Prototypical VoteNet for Few-Shot 3D Point Cloud Object Detection

Anonymous Author(s)
Affiliation
Address
email

Abstract

Most existing 3D point cloud object detection approaches heavily rely on large amounts of labeled training data. However, the labeling process is costly and time-consuming. This paper considers few-shot 3D point cloud object detection, where only a few annotated samples of novel classes are needed with abundant samples of base classes. To this end, we propose Prototypical VoteNet to recognize and localize novel instances, which incorporates two new modules: Prototypical Vote Module (PVM) and Prototypical Head Module (PHM). Specifically, as the 3D basic geometric structures can be shared among categories, PVM is designed to leverage class-agnostic geometric prototypes, which are learned from base classes, to refine local features of novel categories. Then PHM is proposed to utilize class prototypes to enhance the global feature of each object, facilitating subsequent object localization and classification, which is trained by the episodic training strategy. To evaluate the model in this new setting, we contribute two new benchmark datasets, FS-ScanNet and FS-SUNRGBD. We conduct extensive experiments to demonstrate the effectiveness of Prototypical VoteNet, and our proposed method shows significant and consistent improvements compared to baselines on two benchmark datasets. Our code, as well as the new benchmark, will be released to facilitate future works.

1 Introduction

3D object detection aims to localize and recognize objects from point clouds with many applications in augmented reality, autonomous driving, and robotics manipulation. Recently, a number of fully supervised 3D object detection approaches have made remarkable progress with deep learning [22, 18, 31, 24]. Nonetheless, their success heavily relies on large amounts of labeled training data, which are time-consuming and costly to obtain. On the contrary, a human can quickly learn to recognize novel classes by seeing only a few samples. To imitate such human ability, we consider few-shot point cloud 3D object detection, which aims to train a model to recognize novel categorizes from limited annotated samples of novel classes together with sufficient annotated data of base classes.

Few-shot learning has been extensively studied in various 2D visual understanding tasks such as object detection [39, 41, 43, 45], image classification [14, 9, 3, 32], and semantic segmentation [23, 21, 48, 19]. Early attempts [9, 16, 11, 38] employ meta-learning to learn transferable knowledge from a collection of tasks and attained remarkable progress. Recently, benefited from large-scale datasets (*e.g.* ImageNet [6]) and advanced pre-training methods [27, 49, 10, 53], finetuning large-scale pre-trained visual models on down-stream few-shot datasets emerges as an effective approach to address this problem [33, 39, 54]. Among different streams of work, prototype-based methods [42, 52, 20, 17] have been incorporated into both streams and show the great advantages, since they can capture the

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

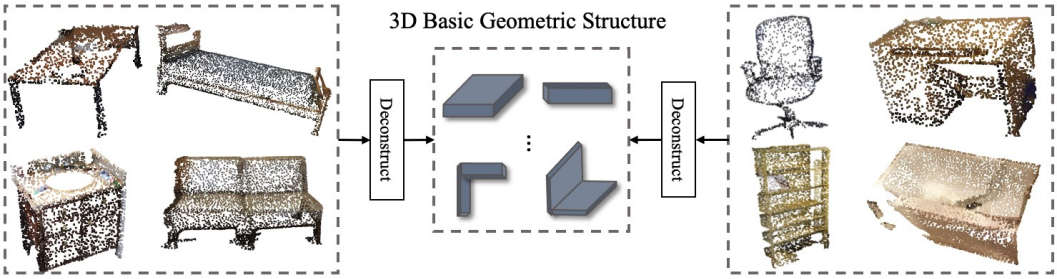

Figure 1: Illustration of the basic geometry of 3D objects, which can be shared among classes.

representative features of categories that can be further utilized for feature refinement [45, 50] or classification [26, 32].

This motivates us to explore effective 3D cues to build prototypes for few-shot 3D detection. Different from 2D visual data, 3D data can get rid of distortions caused by perspective projections, and offer geometric cues with accurate shape and scale information. Besides, 3D primitives to constitute objects can often be shared among different categories. For instance, as shown in Figure 1, rectangular plates and corners can be found in many categories. Based on these observations, in this work, we propose Prototypical VoteNet, which employs such robust 3D shape and primitive clues to design geometric prototypes to facilitate representation learning in the few-shot setting.

Prototypical VoteNet incorporates two new modules, namely Prototypical Vote Module (PVM) and Prototypical Head Module (PHM), to enhance local and global feature learning, respectively, for few-shot 3D object detection. Specifically, based on extracted features from a backbone network (*i.e.* PointNet++ [25]), PVM firstly constructs a class-agnostic 3D primitive memory bank to store geometric prototypes, which are shared by all categories and updated iteratively during training. To exploit the transferability of geometric structures, PVM then incorporates a multi-head attention module to associate geometric prototypes with points in a given scene and utilize them to refine their feature representations. PVM is majorly developed to exploit shared geometric structures among base and novel categories to enhance feature learning of local information in the few-shot setting. Further, to facilitate learning discriminative features for object categorization, PHM is designed to employ a multi-head attention network and leverage class-specific prototypes from a few support samples to refine global representations of objects. Moreover, episodic training [32, 38] is adopted to simulate few-shot circumstances, where PHM is trained by a distribution of similar few-shot tasks instead of only one target object detection task. In this way, supervised by the standard cross-entropy loss, PHM reduces the intra-class variation for the novel categories, with only a few support samples.

Our **contributions** are listed as follows:

- We are the first to study the promising few-shot 3D point cloud object detection task, which allows a model to detect new classes, given a few examples.
- We propose Prototypical VoteNet, which incorporates Prototypical Vote Module and Prototypical Head Module, to address this new challenge. Prototypical Vote Module leverages class-agnostic geometric prototypes to enhance the local features of novel samples. Prototypical Head Module utilizes the class-specific prototypes to refine the object features with the aid of episode training.
- We contribute two new benchmark datasets called FS-ScanNet and FS-SUNRGBD, which are specifically designed for this problem. Our experimental results on these two benchmark datasets show that the proposed model effectively addresses the few-shot 3D point cloud object detection problem, yielding significant improvement over several competitive baseline approaches.

