# Attention-Based Transformation from Latent Features to Point Clouds

**Kaiyi Zhang**[1], **Ximing Yang**[1], **Yuan Wu**[1], **Cheng Jin**[1,2]

[1]School of Computer Science, Fudan University, Shanghai, China
[2]Peng Cheng Laboratory, Shenzhen, China
{zhangky20, xmyang19, wuyuan, jc}@fudan.edu.cn

## Abstract

In point cloud generation and completion, previous methods for transforming latent features to point clouds are generally based on fully connected layers (FC-based) or folding operations (Folding-based). However, point clouds generated by FC-based methods are usually troubled by outliers and rough surfaces. For folding-based methods, their data flow is large, convergence speed is slow, and they are also hard to handle the generation of non-smooth surfaces. In this work, we propose AXform, an attention-based method to transform latent features to point clouds. AXform first generates points in an interim space, using a fully connected layer. These interim points are then aggregated to generate the target point cloud. AXform takes both parameter sharing and data flow into account, which makes it has fewer outliers, fewer network parameters, and a faster convergence speed. The points generated by AXform do not have the strong 2-manifold constraint, which improves the generation of non-smooth surfaces. When AXform is expanded to multiple branches for local generations, the centripetal constraint makes it has properties of self-clustering and space consistency, which further enables unsupervised semantic segmentation. We also adopt this scheme and design AXformNet for point cloud completion. Considerable experiments on different datasets show that our methods achieve state-of-the-art results.

## Introduction

Recently, there have been considerable deep-learning-based point cloud generation and completion methods based on the architecture of autoencoder. Methods such as PointNet (Qi et al. 2017a), PointNet++ (Qi et al. 2017b), DGCNN (Wang et al. 2019), and Point Transformer (Zhao et al. 2020) have studied encoder in detail to obtain better point cloud representations. There are also many methods exploring the design of the decoder. For example, (Fan, Su, and Guibas 2017; Achlioptas et al. 2017; Yuan et al. 2018; Yang et al. 2021) simply use fully connected layers to generate coarse outputs. (Yang et al. 2018; Groueix et al. 2018; Liu et al. 2019a; Wen et al. 2020) deform 2d squares into 3d surfaces. As shown in Figure 1(a), fully connected layers and folding operations are two commonly used methods for the transformation from latent features to point clouds. However, they do not fully explore the transformation from aspects of point constraints, network parameters, data flow, and convergence speed.



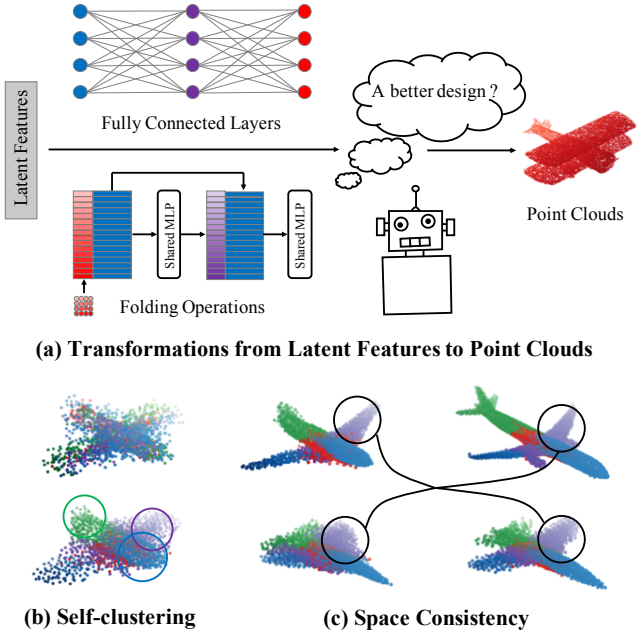



Figure 1: (a) Two commonly used methods for the transformation. (b, c) Two properties of AXform with K branches.

As shown in the upper airplane in Figure 1(b), point clouds generated by the FC-based method usually have outliers and rough surfaces. It is because the generated points do not share parameters, i.e., the available parameters of each point are few. When it is expanded to multiple branches, the generated point clouds do not have the property of self-clustering. The property of self-clustering means points generated by each branch are gathered together and it reflects the locality of an object. The lower airplane in Figure 1(b) is an example of a self-clustered point cloud. The reason for this problem is that each generated point is free and not subject to a centripetal constraint.

Different from the FC-based method, the folding-based method has much fewer parameters and uses grid priors to constrain the generated points on a smooth 2-manifold. This

constraint enables it to be expanded to multiple branches. However, its convergence speed is slow as it is difficult to deform a 2d square into various 3d surfaces with only a few parameters. In the same time, 3d objects often have non-smooth surfaces. Folding operations are also hard to handle these situations as the generated surfaces are folded from smooth 2-manifolds.

To address these issues, we propose an attention-based method AXform to transform latent features to point clouds. It first adopts a fully connected layer to generate points in an interim space. Then weighting operations are performed by an attention mechanism to aggregate target points in the interim space. Finally, these points are mapped to 3d space, which are the final point cloud. AXform has four advantages. First, it has fewer parameters comparing with FC-based and folding-based methods, which is benefit from sharing network parameters. Second, its convergence speed is faster than these two methods. Third, the points generated by AXform are more likely to concentrate due to a centripetal constraint. Thus AXform can be easily expanded to multiple branches for local generations. Last, as shown in Figure 1(c), our AXform with K branches has a property of space consistency, which further enables unsupervised semantic segmentation.

Our main contributions are the following:

- We propose a novel attention-based method for the transformation from latent features to point clouds. Considerable experiments show that it performs better than previous methods.
- The proposed AXform can be expanded to multiple branches for local generations. It has the properties of self-clustering and space consistency, which can be used for unsupervised semantic segmentation on the generated point clouds.
- We apply AXform to point cloud completion, and it achieves state-of-the-art results on the PCN dataset.

## Related Work

**Attention in Point Clouds** Attention mechanism has been widely used in point clouds to get better point cloud representations. (Lee et al. 2019) presents an attention-based network to simulate interactions between elements in point clouds. (Yang et al. 2019b) uses attention layers to capture the relationship between points. (Li et al. 2019) introduces a self-attention unit to enhance the quality of feature integration when upsampling point clouds. (Zhang, Yan, and Xiao 2020) uses an attention module to optimize the input point cloud, which reduces outliers in the generated point cloud. (Wen et al. 2020) proposes a skip-attention model to capture the geometric information from local regions of the input to get a better representation. There are also many other works exploring the application of attention in point clouds (Fuchs et al. 2020; Shajahan, Nayel, and Muthuganapathy 2020; Liu et al. 2019b; Hu et al. 2020; Zhang et al. 2020).

