Progressive Growing of Points with Tree-structured Generators

Hyeontae Son[1]
son.ht@naverlabs.com
Young Min Kim[2]
youngmin.kim@snu.ac.kr

[1] NAVER LABS
Seongnam, South Korea
[2] Department of ECE
Seoul National University
Seoul, South Korea

Abstract

We present progressive growing of points with tree-structured networks that generates high-fidelity point cloud. Because point cloud data lacks the information of inherent topology or connectivity between neighboring points, the data generated from deep neural networks usually fails to faithfully produce local details. Inspired by the recent success of the progressive generation of images and curriculum learning, we suggest that the hierarchical structure of the tree-based network architecture can endow contextual information to enforce the progressive generation of point clouds. When the tree-structured network is incrementally trained by progressively adding the subsequent layers of depth, the quality of generated point cloud is superior to the data generated by the same network structure with naïve end-to-end training. Furthermore, our pipeline simultaneously learns the hierarchical structure within the data set and finds a consistent spatial decomposition of 3D shapes by coherently positioning the nodes with the same ancestors. Extensive experiments show that our method can produce a high-fidelity shape when applied to shape generation and completion as well as auto-encoding point clouds. [1]

1 Introduction

Point cloud is the raw output of 3D measurements of real-life objects and is widely used in applications for vision, robotics, and graphics. While point cloud is a simple yet expressive representation for 3D shape, the representation does not have a structure which can define the local context. In contrast to the CNN-based architecture for images that aggregates neighborhood information, an auto-encoder for 3D point clouds is designed to map the entire shape directly from the condensed global representation. Specifically, the encoder extracts the global feature vector (GFV), and the decoder regresses the entire shape from the GFV. Such encoder-decoder architecture successfully captures the global shape, and it is widely used for shape generation [1, 4, 30], 3D object reconstruction [22, 32], shape matching [6, 8] and shape deformation [26, 27].

The neural networks for the point cloud are trained with the loss that represents the discrepancy between two point cloud shapes. The most commonly used loss to compare two point cloud shapes is the Chamfer Distance (CD), which is the mean of the bi-directional

[1]Code available on **https://github.com/countywest/progressive_growing_of_points**

nearest neighbor distances. CD loss is used since the point-wise correspondences are not provided. The lack of this local context in the training objective results in the reconstructed shape having a highly irregular distribution of points without preserving the fine geometric details. Recent works also state that the training with CD is often trapped in local minima [14] and CD loss fails to make a correct assessment on the similarity of 3D shapes [1]. Such limitation of CD is referred to as Chamfer's blindness [1].

We propose progressive growing of point clouds which guides the point cloud to overcome the limitation of the Chamfer's blindness. The key motivation of our work is the success of coarse-to-fine strategies in generating 2D images [11, 28] and 3D geometries [10, 21, 29, 32]. All of these methods share the spirit of curriculum learning [3], which suggests starting from an easy task and guiding the training process to harder tasks, step by step, to reach the desired optimum. Applying the idea in training the auto-encoder for the point cloud, we start from a sparse set of points and progressively generate dense points in multiple stages.

As a means to steer the progressive growth, we utilize the hierarchical levels in tree-structured point cloud generators [7, 23, 24]. Because there is no connectivity or well-defined structure in point cloud data, it is challenging to decide how to progressively add points on a 3D shape. One common practice is to perform two-stage generation [21, 32] where the second stage upsamples the coarse results from the first stage. Instead, we increase the number of points by persistently adding a layer of hierarchy in the tree structure. As an additional hierarchy is attached, the leaf nodes from the previous stage become the immediate parents of the new leaf nodes and guide them to consistently grow into denser points in higher fidelity with our progressive training scheme.

Our progressive generation is designed to uniformly distribute points with the desired density, leading to high-fidelity reconstruction. In addition to high-quality generation, our process enforces a consistent hierarchical structure for the dataset. We position the points with the same ancestors in proximity and this property induces an interesting pattern in the indices of generated points. As a byproduct, we can find coherent spatial partitions of generated shapes, producing locally consistent correspondences. We visualize the spatial decomposition induced by our progressive generation. The newly endowed structure helps us understand how the latent nodes in different hierarchies are correlated, and we can even discern the intermediate nodes in the tree that participated in generating a specific part in the final shape. We test the efficacy of our method with tree-structured generators for 3 tasks; auto-encoding, generation, and completion of point clouds. Experimental results show that the progressive growing method boosts the performance of reconstructing point clouds over the baseline methods.