2 Related Work

3D Point Cloud Object Detection. Current 3D point cloud object detection approaches can be divided into two streams: Grid Projection/Voxelization based [46, 15, 35, 4, 51, 40] and point-based [30, 22, 18, 8, 2]. The former projects point cloud to 2D grids or 3D voxels so that the

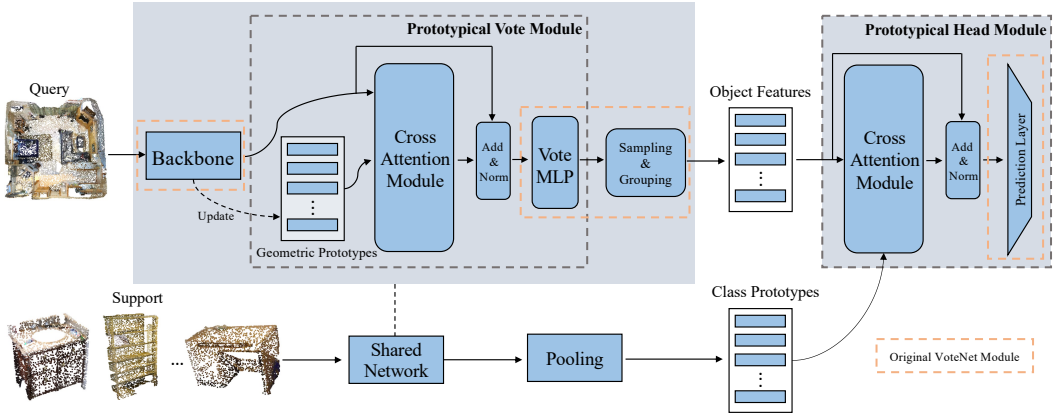

Figure 2: Overview of Prototypical VoteNet. Prototypical VoteNet introduces two new modules: 1) Prototypical Vote Module for enhancing local feature representation of novel samples by leveraging the geometric prototypes, 2) Prototypical Head Module for refining global features of novel objects, utilizing the class-specific prototypes.

advanced convolutional networks can be directly applied. The latter methods take the raw point cloud feature extraction network such as PointNet++ [25] to generate point-wise features for the subsequent detection. Although these fully supervised approaches achieved promising 3D detection performance, their requirement for large amounts of training data precludes their application in many real-world scenarios where training data is costly or hard to acquire. To alleviate this limitation, we explore the direction of few-shot 3D object detection. In this paper, we choose VoteNet [24] as the backbone architecture because of its simplicity and comparable performance in 3D object detection.

Few-Shot Recognition. Few-shot recognition aims to classify novel instances with abundant base samples and a few novel samples. Simple pre-training and finetuning approaches first train the model on the base classes, then finetune the model on the novel categories [3, 7] Meta-learning based methods [9, 16, 11, 38, 32] are proposed to learn classifier across tasks and then transfer to the few-shot classification task. The most related work is Prototypical Network [32], which represents a class as one prototype so that classification can be performed by computing distances to the prototype representation of each class. The above works mainly focus on 2D image understanding. Recently, some few-shot learning approaches for point cloud understanding [29, 50, 47] are proposed. For instance, Sharma *et al.* [50] propose a graph-based method to propagate the knowledge from few-shot samples to the input point cloud. However, there is no work studying few-shot 3D point cloud object detection. In this paper, we first study this problem and introduce the spirit of Prototypical Network into few-shot 3D object detection with 3D geometric prototypes and 3D class-specific prototypes.

2D Few-shot Object Detection. Most existing 2D few-shot detectors employ a meta-learning [41, 14, 45] or fine-tuning based mechanism [44, 43, 26, 36]. Particularly, Kang *et al.* [14] propose a one-stage few-shot detector which contains a meta feature learner and a feature re-weighting module. Meta R-CNN [45] presents meta-learning over RoI (Region-of-Interest) features and incorporates it into Faster R-CNN [28] and Mask R-CNN [11]. TFA [39] reveals that simply fine-tuning the box classifier and regressor outperforms many meta-learning based methods. Cao *et al.* [1] improve the few-shot detection performance by associating each novel class with a well-trained base class based on their semantic similarity.

3 Our Approach

In few-shot 3D point cloud object detection (FSOD), the object class set $\mathbb{C}$ is split into $\mathbb{C}_{\text{base}}$ and $\mathbb{C}_{\text{novel}}$ such that $\mathbb{C} = \mathbb{C}_{\text{base}} \cup \mathbb{C}_{\text{novel}}$ and $\mathbb{C}_{\text{base}} \cap \mathbb{C}_{\text{novel}} = \emptyset$. For each class $r \in \mathbb{C}$, its annotation dataset T_r contains all the data samples with object bounding boxes, that is $T_r = \{(u, P) | u \in \mathbb{R}^6, P \in \mathbb{R}^{N\times 3}\}$. Here, (u, P) is a 3D object bounding box $u = (x, y, z, h, w, l)$, representing box center locations and box dimensions, in a point cloud scene P.

There are only a few examples/shots for each novel class $r \in \mathbb{C}_{\text{novel}}$, which are known as support samples. Besides, there are plenty of annotated samples for each base class $r \in \mathbb{C}_{\text{base}}$. Given the above dataset, FSOD aims to train a model to detect object instances in the novel classes leveraging such sufficient annotations for base categories $\mathbb{C}_{\text{base}}$ and limited annotations for novel categories $\mathbb{C}_{\text{novel}}$.

