Learning Superpoint Graph Cut for 3D Instance Segmentation

Anonymous Author(s)
Affiliation
Address
email

Abstract

3D instance segmentation is a challenging task due to complex local geometric structures of objects in point clouds. In this paper, we propose a learning-based superpoint graph cut method that explicitly learns the local geometric structures of the point cloud for instance segmentation. Specifically, we first oversegment the raw point clouds into superpoints and construct the superpoint graph. Then, we construct an edge score prediction network to predict the edge scores of the superpoint graph, where the similarity vectors of two adjacent nodes learned through cross-graph attention in the coordinate and feature spaces are used for regressing edge scores. By forcing two adjacent nodes of the same instance to be close to the instance center in the coordinate and feature spaces, we formulate a geometry-aware edge loss to train the edge score prediction network. Finally, we develop a superpoint graph cut network that employs the learned edge scores and the predicted semantic classes of nodes to generate instances, where bilateral graph attention is proposed to extract discriminative instance features on the coordinate and feature spaces for predicting semantic labels and scores of instances. Extensive experiments on two challenging datasets, ScanNet v2 and S3DIS, show that our method achieves new state-of-the-art performance.

1 Introduction

In recent years, with the development of 3D sensors, such as LiDAR and Kinect camera, various 3D computer vision tasks have been receiving more and more attention since a large number of large-scale 3D datasets are built. Among them, 3D instance segmentation is a fundamental task in 3D scene understanding and has been widely used in kinds of applications such as self-driving cars, virtual reality, robotic navigation. Although recent progress in 3D instance segmentation is encouraging, it is still a challenging task due to 3D point irregularities and context uncertainties in 3D scenes with complex geometric structures.

Many efforts have been dedicated to 3D instance segmentation and achieved promising performance. These methods can be mainly divided into two categories: detection-based [39, 40] and clustering-based [35, 14] methods. Among detection-based methods, 3D-BoNet [39] first detects the 3D bounding boxes and then employs a mask prediction network to predict the object mask for instance segmentation. However, for objects with complex geometric structures, detection-based methods [40] cannot obtain accurate 3D bounding boxes, thereby degrading the instance segmentation performance. The clustering-based method SGPN [35] clusters points based on the semantic segmentation to generate instances. Different from SGPN, Jiang *et al.* [14] develops offset branch to cluster points based on semantic predictions in dual coordinate spaces, including original and shifted coordinate spaces. Besides, some follow-up methods utilize tree structures [21], hierarchical aggregation [2], and soft semantic segmentation [32] to further boost the instance segmentation performance. Nonetheless,

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

most of these clustering-based methods predict center offsets and semantics without effectively capturing the local geometric structures of the objects. Therefore, the instance segmentation performance is limited, especially for complex scenes.

In this paper, we propose a learning-based superpoint graph cut method that explicitly learns the local geometric structures of point clouds to segment 3D instances. It consists of an edge score prediction network to predict edge scores and a superpoint graph cut network to generate instances. In our method, we oversegment the raw point clouds into superpoints and construct the superpoint graph by linking the k-nearest superpoints in the coordinate space. In the edge score prediction network, we first perform cross-graph attention on the local neighborhoods of two adjacent nodes to extract local geometric features for measuring the similarity of the nodes. Then, based on the learned features form the coordinate and feature spaces, we adopt an edge score branch to predict the edge scores. In addition, we propose a geometry-aware edge loss to train the edge score prediction network by forcing the adjacent nodes of the same instance to be close to the instance center in both the coordinate and feature spaces. Once we obtain the edge scores, in the superpoint graph cut network, the predicted edge scores and predicted semantic classes of the nodes are used to cut the edges for forming object instances. The instances are obtained by applying the breadth-first-search algorithm on the superpoint graph to aggregate nodes in the same connected component. In each instance, we then apply bilateral graph attention to aggregate local geometric features to obtain discriminative instance features for predicting classes and scores of instances. In order to refine the instance, we further adopt a mask learning branch to mask the low confidence superpoints within the instance for refining the instance.

In summary, the contributions of this paper are as follows:

- We present an edge score prediction network that learns the local geometric features of adjacent nodes for producing edge scores. To train it, we also propose a geometry-aware edge loss to keep the instance compact in the coordinate and feature space simultaneously.
- We present a superpoint graph cut network that extracts discriminative instance features to generate accurate instances by using bilateral graph attention in the coordinate and feature spaces.
- Extensive experiments on the ScanNet v2 [4] and S3DIS [1] datasets show that our method achieves new state-of-the-art performance. On the online test set of ScanNet v2, our method achieves the performance of 55.2% in terms of mAP, which is 4.6% higher than the current best results [21]. For S3DIS, our method outperforms the current best results [32] over 2% in terms of mAP.

2 Related Work

3D semantic segmentation. Extracting features from irregular 3D point clouds is crucial for 3D semantic segmentation. Qi *et al.* [26] first proposed PointNet to learn point-wise features from a point set for semantic segmentation through the multi-layer preceptron network. Following it, many efforts [20, 29, 41, 12, 42] have been proposed to improve semantic segmentation performance. Early point-based methods [31, 38, 27, 11] design various local feature aggregation strategies to extract discriminative point-wise features for semantic segmentation. Inspired by successful 2D convolution networks, view-based methods [19, 30, 13, 15] project the point cloud into multiple regular 2D views, where the regular 2D convolution is applied to extract features. In addition to view-based methods, volumetric-based methods [23, 34, 7, 3] first voxelize the point cloud into regular 3D grids and then apply 3D convolution to extract local features of point clouds. In order to capture the local geometric structures of point clouds, graph-based methods [37, 18, 33] construct the graph on point clouds and utilize graph convolution to aggregate local geometric information for semantic segmentation.

3D instance segmentation. 3D instance segmentation is a more challenging task, which further needs to identify each instance. Current methods can be roughly grouped into two classes.

