IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 27, 2021, accepted September 11, 2021, date of publication September 29, 2021, date of current version October 7, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3116304

Point Transformer

NICO ENGEL, (Graduate Student Member, IEEE), VASILEIOS BELAGIANNIS, AND KLAUS DIETMAYER, (Member, IEEE)
Institute of Measurement, Control and Microtechnology, Ulm University, 89081 Ulm, Germany
Corresponding author: Nico Engel (nico.engel@uni-ulm.de)

ABSTRACT In this work, we present Point Transformer, a deep neural network that operates directly on unordered and unstructured point sets. We design Point Transformer to extract local and global features and relate both representations by introducing the local-global attention mechanism, which aims to capture spatial point relations and shape information. For that purpose, we propose SortNet, as part of the Point Transformer, which induces input permutation invariance by selecting points based on a learned score. The output of Point Transformer is a sorted and permutation invariant feature list that can directly be incorporated into common computer vision applications. We evaluate our approach on standard classification and part segmentation benchmarks to demonstrate competitive results compared to the prior work.

INDEX TERMS 3D point processing, artificial neural networks, computer vision, feedforward neural networks, transformer.

I. INTRODUCTION

Processing 3D point sets using deep neural networks has become very popular the past few years. The three-dimensional information has a wide range of applications in autonomous driving [1]–[6] and computer vision [7], [8]. However, training neural networks on point sets is not trivial. First, point sets are unordered, thus require the neural network to be permutation invariant. Second, the number of points in the set is usually dynamic and unstructured. Finally, the network needs to be robust against rotation and translation to operate in the metric space, and since the points describe objects, the network needs to capture the spatial relations between the points.

Standard neural architectures, such as convolutional neural networks (CNN), have shown promising results for structured data. For that reason, several point set processing approaches attempt to transform the points into regular representations such as voxel grids [9], [10] or rendered views of the point clouds [11], [12]. However, transforming the point sets leads to loss of shape information as geometric relations between points are removed. Furthermore, these methods suffer from high computational complexity due to the sparsity of the 3D points.

To address these limitations, there is another family of approaches that act directly on the point set. The main idea is to process each point individually with a multi-layer perceptron (MLP) and then fuse the representation to a vector of fixed size with a set pooling operation over a latent feature space [7], [13]. Set pooling is a symmetric function that is permutation invariant. Additionally, under certain conditions, set pooling acts as a universal set function approximator [14]. Nevertheless, Wagstaff *et al*. [15] argue that reducing the latent representation to a vector of fixed length can be impractical since the cardinality of the input set is usually not considered. Thus, the capacity of the vector may not be sufficient enough to capture the spatial relations of the point set which may reduce the overall performance. Therefore, the set pooling mechanism can become a bottleneck for point processing networks.

Our goal and motivation stems from removing the set pooling method and overcoming the aforementioned bottleneck, while still achieving a permutation invariant representation that models the point set relations in terms of object shape and geometric dependencies. Therefore, it is necessary to introduce a symmetric set function that replaces traditional set pooling operations. For that, we adapt the attention mechanism [16], which was originally introduced for natural language processing, that is used to weight and score sequences (words) based on learned importance. To our understanding, we face a similar problem in 3D point processing, given that we need to relate representations of the input points to capture and describe the object's shape. Additionally, attention itself does not depend on the input ordering, i.e. it is permutation-invariant, as it is comprised of matrix multiplication and summation only, which makes it well-suited

The associate editor coordinating the review of this manuscript and approving it for publication was Jinjia Zhou.

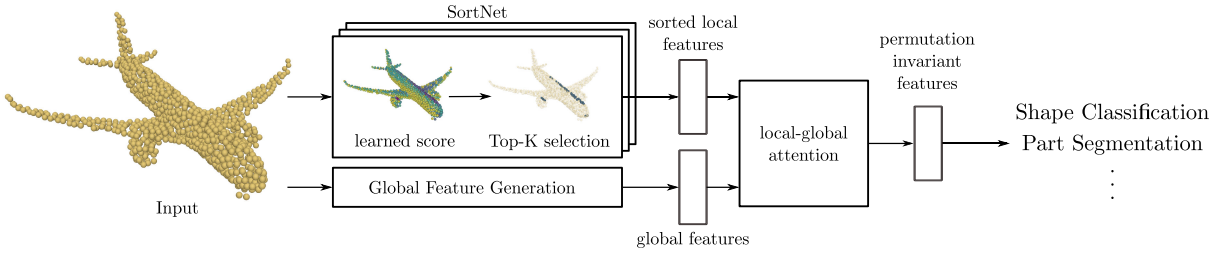

FIGURE 1. Overview of the Point Transformer pipeline. A point cloud serves as input to our network from which local and global features are extracted. We sort local features using SortNet, a module that focuses on important points based on a learned score. We then employ local-global attention to relate global and local features. We aim to capture geometric relations and shape information. The resulting feature representation is permutation invariant and can be used for common computer vision tasks.

for our problem. However, the output is still unordered, thus, directly processing the output of attention for standard computer vision tasks is not possible. Consequently, our goals can be outlined as follows:

- Avoid the bottleneck that can occur while employing set pooling operations [15].
- Present a novel permutation invariant network architecture that adapts the popular and prevalent attention mechanism for 3D point processing.
- Demonstrate superior performance compared to traditional set pooling methods to justify the use of attention and reinforce the claims made by Wagstaff *et al*.

To address these problems, we propose **SortNet**, a permutation invariant network module, that learns ordered subsets of the input with latent features of local geometric and spatial relations. For that, we learn important key points, which we call top-k selections, that replace the set pooling operation. Since current state-of-the-art methods have shown that aggregating local and global information increases the network's capabilities of capturing context information [7], [17], [18], we employ SortNet to generate local features of the point cloud. Moreover, global features of the entire point cloud are related to the sorted local features using **local-global attention**. Local-global attention attends both feature representations to capture the underlying shape. Since the local features are ordered, the output of local-global attention is ordered and permutation invariant; and thus it can be used for a variety of visual tasks such as shape classification and part segmentation. An overview of our network is outlined in Fig. 1. Since we aim to process 3D point sets using the ideas proposed by the Transformer network architecture [19], we took inspiration from [20], and name our network **Point Transformer**.

Overall, our contributions can be summarized as follows:

- We propose **Point Transformer**, a neural network that uses the multi-head attention mechanism and operates directly on unordered and unstructured point sets.
- We present **SortNet**, a key component of Point Transformer, that induces permutation invariance by selecting points based on a learned score.
- We evaluate Point Transformer on two standard benchmarks and show that it delivers competitive results.

II. RELATED WORK

Below, we discuss approaches that process 3D points and are related to our work.

A. POINT SET PROCESSING

Point clouds are irregular and unordered sets of points with a variable amount of elements, thus applying standard neural networks on 3D points is not possible. For that reason, previous approaches rely on transforming the point sets into an ordered representation, such as voxel grids. The metric space is discretized into small regions (voxels), which are labeled as occupied if a point lies inside the voxel. Then, 3D convolutional networks (CNN) can be easily applied to the voxel-based representation [9], [10], [21]. This preprocessing, however, reduces the resolution as multiple points are combined into a single voxel and thus damages important spatial relations of the metric space. Furthermore, voxelization increases the memory requirements and computational complexity due to the sparsity of the 3D points. To address these limitations, multiple extensions have been proposed that try to leverage the sparsity of 3D data [22]–[24], but still fail to process large amounts of input points.

1) VIEW-BASED METHODS

In contrast to building voxel grids, a lot of research has been conducted on rendering point clouds into 2D images, i.e. structured representation of the underlying 3D shape. Then, working with traditional CNNs is possible [12], [25]. Since shape information can be occluded by rendering point clouds from a specific viewpoint, multi-view approaches have been proposed that render multiple images from different angles [11], [12], [26], [27]. Even though images are rendered from different views, the model still fails to capture all geometric and spatial relations. To this day, multi-view approaches achieve impressive results on standard 3D benchmarks. However, the transformation from sparse 3D points into images increases computational complexity as well as required memory.

2) SHAPE-BASED METHODS

PointNet [13] is a pioneering network architecture that operates directly on 3D point sets, and it is invariant to input point permutations. Therefore, a transformation into a structured representation is no longer necessary. PointNet uses

a multi-layer perceptron (MLP) with shared weights that encodes spatial features to each input point separately. Then, a symmetric function, e.g. max pooling, is applied to the latent features to induce permutation invariance and create a global feature representation of the input. PointNet established the de facto standard for point processing that many state-of-the-art approaches still rely on [1], [28]. However, it is not able to encode and capture local information, since the max pooling operation induces permutation invariance, but also destroys local structures and relations of the points in metric space. To address this issue, Qi *et al*. proposed the improved PointNet++ [7] architecture, a hierarchical model that abstracts the input points with every layer to produce sets with fewer elements. First, centroids of local regions are sampled using hand-crafted algorithms, then local features are encoded to the centroids by exploring the local neighborhood. Thus, allowing the network to capture fine-grained patterns and improving the performance on current datasets. A general approach related to unordered sets was introduced by Zaheer *et al*. [14] demonstrating the capabilities of pooling operations to induce permutation invariance. Importantly, they prove that the set pooling method is a universal approximator for any set function. In general, problems arise with set pooling when the reduced feature vector lacks the capacity to capture important geometric relations. Our work addresses this limitation with a network topology that encodes the entire point cloud by relating local information with the global shape structure.

