SegGraph: Leveraging Graphs of SAM Segments for Few-Shot 3D Part Segmentation

Yueyang Hu[1], Haiyong Jiang[1]*, Haoxuan Song[1], Jun Xiao[1]*, Hao Pan[2]
[1]School of Artificial Intelligence, University of Chinese Academy of Sciences
[2]School of Software, Tsinghua University
{huyueyang23, songhaoxuan24}@mails.ucas.ac.cn
{haiyong.jiang, xiaojun}@ucas.ac.cn
haopan@tsinghua.edu.cn

Abstract

This work presents a novel framework for few-shot 3D part segmentation. Recent advances have demonstrated the significant potential of 2D foundation models for low-shot 3D part segmentation. However, it is still an open problem that how to effectively aggregate 2D knowledge from foundation models to 3D. Existing methods either ignore geometric structures for 3D feature learning or neglects the high-quality grouping clues from SAM, leading to under-segmentation and inconsistent part labels. We devise a novel SAM segment graph-based propagation method, named SegGraph, to explicitly learn geometric features encoded within SAM's segmentation masks. Our method encodes geometric features by modeling mutual overlap and adjacency between segments while preserving intra-segment semantic consistency. We construct a segment graph, conceptually similar to an atlas, where nodes represent segments and edges capture their spatial relationships (overlap/adjacency). Each node adaptively modulates 2D foundation model features, which are then propagated via a graph neural network to learn global geometric structures. To enforce intra-segment semantic consistency, we map segment features to 3D points with a novel view-direction-weighted fusion attenuating contributions from low-quality segments. Extensive experiments on PartNet-E demonstrate that our method outperforms all competing baselines by at least 6.9% mIoU. Further analysis reveals that SegGraph achieves particularly strong performance on small components and part boundaries, demonstrating its superior geometric understanding. The code is available at: https://github.com/YueyangHu2000/SegGraph.

1 Introduction

3D part segmentation is a fundamental task in computer vision and graphics with broad implications for shape analysis [1], 3D modeling [2, 3], and robotic manipulations [4, 5]. For instance, a user can edit a 3D shape based on part components and embodied agents need to interact with diverse object parts for different manipulations, such as identifying a drawer's handle or a bottle's neck. Many of these real-world applications involve novel shapes, and it is essential to achieve high-quality 3D part segmentation with only a few number of annotations.

2D foundation models (FMs) with powerful generalization capacities have brought new possibilities to part segmentation. Nevertheless, the modality gap between 2D images and 3D geometric shapes poses a fundamental challenge for their direct deployment in 3D domains. A prevalent strategy [6, 7]

*Joint Corresponding Authors. Haiyong is the Project Lead.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

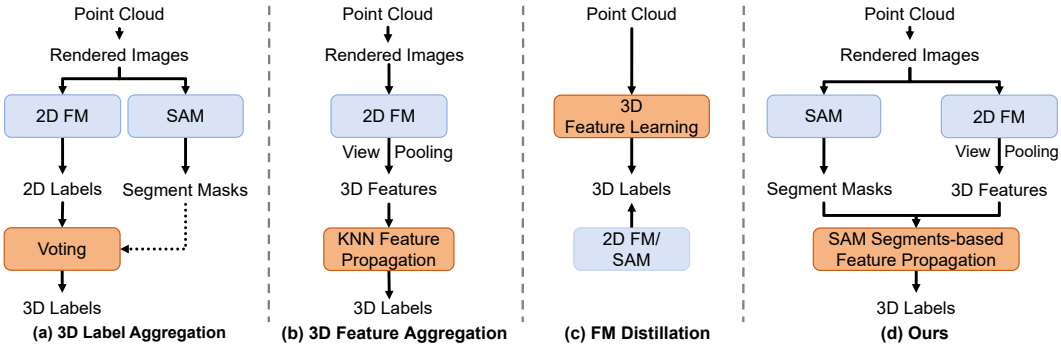

Figure 1: Architecture comparisons of different methods. (a) 3D label aggregation-based methods [6, 18, 11, 12, 7], (b) 3D feature aggregation-based methods [16, 17], (c) distillation-based methods [13, 14, 15] to distill 2D knowledge from foundation models, and (d) ours with an emphasis on SAM segments and SAM-segments-based feature propagation.

is to render a 3D shape into multi-view images, then leverage 2D foundation models such as GLIP [8] or Diffusion [9] for image segmentation, and finally aggregate 2D labels to the 3D domain with voting (see Fig. 1a). During the process, segmentation foundation models, e.g., SAM [10], can further enhance the semantic awareness for finer part segmentation [11, 12, 7]. However, simply aggregating 2D labels to 3D domains overlooks the geometric structures of 3D inputs, resulting in under-segmentation of shape parts and inconsistency among neighbor points.

3D distillation from foundation models [13, 14, 15] and 3D aggregation-based feature learning [16, 17] take a step forward and learn 3D features with geometry awareness (see Fig. 1b,c). Distillation-based methods typically learn geometric features by transferring knowledge from foundation models but rely on large-scale 3D shapes for good performance. 3D feature aggregation-based methods aggregate multi-view image features from foundation models and propagate fused features via KNN-based point structures. However, simple KNN propagation struggles to encode complex geometric patterns and does not take into account grouping clues from SAM. Moreover, these two schemes involve downsampling 3D points (about 100 times for PartNet-E) or using voxel-based 3D feature encoding, leads to ambiguous features for boundary points and small components.

We propose a novel framework, called **SegGraph**, for part segmentation of 3D point clouds. We develop the method based on three key insights. First, SAM's high-quality segmentation provides consistent intra-segment grouping cues for 3D points, effectively reducing misclassification errors near boundaries. These segments can efficiently encode 3D geometric structures with less than 1000 SAM segments for each shape, avoiding input downsampling. Second, spatial relationships among segments offers two critical priors for part segmentation: (1) overlapping segments across different views typically share the same part label, (2) adjacent segments naturally preserve part boundary information. Third, the quality of SAM segments correlates with the viewing direction of an image, exhibiting under-segmentation of small parts in challenging views.

To implement these three observations, we introduce a segment graph propagation-based part learning framework, as illustrated in Fig. 2. First, we encode 3D point features through multi-view feature pooling from 2D foundation model outputs. Simultaneously, we generate view-consistent 3D segments by aggregating SAM-based 2D segmentations across all views. We then model both the spatial relationships (overlapping/adjacent) between segments and the constituent relationships between points and segments. The segment graph takes 3D segments as nodes and represent spatial relationships as nodes. Segment features encoded from its constituent points' aggregated features are propagated via a graph neural network for geometric structure-aware feature encoding. Thereafter, we enhance 3D point features with segment features with a view quality-aware feature unpooling, where points with a normal off a view direction for obtaining the SAM segment are assigned with lower importance.

Extensive experiments demonstrate that SegGraph can significantly outperform competing methods by a large margin of 6.9% mIoU on the PartNet-E dataset [6]. SegGraph is quite extensible and can leverage different kinds of foundation models with an improvement of at least 4% on the PartNet-E dataset.