In the following, we introduce Prototypical VoteNet for few-shot 3D object detection. We will firstly describe VoteNet [24] in Section 3.1 which is the backbone architecture we adopt for 3D object detection. Then, we present Prototypical VoteNet consisting of Prototypical Vote Module (Section 3.2.1) and Prototypical Head Module (Section 3.2.2) to enhance feature learning for few-shot 3D object detection.

3.1 VoteNet Preliminaries

Taking a point cloud scene P_i as input, VoteNet [24] is to localize and categorize 3D objects. As shown in Figure 2, where the orange rectangles represent the original VoteNet modules, it firstly incorporates a 3D backbone network (*i.e.* PointNet++ [25]) parameterized by θ_1 with downsampling layers for point feature extraction as Equation (1).

$$F_i = h_1(P_i; \theta_1), \tag{1}$$

where N and M represent the original and subsampled number of points, respectively, $P_i \in \mathbb{R}^{N \times 3}$ represents an input point cloud scene i, and $F_i \in \mathbb{R}^{M \times (3+d)}$ is the subsampled scene points (also called seeds) with d-dimensional features and 3-dimensional location coordinates.

Then, F_i is fed into a multi-layer perceptron (MLP) with parameters θ_2 (See Figure 2: VoteMLP) which outputs a 3-dimensional coordinate offset $\Delta d_j = (\Delta x_j, \Delta y_j, \Delta z_j)$ relative to its corresponding object center $c = (c_x, c_y, c_z)$ and a residual feature vector Δf_j for each point j in $F_i = \{f_j\}_i$ as in Equation (2).

$$\{\Delta d_j, \Delta f_j\}_i = h_2(F_i; \theta_2). \tag{2}$$

Given the predicted offset Δd_j, the estimated corresponding object center $c_j = (c_{x_j}, c_{y_j}, c_{z_j})$ that point j belongs to can be calculated as Equation (3).

$$c_{x_j} = x_j + \Delta x_j, c_{y_j} = y_j + \Delta y_j, c_{z_j} = z_j + \Delta z_j. \tag{3}$$

Similarly, the point features are updated as $F_i \leftarrow F_i + \Delta F_i$ where $\Delta F_i = \{\Delta f_j\}_i$.

Next, VoteNet samples object centers from $\{(c_{x_j}, c_{y_j}, c_{z_j})\}_i$ using farthest point sampling and group points with nearby centers together (see Figure 2: Sampling & grouping) to form a set of object proposals $O_i = \{o_t\}_i$. Each object proposal is characterized by a feature vector f_{o_t} which is obtained by applying a max pooling operation on features of all points belonging to o_t.

Further, equipped with object features $\{f_{o_t}\}_i$, the prediction layer with parameters θ_3 (see Figure 2: Prediction Layer) is adopted to yield the bounding boxes b_t, objectiveness scores s_t, and classification logits r_t for each object proposal o_t following Equation (4).

$$\{b_t, s_t, r_t\}_i = h_3(\{f_{o_t}\}_i; \theta_3). \tag{4}$$

3.2 Prototypical VoteNet

Here, we present Prototypical VoteNet which incorporates two new designs – Prototypical Vote Module (PVM) and Prototypical Head Module (PHM) to improve feature learning for novel categories with few annotated samples (see Figure 2). Specifically, PVM builds a class-agnostic memory bank of geometric prototypes $\mathcal{G} = \{g_k\}_{k=1}^{K}$ with a size of K, which models transferable class-agnostic 3D primitives learned from rich base categories, and further employs them to enhance local feature representation for novel categories via a multi-head cross-attention module. The enhanced features are then utilized by the Vote MLP module to output the offset of coordinates and features as Equation (2). Second, to facilitate learning discriminative features for novel class prediction, PHM employs an attention-based design to leverage class-specific prototypes $\mathcal{E} = \{e_r\}_{r=1}^{R}$ extracted from the support set D_{support} with R categories to refine global discriminate feature for representing each object proposal (see Figure 2). The output features are fed to the prediction layer for producing results as Equation (4). To make the model more generalizable to novel classes, we exploit a meta-learning strategy, episodic training [32, 38], to train PHM, where a distribution of similar few-shot tasks instead of only one object detection task is learned in the training phase. PVM and PHM are elaborated in the following sections.

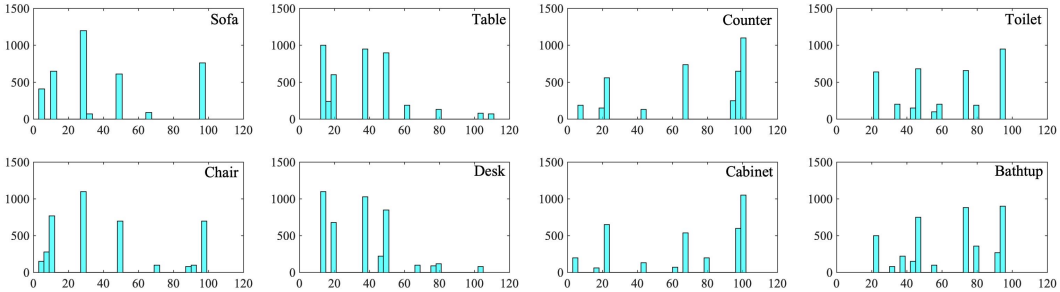

Figure 3: Visualization of the assignment of object point features to the geometric prototypes, where the horizontal axis represents the index of geometric prototypes and the vertical axis represents the number of assignments.

3.2.1 Prototypical Vote Module

Given input features F_i extracted by a backbone network, Prototypical Vote Module constructs class-agnostic geometric prototypes $\mathcal{G} = \{g_k\}_{k=1}^{K}$ and then uses them to enhance local point features with an attention module.

