Collaborative Propagation on Multiple Instance Graphs for 3D Instance Segmentation with Single-point Supervision (Supplementary Material)

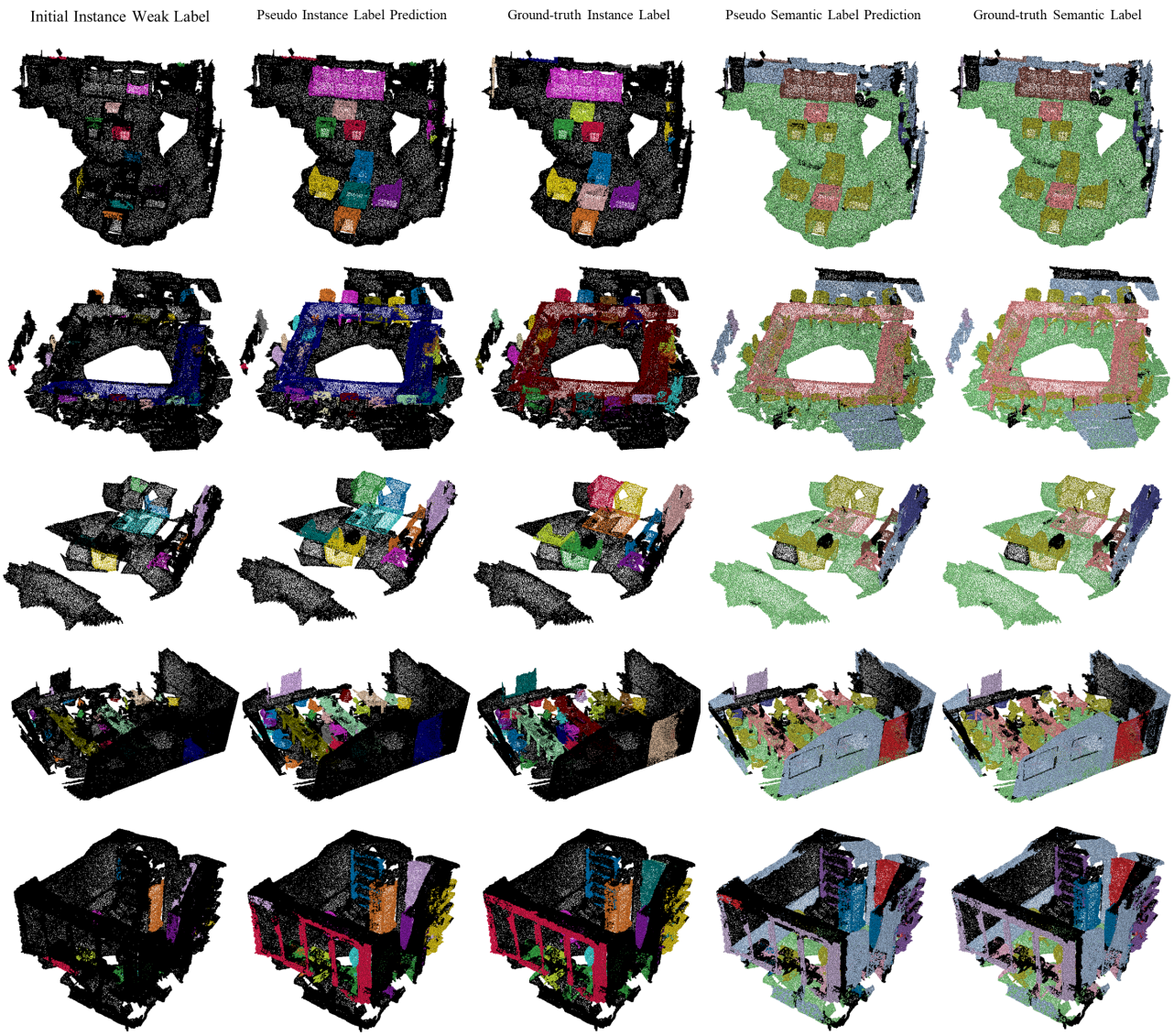

Figure 1. More Qualitative Comparison on ScanNet v2 [3] validation set.

