Communications in Information and Systems
Volume 22, Number 1, 103–129, 2022

Learning point cloud shapes with geometric and topological structures

Yijie Zhu, Zhetong Dong, Chi Zhou, and Hongwei Lin*

3D point cloud semantic analysis is challenging due to irregular locations and ill-posed sparse representations. In this study, we explore the intrinsic structures of point clouds, which assist convolutional neural networks in classification and segmentation tasks. The network is referred to as a geometric and topological structures based convolutional neural network (GTS-CNN). Firstly, the method extracts meaningful geometric adjacency for each surface point as well as the topological persistence information for the whole point cloud. Then the GTS-CNN processes this information with a multi-head mechanism. There are three branches within the network executing graph neighborhood message passing, point position-related inference, and persistence image feature embedding, respectively. In this way, an expressive descriptor is obtained with a combination of three kinds of features, leading to a robust and finely grained representation. Experiments on standard benchmarks, such as ModelNet40 and ShapeNet, show that our network achieves promising performance compared to state-of-the-art methods.

1. Introduction

Recently, 3D point cloud semantic analysis has become popular in many applications of computer vision and computer graphics [1], especially in autonomous driving and robotic manipulation. While 2D images reveal the appearance of an object's surface that is touchable or untouchable, e.g., cup and sky, point clouds record the position and even the surface normal, which is highly useful in scene analysis. With the wide use of 3D sensor devices, including LiDAR scanners and Microsoft Kinect devices, point clouds of physical objects and scenarios are within reach of hands. Because point clouds provide depth information and filter the light effect, agents can focus on just the geometric properties of objects, and do more in 3D space than that with 2D images.

*Corresponding Author.

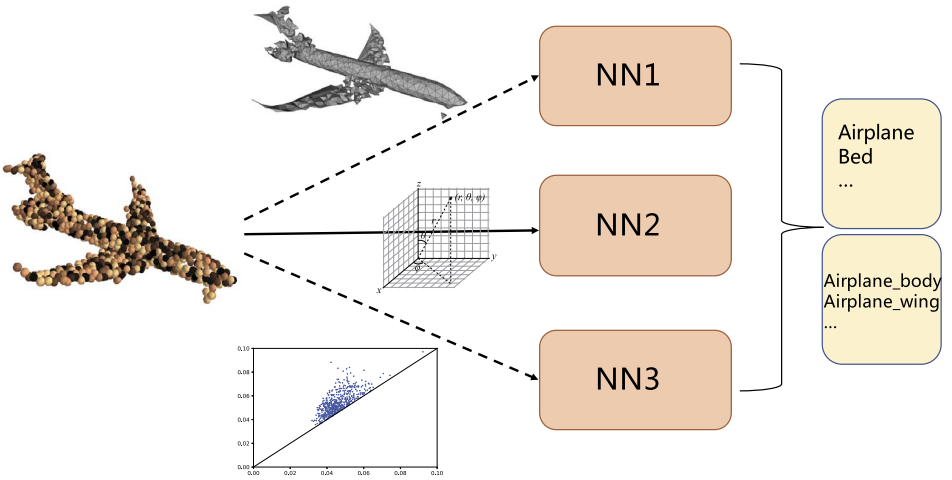


Figure 1: Illustration of our GTS-CNN. It has three branches: The first branch (NN1) deals with the mesh neighborhoods, the second branch (NN2) extracts relative position weighted features, and the third branch (NN3) processes topological homology structures.

For image analysis tasks, past years have witnessed tremendous advances thanks to deep learning technology, especially convolutional neural networks (CNNs). CNNs make use of the local relationships of image patches to extract high-level features from the RGB values of image pixels. With the help of special loss functions for certain tasks, CNNs learn compact representations for targets. Unlike images, point clouds have irregular layouts, limiting the direct application of CNNs. Variants of CNNs have emerged to adapt irregular data, including structured, unstructured and unordered formats.

In this study, we propose a neural network architecture mainly for point cloud shape classification and segmentation tasks (see Figure 1 for details). Our geometric and topological structure-based convolutional neural network (GTS-CNN) consists of three branches, each of which is responsible for one aspect of the point cloud. The first branch extracts features of the neighborhood for each point. Most point cloud networks exploit local relationships via k-nearest neighborhood (KNN) or ϵ-ball, where ambient points connect to the target point, so a graph is established. This way, an unstructured representation is converted to a structured one. However, those methods have not taken the intrinsic geometry into consideration so far. Roughly speaking, a 3D point cloud of the object is a sparse representation of a 2D manifold, which can be used as a prior in the neural network. Therefore in

the first branch, we extract meaningful geometric adjacent relationships using surface reconstruction technology instead of KNN at first, and then use a graph neural network module to process the graph signal. For the second branch, a special transformation is proposed to encode spatial relationships of points, which is a kind of absolute and relative transformation. For the third branch, the topological structure of input data is considered explicitly. Currently, the topological invariants of shapes is seldom exploited in geometric deep learning despite being a significant characteristic. For instance, the topology of a cup with a handle is totally different from that of a ball because there is a handle loop in the cup. Such observations can assist in distinguishing between these two objects. We utilize *persistence diagram*, a commonly used topological summary in *persistent homology* [2], to capture the topological features which describe topology of point clouds via multiple scales. Once we have a suitable discrete representation, we can feed it into the network branch to get a more compact feature for an object.

Furthermore, the whole network provides flexibility for designing. Many neighborhood building methods and topological feature representation methods can be employed. Our contributions are summarized as follows:

- Bring persistence homology to point cloud networks. As far as we know, our work is the first to focus on topological representation of 3D point clouds in a deep learning framework.
- Previous neighbor selection methods cannot effectively reflect the intrinsic geometry of point clouds, so a mesh reconstruction method is employed in this study to exploit a more accurate local neighborhood.
- We propose a radial-position-aware multi-layer perceptron (MLP) layer, and combine it with a graph subnetwork and a convolutional subnetwork using a multi-head mechanism. Experiments show that our hybrid architecture achieves promising results on benchmarks.

2. Related work

2.1. Graph neural network

In geometric deep learning [3], data lying in a manifold with or without connections are called geometric data. Traditional convolutional nerual networks cannot directly process such data. For example, an image is viewed as a function in Euclidean space, sampled on a plane grid. Natural images have the properties of stationarity, locality and compositionality that make the convolution/pooling operator effective. However, for data that can be

modeled as a graph, such as social networks, biomolecular graphs, and surface mesh, no such well-defined stationarity property exists. Geometric data have no regular layout in feature space, but nevertheless we can define a signal on graphs (e.g., on the vertices) and modify the convolutional layer.

Deep learning methods on graphs can be divided into two groups: those based on spatial/patch methods and those based on spectral methods. Spatial methods [1, 4–6] implement convolution directly on each node and its neighbors. The feature of a node at an intermediate layer of a graph neural network is collected from neighbors and itself in the previous layer. After such a message passing stage, there may be a pooling module. It usually performs graph coarsening or clustering to reduce the graph size. Recently, graph attention networks [7] have been developed to compute *importance* e_{ij} of node j's features to node i using a shared attentional mechanism.