2 Related Works

Progressive learning. Progressive learning approaches have shown substantial improvement in high-resolution image generation [11], super-resolution [13, 28] and image-to-image translation [15, 25] by preserving all levels of detail and stabilizing the learning process. Several works apply the same idea to the tasks for 3D point clouds such as point upsampling [29] and point cloud generation [2, 10]. The training data is prepared in multiple resolutions by randomly sampling from the dense point clouds. In contrast, we simply use one ground truth for a shape without explicitly preparing multiple datasets. We also demonstrate that the sparse-to-dense transition overcomes the fundamental limitation of Chamfer distance and

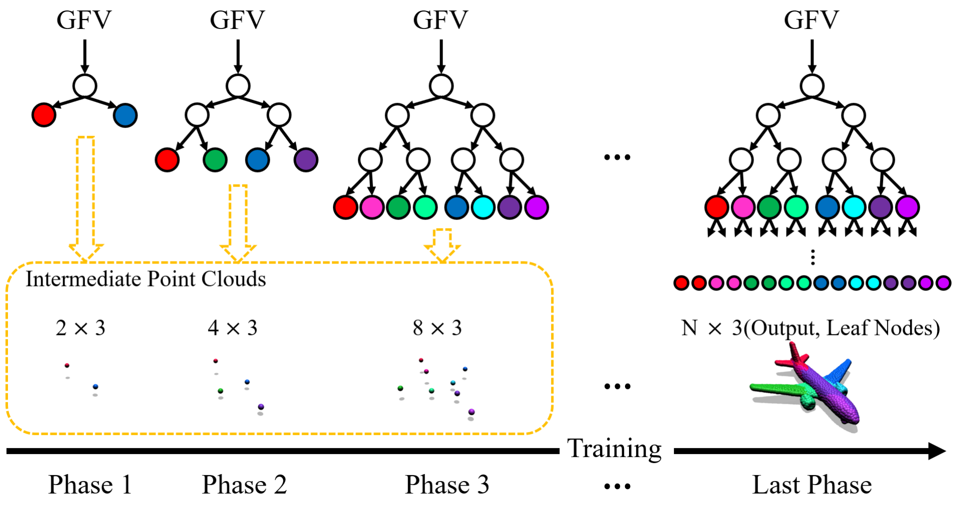

Figure 1: Progressive growing of points with tree-structured network. For any tree-shaped decoder that generates shape from a global feature vector (GFV), our method can progressively train the network to generate high-fidelity shapes in a coarse-to-fine fashion.

uniformly distributes the generated points.

The progressive growing in 3D also requires a method to systematically increase the size of the neural network. Previous works suggest fixing the pre-trained subnets [29], generating multiple resolutions with inter-level skip connections [10, 29] and initializing the new layers by replicating the last branch [2]. We utilize tree nodes, similar to [2], but we encourage smooth transition to higher resolution by linear interpolation [11]. By controlling the coefficient for the interpolation, the newly added nodes faithfully generate higher resolution shapes in the place where the parent nodes faded out.

Tree-structured point cloud generation. Generating point cloud with tree-shaped architecture is suggested by several works for single-image 3D reconstruction [7], 3D shape generation [7, 23], and point cloud completion [24]. The tree structure is composed of hierarchical feature nodes followed by leaf nodes that correspond to individual points. The parent-to-child connection in the network is implemented with 1D transposed convolution with multi-resolution features [7], 1D convolution with the node feature concatenated with the GFV [24], or a graph convolution recursively connecting the ancestor nodes [23]. While our pipeline can be applied in any tree-shaped architecture, progressive growing needs to allow the intermediate nodes at any hierarchy to produce points. We augment the intermediate nodes with auxiliary shared-MLP layers producing points, which are not used once the progressive training is finished.

3 Progressive Growing of Points

We suggest progressive growing of tree-structured network [11] to generate dense point clouds. As we incrementally add layers of the hierarchy, the network generates denser points accordingly as visualized in Figure 1. The i-th node in ℓ-th layer of the tree contains feature f_i^ℓ which can be transformed into an explicit 3D coordinate $p_i^\ell = \texttt{toPoint}^\ell(f_i^\ell)$, where p_i^ℓ is 3D coordinate in $\mathbb{R}^3$ and $\texttt{toPoint}^\ell$ is implemented with a shared-MLP.

For the reconstruction loss, we use Chamfer distance (CD) defined as

$$\mathrm{CD}(P_{out}, P_{gt}) = \frac{1}{2}\Big(\frac{1}{|P_{out}|}\sum_{x\in P_{out}} \min_{y\in P_{gt}} \|x-y\| + \frac{1}{|P_{gt}|}\sum_{y\in P_{gt}} \min_{x\in P_{out}} \|x-y\|\Big) \tag{1}$$