2 Related Work

Supervised 3D Segmentation. Supervised 3D segmentation has been widely studied in recent years, facilitated by the availability of annotated datasets [19, 20, 21, 22, 23, 24]. Prior works have proposed a variety of architectures tailored to 3D data, including point-based models (e.g., PointNet [25] and its variants [26, 25, 27, 28]), volumetric CNNs [29, 30, 31], graph convolutional networks [32, 33, 34] and Transformer-based methods [35, 36, 37]. These methods are typically trained with large amounts of manually annotated 3D data. A particularly challenging sub-task is 3D part segmentation that predicts semantic part labels for a shape. PartNet [23] and DeepGCNs [38] achieves part segmentation via a graph convolutional network (GCN)-based model to learn local geometric features. CSN [39] improves part segmentation by introducing a cross-shape attention mechanism that captures interactions among different shapes in 3D point clouds. However, most of these methods rely on dense part annotations for good performance. Some works explore multi-prototype networks with attention [40] and learning part-specific probability spaces via template morphing and density estimation [41] for few-shot part segmentation.

3D Part Segmentation with Foundation Models. Recent advancements of foundation models [42, 10, 43] have shown remarkable capabilities across diverse tasks. A simple yet effective approach to exploit 2D foundation models for 3D tasks is to render multi-view projections of the 3D data, feed the rendered images into a 2D foundation model, and subsequently aggregate the model outputs back to 3D via voting. Pioneering efforts like PartSLIP [6] and SATR [18] follow this scheme and utilize GLIP [8] for open-vocabulary segmentation. PartSLIP++ [11] and PartSTAD [12] further utilize detections of GLIP as the visual prompts for SAM [10] for better part masks, while 3-by-2 [7] constructs multiview labels based on DIFT [9] and aggregates 2D labels to 3D based on multiview masks from SAM [10]. However, aggregating 2D segmentation into 3D entirely overlooks the inherent geometric structure of the 3D data and consistent point segmentations among neighbors, leading to under-segmentation and noisy parts. COPS [44] and [17] address this problem by directly aggregating 2D features from foundation models to 3D and propagating 3D features via superpoint-based pooling and KNN-based feature smoothing for superpoints. Despite better results, feature aggregation with simple KNN smoothing disables complex 3D context learning and geometric superpoints are often inconsistent with part grouping. Another line of work [13, 14, 15] leverages a distillation-based framework. For example, PartDistill [13] presents a bidirectional distillation between predicted 2D labels and 3D labels from a native 3D network. SAMPart3D [14] and PartField [15] construct training pairs for contrastive learning based on SAM segmentation. These distillation methods require a large amount of 3D data to learn generalizable 3D features. This work combines the benefits of both SAM segmentations and light-weight 3D feature aggregation with shape structural context encoding for easy adaption to few-shot novel shapes.

3 Methodology

Our goal is to predict a part label for each point of a given point cloud with only low-shot training samples for novel shapes. We approach this problem with a graph-based feature aggregation network as illustrated in Fig. 2. The method encodes point-wise features using an off-the-shelf foundation model (Sec. 3.1) and generates a list of segment masks based on SAM segmentation (Sec. 3.2). Then we construct a SAM-segment-based graph to encode different kinds of relations among different segments and 3D points (Sec. 3.3).

3.1 Feature Encoding for 3D Points

To harness the power of image foundation models, such as DINOv2 [43] and CLIP [42], our approach renders a point cloud input into multi-view images from M predefined camera views. We denote rendered images as $\{\boldsymbol{I}_m \in \mathbb{R}^{H\times W\times 3}\}_{m=1}^{M}$. The rendering process follows PartSLIP [6], incorporating occlusion culling [45] to eliminate obscured points. Next, we feed each view image to an image foundation model (DINOv2 in our implementation) to extract image features. However, the resolution of output image feature maps ($\frac{W}{14} * \frac{H}{14}$) is usually lower than the input image. Therefore, image feature maps are upsampled to the original image resolution using bicubic interpolation and are mapped from 768 channels to 96 channels with a linear layer. We can obtain the 3D feature $\boldsymbol{F}^p$ for each point by taking the average of its projected image features at views where the point is not occluded.

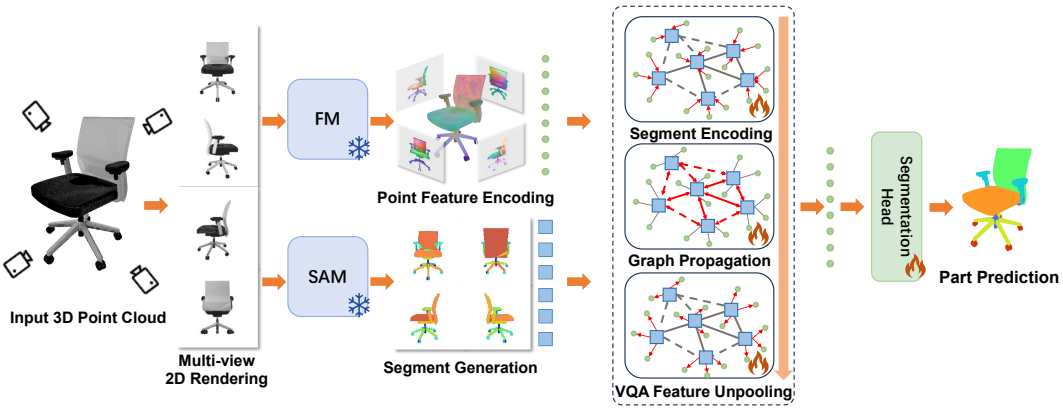

Figure 2: The overview of the pipeline. Given a 3D point cloud, we render it into multi-view images. We extract individual point features by pooling foundation model features (DINOv2) of rendering views. At the same time, multi-view images are fed to SAM for segment generation. Point features and segments are sent to a segment graph to learn geometric features and segment relations for part prediction.

3.2 Segment Generation

In light of the high-quality segmentation masks of SAM [10], we further leverage these masks to enhance the quality of part segmentation. While the semantic granularity of SAM's predictions may vary across views, these masks nevertheless provide reliable grouping clues (see the colored segments in Fig. 3). Our method processes each rendered image using SAM to produce an initial set of segmentation masks. Following the approach in [7], we subsequently decompose overlapping masks under a single view into non-overlapping, over-segmented regions. This decomposition effectively resolves semantic confusions for 3D points whose projections fall within multiple overlapping masks of varying granularity and semantics. Afterwards, all 3D points that project within the same 2D segmentation mask form a corresponding segment. The set of segments is denoted as $\mathcal{S}$.

3.3 Segment-based Feature Refinement via Graph Propagation

Given 3D point features $\boldsymbol{F}^p$ and segments $\mathcal{S}$, we further exploit the grouping cues from individual segments, along with their underlying geometric structures, to enhance 3D features. The key motivations are as follows: 1) points within the same segment typically share the same part label; 2) overlapping segments constructed from different views likely correspond to the same part label; 3) adjacent segments from the same or different views encode the holistic 3D structure and delineate possible part boundaries. To compile these ideas into a framework, we first encode segment features, then propagate them through overlapping segments and adjacent segments, and finally map segment features to individual point for part segmentation.

