Meta-Det3D: Learn to Learn Few-Shot 3D Object Detection

Shuaihang Yuan*, Xiang Li*, Hao Huang, and Yi Fang

NYU Multimedia and Visual Computing Lab, USA
NYUAD Center for Artificial Intelligence and Robotics (CAIR), Abu Dhabi, UAE
NYU Tandon School of Engineering, New York University, USA
New York University Abu Dhabi, UAE
{sy2366,xl1845,hh1811,yfang}@nyu.edu

Abstract. This paper addresses the problem of few-shot indoor 3D object detection by proposing a meta-learning-based framework that only relies on a few labeled samples from novel classes for training. Our model has two major components: a *3D meta-detector* and a *3D object detector*. Given a query 3D point cloud and a few support samples, the 3D meta-detector is trained over different 3D detection tasks to learn task distributions for different object classes and dynamically adapt the 3D object detector to complete a specific detection task. The 3D object detector takes task-specific information as input and produces 3D object detection results for the query point cloud. Specifically, the 3D object detector first extracts object candidates and their features from the query point cloud using a point feature learning network. Then, a class-specific re-weighting module generates class-specific re-weighting vectors from the support samples to characterize the task information, one for each distinct object class. Each re-weighting vector performs channel-wise attention to the candidate features to re-calibrate the query object features, adapting them to detect objects of the same classes. Finally, the adapted features are fed into a detection head to predict classification scores and bounding boxes for novel objects in the query point cloud. Several experiments on two 3D object detection benchmark datasets demonstrate that our proposed method acquired the ability to detect 3D objects in the few-shot setting.

Keywords: 3D object detection · indoor scene · few-shot learning · meta-learning · channel-wise attention.

1 Introduction

In door 3D object detection is one of the key components in various real-world vision applications, such as indoor navigation [1] and visual SLAM [2], and has been attracting numerous research attention in the last decades [3, 4]. Recent progress in deep-learning-based methods has greatly advanced the performance of 3D object detection. However, existing deep-learning-based 3D object detection methods require ground truth supervision. When only a limited number of training data is provided, existing deep-learning-based methods suffer from severe performance degradation. In contrast, humans can

* equal contribution.

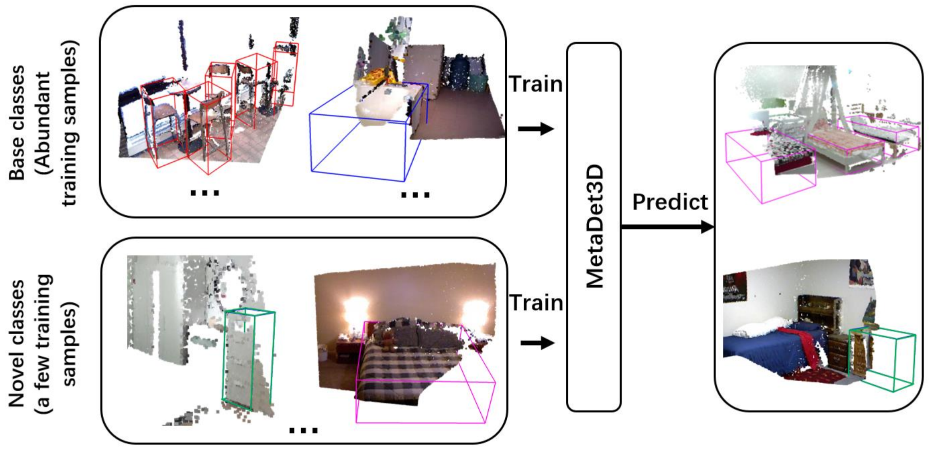

Base classes (Abundant training samples)
Novel classes (a few training samples)
Train
Train
MetaDet3D
Predict

Fig. 1. Given abundant annotated samples of base classes and a few annotated samples of novel classes, our proposed model aims to learn from limited novel class samples to detect unseen 3D objects from the novel classes. (Top) Annotated chairs and sofas in base classes. (Bottom) Beds in the novel classes. Note that the objects are incomplete in the scanned 3D point cloud scenes.

easily learn new concepts and identify new objects with only a few given examples. This observation motivates us to design a novel few-shot 3D object detection framework that leverages the abundant annotated base classes to quickly learn new patterns or concepts from only a few given samples from unseen or novel classes.

Here, we focus on the problem of indoor 3D object detection under the few-shot setting. Specifically, given a sufficient labeled training sample from the base classes and a few labeled samples from the novel classes, our model learns to detect an object from the novel classes. To achieve this goal, we design our detection model based on meta-learning [5, 6]. 3D object detection requires predicting both object category and localization. However, 3D point clouds are unordered and unstructured in nature, which makes it hard to adopt a neural network to learn robust and representative point features. Therefore, 3D object detection under a few-shot scenario is even more challenging. Despite that there are a few previous literatures focusing on meta-learning for 3D point cloud classification [7–9] and segmentation [10, 11], meta-learning for few-shot 3D point cloud detection has seldom been explored, and this work paves a path for few-shot 3D point cloud detection using the meta-learning scheme.

Specifically, our proposed method adopts a meta-learning-based paradigm for few-shot 3D object detection based on 3D object detectors, *e.g.*, VoteNet [4]. In VoteNet, a task-specific 3D object detector is designed for object detection with massive training data and thus is inapplicable in few-shot scenarios. To address the problem of 3D object detection in the few-shot scenario, we design a *3D meta-detector* to guide a *3D object detector* to perform few-shot 3D object detection. Specifically, we introduce a class-specific re-weighting module as the 3D meta-detector to guide the 3D object detector to detect objects from a specific class. Given a query 3D point cloud scene and a few support objects from novel classes, the backbone network of the 3D object detector is first adopted to extract candidates and their features from the input scene. Then, the class-specific re-weighting module receives few-shot objects from novel classes with bounding box annotations and produces re-weighting vectors, one for each novel class. Each vector takes channel-wise attention to all features to re-weight and re-calibrate the query object features. In this manner, the candidate features fuse with support information from novel classes and are adapted to detect objects of the same classes. Finally, the adapted features are fed into a detection head to predict classification scores and bounding boxes for novel objects in the query point cloud scene. We validate our method on two public indoor 3D object detection datasets. Experimental results demonstrate that our proposed meta-learning-based 3D few-shot detection method outperforms several baselines. The contributions of our paper are summarized as follows.