For spectral methods [1, 8], the graph Fourier transformation is applied to the graph signal and the convolution is performed in the spectral space. One main drawback of those methods comes from the difficulty in generalizing across graphs with different sizes. SyncSpecCNN [9] alleviates this problem by aligning all Laplacians to common canonical space.

2.2. Point cloud network

With the seminal work of PointNet [10] in 2016, the deep learning community began to pay attention to 3D point cloud classification and part segmentation. Most works on 3D point cloud networks convert points to a graph or clusters at the beginning. PointNet++ [11] finds that PointNet lacks point-wise geometric context information, so it hierarchically samples a part of the points and applies KNN or ϵ-ball search to build local regions, and then uses mini-PointNet to extract features. Both of them aggregate the local/non-local context with max pooling, and no special convolution is explored in the networks.

Klokov and Lempitsky [12] arrange a point cloud as a KD-tree, which is invariant to permutation and translation, partially invariant to jitter, but variant to rotations. Lei et al. [13] use octree to build a network with the ability to account for more data in point neighbors than KD-tree, and then employ bins of the quantized sphere to encode detailed local geometric information of the points. However, it is still variant to rotations. Methods [14–17] build KNN neighborhoods first, followed by spatial graph neural networks to aggregate high-level features. Methods in [18–21] mine more local geometric details in point clouds. For example, Shen et al. [18] model local shape via kernel templates. Lan et al. [19] project the relative displacement

to the coordinate system and weight neighbor features by angles. The works in [22, 23] extend point clouds to 3D continuous space and estimate local density in different ways. Specifically, Atzmon et al. [22] use weighted basis functions to approximate the point cloud density and convolve with weighted kernel functions in continuous form. On the contrary, Wu et al. [23] directly estimate kernel density offline and use it to weight discrete convolutions.

Spectral methods also appear in point cloud analysis. Te et al. [24] use a distance matrix to build the Laplacian, and then apply graph spectral method. In [25, 26], the surface signal is projected onto a sphere and the spherical spectral transformation equivariant to rotations is implemented via convolution.

Some works apply a point cloud network to other applications, including 3D point cloud denoising [27], adversarial samples generation [28], and geodesic distance estimation [29]. Zhang et al. [30] detect point cloud saliency via shifting some points toward the point cloud center. Their experiments find that the radial positions of points affect the performance of point cloud network. Inspired by this work, the second branch of our network actively takes care of the radial position effect.

2.3. Persistent homology

Homology [31] is an algebraic tool to compute topological invariants that represent independent holes in a topological space. The k-dimensional holes of a topological space X are described by an algebraic element of the k-th homology group $H_k(X)$. Concretely, a 0-dimensional (1-dimensional, 2-dimensional) homology class represents a connected component (a loop, a trapped volume). Note that the inclusion between two topological spaces X and Y induces a homomorphism between the corresponding k-th homology groups $H_k(X)$ and $H_k(Y)$. Persistent homology [2] is a computational tool that indicates the change of homology features as the scale parameter increases. It has a compact expression called persistence diagram (PD), a multiset of points in the Cartesian plane. For a fixed choice of homological dimension k, each topological feature is represented by a point $(x, y) \in \mathbb{R}^2$, whose coordinates x and y are the scale parameters at which the feature first appears and disappears, respectively. The life span $y - x$ $(y \geq x)$ is called the *persistence* of the feature. Practically, features with high persistence are taken as important features, but those with relatively low persistence are typically regarded as noise. In the case of point clouds, PDs can be produced using Vietoris-Rips filtration.

In order to get a stable vector representation of a PD, Adams et al. [32] proposed using normalized symmetric Gaussian to convert a PD to a scalar function over the plane, leading to *persistence image* (PI). Kusano et al. [33] developed a kernel method for PDs. Recently, Dong et al. [34] proposed a vectorizing representation of PD based on B-spline surface (PBSG), of which the control grid is concatenated into a vector. In this study, PI and PBSG are used to encode point clouds. To the best of our knowledge, this is the first use of persistence homology in deep 3D point cloud analysis.

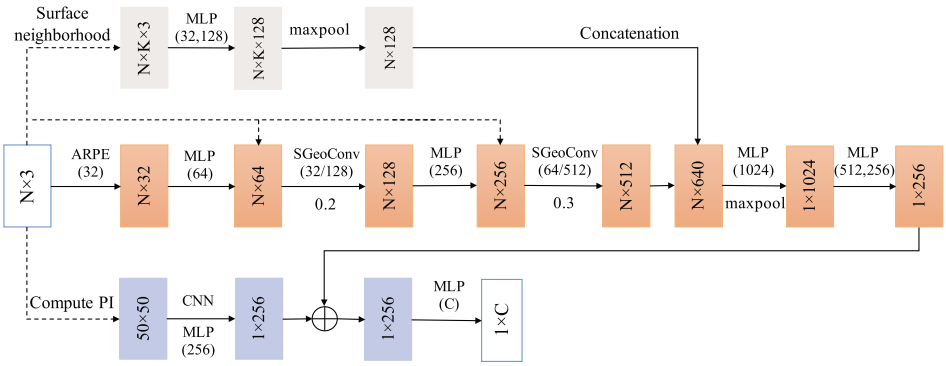
3. Methods

The tasks studied in this paper are to perform 3D point cloud classification and part segmentation. We starts with a set of point clouds $\chi = \{X^{(m)}\}$ with $X^{(m)} = \{x_i^{(m)}\}$ and $m = 1, \cdots, M$, $i = 1, \cdots, N$, where $x_i^{(m)} \in \mathbb{R}^3$. For the classification task, each $X^{(m)}$ needs to be classified to a specific object class, while for the segmentation task, each point in $X^{(m)}$ are segmented to a specific semantic part. We now explain the main modules of GTS-CNN. As stated above, it consists of three branches, each of which processes one aspect of input point clouds. Figure 2 illustrates the full network architectures.

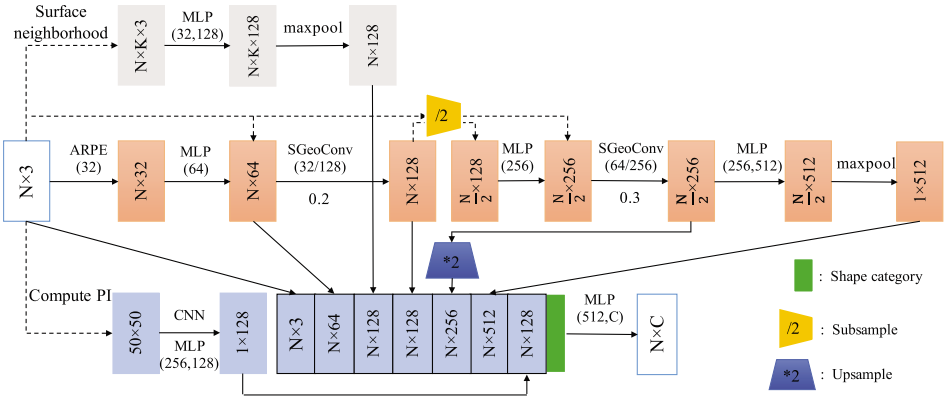
3.1. Mesh neighborhood

The first branch is a pure spatial graph convolutional subnetwork. It is so lightweight that it only contains a neighborhood constructor and multilayer perceptrons. For a point cloud sampled from a 3D shape, methods used to find the neighborhood for each point are well studied in computer graphics. In most point cloud neural networks, the neighborhood is prescribed via KNN or ϵ-ball. A neighborhood should be both large enough so that the points sufficiently describe the local geometry, and appropriately small so that the local features are preserved. In general, ϵ-ball query guarantees a fixed scale of local region, which is more plausible in common space compared to KNN. However, in the case of non-uniform sampling, ϵ-ball query would fail in low density regions. Assuming that data are scattered at a low-dimensional manifold, Zhu et al. [35] regularized the loss of the neural network with a complex constraint over the input and output features, but the loss is hard to optimize. Since point cloud data is located on a 2D manifold, it is natural to construct the neighborhood in a mesh reconstruction view, rather than an underdetermined manifold regularization.

