

Generalized Autoencoder for Volumetric Shape Generation

Yanran Guan Tansin Jahan Oliver van Kaick

School of Computer Science, Carleton University, Canada

{yanran.guan,tansin.jahan,oliver.vankaick}@carleton.ca

Abstract

We introduce a 3D generative shape model based on the generalized autoencoder (GAE). GAEs learn a manifold latent space from data relations explicitly provided during training. In our work, we train a GAE for volumetric shape generation from data similarities derived from the Chamfer distance, and with a loss function which is the combination of the traditional autoencoder loss and the GAE loss. We show that this shape model is able to learn more meaningful structures for the latent manifolds of different categories of shapes, and provides better interpolations between shapes when compared to previous approaches such as autoencoders and variational autoencoders.

1. Introduction and related work

In recent years, significant research effort has been devoted to the development of generative models of 3D shapes based on deep neural networks, since these models have the potential of facilitating applications such as 3D object recognition, 3D modeling, and shape completion. This research effort has concentrated on investigating the two main aspects of these models: (i) the representation used for encoding 3D shapes, and (ii) the neural network architecture and learning framework for training the generative model.

Regarding the shape representation, previous work has investigated different encodings for 3D objects such as volumetric grids [3, 12, 22, 23], distance fields [5, 6, 14], point sets [1], parameterization atlases [9], and part-based representations [4, 11, 13, 24]. When considering specifically volumetric shape representations, early work learned generative models with networks such as autoencoders (AEs) or similar architectures [23], while subsequent efforts use approaches more tailored towards generative tasks such as variational autoencoders (VAEs) [3] and generative adversarial networks (GANs) [22, 12].

This ongoing investigation raised the important question of what constitutes a good 3D generative model, concluding that a generative model should not only be able to reconstruct well the shapes that were used for training the

model, but should also be able to generalize. Generalization implies that the model should at least enable a meaningful interpolation between shapes, possibly even synthesizing shapes that are novel, *i.e.*, surprisingly distinct from the training shapes. Although work on generative models of images has shown that architectures such as VAEs and even GANs can mainly perform a form of extrapolation, generating new data with limited novelty [10], we can still leverage the interpolatory and extrapolatory capabilities of these models for generating shapes distinct from the training data.

In this regard, AEs are easy to train and provide good-quality reconstructions of the training shapes. However, interpolation results are often not satisfactory, since there are no constraints that lead to the learning of a meaningful latent manifold of shapes. VAEs address this problem by effectively learning a probability distribution over a variable rather than predicting single point estimates. However, the synthesized shapes are commonly of low quality, similar to the problem of blurry images sampled from VAEs [8]. On the other hand, GANs learn models tailored to the generative task, but these models are difficult to train. The optimization can easily get stuck in a local minimum, requiring training the network multiple times until a satisfactory model is learned. In addition, GANs can be easily sampled, but the latent encoding of a shape cannot be easily derived from the standard GAN architecture without an encoder.

In this paper, we investigate an alternative architecture for volumetric 3D shape generation: the generalized autoencoder (GAE). Originally introduced by Wang *et al.* [20], GAEs learn a latent manifold of the data from data relationships explicitly provided during training. Specifically, we consider data similarities along with a weighting function in the loss function used for training the GAE. GAEs have certain advantages over AEs, VAEs, or GANs, such as the possibility of explicitly guiding the construction of the latent manifold and being able to map input shapes to this manifold. At the same time, like AEs, GAEs are much easier to train, as they involve training only a simple autoencoder architecture with a single loss function.

We use GAEs for 3D volumetric shape generation by estimating the similarity of data points with the Chamfer dis-

tance [7]. We also introduce a loss function that is a combination of the traditional AE loss and the GAE loss [20]. With experiments on selected categories of shapes, we demonstrate that the GAE loss enables learning a more meaningful manifold for 3D shapes than the traditional AE loss, while modifying the loss into the combined function provides better reconstructions of the training shapes. Note that we demonstrate the GAE in the context of shape generation with a volumetric grid. However, there are no inherent limitations for applying this learning framework to autoencoders based on other shape representations.