- To the best of our knowledge, we are the first to tackle the problem of few-shot 3D object detection using meta-learning, which is of great importance in real-world applications and has seldom been explored in previous literature. Our method learns meta-knowledge through the training process on base classes and transfers the learned meta-knowledge to unseen classes with only a few labeled samples.
- We design a novel few-shot 3D detection framework that consists of a meta-detector to guide a 3D object detector for object detection. To effectively guide the 3D detector to complete a specific task, we propose a class-specific re-weighting module as the 3D meta-detector that receives a few support samples from novel classes and produces class-attentive re-weighting vectors. These re-weighting vectors are used to re-calibrate the cluster features of the same class and strengthen those informative features for detecting novel objects in the query point cloud scene.
- We demonstrate the effectiveness of our proposed model on two 3D object detection benchmark datasets, and our model achieves superior performance than the well-established baseline methods by a large margin.

2 Related Work

2.1 3D Object Detection

Many existing works have been proposed to tackle the task of object detection from 3D point clouds. In this section, we focus on reviewing indoor 3D object detection. A 3D scene can be represented by three-dimensional point clouds. One intuitive way to detect 3D objects from a 3D scene is to employ a 3D deep learning network for object detection. However, this approach requires exhaustive computation, which slows the inference speed when handling large indoor scenes. Moreover, in 3D indoor datasets [12, 13], the complex configuration of the layout of indoor 3D objects also introduces challenges. Various learning-based methods have been proposed to address the challenges above. Those learning-based indoor 3D object detection methods can be generally categorized into three classes. The first class is the sliding-window-based method, which divides the whole 3D scene into multiple patches and further adopts a classifier to detect 3D objects. The second type of indoor 3D object detection method extracts the per-point feature in a latent space and clusters corresponding points to produce the detection results. Qi *et al.* [4] propose a method that generates seed points from the input

3D point cloud and then predicts votes from the generated seed points. Then, Qi *et al.* cluster the votes by gathering the corresponding features and predicting 3D bounding boxes. The third type of 3D indoor detection method is proposal-based. Vote3D [14] uses a grid and predicts bounding boxes to detect 3D indoor objects. In addition, [15] uses MLP to directly predict the 3D bounding boxes from extracted point cloud features. Although the aforementioned methods achieve promising results in various 3D object detection benchmarks, they require the ground truth label as supervision during the training to ensure the model's generalization ability. However, this is not practical in real-world applications. To maintain the model's generalization ability for unseen object categories with only a limited amount of training data, *a.k.a.*, few-shot scenario, we design a meta-learning-based method for 3D object detection for point clouds.

2.2 Meta-learning

Meta-learning aims to handle the scenario where only a few labeled training data are available. Meta-learning can also be categorized into three classes as follows.
Metric-based methods. Directly training deep neural networks with a few training data leads the networks to over-fit on the training data. To avoid this issue, metric-learning methods adopt a non-parametric distance measurement as a classifier to directly compare the features of input data and predict labels. In addition, a parametric network is often adopted to extract representative and discriminative features that have large inter-class variations and small intra-class variations. Typical metric-based frameworks include Siamese Networks [16], Matching Networks [17], Prototypical Networks [18], and Relation Networks [19].
Model-based methods. Distinct from metric-based methods, model-based methods [20, 21] adopt two different parametric networks (often termed as *learners*) where the first one serves as a meta-learner and the second one works as a classifier. The classifier only needs a single feed-forward pass to predict data labels, and its parameters are directly estimated by the meta-learner. Model-based methods only require to calculate gradients of the meta-learner for back-propagation, yielding efficient learning for few-shot problems. However, the generalization ability of the model-based methods on out-of-distribution data is not satisfactory.
Gradient-based methods. The key idea of gradient-based methods is to train a meta-learner to learn meta-knowledge from a dataset, thus to help a classifier to achieve a quick convergence on a few labeled data from unseen classes. In the pioneer work MAML [22], a meta-learner is used to learn a proper parameter initialization of a classifier from training data such that after a small number of back-propagation steps on the classifier, it can achieve good performance on few-shot tasks with limited labeled data. In addition to learning network parameter initialization, this method can also be utilized to update network hyper-parameters [23].

3 Review of VoteNet

We give a brief review of VoteNet [4] that is incorporated as the building block of our proposed method. VoteNet is a 3D object detection model that takes raw 3D point clouds as input and produces 3D bounding boxes without using 2D detectors. The object detection process in VoteNet can be divided into three steps. The first step is to generate seed points from the input 3D point cloud; the second step is to predict votes from the generated seed points, and the final step is to cluster votes by gathering their features and predicting 3D box proposals for each object.

