Point Cloud Super Resolution with Adversarial Residual Graph Networks

Huikai Wu
http://wuhuikai.me/
Kaiqi Huang
kaiqi.huang@nlpr.ia.ac.cn

Institute of Automation
Chinese Academy of Sciences

Abstract

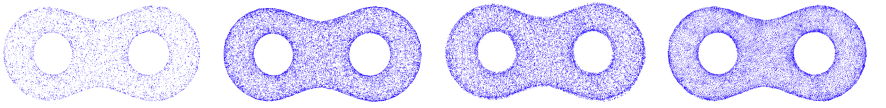
Point cloud super-resolution aims at transforming a low-resolution (LR) point cloud into a high-resolution (HR) one, which is important for 3D reconstruction and point cloud understanding. In this paper, we focus on improving the performance of existing point cloud super-resolution algorithms. The key idea is to exploit the local similarity of point cloud and the analogy between the LR input and the HR output. For the former, we design a deep graph convolutional network. For the latter, we propose residual graph convolution blocks and introduce a skip connection between the input and the output. Our network is trained with a novel loss function, graph adversarial loss, which can capture the characteristics of HR point clouds automatically. Experiments show that our method achieves state-of-the-art performance and generalizes well to unseen data.

1 Introduction

When modeling an object from the real world for 3D printing or animation, a common way is to first obtain the point cloud with depth scanning devices or 3D reconstruction methods [6, 7] and then recover the mesh from the point cloud [3]. However, the captured point cloud is usually sparse and noisy due to the restrictions of devices or the limitations of methods, which leads to a low-quality mesh.

A widely-used method to improve the quality of the recovered mesh is point cloud super-resolution, which takes a LR point cloud as the input and generates a HR point cloud with richer details and fewer noisy points, as shown in Figure 1. In point cloud super-resolution, the most powerful method is PU-Net [25], which is a data-driven algorithm based on deep learning. As the first deep learning based method, PU-Net outperforms many traditional methods such as EAR [8] and achieves state-of-the-art performance.

In this paper, we aim to advance the performance of existing point cloud super-resolution algorithms by overcoming the defects of PU-Net. The first problem is that PU-Net directly regresses the point coordinates based on PointNet++ [18] without exploiting the local similarity of point cloud and the analogy between the LR input and HR output. The second problem is that PU-Net designs a complicated loss function manually with a strong assumption on the uniform distribution of HR point clouds. Manually designed losses tend to overfit human priors, which fail to capture many other properties of HR point clouds like continuity.



(a) Input (b) GT (c) PU-Net [25] (d) Ours

Figure 1: **Point Cloud Super Resolution.** (a) is the input LR point cloud with 5,000 points. (b) is the HR point cloud with 20,000 points. (c) and (d) are the HR point clouds generated by PU-Net [25] and our method. Ours is sharper at edges while having fewer noisy points.

To solve the first problem, we design a deep graph convolutional network (GCN) for exploiting the local similarity of point cloud. To exploit the analogy between the LR and HR point cloud, we introduce residual connections for graph convolutions, resulting in residual graph convolution blocks. Besides, we also add a skip connection between the input and the output. To solve the second problem, we propose a graph adversarial loss. The proposed loss function is more expressive than manually designed ones, which can capture the characteristics of HR point clouds automatically.

In this way, we propose a novel method for point cloud super-resolution, named Adversarial Residual Graph Convolutional Network (AR-GCN). Experiments show that AR-GCN achieves state-of-the-art performance on both seen dataset [25] and unseen dataset (SHREC15). The contributions of our method are three-folds. First, we propose a novel architecture for point cloud super-resolution. Second, we introduce the graph adversarial loss to replace manually designed loss functions. Third, we advance the state-of-the-art performance on both seen and unseen datasets.

