3D POINT CLOUD GENERATION USING ADVERSARIAL TRAINING FOR LARGE-SCALE OUTDOOR SCENE

Takayuki Shinohara, Haoyi Xiu, Masashi Matsuoka

Tokyo Institute of Technology
Department of Architecture and Building Engineering
Yokohama, Japan

IGARSS 2021 - 2021 IEEE International Geoscience and Remote Sensing Symposium | 978-1-6654-0369-6/21/$31.00 ©2021 IEEE | DOI: 10.1109/IGARSS47720.2021.9554523

ABSTRACT

Three-dimensional (3D) point clouds are becoming an important part of the geospatial domain. During research on 3D point clouds, deep-learning models have been widely used for the classification and segmentation of 3D point clouds observed by airborne LiDAR. However, most previous studies used discriminative models, whereas few studies used generative models. Specifically, one unsolved problem is the synthesis of large-scale 3D point clouds, such as those observed in outdoor scenes, because of the 3D point clouds' complex geometric structure. In this paper, we propose a generative model for generating large-scale 3D point clouds observed from airborne LiDAR. Generally, because the training process of the famous generative model called generative adversarial network (GAN) is unstable, we combine a variational autoencoder and GAN to generate a suitable 3D point cloud. We experimentally demonstrate that our framework can generate high-density 3D point clouds by using data from the 2018 IEEE GRSS Data Fusion Contest.

Index Terms— Generative Adversarial Network, Variational Autoencoder, Deep Learning, Point Clouds, Airborne LiDAR

1. INTRODUCTION

Three-dimensional (3D) point clouds play an important role as 3D data structures in the geospatial domain; as a result, automatic analysis methods for point clouds have been studied for many years [1]. Recently, many deep-learning methods for 3D point clouds have been proposed in the field of computer vision [2]; most of them were discriminative models. However, some recent research on generative models for 3D point clouds has shown that generative models can link the semantics to the representation, and we can edit and design new objects.

A generative adversarial network (GAN) [3] is a generative model that can successfully learn data representation and generate realistic samples from complex underlying observed data distributions. GANs for 3D point clouds have been widely investigated; however, simple CAD data have been the main target [4, 5, 6]. GANs for 3D point clouds observed from outdoor scenes remain an unsolved problem because GAN-based generative pipelines are not stable [7] and 3D point clouds include complex geometry. Thus, we extended a GAN-based generative model to deal suitably with 3D point clouds observed from an outdoor scene. To ensure that the training was performed properly, we followed the adversarial variational autoencoder (A-VAE) pipeline [8]. An A-VAE consists of a variational autoencoder (VAE) and a GAN (Figure 1). VAEs extract a latent representation via training to reconstruct input data, and then GANs train to discriminate between real data and reconstructed data. By using this pipeline, we generated suitable large-scale 3D point clouds from the airborne LiDAR.

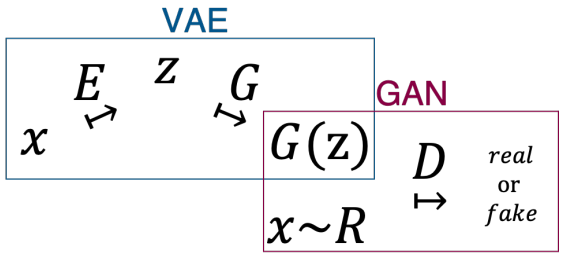

Fig. 1. Pipeline of the A-VAE. Our pipeline contains VAE and GAN.

2. PROPOSED METHOD

2.1. Variational Auto Encoder

The VAE is a generative deep-learning method based on an auto encoder (AE), which is trained to reconstruct its input. Generally, AEs map onto a narrow bottleneck layer from the input and map onto reconstructed data from a bottleneck layer. With successful training, this bottleneck layer provides a low-dimensional representation or latent code for each input data point. The encoder (E) is trained to map a data point x into a bottleneck layer as its latent representation z. The

