Neural Implicit Embedding for Point Cloud Analysis

Kent Fujiwara and Taiichi Hashimoto
LINE Corporation
{kent.fujiwara, taiichi.hashimoto}@linecorp.com

Abstract

We present a novel representation for point clouds that encapsulates the local characteristics of the underlying structure. The key idea is to embed an implicit representation of the point cloud, namely the distance field, into neural networks. One neural network is used to embed a portion of the distance field around a point. The resulting network weights are concatenated to be used as a representation of the corresponding point cloud instance. To enable comparison among the weights, Extreme Learning Machine (ELM) is employed as the embedding network. Invariance to scale and coordinate change can be achieved by introducing a scale commutative activation layer to the ELM, and aligning the distance field into a canonical pose. Experimental results using our representation demonstrate that our proposal is capable of similar or better classification and segmentation performance compared to the state-of-the-art point-based methods, while requiring less time for training.

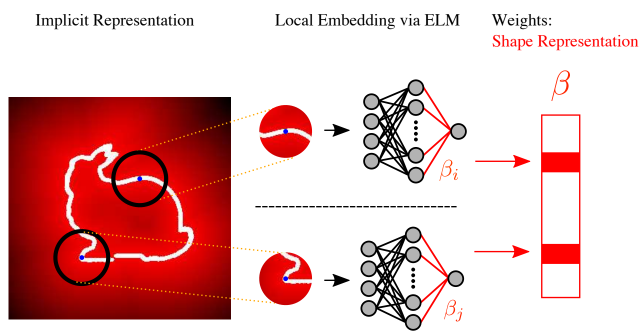


Figure 1: Proposed representation. A specific type of a neural network, Extreme Learning Machine, is responsible for embedding a local portion of the implicit representation around a point in the point cloud. The trainable weights β from the ELMs, concatenated into a single matrix, are the representation of the corresponding point cloud instance (in this case: Stanford Bunny). The representation can be utilized for tasks such as classification and segmentation.

1. Introduction

Analysis of unstructured point cloud data is one of the central topics in computer vision, as 3-dimensional data of various objects can now be easily captured through commercial sensors. Point clouds play an important role in key areas, such as autonomous driving and robotics, where spatial information of the surrounding environment is critical [4, 31]. Point cloud data can also be interpreted as sets, whose analysis is known to have various applications [48].

Unlike 2-dimensional images, 3-dimensional point clouds are generally unordered, unstructured, and represented in an arbitrary coordinate system. Therefore, there is no straightforward method to apply convolutional neural networks to point clouds, despite their recent success in the analysis of 2D data. Many of the current methods attempt to create a regular representation by converting point clouds into voxel data or even rendered images. However, in these cases, information of the original points is lost, making such tasks as point-wise label assignment significantly more difficult. This fact necessitates a different approach to producing a representation of 3D point clouds.

An ideal representation of point clouds should also be robust to arbitrary changes of the origin location, orientation, and scale of the 3D coordinate system used to describe point cloud data. Conventional methods for 3D point cloud data analysis generally attempt to obtain such representation through data augmentation by applying various transformations and adding perturbations to the training data.

We propose a novel representation of points clouds that encapsulates the local information around points clouds, and addresses such issues as robustness to coordinate system change, scaling, and permutation. As shown in Fig. 1, the key idea is to embed an implicit function of one point cloud instance into multiple neural networks, whose weights are used as the feature of the point cloud instance.

We firstly convert each point cloud into an implicit function, the distance field. We acquire the distance field of each instance using a sphere of fixed sampling points, which we call a sampling sphere. We place one sphere on top of each point in the point cloud to acquire the distance field.

We then embed the distance field within each sampling sphere in a neural network to make the implicit representation invariant to sampling point permutation. One network is responsible for representing the distance field within one sampling sphere. The weights from all the networks are concatenated into a matrix to be used as the representation of the corresponding point cloud instance. We enable comparison among the network weights from each instance by employing a specific type of neural network for embedding distance fields: Extreme Learning Machine (ELM).

The representation, consisting of weights obtained from local embedding networks, can be made invariant to coordinate change and scaling by altering the network components and aligning the distance fields. Scale invariance is achieved by using ReLU activation layer in each ELM, and coordinate invariance by using the canonical coordinates of the sampling points, obtained through aligning the distance field to the canonical space determined by the distribution of the distance values unique to each instance.

The major contribution of the representation is that it provides a simple method to capture local details. The experiments demonstrate that our method can provide state-of-the-art accuracy, in classification and segmentation, and is robust against perturbations such as rotation and scaling. Our representation required only simple neural networks to conduct these tasks, leading to reduction in training time.

2. Related Work

With the advances in deep learning techniques and more datasets [35, 2] starting to become available to the public, analysis of point clouds has developed into an important tasks in the field of computer vision. There are various topics in which 3D information is utilized, including shape retrieval [39], correspondence [41] and registration [37].

Methods on point cloud data analysis mainly focus on finding a representation that can be used to train neural networks to extract characteristic information regarding point clouds. Current approaches can be categorized into 2 classes: Grid-based methods [47, 23, 42, 13, 8] and Point-based methods [27, 15, 18, 20, 49]. Aside from the two classes, there are also attempts to combine different representations [28], and approaches [45, 1] that embed shapes into a latent space using generative networks [10, 14].

Grid-based methods attempt to convert point clouds into regular structures to allow convolution of local information. Voxel-based methods [47, 23] convert point clouds into voxel data. As the voxels are ordered and structured, convolution can be conducted simply by applying 3D filters. However, accuracy of these methods depend heavily on the resolution of voxels. Despite the recent efforts to make volumetric approach more efficient [30, 7, 16], these methods are known to be computationally demanding with larger number of voxels. Image-based methods [36, 42, 13] convert points clouds into 2D renderings and use 2D convolutional neural networks to conduct various analyses. Image-based methods are known to be highly successful at shape classification task, as they utilize external pre-trained models, usually trained using various 2D image datasets. However, these methods cannot be applied to tasks such as segmentation, where labels need to be assigned to individual points. The grid-based representations are also covariant to coordinate change, and require data from multiple viewpoints. Some methods convert geometric data to 2D planar data [34, 32]. These methods require the direction of gravity, which is not necessarily available. Our representation does not require such supervision, as we achieve rotational invariance by projection to the canonical pose.