Segment Encoding. A straightforward way to compute segment features is to average the features of their member points. However, simple pooling fails to account for the varying semantic importance of individual points based on their geometric attributes, including their spatial distribution relative to the segment centroid and local surface normal. Inspired by [46], we instead learn adaptive point contributions with a geometric feature encoding module. For each point $\boldsymbol{p}_j$ within a segment, the module calculates its normalized relative position $\boldsymbol{p}_j^r$ w.r.t. the centroid $\boldsymbol{c}_i$ of a segment $\mathcal{S}_i$ as follows:

$$\boldsymbol{p}_j^r = \frac{\boldsymbol{p}_j - \boldsymbol{c}_i}{\max\limits_{k \in \mathcal{S}_i}(\boldsymbol{p}_k) - \min\limits_{k \in \mathcal{S}_i}(\boldsymbol{p}_k)}. \tag{1}$$

Then we compute the local geometric feature $\boldsymbol{F}_j^l \in \mathbb{R}^C$ with a multi-layer perceptron that takes as input the concatenation of point's normal and its normalized position $\boldsymbol{p}_j^r$. Afterwards, we obtain the segment feature $\boldsymbol{F}^s \in \mathbb{R}^C$ by applying max pooling to the local geometric features $\boldsymbol{F}_j^l$ of all points within the segment. The segment feature $\boldsymbol{F}^s$ is further refined through an attentively weighted aggregation of point features $\boldsymbol{F}_j^p$, where attentive weights are computed with an additive attention [47] on $\boldsymbol{F}_j^l$ and $\boldsymbol{F}^s$.

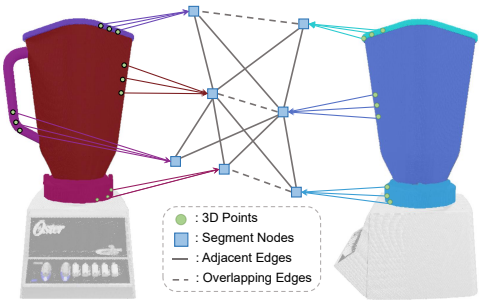

Figure 3: An illustration of graph construction. Features of 3D points (green dots) are first aggregated to segment nodes (blue boxes).

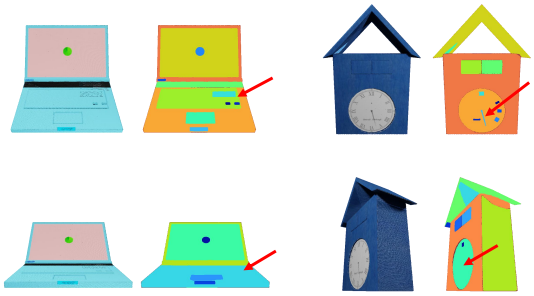

Figure 4: Two examples for the impacts of the rendering views on the quality of SAM segments. For each group, left: input, right: SAM segments.

Building a SAM Segment-based Graph. Denote the graph as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with nodes $\mathcal{V}$ representing 3D segments and edges $\mathcal{E}$ encoding segment relationships. The segment relationships include 3D adjacent relations $\mathcal{E}_a$ and overlapping relations $\mathcal{E}_o$. An overlapping relationship exists between two segment nodes in different views if their corresponding 3D points have significant overlaps (mIoU larger than 10%). An adjacent relation between two segments is constructed if their covered points have rare overlaps but are close enough (with a minimal distance between points of two segments less than 0.01 units in the normalized space). Note that overlapping relations and adjacent relations are mutually exclusive. We show an example of a segment graph in Fig. 3. In this example, overlapping segments (connected by dashed lines) share identical part labels, while adjacent segments (connected by solid lines) demarcate part boundaries. This graph structure is analogous to an atlas in differentiable manifold mapping the 3D shape consistently. Specifically, each segment corresponds to a chart, and transitions between charts are edges derived from the adjacency and overlap relationships among segments; the collection of all charts and their connecting edges constitutes an atlas. In this way, our segment graph provides a structure for analyzing the semantics of object shapes with geometric consistency.

Feature Propagation via the Segment Graph. After building the segment graph, we propagate segment features through the edges using GATv2 [48]. We feed the segment feature $\boldsymbol{F}^s$ to a three GATv2 network layer. Given the distinct nature of overlapping and adjacent relationships, we implement two GATv2 networks with separate weight parameters to model these relationships, respectively. Then we concatenate the output features of two GATv2 at each layer and feed them to a MLP as the node features for the next layer.

Viewing Quality-Aware Feature Unpooling. Subsequently, we propagate segment features to member 3D points to enforce label consistency within segments. This propagation requires careful feature fusion since points usually belong to multiple segments with varying reliability. A naive averaging approach is suboptimal as segments from some challenging views are usually of low quality. As shown in Fig. 4, under certain challenging viewpoints, parts such as laptop keyboards and clock components are not well segmented by SAM. In contrast, under more favorable viewpoints, SAM is able to segment these parts accurately. As the quality of segment masks often depends on the viewing angle, we devise a viewing quality-aware unpooling module for this purpose. This module estimates the quality based on the point normal $\boldsymbol{n}_j$ and the camera view direction:

$$\boldsymbol{w}_{ij}^v = |\boldsymbol{n}_j \cdot \frac{\boldsymbol{p}_j - \boldsymbol{c}_i}{\|\boldsymbol{p}_j - \boldsymbol{c}_i\|}|, \tag{2}$$

where $\boldsymbol{c}_i$ is the camera position for extracting segment i and $\boldsymbol{n}_j$ is the normal direction of point j. The quality score $\boldsymbol{w}_{ij}^v$ is further refined with a learnable MLP with a softmax to reweighing the quality scores. The final 3D point features combine the original point features and a weighted fusion of segment features, and then the features are fed to the segmentation head with a two-layer MLP to produce the logits $\hat{\boldsymbol{Y}}$:

$$\boldsymbol{F}_i^{p\prime} = \boldsymbol{F}_i^p + \sum_{i \in \mathcal{S}_j} \boldsymbol{w}_{ij}^v \cdot \boldsymbol{F}_j^S, \qquad \hat{\boldsymbol{Y}}_i = \text{MLP}(\boldsymbol{F}_i^{p\prime}). \tag{3}$$

For the network training, we use cross-entropy as the objective to train the network with few-shot examples.

4 Experimental Results

We evaluate the effectiveness of our method on two benchmark datasets: PartNet-Ensemble (PartNet-E) [6] and ShapeNetPart [49]. PartNet-E, proposed by PartSLIP [6], consists of 1,906 shapes across 45 categories with RGB colors. The dataset splits follow that of [12]. ShapeNetPart contains 31,963 shapes spanning 16 categories but lacks color information. We adopt the mIoU as the evaluation metric. We evaluate our method on the official testing set using a few-shot setting, where 8-shot examples are sampled from the training set for all experiments. We trained our method three times and reported the average mIoU and the standard deviations to reflect the influence of training variances using different seeds.