Geometric Prototype Construction. At the beginning, $\mathcal{G} = \{g_k\}_{k=1}^{K} \in \mathbb{R}^{d \times K}$ is randomly initialized. During training, $\mathcal{G}$ is iteratively updated with a momentum based on point features of foreground objects. Specifically, for each update, given $\mathcal{G} = \{g_k\}_{k=1}^{K}$ and all the foreground points $\{p_m\}_{m=1}^{M_f}$ with features $\{f_m\}_{m=1}^{M_f}$, where M_f is the number of foreground points in the current batch, we assign each point to its nearest geometric prototype based on feature space distance. Then, for each prototype g_k, we have a group of points $\{p_m\}_k$ with features represented as $\{f_m\}_k$ assigned to it. Point features in one group are averaged to update the corresponding geometric prototype as Equation (5)

$$g_k \leftarrow \gamma * g_k + (1-\gamma)\overline{f}_k, \text{where } \overline{f}_k = \text{average}(\{f_m\}_k). \tag{5}$$

Here $\gamma \in [0, 1]$ is the momentum coefficient for updating geometric prototypes in a momentum manner, serving as a moving average over all training samples. Since one point feature is related to one geometric prototype, we call this one-hot assignment strategy as hard assignment. An alternative to the hard assignment is the soft assignment, which calculates the similarity between a point features with all geometric prototypes. Empirically, we found that hard assignment results in more effective grouping versus soft assignment. More details can be found in the supplementary file.

Geometric Prototypical Guided Local Feature Enrichment. Given the geometric prototypes $\mathcal{G} = \{g_k\}_{k=1}^{K}$ and point features $F_i = \{f_j\}_i$ of a scene i, PVM further employs a multi-head cross-attention module [37] to refine the point features. Specifically, the multi-head attention network uses the point features $F_i = \{f_j\}_i$ as query, geometric prototypes $\mathcal{G} = \{g_k\}_{k=1}^{K}$ as key and value where linear transformation functions with weights represented as Q_h, U_h, V_h are applied to encode query, key and value respectively. Here, h represents the head index. Then, for each head h, the query point feature is updated by softly aggregating the value features where the soft aggregation weight is determined by the similarity between the query point feature and corresponding key feature. The final point feature f_j is updated by summing over outputs from all heads as Equation (6).

$$f_j \leftarrow \text{Cross_Att}(f_j, \{g_k\}) = \sum_{h=1}^{H} W_h(\sum_{k=1}^{K} A_{j,k}^{h} \cdot V_h g_k), \text{where } A_{j,k}^{h} = \frac{\exp[(Q_h f_j)^T (U_h g_k)]}{\sum_{k=1}^{K} \exp[(Q_h f_j)^T (U_h g_k)]}. \tag{6}$$

Here, $A_{j,k}^{h}$ is the soft aggregation weight considering the similarity between the j-th query point feature and the k-th key feature and used to weight the k-th value feature. Through this process, the point feature is refined using geometric prototypes in a weighted manner where prototypes similar to the query point feature will have higher attention weights. This mechanism transfers geometric prototypes learned from base categories with abundant data to model novel points. The multi-head design enables the model to seek similarity measurements from different angles in a learnable manner

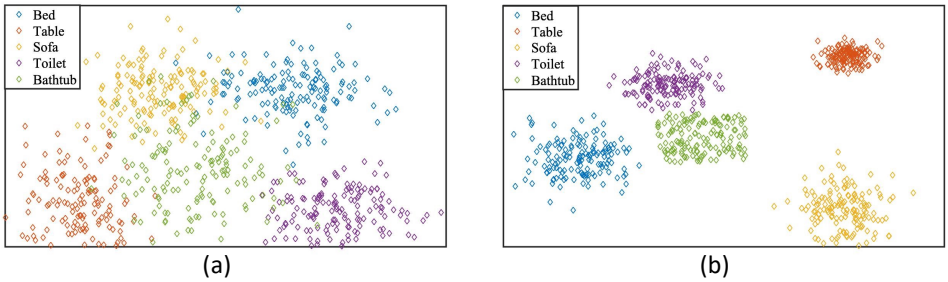

Figure 4: Tsne visualization of the effect of PHM. (a) shows the features without being processed by PHM. (b) shows the features processed by PHM using class-specific prototypes.

to improve robustness. The remaining step of PVM is to feed the refined point features $\{f_j\}_i$ to Vote MLP, which is a multi-layer perceptron, for predicting $\{\Delta d_j, \Delta F_j\}_i$ as stated in Equation (2). In addition, Δd_j is explicitly supervised by a regression loss $\mathcal{L}_{\text{vote}}$ used in [25].

What do the geometric prototypes store? To shed more insights on what the geometric prototypes represent, we visualize the frequency of geometry prototypes in different categories using the "assignment". The number of "assignment"s of object point features to the geometric prototypes is shown in Figure 3, where a point is assigned to the geometric prototype with the highest similarity. In each histogram, the horizontal axis represents the index of geometric prototypes and the vertical axis represents the number of assignments. Note that the first row is the novel classes and the second row is the base classes, which are visually similar to the novel classes in the first row correspondingly. Figure 3 shows that two visually similar categories have a similar assignment histogram since they share the basic geometric structures. This indicates that the memory bank of geometric prototypes successfully learns the 3D basic geometric knowledge, which can be a bridge to transfer the geometric knowledge from base classes to novel ones.

3.2.2 Prototypical Head Module

As shown in Figure 2, given object proposals $O_i = \{o_t\}_i$ with features $\{f_{o_t}\}_i$ from Sampling & Grouping module, PHM module leverages class-specific prototypes $\{e_r\}$ to refine the object features f_{o_t} for the subsequent object classification and localization. Moreover, for better generalizing to novel categories, PHM is trained by the episodic training scheme, where PHM learns a large number of similar few-shot tasks instead of only one task. Considering the function of PHM, we construct the few-shot tasks that, in each iteration, PHM refines the object features $\{f_{o_t}\}_i$ with the aid of class-specific prototypes, which are extracted from randomly sampled support samples. The details are given as follows.