**Point Cloud Generation** Plenty of tasks need to generate point clouds as the output, which attracted a lot of research interest. (Fan, Su, and Guibas 2017) generates point clouds by using a fully connected branch and a 2d deconvolution branch. (Yang et al. 2018) proposes an idea of deforming a 2d square into a 3d surface called folding, which has fewer parameters. (Groueix et al. 2018) further expands the folding operation to multiple branches and achieves better surface reconstructions. (Sun et al. 2020) presents a novel autoregressive model by generating points one by one like 3D printing. (Yang et al. 2021) generates a structural skeleton to achieve controllable generation. (Valsesia, Fracastoro, and Magli 2019; Hui et al. 2020) design deconvolution operations on point clouds to do generation. (Achlioptas et al. 2017; Li et al. 2018; Zamorski et al. 2018; Shu, Park, and Kwon 2019) proposes some GAN models. (Yang et al. 2019a; Klokov, Boyer, and Verbeek 2020; Kim et al. 2020; Luo and Hu 2021) explore flow-based models for reversible point cloud generation.

**Point Cloud Completion** This task aims to complete authentic point clouds when given inputs with various missing patterns. It can contribute to a series of downstream applications like robotics operations (Varley et al. 2017), scene understanding (Dai et al. 2018), and virtual operations of complete shapes (Rui, Plinio, and Alexandre 2017).

Some methods explore to apply deep learning on supervised point cloud completion. (Yuan et al. 2018) provides an autoencoder to combine the global and local shape information for point cloud completion. (Liu et al. 2019a) generates patches by using multiple branches to get a better locality. (Xie et al. 2020) transforms point cloud into a new voxel representation, which better retains the spatial information of original partial point clouds. (Xie et al. 2021) designs a differentiable renderer and applies adversarial training to realize better point supervisions. Different from previous methods, (Wen et al. 2021) first proposes the idea of completing the objects by moving original points step by step. In addition, (Tchapmi et al. 2019; Hu et al. 2019; Sarmad, Lee, and Kim 2019; Huang et al. 2020; Wang et al. 2020; Zhang, Yan, and Xiao 2020; Wen et al. 2020; Wang, Marcelo H., and Lee 2020; Pan 2020; Alliegro et al. 2021; Pan et al. 2021) play an important role in promoting point cloud completion.

## Approach

In this section, we first describe the framework of our method AXform and compare it with previous methods. Then we expand AXform to multiple branches and compare it with AtlasNet (Groueix et al. 2018). Finally, we show how to apply AXform to point cloud completion.

### Our Method

**AXform Framework** AXform aims to achive a better transformation from final latent features to point clouds. As shown in Figure 2(b), it includes three steps: interim points generation by fully connected layers, attention-based interim points aggregation, and 3D-mapping by shared FC.

Assuming that the input features $f_{in} \in R^{K_1}$. AXform first uses a fully connected layer to generate $N$ points in $R^{K_2}$ from $f_{in}$. From these $N$ points, an attention mechanism is adopted to generate an $M * N$ attention map, where $M$ is the final number of points we want to generate. Through this attention map, AXform aggregates these $N$ interim points to

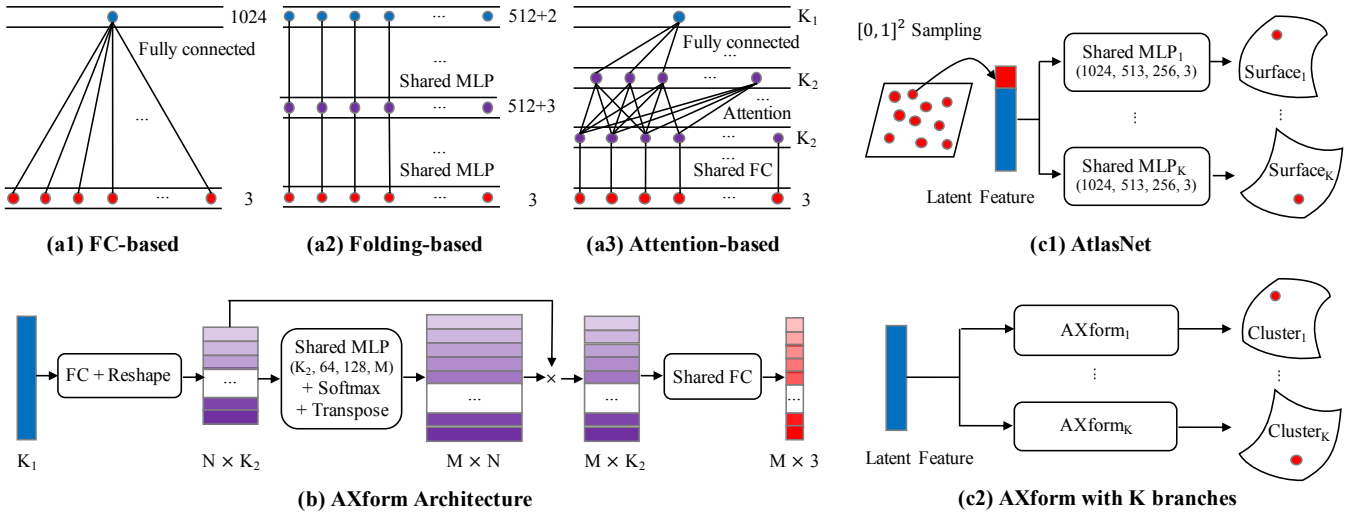



Figure 2: (a-1,2,3) The architectural difference between AXform and previous FC-based and folding-based methods. The number on the right of each figure represents the space dimension. (b) The architecture of AXform with one branch. (c-1,2) The architectural difference between AtlasNet and our AXform with K branches.

$M$ new points. The aggregation process is a convex combination process, therefore the $M$ points are ensured to be in the convex hull constructed by the $N$ interim points in $R^{K_2}$, thus the locality of the $M$ points are guaranteed. In detail, a shared MLP $(K_2, 64, 128, M)$ are firstly adopted to transform $N$ points in $R^{K_2}$ into an $N \times M$ matrix. Then softmax activation function is applied on the dimension of $N$ and the matrix is transposed to form an $M \times N$ attention map. Finally, the original $N$ points are aggregated by the attention map to generate $M$ new points in $R^{K_2}$. After the attention-based interim points aggregation, each point of the $M$ points is mapped to a 3d point by a shared fully connected layer.