where P_{out} represents the point set generated from neural networks and P_{gt} is the ground truth (G.T.) point cloud. CD loss can be calculated even if the number of two point sets are different, namely $|P_{out}| \neq |P_{gt}|$. Minimizing the left term ($\frac{1}{2}(\frac{1}{|P_{out}|}\sum_{x\in P_{out}} \min_{y\in P_{gt}} \|x-y\|)$) enforces the output points to be attached to the nearest G.T. points. On the other hand, minimizing the right term ($\frac{1}{2}(\frac{1}{|P_{gt}|}\sum_{y\in P_{gt}} \min_{x\in P_{out}} \|x-y\|)$) enforces the G.T. points to pull the nearest output points. Note that $|P_{out}|$ for our method increases as we add layers for progressive training. When $|P_{out}| \ll |P_{gt}|$, minimizing the CD loss leads P_{out} to be uniformly distributed on the true 3D shape P_{gt} by minimizing the right term. Thus when the CD loss is applied in a progressive setting, the reconstructed shape does not suffer from cluttering in a particular area, which is commonly observed when the entire shape is generated in an end-to-end fashion.

When we augment an additional layer of nodes, we train the network to smoothly transition from the previous phase,

$$p_i^\ell = (1-\alpha)\cdot \texttt{toPoint}^{\ell-1}(f^{\ell-1}_{\mathcal{P}^\ell(i)}) + \alpha \cdot \texttt{toPoint}^\ell(f_i^\ell). \tag{2}$$

Here $\texttt{toPoint}^{\ell-1}(f^{\ell-1}_{\mathcal{P}^\ell(i)})$ represents the output coordinate of the $(\ell-1)$-th layer, where the index of the parent node for node i at ℓ-th layer $\mathcal{P}^\ell(i) = \left\lfloor \frac{i}{\mathcal{D}^\ell} \right\rfloor$ can be found by considering the number of siblings $\mathcal{D}^\ell$ added with ℓ-th layer. By increasing the weight α linearly from 0 to 1 for each phase ℓ, we can gradually transit to produce an increased number of points. In our experiments, we increase the α by adding $\frac{2}{step^\ell}$ for every back-propagation step and fix the α to 1 after it reaches to 1, where $step^\ell$ means the pre-defined number of iterations for phase ℓ. This technique helps to avoid sudden shocks of adding new layers to the already trained networks [11], stabilizing the learning process.

Strictly speaking, we use equation 2 except for the first and the last phases. In the first phase, we train the first layer by formulating the output point as $p_i^1 = \texttt{toPoint}^1(f_i^1)$, since there is no trained previous layer. In the last phase ($\ell = L$), we design the output as $p_i^L = (1-\alpha)\cdot \texttt{toPoint}^{L-1}(f^{L-1}_{\mathcal{P}^L(i)}) + \alpha \cdot f_i^L$ where the f_i^L is the i-th leaf node which is the 3D coordinate in itself and the ultimate output of the tree networks. After the training process is done, only the leaf nodes are used in the testing and the auxiliary networks named `toPoint` are not needed anymore.

In the early phases, $P_{out} (= \{p_i^\ell\})$ is spread to the G.T. shape by minimizing the CD loss since $|P_{out}| \ll |P_{gt}|$. After the early phases, progressive training with the smooth transition as shown in Equation 2 enforces the sibling nodes to produce points in finer resolution near the coarser points from their parent nodes. p_i^ℓ starts to move from $\texttt{toPoint}^{\ell-1}(f^{\ell-1}_{\mathcal{P}^\ell(i)})$, which is already well trained results in the previous phase. As the α grows slowly, p_i^ℓ tends to stick to the nearest G.T. point smoothly by the CD loss, so the p_i^ℓ will be located near the initial position. As a result, the generated high-resolution point clouds are uniformly distributed on

Table 1: Auto-encoding and l-GAN results. ↑: The higher the better, ↓: The lower the better. Better results are in bold. For the metrics of auto-encoder evaluation, the average CD multiplied by 10^3, EMD multiplied by 10^2, and F-score for different threshold ds are reported. For the metrics of l-GAN evaluation, JSD multiplied by 10^2, MMD-CD multiplied by 10^3, MMD-EMD multiplied by 10^2, COV, and 1-NNA are reported. Ours are post-fixed by PG which is an acronym of *progressive growing*.