4.1 Comparisons on Part Segmentation

Baselines. On the PartNet-E dataset, we compare SegGraph with fully supervised approaches [37, 26, 50] as well as state-of-the-art few-shot methods based on 2D foundation models including label aggregation-based methods [6, 11, 12, 7], and distillation-based methods [13, 15]. The fully supervised methods were trained on 28K objects from 17 overlapping categories in PartNet [23] that overlap with PartNet-E, along with the few-shot training shapes. The few-shot baselines were trained solely on the few-shot set. For the zero-shot method PartField, which successfully distills part-level 3D representations from SAM, we fine-tune an MLP on its extracted features to adapt the model to the few-shot setting. Among the above-mentioned methods, [6, 11, 12, 13] provide testing codes and weight checkpoints for comparisons, while [7] does not have available sources. For methods without sources and that we fail to reproduce, we use their reported results and mark them with *. However, as few-shot methods, e.g., [6, 12, 13], require prompt tuning and do not have training code, we instead compare with PointCLIP [51] and PointCLIPv2 [52] with support of few-shot learning on the ShapeNetPart. Specifically, we replace text embedding for classification in PointCLIP and PointCLIPv2 with a trainable MLP classifier for few-shot example training.

Results on PartNet-E. Tab. 1 presents the comparison results on PartNet-E (see the supplementary for per-category results). All foundation model-based methods achieve superior performance to supervised ones (the top three rows), even with only a few-shot training. This suggests the strong capability of foundation models on downstream tasks. We also observe that label aggregation-based methods [6, 11, 12, 7], performs worse than feature aggregation-based methods [7] and distillation-based methods [13, 15], because label aggregation-based methods rely solely on 2D images to obtain labels, completely ignoring the geometric information crucial for 3D part segmentation. Among all methods, our approach outperforms the current state-of-the-art few-shot part segmentation method, PartDistill, by a significant margin with more than 6% gains in mIoU across all 45 categories, achieving the highest performance in 32 of them. Notably, we observe over 10% absolute improvements on categories such as Door and Lamp, which, based on our observations, can be attributed to our method's superior ability to segment relatively small parts such as door handles and light bulbs.

Despite the overall improvement being substantial, different training runs of our method produce varying performance because of random initial seeds. When training the model three times with the same setting, the average standard deviation across the 45 categories is about 1.09%.

Since most of these foundation model-based methods use GLIP, to ensure a fairer comparison, in addition to employing DINOv2, we also transformed GLIP into a feature extractor for comparison, as detailed in Sec. 4.2. The results show that when using GLIP as the feature extractor, our method still achieves performance that is only slightly lower than that obtained with DINOv2, as shown in Tab. 1.

Further inspection of segmentation results with large variances reveals that semantically ambiguous 3D segments are prone to errors in the few-shot setting, leading to the variance in different training runs with random initial seeds.

We further inspect the improvements on different sized parts. Benefiting from the over-segmented segments and the segment graph, our method demonstrates significantly improved performance (more than $+20\%$) in segmenting small parts. As illustrated in Tab. 2, we select six representative categories that include small parts such as buttons (in Coffee Machine, Remote), knobs (in Coffee Machine),

Table 1: Few-shot part segmentation results on PartNet-E. The mIoU metric is measured in percentage. The number in the top row bracket counts the number of shape categories. SD is for Stable Diffusion [53]. The methods in the top three rows are trained with both few-shot shapes and PartNet shapes that share overlapping categories (17) with PartNet-E.

Methods	FM	Overlapping Categories (17)						None-Overlapping Categories (28)						(45)
		Bottle	Chair	Door	Knife	Lamp	Overall	Camera	Dispe.	Kettle	Oven	Suitca.	Overall	Overall
PointNet++ [26]	-	48.8	84.7	45.7	35.4	68.0	55.3	6.5	12.1	20.9	34.4	40.7	25.0	36.5
SoftGroup [50]	-	41.4	88.3	53.1	31.3	82.2	50.4	23.6	18.9	57.4	13.7	18.3	31.3	38.5
PointNext [37]	-	68.4	91.8	43.8	58.7	64.9	59.1	33.2	26.0	45.1	37.8	13.6	45.5	50.6
PartSLIP [6]	GLIP	83.4	85.3	40.8	65.2	66.1	56.3	58.3	73.8	77.0	73.5	70.4	61.3	59.4
PartSLIP++ [11]	GLIP & SAM	85.5	85.3	45.1	64.3	68.0	59.7	63.2	72.0	85.6	70.3	70.0	63.5	62.1
PartSTAD [12]	GLIP & SAM	83.6	85.3	61.4	63.8	68.4	61.4	64.4	73.7	84.2	71.9	68.3	67.1	65.0
3-By-2 [7]*	SD & SAM	80.9	84.4	54.4	75.1	59.5	60.4	62.6	78.2	81.5	60.0	65.2	66.5	64.2
PartDistill [13]*	GLIP & SAM	84.6	88.4	55.5	71.4	69.2	64.6	60.1	74.7	78.6	72.8	73.4	66.7	65.9
PartField [15]	SAM	75.9	87.6	65.4	72.9	73.8	65.7	52.0	73.9	82.4	50.2	72.0	66.7	66.3
SegGraph (ours)	GLIP & SAM	**87.5** (± 1.73)	**88.5** (± 0.69)	**69.5** (± 0.45)	**79.3** (± 1.70)	**84.2** (± 2.19)	**69.1** (± 2.04)	**66.7** (± 0.88)	**74.8** (± 2.39)	**88.2** (± 1.20)	**70.2** (± 3.40)	**72.8** (± 1.54)	**71.5** (± 2.13)	**70.4** (± 2.08)
	DINOv2 & SAM	**90.0** (± 0.66)	**89.5** (± 0.44)	**73.6** (± 0.15)	**77.3** (± 0.06)	**78.5** (± 0.73)	**70.1** (± 0.80)	**70.8** (± 1.01)	**77.6** (± 0.36)	**87.8** (± 0.47)	**75.7** (± 1.28)	**77.3** (± 1.28)	**74.4** (± 1.23)	**72.8** (± 1.09)

Table 2: Results on shapes with small components of PartNet-E. The mIoU metric is measured in percentage.

	Coffee Machine		Door		Laptop		Lamp		Safe		Remote		Overall (6)	
	Large	Small	Large	Small	Large	Small	Large	Small	Large	Small	Large	Small	Large	Small
PartSLIP [6]	57.1	17.4	44.1	47.3	62.0	10.7	84.1	13.1	68.0	19.4	-	36.5	56.5	24.8
PartSLIP++ [11]	59.2	18.4	43.6	48.5	62.7	7.6	86.1	13.5	71.6	20.1	-	36.4	58.5	25.1
PartSTAD [12]	52.8	18.9	67.7	48.9	71.2	10.2	86.9	12.7	66.5	22.0	-	53.4	62.1	28.9
Ours	55.5	28.7	81.7	55.4	76.5	34.5	84.8	39.3	81.6	54.6	-	82.0	64.1	49.8

Table 3: Few-shot part segmentation results on ShapeNetPart. The mIoU metric is in percentage.

Methods	VLM	Airplane	Cap	Car	Knife	Laptop	Mug	Pistol	Table	Overall(16)
PointCLIP [51]	CLIP	27.2	61.1	30.2	72.2	89.9	59.6	55.6	53.0	52.1
PointCLIPv2 [52]	CLIP	37.1	66.8	38.3	73.3	89.1	75.7	68.4	56.3	57.0
SegGraph (ours)	CLIP & SAM	**37.81** (± 0.73)	**73.96** (± 0.3)	**48.19** (± 1.52)	**74.8** (± 0.93)	**90.98** (± 0.27)	**80.08** (± 0.04)	**70.6** (± 1.16)	**72.72** (± 1.21)	**62.6** (± 1.00)

and cameras (in Laptop). Both the quantitative results in the table and the qualitative visualizations in Fig. 5 indicate that our method exhibits clear advantages in accurately segmenting small parts.

