GENERATION FOR UNSUPERVISED DOMAIN ADAPTATION: A GAN-BASED APPROACH FOR OBJECT CLASSIFICATION WITH 3D POINT CLOUD DATA

Junxuan Huang, Junsong Yuan, Chunming Qiao

Department of Computer Science and Engineering, University at Buffalo
{junxuanh,jsyuan,qiao}@buffalo.edu

ABSTRACT

Recent deep networks have achieved good performance on a variety of 3d points classification tasks. However, these models often face challenges in "wild tasks" where there are considerable differences between the labeled training/source data collected by one Lidar and unseen test/target data collected by a different Lidar. Unsupervised domain adaptation (UDA) seeks to overcome such a problem without target domain labels. Instead of aligning features between source data and target data, we propose a method that uses a Generative Adversarial Network (GAN) to generate synthetic data from the source domain so that the output is close to the target domain. Experiments show that our approach performs better than state-of-the-art UDA methods in three popular 3D object/scene datasets (i.e., ModelNet, ShapeNet and ScanNet) for cross-domain 3D object classification.

Index Terms— 3D object classification, GAN, Unsupervised domain adaptation,

1. INTRODUCTION

3D object classification is the most fundamental task in computer vision [1] and plays a central role in a number of applications, e.g., robots, self-driving cars or virtual reality. Despite their previous success, Deep Neural Network (DNN) based approach usually requires a large amount of labeled point clouds data for training. However, point clouds data can be captured with different sensors (e.g. 64 beam lidar, 128 beam lidar and RGB-D camera) which have different resolution and produce different 3D sampling patterns. In addition, when objects are scanned from LiDAR, some parts can be lost or occluded (e.g.legs of chair) while some training datasets are generated from 3D polygonal models with a complete and uniform surface. As Fig. 1 shows, those datasets have different geometric features, which may lead to a performance drop when we train and test a classification model with different types of sensor signals.

To reduce the domain gap between training and test point clouds, labeling 3D object in the new domain is the most straightforward solution, which is unfortunately time-consuming and expensive. The objective of unsupervised domain adaptation (UDA) is to leverage rich annotations in a source domain to achieve a good performance in a target domain having few annotations. Some of the existing DA methods have focused on mapping features into a shared subspace or minimizing instance-level distances such as MMD [2], CORAL [3]. Other adversarial-training DA methods, like DANN [4], ADDA [5], MCD [6] have tried to use adversarial-training to select domain invariant features during training so that the trained model could perform better in the target domain. PointDan [7] designed a Self-Adaptive Node Construction for aligning 3D local features with points cloud data.

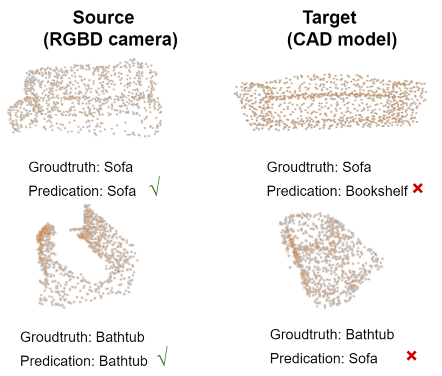

Fig. 1. Point clouds acquired from different sensors and being misclassified

In this paper, we tackle the problem of unsupervised domain adaptation (UDA) for 3D object classification using point cloud data, where the target domain is entirely unlabelled. Inspired by StyleGAN[8], Our approach chooses to generate synthetic data from a source domain according to the target domain data pattern. Unlike other adversarial-training DA methods, we fully utilize the advantage of adversarial-training by generating synthetic data while keeping the label of the source domain. Extensive experiments that involve the use of different pairs of a dataset to perform cross-domain 3D object classification tasks show that our approach's predicting accuracy is 2.6% higher than PointDan on average, 2.8% higher than MCD, and 4.6% higher than ADDA.

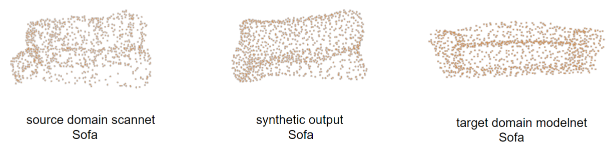

Fig. 2. The distribution of points cloud in target object (right) looks much more uniform than source object (left), and our synthetic outputs keep the shape of the source domain object, but the distribution of 3D points are more similar to the target domain object.

