PU-GAN: a Point Cloud Upsampling Adversarial Network

Ruihui Li[1] Xianzhi Li[1,3] Chi-Wing Fu[1,3] Daniel Cohen-Or[2] Pheng-Ann Heng[1,3]
[1]The Chinese University of Hong Kong [2] Tel Aviv University
[3]Guangdong Provincial Key Laboratory of Computer Vision and Virtual Reality Technology,
Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, China
{lirh,xzli,cwfu,pheng}@cse.cuhk.edu.hk dcor@mail.tau.ac.il

Abstract

Point clouds acquired from range scans are often sparse, noisy, and non-uniform. This paper presents a new point cloud upsampling network called PU-GAN [1], which is formulated based on a generative adversarial network (GAN), to learn a rich variety of point distributions from the latent space and upsample points over patches on object surfaces. To realize a working GAN network, we construct an up-down-up expansion unit in the generator for upsampling point features with error feedback and self-correction, and formulate a self-attention unit to enhance the feature integration. Further, we design a compound loss with adversarial, uniform and reconstruction terms, to encourage the discriminator to learn more latent patterns and enhance the output point distribution uniformity. Qualitative and quantitative evaluations demonstrate the quality of our results over the state-of-the-arts in terms of distribution uniformity, proximity-to-surface, and 3D reconstruction quality.

1. Introduction

Point clouds are the standard outputs from 3D scanning. In recent years, they are gaining more and more popularity as a compact representation for 3D data, and as an effective means for processing 3D geometry. However, raw point clouds produced from depth cameras and LiDAR sensors are often sparse, noisy, and non-uniform. This is evidenced in various public benchmark datasets, such as KITTI [8], SUN RGB-D [28], and ScanNet [6]. Clearly, the raw data is required to be amended, before it can be effectively used for rendering, analysis, or general processing.

Given a sparse, noisy, and non-uniform point cloud, our goal is to upsample it and generate a point cloud that is *dense*, *complete*, and *uniform*, as a faithful representation of the underlying surface. These goals are very challenging to achieve, given such an imperfect input, since, apart from upsampling the data, we need to fill small holes and gaps in data, while improving the point distribution uniformity.

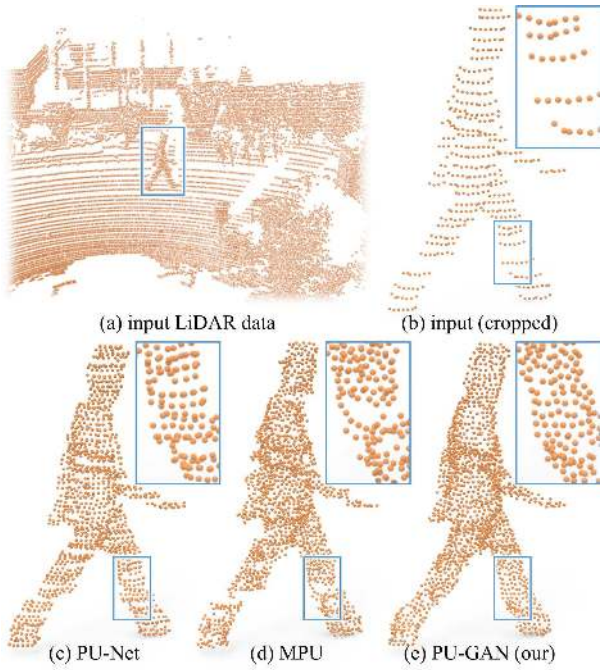

Figure 1. Upsampling (b) a point set cropped from (a) the real-scanned KITTI dataset [8] by using (c) PU-Net [40] (CVPR 2018), (d) MPU [38] (CVPR 2019), and (e) our PU-GAN. Note the distribution non-uniformity and sparsity in the input.

Early methods [3, 21, 14, 15, 33] for the problem are optimization-based, where various shape priors are used to constrain the point cloud generation. Recently, deep neural networks brought the promise of data-driven approaches to the problem. Network architectures, including PU-Net [40], EC-Net [39], and very recently, MPU [38], have demonstrated the advantage of upsampling point clouds through learning. However, these networks may not produce plausible results from low-quality inputs that are particularly sparse and non-uniform; in Figure 1 (c)-(e), we show a typical example that demonstrates the advantage of our method,

[1] https://liruihui.github.io/publication/PU-GAN/

which unlike the other networks, it combines completion and uniformity together with upsampling.

In this work, we present a new point cloud upsampling framework, namely PU-GAN, that combines upsampling with data amendment capability. The key contribution is an adversarial network to enable us to train a generator network to learn to produce a rich variety of point distributions from the latent space, and also a discriminator network to help implicitly evaluate the point sets produced from the generator against the latent point distribution. Particularly, the adversarial learning strategy of the GAN framework can regularize the predictions from a global perspective and implicitly penalize outputs that deviate from the target.

However, successfully training a working GAN framework is known to be challenging [9, 25], particularly the difficulty to balance between the generator and discriminator and to avoid the tendency of poor convergence. Thus, we first improve the point generation quality, or equivalently the feature expansion capability, of the generator, by constructing an up-down-up unit to expand point features by upsampling also their differences for self-correction. Besides, we formulate a self-attention unit to enhance the feature integration quality. Lastly, we further design a compound loss to train the network end-to-end with adversarial, uniform, and reconstruction terms to enhance the distribution uniformity of the results and encourage the discriminator to learn more latent patterns in the target distribution.

To evaluate the upsampling quality of PU-GAN, we employ four metrics to assess its performance against the state-of-the-arts on a variety of synthetic and real-scanned data. Extensive experimental results show that our method outperforms others in terms of distribution uniformity, proximity-to-surface, and 3D reconstruction quality.