**Comparison with FC and Folding** Fully connected layers and folding operations are two commonly used methods for the transformation from latent features to point clouds. As shown in Figure 2(a-1,2,3), we regard features as points in a high-dimensional space and mark them blue. Intermediate feature points are marked purple and final outputs are marked red.

For the FC-based method, each generated point only uses about $1024 \times 3$ parameters. Instead, points generated by AXform share the first fully connected layer $(K_1 \rightarrow K_2)$. Combining Figure 2(b), we can easily infer that each point generated by AXform uses about $K_1 \times K_2 \times N$ parameters. In our experiments, we set $K_1 = 128, K_2 = 32, N = 128$, and thus $K_1 \times K_2 \times N$ is much larger than $1024 \times 3$. Each point use more parameters attentively can not only improves the quality of outputs but also make the outputs have a property of self-clustering. In addition, the number of whole parameters in AXform is mainly contributed by the first fully connected layer $(K_1 \times K_2 \times N)$. It is much fewer than what in the FC-based method $(1024 \times 2048 \times 3)$.

For the folding-based method, since each point in a 2d grid is concatenated by the same 512-dim latent feature, the data flow in the network is large. Its scale depends on the number of target points. Instead, the main memory consumption of data flow in AXform lies in the multiplication of the attention map and the interim point set. It depends on the number of interim points. By setting $N < M$, AXform generally takes up less memory. In addition, the folding-based method uses grid priors to constrain the generated points on a smooth 2-manifold. However, this prior is often too strong, leading to a slow convergence speed. But AXform has a weaker constraint on the generated points based on the attention mechanism, so the convergence speed is quite faster.

## Multiple Branches

Since AXform uses an attention mechanism to generate each target point, and its weighting operation can be regarded as a convex combination. Assuming that the convex hull formed by the interim point set $S$ containing $N$ points is $conv(S)$, the final output containing $M$ points will fall inside $conv(S)$. This centripetal constraint forces the final output to be more concentrated rather than scattered. Therefore, when we expand AXform to multiple branches, points generated by each branch will have a property of self-clustering. Based on this property and referring to the AtlasNet (Groueix et al. 2018) architecture, we propose AXform with K branches, as shown in Figure 2(c2).

Different from AtlasNet which combines each point sampled from a 2d square with the same latent feature and then performs multi-branch folding, our method directly transforms the latent feature into multiple small point clusters by using AXforms. In the following experiments, we will compare them in detail to show the superiority of our method.

## Application on Point Cloud Completion

Since previous methods have done considerable exploration on the use of autoencoder for point cloud completion, we

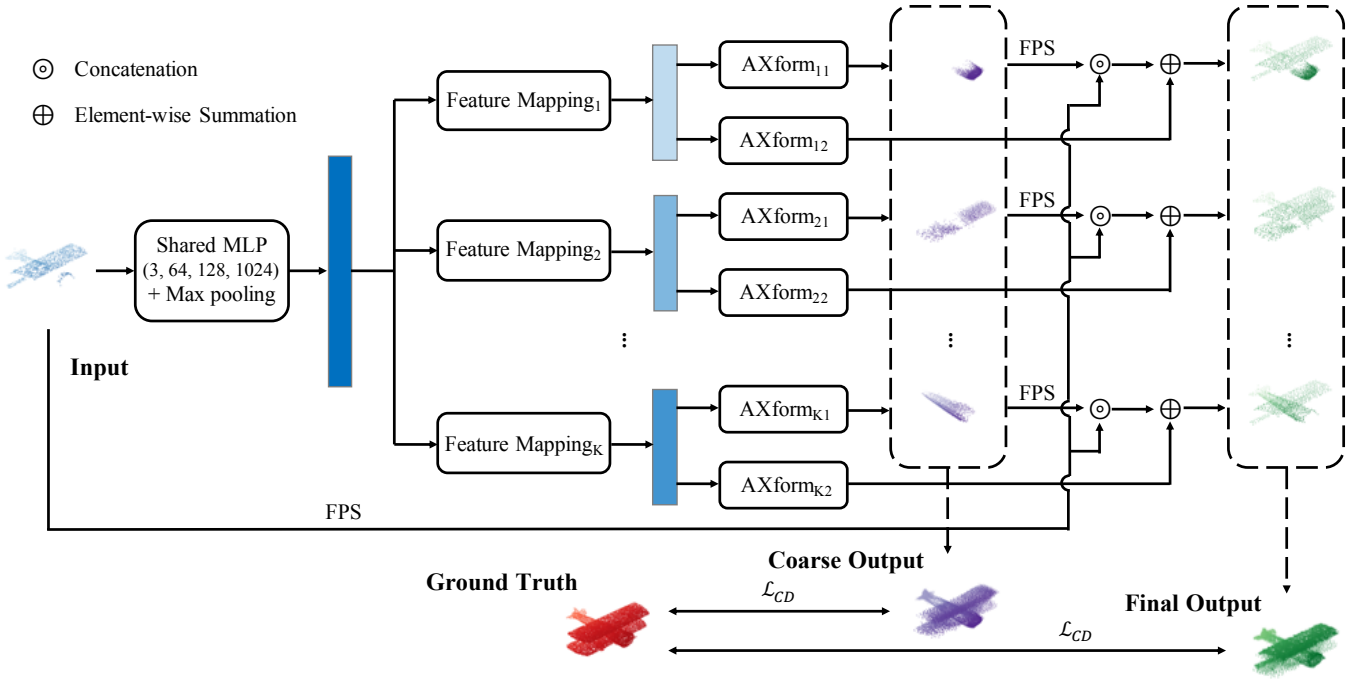



Figure 3: The architecture of AXformNet. It includes K branches and two stages. Each branch generates a part of the target point cloud. The first stage uses AXform to generate a coarse point cloud. In the second stage, refinement is performed by combining input to generate a final point cloud.

propose a network based on AXform for supervised point cloud completion, called AXformNet.

**AXformNet Architecture** As shown in Figure 3, the input of AXformNet is partial point clouds generated by back-projecting 2.5D depth images into 3D and the output is completed point clouds. The framework of AXformNet is based on an autoencoder. To better demonstrate the superiority of our decoder, we simply design our encoder with a shared MLP layer $(3, 64, 128, 1024)$ and a max pooling layer, which is like what in PointNet (Qi et al. 2017a). The design of our decoder is based on AXform with K branches. The difference is that each branch contains two AXforms and an additional feature mapping module.