Category	Model	Auto-encoder					*l*-GAN						
		CD(↓)	EMD(↓)	F-score(↑)			JSD(↓)	MMD(↓)		COV(%, ↑)		1-NNA(%, ↓)	
				@1%	@2%	@3%		CD	EMD	CD	EMD	CD	EMD
Airplane	SRTDec	8.967	2.641	0.268	0.723	0.894	9.71	**11.92**	5.91	41.48	7.16	75.93	99.51
	SRTDec(PG)	**7.618**	**1.383**	**0.343**	**0.810**	**0.937**	**1.71**	12.32	**3.49**	**49.30**	**28.07**	**70.70**	**88.60**
	MRTDec	8.613	3.063	0.283	0.737	0.904	14.22	**11.75**	6.73	44.12	5.68	74.86	99.75
	MRTDec(PG)	**7.777**	**1.410**	**0.345**	**0.798**	**0.928**	**1.95**	12.32	**3.33**	**50.29**	**31.44**	**69.59**	**85.35**
	TopNet	9.050	1.896	0.273	0.724	0.888	4.79	12.35	4.40	45.10	14.24	78.07	96.79
	TopNet(PG)	**8.601**	**1.606**	**0.289**	**0.754**	**0.905**	**2.05**	**12.04**	**3.31**	**48.64**	**34.24**	**72.84**	**87.65**
	TreeGCN	**7.883**	1.863	0.336	0.784	**0.927**	5.27	**12.04**	4.59	42.22	11.69	75.56	97.65
	TreeGCN(PG)	7.962	**1.407**	**0.351**	**0.788**	0.917	**1.67**	12.53	**3.28**	**48.23**	**33.66**	**73.70**	**85.23**
Chair	SRTDec	17.062	4.997	0.0672	0.307	0.584	23.84	**24.38**	15.53	39.87	2.70	**71.21**	99.95
	SRTDec(PG)	**13.740**	**2.457**	**0.0975**	**0.448**	**0.746**	**2.39**	25.40	**5.94**	**48.48**	**26.50**	72.25	**91.15**
	MRTDec	15.779	6.243	0.0772	0.329	0.619	21.61	**23.31**	13.38	43.22	4.52	**66.74**	100.00
	MRTDec(PG)	**14.188**	**3.286**	**0.0918**	**0.413**	**0.716**	**5.62**	24.80	**8.41**	**48.72**	**13.37**	69.62	**98.70**
	TopNet	16.632	3.539	0.0672	0.328	0.617	9.38	**23.95**	10.78	46.95	7.47	66.49	99.85
	TopNet(PG)	**15.285**	**2.940**	**0.0796**	**0.387**	**0.683**	**2.78**	24.35	**5.97**	**49.07**	**30.97**	**65.90**	**92.33**
	TreeGCN	13.990	3.266	0.0912	0.418	0.723	9.31	**24.08**	8.79	47.00	18.19	66.64	99.78
	TreeGCN(PG)	**13.848**	**2.346**	**0.0972**	**0.445**	**0.743**	**1.78**	24.90	**5.46**	**48.72**	**34.91**	**65.46**	**83.48**
Car	SRTDec	14.547	2.601	0.0533	0.304	0.624	23.41	43.79	12.25	6.24	3.74	99.07	99.85
	SRTDec(PG)	**13.108**	**2.161**	**0.0758**	**0.383**	**0.705**	**1.49**	**16.17**	**3.54**	**38.67**	**22.12**	**59.39**	**80.84**
	MRTDec	13.772	**2.429**	0.0656	0.345	0.667	26.18	45.77	13.27	5.21	3.05	98.62	99.98
	MRTDec(PG)	**13.770**	2.500	**0.0659**	0.345	**0.668**	**2.78**	**16.11**	**4.39**	**33.47**	**9.97**	**59.84**	**99.11**
	TopNet	14.604	2.449	0.0560	0.314	0.631	24.31	43.99	10.69	9.59	4.67	99.34	99.75
	TopNet(PG)	**14.051**	**2.329**	**0.0625**	**0.339**	**0.658**	**1.99**	**16.28**	**3.54**	**36.05**	**23.45**	**60.73**	**86.23**
	TreeGCN	**13.056**	2.203	**0.0783**	0.383	0.702	19.09	37.79	11.31	9.34	3.54	97.62	99.83
	TreeGCN(PG)	13.089	**2.136**	0.0770	**0.384**	**0.705**	**1.71**	**16.48**	**3.51**	**35.38**	**23.99**	**63.55**	**80.77**

the G.T. shape. Furthermore, the hierarchical latent structure of generated points enables the consistent part decomposition of 3D shapes without any additional loss term [16], as will be presented in results.

4 Experiments

We evaluate our progressive growing method through 3 tasks; auto-encoding, generation, and completion of point clouds. In our experiment, four tree structured generators are included; **SRTDecoder** [7], **MRTDecoder** [7], **TopNet** [24], and **TreeGCN** [23]. Since we are interested in testing the progressive training of tree-structured generator (decoder), we use the shared PointNet [20] encoder and trained two version of decoders for each experiment; one is trained to directly generate point clouds from the global feature vector (GFV), whereas the other is trained with our progressive growing method. We implemented all of the networks in PyTorch [19] and experimented on Nvidia RTX 2080Ti and Titan RTX. Supplementary material contains evaluation metrics and the detailed method for applying our method to MRTDecoder.

4.1 Auto-encoder and *l*-GAN

For auto-encoding and generating point clouds, we trained neural networks for each of the three categories in the ShapeNet dataset [5]: *airplane*, *chair* and *car*. Following the prior work [1], we use 2048 points for each shape and split the dataset to train/validation/test with 85%-5%-10%.

