Pattern Recognition 139 (2023) 109476

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/patcog

TreeNet: Structure preserving multi-class 3D point cloud completion

Long Xi [a], Wen Tang [a,*], TaoRuan Wan [b]

[a] Department of Creative Technology, Bournemouth University, Poole, BH12 5BB, UK
[b] Department of Informatics, University of Bradford, Bradford, BD7 1DP, UK

ARTICLE INFO

Article history:
Received 15 June 2021
Revised 18 January 2023
Accepted 24 February 2023
Available online 28 February 2023

Keywords:
3D Point cloud completion
Multi-class training
Hierarchical tree
Computer vision
Artificial intelligence
Deep learning

ABSTRACT

Generating the missing data of 3D object point clouds from partial observations is a challenging task. Existing state-of-the-art learning-based 3D point cloud completion methods tend to use a limited number of categories/classes of training data and regenerate the entire point cloud based on the training datasets. As a result, output 3D point clouds generated by such methods may lose details (i.e. sharp edges and topology changes) due to the lack of multi-class training. These methods also lose the structural and spatial details of partial inputs due to the models do not separate the reconstructed partial input from missing points in the output.

In this paper, we propose a novel deep learning network - *TreeNet* for 3D point cloud completion. TreeNet has two networks in hierarchical tree-based structures: TreeNet-multiclass focuses on multi-class training with a specific class of the completion task on each sub-tree to improve the quality of point cloud output; TreeNet-binary focuses on generating points in missing areas and fully preserving the original partial input. TreeNet-multiclass and TreeNet-binary are both network decoders and can be trained independently. TreeNet decoder is the combination of TreeNet-multiclass and TreeNet-binary and is trained with an encoder from existing methods (i.e. PointNet encoder). We compare the proposed TreeNet with five state-of-the-art learning-based methods on fifty classes of the public Shapenet dataset and unknown classes, which shows that TreeNet provides a significant improvement in the overall quality and exhibits strong generalization to unknown classes that are not trained.

1. Introduction

3D point clouds, captured by various sensor technologies such as laser and RGB-D scanners and depth cameras, suffer from large missing data due to complicated occlusions, unreliable measurements, limited viewing angles and the resolution of various sensors in dealing with texture-less regions of the scene. Therefore, generating a complete 3D point cloud (i.e. 3D point cloud completion) from a captured incomplete point cloud is an essential task for a wide range of 3D vision applications from augmented reality [1], robotics [2,3] to navigation and scene understanding [4,5]. Some previous learning-based methods [6,7] convert 3D datasets into structured and ordered 3D voxel grids for network training. However, these methods consume high computer memories, and output qualities are limited by voxel grid resolutions. Recently proposed PointNet [8] processes 3D point cloud dataset directly using multi-layer perception (MLP) models without voxelization. PointNet has been adapted to many computer vision applications such as 3D point cloud classification [8,8,10], object recognition [11], object detection [12,13], object segmentation [14,15] and registration tasks [16–18].

For 3D point cloud completion, a recent deep learning-based approach [19] adopts a PointNet-based encoder [8] to extract 3D point cloud representation for a single class (e.g. airplane objects) of the 3D point cloud completion task. FoldingNet [20] uses a PointNet-based encoder and proposes a folding operation in the decoder that deforms a 2D grid into a 3D point cloud. PCN [21] combines features from the intermediate and the final layer in the PointNet encoder and adopts the folding operation in a decoder. The folding operation, however, specifically enforces the decoder to generate a specific mapping of the input data, which constrains the learning process by limiting the solution search space. To avoid using the folding operation in a decoder, TopNet [22] follows the structure of PointNet [8], using MLP models to extract features in intermediate layers and generating

* Corresponding author.
E-mail address: wtang@bournemouth.ac.uk (W. Tang).

https://doi.org/10.1016/j.patcog.2023.109476

the entire 3D point cloud as the output. PMPNet [23] moves each point in the partial input to complete the point cloud instead of directly generating the complete 3D point clouds. Disp3d [24] mismatches the features between different classes, which results in poor generalization ability on unknown classes. There are two shared problems with these methods for 3D point cloud completion. First, these methods are sensitive to the geometric forms and shapes of training datasets of 3D point clouds. Therefore, they cannot handle multi-classes effectively, even a single class that contains mostly different shapes. Training a class-invariant model with multiple classes of training datasets is difficult with these methods. When the number of classes increases in the training dataset, these methods produce low-quality 3D point cloud outputs that lack structural and spatial details, such as sharp edges and topology changes. Second, these methods lose original structural and spatial details in the final output due to the fact that they regenerate the entire 3D point cloud and do not separate the reconstructed partial input from the missing points in the final output.

Here, we make an important observation that *the partial input should be fully preserved rather than regenerated*. Based on this original idea, in this paper, we propose a novel deep learning-based network by devising hierarchical tree-based decoders to build a learning network-TreeNet for 3D point cloud completion. TreeNet combines two networks. One (TreeNet-multiclass) is for multi-class training and the other (TreeNet-binary) is for missing points generation with original structure-preserving. More specifically, the TreeNet-multiclass decoder assigns each class of a completion task to a specific sub-tree of the root node in the tree. The root node of the tree is a global feature vector generated from a PointNet [8] based encoder. The number of sub-trees to the root node is designed to be identical to the number of classes in the training data. Each sub-tree of the root node is designed to generate 3D point clouds from a specific class and maintains the quality of output for each class of data when the number of classes increases, as shown in Fig. 2. Most importantly, although each sub-tree is designed for a specific class during the training, this does not limit these sub-trees to be used for unknown classes once the model has been trained. (Refer to Section 4.5 for more details.) TreeNet-binary generates points in missing areas and fully preserves the structural and spatial details of partial input in the final output, as shown in Fig. 3. TreeNet-binary is a binary tree structure that has a left leaf node and a right leaf node. The left leaf node generates the reconstruction of the partial input 3D point cloud, and the right leaf node aims to generate points in missing areas. Once the model has been trained, the final output is the combination of the missing data generated from the right leaf node and the original partial input 3D point cloud. TreeNet-binary can be considered a sub-tree of the root node in TreeNet-multiclass, and also be an individual decoder trained separately with a PointNet-based encoder.

We propose three novel forward propagation methods to train our TreeNet-multiclass, TreeNet-binary and TreeNet, respectively. First, to train the TreeNet-multiclass for multi-class 3D point cloud completion, we propose an activation gate for the standard forward propagation to activate and deactivate sub-trees of the root node by assigning each class of a completion task to its corresponding activated sub-tree. Unlike the standard backward propagation, the gradient of the loss function in TreeNet-multiclass is only calculated on the activated sub-trees during each batch of the training. Second, the forward propagation for TreeNet-binary splits features of the root node to a binary tree structure where the left leaf node reconstructs the partial input and the right leaf node generates points in missing areas. Third, TreeNet is trained using the combined forward propagation methods of TreeNet-multiclass and TreeNet-binary.

The main contributions of this paper are:

- A novel TreeNet-multiclass decoder is proposed for multi-class 3D point cloud completion. We evaluate our model on 50 classes of training datasets, whereas the majority of the state-of-the-art methods only use 8 classes.
- A novel TreeNet-binary decoder is proposed, which focuses on generating points in missing areas and fully preserving the original partial input 3D point cloud.
- A novel TreeNet decoder is proposed, which combines the advantages of the TreeNet-multiclass and the TreeNet-binary for 3D point cloud completion.
- Three novel forward and backward propagation methods are proposed to train TreeNet-multiclass, TreeNet-binary and TreeNet decoders, respectively.
- TreeNet-multiclass, TreeNet-binary and TreeNet exhibit strong generalization to unknown classes that are never trained.

2. Related work

Research on 3D shape completion can be categorized into three classes of approaches: geometry-based, data-driven based and learning-based methods. Geometry-based approaches [25,26] complete 3D shapes by using the symmetric information from the partial input, while the data-driven based method [27] relies on the assumption that the database must include very similar shapes. Our work belongs to learning-based methods that can be further categorized according to the forms of the input (i.e. 3D voxel grids or 3D point clouds).

Volumetric 3D Shape Completion: Currently, one major promising progress for the 3D shape completion task is utilizing 3D learning-based neural networks that are successful at learning 3D data representations and features automatically, reducing the incompleteness caused by designing features manually. 3D learning-based architecture largely depends on the representation of the 3D data, such as volumetric voxel grids or 3D point clouds. Since convolutional neural networks can process structured and ordered 3D datasets more effectively than unstructured datasets, most previous 3D learning-based methods [6,7] used voxelized representations for 3D shape completions. However, voxelization causes high computational costs when the resolution increases dramatically, and low resolution results in low-quality output [28].

3D Point Cloud Completion: PointNet [8] and PointNet++ [9] operate directly on 3D point clouds without voxelization, and several state-of-the-art approaches based on PointNet [8] are proposed for 3D point cloud completion. Achlioptas et al. [19] introduce an Auto Encoder [29] and a Generative Adversarial Net (GAN) [30] to learn 3D point cloud representations by focusing on a single class of 3D point cloud completion task. FoldingNet [20] proposes a folding operation in the decoder that deforms a 2D grid into a 3D point cloud and evaluates the different layers of the folding operations to tune the model. PCN [21] evaluates the different number of PointNet layers and fully connected layers in the encoder and decoder to achieve the best performance. PCN also uses a folding operation to generate higher-resolution 3D point clouds from the coarse 3D point clouds in the final stage of the decoder. TopNet [22] first evaluates the encoders in PointNet [8], PointNet++ [9] and PCN [21] and finally chooses the encoder from PCN [21]. TopNet [22] then proposes a decoder following tree structure and evaluates the number of MLP layers with different feature sizes in the decoder to achieve the highest accuracy in the 8 classes of datasets. Specifically, the root node in TopNet [22] is the global feature of the input data. The output of each leaf node represents *a single point* in a 3D point cloud, and all leaf nodes consist of a complete 3D point cloud. Therefore, features of a partial input in the tree root pass through

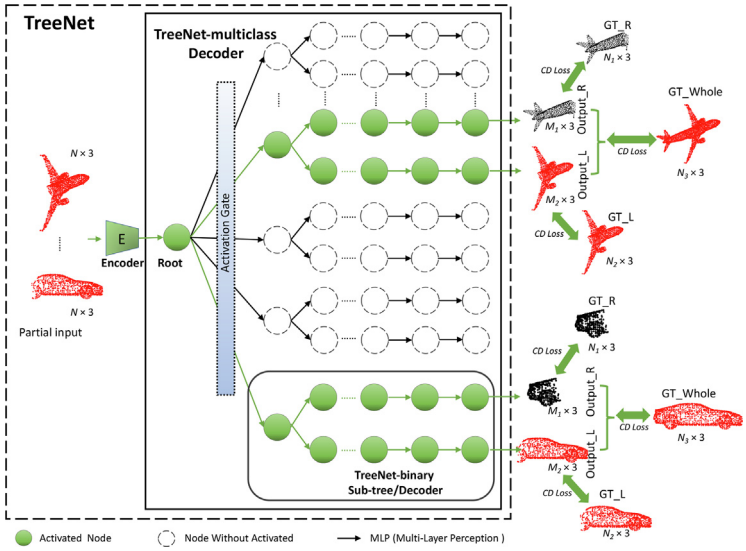


Fig. 1. TreeNet architecture. TreeNet combines TreeNet-multiclass and TreeNet-binary.