As Fig. 2 shows, our model generates a synthetic sofa with a very similar shape to the sofa from the source dataset and its point distribution looks much closer to the target dataset object compared with the sofa from the source dataset. In summary, we have a novel design that generates a synthetic 3D points cloud dataset using GAN with a latent space reconstruction module and an additional discriminator to lower the domain gap between source and target dataset.

2. PROPOSED GAN-BASED DA METHOD

When we directly use a classifier trained on the source dataset over the target dataset, some objects may be misclassified due to the domain gap. As Fig. 1 shows, a sofa from Modelnet may be classified as a bookshelf if we use a classifier trained on Scannet without any domain adaption mechanism . To overcome the domain gap between different point cloud datasets, we propose a generative model, which takes point clouds from the source dataset and target dataset as input and then tries to maintain the shape of an object while keeping the label information when generating the synthetic object according to the target domain dataset's distribution. This objective is that classifier could learn the distribution of point clouds in the target domain and performs better in the test of the target dataset.

As Fig. 3 shows, our model architecture consists of four parts: generator, latent reconstruction module, discriminator, and feature encoder/decoder. In the training, source domain object and target domain object will go through a shared encoder. The encoded features from source domain will be sent to the generator and a discriminator tries to distinguish features from generator or target domain. Following the idea in [9] of adding multimodal information to the model, we also have Gaussian samples z for latent condition input to the generator. To force the generator to use the Gaussian samples z, we introduce a VAE encoder to recover z from the synthetic output. In addition, inspired by [10], in order to enhance the quality of output object from $\mathbf{G}$, we have an additional discriminator, a classifier $\mathbf{C}$ in training the model.

2.1. Definition and Notations

We consider an unsupervised domain adaptation (UDA) setting, a set point clouds from the source domain is represented as $(\mathbf{X}^s \triangleq \{\mathbf{x}_i^s\}_{i=1...N}, y^s)$, where $\mathbf{x}_i^s \in \mathbb{R}^3$ is a 3D point and

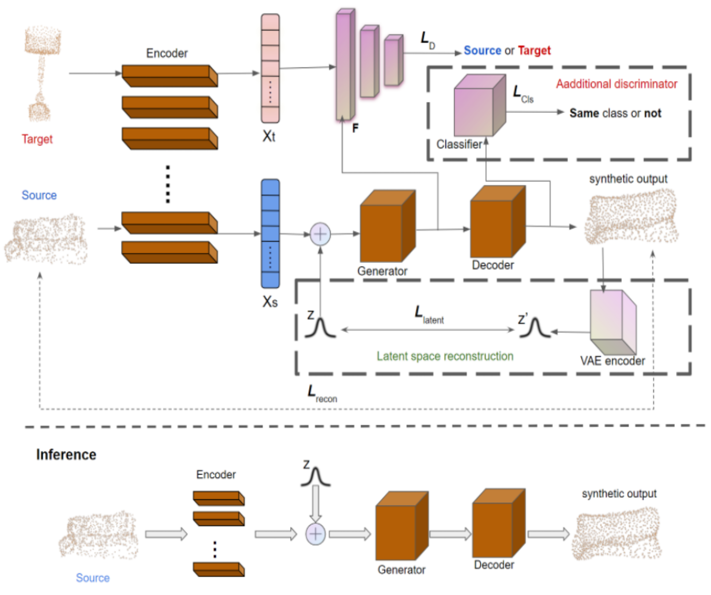

Fig. 3. Overview of the training and inference.

N is the number of points in the point cloud; $y \in \{1, 2, ..., k\}$ is the ground-truth label of the point clouds, where k is the number of classes. In UDA, we have access to a set of labeled LiDAR point clouds, from the source domain and a set of unlabeled LiDAR point clouds $(\mathbf{X}^t \triangleq \{\mathbf{x}_i^t\}_{i=1...N})$ in a target domain. It is assumed that two domains are sampled from the distributions $P_s(\mathbf{X}^s)$ and $P_t(\mathbf{X}^t)$ while the distribution $P_s \neq P_t$. We use classification as the task of domain adaption and denote the classification network as $\mathbf{C}_{\boldsymbol{\theta}}(\cdot) \triangleq \{C_{\boldsymbol{\theta},j} | j = 1...k\}$, whose input is a point clouds object $\mathbf{X}$ and output is a probability vector $\mathbf{C}_{\boldsymbol{\theta}}(\mathbf{X})$. Our approach tend to generate synthetic object $\mathbf{X}'^s$ from $\mathbf{X}^s$ and for the generative adversarial network we have encoder $\mathbf{E}$, decoder $\mathbf{D}$, generator $\mathbf{G}$ and discriminator $\mathbf{F}$. The process is denoted as $\mathbf{X}' = D(G(E(\mathbf{X})))$