In Fig. 5, we compare the predictions of SegGraph with those of previous approaches. Benefiting from the high-quality over-segmentation provided by SAM, our method achieves superior segmentation performance on small-sized parts compared to the 2D label aggregation methods used in the second and third rows. Notable examples include the hands of clocks, buttons on phones and remote controls, and handles on pot lids. Compared to the fourth row, which uses 3D feature-based method, our method exhibits greater advantages along the boundaries of small objects, such as the handle of a bucket and the clock hands. Overall, our approach yields better segmentation performance for small-size parts.

Results on ShapeNetPart. Tab. 3 presents the evaluation results on ShapeNetPart. Since only sparse point clouds without color are available for ShapeNetPart (2048 points), we follow PointCLIP-v2 to densify and smooth point clouds in preprocessing. Therefore, the rendering depth images of ShapeNetPart are of significantly lower quality compared to the rendered images of PartNet-E, leading to worse SAM segmentation and pretrained image features. However, our method still prevails against baseline methods with a 5.6% improvement in mIoU.

4.2 Ablation Studies

We ablate each module in our method. As the vanilla baseline, we aggregate multi-view 2D features extracted by DINOv2 [43] onto the 3D space using average pooling, resulting in 3D feature representation. A MLP is then employed as a classifier to perform point-wise segmentation on the aggregated 3D features without using SAM segments. The performance of this vanilla baseline is shown in the first row of Tab. 4. All experiments are conducted on all shape categories of PartNet-E with an 8-shot setting. We leave hyperparameter analysis and more results to the supplementary.

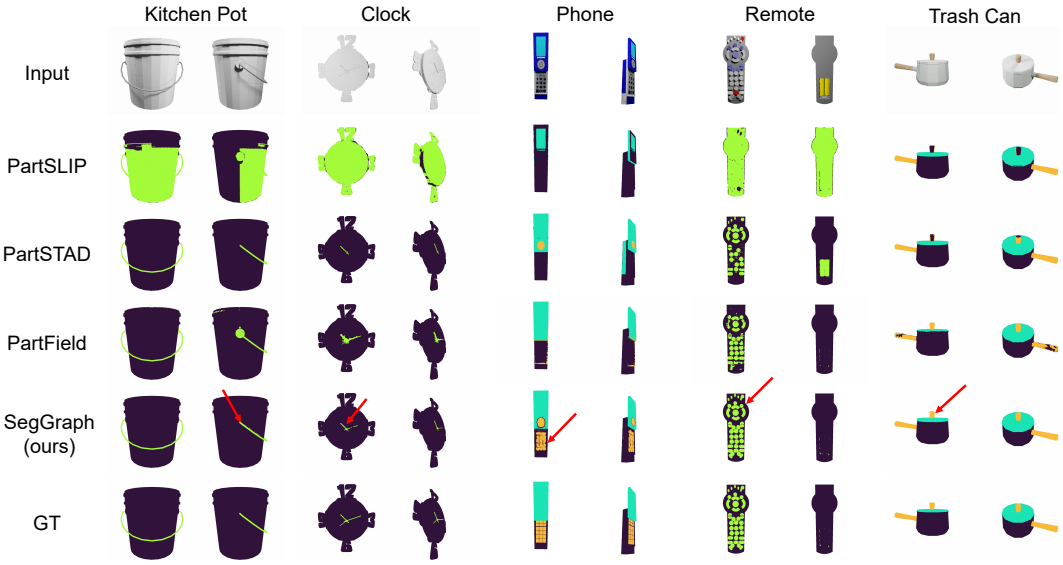

Figure 5: Qualitative comparison of part segmentation results on the PartNet-E dataset under the few-shot setting.

Table 4: Ablation experiments. SE is for segment encoding and AVE for mean pooling or unpooling. SAM means using 3D segments from SAM segmentation.

	SAM	AVE→SE	AVE→Eq.(2)	$\mathcal{E}_o$	$\mathcal{E}_a$	Bottle	Door	Suitca.	Overall(45)
(1)						83.8	62.2	70.3	65.2
(2)	✓					89.0	72.7	70.5	70.5
(3)	✓	✓				88.7	72.8	67.7	70.7
(4)	✓		✓			88.7	72.5	65.5	70.5
(5)	✓	✓	✓			87.9	72.3	66.9	71.0
(6)	✓			✓	✓	87.8	70.5	73.0	71.4
(7)	✓	✓	✓	✓		89.0	73.3	74.4	72.2
(8)	✓	✓	✓		✓	89.7	72.6	74.0	72.3
(9)	✓	✓	✓	✓	✓	90.0	73.6	77.3	72.8

Table 5: Results of using different foundation models and feature propagation (SegGraph vs MLP).

		Bottle	Door	Suitca.	Overall (45)
CLIP [42]	MLP	63.2	35.0	44.0	39.3
	SegGraph	71.2	36.5	52.1	49.5
Diffusion [9]	MLP	78.8	54.0	59.1	56.8
	SegGraph	85.0	59.0	69.4	64.3
GLIP [8]	MLP	82.1	57.7	62.1	61.1
	SegGraph	87.4	69.5	72.8	70.4
PartField [15]	MLP	75.9	65.4	72.0	66.3
	SegGraph	76.6	70.9	75.9	70.3
DINOv2 [43]	MLP	82.0	62.8	68.5	65.2
	SegGraph	90.0	73.6	77.3	72.8

The Role of Segment Encoding (SE) and Viewing Quality-Awareness. The settings without check marks for *AVE→SE* use average pooling, while those without check marks for *AVE→Eq.(2)* removes viewing quality awareness and uses average unpooling, instead. In rows 3-4 of Tab. 4, applying either SE or Eq.(2) individually results in only marginal or even no improvement. However, using both modules together (row 5 vs row 2) leads to slight improvement (0.5%).

The Role of Different Segment Relations. In rows 2,6 of Tab. 4, even without segment encoding and viewing quality awareness, incorporating segment graph leads to segment improvement ($+0.9\%$), suggesting the positive role of graph propagation with GATv2. We conjecture these two graph edges encode geometric features and segment relations, leading to more consistent part segmentation.

Comparing rows 5,7,8 of Tab. 4, using either kind of graph edges for feature propagation with segment encoding and viewing quality awareness exhibits significant performance improvements (1.2% for $\mathcal{E}_a$ and 1.3% for $\mathcal{E}_o$, respectively). Using both edges simultaneously lead to a performance gain of 1.8% (see rows 5,9).

Robustness to Different Foundation Model Features. We conducted experiments by replacing DINOv2 [43] with pretrained features of three different foundation models: CLIP [42], Diffusion [9], GLIP [8], and PartField [15]. Among them, PartField is a 3D model distilled from a 2D foundation model. Since most comparison methods in Table 1 adopt GLIP, for fair comparison, we also modified GLIP into a feature extractor for our experiments. For GLIP, we utilized its multi-scale visual features fused with text features, and further applied a Feature Pyramid Network (FPN) structure to integrate these multi-scale representations, thereby transforming GLIP into a feature extractor. As a comparison, we also replaced the graph propagation module with a MLP to assess the role of structural

Table 6: Detailed runtime analysis of each component in the data preprocessing stage in our method compared with PartSLIP++. All runtime is measured on a per-shape basis.

