A Unified Query-based Paradigm for Point Cloud Understanding

Zetong Yang[1,4*] Li Jiang[2*] Yanan Sun[3] Bernt Schiele[2] Jiaya Jia[1]
[1]CUHK [2]MPI Informatics [3]HKUST [4]SmartMore
{tomztyang, now.syn}@gmail.com {lijiang, schiele}@mpi-inf.mpg.de leojia@cse.cuhk.edu.hk

Abstract

*3D point cloud understanding is an important component in autonomous driving and robotics. In this paper, we present a novel **E**mbedding-**Q**uerying paradigm (EQ-Paradigm) for 3D understanding tasks including detection, segmentation and classification. EQ-Paradigm is a unified paradigm that enables combination of existing 3D backbone architectures with different task heads. Under the EQ-Paradigm, the input is first encoded in the embedding stage with an arbitrary feature extraction architecture, which is independent of tasks and heads. Then, the querying stage enables the encoded features for diverse task heads. This is achieved by introducing an intermediate representation, i.e., Q-representation, in the querying stage to bridge the embedding stage and task heads. We design a novel Q-Net as the querying stage network. Extensive experimental results on various 3D tasks show that EQ-Paradigm in tandem with Q-Net is a general and effective pipeline, which enables flexible collaboration of backbones and heads. It further boosts performance of state-of-the-art methods.*

1. Introduction

3D point cloud understanding is an essential line in computer vision since it could benefit many applications, such as autonomous driving [14], robotics [12], and augmented reality [31].

In point cloud understanding, there are two dominant input representations: points and voxels. Specifically designed for these two representations, mainstream models can be grouped into point- [19, 23, 28, 36, 51, 56, 69] and voxel-based [7, 16, 61, 71] networks. In both cases, state-of-the-art models consist of an encoder network to gradually downsample the points/voxels by sampling algorithms / strided convolution. There are also a decoder network to propagate features of the subsampled points/voxels into original ones and a task-specific head for making predictions. We call these methods **E**ncoder-**D**ecoder paradigm (ED-Paradigm) models. Due to the downsampling-upsampling design, ED-Paradigm models extract features for some fixed positions appearing in the downsampling process.

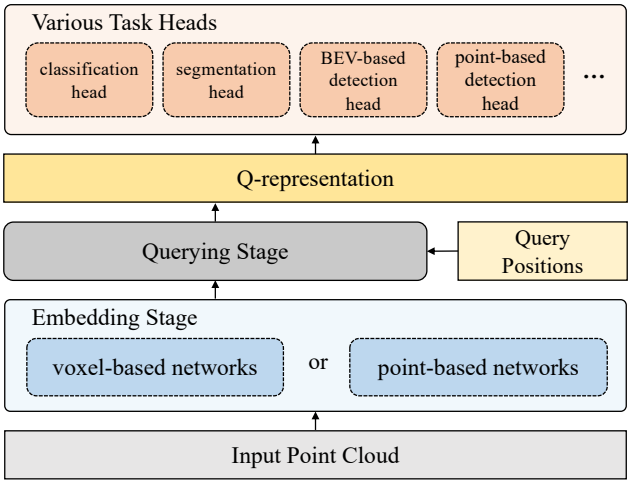

Figure 1. Illustration of the unified query-based EQ-Paradigm. The query position can be randomly designated in the 3D scene, thus making it possible to combine any backbone embedding networks with different task heads.

In this paper, we propose a novel **E**mbedding-**Q**uerying paradigm (EQ-Paradigm) for 3D understanding tasks. Compared to the ED-Paradigm, which extracts features for fixed positions, EQ-Paradigm enables feature generation for any position in the 3D scene. Thus, the EQ-paradigm is generalization of the ED-Paradigm. Any ED-Paradigm model has an EQ-Paradigm counterpart. An EQ-Paradigm model consists of three stages: an **E**mbedding stage, a **Q**uerying stage, and a task-specific head.

The embedding stage can be implemented with any feature extraction architecture, including voxel- and point-based networks, regardless of tasks and heads. We use the embedding network to extract *support features* for following stages. The querying stage then takes a set of positions as *query positions* and generates their intermediate representation, *i.e.*, *Q-representation*, based on the support features. Note that the query positions could be any point in the contiguous 3D space, thus enabling feature generation for any location. We further present a novel querying stage net-

* Equal contribution. Work done during internship at SmartMore.

work called *Q-Net* to effectively extract Q-representation. Afterwards, a task head is employed for generating predictions based on the Q-representation.

Due to the flexibility in query position designation, the EQ-Paradigm is a unified query-based paradigm that can easily combine any state-of-the-art 3D backbone networks with different task heads without extra efforts (Figure 1), which gives a lot of freedom in the head design. For example, SSD head [21] designed for voxel-based detectors [61, 72] can be applied with a point-based embedding network under EQ-Paradigm; an EQ-Paradigm segmentation model can directly obtain point-wise features based on a voxel-based embedding network [7, 16]; also, an EQ-Paradigm version of PVRCNN [39] is able to directly generate proposal grid features from the voxel-based backbones for the following detection head. This greatly increases the flexibility of model design for different tasks.

We evaluate our EQ-Paradigm on multiple important 3D understanding tasks including object detection [26, 32, 39, 40, 61], semantic segmentation [7, 16] and shape classification [36]. Our experiments show that our EQ-Paradigm and Q-Net can be well integrated with any state-of-the-art models regardless of tasks, backbone architectures and head designs, while making consistent performance improvement. Our primary contributions are the following.

- We propose an **E**mbedding-**Q**uerying paradigm for 3D point cloud understanding. It is a unified query-based paradigm enabling combination of arbitrary point- or voxel-based networks with different task heads.
- We present a novel querying stage network Q-Net, to extract intermediate Q-representation, *i.e.*, query features, for the designed query positions.
- We integrate our EQ-Paradigm and Q-Net into multiple state-of-the-art 3D networks for different tasks and achieve consistent performance improvement from extensive experiments.

2. Related Work

ED-Paradigm ED-Paradigm models are widely used. They consist of an encoder network to extract high-level semantic features, a decoder network for feature propagation, and a task head to perform predictions. U-Net [37] is a classical ED-Paradigm network to deal with biomedical image segmentation. It inspires following work on 2D pixel-level tasks including semantic segmentation [5, 6, 27, 70], super resolution (SR) [50] and matting [13, 44, 57]. In 3D tasks, it is also a mainstream paradigm for object detection [18,34,61,72] and semantic segmentation [23,28,45,54,69].