Point-based methods attempt to directly use the coordinates of the points. PointNet [27] proposed to feed point cloud data directly into a neural network. The method avoids the issue of point permutation by applying a symmetric function in higher dimensional space to obtain a global feature. This proposal has led to a new trend of methods directly operating on points [29, 20]. As PointNet produces a global signature for the entire point cloud, recent methods propose strategies to acquire local information from point clouds. Various methods introduce structures, such as k-d trees and graphs, to capture the local relationship among unstructured points [15, 43, 33, 44, 17]. Other methods propose novel strategies for convolution to gather information from the neighboring points [46, 49, 22, 19, 40]. There are also attempts to introduce various local signatures, such as distance to neighboring points and angle between local surface normals, and use them to represent point clouds [5, 6, 50, 21]. Our method shares the same philosophy of incorporating local information surrounding point clouds. We capture the distance field around each point, and encapsulate it in a fixed-size vector using a neural network.

Recently, there have been proposals to design unsupervised representations of the point clouds before conducting analyses [9]. Li *et al.* [18] proposed to first conduct unsupervised learning on each point cloud instance by creating a self-organizing map and identifying a set of nodes unique to each individual. The node information as well as the neighborhood information is used to train a deep neural network. Our method also conducts preprocessing to learn a representation for each instance in an unsupervised manner. We design our embedding strategy to capture the implicit representation around the objects points. The proposed method is also designed to achieve invariance to important elements such as scaling, point permutation, and coordinate change.

Recent methods use neural networks to embed implicit representation of shapes for modeling purposes [24, 3]. Park *et al.* [26] use distance field obtained from a set of sampling points to train an auto-decoder, which returns a latent

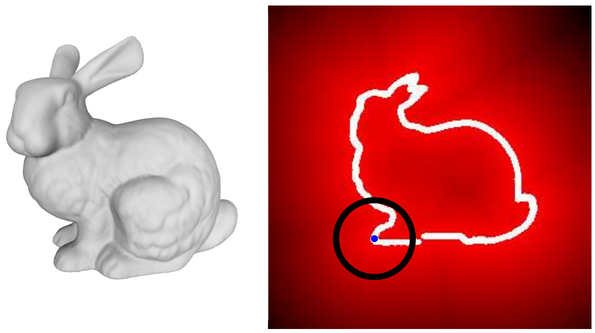


Figure 2: Implicit representation: Distance field $\mathbf{\Phi}$. Darker colors indicate larger distance. Sampling sphere (black circle in 2D) is placed over each surface point (blue).

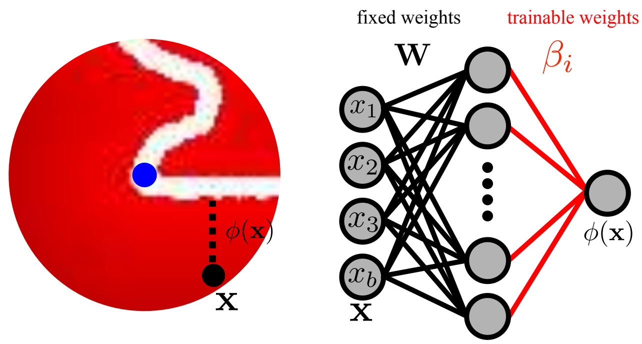


Figure 3: Embedding Neural Network: Extreme Learning Machine. The distance field $\mathbf{\Phi}$ for the i-th point is embedded in the ELM.

vector corresponding to the provided distance field. The latent vector, along with the sampling coordinates is used to train a deep neural network to learn the distance field of various shapes. Instead of using an auto-decoder to obtain a signature for a given distance field, we propose to use the weights of the neural network as the representation. Our method achieves invariance to sampling point permutation, as well as invariance to scaling of coordinate values.

3. Proposed Method

We propose a novel method that overcomes the difficulty of creating a representation for unstructured point clouds. Our method consists of two steps: Conversion of point cloud data into an implicit representation, the distance field, and network embedding of the implicit representation.

3.1. Implicit Representation: Distance Field

There are two reasons behind choosing the distance field as the representation. The first is that it achieves invariance to point permutation. The same set of points results in the same distance field, regardless of the point order. The second reason, which is the key to our method, is that distance is scale-covariant. When the coordinate values are scaled, the distance is also scaled by the same factor. This is critical for achieving scale invariance for embedding distances into a neural network with scale commutative property.

Given a point cloud $\mathcal{P}$ consisting of n surface points $\mathbf{p} \in \mathbb{R}^3$, the distance function ϕ at a sampling point in the surrounding space $\mathbf{x} \in \mathcal{X}$ is defined as

$$\phi(\mathbf{x}) = \min_{\mathbf{p} \in \mathcal{P}} \|\mathbf{x} - \mathbf{p}\| \, . \tag{1}$$

In practice, we prepare what we call a sampling sphere on top of each surface point in the point cloud $\mathcal{P}$ as shown in Fig. 2. The sampling sphere consists of m sampling points $\mathbf{x}$ distributed within the sphere. In this method, the location of the m sampling points $\mathbf{x}$ are identical among sampling spheres of all the point cloud instances. The coordinates of the sampling points in each sphere are normalized, with the corresponding surface point $\mathbf{p}$ as the center of each sphere. Therefore, the distance within the normalized sphere i is

$$\phi_{\mathcal{P}_i}(\mathbf{x}) = \min_{\tilde{\mathbf{p}}} \|\mathbf{x}_i - \tilde{\mathbf{p}}\| \, . \tag{2}$$

where $\tilde{\mathbf{p}} = \mathbf{p} - \mathbf{p}_i$ is the normalized coordinates of the surface points centered at the i-th point $\mathbf{p}_i$.