In each few-shot task, class-specific prototypes are built based on support sets that are randomly sampled. For class r, the class-specific prototype e_r is obtained by averaging the instance features for all support samples in class r. The instance feature is derived by applying a max pooling operation over the features of all points belonging to that instance. As shown in Figure 2, with class-specific prototypes $\mathcal{E} = \{e_r\}_{r=1}^{R}$ for a total of R categories, PHM further employs a multi-head cross-attention module to refine object features $\{f_{o_t}\}_i$. Here, the object features $\{f_{o_t}\}_i$ serve as the query features, class-specific prototypes are used to build value features and key features similar as what has been described in Section 3.2.1. Then, the representation f_{o_t} of each proposal o_t is refined using the outputs of the multi-head attention module, which are weighted sum over the value features and the weight is proportionally to the similarity between the query feature and corresponding key features This process can be formulated as Equation (7), which is similar to Equation (6).

$$f_{o_t} \leftarrow \text{Cross_Att}(f_{o_t}, \{e_r\}). \tag{7}$$

Till now, the f_{o_t} is refined using class-specific prototypes, injecting class-specific features from given support samples into object-level features. Finally, the refined object features $\{f_{o_t}\}_i$ are fed into the prediction module to output the objectness score s_t, bounding box locations b_t, and classification scores r_t following Equation (4).

What does PHM help? Figure 4 visualizes the effect of feature refinement by PHM. The experiment is conducted on FS-ScanNet. Figure 4(a) shows the object features, which have not been processed by PHM. Figure 4(b) shows the object features processed by PHM. We could observe that, after the feature refinement by PHM, the features of each classes become more compact compared to the non-PHM one, which further validates the effectiveness of PHM.

3.3 Model Training

The model is trained by a meta-learning strategy, episodic training [32, 38]. The detailed training strategy is included in the supplementary file. We use the standard cross entropy loss $\mathcal{L}_{\text{cls}}$ and smooth-L_1 loss [28] $\mathcal{L}_{\text{reg}}$ to supervise the classification and the bounding box regression, respectively. As for the objectness prediction, if a vote is located either within 0.3 meters to a ground truth object center or more than 0.6 meters from any center, it is considered to be positive [24], which is supervised by a cross entropy loss $\mathcal{L}_{\text{obj}}$. Note that only the positive proposals need to be supervised by $\mathcal{L}_{\text{cls}}$ and $\mathcal{L}_{\text{reg}}$. Therefore, the overall loss for Prototypical VoteNet is given by,

$$\mathcal{L}_{\text{det}} = \mathcal{L}_{\text{cls}} + \alpha_1 \mathcal{L}_{\text{reg}} + \alpha_2 \mathcal{L}_{\text{obj}} + \alpha_3 \mathcal{L}_{\text{vote}}, \tag{8}$$

where $\alpha_1, \alpha_2, \alpha_3$ is the coefficients to balance the loss contributions.

4 Experiments

To our best knowledge, there is no prior study of few-shot point cloud object detection. Therefore, we setup a new benchmark which is described in Section 4.1 & 4.2. Then, we conduct experiments and compare our method with baseline methods in Section 4.3. Third, a series of ablation studies are performed for further analyzing Prototypical VoteNet in Section 4.4. In addition, the implementation details are included in the supplementary file.

4.1 Benchmark Setup

Datasets. We construct two new benchmark datasets FS-SUNRGBD and FS-ScanNet. Specifically, **FS-SUNRGBD** is derived from SUNRGBD [34]. SUNRGBD consists of 5K RGB-D training images annotated, and the standard evaluation protocol reports performance on 10 categories. We randomly select 4 classes as the novel ones while keeping the remaining ones as the base. In the training set, only K annotated bounding boxes for each novel class are given, where k equals 1, 2, 3, 4 and 5. **FS-ScanNet** is derived from ScanNet [5]. ScanNet consists of 1,513 point clouds, and the annotation of the point clouds corresponds to 18 semantic classes plus one for the unannotated space. Out of its 18 object categories, we randomly select 6 classes as the novel ones, while keeping the remaining as the base. We evaluate with 2 different base/novel splits. In the training set, only K annotated bounding boxes for each novel class are given, where k equals 1, 3 and 5. More details about the new benchmark datasets can be referred to the supplementary file.

Evaluation Metrics. We follow the standard evaluation protocol [24] in 3D point cloud object detection by using mean Average Precision(mAP) under different IoU thresholds (*i.e.* 0.25, 0.50), denoted as AP_{25} and AP_{50}. In addition, in the inference stage, we detect both novel classes and base classes. The performance on base classes is included in the supplementary file.

4.2 Benchmark Few-shot 3D Object Detection

We build the first benchmark for few-shot 3D object detection. The benchmark incorporates 4 competitive baselines, and three of them are built upon the vanilla VoteNet detector with few-shot learning strategies, which have been shown to be successful in 2D few-shot object detection.

- VoteNet-joint: The first one is to train the detector on the base and novel classes together. In this way, it can learn good features from the base classes that are applicable for detecting novel classes. We term this baseline as VoteNet-joint.
- VoteNet+TFA [39]: We incorporate a well-designed few-shot object detection method TFA [39] with VoteNet. TFA first trains VoteNet on the training set with abundant samples of base classes. Then only the classifier and the regressor are finetuned with a small balance set containing both base classes and novel classes.