Given a 3D point cloud of a shape, a triangular mesh can be reconstructed. In this study, the mesh is built based on [36]. It greedily selects



(a) Classification network



(b) Segmentation network

Figure 2: Our GTS-CNN network architectures for classification (a) and segmentation (b). 'SGeoConv' denotes the Spherical GeoConv operator. The numbers of learnable parameters of the networks (a) and (b) are 1.8M and 1.4M, respectively.

local optimal triangles in the Delaunay triangulation of the input points in an incremental way. With a control of the topology of the reconstruction, this approach is robust to sparse point clouds and creates few ambiguities. For a mesh vertex p, its 1-ring neighborhood contains the vertices adjacent to it, its 2-ring neighborhood includes its 1-ring neighbors and the vertices adjacent to its 1-ring neighbors, and so on. To get the K neighbors of a given point p, we first identify its largest n-ring neighborhood with size not more than K, and then expand the neighborhood by randomly adding points in its $(n+1)$-ring neighborhood but not in its n-ring neighborhood, till reaching

the size K. The K neighbors so generated is called the *mesh neighborhood* of the point p.

3.2. Absolute and relative transformations

The second branch of our GTS-CNN is concerned with absolute and relative transformations of points. Denote an intermediate representation of points at layer l by $X^{(l)}_{N\times C}$, where $x^{(l)}_i \in \mathbb{R}^C$ is the i-th row of $X^{(l)}$ as well as the original position $x_i = x^{(0)}_i$. In general, a spatial method can be expressed as:

$$x^{(l+1)}_i = \gamma_\theta(x^{(l)}_i, \square_{j\in\mathcal{N}(i)}\phi_\theta(x^{(l)}_i, x^{(l)}_j, e_{ij})) \tag{1}$$

where $\square$ is the reduce operator (e.g., mean, sum), and $\gamma_\theta, \phi_\theta$ are parameterized functions with learnable parameters θ. For each point i, a local spherical coordinate system is constructed with its origin at point i. For all neighbor points j of i, a relative transformation aggregates their features into point i. Meanwhile, an absolute transformation performs on point i in the world coordinate system. Extending from GeoConv [19], the formula of the convolution (referred to as Spherical GeoConv) in the second branch of GTS-CNN on point i at layer l is:

$$\begin{aligned} x^{(l+1)}_i &= a(i) + r(i) \\ a(i) &= MLP_{ball(i)}(x^{(l)}_i) \\ r(i) &= \sum_{j\in\mathcal{N}(i)} h(i,j,r) \end{aligned} \tag{2}$$

where functions $h(i,j,r)$ and $r(i)$ are the same as in [19]. Compared to GeoConv, the difference is the absolute transform module $a(i)$. Since experiments in [30] indicate that radial distance for some points has a large impact on the fundamental discriminant ability of the network, the module $a(i)$ is adapted to make use of radial position. While GeoConv treats each point at different radial position equally, we divide radial space into several parts $\mathbf{R} = \{0, R_1, R_2, \cdots, R_N\}$ and each part $[R_{k-1}, R_k]$ owns an independent MLP, so that for one point located at a certain shell part, the corresponding MLP is used to extract features. The operator $ball(i)$ identifies which shell the point i lives in and is invariant to rotations. With Spherical GeoConv, the point cloud networks may identify salient points effectively and become more robust.

Absolute and relative transformations are popular in graph and point cloud neural networks, such as in [14, 20, 21]. For example, absolute and relative position embedding (ARPE) is commonly used:

$$x_i^{(l+1)} = \max_{j \in \mathcal{N}(i)} \{MLP([x_i^{(l)}, x_j^{(l)} - x_i^{(l)}])\} \tag{3}$$

Therefore, it is also included as the first layer of the second branch in this study.

3.3. Learning from persistence image

The third branch of GTS-CNN cares about the topological structure of point clouds. As introduced above, the PD of a given 3D point cloud is a set of 2D points. There are various theoretically guaranteed methods to vectorize it. In practice, a 2D convolutional neural network is powerful enough to learn high-level features from 2D data, using PI as the input representation. In the following, the procedure for producing PIs is introduced [32].

Specifically, define the linear map $T : \mathbb{R}^2 \to \mathbb{R}^2$ such that $T(x, y) = (x, y - x)$. Let g represent the normalized symmetric Gaussian with mean $u = (u_x, u_y) = (x, y - x)$, which is defined as

$$g(x, y; u) = \frac{1}{2\pi\sigma^2} e^{-[(x-u_x)^2+(y-u_y)^2]/2\sigma^2}. \tag{4}$$

A PD in birth-death coordinates is denoted as $\mathcal{P}$. The corresponding persistence surface of $\mathcal{P}$ is given by

$$\rho_{\mathcal{P}}(x, y) = \sum_{u \in T(\mathcal{P})} f(u) g(x, y; u), \tag{5}$$

where $f : \mathbb{R}^2 \to \mathbb{R}$ is a nonnegative weighting function. f is critical to ensure the stability of the transformation from a PD to a persistence surface. Finally, the surface is reduced to a finite-size vector by discretizing it on a relevant subdomain and integrating $\rho_{\mathcal{P}}$ over each subdomain. A PI is a collection of pixels $I(\rho_{\mathcal{P}})_p = \iint_p \rho_{\mathcal{P}} dxdy$. As Figure 3 shows, two objects with different topological features have different PIs. With such a topological descriptor, one can distinguish objects with different topologies. Moreover, PI lets us use the topological features in machine learning tasks, e.g., the classification task using support vector machine (SVM). However, few deep learning methods have taken topological descriptors as input so far.

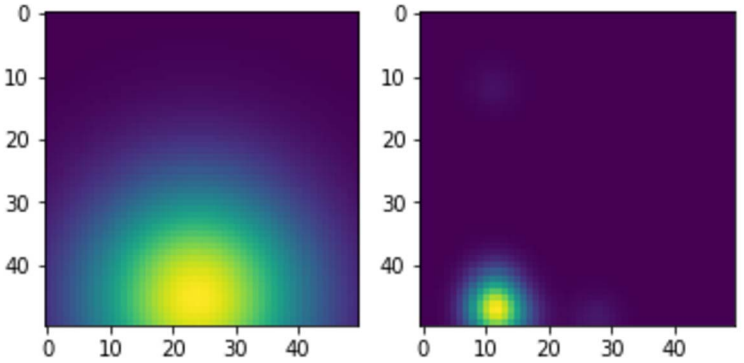


Figure 3: An example of PI. left: Airplane, right: Cup.

