Geodesic-Former: a Geodesic-Guided Few-shot 3D Point Cloud Instance Segmenter

Tuan Ngo and Khoi Nguyen

VinAI Research

Abstract. This paper introduces a new problem in 3D point cloud: few-shot instance segmentation. Given a few annotated point clouds exemplified a target class, our goal is to segment all instances of this target class in a query point cloud. This problem has a wide range of practical applications where point-wise instance segmentation annotation is prohibitively expensive to collect. To address this problem, we present Geodesic-Former – the first geodesic-guided transformer for 3D point cloud instance segmentation. The key idea is to leverage the geodesic distance to tackle the density imbalance of LiDAR 3D point clouds. The LiDAR 3D point clouds are dense near the object surface and sparse or empty elsewhere making the Euclidean distance less effective to distinguish different objects. The geodesic distance, on the other hand, is more suitable since it encodes the scene's geometry which can be used as a guiding signal for the attention mechanism in a transformer decoder to generate kernels representing distinct features of instances. These kernels are then used in a dynamic convolution to obtain the final instance masks. To evaluate Geodesic-Former on the new task, we propose new splits of the two common 3D point cloud instance segmentation datasets: ScannetV2 and S3DIS. Geodesic-Former consistently outperforms strong baselines adapted from state-of-the-art 3D point cloud instance segmentation approaches with a significant margin. The code is available at https://github.com/VinAIResearch/GeoFormer.

Keywords: Few-shot Learning, 3D Point Cloud Instance Segmentation

1 Introduction

This paper introduces a new problem of few-shot 3D point cloud instance segmentation (3DFSIS). As Fig. 1 shows, given a few support point clouds (a.k.a. scenes) with their ground-truth masks to define a target class, we aim to segment all instances of the target class in a query scene. Compared to related vision tasks such as 3D point cloud instance segmentation (3DIS) and 3D point cloud few-shot semantic segmentation (3DF3S), 3DFSIS is fundamentally different. For 3DIS, the training and test classes are the same. One could reliably learn an instance segmenter with abundant annotated examples in training, then apply that segmenter to the test scenes. That is not the case in 3DFSIS where training and test classes are disjoint. For 3DF3S, we need to predict each point

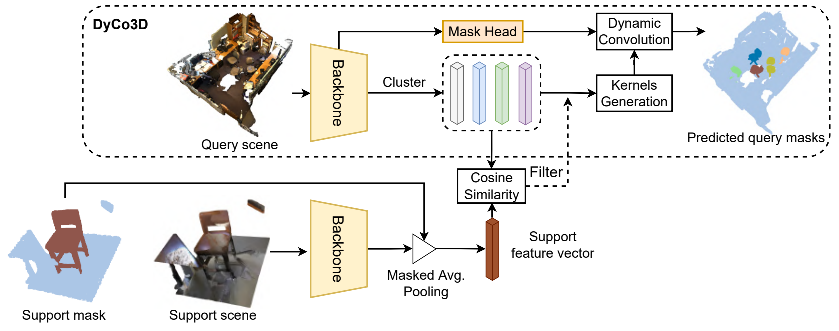

Fig. 1. Our baseline adapted on DyCo3D [16] for 3DFSIS. The query and support point clouds are first input to a shared backbone to extract their features. Then the query points are grouped into candidates based on their semantic and predicted object centroids while the support points are masked-average-pooled to obtain a support feature vector. The cosine similarity between the support feature vector and every candidate's average feature is used to filter out irrelevant candidates. The final candidates are used to generate kernels for dynamic convolution with the feature produced by the mask head in order to obtain the final instance masks of the query scene.

with a semantic label instead of an instance label as in 3DFSIS. That is, we do not need to distinguish different instances of the same class as in 3DF3S. Furthermore, unlike weakly/semi-supervised learning in 3DIS, where all classes are known in training, in the training of 3DFSIS, the new classes are not known in advance. Thus, the model needs to quickly learn from a few examples of new classes whenever they arrive.

3DFSIS is an important vision task and has a wide range of applications including autonomous driving, and augmented reality, especially in applications where training a reliable 3D instance segmenter is prohibitively impossible due to the expensive costs of collecting a sufficient amount of annotated point clouds. However, learning in 3D point clouds is very challenging due to: (1) 3D point clouds are unordered, imbalanced in density (dense near object surface but sparse elsewhere); and (2) the variance in appearance, size, and shape between the support and query scenes is significantly higher than that of 2D images. These two challenges are amplified in the few-shot setting where a very limited number of labeled examples of new classes are provided, e.g. 1 to 5 shots at most compared to 30 to 50 shots with ease in a 2D image. This is due to the reason that one has to label point-by-point in a 3D point cloud rather than labeling approximate polygons for instance masks as in a 2D image. Therefore, it is not trivial to adapt any 2DFSIS to 3DFSIS.

A simple but strong baseline for 3DFSIS can be adapted from a 3D point cloud instance segmenter, e.g. DyCo3D [16], to the few-shot setting. The baseline is depicted in Fig. 1. First, similar points are grouped into candidates based on their Euclidean centroids and semantic predictions. Then each candidate is passed to a subnetwork to generate a kernel for dynamic convolution [40] so as to obtain the final instance mask. To filter out the irrelevant candidates which do