Detection-based methods [40, 10, 22] first detect 3D bounding boxes of each object in point clouds, and then apply a mask prediction network on each box to predict the object mask for instance segmentation. In [39], a 3D instance segmentation framework dubbed 3D-BoNet was proposed, which directly regresses the 3D bounding boxes for all instances and predicts point-level masks for each instance. Yi *et al.* [40] proposed a generative shape proposal network that generates proposals

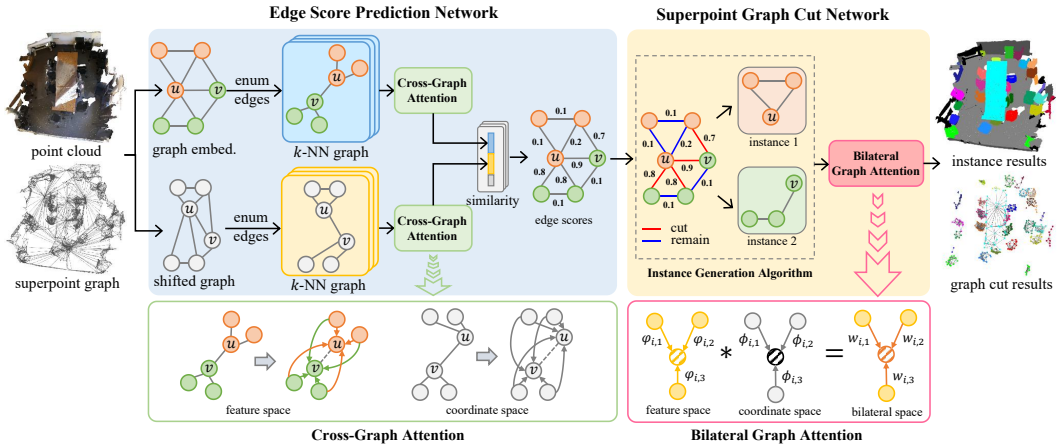

Figure 1: The framework of our learning-based superpoint graph cut method. Given a superpoint graph, we first extract superpoint features (omit in the figure). We then construct an edge score prediction network to extract edge embeddings from the coordinate and feature spaces for predicting edge scores. Finally, we develop a superpoint graph cut network to generate accurate instances.

by reconstructing shapes from noisy observations in a scene for instance segmentation. In addition, using both geometry and RGB inputs, [10] develops a joint 2D-3D feature learning network that combines the 2D and 3D features to regress 3D object bounding boxes and predict instance masks.

Clustering-based methods usually use point similarity [35], semantic maps [8, 16], or geometric shifts [14, 2, 21] to cluster 3D points into object instances. A similarity group proposal network was proposed in [35] to cluster points by learning point-wise similarity for generating instances. [25] proposes a multi-task learning framework that simultaneously learns semantic classes and high-dimensional embeddings of 3D points to cluster the points into object instances. In [36], a segmentation framework was introduced to learn semantic-aware point-wise instance embedding for associatively segmenting instances and semantics of point clouds. Han *et al.* [8] proposed an occupancy-aware method to predict the number of occupied voxels for each instance. PointGroup [14] clusters points by using predicted point-wise center offset vectors and point-wise semantic labels. The follow-up method [2] adopts a hierarchical aggregation strategy for instance segmentation, which first performs point aggregation to cluster points into preliminarily sets, and then performs set aggregation to cluster sets into instances. Lately, Vu *et al.* [32] proposed a soft grouping strategy to mitigate the problem of semantic prediction errors by associating each point with multiple classes, yielding in significant performance gains in 3D instance segmentation. In addition, a semantic superpoint tree network called SSTNet, was proposed in [21] for segmenting point clouds in instances. It first groups superpoints with similar semantic features to build a binary tree, and then generates instances by tree traversal and splitting. To make the network more efficient, a dynamic convolution network combined with a small Transformer network was constructed to propose a lightweight 3D instance segmentation method [9].

3 Method

An overview of our learning-based superpoint graph cut method is illustrated in Figure 1. Based on the superpoint graph, the edge score prediction network (Sec. 3.1) extracts edge embeddings from the coordinate and feature spaces for predicting edge scores. After that, the superpoint graph cut network (Sec. 3.2) generates accurate object instances by learning discriminative instance features to predict classes and scores of instances. Finally, in Sec. 3.3, we describe how to train our method and inference instances from point clouds.

3.1 Edge Score Prediction Network

Given a raw point cloud, we oversegment it into superpoints and construct the superpoint graph $\mathcal{G} = (V, E)$, where V represents the node set of superpoints and E represents the edge set. We use the submanifold sparse convolution [7] followed by the edge-conditioned convolution [28] to extract superpoint features $\boldsymbol{F} \in \mathbb{R}^{|V| \times C}$, where C is the feature dimension.

3.1.1 Edge Feature Embedding

Once we obtain superpoint features, the edge score prediction network extracts edge embeddings to predict edge scores for proposing instances. Given the adjacent nodes $(u, v) \in E$, it is desired that the learned edge embedding can effectively identify whether nodes u and v belong to the same instance. To tackle this problem, we extract the feature vectors of nodes u and v to form the edge embedding. Here, we characterize the similarity of nodes u and v in both coordinate and feature spaces for producing discriminative edge embeddings.

Edge embedding in coordinate space. To characterize the similarity of nodes u and v, we first shift them toward the corresponding instance centroids in the coordinate space. Here, a multi-layer preceptron (MLP) network encodes $\boldsymbol{F}$ to produce $|V|$ offset vectors $\boldsymbol{O} = \{\boldsymbol{o}_1, \ldots, \boldsymbol{o}_{|V|}\} \in \mathbb{R}^{|V|\times 3}$. Give the original superpoint coordinates $X = \{x_1, \ldots, x_{|V|}\} \in \mathbb{R}^{|V|\times 3}$, the shifted superpoint coordinates $\hat{X} = \{\hat{x}_1, \ldots, \hat{x}_{|V|}\}$ are obtained by $\hat{X} = X + O$. In this way, the geometric distance of nodes belonging to different instances will be increased, so that the discrimination of the superpoints will be enhanced. After that, based on the shifted coordinate space, for node u, we leverage its k-nearest superpoints ($i.e.$, $\mathcal{N}_u$) to construct the local k-NN graph G_u. Similarly, we can obtain the graph G_v for node v. Then, we perform cross-graph attention across G_u and G_v to characterize the similarity of nodes through the learned feature vectors, as shown in Figure 1. Taking the node u as an example, the weight $\boldsymbol{\alpha}$ of cross-graph attention is defined as:

$$\alpha_{u,i} = \mathrm{MLP}(\hat{x}_i - \hat{x}_u), \forall i \in \mathcal{N}_u \cup \mathcal{N}_v \tag{1}$$

where $\hat{x}_i$ and $\hat{x}_u$ are the shifted coordinates. Note that i enumerates the all $2{*}k$ neighbors across two graphs. Therefore, the final output feature vector can be formulated as:

$$\boldsymbol{h}_u = \sum_{i \in \mathcal{N}_u \cup \mathcal{N}_v} \hat{\alpha}_{u,i} * \mathrm{MLP}(\hat{x}_i) + b_i \tag{2}$$

where $\hat{\alpha}_{u,i}$ is the weight $\alpha_{u,i}$ after softmax and b_i is a learnable bias. The learned feature vector $\boldsymbol{h}_u \in \mathbb{R}^C$ can characterize the geometry similarity by adaptively learning the geometric differences on two graphs. In the same way, we can obtain feature vector $\boldsymbol{h}_v$ for another node v. We combine the feature vectors $\boldsymbol{h}_u$ and $\boldsymbol{h}_v$ as the edge embedding in coordinate space: $\boldsymbol{e}_{u,v} = [\boldsymbol{h}_u, \boldsymbol{h}_v]$.

Edge embedding in feature space. In addition to the coordinate space, we also consider the feature space to extract discriminative edge embeddings. Firstly, a MLP network encodes $\boldsymbol{F}$ to produce the initial feature embedding $\boldsymbol{Z} \in \mathbb{R}^D$. By pushing feature embeddings of instances away from each other, we enlarge the gap between different instances in the feature space. Given a pair of nodes $(u, v) \in E$, we construct the k-NN graphs $\hat{G}_u$ and $\hat{G}_v$ in the feature space, respectively. In this way, it is expected that each graph can aggregate superpoints within the same instance. Then, we execute cross-graph attention across $\hat{G}_u$ and $\hat{G}_v$ to characterize the similarity of nodes in the feature space. Finally, we can obtain the feature vectors $\hat{\boldsymbol{h}}_u \in \mathbb{R}^C$ and $\hat{\boldsymbol{h}}_v \in \mathbb{R}^C$. If u and v belong to the same instance, they share the similar k-NN graph in the feature space, so that the learned feature vectors $\hat{\boldsymbol{h}}_u$ and $\hat{\boldsymbol{h}}_v$ are similar to each other. Here, we also combine the feature vectors to obtain edge embedding in feature space: $\hat{\boldsymbol{e}}_{u,v} = [\hat{\boldsymbol{h}}_u, \hat{\boldsymbol{h}}_v]$.

Edge score prediction. After obtaining edge embeddings in the coordinate and feature spaces, we utilize a simple MLP network to generate the edge score, which is defined as:

$$a_{u,v} = \sigma(\mathrm{MLP}([\boldsymbol{e}_{u,v}, \hat{\boldsymbol{e}}_{u,v}, d_{u,v}])) \tag{3}$$

where $[\cdot, \cdot, \cdot]$ indicates the concatenation operation, σ denotes the sigmoid function, and $d_{u,v}$ represents the geometric distance between the nodes u and v in the shifted coordinate space. In the experiment, if the edge score $a_{u,v} > 0.5$, it means that the edge between nodes u and v should be cut from the superpoint graph. We use the binary cross-entropy loss $\mathcal{L}_{edge}$ to minimize the edge score.

3.1.2 Geometry-Aware Edge Loss

To train the edge score prediction network, we employ the geometric structures of the superpoint graph to form a geometry-aware edge loss, as shown in Figure 2. Specifically, given the nodes u, v and their corresponding instance centroids c_u and c_v, we draw nodes toward their instance centroids by minimizing the L_2 distance d_{u,c_u} and d_{v,c_v}. Further, when $(u, v) \in E$ belong to the same instance, it is expected that they can collaboratively shift to the same instance centroid c by minimizing the area of triangle $\triangle uvc$. While $(u, v) \in E$ belong to different instances, it is expected that they can collaboratively shift to their own instance centroids c_u and c_v by minimizing the area of triangles

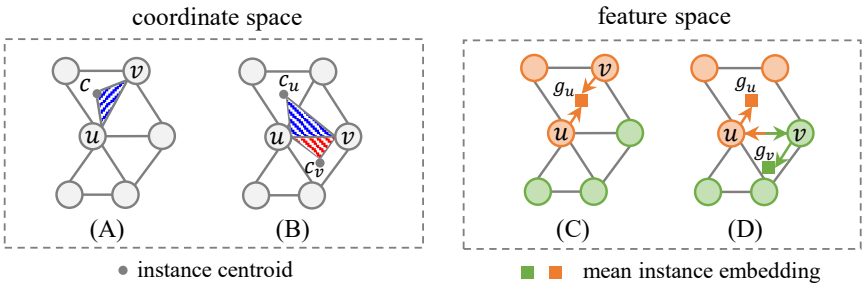

Figure 2: The details of our geometry-aware edge loss. u and v in graphs (A) and (C) belong to the same instance, and u and v in graphs (B) and (D) belong to different instances.