3) CONVOLUTIONS ON POINT CLOUDS

Classic convolutional neural networks require the input data to be ordered, such as images or voxel grids. Since points are unstructured, an active research area is the definition of convolution operations that can operate on irregular 3D point sets such as KPConv [29], SpiderCNN [30] or PointCNN [31]. These methods achieve state-of-the-art performance on a variety of tasks. However, due to the irregularities of the shape and point density, point convolutions are usually hard to design and the kernel needs to be adapted for different input data [32].

B. ATTENTION

Attention itself has its origin in natural language processing [16], [33]. Traditionally, encoder-decoder recurrent neural networks (RNN) were used for machine translation applications, where the last hidden state is used as the context vector for the decoder to sequentially produce the output. The problem is that dependencies between distant inputs are difficult to model using sequential processing. Bahdanau *et al*. [16] introduced the attention mechanism that takes the whole input sequence into account by taking the weighted sum of all hidden states and additionally, models the relative importance between words. Vaswani *et al*. [19] improved the attention mechanism by introducing multi-head attention and proposing an encoder-decoder structure that solely relies on attention instead of RNNs or convolutions. Therefore, they reduce the computational complexity. In this work, multi-head attention is the basis for Point Transformer.

1) ATTENTION WITH POINT CLOUD PROCESSING

Neural networks that rely on attention achieved impressive results in machine translation, and were adopted to function on point clouds by utilizing the points as sequences. Vinyals *et al*. [34] proposed a network that processes unordered sets using attention. They show that the network is able to sort numbers. However, they only focus on generic sets. In contrast, we present an approach that is applied to different point cloud related tasks for capturing shape and geometry information. Recently, Lee *et al*. [20] proposed Set Transformer, a method that is related to our approach. They adapt the original Transformer network to process unordered sets by using induced points, i.e. trainable parameters of the network, that are attended to the input. Set Transformer focuses on general sets as input. Furthermore, Lee *et al*. demonstrate that it is applicable to point sets. In our work, Point Transformer is specifically designed to process point clouds and leverage important characteristics of points in metric space such as shape and geometric relations.

Xie *et al*. [35] propose ShapeContextNet, where they hierarchically apply the shape context approach that acts as a convolutional building block. To overcome the difficulties of manually tuning the shape context parameters, Xie *et al*. employ self-attention to combine the selection and feature aggregation process into one trainable operation. However, similar to point cloud convolutions, shape context relies on a manual selection of the shape context kernels which is sensitive to the irregularities of point cloud data.

The Point2Sequence model [17] uses an attention-based sequence-to-sequence network. The approach first extracts local regions and produces local features using an LSTM-based attention module. Using a set pooling method, a global feature vector is generated following the ideas of [14] and [13]. However, it relies on a sequence-to-sequence architecture that tends to be more computational complex than multi-head attention [19]. Furthermore, in contrast to our method, Point2Sequence uses a max-pooling operation to make the network permutation invariant. Yang *et al*. [36] introduce a network architecture that replaces traditional subsampling methods like furthest point sampling (FPS) with an attention-based selection process using the gumbel-softmax function, which is similar to the proposed SortNet module.

Recently, Tao *et al.* [37] proposed a multi-head attentional point cloud processing network that uses a rotation invariant representation of point clouds as input. For that, they employ a multi-head attentional convolution layer (MACL) with attention coding. However, their work focuses on designing a rotation invariant network that relies on global max pooling operations, whereas Point Transformer together with SortNet leverages the strengths and advantages of the attention operation to select useful local point structures and relates them to the global shape to induce permutation invariance.

III. FUNDAMENTALS

Attention has been first proposed for natural language processing, where the goal is to focus on a subset of important words [16]. Here, we frame the problem in the context of point sets. We consider the unordered point set $\mathcal{P} = \{p_i \in \mathbb{R}^D, i = 1, \ldots, N\}$. Our goal is to map $\mathcal{P}$ to the output space $\mathbb{R}^O$ with the set function $f : \mathcal{P} \to \mathbb{R}^O$. Furthermore, we assume that f is invariant to input permutations. Since the input point set represents some object, e.g. from laser scans, the points are not independent of each other. We aim to make use of the attention mechanism to capture the relations between the points, as well as shape information for performing visual tasks such as object classification or segmentation. Next, we shortly present attention and introduce the Transformer architecture in the context of point sets.

A. ATTENTION

The idea of the attention mechanism is to set an importance-based focus on different parts of an input sequence. Consequently, relations between inputs are highlighted that can be used to capture context and higher-order dependencies. The attention function $\mathcal{A}(\cdot)$ describes a mapping of N queries $Q \in \mathbb{R}^{N \times d_k}$ and N_k key-value pairs $K \in \mathbb{R}^{N_k \times d_k}$, $V \in \mathbb{R}^{N_k \times d_v}$ to an output $\mathbb{R}^{N \times d_k}$ [19]. Using the pairwise dot product $QK^T \in \mathbb{R}^{N \times N_k}$, a score is calculated indicating which part of the input sequence to focus on

$$\text{score}(Q, K) = \sigma(QK^T), \tag{1}$$

where $\text{score}(\cdot) : \mathbb{R}^{N \times d_q}, \mathbb{R}^{N_k \times d_k} \to \mathbb{R}^{N \times N_k}$. Furthermore, we set the activation function $\sigma(\cdot) = \text{softmax}(\cdot)$ and scale QK^T by $1/\sqrt{d_k}$ to increase stability [19]. To capture the relations between the input points, the values V are weighted by the scores from Equation (1). Therefore, we have

$$\mathcal{A}(Q, K, V) = \text{score}(Q, K)V, \tag{2}$$

with $\mathcal{A}(Q, K, V) : \mathbb{R}^{N \times d_k}, \mathbb{R}^{N_k \times d_k}, \mathbb{R}^{N_k \times d_v} \to \mathbb{R}^{N \times d_k}$. It is apparent, that the attention function (2) is a weighted sum of V, where a value gets more weight if the dot product between the keys and values yields a higher score. If not specified otherwise, we set the model dimension to $d_k = d_q = d_m$.

B. TRANSFORMER

The Transformer network [19] is an extension of the attention mechanism from Equation (2) that consists of an encoder-decoder structure and introduces multi-head attention. In the following, we explain multi-head attention in detail, as our Point Transformer architecture relies on it.

Instead of employing a single attention function, multi-head attention first linearly projects the queries, keys and values Q, K, V h times to d_k, d_k and d_v dimensions, respectively, using separate feed-forward networks to learn relations from different subspaces. Then, attention is applied to each projection in parallel. The output is then concatenated and projected again using a feed-forward network. Thus, multi-head attention can be defined as follows:

$$\text{Multihead}(Q, K, V) = (\text{head}_1 \oplus \ldots \oplus \text{head}_h)W^O, \tag{3}$$

where $\text{head}_i = \mathcal{A}(QW_i^Q, KW_i^K, VW_i^V)$ with learnable parameters $W_i^Q \in \mathbb{R}^{d_m \times d_k}$, $W_i^K \in \mathbb{R}^{d_m \times d_k}$ and $W_i^V \in \mathbb{R}^{d_m \times d_v}$. The $\oplus$ operation denotes matrix concatenation and $W^O \in \mathbb{R}^{hd_v \times d_m}$ is a learnable parameter matrix [19]. To achieve similar computational complexity as traditional attention, the dimensions of each head d_k, d_v are reduced such that $d_k = d_v = d_m/h$. For the transformer architecture, Vaswani *et al.* [19] define encoder and decoder stacks of identical layers that are comprised of multi-head attention and a point-wise fully connected layer, each with a residual connection followed by layer normalization [38]. We call this layer multi-head attention and define it as follows:

$$\mathcal{A}^{\text{MH}}(X, Y) = \text{LayerNorm}(S + \text{rFF}(S)), \tag{4}$$

where $\mathcal{A}^{\text{MH}} : \mathbb{R}^{N \times d_m}, \mathbb{R}^{N_k \times d_m} \to \mathbb{R}^{N \times d_m}$. The sublayer S is defined as $S = \text{LayerNorm}(X + \text{Multihead}(X, Y, Y))$ and rFF is a row-wise feed-forward network that is applied to each input independently. In practice, multiple multi-head attention layers can be deployed in sequence to further capture higher-order dependencies. Note that the output of $\mathcal{A}^{\text{MH}}$ depends on the ordering of X, thus it is not permutation invariant. However, the values of the corresponding outputs for each input point are always the same regardless of the input order, since $\mathcal{A}^{\text{MH}}$ only consists of matrix multiplication and summation.