2.2. Learning mapping of point cloud to latent space

We obtain feature vector $\mathbb{X}_s$ of input point clouds by training an autoencoder $\mathbf{E}_{\text{AE}}$. The object $\mathbf{X}^s$ from the source dataset is encoded as feature vector $\mathbb{X}_s$ and decoder $\mathbf{D}_{\text{AE}}$ reconstructs object $\tilde{\mathbf{X}}$ from the latent feature vector $\mathbb{X}_s$. The encoder and decoder are trained with the reconstruction loss, and we choose Earth Mover's Distance (EMD) to measure the distance between reconstructed object $\tilde{\mathbf{X}}$ and input object $\mathbf{X}^s$.

$$\mathcal{L}^{\text{recon}} = d^{\text{EMD}}(\mathbf{X}^s, \mathbf{D}_{\text{AE}}(\mathbf{E}_{\text{AE}}(\mathbf{X}^s))), \tag{1}$$

As for the object $\mathbf{X}^t$ from the target dataset, instead of training another autoencoder for the target domain, we directly feed $\mathbf{X}^t$ to $\mathbf{E}_{\text{AE}}$ because [11] showed that doing this would achieve a better performance in subsequent adversarial training.

2.3. Latent space reconstruction

Like most of the generative adversarial networks, we set a min-max game between generator and discriminator. The generator is trained to fool the discriminator so that the discriminator fails to tell if the latent vector comes from the source domain $\mathbf{X}^s$ or target domain $\mathbf{X}^t$. Use Earth Mover's Distance (EMD) to restrict the synthetic outputs can make it close to the input's shape, but we do not want the synthetic object to having the exact shape of input. Because we are building a synthetic dataset which means the variety is also significant[9]. So we bring a random sampled variable $\mathbf{z}$ into our model and a Variational Autoencoder (VAE) is trained to encode synthetic objects to recover latent input vector, encouraging the use of conditional mode input $\mathbf{z}$.

Formally, the latent representation of the source domain input $\mathbf{x}_s = E_{\text{AE}}(\mathbf{X}^s)$, along with random sampled variable $\mathbf{z}$ from a standard Gaussian distribution $\mathcal{N}(0, \mathcal{I})$. Thus, a latent representation $\tilde{\mathbf{x}}_t = G(\mathbf{x}_s, \mathbf{z})$ will be generated by generator. Then discriminator will try to distinguish between $\tilde{\mathbf{x}}_t$ and $\mathbf{x}_t = E_{\text{AE}}(\mathbf{X}^t)$. The mode encoder E_z will encode the synthetic output $\tilde{\mathbf{X}}^t$, which is decoded from the latent representation $\tilde{\mathbf{X}}^t = D_{\text{AE}}(\tilde{\mathbf{x}}_t)$, to reconstruct the conditional input $\tilde{\mathbf{z}} = E_z(\tilde{\mathbf{X}}^t)$.

2.4. Train the generator with classification loss

There is no guarantee that the synthetic output will be the same class as input object, though we use a reconstruction loss during training to make sure the output point set will be close to the input point set in Earth Mover's Distance (EMD). So we add a discriminator to utilize label information from the source dataset fully. The classifier $\mathbf{C}$ will predict the class of output point set and compare it with the ground truth label in the source dataset. The loss will be backward to the generator to encourage it generates synthetic objects in the same class as input objects from the source dataset.

2.5. Overall loss function and training

To optimize GAN's output object quality, we set a min-max game between the generator, the discriminator, and the classifier. Given training examples of source domain object $\mathbf{X}^s$, and Gaussian samples $\mathbf{z}$, we seek to optimize the following training losses over the generator $\mathbf{G}$, the discriminator $\mathbf{F}$, the encoder $\mathbf{E}_z$ and the classifier $\mathbf{C}$:

Adversarial loss. For trianing of the generator and discriminator we add the an adversarial loss and we implement least square GAN [12] for stabilizing the training. Hence, the adversarial losses will be minimized for the generator and the discriminator are defined as:

$$\begin{aligned}\mathcal{L}_F^{\text{GAN}} &= \mathbb{E}_{\mathbf{X}^t \sim p(\mathbf{X}^t)}[F(E_{\text{AE}}(\mathbf{X}^t)) - 1]^2 \\ &\quad + \mathbb{E}_{\mathbf{X}^s \sim p(\mathbf{X}^s), \mathbf{z} \sim p(\mathbf{z})}[F(G(E_{\text{AE}}(\mathbf{X}^s), \mathbf{z}))]^2\end{aligned} \tag{2}$$

$$\mathcal{L}_G^{\text{GAN}} = \mathbb{E}_{\mathbf{X}^s \sim p(\mathbf{X}^s), \mathbf{z} \sim p(\mathbf{z})}[F(G(E_{\text{AE}}(\mathbf{X}^s), \mathbf{z})) - 1]^2, \tag{3}$$

where $\mathbf{X}^t \sim p(\mathbf{X}^t)$, $\mathbf{X}^s \sim p(\mathbf{X}^s)$ and $\mathbf{z} \sim p(\mathbf{z})$ denotes samples drawn from the set of complete point sets, the set of partial point sets, and $\mathcal{N}(0, \mathcal{I})$.

Reconstruction loss. To make the output object similar to the input object in shape, we add a reconstruction loss to encourage the generator to reconstruct the input so that the output object is more likely to be considered the same class as the input object. Here we use Earth Mover's Distance (EMD) to measure the distance between the reconstructed object $\tilde{\mathbf{X}}$ and the input object $\mathbf{X}^s$.

$$\mathcal{L}_G^{\text{recon}} = \mathbb{E}_{\mathbf{X}^s \sim p(\mathbf{X}^s), \mathbf{z} \sim p(\mathbf{z})}\left[d^{\text{EMD}}(\mathbf{X}^s, D_{\text{AE}}(G(E_{\text{AE}}(\mathbf{X}^s), \mathbf{z}))\right]), \tag{4}$$

Latent space reconstruction. A reconstruction loss on the $\mathbf{z}$ latent space is also added to force G to use the conditional mode vector $\mathbf{z}$ in generate output object:

$$\mathcal{L}_{G,E_z}^{\text{latent}} = \mathbb{E}_{\mathbf{X}^s \sim p(\mathbf{X}^s), \mathbf{z} \sim p(\mathbf{z})}[\|\mathbf{z}, E_z(D_{\text{AE}}(G(E_{\text{AE}}(\mathbf{X}^s), \mathbf{z})))\|_1], \tag{5}$$

Classification loss. To restrict the class of output object, we added the classification loss to the classifier and the generator.

$$\mathcal{L}_C^{\text{Cls}} = \mathbb{E}_{\mathbf{X}^s \sim p(\mathbf{X}^s)}[-\sum_{k=1}^{K} l_k \log{(C(\mathbf{X}^s))}] \tag{6}$$

$$\mathcal{L}_G^{\text{Cls}} = \mathbb{E}_{\mathbf{X}^s \sim p(\mathbf{X}^s), \mathbf{z} \sim p(\mathbf{z})}[-\sum_{k=1}^{K} l_k \log{(C(D_{\text{AE}}(G(E_{\text{AE}}(\mathbf{X}^s), \mathbf{z}))))}] \tag{7}$$

where l_k is the k-th label among all classes in source dataset The full objective function for training the domain transfer network is described as:

$$\underset{(G,E_z,C)}{\text{argmin}}\ \underset{F}{\text{argmax}}\ \mathcal{L}_F^{\text{GAN}} + \mathcal{L}_G^{\text{GAN}} + \alpha\mathcal{L}_G^{\text{recon}} + \beta\mathcal{L}_{G,E_z}^{\text{latent}} + \gamma\mathcal{L}_G^{\text{Cls}}, \tag{8}$$

where α, β and γ are importance weights for the reconstruction loss, the latent space reconstruction loss and classification loss respectively.

In choosing the importance weights, α controls how similar the shape of synthetic output object with the shape of the input object in 3D space, β determines how close the synthetic output object with input object in latent space, and γ influences how much probability the synthetic object will be considered as same class as input object under classifier.

