Eurographics Symposium on Geometry Processing 2023
P. Memari and J. Solomon
(Guest Editors)

Volume 42 (2023), Number 5

Cross-Shape Attention for Part Segmentation of 3D Point Clouds

paper1011

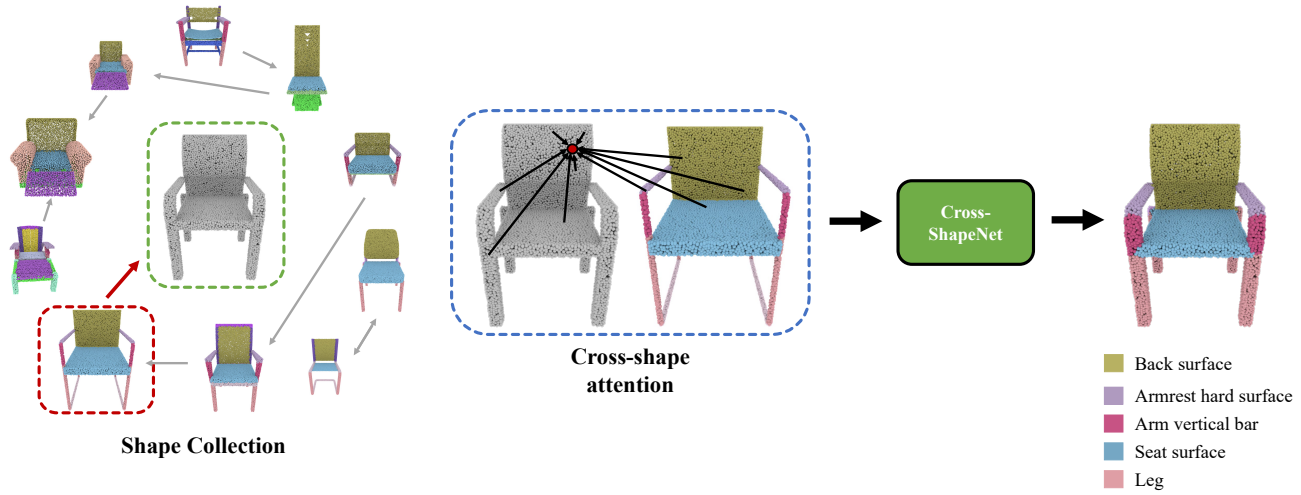

Figure 1: Left: *Given an input shape collection, our method constructs a graph where each shape is represented as a node and edges indicate shape pairs that are deemed compatible for cross-shape feature propagation.* Middle: *Our network is designed to compute point-wise feature representations for a given shape (grey shape) by enabling interactions between its own point-wise features and those of other shapes using our cross-shape attention mechanism.* Right: *As a result, the point-wise features of the shape become more synchronized with ones of other relevant shapes leading to more accurate fine-grained segmentation.*

Abstract

We present a deep learning method that propagates point-wise feature representations across shapes within a collection for the purpose of 3D shape segmentation. We propose a cross-shape attention mechanism to enable interactions between a shape's point-wise features and those of other shapes. The mechanism assesses both the degree of interaction between points and also mediates feature propagation across shapes, improving the accuracy and consistency of the resulting point-wise feature representations for shape segmentation. Our method also proposes a shape retrieval measure to select suitable shapes for cross-shape attention operations for each test shape. Our experiments demonstrate that our approach yields state-of-the-art results in the popular PartNet dataset.

CCS Concepts

• ***Computing methodologies*** → ***Point-based models;*** *Shape representations;* • ***Computer systems organization*** → ***Neural networks;***

1. Introduction

Learning effective point-based representations is fundamental to shape understanding and processing. In recent years, there has been significant research in developing deep neural architectures to learn point-wise representations of shapes through convolution and attention layers, useful for performing high-level tasks, such as shape segmentation. The common denominator of these networks is that they output a representation for each shape point by weighting and aggregating representations and relations with other points within the same shape.

In this work, we propose a *cross-shape attention* mechanism that enables interaction and propagation of point-wise feature representations across shapes of an input collection. In our architecture, the representation of a point in a shape is learned by combining representations originating from points in the same shape as well as other shapes. The rationale for such an approach is that if a point on one shape is related to a point on another shape e.g., they lie on geometrically or semantically similar patches or parts, then cross-shape attention can promote consistency in their resulting representations and part label assignments. We leverage neural attention

to determine and weigh pairs of points on different shapes. We integrate these weights in our cross-shape attention scheme to learn more consistent point representations for the purpose of semantic shape segmentation.

Developing such a cross-shape attention mechanism is challenging. Performing cross-attention across all pairs of shapes becomes prohibitively expensive for large input collections of shapes. Our method learns a measure that allows us to select a small set of other shapes useful for such cross-attention operations with a given input shape. For example, given an input office chair, it is more useful to allow interactions of its points with points of another structurally similar office chair rather than a stool. During training, we maintain a sparse graph (Figure 1), whose nodes represent training shapes and edges specify which pairs of shapes should interact for training our cross-shape attention mechanism. At test time, the shape collection graph is augmented with additional nodes representing test shapes. New edges are added connecting them to training shapes for propagating representations from relevant training shapes.

We tested our cross-shape attention mechanism on two different backbones to extract the initial point-wise features per shape for the task of part segmentation: a sparse tensor network based on MinkowskiNet [CGS19] and the octree-based network MID-FC [WYZ*21]. For both backbones, we observed that our mechanism significantly improves the point-wise features for segmentation. Compared to the MinkowskiNet baseline, we found an improvement of 3.1% in mean part IoU in the PartNet benchmark [MZC*19a]. Compared to MID-FC, we found an improvement of $+1.3\%$ in mean part IoU, achieving a new state-of-the-art result in PartNet (MID-FC: 60.8% $\rightarrow$ Ours: 62.1%).

In summary, our main technical contribution is an attention-based mechanism that enables point-wise feature interaction and propagation within and across shapes for more consistent segmentation. Our experiments show state-of-the-art performance on the recent PartNet dataset.

2. Related work

We briefly overview related work on 3D deep learning for point clouds. We also discuss cross-attention networks developed in other domains.

2.1. 3D deep learning for processing point clouds

Several different types of neural networks have been proposed for processing point sets over the recent years. After the pioneering work of PointNet [QSMG17, QYSG17], several works further investigated hierarchical point aggregation mechanisms to better model the spatial distribution of points [LCL18, SGS19, LKM19, LHC*20]. Alternatively, point clouds can be projected onto local views [SMKLM15, QSN*16, KAMC17, HKC*17] and processed as regular grids through image-based convolutional networks. Another line of work converts point representations into volumetric grids [WSK*15, MS15, DCS*17, RWS*18, SWL19, LTLH19] and processes them through 3D convolutions. Instead of uniform grids, hierarchical space partitioning structures (e.g., kd-trees, octrees, lattices) can be used to define regular convolutions [RUG17, KL17, WLG*17, WSLT18, SJS*18, WYZ*21]. Another type of networks incorporate point-wise convolution operators to directly process point clouds [LBS*18, HTY18, XLCT18, LFM*19, GWL18, AML18, HRV*18, WSM*18, XFX*18, WQF19, KZH19, TQD*19]. Alternatively, shapes can be treated as graphs by connecting each point to other points within neighborhoods in a feature space. Then graph convolution and pooling operations can be performed either in the spatial domain [WSL*19, SFYT18, LYYD19, WSS18, ZHW*19, LFXP19, LB19, LMQ*21, JZL*19, XSyW*19, WHH*19, HWW*19, LKM19], or spectral domain [YSGG17, BMM*15, BMRB16, MBM*17]. Attention mechanisms have also been investigated to modulate the importance of graph edges and point-wise convolutions [XLCT18, XSyW*19, WHH*19, YHJY19]. Graph neural network approaches have been shown to model non-local interactions between points within the same shape [WSL*19, LMQ*21, XSyW*19, HWW*19]. Finally, several recent works [ZJJ*21, GCL*21, EBD21, ML21, XWL*21, LLJ*22] introduced a variety of transformer-inspired models for point cloud processing tasks. None of the above approaches have investigated the possibility of extending attention across shapes. A notable exception are the methods by Wang et al. [WS19] and Cao et al. [CPBM21] that propose cross-attention mechanisms across given pairs of point cloud instances representing different transformations of the same underlying shape for the specific task of rigid registration. Our method instead introduces cross-attention across shapes within a large collection without assuming any pre-specified shape pairs. Our method aims to discover useful pairs for cross-shape attention and learns representations by propagating them within the shape collection. Our method shows that the resulting features yield more consistent 3D shape segmentation than several other existing point-based networks.