For the task of point processing, we take the unordered point set $\mathcal{P}$ and generate a latent feature representation p_i^{latent} with dimension d_m for every $p_i \in \mathcal{P}$ using a rFF and concatenate them to form $P = [p_1^{\text{latent}}, \ldots, p_N^{\text{latent}}] \in \mathbb{R}^{N \times d_m}$. Based on P we now define the **self multi-head attention** as:

$$\mathcal{A}^{\text{self}}(P) := \mathcal{A}^{\text{MH}}(P, P), \tag{5}$$

which performs multi-head attention between all elements of P, thus resulting in a matrix of same size as P. To attend elements of different sets, we additionally introduce a second matrix representation Q of another set $\mathcal{Q} = \{q_j \in \mathbb{R}^D, j = 1, \ldots, N_k\}$ that has been projected to latent feature dimension d_m, thus $Q \in \mathbb{R}^{N_k \times d_m}$. We can now define **cross multi-head attention** as:

$$\mathcal{A}^{\text{cross}}(P, Q) := \mathcal{A}^{\text{MH}}(P, Q), \tag{6}$$

that outputs a matrix of dimension $N \times d_m$ which order depends on the ordering of P. Since the output is not permutation invariant but follows the ordering of the input, Transformer and multi-head attention cannot be used directly for point data without further processing. To solve this problem, we introduce our novel Point Transformer architecture that handles unordered point sets.

IV. POINT TRANSFORMER

This section presents Point Transformer, a neural network that operates on point set data and it is based on the multi-head attention mechanism. The network is permutation invariant due to a new module that we name **SortNet**. Our goal is to explore shape information of the point set by relating local

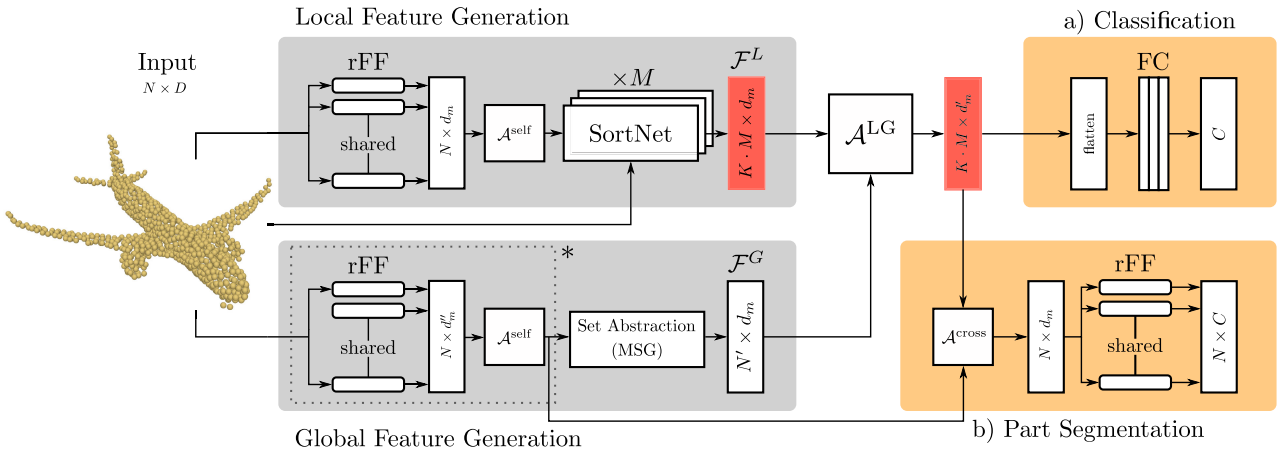

FIGURE 2. Overview of the Point Transformer architecture which consists of two branches to generate local and global features. SortNet produces an ordered set of local features that are attended against the global structure of the input point cloud. Depending on the task, classification or part segmentation heads are employed. Red Boxes denote sorted sets. * only for part segmentation.

and global features of the input. This is done using cross multi-head attention. To introduce our method, we first give an overview of the complete Point Transformer architecture, which is shown in Fig. 2. Our approach is divided into three parts:

1) **SortNet** that extracts ordered local feature sets from different subspaces.
2) **Global feature generation** of the whole point set.
3) **Local-Global attention**, which relates local and global features.

As introduced in Sec. III, we consider the point set $\mathcal{P} = \{p_i \in \mathbb{R}^D, i = 1, \ldots, N\}$ as input to our network. In most cases, the point dimension is given by $D = 3$ when *xyz* coordinates are considered. Moreover, it is possible to append additional point features, for example lidar intensity values ($D = 4$) or point normal vectors ($D = 6$). Point Transformer consists of two independent branches: a local feature generation module, i.e. SortNet, and a global feature extraction network. For the local feature branch, the input $\mathcal{P}$ is projected to latent space with dimension d_m using a row-wise feed-forward network. Then, we employ self multi-head attention on the latent features to relate the points to each other. Finally, SortNet outputs a sorted set of fixed length. This module is comparable to a kernel in convolutional neural networks, where the activation of a kernel depends on regions of the input space, i.e. the receptive field. SortNet works in a similar fashion: It focuses on points of interest according to the learnable score derived from the latent feature representation. For the extraction of global features, we employ set abstraction with multi-scale grouping introduced by [7]. After obtaining features from both branches, we employ our proposed local-global attention to combine and aggregate local and global features of the input point cloud. Since we use local-global attention such that the ordering of the output depends on the local features, the output of Point Transformer is permutation invariant and ordered as well and can directly be incorporated into computer vision applications such as shape classification and part segmentation.

A. SortNet

The local feature generation module, i.e. SortNet, is one of our key contributions. It produces local features from different subspaces that are permutation invariant by relying on a learnable score. We show the architecture in Fig. 3. SortNet receives the original point cloud $\mathcal{P} \in \mathbb{R}^{N \times D}$ and the projected latent feature representation$P = [p_1^{\text{latent}}, \ldots, p_N^{\text{latent}}] \in \mathbb{R}^{N \times d_m}$ from the row-wise feed forward network. We employ an additional self multi-head attention layer on the latent features to capture spatial and higher-order relations between each $p_i \in \mathcal{P}$.

Subsequently, a row-wise feed forward (rFF) network is used to reduce the feature dimension to one, thus creating a learnable scalar score $s_i \in \mathbb{R}$ for each input point p_i, which incorporates spatial relations due to the self multi-head attention layer. We now define the pair which assigns the corresponding score to every input point $\langle p_i, s_i \rangle_{i=1}^N$. Let $(\mathcal{Q}, \geq)$ be a totally ordered set. We select from the original input point list $K \leq N$ points with the highest score value and sort them accordingly such that:

$$\mathcal{Q} = \{q_j, j = 1, \ldots, K\}, \tag{7}$$

where $q_j = \langle p_i^j, s_i^j \rangle_{j=1}^K, p_i^j \in \mathcal{P}$ such that $s_i^1 \geq \ldots \geq s_i^K$. In other words, we employ the top-k operation to search for the K highest scores s_i and select the associated input points p_i. After selecting K points using the learnable score, we now capture localities by grouping all points from $\mathcal{P}$ that are within the euclidean distance r of each selected points, i.e. we perform a ball query search similar to [7]. The grouped points are then used to encode local features, denoted by $g^j \in \mathbb{R}^{d_m - 1 - D}, j = 1, \ldots, K$. We choose the feature dimension of the grouped points g^j such that the resulting dimension of the local feature vector corresponds to the model dimension d_m. The scores s_i^j, as well as the local features g^j from the grouping layer, are concatenated to the corresponding input points p_i^j to include the score calculation into our optimization problem and encode local characteristics to the selected point.

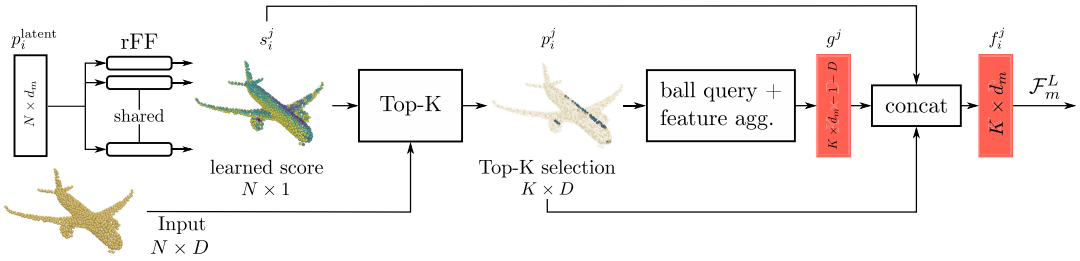

FIGURE 3. Overview of the SortNet. A score is learned from a latent feature representation to extract important points from the input. Local features are aggregated from neighboring points. SortNet outputs a permutation invariant and sorted feature set. Red boxes denote sorted sets.

Thus, we obtain our local feature vector

$$f_i^j = p_i^j \oplus s_i^j \oplus g^j, \quad f_i^j \in \mathbb{R}^{d_m}. \tag{8}$$

Consequently, the output of SortNet constitutes one local feature set

$$\mathcal{F}_m^L = \{f_i^j, j = 1, \ldots, K\}. \tag{9}$$

Since $\mathcal{Q}$ is an ordered set, it follows that $\mathcal{F}_m^L$ is ordered as well. To capture dependencies and local features from different subspaces, we employ M separate SortNets. Finally, the M feature sets are concatenated to obtain an ordered local feature set of fixed size

$$\mathcal{F}^L = \mathcal{F}_1^L \cup \ldots \cup \mathcal{F}_M^L, \quad \mathcal{F}^L \in \mathbb{R}^{K \cdot M \times d_m}. \tag{10}$$

B. GLOBAL FEATURE GENERATION

The second branch of Point Transformer is responsible for extracting global features from the input point cloud. To reduce the total number of points to save computational time and memory, we employ the set abstraction multi-scale grouping (MSG) layer introduced by Qi *et al.* [7]. We subsample the entire point cloud to $N' < N$ points using the furthest point sampling algorithm (FPS) and find neighboring points to aggregate features of dimension d_m resulting in a global representation of dimension $N' \times d_m$. Note that the global feature representation is still unordered since no sorting or set pooling operation was performed.