2 Related Work

Point Cloud Super Resolution In most earlier works, point cloud super-resolution is formulated as an optimization problem. [2] first compute the Voronoi diagram on the moving least squares surface and then add points at the vertices of the Voronoi diagram. [13] present a locally optimal projection operator for surface approximation based on L_1 median, which is parameter-free and robust to noisy points. The above methods have a strong assumption on the smoothness of the underlying surface, which tend to generate vague edges. [8] introduce an edge-aware method, which first samples away from the edges and then progressively samples the point cloud while approaching the edge singularities.

Different from optimization based methods, [25] propose a data-driven method, PU-Net, to exploit massive 3D data. Concretely, PU-Net first learns multi-level features per point with PointNet++ [18] and then expands the point set via a multi-branch convolution. Through end-to-end learning, PU-Net outperforms optimization based methods on multiple datasets, achieving state-of-the-art performance. Following PU-Net, [24] advances the performance with extra edge annotations. 3PU [23] proposes a progressive upsampling strategy, resulting in a better performance than PU-Net. PU-GAN [10] proposes an adversarial loss to replace the manually designed loss functions, which is the current art of point cloud super resolution.

Deep Learning for 3D Data [15, 19, 22] employ 3D CNNs to process 3D data in volumetric grids. Differently, [11, 17, 18] process point clouds directly instead of volumetric grids.

[21, 26] employ GCNs [5] for point cloud classification. [16] design a generative adversarial network (GAN) for learning graph embeddings.

3 Method

3.1 Point Cloud Super Resolution

Given a point cloud x with shape $n \times 3$, the goal of point cloud super-resolution is to generate a point cloud $\hat{y}$ with shape $N \times 3$ ($N = \gamma n, \gamma > 1$). Each point of $\hat{y}$ lies on the underlying surface described by x, as shown in Figure 1.

3.2 Method Overview: AR-GCN

AR-GCN is a novel approach that contains three major components: the adaptive adversarial loss L_G, the graph generator G, and the graph discriminator D. As shown in Figure 2.a, the LR point cloud x is directly fed into G to generate the HR output $\hat{y}$. Then, $\hat{y}$ is sent into D to produce L_G, while another loss L_{cd} is calculated given $\hat{y}$ and the ground truth y.

AR-GCN consists of two networks, the generator G and the discriminator D. G aims to generate the HR point cloud by upsampling the LR input progressively, while D is responsible for distinguishing the fake HR point cloud ($\hat{y}$) from the real one (y). To train G and D simultaneously, we propose a joint loss function as shown in Equation 1:

$$L(x,y) = \lambda L_{cd}(G(x),y) + L_G(G(x)), \tag{1}$$

where λ controls the balance between L_{cd} and L_G. L_{cd} measures the distance from y to $\hat{y}$,

$$L_{cd}(\hat{y},y) = \Sigma_{p\in y} min_{q\in\hat{y}} ||p-q||_2^2, \tag{2}$$

which is a simplified version of Chamfer Distance, measuring how well $\hat{y}$ covers y.

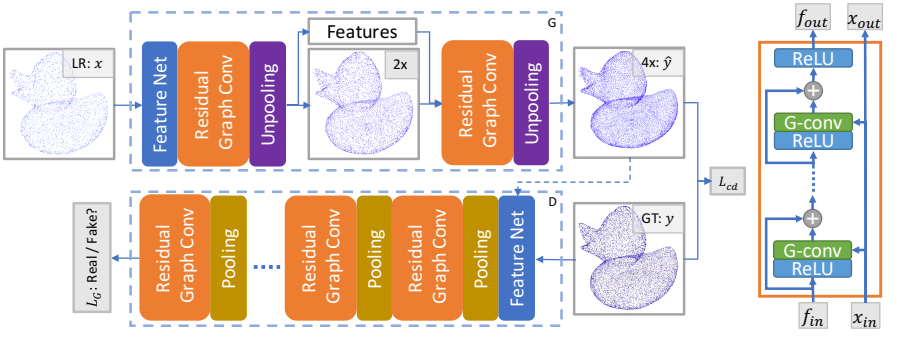
However, L_{cd} only measures the point-wise distance between y and $\hat{y}$, which ignores high-order properties of HR point clouds, such as continuity. Existing methods like PU-Net usually manually design a complex loss function to capture high-order properties, which is inefficient and has strong assumptions about HR point clouds based on human priors. Alternatively, we propose a loss function L_G that is defined by the graph discriminator D, which can automatically capture such properties by learning from data. Concretely, L_G is a graph adversarial loss that is adapted from LS-GAN [14],

$$L_G(\hat{y}) = \max_D ||1 - D(\hat{y})||_2^2 + ||D(y)||_2^2. \tag{3}$$

Different from LS-GAN, D in this paper is a GCN designed for point clouds, while that in LS-GAN is a CNN designed for image data.