$\triangle uvc_u$ and $\triangle uvc_v$. The area constraint in the coordinate space is written as:

$$\mathcal{L}_{area} = \frac{1}{|E|} \sum_{(u,v)\in E} \|\hat{x}_u - c_u\|_2 + \|\hat{x}_v - c_v\|_2 + \frac{1}{2}(|(c - \hat{x}_u) \times (c - \hat{x}_v)|\mathbb{I}(u,v) \\ +(|(c_u - \hat{x}_u) \times (\hat{x}_v - \hat{x}_u)| + |(c_v - \hat{x}_v) \times (\hat{x}_u - \hat{x}_v)|)(1 - \mathbb{I}(u,v))) \tag{4}$$

where $\mathbb{I}(u,v)$ is the indicator function, and $\mathbb{I}(u,v)$ equals to 1 if u and v belong to the same instance, and 0, otherwise. Note that "$\times$" represents the outer product of vectors for computing the area of triangles. Likewise, we expect the nodes in the same instance to be compact in the feature space by constraining their feature embeddings. For $(u,v) \in E$ belong to the same instance, we draw embeddings of u and v toward the mean embedding of the instance, and also pull them to each other. For $(u,v) \in E$ belong to different instances, we push embeddings of u and v away from each other. In addition, the instances are pushed away from each other by increasing the distance of their own mean embedding of instances. Thus, the constraint in the feature space is written as:

$$\mathcal{L}_{feat} = \frac{1}{|E|} \sum_{(u,v)\in E} ([\|\boldsymbol{z}_u - \boldsymbol{z}_v\|_2 - \delta]_+^2 + [\|\boldsymbol{z}_u - \boldsymbol{g}_u\|_2 - \delta]_+^2)\mathbb{I}(u,v) \\ +([2\beta - \|\boldsymbol{z}_u - \boldsymbol{z}_v\|_2]_+^2 + [2\beta - \|\boldsymbol{g}_u - \boldsymbol{g}_v\|_2]_+^2)(1 - \mathbb{I}(u,v)) \tag{5}$$

where $\boldsymbol{z}_u \in \mathbb{R}^D$ and $\boldsymbol{z}_v \in \mathbb{R}^D$ are the feature embeddings. Note that $\boldsymbol{g}_u \in \mathbb{R}^D$ and $\boldsymbol{g}_v \in \mathbb{R}^D$ indicate the mean feature embeddings of the instances that u and v belong to, respectively. The thresholds δ and β are set to be 0.1 and 1.5 to ensure that the inter-instance distance is higher than the intra-instance. Finally, the geometry-aware edge loss is defined as:

$$\mathcal{L}_{geo} = \mathcal{L}_{area} + \mathcal{L}_{feat} + \mathcal{L}_{edge} \tag{6}$$

3.2 Superpoint Graph Cut Network

3.2.1 Instance Generation via Superpoint Graph Cut

Given the edge score $\boldsymbol{A} = \{a_{u,v}\} \in \mathbb{R}^{|E|\times 1}$, we propose an instance generation algorithm to generate object instances by simultaneously employing the learned edge scores and the predicted semantic classes of nodes. Specifically, in order to mitigate semantic prediction errors, we first adopt a soft threshold θ to associate the superpoint nodes with multiple classes. Given semantic scores of superpoints $\boldsymbol{S} = \{\boldsymbol{s}_1, \ldots, \boldsymbol{s}_{|V|} \mid \boldsymbol{s}_i \in \mathbb{R}^N \text{ for } i = 1, \ldots, |V|\}$, where N is the number of classes, if $s_v^i < \theta$, the v-th superpoint can be associated with the i-th class. In this way, for the i-th class, we can slice a superpoint subset C_i on the superpoint graph, where the semantic score of the superpoint on the i-th class index is higher than θ. Then, on the superpoint graph, for the edge $(u,v) \in E$, if nodes $u \in C_i$ and $v \in C_i$, the edge (u,v) will be preserved, otherwise the edge will be deleted. In other words, we remove the edge between two superpoint nodes with different semantics. After that, for the preserved edges (u,v) on the superpoint graph, we utilize the edge score $a_{u,v}$ to determine whether the edge should be cut from the superpoint graph. In the experiment, the threshold for cutting the edge is set to 0.5. If the edge score is higher than 0.5, the edge will be cut from the superpoint graph. Finally, we adopt the breadth-first-search on the superpoint graph to aggregate nodes in the same connected component for generating instances for the i-th class. In this way, we can generate all instances for N classes by iterating through N classes. The details are shown in Algorithm 1.

3.2.2 Bilateral Graph Attention for Instance Embedding

As we obtain instances $\boldsymbol{I} = \{I_1, \ldots, I_m\}$ from the point cloud, we propose bilateral graph attention to extract discriminative instance embeddings by simultaneously considering the coordinate and

Algorithm 1 Instance Generation Algorithm

Input: node semantic scores $\boldsymbol{S} = \{\boldsymbol{s}_1, \ldots, \boldsymbol{s}_{|V|} \mid \boldsymbol{s}_i \in \mathbb{R}^N \text{ for } i = 1, \ldots, |V|\}$, N is the number of classes; semantic threshold θ; edge scores $\mathbf{A} = \{a_{u,v}\} \in \mathbb{R}^{|E| \times 1}$, $a_{u,v}$ indicates the score of edge which connects nodes u and v;
Output: instances $\mathbf{I} = \{I_1, \ldots, I_m\}$, m is the number of instances.

```
initialize an empty instance set I
for i = 1 to N do
    if i is valid class (excluding wall, floor) then
        initialize an array f (visited) of length |V| with all zeros
        for v = 1 to |V| do
            if f_v == 0 and s_v^i > θ then
                initialize an empty queue Q
                initialize an empty set I
                f_v = 1 ; Q.pushBack(v) ; add v to I
                while Q is not empty do
                    h = Q.popFront()
                    for each k ∈ {k | a_{h,k} < 0.5} do
                        if f_k == 0 and s_k^i > θ then
                            f_k = 1 ; Q.pushBack(k) ; add k to I
                add I to I
return I
```

feature spaces. Specifically, given the i-th instances I_i, we first compute instance centroid c_i by averaging the shifted superpoint coordinates within it. After that, we adopt the inverse distance weighted average of the corresponding superpoints to interpolate the embedding of the instance centroid, which is formulated as:

$$\boldsymbol{f}_i^{'}(c_i) = \frac{\sum_{j \in I_i} \psi_j(c_i) * \boldsymbol{f}_j}{\sum_{j \in I_i} \psi_j(c_i)}, \psi_j(c_i) = \frac{1}{\|x_j - c_i\|_2} \tag{7}$$