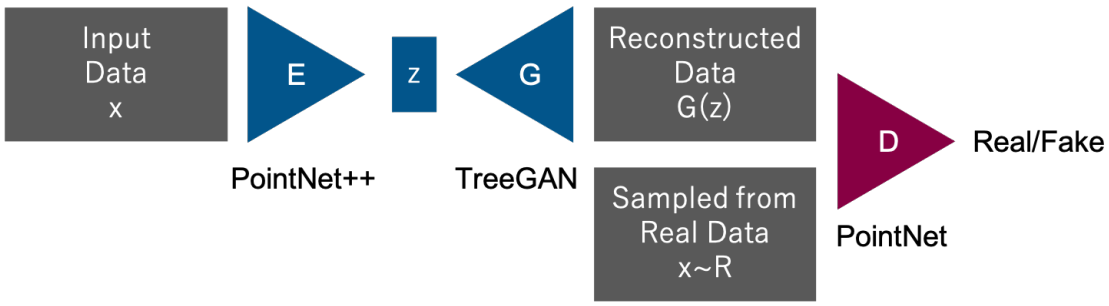

Fig. 2. A-VAE contains two generative models: VAE and GAN. The VAE includes the encoder(E) and the generator (G). The encoder produces bottleneck layer z, and the generator reconstructs input data from z. The GAN has a discriminator that estimates the distance between samples from real data ($x \sim R$) and fake data generated from the generator($G(z)$).

generator (G) can then produce a reconstruction $G(z)$ of x from the bottleneck layer z.

VAEs [9]—the generative models based on AEs—consider that the bottleneck layer has latent stochastic space z and optimizes the upper boundary on the negative log-likelihood of x:

$$\begin{aligned}\mathbb{E}_{x\sim p_d(x)}[-\log p(x)] <& \mathbb{E}_x[\mathbb{E}_{z\sim q(z|x)}[-\log p(x|z)]] \\ &+ \mathbb{E}_x\left[KL\left(q(z|x)\|p(z)\right)\right],\end{aligned}$$

in which $p_d(x)$ is the empirical distribution, $q(z|x)$ is the variational posterior distribution (i.e., the encoder E), $p(x|z)$ is the generative model from the generator G, $p(z)$ is the prior distribution, and KL is the Kullback–Leibler divergence. In a practical training process, $p(x|z)$ and $q(z|x)$ are parametrized with neural networks, and sampling from $q(z|x)$ is performed by a so-called "reparametrization trick". The total loss function used in the VAE's training process can be represented by two functions: the reconstruction function and the regularization function. The reconstruction function calculates the distance between the input data and reconstructed data by determining the difference between them, assuming that $p(x|z)$ is a normal distribution. The regularization function forces z generated from the $q(z|x)$ network from a prior distribution $p(z)$. This can be formulated as follows:

$$L_{\text{VAE}} = d_{Rec} + \text{regularization}, \tag{1}$$

in which d_{Rec} is the distance between the input data x and the reconstructed data $G(z)$. Here, the input data and reconstructed data are defined sets of points, and the distance between two sets of points (d_{Rec}) requires the idea of a permutation-invariant distance. A previous study [10] proposed two permutation-invariant distances between input point sets and reconstructed point sets. The first one is the *earth mover's distance* (EMD) [11], in which an EMD is an optimal transportation problem between one set and another. In this study, S_1 and S_2 were defined as sets of points. For two subsets of the same size $S_1 \subseteq \mathbb{R}^3$ and $S_2 \subseteq \mathbb{R}^3$, the EMD between these subsets is defined by

$$d_{EMD}(S_1, S_2) = \min_{\phi: S_1 \to S_2} \sum_{x \in S_1} \|x - \phi(x)\|_2,$$

in which ϕ is a bijection. As it is a loss function during the training process, EMD is differentiable almost everywhere. The second one is the *chamfer* (pseudo)-distance (CD), which represents the squared distance between each input point and its nearest neighbor in the reconstructed points, as follows:

$$d_{CH}(S_1, S_2) = \sum_{x \in S_1} \min_{y \in S_2} \|x - y\|_2^2 + \sum_{y \in S_2} \min_{x \in S_1} \|x - y\|_2^2.$$

Because CD is differentiable, it is more efficient to compute than the EMD. In this study, we combine these two distances as a loss function to train the VAE model:

$$d_{Rec}(S_1, S_2) = d_{EMD}(S_1, S_2) + d_{CH}(S_1, S_2). \tag{2}$$

The input to our VAE network was a set of point clouds with 2,048 points (2,048 $\times$ 3 matrix). The VAE network consisted of two networks—namely, an encoder and a decoder—that behaved like a generator (Figure 2). The encoder architecture followed the design principle of PointNet++ [12]. To remove the dependence on the input order, the max pool as a permutation-invariant function was used after the convolutions to produce a bottleneck layer with 1,024 dimensions. By using TreeGAN, our generator reconstructed the input data from the bottlenecks to produce 3D point clouds with the same dimensions as the input (2,048 $\times$ 3) [5].