Method	Novel Split 1						Novel Split 2					
	1-shot		3-shot		5-shot		1-shot		3-shot		5-shot	
	AP_{25}	AP_{50}	AP_{25}	AP_{50}	AP_{25}	AP_{50}	AP_{25}	AP_{50}	AP_{25}	AP_{50}	AP_{25}	AP_{50}
VoteNet+joint	9.21	3.14	22.64	9.04	24.93	12.82	4.92	0.94	15.86	3.15	20.72	6.13
VoteNet+TFA	0.48	0.09	8.07	1.03	16.36	7.91	1.00	0.15	2.64	0.22	5.71	2.40
VoteNet+PT+TFA	2.58	1.04	10.37	2.13	17.21	8.94	2.13	0.56	4.85	1.25	7.25	2.49
Meta VoteNet	11.01	4.20	25.73	10.99	26.68	14.40	6.06	1.01	16.93	4.51	23.83	7.17
Ours	**15.34**	**8.25**	**31.25**	**16.01**	**32.25**	**19.52**	**11.01**	**2.21**	**21.14**	**8.39**	**28.52**	**12.35**

Table 1: Results on **FS-ScanNet** using mean Average Precision (mAP) at two different IoU thresholds of 0.25 and 0.50, denoted as AP_{25} and AP_{50}.

Method	1-shot		2-shot		3-shot		4-shot		5-shot	
	AP_{25}	AP_{50}	AP_{25}	AP_{50}	AP_{25}	AP_{50}	AP_{25}	AP_{50}	AP_{25}	AP_{50}
VoteNet+joint	5.46	0.22	6.52	0.77	13.73	2.20	20.47	4.50	22.99	5.90
VoteNet+TFA	1.41	0.03	3.70	0.78	4.03	1.09	7.91	2.10	8.50	2.81
VoteNet+PT+TFA	3.40	0.51	5.13	1.22	7.94	2.31	10.05	3.12	11.32	4.01
Meta VoteNet	7.04	0.98	9.23	1.34	16.24	3.12	20.10	4.69	24.41	6.05
Ours	**12.39**	**1.52**	**14.54**	**3.05**	**21.51**	**6.13**	**24.78**	**7.17**	**29.95**	**8.16**

Table 2: Results on **FS-SUNRGBD** using mean Average Precision (mAP) at two different IoU thresholds of 0.25 and 0.50, denoted as AP_{25} and AP_{50}.

- VoteNet+PT+TFA: The pretraining is proven to be important in 2D few-shot learning, as it learns more generic features, facilitating knowledge transfer from base classes to novel classes. Therefore, we add a pretraining stage, which is borrowed from a self-supervised point cloud contrastive learning method [13], before the training stage of VoteNet+TFA.
- Meta VoteNet: Meta VoteNet is inspired by Meta RCNN [45]. Meta RCNN develops a meta-learning prediction head, which leverages class prototypes to multiply the RoI features and then predicts if a RoI feature is the category that the class prototype belongs to. In VoteNet, the grouped point features after the vote stage can be treated as the RoI features. Therefore, We implement a meta-learning prediction head on VoteNet as a strong baseline.

4.3 Main Results

FS-ScanNet. We first compare the proposed approach with the baselines on FS-ScanNet in Table 1. These results show: 1) VoteNet+TFA has the worst performance. For example, it only achieves 8.07% AP_{25} and 1.03% AP_{50} on 3-shot in Novel Split-1. The reason is that VoteNet+TFA is trained from scratch, and only a few layers are funetuned by novel samples. Therefore, it tends to be overfitting with poor generalization ability on novel classes. 2) Even adding a pretaining stage to VoteNet+TFA, which is termed as VoteNet+PT+TFA, it still performs poorly. For example, it only contributes the improvement of +2.30% AP_{25} and +1.10% AP_{50} to VoteNet+TFA on 3-shot in Novel Split-1. This is different from 2D few-shot object detection. This is potentially caused by the lack of large-scale datasets for pre-training in 3D, while the 2D backbone ResNet [12] has a dataset of over one million images for pre-training. 3) Besides our designed method, Meta VoteNet is superior. For example, Meta VoteNet outperforms VoteNet+joint by +3.09% AP_{25} and +1.95% AP_{50} on 3-shot in Novel Split-1. This shows the meta-learning prediction head, derived from Meta RCNN [45], helps to generalize from base classes to novel classes. 4) Our new model Prototypical VoteNet surpasses all competitors by significant margins. For example, Prototypical VoteNet achieves 32.25% AP_{25} and 16.01% AP_{50} on 3-shot in Novel Split-1. This is because our proposed method is equipped with a more generic vote module by learning geometric prototypes, and leverage class prototypes and episode learning scheme to promote the discriminative feature learning.

FS-SUNRGBD. Table 2 shows the result comparison on FS-SUNRGBD. In FS-SUNRGBD, we find that VoteNet+TFA and VoteNet+PT+TFA still have a large gap with other methods due to the overfitting and insufficient pretraining. Meanwhile, Meta VoteNet outperforms other baseline methods. For example, on 3-shot, Meta VoteNet surpasses VoteNet+PT+TFA by +8.34% AP_{25} and +0.81% AP_{50} and VoteNet+joint by +2.51% AP_{25} and +0.92% AP_{50}. Finally, our designed Prototypical VoteNet outperforms all the methods. For instance, on 3-shot, Prototypical VoteNet

surpasses Meta VoteNet by +5.37% AP_{25} and +3.01% AP_{50}, which further validates the effectiveness of our proposed PVM and PHM.

4.4 Ablation Study

Method	3-shot		5-shot	
	AP_{25}	AP_{50}	AP_{25}	AP_{50}
Baseline	22.64	9.04	24.93	12.82
+PVM	27.43	13.63	28.44	16.45
+PHM	28.76	14.04	30.13	17.51
+PVM+PHM	**31.25**	**16.01**	**32.25**	**19.52**

Table 3: Ablation study of individual components.

	Prototype	$AP_{0.25}$	$AP_{0.50}$
PVM	Geometric	31.25	16.01
	Self-learning	28.34	14.01
PHM	Class	31.25	16.01
	Self-learning	27.45	13.67

Table 4: Ablation study of Prototypes.