In this study, an image convolutional neural network is used to deal with PIs for classification. Different from Adams et al. [32], the weighting function in Equation (5) is set to be sigmoid $f(u) = 1/(1 + e^{-su_y+t})$ with translation and scale factors t and s, where u_y is the persistence length. PI can be viewed as a gray image that has a local relationship, and the sigmoid weighting function indicates the point region that we care about. Points in a PD that has short persistence are ignored as noise. Those having excessively long persistence are regarded as trivial features. Therefore, the larger its persistence is, the slower the corresponding weight increases.

3.4. Final architecture

The final neural network is a hybrid architecture, which is based on Geo-CNN [19]. It explicitly combines geometric structure and topological structure. The first branch takes a point cloud and the mesh neighborhood as input, followed by a simple graph neural network. The graph neural network generates new feature at each vertex by aggregating the features at its adjacent mesh vertices. The second branch is a modified point cloud neural network, which is the main branch of the three. It implements comprehensive absolute and relative transformations on point clouds. The third branch actually is a 2D convolutional neural network. Without bells and whistles, we design a small network (referred to as PI-Net) to extract the features from PI of point clouds. The parameters of PI-Net are shown in Table 1. The output features of the third branch will be added to the output of the main branch (i.e., the second branch).

Table 1: Network structure in the third branch. Each layer is followed by batch normalization and ReLU

Layer	Type	Parameters
1	conv	size: $4 \times 4 \times 1 \times 4$ stride: 2, pad: 0
2	conv	size: $3 \times 3 \times 4 \times 16$ stride: 2, pad: 0
3	conv	size: $3 \times 3 \times 16 \times 64$ stride: 2, pad: 0
4	conv	size: $3 \times 3 \times 64 \times 128$ stride: 2, pad: 0
5	linear	size: 512×256

4. Numerical experiments

All experiments in this study are implemented using Pytorch on an Nvidia RTX 2080Ti. The loss function of networks is a cross entropy function with weight decay regularization. It is trained by Adam optimizer. The weight decay factor is set to 10^{-5} and the learning rate is initialized to 0.001, which drops with a rate of 0.6 every 10 epochs. The mini-batch size is set to 32 during training. The parameters of the network are initialized using the He initialization method [37]. Each Spherical GeoConv has two parts with $\mathbf{R} = \{0, 0.5, 1.0\}$. The translation and scale factors t and s are 0.5 and 10, respectively. The mesh neighborhood size K is set to 16 in the first branch and 32 in ARPE. In addition, our code is released and publicly available at https://github.com/ZJUCAGD/GTS-CNN.

4.1. Datasets

Two public datasets are used to evaluate point cloud networks. One is ModelNet40 [38], which contains 12,311 object meshes in 40 categories. In the dataset, 9,843 models are used for training and 2,468 models for testing. The other is the ShapeNet part dataset [39], used for shape segmentation tasks. It contains 16,881 shapes with 16 categories. Shapes are labeled into 50 parts in total. Each category is composed of two to six specific parts. Models of these two datasets are represented by vertices and faces. For each model, thousands of points are sampled from the mesh for the following numerical experiments.

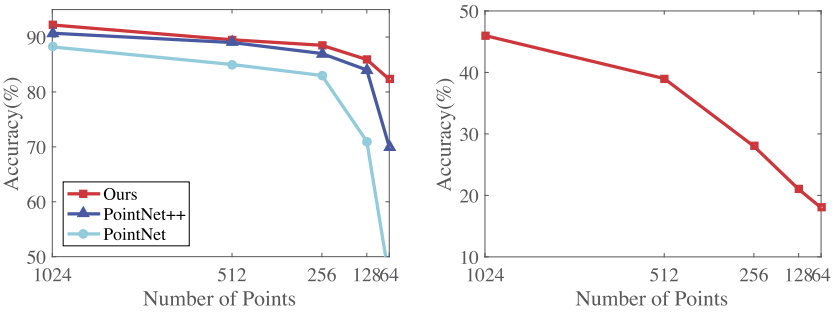
Table 2: ModelNet40 shape classification results: The methods are sorted by time

Method	input	#points	acc. (%)	#params(M)/acc
PointNet [10]	xyz	1k	89.2	1.6/88.4
PointNet++ [11]	xyz	1k	90.7	1.7/90.4
ECC [5]	xyz	1k	87.4	–
Kd-Net [12]	xyz	32k	91.8	3.6/81.0
DGCNN [14]	xyz	1k	92.2	1.8/91.7
SO-Net [16]	xyz	2k	90.9	1.7/90.5
KCNet [18]	xyz	1k	91.0	0.9/88.8
PCNN [22]	xyz	1k	92.3	8.2/90.8
PointCNN [17]	xyz	1k	92.5	0.9/88.0
Ψ-CNN [13]	xyz	10k	92.0	0.8/85.8
PAT [20]	xyz	10k	91.7	–
RS-CNN(no voting) [21]	xyz	1k	**92.9**	1.3/90.9
Ours	**xyz**	**1k**	**92.2**	**1.8/92.2**
PointNet++ [11]	xyz, nor	5k	91.9	–
SO-Net [16]	xyz, nor	5k	93.4	–
PointConv [23]	xyz, nor	1k	92.5	20/89.8
Geo-CNN [19]	xyz, nor	1k	93.4	4.1/90.3

4.2. Point cloud classification

The same as in PointNet++, to obtain 3D point clouds, 1024 points are uniformly sampled from each object in ModelNet40 and normalized into a unit ball centered at the origin. Therefore, all objects are in a similar spatial scale. During training, each point in a point cloud is jittered by a Gaussian random vector with zero mean and 0.01 standard deviation. The mesh neighborhood and PI are computed offline in advance for saving training time. Because of heterogeneous structures, the whole network, shown in Figure 2(a), is separated into two parts (the first two branches and the third branch) to train 100 epochs and 50 epochs respectively, and then they are combined to fine-tune through 60 epochs.

Table 2 summaries previous methods and our method for a quantitative comparison. The majority of methods classify point clouds based on input point positions; in addition, some of them may incorporate normal vector information or sample more points to achieve state-of-the-art scores. We notice that some methods, e.g., RS-CNN, adopt sufficient augmentation/learning techniques, like voting, to make the performance satisfactory, which is important in reality tasks. However, the abuse of these delicate strategies makes things confusing. Since the effectiveness of the methods



(a) Full Net (b) PI-Net branch

Figure 4: Robustness to subsampling of point clouds.

listed above should be verified under the same experimental background, we re-implement the neural networks that are available at authors' websites using Pytorch and set all configurations (including optimizer, number of input points 1024, random seed, etc.) the same as ours. All networks are trained from scratch and the results are listed in column "#params(M)/acc" in Table 2. The column shows the number of parameters of networks and records the overall accuracy. The accuracies of methods drop more or less with unified training and prediction strategies. Among them, our GTS-CNN achieves comparable performance (92.2%).

In some cases, the number of points is less than 1024, so it is necessary to evaluate the robustness to subsampling of point clouds. Figure 4(a) shows the result that the networks are not very sensitive to uniformly subsampling. When the number of points drops to 64, our classification network still achieves high accuracy above 80%, but the compared networks PointNet and PointNet++ achieve accuracies lower than 70%.