2. Related Work

Optimization-based upsampling. To upsample a point set, a pioneering method by Alexa *et al.* [3] inserts points at Voronoi diagram vertices in the local tangent space. Later, Lipman *et al.* [21] introduced the locally optimal projection (LOP) operator to resample points and reconstruct surfaces based on an L_1 norm. Soon after, Huang *et al.* [14] devised a weighted LOP with an iterative normal estimation to consolidate point sets with noise, outliers, and non-uniformities. Huang *et al.* [15] further developed a progressive method called EAR for edge-aware resampling of point sets. Later, Wu *et al.* [33] proposed a consolidation method to fill large holes and complete missing regions by introducing a new deep point representation. Overall, these methods are *not* data-driven; they heavily rely on priors, *e.g.*, the assumption of smooth surface, normal estimation, *etc.*

Deep-learning-based upsampling. In recent few years, several deep neural networks have been designed to learn features directly from point sets, *e.g.*, PointNet [23], PointNet++ [24], SpiderCNN [35], KCNet [26], SPLATNet [29], PointCNN [20], PointConv [13], PointWeb [44], DGCNN [31], *etc*. Different from point completion [41, 10] which generates the entire object from partial input, point cloud upsampling tends to improve the point distribution uniformity within local patches. Based on the PointNet++ architecture, Yu *et al.* [40] introduced a deep neural network PU-Net to upsample point sets. PU-Net works on patches and expands a point set by mixing-and-blending point features in the feature space. Later, they introduced EC-Net [39] for edge-aware point cloud upsampling by formulating an edge-aware joint loss function to minimize the point-to-edge distances. Very recently, Wang *et al.* [38] presented a multi-step progressive upsampling (MPU) network to further suppress noise and retain details. In this work, we present a new method to upsample point clouds by formulating a GAN framework, enabling us to generate higher-quality point samples with completion and uniformity.

GAN-based 3D shape processing. Compared with the conventional CNN, the GAN framework makes use of a discriminator to implicitly learn to evaluate the point sets produced from the generator. Such flexibility particularly benefits generative tasks [19, 16, 43], since we hope the generator can produce a rich variety of output patterns.

Recently, there are some inspiring works in applying GAN to 3D shape processing. Most of them focus on generating 3D objects either from a probabilistic space [32] or from 2D images [36, 11, 27]. Moreover, Wang *et al.* [30] introduced a 3D-ED-GAN for shape completion given a corrupted 3D scan as input. However, these methods can only consume 3D volumes as inputs or output shapes in voxel representations. Achlioptas *et al.* [2] first adapted the GAN model to operate on raw point clouds for enhancing the representation learning. To the best of our knowledge, no prior works develop GAN models for point cloud upsampling.

3. Method

3.1. Overview

Given an unordered sparse point set $\mathcal{P} = \{p_i\}_{i=1}^{N}$ of N points, we aim to generate a dense point set $\mathcal{Q} = \{q_i\}_{i=1}^{rN}$ of rN points, where r is the upsampling rate. While output $\mathcal{Q}$ is not necessarily a superset of $\mathcal{P}$, we want it to satisfy two requirements: (i) $\mathcal{Q}$ should describe the *same underlying geometry* of a latent target object as $\mathcal{P}$, so points in $\mathcal{Q}$ should lie on and cover the target object surface; and (ii) points in $\mathcal{Q}$ should be *uniformly-distributed* on the target object surface, even for sparse and non-uniform input $\mathcal{P}$.

Figure 2 shows the overall network architecture of PU-GAN, where the generator produces dense output $\mathcal{Q}$ from sparse input $\mathcal{P}$, and the discriminator aims to find the fake generated ones. Following, we first elaborate on the archi-

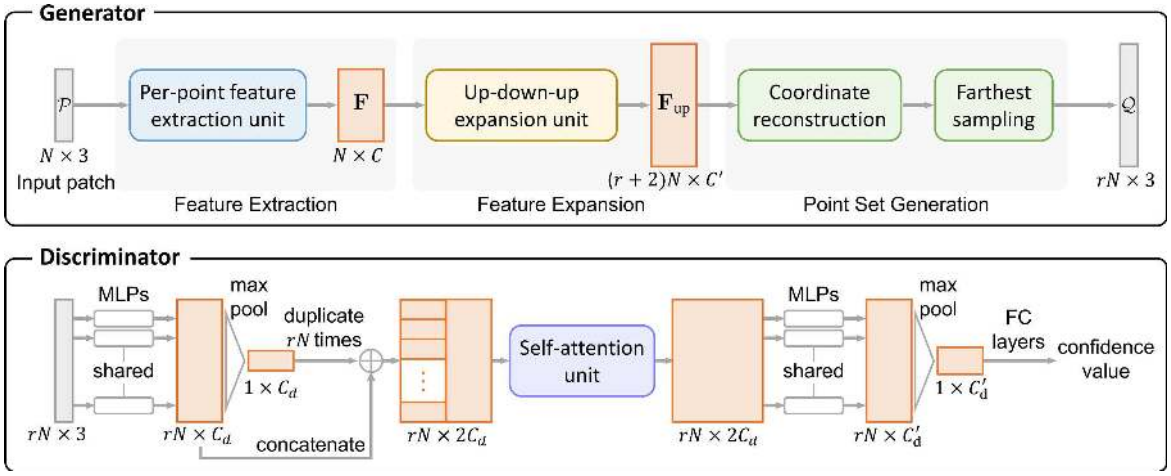

Figure 2. Overview of PU-GAN's generator and discriminator architecture. Note that N is the number of points in input $\mathcal{P}$; r is the upsampling rate; and C, C', C_d, and C'_d are the number of feature channels that are 480, 128, 64, and 256, respectively, in our implementation.

tecture of the generator and discriminator (Section 3.2). We then present two building blocks in our architecture: the up-down-up expansion unit (Sections 3.3) and the self-attention unit (Section 3.4). Lastly, we present the patch-based training strategy with the compound loss (Section 3.5).

3.2. Network Architecture

3.2.1 Generator

As shown on top of Figure 2, our generator network has three components to successively process input $\mathcal{P}$:

The feature extraction component aims to extract features $\mathbf{F}$ from input $\mathcal{P}$ of $N \times d$, where d is the number of dimensions in the input point attributes, *i.e.*, coordinates, color, normal, *etc.* Here, we focus on the simplest case with $d = 3$, considering only the 3D coordinates, and adopt the recent feature extraction method in [38], where dense connections are introduced to integrate features across different layers.

The feature expansion component expands $\mathbf{F}$ to produce the expanded features $\mathbf{F}_{\text{up}}$; here, we design the *up-down-up expansion unit* (see Figure 2 (top)) to enhance the feature variations in $\mathbf{F}_{\text{up}}$, enabling the generator to produce more diverse point distributions; see Section 3.3 for details.

The point set generation component first regresses a set of 3D coordinates from $\mathbf{F}_{\text{up}}$ via a set of multilayer perceptrons (MLPs). Since the feature expansion process is still local, meaning that the features in $\mathbf{F}_{\text{up}}$ (or equivalently, points in the latent space) are intrinsically close to the inputs, we thus include a farthest sampling step in the network to retain only rN points that are further away from one another; see the green boxes in Figure 2. To allow this selection, when we expand $\mathbf{F}$ to $\mathbf{F}_{\text{up}}$, we actually generate $(r+2)N$ features in $\mathbf{F}_{\text{up}}$. This strategy further enhances the point set distribution uniformity, particularly from a global perspective.