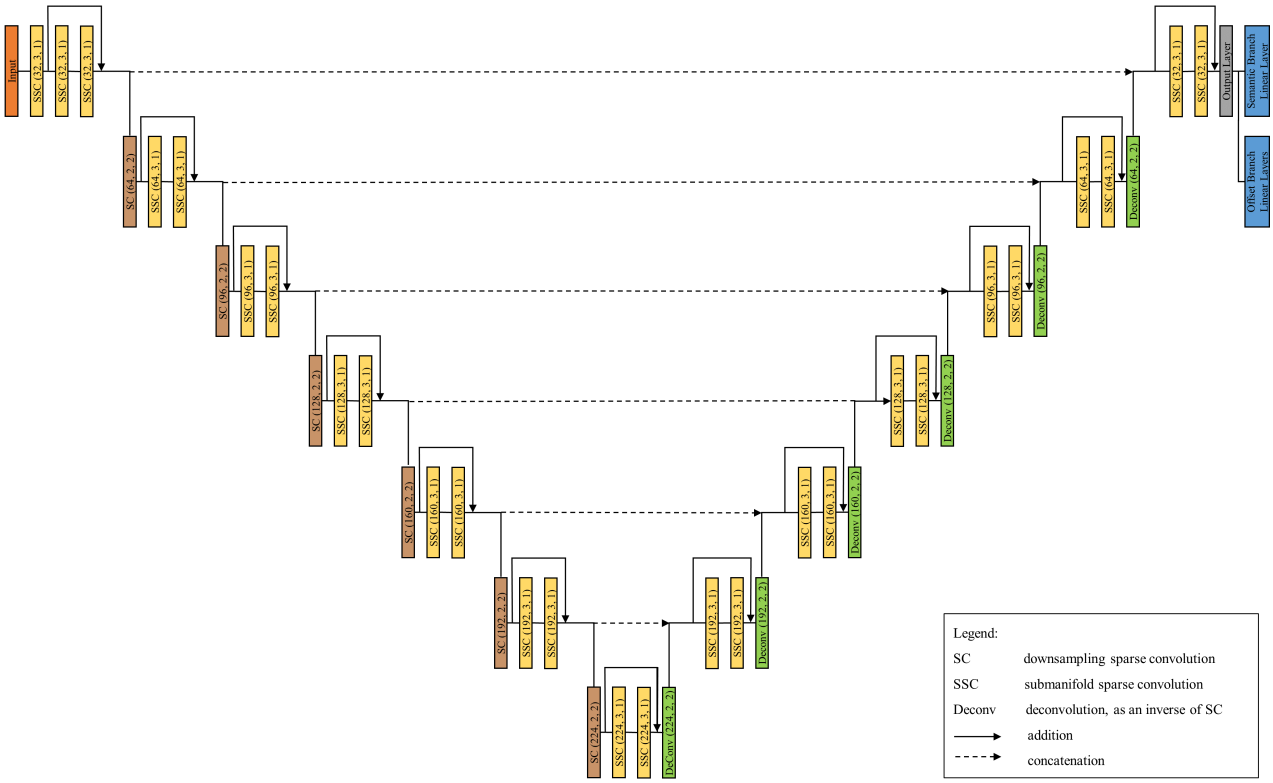

Figure 2. The detailed structure of 3D U-Net backbone with submanifold sparse convolution [6].

1. Additional Qualitative Results

In this section, we show some more visualization results of generated pseudo labels on the training set of ScanNet dataset [3]. As shown in Figure 1, initial instance weak labels are first derived from "one object one point" weak annotations. Then, the proposed method RWSeg can propagate information to unlabelled points. Generated pseudo labels are compared with fully annotated ground-truth for semantic segmentation and instance segmentation respectively. The results show our high-quality pseudo labels have very similar patterns to the actual annotations and contain only minor errors.

2. Network Architecture Details

In this section, we present the detailed structure of our 3D U-Net backbone with submanifold sparse convolution [6] and self-attention module. The backbone network is originally introduced by Graham [5] and has been widely used for feature extraction in point cloud segmentation tasks [13, 10, 8, 7, 2, 11]. The core idea of submanifold sparse convolution is to efficiently process spatially-sparse data, otherwise using normal 3D convolution can be very computationally expensive.