2.6. Network implementation

In the experiments, each point cloud object is set to 1024 points the VAE which follows [13, 14], using PointNet[1] as the encoder and a 3-layer MLP as the decoder. The autoencoder encodes a point set into a latent vector of fixed dimension $|\mathbf{x}| = 256$ with PointNet[1]. We use 3-layer MLP for both generator G and discriminator F. The classifier is also using Pointnet structure. To train the VAE, we use the Adam optimizer[15] with an initial learning rate 0.0005, $\beta_1 = 0.9$ and train 2000 epochs with a batch size of 200. To train the

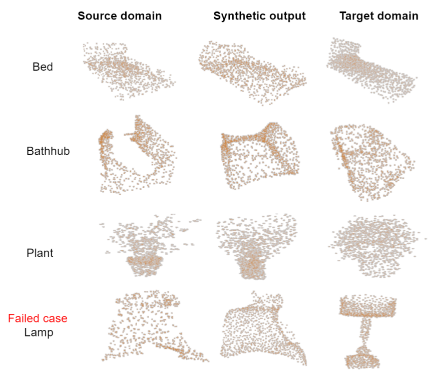

Fig. 4. Visualization for source domain object, target domain object and synthetic object

autoencoder we use the Adam optimizer with an initial learning rate 0.0005, $\beta_1 = 0.5$ and train for a maximum of 1000 epochs with a batch size of 32. The parameters of the pre-trained autoencoder and VAE are fixed during GAN training. To train the GAN, we use the Adam optimizer with an initial learning rate 0.0005, $\beta_1 = 0.5$ and train for a maximum of 1000 epochs with a batch size of 50. The classifier used to train the GAN was pre-trained on the source dataset with the Adam optimizer in an initial learning rate 0.0001 for 200 epochs.

3. EXPERIMENTS

3.1. Datasets

We verify our domain adaption model on three public point cloud datasets: shapenet [16], scannet [17] and modelnet [18]. Following the same setting as PointDA-10 [7], we choose ten common classes among three datasets. We consider six types of adaptation scenarios which are **M** $\rightarrow$ **S**, **M** $\rightarrow$ **S***, **S** $\rightarrow$ **M**, **S** $\rightarrow$ **S***, **S*** $\rightarrow$ **M** and **S*** $\rightarrow$ **S**, where **M**, **S** and **S*** represent subset of Modelnet, Shapenet and Scannet respectively.

Table 1. Quantitative classification results (%) on PointDA-10 Dataset[7].

	M→S	M→S*	S→M	S→S*	S*→M	S*→S	Avg
w/o Adapt	42.5	22.3	39.9	23.5	34.2	46.9	34.9
MMD[19]	57.5	27.9	40.7	26.7	47.3	54.8	42.5
DANN[4]	58.7	29.4	**42.3**	30.5	48.1	56.7	44.2
ADDA[5]	61.0	30.5	40.4	29.3	48.9	51.1	43.5
MCD[6]	62.0	31.0	41.4	31.3	46.8	59.3	45.3
PointDAN[7]	62.5	31.2	41.5	31.5	46.9	59.3	45.5
Ours	**62.8**	**36.5**	41.9	**31.6**	**50.4**	**65.7**	**48.1**
Supervised	90.5	53.2	86.2	53.2	86.2	90.5	76.6

M means ModelNet and **S** denotes ShapeNet while **S*** represents ScanNet.

3.2. Experiments Setup

We choose PointNet [1] as the backbone of the classifier. The learning rate is set to 0.0001 under the weight decay 0.0005. All models have been trained for 200 epochs of batch size 64 in both source domain and synthetic datasets. **Baselines:** In our experiments, we evaluate the performance when the model is trained only by source training samples (**w/o Adapt**). We also compare our model with five general-purpose UDA methods including: Maximum Mean Discrepancy (**MMD**) [19], Adversarial Discriminative Domain Adaptation (**ADDA**) [5], Domain Adversarial Neural Network (**DANN**) [4], Maximum Classifier Discrepancy (**MCD**) [6] and **PointDAN** [7] in the same training policy. Finally, we show the performance of a full supervised method (**Supervised**). As expected, supervised performs better than all unsupervised methods.

As shown from table 1, our approach can improve over all five existing UDA models in all six scenarios. In particular, for **S*** $\rightarrow$ **S**, our approach achieves 65.7% predicting accuracy, which represents about 10.8% improvement over the best result from the existing approaches. For **S***→**M**, our model also has over 5.7% improvement. Note that, among the three different datasets, Scannet (**S***) is the most challenging dataset because Scannet's point cloud objects are scanned from the real world, while the other two datasets are generated from 3D polygonal models. This implies that the difference between **S*** and **S**, and between **S*** and **M** are more significant from the observation and our approach is better at generating synthetic objects by minimizing the gap in point distributions between the source and target datasets. However, in some corner cases, the adaption may fail because of the huge shape difference of one category between source and target dataset Fig. 4.