To generate accurate votes for 3D object detection, VoteNet leverage PointNet++ [24], a hierarchical point signature learning model, as the backbone network to learn representative point features for geometric reasoning. The backbone network takes raw point clouds as input and produces a subset of the input point clouds with the coordinates of each point and the corresponding feature vector. Each output point is called a *seed* point and will produce one vote. Inspired by the generalized Hough voting for object detection [25], VoteNet leverages the power of a deep neural network to generate votes from seed points. Given the seed set $\{seed_i\}_{i=1}^{P}$, consisting P seed points, where each $seed_i$ is represented by the 3D coordinate x_i and feature vector f_i, a shared voting module is used to generate votes from each seed point independently. The voting module is simply a multi-layer perceptron (MLP). The MLP takes $[x_i, f_i]$ as input and outputs the displacement of each seed point in Euclidean space as well as the feature offset in feature space represented as $[\Delta x_i, \Delta f_i]$. The final vote is then represented as $v_i = [y_i = x_i + \Delta x_i, g_i = f_i + \Delta f_i]$. This voting process is supervised by the ℓ_2 distance that penalizes the offsets between the predicted and ground-truth distances from the seed points on the object surface to the object centers. During training, the MLP learns to drive the seed points to move towards the associating object centers and making it easier for the following vote clustering process.

Then, VoteNet adopts farthest point sampling (FPS) [24] to select Q votes based on the vote coordinates in Euclidean space and group votes by finding the nearest votes to the cluster centers. A cluster $\{C_q\}_{q=1}^{Q}$ is represented by $[y_{qi}, g_{qi}]$ where y_{qi} is the 3D coordinate of the i^{th} vote in this cluster and g_{qi} is the corresponding feature vector. After that, each cluster is transformed into a local coordinate by subtracting each vote's coordinate from its associating cluster center. Finally, a detection head based on PointNet [26] is introduced to generate 3D bounding boxes from each of the vote clusters. The head network takes the normalized vote clusters as input and predicts the bounding boxes and classification scores. However, directly applying VoteNet in few-shot scenarios yields inferior performance as VoteNet requires a large amount of labeled data for training. In the next section, we describe our model utilizing VoteNet to overcome the challenges of few-shot 3D object detection with limited training labels.

4 Method

4.1 Few-shot 3D Object Detection

In this section, we introduce the basic setting of the meta-learning-based few-shot 3D object detection. Different from conventional 3D object detection, in the few-shot scenario, the whole dataset $\mathscr{D}$ is separated into two parts, *i.e.*, base classes $\mathscr{D}_{base}$ and novel classes $\mathscr{D}_{novel}$ where $\mathscr{D}_{base} \cap \mathscr{D}_{novel} = \emptyset$. For each of the object categories from the base classes, abundant labeled data is available and $\mathscr{D}_{base}$ is divided into training and

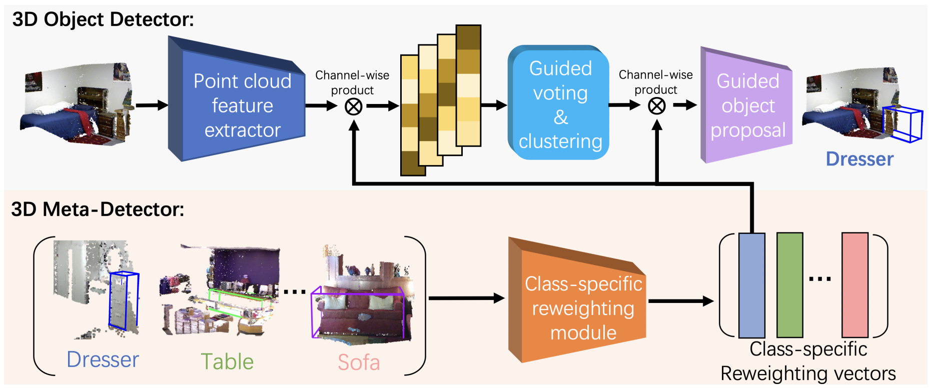

3D Object Detector:
Point cloud feature extractor
Channel-wise product
Guided voting & clustering
Channel-wise product
Guided object proposal
Dresser
3D Meta-Detector:
Dresser
Table
Sofa
Class-specific reweighting module
Class-specific Reweighting vectors

Fig. 2. Our MetaDet3D contains two components. The first component (bottom) is the 3D meta-detector which provides task/class-specific object detection guidance through a group of class-specific re-weighting vectors. The second component (top) is the 3D object detector that takes class-specific re-weighting vectors as input and predicts object bounding boxes and labels using three sub-components. A detailed description of each component and the three sub-components are depicted in section 4.3.

testing set $\mathcal{D}_{base}^{train}$ and $\mathcal{D}_{base}^{test}$. Similarly, $\mathscr{D}_{novel}$ is also divided into training and testing sets $\mathcal{D}_{novel}^{train}$ or $\mathcal{D}_{novel}^{test}$. However, in the few-shot setting, only a few labeled samples from the novel class can be used for training. Few-shot 3D object detection aims to predict 3D bounding boxes and class labels for each object from the *novel* classes with a few labeled data during training.

To facilitate the training and evaluation of a few-shot 3D object detection model, we construct several episodes from the training and testing set for $\mathscr{D}_{base}$ and $\mathscr{D}_{novel}$ following the settings proposed in [27]. Each episode $\mathcal{E}$ is constructed from a set of support point clouds $\mathbf{S}$ with annotations and a set of query point clouds $\mathbf{Q}$. In a N-*way*-K-*shot* setting, the support set $\mathbf{S}$ has K instances with bounding box annotations for each of the N classes, *i.e.*, $\mathbf{S} = \{(S_{n,k}, r_{n,k})\}$, where $c = 1, 2, ..., N$ and $k = 1, 2, ..., K$, $S_{n,k}$ denotes the input point cloud, and $r_{n,k}$ denotes the bounding box annotation of the support set instances. We denote the query set as $\mathbf{Q}$, which contains N_q instances from the object classes that are the same as the support set. Our few-shot detection model aims to perform 3D object detection over the query point clouds with the support set as guidance.

