Deep Adversarial 3D Shape Net

Chieh Lin
Machine Learning
chiehl1@cs.cmu.edu

Derek Liu
Mechanical Engineering
hsuehtil@andrew.cmu.edu

Alex Kelly
Information Systems
ajk1@andrew.cmu.edu

1 Introduction

3D shapes are a crucial but heavily underutilized resource in today's computer vision research. The advancement of scanning technology, modeling and big data have led to a huge growth of 3D shape repositories. Deep learning is an emerging approach in this field and proper 3D shape representations are still being developed.

In just the past two years, people have been exploring deep learning models for generating 3D shapes. Two crucial questions are (1) What is the suitable 3D shape representation (2) How to make generated shapes look realistic. In other words, we want all components of the shape to be connected. In this project, we are addressing the three issues in 3D shape net development.

In terms of applications, we delve into reconstructing 3D shapes using 2.5D (2D + depth) images. With depth sensors such as LiDAR and RGB-D cameras now prevalent in robotics, it will be very useful to be able to infer the 3D shape from an image that a robot might see.

1.1 Related Works of Deep 3D shape nets

3D object understanding and reconstruction are important research questions in computer graphics and computer vision. In graphics, researchers tried to solve this question by breaking the shape down into separate shape parts (Huang et al. (2015)). This approach has achieved plausible results in shape synthesis. However, this requires pre-processing steps, including shape segmentation and resampling, and it does not work well for shapes with different part compositions.

The development of deep learning models and 3D shape repositories, such as Princeton Model Net (Shilane et al. (2004)), has lead this research field toward using deep learning approaches. Last year, the first 3D shape net using Deep Belief Net (DBN) (Mohamed et al. (2012)) had been proposed by Z. Wu (Wu et al. (2015)). This is the first attempt treating the whole input shape, represented as 3D voxel, as the input without doing pre-processing. This approach shows that the DBN is able to learn shape features, such as planes, from the shape data. Also, the learned representation has been used to reconstruct and classify 3D shapes. Although the results still have space to be improved, it has shown that using deep learning to learn 3D shape representation is a plausible direction.

Concurrently, D. Maturana proposed a 3D Convolutional Neural Networks for 3D shape classification (Maturana and Scherer (2015)). It also treats 3D shapes as voxel and trains a CNN for classification. It achieves better classification accuracy, 83% for ModelNet40 dataset, then the 3D shape net (Wu et al. (2015)). Su, Hang also proposed a model for 3D shape classification using images taken from different angles (Su et al. (2015)). It is a clever idea to utilize large amount of image data to help 3D shape classification. This approach achieves 90.1% classification accuracy on ModelNet40.

This year, deep learning models for 3D shapes have been actively developed. Brock Andrew uses Variational Autoencoder (VAE) (Kingma and Welling (2013)) to replace the DBN in 3D shape net (Goodfellow et al. (2014)). This work tries to use deeper network structure to learn better shape representations. It achieves over 95% classification accuracy and is able to generate more plausible shapes.

10807 Class Project (2016).

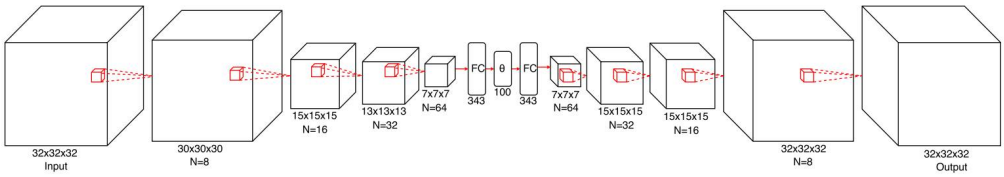

Figure 1: 3D VAE

Besides using VAE, Jiajun Wu proposed another deep learning model for 3D shapes (Wu et al. (2016)) using a popular training approach, Generative Adversarial Networks (GAN) (Goodfellow et al. (2014)). Generative Adversarial Networks (GAN) are a contemporary method that sets up a generative model and a discriminative model in a competition. The discriminator attempts to classify shapes as being real or synthetic, while the generator tries to produce realistic shapes to confuse it. The intuition behind is that adversarial training is able to learn a more structured latent representation for images. If 3D shapes share similar characteristics as images, GAN should be able to learn a better representation and reconstruct more plausible 3D shapes. As a result, 3D-GAN performs worse in classification but the generated shapes are visually more plausible.