Table 1 shows the quantitative comparison of the baseline and our method. The proposed learning method boosts the reconstruction quality in all metrics. The description of individual evaluation metrics is deferred in the supplementary material. We can clearly observe

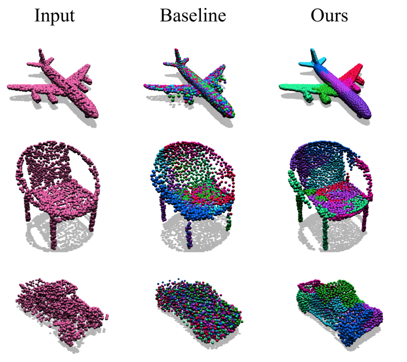

Figure 2: Auto-encoder results.

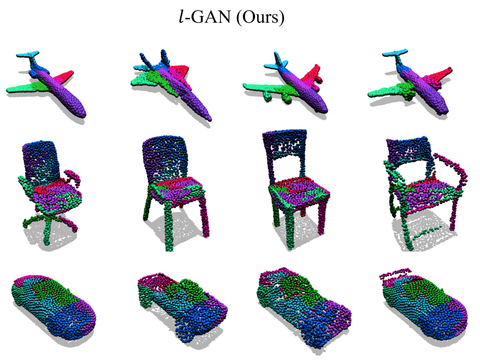

Figure 3: *l*-GAN results.

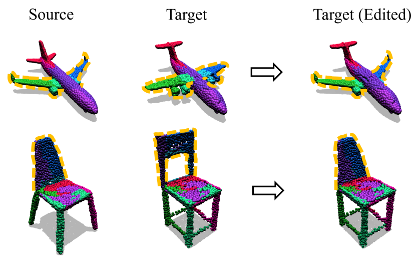

Figure 4: Part editing by index-wise copying.

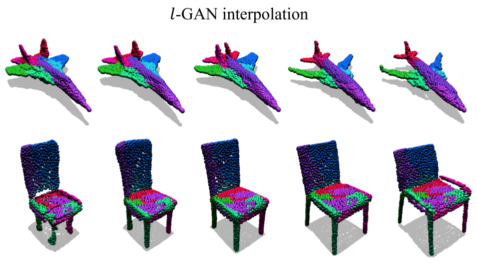

Figure 5: *l*-GAN interpolation results.

in Figure 2 that our reconstruction uniformly distributes the points with high-quality reconstruction, similar to the ground truth input. As a result, our method decreases the EMDs by a large margin and achieves higher F-scores. The benefit of progressive training is especially prominent in narrow chair legs or fine geometric details, where the baseline method suffers from irregular, blurry distribution of points.

The pre-trained auto-encoders can generate the shape by training *l*-GAN [1] to generate GFV which can be decoded into a point cloud. Table 1 also shows our method outperforms the baseline consistently in all metrics except some cases in MMD-CD. The qualitative results in Figure 3 show that our results create uniformly distributed high-quality shapes. The superior performance is demonstrated with the significant boost in JSD and EMD-based metrics.

Spatial decomposition. In addition to the high-quality generation, shapes created by our method exhibit consistent spatial structure. The consistent structure is visualized in Figure 2 and 3, where points are colored according to the intermediate ancestor nodes they are generated from. Clearly, all of the tree-structured networks enjoy the spatially-coherent part decomposition even though our method does not utilize any explicit supervision about the part labels [17, 18, 31]. The coherent decomposition is induced from our generation method, where the additional layer of nodes in the tree structure has been progressively guided to generate shapes in the spatial vicinity of its ancestors. In contrast, the end-to-end training of baseline methods with latent tree structure cannot bring the explicable structure into the ambient space.

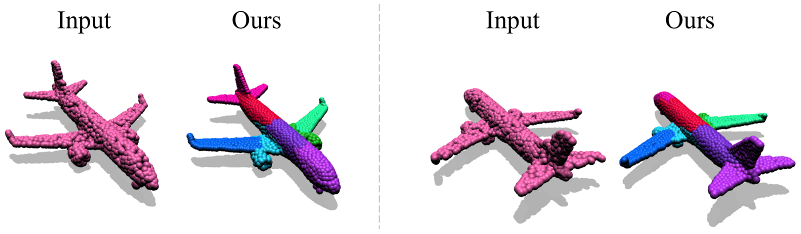

Figure 6: Outputs of the auto-encoder from different poses of the inputs. For this figure, we trained the networks by feeding point clouds with random rotation around the up-axis. Points of the same color are consistently located in the nearby space. However, they are not correlated to semantic information.