The distance field, as is, is covariant to coordinate change. When sampling spheres and the target shape is rotated, the coordinates of the sampling points change and but the distance field inside the sphere remains the same.

We define the radius of each sampling sphere to be relative to the radius of the sphere encapsulating the entire point cloud. If the underlying shape is either partial or open, we define the radius of the sampling spheres to be the average distance from each point to its k-th nearest point, assuming that the density of points are even among all point clouds.

3.2. Parameterization of Implicit Representation

The distance field from each sampling sphere is then embedded into a neural network so that its weights capture the characteristics of the distances within each sampling sphere. For each surface point $\mathbf{p}$, we train one neural network to capture the characteristics of the distance field around it.

A regular neural network has numerous possibilities of weight combinations, as both the input-to-hidden-layer weights $\mathbf{W}$ and the corresponding hidden-to-output-layer weights $\boldsymbol{\beta}$ are simultaneously optimized during training. To enable comparison between the neural network weights, they have to be embedded in the same metric space. We, therefore, employ a specific type of a neural network, namely an Extreme Learning Machine (ELM) [12].

In this method, we employ a simple 3-layer ELM. An ELM is a feed forward neural network whose input-to-

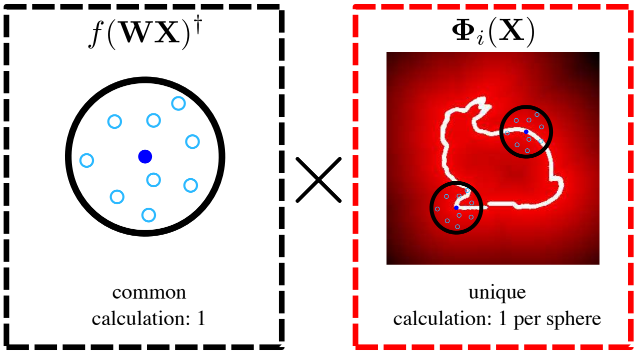


Figure 4: Efficient embedding strategy. As the points in the sampling sphere are normalized with respect to the center, the pseudoinverse $\mathbf{H}^{\dagger} = f(\mathbf{WX})^{\dagger}$ has to be calculated only once. This matrix is multiplied by the distance values obtained from the sampling sphere around each point, providing us with a unique weight $\boldsymbol{\beta}$ for each sampling sphere without repetitive calculation of the pseudoinverse.

hidden-layer weights are fixed $\mathbf{W}$, as shown in Fig. 3. The input to the ELM is the sampling points $\mathbf{X} \in \mathbb{R}^{m\times 3}$ in each sampling sphere centered around a surface point. We train the ELM to return $\boldsymbol{\Phi}_{\mathcal{P}_i}(\mathbf{X})$, the distance from the sampling points to the closest point in the point cloud $\mathcal{P}$. The objective function of ELM is

$$\boldsymbol{\beta}_i^* =_{\boldsymbol{\beta}_i} \|\boldsymbol{\Phi}_{\mathcal{P}_i}(\mathbf{X}) - \boldsymbol{\beta}_i^{\top} f(\mathbf{WX}^{\top} + \mathbf{b})\|_F^2 \,, \tag{3}$$

where $\boldsymbol{\beta}_i \in \mathbb{R}^k$ is the network weights, f is a non-linear activation function, $\mathbf{W} \in \mathbb{R}^{k\times 3}$ is the random weight, and $\mathbf{b} \in \mathbb{R}^k$ is the random bias. To obtain the weights $\boldsymbol{\beta}$ so that the network output matches the target $\boldsymbol{\Phi}_{\mathcal{P}_i}(\mathbf{X})$, we simply need to solve for the pseudoinverse of $\mathbf{H} = f(\mathbf{WX} + \mathbf{b})$ to obtain $\boldsymbol{\beta}^* = \mathbf{H}^{\dagger}\boldsymbol{\Phi}_{\mathcal{P}}(\mathbf{X})$, or more robustly, obtain

$$\boldsymbol{\beta}_i^* = (c\mathbf{I} + \mathbf{H}^{\top}\mathbf{H})^{-1}\mathbf{H}^{\top}\boldsymbol{\Phi}_{\mathcal{P}_i}(\mathbf{X}) \,, \tag{4}$$

where, c is a constraint added to the diagonal elements of $\mathbf{H}^{\top}\mathbf{H}$. To reflect the scale, we set c as the variance of the sampling points $\mathbf{X}$. By definition, we are able to obtain a unique solution $\boldsymbol{\beta}^*$ regardless of the permutation of $\mathbf{X}$.

We fix $\mathbf{W}$ of all ELMs to a common orthogonalized random matrix, as proposed in a prior work [12]. The weights $\boldsymbol{\beta}$ are determined by the distance field values. We will exploit this characteristic of ELMs to provide a unique set of weights $\boldsymbol{\beta}$ for each sampling sphere.

3.2.1 Fixing W

We have two purposes behind fixing the input-to-hidden-layer weights $\mathbf{W}$. The first can be observed from the formulation of ELM. ELM obtains the output value by calculating $\boldsymbol{\beta}^{\top} f(\mathbf{WX}^{\top} + \mathbf{b})$, which is the inner product between a vector $f(\mathbf{WX}^{\top} + \mathbf{b})$ and the trainable weight $\boldsymbol{\beta}$. Fixing $\mathbf{W}$ is equivalent to introducing the Euclidean metric to the metric space of $\boldsymbol{\beta}$. Without defining the inner product, trainable weights $\boldsymbol{\beta}$ obtained from ELMs cannot be easily compared.

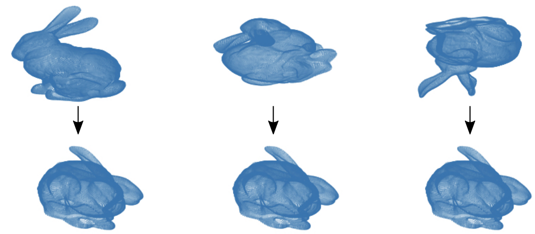


Figure 5: Coordinate invariant representation. The original points, which can be considered as the 0-level set of the distance field, are shown. Despite randomly rotating the original model, our method aligns the distance fields.