In summary, our contributions are as follows:

- We introduce a 3D generative model based on the GAE [20], which allows one to control the latent manifold learned by the model.
- We guide the construction of a latent manifold of 3D shapes with data similarities computed via the Chamfer distance, and train the model with a loss that is the combination of the traditional AE and GAE losses. We show that this model leads to more meaningful manifold structures and better interpolations between shapes when compared to previous approaches.

2. GAE for 3D shape generation

In this section, we introduce the 3D generalized autoencoder (3D-GAE). We first define the GAE [20], and then explain how to adapt the GAE for 3D shape generation.

2.1. Generalized autoencoder (GAE)

The GAE, proposed by Wang *et al.* [20] as an extension of the traditional AE, consists of an encoder and a decoder, with the encoder mapping an input to a reduced latent representation and the decoder reconstructing the input from the learned latent representation. However, different from traditional AEs, the GAE is able to learn a manifold latent space by exploring data relations in the training set. Specifically, the traditional AE learns how to reconstruct a single instance \mathbf{x}_i according to the error $\|\mathbf{x}_i - \mathbf{x}_i'\|^2$, where \mathbf{x}_i' is the reconstruction of \mathbf{x}_i . On the other hand, the GAE uses each instance \mathbf{x}_i to reconstruct a set of instances $\Omega_{\mathbf{x}_i}$, called the *reconstruction set* of \mathbf{x}_i . The reconstruction error $\|\mathbf{x}_j - \mathbf{x}_i'\|^2$ of each instance $\mathbf{x}_j \in \Omega_{\mathbf{x}_i}$ is weighted by a relational weight $s_{i,j}$, called the *reconstruction weight*. Thus, the reconstruction loss of GAE for each instance can be written as:

$$\mathcal{L}_{\text{GAE}}(\Omega_{\mathbf{x}_i}, \mathbf{x}_i') = \sum_{\mathbf{x}_j \in \Omega_{\mathbf{x}_i}} s_{i,j} \|\mathbf{x}_j - \mathbf{x}_i'\|^2, \quad (1)$$

where the reconstruction set $\Omega_{\mathbf{x}_i}$ and the reconstruction weight $s_{i,j}$ can be given by different schemes. Most of

the existing schemes attempt to replicate a dimensionality reduction method, such as principal component analysis (PCA) [15], Isomap [19], locally linear embedding (LLE) [17], or Laplacian eigenmaps (LE) [2].

2.2. 3D generalized autoencoder (3D-GAE)

Our 3D-GAE follows the architecture of a typical autoencoder, but adapted to 3D shape generation.

Input and output. The encoder of 3D-GAE takes volumetric shapes as input. Each input is represented as a set of $32 \times 32 \times 32$ voxels. A voxel can be defined as either occupied (1) or empty (0). The encoder outputs a latent representation $\mathbf{z} \in \mathbb{R}^n$ of the shape. The decoder then translates a latent vector back into a shape represented as a volume. Given that the last layer of the decoder uses a sigmoid activation function, each output voxel contains a real value in the range $[0, 1]$. Thus, we transform each voxel into a binary value according to a threshold of 0.5, that is, the voxel becomes 0 if the real value is below 0.5, or 1 otherwise.

Network architecture. We use a symmetric architecture where the encoder and decoder have the same number of layers. The encoder performs three levels of downsampling, while the decoder performs upsampling also at three levels. Specifically, the input is downsampled from $32 \times 32 \times 32$ to $4 \times 4 \times 4$ using three convolutional layers with a ReLU activation function followed by a batch normalization layer that helps the network to learn features independently from the output of previous layers. The filter sizes of the three convolutional layers are respectively 32, 16, and 8. We use a kernel of size $4 \times 4 \times 4$ and stride 2 for all the convolutional layers. The output layer of the encoder is a fully-connected layer which generates the latent vector \mathbf{z} . We use 128 as the size of the latent vector, which provides a balance between the reconstruction error and the complexity of the model. The decoder follows the inverse of this architecture for performing upsampling, with the exception that the output layer uses a sigmoid activation function.