Since different branches focus on the generation of different parts of an object, the input feature of AXform in each branch should be different. Therefore, we add a feature mapping module before each branch to achieve adaptive feature transformation, which is a 4-layer fully connected network $(1024, 1024, 1024, 128)$.

Following the idea of considerable previous methods like (Liu et al. 2019a; Wang, Marcelo H., and Lee 2020), we also add a refinement network after the generation of coarse outputs. Specifically, the coarse output of each branch and the input partial point cloud are first concatenated after farthest point sampling (FPS) respectively. Then a bias calculated by AXform is added to generate the final output. Here we need to point out that for a fair comparison, we also combine the partial point cloud when performing the refinement. This operation contributes to the metric of Chamfer Distance a lot. Assuming that a partial point cloud occupies $\lambda$ of its corresponding ground truth point cloud in space, and their points are completely coincident. If $CD(complete, gt) = A$ and we replace a part of the complete point cloud with the original partial point cloud to get $complete'$, it is easy to infer that $CD(complete', gt) = (1 - \lambda)A$, which is a large improvement of the metric.

**Loss Functions** We use L1 Chamfer Distance (Yuan et al. 2018) to supervise the training process of AXformNet. $Y_{coarse}$, $Y_{final}$ and $Y_{gt}$ represent the coarse output point cloud, the final output point cloud, and the ground truth point cloud respectively. $\alpha$ is a weighting factor. The total loss for training is then given as:

$$\mathcal{L} = \alpha \mathcal{L}_{CD}(Y_{coarse}, Y_{gt}) + \mathcal{L}_{CD}(Y_{final}, Y_{gt}) \quad (1)$$

## Experiments

### Datasets and Implementation Details

We evaluate AXform on three representative categories in the ShapeNet (Chang et al. 2015) dataset: *airplane*, *car*, and *chair*. The point clouds are obtained by sampling points uniformly from the mesh surface. All the point clouds are then centered and scaled to [-0.5, 0.5]. We follow the train/-val/test split in ShapeNet official documents and use 2048 points for each shape during both training and testing. A simple shared MLP layer $(3, 64, 128, K_1)$ with a max pooling layer is used as the encoder. All the experiments are performed for 200 epochs with a batch size of 32. Adam

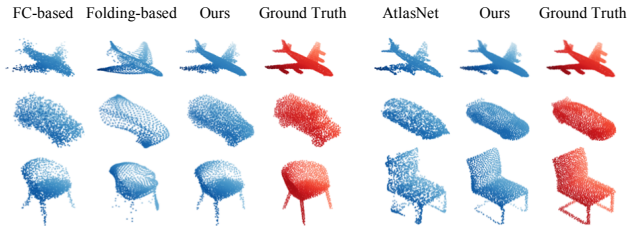
is used as the optimizer and the initial learning rate is 1e-4. The Chamfer Distance used in these experiments is L2 Chamfer Distance (Fan, Su, and Guibas 2017).

We also evaluate AXformNet on the PCN (Yuan et al. 2018) dataset for point cloud completion. It includes 30974 shapes with 8 categories. We set branch number $K = 16$ and train our method for 100 epochs with a batch size of 128. $\alpha$ increases from 0.01 to 1 within the first 25 epochs. Adam is used as the optimizer and the initial learning rate is 1e-3.

## Reconstruction and Generation

| #Branches | Category | Methods | $K_1$ | $K_2$ | $N$ | CD ↓ | Params. ↓ |
|---|---|---|---|---|---|---|---|
| K = 1 | Airplane | FC-based | 1024 | | | 7.895 | 7.4M |
| | | Folding-based | 512 | | | 9.208 | 1.1M |
| | | Ours | 128 | 32 | 128 | **4.386** | **0.8M** |
| | Car | FC-based | 1024 | | | 11.523 | 7.4M |
| | | Folding-based | 512 | | | 20.989 | 1.1M |
| | | Ours | 128 | 32 | 128 | **8.008** | **0.8M** |
| | Chair | FC-based | 1024 | | | 13.861 | 7.4M |
| | | Folding-based | 512 | | | 23.103 | 1.1M |
| | | Ours | 128 | 32 | 128 | **11.606** | **0.8M** |
| | All | FC-based | 1024 | | | 8.578 | 7.4M |
| | | Folding-based | 512 | | | 12.980 | 1.1M |
| | | Ours | 128 | 32 | 128 | **8.046** | **0.8M** |
| K = 16 | Airplane | AtlasNet | 1024 | | | 6.307 | 27.5M |
| | | Ours | 128 | 32 | 128 | **3.607** | **8.9M** |
| | Car | AtlasNet | 1024 | | | 15.269 | 27.5M |
| | | Ours | 128 | 32 | 128 | **7.670** | **8.9M** |
| | Chair | AtlasNet | 1024 | | | 17.154 | 27.5M |
| | | Ours | 128 | 32 | 128 | **9.423** | **8.9M** |
| | All | AtlasNet | 1024 | | | 11.057 | 27.5M |
| | | Ours | 128 | 32 | 128 | **6.867** | **8.9M** |

Table 1: Quantitative comparison of reconstruction results on our sampled ShapeNet dataset. CD is multiplied by $10^4$.




(a) AXform with one branch (b) AXform with K branches

Figure 4: Visualized comparison of reconstruction results on our sampled ShapeNet dataset. Trained on each category.

AXform with one branch is compared with FC-based and folding-based methods on our sampled ShapeNet dataset. According to FoldingNet (Yang et al. 2018), the input feature dimension $K_1$ of the FC-based method and the folding-based method is set to 1024 and 512 respectively. The folding operation is performed in two rounds. To show that our method can still have better reconstruction results even network parameters are fewer than previous methods, we set $K_1 = 128$, $K_2 = 32$, and $N = 128$. As shown in Table 1, Chamfer Distance of our method is much better than previous two methods regardless of training on a single category or all three categories. As shown in Figure 4(a), it can be found that the reconstruction results of our method are closer to the ground truth. In addition, the reconstruction details of our method are better, such as the engines and tail of the airplane.