2.2. Adversarial Training

Generally, VAEs can suitably generate data; however, the generated data typically lack diversity. As a result, we used the GAN framework and VAEs to ensure the diversity of the generated 3D point clouds (Figure 2). We used A-VAE, which uses adversarial training to force a particular distribution on the generated data space.

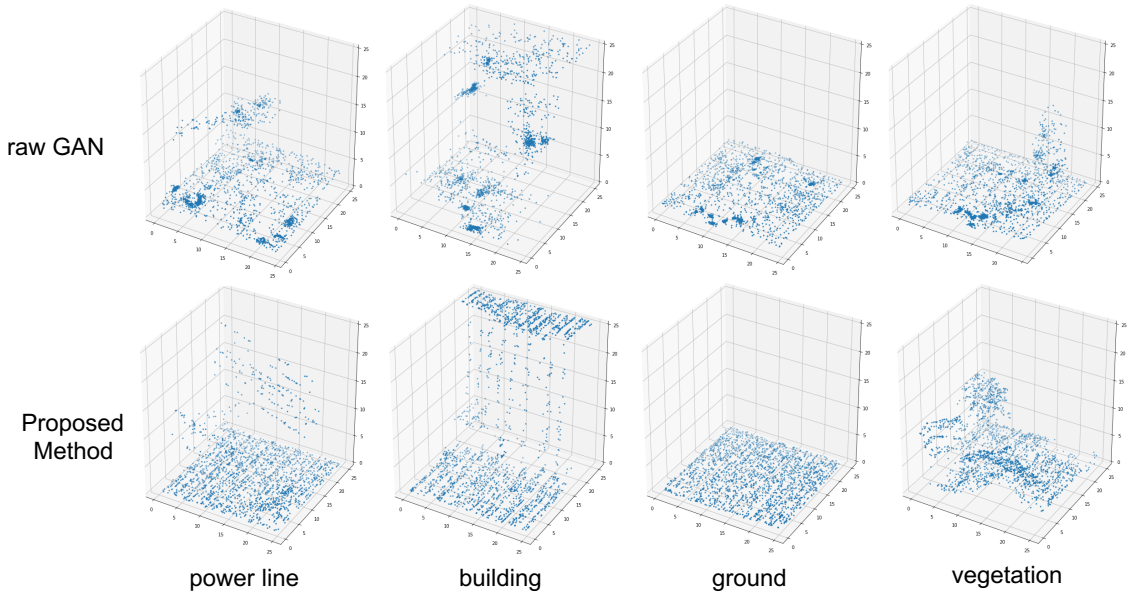

Fig. 3. Generated 3D point clouds including typical objects from our generator using randomly sampled z.

By using the A-VAE, it was assumed that an additional neural network, the discriminator D, was responsible for distinguishing between fake and real samples. The real samples were sampled from the assumed normal prior distribution $p(z)$—that is, the input data. The fake samples were generated via the decoder network $q(z|x)$ in the VAE.

The loss function of a generator, L_{gen}, is defined as

$$L_{gen} = -\mathbb{E}_z[D(G(z))], \tag{3}$$

in which G and D are the generator and discriminator, respectively, and z represents a bottleneck in the latent code distribution of the VAE.

We used the loss function for the discriminator D proposed in Wasserstein GAN [13]. The loss function of a discriminator, L_{disc}, is defined as

$$\begin{aligned} L_{disc} = \mathbb{E}_z[D(G(z))] - \mathbb{E}_{x\sim\mathcal{R}}[D(x)] \\ + \lambda_{gp}\mathbb{E}_{\hat{x}}[(\nabla_{\hat{x}}D(\hat{x})_2 - 1)^2]. \end{aligned} \tag{4}$$

in which z is the bottleneck, x represents real point clouds, $\mathcal{R}$ represents a real point cloud distribution, and the values of $\hat{x}$ are calculated from the real and fake point clouds. (4) also has a gradient penalty coefficient that satisfies the 1-Lipschitz condition [14] for each parameter in the discriminator, in which λ_{gp} is a weighting parameter.

The training pipeline characteristic of GAN models is performed by alternating updates of the parameters of the VAE (E and G) and discriminator D. The parameters for the discriminator D are updated by minimizing the function $L_D = -V(E, D)$. The parameters for the encoder E and generator G are optimized by minimizing the VAE loss function (2) and $V(E, D)$. Also, we used a PointNet-based network as the discriminator [15].