The second purpose is efficiency. We can make the training process significantly efficient by taking advantage of the structure of sampling spheres, using the fact that the sampling points in all the spheres are identical and normalized. Generally, calculation of trainable weights $\boldsymbol{\beta}$ requires the pseudoinverse $\mathbf{H}^{\dagger} = (c\mathbf{I} + \mathbf{H}^{\top}\mathbf{H})^{-1}\mathbf{H}^{\top}$ to be calculated as many times as the number of the sampling spheres n. However, as mentioned previously, we use the same set of normalized sampling points to capture the local distance field within each sphere. Therefore, we can define $\mathbf{H} = f(\mathbf{WX} + \mathbf{b})$ to be the same for all sampling points. This means that the pseudoinverse $\mathbf{H}^{\dagger}$ only has to be calculated once throughout the dataset, as shown in Fig 4. To obtain the weights $\boldsymbol{\beta}$ of all the sampling spheres simultaneously, we concatenate all the distance fields obtained by the sampling spheres into a single matrix, so that

$$\boldsymbol{\beta}^* = (c\mathbf{I} + \mathbf{H}^{\top}\mathbf{H})^{-1}\mathbf{H}^{\top}\boldsymbol{\Phi}_{\mathcal{P}}(\mathbf{X}) \,, \tag{5}$$

where $\boldsymbol{\Phi}_{\mathcal{P}}(\mathbf{X}) \in \mathbb{R}^{m\times n}$ and $\boldsymbol{\beta}^* \in \mathbb{R}^{k\times n}$ are matrices whose i-th columns contain the distances and the resulting ELM weights from the i-th sampling sphere, respectively.

3.3. Achieving Coordinate and Scale Invariance

We can design the ELM weights $\boldsymbol{\beta}$ to be invariant to coordinate values and scale. The invariance is achieved by modifying the input to the embedding ELM, and by selecting a specific activation layer for the ELM.

3.3.1 Coordinate Invariance: Canonical Projection

We project the distance field onto a 4-dimensional canonical space to achieve invariance to rotation. We introduce a

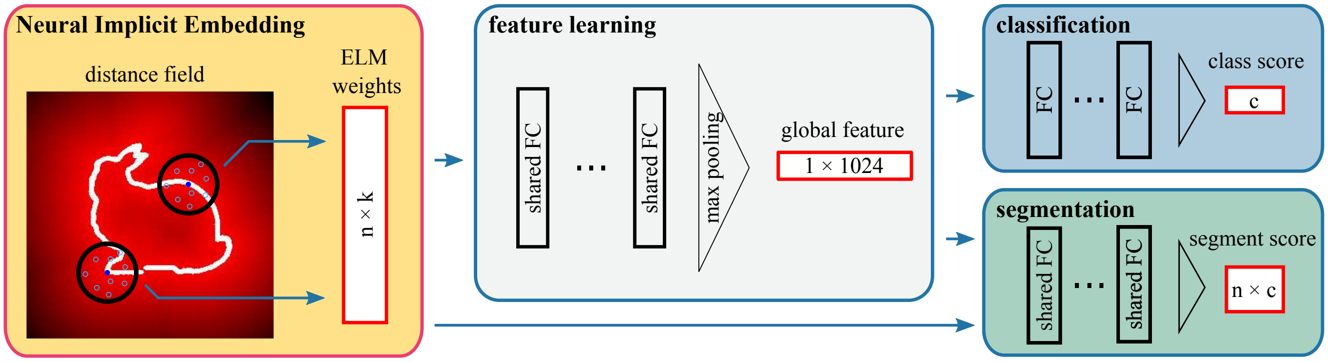


Figure 6: Network architecture for the proposed representation. The ELM weights containing local distance field information are used as input to train a global feature vector. The feature is fed into subsequent networks to conduct analysis. In the segmentation task, we concatenate the global feature with the original ELM weights corresponding to each point.

sampling matrix $\mathbf{X} = \begin{bmatrix}\mathbf{x}_1, & \mathbf{x}_2, & \cdots & ,\mathbf{x}_m\end{bmatrix}^\top \in \mathbb{R}^{m\times 3}$ and the corresponding sampled distance vector $\boldsymbol{\Phi}_{\mathcal{P}}(\mathbf{X}) = \begin{bmatrix}\phi(\mathbf{x}_1), & \phi(\mathbf{x}_2), & \cdots & ,\phi(\mathbf{x}_m)\end{bmatrix}^\top \in \mathbb{R}^m$ and concatenate them into a matrix $\mathbf{M} = \begin{bmatrix}\mathbf{X} & \boldsymbol{\Phi}_{\mathcal{P}}(\mathbf{X})\end{bmatrix}$. We then apply singular value decomposition (SVD) on $\mathbf{M}$ to obtain

$$\mathbf{M} = \mathbf{U}\mathbf{S}\mathbf{V}^\top . \tag{6}$$

As the results from SVD may contain sign ambiguity, we propose to transform the data using all the possible sign permutations. Given an input data, we prepare a vector $\mathbf{c}$ consisting of 1 and -1. We apply the signs to $\mathbf{V}^\top$ so that

$$\bar{\mathbf{V}}^\top = \mathbf{V}^\top \mathbf{C} , \tag{7}$$

where $\mathbf{C}$ is a diagonal matrix with $\mathbf{c}$ as its elements.

As can be seen by modifying eq. 6 to $\bar{\mathbf{M}} = \mathbf{M}\bar{\mathbf{V}}$, $\bar{\mathbf{V}}$ is the projection of the data matrix to a 4-dimensional canonical space. The projection allows the distance field to be aligned to a unique pose in the 4D space based on the variance of the distance field, as can be seen in Fig. 5 demonstrates the effect of applying $\bar{\mathbf{V}}$. We will demonstrate the robustness to surface point density in the experiments.