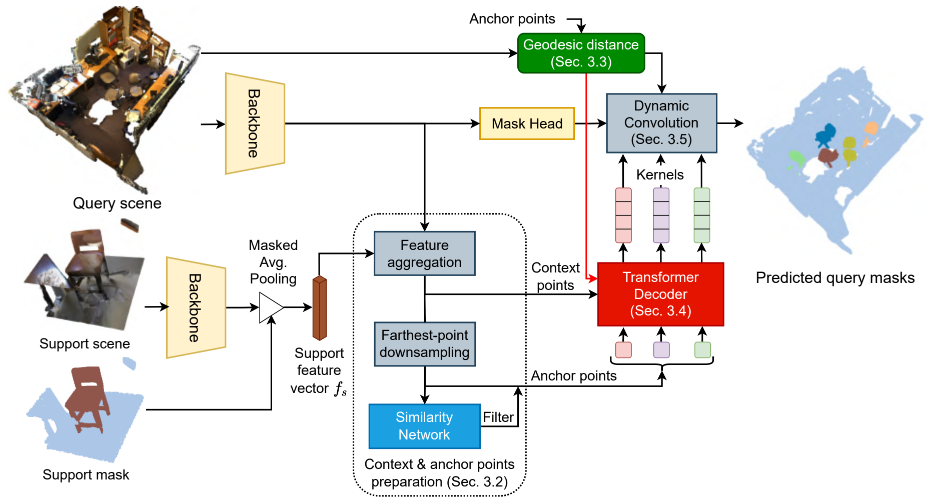

Fig. 2. Our proposed approach, Geodesic-Former, for 3DFSIS. Given support and query scenes, a shared backbone is used to extract their features. Support features are further masked average pooled with the support mask to obtain a support feature vector representing the target class. Then the query features and support feature vector are aggregated to obtain *context points* which are further sub-sampled in farthest-point-downsampling operation to obtain *anchor points* representing the initial prediction of the object location. Next, the geodesic distances between every anchor point to all the context points are computed taking into account the imbalance in point cloud density (distributed near object surface only). This geodesic distance is used as the positional encoding to guide the transformer decoder whose key/value and query are context and anchor points, respectively, so as to produce a kernel for each anchor point. Finally, each kernel dynamically convolves with the features produced by the mask head along with the geodesic distance embedding to obtain the final object instance mask.

not belong to the target class, one can use cosine similarity between the support feature vector and the average feature vector of all points of each candidate. This framework has several limitations. First, as mentioned above, 3D point clouds are imbalanced in density and mostly distributed near the object surface so that the Euclidean distance for clustering is unreliable, i.e. points that are close together might not necessarily belong to the same object and vice versa. Second, the clustering in DyCo3D relies heavily on the performance of the offset centroid predictions, hence, it might be overfitting to some 3D shapes and sizes of the training object classes, resulting in poor generalization to the test classes.

To address these limitations, we propose a new geodesic-guided transformer decoder to generate the kernel for the dynamic convolution from a set of initial anchor points, giving the name of our approach, Geodesic-Former. The overview of Geodesic-Former is depicted in Fig. 2. First, geodesic distance embedding based on the geodesic distances between each of the anchor points to all context points is computed. In this way, the geodesic distance between two points belonging to different objects is very large, helping differentiate different objects. Then this embedding is used as positional encoding to guide the later transformer decoder and dynamic convolution. Second, to avoid overfitting to the shape and size of training classes, we use a combination of the Farthest Point Sampling [37], a similarity network, and a transformer decoder. The first samples initial seeds from the query point cloud representing the initial locations of the object candidates, the second filters out irrelevant candidates, and the third contextualizes the foreground (FG) candidates to precisely represent objects with the information of the context points in order to generate the convolution kernels. In this way, as long as an initial seed belongs to an object, it can represent that object. In contrast, for each point, DyCo3D has to predict exactly the center point of the object it belongs to in order for the clustering to work well. This is even harder when transferring to the new object classes in testing. Also, to the best of our knowledge, we are the first to adopt the transformer decoder architecture to the 3DIS and 3DFSIS.

In sum, our technical contributions are summarized as follows:

- We introduce a new 3D point cloud few-shot instance segmentation task.
- To evaluate the new task, we introduce new splits adapted from the ScannetV2 and S3DIS datasets.
- To address the new task, we propose a strong baseline (adapted from SOTA 3DIS methods) and our novel proposed approach, Geodesic-Former, combining the transformer decoder with dynamic convolution in respect of the geodesic distance encoding scene's geometry.

In the following, Sec. 2 reviews prior work; Sec. 3 specifies Geodesic-Former; and Sec. 4 presents our implementation details and experimental results. Sec. 5 concludes with some remarks and discussions.

2 Related Work

This section reviews closely related work in 2D and 3D instance segmentation.

3D point cloud instance segmentation (3DIS) approaches can be divided into two groups: proposal-based and proposal-free. The *proposal-based* approaches [17,51,52] first detect 3D bounding boxes, then segment the foreground region inside them. 3D-SIS [17] proposes a Mask R-CNN-based 3D instance segmentation architecture, jointly learns features from both RGB images and 3D point cloud. 3D-BoNet [51] simplifies the detection network by directly predicting a fixed number of unoriented 3D bounding boxes from a global feature vector, then segmenting foreground points inside each box. GSPN [52] generates proposals by reconstructing shapes from noisy observations and further refining these proposals with a Region-based PointNet [36]. On the other hand, the *proposal-free* approaches [44,20,3,16,8] learn embedding features then group points to instances. SGPN [44] adopts the double-hinge loss to learn discriminative features in order to compute the similarity matrix of paired points for grouping points. PointGroup [20] predicts the 3D offset from each point to its instance's centroid and generates clusters from two sets: original points and shifted points. HAIS

[3] proposes a hierarchical clustering method where a small cluster can be either filtered out or absorbed by a larger cluster to become its part. DyCo3D [16] adopts the same clustering approach but leverages dynamic convolution [46,40] to generate 3D instance masks. SSTNet [25] constructs a super point tree based on the point cloud's semantic features and uses tree traversal to split nodes into instances. All the 3DIS approaches assume the training and test classes are the same, and there is a large number of annotated data for training. The setting of 3DFIS is fundamentally different: the training and test classes are disjoint and we only have a few annotated examples for each test class.

Few-shot 2D instance segmentation approaches [30,50,9,33,34] extend the Mask R-CNN[13] – a common 2D image instance segmenter – to the few-shot setting. The support features are extracted from a few labeled examples and incorporated into the query feature map to segment objects of the target class. [33] utilizes the anchor-free detector [41] to alleviate the overfitting problem of the anchor boxes to the training classes and assembles the predicted object's latent parts into an object mask. However, 2D images are structured, grid-based, and dense whereas 3D point clouds are unordered, irregular, and sparse. Therefore, these approaches cannot be applied directly to 3DFSIS.