Backbone network In Figure 2, the backbone network takes the sparse voxelized representation of point cloud as input. The U-Net structure is mainly built based on sparse convolution (SC) layers and submanifold sparse convolution (SSC) layers. SC(m, f, s) represents a downsampling sparse convolution (SC) layer with feature dimension m, kernel size f and stride s. Residual connection is used to contain two submanifold sparse convolution (SSC) layers. Deconvolution represents an inverse operation of sparse convolution (SC). The output of the backbone network is split into the semantic branch and offset branch. The semantic branch further utilizes a self-attention layer for feature propagation. For offset branch, point feature vectors are transformed via a two-layer MLP to the dimension of 3, which is then supervised by a regression loss for predicting the centroid shift vectors.

Self-attention module Figure 3 illustrates the process of representing each supervoxel set $\mathcal{V} = p_1, p_2, ..., p_i$ as a super-point. This is achieved by performing an average pooling operation on both the semantic features S and the point coordinates

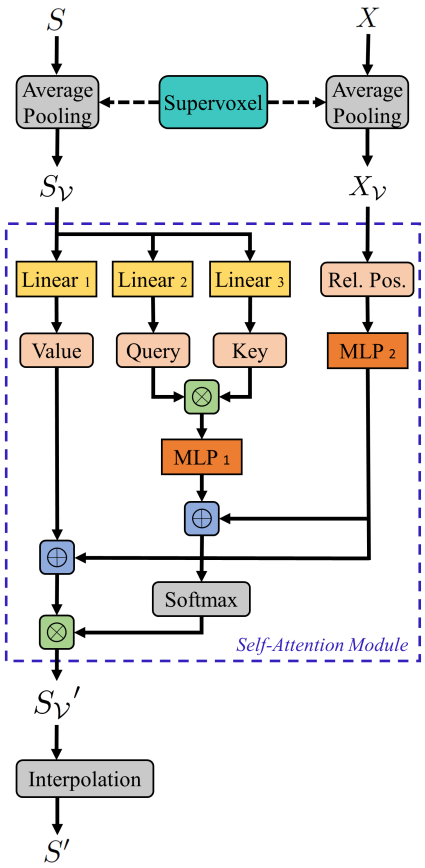

Figure 3. Illustration of self-attention module in semantic branch for feature propagation. $\oplus$ denotes the broadcasting addition and $\otimes$ denotes the element-wise multiplication. Rel. Pos. represents the relative positional similarity of input coordinates.

X for all points belonging to the set. Following [15, 17], we first perform linear transformations of the input semantic features $S_{\mathcal{V}}$ to three matrices as query, key, and value (Q, K, V). Then, matrix $\boldsymbol{A}$ captures the similarity between queries and keys and also includes encoded positional information for adjustment. This can be written as

$$\boldsymbol{Q} = S_{\mathcal{V}} W_Q, \quad \boldsymbol{K} = S_{\mathcal{V}} W_K, \quad \boldsymbol{V} = S_{\mathcal{V}} W_V, \tag{1}$$

$$\boldsymbol{A} = \gamma(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{d}}) + \delta(X_{\mathcal{V}}, X_{\mathcal{V}}), \tag{2}$$

where d is the dimension of Q and V, $X_{\mathcal{V}}$ the coordinates of supervoxels, γ is a mapping function via MLP, δ is a positional similarity function via MLP. The output of self-attention can be formulated as

$$\mathrm{Attn}(S_{\mathcal{V}}) = \sigma(\boldsymbol{A})(\bar{\boldsymbol{V}} + \delta(X_{\mathcal{V}}, X_{\mathcal{V}})), \tag{3}$$

where $\sigma(\cdot)$ is a softmax activation function. $\bar{\boldsymbol{V}}$ denotes a symmetric matrix created by repeatedly expanding $\boldsymbol{V}$. More details can be found in the supplementary material.