3.2.2 Discriminator

The goal of the discriminator is to distinguish whether its input (a set of rN points) is produced by the generator. To do so, we first adopt the basic network architecture in [41] to extract the global features, since it efficiently combines the local and global information, and ensures a lightweight network. To improve the feature learning, we add a self-attention unit (see Section 3.4) after the feature concatenation; see the middle part in Figure 2 (bottom). Compared with the basic MLPs, this attention unit helps enhance the feature integration and improve the subsequent feature extraction capability. Next, we apply a set of MLPs and a max pooling to produce the global features, and further regress the final confidence value via a set of fully connected layers. If this confidence value is close to 1, the discriminator predicts that the input likely comes from a target distribution with high confidence, and otherwise from the generator.

3.3. Up-down-up Expansion Unit

To upsample a point set, PU-Net [40] duplicates the point features and uses separate MLPs to process each copy independently. However, the expanded features would be too similar to the inputs, so affecting the upsampling quality. Rather than a single-step expansion, the very recent MPU method [38] breaks a $16\times$ upsampling network into four successive $2\times$ upsampling subnets to progressively upsample in multiple steps. Though details are better preserved in the upsampled results, the training process is complex and requires more subsets for a higher upsampling rate.

In this work, we construct an up-down-up expansion unit to expand the point features. To do so, we first upsample the point features (after MLPs) to generate $\mathbf{F}'_{\text{up}}$ and downsample it back (see Figure 3 (top)); then, we compute the difference (denoted as Δ) between the features before the upsampling and after the downsampling. By also upsampling

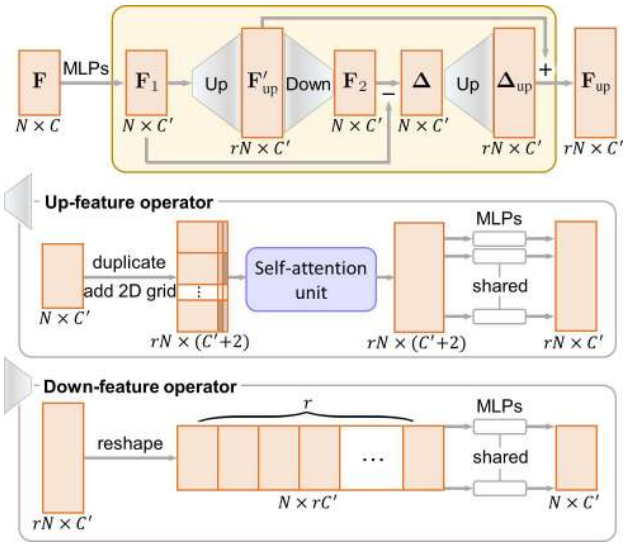

Figure 3. The up-down-up expansion unit (top), up-feature operator (middle), and down-feature operator (bottom).

Δ to Δ_{up}, we add Δ_{up} to $\mathbf{F}'_{\text{up}}$ to self-correct the expanded features; see again Figure 3 (top) for the whole procedure. Such a strategy not only avoids tedious multi-step training, but also facilitates the production of fine-grained features.

Next, we discuss the up-feature and down-feature operators in the up-down-up expansion unit (see also Figure 3):

Up-feature operator. To upsample the point features r times, we should increase the variations among the duplicated features. This is equivalent to pushing the new points away from the input ones. After we duplicate the input feature map (of N feature vectors and C' channels) r times, we adopt the 2D grid mechanism in FoldingNet [37] to generate a unique 2D vector per feature-map copy, and append such vector to each point feature vector in the same feature-map copy; see Figure 3 (middle). Further, we use the self-attention unit (see Section 3.4) followed by a set of MLPs to produce the output upsampled features.

Down-feature operator. To downsample the expanded features, we reshape the features and then use a set of MLPs to regress the original features; see Figure 3 (bottom).

3.4. Self-attention Unit

To introduce long-range context dependencies to enhance feature integration after concatenation, we adopt the self-attention unit [42] in the generator (see Figure 3 (middle)), as well as in the discriminator (see Figure 2 (bottom)). Figure 4 presents its architecture. Specifically, we transform the input features into $\mathbf{G}$ and $\mathbf{H}$ via two separate MLPs, and then generate the attention weights $\mathbf{W}$ from $\mathbf{G}$ and $\mathbf{H}$ by

$$\mathbf{W} = f_{\text{softmax}}(\mathbf{G}^T\mathbf{H}) \,, \tag{1}$$

where f_{softmax} denotes the softmax function. After that, we obtain the weighted features $\mathbf{W}^T\mathbf{K}$, where $\mathbf{K}$ is the set of features extracted from the input via another MLP. Lastly, we generate the output features, which is the sum of the input features and the weighted features.

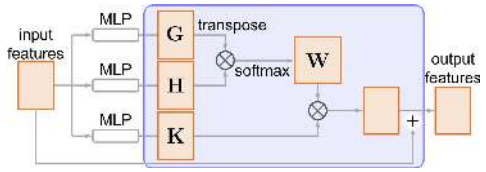

Figure 4. Illustration of the self-attention unit.

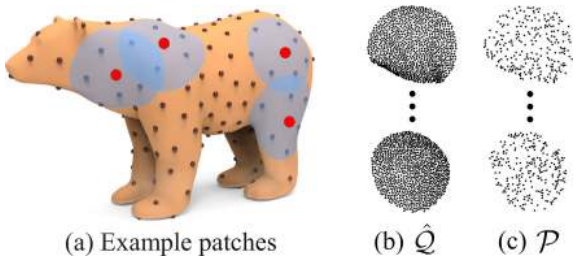

Figure 5. (a) Seed points (black dots) and patches (blue disks) on a 3D mesh in training data. (b) & (c) Example $\hat{\mathcal{Q}}$ and $\mathcal{P}$ on patches.

3.5. Patch-based End-to-end Training

3.5.1 Training data preparation

We trained our network to upsample local groups of points over patches on object surface. Specifically, for each 3D mesh (normalized in a unit sphere) in training set (see Section 4.1), we use randomly find 200 seed positions on each mesh surface, geodesically grow a patch from each seed (each patch occupies ~5% of the object surface), and then normalize each patch within a unit sphere; see Figure 5(a). On each patch, we further use Poisson disk sampling [5] to generate $\hat{\mathcal{Q}}$, which is a target point set of rN points on the patch. During the training, we generate the network input $\mathcal{P}$ by randomly selecting N points on-the-fly from $\hat{\mathcal{Q}}$.