Few-shot 3D point cloud semantic segmentation (3DF3S). Recently, [55] introduced the problem of few-shot 3D point cloud semantic segmentation. From the support scene, multiple prototypes are extracted and propagated to the query points based on their affinity matrix. However, this approach does not distinguish different instances of the same object class. 3DFSIS is arguably harder than 3DFSSS since we need to classify all points into instance labels instead of semantic labels only.

Vision transformer has been applied to 2D image classification [7,42,53], object detection [2,56,45,10,29,47], semantic segmentation [49,38], and instance segmentation [24,12,6,18]. Furthermore, the transformer architecture is naturally fit to process unordered 3D point clouds since its attention mechanism is permutation invariant. Some recent approaches [27,54,31] have shown the potential of transformers in some 3D tasks. [54] designs a self-attention network to process 3D point clouds and achieve good results on 3D semantic segmentation, object part segmentation, and object classification. [27,35,31] leverage transformer-based architecture for 3D object detection. We are the first to adopt the cross-attention transformer decoder with a special design for the 3DIS and 3DFSIS tasks.

3 Our Geodesic-Former

3.1 Problem setting

In training, we are provided a sufficiently large training set of base classes C_{train}, i.e. $\{P^t, m^t\}_{t=1}^{T}$, where P^t, m^t are the 3D point cloud of the scene t and its ground-truth segmentation masks, respectively, and T is the number of training samples. In testing, given K support 3D point cloud scenes P_s and their ground-truth masks m_s to define a new target class c_{test}, we seek to segment all instances

m_q of the target class in a query scene P_q. It is worth noting that the target class is different from the base classes, or $c_{test} \notin C_{train}$. In this paper, we explore two configurations: 1-shot and 5-shot instance segmentation.

To address this problem, we design our approach Geodesic-Former inspired by DyCo3D [16]. The overview of Geodesic-Former is illustrated in Fig. 2. To extract features F_s, F_q from the support and query point clouds P_s, P_q, respectively, we employ a 3D U-Net with sparse convolution [11] used in [16]. In addition, a support feature vector f_s is extracted from the support features F_s via a masked-average-pooling operation representing the target class.

In the following, Sec. 3.2 first describes how to prepare the context and anchor points for the transformer decoder. Sec. 3.3 specifies how to compute the geodesic distance between each anchor point to every context point to guide the transformer decoder and dynamic convolution. Sec. 3.4 discusses how the transformer decoder generates the convolution kernel for the dynamic convolution, which is presented in Sec. 3.5, in order to produce the final instance mask. Finally, Sec. 3.6 proposes the strategy to train our approach.

3.2 Context and Anchor Points Preparation

First, we aggregate support feature $f_s \in \mathbb{R}^{1\times d}$ into the query features $F_q \in \mathbb{R}^{N_q\times d}$ inspired by [48], resulting in integrated features of the *context points* $F_c \in \mathbb{R}^{N_q\times d}$ as follows:

$$F_c = W_{proj} * [F_q \odot f_s; F_q - f_s; F_q], \tag{1}$$

where d is the number of feature channels, N_q is the number of query points, $W_{proj} \in \mathbb{R}^{3d\times d}$ is the linear layer weight; $*, [\cdot;\cdot], \odot, -$ are the convolution, concatenation, channel-wise multiplication, and subtraction operations, respectively. In this way, we preserve the original query point features along with the newly rectified and subtracted features from the support.

Next, from the context points, a smaller set of points is sampled by a farthest-point-down sampling to represent distinct object candidates. In our work, we sample a large enough number of candidates so that they can cover all objects in all cases. Then, a similarity network, which is a multi-layer perceptron (MLP), is used to filter relevant candidates as *anchor points* $F_a \in \mathbb{R}^{N_a\times d}$, N_a is the number of anchor points, having high appearance similarity with the support examples. After this step, we take the context points F_c and anchor points F_a as input to the transformer decoder to generate the kernel of each anchor point.

3.3 Geodesic Distance Embedding Computation

The 3D point clouds captured by LiDAR sensors have an important property that it is distributed unequally in the 3D space (dense near the object surface and sparse elsewhere). As a result, two points are close in 3D Euclidean distance but they might belong to two different objects. In this case, the geodesic distance [21] which encodes the scene's geometry would be a better choice as visualized in

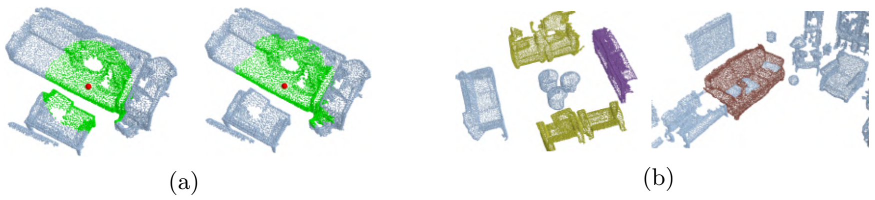

Fig. 3. (a) Comparison between Euclidean distance and geodesic distance. For each image, the green points are the top-2000 nearest neighbors of the red point in Euclidean distance (left) and geodesic distance (right). (b) An example of FG/BG flipping in training and testing making transformer classifier confused, i.e. sofa is labeled as BG in training (left) but FG in testing (right).

Fig. 3a. In other words, if two points are close in Euclidean distance but there is no path or their geodesic distance is too high, they clearly belong to two separate objects. Therefore, we propose to use the geodesic distance between the anchor points and every context point as geometry guidance to distinguish objects in subsequent modules.

To obtain the geodesic distance, we first employ the ball query algorithm [37] to get a directed sparse graph whose nodes are context points and each node only connects to at most κ other nodes. There exists a directed edge from node 1 to node 2 if node 2 is among the κ nearest neighbors of node 1 and within a radius τ, and the weight of the edge is always positive and equal to the local Euclidean distance between the two nodes. After that, we use the shortest path algorithm, i.e. Djikstra [5], to compute the length of the shortest path from each anchor point to every context point in the obtained sparse graph as its geodesic distance. Finally, the geodesic distance embedding $G^i \in \mathbb{R}^{N_q \times d}$ of an anchor point i is obtained by encoding its geodesic distance using the sine/cosine function in [43].