3.3 Graph Generator

As shown in Figure 2.a, the graph generator G consists of three novel building blocks, namely residual graph convolution block (R-GCB), unpooling block, and feature net.



(a) AR-GCN (b) R-GCB

Figure 2: (a) is the proposed AR-GCN consisting of a graph generator G, a graph discriminator D, a graph adversarial loss L_G, and a point-wise loss L_{cd}. Specially, G upsamples the input point cloud progressively. (b) is Residual Graph Convolution Block (R-GCB), which is noted as Residual Graph Conv in (a).

Residual Graph Convolution Block The core of R-GCB (Figure 2.b) is the graph convolution layer, G-conv, which is defined on a graph G=(υ, ε) and calculated as follows,

$$f_{l+1}^p = w_0 f_l^p + w_1 \Sigma_{q \in N(p)} f_l^q, \forall p \in \upsilon. \tag{4}$$

w is the learnable parameters and f_l^p is the feature of vertex p at layer l. $N(p)$ is the vertices that directly connect to p as defined in the adjacency matrix ε. Point clouds do not have a predefined adjacency matrix. Instead, we define $N(p)$ as the k nearest neighbors of p in point cloud x_{in}. Besides G-conv, we also introduce residual connections into R-GCB. It helps to exploit the similarity between LR and HR point cloud.

Unpooling Block The unpooling block takes the point cloud x_{in} and the corresponding feature f_{in} as inputs. It first employs a G-conv layer to transform f_{in} with shape $\hat{n} \times c$ to a tensor with shape $\hat{n} \times 6$. The tensor is then reshaped to $\hat{n} \times 2 \times 3$, noted as δx. The upsampled point cloud x_{out} is obtained by adding x_{in} and δx pointwisely, where each point is transformed into 2 points. The unpooling block is designed to predict the residual between x_{in} and x_{out} instead of regressing x_{out} directly, which exploits the similarity between x_{in} and x_{out}, leading to a better performance. The corresponding output feature f_{out} is obtained as follows, where $N[x_{in}](p)$ means the k nearest neighbors of point p in point cloud x_{in}.

$$f_{out}^p = \frac{1}{k} \Sigma_{q \in N[x_{in}](p)} f_{in}^q, \forall p \in x_{out}. \tag{5}$$

Feature Net As shown in Figure 2.b, R-GCB takes both the point cloud and the corresponding feature as inputs. Given the input point cloud x, we design a simple block named feature net to generate the corresponding feature f. Specifically, for each point $p \in x$ with shape 1×3, we first obtain its k nearest neighbors P with shape $k \times 3$. Then, several point-wise convolutions and a max-pooling layer transform $\hat{P} = P - p$ into f^p with shape $1 \times c$.

Progressive Super Resolution Instead of directly upsampling the LR point cloud with the desired upscale ratio, we generate the HR point cloud step by step. The point cloud is upsampled by $2\times$ at each step, as shown in Figure 2.a. Experiment shows that such an approach results in better accuracy.

3.4 Graph Discriminator

As shown in Figure 2.a, D is composed of a feature net, R-GCBs, and pooling blocks.

Pooling Block Given the input point cloud x_{in} with shape $4n \times 3$, we first employ farthest point sampling (FPS) to generate x_{out} with shape $n \times 3$. The corresponding feature f_{out} is then obtained by $f_{out}^p = max_{q \in N[x_{in}](p)} f_{in}^q, \forall p \in x_{out}$, where max is applied elementwisely.