Point-based 3D Architectures Point-based 3D models deal with raw point clouds, which extract sparse point features, downsample point cloud in encoder networks, propagate features to original points by decoders, and make predictions by task-specific heads. PointNet++ [36] is a fundamental point-based backbone and has been widely applied to many point-based models [26,33,34,40,52,63,64]. These models utilize a series of set-abstraction layers as their encoders and multiple feature propagation layers as decoders.

Some models focus on developing elegant heads to leverage the sparse point features. For example, F-PointNet [34] achieves amodal 3D box estimation. In [40, 65], canonical 3D bounding box refinement and PointsPool layer are proposed respectively. Other point-based backbones [19, 59, 68] focus on improving the feature aggregation operation in PointNet++ by introducing graph convolutions [42, 48], convolution-like operations [45, 54, 56] or transformer structure [30,69]. For raw point cloud, a point-based model can extract features with accurate relative positions and structural information. But it is limited in dealing with large-scale point cloud due to high time complexity for ball query and farthest point sampling.

Voxel-based 3D Architectures Voxel-based methods first divide raw point cloud into regular voxels, then apply convolutional neural networks (CNNs) composed of sparse [7, 15, 16] or dense [72] convolutions as their encoder and decoder networks to extract voxel features. Still, voxel-based models are widely applied in various methods [7, 18, 39, 41, 61, 72] on different tasks. Compared to point-based architectures, voxel-based methods reduce a large number of redundant points in the same voxels but sacrifice the data precision during the voxelization process. However, voxel-based methods are able to deal with the large-scale scenario. In this paper, we propose an EQ-Paradigm for enabling models on different tasks to easily switch between these two backbone architectures and providing great flexibility in head design.

3. EQ-Paradigm

We first give an overview of the EQ-paradigm in this section, then elaborate our novel querying stage design, Q-Net, in the next section.

3.1. Overview

As shown in Figure 2, our EQ-Paradigm has three stages: an **E**mbedding stage, a **Q**uerying stage and a task head. First, the embedding stage extracts features from the input point cloud $I \in \mathbb{R}^{N\times 3}$. We take those features as support features F_S for the following querying stage. The corresponding 3D positions of F_S are denoted as support points $S \in \mathbb{R}^{n\times 3}$. The querying stage is then responsible for generating Q-representation, that is, query features F_Q for query positions $Q \in \mathbb{R}^{m\times 3}$ based on support points S and support features F_S. Notably, Q is not required to be a subset of I. Instead, a query point is expected to be any position

in the continuous 3D space. We provide a novel querying stage design called Q-Net. Finally, the task head produces predictions based on query positions Q and features F_Q. Our EQ-Paradigm is expressed as

$$\begin{aligned} F_S, S &= \text{Embedding}(I), \\ F_Q &= \text{Querying}(F_S, S, Q), \\ O &= \text{Head}(F_Q, Q), \end{aligned} \tag{1}$$

where O indicates the final outputs for specific tasks.

3.2. Embedding Stage

In the EQ-Paradigm, the feature extraction network in the embedding stage can be any 3D network, including voxel-based networks with voxelized input [7, 15, 16] and point-based networks with raw point clouds [23, 35, 36, 54], independent of tasks and heads. The goal of the embedding stage is to generate support points S and support features F_S. For point-based embedding networks, the support points S are usually a subsample of the input point cloud I, depending on the downsampling strategy (*e.g.*, farthest point sampling) of the network. In the voxel-based situation, downsampling is usually achieved by strided convolutions, and we take the downsampled voxel centers as S.

As mentioned in Section 2, a voxel-based backbone is able to deal with large-scale point cloud scenarios, while a point-based backbone can extract more precise structural information. In the EQ-Paradigm, a model can arbitrarily specify its embedding stage network according to the practical demand, which brings flexibility in model design.

3.3. Querying Stage

The querying stage is utilized for extracting query features F_Q for a set of manually designated query positions Q from support features F_S and their positions S. The queried features are then sent to the task-specific head for generating final predictions.

The key aspect of the querying stage lies in the selection of query positions according to different tasks and head designs, as illustrated in Figure 2 and the following.

- *Query positions in detection.* To deploy an SSD [21, 61] head in an outdoor 3D object detection model, query positions are selected to be the pixel centers within the target Bird-Eye-View (BEV) map (Figure 2(a)). To utilize point-based heads proposed in [32, 40, 63], query positions are subsampled points from the raw input point cloud by uniform or farthest point sampling (Figure 2(b)).
- *Query positions in segmentation.* In semantic segmentation, query positions are the points requiring point-wise class predictions in a 3D scene (Figure 2(c)). Usually, the whole input point cloud I is taken as Q.

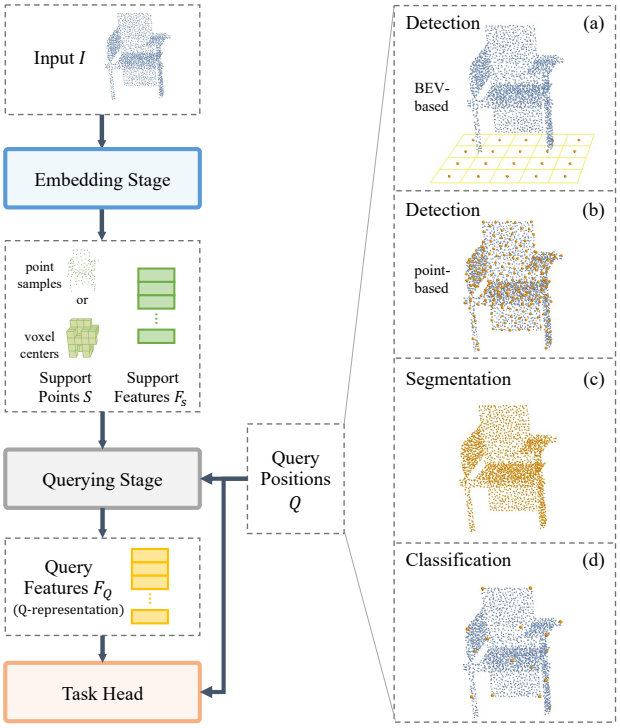

Figure 2. Overview of our EQ-Paradigm. Given an input point cloud I, a set of support features F_S for support points S are generated in the embedding stage. The support points (marked in **green**) can be voxel centers or point samples for voxel- or point-based embedding networks, respectively. The querying stage network generates the query features F_Q (also known as Q-representation) used in the task head for query positions Q based on S and F_S. The query positions Q (marked in **yellow**) for different tasks and heads are shown in (a)-(d).