3.4 Transformer Decoder

The transformer decoder takes as input the anchor points $F_a \in \mathbb{R}^{N_a \times d}$ and context points $F_c \in \mathbb{R}^{N_q \times d}$ to produce the kernel $W^i \in \mathbb{R}^L$ for each anchor point i, where L is the number of parameters in dynamic convolution. The decoder follows the design of DETR [2] consisting of a multi-block of transformer layers with two kinds of attention: self-attention between anchor points and cross-attention between anchor and context points. Hence, each anchor point knows each other and captures a complete object structure to generate a kernel for the dynamic convolution. Notably, the attention mechanism in a transformer is inherently fitted to the 3D point cloud since they are both unordered.

Importantly, to address the 3DFSIS, we make substantial modifications to the positional encoding and output of the decoder. First, to guide the attention in the transformer with the geodesic geometry structure as discussed in Sec. 3.3, the geodesic distance embedding G is used as the positional encoding instead of the embedding of 3D point coordinates. Second, we do not predict the object class, i.e., the foreground (FG)/background (BG) classification, due to the FG and BG confusion of few-shot settings during the training and testing phase. In particular, a lot of new classes presenting in training scenes but are labeled as BG causing the trained classification head to predict them as BG (false negative) in testing as depicted in Fig. 3b. Instead, the similarity network filters the FG anchor points as described in Sec. 3.2.

3.5 Dynamic Convolution

To prepare features for dynamic convolution whose weights are predicted by the transformer decoder, a **mask head** takes as input the query point features $F_q \in \mathbb{R}^{N_q \times d}$ to produce the mask features $F_{mask} \in \mathbb{R}^{N_q \times d}$ by applying two blocks of MLP with Batch Norm [19], and ReLU [32] in between. Also, the geodesic distance is critical geometric cue to distinguish instances, we directly append the geodesic distance embedding $G^i \in \mathbb{R}^{N_q \times d}$ of each anchor point i to the mask features in order to obtain the final instance mask $\widehat{m}^i \in [0,1]^{N_q \times 1}$ in a **dynamic convolution** as follows :

$$\widehat{m}^i = \text{Conv}\left([F_{mask}; G^i]; W^i\right), \tag{2}$$

where $[\cdot;\cdot]$ is the concatenation operation, and Conv is implemented with several convolutional layers as in DyCo3D [16].

3.6 Training Strategy

Pretraining: First, we pretrain the U-Net backbone, mask head, and the transformer decoder with the standard 3D point cloud instance segmentation task on the base classes. In this stage, the feature aggregation in Fig. 2 is not used since we do not have support feature, instead, we copy the features of query points to the context points directly. Also, we add a classification head on top of the output of the transformer decoder to predict the semantic category $\widehat{\gamma}^i$ along with the kernel generation to predict the mask $\widehat{m}^i$ for each anchor point i. The number of classes is $\Gamma + 1$ where Γ is the total number of base classes of C_{train} and one additional background class. The matching cost $C^{\text{pretrain}}_{\text{match}} \in \mathbb{R}_+^{N_a \times N_{gt}}$ between the prediction $(\widehat{\gamma}^i, \widehat{m}^i)$ and the ground truth (γ^j, m^j) is computed as:

$$C^{\text{pretrain}}_{\text{match}}(i,j) = L_{seg}(\widehat{m}^i, m^j) + L_{cls}(\widehat{\gamma}^i, \gamma^j), \tag{3}$$

where L_{seg} is the dice loss [39], and L_{cls} is the sigmoid focal loss [26]. Based on the matching cost $C^{\text{pretrain}}_{\text{match}}$, the Hungarian algorithm [23] is leveraged to find the optimal 1-to-1 matching π^*, then the following loss is used for training:

$$L^{\text{pretrain}}_{\text{Hungarian}} = \sum_{i=1}^{N_{GT}} L_{seg}(\widehat{m}^i, m^{\pi^*(i)}) + \sum_{i=1}^{N_a} L_{cls}(\widehat{\gamma}^i, \gamma^{\pi^*(i)}). \tag{4}$$

If a class prediction $\widehat{\gamma}^i$ has no ground truths matched, it will be matched with the background class.

Table 1. Class splits of the ScannetV2 and S3DIS datasets. Fold 0 is used for training while fold 1 is used for testing.

ScannetV2		S3DIS	
Fold 0	Fold 1	Fold 0	Fold 1
cabinet	sofa	beam	door
bed	table	board	floor
chair	window	bookcase	sofa
door	picture	ceiling	table
bookshelf	shower curtain	chair	wall
counter	refrigerator	column	window
desk	toilet		
curtain	sink		
bathtub	other furniture		

Episodic training: We leverage the episodic training strategy – a common approach for few-shot image classification – to mimic the test scenario in training. That is, for each episode, we randomly sample a pair of support and query point clouds P_s, P_q and their masks m_s, m_q from training examples of the base classes. In this stage, the classification head is removed and we add feature aggregation and similarity network to train with the transformer decoder while freezing the backbone and mask head. This is the final architecture of Geodesic-Former as depicted in Fig. 2. The following matching cost and loss are used to train and Approach in this stage:

$$C_{\text{match}}^{\text{episodic}}(i,j) = L_{seg}(\widehat{m}^i, m^j), \quad L_{\text{Hungarian}}^{\text{episodic}} = \sum_{j=1}^{N_{GT}} L_{seg}(\widehat{m}^i, m^{\pi^*(i)}). \tag{5}$$

For $K > 1$ **shots**, we additionally apply the episodic training on a set of balanced support-query pairs of the base and new classes to further fine-tune the Geodesic-Former. In testing, the final support feature vector f_s is the average vector of all feature vectors f_s^k of K support scenes.