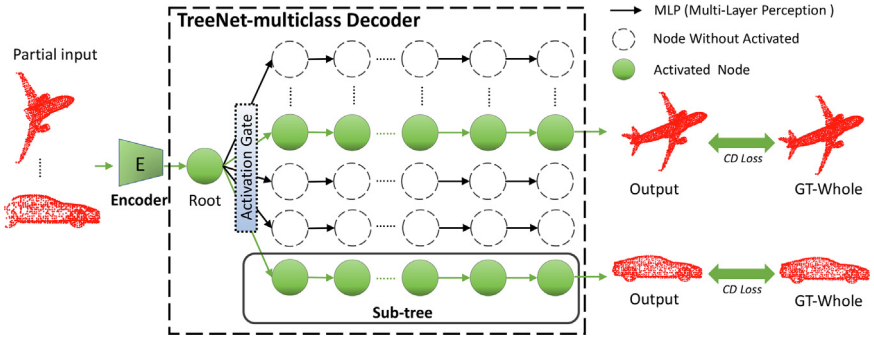


Fig. 2. TreeNet-multiclass architecture. Each sub-tree is designed to focus on a specific class of 3D point cloud completion tasks during the training. Once trained, these sub-trees can be used for unknown classes. (Refer to Section 4.5 for more details about unknown classes.)

all nodes to regenerate an entire 3D point cloud. The architecture of the TopNet [22] is not designed for multi-class 3D point cloud completion, and it loses the structural and spatial details of partial inputs. In contrast, our TreeNet focuses on multi-class training and missing points generation with original structural preservation, and the architecture of our proposed TreeNet is completely different to the tree in TopNet [22]. PMPNet [23] applies PointNet++ [9] encoder and generates translation matrices for each point in the partial input in the decoder, which translates the incomplete input to the nearest occluded regions. PMPNet [23] also analyses the different recurrent units and the different searching radii in the proposed recurrent path aggregation module to tune the decoder. Disp3d [24] proposes a down-sampling operation, a neighbour pooling and an up-sampling operation to generate complete 3D point clouds.

3. Methods

Given a partial 3D point cloud input with N points where each point is defined as $P_i = (x, y, z)$, our novel tree-based decoder-TreeNet generates M missing points. Our TreeNet decoder, as shown in Fig. 1, contains TreeNet-multiclass and TreeNet-binary. TreeNet-multiclass (Section 3.1) is for multi-class 3D point cloud completion, as shown in Fig. 2. TreeNet-binary (Section 3.2) generates points in missing areas and preserves the original partial input, as shown in Fig. 3. Currently, we use TreeNet-binary as the sub-tree of the root node in TreeNet-multiclass and achieve our final output of TreeNet. However, this does not limit the TreeNet-multiclass and TreeNet-binary to be used as individual decoders. TreeNet-multiclass, TreeNet-binary and TreeNet are trained using their own forward propagation methods (Section 3.3) and a combined Chamfer distance [31] as loss (Section 3.4) and within an auto-encoder framework that includes a PointNet [8] based encoder as a first stage.

3.1. Multi-class point cloud completion

To train a class-invariant model for multi-class 3D point cloud completion, TreeNet-multiclass assigns each class of the completion task to a specific sub-tree of its root node, where each of the sub-tree is identical to others and designed to focus on generating 3D point clouds from a specific class. To this end, The number of branches of the root node is identical to the number of classes in the training data. The architecture of our TreeNet-multiclass is shown in Fig. 2. The root node contains common features of partial inputs generated from a PointNet-based encoder and each sub-tree uses these common features that are passed from the root node. The MLP is used for generating features at each level of the tree, and the nodes at the same level of the tree are processed by the same shared MLP. The root node is connected with D sub-trees that correspond to D classes of data. To assign D classes of completion tasks to D corresponding sub-trees of the root, an activation gate (Eq. (2)) is proposed for the forward propagation to activate and deactivate the sub-trees in our decoder, as illustrated in Section 3.3.1. The feature only passes through its corresponding activated sub-tree, and other sub-trees will be temporarily deacti-

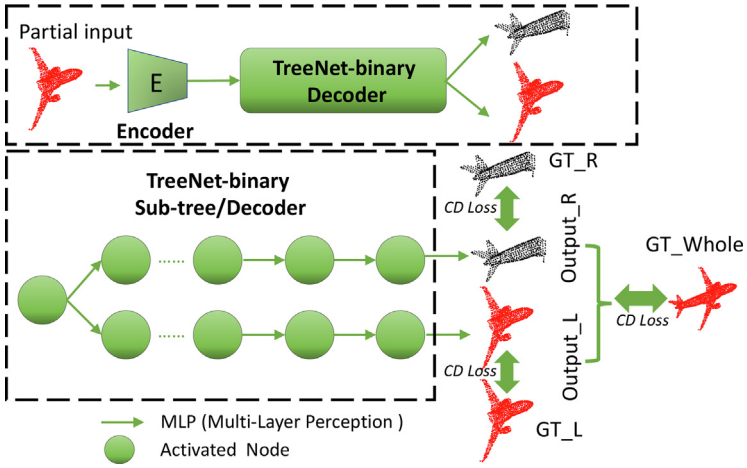


Fig. 3. TreeNet-binary architecture. The output 3D point cloud is divided into two segments (Output_L and Output_R). Output_L is the reconstruction of the partial input, and Output_R represents points in missing areas. Output_L and Output_R are used to calculate the combined loss during training.

vated. The deactivated sub-trees will be activated again when the corresponding features of input data pass through. The leaf node in each sub-tree represents a complete 3D point cloud. TreeNet-multiclass is trained by using our novel forward and backward method described in Section 3.3 and the Chamfer distance loss as shown in Eq. 5.

Since the output of our TreeNet-multiclass decoder is a complete 3D point cloud by regenerating all points in the output, we further design our tree-based decoder to generate points in missing areas and preserve the original partial input structure (Section 3.2).

3.2. Missing points generation with original structure preserving

To generate points in missing areas and fully preserve the structure of the original partial input, we further propose a novel TreeNet-binary decoder that follows a binary tree structure. The architecture of our TreeNet-binary is shown in Fig. 3. The left leaf node produces the reconstruction of the partial input, and the right leaf node generates points in missing areas. The outputs from the left and the right leaf nodes are used to calculate the combined loss during training, as shown in Eq. (6) in Section 3.4. Once trained, the final output 3D point cloud is the combination of the output of the right leaf node and the partial input. Therefore, the original partial input is fully preserved.

Most importantly, there are two uses for our proposed TreeNet-binary. First, the TreeNet-binary can be considered an individual decoder and trained with the novel forward propagation, as illustrated in Eq. (3) in Section 3.3.1. Second, we use the TreeNet-binary as the sub-tree of the root node in TreeNet-multiclass and achieve our final TreeNet, as shown in Fig. 1.

3.3. Forward and backward propagation in tree-based decoder

In this section, we illustrate the forward and backward propagation methods for training TreeNet-multiclass, TreeNet-binary and TreeNet, respectively.

3.3.1. Forward propagation

Based on the design of our TreeNet-multiclass, we propose an activation gate to activate and deactivate sub-trees of the root node for our novel forward propagation. The features pass through all neurons in standard forward propagation (Eq. (1)), whereas the features only pass through their corresponding activated sub-tree in our proposed forward propagation for TreeNet-multiclass (Eq. (2)).

$$z_i^{(l+1)} = \boldsymbol{w}_i^{(l+1)} \boldsymbol{y}_i^{(l)} + \boldsymbol{b}_i^{(l+1)}; \qquad \boldsymbol{y}_i^{(l+1)} = f(z_i^{(l+1)}) \tag{1}$$

where i indexes the hidden neuron in each layer and l indexes the hidden layer. $\boldsymbol{w}_i^{(l+1)}$ and $\boldsymbol{b}_i^{(l+1)}$ denote the ith weight and bias at layer $l+1$. $\boldsymbol{y}_i^{(l)}$ is the ith features of inputs at the layer l. $z_i^{(l+1)}$ is the ith feature of inputs at the layer $l+1$. $f(\cdot)$ is any activation function, e.g. Tanh.