3.5.2 Loss functions

We now present the compound loss designed for training PU-GAN in an end-to-end fashion.

Adversarial loss. To train the generator network G and discriminator network D in an adversarial manner, we use the least-squared loss [22] as our adversarial loss:

$$\mathcal{L}_{\text{gan}}(G) = \frac{1}{2}[D(\mathcal{Q}) - 1]^2 \tag{2}$$

$$\text{and } \mathcal{L}_{\text{gan}}(D) = \frac{1}{2}[D(\mathcal{Q})^2 + (D(\hat{\mathcal{Q}}) - 1)^2], \tag{3}$$

where $D(\mathcal{Q})$ is the confidence value predicted by D from generator output $\mathcal{Q}$. During the network training, G aims to generate $\mathcal{Q}$ to fool D by minimizing $\mathcal{L}_{\text{gan}}(G)$, while D aims to minimize $\mathcal{L}_{\text{gan}}(D)$ to learn to distinguish $\mathcal{Q}$ from $\hat{\mathcal{Q}}$.

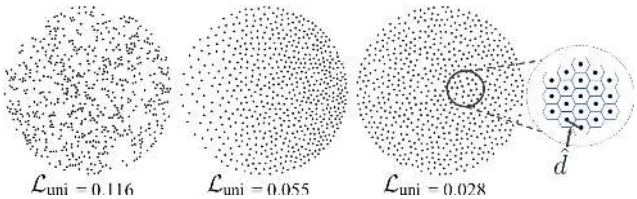

Figure 6. Example point sets with same number of points (625) but different point distribution patterns; $\mathcal{L}_{\text{uni}}$ is computed with p=1%.

Uniform loss. The problem of learning to generate point sets in 3D is complex with an immense space for exploration during the network training. Particularly, we aim for uniform distributions; using the adversarial loss alone is hard for the network to converge well. Thus, we formulate a uniform loss to evaluate $\mathcal{Q}$ from the generator, aiming to improve the generative ability of the generator.

To evaluate a point set's uniformity, the NUC metric in PU-Net [40] crops equal-sized disks on the object surface and counts the number of points in the disks. So, the metric neglects the local clutter of points. Figure 6 shows three patches of points with very different point distributions; since they contain the same number of points, the NUC metric cannot distinguish their distribution uniformity.

Our method evaluates $\mathcal{Q}$ (a patch of rN points) during the network training by first using the farthest sampling to pick M seed points in $\mathcal{Q}$ and using a ball query of radius r_d to crop a point subset (denoted as S_j, $j = 1..M$) in $\mathcal{Q}$ at each seed. Here, we use a small r_d, so S_j roughly lies on a small local disk of area πr_d^2 on the underlying surface. On the other hand, since we form patches by geodesics and normalize them in a unit sphere, patch area is $\sim\pi 1^2$. So, the expected percentage of points in S_j (denoted as p) is $\pi r_d^2/\pi 1^2 = r_d^2$, and the expected number of points in S_j (denoted as $\hat{n}$) is rNp. Hence, we follow the chi-squared model to measure the deviation of $|S_j|$ from $\hat{n}$, and define

$$U_{\text{imbalance}}(S_j) = \frac{(|S_j| - \hat{n})^2}{\hat{n}} \,. \tag{4}$$

To account for local point clutter, for each point in S_j, we find its distance to the nearest neighbor, denoted as $d_{j,k}$ for the k-th point in S_j. If S_j has a uniform distribution, the expected point-to-neighbor distance $\hat{d}$ should be roughly $\sqrt{\frac{2\pi r_d^2}{|S_j|\sqrt{3}}}$, which is derived based on the assumption that S_j is flat and neighboring points are hexagonal; see Figure 6 (right) for an illustration and supplemental material for the derivation. Again, we follow the chi-squared model to measure the deviation of $d_{j,k}$ from $\hat{d}$, and define

$$U_{\text{clutter}}(S_j) = \sum_{k=1}^{|S_j|} \frac{(d_{j,k} - \hat{d})^2}{\hat{d}} \,. \tag{5}$$

Here, U_{clutter} accounts for the local distribution uniformity, while $U_{\text{imbalance}}$ accounts for the nonlocal uniformity to encourage better point coverage. Putting them together, we formulate the uniform loss as

$$\mathcal{L}_{\text{uni}} = \sum_{j=1}^{M} U_{\text{imbalance}}(S_j) \cdot U_{\text{clutter}}(S_j) \,. \tag{6}$$

Figure 6 shows three example point sets with the same number of points but different point distribution patterns. Using $\mathcal{L}_{\text{uni}}$, we can distinguish the point uniformity among them.

Reconstruction loss. Both the adversarial and uniform losses do not encourage the generated points to lie on the target surface. Thus, we include a reconstruction loss using the Earth Mover's distance (EMD) [7]:

$$\mathcal{L}_{\text{rec}} = \min_{\phi:\mathcal{Q}\to\hat{\mathcal{Q}}} \sum_{q_i\in\mathcal{Q}} \|q_i - \phi(q_i)\|_2, \tag{7}$$

where $\phi : \mathcal{Q} \to \hat{\mathcal{Q}}$ is the bijection mapping.

Compound loss. Overall, we train PU-GAN end-to-end by minimizing $\mathcal{L}_G$ for generator and $\mathcal{L}_D$ for discriminator:

$$\mathcal{L}_G = \lambda_{\text{gan}}\mathcal{L}_{\text{gan}}(G) + \lambda_{\text{rec}}\mathcal{L}_{\text{rec}} + \lambda_{\text{uni}}\mathcal{L}_{\text{uni}}, \tag{8}$$

$$\text{and } \mathcal{L}_D = \mathcal{L}_{\text{gan}}(D), \tag{9}$$

where λ_{gan}, λ_{rec}, and λ_{uni} are weights. During the training process, G and D are optimized alternatively.

4. Experiments

4.1. Datasets and Implementation Details

We collected 147 3D models from the released datasets of PU-Net [40] and MPU [38], as well as from the Visionair repository [1], covering a rich variety of objects, ranging from simple and smooth models (*e.g.*, Icosahedron) to complex and high-detailed objects (*e.g.*, Statue); see the supplemental material for all of them. Among them, we randomly select 120 models for training and use the rest for testing.

In the training phase, we cropped 200 patches per training model (see Section 3.5.1) and produced 24,000 patches in total. By default, we set $N = 256$, $r = 4$, and $M = 50$. Moreover, for the uniform loss, we cropped one set of S_j with radius $r_d = \sqrt{p}$ for each $p \in \{$ 0.4%, 0.6%, 0.8%, 1.0%, 1.2% $\}$, compute Eq. (6) five times for each set, and then sum up the results as the $\mathcal{L}_{\text{uni}}$ term in Eq. (8).