Such a few-shot 3D object detection setting is practical in real-world scenarios. One may need to develop an object detection model, but collecting a large-scale well-annotated dataset for the target classes is time-consuming. An alternative would be deploying a detection model pre-trained on some existing large-scale object detection datasets (*e.g.*, Microsoft COCO [28] and SUN RGB-D [13]). However, one may only hope to detect several specific object categories of which these datasets only cover a very limited number of samples. These real-world applications pose a challenge and demand for few-shot object detection models.

In the following sections, we introduce our novel design of a few-shot 3D object detection framework, *MetaDet3D*, short for **Meta-Det**ection-**3D**. The illustration of our model is shown in Figure 2. Our detection process is divided into two components: 1) a 3D meta-detector, and 2) a 3D object detector. We treat the object detection problem for each class as a learning task. 3D meta-detector is designed to learn the class-specific task distribution over different tasks and dynamically adapt the 3D object detector to complete a specific detection task. The 3D meta-detector learns class-specific re-weighting vectors that are used to guide the 3D object detector. The 3D object detector aims to complete a specific 3D object detection task by leveraging the class-specific information from the 3D meta-detector to detect objects from the same classes. The details of the proposed 3D meta-detector and 3D object detector are described as follows.

4.2 3D Meta-Detector

In this section, we introduce a novel 3D meta-detector, which is designed to learn the class-specific task distribution from different 3D object detection tasks.

We propose a class-specific re-weighting module $\mathcal{M}$ to produce the guidance for the 3D object detector. The 3D meta-detector takes 3D point cloud objects of different classes as input and learns a group of compact and robust re-weighting vectors in the embedding space. Formally, we first crop support object instances from support set $\mathbf{S}$ by using the ground-truth bounding box $r_{n,k}$ to construct supporting instance set $\mathbf{I} = \{I_{n,k}\}$, where $I_{n,k}$ denotes the k^{th} cropped instance from the n^{th} class.

As deep neural networks have been proved effective and efficient in 3D feature learning, we adopt PointNet++ [24] to produce a set of compact re-weighting vectors, one for each class. Specifically, given $N \times K$ support instances, our re-weighting module $\mathcal{M}$ generates generates N re-weighting vectors $\{z_n\}_{n=1}^{N}$ where $z_n \in \mathbb{R}^W$, one for each class. This step is formulated as:

$$z_n = \frac{1}{K} \sum \mathcal{M}_\theta(I_{n,k}), \quad n = 1, 2, ..., N \ . \tag{1}$$

The output z_n represents the re-weighting vector for the n^{th} class. We use θ to represent the network parameters of our re-weighting module $\mathcal{M}$. The generated z_n is responsible for tuning object features in the 3D object detector for an accurate object candidate generation for the corresponding n^{th} class, which is detailed in the following section.

4.3 3D Object Detector

In this section, we introduce a 3D object detector that is used to complete a 3D object detection task for the given query set $\mathbf{Q}$ with the guidance provided by the 3D meta-detector. Inspired by the previous 3D object detection works [4, 3, 29], the 3D object detector can be divided into three steps. The first step is to extract point features from the input 3D point cloud. The second step is to generate object candidates. The final step is to predict 3D box proposals for each object.

Point Feature Extraction. To extract point features from a point cloud, we leverage PointNet++ [24], a hierarchical point feature learning model, as the backbone network to learn representative point features for geometric reasoning. The backbone network takes raw point clouds as input and produces a subset of the input point clouds with the coordinates of each point and the corresponding feature vectors. Each output point is called a *seed* and will produce one vote to generate a candidate. Specifically, for a given query scan Q_i from $\mathbf{Q}$, the network (denotes as $\mathcal{F}$) outputs seeds represented by a combination of seeds' coordinates and point features, *i.e.*, $[x, f]$ where $x \in \mathbb{R}^3$ and $f \in \mathbb{R}^W$:

$$[x, f] = \mathcal{F}_\delta(q_i) \ , \tag{2}$$

where δ denotes the network parameters.

Guided Voting & Clustering. Inspired by the voting-based 3D object detection works [4, 3, 29], we leverage the power of a deep neural network to generate candidates from seeds. Given a group of seeds, we apply channel-wise multiplication $\otimes$ between seeds' feature f and the learned class-specific re-weighting vectors z_n to generate re-weighted feature$f^{'}$ where $f^{'} = z_n \otimes f$.

A shared voting module is then utilized to generate candidates from the seeds. The module is implemented as a multi-layer perceptron (MLP) network with ϕ to denote the network parameters. The MLP takes $[x, f^{'}]$ as input and outputs the displacement of the seeds in Euclidean space as well as the feature offset in feature space. The guided voting for object candidates generation is formulated as:

$$[\varDelta x, \varDelta f^{'}] = MLP_\phi(f^{'}). \tag{3}$$

$$[y, g] = [x + \varDelta x, f^{'} + \varDelta f^{'}]. \tag{4}$$

The final candidates are then represented by $[y, g]$. This voting process is supervised by the Euclidean distance from the seeds on the object surface to the object centers. We denote this voting module as $\mathcal{V}$:

$$\begin{aligned} \mathcal{V}_\phi([x, f], z_n) &= [x, f \otimes z_n] + MLP_\phi(f \otimes z_n) \\ &= [y, g]. \end{aligned} \tag{5}$$

After object candidates generation, we adopt farthest point sampling [24] to select T candidates as center points in Euclidean space and generate clusters $\{\mathcal{C}_t\}_{t=1}^T$ by searching the nearest neighboring points for each center candidates, *i.e.*, $\mathcal{C}_t = \{[y_i, g_i] | \|y_i - y_t\| \leq d\}$, where y_t is the center coordinates of cluster t, y_i and g_i denote the coordinates and features of the i^{th} candidates, and d is the searching radius for each cluster. We denote the farthest point sampling and grouping process as $\mathcal{G}(\cdot)$.