2.2. Cross-attention in other domains

Our method is inspired by recent cross-attention models proposed for video classification, image classification, keypoint recognition, and image-text matching. Wang et al. [WGGH18] introduced non-local networks that allow any image query position to perceive features of all the other positions within the same image or across frames in a video. To avoid huge attention maps, Huang et al. proposes a "criss-cross" attention module [HWH*19] to maintain sparse connections for each position in image feature maps. Cao et al. [CXL*19] simplifies non-local blocks with query-independent attention maps [CXL*19]. Lee et al. [LCH*18] propose cross-attention between text and images to discover latent alignments between image regions and words in a sentence. Hou et al. [HCB*19] models the semantic relevance between class and query feature maps in images through cross-attention to localize more relevant image regions for classification and generate more discriminative features. Sarlin et al. [SDMR19] learns keypoint matching between two indoor images from different viewpoints by leveraging self-attention and cross-attention to boost the receptive field of local descriptors and allow cross-image communication. Chen et al. [CFP21] propose cross-attention between multiscape representations for image classification. Finally, Doersch et al. [DGZ20] introduced a CrossTransformer model for few-shot learning on images. Give an unlabeled query image, their model computes local

cross-attention similarities with a number of labeled images and then infers class membership.

Our method instead introduces attention mechanisms across 3D shapes. In contrast to cross-attention approaches in the above domains, we do not assume any pre-existing paired data. The usefulness of shape pairs is determined based on a learned shape compatibility measure.

3. Method

Given an input collection of 3D shapes represented as point clouds, the goal of our method is to extract and propagate point-based feature representations from one shape to another, and use the resulting representations for 3D semantic segmentation. To perform the feature propagation, we propose a Cross-Shape Attention (CSA) mechanism. The mechanism first assesses the degree of interaction between pairs of points on different shapes. Then it allows point-wise features on one shape to influence the point-wise features of the other shape based on their assessed degree of interaction. In addition, we provide a mechanism that automatically selects shapes ("key shapes") to pair with an input test shape ("query shape") to execute these cross-shape attention operations. In the following sections, we first discuss the CSA layer at test time (Section 3.1). Then we discuss our retrieval mechanism to find key shapes given a test shape (Section 3.2), our training (Section 3.3), test stage (Section 3.4), and finally our network architecture details (Section 3.5).

3.1. Cross-shape attention for a shape pair

The input to our CSA layer is a pair of shapes represented as point clouds: $\mathcal{S}_m = \{\mathbf{p}_i\}_{i=1}^{P_m}$ and $\mathcal{S}_n = \{\mathbf{p}_j\}_{j=1}^{P_n}$ where $\mathbf{p}_i, \mathbf{p}_j \in \mathcal{R}^3$ represent 3D point positions and P_m, P_n are the number of points for each shape respectively. Our first step is to extract point-wise features for each shape.

In our implementation, we experimented with two backbones for point-wise feature extraction: a sparse tensor network based on a modified version of MinkowskiNet [CGS19], and an octree-based network, called MID-FC [WYZ*21] (architecture details for the two backbones are provided in Section 3.5 and supplementary material). The output from the backbone is a per-point D-dimensional representation stacked into a matrix for each of the two shapes respectively: $\mathbf{X}_m \in \mathcal{R}^{P_m \times D}$ and $\mathbf{X}_n \in \mathcal{R}^{P_n \times D}$. The CSA layer produces new D-dimensional point-wise representations for both shapes:

$$\boldsymbol{X}'_m = f\left(\boldsymbol{X}_m, \boldsymbol{X}_n; \boldsymbol{\theta}\right), \;\; \boldsymbol{X}'_n = f\left(\boldsymbol{X}_n, \boldsymbol{X}_m; \boldsymbol{\theta}\right) \tag{1}$$

where f is the cross-shape attention function with learned parameters θ described in the next paragraphs.

Key and query intermediate representations. Inspired by Transformers [VSP*17], we first transform the input point representations of the first shape in the pair to intermediate representations, called "query" representations. The input point representations of the second shape are transformed to intermediate "key" representations. The keys will be compared to queries to determine the degree of influence of one point on another. Specifically, these transformations are expressed as follows:

$$\boldsymbol{q}^{(h)}_{m,i} = \boldsymbol{W}^{(h)}_q \boldsymbol{x}_{m,i}, \;\; \boldsymbol{k}^{(h)}_{n,j} = \boldsymbol{W}^{(h)}_k \boldsymbol{x}_{n,j} \tag{2}$$

where $\boldsymbol{x}_{m,i}$ and $\boldsymbol{x}_{n,j}$ are point representations for the query shape $\mathcal{S}_m$ and key shape $\mathcal{S}_n$, $\boldsymbol{W}^{(h)}_q$ and $\boldsymbol{W}^{(h)}_k$ are $D' \times D$ learned transformation matrices shared across all points of the query and key shape respectively, and h is an index denoting each different transformation ("head"). The dimensionality of the key and query representations D' is set to $\lfloor D/H \rfloor$, where H is the number of heads. These intermediate representations are stacked into the matrices $\boldsymbol{Q}^{(h)}_m \in \mathcal{R}^{P_m \times D'}$ and $\boldsymbol{K}^{(h)}_n \in \mathcal{R}^{P_n \times D'}$. Furthermore, the point representations of the key shape $\mathcal{S}_n$ are transformed to value representations as:

$$\boldsymbol{v}^{(h)}_{n,j} = \boldsymbol{W}^{(h)}_v \boldsymbol{x}_{n,j} \tag{3}$$

where $\boldsymbol{W}^{(h)}_v$ is a learned $D' \times D$ transformation shared across the points of the key shape. These are also stacked to a matrix $\boldsymbol{V}^{(h)}_n \in \mathcal{R}^{P_n \times D'}$.

Pairwise point attention. The similarity of key and query representations is determined through scaled dot product [VSP*17]. This provides a measure of how much one shape point influences the point on the other shape. The similarity of key and query representations is determined for each head as:

$$\boldsymbol{A}^{(h)}_{m,n} = softmax\left(\frac{\boldsymbol{Q}^{(h)}_m \cdot \left(\boldsymbol{K}^{(h)}_n\right)^\top}{\sqrt{D'}}\right) \tag{4}$$

where $\boldsymbol{A}^{(h)}_{m,n} \in \mathcal{R}^{P_m \times P_n}$ is a cross-attention matrix between the two shapes for each head.

Feature representation updates. The cross-attention matrix is used to update the point representations for the query shape $\mathcal{S}_m$:

$$\boldsymbol{z}^{(h)}_{m,i} = \sum_{j=1}^{P_n} \boldsymbol{A}^{(h)}_{m,n}[i,j] \boldsymbol{W}^{(h)}_v \boldsymbol{x}_{n,j} \tag{5}$$

The point-wise features are concatenated across all heads, then a linear transformation layer projects them back to D-dimensional space and they are added back to the original point-wise features of the query shape:

$$\boldsymbol{x}'_{m,i} = \boldsymbol{x}_{m,i} + \mathbf{W}_d \cdot [\boldsymbol{z}^{(1)}_{m,i}, \boldsymbol{z}^{(2)}_{m,i}, ..., \boldsymbol{z}^{(H)}_{m,i}] \tag{6}$$

where H is the number of heads and $\mathbf{W}_d$ is another linear transformation. The features are stacked into a matrix $\boldsymbol{X}'_m \in \mathcal{R}^{P_m \times D}$, followed by layer normalization [BKH16].