- *Query positions in classification.* In classification, Q can be the shape center to produce a representative feature for the classifier, or can also be multiple uniformly-distributed positions indicating different parts of an object to vote for the category (Figure 2(d)). In this paper, we intend to vote category using 16 sampled points as query positions.

The querying stage is agnostic of the embedding network type and has great flexibility in query position selection. The point or voxel features extracted in the embedding stage can be well propagated to the query positions required by different tasks and heads. Also, for a specific task head, it is possible to switch the point- or voxel-based embedding networks depending on which representation is better for the head. This is valuable for tasks like detection, where the head and backbone designs are both important, as shown in the ablation study in Section 5.5.

4. Q-Net

Our querying stage network Q-Net is based on the transformer structure [11, 47] to extract Q-representations, *i.e.*,

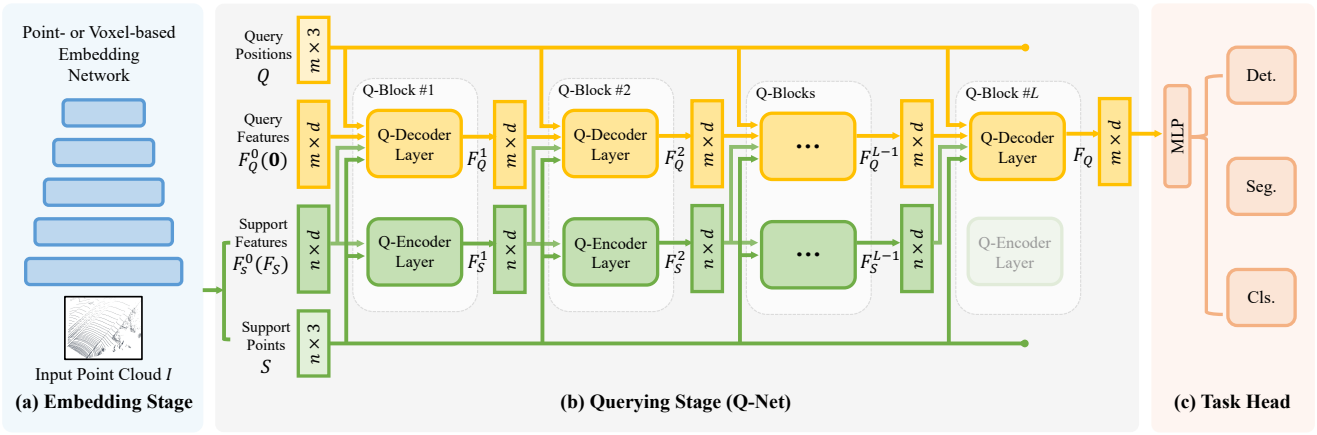

Figure 3. Illustration of our Q-Net in EQ-Paradigm. Taking as input the support points S and support features F_S, Q-Net generates query features F_Q for query positions Q. Q-Net consists of L consecutive Q-Blocks, each containing a Q-Encoder layer for updating support features and a Q-Decoder layer for refining query features. We initialize F_Q^0 by $\mathbf{0}$ and take F_S as the initial support features F_S^0.

query features F_Q. Recently, transformer models have shown great potential in the field of 2D computer vision [4, 10, 11, 25, 46, 62] as well as 3D tasks [29, 30, 66, 69]. Here, we develop our Q-Net based on the transformer to effectively generate features for query positions due to the flexible receptive field and strong representation ability of transformer layers. Note that the transformer mechanism suits the querying stage because the attention operator with positional encoding both contributes a global perspective and considers the relative positions between points, which satisfies the need of feature generation for flexible query positions. Figure 3 shows the architecture of Q-Net.

4.1. Q-Block

Q-Net is a stack of L Q-Blocks. Each Q-Block has four input elements. For the l-th block, the four inputs are query positions Q, support points S, query features F_Q^{l-1}, and support features F_S^{l-1}, where F_Q^{l-1} and F_S^{l-1} are the output of the $(l-1)$-th block. For the first Q-Block, we initialize F_Q^0 as $\mathbf{0}$. Since query positions Q are not necessarily a subset of input point cloud I, initializing their features as zeros does not introduce any inductive bias. Meanwhile, F_S^0 is initialized by the support features F_S from the embedding stage. These L Q-Blocks update the query and support features in iterations. L is set to 3 in our implementation. Ablation study on L is included in the supplementary material.

Each Q-Block utilizes two layers. They are a Q-Encoder layer and a Q-Decoder layer to update the support features and refine the query features, respectively. The support features are updated to encode richer global semantic information, thus benefiting the query feature refinement. We abandon the Q-Encoder layer in the last Q-Block, since we do not need updated support features anymore without the next Q-Decoder layer. The output of the last Q-Block is the final query features F_Q, which are fed into the task head for making predictions. Formally, the Q-Block is depicted as

$$\begin{aligned} F_Q^l &= \text{Q-Decoder}(Q, F_Q^{l-1}, S, F_S^{l-1}), \\ F_S^l &= \text{Q-Encoder}(S, F_S^{l-1}). \end{aligned} \tag{2}$$

We follow the original transformer [47] to build our Q-Encoder and Q-Decoder layers. We adopt the transformer encoder layer as our Q-Encoder layer, while the Q-Decoder layer is adapted from the transformer decoder layer.

Q-Encoder Layer We use the Q-Encoder layer to update the support features. Architecture of our Q-Encoder layer follows the widely-used transformer encoder layer, which consists of an attention layer (Attention) and a feed-forward network (FFN). We formulate the Q-Encoder layer as

$$\begin{aligned} \hat{F}_S^l &= \text{Attention}(S, F_S^{l-1}, S, F_S^{l-1}) + F_S^{l-1}, \\ F_S^l &= \text{FFN}(\hat{F}_S^l) + \hat{F}_S^l. \end{aligned} \tag{3}$$

The attention layer here is a classical $\mathbf{qkv}$-based multi-head self-attention [47], where $\mathbf{q}$, $\mathbf{k}$ and $\mathbf{v}$ are all from the support features F_S^{l-1}. We use LayerNorm [3] to normalize features before each Attention and FFN module.