The representation in the canonical space is now invariant to rotation up to ambiguity of sign of $\mathbf{V}$. We resolve this ambiguity by preparing all the variations from possible sign permutations. The input to the ELM is $\bar{\mathbf{X}} = \mathbf{X}\bar{\mathbf{V}}_{\mathbf{X}}$, the coordinates of the sampling points in the 4D canonical space. $\bar{\mathbf{V}}_{\mathbf{X}} \in \mathbb{R}^{3\times 4}$ is the first three rows of $\bar{\mathbf{V}}$. The ELM is trained to return the distance vector $\boldsymbol{\Phi}_{\mathcal{P}}(\mathbf{X})$ corresponding to the sampling points. The elements of the distance vector is removed from the input to avoid trivial solutions.

To efficiently solve for $\bar{\mathbf{V}}_{\mathbf{X}}$, we place a global sphere surrounding the entire point cloud instance. The distance values within the sphere is used to align each instance to a unique pose. The global sphere is solely used to align the global distance field. After alignment, sampling spheres are placed on surface points to acquire the local distance fields.

3.3.2 Scale Invariance: ReLU Activation Function

We achieve scale invariance by taking advantage of the scale commutative property of the rectified linear unit (ReLU) [25]. We adopt it as the activation function f of the ELM and remove the bias term from eq. (3). The modified embedding ELM is now denoted as

$$\boldsymbol{\beta}^* =_{\boldsymbol{\beta}} \|\boldsymbol{\Phi}_{\mathcal{P}}(\mathbf{X}) - \boldsymbol{\beta}^\top f(\mathbf{W}\bar{\mathbf{X}}_b)\|_F^2 , \tag{8}$$

where $\bar{\mathbf{X}}_b \in \mathbb{R}^{m\times(4+1)}$ is $\bar{\mathbf{X}}$ with an additional column. The extra column is the bias scaled by the standard deviation of all the values in $\bar{\mathbf{X}}$. In this process, we have removed the bias term and inserted it into the input matrix.

To prove the scale invariance, we consider the case of applying a scaling factor s to the input $\bar{\mathbf{X}}_b$. The output $\boldsymbol{\Phi}_{\mathcal{P}}(\mathbf{X})$ is also scaled due to the property of distance. Therefore, eq. (8) is modified to

$$s\boldsymbol{\Phi}_{\mathcal{P}}(\mathbf{X}) \approx \boldsymbol{\beta}^{*\top} f(\mathbf{W}s\bar{\mathbf{X}}_b) . \tag{9}$$

As ReLU lets through positive values, the scaling factor s can be moved outside the activation function:

$$s\boldsymbol{\Phi}_{\mathcal{P}}(\mathbf{X}) \approx s\boldsymbol{\beta}^{*\top} f(\mathbf{W}\bar{\mathbf{X}}_b) . \tag{10}$$

The scaling factor s cancels each other out, leaving the network weights $\boldsymbol{\beta}$ unchanged. This allows our modified version of ELM to be invariant to scaling of the point clouds. Variants of ReLU, such as LeakyReLU, also preserves this quality, and can be used as the activation function.

4. Experiments

Our method is implemented with Keras, and all the calculations were executed on an Intel(R) Xeon(R) Silver 4114 CPU @ 2.20GHz computer with NVIDIA V100 graphics card. We utilized the graphics card to accelerate multiplication of the matrices involved in distance calculation. We used a network configuration similar to PointNet [27], as shown in Fig 6. Feature learning network contains 4 shared fully connected layers (1024 nodes). Batch normalization and ReLU are applied after each layer. Batch size is 16.

4.1. Classification Accuracy

We firstly use the proposed representation to classify the benchmark dataset ModelNet 10/40 [47]. To make the comparison with other methods fair, we first normalize the CAD models in the dataset by setting them to zero-mean, encapsulating the models in a unit sphere, and then uniformly sample over the model surface, as conducted in prior work [27]. 2048 surface points are sampled. The direction of gravity is known in this dataset. For comparison to other methods, we also use the original point cloud data before the proposed projection for calculation of distance fields.

We prepared 2048 sampling spheres, equivalent to the number of the surface points, and trained one ELM per each sphere to parameterize each instance in the dataset, and converted all the data in the training and testing set to ELM weights. These are used as features representing each individual shape. The fixed random weights are orthogonalized, as conducted in [38], to improve the result of regression. MLP with 3 hidden layers ($512, 256, 128$ nodes) with dropout (0.4 keep rate) is used as the classification network.

Table 1 shows the classification accuracy on both ModelNet 10 and ModelNet 40 dataset. We compared our results to some of the state-of-the-art methods. The top half of the table shows the results from methods that rely on the renderings of meshed data of the point clouds. These methods generally conduct pretraining using external datasets. The results of these methods, therefore, cannot be directly compared with point-based methods. The bottom half of the table shows the point-based methods. Our representation achieves the best results on ModelNet 10 and achieves second best on ModelNet 40. The best results were achieved using $m = 1024$ sampling points, embedding ELMs with dimension of $k = 256$, and sampling spheres with radius of 0.3. Similar to other methods, concatenating the normals and point coordinate information improved the classification result using our representation.However, the scale invariance is lost, as point coordinates vary with scaling, while our representation does not. Using the average distance to k-nearest neighbor ($k = 256$) as the sampling sphere radius achieved similar results: 96.2% and 93.2% for ModelNet 10 and 40 respectively.

Interestingly, adding noise to our representation, a common data augmentation practice, resulted in a lower accuracy. This is because the embedding using ELM already contains some margin of error, which is essentially equivalent to adding noise directly to the original point cloud data.

[1]The method applies various transformation during testing phase to conduct voting. Other methods do not explicitly conduct this operation, therefore, excluded the results achieved using this voting scheme.