4 Experiments

Datasets: To evaluate Geodesic-Former on the new 3DFSIS task, we introduce two new datasets derived from ScannetV2 [4] and S3DIS [1] used for 3D point cloud instance segmentation. ScannetV2 consists of 1613 point clouds of scans from 707 unique indoor scenes with 20 semantic classes in total and 18 classes for instance segmentation. We follow the common split of 1201, 312, and 100 for training, evaluating, and testing, respectively [16]. Inspired by [55] for 3D few-shot semantic segmentation, we split the 18 foreground classes into two non-overlapping folds based on the alphabetical order with nine classes each, one for training classes (fold 0) and the other for test classes (fold 1). S3DIS is another benchmark for 3D indoor scenes which contains 272 point clouds collected from 6 large-scale areas with 13 semantic categories. We only keep 12 main categories and remove the "clutter" class. We also split it into two folds with six classes each. Area 5 containing 68 point clouds is used for testing while the rest is used for training. Tab. 1 summarizes the class splits of ScannetV2 and S3DIS.

We report the results for the test classes in the following procedure: (1) we randomly sample $K = \{1, 5\}$ support examples, with their binary masks for every class in the training set (with the purpose of saving the whole test set for the query scenes only) and apply them to the whole test set, a.k.a the fixed support set; (2) for each query scene in the test set, if a test class does not present in the scene, we skip the evaluation of that class for that scene. To improve the reliability of the measured metrics, we sample and evaluate all the approaches on ten disjoint fixed support sets, and report the average with standard deviation. In this setting, we consider the unlabeled points of new classes in the training set as unseen points commonly used in 2DFSIS.

Evaluation metrics: For ScannetV2, we adopt the mean average precision (mAP) and AP50 used in the instance segmentation task. For S3DIS, we apply the metrics that are used in [20,16,14,15] to test classes: mCov, mPrec, and mRec. They are the mean instance-wise IoU, mean precision, and mean recall.

Implementation details: We adopt the sparse convolution [11] to implement the backbone network. The voxel size is set to 0.02 m for ScannetV2 and 0.05 m for S3DIS, and the output channel of the backbone network is set to 16. To calculate the geodesic distance, we employ the FAISS[1] library for ball-query search, then we re-implement by vectorizing the shortest path algorithm, i.e., Dijkstra's algorithm, in Pytorch to speed up the processing time. The transformer decoder is the same as [31] consisting of four layers, each uses multi-head attention with four heads, and the output dimension and the hidden dimension are set to 64. We train our model using the Adam optimizer [22] with a cosine learning rate scheduler [28]. During the pretraining phase, the initial learning rate is set to 10^{-2}, and the number of training epochs is 500. After that, we train for another 200 epochs in episodic training with the learning rate of 5×10^{-3}. Our data augmentation is the same as [20]'s.

4.1 Ablation Study

We conduct several experiments on the validation set of ScannetV2 to study the contribution of various components of our method with one shot, $K = 1$.

Similarity network, transformer decoder, geodesic distance. In Tab. 2, the first and second rows show the performance of our baseline in Fig. 1, and a per-point classification variant where we use cosine similarity to filter out irrelevant points before clustering by predicted objects' centers. This variant performs poorly as each point is classified independently without geometric cues of objects, and the classified points are so cluttered to form a complete shape. When replacing the cosine similarity in the baseline with a similarity network, the performance slightly increases, +0.4 in row 3. When the clustering algorithm

[1] https://github.com/facebookresearch/faiss

Table 2. Ablation study on each component's contribution to the final results. "SN", "TD", and "GDE" denote similarity network, transformer decoder, and geodesic distance embedding, respectively. (*) denotes the baseline of per-point classification.

	Combination			Metric	
	SN	TD	GDE	mAP	AP50
Baseline (DyCo3D [16])				6.2	11.7
Baseline (*)				4.9	9.7
	✓			6.6	12.5
		✓		6.7	13.1
			✓	7.8	14.2
	✓	✓		7.6	14.3
	✓		✓	8.7	14.9
		✓	✓	9.4	17.1
Geodesic-Former	✓	✓	✓	**10.6**	**19.8**
Geodesic-Former w/ cls.		✓	✓	4.5	10.2

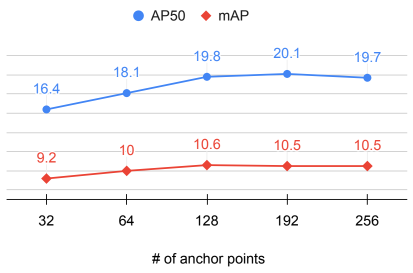

Fig. 4. Study on the number of anchor points N_a.

in the baseline is replaced by the transformer decoder, the performance also slightly improves, +0.5 in row 4. Especially, when adding the geodesic distance embedding to the dynamic convolution of the baseline, the performance is significantly boosted, +1.6 in row 5. This justifies the importance of geodesic distance to the segmentation. When combining each pair of the three components, the performance improves substantially over each component alone. Finally, our full approach, Geodesic-Former achieves the best performance, 10.6 in mAP and 19.8 in AP50. These results show that when combining these components together, the performance gain is much larger than using them separately. We also have an ablation when turning off the similarity network and using the classification head in the pretraining phase, the performance drops significantly, -6.1 in row 9. This justifies our claims that using the classification head in our 3DFSIS is sub-optimal due to the FG/BG confusion as described in Sec. 3.4.

Number of anchor points. The results are summarized in Fig. 4. Using the number of anchor points of 128 gives the best results. This is because using too few anchor points cannot capture the diversity of objects in the scene, whereas using too many does not boost the performance significantly.

Table 3. Study on the number of dynamic convolution layers.

# of layers	1	2	3	4
mAP on training set	22.6 ± 1.4	28.1 ± 1.7	28.0 ± 1.3	28.3 ± 1.5
mAP on testing set	3.4 ± 0.2	10.6 ± 0.6	9.3 ± 1.3	6.4 ± 1.9

Table 4. Study on ball query settings in Sec. 3.3 to form sparse directed graph.

mAP	$\kappa = 16$	$\kappa = 32$	$\kappa = 64$	$\kappa = 128$
$\tau = 0.03$ m	9.0 ± 0.9	9.3 ± 0.8	9.9 ± 0.7	10.1 ± 0.8
$\tau = 0.05$ m	9.2 ± 0.6	9.7 ± 0.8	**10.6 ± 0.6**	10.6 ± 0.7
$\tau = 0.1$ m	8.9 ± 1.2	9.3 ± 0.7	10.5 ± 1.0	10.3 ± 0.9

Number of layers in the dynamic convolution. As can be seen in Tab. 3, using only a single layer of dynamic convolution leads to a significant drop in performance (-7.2 in mAP). On the other hand, using too many layers may be prone to overfitting the training data and harder to optimize due to a large number of generated parameters. Using two layers gives the best results.