Contributions of individual components. We conduct ablation studies on 3-shot and 5-shot in split-1 of FS-ScanNet with results shown in Table 3. The results show that both of them are effective on their own. For example, on 3-shot, PVM contributes the improvement of +4.19% AP_{25} and +4.59% AP_{50}, and PHM contributes the improvement of +6.12% AP_{25} and +5.00% AP_{50}. Moreover, when combined, the best performance, 31.25% AP_{25} and 16.01% AP_{50}, is achieved.

# Prototype	$AP_{0.25}$	$AP_{0.50}$
$K = 30$	29.98	15.01
$K = 60$	30.24	15.54
$K = 90$	31.10	15.98
$K = 120$	31.25	16.01
$K = 150$	31.01	15.89

Table 5: Ablation study of memory bank size.

Coefficient m	$AP_{0.25}$	$AP_{0.50}$
$\gamma = 0.2$	29.50	14.65
$\gamma = 0.9$	30.55	15.15
$\gamma = 0.99$	30.71	15.30
$\gamma = 0.999$	31.01	15.89
$\gamma = 0.9999$	30.90	15.79

Table 6: Ablation study of coefficient γ.

Effectiveness of Prototypes. Table 4 shows the effectiveness of two kinds of prototypes. Specifically, in order to validate the geometric prototypes, we first displace them by the self-learning ones, which are randomly initialized and updated by the gradient descend during the model training. The experiment results show that the performance significantly degrades. To validate the effectiveness of class prototypes, we also alter them by the randomly initialized self-learning prototypes. Similarly, the performance drops drastically due to the lack of class prototypes.

Size of Memory Bank. Table 5 reports the performance of our approach with regard to the size of the memory bank containing the geometric prototypes. This ablation study is performed on 3-shot in split-1 of FS-ScanNet. For K $= 30$, the memory bank only contains 30 geometric prototypes, which only achieves 29.98% AP_{25} and 15.01% AP_{50}. Further, when using more prototypes (i.e., K $= 120$), we see a clear performance improvement, which reaches 31.25% AP_{25} and 16.01% AP_{50}. However, when continuous increasing K, there will be no improvement.

Coefficient γ. Table 6 shows the effect of momentum coefficient (γ in Equation (5)). The experiment is performed on 3-shot in split 1 of FS-ScanNet. The results show that, when using a relatively large coefficient (*i.e.*, $\gamma \in [0.999, 0.9999]$), the model performs well, compared with the model using a small momentum coefficient (*i.e.*, $\gamma \in [0.9, 0.99]$). Moreover, the performance drops considerably at the extreme case of $\gamma = 0.2$. This indicates that a slow updating process is beneficial.

5 Conclusion

We have presented Prototypical VoteNet for few-shot 3D object detection along with a new benchmark for evaluation. Prototypical VoteNet enjoys the advantages of two new designs, namely Prototypical Vote Module (PVM) and Prototypical Head Module (PHM), for enhancing feature learning in the few-shot setting. Specifically, PVM exploits geometric prototypes learned from base categories to refine local features of novel categories. PHM is proposed to utilize class-specific prototypes to promote discriminativeness of object-level features. Extensive experiments on two new benchmark datasets demonstrate the superiority of our approach. We hope our studies on 3D propotypes and proposed new benchmark could inspire further investigations in 3D few-shot learning.

References

[1] Yuhang Cao, Jiaqi Wang, Ying Jin, Tong Wu, Kai Chen, Ziwei Liu, and Dahua Lin. Few-shot object detection via association and discrimination. In *Advances in Neural Information Processing Systems*, 2021.

[2] Jintai Chen, Biwen Lei, Qingyu Song, Haochao Ying, Danny Z. Chen, and Jian Wu. A hierarchical graph network for 3d object detection on point clouds. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[3] Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang, and Huang Jia-Bin. A closer look at few-shot classification. In *International Conference on Learning Representations*, 2019.

[4] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3d object detection network for autonomous driving. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[5] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[6] Jia Deng, Dong Wei, Socher Richard, Li Li-Jia, Li Kai, and Fei-Fei Li. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[7] Guneet S Dhillon, Pratik Chaudhari, Avinash Ravichandran, and Stefano Soatto. A baseline for few-shot image classification. *arXiv preprint arXiv:1912.03083*, 2019.

[8] Francis Engelmann, Martin Bokeloh, Alireza Fathi, Bastian Leibe, and Matthias Niessner. 3d-mpa: Multi-proposal aggregation for 3d semantic instance segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[9] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, 2017.

[10] Peng Gao, Shijie Geng, Renrui Zhang, Teli Ma, Rongyao Fang, Yongfeng Zhang, Hongsheng Li, and Yu Qiao. Clip-adapter: Better vision-language models with feature adapters. *arXiv preprint arXiv:2110.04544*, 2021.

[11] Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick. Mask r-cnn. In *IEEE International Conference on Computer Vision*, 2017.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[13] Ji Hou, Benjamin Graham, Matthias Niessner, and Saining Xie. Exploring data-efficient 3d scene understanding with contrastive scene contexts. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2021.

[14] Bingyi Kang, Zhuang Liu, Xin Wang, Fisher Yu, Jiashi Feng, and Trevor Darrell. Few-shot object detection via feature re-weighting. In *IEEE International Conference on Computer Vision*, 2019.

[15] Alex H. Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[16] Kwonjoon Lee, Subhransu Maji, Avinash Ravichandran, and Stefano Soatto. Meta-learning with differentiable convex optimization. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[17] Jinlu Liu, Liang Song, and Yongqiang Qin. Prototype rectification for few-shot learning. In *European Conference on Computer Vision*, 2022.

[18] Ze Liu, Zheng Zhang, Yue Cao, Han Hu, and Xin Tong. Group-free 3d object detection via transformers. In *IEEE International Conference on Computer Vision*, 2021.