			Classification Accuracy (%)	
Method	Pretrain	Data	MN 10	MN 40
MV [36]	Yes	Img	-	90.1
Dom. [42]			-	93.8
VIP [11]			94.1	92.0
RotNet [13]			98.5	97.4
Voxel [47]	None	Pt+N	83.5	77.0
Auth. [34]			88.4	83.9
Kdtree [15]			94.0	91.8
PNet++ [29]			-	91.9
SONET [18]			95.7	**93.4**
Pt2Sq [20]			95.3	92.6
PConv [46]			-	92.5
KPConv [40]			-	92.9
RSCNN [21][1]			-	92.9
ShellNet [49]			-	93.1
Ours	None	Weights	95.7	92.2
Ours	None	W+Pt+N	**96.7**	93.2

Table 1: Comparison of results on ModelNet 10 and 40. Our method performs better than most of the state of the art methods with just one representation per instance.

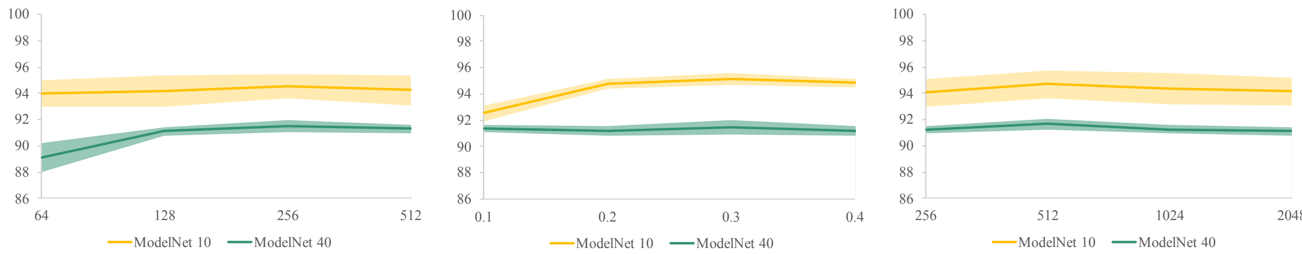
4.2. Effects of Elements Pertaining to Embedding

To observe the effect of each element to the descriptiveness of the resulting representation, we alter the dimension k of ELM weight $\mathbf{W}$, the radius of the sampling spheres, and the number m of sampling points within each sphere. In each experiment, we fixed two of the elements and changed the third one to see the change in classification accuracy.

Fig. 7 are the results from the experiment observing the effect of each element for ModelNet 10 and ModelNet 40 datasets. The averages and standard deviations of 10 evaluations from each combination are used in the plot. As can be seen from the results, the classification accuracy is low when all the elements are set too low, which was expected. However, as the elements are set to higher values, the accuracy reaches a peak before showing slight regression in accuracy. More information is involved in the embedding, and ELMs with small number of nodes cannot fully encapsulate the details that the surrounding distance fields present.

We also note that, theoretically, setting all the elements to a much larger value would result in a finer representation around each surface point, leading to improvement in classification accuracy. However, the size of the data would grow exponentially, requiring heavy memory usage. In this experiment, we set the upper threshold to each of the elements to keep the proposed representation compact.

While most point classification methods rely on deep



(a) Dimension of k

(b) Radius of sampling sphere

(c) Number of sampling points m

Figure 7: Classification accuracy (%) change under different settings. (7a) Effect of dimension k of ELM weight. (7b) Effect of sampling sphere radius. (7c) Effect of samplings points in each sphere.

	Classification Accuracy (%) points (n)			
	2048	1024	512	256
ModelNet 10	96.7	95.9	94.3	93.5
ModelNet 40	93.2	92.7	91.1	90.1

Table 2: Classification accuracy after changing number of original point cloud points n.

	Classification Accuracy (%) Maximum Angle					
	0	$\frac{\pi}{4}$	$\frac{\pi}{2}$	$\frac{3}{4}\pi$	π	$\frac{\pi}{4}$ sc.
P++ [29]	91.9	36.4	40.5	41.1	44.3	44.2
SNet [18]	**93.4**	84.5	77.4	79.0	69.4	77.1
Ours	93.2	**85.8**	**84.9**	**82.6**	**81.8**	**83.9**

Table 3: Comparison of classification results on ModelNet 40 after perturbation on test data.

neural network structures, our classification network is relatively simple, requiring shorter time for training. For classification of ModelNet 40 dataset with 512 sample points in each sampling sphere, and with the dimension of the ELM weights set to $k = 256$, the proposed method requires approximately **110 minutes** for training the neural network, compared to approximately 180 minutes for SO-Net, the most efficient method introduced in this paper.

4.3. Robustness to Number of Surface Points

To observe the robustness of the proposed implicit representation to number of points in the point clouds, we sampled a subset of points from the point clouds before calculating the distance field and used it to train ELMs in the proposed method. We prepared a sampling sphere over each point in the point set, each containing 1024 sampling points within the radius of 0.3, and embedded it in a ELM with $k = 256$. We used the same classification network to obtain the results. As can be seen from the results in Table 2, the method can classify instances relatively accurately. This can be attributed to the fact that the implicit representation, distance field, is robust to the density of the original point cloud. The results demonstrate that our representation takes full advantage of the robustness.

4.4. Effect of Canonical Embedding

To demonstrate the invariance to various factors such as coordinate change and scale, we test the accuracy of the proposed method when test data is transformed and scaled. This is common in real life situations, as not all objects are upright on the tabletop. We apply random rotation along all three axes and scaling to the test data and observe the effects on the classification accuracy. To make the comparison fair, we did not conduct data augmentation on the training data for any of the methods. Here, we compared our method to representative point-based methods, PointNet++ [29] and SO-Net [18], based on the codes made publicly available. We selected these methods, due to the fact that the calculation efficiency of our method and these methods are relatively high compared to other point-based methods.