C. LOCAL-GLOBAL ATTENTION

The goal of Point Transformer is to relate local and global feature sets, $\mathcal{F}^L$ and $\mathcal{F}^G$ respectively, to capture shape and context information of the point cloud. After obtaining both feature lists, we employ self multi-head attention $\mathcal{A}^{\text{self}}$ on the local features $\mathcal{F}^L$ as well as the global features $\mathcal{F}^G$. Then, cross multi-head attention layer $\mathcal{A}^{\text{cross}}$ from Equation (6) is applied such that every global feature is scored against every local feature, thus relating local context with the underlying shape. We call this operation local-global attention $\mathcal{A}^{\text{LG}}$ (see Fig. 2) and define it as follows:

$$\mathcal{A}^{\text{LG}} := \mathcal{A}^{\text{cross}}(\mathcal{A}^{\text{self}}(F^L), \mathcal{A}^{\text{self}}(F^G)), \tag{11}$$

where F^L and F^G are the matrix representations of $\mathcal{F}^L$ and $\mathcal{F}^G$, respectively. The last row-wise feed forward layer in the multi-head attention mechanism of $\mathcal{A}^{\text{LG}}$ reduces the feature dimension to $d'_m < d_m$ in order to decrease computational complexity, thus we have $\mathcal{A}^{\text{LG}} : \mathbb{R}^{K \cdot M \times d_m}, \mathbb{R}^{N' \times d_m} \to \mathbb{R}^{K \cdot M \times d'_m}$. In other words, we take every local feature from SortNet and score the global features against it. At this point, it is important to note that we relate the local features, i.e. a subset of the input $\mathcal{F}^L \subseteq \mathcal{P}$, with the global structure. Thus, we avoid reducing the shape representation using set pooling; instead, the output of local-global attention includes information of the entire point cloud, i.e. the underlying shape, as well as local characteristics. As with multi-head attention, for local-global attention, we employ multiple cross and self multi-head attention layers in sequence to learn higher-order dependencies [19]. Since the ordering of the local features $\mathcal{F}^L$ defines the order of the output of local-global attention, we obtain a permutation invariant latent representation of fixed size of the aggregated features, that can directly be incorporated into computer vision tasks.

D. COMPLETE MODEL

To recap, Point Transformer functions as follows: Our architecture is comprised of two independent branches, SortNet for the extraction of local features and a global feature generation module. SortNet constitutes a novel architecture that selects a number of input points based on a learned score from latent features, resulting in $M \cdot K$ ordered feature vectors with dimension d_{m}. In the global feature branch, we employ multi-scale grouping to reduce the total number of points to N' while aggregating spatial information. Then, local-global attention is used to relate both spatial signatures, producing a permutation invariant and ordered representation of length $K \cdot M$ with reduced dimension d'_m (see Fig. 2), which can be used for different tasks such as shape classification or part segmentation. Additionally, we demonstrate the processing chain of our model as a flowchart in Fig. 4.

1) SHAPE CLASSIFICATION

Assigns the point cloud to one of C object classes. For this, we flatten the sorted output of local-global attention to a vector of fixed size $\mathbb{R}^{M \cdot K \cdot d'_m}$ and reduce the dimensions using a row-wise feed-forward network to $\mathbb{R}^C$. Thus, each output represents one class. Using a final softmax layer, class probabilities are produced. The shape classification head is shown in Fig. 2 a).

TABLE 1. Here, we compare Point Transformer to related approaches that use either set pooling or attention. We evaluate on popular benchmarks for object classification (ModelNet) and part segmentation (ShapeNet).

Method	ModelNet	ShapeNet
PointNet [13]	89.2	83.7
PointNet++ [7]	91.9	85.1
ShapeContextNet [35]	89.8	84.6
Deep Sets [14]	90.3	-
Point2Sequence [17]	92.6	85.2
Set Transformer [20]	90.4	-
PAT [36]	91.7	-
Tao et. al [37]	87.5	75.2
Point Transformer	92.8	85.9
KPConv [29]	92.9	86.2
PointCNN [31]	92.2	86.1
SpiderCNN [30]	90.5	85.3

2) PART SEGMENTATION

Assigns a label to each point of the input set. State-of-the-art methods [7], [17] upsample a global feature vector obtained from a set pooling operation using interpolation. We, however, employ an additional cross multi-head attention layer to attend the output of $\mathcal{A}^{\text{LG}}$, i.e. the aggregated shape and context information, to each point of the input set $\mathcal{P}$. It is important to note that we project the points in the global feature generation branch to d''_m dimensions and apply self multi-head attention. The features are additionally used for the set abstraction layer. Later, we attend the projected features with the output of Point Transformer. Thus, we can relate each point to the entire point cloud. The result is a matrix of dimension $\mathbb{R}^{N\times d'_m}$. Then, a row-wise feed-forward layer reduces the dimension of each point to the C possible classes $\mathbb{R}^{N\times C}$. Again, using a final softmax layer, per-point class probabilities are produced as shown in Fig. 2 b).

V. EXPERIMENTS

In this section, we perform two standard evaluations on Point Transformer. We compare our results with approaches that operate directly on 3D point sets [7], [13], [14], attention-based approaches [17], [20], [35] and methods that use point cloud convolutions [29]–[31], [39]. Moreover, we provide a thoughtful analysis and visualizations of the components of our approach. We implement our network in Pytorch [40] where we rely on the RAdam optimizer [41] for all experiments. The weights of each layer are initialized using the popular Kaiming normal initialization method [42]. Our implementation will be made publicly available.

A. POINT CLOUD CLASSIFICATION

We evaluate Point Transformer on the ModelNet40 dataset [10] and use the modified version by Qi *et al.* [7] that provides 10.000 points sampled from the mesh of the CAD model, as well as the normal vectors for each point. The dataset consists of 40 categories and it is composed of 9843 training samples and 2468 test samples. During the training for classification, we augment the input by randomly scaling the shape in the range of [0.8, 1.25] and randomly translating in the range of $[-0.1, 0.1]$. Additionally, we apply random dropout of the input points as proposed in [7], [13]. For the experiments, we set $N = 1024$, $D = 6$ (xyz and normals), $d_m = 512$, $d'_m = 64$, $M = 4$ and $K = 64$. The results of the shape classification are shown in Table 1. Point Transformer outperforms attention-based methods (top part of Table 1) and achieves on par accuracy when compared to state-of-the art methods (bottom part of Table 1) with a classification accuracy of 92.8%.

TABLE 2. Hyperparameters of Point Transformer for the classification and the part segmentation task.

Parameter	Explanation	Classification	Part Segmentation
General Hyperparameters			
B	Batch size	11	8
N	Number of input points	1024	1024
D	Input dimension	6	6
l_r	Learning rate	0.001	0.005
w_d	Weight decay	1×10^{-6}	0.0001
d_m	Latent feature dimension	512	512
-	Weight initializer	Kaiming Normal [42]	
Local Feature Generation			
-	rFF feature dimension	$(64, 128, 512)$	$(64, 128, 512)$
-	Dropout	0.4	0.3
n_{head}	Number of local attention heads	8	8
n_{layers}	Number of local attention layers	1	1
SortNet			
M	Number of SortNets	4	10
K	Number of top-k selections	64	16
-	rFF feature dimension	$(128, 256, 512)$	$(64, 128, 512)$
-	Dropout	0.4	0.3
Global Feature Generation			
N'	Reduced point set	128	64
-	Segmentation rFF	-	$(64, 128, 256)$
d''_m	Segmentation feature dimension	-	256
n_{head}	Number of local attention heads	8	8
n_{layers}	Number of local attention layers	1	1
-	Dropout	0.4	0.3
Local-Global Attention			
d'_m	Reduced feature dimension	64	256
n_{head}	Number of local attention heads	8	8
n_{layers}	Number of local attention layers	4	4
Classification Head			
C	Number of classes	40	
-	Fully connected dimension	$(4096, 1024, 512, 128, 40)$	
-	Dropout	0.4	
Segmentation Head			
C	Number of classes	50	
-	Output rFF	$(256, 128, 50)$	
n_{head}	Number of local attention heads	8	8
n_{layers}	Number of local attention layers	1	1

TABLE 3. Model complexity study. Here, we compare the network size and the inference time against related approaches.

Method	# of params	Network size	Inference time
Point Transformer	13.5 M	51 MB	110 ms
SortNet	10 k	0.04 MB	1.25 ms
PAT [36]	-	5.8 MB	88 ms
KPConv [29]	15 M	-	210 ms
PointNet++ [7]	2.1 M	24 MB	160 ms
Point2Seq [17]	2 M	-	-

B. POINT CLOUD PART SEGMENTATION

Here, we evaluate Point Transformer on the challenging task of point cloud part segmentation on the ShapeNet dataset [43], which contains 13.998 train samples and 2874 test samples. The dataset is composed of objects from 16 categories with a total of 50 part labels. The goal is to predict the class category of every point. To address this task, the network has to learn a deep understanding of the underlying shape. For the part segmentation, we set $M = 10$ and $K = 16$.