Method	Render (s)	GLIP+SAM / SAM (s)	Voting Preprocess / Build Graph (s)	Total (s)
PartSLIP++	2.12	16.33 (GLIP+SAM)	92.38 (Voting Preprocess)	110.30
Ours	2.12	52.40 (SAM)	10.40 (Build Graph)	64.92
Ours	2.12	1.30 (FastSAM)	10.40 (Build Graph)	13.82

Table 7: Training and inference time comparison per shape.

Method	Train (s)	Inference (s)
PartSLIP++	—	5.83
Ours	2.25	1.46

reasoning. As shown in Tab. 5, when replacing the MLP with our proposed graph propagation module, we observe consistent and significant performance improvements across all foundation models. This demonstrates that our method is not only well-suited for 2D-to-3D knowledge transfer, but also effectively enhances the performance of 3D segmentation models. These results validate the strong generalizability of our approach across different types of feature representations.

4.3 Runtime Analysis

Tab. 6 and Tab. 7 present the runtime analysis of each component and comparison with PartSLIP++ on the PartNet-E dataset. All measurements are on a per-shape basis, with training time referring to one epoch for a single shape, using a single NVIDIA V100 GPU.

For PartSLIP++, most preprocessing time is spent on superpoint generation via L0-cut pursuit and voting weight computation (92.38 s). Its weighted voting during inference also results in high latency (5.8 s).

For our method, the main overhead stems from SAM segmentation on ten multi-view images. As the SamAutomaticMaskGenerator processes one image at a time, this step is relatively slow but still faster than PartSLIP++ (64.9 s vs. 110.8 s). Employing faster variants such as FastSAM [54] further reduces the time but slightly degrades accuracy (72.8 $\rightarrow$ 70.9 mIoU).

During inference, our method is more efficient (1.46 s vs. 5.83 s). Since PartSLIP++'s training code is unavailable, its training time is omitted. However, we can reasonably expect that our training time is much faster than PartSLIP++'s based on the inference time comparison. Note that both the "Train" and "Inference" time measurements include the time for DINOv2 feature encoding.

4.4 More Analysis

Feature Visualization To more clearly demonstrate the capability of our method in learning 3D structural information, we visualize extracted 3D features in Fig. 6. As a baseline for comparison, we use 3D features obtained by directly averaging DINOv2 [43] features of multi-view rendered images, without any graph-based propagation. We apply PCA to reduce the feature channels to 3, corresponding to the RGB color channels for visualization. Fig. 6 suggests that the features processed by SegGraph exhibit clear separability among different parts. This strong discriminative ability arises from two aspects. First, the few-shot fine-tuning process enables the model to learn meaningful part-level distinctions even the number of shots is quite low (8-shot). Second, it benefits from the proposed graph design to encode 3D structural information. For instance, regions such as the screen area of the phone and the supporting bracket at the top of the coffee machine on the far right-both of which lack annotations in the training set—still exhibit clear part-level separability in the feature visualization.

Feature Similarities Across Shapes. Given the strong part-level separability observed in Fig. 6, we further explore feature similarities in Fig. 7. Specifically, after selecting an anchor point within a shape (indicated by the red arrow), we compute the Euclidean distance between the feature of each point and that of the anchor point, either within the same shape or across different shapes (as shown in the second row). The distances are then normalized, with a distance of 0 mapped to blue and a distance of 1 mapped to gray. As demonstrated in the figure, the selected anchor point yields consistent part-level feature similarity, both within the same shape and across different shapes.

Failure Cases. We examined the visual results and observed an interesting failure case in the USB category (Fig. 8). The USB model contains two disconnected parts—the cap and the body—each with a visually and geometrically similar connection subpart. While SAM correctly distinguishes these subparts, DINOv2 features for them are nearly identical despite their spatial separation. Consequently,

SegGraph misclassifies the two as belonging to the same category. This case indicates that SegGraph can be confused by repeated or similar sub-parts even when they are spatially distant.

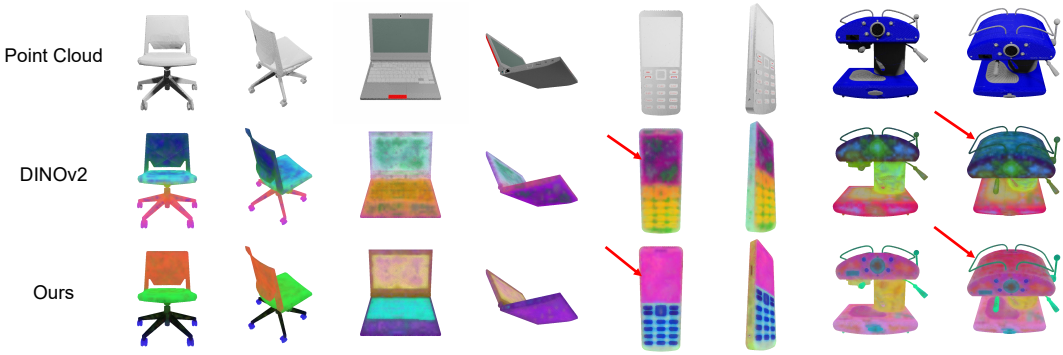

Figure 6: Feature Visualization. Points with similar colors are likely to share the same part label.

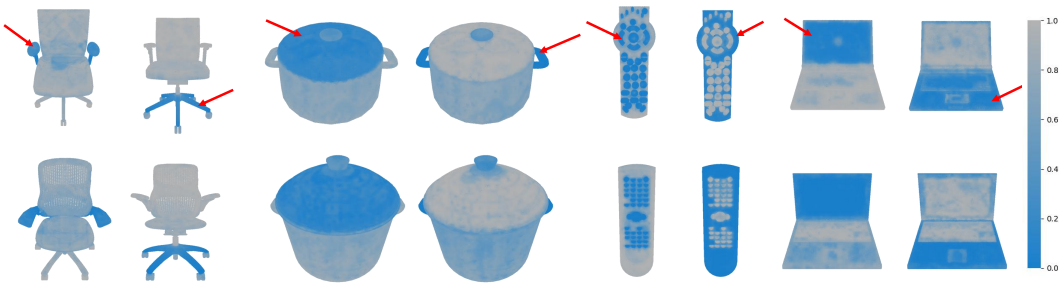

Figure 7: Feature similarities between an anchor point (red arrows) and points in the same shape (top row) and points in a different shape (bottom row). We use colors to encode feature similarities.

