	51.6	42.0	54.1	38.2	9.6	29.4	35.2
	Engelmann et al. [1a]	81.1	66.4	49.7	90.3	92.1	67.9	44.7	24.2	52.3	51.2	47.4	58.1	39.0	6.9	30.0	41.9
	SPG [19]	85.5	73.0	62.1	89.9	95.1	76.4	62.8	47.1	55.3	68.4	73.5	69.2	63.2	45.9	8.7	52.9
	PointCNN [24]	88.1	75.6	65.4	94.8	97.3	75.8	63.3	51.7	58.4	57.2	71.6	69.1	39.1	61.2	52.2	58.6
	SSP+SPG [18]	87.9	**78.3**	68.4	-	-	-	-	-	-	-	-	-	-	-	-	-
	DeepGCN [2a]	85.9	-	60.0	93.1	95.3	78.2	33.9	37.4	56.1	68.2	64.9	61.0	34.6	51.5	51.1	54.4
	SegGCN (Prop.)	87.8	77.1	**68.5**	92.5	**97.6**	**78.9**	44.6	**58.2**	53.7	67.3	**74.6**	**83.9**	**68.0**	**65.7**	46.8	58.5

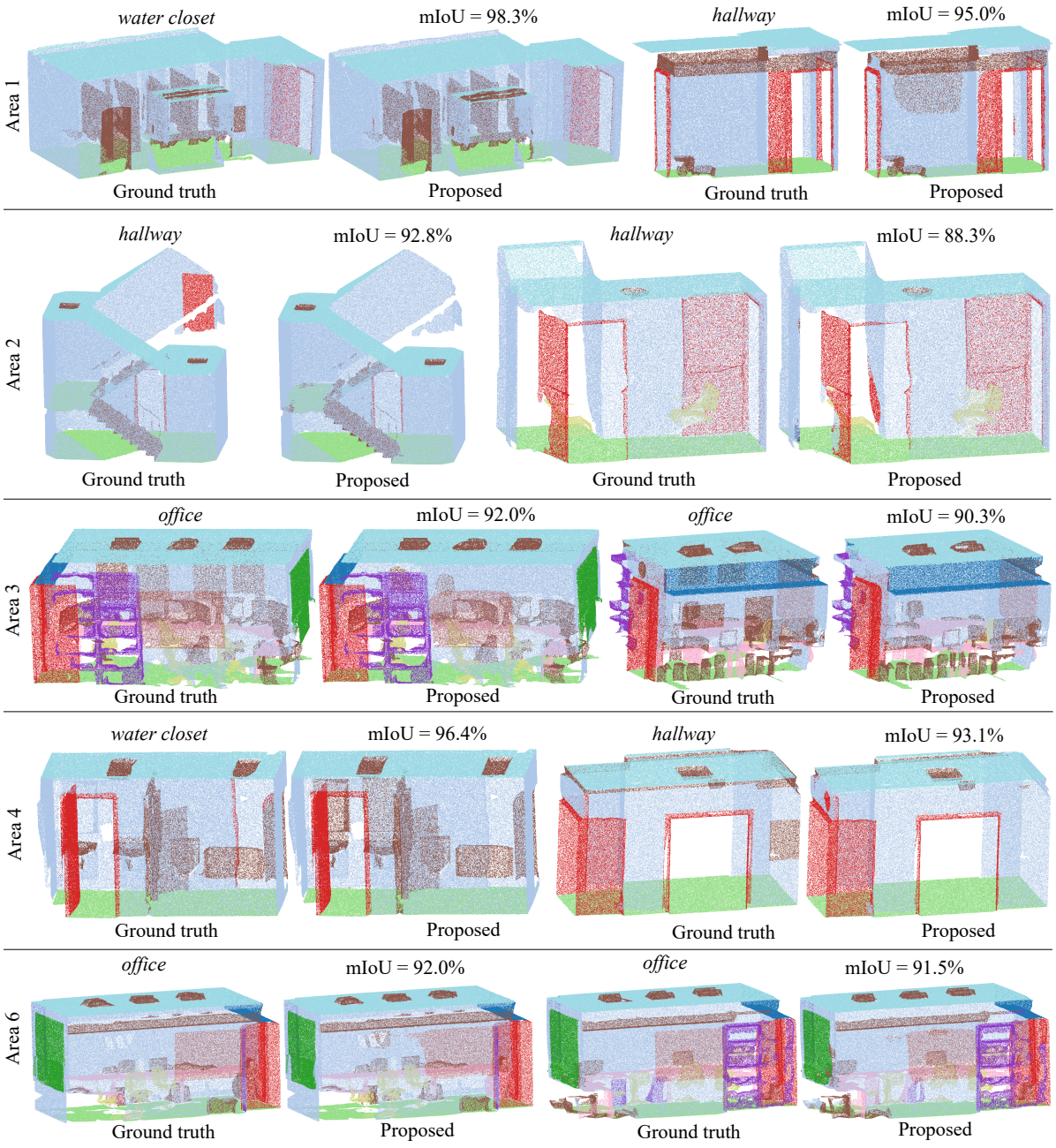

Figure F.1. Visualization of representative segmentations generated by the proposed SegGCN on different Areas of the S3DIS dataset. For a diverse range of scenes (simple to complex), SegGCN is able to segment the point cloud semantics effectively.