Low-dimensional embedding. In our 3D-GAE, we use LE [2] as the parametric model to derive the reconstruction set and the reconstruction weights. The reconstruction set of an input shape \mathbf{x}_i is its k -nearest neighbors, *i.e.*, $\Omega_{\mathbf{x}_i} = \mathcal{N}_k(\mathbf{x}_i)$. The reconstruction weight is computed as:

$$s_{i,j} = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{t}\right), \quad (2)$$

where $\mathbf{x}_j \in \Omega_{\mathbf{x}_i}$ and t is a tuning parameter. In our work, we set $t = 200$, to give proportionally more weight to the closest neighbors.

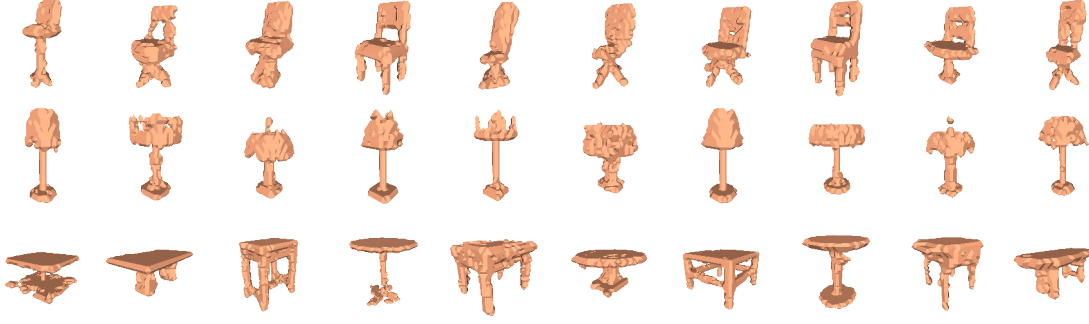


Figure 1. A collection of shapes synthesized by 3D-GAE via linear interpolation.

Similarity encoding. To derive the k -nearest neighbors of each input shape, we use the Chamfer distance [7] to compute the similarity of shapes. The Chamfer distance can be seen as a simplified form of the Hausdorff distance. Thus, we assume that the input shapes are consistently pre-aligned. Considering that our input is a set of volumetric shapes, each shape \mathbf{x}_i can be denoted as a volume $\mathbf{V}_i = (v_{p,q,r})_{32 \times 32 \times 32}$, with each entry $v_{p,q,r} \in \{0, 1\}$. We then convert the volume \mathbf{V}_i to a point cloud \mathcal{P}_i by collecting the indices of occupied voxels in \mathbf{V}_i , i.e., $\mathcal{P}_i = \{(p, q, r) \mid v_{p,q,r} = 1\}$. After that, we can compute the Chamfer distance between any two point clouds \mathcal{P}_i and \mathcal{P}_j :

$$d(\mathcal{P}_i, \mathcal{P}_j) = \sum_{\mathbf{a} \in \mathcal{P}_i} \min_{\mathbf{b} \in \mathcal{P}_j} \|\mathbf{a} - \mathbf{b}\|^2 + \sum_{\mathbf{b} \in \mathcal{P}_j} \min_{\mathbf{a} \in \mathcal{P}_i} \|\mathbf{a} - \mathbf{b}\|^2. \quad (3)$$

With the pairwise Chamfer distances computed for the entire input set, for each shape, we sort the remaining shapes in the set in ascending order according to their Chamfer distances. We then select the top k shapes as the nearest neighbors. In our work, we select $k = 10$.

Training and loss function. In practice, when training deep neural networks, the training is carried out for batches of data instead of individual instances, so that the model can converge to global optima, where a batch is a group of instances sampled from the training set. Thus, we also define the loss function of 3D-GAE over batches. We first compute the reconstruction error between each input batch \mathcal{B} and its reconstruction \mathcal{B}' , and then sum up the GAE losses, as given in Equation 1, per instance in the batch. The loss function of 3D-GAE is thus:

$$\mathcal{L}_{3D-GAE}(\mathcal{B}, \mathcal{B}') = \|\mathcal{B} - \mathcal{B}'\|^2 + \sum_{\mathbf{x} \in \mathcal{B}} \mathcal{L}_{GAE}(\Omega_{\mathbf{x}}, \mathbf{x}'). \quad (4)$$

3. Results and evaluation

We first explain the setup of our experiments, and then analyze our results with different forms of evaluation.