Q-Decoder Layer The Q-Decoder layer generates enhanced feature representations for query positions. Different from the transformer decoder layer, in the Q-Decoder layer, we do not apply self-attention to query features and instead directly adopt the cross-attention layer to generate query features from the support features, formulated as

$$\begin{aligned} \hat{F}_Q^l &= \text{Attention}(Q, F_Q^{l-1}, S, F_S^{l-1}) + F_Q^{l-1}, \\ F_Q^l &= \text{FFN}(\hat{F}_Q^l) + \hat{F}_Q^l, \end{aligned} \tag{4}$$

where the attention layer is a $\mathbf{qkv}$-based multi-head cross-attention, in which $\mathbf{q}$ is from the query features while $\mathbf{k}$ and $\mathbf{v}$ are from the support features. Removing the self-attention

in the conventional transformer decoder layer keeps independence of query positions. That is, the query feature of a query position only depends on its relationship with the support points/features, but not with other query positions/features, thus providing more freedom in the choice of query positions. For example, we can query the features of only parts of interest in the whole scene. The ablation study in Section 5.5 shows the advantages of this design.

Attention Layer The Attention layer, formulated as

$$\tilde{F}_Y = \text{Attention}(Y, F_Y, X, F_X), \tag{5}$$

plays a fundamental role in a Q-Block. It leverages m target positions $Y \in \mathbb{R}^{m\times 3}$ with features $F_Y \in \mathbb{R}^{m\times d}$ and n source positions $X \in \mathbb{R}^{n\times 3}$ with features $F_X \in \mathbb{R}^{n\times d}$ to obtain new target features $\tilde{F}_Y \in \mathbb{R}^{m\times d}$. Here, d denotes the channel number of features. A $\mathbf{qkv}$-based attention layer can be viewed as applying attention weights to the source features F_X for computing new target features. Here, we describe the single-head calculation for clarity. The computation of the i-th new target feature $\tilde{F}_Y^{(i)}$ is formulated as

$$\tilde{F}_Y^{(i)} = \mathcal{A}^{(i)}(F_X W_{\mathbf{v}} + B_{\mathbf{v}}^{(i)}). \tag{6}$$

The attention weight $\mathcal{A} \in \mathbb{R}^{m\times n}$ is obtained by utilizing a softmax function on the result of dot product between target features F_Y and source features F_X as

$$\mathcal{A} = \text{SoftMax}\left(\frac{(F_Y W_{\mathbf{q}})(F_X W_{\mathbf{k}})^T + B_{\mathbf{qk}}}{\sqrt{d}}\right). \tag{7}$$

$W_{\mathbf{q}}$, $W_{\mathbf{k}}$ and $W_{\mathbf{v}}$ are weights of the linear layers for $\mathbf{q}$, $\mathbf{k}$ and $\mathbf{v}$, respectively. Also, in our Q-Block, we apply two types of relative positional encoding. The first $B_{\mathbf{v}} \in \mathbb{R}^{m\times n\times d}$ in Eq. (6) is for providing relative geometric information in the value vectors. The second $B_{\mathbf{qk}} \in \mathbb{R}^{m\times n}$ in Eq. (7) encodes the Euclidean positional difference between the target Y and source X in the attention weights.

Relative Positional Encoding Relative positional encoding is an indispensable component in our Q-Net. Unlike previous transformer structures [4, 47] that adopt input features with effective semantic and position information, we initialize query features F_Q^0 by $\mathbf{0}$ in the first Q-Block, which avoids introducing inductive biases and provides no effective information. Hence, at the beginning of our Q-Net, query positions are the only hints for generating query features from support points and support features.

Meanwhile, it is not optimal to update query features in the first block only depending on the coordinate difference between query and support points, since it makes no difference in attention weights for object points with the same relative position but in various scales and shapes. Inspired by [38, 53], we adopt contextual relative positional encoding, which fits our Q-Block well.

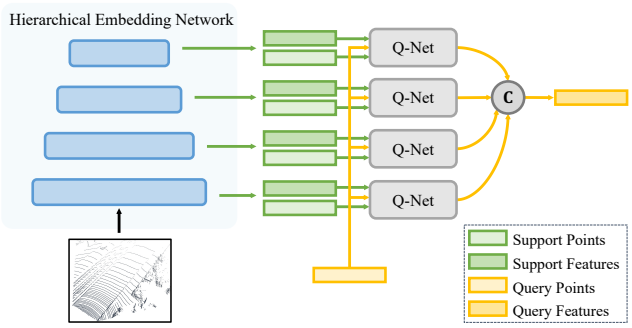

Figure 4. The hierarchical extension of our Q-Net.

Compared to bias-mode relative positional encoding [25, 53, 62], contextual relative positional encoding considers the interactions of positional embeddings with the $\mathbf{q}$, $\mathbf{k}$, $\mathbf{v}$ features, making the relative positional encoding automatically adapt to features with different contextual information. Hence, it produces various responses for points in objects with diverse scales and shapes, even when some point pairs share the same relative positional difference. We provide the details of our relative positional encoding strategy and its effect in the supplementary material.

Local Attention When the numbers of target m and source n are large, *e.g.*, 40k, applying global attention to them is extremely GPU memory-consuming, since the attention weight $\mathcal{A} \in \mathbb{R}^{m\times n}$ is too large to store. To address this issue, we instead apply local attention in our Q-Net inspired by [30, 69]. Specifically, for each target point, we figure out its K nearest neighbors (KNN) in source points according to Euclidean distances and compute attention only on these neighbors. In this way, the size of attention weight $\mathcal{A}$ is greatly reduced to $m \times K$, and K is far smaller than n.

4.2. Hierarchical Q-Net

Hierarchical multi-level architecture is proven to be essential for 3D tasks [7, 36] considering the diversity in 3D scene scales and object sizes. Especially for a point-wise prediction task like semantic segmentation, the multi-level features are of great importance in producing state-of-the-art results [7, 69], since the fine-grained features are needed to make detailed per-point segmentation.

We develop a hierarchical Q-Net for exploiting multi-level features. As illustrated in Figure 4, we apply a series of Q-Nets on support features from multiple levels of the hierarchical embedding network and concatenate the query features from different levels to generate final predictions.