3. EXPERIMENTAL RESULTS

3.1. Implementation

We used an Adam optimizer for the encoder, generator, and discriminator networks with a learning rate of $\alpha = 10^{-4}$; the other coefficients were $\beta_1 = 0$ and $\beta_2 = 0.99$. The gradient penalty coefficient of the loss function (λ_{gp}) is set to 10 and the discriminator is updated five times per iteration; however, the generator is updated once per iteration. For the training and inference processes, we used Pytorch for the neural network implementation and TSUBAME3.0 for the computing environment.

3.2. Dataset

We trained the proposed network by using data from the 2018 IEEE GRSS Data Fusion Contest [1]. This public large-scale outdoor dataset of 3D point clouds was observed from airborne LiDAR. Because the 3D point clouds that we used had some noise, such as points that appeared to be underground, we performed denoising on the original 3D point clouds by using Open3D [16]. Because the original data was on the order of several hundred meters, it could not be loaded into the

[1] https://site.ieee.org/france-grss/2018/03/15/data-fusion-contest-2018-test-data-are-available/

GPU memory during the training process. As a result, dividing all the training data into smaller patches was necessary. In this study, we cut the small patches from the large original data into 25 m intervals without overlap. Because the number of 3D point clouds among the patches varied, we subsampled them and settled at 2,048 points.

3.3. Generated Point Clouds

Figure 3 (Proposed Method) shows the fake 3D point clouds that include typical objects generated from z by our trained TreeGAN-based generator. Notice that the 3D point clouds generated by the trained model reproduced the power lines, ground, buildings, and vegetation. Compared with the results of the GAN-only models (Figure 3 (raw GAN)), our model combines a VAE and GAN, demonstrating the possibility of a variety of 3D point cloud generation.

4. CONCLUSION

In this paper, we propose a generative model combining a VAE and GAN that can generate large-scale 3D point clouds observed by airborne LiDAR. Our generative models produced faithful samples and covered most of the ground truth distribution. Future work should include applications to other tasks, such as 3D point cloud reconstruction that uses a conditional GAN from a single image.

5. ACKNOWLEDGE

We gratefully acknowledge the benchmark data owners for providing the airborne LiDAR data. This work was partially supported by KAKENHI (19H02408). Numerical calculations were performed using the TSUBAME3.0 supercomputer at the Tokyo Institute of Technology.

6. REFERENCES

[1] B. Lohani and S. Ghosh, "Airborne lidar technology: A review of data collection and processing systems," *Proceedings of the National Academy of Sciences, India Section A: Physical Sciences*, vol. 87, pp. 567–579, 2017.

[2] Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu, and M. Bennamoun, "Deep learning for 3d point clouds: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2020.

[3] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio, "Generative adversarial nets," in *NeurIPS*, 2014.

[4] Chun-Liang Li, Manzil Zaheer, Yang Zhang, Barnabas Poczos, and Ruslan Salakhutdinov, "Point cloud gan," *arXiv preprint arXiv:1810.05795*, 2018.

[5] Dong Wook Shu, Sung Woo Park, and Junseok Kwon, "3d point cloud generative adversarial network based on tree structured graph convolutions," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[6] Jiajun Wu, Chengkai Zhang, Tianfan Xue, William T Freeman, and Joshua B Tenenbaum, "Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling," in *NeurIPS*, 2016.

[7] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen, "Improved techniques for training gans," *Advances in neural information processing systems*, vol. 29, pp. 2234–2242, 2016.

[8] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther, "Autoencoding beyond pixels using a learned similarity metric," in *International conference on machine learning*. PMLR, 2016, pp. 1558–1566.

[9] Diederik P Kingma and Max Welling, "Auto-encoding variational bayes," in *ICLR*, 2014.

[10] H. Fan, H. Su, and L. Guibas, "A point set generation network for 3d object reconstruction from a single image," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2463–2471.

[11] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas, "The earth mover's distance as a metric for image retrieval," *International journal of computer vision*, vol. 40, no. 2, pp. 99–121, 2000.

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

[13] Martin Arjovsky, Soumith Chintala, and Léon Bottou, "Wasserstein generative adversarial networks," in *ICML*, 2017.

[14] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville, "Improved training of wasserstein gans," in *Advances in neural information processing systems*, 2017, pp. 5767–5777.

[15] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *CVPR*, 2017.

[16] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun, "Open3D: A modern library for 3D data processing," *arXiv:1801.09847*, 2018.