Datasets. We use three sets of objects taken from the COSEG [21] and ModelNet40 [18] datasets. Although each of these datasets contains several categories of objects, we select the *chair*, *lamp*, and *table* categories from these datasets since they have shapes with more pronounced structural variations compared to other categories. Specifically, our selected datasets are composed of 400 chairs from the COSEG dataset, and 100 lamps and 400 tables from the Modelnet40 dataset. We voxelize all the shapes into $32 \times 32 \times 32$ volumes. Each voxel contains either the value 0 or 1, representing an empty or occupied voxel. All the volumes are consistently pre-aligned.

Training procedure. For training the 3D-GAE, we split the datasets into training batches of 10 shapes each. We adjust the learning rate according to the size of the datasets. Specifically, we set the learning rate to 0.001 for both chairs and tables, and 0.002 for lamps. We train the network for 200 epochs with the Adam optimizer. More details about our implementation and training procedure of 3D-GAE can be found in our code¹.

Evaluation. To assess the quality of the latent manifolds learned by the 3D-GAE, we evaluate two aspects of the resulting model: (i) the interpolatory and extrapolatory capabilities of the model, and (ii) the structure of the learned latent space. We perform these two evaluations in a qualitative and quantitative manner by comparing our results to two other baseline approaches: a volumetric autoencoder (3D-AE) that uses the same network architecture as the 3D-GAE, discussed in Section 2.2, but trained with the standard mean squared error (MSE) loss, and a volumetric variational autoencoder (3D-VAE), trained with the network architecture and the weighted binary cross-entropy (BCE) loss as introduced by Brock *et al.* [3].

¹<https://github.com/IsaacGuan/3D-GAE>




































































































Dataset	Method	Linear interpolations										
Chair	3D-GAE											
	3D-AE											
	3D-VAE											
Lamp	3D-GAE											
	3D-AE											
	3D-VAE											
Table	3D-GAE											
	3D-AE											
	3D-VAE											

Table 1. Linear interpolations of selected shapes obtained with different network architectures.

Generated shapes. In our evaluation, we linearly interpolate reference shapes to synthesize new shapes. Specifically, we take the latent vectors \mathbf{z}_1 and \mathbf{z}_2 of two endpoint shapes and interpolate them with the formula $\mathbf{z}_u = (1 - u) \cdot \mathbf{z}_1 + u \cdot \mathbf{z}_2$, where u is a scalar interpolation parameter. We then feed \mathbf{z}_u to the decoder to obtain a 3D shape. A collection of shapes interpolated by the 3D-GAE is shown in Figure 1, where we randomly select the reference shapes and interpolation parameter, and handpick shapes with good visual quality. Table 1 presents examples of shapes interpolated with the 3D-GAE and the baseline approaches. The left and right endpoint shapes of each row are reconstructions of two selected training shapes, while the shapes in the middle of each row are obtained by linearly interpolating the latent vectors of the endpoint shapes by varying u in steps of 0.1.

By examining the endpoint shapes in Table 1, we notice how the reconstructions for the 3D-AE and 3D-GAE are much sharper and better connected when compared to the

Dataset	3D-GAE	3D-AE	3D-VAE
Chair	3.09 ± 0.36	2.98 ± 0.41	2.92 ± 0.38
Lamp	1.57 ± 0.24	1.33 ± 0.31	1.19 ± 0.25
Table	2.45 ± 0.35	2.19 ± 0.28	1.78 ± 0.32

Table 2. Inception scores of results synthesized with different methods, where higher values are better.

3D-VAE, without spurious blobs floating around the shapes. Next, by examining the interpolated shapes, we see that the shapes obtained with the 3D-GAE are in general also better connected (*e.g.*, the first row of chairs) and have less missing regions (*e.g.*, the first row of tables). In addition, the gradual interpolation looks slightly smoother for the 3D-GAE than the 3D-AE (*e.g.*, the lamp shades). Thus, the 3D-GAE provides reconstructions of quality similar to the 3D-AE, while at the same time providing better interpolation than the 3D-VAE.