To avoid overfitting in training, we augment the network inputs by random rotation, scaling, and point perturbation with Gaussian noise. We trained the network for 100 epochs using the Adam algorithm [18] with a two time-scale update rule (TTUR) [12]. We set the learning rates of generator and discriminator as 0.001 and 0.0001, respectively; after 50k iterations, we gradually reduce both rates by a decay rate of 0.7 per 50k iterations until 10^{-6}. The batch size is 28, and λ_{rec}, λ_{uni}, and λ_{gan} are empirically set as 100, 10, and 0.5, respectively. We implemented our network using TensorFlow and trained it on NVidia Titan Xp GPU.

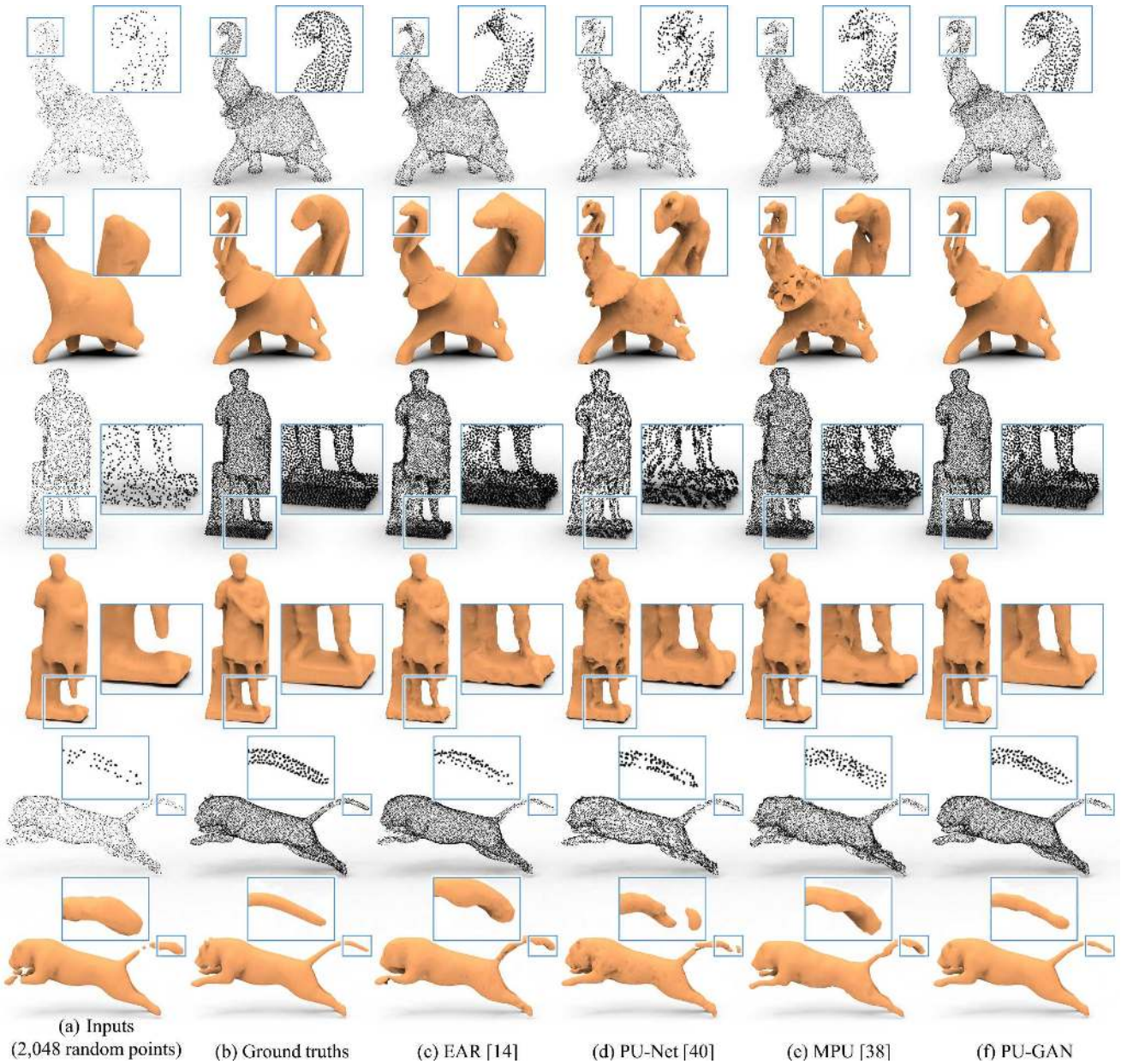

Figure 7. Comparing point set upsampling (x4) and surface reconstruction results produced with different methods (c-f) from inputs (a).

During the testing, given a point set, we follow the patch-based strategies in MPU [38] and EC-Net [39] to use the farthest sampling to pick seeds and extract a local patch of N points per seed. Then, we feed the patches to the generator and combine the upsampled results as the final output.

4.2. Evaluation Metrics

We employ four evaluation metrics: (i) uniformity using $\mathcal{L}_{\text{uni}}$ in Eq. (6), (ii) point-to-surface (P2F) distance using the testing models, (iii) Chamfer distance (CD), and (iv) Hausdorff distance (HD) [4]. For quantitative evaluation, we use Poisson disk sampling to sample 8,192 points as the ground truth (*e.g.*, see Figure 7(b)) on each of the testing models, and randomly select 2,048 points as testing input. To evaluate the uniformity of the test results (using Eq. (6)), we randomly pick $M = 1,000$ seeds on each result, and instead of using ball query to crop S_j, we use the actual mesh (testing model) to geodesically find S_j at each seed of different p for higher-quality evaluation. The lower the metric values are, the better the upsampling results are.