The consistent 3D correspondences can be utilized for part-level control of generated shapes. For example, we can cut and paste the points with the same indices as shown in Figure 4 (see the yellow dashed line). We can also generate the 3D point clouds of intermediate shapes by interpolating the latent code of l-GAN as shown in Figure 5. The consistent segmentation along the smooth transition between shapes indicates that the positions of intermediate nodes and the part decomposition are persistent over the interpolated shapes.

Our decomposition is a byproduct of increasing the spatial resolution with uniform density and does not contain semantic context. Our generation process produces higher resolution points in the neighborhood of the immediate ancestors and results in spatial partition given aligned data set. For example, if we perturb the initial poses as shown in Figure 6, the consistent point indices are bound to the same spatial position rather than the semantic parts.

Implementation details. For auto-encoding point clouds, we used the PointNet encoder implemented using 5 1-D convolutional layers with ReLU activation and max pooling operator in all experiments. The decoder networks are structured to be a tree of which the depth is 6 (SRTDecoder, MRTDecoder, TopNet) or 7 (TreeGCN). We used Adam optimizer [12] with $\beta_1 = 0.9$, $\beta_2 = 0.99$. The learning rate was chosen to be 10^{-4} with no decaying and the networks are trained with a batch size of 64 up to 300K iterations. When we apply our progressive method, we scheduled the learning phase with transition steps(list of the $step^{\ell}$ in the section 3) of [4K, 8K, 32K, 64K, 128K, 300K] for the 6-layered decoders and [4K, 8K, 16K, 32K, 64K, 128K, 300K] for the 7-layered decoder (TreeGCN).

For the generation of point clouds, we followed the l-GAN framework. We trained GANs for learning the distribution of the latent space following WGAN-GP [9] with the two kinds of pre-trained auto-encoders. The networks for the generators and discriminators are implemented as fully connected layers followed by ReLU activation with 2-layers and 3-layers each. In all experiments for 3D shape generation, the networks are trained up to 2K epochs with a batch size of 32, and the best models with the lowest JSD in the validation set were chosen for the test. We conducted the same tests 3 times and reported the average of results since they could be volatile to sampling noise for generating S_{out}. For the visualizations in Figure 2 and 3, we used MRTDecoder [7] for the airplane, TreeGCN [23] for the chair, and SRTDecoder [7] for the car.

4.2 Point Cloud Completion

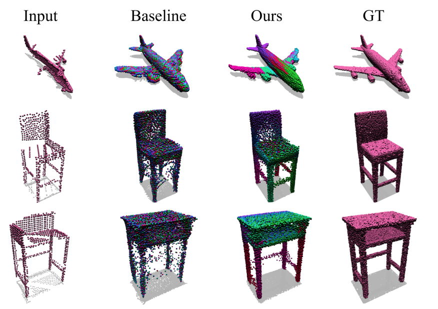

Figure 7: Completion results in PCN dataset.

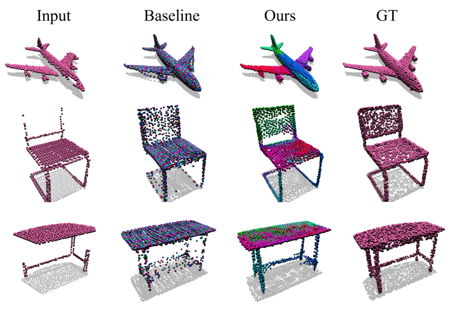

Figure 8: Completion results in TopNet dataset.

The progressive growing can faithfully reconstruct missing data given a partial point cloud. For the point cloud completion task, we used two datasets: PCN dataset [32] and TopNet dataset [24]. PCN dataset consists of 30974 pairs of partial and corresponding complete point cloud from 8 categories of ShapeNet [5]. TopNet dataset is composed of 28974 and 800 pairs for training and validation. The key difference between the two datasets is the density of the ground truth point clouds. Each complete point cloud contains 16384 points in the PCN dataset and 2048 points in the TopNet dataset. For the PCN dataset, we used the provided train/validation/test splits as [32]. Since the ground truth shapes of the test split are not provided in the TopNet dataset, we used the provided validation set as a test set and randomly sampled 600 pairs from the training set for the validation.

The progressive training of tree-structured decoders increases the performance of point cloud completion. Table 2 shows the quantitative results of the point cloud completion and a few exemplar results are available in Figure 7 and 8. Since the progressive generation effectively avoids falling in the local minima of CD, the points are uniformly distributed to the correct location. In contrast, the direct generation of the full point cloud fails to reconstruct the thin structures and fills the space recklessly which should be empty. As a result, the quantitative metric confirms superior performance of progressive growing, with lower CD and EMD, and higher F-score. Also, our method still enjoys the spatial decomposition visualized by color-coded indices of point clouds, which is not available in the baseline reconstruction.