Given D classes of training data, the number of sub-trees in our TreeNet-multiclass is D. Let $d \in \{0, 1, \ldots, D\}$ and defines the dth sub-tree from the left to the right. Assuming the feature size of the root node is reshaped to $[B, D, m]$, where B is the batch size and $[D, m]$ is the size of a feature vector F from a partial input 3D point cloud. Thus, there are B feature vectors in the root node preparing to pass through the tree. The purpose is to assign B feature vectors to their corresponding sub-trees during each batch of training. An activation gate (Eq. (2)) activates and deactivates sub-trees of the root node, and the size of the activation gate is $[B, D, m]$ which is as same as the feature size of the root node. Thus, there are D inner gates in the activation gate, where each inner gate g_d is a vector with the size of $m \times 1$ and all values in each g_d are the same, either all 0 or all 1. Each inner gate corresponds to a sub-tree and decides whether the corresponding sub-tree is activated or deactivated. For the sub-trees with the corresponding features F coming through, all values in g_d become 1, and sub-trees without the corresponding F values in g_d become 0. *Activation_Gate* is an element-wise product with B feature vectors in the root node to activate and deactivate sub-trees. Thus, the forward propagation for TreeNet-multiclass is defined by Eq. (2).

$$\begin{aligned} &Activation\_Gate = [g_1, g_2, g_3, \ldots, g_D] \\ &\tilde{\boldsymbol{y}}_{subtree\_Root} = Activation\_Gate * \boldsymbol{y}_{Root} \\ &z_i^{d(l+1)} = \boldsymbol{w}_i^{d(l+1)} \tilde{\boldsymbol{y}}_{subtree\_Root^i}^{d(l)} + \boldsymbol{b}_i^{d(l+1)} \quad (d \in \{0, 1, \ldots, D\}) \\ &\boldsymbol{y}_i^{d(l+1)} = f(z_i^{d(l+1)}) \end{aligned} \tag{2}$$

where $*$ denotes an element-wise product. $\boldsymbol{y}_{Root}$ denotes the features in the root node of the tree. $\tilde{\boldsymbol{y}}_{subtree\_Root}$ denotes features in the root of each sub-tree. d defines the dth sub-tree from the left to the right. l indexes the hidden layer and i indexes the hidden neuron in each layer. $\tilde{\boldsymbol{y}}_{subtree\_Root^i}^{d(l)}$ denotes the ith activated or deactivate neuron in dth sub-tree at the layer l. $\boldsymbol{w}_i^{d(l+1)}$ and $\boldsymbol{b}_i^{d(l+1)}$ denote the ith weight and bias in dth sub-tree at layer $l+1$. $z_i^{d(l+1)}$ denotes the ith feature vector in dth sub-tree at the layer $l+1$.

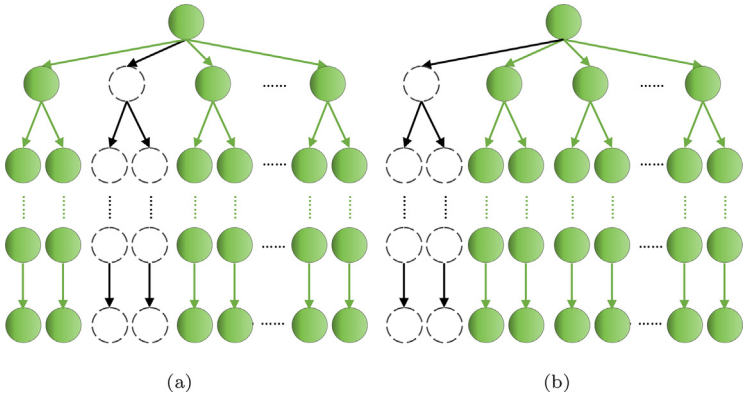


Fig. 4. Forward propagation. The forward propagation only operates on the activated nodes in each batch of training. (a) One possible situation during a batch of training. (b) Another possible situation of another batch of training.

$\boldsymbol{y}_i^{d(l+1)}$ denotes the ith feature vector in dth sub-tree at the layer $l+1$. $f(\cdot)$ is a Tanh activation function.

The forward propagation for TreeNet-binary, as defined by Eq. (3), is proposed to train TreeNet-binary.

$$\begin{aligned}
&\boldsymbol{y}_{RST} = \boldsymbol{y}_{Root_{idx}} \quad (idx = 0, \ldots, m-1)\\
&\boldsymbol{y}_{LST} = \boldsymbol{y}_{Root_{idx}} \quad (idx = m, \ldots, 2m)\\
&z_{RST^{(i)}}^{(l+1)} = \boldsymbol{w}_{RST^{(i)}}^{(l+1)} \boldsymbol{y}_{RST^{(i)}}^{(l)} + \boldsymbol{b}_{RST^{(i)}}^{(l+1)}; \quad \boldsymbol{y}_{RST^{(i)}}^{(l+1)} = f(z_{RST^{(i)}}^{(l+1)})\\
&z_{LST^{(i)}}^{(l+1)} = \boldsymbol{w}_{LST^{(i)}}^{(l+1)} \boldsymbol{y}_{LST^{(i)}}^{(l)} + \boldsymbol{b}_{LST^{(i)}}^{(l+1)}; \quad \boldsymbol{y}_{LST^{(i)}}^{(l+1)} = f(z_{LST^{(i)}}^{(l+1)})
\end{aligned} \tag{3}$$

where $\boldsymbol{y}_{Root_{idx}}$ denotes the feature vector in the root node of the tree. $\boldsymbol{y}_{RST}$ and $\boldsymbol{y}_{LST}$ denote features for the right and left sub-trees, respectively. m is the feature size of the left and right sub-trees. $\boldsymbol{y}_{RST^{(i)}}^{(l)}$ and $\boldsymbol{y}_{LST^{(i)}}^{(l)}$ denote the ith neuron in the right and left sub-trees at the layer l, respectively. $\boldsymbol{w}_{RST^{(i)}}^{(l+1)}$ and $\boldsymbol{b}_{RST^{(i)}}^{(l+1)}$ denote the ith weight and bias for right sub-tree at layer $l+1$. $\boldsymbol{w}_{LST^{(i)}}^{(l+1)}$ and $\boldsymbol{b}_{LST^{(i)}}^{(l+1)}$ denote the ith weight and bias for left sub-tree at layer $l+1$. $z_{RST^{(i)}}^{(l+1)}$ and $z_{LST^{(i)}}^{(l+1)}$ denote the ith feature vector in right and left sub-trees at layer $l+1$, respectively. $\boldsymbol{y}_{RST^{(i)}}^{(l+1)}$ and $\boldsymbol{y}_{LST^{(i)}}^{(l+1)}$ denote the ith feature vector after an activation function $f(\cdot)$ in right and left sub-trees at layer $l+1$, respectively.

The TreeNet combines TreeNet-multiclass and TreeNet-binary and is trained using the combined forward propagation of TreeNet-multiclass and TreeNet-binary, as defined by Eq. (4).

$$\begin{aligned}
&Activation_Gate = [g_1, g_2, g_3, \ldots, g_D]\\
&\tilde{\boldsymbol{y}}_{subtree_Root} = Activation_Gate * \boldsymbol{y}_{Root}\\
&\tilde{\boldsymbol{y}}_{RST}^{d(l)} = \tilde{\boldsymbol{y}}_{subtree_Root_{idx}}^{d(l)} \quad (d \in \{0, 1, \ldots, D\})\\
&\qquad (idx = 2d \times m, \ldots, (2d+1) \times m - 1);\\
&\tilde{\boldsymbol{y}}_{LST}^{d(l)} = \tilde{\boldsymbol{y}}_{subtree_Root_{idx}}^{d(l)} \quad (d \in \{0, 1, \ldots, D\})\\
&\qquad (idx = (2d+1) \times m, \ldots, (d+1) \times 2m - 1)\\
&z_{RST^{(i)}}^{d(l+1)} = \boldsymbol{w}_{RST^{(i)}}^{d(l+1)} \boldsymbol{y}_{RST^{(i)}}^{d(l)} + \boldsymbol{b}_{RST^{(i)}}^{d(l+1)}; \quad \boldsymbol{y}_{RST^{(i)}}^{d(l+1)} = f(z_{RST^{(i)}}^{d(l+1)})\\
&z_{LST^{(i)}}^{d(l+1)} = \boldsymbol{w}_{LST^{(i)}}^{d(l+1)} \boldsymbol{y}_{LST^{(i)}}^{d(l)} + \boldsymbol{b}_{LST^{(i)}}^{d(l+1)}; \quad \boldsymbol{y}_{LST^{(i)}}^{d(l+1)} = f(z_{LST^{(i)}}^{d(l+1)})
\end{aligned} \tag{4}$$

where $\tilde{\boldsymbol{y}}_{RST}^{d(l)}$ and $\tilde{\boldsymbol{y}}_{LST}^{d(l)}$ denote the dth right and left sub-trees at layer l, respectively. m is the feature size of each sub-tree root node and is also the size of each inner gate g_d. $\boldsymbol{w}_{RST^{(i)}}^{d(l+1)}$ and $\boldsymbol{b}_{RST^{(i)}}^{d(l+1)}$ are the ith weight and bias in dth right sub-tree at layer $l+1$. $\boldsymbol{w}_{LST^{(i)}}^{d(l+1)}$ and $\boldsymbol{b}_{LST^{(i)}}^{d(l+1)}$ are the ith weight and bias in dth left sub-tree at layer $l+1$. $\boldsymbol{y}_{RST^{(i)}}^{d(l+1)}$ and $\boldsymbol{y}_{LST^{(i)}}^{d(l+1)}$ denote the ith feature vector after an activation function $f(\cdot)$ in dth right and left sub-trees at layer $l+1$, respectively.

Assuming there are T 3D point clouds in the training set, including D classes of datasets. The B training point clouds in each batch of the training are randomly selected from the training set, and the whole training set is trained with E epochs. Thus, there are $\frac{T}{B} \times E$ different situations of the forward propagation during the TreeNet training, and the values in the activation gate are automatically changed $\frac{T}{B} \times E$ times based on the class of the randomly selected data. We show two possible situations of the forward propagation for TreeNet during two different batches of training in Fig. 4. The second sub-tree from the left to the right is deactivated in (a) but activated in (b). The first sub-tree from the left is deactivated in (b) but activated in (a).

3.3.2. Backward propagation

Backward propagation (BP) is the major learning procedure that repeatedly adjusts the weights and biases of the connections in the network to minimize a measure of the difference between the output and the ground truth. Since the values of nodes in deactivated sub-trees are zero, the gradient of the loss function in our BP is only calculated on the activated sub-trees during each batch of training. Whereas the existing learning-based methods for 3D point cloud completion [19–24] need all neurons in the decoder to participate in backward propagation during training. Therefore, our TreeNet is more efficient and effective in training multi-classes of data. Similarly to the forward propagation, we show two possible situations of the backward propagation during two different batches of training in Fig. 5. The first sub-tree of the root node from the left to the right is deactivated in (a) but activated in (b). The second sub-tree of the root node from the left to the right is activated in (a) but deactivated in (b). The last sub-trees to the right are all deactivated in both (a) and (b).