Self-shape attention. The pairwise attention of Equation 4 and update operation of Equation 5 can also be applied to a pair that consists of the shape and itself. In this case, the CSA layer is equivalent to Self-Shape Attention (SSA), enabling long-range interactions between shape points within the same shape.

Cross-shape attention for multiple shapes. We can further generalize the cross-shape operation in order to handle multiple shapes and also combine it with self-shape attention. Given a selected set

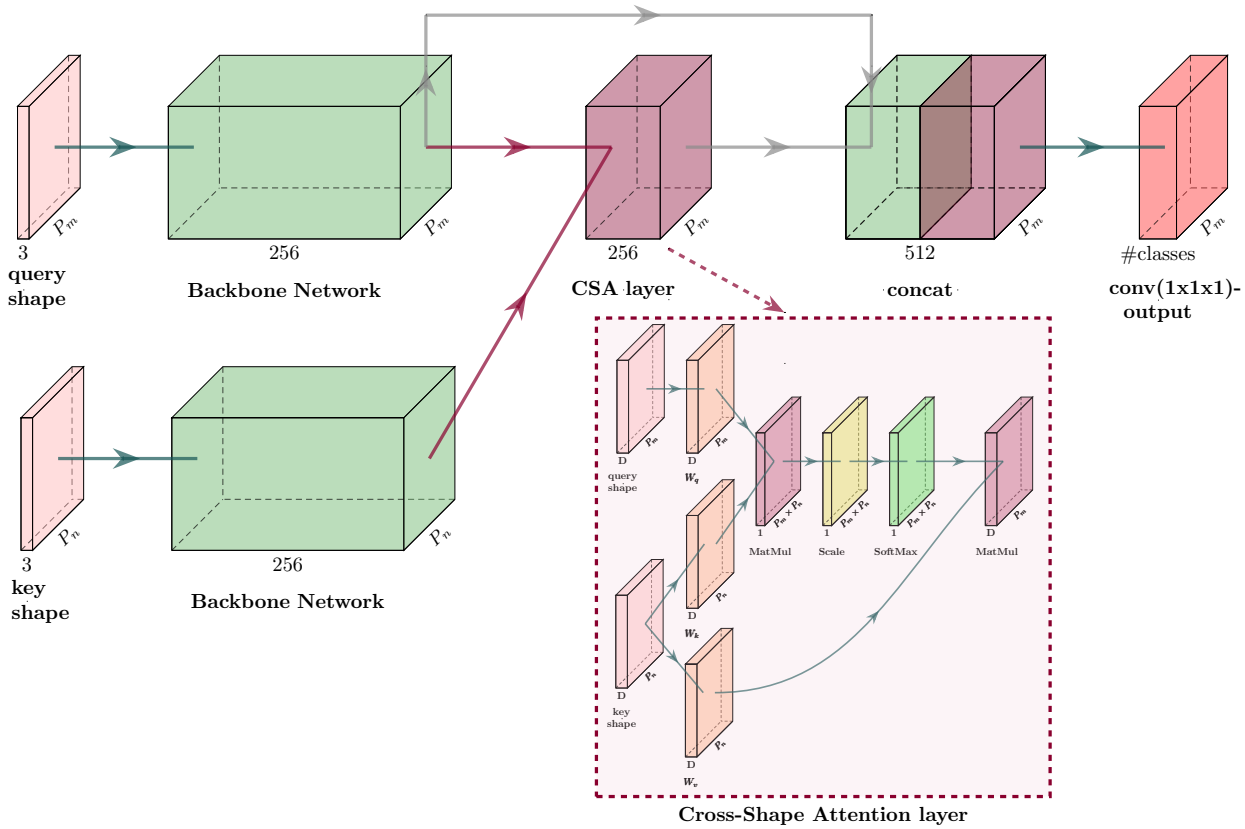

Figure 2: *Our cross-shape network architecture: given an input test shape ("query shape") represented as an input point set, we first extract initial point-wise features through a backbone (our MinkowskiNet variant, called "MinkNetHRNet", or alternatively the MID-FC network [WYZ*21]). Then our proposed cross-attention layer, called CSA layer, propagates features extracted from another shape of the input shape collection ("key shape") to the query shape such that their semantic segmentation becomes more synchronized. The output point-wise features of the CSA layer are concatenated with the original features of the query shape, then they are passed to a classification layer for semantic segmentation. Note that the illustrated CSA layer in the inlet figure uses only one head ($H = 1$).*

of key shapes, our CSA layer outputs point representations for the query shape $\mathcal{S}_m$ as follows:

$$\boldsymbol{X}'_m = \sum_{n \in \{\mathcal{C}(m), m\}} c(m,n) \boldsymbol{A}_{m,n} \boldsymbol{V}_n \tag{7}$$

where $\mathcal{C}(m)$ is a set of key shapes deemed compatible for cross-shape attention with shape $\mathcal{S}_m$ and $c(m,n)$ is a learned pairwise compatibility function between the query shape $\mathcal{S}_m$ and each key shape $\mathcal{S}_n$. The key idea of the above operation is to update point representations of the query shape $\mathcal{S}_m$ as a weighted average of attention-modulated representations computed by using other key shapes as well as the shape itself. The compatibility function $c(m,n)$ assesses these weights that different shapes should have for cross-shape attention. It also implicitly provides the weight of self-shape attention when $\mathcal{S}_m = \mathcal{S}_n$.

Compatibility function. To compute the compatibility function, we first extract a global, D-dimensional vector representation $\boldsymbol{y}_m$ and $\boldsymbol{y}_n$ for the query shape $\mathcal{S}_m$ and each key shape $\mathcal{S}_n$ respectively through mean-pooling on their self-shape attention representations:

$$\boldsymbol{y}_m^{(SSA)} = \text{avg}_i \boldsymbol{X}'^{(SSA)}_{m,i} = \text{avg}_i \left(\boldsymbol{A}_{m,m} \boldsymbol{V}_m \right) \tag{8}$$

$$\boldsymbol{y}_n^{(SSA)} = \text{avg}_i \boldsymbol{X}'^{(SSA)}_{n,i} = \text{avg}_i \left(\boldsymbol{A}_{n,n} \boldsymbol{V}_n \right) \tag{9}$$

In this manner, the self-attention representations of both shapes provide cues for the compatibility between them expressed using their scaled dot product similarity [VSP*17]:

$$\begin{aligned} \boldsymbol{u}_m &= \boldsymbol{U}_q \boldsymbol{y}_m^{(SSA)} \\ \boldsymbol{u}_n &= \boldsymbol{U}_k \boldsymbol{y}_n^{(SSA)} \\ s(m,n) &= \hat{\boldsymbol{u}}_m \cdot \hat{\boldsymbol{u}}_n^\top \end{aligned} \tag{10}$$

where $\boldsymbol{U}_q$ and $\boldsymbol{U}_k$ are learned $D \times D$ transformations for the query and key shape respectively, and $\hat{\boldsymbol{u}}_m = \boldsymbol{u}_m / ||\boldsymbol{u}_m||, \hat{\boldsymbol{u}}_n = \boldsymbol{u}_n / ||\boldsymbol{u}_n||$. The final compatibility function $c(m,n)$ is computed as a normalized measure using a softmax transformation of compatibilities of the shape m with all other shapes in the set $\mathcal{C}(m)$, including the

self-compatibility:

$$c(m,n) = \frac{exp\big(s(m,n)\big)}{\sum_{n\in\{C(m),m\}} exp\big(s(m,n)\big)} \tag{11}$$

3.2. Key shape retrieval

To perform cross-shape attention, we need to retrieve one or more key shapes for each query shape. One possibility is to use the measure of Eq. 10 to evaluate the compatibility of the query shape with each candidate key shape from an input collection. However, we found that this compatibility is more appropriate for the particular task of weighting the contribution of each selected key shape for cross-shape attention, rather than retrieving key shapes themselves (see supplementary for additional discussion). We instead found that it is better to retrieve key shapes whose point-wise representations are on average more similar to the ones of the query shape. To achieve this, we perform the following steps:

(i) We compute the similarity between points of the query shape and the points of candidate key shapes in terms of cosine similarity of their SSA representations:

$$\boldsymbol{S}_{m,n} = \boldsymbol{X}_m'^{(SSA)} \cdot (\boldsymbol{X}_n'^{(SSA)})^\top \tag{12}$$

where $\boldsymbol{S}_{m,n} \in \mathcal{R}^{P_m \times P_n}$.