Lastly, refined semantic features are interpolated to the original size in point cloud. The training process is supervised by a conventional cross-entropy loss H_{CE} with incomplete labels. We define the semantic loss as

$$L_{sem} = -\frac{1}{N}\sum_{i=1}^{N} H_{CE}(y_i, \widehat{c}_i). \tag{4}$$

where $\widehat{c}_i$ is the weak semantic label. Unlabelled points are ignored here.

Offset loss function Following [8], We use a L_1 regression loss and a cosine similarity based direction loss to train the offset prediction,

$$L_{offset} = \frac{1}{\sum_i m_i} \sum_i ||d_i - (\hat{q}_i - p_i)|| \cdot m_i - \frac{1}{\sum_i m_i} \sum_i \frac{d_i}{||d_i||_2} \cdot \frac{\hat{q}_i - p_i}{||\hat{q}_i - p_i||_2} \cdot m_i. \tag{5}$$

where $\mathbf{m} = \{m_1, ..., m_N\}$ is a binary mask. The value of m_i indicates whether point i is on an instance or not. This means we only consider foreground points with weak labels for supervision.

3. Ablations on Self-attention Module

In Table 1, we perform ablation study on self-attention module by blocking relative position feature on ScanNet v2 [3]. The structure with relative position feature broadcasting addition to both feature branch and attention branch can bring more performance gain.

Relative position usage	Train	Val
Baseline - backbone only	74.6	61.7
None	77.3	64.1
Feature branch only	77.6	64.3
Attention branch only	78.3	65.3
Feature branch + Attention branch	**78.9**	**66**

Table 1. Ablations on Self-attention Module

4. Random Walk with multiple Steps

This section explains how to inference the equation as the final steady-state of the random walk algorithm (From Eq.6 to Eq.7 in original paper).

Random walk algorithm is performed by repeatedly adjusting node vector b via transition matrix $\boldsymbol{A}$. At t-th iteration, the adjustment can be expressed as

$$\boldsymbol{b}_{t+1}^l = \alpha \boldsymbol{A} \boldsymbol{b}_t^l + (1 - \alpha) \boldsymbol{b}_0^l, \tag{6}$$

where b_t is the existing node vector derived at the previous random walk step, b_0 is the initial node vector, $\alpha \in [0, 1]$ is a blending coefficient between propagated score and initial score.

For random walk with multiple steps, we use t to represent the t-th iteration and Expand Eq. (6) to

$$\boldsymbol{b}_{t+1}^l = (\alpha \boldsymbol{A})^{t+1} \boldsymbol{b}_0^l + (1 - \alpha) \sum_{i=0}^{t} (\alpha \boldsymbol{A})^i \boldsymbol{b}_0^l. \tag{7}$$

Applying $t \to \infty$, since $\alpha \in [0, 1]$, the first term in Eq. (7) turns into

$$\lim_{t \to \infty} (\alpha \boldsymbol{A})^{t+1} \boldsymbol{b}_0^l = 0. \tag{8}$$

For the second term with matrices can be expanded as

$$\lim_{t \to \infty} \sum_{i=0}^{t} (\alpha \boldsymbol{A})^i \boldsymbol{b}_0^l = (\boldsymbol{I} - \alpha \boldsymbol{A})^{-1} \boldsymbol{b}_0^l, \tag{9}$$

where $\boldsymbol{I}$ is the identity matrix. Thus, the final steady-state of random walk algorithm can be written as

$$\boldsymbol{b}_{(\infty)}^l = (1 - \alpha)(\boldsymbol{I} - \alpha \boldsymbol{A})^{-1} \boldsymbol{b}_0^l. \tag{10}$$