4.3. Point cloud segmentation

Following the experiment setup in PointNet, 2048 points are uniformly sampled from each object in the ShapeNet part segmentation dataset. The data preprocessing is the same as that in the point cloud classification (Section 4.2). In the point cloud segmentation task, point intersection-over-union (IoU) is computed at test stage. Per-class and per-instance mean IoUs are reported as evaluation criteria.

Our segmentation architecture is shown in Figure 2(b). It is similar to the classification network, but contains a subsample module and an upsample module, which are implemented with furthest point sampling and nearest

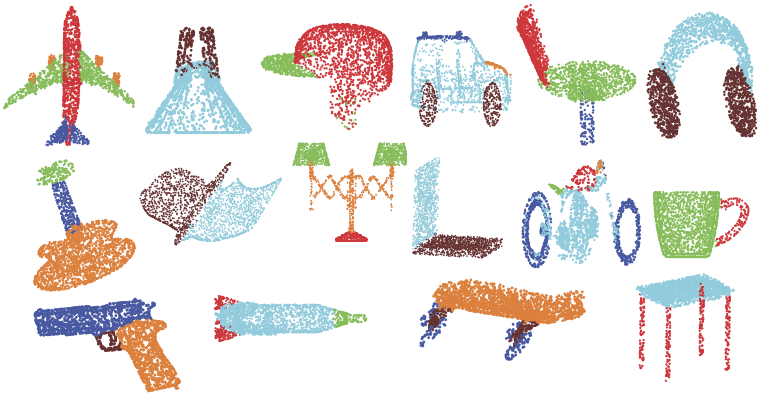


Figure 5: Visualization of prediction results on ShapeNet test dataset. Best viewed in color.

neighborhood interpolation. Table 3 summarizes the quantitative results on the ShapeNet test dataset. Compared with previous methods, our network achieves better performance with point positions as input. The network obtains class mIoU of 82.8% and instance mIoU of 85.2%. Figure 5 depicts several segmentation results visually for 16 categories.

4.4. Ablation study

Deep neural networks have many uncertain aspects and confusing behaviors that influence experimental performance. In this section, we will investigate persistence homology used in the network, compare neighborhood selection methods possibly employed in the first branch, and study radial space partition in SGeoConv. Finally, we will validate the effectiveness of the multi-head design.

Influence of persistent homology. There are some ways to use persistence homology to classify shape point clouds. Firstly, we can choose different vector representations of the PD. As stated above, PI and PBSG [34] have properties that are stable to shape deformation. Secondly, hyperparameters of vector representation, including vector/image size, weight function, etc., are crucial for capturing topological features with long persistence, and are insensitive to points packed closely together. Experiments indicate that a sigmoid function is a good choice for the weight function. Thirdly, classifiers are important for classification tasks. Recent topological data analysis

Table 3: Point cloud segmentation quantitative results on ShapeNet test dataset. 'nor' stands for normal vector information

method	Kd-Net [12]	PointNet [10]	SCN [40]	PCNN [22]	KCNet [18]	DGCNN [14]	**Ours**	PointNet++ [11]	SyncSpecCNN [9]	SO-Net [16]
Input	4k	2k	1k	2k	2k	2k	2k	2k nor	mesh	1k nor
Class mIoU	77.4	80.4	81.8	81.8	82.2	82.3	**82.8**	81.9	82.0	80.8
Instance mIoU	82.3	83.7	84.6	85.1	84.7	85.1	**85.2**	85.1	84.7	84.6
airplane	80.1	83.4	83.8	82.4	82.8	**84.2**	83.4	82.4	81.6	81.9
bag	74.6	78.7	80.8	80.1	81.5	**83.7**	82.3	79.0	81.7	83.5
cap	74.3	82.5	83.5	85.5	86.4	84.4	86.8	**87.7**	81.9	84.8
car	70.3	74.9	79.3	**79.5**	77.6	77.1	78.2	77.3	75.2	78.1
chair	88.6	89.6	90.5	90.8	90.3	**90.9**	90.8	90.8	90.2	90.8
earphone	73.5	73.0	69.8	73.2	76.8	78.5	**80.2**	71.8	74.9	72.2
guitar	90.2	91.5	**91.7**	91.3	91.0	91.5	91.5	91.0	93.0	90.1
knife	87.2	85.9	86.5	86.0	87.2	**87.3**	86.7	85.9	86.1	83.6
lamp	81.0	80.8	82.9	**85.0**	84.5	82.9	84.1	83.7	84.7	82.3
laptop	94.9	95.3	96.0	95.7	95.5	**96.0**	95.4	95.3	95.6	95.2
motorbike	57.4	65.2	69.2	**73.2**	69.2	67.8	71.6	71.6	66.7	69.3
mug	86.7	93.0	93.8	94.8	94.4	93.3	**94.9**	94.1	92.7	94.2
pistol	78.1	81.2	82.5	83.3	81.6	82.6	**83.9**	81.3	81.6	80.0
rocket	51.8	57.9	62.9	51.0	60.1	59.7	59.1	58.7	60.6	51.6
skateboard	69.9	72.8	74.4	75.0	75.2	75.5	74.3	76.4	**82.9**	72.1
table	80.3	80.6	80.8	81.8	81.3	82.0	81.9	**82.6**	82.1	82.6

Table 4: Testing accuracy for different persistence diagram configurations. "NN" stands for the simple neural network PI-Net

Method	size	classifier	acc. (%)
PBSG	10×10	SVM	28.1
	10×10	NN	33.3
	20×20	NN	35.5
PI	10×10	NN	38.5
	20×20	NN	42.5
	50×50	NN	**46.0**
	100×100	NN	43.2

uses machine learning methods to prove the validity itself, but ignores promoting the learning methods. Although deep neural networks are designed for end-to-end learning and PD is just an intermedia feature representation of raw data, it is novel to revisit classification with a combination of both. Table 4 shows the results when we only use topological features to classify shapes in ModelNet40. It suggests that a simple neural network is better than the classic classifier SVM. Too large of the image size, such as a PI with 100×100 pixels, is not the best for the task, because a PD contains finite discrete points so that most areas of a large image become meaningless.

Another key point is that PD is robust to rigid transformation and slight perturbation, so that a PI-based CNN can works well with moderate shape deformation. To verify this point, a small 2D point cloud dataset is constructed. The dataset samples points from cups with various shapes and orientations in the planar space. It consists of 100 rectangular cups with handle, 100 rectangular cups without handle, 100 elliptical cups with handle, and 100 elliptical cups without handle. The whole dataset is uniformly divided into training and testing splits, keeping the ratio of the number of objects at 7:3. The task of PI-based CNN is to identify whether an input point cloud has a handle or not. As Figure 6 shows, when the handle of a cup shrinks, the persistence of the highlighted point in the PD becomes small but the feature point does not disappear. Different shapes of the main cup bodies scarcely influence PDs. To the end, experiments on such a toy dataset suggest a lightweight CNN (e.g., PI-Net) makes satisfactory predictions with an accuracy up to 88.3%. There are theorems ensuring the stability of PD [41] and the stability of PI [32].