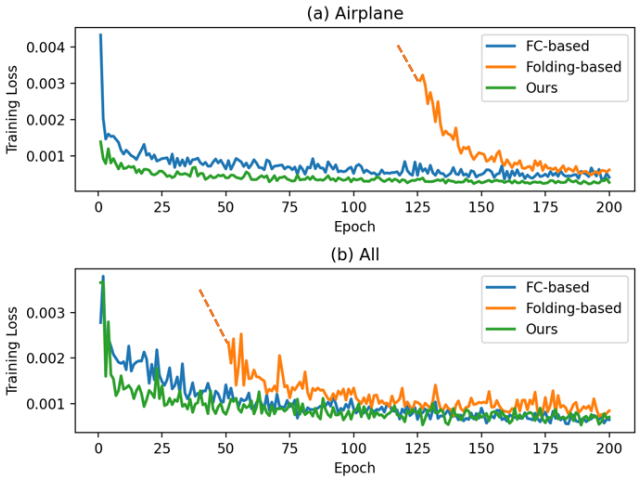



Figure 5: Loss curve during training.

| | #Branches | 2 | 4 | 8 | 16 | 32 |
|---|---|---|---|---|---|---|
| Fix $N = 128$ | CD | 4.280 | 4.041 | 3.840 | 3.607 | 3.520 |
| | Params. | 1.3M | 2.4M | 4.6M | 8.9M | 17.5M |
| Fix Params. ≈ 8.9M | CD | 3.844 | 3.774 | 3.724 | 3.607 | 3.643 |
| | $N$ | 1024 | 512 | 256 | 128 | 64 |

Table 2: Ablation studies for the branch number K. $K_1 = 128$ and $K_2 = 32$ are fixed. CD is multiplied by $10^4$. Trained on airplane category.

For AXform with K branches, we first do some ablation studies on the number of branches. As shown in Table 2, when the AXform in each branch is fixed, more branches will lead to better Chamfer Distance. However, the decline of Chamfer Distance decreases a lot when $K \geq 16$. When the network parameters are approximately kept unchanged,

| Methods | JSD ↓ | MMD ↓ | | COV %, ↑ | | 1-NNA %, ↓ | |
|---|---|---|---|---|---|---|---|
| | | CD | EMD | CD | EMD | CD | EMD |
| l-GAN (CD) | 7.24 | **0.454** | **4.43** | **33.66** | **25.70** | 63.00 | 81.23 |
| l-GAN-AXform | **6.27** | 0.498 | 4.54 | 33.33 | 24.59 | **61.59** | **78.55** |

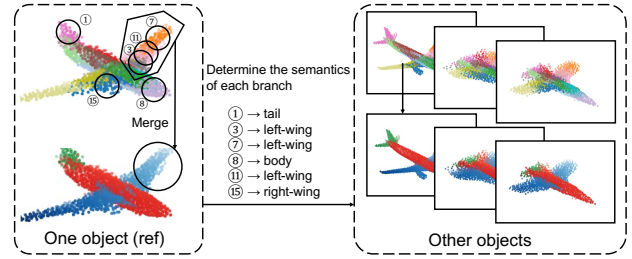
Table 3: Quantitative comparison of replacing the MLP decoder in l-GAN (CD) with our AXform. JSD, MMD-CD/EMD are multiplied by $10^2$, $10^2$, and $10^3$ respectively. Trained on airplane category.

| Methods | Airp. | Cab. | Car | Chair | Lamp | Couch | Table | Vessel | Avg. | Methods | Airp. | Cab. | Car | Chair | Lamp | Couch | Table | Vessel | Avg. |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FoldingNet | 9.49 | 15.80 | 12.61 | 15.55 | 16.41 | 15.97 | 13.65 | 14.99 | 14.31 | MSN | 5.60 | 11.96 | 10.78 | 10.62 | 10.71 | 11.90 | 8.70 | 9.49 | 9.97 |
| AtlasNet | 6.37 | 11.94 | 10.11 | 12.06 | 12.37 | 12.99 | 10.33 | 10.61 | 10.85 | GRNet | 6.45 | 10.37 | 9.45 | 9.41 | 7.96 | 10.51 | 8.44 | 8.04 | 8.83 |
| PCN | 5.50 | **10.63** | 8.70 | 11.00 | 11.34 | 11.68 | **8.59** | 9.67 | 9.64 | SpareNet | 5.96 | 12.57 | 9.96 | 11.93 | 11.11 | 13.39 | 9.95 | 9.59 | 10.56 |
| TopNet | 7.61 | 13.31 | 10.90 | 13.82 | 14.44 | 14.78 | 11.22 | 11.12 | 12.15 | PMP-Net | 5.65 | 11.24 | 9.64 | 9.51 | **6.95** | 10.83 | 8.72 | **7.25** | 8.73 |
| Ours(vanilla) | **5.37** | 10.68 | **8.65** | **10.74** | **10.46** | **11.68** | 8.73 | **9.30** | **9.45** | Ours | **4.76** | **10.18** | **8.60** | **9.13** | 8.17 | **10.40** | **7.75** | 7.80 | **8.35** |

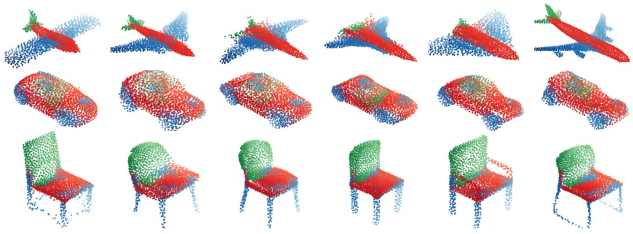
Table 4: Quantitative comparison between our methods and existing methods. The metric is CD, multiplied by $10^3$.

| Methods | Airp. | Cab. | Car | Chair | Lamp | Couch | Table | Vessel | Avg. | Methods | Airp. | Cab. | Car | Chair | Lamp | Couch | Table | Vessel | Avg. |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FoldingNet | 0.642 | 0.237 | 0.382 | 0.236 | 0.219 | 0.197 | 0.361 | 0.299 | 0.322 | MSN | 0.885 | **0.644** | 0.665 | 0.657 | 0.699 | 0.604 | 0.782 | 0.708 | 0.705 |
| AtlasNet | 0.845 | 0.552 | 0.630 | 0.552 | 0.565 | 0.500 | 0.660 | 0.624 | 0.616 | GRNet | 0.843 | 0.618 | 0.682 | 0.673 | 0.761 | 0.605 | 0.751 | 0.750 | 0.708 |
| PCN | 0.881 | **0.651** | 0.725 | 0.625 | 0.638 | **0.581** | **0.765** | 0.697 | 0.695 | SpareNet | 0.869 | 0.571 | 0.672 | 0.592 | 0.647 | 0.527 | 0.719 | 0.690 | 0.661 |
| TopNet | 0.771 | 0.404 | 0.544 | 0.413 | 0.408 | 0.350 | 0.572 | 0.560 | 0.503 | PMP-Net | 0.860 | 0.495 | 0.570 | 0.600 | 0.778 | 0.516 | 0.639 | 0.742 | 0.650 |
| Ours(vanilla) | **0.893** | 0.634 | **0.725** | **0.632** | **0.667** | 0.567 | 0.756 | **0.703** | **0.697** | Ours | **0.920** | 0.642 | **0.734** | **0.704** | **0.782** | **0.605** | **0.801** | **0.778** | **0.746** |

Table 5: Quantitative comparison between our methods and existing methods. The metric is F-Score@1%.