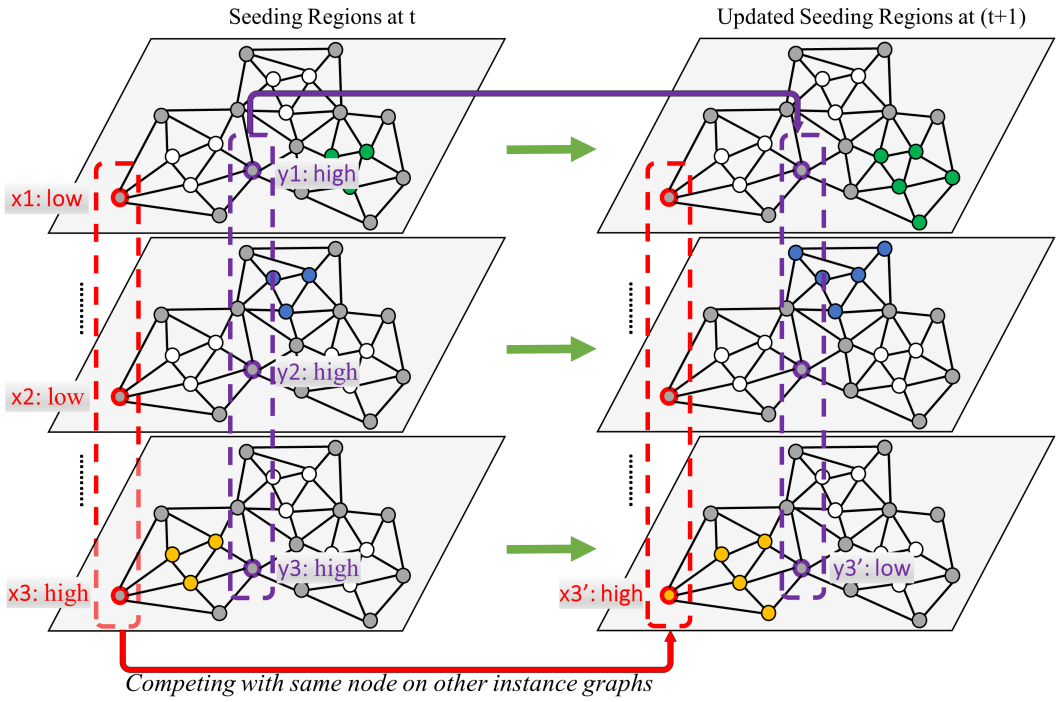

Figure 4. Illustration of competing mechanism in CRW. This example shows the effect of competition on two different nodes (x3 in red, y3 in purple).

5. Competing Mechanism in CRW

For illustrative purposes, we present an example in Figure 4. In this case, the foreground category consists of three instance graphs, each with a distinct seeding point marked in green, blue, and yellow, respectively. Node $x3$ (in red) has two nodes in the same position, $x1$ and $x2$. At step t, their node scores are determined by their overall distance to the seeding points. Since $x1$ is far from the seeding points marked in green, its score will be low after a random walk step, whereas $x3$, which is closer to the seeding points marked in yellow, will have a higher score. After applying SoftMax normalization to the scores of $x1$, $x2$, and $x3$, the output score for $x3'$ at step $t+1$ will be high, as it faces less competition from the other two nodes.

Similarly, we have a node $y3$ with two same-positioned nodes, $y1$ and $y2$, placed in the center of three instances. At step t, the scores of $y1$, $y2$, and $y3$ are all high. However, during normalization, $y3$ receives a strong repulsive effect from $y1$ and $y2$. Thus, the output score $y3'$ at step $t+1$ will be low.

Finally, the proposed algorithm compares the node scores at step $t+1$. In this case, the node $x3'$ will have a higher priority to be grouped into seeds than $y3'$. This is because node $x3'$ is highly likely from the instance in yellow, whereas there is lower confidence in $y3'$. Therefore, we leave this node to be grouped in the later steps.

6. Ablations on hyperparamters in CRW

In Table 2, we show the experimental results with varying hyperparameters for the competing mechanism in CRW. The considered baseline is the proposed baseline random walk algorithm, which is represented by $t_{2_{max}} = 0$.

The table illustrates that a lower update percentage θ typically leads to better results but requires more iterations $t_{2_{max}}$, as it gradually groups the most confident points with our competing mechanism. The improvements over the random walk baseline are consistently observed. As discussed in the paper, the extent of the improvement depends on the distribution of the dataset. Notably, the proposed design in CRW is particularly effective in solving challenging cases, such as those with compacted objects of the same class.