Guided Object Proposal. We apply a 3D point feature learning network to extract cluster features and generate 3D object proposals. Specifically, for each cluster $\mathcal{C}_t$ represented by $\{[y_i, g_i]\}$, we normalize candidate locations to a local normalized coordinate system by $y_i' = (y_i - y_t)/d$. A shared PointNet [26], represented by $\mathcal{H}$, is adopted to produce cluster features. To generate object proposals with the guidance of 3D meta-detectors, we perform a channel-wise multiplication between cluster features and z_n to generate the re-calibrated cluster features. The guided object proposal module for the n^{th} class is defined as:

$$\mathcal{P}_\psi(\mathcal{C}) = MLP(\mathcal{H}([y_i', g_i]) \otimes z_c) \ , \tag{6}$$

where MLP is a multi-layer perceptron network serving as a prediction head to predict parameters of 3D bounding boxes and class labels. The whole detection head network is denoted by $\mathcal{P}$ with network parameters ψ.

4.4 Few-shot Learning Strategy

In this section, we introduce our learning strategy for N-way-K-shot detection. Our model takes support instances $\mathbf{S} = \{S_{n,k}\}$ and query scan q_i as input. We denote the ground-truth bounding boxes of query scan q_i as $\mathcal{R}_{q_i}$. The learning of this task can be reorganized as jointly optimizing network parameters θ, δ, ϕ, and ψ using back-propagation by minimizing the following loss:

$$\min_{\theta,\delta,\phi,\psi} \mathcal{L} = \sum_i \mathcal{L}(\mathcal{P}_\psi(\mathcal{V}_\phi(\mathcal{F}_\delta(q_i), \mathcal{M}_\theta(S_{n,k}))), \mathcal{R}_{q_i}) \ , \tag{7}$$

It is of importance to employ an appropriate learning strategy such that the 3D meta-detector $\mathcal{M}$ can produce representative re-weighting vectors to guide the 3D object detector $\{\mathcal{F}, \mathcal{V}, \mathcal{P}\}$ to generate the correct object proposals.

To achieve this goal, we adopt a two-phase learning strategy to train our proposed model. We refer to the first phase as *base-training*. During base-training, only episode data from the base classes are used, and we jointly train $\mathcal{F}$, $\mathcal{V}$, $\mathcal{P}$ and $\mathcal{M}$ using ground-truth annotations. This phase guarantees the learner, composed of $\mathcal{F}$, $\mathcal{V}$, and $\mathcal{P}$, to learn to detect class-specific objects by referring to re-weighting vectors. We denote the second phase as *fine-tuning*. In this step, we train our model using episode data from both base classes and novel classes. Similar to the first base-training phase, we jointly optimize θ,δ,ϕ and ψ by minimizing Eq. 7.

Loss function. Our model simultaneously estimates category labels of the bounding boxes, objectiveness scores, and bounding box parameters. We adopt a similar loss function as in VoteNet [4]. Specifically, We use the cross-entropy loss $\mathcal{L}_{sem}$ and $\mathcal{L}_{obj}$ to supervise the bounding box's label prediction as well as the objective score for candidates. Proposals generated from positive candidates that are close to the object center are further regressed to bounding parameters which include box center, heading angle, and box size. We adopt Huber loss $\mathcal{L}_{box}$ to supervise the bounding box prediction. Hence, the overall detection loss function is $\mathcal{L} = \mathcal{L}_{sem} + \mathcal{L}_{obj} + \mathcal{L}_{box}$.

5 Experiments and Results

In this section, we conduct experiments on two 3D object detection benchmarks to validate the object detection ability in the few-shot setting. In section 5.1, we first give a description of the benchmarks used in our experiments and how we adapt them to few-shot settings. In section 5.2, we describe the detailed implementation and the network architecture of our proposed model. In section 5.3, we compare our model with several baselines approaches for few-shot 3D object detection in an indoor environment. In section 5.4, we further analyze the performance of our model. In section 5.5, we validate the effectiveness of our proposed modules by comparing different designs.

5.1 Datasets

SUN RGB-D. We conduct our experiments on SUN RGB-D [13] dataset. It is a widely used benchmark dataset for object detection. The dataset consists of around 10k RGB-D images captured from indoor scenes. Half of them are used for training, and the rest are used for testing. Each sample in the dataset is annotated with amodal-oriented 3D bounding boxes and category labels. The dataset includes a total of 37 object categories. In the common 3D object detection setting, the training and evaluation process is based on the ten most common categories, including nightstand, sofa, and table. In our few-shot experiment setting, we divide the whole dataset into $\mathscr{D}_{novel}$ and $\mathscr{D}_{base}$ according to the categories, where $\mathscr{D}_{base} \cap \mathscr{D}_{novel} = \emptyset$. More specifically, we select N out of the ten categories as novel categories, and we refer to the other $10 - N$ categories as base categories. We further split each category into support/training and query/test sets.
ScanNet (v2). In addition to the SUN RGB-D dataset, we also conduct our experiment on the ScanNet (v2) [12] benchmark, which is a richly-annotated 3D indoor scene dataset that consists of 1,513 real-world scans. The whole dataset is divided into 1,201 training scenes, 312 validation scenes, and 100 test scenes. In contrast to the SUN RGB-D dataset, neither amodal bounding boxes nor their orientations are available in ScanNet (v2). Moreover, different from the SUN RGB-D dataset, ScanNet (v2) contains 18 object categories and provides complete reconstructed meshes of indoor scenes. We select N out of the 18 categories as novel categories and others as base categories. We also split each category into support/training and query/test sets for the SUN RGB-D dataset.