The computation of the topological features in the third branch is not only invariant to rotations and translations w.r.t. input point clouds, but also robust to subsampling of point clouds. Figure 4(b) shows that the accuracy of the PI-Net branch linearly decays as the point cloud becomes

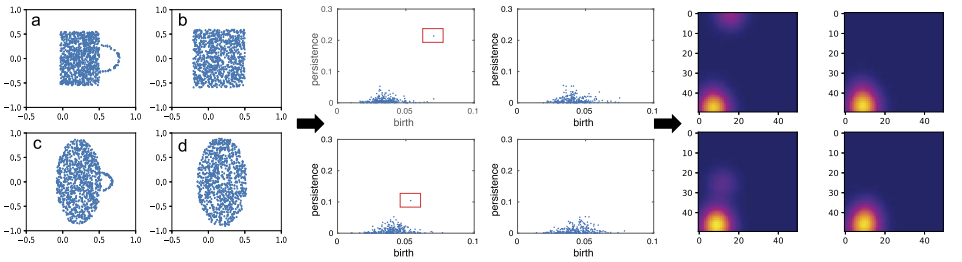
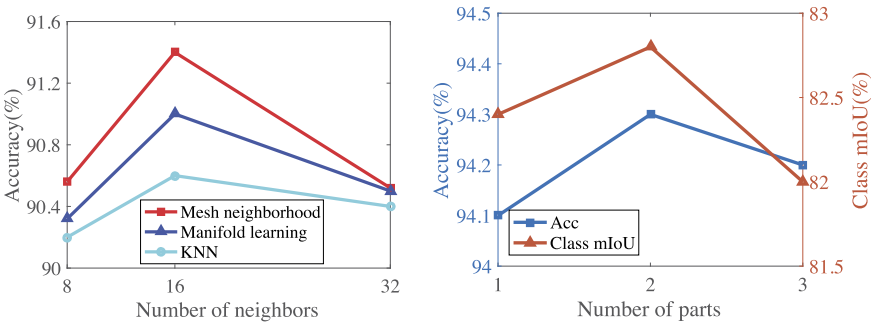


Figure 6: Illustration of persistence image behavior under shape deformation. First, the 4 different point clouds at left are transformed into PDs, shown in the middle subplots. Then they are transformed into PIs, shown at right. The probabilities of the existence of a handle for (a–d) predicted by a lightweight CNN are 99.9%, 12.1%, 99.9% and 3.1%, respectively.

sparse, putting such deterioration in a smooth and predictable direction. In a word, persistence homology does help neural networks to be robust against common transformations applied to point cloud in an effective way.

Comparison of neighbor selection methods. We compare three neighbor selection methods (KNN, manifold learning and mesh neighborhood) that can be employed in the network. The baseline network is the first two branches of a GTS-CNN classification network with KNN method for neighborhood construction, and the other two neighbor selection methods are used in the same network in replacement of KNN for comparison. As the manifold learning neighbor selection method, the *neighborhood selection algorithm* in adaptive manifold learning [42] is employed to build the neighborhood. It starts with the KNN neighborhood for each data point, then prunes noisy neighbors step by step using a geometric criterion, and expands the neighborhood set by adding back some pruned points. Moreover, in the mesh neighborhood method, the neighborhood is induced from a mesh as introduced in Section 3.1. Figure 7(a) presents the experimental results of the three methods. Evidently, the KNN method achieves the lowest scores compared to the other two methods. The mesh neighborhood method is better than the adaptive manifold learning method. The latter does not take global surface situation into consideration and is a subset of KNN after all, while the mesh neighborhood method maintains the topological quality of the shape.

Radial space partition for absolute transformation in SGeoConv. In world space, absolute transformation aims to get global features of points.



(a) Neighborhood (b) SGeoConv

Figure 7: Testing performance for different neighborhood construction methods (a), and different number of radial parts in SGeoConv (b).

Table 5: Ablation study of GTS-CNN on ModelNet40. Model A is GeoCNN [20] with 2 GeoConv

Model	mesh neighborhood	Spherical Geoconv	PI	acc. (%)
A				89.8
B	✓			90.6
C	✓	✓		91.4
D	✓	✓	✓	**92.2**

We study how many shell parts the unit sphere should be split into. Figure 7(b) shows the result that when the sphere is split into two parts, the network achieves higher performance (both accuracy and mIoU) on ShapeNet part segmentation compared to 1 part (no split) and 3 parts.

Contributions of three branches. We train the classification network on ModelNet40 with different configurations of the three branches. Results are presented in Table 5. Model A is the baseline, and Model D is the final network we use, which achieves the highest score on the dataset. See Appendix A for more experimental results.

5. Conclusion

In this paper, we have presented GTS-CNN, a hybrid deep learning architecture for inferring semantic information within point clouds. The contribution is that we use the geometric prior of a given point cloud and persistent homology to improve the performance of point cloud networks. With the

geometric prior, it is possible to fit the local points with a smooth surface or graph mesh, thus aiding the network in training and protecting it from unrelated points. For topological information, it is difficult to dig out topological features within a point cloud, and some methods have been tried in previous work, such as multiscale neighborhoods. Persistent homology is an advanced tool for describing the topological structure of objects. Most neural networks are able to learn features by themselves; however, the existence of persistence homology makes things more understandable and controllable. Consequently, the proposed networks equipped with geometric and topological structures achieve high performance on two tasks. Experimental and theoretical analysis shows the effectiveness of the method. However, there are still some problems unsolved. The computation burden is unavoidable in neighborhood construction and PI generation. GPU-accelerated algorithms are one solution. Another avenue for future work is the study of the philosophy of usage for geometric and topological structures in different 3D tasks, because there are so many choices to select and combine together.

Appendix A

This appendix shows the details of our study. The topological representations are discussed from theoretical and experimental views in Section A.1. The training loss of the final networks is showed in Section A.2. And a procedure of constructing a 2D point cloud dataset is introduced in Section A.3.

A.1. More studies on the topological representation

Stability of PI-Net

Lemma 1. *Let $f, g : \mathbb{X} \to \mathbb{R}$ be tame Lipschitz functions on a metric space whose triangulation grows polynomially with a constant exponent j. Then there are constants C and $k > j$ no smaller than one such that the degree q Wasserstein distance between persistence diagrams $PD(f)$ and $PD(g)$ is $W_q(PD(f), PD(g)) \leqslant C \cdot \|f - g\|_\infty^{1-k/q}$, for every $q \geqslant k$.*

Lemma 1 states that a persistence diagram of a tame Lipschitz function that is extended from a 3D point cloud is stable with regard to local deformation of the point cloud. See [41, 43] for details.

Lemma 2. *The persistence image $I(\rho_{\mathcal{P}})$ with Gaussian distributions is stable with respect to the 1-Wasserstein distance between diagrams $\mathcal{P}, \mathcal{P}'$. More*

precisely,

$$\|I(\rho_{\mathcal{P}}) - I(\rho_{\mathcal{P}'})\|_2 \leqslant (\sqrt{5}|\nabla f| + \sqrt{\frac{10}{\pi}}\frac{\|f\|_\infty}{\sigma})W_1(\mathcal{P}, \mathcal{P}'),$$

where f is a weighting function.

Lemma 2 describes that PI is stable when it is transformed from PD with a sigmoid function f. See [32] for details.