Update percentage θ	Iteration number t_{2max}	AP (chair)	AP (bksf)
N.A.	0	64.2	48.1
80%	5	66.7 (+2.5)	49.9 (+1.8)
50%	5	67.4 (+3.2)	52.3 (+4.2)
20%	5	67 (+2.8)	53.4 (+5.3)
20%	20	67.3 (+3.1)	54.4 (+6.3)
10%	50	67.3 (+3.1)	55.1 (+7.0)

Table 2. Experiments with different CRW hyperparameters on ScanNet v2 [3]

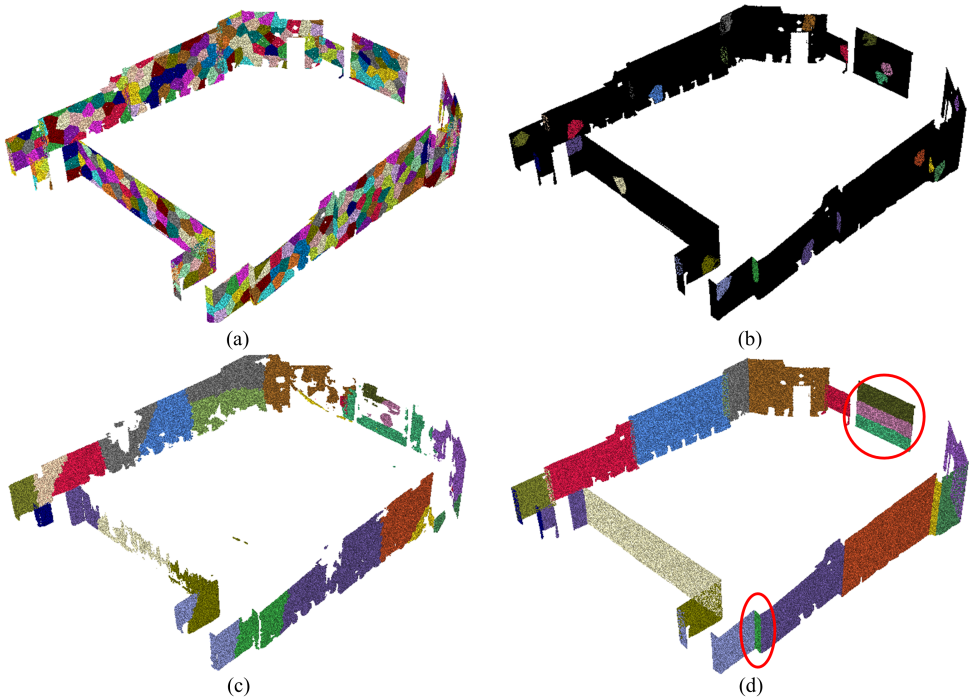

Figure 5. Limitations of RWSeg on S3DIS [1] dataset. (a) generated supervoxels (b) initial instance weak labels (c) generated instance pseudo labels (d) ground-truth instance labels

7. Additional Analysis in CRW

After conducting experiments with various values of hyperparameters for t_1max and α, we have observed that our algorithm can converge after just a single random walk step. Further increasing the iteration number of t_1max only results in marginal improvements. We argue that the fully connected graph used in our algorithm has a wide influence field. This means that it can exert its influence over a large area, and consequently, reduces the need for multiple random walk steps.

Hyperparameter $\alpha \in [0, 1]$ is used to control the trade-off between propagated node values and initial node values. Intuitively, it prevents deviating too fast from initial segmentation values. In our experiments, different values of α create a minor influence on the final converged results (less than 0.2% in mAP). However, if we set the value of α to 1 to remove the effect from initial values, the performance of random walk is dropped by 1% in mAP.

8. Limitations of RWSeg

In S3DIS [1] dataset, some background stuff such as walls, ceilings, boards are also treated as instances by their setting. As shown in Figure 5 (d), walls are intentionally labeled as separate instances, even though they are part of the background. Additionally, these walls can vary greatly in size, which poses a challenge for our method. Our method is primarily designed for common instance types and may struggle to make accurate predictions on these cases, especially with limited initial weak labels. In practice, one possible solution is to use surface normals as a clue and apply unsupervised plane estimation methods. However, this is beyond the scope of this work and goes beyond our objectives.