5.2 Implementation Details

Few-shot input and data augmentation. Our model takes point clouds from depth images (SUN RGB-D) or 3D indoor scans (ScanNet (v2)) as input. We uniformly sample 20k points from the source data, and each point is represented by 3D coordinates and the 1D height feature, which is the distance from the point to the floor. Following the VoteNet [4], we set the 1% percentile of all heights as the floor to calculate the height features. After the sampling, we apply random scaling from 0.8 to 1.2 on the data, followed by the random rotation for $-3°$ to $3°$ and random flipping.

	Novel Set 1				**Novel Set 2**				**Novel Set 3**			
Method	**bathtub**	**bed**	**chair**	**mAP**	**dresser**	**sofa**	**table**	**mAP**	**chair**	**desk**	**table**	**mAP**
VoteNet-JT	0.0	5.1	6.7	3.9	0.0	1.0	5.6	2.2	12.4	0.8	0.8	4.7
VoteNet-FT	0.0	16.1	4.8	7.0	0.1	4.2	6.9	3.7	13.7	2.5	2.2	6.1
VoteNet-2	0.0	16.6	5.5	7.4	0.1	4.8	7.5	4.1	16.5	2.0	2.1	6.9
Ours	3.5	21.8	10.3	11.9	10.4	9.9	11.0	10.4	10.2	12.7	12.1	11.6

Table 1. Quantitative results for few-shot 3D object detection on **SUN RGB-D**. Experiments are conducted on three different novel classes with $N = 3$, $K = 10$.

	Novel Set 1				Novel Set 2				Novel Set 3			
Method	**curtain**	**shower cur.**	**desk**	**mAP**	**garb. bin**	**bed**	**sink**	**mAP**	**sofa**	**table**	**chair**	**mAP**
VoteNet-JT	1.9	2.2	1.0	1.7	2.4	1.0	2.0	1.8	0.3	0.5	0.0	0.3
VoteNet-FT	3.4	4.9	2.2	3.5	6.1	1.7	4.4	4.1	1.6	1.8	0.5	1.3
VoteNet-2	4.8	5.2	2.9	4.3	7.5	2.1	5.1	4.9	2.2	2.6	1.1	1.9
Ours	26.5	11.0	3.0	13.5	19.4	10.8	1.1	10.4	9.2	8.0	8.4	8.5

Table 2. Quantitative results for few-shot 3D object detection on **ScanNet (v2)**. Experiments are conducted on three different novel classes with $N = 3$, $K = 10$. "shower cur." is the abbreviation for "shower curtain", and "garb. bin" is the abbreviation for "garbage bin".

Network architecture. We use PointNet++ [24] as backbone for the class-specific re-weighting module. The feature extractor network contains three set abstraction (SA) layers followed by a max-pooling layer. The radius of three SA layers are $[0.3, 0.5, 1]$ with the output channel sizes of $[1024, 512, 256]$. The 256-dimensional feature vector is fed into the weight learning network which is an MLP with two hidden layers of size $[512, 256]$. We fuse the class-specific re-weighting module with VoteNet [4] as our 3D object detector.
Network training. We follow the two-phase learning strategy to train our model from scratch with an Adam optimizer. In the base-training phase, we set the initial learning rate to 0.001 on $\mathcal{D}_{base}^{train}$. We decrease the learning rate by ten times after 50 epochs and 100 epochs. In the fine-tuning phase, we reset the initial learning rate to 0.001, and we decrease the learning rate every 30 epochs on $\mathcal{D}_{base}^{train}$ and $\mathcal{D}_{novel}^{train}$. Our model is implemented using PyTorch [30] and runs on NVIDIA GTX 2080 GPUs.

	Novel Set 1		Novel Set 2		Novel Set 3	
Method	**30**	**50**	**30**	**50**	**30**	**50**
VoteNet-JT	5.8	6.8	3.4	4.8	5.3	5.9
VoteNet-FT	8.1	9.8	4.6	5.6	7.3	8.1
VoteNet-2	9.0	11.3	5.1	6.1	8.0	8.9
Ours	12.1	14.0	11.7	13.6	13.2	13.8

Table 3. Results on **SUN RGB-D**. The table shows the evaluation results for different methods on three novel classes with different numbers of shots $K = 30, 50$. The evaluation metric is mAP with IoU threshold 0.25.

5.3 Comparison with the Baselines

Experiment settings. To the best of our knowledge, this is the first work focusing on the few-shot 3D object detection task. In order to explore the generalization ability of our proposed model on few-shot 3D object detection settings, we conduct experiments on SUN RGB-D, and ScanNet (v2) benchmarks to compare our model against various baseline models built on VoteNet [4]. Specifically, we adopt three baseline models for

	Novel Set 1		Novel Set 2		Novel Set 3	
Method	**30**	**50**	**30**	**50**	**30**	**50**
VoteNet-JT	2.2	4.1	2.4	3.1	0.6	0.8
VoteNet-FT	4.8	6.0	5.9	7.0	1.9	2.7
VoteNet-2	6.0	6.9	7.3	8.8	2.5	4.0
Ours	14.7	16.1	11.4	12.7	9.9	10.8

Table 4. Results on **ScanNet (v2)**. The table shows the evaluation results for different methods on three novel classes with different numbers of shots $K = 30, 50$. The evaluation metric is mAP with IoU threshold 0.25.

the comparison: 1) VoteNet-JT. In this baseline method, VoteNet is alternately trained on the base classes and novel classes without base-training and fine-tuning phases. 2) VoteNet-FT. In this baseline method, VoteNet first uses a base-training phase to train on the data from base classes, and then it only uses data from novel classes to fine-tune the model. 3) VoteNet-2. In this method, we train the VoteNet following the two-phase learning strategy until full convergence. Both our model and the three baseline models are trained in the few-shot setting where $N = 3$, $K = \{10, 30, 50\}$. During inference, we use the class-specific re-weighting vectors learned from the whole supporting set to guide 3D object detection on the query set.