Implementation details. The networks for the point cloud completion are designed in the same manner as the auto-encoders. In all experiments, we used the PointNet encoder and Adam optimizer similar to those of auto-encoder. The four tree networks were trained up to 300K steps, and we scheduled the transition steps following the auto-encoder experiments to apply our methods. Unlike auto-encoder experiments, we used 4-layered encoders and trained networks with a batch size of 32. Since the number of points in the GT shapes varies depending on the dataset, we regulated the number of output points to be equal to that of GT of the datasets by controlling the number of children for each node. We used TopNet [24] for the visualizations in Figure 7 and 8, and painted the generated point clouds according to the index of the intermediate nodes as before.

Table 2: Point cloud completion results. ↑: The higher the better, ↓: The lower the better. Better results are in bold. The average CD multiplied by 10^3, EMD multiplied by 10^2, and F-score for different threshold ds are reported. Ours are post-fixed by PG which is an acronym of *progressive growing*.

Dataset	Model	CD(↓)	EMD(↓)	F-Score(↑)		
				@1%	@2%	@3%
PCN	SRTDec	13.255	5.018	0.105	0.396	0.658
	SRTDec(PG)	**10.960**	**2.546**	**0.276**	**0.645**	**0.816**
	MRTDec	11.267	3.645	0.236	0.581	0.787
	MRTDec(PG)	**10.401**	**2.447**	**0.295**	**0.670**	**0.834**
	TopNet	10.548	2.948	0.260	0.646	0.829
	TopNet(PG)	**9.837**	**2.482**	**0.329**	**0.696**	**0.848**
	TreeGCN	9.876	2.916	0.292	0.692	**0.856**
	TreeGCN(PG)	**9.820**	**2.331**	**0.334**	**0.707**	0.853
TopNet	SRTDec	25.470	7.950	0.0504	0.226	0.432
	SRTDec(PG)	**23.503**	**3.923**	**0.0603**	**0.298**	**0.548**
	MRTDec	23.183	6.191	0.0650	0.280	0.513
	MRTDec(PG)	**22.830**	**3.770**	**0.0711**	**0.319**	**0.574**
	TopNet	**21.845**	3.902	0.0735	0.328	0.581
	TopNet(PG)	21.930	**3.625**	**0.0806**	**0.350**	**0.605**
	TreeGCN	**21.410**	3.854	**0.0843**	0.347	**0.610**
	TreeGCN(PG)	21.939	**3.518**	0.0831	**0.349**	0.608

5 Conclusions

In this paper, we propose a progressive learning method for the tree-structured point cloud generators. Motivated by previous works on progressive growing, we adapt the coarse-to-fine strategy for the point cloud generation. The transition to an increasing number of points is implemented by augmenting the intermediate nodes of the tree-structured network to generate points at each layer, then smoothly interpolating into the increased hierarchy of the layer. The hierarchy is gradually added as the learning phase progresses, and the point cloud is guided to distribute uniformly, avoiding local minima of Chamfer distance (CD). The effectiveness of the progressive growing is demonstrated with three tasks; point cloud auto-encoder, generation, and completion. The results not only show the superior performance of the proposed training method but also suggest a consistent structural relationship in the generated shapes which spatially decomposes parts without any prior labeling.

Acknowledgements.
This research was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2020R1C1C1008195), the National Convergence Research of Scientific Challenges through the National Research Foundation of Korea (NRF) funded by Ministry of Science and ICT (NRF-2020M3F7A1094300), and the BK21 FOUR program of the Education and Research Program for Future ICT Pioneers, Seoul National University in 2021.

References

[1] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas J. Guibas. Learning Representations and Generative Models for 3D Point Clouds. In *Proceedings of the 35th International Conference on Machine Learning, ICML*, pages 40–49, 2018.

[2] Mohammad Samiul Arshad and William J. Beksi. A Progressive Conditional Generative Adversarial Network for Generating Dense and Colored 3D Point Clouds. In Vitomir Struc and Francisco Gómez Fernández, editors, *8th International Conference on 3D Vision, 3DV 2020, Virtual Event, Japan, November 25-28, 2020*.

[3] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In Andrea Pohoreckyj Danyluk, Léon Bottou, and Michael L. Littman, editors, *Proceedings of the 26th Annual International Conference on Machine Learning, ICML 2009, Montreal, Quebec, Canada, June 14-18, 2009*.

[4] Ruojin Cai, Guandao Yang, Hadar Averbuch-Elor, Zekun Hao, Serge J. Belongie, Noah Snavely, and Bharath Hariharan. Learning Gradient Fields for Shape Generation. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part III*.

[5] Angel X. Chang, Thomas A. Funkhouser, Leonidas J. Guibas, Pat Hanrahan, Qi-Xing 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. *CoRR*, abs/1512.03012, 2015.

[6] Theo Deprelle, Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan C. Russell, and Mathieu Aubry. Learning elementary structures for 3d shape generation and matching. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*.