4.3. Qualitative and Quantitative Comparisons

We qualitatively and quantitatively compared PU-GAN with three state-of-the-art point set upsampling methods:

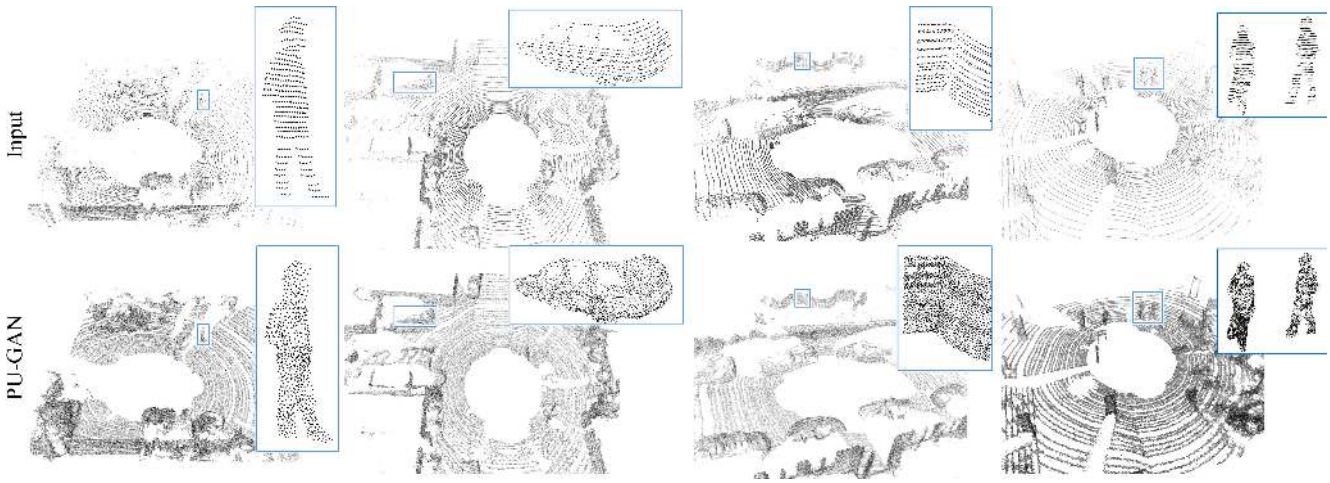

Figure 8. Using PU-GAN to upsample real-scanned point cloud data acquired by LiDAR.

EAR [15], PU-Net [40], and MPU [38]. For EAR, we employed the released demo code and generated the best results by exhaustively fine-tuning every associated parameter. For PU-Net and MPU, we used their public code and retrained their networks using our training data.

Table 1 shows the quantitative comparison results. Our PU-GAN achieves the lowest values consistently for all the evaluation metrics. Particularly, the uniformity of our results stays the lowest for all different p, indicating that the points generated by PU-GAN are much more uniform than those produced by others over varying scales.

Besides quantitative results, we show point set upsampling and surface reconstruction (using [17]) results for various models in Figure 7. Comparing the results produced by (f) our method and by (c-e) others, against (b) the ground truth points that are uniformly-sampled on the original testing models, we can see that other methods tend to produce more noisy and nonuniform point sets, thus leading to more artifacts in the reconstructed surfaces. See, particularly, the blown-up views, showing that PU-GAN can produce more fine-grained details in the upsampled results, *e.g.*, elephant's nose (top) and tiger's tail (bottom). More comparison results can be found in the supplemental material.

4.4. Upsampling Real-scanned Data

Figure 8 shows upsampling results on LiDAR point clouds (downloaded from KITTI dataset [8]) produced by PU-GAN. From the first row, we can see the sparsity and non-uniformity of the inputs. Our PU-GAN can still fill some holes and output more uniform points in the results; please see the supplemental material for more results.

4.5. Ablation Study and Baseline Comparison

To evaluate the components in PU-GAN, including the GAN framework (*i.e.*, remove the discriminator and keep only the generator), up-down-up expansion unit, self-attention unit, uniform loss, and farthest sampling, we removed each of them from the network and generated upsampling results for the testing models. Table 2 shows the evaluation results. Our full pipeline performs the best, and removing any component reduces the overall performance, meaning that each component contributes. Particularly, removing the GAN framework (*i.e.*, removing the discriminator) causes the largest performance drop, thereby showing the effectiveness of adversarial learning in the framework.

Further, we designed a GAN baseline for comparison by removing $\mathcal{L}_{\text{uni}}$, self-attention unit, up-down-up expansion unit, and farthest sampling altogether; see supplemental material for its architecture. Quantitative results shown in the second last row of Table 2 and a visual comparison example shown in Figure 9 demonstrate that simply applying a basic GAN framework is insufficient for the upsampling problem; our adaptations to the GAN model plus the various formulations contribute to realizing a working GAN model.

Table 1. Quantitative comparisons with the state-of-the-arts.

Methods	Uniformity for different p (10^{-3})					P2F (10^{-3})	CD (10^{-3})	HD (10^{-3})
	0.4%	0.6%	0.8%	1.0%	1.2%			
EAR [15]	16.84	20.27	23.98	26.15	29.18	5.82	0.52	7.37
PU-Net [40]	29.74	31.33	33.86	36.94	40.43	6.84	0.72	8.94
MPU [38]	7.51	7.41	8.35	9.62	11.13	3.96	0.49	6.11
PU-GAN	**3.38**	**3.49**	**3.44**	**3.91**	**4.64**	**2.33**	**0.28**	**4.64**

Table 2. Quantitative comparisons: removing each specific component from the full PU-GAN pipeline (top five rows) vs. baseline GAN model (6-th row) vs. full PU-GAN pipeline (last row).

	Uniformity for different p (10^{-3})					P2F (10^{-3})	CD (10^{-3})	HD (10^{-3})
	0.4%	0.6%	0.8%	1.0%	1.2%			
Discriminator	7.02	7.31	8.36	9.70	11.17	4.61	0.57	7.25
$\mathcal{L}_{\text{uni}}$	5.15	5.71	6.13	6.82	7.32	3.99	0.51	6.22
self-attention	4.19	4.66	4.87	5.52	6.47	3.97	0.46	6.01
up-down-up	3.89	4.16	4.63	5.14	5.89	3.02	0.35	5.15
farthest samp.	3.56	3.76	3.78	4.15	4.97	2.72	0.31	4.96
baseline GAN	8.12	8.18	8.76	9.89	11.32	4.79	0.59	7.31
full pipeline	**3.38**	**3.49**	**3.44**	**3.91**	**4.64**	**2.33**	**0.28**	**4.64**

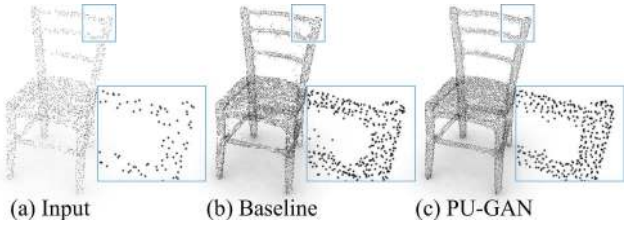

Figure 9. Visual comparison of (c) PU-GAN against (b) the baseline GAN model, given (a) an input of 2,048 non-uniform points.