TABLE 4. Hyperparameter study results on ModelNet40 for different combinations of the hyperparameters M (number of SortNets) and K (Top-K selections).

M	4	4	4	4	8	8	8	8	8	16	16	16	16	32	32	32
K	16	32	64	96	8	16	32	64	96	8	16	32	64	8	16	32
Accuracy	91.7	92.3	**92.8**	91.7	90.2	90.5	91.9	92.4	92.0	91.2	91.6	92.0	91.7	90.8	91.3	91.1

Again, we use *xyz* coordinates with normal vectors ($D = 6$) and $N = 1024$ input points. For this experiment, we follow the setup of [13] where a one-hot encoding of the category is concatenated to the input points as an additional feature. We report the mean IoU (Intersection-over-Union) in Table 1. Finally, we visualize exemplary results of the part segmentation task in Fig. 5.

C. NETWORK COMPLEXITY

We examine the network complexity of Point Transformer and perform a comparison to related approaches. The results of this experiment are shown in Table 3. We performed all experiments on a Nvidia GeForce 1080Ti. Point Transformer has about 13.5 million learnable parameters (51 MB), which is less when compared to KPConv (15 million learnable parameters). However, our model is about 6 times bigger than PointNet++ and Point2Seq. This is mainly due to the fact that the Transformer model itself has a lot of learnable parameters. For example, one SortNet only has about 10.000 learnable parameters which shows that SortNet can be incorporated into any existing network architecture without much space requirements and computational overhead, as it only adds about 1.2 ms of inference time. In many cases, the forward pass of multiple SortNets can additionally be performed in parallel. Even though, Point Transformer has more learnable parameters than, e.g, PointNet++, it still has a faster inference time because multi-head attention blocks are highly optimized and computation is also performed in parallel by employing multiple attention heads. For the computational complexity of the network, an upper bound can be estimated from the most expensive operation, which in our case is the multi-head attention mechanism. The complexity is given by $\mathcal{O}(N^2 \cdot d_m)$, thus it scales quadratic with respect to the total number of input points.

D. HYPERPARAMETER STUDY

Here, we analyze the effects of different numbers of SortNets in our Point Transformer architecture as well as the amount of Top-K selections on the ModelNet40 dataset [10]. The results are shown in Table 4. Furthermore, we present the hyperparameters that were used for the reported results for the classification and the part segmentation task in Table 2. The parameters follow the notation introduced in Fig. 2 and Fig. 3. The values were found by performing a hyperparameter grid search experiment for the classification and the part segmentation, similar to Table 4. We report the set of parameters that achieved the best overall performance. Note, that for the rFF, each value in the parenthesis denotes one layer, where the value represents the feature dimension for that layer.

TABLE 5. Results of network design analysis. We evaluate different SortNet architectures to highlight that the learnable score increases the networks performance. Additionally, we compare different sampling methods for the global feature generation branch.

a) Ablation Study SortNet	Accuracy (%)
SortNet with learnable score	83.4
SortNet with FPS	74.8
SortNet with random points	60.1
b) Ablation Study Global Feature Generation	**Accuracy (%)**
No sampling	91.9
FPS ($N' = 128$)	92.3
Set Abstraction (MSG) ($N' = 128$)	92.8

E. POINT TRANSFORMER DESIGN ANALYSIS

We conduct an ablation study to show the influence of each Point Transformer module. Afterward, we qualitatively examine our classification results by visualizing the learned point set regions that contribute to the classification output.

1) ABLATION STUDY OF SortNet

We first evaluate Point Transformer using only the SortNet module from Fig. 3 with the classification head from Fig. 2 a). Our aim is to show that the learned scores are based on the importance of points for the classification task. In addition, we want to verify that SortNet selects points that help to understand the underlying shape. Since we cannot explicitly define which are the most important points, we rely on the accuracy score. In detail, we train SortNet based on three different experiments and deliberately set $M = 10$ and $K = 12$, selecting only a subset of the entire point cloud ($M \cdot K = 120$, $N = 1024$). In the first experiment, we train SortNet as it is implemented in the Point Transformer pipeline. In the second experiment, we replace the Top-K selection process with the furthest point sampling. Finally, we randomly select K points from the input set instead of the learned Top-K selection. It is important to note, that the last two experiments remove the permutation invariance property. However, we want to show that SortNet performs better than a random selection of points and handcrafted sampling methods. Thus, we rely on random sampling and FPS as baselines. The results are shown in Table 5 a). With randomly sampled points, SortNet achieves 60.1% classification accuracy. When we apply the FPS to cover most of the underlying shape, the accuracy increases to 74.8%, indicating spatial information preservation. Finally, when we use learned Top-K selection, we achieve the highest classification accuracy of 83.4%. This empirically shows that SortNet learns to focus on important shape regions.

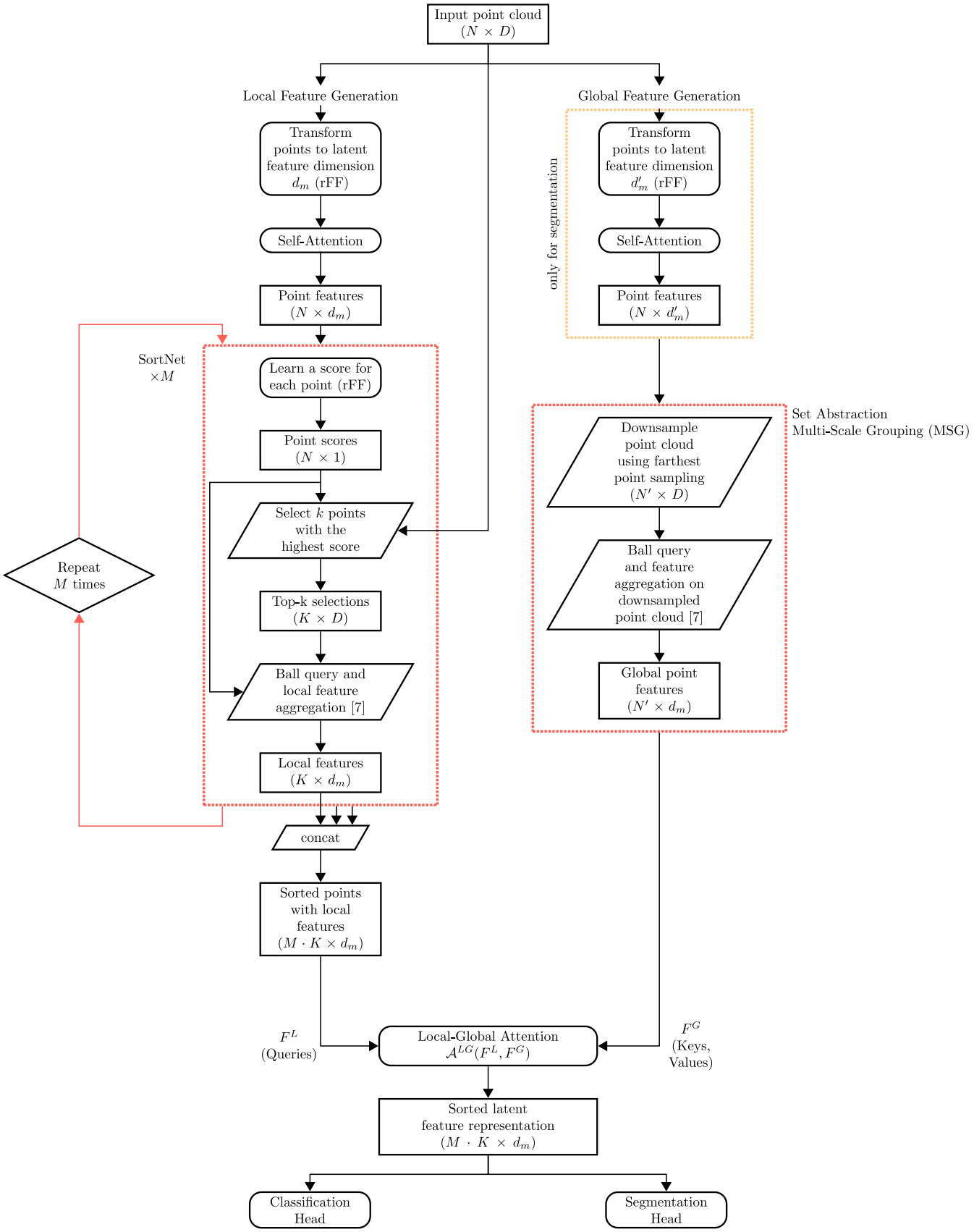

FIGURE 4. Overview of the processing chain of Point Transformer. Data is shown as rectangles with the respective dimensions. Networks modules, for example row-wise feed forward networks (rFF), are denoted by rectangles with rounded corners and additional process steps are shown as parallelograms. Here, it is important to note that individual rFF's with separate weights are deployed in each of the *M* SortNet modules.