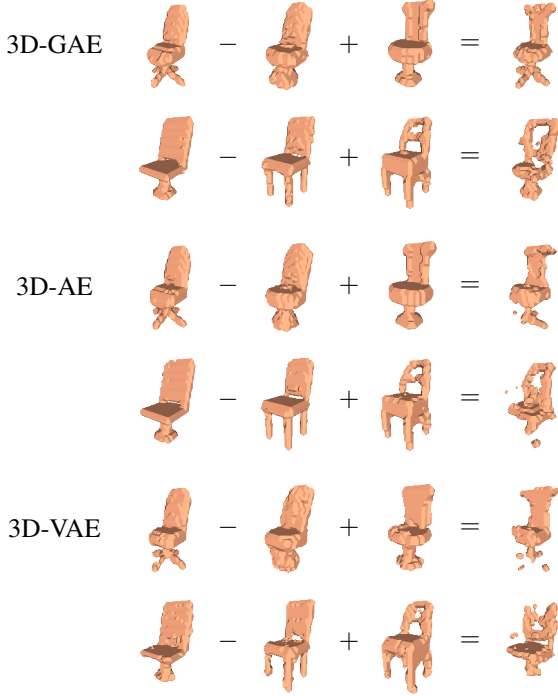


Figure 2. Examples of extrapolation via shape arithmetic.

Moreover, Figure 2 shows a few examples of extrapolation obtained with the 3D-GAE by applying arithmetic operations to the latent vectors. For example, in the first row, we take the input chair on the left and remove the components of the latent vector corresponding to the back of the chair. We accomplish this by subtracting the latent vector of the shape in the center-left which has the same type of back. Next, we add the latent vector of the shape in the center-right with a different type of back. In this manner, we exchange the type of back of the input shape, obtaining the shape on the right. Note that the resulting shape does not exist in the training set. The second row shows another example of exchanging the back of a chair. However, in this case, the resulting shape has a few visual artifacts since the new back does not match the input so well. In comparison, the same arithmetic operations carried out with the other two networks provide results with worse visual quality.

For a quantitative evaluation of the quality of the interpolated shapes, we automatically generate new sets of shapes by sampling shapes interpolated with the three methods, and compute their inception scores, which we report in Table 2. Specifically, we choose 400 interpolated shapes for chairs and tables, and 100 for lamps. We compute the inception scores with the method of Xie *et al.* [25] and the pre-trained model of Qi *et al.* [16]. This method evaluates the quality of the shapes according to the ModelNet40 dataset. We compare the results of shapes interpolated with the 3D-

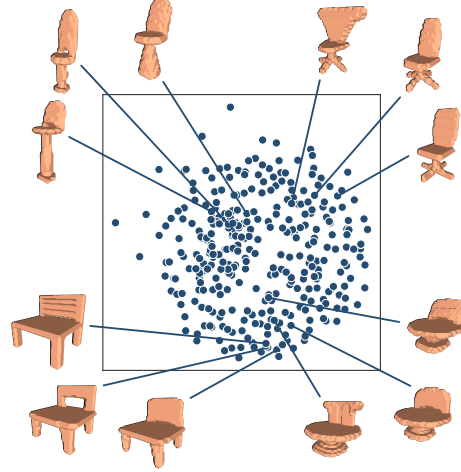


Figure 3. Multi-dimensional scaling diagram reflecting similarity among latent vectors of training shapes for the set of chairs.

Distance	Dataset	3D-GAE	3D-AE	3D-VAE
Chamfer	Chair	0.6425	0.5018	0.4239
	Lamp	0.8664	0.5281	0.6623
	Table	0.7732	0.6984	0.6970
Euclidean	Chair	0.7418	0.6408	0.5770
	Lamp	0.9294	0.7330	0.5485
	Table	0.8449	0.7277	0.8305

Table 3. Pearson correlation coefficients between shape distances and latent spaces learned by the different methods.

GAE to the baselines, where we generate sets with the same endpoint shapes for these approaches. We observe that the scores provided by the 3D-GAE are higher than the scores of 3D-AE and 3D-VAE, which indicates that the synthesized shapes generated by the 3D-GAE are classified better than the others, and likely of higher quality.