Ball query configuration. Tab. 4 reports the results with different nearest neighbors κ and radii τ to form the directed sparse graph (as described in Sec. 3.3) in order to compute the geodesic distance. From this table, $\kappa = 64$ and $\tau = 0.05$ m give the best results.

4.2 Comparison with Prior Work

Since there is no prior work on 3DFSIS, we adapt three state-of-the-art (SOTA) approaches on 3DIS: DyCo3D [16], PointGroup [20], and HAIS [3] to the few-shot setting for comparing with our approach. The adapted version of DyCo3D is exactly our baseline as depicted in Fig. 1. We apply the cosine similarity filter to all methods to remove irrelevant proposals after the clustering stage and the other modules are kept exactly the same as in their original papers. The similarity thresholds for these methods are carefully fine-tuned to achieve the best performance for a fair comparison, i.e. 0.95, 0.9, and 0.8 for DyCo3D, PointGroup, and HAIS, respectively. Notably, the set aggregation module in HAIS requires another statistical class-specific radius to aggregate fragments into larger components. We calculate this radius based on the support scenes and then apply it to the query scene.

Tab. 5 and Tab. 6 show the comparison results on the S3DIS and ScannetV2 datasets, respectively. For ScannetV2, HAIS performs worst among the four, probably due to the sensitive class-specific radius in its set aggregation module. Geodesic-Former consistently outperforms all of them by a large margin in all metrics, i.e., +4.4 for one shot and +6.8 for five shots in the mAP. Moreover, our method is more robust across different runs where the standard variations of mAP and AP50 are only 0.7 and 1.4, respectively, compared with 2.0 and 3.1 of the second-best DyCo3D's. For S3DIS, Geodesic-Former outperforms others with a significant margin, i.e. in mCov and mRec, about +4 for one shot and

Table 5. Comparison of Geodesic-Former and the strong baselines on ScannetV2.

	$K = 1$		$K = 5$	
	mAP	AP50	mAP	AP50
DyCo3D [16]	6.2 ± 2.0	11.7 ± 3.1	6.4 ± 1.2	11.9 ± 2.2
PointGroup [20]	5.3 ± 1.2	10.3 ± 2.5	5.3 ± 0.5	11.7 ± 0.8
HAIS [3]	1.6 ± 0.6	3.5 ± 0.8	1.0 ± 0.2	2.3 ± 0.4
Geodesic-Former	**10.6 ± 0.7**	**19.8 ± 1.4**	**13.2 ± 0.9**	**24.8 ± 1.3**

Table 6. Comparison of Geodesic-Former and the strong baselines on S3DIS.

	$K = 1$			$K = 5$		
	mCov	mPre	mRec	mCov	mPre	mRec
DyCo3D [16]	13.5 ± 2.1	2.9 ± 1.0	4.1 ± 1.4	14.5 ± 1.3	3.1 ± 0.5	4.1 ± 1.4
PointGroup[20]	12.9 ± 2.8	4.6 ± 1.4	3.8 ± 1.3	13.7 ± 0.8	4.6 ± 0.6	3.8 ± 0.8
HAIS [3]	4.6 ± 1.2	**8.1 ± 0.9**	3.9 ± 1.3	5.0 ± 1.9	**11.8 ± 2.0**	4.1 ± 0.4
Ours	**17.8 ± 1.5**	7.0 ± 0.4	**8.5 ± 1.7**	**20.2 ± 2.1**	10.8 ± 1.3	**12.2 ± 1.8**

+7 for five shots. HAIS's results are slightly better than ours in mPre due to its strict threshold to get high precision but low recall rate.

4.3 Qualitative Results

Fig. 5 shows the qualitative results of our approach and others on ScannetV2. For the training class "chair" shown on row 1, all approaches perform well. For the test classes (rows 2-5), there are differences in the segmentation results. Geodesic-Former outperforms others in the hard cases such as in the thin object ("show curtain" - row 2), in the big object ("table" - row 3), and in the incomplete object ("window" - row 4). These examples demonstrate the strong capability of our approach when handling objects to various extent thanks to the transformer decoder and the geodesic distance embedding. However, Geodesic-Former mis-segments the sofa-stool as sofa due to their similar appearance (row 5).

Also, Fig. 6 illustrates the quality of the computed geodesic distance. For each red point, we visualize the top reachable geodesic-distance nearest neighbors (green points) and unreachable points (gray points) which have *infinite* geodesic distance. It justifies that the geodesic distance helps distinguish objects much better than Euclidean distance.

5 Discussion and Conclusion

Discussion. We have succeeded in applying our approach on a lower number of shots only, i.e., 1 and 5 shots. For a higher number of shots ($K > 5$), the improvement is insignificant due to the simple averaging operation. The study on how to aggregate features from multiple supports in a 3D point cloud to leverage their geometric structure would be an interesting research topic.