3.4. Loss function

The loss function for 3D point cloud completion measures the difference between the output 3D point cloud S_{output} and the ground truth point cloud S_{gt}. The loss is defined to be invariant to any permutation of 3D point clouds in both S_{output} and S_{gt}. Similarly to the state-of-the-art methods [20–24], we also use the

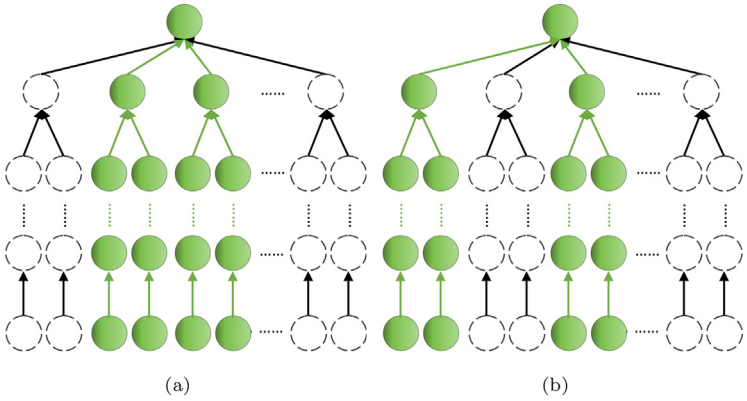


Fig. 5. Backward propagation. The Backward propagation only operates on the activated nodes in each batch of training. (a) One possible situation during a batch of training. (b) Another possible situation of another batch of training.

Chamfer distance (CD) [31] in the loss function.

$$CD(S_{output}, S_{gt}) = \frac{1}{S_{output}} \sum_{x \in S_{output}} \min_{y \in S_{gt}} ||x - y||_2 + \frac{1}{S_{gt}} \sum_{y \in S_{gt}} \min_{s \in S_{output}} ||y - x||_2 \quad (5)$$

Chamfer distance calculates the average nearest point distance between S_{output} and S_{gt} by finding the closest neighbour with $O(nlogn)$ complexity. In addition, S_{output} and S_{gt} can be the different sizes of 3D point clouds.

The loss function for TreeNet-multiclass is CD, as illustrated in Eq. (5). Based on the Chamfer distance, the final loss functions for TreeNet-binary and TreeNet are the same, as defined in Eq. (6).

$$\begin{aligned} Loss_1 &= CD(S_{Output\_L}, S_{GT\_L}) \\ Loss_2 &= CD(S_{Output\_R}, S_{GT\_R}) \\ Loss_3 &= CD(S_{Output\_L\&R}, S_{GT\_Whole}) \\ Loss &= (\lambda_1 \cdot Loss_1 + \lambda_2 \cdot Loss_2) + Loss_3 \end{aligned} \quad (6)$$

where $Output\_L$ is the output of the left leaf node, and GT_L is the ground truth corresponding to the $Output\_L$. $Output\_R$ is the output of the right leaf node, and GT_R is the ground truth corresponding to the $Output\_R$. $Output\_L\&R$ is the combination of the $Output\_L$ and $Output\_R$, and GT_Whole is the ground truth corresponding to the $Output\_L\&R$. λ is the corresponding weight to balance the effect of gradients of the backward propagation.

4. Experiments

Our tree-based decoder is trained within an auto-encoder framework that includes a PointNet-based encoder [21]. An Nvidia Geforce 2080Ti GPU with 12G memory is used for network training. We conduct three-stage experiments. First, we conduct a set of experiments to decide the final design of our TreeNet-multiclass, TreeNet-binary and TreeNet, including the effectiveness of the existing point cloud encoders, depth of the trees and values of λ in the loss function (Section 4.3). Second, we compare our TreeNet-multiclass, TreeNet-binary and TreeNet with the state-of-the-art methods [20–24] on testing datasets from trained classes (Section 4.4). Third, we evaluate the generalization ability of each network on unknown classes that are never trained (Section 4.5). Similar to the state-of-the-art methods [20–24], the Chamfer distance (Eq. (5)) is used as the evaluation metric to compare the output 3D point clouds with the corresponding ground truth.

4.1. Implementation details

We train TreeNet-multiclass, TreeNet-binary and TreeNet for 600 epochs with a batch size of 32, a learning rate of 0.005, and an Adagrad optimizer. Our TreeNet-binary and TreeNet have $L = 7$ levels for generating the output 3D point cloud with the size of 2048×3, and TreeNet-multiclass has $L = 3$ levels, as illustrated in Section 4.3.2. The weights defined in our total loss for TreeNet-binary and TreeNet are $\lambda_1 = 0.2$ and $\lambda_2 = 0.8$ (Section 4.3.2).

For TreeNet, the feature size for the tree root is 1024, and the feature size for each sub-tree root is 2048. The filter sizes for the left sub-tree in each level are $[1024, 1536, 2048, 2560, 1024 \times 3]$. The filter sizes for the right sub-tree in each level are also $[1024, 1536, 2048, 2560, 1024 \times 3]$. The class label of each partial input 3D point cloud is automatically saved and decides the values in $Activation\_Gate^{(l)}$ (Eqs. (2) and (4)) during the training.

As shown in Fig. 1, TreeNet is trained with the input size of $N \times 3$ and three ground truth sizes of $N_1 \times 3$, $N_2 \times 3$ and $N_3 \times 3$, generating the output 3D point cloud with the sizes of $M_1 \times 3$ and $M_2 \times 3$. Once trained, the size of the final output 3D point cloud is $(N + M_1) \times 3$. Note that N, N_1, N_2, N_3, M_1 and M_2 can be any number. In our experiment, we set $N = 1024$, $N_1 = 512$, $N_2 = 1024$, $N_3 = 2048$, $M_1 = 1024$ and $M_2 = 1024$.

4.2. Datasets

Training and Testing Datasets. For a fair comparison, we use the ShapeNetCore [32] dataset as the training and testing datasets following FoldingNet [20], PCN [21], TopNet [22], PMPNet [23] and Disp3d [24]. ShapeNetCore [32] is composed of 55 object classes with more than 50,000 3D point clouds, where each 3D point cloud contains 2048 points.

For the ShapeNetCore dataset of the 8 classes (29,774 point clouds), the data used for training is 97%, and the data used for testing is 3%. We followed the percentage of dataset splitting used in the recent state-of-the-art methods PCN [21], TopNet [22], PMPNet [23] and Disp3d [24], because, we have conducted comparative studies with the results of these methods. This dataset includes airplanes, cabinets, cars, chairs, lamps, sofas, tables and vessels.

In addition to the 8 classes, we randomly select the further 42 classes in ShapeNetCore and combine these with the 8 classes of ShapeNetCore. This dataset contains 50 classes (51,188 point clouds), the data used for training is 90%, and the data used for testing is 10%. The testing dataset is not used for training and only for testing. This dataset includes airplanes, lamps, mugs, bowls, caps, laptops, buses, pillows, etc.

Since all 3D point clouds in ShapeNetCore are complete 3D point clouds, we create partial 3D point clouds from all training

Table 1
Encoder analysis: Quantitative comparison of our approach against previous works using different encoders. E represents the encoder, and D is our TreeNet decoder. The encoders in TopNet [22] and PCN [21] are identical. The Chamfer distance is reported multiplied by (10^3).

Methods	CD
PointNet [8](E) + TreeNet(D)	1.002
PointNet+ [9](E) + TreeNet(D)	7.284
AE [19](E) + TreeNet(D)	1.267
PCN [21]/TopNet [22](E) + TreeNet(D)	**0.817**

Table 2
Quantitative comparison between different levels of our networks tested on 8 classes of testing data. The Chamfer distance is reported multiplied by (10^3).

Level	3	4	5	6	7	8
TreeNet-multiclass	**1.31**	1.37	1.38	1.37	1.47	1.45
TreeNet-bianry	1.999	1.098	0.990	0.989	**0.919**	0.957
TreeNet	1.065	0.853	0.853	0.853	**0.817**	0.828

and testing sets. During each epoch of training, the partial 3D point clouds are created from the training set with missing rates from 20% to 50%. During the testing, the partial inputs are created from the testing set with missing rates from 20% to 50%. The removed areas are selected from k nearest points around a randomly selected point.

Testing Dataset from Unknown Classes. To evaluate the robustness and generalization ability of a network, we use the remaining five classes of the ShapeNetCore dataset as the unknown classes for the evaluation, including cameras, baskets, stoves, towers and printers. These five unknown classes consist of 843 3D point clouds that are not trained and strange to all networks. The testing partial point clouds are created with 50% of the missing rate.

4.3. Ablation studies

In this section, we present the results of the ablation studies to analyse the effectiveness of four state-of-the-art PointNet-based encoders for feature extraction modules in methods [8,9,19,21,22] and our tree-based decoders. Following PCN [21], TopNet [22], PMPNet [23] and Disp3d [24], the model training and testing are based on the same 8 classes of the ShapeNetCore datasets.

4.3.1. Encoder analysis

We analyse the effectiveness of five state-of-the-art PointNet-based encoders in PointNet [8], PointNet++ [9], AE [19], PCN [21] and TopNet [22].

In this experiment, to analyse the most effective encoder, we use the same TreeNet decoder as described in Section 4.3.2 but only change the encoder. The results of this analysis are reported in Table 1. As can be seen, the PCNs encoder shows a better performance than others. Thus, we choose PCNs encoder as our final encoder. Note that compared across these methods using the same encoder, our model outperforms the state-of-the-art methods (see Section 4.4.3).

4.3.2. Tree-based decoders analysis

Tree-based decoders analysis is based on choosing the number of tree levels L in TreeNet-multiclass, TreeNet-binary and TreeNet, and weights λ_1 and λ_2 in Eq. (6).