4 Experiment

In this section, we first introduce the datasets and the implementation details. We then present the quantitative and qualitative results to show the effectiveness of our method. To demonstrate the effect of each proposed component in AR-GCN, we also conduct a comprehensive ablation study. Finally, we show several potential applications of our method.

4.1 Datasets

Two datasets are employed in the experiments. One is the train-test dataset, which our method is trained with and tested on. The other is the unseen dataset, where our method is directly tested without training or finetuning.

Train-Test Dataset The dataset proposed in PU-Net [25] is used for training and testing, which contains 60 models from the Visionair repository. Following the protocol in PU-Net, we use 40 models for training and the rest 20 models for testing. For training, we extract 100 patches from each model as the ground truth. Each patch contains $4,096$ points. The input patch is generated by randomly sampling $1,024$ points from the ground truth patch. For testing, we sample $20,000$ points uniformly per model as the ground truth, while sampling $5,000$ points as the input.

Unseen Dataset: SHREC15 To validate the generalization ability, we directly test AR-GCN on SHREC15 [12] after training with the train-test dataset without finetuning. There are 50 categories in SHREC15 and 24 models in each category. We randomly choose one model from each category for testing, since the models in each category only differ in poses. We sample $20,000$ points per model as ground truth and $5,000$ points as input.

4.2 Implementation Details

Our method is implemented in Tensorflow [1]. To avoid overfitting, the training data is augmented with random rotation, shift, and scaling. Adam [9] is employed for optimization, where batch size is 28 and learning rate is 0.001. The network is first trained with L_{cd} for 80 epochs and then finetuned with L for another 40 epochs. As for the network architecture, the number of neighbors k is set to 8. The number of channels c in G is set to 128 and that in D is set to 64. Each R-GCB in G contains 12 G-convs while that in D contains 4 G-convs.

Dataset	Method	CD	EMD	F-score	NUC with different p			Deviation
				$\tau = 0.01$	0.2%	0.4%	0.8%	mean
Train-Test	Input	0.0120	0.0036	41.33%	0.315	0.224	0.163	-
	MLS	0.0117	0.0043	57.70%	0.364	0.272	0.204	**0.18**
	PU-Net	0.0118	0.0041	43.24%	0.206	0.165	0.137	0.78
	PU-GAN	**0.0081**	0.0050	**73.53%**	**0.202**	0.170	0.150	0.32
	AR-GCN	0.0084	**0.0035**	70.28%	0.204	**0.164**	**0.134**	0.26
SHREC15	Input	0.0077	**0.0031**	27.86%	0.310	0.220	0.163	-
	MLS	0.0067	0.0032	84.69%	0.253	0.199	0.159	0.33
	PU-Net	0.0103	0.0050	56.39%	0.283	0.230	0.189	0.90
	PU-GAN	**0.0054**	0.0051	92.08%	0.214	0.184	0.166	0.24
	AR-GCN	**0.0054**	**0.0031**	**93.07%**	**0.201**	**0.162**	**0.135**	**0.18**

Table 1: **Quantitative Comparison on the Train-Test Dataset and SHREC15 Dataset.**