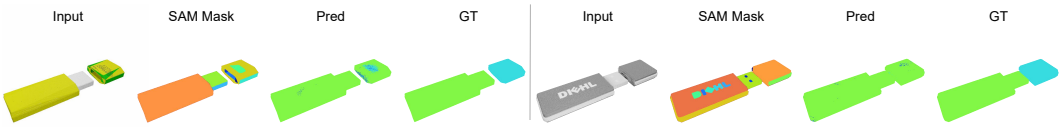

Figure 8: Failure cases on the USB category

5 Conclusion and Discussion

We presented SegGraph, a SAM segment-based graph propagation method for utilizing 2D features of foundation models for 3D part segmentation. Based on multi-view segmentation masks generated by SAM, we construct a graph to encode geometric features with segments as nodes and spatial relationships between nodes as edges. Through feature propagation on this graph, our method effectively adapts features from 2D foundation models to the 3D domain. SegGraph achieves state-of-the-art performance on 3D part semantic segmentation, especially for part boundaries and small components.

While our method demonstrates strong performance, there are some limitations. First, the rendering-view based feature aggregation paradigm fundamentally limits our ability to handle occluded or internal structures of 3D models. Second, the current framework operates at a fixed segmentation scale, without accommodating multi-scale part granularity. Future work could explore constructing hierarchical semantic representations of shapes, facilitating more versatile applications.

Acknowledgement. We thank all the anonymous reviewers for their insightful comments. This work was partially supported by the National Natural Science Foundation of China (62271467, 62476262, 62206263, 62306297, 62306296), Beijing Nova Program, and Beijing Natural Science Foundation (4242053, L242096).

References

[1] Niloy J. Mitra, Michael Wand, Hao Zhang, Daniel Cohen-Or, Vladimir G. Kim, and Qi-Xing Huang. Structure-aware shape processing. In Special Interest Group on Computer Graphics and Interactive Techniques Conference, SIGGRAPH '14, Vancouver, Canada, August 10-14, 2014, Courses, pages 13:1–13:21. ACM, 2014.

[2] Thomas A. Funkhouser, Michael M. Kazhdan, Philip Shilane, Patrick Min, William Kiefer, Ayellet Tal, Szymon Rusinkiewicz, and David P. Dobkin. Modeling by example. ACM Trans. Graph., 23(3):652–663, 2004.

[3] Minghao Chen, Roman Shapovalov, Iro Laina, Tom Monnier, Jianyuan Wang, David Novotny, and Andrea Vedaldi. Partgen: Part-level 3d generation and reconstruction with multi-view diffusion models. arXiv preprint arXiv:2412.18608, 2024.

[4] Jacopo Aleotti and Stefano Caselli. A 3d shape segmentation approach for robot grasping by parts. Robotics Auton. Syst., 60(3):358–366, 2012.

[5] Weiyu Liu, Jiayuan Mao, Joy Hsu, Tucker Hermans, Animesh Garg, and Jiajun Wu. Composable part-based manipulation. In Proceedings of The 7th Conference on Robot Learning, volume 229 of Proceedings of Machine Learning Research, pages 1300–1315. PMLR, 06–09 Nov 2023.

[6] Minghua Liu, Yinhao Zhu, Hong Cai, Shizhong Han, Zhan Ling, Fatih Porikli, and Hao Su. Partslip: Low-shot part segmentation for 3d point clouds via pretrained image-language models. In IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023, pages 21736–21746, 2023.

[7] Anh Thai, Weiyao Wang, Hao Tang, Stefan Stojanov, James M. Rehg, and Matt Feiszli. $3^{\times}$ 2: 3d object part segmentation by 2d semantic correspondences. In Computer Vision - ECCV 2024 - 18th European Conference, Milan, Italy, September 29-October 4, 2024, Proceedings, Part XXXVIII, volume 15096, pages 149–166, 2024.

[8] Liunian Harold Li, Pengchuan Zhang, Haotian Zhang, Jianwei Yang, Chunyuan Li, Yiwu Zhong, Lijuan Wang, Lu Yuan, Lei Zhang, Jenq-Neng Hwang, Kai-Wei Chang, and Jianfeng Gao. Grounded language-image pre-training. In IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022, pages 10955–10965. IEEE, 2022.

[9] Luming Tang, Menglin Jia, Qianqian Wang, Cheng Perng Phoo, and Bharath Hariharan. Emergent correspondence from image diffusion. In Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023, 2023.

[10] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloé Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross B. Girshick. Segment anything. In IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023, pages 3992–4003. IEEE, 2023.

[11] Yuchen Zhou, Jiayuan Gu, Xuanlin Li, Minghua Liu, Yunhao Fang, and Hao Su. Partslip++: Enhancing low-shot 3d part segmentation via multi-view instance segmentation and maximum likelihood estimation. CoRR, abs/2312.03015, 2023.

[12] Hyunjin Kim and Minhyuk Sung. Partstad: 2d-to-3d part segmentation task adaptation. In Computer Vision - ECCV 2024 - 18th European Conference, Milan, Italy, September 29-October 4, 2024, Proceedings, Part V, volume 15063, pages 422–439, 2024.

[13] Ardian Umam, Cheng-Kun Yang, Min-Hung Chen, Jen-Hui Chuang, and Yen-Yu Lin. Partdistill: 3d shape part segmentation by vision-language model distillation. In IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024, pages 3470–3479. IEEE, 2024.

[14] Yunhan Yang, Yukun Huang, Yuan-Chen Guo, Liangjun Lu, Xiaoyang Wu, Edmund Y. Lam, Yan-Pei Cao, and Xihui Liu. Sampart3d: Segment any part in 3d objects. CoRR, abs/2411.07184, 2024.

[15] Minghua Liu, Mikaela Angelina Uy, Donglai Xiang, Hao Su, Sanja Fidler, Nicholas Sharp, and Jun Gao. Partfield: Learning 3d feature fields for part segmentation and beyond. arXiv preprint arXiv:2504.11451, 2025.

[16] Marco Garosi, Riccardo Tedoldi, Davide Boscaini, Massimiliano Mancini, Nicu Sebe, and Fabio Poiesi. 3d part segmentation via geometric aggregation of 2d visual features. In IEEE/CVF Winter Conference on Applications of Computer Vision, WACV 2025, Tucson, AZ, USA, February 26 - March 6, 2025, pages 3257–3267, 2025.

[17] Guofeng Mei, Luigi Riz, Yiming Wang, and Fabio Poiesi. Geometrically-driven aggregation for zero-shot 3d point cloud understanding. In IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024, pages 27896–27905. IEEE, 2024.

[18] Ahmed Abdelreheem, Ivan Skorokhodov, Maks Ovsjanikov, and Peter Wonka. SATR: zero-shot semantic segmentation of 3d shapes. In IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023, pages 15120–15133. IEEE, 2023.

[19] Angel X. Chang, Thomas A. Funkhouser, Leonidas J. Guibas, Pat Hanrahan, Qi-Xing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. Shapenet: An information-rich 3d model repository. CoRR, abs/1512.03012, 2015.

[20] Jens Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Cyrill Stachniss, and Jürgen Gall. Semantickitti: A dataset for semantic scene understanding of lidar sequences. In 2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019, pages 9296–9306, 2019.

[21] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas A. Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017, pages 2432–2443, 2017.

[22] Iro Armeni, Ozan Sener, Amir R. Zamir, Helen Jiang, Ioannis K. Brilakis, Martin Fischer, and Silvio Savarese. 3d semantic parsing of large-scale indoor spaces. In 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016, pages 1534–1543, 2016.

[23] 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 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019, pages 909–918, 2019.