We choose the number of tree levels L in TreeNet-multiclass, TreeNet-binary and TreeNet for an output point cloud size 2048×3 by varying L in $3, 4, 5, 6, 7, 8$, respectively. The results on 8 classes of the ShapeNetCore dataset are reported in Table 2. Based on the results, we use 3 levels in TreeNet-multiclass and 7 levels in TreeNet-binary and TreeNet. As shown in the table, one notices that after a certain level, the increase of the tree depth does not necessarily increase the quality of the output, which is due to the fact that 3 levels in TreeNet-multiclass and 7 levels in TreeNet-binary and TreeNet already sufficiently include the majority of features in the output of 2048 points in this experiment. The results for different weights λ_1 and λ_2 are reported in Table 3. We set $\lambda_1 = 0.2$ and $\lambda_2 = 0.8$ based on these results. In all experiments, regardless of the value for L and λ above, our TreeNet-multiclass, TreeNet-binary and TreeNet outperform the state-of-the-art methods (see Table 4 in Section 4.4.3).

Table 3
Quantitative comparison between different weights for our final loss on 8 classes of ShapeNet. The Chamfer distance is reported multiplied by (10^3).

λ_1, λ_2	TreeNet-binary	TreeNet
$\lambda_1 = 0.1$, $\lambda_2 = 0.9$	0.980	0.866
$\lambda_1 = 0.2$, $\lambda_2 = 0.8$	**0.919**	**0.817**
$\lambda_1 = 0.3$, $\lambda_2 = 0.7$	0.943	0.824
$\lambda_1 = 0.4$, $\lambda_2 = 0.6$	0.930	0.858
$\lambda_1 = 0.5$, $\lambda_2 = 0.5$	0.976	0.870
$\lambda_1 = 0.6$, $\lambda_2 = 0.4$	0.922	0.838
$\lambda_1 = 0.7$, $\lambda_2 = 0.3$	0.959	0.895
$\lambda_1 = 0.8$, $\lambda_2 = 0.2$	0.946	0.884
$\lambda_1 = 0.9$, $\lambda_2 = 0.1$	0.952	0.896

Table 4
Quantitative comparison of our approach against previous works tested on 8 and 50 classes of testing data. The Chamfer distances are the average results on all 8 and 50 classes and are reported multiplied by (10^3). Bold denotes the top three performing measures.

Methods	8 Classes	50 Classes
FoldingNet [20]	1.862	1.242
PCN [21]	1.946	1.251
TopNet [22]	1.378	1.079
PMPNet [23]	1.911	1.081
Disp3d [24]	1.880	1.649
TreeNet-multiclass(Ours)	**1.308**	**0.969**
TreeNet-binary(Ours)	**0.926**	**0.757**
TreeNet(Ours)	**0.823**	**0.596**

4.4. Evaluation of tree-based decoder

In this section, we analyse the effectiveness of TreeNet-multiclass, TreeNet-binary and TreeNet trained on 8 and 50 classes of training data by comparing our methods with state-of-the-art learning-based methods FoldingNet [20], PCN [21], TopNet [22], PMPNet [23] and Disp3d [24].

4.4.1. Effectiveness of TreeNet-multiclass

To analyse the effectiveness of the proposed TreeNet-multiclass for multi-class 3D point cloud completion, we train all networks including FoldingNet [20], PCN [21], TopNet [22], PMPNet [23], Disp3d [24] and our TreeNet-multiclass on the same 8 and 50 classes of training datasets, respectively. All networks are evaluated on the 8 and 50 classes of testing datasets (Section 4.2), respectively.

Figure 6 shows the results of each trained model from the same partial input. The second row in Fig. 6 shows the results of each method based on 8 classes of training data, and the third row shows the results based on 50 classes of training data. Chamfer distance (CD) has been calculated between each result and ground truth and shown at the bottom of each figure. Based on the Chamfer distances, the quality of output 3D point clouds from FoldingNet [20], PCN [21], TopNet [22] and PMPNet [23] decreases when the number of classes increases from 8 to 50 in the train-

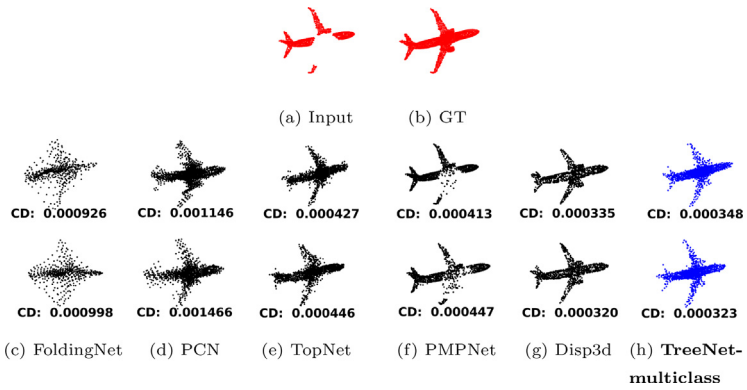


Fig. 6. Qualitative completion results from the same partial input based on 8 and 50 classes of training data. The second and third rows show the completion results of models trained on 8 and 50 classes of data, respectively.

Table 5
Evaluations of every class in 8 classes of testing datasets. The Chamfer distances reported are multiplied by (10^3). Bold denotes the top three performing measures.

Methods	plane	cabinet	car	chair	lamp	sofa	table	vessel
FoldingNet [20]	1.56	1.99	1.15	2.87	2.39	1.78	1.95	1.21
PCN [21]	1.33	2.11	1.18	2.94	2.43	1.95	2.06	1.57
TopNet [22]	0.89	1.57	1.02	1.99	**1.70**	1.48	**1.35**	1.03
PMPNet [23]	1.18	2.65	1.53	2.46	2.27	1.75	2.50	0.95
Disp3d [24]	0.95	1.51	0.92	2.73	3.73	1.46	2.42	1.32
TreeNet-multiclass	**0.75**	**1.40**	**0.91**	**1.97**	1.75	**1.38**	1.48	**0.82**
TreeNet-binary	**0.53**	**1.17**	**0.60**	**1.30**	**1.37**	**0.82**	**1.00**	**0.62**
TreeNet	**0.44**	**0.96**	**0.53**	**1.12**	**1.29**	**0.74**	**0.98**	**0.53**

Table 6
Evaluations on the 50 classes of testing datasets. Nine classes of results are selected and displayed. The Chamfer distance is reported multiplied by (10^3). Bold denotes the top three performing measures.

Methods	table	bench	bus	laptop	pistol	pot	monitor	bed	mug
FoldingNet [20]	1.98	1.42	0.60	1.40	1.21	1.44	1.16	1.98	1.85
PCN [21]	1.96	1.14	0.63	1.01	1.07	1.49	1.15	2.23	1.92
TopNet [22]	1.49	0.92	0.63	0.86	0.96	1.58	0.98	1.99	1.59
PMPNet [23]	2.46	0.83	0.77	0.92	**0.80**	**0.97**	**0.72**	**1.02**	1.17
Disp3d [24]	5.05	1.33	**0.49**	**0.61**	0.89	1.99	1.53	3.05	1.46
TreeNet-multiclass	**1.44**	**0.81**	0.57	0.62	0.84	1.50	0.88	**1.98**	**1.01**
TreeNet-binary	**1.02**	**0.69**	**0.37**	**0.59**	**0.55**	**0.93**	**0.62**	2.11	**1.09**
TreeNet	**0.87**	**0.48**	**0.30**	**0.38**	**0.48**	**0.87**	**0.50**	**1.15**	**0.78**

ing data, whereas our TreeNet-multiclass model has shown stable results when the number of classes increases.

The average Chamfer distances of our TreeNet-multiclass against the state-of-the-art methods tested on 8 and 50 classes of the testing data are shown in Tables 4, 5 and 6 with qualitative results in Fig. 7. Our TreeNet-multiclass outperforms FoldingNet [20], PCN [21], TopNet [22], PMPNet [23] and Disp3d [24] across all 8 and 50 classes of testing data. As shown in Fig. 7, Disp3d [24] generates a car from a partial vessel as an input, which shows the poor ability on multi-classes training. The multi-classes training for our TreeNet-multiclass solves this problem and distinguishes features between each class. Most noticeably, the result of TreeNet-multiclass has shown a 5.08% improvement trained on 8 classes of data over the next best method TopNet [22], and a 10.19% improvement trained on 50 classes over TopNet [22], demonstrating the capability of TreeNet-multiclass in handling multi-class 3D point cloud completion tasks effectively.

4.4.2. Effectiveness of TreeNet-binary

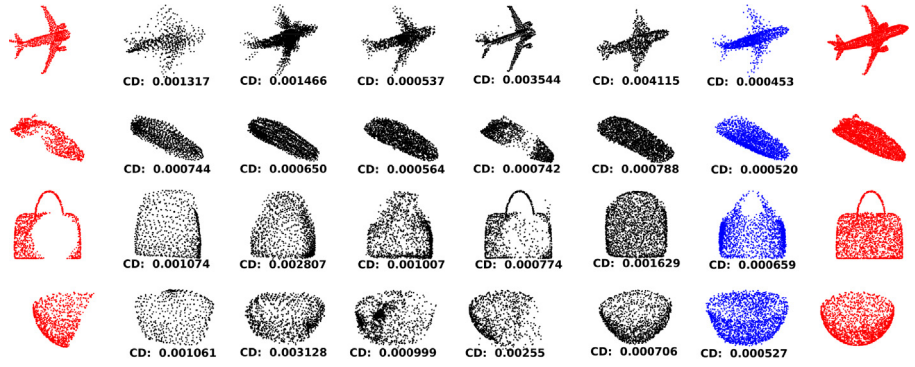
To analyse the effectiveness of our proposed TreeNet-binary, all networks are trained on the same 8 and 50 classes of training datasets and evaluated on the same 8 and 50 classes of testing datasets (Section 4.2), respectively. Figure 8 illustrates the final output (h) of the TreeNet-binary which is the combination of the partial input (a) and the output of the right leaf node (g), which proves that TreeNet-binary is able to generate points in missing areas and fully preserve the original partial input.