[7] Matheus Gadelha, Rui Wang, and Subhransu Maji. Multiresolution Tree Networks for 3D Point Cloud Processing. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part VII*.

[8] Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan C. Russell, and Mathieu Aubry. 3D-CODED: 3D Correspondences by Deep Deformation. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part II*.

[9] Ishaan Gulrajani, Faruk Ahmed, Martín Arjovsky, Vincent Dumoulin, and Aaron C. Courville. Improved Training of Wasserstein GANs. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*.

[10] Le Hui, Rui Xu, Jin Xie, Jianjun Qian, and Jian Yang. Progressive Point Cloud Deconvolution Generation Network. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XV*.

[11] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive Growing of GANs for Improved Quality, Stability, and Variation. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*.

[12] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, ICLR*, 2015.

[13] Wei-Sheng Lai, Jia-Bin Huang, Narendra Ahuja, and Ming-Hsuan Yang. Deep Laplacian Pyramid Networks for Fast and Accurate Super-Resolution. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*.

[14] Chun-Liang Li, Tomas Simon, Jason M. Saragih, Barnabás Póczos, and Yaser Sheikh. LBS Autoencoder: Self-Supervised Fitting of Articulated Meshes to Point Clouds. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*.

[15] Jianxin Lin, Yingxue Pang, Yingce Xia, Zhibo Chen, and Jiebo Luo. TuiGAN: Learning Versatile Image-to-Image Translation with Two Unpaired Images. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part IV*.

[16] Minghua Liu, Lu Sheng, Sheng Yang, Jing Shao, and Shi-Min Hu. Morphing and Sampling Network for Dense Point Cloud Completion. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*.

[17] Kaichun Mo, Shilin Zhu, Angel X. Chang, Li Yi, Subarna Tripathi, Leonidas J. Guibas, and Hao Su. PartNet: A Large-scale Benchmark for Fine-grained and Hierarchical Part-level 3D Object Understanding. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*.

[18] Kaichun Mo, Paul Guerrero, Li Yi, Hao Su, Peter Wonka, Niloy J. Mitra, and Leonidas J. Guibas. StructureNet: Hierarchical Graph Networks for 3D Shape Generation. *ACM Trans. Graph.*, 38(6):242:1–242:19, 2019.

[19] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. 2017.

[20] Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR*.

[21] Shunsuke Saito, Tomas Simon, Jason M. Saragih, and Hanbyul Joo. PIFuHD: Multi-Level Pixel-Aligned Implicit Function for High-Resolution 3D Human Digitization. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*.

[22] Muhammad Sarmad, Hyunjoo Jenny Lee, and Young Min Kim. RL-GAN-Net: A Reinforcement Learning Agent Controlled GAN Network for Real-Time Point Cloud Shape Completion. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[23] Dong Wook Shu, Sung Woo Park, and Junseok Kwon. 3D Point Cloud Generative Adversarial Network Based on Tree Structured Graph Convolutions. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[24] Lyne P. Tchapmi, Vineet Kosaraju, Hamid Rezatofighi, Ian Reid, and Silvio Savarese. TopNet: Structural Point Cloud Decoder. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[25] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-Resolution Image Synthesis and Semantic Manipulation With Conditional GANs. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, .

[26] Weiyue Wang, Duygu Ceylan, Radomír Mech, and Ulrich Neumann. 3DN: 3D Deformation Network. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, .

[27] Yifan Wang, Noam Aigerman, Vladimir G. Kim, Siddhartha Chaudhuri, and Olga Sorkine-Hornung. Neural Cages for Detail-Preserving 3D Deformations. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, .

[28] Yifan Wang, Federico Perazzi, Brian McWilliams, Alexander Sorkine-Hornung, Olga Sorkine-Hornung, and Christopher Schroers. A Fully Progressive Approach to Single-Image Super-Resolution. In *2018 IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2018, Salt Lake City, UT, USA, June 18-22, 2018*, .

[29] Yifan Wang, Shihao Wu, Hui Huang, Daniel Cohen-Or, and Olga Sorkine-Hornung. Patch-Based Progressive 3D Point Set Upsampling. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, pages 5958–5967, 2019.

[30] Guandao Yang, Xun Huang, Zekun Hao, Ming-Yu Liu, Serge J. Belongie, and Bharath Hariharan. PointFlow: 3D Point Cloud Generation With Continuous Normalizing Flows. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*.

[31] Fenggen Yu, Kun Liu, Yan Zhang, Chenyang Zhu, and Kai Xu. PartNet: A Recursive Part Decomposition Network for Fine-grained and Hierarchical Shape Segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*.

[32] Wentao Yuan, Tejas Khot, David Held, Christoph Mertz, and Martial Hebert. PCN: Point Completion Network. In *Proceedings of 2018 International Conference on 3D Vision (3DV)*, 2018.