(ii) Then for each query point, we find its best matching candidate key shape point yielding the highest cosine similarity:

$$r_i(m,n) = \max_j \boldsymbol{S}_{m,n}[i,j] \tag{13}$$

(iii) Finally, we compute the average of these highest similarities across all query points:

$$r(m,n) = \underset{i}{avg}\, r_i(m,n) \tag{14}$$

The retrieval measure $r(m,n)$ is used to compare the query shape $\mathcal{S}_n$ with candidate key shapes from a collection.

3.3. Training

The input to our training procedure is a collection of point clouds with part annotations along with a smaller annotated collection used for hold-out validation. We first train our backbone including a layer that implements self-shape attention alone according to Eq. 3-6 (i.e., $\mathcal{S}_m = \mathcal{S}_n$ in this case). The resulting output features are passed to a softmax layer for semantic segmentation. The network is trained according to softmax loss. Based on the resulting SSA features, we construct a graph (Figure 1), where each training shape is connected with K shapes, deemed as "key" shapes, according to our retrieval measure of Eq. 14. One such graph is constructed for the training split, and another for the validation split. We then fine-tune our backbone and train a layer that implements our full cross-shape attention involving all K key shapes per training shape using the same loss. During training, we measure the performance of the network on the validation split, in terms of part IoU [MZC*19b], and if it reaches a plateau state, we recalculate the K-neighborhood of each shape based on the updated features. We further fine-tune our backbone and CSA layer. This iteration of graph update and fine-tuning of our network is performed two times in our implementation.

Variant	**avg part IoU**
MinkResUNet	46.8
MinkHRNet	48.0
MinkHRNetCSN-SSA	48.7
MinkHRNetCSN-K1	**49.9**
MinkHRNetCSN-K2	49.7
MinkHRNetCSN-K3	47.2
MID-FC	60.8
MID-FC-SSA	61.8
MID-FC-CSN-K1	61.9
MID-FC-CSN-K2	61.9
MID-FC-CSN-K3	62.0
MID-FC-CSN-K4	**62.1**
MID-FC-CSN-K5	62.0

Table 1: *Ablation study for all our variants in PartNet.*

3.4. Inference

During inference, we create a graph connecting each test shape with K training shapes retrieved by the measure of Eq. 14. We then perform a feed-forward pass through our backbone, CSA layer, and classification layer to assess the label probabilities for each test shape.

3.5. Architecture

Here we describe the two backbones (MinkNetHRNet, MID-FC) we used to provide point-wise features to our CSA layer.

MinkNetHRNet. The first backbone is a variant of the sparse tensor network based on MinkowskiNet [CGS19]. We note that our variant performed better than the original MinkowskiNet for 3D segmentation [CGS19], as discussed in our experiments. In a preprocessing step, we normalize the point clouds to a unit sphere and convert them to a sparse voxel grid (voxel size = 0.05). After two convolutional layers, the network branches into three stages inspired by the HRNet [WSC*21], a network that processes 2D images in a multi-resolution manner. In our case, the first stage consists of three residual blocks processing the sparse voxel grid in its original resolution. The second stage downsamples the voxel grid by a factor of 2 and processes it through two other residual blocks. The third stage further downsamples the voxel grid by a factor of 2 and processes it through another residual block. The multi-resolution features from the three stages are combined into one feature map through upsampling following [WSC*21]. The resulting feature map is further processed by a $1D$ convolutional block. The sparse voxel features are then mapped back to points as done in the original MinkowskiNet [CGS19]. Details about the architecture of this backbone are provided in the supplementary.

MID-FC. The second variant utilizes an octree-based architecture based on the MID-FC network [WYZ*21]. This network also incorporates a three-stage HRNet [WSC*21] to effectively maintain and merge multi-scale resolution feature maps. To implement this architecture, each point cloud is first converted into an octree representation with a resolution of 64^3. To train this network,

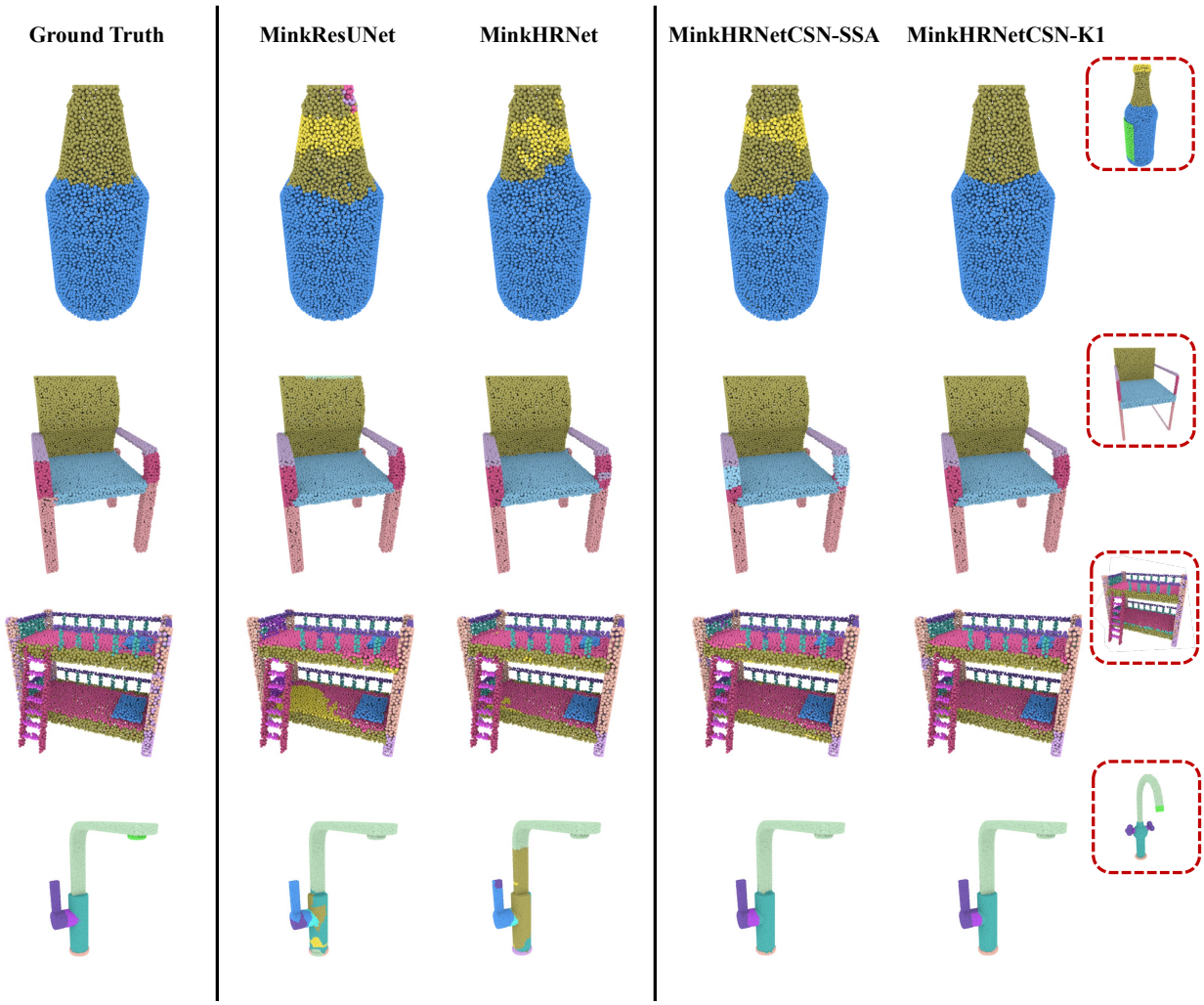

Figure 3: *Qualitative comparisons for a few characteristic test shapes of PartNet between the original MinkowskiNet for 3D shape segmentation ("MinkResUNet"), our backbone ("MinkHRNet"), and CrossShapeNet (CSN) in case of using self-shape attention alone ("MinkHRNetCSN-SSA") and using cross-shape attention with $K = 1$ key shape per query shape ("MinkHRNetCSN-K1"). The inlet images (red dotted box) show this key shape retrieved for each of the test shapes.*