We show average Chamfer distances in Tables 4, 5 and 6 with qualitative completion results in Fig. 9. As can be seen that TreeNet-binary also outperforms FoldingNet [20], PCN [21], TopNet [22], PMPNet [23] and Disp3d [24] across 8 and 50 classes of testing data, respectively. The result of TreeNet-binary has shown a 32.80% improvement trained on 8 classes of data over the next best method TopNet [22] and a 29.84% improvement trained on 50 classes of data over TopNet [22].

4.4.3. Effectiveness of TreeNet

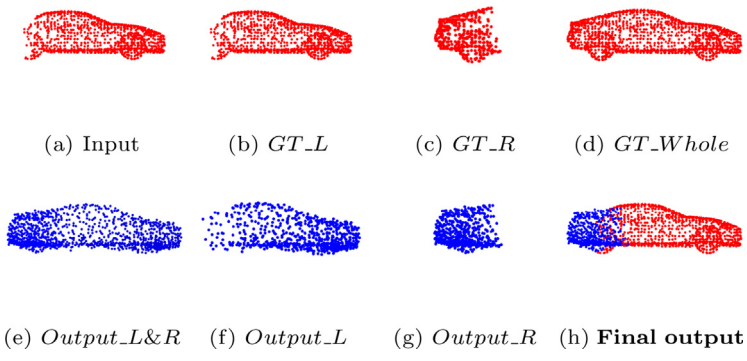
To assess the effectiveness of TreeNet, we not only compare TreeNet with FoldingNet [20], PCN [21], TopNet [22], PMPNet [23] and Disp3d [24] but also with TreeNet-multiclass and TreeNet-binary.

The average Chamfer distances of our TreeNet against the state-of-the-art methods on 8 and 50 classes of the testing data are shown in Tables 4, 5 and 6, with qualitative results shown in Fig. 10. TreeNet combines the advantages of TreeNet-multiclass and TreeNet-binary and outperforms TreeNet-multiclass and TreeNet-binary on 8 and 50 classes of testing data. In addition, TreeNet significantly outperforms FoldingNet [20], PCN [21], TopNet [22],



(a) Input (b) FoldingNet (c) PCN (d) TopNet (e) PMP-Net (f) Disp3d (g) **TreeNet-multiclass** (h) GT

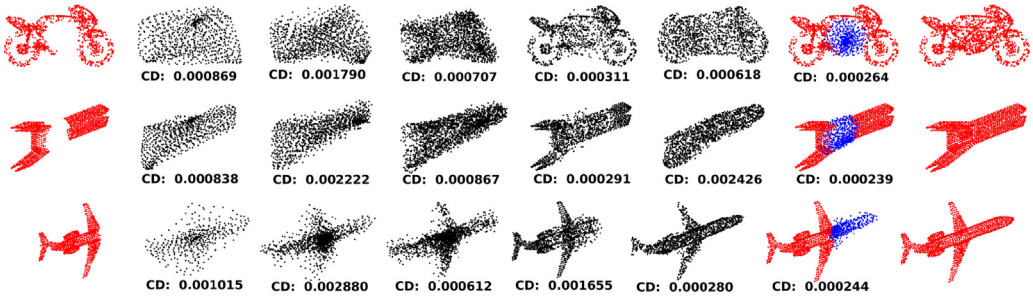
Fig. 7. Qualitative completion results of models trained on 8 (first and second row) and 50 (third and fourth row) classes of data, respectively.



(a) Input (b) GT_L (c) GT_R (d) GT_Whole

(e) $Output\_L\&R$ (f) $Output\_L$ (g) $Output\_R$ (h) **Final output**

Fig. 8. Illustration of TreeNet-binary output.



(a) Input (b) FoldingNet (c) PCN (d) TopNet (e) PMP-Net (f) Disp3d (g) **TreeNet-binary** (h) GT

Fig. 9. Qualitative completion results of methods trained on 50 classes of data.

PMPNet [23] and Disp3d [24] across 8 and 50 classes of testing data. The result of TreeNet has shown a 40.28% improvement on 8 classes over the next best method TopNet [22], and a 44.76% improvement on 50 classes over TopNet [22]. As shown in Fig. 10, FoldingNet [20], PCN [21] and TopNet [22] have failed to recover structural and spatial details such as sharp edges of a bench (2nd column), a laptop (6th column) and a monitor (7th column) and topology change of a jar (5th column). Disp3d [24] confuses features between classes to some extent and generates a pot from a partial jar (5th column). In contrast, TreeNet has successfully generated these details and never confuses features between classes. Most importantly, FoldingNet [20], PCN [21], TopNet [22] and Disp3d [24] all lose the structural and spatial details of the original partial input. On the contrary, our TreeNet-binary and TreeNet generate points in missing areas and preserve the partial input.

4.5. Generalization on unknown classes

We extensively conduct several comparison experiments to evaluate the generalization capability of our networks. All learned models including FoldingNet [20], PCN [21], TopNet [22], PMP-Net [23], Disp3d [24] and ours are trained on the same 50 classes of training data and directly tested on the unknown classes (Section 4.2) that are never trained. These shape datasets in unknown classes are different from the training datasets, which poses a significant challenge to the generalization of all learning-based methods.

A partial 3D point cloud from an unknown class can be passed through the encoder to all sub-trees in the decoder. The root nodes in TreeNet-multiclass and TreeNet contain common features extracted from the encoder, and each sub-tree in the decoder uses these common features to generate a complete 3D point cloud. As

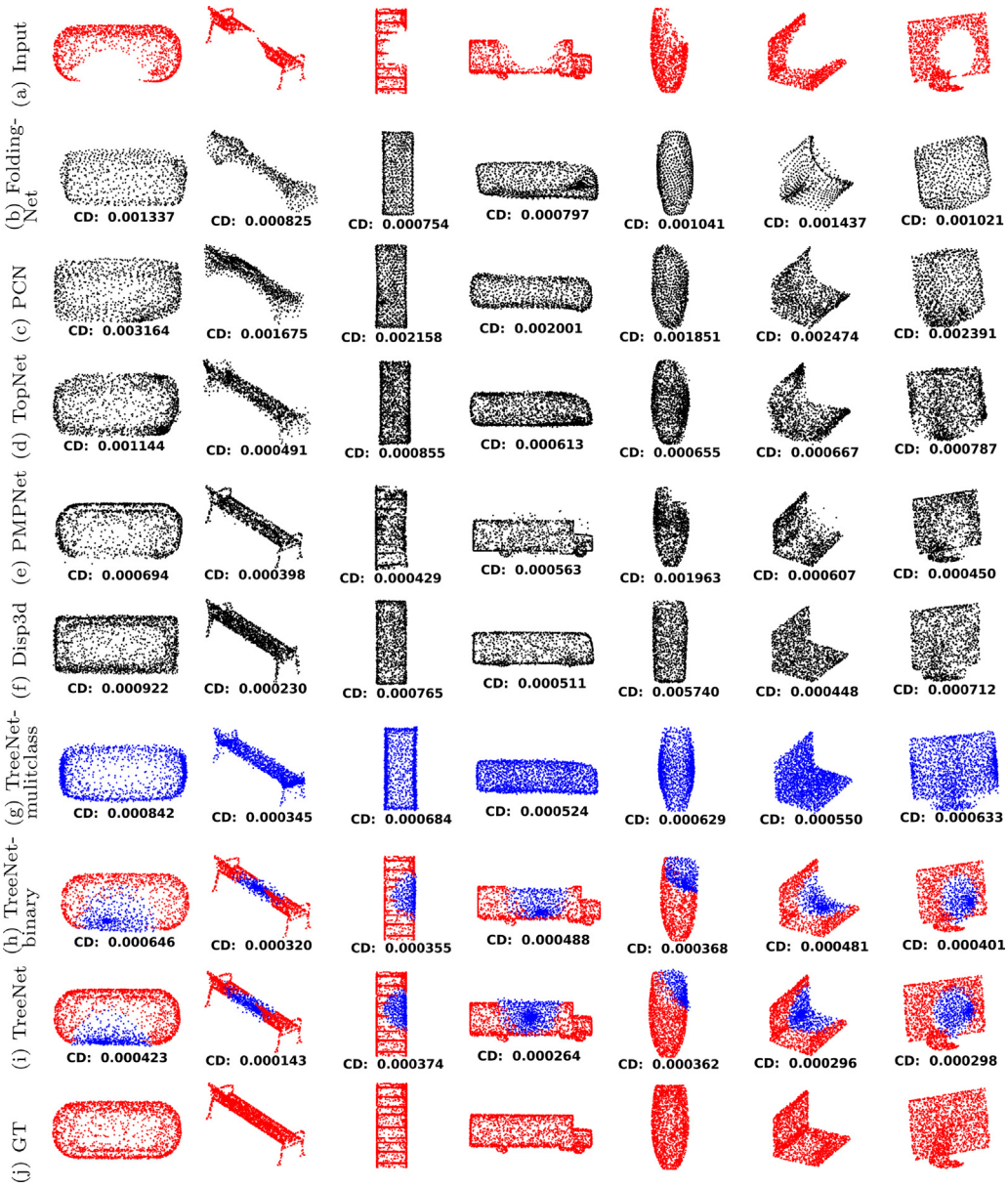


Fig. 10. Qualitative completion results of models trained on 50 classes of data.

shown in Fig. 11, a partial 3D shape from an unknown class is passed through the encoder to all sub-trees in the decoder and we show the generalization abilities from different sub-trees in TreeNet-multiclass and TreeNet, respectively. The final output is the best completion result with the lowest Chamfer distance between each output and the ground truth.

The performance of each method on unknown classes is reported in Table 7 with qualitative completion results of each method in Fig. 12. Our TreeNet ranks first and achieves the best completion results on all five unknown classes, which shows the strong generalization and robustness evaluated on unknown classes and also proves that the sub-trees can share common features among different classes. Refer to Section 5 for more detailed discussions and illustrations about unknown data.

Table 7
Evaluations on unknown classes. The Chamfer distance is reported multiplied by (10^3). Bold denotes the top three performing measures.