[19] Zhihe Lu, Sen He, Xiatian Zhu, Li Zhang, Yi-Zhe Song, and Tao Xiang. Simpler is better: Few-shot semantic segmentation with classifier weight transformer. In *IEEE International Conference on Computer Vision*, 2021.

[20] Jiawei Ma, Hanchen Xie, Guangxing Han, Shih-Fu Chang, Aram Galstyan, and Wael Abd-Almageed. Partner-assisted learning for few-shot image classification. In *IEEE International Conference on Computer Vision*, 2021.

[21] Juhong Min, Dahyun Kang, and Minsu Cho. Hypercorrelation squeeze for few-shot segmentation. In *IEEE International Conference on Computer Vision*, 2021.

[22] Ishan Misra, Rohit Girdhar, and Armand Joulin. An end-to-end transformer model for 3d object detection. In *IEEE International Conference on Computer Vision*, 2021.

[23] Khoi Nguyen and Sinisa Todorovic. Feature weighting and boosting for few-shot segmentation. In *IEEE International Conference on Computer Vision*, 2019.

[24] Charles R. Qi, Or Litany, Kaiming He, and Leonidas J. Guibas. Deep hough voting for 3d object detection in point clouds. In *IEEE International Conference on Computer Vision*, 2019.

[25] 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*, 2017.

[26] Limeng Qiao, Yuxuan Zhao, Zhiyuan Li, Xi Qiu, Jianan Wu, and Chi Zhang. Defrcn: Decoupled faster r-cnn for few-shot object detection. In *IEEE International Conference on Computer Vision*, 2021.

[27] 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. *arXiv preprint arXiv:2103.00020*, 2021.

[28] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, 2015.

[29] Charu Sharma and Manohar Kaul. Self-supervised few-shot learning on point clouds. In *Advances in Neural Information Processing Systems*, 2020.

[30] Shaoshuai Shi, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[31] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[32] Jake Snell, Kevin Swersky, and Richard S Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, 2017.

[33] Haoyu Song, Li Dong, Wei-Nan Zhang, Ting Liu, and Furu Wei. Clip models are few-shot learners: Empirical studies on vqa and visual entailment. In *Association for Computational Linguistics*, 2022.

[34] Shuran Song, Samuel P. Lichtenberg, and Jianxiong Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[35] Shuran Song and Jianxiong Xiao. Deep sliding shapes for amodal 3d object detection in rgb-d images. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[36] Bo Sun, Banghuai Li, Shengcai Cai, Ye Yuan, and Chi Zhang. Fsce: Few-shot object detection via contrastive proposal encoding. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2021.

[37] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.

[38] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*, 2016.

[39] Xin Wang, Thomas E Huang, Trevor Darrell, Joseph E Gonzalez, and Fisher Yu. Frustratingly simple few-shot object detection. In *International Conference on Machine Learning*, 2020.

[40] Yue Wang, Alireza Fathi, Abhijit Kundu, David A. Ross, Caroline Pantofaru, Tom Funkhouser, and Justin Solomon. Pillar-based object detection for autonomous driving. In *European Conference on Computer Vision*, 2020.

[41] Yu-Xiong Wang, Deva Ramanan, and Martial Hebert. Meta-learning to detect rare objects. In *IEEE International Conference on Computer Vision*, 2019.

[42] Aming Wu, Yahong Han, Linchao Zhu, and Yi Yang. Universal-prototype enhancing for few-shot object detection. In *IEEE International Conference on Computer Vision*, 2021.

[43] Aming Wu, Suqi Zhao, Cheng Deng, and Wei Liu. Generalized and discriminative few-shot object detection via svd-dictionary enhancement. In *Advances in Neural Information Processing Systems*, 2021.

[44] Jiaxi Wu, Songtao Liu, Di Huang, and Yunhong Wang. Multi-scale positive sample refinement for few-shot object detection. In *European Conference on Computer Vision*, 2020.

[45] Xiaopeng Yan, Ziliang Chen, Anni Xu, Xiaoxi Wang, Xiaodan Liang, and Liang Lin. Meta r-cnn: Towards general solver for instance-level low-shot learning. In *IEEE International Conference on Computer Vision*, 2019.

[46] Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. *Sensors*, 2018.

[47] Chuangguan Ye, Hongyuan Zhu, Yongbin Liao, Yanggang Zhang, Tao Chen, and Jiayuan Fan. What makes for effective few-shot point cloud classification? In *IEEE Winter Conference on Applications of Computer Vision*, 2022.
[48] Bingfeng Zhang, Jimin Xiao, and Terry Qin. Self-guided and cross-guided learning for few-shot segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2021.
[49] Renrui Zhang, Rongyao Fang, Wei Zhang, Peng Gao, Kunchang Li, Jifeng Dai, Yu Qiao, and Hongsheng Li. Tip-adapter: Training-free clip-adapter for better vision-language modeling. *arXiv preprint arXiv:2111.03930*, 2021.
[50] Na Zhao, Tat-Seng Chua, and Gim Hee Lee. Few-shot 3d point cloud semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2021.
[51] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.
[52] Kai Zhu, Yang Cao, Wei Zhai, Jie Cheng, and Zheng-Jun Zha. Self-promoted prototype refinement for few-shot class-incremental learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2021.
[53] Kaiyang Zhu, Jingkang Yang, Hao Li, Change Loy Chen, Xiaogang Wang, and Ziwei Liu. Conditional prompt learning for vision-language models. 2022.
[54] Xizhou Zhu, Jinguo Zhu, Hao Li, Xiaoshi Wu, Xiaogang Wang, Hongsheng Li, Wang Xiaohua, and Jifeng Dai. Uni-perceiver: Pre-training unified architecture for generic perception for zero-shot and few-shot tasks. 2022.

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes] See Section **??**.
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [N/A]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]