(a) Workflow



(b) Unsupervised semantic segmentation results

Figure 6: AXform with K branches has a property of space consistency which enables unsupervised semantic segmentation on the generated point clouds.

| Methods | Airp. | Cab. | Car | Chair | Lamp | Couch | Table | Vessel | Avg. |
|---|---|---|---|---|---|---|---|---|---|
| w/o fm | 4.98 | 10.32 | 8.75 | 9.73 | 8.85 | 10.75 | 8.10 | 8.24 | 8.71 |
| Ours | **4.76** | **10.18** | **8.60** | **9.13** | **8.17** | **10.40** | **7.75** | **7.80** | **8.35** |

(a) CD.

| Methods | Airp. | Cab. | Car | Chair | Lamp | Couch | Table | Vessel | Avg. |
|---|---|---|---|---|---|---|---|---|---|
| w/o fm | 0.912 | 0.629 | 0.717 | 0.671 | 0.747 | 0.584 | 0.782 | 0.758 | 0.725 |
| Ours | **0.920** | **0.642** | **0.734** | **0.704** | **0.782** | **0.605** | **0.801** | **0.778** | **0.746** |

(b) F-Score@1%.

Table 6: Ablation studies for the feature mapping module.

$K = 16$ obtains the optimal Chamfer Distance. Therefore, we choose $K = 16$ for the comparison experiments. Under the condition that network parameters are fewer than AtlasNet (Groueix et al. 2018), as shown in Table 1, our method can still achieve a lower Chamfer Distance. As shown in Figure 4(b), it can be found that the reconstruction results of our method are smoother and more even. For example, thin structures like chair legs are difficult to be deformed from 2d squares, so the result of AtlasNet is inferior to our method.

Figure 5 shows the loss curves of three methods during training. (a) and (b) represents the training process on a single airplane category and all three categories respectively. Since the loss value of the folding-based method is huge at the beginning, we use a dotted line to represent the huge value, which contributes to a clear comparison between the methods. It can be found that the convergence speed of AXform is significantly faster than the previous two methods, and our loss curve is relatively smoother.

AXform can be used on existing network architectures. For example, we replace the MLP decoder in l-GAN (CD) (Achlioptas et al. 2017) to get better generation results. The generation set is the same size as the test set. To reduce the sampling bias of the evaluation metrics in Table 3, the process is repeated 3 times and reported the averages. It can be found that l-GAN-AXform achieves better JSD and 1-NNA and comparable MMD and COV. Compared with other metrics, (Yang et al. 2019a) points out that 1-NNA is better suited for evaluating generative models of point clouds. Therefore, AXform improves the point cloud generation model l-GAN (CD).

## Unsupervised Semantic Segmentation

AXform with K branches can realize unsupervised semantic segmentation on the generated point clouds due to the property of space consistency. As shown in Figure 6(a), we give a realization workflow. When there are enough branches, the space consistency can be regarded as "semantic consis-

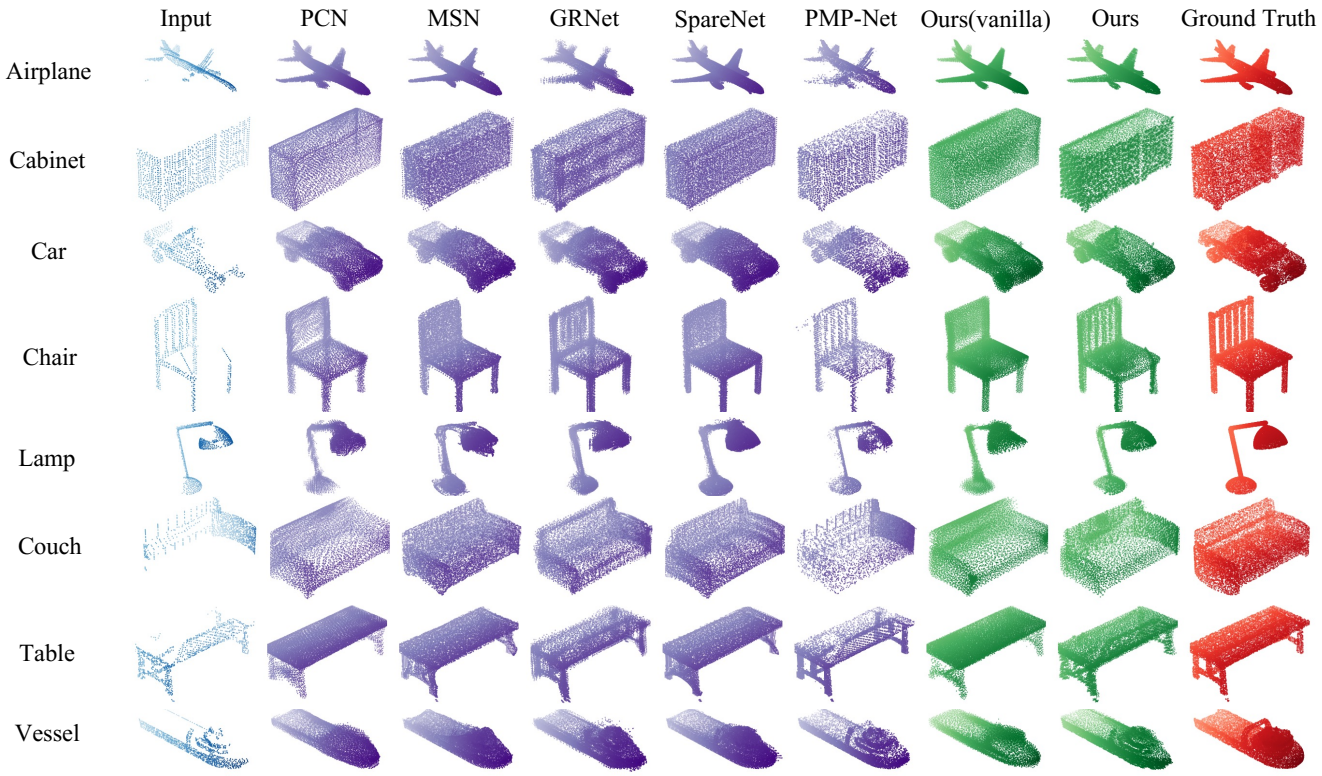



Figure 7: Visualized completion comparison on PCN dataset.

tency" which means each branch focuses on a certain semantics. Assume that the 16 branches are labeled 1-16. We take an airplane in the training set as a reference and find that branches 3, 4, and 7 generate the left-wing. Then, when reconstructing all the airplanes in the test set, branches 3, 4, and 7 will still generate the left-wing due to the property of space consistency. Finally, unsupervised semantic segmentation can be realized by merging the output of branches with the same semantics.