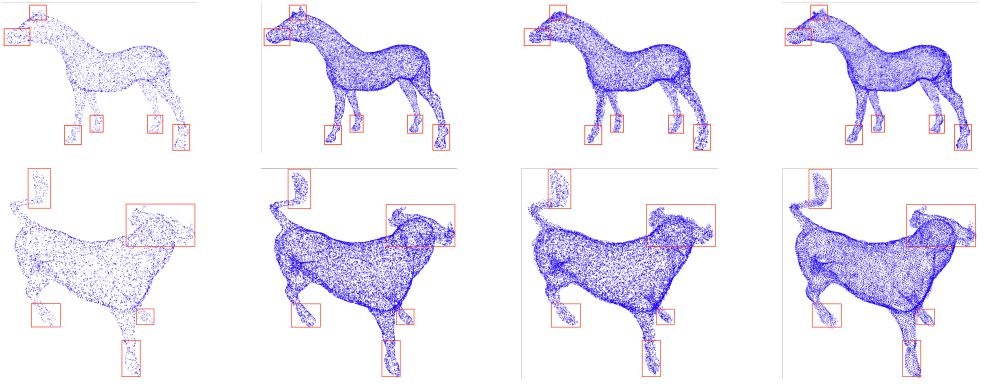
4.3 Quantitative Results

Evaluation Metrics To measure the difference between $\hat{y}$ and y pointwisely, Chamfer Distance (CD) and Earth Mover's Distance (EMD) are employed, for which the smaller is the better. F-score [20] is also used as the metric, which treats point cloud super-resolution as a classification problem. Concretely, precision and recall are first obtained by checking the percentage of points in $\hat{y}$ or y that can find a neighbor from the other within a certain threshold τ. F-score is then calculated as the harmonic mean of precision and recall. For F-score, the larger is the better. The metrics in [25] are also utilized. Deviation is used to measure the difference between $\hat{y}$ and the ground truth mesh, while normalized uniformity coefficient (NUC) is for measuring the uniformity. For Deviation and NUC, the smaller is the better.

Quantitative Comparison Results on train-test dataset and SHREC15 are reported in Table 1. The LR input is used as the preliminary baseline. For traditional methods, we choose Moving Least Squares (MLS) [2] as the baseline.[1] As for deep learning based methods, PU-Net and PU-GAN serve as the baselines, which are the best existing algorithms.[2] As shown in Table 1, our method outperforms all the other methods on both datasets in most metrics. In train-test dataset, our method outperforms PU-GAN in 4 metrics. Compared to MLS and PU-Net, our method improves the F-score by more than 10% and advances CD a large step. Surprisingly, our method even outperforms PU-Net in NUC, although it is not trained with the repulsion loss in PU-Net [25] that forces the generated point cloud uniform. We attribute this to the effect of the proposed adversarial loss. In SHREC15, our method outperforms all the other methods in all metrics by a large margin, which shows the generalization ability of our method on the unseen dataset.

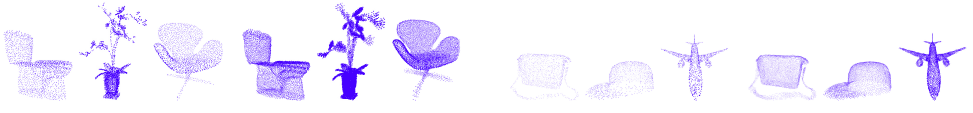
Experiments on Large Datasets We directly test our model on the test set of two large 3D datasets: ModelNet40 [22] and ShapeNet [4] without retraining or fine-tuning. 512/640 points are sampled as the input and the original point clouds serve as the ground truth. Experiments show that our method (33.37%/56.89%) outperforms PU-Net (32.48%/36.53%) by a large margin in F-score on both datasets.

[1] The result of MLS is produced by Point Cloud Library (PCL).
[2] The results are reproduced with the official code by following the authors' instructions.



(a) Input (b) GT (c) PU-Net (d) Ours

Figure 3: **Qualitative Results.** (a) is the LR input point cloud and (b) is the corresponding HR point cloud. (c) and (d) are the HR point clouds generated by PU-Net [25] and our method. Ours are sharper at edges, which have richer details but fewer noisy points, especially inside the red boxes.



(a) Input-ModelNet40 (b) Ours-ModelNet40 (c) Input-ShapeNet (d) Ours-ShapeNet

Figure 4: **Visual results on ModelNet40 and ShapeNet with 2,048 points as input.**