G. Visualization of the fuzzy and hard spherical kernels

We show pairs of fuzzy and hard spherical kernels learned in different hierarchical layers of the graph encoder in Fig. G.1. The networks using the two kinds of kernels are both trained on the S3DIS dataset. Their kernel sizes are identical, i.e. $8 \times 4 \times 1 + 1$. For better visualization, we color each bin only on the sphere surface and omit the the weight of self-loop.

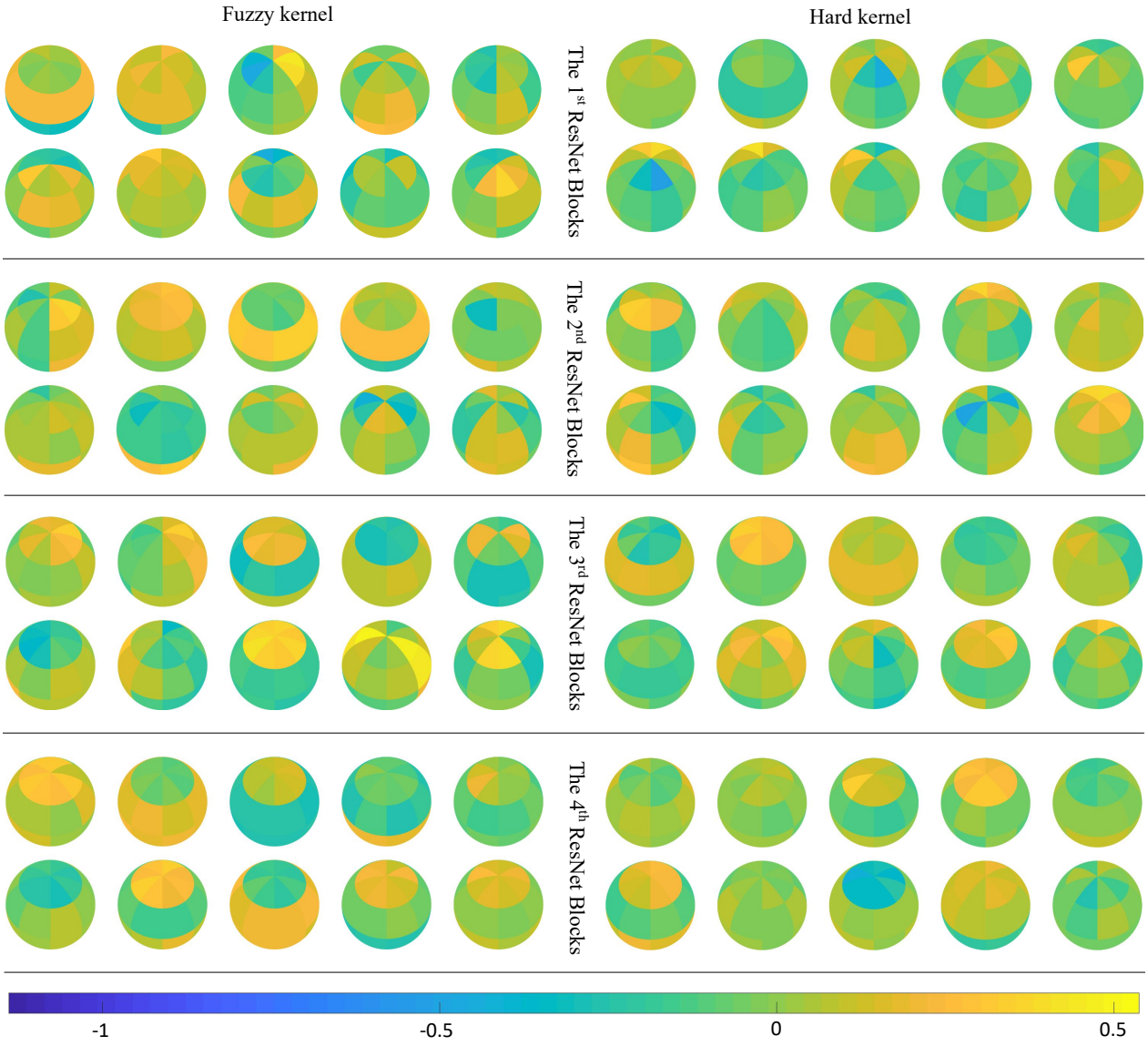

Figure G.1. Pairs of fuzzy and hard spherical kernels learned by SegGCN on the S3DIS dataset. In rows from top to bottom, we show the fuzzy kernels in the left side, and the corresponding hard kernels in the right side. All of the weights are visualized with the same color map shown at the bottom. We can notice that the learned weights generally vary from -1 to 0.5. However, due to the application of fuzzy coefficients in the fuzzy kernel, it learns very different parameters from the hard kernel.

References

[1a] Francis Engelmann, Theodora Kontogianni, Alexander Hermans, and Bastian Leibe. Exploring spatial context for 3D semantic segmentation of point clouds. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 716–724, 2017.

[2a] Guohao Li, Matthias Müller, Ali Thabet, and Bernard Ghanem. DeepGCNs: Can GCNs go as deep as CNNs? In Proceedings of the IEEE International Conference on Computer Vision, 2019.