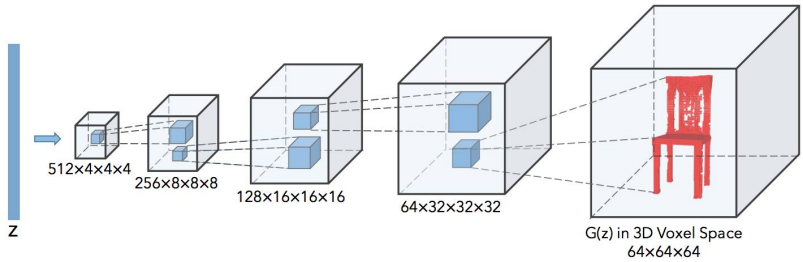

Figure 2: 3D GAN

The field of using deep learning for 3D shape understanding is only starting to emerge since the past two years. Voxel has been proved to be a good shape representation for deep learning models to learn shape representation. However, in graphics perspective, voxel is not a suitable representation because it can only capture large scale shape features. Therefore, a future direction is to find a different encoding method, such as point cloud, to train those models. Another problem shared for all existing models is that they don't constrain generated shapes to be "valid". For example, a valid shape should not have disconnected parts flying in the air. Further research in the ways of including those constraints into training process is also crucial for making generation more plausible.

2 Methods

2.1 Dataset and Evaluation Metrics, Data processing

We extract shapes from the ModelNet10 and ModelNet40 benchmark dataset provided by (Wu et al. (2015)). This dataset is composed of synthetic CAD models with 10 classes or 40 classes, such as desks and chairs. Each object is converted to voxel representation and has a maximum number of 12 rotations. Figure 3 shows the voxel representation of the chair CAD data.

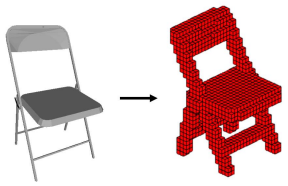

Figure 3: Voxel representation

2.2 3D Convolution Neural Network (3D CNN)

We use Keras tool (Chollet (2015)) to build a 3D convolutional neural network to perform classification and retrieval tasks in order to evaluate on ModelNet10 benchmark datasets. Our model is constructed with 2 layers of 3D convolution layers (with 3*3*3 kernel, (1,1,1) strides, 32 filters), 1 max pooling layer (with 2*2*2 pool and 0.25 dropout), 1 dense layer of size 128 (with 0.5 dropout) and an output layer with size equal to the number of classes for classification. Each layer uses rectified linear unit as the activation function except that the output layer uses softmax activation function. The loss function is cross-entropy optimized with Adam optimizer (Kingma and Ba (2014), we use default parameters provided by original paper). We train the ModelNet10 training set with batch size 32 for 20 epochs (enough for convergence) and select the best epoch as our performance.

To perform the retrieval task and calculate mean average precision (mAP) , we first extract the intermediate output of the size 128 dense layer and compute euclidean distance matrix of testing set. After that, for each testing sample, we generate a ranked list with nearest neighbors and compute the mAP based on the true labels and ranked list.

2.3 Variational Autoencoder (VAE)

The variational autoencoder (Kingma and Welling (2013)) is a generative model which pairs a bottom-up encoder network with a top-down decoder network. VAEs have been successful at separating style and content (Kingma and Ba (2014)) and at learning to "draw" images in a realistic manner (Gregor et al. (2015)). Although it is named "autoencoder", the mathematical basis of VAE has little to do with classical autoencoders and its variants, such as sparse autoencoder (Lee et al. (2006)) and denoising autoencoder (Vincent et al. (2008)).