5. Experiments

We conduct experiments on four popular 3D tasks: semantic segmentation, indoor object detection, outdoor object detection and shape classification. Implementation details of training schedule, hyper-parameters and network

Method	ScanNet		S3DIS	
	Validation	Test	Area 5	6-fold
PointNet [35]	-	-	41.1	47.6
PointNet++ [36]	-	33.9	-	-
PointCNN [19]	-	45.8	57.3	65.4
PointWeb [68]	-	-	60.3	66.7
PointEdge [17]	63.4	61.8	61.9	67.8
PointConv [54]	61.0	66.6	-	-
PointASNL [60]	66.4	66.6	62.6	68.7
KPConv [45]	69.2	68.6	67.1	70.6
FusionNet [67]	-	68.8	67.2	-
SparseConvNet [16]	-	72.5	-	-
MinkowskiNet [7]	72.2	73.6	65.4	-
PAConv [56]	-	-	66.6	69.3
PointTransformer [69]	-	-	70.4	73.5
Sparse U-Net (Baseline)	72.9	-	66.9	72.6
Sparse EQ-Net (Ours)	**75.3**	**74.3**	**71.3**	**77.5**
Improvement	*+2.4*	-	*+4.4*	*+4.9*

Table 1. Semantic segmentation results on mIoU(%) of our method and other 3D networks on ScanNet and S3DIS. The Sparse U-Net is our re-implemented version of SparseConvNet.

Method	Network	mAP @0.25	mAP @0.5
ScanNetV2			
VoteNet [32]	PointNet++	58.6	33.5
VoteNet$^+$	PointNet++	62.9	39.9
VoteNet (Ours)	EQ-PointNet++	**64.3**	**45.4**
GroupFree [26]	PointNet++ (L6, O256)	67.3	48.9
GroupFree$^+$	PointNet++ (L6, O256)	66.3	47.8
GroupFree (Ours)	EQ-PointNet++ (L6, O256)	**68.0**	**50.0**
SUN RGB-D			
VoteNet [32]	PointNet++	57.7	32.9
VoteNet$^+$	PointNet++	59.1	35.8
VoteNet (Ours)	EQ-PointNet++	**60.5**	**38.5**

Table 2. Performance of different methods with PointNet++ and EQ-PointNet++ on ScanNetV2 and SUN RGB-D datasets. $^+$ denotes the models reproduced by MMDetection3D [8].

structure are included in the supplementary material.

5.1. Semantic Segmentation

Datasets For point cloud semantic segmentation, we use competitive and popular datasets of ScanNetV2 [9] and S3DIS [1] in our experiments. ScanNetV2 comprises 1,613 indoor scans (1,201/312/100 for train/val/test) with point-wise semantic labels in 20 object categories. S3DIS is composed of 271 point cloud scenes collected from 6 large-scale indoor areas, annotated with 13 semantic classes. For evaluation, we follow the commonly-used S3DIS dataset split [7, 19, 68] to test on Area 5 and train on other 5 areas, and also apply the 6-fold cross validation, which takes each area as test set once. For the evaluation metrics, we adopt the mean Intersection-over-Union (mIoU).

Models We utilize the voxel-based residual U-Net structure with sparse convolutions [7, 16] as the baseline model in our experiments. The sparse U-Net follows the ED-Paradigm with an encoder network and a decoder one. It is a powerful backbone structure in 3D segmentation. We develop our network with EQ-Paradigm based on the sparse U-Net by keeping the encoder as our embedding network and adopting the Q-Net to extract the features for each point. In the embedding stage, the input 3D volume is downsampled for 6 times, providing multi-level support features. We use the center coordinates of the voxels as the support positions and apply the hierarchical Q-Net to fuse the multi-level features to get better feature representations for the query positions. The queried features are then fed into a classifier to produce point-wise semantic predictions.

Experimental Results We compare our EQ-Net with our baseline model and other 3D networks. The results are shown in Table 1. On both datasets, our method attains higher mIoU than the strong baseline models, with significant gain of 2.4%, 4.4% and 4.9% on ScanNet validation set, S3DIS Area 5 and 6-fold, respectively. Also, compared with recent state-of-the-art 3D segmentation networks, our EQ-Net still achieves higher performance on these two datasets, showing effectiveness of the EQ-Paradigm and our well-designed Q-Net in point-wise prediction tasks.

5.2. Indoor Object Detection

Datasets We evaluate our method on two popular datasets of ScanNetV2 [9] and SUN RGB-D [43]. ScanNetV2 contains 1,513 scenes with bounding boxes labeled in 18 categories; SUN RGB-D includes 5k training scenes with bounding boxes in 10 classes. Evaluation metric is the mean Average Precision (mAP) with intersection-over-union (IoU) 0.25 (mAP@0.25) and 0.5 (mAP@0.5) following [32].

Baseline Models We test our approach on VoteNet [32] and GroupFree [26] for ScanNetV2 dataset, and on VoteNet for SUN RGB-D dataset. All baseline models are publicly available at MMDetection3D [8] codebase. VoteNet is the classical indoor detector serving as the baseline model for all modern methods; GroupFree is the current state-of-the-art indoor detector.

EQ-PointNet++ PointNet++ [36] is the cornerstone of indoor 3D object detection. Recent methods [26, 32] all utilize it to extract sparse point features for detection heads.

EQ-PointNet++ is the EQ-Paradigm version of PointNet++. It treats a stack of set-abstraction layers as its embedding stage similar to PointNet++ and applies a hierarchical Q-Net in its querying stage to extract query features with multi-level information. Query positions are 1,024 points obtained by applying furthest point sampling (FPS) on the raw input point cloud following [26, 32]. For all models, we replace their PointNet++ backbone networks by our EQ-PointNet++ networks.