Methods	camera	basket	stove	tower	printer	*Avg.*
FoldingNet [20]	2.05	1.62	1.59	1.42	1.99	1.73
PCN [21]	2.03	1.64	1.49	1.34	2.02	1.70
TopNet [22]	2.09	1.67	1.49	1.34	1.81	1.68
PMPNet [23]	**1.73**	2.34	1.75	1.63	**1.47**	1.78
Disp3d [24]	5.58	2.71	2.76	5.47	3.93	4.09
TreeNet-multiclass	1.88	**1.33**	**1.20**	**1.13**	1.57	**1.42**
TreeNet-binary	**1.73**	**1.11**	**1.34**	**1.02**	**1.39**	**1.32**
TreeNet	**0.98**	**0.85**	**0.77**	**0.62**	**0.90**	**0.82**

5. Discussion and limitation

In this section, we first discuss the performance of completion results on unknown datasets and then discuss the limitations of our proposed TreeNet.

To show the common features that can be shared among sub-trees, we tested six types of car models using a single sub-tree in the decoders of TreeNet-multiclass and TreeNet. The results show that our networks can handle the six types of 3D car models with better performance than the other five state-of-the-art networks (FoldingNet [20], PCN [21], TopNet [22], PMPNet [23] and Disp3d [24]), as shown in Fig. 13. This experiment indicates that

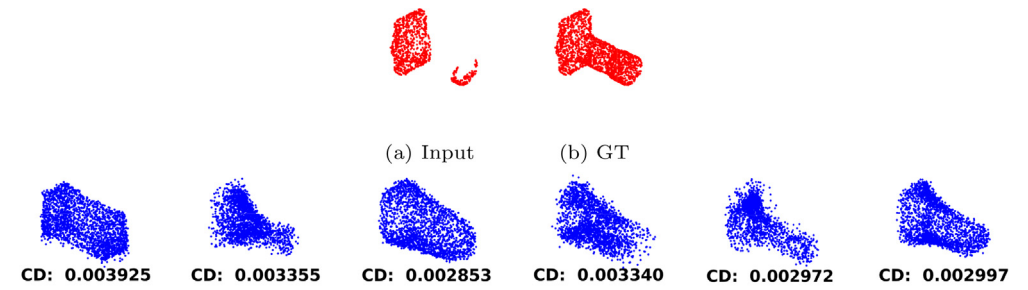


Fig. 11. Completion results from different sub-trees in the tree for data in unknown classes.

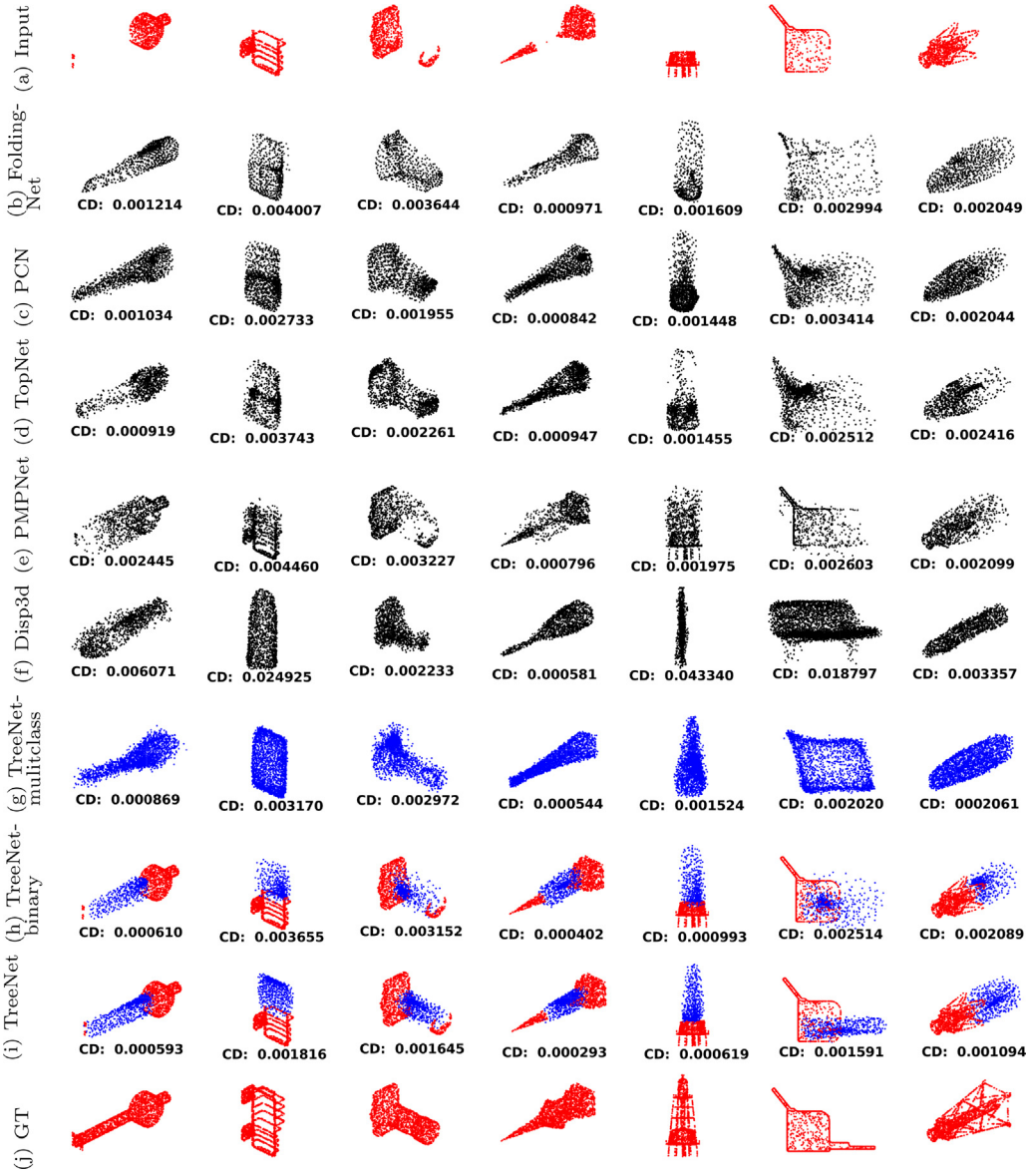


Fig. 12. Qualitative completion results of models on unknown classes with unknown shapes.

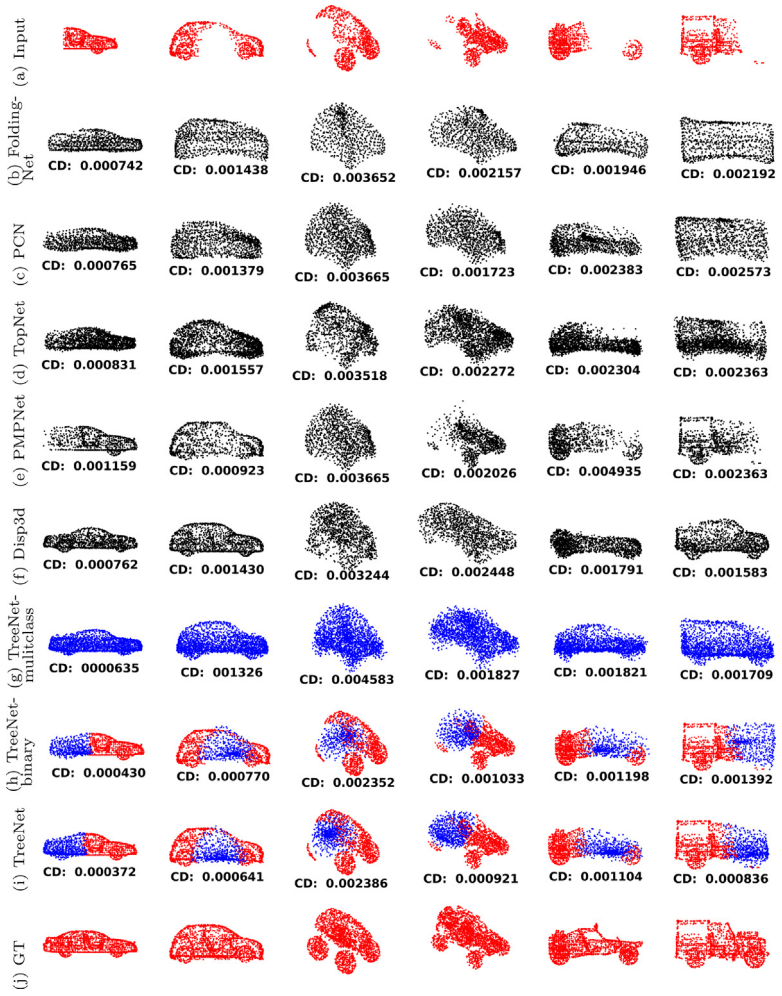


Fig. 13. Qualitative completion results tested on different types of car models.

a single sub-tree can share common features for different types of 3D models in one class. Also shown in our experimental results on unknown classes, the sub-trees can share common features among different classes, as shown in Figs. 12 and 11.

For datasets from unknown classes with unknown shapes, all networks have certain generalizations to generate complete shapes under such challenging conditions. Although our TreeNet ranks first and achieves the best completion results on all five unknown classes, which shows the strong generalization and robustness evaluated on unknown classes, all networks fail to generate detailed information in missing areas, as shown in Fig. 12 and Table 7. One possible reason could be that the current loss function measures geometrical features based on the global shape information, which needs more local feature information, thus, resulting in insufficient features for fine details locally. This limitation will be addressed in our feature work.

One limitation of our TreeNet-multiclass and TreeNet is that the model size is linearly correlated with the number of training classes, as shown in Table 8. However, the number of parameters is still much smaller than that of Disp3d [24]. It is worth noting that the number of parameters in our TreeNet-binary does not increase when the number of training classes increases, which is similar to the state-of-the-art networks (FoldingNet [20], PCN [21], TopNet [22], PMPNet [23] and Disp3d [24]). Although the number of parameters of our TreeNet-multiclass and TreeNet is around 3 (8 classes) to 10 (50 classes) times bigger than that of FoldingNet [20], PCN [21], TopNet [22] and PMPNet [23], the result of TreeNet is 40.28% better on 8 classes than the next best method TopNet [22], and 44.76% better on 50 classes than TopNet [22]. This limitation will be addressed in our feature work.