a self-supervised learning approach is employed using a multi-resolution instance discrimination pretext task with ShapeNet-Core55 [WYZ*21]. The training process involves two losses: a shape instance discrimination loss to classify augmented copies of each shape instance and a point instance discrimination loss to classify the same points on the augmented copies of a shape instance. This joint learning approach enables the network to acquire generic shape and point encodings that can be used for shape analysis tasks. Finally, the pre-trained network is combined with two fully-connected layers and our CSA layer. During training for our segmentation task, the HRNet is frozen, while we train only the two fully-connected layers and CSA layer for efficiency reasons. Details about the architecture of this backbone are provided in the supplementary.

3.6. Implementation details

We train our Cross Shape Network for each object category of PartNet [MZC*19b] separately, using the standard cross entropy loss, for 200 epochs. We set the batch size equal to 8 for all variants (SSA, K=1,2,3). For optimization we use the SGD optimizer [Rud16] with a learning rate of 0.5 and momentum = 0.9. We scale learning rate by a factor of 0.5, whenever the loss of the hold-out validation split saturates (patience = 10 epochs, cooldown = 10 epochs). For updating the shape graph for the training and validation split, we measure the performance of the validation split in terms of Part IoU. If it reaches a saturation point (patience = 10 epochs, cooldown = 5 epochs), we load the best model up to that moment, based on Part IoU performance, and update the graph for both splits. The graph is updated twice throughout our training procedure. For all layers we use batch normalization [IS15] with momentum = 0.02, except for the CSA module, where the layer nor-

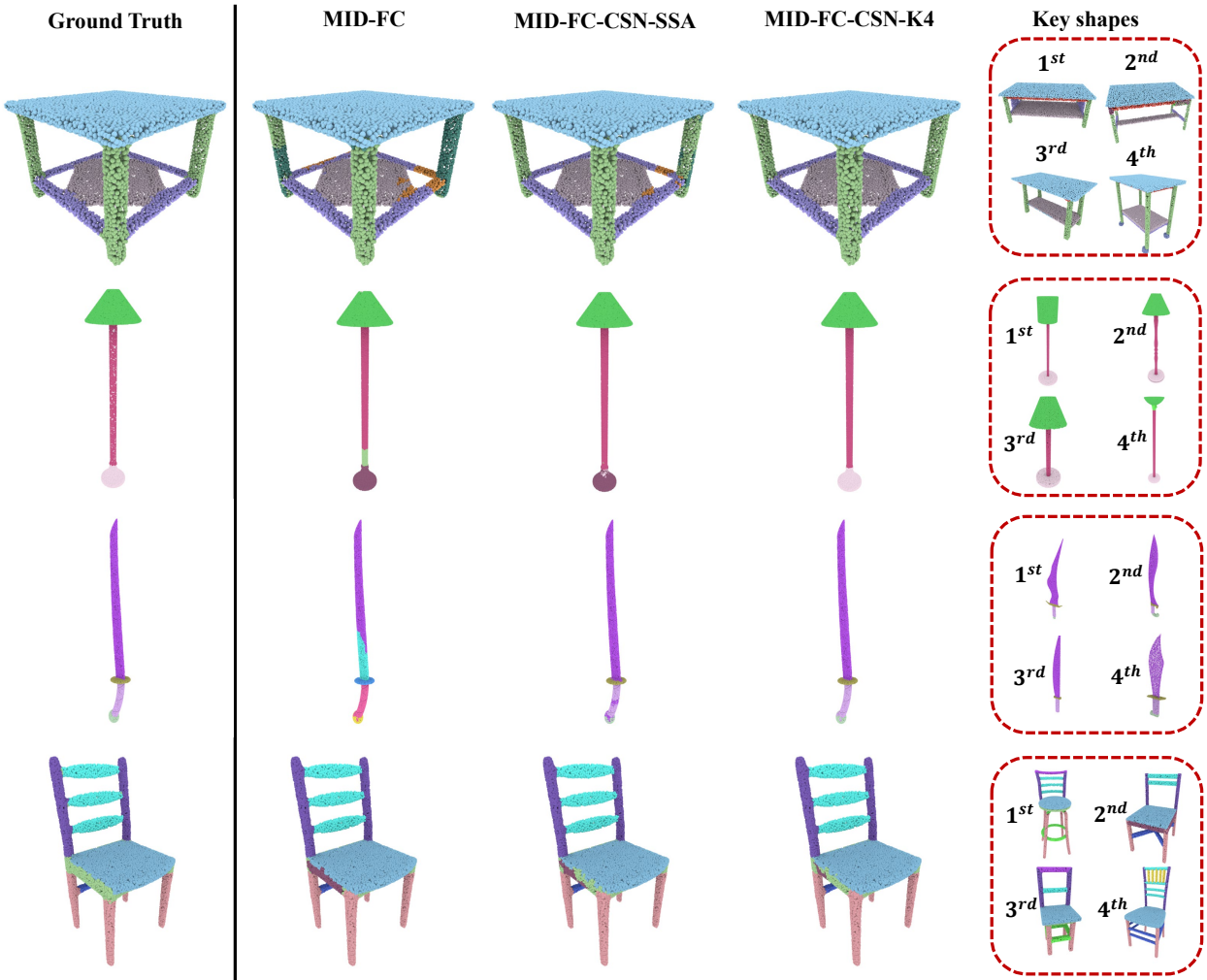

Figure 4: *Qualitative comparisons for a few characteristic test shapes of PartNet between the original MID-FC network for 3D shape segmentation ("MID-FC") [WYZ*21], and CrossShapeNet (CSN) in case of using self-shape attention alone ("MID-FC-CSN-SSA") and using cross-shape attention with $K = 4$ key shape per query shape ("MID-FC-CSN-K4"). The last column shows the key shapes and their ordering, retrieved for each test shape.*

malization [BKH16] is adopted. We also refer readers to our project page with source code for more details. †

4. Results

We evaluated our method for fine-grained shape segmentation qualitatively and quantitatively. In the next paragraphs, we discuss the used dataset, evaluation metrics, comparisons, and an analysis considering the computation time and size of our CSA layer.

Dataset. We use the PartNet dataset [MZC*19b] for training and evaluating our method according to its provided training, validation, and testing splits. Our evaluation focuses on the fine-grained level of semantic segmentation, which includes 17 out of the 24 object categories present in the PartNet dataset. We trained our network and competing variants separately for each object category.

† *Our code and project page will become available upon acceptance.*

Evaluation Metrics. For evaluating the performance of our method and variants, we used the standard Part IoU metric, as also proposed in the PartNet benchmark [MZC*19b]. The goal of our evaluation is to verify the hypothesis that our self-attention and cross-shape attention mechanisms yield better features for segmentation than the ones produced by any of the two original backbones on the task of semantic shape segmentation.