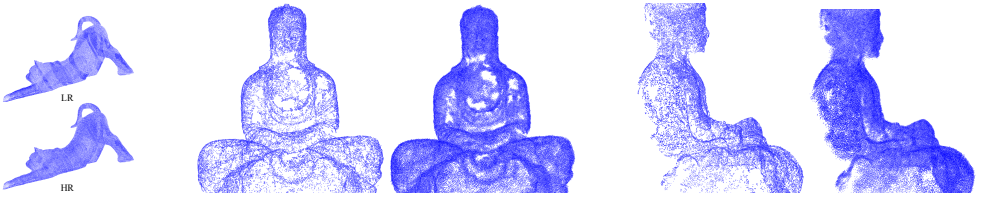
4.4 Qualitative Results

Figure 3 presents the point clouds in both datasets visually, where the 1st row is from the train-test dataset and the 2nd row is from SHREC15. Compared to PU-Net, the HR point clouds of our method have richer details and sharper edges. Besides, ours are more uniformly distributed in the smooth area with fewer noisy points. On the contrary, the results from PU-Net are noisy and blurry on the edges. It tends to outspread uniformly without preserving the underlying surface. The differences around the legs, feet and horns are most significant, as shown in the red boxes. Figure 4 shows the visual results on ModelNet40 and ShapeNet. Our method can generate a denser point cloud while preserving the underlying structure.

Real-Scanned Point Cloud We also conduct an experiment on real-scanned and un-even point clouds. As shown in Figure 5, our method can generate a denser point cloud with more uniformly distributed points while maintains the underlying surface. It can also fill small holes by uniformly adding points in sparse regions.

4.5 Ablation Study

The ablation study is shown in Table 2, which verifies the effect of each proposed component. **Residual Prediction:** Compared to $\text{GCN}^{4\times}_{points}$, $\text{GCN}^{4\times}$ adds a skip connection between input and output, which forces the model to predict the residual δx instead of directly regress-



(a) (b) LR (c) HR (d) LR (e) HR

Figure 5: Visual results on real-scanned and un-even point clouds.

Method	Train-Test Dataset			SHREC15		
	CD	F-score	Deviation	CD	F-score	Deviation
$GCN^{4\times}_{points}$	0.0971	3.96%	13.7	0.0925	8.03%	10.3
$GCN^{4\times}$	0.0092	63.79%	0.31	0.0061	87.98%	0.24
$ResGCN^{4\times}$	0.0090	65.04%	0.36	0.0059	90.57%	0.25
PU-Net [25]	0.0118	43.24%	0.78	0.0103	56.39%	0.90
ResGCN	0.0086	68.75%	0.30	0.0056	92.32%	0.21
ResPoint	0.0121	41.45%	0.85	0.0076	78.28%	0.56
ResGCN + L_{pu}	0.0096	60.71%	0.36	0.0064	86.46%	0.26
AR-GCN w/o FT	0.0085	69.53%	0.27	0.0055	92.46%	**0.18**
AR-GCN	**0.0084**	**70.28%**	**0.26**	**0.0054**	**93.07%**	**0.18**

Table 2: **Ablation Study on the Train-Test Dataset and SHREC15 Dataset.**

ing the point coordinates. The skip connection largely improves the stability of learning, resulting in good performance. Without the skip connection, the model does not work at all.

Residual G-conv: Compared to $GCN^{4\times}$, $ResGCN^{4\times}$ employs residual graph conv instead of graph conv. The residual connection introduces a better initialization, making the optimization more stable, which improves F-score by about 1.5% and 3% on both datasets.

Progressive Upsampling: Compared to $ResGCN^{4\times}$, ResGCN introduces progressive super-resolution. As a result, the F-score increases by nearly 4% on the train-test dataset. The Deviation also improves a lot on both datasets.

G-conv: Different from ResGCN, ResPoint employs the building block of PointNet [17] to replace G-conv (Figure 2.b). Since G-conv has a better exploitation of neighbor points, ResGCN significantly outperforms ResPoint.

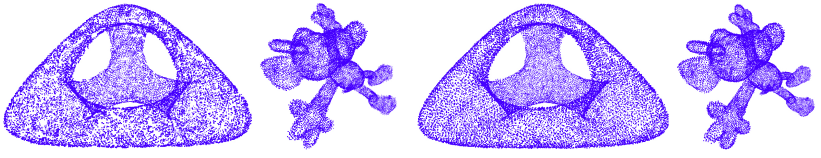
Loss Function: (1) By changing L_{cd} into the loss proposed in [25], the F-score decreases by about 8% as shown by ResGCN + L_{pu} and ResGCN. (2) With the adversarial loss, AR-GCN achieves a better Deviation than ResGCN. As shown in Figure 6, by adding the adversarial loss (L_G), the points distribute more uniformly compared to that without the adversarial loss.