Proposition 3. *The PI-Net in the third branch of GTS-CNN is stable with respect to local deformations of point clouds.*

Here, we make a brief explanation. Given a point cloud X, a second point cloud X' is generated from X via local deformations or noisy point addition. There are some well-defined Lipschitz functions g, g' extended from X, X' such that $W_1^q(PD(g), PD(g')) \leqslant C^q \cdot \|g - g'\|_\infty^{q-k}$. Thus the distance between PIs of two point clouds $\|I(X) - I(X')\|_2$ is controlled by $\|g - g'\|_\infty$, indicating only local deformations between two images. Assuming a simple image convolutional neural network owns deformation stability as in [3, 44, 45], the feature representation output from PI-Net is stable with local deformations of input image, so that PI-Net is stable with respect to local deformations of point clouds.

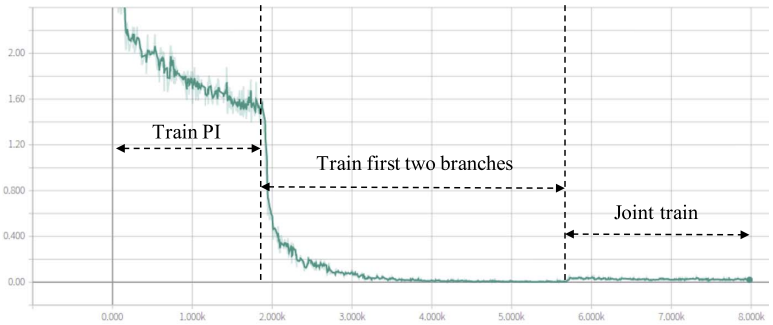
A.1.1. Aggregation for topological features To investigate how to combine the output topological features of the PI-Net and the features of the second branch, we tested several symmetric functions, where *Max, Sum* and *Mean* are element-wise functions. Table 6 suggests that *Mean* function is the most effective aggregation operation for testing accuracy on ModelNet40 dataset.

A.2. Training of networks

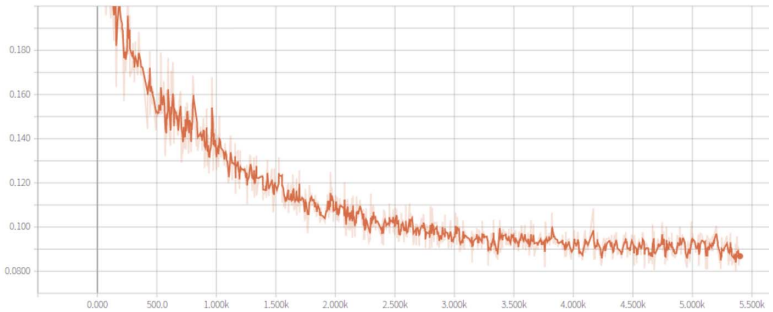
To see how networks evolve at the training stage, we measured loss curves, shown in Figure 8, where loss value was recorded every 8 batches with mini-batch size 32. For the classification task, the network was trained via three sub-processes, which gained a higher test accuracy compared to direct training according to our experimental observation. For the segmentation task, the network was trained directly and was not necessary to fine-tune later. Both loss curves converged at the end, indicating the effectiveness of network models.

Table 6: Different aggregation operations for combining output features of second branch (*i.e.*, geometric feature $\mathbf{x}$) and the third branch (*i.e.*, topological feature $\mathbf{y}$)

Aggregation operation	acc. (%)
Max ($max(\mathbf{x}, \mathbf{y})$)	91.3
Sum ($\mathbf{x} + \mathbf{y}$)	91.9
Mean ($\frac{\mathbf{x}+\mathbf{y}}{2}$)	**92.2**
Concat ($[\mathbf{x}, \mathbf{y}]$)	91.3



(a) Loss of the classification network



(b) Loss of the segmentation network

Figure 8: Visualization of loss evolution at the training stage for the classification network (a) and the segmentation network (b): The training stage of (a) was separated into three sub-processes while the training stage of (b) only contained a joint training process.

A.3. Construction of toy dataset

In Section 4.4, a toy dataset was constructed to assist in our study. In this part, we describe the procedure of construction in detail.

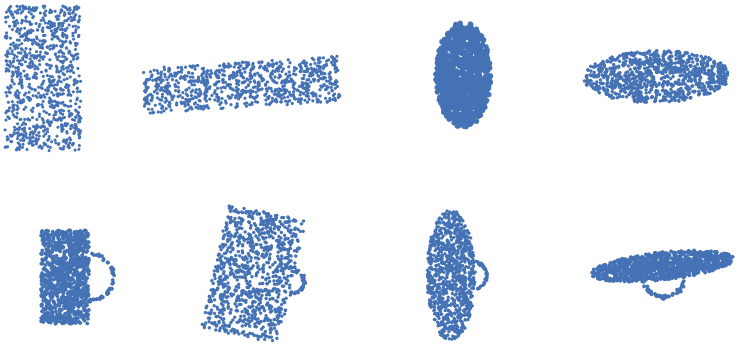


Figure 9: Visualization of cups in the toy dataset.

- Main body. Because both rectangle and ellipse could be determined by two parameters (*e.g.*, major axis and minor axis), two numbers were randomly sampled from corresponding ranges ($[0.8, 2.8]$ and $[0.2, 1.0]$ for ellipse, $[0.4, 3.4]$ and $[0.2, 1.5]$ for rectangle) to generate a rectangle/ellipse. Then 950 points were randomly sampled inside the rectangle/ellipse.
- Handle. A handle was a semi-circle arc, of which radius was in the range $[0.1, 0.3]$. Then 50 points were randomly sampled from the arc followed by small perturbation.
- Finally, a point cloud of a 2D cup was generated after rotating points from both the main body and the handle by a random angle. Figure 9 presents several samples.

Acknowledgments

This work is supported by the National Natural Science Foundation of China under Grant nos. 61872316, 61932018, and the National Key R&D Plan of China under Grant no. 2020YFB1708900.

References

[1] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.

[2] Herbert Edelsbrunner, David Letscher, and Afra Zomorodian. Topological persistence and simplification. *Discrete and Computational Geometry*, 28:511–533, 2002. MR1949898

[3] Michael M. Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond Euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

[4] Jonathan Masci, Davide Boscaini, Michael Bronstein, and Pierre Vandergheynst. Geodesic convolutional neural networks on riemannian manifolds. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 37–45, 2015.

[5] Martin Simonovsky and Nikos Komodakis. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3693–3702, 2017.

[6] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M. Bronstein. Geometric deep learning on graphs and manifolds using mixture model CNNs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5115–5124, 2017.

[7] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. *International Conference on Learning Representations*, 2018.

[8] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*, pages 3844–3852, 2016.

[9] Li Yi, Hao Su, Xingwen Guo, and Leonidas J. Guibas. SyncSpecCNN: Synchronized spectral CNN for 3D shape segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2282–2290, 2017.

[10] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3D classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 652–660, 2017.

[11] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J. Guibas. PointNet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems*, pages 5099–5108, 2017.

[12] Roman Klokov and Victor Lempitsky. Escape from cells: Deep kd-networks for the recognition of 3D point cloud models. In *Proceedings*

of the IEEE International Conference on Computer Vision, pages 863–872, 2017.