Ablation. Table 1 reports the mean part IoU performance averaged the PartNet's part categories for the original backbones ("MinkHRNet") and ("MID-FC"). We first observe that our backbone variant "MinkHRNet" improves over the original "MinkResUNet" proposed in [CGS19], yielding an improvement of 1.2% in mean Part IoU. Our variant based on self-shape attention alone ("MinkHRNetCSN-SSA") further improves our backbone by 0.7% in Part IoU. We further examined the performance of our cross-shape attention (CSA layer) tested in the variants "MinkHRNetCSN-K1", "MinkHRNetCSN-K2", and

Category	Bed	Bott	Chai	Cloc	Dish	Disp	Door	Ear	Fauc	Knif	Lamp	Micr	Frid	Stor	Tabl	Tras	Vase	**avg.**	**#cat.**
SpiderCNN [XFX*18]	36.2	32.2	30.0	24.8	50.0	80.1	30.5	37.2	44.1	22.2	19.6	43.9	39.1	44.6	20.1	42.4	32.4	37.0	0
PointNet++ [QYSG17]	30.3	41.4	39.2	41.6	50.1	80.7	32.6	38.4	52.4	34.1	25.3	48.5	36.4	40.5	33.9	46.7	49.8	42.5	0
ResGCN-28 [LMQ*21]	35.9	49.3	41.1	33.8	56.2	81.0	31.1	45.8	52.8	44.5	23.1	51.8	34.9	47.2	33.6	50.8	54.2	45.1	0
PointCNN [LBS*18]	41.9	41.8	43.9	36.3	58.7	82.5	37.8	48.9	60.5	34.1	20.1	58.2	42.9	49.4	21.3	53.1	58.9	46.5	0
CloserLook3D [LHC*20]	49.5	49.4	48.3	49.0	65.6	84.2	56.8	53.8	62.4	39.3	24.7	61.3	55.5	54.6	44.8	56.9	58.2	53.8	0
MinkResUNet [CGS19]	39.4	44.2	42.3	35.4	57.8	82.4	33.9	45.8	57.8	46.7	25.0	53.7	40.5	45.0	35.7	50.6	58.8	46.8	0
MinkHRNetCSN-K1 (ours)	42.1	54.0	42.5	42.9	58.2	83.2	43.5	51.5	59.4	47.8	27.9	57.4	43.7	46.2	36.8	51.5	60.0	49.9	0
MID-FC [WYZ*21]	51.6	56.5	**55.7**	55.3	75.6	91.3	56.6	53.8	64.6	55.4	31.2	78.7	63.1	62.8	45.7	65.8	69.3	60.8	1
MID-FC-CSN-K4 (ours)	**52.2**	**58.6**	**55.7**	**57.7**	**76.4**	**91.4**	**58.9**	**54.5**	**65.2**	**62.2**	**33.1**	**79.2**	**64.0**	**62.9**	**46.0**	**67.2**	**69.9**	**62.1**	**16**

Table 2: *Comparisons with other methods reporting performance in PartNet. The column "avg." reports the mean Part IoU (averaged over all* 17 *categories). The last column "#cat" counts the number of categories that a method wins over others.*

"MinkHRNetCSN-K3", where we use $K = 1, 2, 3$ key shapes per query shape. Our CrossShapeNet with $K = 1$ ("MinkHRNetCSN-K1") offers the best performance on average by improving Part IoU by another 1.2% with respect to using self-shape attention alone. When using $K = 2$ key shapes in cross-shape attention, the performance drops slightly (-0.2% in Part IoU on average) compared to using $K = 1$, and drops even more when using $K = 3$. Thus, for the MinkowskiNet variants, it appears that the optimal number of key shapes is $K = 1$; we suspect that the performance drop for higher K is due to the issue that the chance of retrieving shapes that are incompatible to the query shape is increased with larger numbers of retrieved key shapes.

We also observe improvements using the MID-FC backbone. Note that this backbone has higher performance than the MinkowskiNet variants due to its pretraining and fine-tuning strategies [WYZ*21]. Our variant based on self-shape attention alone ("MID-FC-SSA") further improves the original MID-FC backbone by 1.0% in mean Part IoU. When using cross-shape attention, the optimal performance is achieved when using $K = 4$ key shapes ("MID-FC-CSN-K4"), which improves Part IoU by another 0.3% with respect to using self-shape attention alone. We note that the above improvements are quite stable – by repeating all experiments 15 times, the standard deviation of mean Part IoU is $\sigma = 0.03\%$. This means that the above differences are significant – even the improvement of 0.3% of "MID-FC-CSN-K4" over "MID-FC-SSA" is of scale 10σ.

Comparisons with other methods. Table 2 includes comparisons with other methods reporting their performance on PartNet per category [XFX*18, QYSG17, LMQ*21, LBS*18, LHC*20, WYZ*21] along with our best performing variants ("MinkHRNetCSN-K1" and "MID-FC-CSN-K4"). *Compared to the original MinkowskiNet ("MinkResUNet"), our "MinkHRNetCSN-K1" variant achieves an improvement of* 3.1% *in terms of mean Part IoU in PartNet. Compared to "MID-FC", our best variant ("MID-FC-CSN-K4") also offers a noticeable improvement of* 1.3% *in mean Part IoU. To the best of our knowledge, the result of our best variant represents the highest mean Part IoU performance achieved in the PartNet benchmark so far. As it can been in the last column of Table 2, our method improves performance for* 16 *out of* 17 *categories.*

Qualitative Results. Figures 3 shows qualitative comparisons for MinkowskiNet-based variants – specifically our best variant in this case using cross-shape attention with $K = 1$, self-shape attention, our MinkNetHR backbone, and the original MinkowskiNet. Our backbone often improves the labeling relative to the original MinkowskiNet (e.g., see bed mattress, or monitor base). Our cross-shape attention tends to further improve upon fine-grained details in the segmentation e.g., see the top of the bottle, the armrests in the chair, and the bottom of the blade, pushing the segmentation to be more consistent with the retrieved key shape shown in the inlet images. Figure 4 shows comparisons for the MID-FC-based variants, including using cross-shape attention with $K = 4$, self-shape attention, and the original MID-FC. We can drive similar conclusions – our method improves the consistency of segmentation especially for fine-grained details e.g., the lower bars of the table, the bottom of the lamp, the handle of the sword, and the sides of the seat.

Number of parameters. Our CSA layer adds a relatively small overhead in terms of number of parameters. The MID-FC backbone has $1.8M$ parameters, the MinkHRNet has $24.8M$ parameters, while the CSA layer adds $0.4M$ parameters.

Computation. Training our CSA layer for $K = 1$ takes 105 hours on a NVidia $V100$ for the largest PartNet category (3.5x more compared to training either backbone alone) due to the iterative graph construction and fine-tuning discussed in Section 3.3. The inference time per test shape increases linearly with the input collection size used for retrieval. In our experiments, testing time ranges from 0.7 sec ("Dish" class with the smallest number of shapes) to 7.8 sec ("Table" class with the largest number of shapes).

5. Conclusion

We presented a method that enables interaction of point-wise features across different shapes in a collection. The interaction is mediated through a new cross-shape attention mechanism. Our experiments show improvements of this interaction in the case of fine-grained shape segmentation.

Limitations. First, we note that the performance increase comes with a higher computational cost at training and test time. It would be interesting to explore if further performance gains can be achieved through self-supervised pre-training of point correspondences as done in e.g., PointContrast [XGG*20] that could in turn guide the attention mechanism. Sparsifying the attention mechanism [CGRS19] would be beneficial in this case to handle the quadratic computational complexity of our current attention mechanism. Accelerating the key shape retrieval mechanism would also be useful to further decrease the test time. Another future research direction is to explore how to generalize the cross-shape attention mechanism from single shapes to entire scenes.