To compare the variance of each representation, we firstly applied random rotation to the test data to observe whether the learnt model can be used to classify rotated data. Table 3 shows the classification accuracy of the three methods after applying random rotation to test data with limits to the maximum angles. The numbers indicate the median values from 10 attempts. When random rotation is applied to all three axes of rotation, Prior methods start to underperform with more rotation. Our method, in contrast, outperforms others even after large amount of rotation is applied. This demonstrates that our representation is aligned in the canonical space, and is only slightly perturbed by sampling noise caused by the rotation. Augmenting training data by applying random rotation only slightly improves the results of prior methods, as there are endless possible

		Intersection over Union (IoU)															
	mean	air	bag	cap	car	chair	ear.	guit.	kni.	lam.	lap.	mot.	mug	pist.	rock.	ska.	tab.
P [27]	83.7	83.4	78.7	82.5	74.9	89.6	73.0	91.5	85.9	80.8	95.3	65.2	93.0	81.2	57.9	72.8	80.6
P++ [29]	85.1	82.4	79.0	87.7	77.3	90.8	71.8	91.0	85.9	83.7	95.3	71.6	94.1	81.3	58.7	76.4	82.6
Kd [15]	82.3	80.1	74.6	74.3	70.3	88.6	73.5	90.2	87.2	81.0	94.9	57.4	86.7	78.1	51.8	69.9	80.3
SO [18]	84.9	82.8	77.8	88.0	77.3	90.6	73.5	90.7	83.9	82.8	94.8	69.1	94.2	80.9	53.1	72.9	83.0
Ours	**85.2**	84.0	80.4	88.0	80.2	90.7	77.5	91.2	86.4	82.6	95.5	70.0	93.9	84.1	55.6	75.6	82.1

Table 4: ShapeNetPart segmentation results from methods using PointNet based architecture.

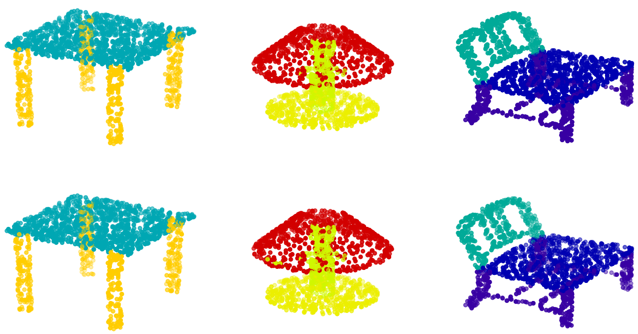


Figure 8: Visualization of segmentation results using our representation. Top: Ground truth. Bottom: Our results.

poses that shapes can take. This fact suggest much more data augmentation would be required for other point-based methods to handle all the possible rotational perturbations. The last column compares the classification accuracy after rotational and scale perturbation. Rotation with maximal angle of $\pi/4$ and scaling of 0.5 is applied to all the point cloud coordinates. The results from other methods further degrade, while our method maintains an accuracy of 83.9%.

4.5. Point Cloud Segmentation

We also verify the descriptiveness of the proposed representation through segmentation of point cloud data based on per-point labels. We use the ShapeNetCore Part dataset for the experiment. The dataset consists of 12145 training data and 2873 test data, consisting of 16 classes and 50 labeled parts. For fair comparison, we follow the protocol indicated in prior works, such as PointNet++ [29] and SO-Net [18].

We place the sampling sphere on top of each point in the same manner as in classification process. The parameter from the ELM is used as the local representation of the corresponding point. As shown in Fig. 6, we concatenate the local ELM weights and the intermediate global feature obtained from pooling the output of the first section of the segmentation network. We also use the normals and the original point coordinates along with the concatenated representation and to classify each point. The setting for factors, such as the number of sampling points, ELM dimension, and sampling sphere radius, are inherited from the classification experiment: 1024, 256, and 0.3, respectively.

The segmentation results are visualized in Fig. 8. We compare the results from our representation with state-of-the-art approaches that employ PointNet based network architecture in Table 4. We followed the prior works and used the point intersection over union (IoU) to compare the results. The segmentation result using our representation achieves the best mean IoU and outperforms the state-of-the-art methods in most of the classes. Compared to the results from the SO-Net, our method performs better in 12 of the total 16 categories. As our proposal can be used as a feature along with point coordinates, the representation can be directly inserted into more complex networks [19, 22].

5. Conclusion

The proposal is inspired by the observation that if neural networks are capable of learning extremely complex functions that separate various classes accurately, they should also be able to learn a function that represents one shape. The results demonstrate that our proposal is more descriptive, yet more efficient to process than the prior methods. No end-to-end training is conducted, and only unsupervised learning to encapsulate local distance fields is required. Our method is useful for achieving various invariances that made representation of unstructured point complex.

Our method derives from the belief that shapes should be preprocessed into a unified parametric space rather than trying to manually prepare all possible variations of the shapes through data augmentation. As was demonstrated in the experiment with coordinate and scaling perturbations, many of the current methods fail when the axis of gravitational direction is unknown. Methods proposed to conduct point cloud analysis need to consider various circumstances where the orientation of the point cloud is unavailable.

The results presented in this paper can theoretically be improved, as the random fixed weights used in the embedding ELMs are not trained to improve the accuracy of the following tasks, such as classification. As future work, we will pursue a method to tune the fixed ELM weights. Finding efficient methods to encapsulate more detailed information is also another direction of research.

References

[1] Amir Arsalan Soltani, Haibin Huang, Jiajun Wu, Tejas D Kulkarni, and Joshua B Tenenbaum. Synthesizing 3d shapes via modeling multi-view depth maps and silhouettes with deep generative networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1511–1519, 2017.

[2] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

[3] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5939–5948, 2019.

[4] David M Cole and Paul M Newman. Using laser range data for 3d slam in outdoor environments. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 1556–1563. IEEE, 2006.

[5] Haowen Deng, Tolga Birdal, and Slobodan Ilic. Ppf-foldnet: Unsupervised learning of rotation invariant 3d local descriptors. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 602–618, 2018.

[6] Haowen Deng, Tolga Birdal, and Slobodan Ilic. Ppfnet: Global context aware local features for robust 3d point matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 195–205, 2018.

[7] Martin Engelcke, Dushyant Rao, Dominic Zeng Wang, Chi Hay Tong, and Ingmar Posner. Vote3deep: Fast object detection in 3d point clouds using efficient convolutional neural networks. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1355–1361, 2017.