[24] Timo Hackel, Nikolay Savinov, Lubor Ladicky, Jan Dirk Wegner, Konrad Schindler, and Marc Pollefeys. Semantic3d.net: A new large-scale point cloud classification benchmark. CoRR, abs/1704.03847, 2017.

[25] Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017, pages 77–85, 2017.

[26] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA, pages 5099–5108, 2017.

[27] Renrui Zhang, Liuhui Wang, Yali Wang, Peng Gao, Hongsheng Li, and Jianbo Shi. Parameter is not all you need: Starting from non-parametric networks for 3d point cloud analysis. CoRR, abs/2303.08134, 2023.

[28] Xu Ma, Can Qin, Haoxuan You, Haoxi Ran, and Yun Fu. Rethinking network design and local geometry in point cloud: A simple residual MLP framework. In The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022. OpenReview.net, 2022.

[29] Daniel Maturana and Sebastian A. Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2015, Hamburg, Germany, September 28 - October 2, 2015, pages 922–928. IEEE, 2015.

[30] Ji Hou, Angela Dai, and Matthias Nießner. 3d-sis: 3d semantic instance segmentation of RGB-D scans. In IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019, pages 4421–4430, 2019.

[31] Özgün Çiçek, Ahmed Abdulkadir, Soeren S. Lienkamp, Thomas Brox, and Olaf Ronneberger. 3d u-net: Learning dense volumetric segmentation from sparse annotation. In Medical Image Computing and Computer-Assisted Intervention - MICCAI 2016 - 19th International Conference, Athens, Greece, October 17-21, 2016, Proceedings, Part II, volume 9901, pages 424–432, 2016.

[32] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph CNN for learning on point clouds. ACM Trans. Graph., 38(5):146:1–146:12, 2019.

[33] Xin Wei, Ruixuan Yu, and Jian Sun. View-gcn: View-based graph convolutional network for 3d shape analysis. In 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020, pages 1847–1856, 2020.

[34] Wenkai Han, Chenglu Wen, Cheng Wang, Xin Li, and Qing Li. Point2node: Correlation learning of dynamic-node for point cloud feature modeling. In The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020, pages 10925–10932. AAAI Press, 2020.

[35] Meng-Hao Guo, Junxiong Cai, Zheng-Ning Liu, Tai-Jiang Mu, Ralph R. Martin, and Shi-Min Hu. PCT: point cloud transformer. Comput. Vis. Media, 7(2):187–199, 2021.

[36] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip H. S. Torr, and Vladlen Koltun. Point transformer. In 2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021, pages 16239–16248. IEEE, 2021.

[37] Guocheng Qian, Yuchen Li, Houwen Peng, Jinjie Mai, Hasan Hammoud, Mohamed Elhoseiny, and Bernard Ghanem. Pointnext: Revisiting pointnet++ with improved training and scaling strategies. In Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022, 2022.

[38] Guohao Li, Matthias Müller, Ali K. Thabet, and Bernard Ghanem. Deepgcns: Can gcns go as deep as cnns? In 2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019, pages 9266–9275, 2019.

[39] Marios Loizou, Siddhant Garg, Dmitry Petrov, Melinos Averkiou, and Evangelos Kalogerakis. Cross-shape attention for part segmentation of 3d point clouds. Comput. Graph. Forum, 42(5):i–viii, 2023.

[40] Jiahui Wang, Haiyue Zhu, Haoren Guo, Abdullah Al Mamun, Prahlad Vadakkepat, and Tong Heng Lee. Cam/cad point cloud part segmentation via few-shot learning. In 2022 IEEE 20th International Conference on Industrial Informatics (INDIN), pages 359–365. IEEE, 2022.

[41] Lingjing Wang, Xiang Li, and Yi Fang. Few-shot learning of part-specific probability space for 3d shape segmentation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2020.

[42] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event, volume 139, pages 8748–8763, 2021.

[43] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mido Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jégou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision. Trans. Mach. Learn. Res., 2024, 2024.

[44] Marco Garosi, Riccardo Tedoldi, Davide Boscaini, Massimiliano Mancini, Nicu Sebe, and Fabio Poiesi. 3d part segmentation via geometric aggregation of 2d visual features. In IEEE/CVF Winter Conference on Applications of Computer Vision, WACV 2025, Tucson, AZ, USA, February 26 - March 6, 2025, pages 3257–3267, 2025.

[45] Sagi Katz, Ayellet Tal, and Ronen Basri. Direct visibility of point sets. ACM Trans. Graph., 26(3):24, 2007.

[46] Xiuwei Xu, Huangxing Chen, Linqing Zhao, Ziwei Wang, Jie Zhou, and Jiwen Lu. Embodied-sam: Online segment any 3d thing in real time. CoRR, abs/2408.11811, 2024.

[47] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, 2015.

[48] Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks? In The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022, 2022.

[49] Li Yi, Vladimir G. Kim, Duygu Ceylan, I-Chao Shen, Mengyan Yan, Hao Su, Cewu Lu, Qixing Huang, Alla Sheffer, and Leonidas J. Guibas. A scalable active framework for region annotation in 3d shape collections. ACM Trans. Graph., 35(6):210:1–210:12, 2016.

[50] Thang Vu, Kookhoi Kim, Tung Minh Luu, Thanh Xuan Nguyen, and Chang D. Yoo. Softgroup for 3d instance segmentation on point clouds. In IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022, pages 2698–2707. IEEE, 2022.

[51] Renrui Zhang, Ziyu Guo, Wei Zhang, Kunchang Li, Xupeng Miao, Bin Cui, Yu Qiao, Peng Gao, and Hongsheng Li. Pointclip: Point cloud understanding by CLIP. In IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022, pages 8542–8552, 2022.

[52] Xiangyang Zhu, Renrui Zhang, Bowei He, Ziyu Guo, Ziyao Zeng, Zipeng Qin, Shanghang Zhang, and Peng Gao. Pointclip V2: prompting CLIP and GPT for powerful 3d open-world learning. In IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023, pages 2639–2650, 2023.

[53] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022, pages 10674–10685. IEEE, 2022.

[54] Xu Zhao, Wenchao Ding, Yongqi An, Yinglong Du, Tao Yu, Min Li, Ming Tang, and Jinqiao Wang. Fast segment anything. CoRR, abs/2306.12156, 2023.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes] , [No] , or [NA] .
- [NA] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for NA).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will be asked to also include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While "[Yes] " is generally preferable to "[No] ", it is perfectly acceptable to answer "[No] " provided a proper justification is given (e.g., "error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering "[No] " or "[NA] " is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Relevant content can be found in the abstract and Sec. 1.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Relevant content can be found in the Sec. 5.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

   Justification: The paper does not involve theoretical claims that necessitate formal proofs.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The detailed experimental procedure and settings can be found in Sec. 4

   Guidelines:

   - The answer NA means that the paper does not include experiments.

- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We will submit the code in the supplementary material.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).

- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: A comprehensive description of all training and testing details will be provided in the appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: Due to limited computational resources, we were unable to compute error bars through multiple runs.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
   - If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: A description of the computational resources utilized for the experiments will be provided in the appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.

- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Our research fully adheres to the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: The work does not have any societal impact.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: This study poses no such risks.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We have cited all the creators of the code packages and datasets used.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: We only use large language models (LLMs) for editing purposes.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.