Latent spaces. We evaluate the quality of the latent spaces learned with a qualitative and quantitative evaluation. Figure 3 shows a multi-dimensional scaling (MDS) plot of the latent space of the set of chairs, based on the L_2 -norm computed between all pairs of latent vectors. We observe in this example how shapes with similar structure are grouped closely together in the same region of the plot, reflecting a meaningful grouping in the latent space.

To evaluate the latent spaces in a quantitative manner, we compute the correlation between all the pairwise shape distances and all the pairwise distances of the corresponding latent vectors. The shape distances are computed according to the Chamfer distance and Euclidean distance of the volumes. The correlation of pairwise distances is a simplified form of the residual variance measure proposed by Tenen-

Dataset	3D-GAE	3D-GAE*	3D-AE	3D-VAE
Chair	0.0058	0.0068	0.0034	0.0128
Lamp	0.0079	0.0095	0.0034	0.0174
Table	0.0098	0.0106	0.0065	0.0193

Table 4. Average reconstruction errors after training the models.

Dataset	3D-GAE	3D-AE	3D-VAE
Chair	504.75	221.25	406.21
Lamp	134.32	59.65	109.42
Table	526.97	221.69	407.55

Table 5. Training time of the three networks (in seconds).

baum *et al.* [19] for evaluating the quality of an embedding. Table 3 shows that our method provides the highest correlation coefficients among all the methods for both types of shape distance, where higher coefficients indicate higher correlation. On one hand, this result is expected, as the 3D-GAE optimizes the network according to the Chamfer distances, while the other networks do not directly optimize this objective. On the other hand, this result confirms that the 3D-GAE indeed allows one to better control the latent manifold learned by the network.

Reconstruction error and loss function. Table 4 presents the final average reconstruction errors of all training shapes for all the networks. We notice that, as expected from results reported by previous work, the 3D-AE has lower reconstruction errors, while the other networks provide better interpolation at the expense of higher reconstruction errors. The 3D-GAE sits in-between the 3D-AE and 3D-VAE. In addition, 3D-GAE* denotes the 3D-GAE trained only with the original GAE loss rather than the combined loss. We notice how the reconstruction errors for this network are slightly higher than the network with the combined loss. The main reason is that, with the combined loss, batches are taken into consideration during training, making the model more efficiently converge to a global optimum. Although the difference between the reconstruction errors of 3D-GAE and 3D-GAE* is small, we conjecture that the combined loss may have more impact on larger datasets.

Execution time. The deep networks are trained with an NVIDIA GeForce GTX 1070 Ti GPU with 8 GB of memory and CUDA version 10.0. Table 5 presents the training time for each type of network. We see that the 3D-GAE takes around 2.4 and 1.3 times longer to train than the 3D-AE and 3D-VAE, respectively, given that its loss function considers a neighborhood of data points for each training sample.

4. Conclusion, limitations, and future work

In this paper, we introduced the 3D-GAE, a generative model of 3D shapes based on the GAE [20]. We demonstrated that the 3D-GAE is able to learn a meaningful latent manifold by exploring shape relations derived from the Chamfer distance. We showed that the latent manifold leads to better-quality interpolation and extrapolation of shapes.

Despite its advantages over AEs and VAEs, the 3D-GAE also has certain limitations. First, the training time of 3D-GAE is relatively longer than that of previous methods, since its loss function considers the neighborhood of each training sample. For similar reasons, the 3D-GAE requires the construction of a larger tensor graph for training. Thus, it requires more memory than other types of networks, and with the current hardware capabilities, it would be difficult to adapt the 3D-GAE to volumes with higher resolution, such as volumes with $64 \times 64 \times 64$ voxels.

Given these limitations, one direction for future work is to base the generative model on a more suitable representation for shapes, such as an implicit representation [5, 14], which leads to a sparser input. A more natural shape representation would likely also lead to shapes with better visual quality. In addition, it would be interesting to explore other data relationships beyond the Chamfer distance for learning application-specific latent manifolds. For example, semantic relations or part structures could be considered in the loss function used for training.

Acknowledgments

We thank the reviewers for their valuable comments. This work was supported by NSERC (2015-05407).