Result analysis. The quantitative results on SUN RGB-D and ScanNet (v2) benchmarks are presented in Table 1 and Table 2 respectively. In both tables, the average precision values with the IoU threshold 0.25 are reported. Results presented in Table 1 and Table 2 show that our model outperforms the baseline models. The comparison results for different numbers of shots on two benchmarks are presented in Table 3 and Table 4. As shown in the tables, our model improves the mAP by at least 3.1% on SUN RGB-D benchmark when $K = 30$, and 2.7% when $K = 50$. Moreover, our method improves the mAP by at least 4.1% on ScanNet (v2) benchmark when $K = 30$, and 3.9% when $K = 50$. Refer to the supplementary material for the per-instance mAP. The results also demonstrate that directly adopting conventional object detectors leads to limited generalization abilities for novel classes. In Figure 4 and Figure 5, we show several qualitative examples of the 3D detection results on SUN RGB-D and ScanNet (v2) benchmarks respectively.

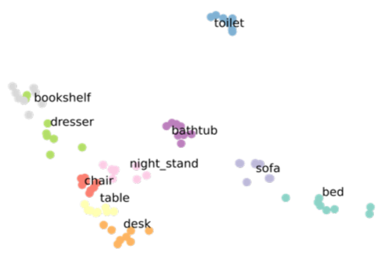
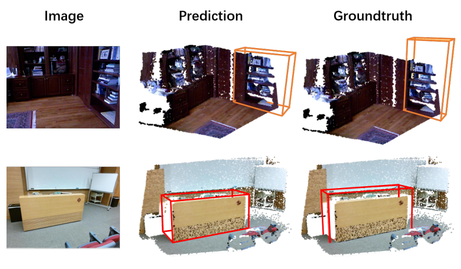


Fig. 3. The t-SNE visualization of class-specific re-weighting vectors from SUN-RGB D.

5.4 Performance Analysis

Class-specific re-weighting vector. The re-weighting vectors learned from the 3D meta-detector serve as important task-specific information to guide the detection process of the 3D object detector. In this section, we project the re-weighting vectors for



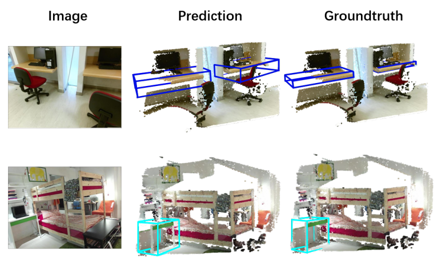
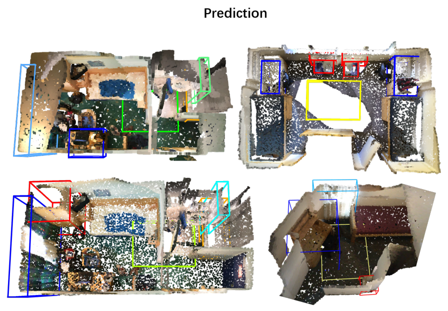


Fig. 4. Qualitative results on **SUN RGB-D**.

different object categories onto a 2D plane to verify that the re-weighting vectors are learned to contain class-specific information. We conduct experiments on the SUN RGB-D dataset. Figure 3 shows the projections of re-weight vectors by using t-SNE [31]. It is obvious that the vectors that belong to the same class tend to cluster with each other. Moreover, we also observe that classes sharing similar visual features are projected close to each other. Specifically, the desk, table, and chair classes are projected to the bottom-left in the figure, while the sofa and bed classes are projected to the bottom-right.



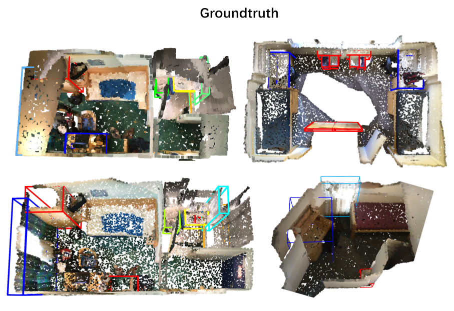


Fig. 5. Qualitative results on **ScanNet (v2)**.

Method	Model size	SUN RGB-D	ScanNet (v2)
VoteNet	11.2MB	0.10s	0.14s
Ours	11.6MB	0.11s	0.16s

Table 5. The comparison for the model size and the inference time for the object detection in one query scan.

Speed and size. We also analyze the inference speed and model size on different benchmark datasets. In Table 5, the comparison of our model and two baseline models is presented. Our model takes around 0.16s to detect objects in one query scan in ScanNet (v2) dataset, which is almost the same as the VoteNet. In addition, our proposed model has a size of 11.7MB, which is only 0.5MB larger than VoteNet. This observation demonstrates that our model can effectively detect novel objects with almost the same memory cost as VoteNet.

5.5 Where to Re-weight

We analyze the effect of different re-weighting locations on the final 3D object detection performance. In this experiment, we compare the object detection performance on ScanNet (v2) using three different designs of our proposed model. In the first design, we only use re-weighting in the "Guided Voting" module, and we denote this design as **V.R.**. In the second design, we only employ re-weighting in the object proposal module, which is referred as **O.R.**, and the last design corresponds to our proposed method that applies re-weighting in both object proposal modules. Table 6 shows the mAP of the three different designs. We notice that our proposed model that applies re-weighting in both object proposal modules achieves the best performance.

	V.R.	**O.R.**	**Ours**
Base classes	52.1	52.5	59.0
Novel classes	10.9	11.1	11.3

Table 6. The object detection results on SUN RGB-D with different model designs.