where I_i represents the superpoints within the i-th instance and x_j indicates the original coordinates of superpoints. Note that $*$ indicates the Hadamard product, which outputs the element-wise production of two vectors. After obtaining the coordinate c_i and embedding $\boldsymbol{f}_i^{'}$ for the i-th instance, we then link the superpoints of instance to instance centroid to construct the k-NN graph. To extract discriminative instance embedding, we develop bilateral graph attention to achieve this. The bilateral weight $\boldsymbol{w}_{i,j}$ between the superpoint $j \in I_i$ and the i-th instance is formulated as:

$$\boldsymbol{w}_{i,j} = \phi(\boldsymbol{f}_i^{'}, \boldsymbol{f}_j) * \varphi(c_i, x_j) \tag{8}$$

where $\phi(\cdot, \cdot) : \mathbb{R}^C \times \mathbb{R}^C \to \mathbb{R}^C$ and $\varphi(\cdot, \cdot) : \mathbb{R}^3 \times \mathbb{R}^3 \to \mathbb{R}^C$ are two mapping functions implemented by MLP networks. $\phi(\boldsymbol{f}_i^{'}, \boldsymbol{f}_j) = \mathrm{ReLU}(\boldsymbol{W}_\phi^\top(\boldsymbol{f}_i^{'} - \boldsymbol{f}_j))$ encodes the difference between the superpoint and the instance centroid in the feature space, while $\varphi(c_i, x_j) = \mathrm{ReLU}(\boldsymbol{W}_\varphi^\top(c_i - x_j))$ encodes the difference between the superpoint and the instance centroid in the coordinate space. Thus, $\boldsymbol{w}_{i,j} \in \mathbb{R}^C$ captures the channel-wise relationship between the superpoint and instance in the coordinate and feature spaces. We use the softmax function to obtain normalized weight $\hat{\boldsymbol{w}}_{i,j}$ across the instance I_i, which is written as:

$$\hat{\boldsymbol{w}}_{i,j} = \frac{\exp(\boldsymbol{w}_{i,j})}{\sum_{k \in I_i} \exp(\boldsymbol{w}_{i,k})} \tag{9}$$

Finally, we sum the weighted superpoint embeddings to obtain the instance embedding, which is given by:

$$\hat{\boldsymbol{f}}_i = \sum\nolimits_{j \in I_i} \boldsymbol{w}_{i,j} * \boldsymbol{f}_j \tag{10}$$

Subsequently, we adopt a classification head and a score head to predict the class and score of the instance I_i. In addition, we use a superpoint mask head to predict the superpoint score for masking the low confidence superpoints within the instance. According to these three heads, we use the cross-entropy as the classification loss $\mathcal{L}_{cls}$, the binary cross-entropy as the score loss $\mathcal{L}_{score}$, and the mean squared error as the mask loss $\mathcal{L}_{mask}$ to form the final loss $\mathcal{L}_{ins} = \mathcal{L}_{cls} + \mathcal{L}_{score} + \mathcal{L}_{mask}$ for training the superpoint graph cut network.

Table 1: Instance segmentation results on ScanNet v2 hidden test set. The reported results are from the ScanNet benchmark on 19/5/2022.

Method	AP	bathtub	bed	bookshe	cabinet	chair	counter	curtain	desk	door	other	picture	fridge	s.curtain	sink	sofa	table	toilet	window
3D-SIS [10]	16.1	40.7	15.5	6.8	4.3	34.6	0.1	13.4	0.5	8.8	10.6	3.7	13.5	32.1	2.8	33.9	11.6	46.6	9.3
GSPN [40]	15.8	35.6	17.3	11.3	14.0	35.9	1.2	2.3	3.9	13.4	12.3	0.8	8.9	14.9	11.7	22.1	12.8	56.3	9.4
3D-MPA [5]	35.5	45.7	48.4	29.9	27.7	59.1	4.7	33.2	21.2	21.7	27.8	19.3	41.3	41.0	19.5	57.4	35.2	84.9	21.3
PointGroup [14]	40.7	63.9	49.6	41.5	24.3	64.5	2.1	57.0	11.4	21.1	35.9	21.7	42.8	66.0	25.6	56.2	34.1	86.0	29.1
SSTNet [21]	50.6	73.8	54.9	49.7	31.6	69.3	17.8	37.7	19.8	33.0	46.3	57.6	51.5	85.7	49.4	63.7	45.7	94.3	29.0
HAIS [2]	45.7	70.4	56.1	45.7	36.4	67.3	4.6	54.7	19.4	30.8	42.6	28.8	45.4	71.1	26.2	56.3	43.4	88.9	34.4
SoftGroup [32]	50.4	66.7	57.9	37.2	38.1	69.4	7.2	67.7	30.3	38.7	53.1	31.9	58.2	75.4	31.8	64.3	49.2	90.7	38.8
GraphCut (ours)	**55.2**	100	61.1	43.8	39.2	71.4	13.9	59.8	32.7	38.9	51.0	59.8	42.7	75.4	46.3	76.1	58.8	90.3	32.9

3.3 Training and Inference

In the training process, the whole framework is optimized by a joint loss, which is defined as:

$$\mathcal{L} = \mathcal{L}_{sem} + \mathcal{L}_{geo} + \mathcal{L}_{ins} \tag{11}$$

where $\mathcal{L}_{sem}$ is the conventional cross-entropy loss for semantic scores, $\mathcal{L}_{geo}$ is the geometry-aware edge loss for edge scores, and $\mathcal{L}_{ins}$ is the instance loss for instance classification, score prediction, and superpoint mask prediction. In the inference process, our method directly outputs object instances after a forward pass of our network. Note that non-maximum suppression is not necessary for our method.