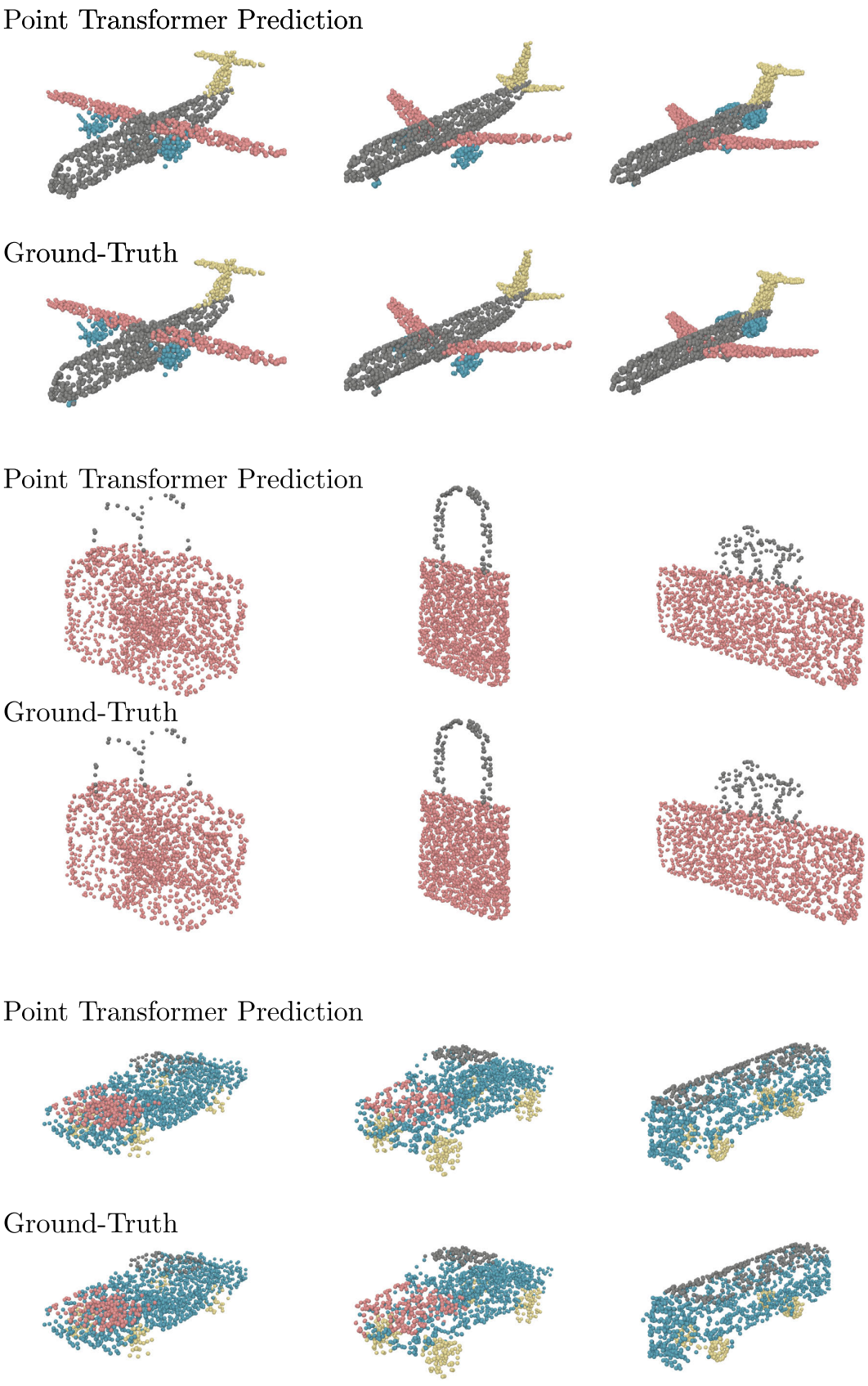

FIGURE 5. Additional results of the part segmentation task for different object categories. We show the prediction of Point Transformer (top) in comparison with the ground-truth (bottom).

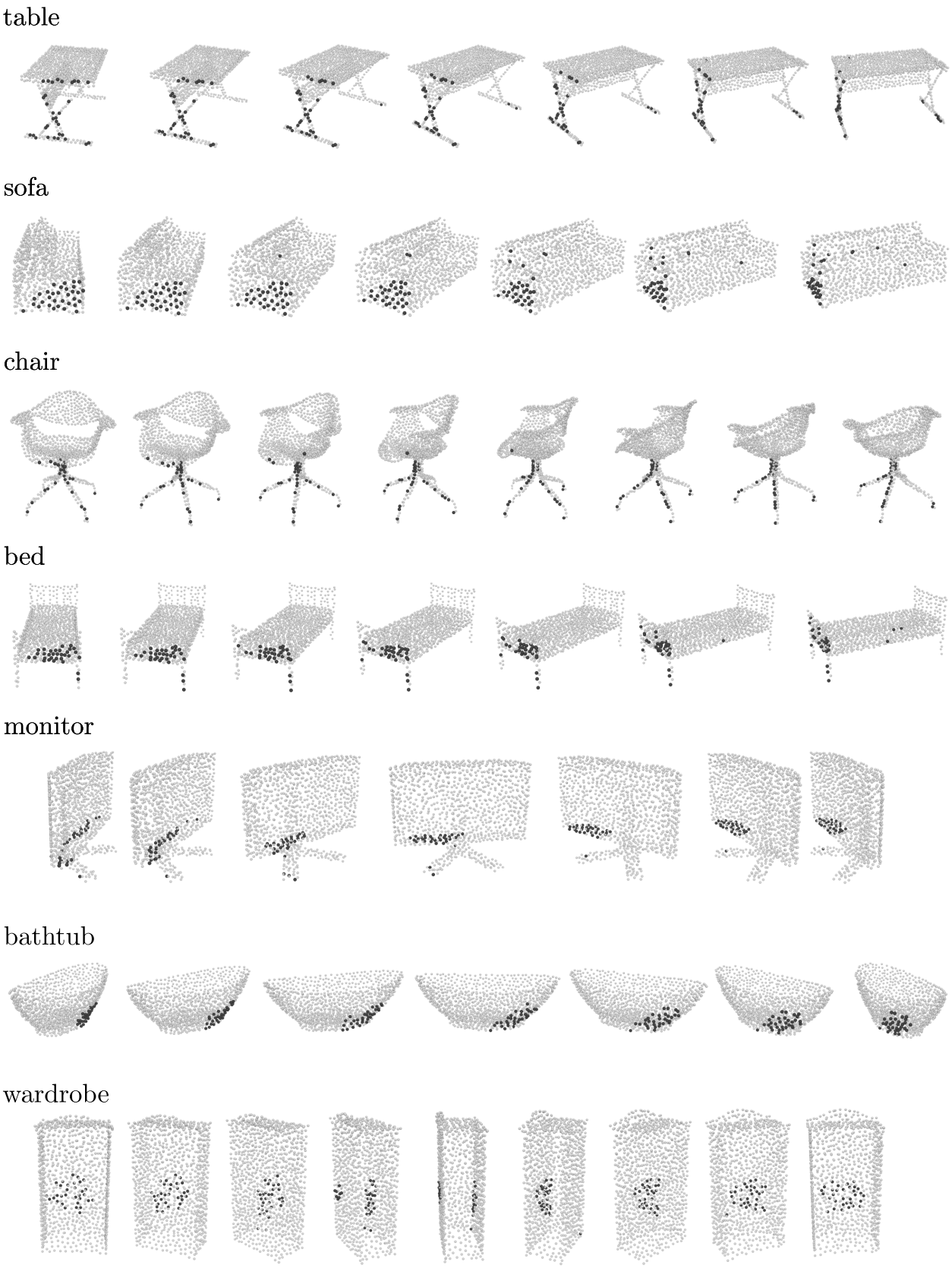

FIGURE 6. Influence of input point rotations on the top-K selection process. • Top-K selection (dark points), · input points (light points). When the input point cloud is rotated, SortNet still focuses on similar local regions of the underlying shape.