Figure 6(b) shows the unsupervised semantic segmentation results on three categories by using AXform with 16 branches. Here different colors represent different semantics. Airplane has three semantics of wing, body, and tail; Car has three semantics of wheel, body, and roof; Chair has three semantics of leg, surface, and back. It can be found that, except for some joints, the results are quite good.

### Point Cloud Completion

We compare AXformNet with previous methods on the PCN dataset. Chamfer Distance and F-Score@1% (Tatarchenko et al. 2019) are used as evaluation metrics. In the approach section, we have pointed out that keeping the partial point cloud unchanged contributes to the metrics a lot. For a fair comparison, we run a vanilla version of AXformNet, which does not include the second refinement stage.

The metrics of SpareNet (Xie et al. 2021) and PMP-Net (Wen et al. 2021) in Table 4 and Table 5 are obtained from their given pretrained model and the others are obtained from the paper GRNet (Xie et al. 2020). It can be found that AXformNet is the best in most categories and on average. The F-Score@1% which is more convincing than Chamfer Distance is greatly improved. Figure 7 shows the visualized completion comparison on the PCN dataset. AXformNet can generate better complete point clouds than the other methods by using AXform. Since AXformNet(vanilla) is the ablation of the refinement module, here we only do ablation studies for the feature mapping module. As shown in Table 6, the feature mapping module contributes to the improvement of the results. Without it, AXformNet can still behave well.

## Conclusion

Since the transformation from latent features to point clouds has not been fully explored, we propose a novel attention-based method called AXform. It generates point clouds by weighting the points in an interim space and achieves better results than previous FC-based and folding-based methods. In addition, AXform has properties of self-clustering and space consistency when been expanded to multiple branches, which can be used for unsupervised semantic segmentation. We apply AXform to point cloud completion and it achieves state-of-the-art results on the PCN dataset.

## Acknowledgments

This work was supported by National Natural Science Fund of China (62176064) and Zhejiang Lab (2019KD0AB06). Cheng Jin is the corresponding author.

## References


Achlioptas, P.; Diamanti, O.; Mitliagkas, I.; and Guibas, L. J. 2017. Learning Representations and Generative Models For 3D Point Clouds. *arXiv preprint arXiv:1707.02392*.

Alliegro, A.; Valsesia, D.; Fracastoro, G.; Magli, E.; and Tommasi, T. 2021. Denoise and Contrast for Category Agnostic Shape Completion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 4629–4638.

Chang, A. X.; Funkhouser, T.; Guibas, L.; Hanrahan, P.; Huang, Q.; Li, Z.; Savarese, S.; Savva, M.; Song, S.; Su, H.; Xiao, J.; Yi, L.; and Yu, F. 2015. ShapeNet: An Information-Rich 3D Model Repository. Technical Report 1512.03012, arXiv preprint.

Dai, A.; Ritchie, D.; Bokeloh, M.; Reed, S.; Sturm, J.; and Nießner, M. 2018. ScanComplete: Large-Scale Scene Completion and Semantic Segmentation for 3D Scans. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*.

Fan, H.; Su, H.; and Guibas, L. J. 2017. A Point Set Generation Network for 3D Object Reconstruction From a Single Image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Fuchs, F. B.; Worrall, D. E.; Fischer, V.; and Welling, M. 2020. SE(3)-Transformers: 3D Roto-Translation Equivariant Attention Networks. In *Advances in Neural Information Processing Systems 34 (NeurIPS)*.

Groueix, T.; Fisher, M.; Kim, V. G.; Russell, B.; and Aubry, M. 2018. AtlasNet: A Papier-Mâché Approach to Learning 3D Surface Generation. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Hu, Q.; Yang, B.; Xie, L.; Rosa, S.; Guo, Y.; Wang, Z.; Trigoni, N.; and Markham, A. 2020. RandLA-Net: Efficient Semantic Segmentation of Large-Scale Point Clouds. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Hu, T.; Han, Z.; Shrivastava, A.; and Zwicker, M. 2019. Render4Completion: Synthesizing Multi-View Depth Maps for 3D Shape Completion. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*.

Huang, Z.; Yu, Y.; Xu, J.; Ni, F.; and Le, X. 2020. PF-Net: Point Fractal Network for 3D Point Cloud Completion. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Hui, L.; Xu, R.; Xie, J.; Qian, J.; and Yang, J. 2020. Progressive Point Cloud Deconvolution Generation Network. In *ECCV*.

Kim, H.; Lee, H.; Kang, W. H.; Lee, J. Y.; and Kim, N. S. 2020. SoftFlow: Probabilistic Framework for Normalizing Flow on Manifolds. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M. F.; and Lin, H., eds., *Advances in Neural Information Processing Systems*, volume 33, 16388–16397. Curran Associates, Inc.

Klokov, R.; Boyer, E.; and Verbeek, J. 2020. Discrete Point Flow Networks for Efficient Point Cloud Generation. In *Proceedings of the 16th European Conference on Computer Vision (ECCV)*.

Lee, J.; Lee, Y.; Kim, J.; Kosiorek, A.; Choi, S.; and Teh, Y. W. 2019. Set Transformer: A Framework for Attention-based Permutation-Invariant Neural Networks. In *Proceedings of the 36th International Conference on Machine Learning*, 3744–3753.

Li, C.-L.; Zaheer, M.; Zhang, Y.; Poczos, B.; and Salakhutdinov, R. 2018. Point cloud gan. *arXiv preprint arXiv:1810.05795*.

Li, R.; Li, X.; Fu, C.-W.; Cohen-Or, D.; and Heng, P.-A. 2019. PU-GAN: a Point Cloud Upsampling Adversarial Network. In *IEEE International Conference on Computer Vision (ICCV)*.

Liu, M.; Sheng, L.; Yang, S.; Shao, J.; and Hu, S.-M. 2019a. Morphing and Sampling Network for Dense Point Cloud Completion. *arXiv preprint arXiv:1912.00280*.