ResGCN: The difference between PU-Net and ResGCN + L_{pu} is the whole network architecture. ResGCN + L_{pu} improves the F-score by more than 17%, which shows the effectiveness of the proposed ResGCN.

Training Strategy: AR-GCN w/o FT is trained with $L(x,y)$, while AR-GCN is trained with L_{cd} firstly and then fine-tuned with $L(x,y)$. Results show that AR-GCN outperforms AR-GCN w/o FT consistently, which demonstrates the superiority of the 2-step training strategy.

Method	Train-Test			SHREC15		
	F-score	NUC	Deviation	F-score	NUC	Deviation
PU-Net	43.2%	0.131	0.78	56.4%	0.180	0.90
Ours	70.3%	0.128	0.26	93.1%	0.130	0.18
Ours+noisy	55.4%	0.128	0.59	78.8%	0.130	0.55
Ours+uneven	46.0%	0.202	0.56	75.5%	0.201	0.37

Table 3: **Experiments on noisy data and uneven data.**



(a) w/o L_G (b) w/ L_G

Figure 6: **Visual results of the proposed adversarial loss.**

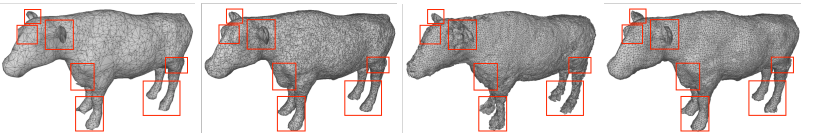
4.6 Robustness of AR-GCN

To show the robustness of our method, we take experiments on noisy point clouds and non-uniformly sampled point clouds. The quantitative results on the Train-Test dataset and SHREC15 are shown in Table 3. Ours+noisy represents applying AR-GCN on noisy point clouds (Gaussian noise z, where $z \sim \mathcal{N}(0, 0.01)$). Results show that our method with noisy point clouds outperforms PU-Net with clean ones. Ours+uneven means employing our method on non-uniformly sampled point clouds. Our method with uneven point clouds achieves similar performance to PU-Net with uniform ones on all the metrics except NUC. We attribute this to the non-uniform distribution of the input point clouds.

4.7 Applications

3D Reconstruction: In 3D reconstruction, the quality and density of the point cloud have a huge impact on the quality of the reconstructed mesh. Due to the limitations of scanning devices, the point cloud is usually sparse and noisy. Thus, point cloud super-resolution is important for improving the quality of 3D reconstruction. We employ our method and PU-Net to generate the HR point cloud, which is then fed into Ball pivoting algorithm [3] for mesh reconstruction. As shown in Figure 7, the mesh reconstructed from our method contains richer details compared to that from the LR input. Besides, ours is smoother in the flat area and sharper at the edges, while the mesh from PU-Net is noisy with unpleasant artifacts, especially inside the red boxes.

LR Point Cloud Classification: The classification accuracy on the LR point cloud is usually lower than that on the HR point cloud. To improve the performance of LR point cloud classification, one possible way is to upsample the LR point cloud into a HR point cloud before classification. In the experiment, we employ PointNet++ [18] on ModelNet40 dataset [22] for classification. PointNet++ achieves 91.05% (Acc) with $1,024$ points as input. When we randomly sample 256 points as the input, the performance drops from 91.05% to 46.96%. By upsampling the 256 points $4\times$ with our method, PointNet++ achieves 79.34% in Acc,



(a) Input (b) GT (c) PU-Net [] (d) Ours

Figure 7: **Mesh Reconstruction from Point Cloud.**

which outperforms 46.96% by a large margin. This experiment shows that point cloud super-resolution is important for understanding LR point cloud.