VAE is trained to maximize the variational lower bound on the data's log-likelihood:

$$\log P(X) = \log E_{Q(z|X)}[\frac{P(X,z)}{Q(z|X)}] \geq E_{Q(z|X)}[\log \frac{P(X,z)}{Q(z|X)}] = \mathcal{L}(x)$$

After some derivation, the loss function $\mathcal{L}(x)$ can be optimized using (Doersch (2016)):

$$\mathcal{L}(x) = \log P(X) - \mathcal{D}[Q(z|X)||P(z|X)] = E_{z\ Q}[\log P(X|z] - \mathcal{D}[Q(z|X)||P(z)]$$

in which X is input data, z is latent variables, and $\mathcal{Q}$ is any distribution. The left hand side is the loss function we want to optimize. The right hand side is what we can optimize using gradient descent given $\mathcal{Q}$. After the "reparameterization trick" proposed in (Kingma and Welling (2013)), we then can compute the gradient using:

$$E_{X\ D}[E_{\epsilon\ \mathcal{N}(0,I)}[\log P(X|z = \mu(X) + \Sigma^{1/2}(X) * \epsilon)] - \mathcal{D}[Q(z|X)||P(z)]]$$

2.4 Adversarial Autoencoder

Adversarial Autoencoder (AAE) (Makhzani et al. (2015)) is a probabilistic autoencoder that uses the generative adversarial networks (GAN) (Goodfellow et al. (2014)) to perform variational inference. It is a typical autoencoder regularized by matching the aggregated posterior, $q(z)$, to an arbitrary prior, $p(z)$. In order to do so, a discriminator in standard GAN is attached on top of "neck" of the autoencoder. Because the encoder of the autoencoder acts the same as the generator in GAN, the encoder is regularized through fooling the discriminator. Meanwhile, it attempts to minimize reconstruction cost using the error propagated from the decoder.

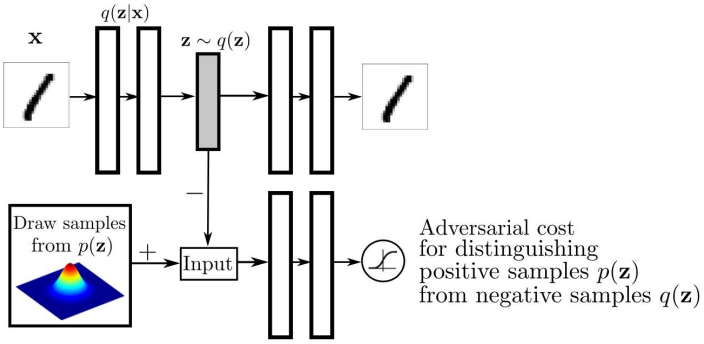

Figure 4: AAE model architecture

Training AAE includes three steps:

- A minibatch of input vectors is encoded and decoded by autoencoder. Then update both encoder and decoder using the standard reconstruction loss.
- Encode input vectors using the encoder and generate sampled vectors using the hidden code in autoencoder separately. Then use the two vectors as the input of discriminator. The discriminator is trained to differentiate the two vectors using binary cross-entropy cost.
- After the discriminator making prediction, update the encoder again using binary cross-entropy loss based on its ability to fool the discriminator

Usually, the discriminator learns faster than the encoder. To successfully train the AAE, we need to either reduce the learning rate for discriminator or use an adaptive learning scheme to slow down the discriminator.

Comparing to Variational Autoencoder (VAE) (Kingma and Welling (2013)), VAE needs to access the functional form of the prior distribution in order to perform back-propagation. But AAE only needs to be able to sample from $q(z)$ to match $p(z)$, which results in a more plausible and flexible learned latent space.