Experimental Results As shown in Table 2, models with EQ-PointNet++ achieve better performance on both

Set	Method	Car (%)			Pedestrian (%)			Cyclist (%)		
		Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard
Val	SECOND [61]	90.85	**81.66**	78.57	56.07	51.12	46.14	83.06	66.69	63.02
	EQ-SECOND (Ours)	**91.74**	81.49	**78.62**	**57.48**	**53.64**	**49.55**	**85.01**	**67.13**	**63.34**
	PointRCNN [40]	91.35	80.25	77.84	61.19	54.33	47.43	89.77	**71.55**	**67.20**
	EQ-PointRCNN (Ours)	**91.80**	**84.00**	**82.29**	**64.80**	**58.36**	**52.55**	**91.23**	71.09	66.35
	PVRCNN [39]	92.07	84.75	82.46	62.32	54.42	49.81	90.39	70.42	65.99
	EQ-PVRCNN† (Ours)	**92.63**	85.41	82.97	66.78	59.23	54.34	**93.34**	**75.71**	**71.11**
	EQ-PVRCNN§ (Ours)	92.52	**85.61**	**83.13**	**69.95**	**62.55**	**56.51**	91.51	74.02	69.46
Test	PVRCNN [39]	**90.25**	81.43	76.82	52.17	43.29	40.29	78.60	63.71	57.65
	EQ-PVRCNN§ (Ours)	90.13	**82.01**	**77.53**	**55.84**	**47.02**	**42.94**	**85.41**	**69.10**	**62.30**

Table 3. Performance comparison on the KITTI val and test sets.

datasets. Specifically, VoteNet with EQ-PointNet++ gains 5.5% and 2.7% mAP@0.5 improvement on ScanNetV2 and SUN RGB-D datasets respectively. On state-of-the-art indoor detector GroupFree [26], our approach brings consistent performance improvement in terms of mAP@0.25 and mAP@0.5 by 0.7% and 1.1% compared to the official results [26] and by 1.7% and 2.2% compared to our reproduced results [8]. These experiments demonstrate our EQ-Paradigm and Q-Net are well adapted into indoor detectors and boost their performance.

5.3. Outdoor Object Detection

Datasets For outdoor detection, we conduct experiments on the widely adopted KITTI dataset [14]. There are 7,481 training point clouds and 7,518 testing point clouds with 3 categories of "Car", "Pedestrian" and "Cyclist". Following [72], we split the original KITTI training dataset into 3,717 images/scenes train set and 3,769 images/scenes val set. All "AP" results are calculated with 40 recall positions following the official KITTI protocol.

Baseline Models We select 3 outdoor detectors to demonstrate the superiority of our approach. They are SECOND [61], PointRCNN [40] and PVRCNN [39]. These methods with different heads require varying query position designation. In SECOND, query positions are the pixel centers within the target BEV map; in PointRCNN, all points within the input point cloud serve as query positions; while in PVRCNN, they can be coordinates either of keypoints (EQ-PVRCNN§) following the original PVRCNN design or of proposal grids in a straightforward way (EQ-PVRCNN†).

Experimental Results As listed in Table 3, our approach yields consistent improvement on different detectors. Especially on PointRCNN, EQ-PointRCNN obtains significant improvements, *e.g.*, 3.75% "AP" improvement on "Car" instances labeled as "Moderate" difficulty. Compared to the state-of-the-art model PVRCNN, our approach achieves remarkable improvement on both KITTI val and test sets. On the test set, EQ-PVRCNN§ attains 3.73% and 5.39% improvement on "Pedestrian" and "Cyclist" instances labeled as "Moderate" difficulty level.

These practical improvement indicates that EQ-Paradigm and Q-Net can be widely applied to any 3D outdoor detectors and deliver sustained performance improvement. Meanwhile, by altering query positions, our approach can inspire some new designs on existing methods. As shown in Table 3, by directly obtaining proposal grid features for box prediction to get rid of a few modules (including voxel-set abstraction, predicted keypoint weighting, and RoI-grid Pooling in PVRCNN), EQ-PVRCNN† still achieves impressive performance improvement with concise head design.

Method	Input	Accuracy (%)
PCNN [2]	1k points	92.3
RS-CNN (SSG) [23]	1k points	92.4
PointCNN [19]	1k points	92.5
KPConv [45]	1k points	92.9
DGCNN [51]	1k points	92.9
InterpCNN [28]	1k points	93.0
DensePoint [22]	1k points	93.2
Grid-GCN [58]	1k points	93.1
PosPool [24]	5k points	93.2
SpecGCN [49]	2k points+normal	92.1
PointWeb [68]	1k points + normal	92.3
SpiderCNN [59]	1k points+normal	92.4
PointConv [54]	1k points+normal	92.5
PointNet++ (SSG)	1k points	92.1
EQ-PointNet++ (SSG)	1k points	**93.2**

Table 4. Accuracy comparison on the ModelNet40 dataset.

5.4. Shape Classification

Datasets and Models We conduct classification experiments on ModelNet40 dataset [55], which includes 9,843 training and 2, 468 testing meshed models in 40 categories. We employ EQ-PointNet++ as our classification model. Query positions are 16 points obtained by furthest point sampling on the input point cloud. In the recognition head, we deploy another set-abstraction layer to summarize the 16 query features for category prediction.

Experimental Results As shown in Table 4, EQ-PointNet++ surpasses PointNet++ with single-scale grouping (SSG) by 1.1% in terms of classification accuracy. Compared with other classifiers [51], EQ-PointNet++ still yields better performance, showing the generalization ability of EQ-Paradigm and effectiveness of Q-Net.

Head	Embedding Network		AP (%)
	voxel-based	point-based	
SECOND head (voxel-based)	√	-	81.49
	-	√	82.70
	√	√	82.94
PointRCNN head (point-based)	√	-	82.65
	-	√	84.00
	√	√	84.38

Table 5. AP comparison on different head designs with point- and voxel-based embedding networks.

5.5. Ablation Study

Analysis on the EQ-Paradigm In Table 5, we verify the capacity of EQ-Paradigm in combining point- or voxel-based backbone networks with different task heads by adopting different embedding structures in voxel- and point-based detectors, SECOND [61] and PointRCNN [40]. Experiments are conducted on KITTI validation set with "AP" calculated on "Moderate" difficulty level in class "Car". We use the SparseConvNet in SECOND [61] as the voxel-based embedding network, and PointNet++ without decoder in PointRCNN [40] as the point-based embedding network.

As illustrated in Table 5, heads in SECOND and PointRCNN are both applicable to point- and voxel-based embedding stage networks and produce promising performance. This manifests the EQ-Paradigm unifies different 3D architectures. Notably, SECOND with point-based embeddings achieves 1.21% improvement over its voxel-based baseline. This demonstrates that different architectures have unique advantages. For example, point-based architectures extract more precise structural information.

Meanwhile, in Table 5, we show that voxel- and point-based embedding networks can be simultaneously utilized in an EQ-Paradigm model to yield further improvement. These experiments demonstrate that EQ-Paradigm is vastly flexible in backbone and head selection and is able to combine strengths of points and voxels.