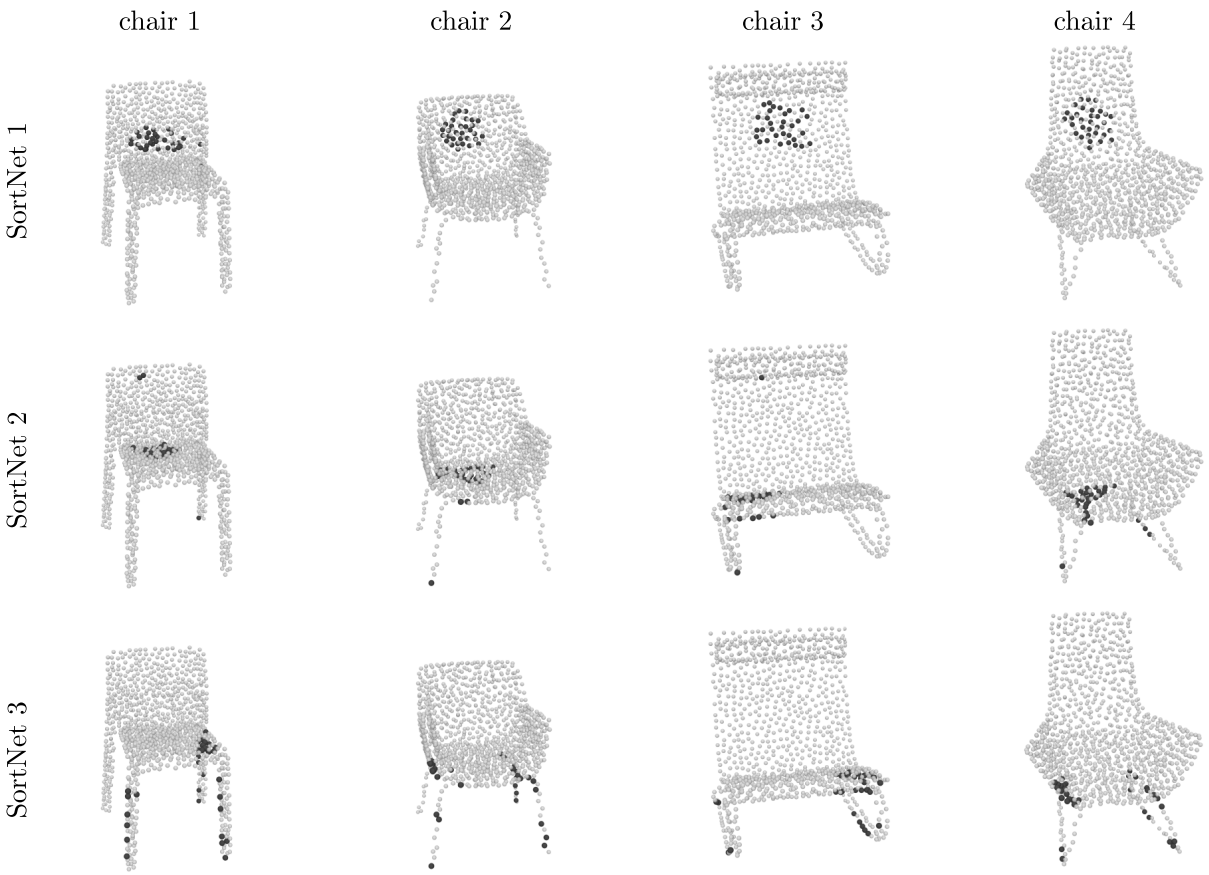

FIGURE 7. Top-K selections for different chair models. • Top-K selection (dark points), • input points (light points). SortNet selects points from similar local regions when applied to objects of the same category, suggesting that it is aware of the underlying shape.

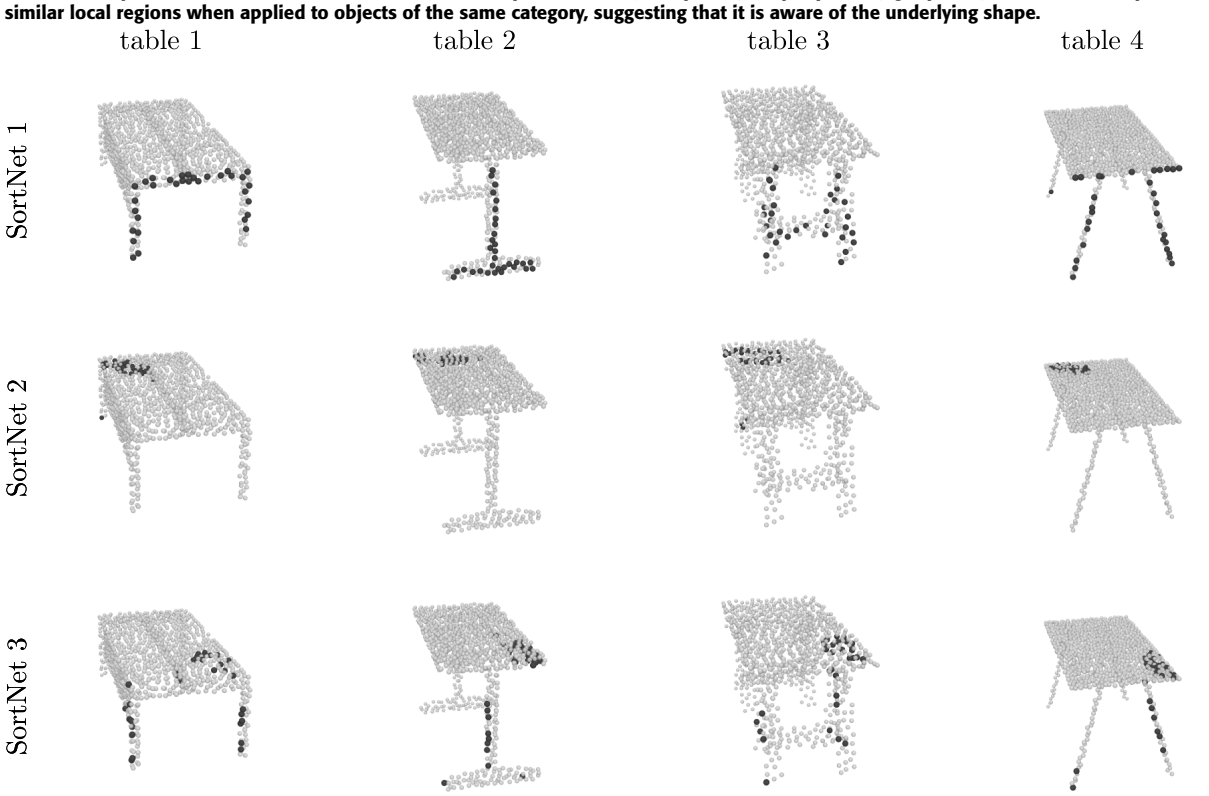

FIGURE 8. Top-K selections for different table models. • Top-K selection (dark points), • input points (light points). SortNet selects points from similar local regions when applied to objects of the same category, suggesting that it is aware of the underlying shape.

FIGURE 9. Here all selected points from the local feature generation branch (right) are shown in comparison with the complete input point cloud (left). The selected points of each SortNet are shown in the same color. It is clear that every SortNet focuses on different local regions of the object. When the selected points are visualized together, the input point cloud is still recognizable, suggesting that in combination, all SortNets try to retain as much as possible of the underlying shape.

2) ABLATION STUDY GLOBAL FEATURE GENERATION

In this ablation study, we compare different sampling methods for the extraction of global features. We rely on the complete Point Transformer pipeline as shown in Fig. 2 and replace the set abstraction (MSG) with different sampling approaches. Again, we evaluate the accuracy of the classification task. The results are presented in Table 5 b). In the first experiment, we use the complete input point cloud. Then, we sample $N' = 128$ points using the furthest point sampling, which slightly improves our result by 0.4%. When we additionally aggregate features from local regions around the sampled points, i.e. set abstraction with multiscale grouping (MSG) [7], the accuracy can be further increased to 92.8%. This indicates that scoring the local features against every input point makes it harder to find important relations. Additionally, by uniformly selecting fewer points and aggregating local features the network can concentrate on meaningful parts of the underlying shape.

3) ROTATION ROBUSTNESS OF SortNet

In this section we evaluate the robustness of SortNet against rotations of the input cloud. For this, we first evaluate Point Transformer on the ModelNet40 test set and randomly rotate the input point cloud. Even though we did not train the network with rotations, we still achieve a classification accuracy of 92.3% compared to 92.8% without rotations. We applied the same input point rotation to PointNet++ and classification accuracy dropped from 91.9% to 88.6%. To qualitatively support this claim, we visualize the learned Top-K selections of one SortNet for different rotations in Fig. 6, which shows that SortNet still focuses on the similar local regions even when the input point cloud is rotated.

4) VISUALIZATIONS OF LEARNED LOCAL REGIONS

Here, we show that SortNet focuses on local regions similar to the receptive field of a CNN. For this, we visualize the learned Top-K selections of multiple trained SortNet modules on different models of the same object class in Fig. 7 and Fig. 8. It is apparent, that each SortNet tries to select similar regions even when the shape of the model is slightly different. This, together with the results from the rotational robustness, suggests that SortNet is aware of the underlying shape.

5) ALL TOP-K SELECTIONS

As an additional evaluation, we show all selected points of $M = 8$ SortNet modules in Fig. 9 for the classification task. We visualize points that were selected from the same SortNet with the same color. It is apparent, that different SortNet modules focus on different parts of the object and in combination, still retain as much as possible of the underlying shape.

VI. CONCLUSION AND FUTURE WORK

In this work, we proposed Point Transformer, a permutation invariant neural network that relies on the multi-head attention mechanism and operates on irregular point clouds. The core of Point Transformer is a novel module that receives a latent feature representation of the input point cloud and selects points based on a learned score. We relate local features to the global structure of the point cloud, thus exploiting context and inducing shape-awareness. The output of Point Transformer is a sorted and permutation invariant feature list that is used for shape classification and part segmentation. Finally, we show that our point selection mechanism is based on importance for the specified task. As future work, we want to focus on improving the efficiency of the Transformer architecture by implementing recent advances for self-attention, such as [44], [45].

REFERENCES

[1] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "PointPillars: Fast encoders for object detection from point clouds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 12697–12705.

[2] N. Engel, S. Hoermann, P. Henzler, and K. Dietmayer, "Deep object tracking on dynamic occupancy grid maps using RNNs," in *Proc. 21st Int. Conf. Intell. Transp. Syst. (ITSC)*, Nov. 2018, pp. 3852–3858.