[8] Yifan Feng, Haoxuan You, Zizhao Zhang, Rongrong Ji, and Yue Gao. Hypergraph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3558–3565, 2019.

[9] Kent Fujiwara, Ikuro Sato, Mitsuru Ambai, Yuichi Yoshida, and Yoshiaki Sakakura. Canonical and compact point cloud representation for shape classification. *arXiv preprint arXiv:1809.04820*, 2018.

[10] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[11] Zhizhong Han, Mingyang Shang, Yuhang Liu, and Matthias Zwicker. View inter-prediction gan: Unsupervised representation learning for 3d shapes by learning global shape memories to support local view predictions. In *AAAI conference on Artificial Intelligence*, 2019.

[12] Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew. Extreme learning machine: theory and applications. *Neurocomputing*, 70(1):489–501, 2006.

[13] Asako Kanezaki, Yasuyuki Matsushita, and Yoshifumi Nishida. Rotationnet: Joint object categorization and pose estimation using multiviews from unsupervised viewpoints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5010–5019, 2018.

[14] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *Proceedings of the Second International Conference on Learning Representations (ICLR 2014)*, Apr. 2014.

[15] 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.

[16] Sudhakar Kumawat and Shanmuganathan Raman. Lp-3dcnn: Unveiling local phase in 3d convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4903–4912, 2019.

[17] Huan Lei, Naveed Akhtar, and Ajmal Mian. Octree guided cnn with spherical kernels for 3d point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9631–9640, 2019.

[18] 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.

[19] Jinxian Liu, Bingbing Ni, Caiyuan Li, Jiancheng Yang, and Qi Tian. Dynamic points agglomeration for hierarchical point sets learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7546–7555, 2019.

[20] Xinhai Liu, Zhizhong Han, Yu-Shen Liu, and Matthias Zwicker. Point2sequence: Learning the shape representation of 3d point clouds with an attention-based sequence to sequence network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 8778–8785, 2019.

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

[22] Jiageng Mao, Xiaogang Wang, and Hongsheng Li. Interpolated convolutional networks for 3d point cloud understanding. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1578–1587, 2019.

[23] Daniel Maturana and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 922–928. IEEE, 2015.

[24] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4460–4470, 2019.

[25] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

[26] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 165–174, 2019.

[27] 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.

[28] Charles R Qi, Hao Su, Matthias Nießner, Angela Dai, Mengyuan Yan, and Leonidas J Guibas. Volumetric and multi-view cnns for object classification on 3d data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5648–5656, 2016.

[29] 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 5105–5114, 2017.

[30] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. Octnet: Learning deep 3d representations at high resolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3577–3586, 2017.

[31] Radu Bogdan Rusu, Zoltan Csaba Marton, Nico Blodow, Mihai Dolha, and Michael Beetz. Towards 3d point cloud based object maps for household environments. *Robotics and Autonomous Systems*, 56(11):927–941, 2008.

[32] Baoguang Shi, Song Bai, Zhichao Zhou, and Xiang Bai. Deeppano: Deep panoramic representation for 3-d shape recognition. *IEEE Signal Processing Letters*, 22(12):2339–2343, 2015.

[33] 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.

[34] Ayan Sinha, Jing Bai, and Karthik Ramani. Deep learning 3d shape surfaces using geometry images. In *European Conference on Computer Vision*, pages 223–240, 2016.

[35] Shuran Song, Samuel P Lichtenberg, and Jianxiong Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 567–576, 2015.

[36] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 945–953, 2015.

[37] Gary KL Tam, Zhi-Quan Cheng, Yu-Kun Lai, Frank C Langbein, Yonghuai Liu, David Marshall, Ralph R Martin, Xian-Fang Sun, and Paul L Rosin. Registration of 3d point clouds and meshes: a survey from rigid to nonrigid. *IEEE transactions on visualization and computer graphics*, 19(7):1199–1217, 2013.

[38] Jiexiong Tang, Chenwei Deng, and Guang-Bin Huang. Extreme learning machine for multilayer perceptron. *IEEE transactions on neural networks and learning systems*, 27(4):809–821, 2016.

[39] Johan WH Tangelder and Remco C Veltkamp. A survey of content based 3d shape retrieval methods. In *Shape Modeling Applications, 2004*, pages 145–156, 2004.

[40] Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6411–6420, 2019.

[41] Oliver Van Kaick, Hao Zhang, Ghassan Hamarneh, and Daniel Cohen-Or. A survey on shape correspondence. In *Computer Graphics Forum*, volume 30, pages 1681–1707. Wiley Online Library, 2011.

[42] Chu Wang, Marcello Pelillo, and Kaleem Siddiqi. Dominant set clustering and pooling for multi-view 3d object recognition. In *Proceedings of the British machine vision conference*, 2017.

[43] Peng-Shuai Wang, Yang Liu, Yu-Xiao Guo, Chun-Yu Sun, and Xin Tong. O-cnn: Octree-based convolutional neural networks for 3d shape analysis. *ACM Transactions on Graphics (TOG)*, 36(4):72, 2017.

[44] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics (TOG)*, 38(5):1–12, 2019.

[45] Jiajun Wu, Chengkai Zhang, Tianfan Xue, Bill Freeman, and Josh Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *Advances in Neural Information Processing Systems*, pages 82–90, 2016.

[46] Wenxuan Wu, Zhongang Qi, and Li Fuxin. Pointconv: Deep convolutional networks on 3d point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9621–9630, 2019.

[47] 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.

[48] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan R Salakhutdinov, and Alexander J Smola. Deep sets. In *Advances in Neural Information Processing Systems*, pages 3394–3404, 2017.

[49] Zhiyuan Zhang, Binh-Son Hua, and Sai-Kit Yeung. Shellnet: Efficient point cloud convolutional neural networks using concentric shells statistics. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1607–1616, 2019.

[50] Hengshuang Zhao, Li Jiang, Chi-Wing Fu, and Jiaya Jia. Pointweb: Enhancing local neighborhood features for point cloud processing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5565–5573, 2019.