9. Comparisons between Feature Propagation Methods

Graph Neural Network (GNN) is a widely used message passing network for feature propagation. It usually iteratively updates the representation of a node by aggregating and fusing information from its local neighbors. A major advantage of self-attention over GNN is its global receptive field.

Besides, some graph propagation mechanisms require minimizing a separated pairwise energy function, such as conditional random field (CRF) [9] loss. Whereas our self-attention module can be directly integrated after a CNN-based backbone and supervised by the original loss functions.

10. Implementation of RWSeg

The license information of assets used in our paper are listed in Table 3. Our code will be available on the Github once the paper is accepted.

Asset	License websites
Pytorch [14]	https://github.com/pytorch/pytorch/blob/master/LICENSE
SparseConvNet [5]	https://github.com/facebookresearch/SparseConvNet/blob/main/LICENSE
Mesh-Segmentator [4]	http://cs.brown.edu/people/pfelzens/segment/index.html
Supervoxel-3D [12]	https://github.com/yblin/Supervoxel-for-3D-point-clouds
PointGroup [8]	https://github.com/dvlab-research/PointGroup/blob/master/LICENSE
PtXFMR [16]	https://github.com/lucidrains/point-transformer-pytorch/blob/main/LICENSE
ScanNet dataset [3]	https://github.com/ScanNet/ScanNet/blob/master/LICENSE
S3DIS dataset [1]	http://buildingparser.stanford.edu/dataset.html

Table 3. The assets used in the paper and their corresponding license websites

References

[1] Iro Armeni, Ozan Sener, Amir R. Zamir, Helen Jiang, Ioannis Brilakis, Martin Fischer, and Silvio Savarese. 3d semantic parsing of large-scale indoor spaces. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, 2016. 6, 7

[2] Shaoyu Chen, Jiemin Fang, Qian Zhang, Wenyu Liu, and Xinggang Wang. Hierarchical aggregation for 3d instance segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 15467–15476, October 2021. 2

[3] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2017. 1, 2, 4, 6, 7

[4] Pedro Felzenszwalb and Daniel Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59:167–181, 09 2004. 7

[5] Benjamin Graham, Martin Engelcke, and Laurens van der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks. *CVPR*, 2018. 2, 7

[6] Benjamin Graham and Laurens van der Maaten. Submanifold sparse convolutional networks. *CoRR*, abs/1706.01307, 2017. 2

[7] Lei Han, Tian Zheng, Lan Xu, and Lu Fang. Occuseg: Occupancy-aware 3d instance segmentation, 2020. 2

[8] Li Jiang, Hengshuang Zhao, Shaoshuai Shi, Shu Liu, Chi-Wing Fu, and Jiaya Jia. Pointgroup: Dual-set point grouping for 3d instance segmentation, 2020. 2, 4, 7

[9] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc. 7

[10] Jean Lahoud, Bernard Ghanem, Marc Pollefeys, and Martin R. Oswald. 3d instance segmentation via multi-task metric learning, 2019. 2

[11] Zhihao Liang, Zhihao Li, Songcen Xu, Mingkui Tan, and Kui Jia. Instance segmentation in 3d scenes using semantic superpoint tree networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2783–2792, October 2021. 2

[12] Yangbin Lin, Cheng Wang, Dawei Zhai, Wei Li, and Jonathan Li. Toward better boundary preserved supervoxel segmentation for 3d point clouds. *ISPRS Journal of Photogrammetry and Remote Sensing*, 143:39 – 47, 2018. ISPRS Journal of Photogrammetry and Remote Sensing Theme Issue "Point Cloud Processing". 7

[13] Chen Liu and Yasutaka Furukawa. MASC: multi-scale affinity with sparse convolution for 3d instance segmentation. *CoRR*, 2019. 2

[14] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. 7

[15] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. 3

[16] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip Torr, and Vladlen Koltun. Point transformer, 2020. 7

[17] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip Torr, and Vladlen Koltun. Point transformer. In *ICCV*, 2021. 3