References

[AML18] ATZMON M., MARON H., LIPMAN Y.: Point Convolutional Neural Networks by Extension Operators. *ACM Trans. on Graphics 37*, 4 (2018). 2

[BKH16] BA J., KIROS J. R., HINTON G. E.: Layer Normalization. *arXiv preprint arXiv:1607.06450* (2016). 3, 7

[BMM*15] BOSCAINI D., MASCI J., MELZI S., BRONSTEIN M. M., CASTELLANI U., VANDERGHEYNST P.: Learning class-specific descriptors for deformable shapes using localized spectral convolutional networks. *Computer Graphics Forum 34*, 5 (2015). 2

[BMRB16] BOSCAINI D., MASCI J., RODOLÀ E., BRONSTEIN M.: Learning shape correspondence with anisotropic convolutional neural networks. In *Proc. NeurIPS* (2016). 2

[CFP21] CHEN C.-F. R., FAN Q., PANDA R.: CrossViT: Cross-Attention Multi-Scale Vision Transformer for Image Classification. In *Proc. ICCV* (2021). 2

[CGRS19] CHILD R., GRAY S., RADFORD A., SUTSKEVER I.: Generating Long Sequences with Sparse Transformers. *arXiv preprint arXiv:1904.10509* (2019). 8

[CGS19] CHOY C., GWAK J., SAVARESE S.: 4D Spatio-Temporal ConvNets: Minkowski Convolutional Neural Networks. In *Proc. CVPR* (2019). 2, 3, 5, 7, 8

[CPBM21] CAO A.-Q., PUY G., BOULCH A., MARLET R.: PCAM: Product of Cross-Attention Matrices for Rigid Registration of Point Clouds. In *Proc. ICCV* (2021). 2

[CXL*19] CAO Y., XU J., LIN S., WEI F., HU H.: GCNet: Non-local networks meet squeeze-excitation networks and beyond. In *Proc. CVPR Workshops* (2019). 2

[DCS*17] DAI A., CHANG A. X., SAVVA M., HALBER M., FUNKHOUSER T., NIESSNER M.: ScanNet: Richly-annotated 3d reconstructions of indoor scenes. In *Proc. CVPR* (2017). 2

[DGZ20] DOERSCH C., GUPTA A., ZISSERMAN A.: Crosstransformers: spatially-aware few-shot transfer. In *Proc. NeurIPS* (2020). 2

[EBD21] ENGEL N., BELAGIANNIS V., DIETMAYER K.: Point transformer. *IEEE Access 9* (2021). 2

[GCL*21] GUO M.-H., CAI J.-X., LIU Z.-N., MU T.-J., MARTIN R. R., HU S.-M.: Pct: Point cloud transformer. *Computational Visual Media 7*, 2 (2021). 2

[GWL18] GROH F., WIESCHOLLEK P., LENSCH H. P. A.: Flex-Convolution (Million-Scale Point-Cloud Learning Beyond Grid-Worlds). In *Proc. ACCV* (2018). 2

[HCB*19] HOU R., CHANG H., BINGPENG M., SHAN S., CHEN X.: Cross Attention Network for Few-shot Classification. In *Proc. NeurIPS* (2019). 2

[HKC*17] HUANG H., KALOGERAKIS E., CHAUDHURI S., CEYLAN D., KIM V. G., YUMER E.: Learning Local Shape Descriptors from Part Correspondences with Multiview Convolutional Networks. *ACM Trans. on Graphics 37*, 1 (2017). 2

[HRV*18] HERMOSILLA P., RITSCHEL T., VÁZQUEZ P.-P., VINACUA A., ROPINSKI T.: Monte Carlo Convolution for Learning on Non-Uniformly Sampled Point Clouds. *ACM Trans. on Graphics 37*, 6 (2018). 2

[HTY18] HUA B.-S., TRAN M.-K., YEUNG S.-K.: Pointwise convolutional neural networks. In *Proc. CVPR* (2018). 2

[HWH*19] HUANG Z., WANG X., HUANG L., HUANG C., WEI Y., LIU W.: CCNet: Criss-cross attention for semantic segmentation. In *Proc. CVPR* (2019). 2

[HWW*19] HAN W., WEN C., WANG C., LI X., LI Q.: Point2Node: Correlation Learning of Dynamic-Node for Point Cloud Feature Modeling. *arXiv preprint arXiv:1912.10775* (2019). 2

[IS15] IOFFE S., SZEGEDY C.: Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *Proc. ICML* (2015). 6

[JZL*19] JIANG L., ZHAO H., LIU S., SHEN X., FU C.-W., JIA J.: Hierarchical point-edge interaction network for point cloud semantic segmentation. In *Proc. CVPR* (2019). 2

[KAMC17] KALOGERAKIS E., AVERKIOU M., MAJI S., CHAUDHURI S.: 3D shape segmentation with projective convolutional networks. In *Proc. CVPR* (2017). 2

[KL17] KLOKOV R., LEMPITSKY V.: Escape from cells: Deep KD-networks for the recognition of 3d point cloud models. In *Proc. CVPR* (2017). 2

[KZH19] KOMARICHEV A., ZHONG Z., HUA J.: A-CNN: Annularly convolutional neural networks on point clouds. In *Proc. CVPR* (2019). 2

[LB19] LANDRIEU L., BOUSSAHA M.: Point cloud oversegmentation with graph-structured deep metric learning. In *Proc. CVPR* (2019). 2

[LBS*18] LI Y., BU R., SUN M., WU W., DI X., CHEN B.: PointCNN: Convolution On X-Transformed Points. In *Proc. NeurIPS* (2018). 2, 8

[LCH*18] LEE K.-H., CHEN X., HUA G., HU H., HE X.: Stacked cross attention for image-text matching. In *Proc. ECCV* (2018). 2

[LCL18] LI J., CHEN B., LEE G.: SO-Net: Self-Organizing Network for Point Cloud Analysis. In *Proc. CVPR* (2018). 2

[LFM*19] LIU Y., FAN B., MENG G., LU J., XIANG S., PAN C.: DensePoint: Learning Densely Contextual Representation for Efficient Point Cloud Processing. In *Proc. ICCV* (2019). 2

[LFXP19] LIU Y., FAN B., XIANG S., PAN C.: Relation-shape convolutional neural network for point cloud analysis. In *Proc. CVPR* (2019). 2

[LHC*20] LIU Z., HU H., CAO Y., ZHANG Z., TONG X.: A Closer Look at Local Aggregation Operators in Point Cloud Analysis. In *Proc. ECCV* (2020). 2, 8

[LKM19] LE E.-T., KOKKINOS I., MITRA N. J.: Going Deeper with Point Networks. *arXiv preprint arXiv:1907.00960* (2019). 2

[LLJ*22] LAI X., LIU J., JIANG L., WANG L., ZHAO H., LIU S., QI X., JIA J.: Stratified transformer for 3d point cloud segmentation. In *Proc. CVPR* (2022). 2

[LMQ*21] LI G., MÜLLER M., QIAN G., PEREZ I. C. D., ABUALSHOUR A., THABET A. K., GHANEM B.: Deepgcns: Making gcns go as deep as cnns. *IEEE Trans. Pat. Ana. & Mach. Int.* (2021). 2, 8

[LTLH19] LIU Z., TANG H., LIN Y., HAN S.: Point-Voxel CNN for efficient 3D deep learning. In *Proc. NeurIPS* (2019). 2

[LYYD19] LAN S., YU R., YU G., DAVIS L. S.: Modeling local geometric structure of 3D point clouds using Geo-CNN. In *Proc. CVPR* (2019). 2

[MBM*17] MONTI F., BOSCAINI D., MASCI J., RODOLA E., SVOBODA J., BRONSTEIN M. M.: Geometric deep learning on graphs and manifolds using mixture model cnns. In *Proc. CVPR* (2017). 2

[ML21] MAZUR K., LEMPITSKY V.: Cloud Transformers: A Universal Approach to Point Cloud Processing Tasks. In *Proc. ICCV* (2021). 2

[MS15] MATURANA D., SCHERER S.: Voxnet: A 3D convolutional neural network for real-time object recognition. In *Proc. IROS* (2015). 2