[3] N. Engel, S. Hoermann, M. Horn, V. Belagiannis, and K. Dietmayer, "DeepLocalization: Landmark-based self-localization with deep neural networks," in *Proc. IEEE Intell. Transp. Syst. Conf. (ITSC)*, Oct. 2019, pp. 926–933.

[4] K. Fischer, M. Simon, F. Oelsner, S. Milz, H.-M. Gross, and P. Maeder, "StickyPillars: Robust and efficient feature matching on point clouds using graph neural networks," 2020, *arXiv:2002.03983*. [Online]. Available: http://arxiv.org/abs/2002.03983

[5] M. Horn, N. Engel, V. Belagiannis, M. Buchholz, and K. Dietmayer, "DeepCLR: Correspondence-less architecture for deep end-to-end point cloud registration," in *Proc. IEEE 23rd Int. Conf. Intell. Transp. Syst. (ITSC)*, Sep. 2020, pp. 1–7.

[6] J. Wiederer, A. Bouazizi, U. Kressel, and V. Belagiannis, "Traffic control gesture recognition for autonomous vehicles," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2020, pp. 10676–10683.

[7] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep hierarchical feature learning on point sets in a metric space," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5099–5108.

[8] C. R. Qi, O. Litany, K. He, and L. Guibas, "Deep Hough voting for 3D object detection in point clouds," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 9277–9286.

[9] D. Maturana and S. Scherer, "VoxNet: A 3D convolutional neural network for real-time object recognition," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2015, pp. 922–928.

[10] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3D ShapeNets: A deep representation for volumetric shapes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1912–1920.

[11] C. R. Qi, H. Su, M. Nießner, A. Dai, M. Yan, and L. J. Guibas, "Volumetric and multi-view CNNs for object classification on 3D data," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 5648–5656.

[12] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller, "Multi-view convolutional neural networks for 3D shape recognition," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 945–953.

[13] R. Q. Charles, H. Su, M. Kaichun, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 652–660.

[14] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola, "Deep sets," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 3391–3401.

[15] E. Wagstaff, F. Fuchs, M. Engelcke, I. Posner, and A. M. Osborne, "On the limitations of representing functions on sets," in *Proc. 36th Int. Conf. Mach. Learn. (ICML)*, vol. 97, 2019, pp. 6487–6494.

[16] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *Proc. 3rd Int. Conf. Learn. Represent. (ICLR)*, 2015.

[17] X. Liu, Z. Han, Y.-S. Liu, and M. Zwicker, "Point2Sequence: Learning the shape representation of 3D point clouds with an attention-based sequence to sequence network," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, 2019, pp. 8778–8785.
[18] J. Li, B. M. Chen, and G. H. Lee, "SO-Net: Self-organizing network for point cloud analysis," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 9397–9406.
[19] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5998–6008.
[20] J. Lee, Y. Lee, J. Kim, A. Kosiorek, S. Choi, and Y. W. Teh, "Set transformer: A framework for attention-based permutation-invariant neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 3744–3753.
[21] Y. Zhou and O. Tuzel, "VoxelNet: End-to-end learning for point cloud based 3D object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4490–4499.
[22] D. Z. Wang and I. Posner, "Voting for voting in online point cloud object detection," in *Proc. Robot., Sci. Syst.*, Rome, Italy, Jul. 2015. [Online]. Available: http://www.roboticsproceedings.org/rss11/p35.html, doi: 10.15607/RSS.2015.XI.035.
[23] Y. Li, S. Pirk, H. Su, C. R. Qi, and L. J. Guibas, "FPNN: Field probing neural networks for 3D data," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 307–315.
[24] G. Riegler, A. O. Ulusoy, and A. Geiger, "OctNet: Learning deep 3D representations at high resolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 3577–3586.
[25] B. Shi, S. Bai, Z. Zhou, and X. Bai, "DeepPano: Deep panoramic representation for 3-D shape recognition," *IEEE Signal Process. Lett.*, vol. 22, no. 12, pp. 2339–2343, Dec. 2015.
[26] A. Kanezaki, Y. Matsushita, and Y. Nishida, "RotationNet: Joint object categorization and pose estimation using multiviews from unsupervised viewpoints," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 5010–5019.
[27] J.-C. Su, M. Gadelha, R. Wang, and S. Maji, "A deeper look at 3D shape classifiers," in *Proc. Eur. Conf. Comput. Vis.* Springer, 2018, pp. 645–661.
[28] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas, "Frustum PointNets for 3D object detection from RGB-D data," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 918–927.
[29] H. Thomas, C. R. Qi, J.-E. Deschaud, B. Marcotegui, F. Goulette, and L. Guibas, "KPConv: Flexible and deformable convolution for point clouds," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 6411–6420.
[30] Y. Xu, T. Fan, M. Xu, L. Zeng, and Y. Qiao, "SpiderCNN: Deep learning on point sets with parameterized convolutional filters," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 87–102.
[31] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen, "PointCNN: Convolution on *X*-transformed points," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 820–830.
[32] Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu, and M. Bennamoun, "Deep learning for 3D point clouds: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, early access, Jun. 29, 2020, doi: 10.1109/TPAMI.2020.3005434.
[33] M.-T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," 2015, *arXiv:1508.04025*. [Online]. Available: http://arxiv.org/abs/1508.04025
[34] O. Vinyals, S. Bengio, and M. Kudlur, "Order matters: Sequence to sequence for sets," in *Proc. 4th Int. Conf. Learn. Represent. (ICLR)*, 2016.
[35] S. Xie, S. Liu, Z. Chen, and Z. Tu, "Attentional ShapeContextNet for point cloud recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4606–4615.
[36] J. Yang, Q. Zhang, B. Ni, L. Li, J. Liu, M. Zhou, and Q. Tian, "Modeling point clouds with self-attention and Gumbel subset sampling," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 3323–3332.
[37] Z. Tao, Y. Zhu, T. Wei, and S. Lin, "Multi-head attentional point cloud classification and segmentation using strictly rotation-invariant representations," *IEEE Access*, vol. 9, pp. 71133–71144, 2021.
[38] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," 2016, *arXiv:1607.06450*. [Online]. Available: http://arxiv.org/abs/1607.06450
[39] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph CNN for learning on point clouds," *ACM Trans. Graph.*, vol. 38, no. 5, pp. 1–12, 2019.
[40] A. Paszke *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 8026–8037. [Online]. Available: https://github.com/pytorch/pytorch/blob/master/CITATION
[41] L. Liu, H. Jiang, P. He, W. Chen, X. Liu, J. Gao, and J. Han, "On the variance of the adaptive learning rate and beyond," 2019, *arXiv:1908.03265*. [Online]. Available: http://arxiv.org/abs/1908.03265
[42] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1026–1034.
[43] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu, "ShapeNet: An information-rich 3D model repository," 2015, *arXiv:1512.03012*. [Online]. Available: http://arxiv.org/abs/1512.03012
[44] S. Wang, B. Z. Li, M. Khabsa, H. Fang, and H. Ma, "Linformer: Self-attention with linear complexity," 2020, *arXiv:2006.04768*. [Online]. Available: http://arxiv.org/abs/2006.04768
[45] Y. Xiong, Z. Zeng, R. Chakraborty, M. Tan, G. Fung, Y. Li, and V. Singh, "Nyströmformer: A Nyström-based algorithm for approximating self-attention," 2021, *arXiv:2102.03902*. [Online]. Available: http://arxiv.org/abs/2102.03902

NICO ENGEL (Graduate Student Member, IEEE) received the B.Sc. and M.Sc. degrees in electrical engineering and business administration from Christian-Albrecht University of Kiel, Kiel, Germany, in 2015 and 2017, respectively. He is currently pursuing the Doctoral degree (Ph.D.) with Ulm University, Ulm, Germany. Currently, he is a Research Assistant with the Institute of Measurement, Control and Microtechnology, Ulm University. His research interests include vehicle self-localization methods for autonomous driving and current deep learning advances in the field of 3D computer vision and point cloud processing.

VASILEIOS BELAGIANNIS received the degree in engineering from Democritus University of Thrace, and the M.Sc. degree in computational science and engineering and the Ph.D. degree from TU München. He was a Postdoctoral Research Assistant with the University of Oxford (Visual Geometry Group). He is currently a W1-Professor with Ulm University, Germany. Prior to joining Ulm University, he was conducting research at OSRAM, Germany. His research interests include deep learning and machine learning, including applications in computer vision, automated driving, and medical image analysis.

KLAUS DIETMAYER (Member, IEEE) was born in Celle, Germany, in 1962. He received the Diploma degree (equivalent to M.Sc. degree) in electrical engineering from the Technical University of Braunschweig, Germany, in 1989, and the Dr.-Ing. degree (equivalent to Ph.D. degree) from the University of Armed Forces, Hamburg, Germany, in 1994. In 1994, he joined Philips Semiconductors Systems Laboratory, Hamburg, as a Research Engineer. In 1996, he became a Manager in the field of networks and sensors for automotive applications. In 2000, he was appointed as a Professor at Ulm University in the field of measurement and control. He is currently a Full Professor and the Director of the Institute of Measurement, Control and Microtechnology, School of Engineering and Computer Science, Ulm University. His research interests include information fusion, multi-object tracking, environment perception for advanced automotive driver assistance, and E-mobility. He is a member of the German Society of Engineers VDI/VDE.