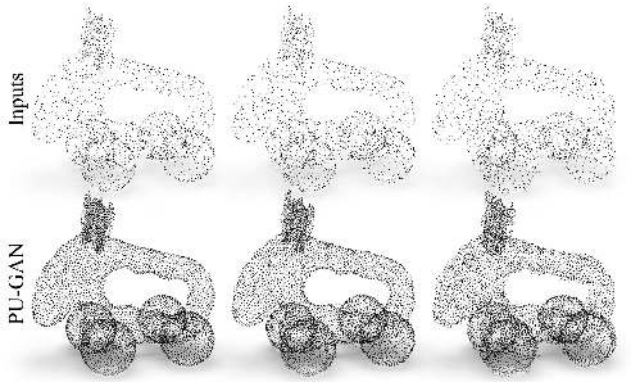

Figure 10. Upsampling results by applying PU-GAN to inputs with noise level of 0, 0.5%, and 1%.

4.6. Other Experiments

Upsampling point sets of varying noise levels. Figure 10 shows the results of using PU-GAN to upsample point sets of increasing Gaussian noise levels, indicating the robustness of PU-GAN to noise and sparsity. Please refer to the supplemental material for more results.

Upsampling point sets of varying sizes. Figure 11 shows the results of upsampling input point sets of different sizes; our method is stable even for input with only 512 points; more results are shown in the supplemental material.

Applications. Besides surface reconstruction and rendering (see Figure 7), point cloud upsampling is also helpful for object recognition. To demonstrate this, we trained PointNet (vanilla version) [23] on ModelNet40 dataset [34] for classification under two cases: in case (i), we directly trained PointNet to take sparse training set (512 points) as inputs for classifying objects in the sparse testing set; and in case (ii), we upsampled the point clouds in both sparse training and testing sets to 2,048 points using PU-GAN and then trained another PointNet with the upsampled training set for classifying the upsampled data in the testing set. Results show that, the overall classification accuracy increased from 82.4% (case (i)) to 83.8% (case (ii)), indicating that PU-GAN helps improve the classification performance.

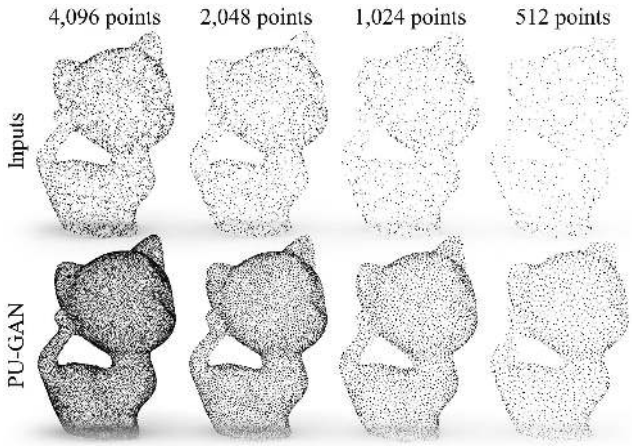

Figure 11. Upsampling point sets (top row) of varying sizes.

5. Conclusion

In this paper, we presented, PU-GAN, a novel GAN-based point cloud upsampling network, that combines upsampling with data amendment capabilities. Such adversarial network enables the generator to produce a uniformly-distributed point set, and the discriminator to implicitly penalize outputs that deviate from the expected target. To facilitate the GAN framework, we introduced an up-down-up unit for feature expansion with error feedback and self-correction, as well as a self-attention unit for better feature fusion. Further, we designed a compound loss to guide the learning of the generator and discriminator. We demonstrated the effectiveness of our PU-GAN via extensive experiments, showing that it outperforms the state-of-the-art methods for various metrics, and presented the upsampling performance on real-scanned LiDAR inputs.

However, since PU-GAN is designed to complete tiny holes at patch level, it has limited ability in filling large gaps or holes in point clouds; see Figure 8. Analyzing and synthesizing at the patch level lacks a global view of the overall shape. Please refer to the supplemental material for a typical failure case. In the future, we will explore a multi-scale training strategy to encourage the network to learn from both local small patched and global structures. We are also considering exploring conditional GANs, to let the network learn the uniformity and semantic consistency at the same time.

Acknowledgments. We thank anonymous reviewers for the valuable comments. The work is supported by the 973 Program (Proj. No. 2015CB351706), the National Natural Science Foundation of China with Proj. No. U1613219, the Research Grants Council of the Hong Kong Special Administrative Region (No. CUHK 14203416 & 14201717), and the Israel Science Foundation grants 2366/16 and 2472/7.

References

[1] Visionair. http://www.infra-visionair.eu/. Accessed: 2019-07-24. 5

[2] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas J. Guibas. Learning representations and generative models for 3D point clouds. In *Int. Conf. on Machine Learning. (ICML)*, pages 40–49, 2018. 2

[3] Marc Alexa, Johannes Behr, Daniel Cohen-Or, Shachar Fleishman, David Levin, and Claudio T. Silva. Computing and rendering point set surfaces. *IEEE Trans. Vis. & Comp. Graphics*, 9(1):3–15, 2003. 1, 2

[4] Matthew Berger, Joshua A. Levine, Luis Gustavo Nonato, Gabriel Taubin, and Claudio T. Silva. A benchmark for surface reconstruction. *ACM Trans. on Graphics*, 32(2):20:1–17, 2013. 6

[5] Massimiliano Corsini, Paolo Cignoni, and Roberto Scopigno. Efficient and flexible sampling with blue noise properties of triangular meshes. *IEEE Trans. Vis. & Comp. Graphics*, 18(6):914–924, 2012. 4

[6] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. ScanNet: Richly-annotated 3D reconstructions of indoor scenes. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 5828–5839, 2017. 1

[7] Haoqiang Fan, Hao Su, and Leonidas J. Guibas. A point set generation network for 3D object reconstruction from a single image. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 605–613, 2017. 5

[8] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The KITTI dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013. 1, 7

[9] Ian J. Goodfellow. On distinguishability criteria for estimating generative models. In *Int. Conf. on Learning Representations (ICLR) Workshops*, 2015. 2

[10] Swaminathan Gurumurthy and Shubham Agrawal. High fidelity semantic shape completion for point clouds using latent optimization. In *IEEE Winter Conf. on Applications of Computer Vision (WACV)*, pages 1099–1108, 2019. 2