Table 2. Ablation analysis

	AE	**L**	**C**	M→S	M→S*	S→M	S→S*	S*→M	S*→S	Avg
w/o Adapt				42.5	22.3	39.9	23.5	34.2	46.9	34.9
only AE	√			59.5	33.5	34.2	16.1	43.3	55.4	40.3
AE+L	√	√		62.6	34.1	40.4	29.1	49.6	64.3	46.7
GFA	√	√	√	**62.8**	**36.5**	**41.9**	**31.6**	**50.4**	**65.7**	**48.1**
Supervised				90.5	53.2	86.2	53.2	86.2	90.5	76.6

AE means use autoencoder with reconstruction loss in model ,**L** denotes latent space reconstruction with VAE , **C** represents the additional discriminator, a classifier.

Ablation Study Setup: To study the influence of latent reconstruction module and additional discriminator, we first construct a model that only keeps encoder/decoder and generator. From table 2, we can see this ablated model could only slightly improve the classifier over the none-adaption case. After applying the latent space reconstruction module to this ablated model, the classifier's performance significantly increases in all six scenarios due to the high fidelity synthetic output. Even the ablated model can outperform the existing models most of the time, but adding classifier **C** to our model as an additional discriminator will further improve the result.

4. CONCLUSION

We have proposed a novel generative approach to unsupervised domain adaptation in the 3D classification task. The basic idea is to transfer source training data into the style of target domain rather than selecting domain invariant feature or implementing feature alignment. Furthermore, we implemented latent reconstruction module and an addition discriminator for enhancing the performance.We have demonstrated our approach's superiority over the state-of-the-art domain adaptation methods in three datasets.

5. REFERENCES

[1] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, vol. 1, no. 2, pp. 4, 2017.

[2] Karsten M Borgwardt, Arthur Gretton, Malte J Rasch, Hans-Peter Kriegel, Bernhard Schölkopf, and Alex J Smola, "Integrating structured biological data by kernel maximum mean discrepancy," *Bioinformatics*, vol. 22, no. 14, pp. e49–e57, 2006.

[3] Baochen Sun and Kate Saenko, "Deep coral: Correlation alignment for deep domain adaptation," in *Proceedings of the European Conference on Computer Vision*, 2016.

[4] Yaroslav Ganin and Victor Lempitsky, "Unsupervised domain adaptation by backpropagation," *arXiv preprint arXiv:1409.7495*, 2014.

[5] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell, "Adversarial discriminative domain adaptation," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2017.

[6] Kuniaki Saito, Kohei Watanabe, Yoshitaka Ushiku, and Tatsuya Harada, "Maximum classifier discrepancy for unsupervised domain adaptation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3723–3732.

[7] Can Qin, Haoxuan You, Lichen Wang, C.-C. Jay Kuo, and Yun Fu, "Pointdan: A multi-scale 3d domain adaption network for point cloud representation," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., pp. 7190–7201. Curran Associates, Inc., 2019.

[8] Tero Karras, Samuli Laine, and Timo Aila, "A style-based generator architecture for generative adversarial networks," 2019.

[9] Rundi Wu, Xuelin Chen, Yixin Zhuang, and Baoquan Chen, "Multimodal shape completion via conditional generative adversarial networks," in *The European Conference on Computer Vision (ECCV)*, August 2020.

[10] Lanlan Liu, Michael Muelly, Jia Deng, Tomas Pfister, and Li-Jia Li, "Generative modeling for small-data object detection," 2019.

[11] Mitra NJ Chen X, Chen B, "Unpaired point cloud completion on real scans using adversarial training.," *International Conference on Learning Representations (ICLR)*, 2020.

[12] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley, "Least squares generative adversarial networks," 2017, pp. 2794–2802.

[13] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas, "Learning representations and generative models for 3d point clouds," 2018, pp. 40–49.

[14] Xuelin Chen, Baoquan Chen, and Niloy J. Mitra, "Unpaired point cloud completion on real scans using adversarial training," 2020.

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

[16] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al., "ShapeNet: An information-rich 3D model repository," *arXiv preprint arXiv:1512.03012*, 2015.

[17] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Niener, "ScanNet: Richly annotated 3D reconstructions of indoor scenes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5828–5839.

[18] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao, "3d shapenets: A deep representation for volumetric shapes," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1912–1920.

[19] Mingsheng Long, Jianmin Wang, Guiguang Ding, Jiaguang Sun, and Philip S Yu, "Transfer feature learning with joint distribution adaptation," in *Proceedings of IEEE International Conference on Computer Vision*, 2013.