[13] Huan Lei, Naveed Akhtar, and Ajmal Mian. Octree guided CNN with spherical kernels for 3D point clouds. *arXiv preprint arXiv:1903.00343*, 2019.

[14] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph CNN for learning on point clouds. *arXiv preprint arXiv:1801.07829*, 2018.

[15] Francis Engelmann, Theodora Kontogianni, Jonas Schult, and Bastian Leibe. Know what your neighbors do: 3D semantic segmentation of point clouds. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[16] Jiaxin Li, Ben M. Chen, and Gim Hee Lee. So-Net: Self-organizing network for point cloud analysis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9397–9406, 2018.

[17] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. PointCNN: Convolution on X-transformed points. In *Advances in Neural Information Processing Systems*, pages 820–830, 2018.

[18] Yiru Shen, Chen Feng, Yaoqing Yang, and Dong Tian. Mining point cloud local structures by kernel correlation and graph pooling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4548–4557, 2018.

[19] Shiyi Lan, Ruichi Yu, Gang Yu, and Larry S. Davis. Modeling local geometric structure of 3D point clouds using Geo-CNN. *arXiv preprint arXiv:1811.07782*, 2018.

[20] Jiancheng Yang, Qiang Zhang, Bingbing Ni, Linguo Li, Jinxian Liu, Mengdie Zhou, and Qi Tian. Modeling point clouds with self-attention and gumbel subset sampling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3323–3332, 2019.

[21] Yongcheng Liu, Bin Fan, Shiming Xiang, and Chunhong Pan. Relation-shape convolutional neural network for point cloud analysis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8895–8904, 2019.

[22] Matan Atzmon, Haggai Maron, and Yaron Lipman. Point convolutional neural networks by extension operators. *ACM Transactions on Graphics (TOG)*, 37(4):71, 2018.

[23] Wenxuan Wu, Zhongang Qi, and Li Fuxin. PointConv: Deep convolutional networks on 3D point clouds. *arXiv preprint arXiv:1811.07246*, 2018.

[24] Gusi Te, Wei Hu, Amin Zheng, and Zongming Guo. RG-CNN: Regularized graph CNN for point cloud segmentation. In *2018 ACM Multimedia Conference on Multimedia Conference*, pages 746–754. ACM, 2018.

[25] Taco S. Cohen, Mario Geiger, Jonas Köhler, and Max Welling. Spherical CNNs. *arXiv preprint arXiv:1801.10130*, 2018.

[26] Carlos Esteves, Christine Allen-Blanchette, Ameesh Makadia, and Kostas Daniilidis. Learning SO(3) equivariant representations with spherical CNNs. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 52–68, 2018.

[27] Chaojing Duan, Siheng Chen, and Jelena Kovacevic. 3D point cloud denoising via deep neural network based local surface estimation. In *ICASSP 2019–2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8553–8557. IEEE, 2019.

[28] Chong Xiang, Charles R. Qi, and Bo Li. Generating 3D adversarial point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9136–9144, 2019.

[29] Tong He, Haibin Huang, Li Yi, Yuqian Zhou, Chihao Wu, Jue Wang, and Stefano Soatto. GeoNet: Deep geodesic networks for point cloud analysis. *arXiv preprint arXiv:1901.00680*, 2019.

[30] Tianhang Zheng, Changyou Chen, Kui Ren, et al. Learning saliency maps for adversarial point-cloud generation. *arXiv preprint arXiv:1812.01687*, 2018.

[31] Allen Hatcher. *Algebraic Topology*. Cambridge University Press, 2002. MR1867354

[32] Henry Adams, Tegan Emerson, Michael Kirby, Rachel Neville, Chris Peterson, Patrick Shipman, Sofya Chepushtanova, Eric Hanson, Francis Motta, and Lori Ziegelmeier. Persistence images: A stable vector representation of persistent homology. *The Journal of Machine Learning Research*, 18(1):218–252, 2017. MR3625712

[33] Genki Kusano, Kenji Fukumizu, and Yasuaki Hiraoka. Kernel method for persistence diagrams via kernel embedding and weight factor. *The Journal of Machine Learning Research*, 18(1):6947–6987, 2017. MR3827077

[34] Zhetong Dong, Hongwei Lin, and Chi Zhou. Persistence B-spline grids: Stable vector representation of persistence diagrams based on data fitting. *arXiv preprint arXiv:1909.08417*, 2019.

[35] Wei Zhu, Qiang Qiu, Jiaji Huang, Robert Calderbank, Guillermo Sapiro, and Ingrid Daubechies. Ldmnet: Low dimensional manifold regularized neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2743–2751, 2018.

[36] David Cohen-Steiner and Frank Da. A greedy Delaunay-based surface reconstruction algorithm. *The Visual Computer*, 20(1):4–16, 2004.

[37] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In *CVPR*, 2015.

[38] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3D ShapeNets: A deep representation for volumetric shapes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1912–1920, 2015.

[39] Li Yi, Vladimir G. Kim, Duygu Ceylan, I. Shen, Mengyan Yan, Hao Su, Cewu Lu, Qixing Huang, Alla Sheffer, Leonidas Guibas, et al. A scalable active framework for region annotation in 3D shape collections. *ACM Transactions on Graphics (TOG)*, 35(6):210, 2016.

[40] Saining Xie, Sainan Liu, Zeyu Chen, and Zhuowen Tu. Attentional ShapeContextNet for point cloud recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4606–4615, 2018.

[41] David Cohensteiner, Herbert Edelsbrunner, and John Harer. Stability of persistence diagrams. *Discrete and Computational Geometry*, 37(1):103–120, 2007. MR2279866

[42] Zhenyue Zhang, Jing Wang, and Hongyuan Zha. Adaptive manifold learning. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (2):253–265, 2012.

[43] David Cohen-Steiner, Herbert Edelsbrunner, John Harer, and Yuriy Mileyko. Lipschitz functions have Lp-stable persistence. *Foundations of Computational Mathematics*, 10(2):127–139, 2010. MR2594441

[44] Avraham Ruderman, Neil C. Rabinowitz, Ari S. Morcos, and Daniel Zoran. Pooling is neither necessary nor sufficient for appropriate deformation stability in CNNs. *arXiv preprint arXiv:1804.04438*, 2018.

[45] Alberto Bietti and Julien Mairal. Group invariance, stability to deformations, and complexity of deep convolutional representations. *Journal of Machine Learning Research*, 20(25):1–49, 2019. MR3911432

Yijie Zhu
State Key Lab. of CAD&CG
Zhejiang University
Hangzhou, 310058
China
E-mail address: 3130102998@zju.edu.cn

Zhetong Dong
School of Mathematical Sciences
Zhejiang University
Hangzhou, 310027
China
E-mail address: ztdong@zju.edu.cn

Chi Zhou
School of Mathematical Sciences
Zhejiang University
Hangzhou, 310027
China
E-mail address: elonzhou@zju.edu.cn

Hongwei Lin
State Key Lab. of CAD&CG
School of Mathematical Sciences
Zhejiang University
Hangzhou, 310027
China
E-mail address: hwlin@zju.edu.cn

Received July 15, 2020