5 Conclusion

We proposed a graph convolutional network AR-GCN for point cloud super-resolution, which is composed of a graph generator, a graph discriminator, and a graph adversarial loss. With comprehensive experiments, we demonstrated that residual connections and residual prediction are effective for stable training and better performance. With the proposed graph adversarial loss, our method generates more realistic HR point cloud compared to the manually designed loss function. The experiment on train-test dataset showed that our method outperforms other methods. The experiment on the unseen dataset SHREC15 further demonstrated the better performance and generalization ability of our method.

References

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: a system for large-scale machine learning. In *OSDI*, 2016.

[2] Marc Alexa, Johannes Behr, Daniel Cohen-Or, Shachar Fleishman, David Levin, and Claudio T Silva. Computing and rendering point set surfaces. *IEEE Transactions on visualization and computer graphics*, 2003.

[3] Fausto Bernardini, Joshua Mittleman, Holly Rushmeier, Cláudio Silva, and Gabriel Taubin. The ball-pivoting algorithm for surface reconstruction. *IEEE transactions on visualization and computer graphics*, 1999.

[4] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An Information-Rich 3D Model Repository. Technical Report arXiv, 2015.

[5] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *NIPS*, 2016.

[6] Haoqiang Fan, Hao Su, and Leonidas J Guibas. A point set generation network for 3d object reconstruction from a single image. In *CVPR*, 2017.

[7] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[8] Hui Huang, Shihao Wu, Minglun Gong, Daniel Cohen-Or, Uri Ascher, and Hao Richard Zhang. Edge-aware point set resampling. *TOG*, 2013.

[9] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980*, 2014.

[10] Ruihui Li, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. Pu-gan: a point cloud upsampling adversarial network. In *ICCV*, 2019.

[11] Yangyan Li, Rui Bu, Mingchao Sun, and Baoquan Chen. Pointcnn. *arXiv:1801.07791*, 2018.

[12] Z. Lian, J. Zhang, S. Choi, H. ElNaghy, J. El-Sana, T. Furuya, A. Giachetti, R. A. Guler, L. Lai, C. Li, H. Li, F. A. Limberger, R. Martin, R. U. Nakanishi, A. P. Neto, L. G. Nonato, R. Ohbuchi, K. Pevzner, D. Pickup, P. Rosin, A. Sharf, L. Sun, X. Sun, S. Tari, G. Unal, and R. C. Wilson. Non-rigid 3d shape retrieval. In *3DOR*, 2015.

[13] Yaron Lipman, Daniel Cohen-Or, David Levin, and Hillel Tal-Ezer. Parameterization-free projection for geometry reconstruction. *TOG*, 2007.

[14] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *ICCV*, 2017.

[15] Daniel Maturana and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *IROS*, 2015.

[16] Shirui Pan and etc. Adversarially regularized graph autoencoder for graph embedding. In *IJCAI*, 2018.

[17] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *arXiv:1612.00593*, 2016.

[18] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *NIPS*, 2017.

[19] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. Octnet: Learning deep 3d representations at high resolutions. In *CVPR*, 2017.

[20] Marina Sokolova, Nathalie Japkowicz, and Stan Szpakowicz. Beyond accuracy, f-score and roc: a family of discriminant measures for performance evaluation. In *Australasian joint conference on artificial intelligence*, 2006.

[21] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *TOG*, 2019.

[22] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *CVPR*, 2015.

[23] Wang Yifan, Shihao Wu, Hui Huang, Daniel Cohen-Or, and Olga Sorkine-Hornung. Patch-based progressive 3d point set upsampling. In *CVPR*, 2019.

[24] Lequan Yu, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. Ec-net: an edge-aware point set consolidation network. In *ECCV*, 2018.

[25] Lequan Yu, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. Pu-net: Point cloud upsampling network. In *CVPR*, 2018.

[26] Yingxue Zhang and Michael Rabbat. A graph-cnn for 3d point cloud classification. In *ICASSP*, 2018.