Liu, X.; Han, Z.; Wen, X.; Liu, Y.-S.; and Zwicker, M. 2019b. L2G Auto-encoder: Understanding Point Clouds by Local-to-Global Reconstruction with Hierarchical Self-Attention. In *Proceedings of the 27th ACM International Conference on Multimedia*.

Luo, S.; and Hu, W. 2021. Diffusion Probabilistic Models for 3D Point Cloud Generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Pan, L. 2020. ECG: Edge-aware Point Cloud Completion with Graph Convolution. *IEEE Robotics and Automation Letters*, 5(3): 4392–4398.

Pan, L.; Chen, X.; Cai, Z.; Zhang, J.; Zhao, H.; Yi, S.; and Liu, Z. 2021. Variational Relational Point Completion Network. *arXiv preprint arXiv:2104.10154*.

Qi, C. R.; Su, H.; Mo, K.; and Guibas, L. J. 2017a. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Qi, C. R.; Yi, L.; Su, H.; and Guibas, L. J. 2017b. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. *arXiv preprint arXiv:1706.02413*.

Rui, F.; Plinio, M.; and Alexandre, B. 2017. Automatic Object Shape Completion from 3D Point Clouds for Object Manipulation. In *Proceedings of the 12th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 4: VISAPP, (VISIGRAPP 2017)*, 565–570. INSTICC, SciTePress. ISBN 978-989-758-225-7.

Sarmad, M.; Lee, H. J.; and Kim, Y. M. 2019. RL-GAN-Net: A Reinforcement Learning Agent Controlled GAN Network for Real-Time Point Cloud Shape Completion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Shajahan, D. A.; Nayel, V.; and Muthuganapathy, R. 2020. Roof Classification From 3-D LiDAR Point Clouds Using Multiview CNN With Self-Attention. *IEEE Geoscience and Remote Sensing Letters*, 17(8): 1465–1469.

Shu, D. W.; Park, S. W.; and Kwon, J. 2019. 3D Point Cloud Generative Adversarial Network Based on Tree Structured Graph Convolutions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

Sun, Y.; Wang, Y.; Liu, Z.; Siegel, J. E.; and Sarma, S. E. 2020. PointGrow: Autoregressively Learned Point Cloud Generation with Self-Attention. In *Winter Conference on Applications of Computer Vision*.

Tatarchenko, M.; Richter, S. R.; Ranftl, R.; Li, Z.; Koltun, V.; and Brox, T. 2019. What Do Single-View 3D Reconstruction Networks Learn? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Tchapmi, L. P.; Kosaraju, V.; Rezatofighi, S. H.; Reid, I.; and Savarese, S. 2019. TopNet: Structural Point Cloud Decoder. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Valsesia, D.; Fracastoro, G.; and Magli, E. 2019. Learning Localized Generative Models for 3D Point Clouds via Graph Convolution. In *International Conference on Learning Representations*.

Varley, J.; DeChant, C.; Richardson, A.; Nair, A.; Ruales, J.; and Allen, P. 2017. Shape Completion Enabled Robotic Grasping. In *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*. IEEE.

Wang, X.; Marcelo H., A. J.; and Lee, G. H. 2020. Cascaded Refinement Network for Point Cloud Completion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Wang, Y.; Sun, Y.; Liu, Z.; Sarma, S. E.; Bronstein, M. M.; and Solomon, J. M. 2019. Dynamic Graph CNN for Learning on Point Clouds. *ACM Transactions on Graphics (TOG)*.

Wang, Y.; Tan, D. J.; Navab, N.; and Tombari, F. 2020. SoftPoolNet: Shape Descriptor for Point Cloud Completion and Classification. *CoRR*, abs/2008.07358.

Wen, X.; Li, T.; Han, Z.; and Liu, Y.-S. 2020. Point Cloud Completion by Skip-Attention Network With Hierarchical Folding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Wen, X.; Xiang, P.; Han, Z.; Cao, Y.-P.; Wan, P.; Zheng, W.; and Liu, Y.-S. 2021. PMP-Net: Point cloud completion by learning multi-step point moving paths. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Xie, C.; Wang, C.; Zhang, B.; Yang, H.; Chen, D.; and Wen, F. 2021. Style-Based Point Generator With Adversarial Rendering for Point Cloud Completion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 4619–4628.

Xie, H.; Yao, H.; Zhou, S.; Mao, J.; Zhang, S.; and Sun, W. 2020. GRNet: Gridding Residual Network for Dense Point Cloud Completion. In *ECCV*.

Yang, G.; Huang, X.; Hao, Z.; Liu, M.-Y.; Belongie, S.; and Hariharan, B. 2019a. PointFlow: 3D Point Cloud Generation With Continuous Normalizing Flows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

Yang, J.; Zhang, Q.; Ni, B.; Li, L.; Liu, J.; Zhou, M.; and Tian, Q. 2019b. Modeling Point Clouds With Self-Attention and Gumbel Subset Sampling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Yang, X.; Wu, Y.; Zhang, K.; and Jin, C. 2021. CPCGAN: A Controllable 3D Point Cloud Generative Adversarial Network with Semantic Label Generating. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(4): 3154–3162.

Yang, Y.; Feng, C.; Shen, Y.; and Tian, D. 2018. FoldingNet: Point Cloud Auto-Encoder via Deep Grid Deformation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Yuan, W.; Khot, T.; Held, D.; Mertz, C.; and Hebert, M. 2018. PCN: Point Completion Network. In *International Conference on 3D Vision (3DV)*.

Zamorski, M.; Zieba, M.; Klukowski, P.; Nowak, R.; Kurach, K.; Stokowiec, W.; and Trzciński, T. 2018. Adversarial Autoencoders for Compact Representations of 3D Point Clouds. *arXiv preprint arXiv:1811.07605*.

Zhang, G.; Ma, Q.; Jiao, L.; Liu, F.; and Sun, Q. 2020. AttAN: Attention Adversarial Networks for 3D Point Cloud Semantic Segmentation. In Bessiere, C., ed., *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, 789–796. International Joint Conferences on Artificial Intelligence Organization. Main track.

Zhang, W.; Yan, Q.; and Xiao, C. 2020. Detail Preserved Point Cloud Completion via Separated Feature Aggregation. In *Computer Vision – ECCV 2020*.

Zhao, H.; Jiang, L.; Jia, J.; Torr, P.; and Koltun, V. 2020. Point Transformer. arXiv:2012.09164.