Analysis on the Hierarchical Q-Net Multi-level features play an important role in recognition [5, 20, 70]. Our EQ-Paradigm is naturally compatible with the multi-level scheme by simply employing multi-level features as support features in the querying stage. We accordingly design a simple and yet effective hierarchical Q-Net structure. We validate the advantage of fusing multi-level information by conducting experiments on point cloud semantic segmentation, which calls for fine-grained features to better segment points on object boundaries.

Table 6 lists effect of incorporating levels of features in the querying stage on ScanNet validation set. We start from the coarsest layer and gradually include more finer features. Continuous performance improvement is observed with the increasing number of feature levels, manifesting the effectiveness of our hierarchical Q-Net.

No. of Levels	1	2	3	4	5	6
mIoU (%)	58.4	64.1	68.3	71.9	74.2	75.3

Table 6. Effects of the number of feature levels in our hierarchical Q-Net. The experiments are conducted on ScanNet validation set.

Method	SA	Query Position Selection		AP (%)
		train	test	
EQ-SECOND	√	patch	patch	81.61
	√	patch	random	74.96
	-	patch	patch	81.49
	-	patch	random	81.49

Table 7. AP comparison on EQ-SECOND utilizing Q-Decoder layer with or without self-attention ("SA") layers.

Analysis on the Q-Decoder In Q-Decoder layer, to make query positions independent of each other to allow arbitrary query position selection, we remove the self-attention layer for query points in the transformer decoder layer. In Table 7, we compare the performance of EQ-SECOND with and without the self-attention layer on different test modes. Both models are trained in "patch" mode, and tested in modes of "patch" and "random". In "patch" mode, we split the target BEV map into patches with equal sizes, randomly select one patch at each iteration, and treat all pixel centers within the patch as query positions. In "random" mode, we arbitrarily choose pixel centers within the BEV map as query positions.

The self-attention layer encodes the relation among query positions, thus restricting the choice of query positions at test time. Table 7 shows AP drop of 6.65% on the model with self-attention layer when tested with randomly-selected query positions. Great negative effect of self-attention layer to arbitrary query position selection is observed. In contrast, our model free of self-attention enables arbitrary selection without influencing performance. It is also noticeable that self-attention layer brings limited AP improvement (0.12%) and incurs large computation overhead when dealing with a large number of query positions.

6. Conclusion

We have presented a novel unified pipeline of EQ-Paradigm for 3D understanding tasks including object detection, semantic segmentation and classification. The EQ-Paradigm enables combination of 3D backbone architectures, heads and tasks freely. We achieve this by proposing a querying stage to transfer the support features extracted in the embedding stage to the positions required by heads and tasks. We further develop a dedicated Q-Net for the querying stage, applicable to different state-of-the-art models for clear performance improvement. In the future, we plan to generalize our EQ-Paradigm to other 3D tasks like scene completion and instance segmentation. We will extend the EQ-Paradigm as a unified pipeline for all 3D tasks.

References

[1] Iro Armeni, Ozan Sener, Amir Roshan Zamir, Helen Jiang, Ioannis K. Brilakis, Martin Fischer, and Silvio Savarese. 3d semantic parsing of large-scale indoor spaces. In *CVPR*, 2016. 6

[2] Matan Atzmon, Haggai Maron, and Yaron Lipman. Point convolutional neural networks by extension operators. *ACM Trans. Graph.*, 2018. 7

[3] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 4

[4] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *ECCV*, 2020. 4, 5

[5] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *TPAMI*, 2018. 2, 8

[6] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *ECCV*, 2018. 2

[7] Christopher B. Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *CVPR*, 2019. 1, 2, 3, 5, 6

[8] MMDetection3D Contributors. MMDetection3D: OpenMMLab next-generation platform for general 3D object detection. https://github.com/open-mmlab/mmdetection3d, 2020. 6, 7

[9] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas A. Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *CVPR*, 2017. 6

[10] Stéphane d'Ascoli, Hugo Touvron, Matthew L. Leavitt, Ari S. Morcos, Giulio Biroli, and Levent Sagun. Convit: Improving vision transformers with soft convolutional inductive biases. In *ICML*, 2021. 4

[11] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021. 3, 4

[12] Haoshu Fang, Chenxi Wang, Minghao Gou, and Cewu Lu. Graspnet-1billion: A large-scale benchmark for general object grasping. In *CVPR*, 2020. 1

[13] Marco Forte and François Pitié. F, b, alpha matting. *CoRR*, abs/2003.07711, 2020. 2

[14] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The KITTI dataset. *I. J. Robotics Res.*, 2013. 1, 7

[15] Ben Graham. Sparse 3d convolutional neural networks. In *BMVC*, 2015. 2, 3

[16] Benjamin Graham, Martin Engelcke, and Laurens van der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks. In *CVPR*, 2018. 1, 2, 3, 6

[17] Li Jiang, Hengshuang Zhao, Shu Liu, Xiaoyong Shen, Chi-Wing Fu, and Jiaya Jia. Hierarchical point-edge interaction network for point cloud semantic segmentation. In *ICCV*, 2019. 6

[18] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. *CVPR*, 2019. 2

[19] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on x-transformed points. In *NIPS*, 2018. 1, 2, 6, 7

[20] Tsung-Yi Lin, Piotr Dollár, Ross B. Girshick, Kaiming He, Bharath Hariharan, and Serge J. Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017. 8

[21] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott E. Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: single shot multibox detector. In *ECCV*, 2016. 2, 3

[22] Yongcheng Liu, Bin Fan, Gaofeng Meng, Jiwen Lu, Shiming Xiang, and Chunhong Pan. Densepoint: Learning densely contextual representation for efficient point cloud processing. In *ICCV*, 2019. 7

[23] Yongcheng Liu, Bin Fan, Shiming Xiang, and Chunhong Pan. Relation-shape convolutional neural network for point cloud analysis. In *CVPR*, 2019. 1, 2, 3, 7

[24] Ze Liu, Han Hu, Yue Cao, Zheng Zhang, and Xin Tong. A closer look at local aggregation operators in point cloud analysis. In *ECCV*, 2020. 7

[25] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *CoRR*, 2021. 4, 5

[26] Ze Liu, Zheng Zhang, Yue Cao, Han Hu, and Xin Tong. Group-free 3d object detection via transformers. *ICCV*, 2021. 2, 6, 7

[27] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. 2

[28] Jiageng Mao, Xiaogang Wang, and Hongsheng Li. Interpolated convolutional networks for 3d point cloud understanding. *ICCV*, 2019. 1, 2, 7