References

- [1] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. Learning representations and generative models for 3D point clouds. In *Proc. Int. Conf. on Machine Learning*, pages 40–49, 2018. 1
- [2] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in Neural Information Processing Systems*, pages 585–591, 2002. 2
- [3] Andrew Brock, Theodore Lim, James M. Ritchie, and Nick Weston. Generative and discriminative voxel modeling with convolutional neural networks. *CoRR*, abs/1608.04236, 2016. 1, 3
- [4] Siddhartha Chaudhuri, Daniel Ritchie, Jiajun Wu, Kai Xu, and Hao Zhang. Learning generative models of 3D structures. In *Eurographics State of the Art Reports*, 2020. 1
- [5] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 5939–5948, 2019. 1, 6

- [6] Angela Dai, Charles R. Qi, and Matthias Nießner. Shape completion using 3D-encoder-predictor CNNs and shape synthesis. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 6545–6554, 2017. 1
- [7] Haoqiang Fan, Hao Su, and Leonidas Guibas. A point set generation network for 3D object reconstruction from a single image. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 2463–2471, 2017. 2, 3
- [8] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. Adaptive computation and machine learning. MIT Press, 2016. 1
- [9] Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan C. Russell, and Mathieu Aubry. A papier-mâché approach to learning 3D surface generation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 216–224, 2018. 1
- [10] Ali Jahanian, Lucy Chai, and Phillip Isola. On the “steerability” of generative adversarial networks. *CoRR*, abs/1907.07171, 2019. 1
- [11] Jun Li, Kai Xu, Siddhartha Chaudhuri, Ersin Yumer, Hao Zhang, and Leonidas Guibas. GRASS: Generative recursive autoencoders for shape structures. *ACM Trans. on Graphics*, 36(4):52:1–52:14, 2017. 1
- [12] Jerry Liu, Fisher Yu, and Thomas A. Funkhouser. Interactive 3D modeling with a generative adversarial network. In *Proc. Int. Conf. on 3D Vision*, pages 126–134, 2017. 1
- [13] Charlie Nash and Christopher K. I. Williams. The shape variational autoencoder: A deep generative model of part-segmented 3D objects. *Computer Graphics Forum*, 36(5):1–12, 2017. 1
- [14] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. DeepSDF: Learning continuous signed distance functions for shape representation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 165–174, 2019. 1, 6
- [15] Karl Pearson. On lines and planes of closest fit to points in space. *Philosophical Magazine*, 2(11):559–572, 1901. 2
- [16] Charles R. Qi, Hao Su, Matthias Nießner, Angela Dai, Mengyuan Yan, and Leonidas J. Guibas. Volumetric and multi-view CNNs for object classification on 3D data. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 5648–5656, 2016. 5
- [17] Sam T. Roweis and Lawrence K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000. 2
- [18] Nima Sedaghat and Thomas Brox. Unsupervised generation of a viewpoint annotated car dataset from videos. In *Proc. IEEE Int. Conf. on Computer Vision*, pages 1314–1322, 2015. 3
- [19] Joshua B. Tenenbaum, Vin de Silva, and John C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000. 2, 6
- [20] Wei Wang, Yan Huang, Yizhou Wang, and Liang Wang. Generalized autoencoder: A neural network framework for dimensionality reduction. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition Workshops*, pages 496–503, 2014. 1, 2, 6
- [21] Yunhai Wang, Shmulik Asafi, Oliver van Kaick, Hao Zhang, Daniel Cohen-Or, and Baoquan Chen. Active co-analysis of a set of shapes. *ACM Trans. on Graphics*, 31(6):165:1–165:10, 2012. 3
- [22] 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 *Advances in Neural Information Processing Systems*, pages 82–90, 2016. 1
- [23] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3D ShapeNets: A deep representation for volumetric shapes. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 1912–1920, 2015. 1
- [24] Zhijie Wu, Xiang Wang, Di Lin, Dani Lischinski, Daniel Cohen-Or, and Hui Huang. SAGNet: Structure-aware generative network for 3D-shape modeling. *ACM Trans. on Graphics*, 38(4):91:1–91:14, 2019. 1
- [25] Jianwen Xie, Zilong Zheng, Ruiqi Gao, Wenguan Wang, Song-Chun Zhu, and Ying Nian Wu. Learning descriptor networks for 3D shape synthesis and analysis. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 8629–8638, 2018. 5