4 Experiments

4.1 Experimental Settings

Datesets. We conduct experiments on two benchmark datasets, ScanNet v2 [4] and S3DIS [1]. The ScanNet v2 dataset contains 1,613 3D scenes which are split into 1,201 training, 312 validation, and 100 test scenes, respectively. The results of instance segmentation are evaluated on 18 object categories. We report the results on validation and hidden test set. The ablation study is conducted on validation set. The S3DIS dataset has 272 3D scans in 6 different areas with 13 object classes. The instance segmentation is evaluated on all classes. We report Area 5 and 6-fold cross-validation results, respectively.

Evaluation metrics. Following the ScanNet v2 official protocol, we use the mean average precision as the evaluation metric for both ScanNet v2 and S3DIS. The mean average precision with IoU thresholds of 50% and 25% are denoted as AP_{50} and AP_{25} respectively. Also, AP denotes the mean average precision while IoU threshold from 50% to 95% with a step size of 5%. Additionally, following existing methods [36, 2, 32], we use mean coverage (mCov), mean weighted coverage (mWCov), mean precision (mPrec), and mean recall (mRec) for S3DIS evaluation.

Implementation details. Our model is trained on a single TITAN RTX GPU. We use the Adam optimizer with a base learning rate of 0.001 for the network training, which is scheduled by a cosine annealing. The voxel size is set to 0.02m. A graph-based segmentation method [6] and SSP+SPG [18, 17] are used to generate superpoints for ScanNet scene and S3DIS room, respectively. At training time, we limit the maximum number of points in a scene to 250k and crop the excess randomly. Due to the high point density of S3DIS, we randomly downsample its 1/4 points before cropping. At inference, the whole scene is fed into the network without downsampling and cropping.

4.2 Benchmarking Results

ScanNet v2. We compare our model with recent state-of-the-art methods on the unreleased test set of ScanNet v2. Table 1 reports the results on the leaderboard of the official testing server [1]. It can be observed that our method achieves the highest performance in terms of AP. The results on the leaderboard can demonstrate the effectiveness of our method for 3D instance segmentation.

[1] http://kaldir.vc.in.tum.de/scannet_benchmark/semantic_instance_3d.php?metric=ap

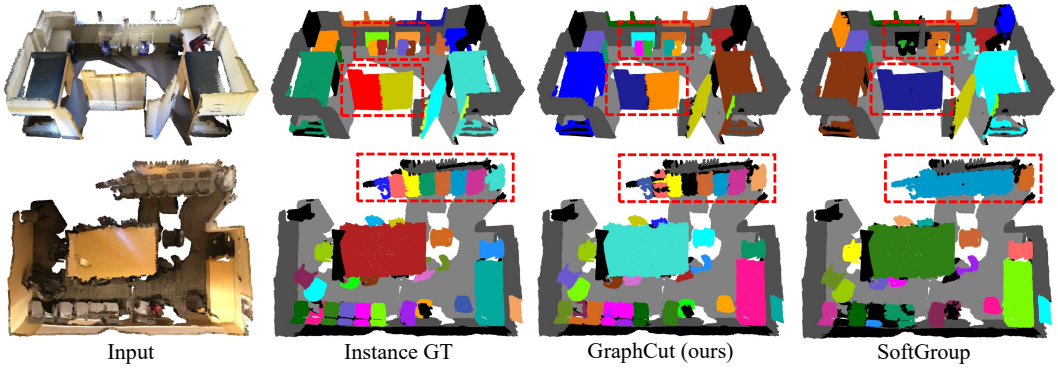

Figure 3: Visualization results of our GraphCut v.s. SoftGroup [32] on the ScanNet v2 validation set.

Moreover, we evaluate our method on the validation set of ScanNet v2. From the results listed in Table 2, one can observe that our model can achieve better results. Especially, our model brings 2.8% gains for the metric AP and 1.5% gains for the metric AP_{50} to the second-best methods. In addition, we provide the visualization results of our method v.s. SoftGroup [32] in Figure 3. We use the red rectangular boxes to show the differences between two methods. It can be found that our method can generate good instances with clear boundaries for objects clustered together, such as chairs. SoftGroup relies on point grouping by using offset-shifted point coordinates, which cannot make full use of local geometric information of point clouds. Since our method can fully utilize the local geometric information of point clouds by constructing an edge score prediction network and a superpoint graph cut network, our method achieves better results than SoftGroup on these clustered objects. Please refer to the supplementary material for more visualization results and runtime analysis.

Table 2: Instance segmentation results on ScanNet v2 validation set.

Method	AP	AP_{50}	AP_{25}
GSPN [40]	19.3	37.8	53.4
3D-MPA [5]	35.3	51.9	72.4
PointGroup [14]	34.8	56.9	71.3
SSTNet [21]	49.4	64.3	74.0
HAIS [2]	43.5	64.1	75.6
SoftGroup [32]	46.0	67.6	78.9
GraphCut (ours)	**52.2**	**69.1**	**79.3**

S3DIS. In Table 3, we list the results on Area 5 and 6-fold cross validation of S3DIS. Regarding to the evaluation on Area 5, our method can outperform all compared methods. It is worth noting that our model improves the SoftGroup by 2.5% in terms of AP. For the 6-fold cross validation of S3DIS, our method is superior to the state-of-the-art methods on most of the metrics.

4.3 Ablation Studies and Analysis

Different k in edge score prediction network. In our edge score prediction network, we learn the similarity from the local k-NN graphs of two adjacent nodes to identify whether they belong to the same instance. Here, we study the impact of different k on the instance segmentation performance. We select $k \in \{0, 2, 4, 8, 16\}$. Notably, k=0 means that we only concatenate two adjacent superpoint features as edge embedding. The results of AP, AP_{50}, and AP_{25} are 52.0%, 68.8%, 78.7% (k=2), **52.2**%, **69.1**%, **79.3**% (k=4), 51.4%, 68.1%, 79.1% (k=8), and 50.8%, 67.7%, 79.1% (k=16), respectively. Since k=4 achieves the best results, we set k=4 in our experiment.