6 Conclusion

To the best of our knowledge, this is the first work to tackle the few-shot 3D object detection problem. Our proposed method generalizes its ability from the meta-training process to infer 3D object proposals and predict 3D bounding boxes for unseen 3D objects in novel classes. A novel class-specific re-weighting vector is introduced with the goal of facilitating the meta-detector to learn task distributions for different object classes and then dynamically adapt the 3D object detector to complete a specific detection task. Our method can be easily adapted to other two-stage detection methods that contain the object proposal stage. Experiments on two public 3D object detection benchmarks demonstrate that our model can effectively detect 3D objects from novel classes with superior performance over the well-established baseline models. In-depth analyses of our model further indicate the effectiveness and efficiency of each proposed component.

Acknowledgements The authors appreciate the generous support provided by Inception Institute of Artificial Intelligence (IIAI) in the form of NYUAD Global Ph.D. Student Fellowship. This work was also partially supported by the NYUAD Center for Artificial Intelligence and Robotics (CAIR), funded by Tamkeen under the NYUAD Research Institute Award CG010.

References

1. Afif, M., Ayachi, R., Said, Y., Pissaloux, E., Atri, M.: An evaluation of retinanet on indoor object detection for blind and visually impaired persons assistance navigation. Neural Processing Letters **51** (2020) 2265–2279
2. Yang, S., Scherer, S.: Cubeslam: Monocular 3-d object slam. IEEE Transactions on Robotics **35** (2019) 925–938
3. Qi, C.R., Chen, X., Litany, O., Guibas, L.J.: Imvotenet: Boosting 3d object detection in point clouds with image votes. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. (2020) 4404–4413
4. Qi, C.R., Litany, O., He, K., Guibas, L.J.: Deep hough voting for 3d object detection in point clouds. In: Proceedings of the IEEE International Conference on Computer Vision. (2019) 9277–9286
5. Hospedales, T.M., Antoniou, A., Micaelli, P., Storkey, A.J.: Meta-learning in neural networks: A survey. Transactions on Pattern Analysis and Machine Intelligence (2021)
6. Huisman, M., Van Rijn, J.N., Plaat, A.: A survey of deep meta-learning. Artificial Intelligence Review **54** (2021) 4483–4541
7. Nie, J., Xu, N., Zhou, M., Yan, G., Wei, Z.: 3d model classification based on few-shot learning. Neurocomputing **398** (2020) 539–546
8. Zhang, B., Wonka, P.: Training data generating networks: Linking 3d shapes and few-shot classification. arXiv preprint arXiv:2010.08276 (2020)
9. Huang, H., Li, X., Wang, L., Fang, Y.: 3d-metaconnet: Meta-learning for 3d shape classification and segmentation. In: International Conference on 3D Vision, IEEE (2021) 982–991
10. Yuan, S., Fang, Y.: Ross: Robust learning of one-shot 3d shape segmentation. In: The IEEE Winter Conference on Applications of Computer Vision. (2020) 1961–1969
11. Wang, L., Li, X., Fang, Y.: Few-shot learning of part-specific probability space for 3d shape segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. (2020) 4504–4513
12. 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 Conference on Computer Vision and Pattern Recognition. (2017) 5828–5839
13. Song, S., Lichtenberg, S.P., Xiao, J.: Sun rgb-d: A rgb-d scene understanding benchmark suite. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2015) 567–576
14. Wang, D.Z., Posner, I.: Voting for voting in online point cloud object detection. In: Robotics: Science and Systems. Volume 1., Rome, Italy (2015) 10–15
15. 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. Advances in neural information processing systems **32** (2019)
16. Koch, G., Zemel, R., Salakhutdinov, R.: Siamese neural networks for one-shot image recognition. In: ICML deep learning workshop. Volume 2., Lille (2015)
17. Vinyals, O., Blundell, C., Lillicrap, T., Wierstra, D., et al.: Matching networks for one shot learning. In: Advances in neural information processing systems. (2016) 3630–3638
18. Snell, J., Swersky, K., Zemel, R.: Prototypical networks for few-shot learning. In: Advances in Neural Information Processing Systems. (2017) 4077–4087
19. Sung, F., Yang, Y., Zhang, L., Xiang, T., Torr, P.H., Hospedales, T.M.: Learning to compare: Relation network for few-shot learning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2018) 1199–1208
20. Bertinetto, L., Henriques, J.F., Valmadre, J., Torr, P., Vedaldi, A.: Learning feed-forward one-shot learners. In: Advances in neural information processing systems. (2016) 523–531

21. Graves, A., Wayne, G., Danihelka, I.: Neural turing machines. arXiv preprint arXiv:1410.5401 (2014)
22. Finn, C., Abbeel, P., Levine, S.: Model-agnostic meta-learning for fast adaptation of deep networks. arXiv preprint arXiv:1703.03400 (2017)
23. Li, Z., Zhou, F., Chen, F., Li, H.: Meta-sgd: Learning to learn quickly for few-shot learning. arXiv preprint arXiv:1707.09835 (2017)
24. 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) 5099–5108
25. Leibe, B., Leonardis, A., Schiele, B.: Combined object categorization and segmentation with an implicit shape model. In: Workshop on statistical learning in computer vision, ECCV. Volume 2. (2004) 7
26. 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 Conference on Computer Vision and Pattern Recognition. (2017) 652–660
27. Santoro, A., Bartunov, S., Botvinick, M., Wierstra, D., Lillicrap, T.: Meta-learning with memory-augmented neural networks. In: International Conference on Machine Learning, PMLR (2016) 1842–1850
28. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: European Conference on Computer Vision, Springer (2014) 740–755
29. Xie, Q., Lai, Y.K., Wu, J., Wang, Z., Zhang, Y., Xu, K., Wang, J.: Mlcvnet: Multi-level context votenet for 3d object detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. (2020) 10447–10456
30. Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A.: Automatic differentiation in pytorch. (2017)
31. Maaten, L.v.d., Hinton, G.: Visualizing data using t-sne. Journal of machine learning research **9** (2008) 2579–2605