Table 8
The number of network parameters in each method. '-' means the number of network parameters does not increase when the number of training classes increases.

Methods	Parameters (trained on 8 classes)	Parameters (trained on 50 classes)
FoldingNet [20]	2,402,054	-
PCN [21]	5,286,659	-
TopNet [22]	9,965,117	-
PMPNet [23]	5,435,163	-
Disp3d [24]	100,318,976	-
TreeNet-multiclass	15,515,904	59,599,104
TreeNet-binary	22,869,248	-
TreeNet	30,216,448	74,299,648

6. Conclusion

In this paper, we show the proposed networks, TreeNet-multiclass, TreeNet-binary and TreeNet, can produce high-quality 3D point clouds on multi-class datasets for point cloud completion, and these novel network structures outperform the state-of-the-art learning-based methods in terms of the quality of the completion results on trained and unknown classes. TreeNet-multiclass is for multi-class training and assigns a specific class of the completion task to each sub-tree, while TreeNet-binary preserves the original partial input and generates points in missing areas. We propose

three forward propagation methods to train TreeNet-multiclass, TreeNet-binary and TreeNet separately. Our models achieve high-quality completion results and show remarkable generalization and robustness to unknown classes that are not trained.

In future work, we will set up the dataset data in each class based on semantic information. Shapes with similar semantic features will pass through their corresponding sub-trees, which may vastly reduce the number of sub-trees when the number of training classes increases drastically. The loss function for calculating the semantic features can also generate detailed information on the missing areas locally.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

[1] L. Chen, T.W. Day, W. Tang, N.W. John, Recent developments and future challenges in medical mixed reality, in: 2017 IEEE International Symposium on Mixed and Augmented Reality (ISMAR), IEEE, 2017, pp. 123–135, doi:10.1109/ISMAR.2017.29.

[2] B. Fang, G. Mei, X. Yuan, L. Wang, Z. Wang, J. Wang, Visual slam for robot navigation in healthcare facility, Pattern Recognit. 113 (2021) 107822, doi:10.1016/j.patcog.2021.107822.

[3] H. Wei, L. Meng, An accurate stereo matching method based on color segments and edges, Pattern Recognit. 133 (2023) 108996, doi:10.1016/j.patcog.2022.108996.

[4] M. Li, Y. Xie, L. Ma, Paying attention for adjacent areas: learning discriminative features for large-scale 3d scene segmentation, Pattern Recognit. 129 (2022) 108722, doi:10.1016/j.patcog.2022.108722.

[5] L. Xi, Y. Zhao, L. Chen, Q.H. Gao, W. Tang, T.R. Wan, T. Xue, Recovering dense 3D point clouds from single endoscopic image, Comput. Methods Programs Biomed. 205 (2021) 106077, doi:10.1016/j.cmpb.2021.106077.

[6] A. Dai, C.R. Qi, M. Nießner, Shape completion using 3D-encoder-predictor CNNs and shape synthesis, in: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 6545–6554, doi:10.1109/CVPR.2017.693.

[7] X. Han, Z. Li, H. Huang, E. Kalogerakis, Y. Yu, High-resolution shape completion using deep neural networks for global structure and local geometry inference, in: 2017 IEEE International Conference on Computer Vision (ICCV), 2017, pp. 85–93, doi:10.1109/ICCV.2017.19.

[8] R.Q. Charles, H. Su, M. Kaichun, L.J. Guibas, PointNet: deep learning on point sets for 3D classification and segmentation, in: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 77–85, doi:10.1109/CVPR.2017.16.

[9] C.R. Qi, L. Yi, H. Su, L.J. Guibas, PointNet++: deep hierarchical feature learning on point sets in a metric space, in: NIPS'17: Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS), Curran Associates Inc., Red Hook, NY, USA, 2017, pp. 5105–5114.

[10] H. Wang, Y. Zhang, W. Liu, X. Gu, X. Jing, Z. Liu, A novel GCN-based point cloud classification model robust to pose variances, Pattern Recognit. 121 (2022) 108251, doi:10.1016/j.patcog.2021.108251.

[11] C. Yu, Z. Zhang, H. Li, J. Sun, Z. Xu, Meta-learning-based adversarial training for deep 3d face recognition on point clouds, Pattern Recognit. 134 (2023) 109065, doi:10.1016/j.patcog.2022.109065.

[12] J.S. K. Hu, T. Kuai, S.L. Waslander, Point density-aware voxels for LiDAR 3D object detection, in: 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2022, pp. 8459–8468, doi:10.1109/CVPR52688.2022.00828.

[13] R. Qian, X. Lai, X. Li, 3D object detection for autonomous driving: asurvey, Pattern Recognit. 130 (2022) 108796, doi:10.1016/j.patcog.2022.108796.

[14] Y. Su, W. Liu, Z. Yuan, M. Cheng, Z. Zhang, X. Shen, C. Wang, DLA-Net: learning dual local attention features for semantic segmentation of large-scale building facade point clouds, Pattern Recognit. 123 (2022) 108372, doi:10.1016/j.patcog.2021.108372.

[15] F. Yang, F. Davoine, H. Wang, Z. Jin, Continuous conditional random field convolution for point cloud segmentation, Pattern Recognit. 122 (2022) 108357, doi:10.1016/j.patcog.2021.108357.

[16] K. Fu, S. Liu, X. Luo, M. Wang, Robust point cloud registration framework based on deep graph matching, in: 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2021, pp. 8889–8898, doi:10.1109/CVPR46437.2021.00878.

[17] L. Xi, W. Tang, T. Xue, T. Wan, Iterative BTreeNet: unsupervised learning for large and dense 3d point cloud registration, Neurocomputing 506 (2022) 336–354, doi:10.1016/j.neucom.2022.07.082.

[18] Z. Zhang, J. Sun, Y. Dai, D. Zhou, X. Song, M. He, Self-supervised rigid transformation equivariance for accurate 3D point cloud registration, Pattern Recognit. 130 (2022) 108784, doi:10.1016/j.patcog.2022.108784.

[19] P. Achlioptas, O. Diamanti, I. Mitliagkas, L.J. Guibas, Learning representations and generative models for 3D point clouds, in: ICML, 2018, pp. 40–49. http://proceedings.mlr.press/v80/achlioptas18a.html

[20] Y. Yang, C. Feng, Y. Shen, D. Tian, FoldingNet: point cloud auto-encoder via deep grid deformation, in: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 206–215, doi:10.1109/CVPR.2018.00029.

[21] W. Yuan, T. Khot, D. Held, C. Mertz, M. Hebert, PCN: point completion network, in: 2018 International Conference on 3D Vision (3DV), IEEE, 2018, pp. 728–737, doi:10.1109/3DV.2018.00088.

[22] L.P. Tchapmi, V. Kosaraju, H. Rezatofighi, I. Reid, S. Savarese, TopNet: structural point cloud decoder, in: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 383–392, doi:10.1109/CVPR.2019.00047.

[23] X. Wen, P. Xiang, Z. Han, Y.-P. Cao, P. Wan, W. Zheng, Y.-S. Liu, PMP-Net: point cloud completion by learning multi-step point moving paths, in: 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2021, pp. 7439–7448, doi:10.1109/CVPR46437.2021.00736.

[24] Y. Wang, D.J. Tan, N. Navab, F. Tombari, Learning local displacements for point cloud completion, in: 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2022, pp. 1558–1567, doi:10.1109/CVPR52688.2022.00162.

[25] S. Thrun, B. Wegbreit, Shape from symmetry, in: Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1, Vol. 2, 2005, pp. 1824–1831Vol. 2, doi:10.1109/ICCV.2005.221.

[26] N.J. Mitra, L.J. Guibas, M. Pauly, Partial and approximate symmetry detection for 3D geometry, in: ACM SIGGRAPH 2006 Papers, in: SIGGRAPH '06, Association for Computing Machinery, New York, NY, USA, 2006, pp. 560–568, doi:10.1145/1179352.1141924.

[27] M. Sung, V.G. Kim, R. Angst, L. Guibas, Data-driven structural priors for shape completion, ACM Trans. Graph. (TOG) 34 (6) (2015), doi:10.1145/2816795.2818094.

[28] Z. Wang, F. Lu, VoxSegNet: volumetric CNNs for semantic part segmentation of 3D shapes, IEEE Trans. Vis. Comput.Graph. (TVCG) 26 (9) (2020) 2919–2930, doi:10.1109/TVCG.2019.2896310.

[29] D.P. Kingma, M. Welling, Auto-encoding variational Bayes, in: 2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14–16, 2014, Conference Track Proceedings, 2014.

[30] I.J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, in: NIPS'14: Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2 (NIPS), MIT Press, 2014, pp. 2672–2680.

[31] H. Fan, H. Su, L. Guibas, A point set generation network for 3D object reconstruction from a single image, in: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 2463–2471, doi:10.1109/CVPR.2017.264.

[32] A.X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, ShapeNet: an information-rich 3D model repository, CoRR (2015) abs/1512.03012.

Long Xi received his MS degree in 2019 from the department of computer science, Xi'an Polytechnic University, Xi'an, China. He is currently a PhD candidate at Bournemouth University, UK. His research interests include computer vision, artificial intelligence and pattern recognition.

Wen Tang received her BA degree in 1984 from Xi'an Polytechnic University, Xi'an, China, and PhD degree from University of Leeds, UK, in 1998. She is currently a full professor at Bournemouth University, UK, and a guest professor at Xi'an Polytechnic University, China. Her research interests include interactive virtual reality software technologies, physicallybased simulation, computer animation algorithms and computer games technology.

TaoRuan Wan is a senior lecturer in the School of Informatics at the University of Bradford, Director of Post Graduate Research EIMC, and a guest professor at Xi'an Polytechnic University, China. His research interests are focused in the areas of Digital Health, VR/AR technologies, Computer Vision, AI and deep learning, Computational Models for visualization, Real-time Modeling and Simulation.