Effectiveness of edge feature embedding. To verify the effectiveness of our edge feature embedding, we consider three cases: (1) Only with edge embedding in coordinate space (dubbed as "Coordinate"), (2) Only with edge embedding in feature space (dubbed as "Feature"), (3) Only with embeddings of adjacent nodes as edge embedding. From the instance segmentation results on ScanNet v2 validation set listed in Table 4, the best performance is achieved with the combination of the edge embeddings in both the coordinate and feature spaces. In the edge feature embedding, employing both geometry and feature embeddings of point clouds can improve performance of the instance segmentation of point clouds.

Ablation study on geometry-aware edge loss. Here, we conduct the experiments on the ScanNet v2 validation set to verify the effectiveness of the propose geometry-aware edge loss. Specifically, we also consider three ablations: (1) Only with area constraint in the coordinate space ($i.e.$, "$\mathcal{L}_{area}$"), (2)

Table 3: Instance segmentation results on the S3DIS dataset.

Setting	Methodc	AP	AP_{50}	mCov	mWCov	$mPrec_{50}$	$mRec_{50}$
Area 5	SGPN [35]	-	-	32.7	35.5	36.0	28.7
	ASIS [36]	-	-	44.6	47.8	55.3	42.4
	PointGroup [14]	-	57.8	-	-	61.9	62.1
	SSTNet [21]	42.7	59.3	-	-	65.5	64.2
	HAIS [2]	-	-	64.3	66.0	71.1	65.0
	SoftGroup [32]	51.6	66.1	66.1	68.0	73.6	66.6
	GraphCut (ours)	**54.1**	**66.4**	**67.5**	**68.7**	**74.7**	**67.8**
6-fold	SGPN [35]	-	-	37.9	40.8	38.2	31.2
	PartNet [24]	-	-	-	-	56.4	43.4
	ASIS [36]	-	-	51.2	55.1	63.6	47.5
	3D-BoNet [39]	-	-	-	-	65.6	47.7
	OccuSeg [8]	-	-	-	-	72.8	60.3
	GICN [22]	-	-	-	-	68.5	50.8
	PointGroup [14]	-	64.0	-	-	69.6	69.2
	SSTNet [21]	54.1	67.8	-	-	73.5	73.4
	HAIS [2]	-	-	67.0	70.4	73.2	69.4
	SoftGroup [32]	54.4	**68.9**	69.3	71.7	**75.3**	69.8
	GraphCut (ours)	**56.3**	68.2	**72.8**	**75.0**	74.4	**73.7**

Table 4: Ablation study on ScanNet v2 validation set for edge feature embedding.

Coordinate	Feature	AP	AP_{50}	AP_{25}
		50.1	66.6	77.4
✓		50.9	68.0	78.3
	✓	51.5	68.8	78.5
✓	✓	**52.2**	**69.1**	**79.3**

Table 5: Ablation study on ScanNet v2 validation set for geometry-aware edge loss.

$\mathcal{L}_{area}$	$\mathcal{L}_{feat}$	AP	AP_{50}	AP_{25}
		47.8	65.1	75.3
✓		51.7	68.6	78.8
	✓	51.2	67.8	77.8
✓	✓	**52.2**	**69.1**	**79.3**

Only with instance constraint in the feature space ($i.e.$, "$\mathcal{L}_{feat}$"), (3) Only with binary cross-entropy loss, $i.e.$, $\mathcal{L}_{edge}$. The results are listed in Table 5. It can be observed that the geometry constraints bring substantial gains to our method. By using the area constraint, it is easier to draw the nodes of the instance toward the instance center, making the boundary between different instances clearer. Please refer to the supplementary material for visualization results.

Effectiveness of bilateral graph attention. In order to validate the effectiveness of the proposed bilateral graph attention, we replace the bilateral graph attention with a simple MLP network followed by max-pooling and conduct experiments on the ScanNet v2 validation set. The results of AP, AP_{50}, and AP_{25} are 49.9%, 66.8%, 77.3% (MLP network), and **52.2**%, **69.1**%, **79.3**% (our bilateral graph attention), respectively. Without our designed bilateral graph attention, the performance drops a lot. This is because the bilateral graph attention can adaptively aggregate the information of superpoints in the same instance, which is more reasonable than the simple max-pooling operation for instance embedding.

5 Conclusion

In this paper, we proposed a learning-based superpoint graph cut method for 3D instance segmentation, which prunes the irrelevant edges off the superpoint graph for instance generation. Specifically, we proposed an edge score prediction network with cross-graph attention in the coordinate and feature spaces to capture local geometric information of two adjacent nodes and predict the edge scores. A geometry-aware edge loss was proposed to train the edge score prediction network, which encourages two adjacent nodes in the same instance to be close to the instance center in both coordinate and feature spaces. Based on the learned edge scores, a superpoint graph cut network was developed to cut irrelevant edges for instance generation. For the generated instances, we further adopted bilateral graph attention to predict semantic classes and scores of instances. Extensive experiments on ScanNet v2 and S3DIS benchmarks show that our method achieves new state-of-the-art performance.

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 *CVPR*, 2016.

[2] Shaoyu Chen, Jiemin Fang, Qian Zhang, Wenyu Liu, and Xinggang Wang. Hierarchical aggregation for 3D instance segmentation. In *ICCV*, 2021.

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

[4] 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.

[5] Francis Engelmann, Martin Bokeloh, Alireza Fathi, Bastian Leibe, and Matthias Nießner. 3D-MPA: Multi proposal aggregation for 3D semantic instance segmentation. In *CVPR*, 2020.

[6] Pedro F Felzenszwalb and Daniel P Huttenlocher. Efficient graph-based image segmentation. *International journal of computer vision*, 2004.

[7] Benjamin Graham, Martin Engelcke, and Laurens Van Der Maaten. 3D semantic segmentation with submanifold sparse convolutional networks. In *CVPR*, 2018.

[8] Lei Han, Tian Zheng, Lan Xu, and Lu Fang. OccuSeg: Occupancy-aware 3D instance segmentation. In *CVPR*, 2020.

[9] Tong He, Chunhua Shen, and Anton van den Hengel. DyCo3D: Robust instance segmentation of 3D point clouds through dynamic convolution. In *CVPR*, 2021.