[MZC*19a] MO K., ZHU S., CHANG A. X., YI L., TRIPATHI S., GUIBAS L. J., SU H.: PartNet: A Large-scale Benchmark for Fine-grained and Hierarchical Part-level 3D Object Understanding. In *Proc. CVPR* (2019). 2

[MZC*19b] MO K., ZHU S., CHANG A. X., YI L., TRIPATHI S., GUIBAS L. J., SU H.: PartNet: A Large-Scale Benchmark for Fine-Grained and Hierarchical Part-Level 3D Object Understanding. In *Proc. CVPR* (2019). 5, 6, 7

[QSMG17] QI C. R., SU H., MO K., GUIBAS L.: PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *Proc. CVPR* (2017). 2

[QSN*16] QI C. R., SU H., NIESSNER M., DAI A., YAN M., GUIBAS L. J.: Volumetric and multi-view cnns for object classification on 3d data. In *Proc. CVPR* (2016). 2

[QYSG17] QI C. R., YI L., SU H., GUIBAS L. J.: PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. In *Proc. NeurIPS* (2017). 2, 8

[Rud16] RUDER S.: An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747* (2016). 6

[RUG17] RIEGLER G., ULUSOY A. O., GEIGER A.: OctNet: Learning Deep 3D Representations at High Resolutions. In *CVPR* (2017). 2

[RWS*18] RETHAGE D., WALD J., STURM J., NAVAB N., TOMBARI F.: Fully-convolutional point networks for large-scale point clouds. In *Proc. ECCV* (2018). 2

[SDMR19] SARLIN P.-E., DETONE D., MALISIEWICZ T., RABINOVICH A.: SuperGlue: Learning Feature Matching with Graph Neural Networks. *arXiv preprint arXiv:1911.11763* (2019). 2

[SFYT18] SHEN Y., FENG C., YANG Y., TIAN D.: Mining Point Cloud Local Structures by Kernel Correlation and Graph Pooling. In *Proc. CVPR* (2018). 2

[SGS19] SRIVASTAVA N., GOH H., SALAKHUTDINOV R.: Geometric Capsule Autoencoders for 3D Point Clouds. *arXiv preprint arXiv:1912.03310* (2019). 2

[SJS*18] SU H., JAMPANI V., SUN D., MAJI S., KALOGERAKIS E., YANG M.-H., KAUTZ J.: SPLATNet: Sparse Lattice Networks for Point Cloud Processing. In *Proc. CVPR* (2018). 2

[SMKLM15] SU H., MAJI S., KALOGERAKIS E., LEARNED-MILLER E.: Multi-view convolutional neural networks for 3d shape recognition. In *Proc. ICCV* (2015). 2

[SWL19] SHI S., WANG X., LI H.: PointRCNN: 3D Object Proposal Generation and Detection From Point Cloud. In *Proc. CVPR* (2019). 2

[TQD*19] THOMAS H., QI C. R., DESCHAUD J.-E., MARCOTEGUI B., GOULETTE F., GUIBAS L. J.: KPConv: Flexible and deformable convolution for point clouds. In *Proc. CVPR* (2019). 2

[VSP*17] VASWANI A., SHAZEER N., PARMAR N., USZKOREIT J., JONES L., GOMEZ A. N., KAISER L. U., POLOSUKHIN I.: Attention is All you Need. In *Proc. NeurIPS* (2017). 3, 4

[WGGH18] WANG X., GIRSHICK R., GUPTA A., HE K.: Non-local neural networks. In *Proc. CVPR* (2018). 2

[WHH*19] WANG L., HUANG Y., HOU Y., ZHANG S., SHAN J.: Graph Attention Convolution for Point Cloud Semantic Segmentation. In *Proc. CVPR* (2019). 2

[WLG*17] WANG P.-S., LIU Y., GUO Y.-X., SUN C.-Y., TONG X.: O-CNN: Octree-based Convolutional Neural Networks for 3D Shape Analysis. *ACM Trans. on Graphics 36*, 4 (2017). 2

[WQF19] WU W., QI Z., FUXIN L.: Pointconv: Deep convolutional networks on 3d point clouds. In *Proc. CVPR* (2019). 2

[WS19] WANG Y., SOLOMON J. M.: Deep Closest Point: Learning Representations for Point Cloud Registration. In *Proc. ICCV* (2019). 2

[WSC*21] WANG J., SUN K., CHENG T., JIANG B., DENG C., ZHAO Y., LIU D., MU Y., TAN M., WANG X., LIU W., XIAO B.: Deep High-Resolution Representation Learning for Visual Recognition. *IEEE Trans. Pat. Ana. & Mach. Int. 43*, 10 (2021). 5

[WSK*15] WU Z., SONG S., KHOSLA A., YU F., ZHANG L., TANG X., XIAO J.: 3D shapenets: A deep representation for volumetric shapes. In *Proc. CVPR* (2015). 2

[WSL*19] WANG Y., SUN Y., LIU Z., SARMA S. E., BRONSTEIN M. M., SOLOMON J. M.: Dynamic Graph CNN for Learning on Point Clouds. *ACM Trans. on Graphics 38*, 5 (2019). 2

[WSLT18] WANG P.-S., SUN C.-Y., LIU Y., TONG X.: Adaptive O-CNN: A Patch-based Deep Representation of 3D Shapes. *ACM Trans. on Graphics 37*, 6 (2018). 2

[WSM*18] WANG S., SUO S., MA W.-C., POKROVSKY A., URTASUN R.: Deep parametric continuous convolutional neural networks. In *Proc. CVPR* (2018). 2

[WSS18] WANG C., SAMARI B., SIDDIQI K.: Local Spectral Graph Convolution for Point Set Feature Learning. *arXiv preprint arXiv:1803.05827* (2018). 2

[WYZ*21] WANG P.-S., YANG Y.-Q., ZOU Q.-F., WU Z., LIU Y., TONG X.: Unsupervised 3D Learning for Shape Analysis via Multiresolution Instance Discrimination. In *Proc. AAAI* (2021). 2, 3, 4, 5, 6, 7, 8

[XFX*18] XU Y., FAN T., XU M., ZENG L., QIAO Y.: SpiderCNN: Deep Learning on Point Sets with Parameterized Convolutional Filters. *arXiv preprint arXiv:1803.11527* (2018). 2, 8

[XGG*20] XIE S., GU J., GUO D., QI C. R., GUIBAS L., LITANY O.: PointContrast: Unsupervised Pre-training for 3D Point Cloud Understanding. In *Proc. ECCV* (2020). 8

[XLCT18] XIE S., LIU S., CHEN Z., TU Z.: Attentional shapecontextnet for point cloud recognition. In *Proc. CVPR* (2018). 2

[XSyW*19] XU Q., SUN X., YING WU C., WANG P., NEUMANN U.: Grid-GCN for Fast and Scalable Point Cloud Learning. *arXiv preprint arXiv:1912.02984* (2019). 2

[XWL*21] XIANG P., WEN X., LIU Y.-S., CAO Y.-P., WAN P., ZHENG W., HAN Z.: SnowflakeNet: Point Cloud Completion by Snowflake Point Deconvolution With Skip-Transformer. In *Proc. ICCV* (2021). 2

[YHJY19] YUNXIAO S., HAOYU F., JING Z., YI F.: Pairwise Attention Encoding for Point Cloud Feature Learning. In *Proc. 3DV* (2019). 2

[YSGG17] YI L., SU H., GUO X., GUIBAS L. J.: SyncSpecCNN: Synchronized spectral cnn for 3d shape segmentation. In *Proc. CVPR* (2017). 2

[ZHW*19] ZHANG K., HAO M., WANG J., DE SILVA C. W., FU C.: Linked Dynamic Graph CNN: Learning on point cloud via linking hierarchical features. *arXiv preprint arXiv:1904.10014* (2019). 2

[ZJJ*21] ZHAO H., JIANG L., JIA J., TORR P. H., KOLTUN V.: Point transformer. In *Proc. ICCV* (2021). 2