[11] JunYoung Gwak, Christopher B. Choy, Animesh Garg, Manmohan Chandraker, and Silvio Savarese. Weakly supervised 3D reconstruction with adversarial constraint. In *2017 International Conference on 3D Vision (3DV)*, pages 263–272, 2017. 2

[12] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a local Nash equilibrium. In *Int. Conf. on Advances in Neural Information Processing Systems (NIPS)*, pages 6626–6637, 2017. 5

[13] Binh-Son Hua, Minh-Khoi Tran, and Sai-Kit Yeung. Pointwise convolutional neural networks. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 984–993, 2018. 2

[14] Hui Huang, Dan Li, Hao Zhang, Uri Ascher, and Daniel Cohen-Or. Consolidation of unorganized point clouds for surface reconstruction. *ACM Trans. on Graphics (SIGGRAPH Asia)*, 28(5):176:1–7, 2009. 1, 2

[15] Hui Huang, Shihao Wu, Minglun Gong, Daniel Cohen-Or, Uri Ascher, and Hao Zhang. Edge-aware point set resampling. *ACM Trans. on Graphics*, 32(1):9:1–12, 2013. 1, 2, 7

[16] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1125–1134, 2017. 2

[17] Michael Kazhdan and Hugues Hoppe. Screened Poisson surface reconstruction. *ACM Trans. on Graphics*, 32(3):29:1–13, 2013. 7

[18] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Int. Conf. on Learning Representations (ICLR)*, 2015. 5

[19] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 4681–4690, 2017. 2

[20] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. PointCNN: Convolution on $\mathcal{X}$-transformed points. In *Int. Conf. on Advances in Neural Information Processing Systems (NIPS)*, pages 828–838, 2018. 2

[21] Yaron Lipman, Daniel Cohen-Or, David Levin, and Hillel Tal-Ezer. Parameterization-free projection for geometry reconstruction. *ACM Trans. on Graphics (SIGGRAPH)*, 26(3):22:1–5, 2007. 1, 2

[22] Xudong Mao, Qing Li, Haoran Xie, Raymond Y.K. Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *IEEE Int. Conf. on Computer Vision (ICCV)*, pages 2794–2802, 2017. 4

[23] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. PointNet: Deep learning on point sets for 3D classification and segmentation. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 652–660, 2017. 2, 8

[24] Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. PointNet++: Deep hierarchical feature learning on point sets in a metric space. In *Int. Conf. on Advances in Neural Information Processing Systems (NIPS)*, pages 5099–5108, 2017. 2

[25] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *Int. Conf. on Learning Representations (ICLR)*, 2016. 2

[26] Yiru Shen, Chen Feng, Yaoqing Yang, and Dong Tian. Mining point cloud local structures by kernel correlation and graph pooling. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 4548–4557, 2018. 2

[27] Edward Smith and David Meger. Improved adversarial systems for 3D object generation and reconstruction. In *1st Annual Conference on Robot Learning(CoRL)*, pages 87–96, 2017. 2

[28] Shuran Song, Samuel P. Lichtenberg, and Jianxiong Xiao. SUN RGB-D: A RGB-D scene understanding benchmark suite. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 567–576, 2015. 1

[29] Hang Su, Varun Jampani, Deqing Sun, Subhransu Maji, Evangelos Kalogerakis, Ming-Hsuan Yang, and Jan Kautz. SPLATNet: Sparse lattice networks for point cloud processing. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2530–2539, 2018. 2

[30] Weiyue Wang, Qiangui Huang, Suya You, Chao Yang, and Ulrich Neumann. Shape inpainting using 3D generative adversarial network and recurrent convolutional networks. In *IEEE Int. Conf. on Computer Vision (ICCV)*, pages 2298–2306, 2017. 2

[31] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph CNN for learning on point clouds. *ACM Trans. on Graphics*, 2019. to appear. 2

[32] Jiajun Wu, Chengkai Zhang, Tianfan Xue, Bill Freeman, and Josh Tenenbaum. Learning a probabilistic latent space of object shapes via 3D generative-adversarial modeling. In *Int. Conf. on Advances in Neural Information Processing Systems (NIPS)*, pages 82–90, 2016. 2

[33] Shihao Wu, Hui Huang, Minglun Gong, Matthias Zwicker, and Daniel Cohen-Or. Deep points consolidation. *ACM Trans. on Graphics (SIGGRAPH Asia)*, 34(6):176:1–13, 2015. 1, 2

[34] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3D ShapeNets: A deep representation for volumetric shapes. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1912–1920, 2015. 8

[35] Yifan Xu, Tianqi Fan, Mingye Xu, Long Zeng, and Yu Qiao. SpiderCNN: Deep learning on point sets with parameterized convolutional filters. In *European Conf. on Computer Vision (ECCV)*, pages 87–102, 2018. 2

[36] Bo Yang, Hongkai Wen, Sen Wang, Ronald Clark, Andrew Markham, and Niki Trigoni. 3D object reconstruction from a single depth view with adversarial learning. In *IEEE Int. Conf. on Computer Vision (ICCV)*, pages 679–688, 2017. 2

[37] Yaoqing Yang, Chen Feng, Yiru Shen, and Dong Tian. FoldingNet: Point cloud auto-encoder via deep grid deformation. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 206–215, 2018. 4

[38] Wang Yifan, Shihao Wu, Hui Huang, Daniel Cohen-Or, and Olga Sorkine-Hornung. Patch-based progressive 3D point set upsampling. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 5958–5967, 2019. 1, 2, 3, 5, 6, 7

[39] Lequan Yu, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. EC-Net: An edge-aware point set consolidation network. In *European Conf. on Computer Vision (ECCV)*, pages 386–402, 2018. 1, 2, 6

[40] Lequan Yu, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. PU-Net: Point cloud upsampling network. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2790–2799, 2018. 1, 2, 3, 5, 7

[41] Wentao Yuan, Tejas Khot, David Held, Christoph Mertz, and Martial Hebert. PCN: Point completion network. In *2018 International Conference on 3D Vision (3DV)*, pages 728–737, 2018. 2, 3

[42] Han Zhang, Ian J. Goodfellow, Dimitris N. Metaxas, and Augustus Odena. Self-attention generative adversarial networks. In *Int. Conf. on Machine Learning. (ICML)*, pages 7354–7363, 2019. 4

[43] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris N. Metaxas. StackGAN: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *IEEE Int. Conf. on Computer Vision (ICCV)*, pages 5907–5915, 2017. 2

[44] Hengshuang Zhao, Li Jiang, Chi-Wing Fu, and Jiaya Jia. PointWeb: Enhancing local neighborhood features for point cloud processing. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 5565–5573, 2019. 2