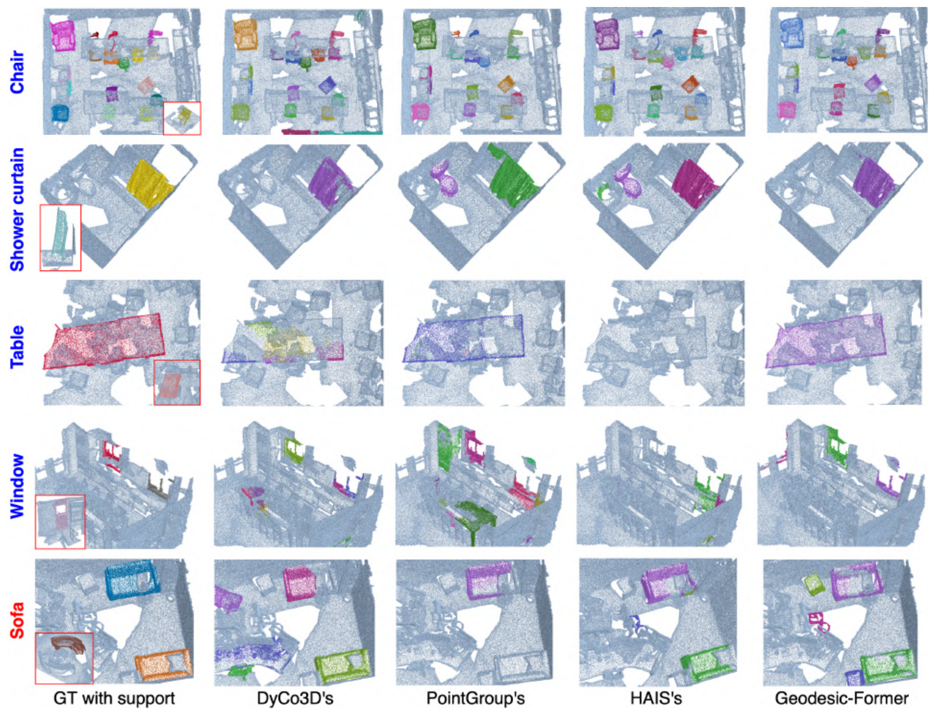

Fig. 5. Qualitative results of Geodesic-Former and the strong baselines on the ScannetV2 dataset. Each row shows an example of the query scene with its GT mask and the support scene with its GT mask (the smaller red-border box) on the first column. The name of the support class is on the left next to GT.

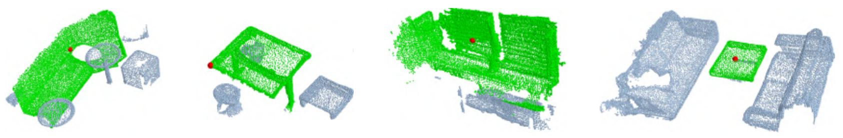

Fig. 6. Representative examples of computed geodesic distance. For each image, the green points is the top reachable geodesic-distance nearest neighbors of the red point.

Conclusion. In this work, we have introduced the new few-shot 3D point cloud instance segmentation task and have proposed the Geodesic-Former – a new geodesic-guided transformer with dynamic convolution to address it. Extensive experiments have been conducted on the newly introduced splits of ScannetV2 and S3DIS datasets showing that our approach achieves robust and significant performance gain on both datasets from the very strong baselines adapted from the state-of-the-art approaches in 3D instance segmentation, i.e., +4.4 for one shot and +6.8 for five shots in mAP on ScannetV2; +4.3 for one shot and +5.7 for five shots in mCov on S3DIS. We hope that our proposed problem, datasets, and approach could facilitate future work in this direction.

References

1. Armeni, I., Sener, O., Zamir, A.R., Jiang, H., Brilakis, I., Fischer, M., Savarese, S.: 3d semantic parsing of large-scale indoor spaces. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2016) 9
2. Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S.: End-to-end object detection with transformers. In: European Conference on Computer Vision. Springer (2020) 5, 7
3. Chen, S., Fang, J., Zhang, Q., Liu, W., Wang, X.: Hierarchical aggregation for 3d instance segmentation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (2021) 4, 5, 12, 13
4. Dai, A., Chang, A.X., Savva, M., Halber, M., Funkhouser, T., Nießner, M.: Scannet: Richly-annotated 3d reconstructions of indoor scenes. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2017) 9
5. Dijkstra, E.W., et al.: A note on two problems in connexion with graphs. Numerische mathematik **1**(1), 269–271 (1959) 7
6. Dong, B., Zeng, F., Wang, T., Zhang, X., Wei, Y.: Solq: Segmenting objects by learning queries. arXiv preprint arXiv:2106.02351 (2021) 5
7. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al.: An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929 (2020) 5
8. Engelmann, F., Bokeloh, M., Fathi, A., Leibe, B., Nießner, M.: 3d-mpa: Multi-proposal aggregation for 3d semantic instance segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2020) 4
9. Fan, Z., Yu, J.G., Liang, Z., Ou, J., Gao, C., Xia, G.S., Li, Y.: Fgn: Fully guided network for few-shot instance segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2020) 5
10. Fang, Y., Liao, B., Wang, X., Fang, J., Qi, J., Wu, R., Niu, J., Liu, W.: You only look at one sequence: Rethinking transformer in vision through object detection. arXiv preprint arXiv:2106.00666 (2021) 5
11. Graham, B., Engelcke, M., Van Der Maaten, L.: 3d semantic segmentation with submanifold sparse convolutional networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2018) 6, 10
12. Guo, R., Niu, D., Qu, L., Li, Z.: Sotr: Segmenting objects with transformers. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (2021) 5
13. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask r-cnn. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (2017) 5
14. He, T., Gong, D., Tian, Z., Shen, C.: Learning and memorizing representative prototypes for 3d point cloud semantic and instance segmentation. In: European Conference on Computer Vision. Springer (2020) 10
15. He, T., Liu, Y., Shen, C., Wang, X., Sun, C.: Instance-aware embedding for point cloud instance segmentation. In: European Conference on Computer Vision. Springer (2020) 10
16. He, T., Shen, C., van den Hengel, A.: Dyco3d: Robust instance segmentation of 3d point clouds through dynamic convolution. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2021) 2, 4, 5, 6, 8, 9, 10, 11, 12, 13