[10] Ji Hou, Angela Dai, and Matthias Nießner. 3D-SIS: 3D semantic instance segmentation of RGB-D scans. In *CVPR*, 2019.

[11] Qingyong Hu, Bo Yang, Linhai Xie, Stefano Rosa, Yulan Guo, Zhihua Wang, Niki Trigoni, and Andrew Markham. RandLA-Net: Efficient semantic segmentation of large-scale point clouds. In *CVPR*, 2020.

[12] Zeyu Hu, Mingmin Zhen, Xuyang Bai, Hongbo Fu, and Chiew-lan Tai. JSEnet: Joint semantic segmentation and edge detection network for 3D point clouds. In *ECCV*, 2020.

[13] Maximilian Jaritz, Jiayuan Gu, and Hao Su. Multi-view pointnet for 3D scene understanding. In *ICCVW*, 2019.

[14] Li Jiang, Hengshuang Zhao, Shaoshuai Shi, Shu Liu, Chi-Wing Fu, and Jiaya Jia. PointGroup: Dual-set point grouping for 3D instance segmentation. In *CVPR*, 2020.

[15] Abhijit Kundu, Xiaoqi Yin, Alireza Fathi, David Ross, Brian Brewington, Thomas Funkhouser, and Caroline Pantofaru. Virtual multi-view fusion for 3D semantic segmentation. In *ECCV*, 2020.

[16] Jean Lahoud, Bernard Ghanem, Marc Pollefeys, and Martin R Oswald. 3D instance segmentation via multi-task metric learning. In *ICCV*, 2019.

[17] Loic Landrieu and Mohamed Boussaha. Point cloud oversegmentation with graph-structured deep metric learning. In *CVPR*, 2019.

[18] Loic Landrieu and Martin Simonovsky. Large-scale point cloud semantic segmentation with superpoint graphs. In *CVPR*, 2018.

[19] Felix Järemo Lawin, Martin Danelljan, Patrik Tosteberg, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. Deep projective 3D semantic segmentation. In *CAIP*. Springer, 2017.

[20] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. PointCNN: Convolution on
$mathcalX$-transformed points. In *NIPS*, 2018.

[21] Zhihao Liang, Zhihao Li, Songcen Xu, Mingkui Tan, and Kui Jia. Instance segmentation in 3D scenes using semantic superpoint tree networks. In *ICCV*, 2021.

[22] Shih-Hung Liu, Shang-Yi Yu, Shao-Chi Wu, Hwann-Tzong Chen, and Tyng-Luh Liu. Learning gaussian instance segmentation in point clouds. *arXiv preprint arXiv:2007.09860*, 2020.

[23] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015.

[24] Kaichun Mo, Shilin Zhu, Angel X Chang, Li Yi, Subarna Tripathi, Leonidas J Guibas, and Hao Su. PartNet: A large-scale benchmark for fine-grained and hierarchical part-level 3d object understanding. In *CVPR*, 2019.

[25] Quang-Hieu Pham, Thanh Nguyen, Binh-Son Hua, Gemma Roig, and Sai-Kit Yeung. JSIS3D: Joint semantic-instance segmentation of 3D point clouds with multi-task pointwise networks and multi-value conditional random fields. In *CVPR*, 2019.

[26] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. PointNet: Deep learning on point sets for 3D classification and segmentation. In *CVPR*, 2017.

[27] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. PointNet++: Deep hierarchical feature learning on point sets in a metric space. In *NeurIPS*, 2017.

[28] Martin Simonovsky and Nikos Komodakis. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *CVPR*, 2017.

[29] Liyao Tang, Yibing Zhan, Zhe Chen, Baosheng Yu, and Dacheng Tao. Contrastive boundary learning for point cloud segmentation. *ArXiv*, 2022.

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

[31] 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.

[32] Thang Vu, Kookhoi Kim, Tung M. Luu, Xuan Thanh Nguyen, and Chang D. Yoo. SoftGroup for 3D instance segmentation on 3D point clouds. In *CVPR*, 2022.

[33] Lei Wang, Yuchun Huang, Yaolin Hou, Shenman Zhang, and Jie Shan. Graph attention convolution for point cloud semantic segmentation. In *CVPR*, 2019.

[34] Peng-Shuai Wang, Yang Liu, Yu-Xiao Guo, Chun-Yu Sun, and Xin Tong. O-CNN: Octree-based convolutional neural networks for 3D shape analysis. *TOG*, 2017.

[35] Weiyue Wang, Ronald Yu, Qiangui Huang, and Ulrich Neumann. SGPN: Similarity group proposal network for 3D point cloud instance segmentation. In *CVPR*, 2018.

[36] Xinlong Wang, Shu Liu, Xiaoyong Shen, Chunhua Shen, and Jiaya Jia. Associatively segmenting instances and semantics in point clouds. In *CVPR*, 2019.

[37] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *TOG*, 2019.

[38] Wenxuan Wu, Zhongang Qi, and Li Fuxin. PointConv: Deep convolutional networks on 3D point clouds. In *CVPR*, 2019.

[39] Bo Yang, Jianan Wang, Ronald Clark, Qingyong Hu, Sen Wang, Andrew Markham, and Niki Trigoni. Learning object bounding boxes for 3D instance segmentation on point clouds. In *NeurIPS*, 2019.

[40] Li Yi, Wang Zhao, He Wang, Minhyuk Sung, and Leonidas J Guibas. GSPN: Generative shape proposal network for 3d instance segmentation in point cloud. In *CVPR*, 2019.

[41] Hengshuang Zhao, Li Jiang, Chi-Wing Fu, and Jiaya Jia. PointWeb: Enhancing local neighborhood features for point cloud processing. In *CVPR*, 2019.

[42] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. Point Transformer. In *ICCV*, 2021.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] Please refer to the supplementary material.
   (c) Did you discuss any potential negative societal impacts of your work? [No]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] Please refer to the supplementary material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] Please refer to the experimental part of the main paper.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] Please refer to the experimental part of the main paper.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [No]
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [No]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [No]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]