[29] Ishan Misra, Rohit Girdhar, and Armand Joulin. An End-to-End Transformer Model for 3D Object Detection. In *ICCV*, 2021. 4

[30] Xuran Pan, Zhuofan Xia, Shiji Song, Li Erran Li, and Gao Huang. 3d object detection with pointformer. In *CVPR*, 2021. 2, 4, 5

[31] Youngmin Park, Vincent Lepetit, and Woontack Woo. Multiple 3d object tracking for augmented reality. In *ISMAR*, 2008. 1

[32] Charles R Qi, Or Litany, Kaiming He, and Leonidas J Guibas. Deep hough voting for 3d object detection in point clouds. In *ICCV*, 2019. 2, 3, 6

[33] Charles R. Qi, Or Litany, Kaiming He, and Leonidas J. Guibas. Deep hough voting for 3d object detection in point clouds. *ICCV*, 2019. 2

[34] Charles Ruizhongtai Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J. Guibas. Frustum pointnets for 3d object detection from RGB-D data. *CVPR*, 2018. 2

[35] Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, 2017. 3, 6

[36] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *NIPS*, 2017. 1, 2, 3, 5, 6

[37] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015. 2

[38] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. In *NAACL-HLT*, 2018. 5

[39] 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 *CVPR*, 2020. 2, 7

[40] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *CVPR*, 2019. 2, 3, 7, 8

[41] Shaoshuai Shi, Zhe Wang, Xiaogang Wang, and Hongsheng Li. Part-aˆ2 net: 3d part-aware and aggregation neural network for object detection from point cloud. *arXiv preprint arXiv:1907.03670*, 2019. 2

[42] Martin Simonovsky and Nikos Komodakis. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *CVPR*, 2017. 2

[43] Shuran Song, Samuel P. Lichtenberg, and Jianxiong Xiao. SUN RGB-D: A RGB-D scene understanding benchmark suite. In *CVPR*, 2015. 6

[44] Yanan Sun, Chi-Keung Tang, and Yu-Wing Tai. Semantic image matting. In *CVPR*, 2021. 2

[45] Hugues Thomas, Charles R. Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J. Guibas. Kpconv: Flexible and deformable convolution for point clouds. *ICCV*, 2019. 2, 6, 7

[46] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *ICML*, 2021. 4

[47] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *NIPS*, 2017. 3, 4, 5

[48] Chu Wang, Babak Samari, and Kaleem Siddiqi. Local spectral graph convolution for point set feature learning. In *ECCV*, 2018. 2

[49] Chu Wang, Babak Samari, and Kaleem Siddiqi. Local spectral graph convolution for point set feature learning. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *ECCV*, 2018. 7

[50] Xintao Wang, Kelvin C.K. Chan, Ke Yu, Chao Dong, and Chen Change Loy. Edvr: Video restoration with enhanced deformable convolutional networks. In *CVPR*, 2019. 2

[51] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph CNN for learning on point clouds. *ACM Trans. Graph.*, 2019. 1, 7

[52] Zhixin Wang and Kui Jia. Frustum convnet: Sliding frustums to aggregate local point-wise features for amodal 3d object detection. In *IROS*. IEEE, 2019. 2

[53] Kan Wu, Houwen Peng, Minghao Chen, Jianlong Fu, and Hongyang Chao. Rethinking and improving relative position encoding for vision transformer. 2021. 5

[54] Wenxuan Wu, Zhongang Qi, and Fuxin Li. Pointconv: Deep convolutional networks on 3d point clouds. In *CVPR*, 2019. 2, 3, 6, 7

[55] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *CVPR*, 2015. 7

[56] Mutian Xu, Runyu Ding, Hengshuang Zhao, and Xiaojuan Qi. Paconv: Position adaptive convolution with dynamic kernel assembling on point clouds. In *CVPR*, 2021. 1, 2, 6

[57] Ning Xu, Brian L. Price, Scott Cohen, and Thomas S. Huang. Deep image matting. In *CVPR*, 2017. 2

[58] Qiangeng Xu, Xudong Sun, Cho-Ying Wu, Panqu Wang, and Ulrich Neumann. Grid-gcn for fast and scalable point cloud learning. In *CVPR*, 2020. 7

[59] Yifan Xu, Tianqi Fan, Mingye Xu, Long Zeng, and Yu Qiao. Spidercnn: Deep learning on point sets with parameterized convolutional filters. In *ECCV*, 2018. 2, 7

[60] Xu Yan, Chaoda Zheng, Zhen Li, Sheng Wang, and Shuguang Cui. Pointasnl: Robust point clouds processing using nonlocal neural networks with adaptive sampling. In *CVPR*, 2020. 6

[61] Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. *Sensors*, 2018. 1, 2, 3, 7, 8

[62] Jianwei Yang, Chunyuan Li, Pengchuan Zhang, Xiyang Dai, Bin Xiao, Lu Yuan, and Jianfeng Gao. Focal self-attention for local-global interactions in vision transformers. *CoRR*, 2021. 4, 5

[63] Zetong Yang, Yanan Sun, Shu Liu, and Jiaya Jia. 3dssd: Point-based 3d single stage object detector, 2020. 2, 3

[64] Zetong Yang, Yanan Sun, Shu Liu, Xiaoyong Shen, and Jiaya Jia. IPOD: intensive point-based object detector for point cloud. *CoRR*, 2018. 2

[65] Zetong Yang, Yanan Sun, Shu Liu, Xiaoyong Shen, and Jiaya Jia. STD: sparse-to-dense 3d object detector for point cloud. *ICCV*, 2019. 2

[66] Zetong Yang, Yin Zhou, Zhifeng Chen, and Jiquan Ngiam. 3d-man: 3d multi-frame attention network for object detection. In *CVPR*, 2021. 4

[67] Feihu Zhang, Jin Fang, Benjamin Wah, and Philip Torr. Deep fusionnet for point cloud semantic segmentation. In *ECCV*, 2020. 6

[68] Hengshuang Zhao, Li Jiang, Chi-Wing Fu, and Jiaya Jia. Pointweb: Enhancing local neighborhood features for point cloud processing. In *CVPR*, 2019. 2, 6, 7

[69] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip H. S. Torr, and Vladlen Koltun. Point transformer. In *ICCV*, 2021. 1, 2, 4, 5, 6

[70] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *CVPR*, 2017. 2, 8

[71] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. *CoRR*, 2019. 1

[72] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. *CVPR*, 2018. 2, 7