17. Hou, J., Dai, A., Nießner, M.: 3d-sis: 3d semantic instance segmentation of rgb-d scans. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4421–4430 (2019) 4
18. Hu, J., Cao, L., Lu, Y., Zhang, S., Wang, Y., Li, K., Huang, F., Shao, L., Ji, R.: Istr: End-to-end instance segmentation with transformers. arXiv preprint arXiv:2105.00637 (2021) 5
19. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: International Conference on Machine Learning. PMLR (2015) 8
20. Jiang, L., Zhao, H., Shi, S., Liu, S., Fu, C.W., Jia, J.: Pointgroup: Dual-set point grouping for 3d instance segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2020) 4, 10, 12, 13
21. Kimmel, R., Sethian, J.A.: Computing geodesic paths on manifolds. Proceedings of the national academy of Sciences **95**(15), 8431–8435 (1998) 6
22. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014) 10
23. Kuhn, H.W.: The hungarian method for the assignment problem. Naval research logistics quarterly **2**(1-2), 83–97 (1955) 8
24. Li, Z., Wang, W., Xie, E., Yu, Z., Anandkumar, A., Alvarez, J.M., Lu, T., Luo, P.: Panoptic segformer. arXiv preprint arXiv:2109.03814 (2021) 5
25. Liang, Z., Li, Z., Xu, S., Tan, M., Jia, K.: Instance segmentation in 3d scenes using semantic superpoint tree networks. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (2021) 5
26. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (2017) 8
27. Liu, Z., Zhao, X., Huang, T., Hu, R., Zhou, Y., Bai, X.: Tanet: Robust 3d object detection from point clouds with triple attention. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 34 (2020) 5
28. Loshchilov, I., Hutter, F.: Sgdr: Stochastic gradient descent with warm restarts. arXiv preprint arXiv:1608.03983 (2016) 10
29. Meng, D., Chen, X., Fan, Z., Zeng, G., Li, H., Yuan, Y., Sun, L., Wang, J.: Conditional detr for fast training convergence. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (2021) 5
30. Michaelis, C., Ustyuzhaninov, I., Bethge, M., Ecker, A.S.: One-shot instance segmentation. arXiv preprint arXiv:1811.11507 (2018) 5
31. Misra, I., Girdhar, R., Joulin, A.: An end-to-end transformer model for 3d object detection. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (2021) 5, 10
32. Nair, V., Hinton, G.E.: Rectified linear units improve restricted boltzmann machines. In: International Conference on Machine Learning (2010) 8
33. Nguyen, K., Todorovic, S.: Fapis: A few-shot anchor-free part-based instance segmenter. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 11099–11108 (2021) 5
34. Nguyen, K., Todorovic, S.: ifs-rcnn: An incremental few-shot instance segmenter. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 7010–7019 (2022) 5
35. Pan, X., Xia, Z., Song, S., Li, L.E., Huang, G.: 3d object detection with pointformer. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2021) 5

36. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2017) 4
37. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In: Advances in Neural Information Processing Systems (2017) 4, 7
38. Strudel, R., Garcia, R., Laptev, I., Schmid, C.: Segmenter: Transformer for semantic segmentation. arXiv preprint arXiv:2105.05633 (2021) 5
39. Sudre, C.H., Li, W., Vercauteren, T., Ourselin, S., Cardoso, M.J.: Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations. In: Deep learning in medical image analysis and multimodal learning for clinical decision support, pp. 240–248. Springer (2017) 8
40. Tian, Z., Shen, C., Chen, H.: Conditional convolutions for instance segmentation. In: European Conference on Computer Vision. Springer (2020) 2, 5
41. Tian, Z., Shen, C., Chen, H., He, T.: Fcos: Fully convolutional one-stage object detection. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (2019) 5
42. Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., Jégou, H.: Training data-efficient image transformers & distillation through attention. In: International Conference on Machine Learning. PMLR (2021) 5
43. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, **L**., Polosukhin, I.: Attention is all you need. In: Advances in Neural Information Processing Systems (2017) 7
44. Wang, W., Yu, R., Huang, Q., Neumann, U.: Sgpn: Similarity group proposal network for 3d point cloud instance segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2569–2578 (2018) 4
45. Wang, W., Xie, E., Li, X., Fan, D.P., Song, K., Liang, D., Lu, T., Luo, P., Shao, L.: Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. arXiv preprint arXiv:2102.12122 (2021) 5
46. Wang, X., Zhang, R., Kong, T., Li, L., Shen, C.: Solov2: Dynamic and fast instance segmentation. In: Advances in Neural information processing systems. vol. 33 (2020) 5
47. Wang, Y., Zhang, X., Yang, T., Sun, J.: Anchor detr: Query design for transformer-based detector. arXiv preprint arXiv:2109.07107 (2021) 5
48. Xiao, Y., Marlet, R.: Few-shot object detection and viewpoint estimation for objects in the wild. In: European Conference on Computer Vision. Springer (2020) 6
49. Xie, E., Wang, W., Yu, Z., Anandkumar, A., Alvarez, J.M., Luo, P.: Segformer: Simple and efficient design for semantic segmentation with transformers. arXiv preprint arXiv:2105.15203 (2021) 5
50. Yan, X., Chen, Z., Xu, A., Wang, X., Liang, X., Lin, L.: Meta r-cnn: Towards general solver for instance-level low-shot learning. In: Proceedings of the IEEE International Conference on Computer Vision (2019) 5
51. Yang, B., Wang, J., Clark, R., Hu, Q., Wang, S., Markham, A., Trigoni, N.: Learning object bounding boxes for 3d instance segmentation on point clouds. In: Advances in Neural Information Processing Systems (2019) 4
52. Yi, L., Zhao, W., Wang, H., Sung, M., Guibas, L.J.: Gspn: Generative shape proposal network for 3d instance segmentation in point cloud. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2019) 4

53. Yuan, L., Chen, Y., Wang, T., Yu, W., Shi, Y., Jiang, Z., Tay, F.E., Feng, J., Yan, S.: Tokens-to-token vit: Training vision transformers from scratch on imagenet. arXiv preprint arXiv:2101.11986 (2021) 5
54. Zhao, H., Jiang, L., Jia, J., Torr, P.H., Koltun, V.: Point transformer. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (2021) 5
55. Zhao, N., Chua, T.S., Lee, G.H.: Few-shot 3d point cloud semantic segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2021) 5, 9
56. Zhu, X., Su, W., Lu, L., Li, B., Wang, X., Dai, J.: Deformable detr: Deformable transformers for end-to-end object detection. arXiv preprint arXiv:2010.04159 (2020) 5