Our proposed 3D AAE is based on this 2D AAE. However, we need to modify the error, convolution, pooling and so on in 3D space. The input of 3D AAE would be binary voxel representation of shapes, the output would be the same form as the input.

2.5 3D-GAN

We implement our 3D-GAN model based on the torch implemention of DCGAN (Deep Convolutional Generative Adversarial Networks) (Radford et al., 2015) on 2D images. There is also a very recent paper (2016 NIPS) that implements the 3D version of DCGAN called 3D-GAN (Zhang et al., 2016). Originally we ask the authors for 3D-GAN source code but they didn't send their code in time. Therefore, we implement it ourselves and set our parameters similar to the 3D-GAN model but reduce the dimensions of voxels to speed up the training process. The model structure is basically the same as Figure 2 except that we only have 4 deconvolution layers in the generator model with the number of filters to be (256, 128, 64, 1), the kernel size 4*4*4, suitable padding, ReLU activation layers and batch normalization layers in between. The sturcuture of discriminator basically mirrors the structure of generator but use LeakyReLU with parameter 0.2 instead of ReLU. The output layer of generator uses sigmoid activation to represent the probability of a voxel while the output layer of discriminator represents the probability to be real 3D voxel model. The detailed parameters of the generator and discriminator are in Table 1. We use binary cross entropy as the loss function and set batch size to 100. We adopt ADAM optimizer with learning rate of generator to be 0.00025 and learning rate of discriminator to be 0.00001, beta parameter 0.5.

Table 1: Detailed parameters of our 3D-GAN model

layer	n_filter	kernel size	stride	pad	activation
generator_deconv_layer1	256	4*4*4	2,2,2	X	ReLU
generator_deconv_layer2	128	4*4*4	2,2,2	1,1,1	ReLU
generator_deconv_layer3	64	4*4*4	2,2,2	1,1,1	ReLU
generator_deconv_layer4	1	4*4*4	2,2,2	2,2,2	Sigmoid
discriminator_conv_layer1	64	4*4*4	2,2,2	2,2,2	LeakyReLU_0.2
discriminator_conv_layer2	128	4*4*4	2,2,2	1,1,1	LeakyReLU_0.2
discriminator_conv_layer3	256	4*4*4	2,2,2	1,1,1	LeakyReLU_0.2
discriminator_conv_layer4	1	4*4*4	2,2,2	X	Sigmoid

2.6 2.5D Reconstruction with DBM

With depth sensors such as LiDAR and RGB-D cameras now prevalent in robotics, it will be useful to be able to reconstruct 3D shapes from a 2.5D image. This task is analogous to image completion in 2D, where there is some missing information that we would like to infer. However, current geometric and structural methods do not reflect modern trends in deep learning. (Wu et al. (2015)) is the first to propose a generative approach, using a Convolutional Deep Belief Network (CDBN) to represent the probability distribution of 3D voxel shapes.

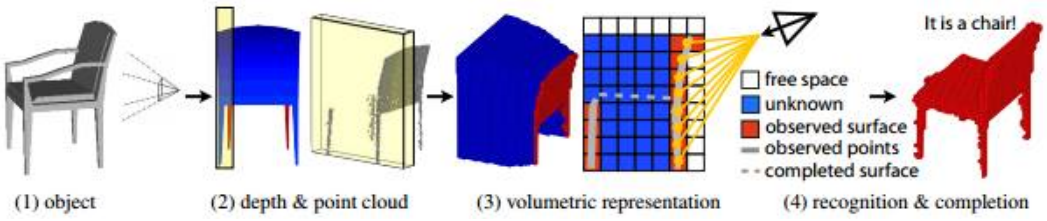

Figure 5: View-based 2.5D Shape Completion

Deep Boltzmann Machines (DBM) have proven effective for 2D image completion, most notably ShapeBM's powerful partial image completion (Eslami et al. (2012)). We extended this model to be able to reconstruct 3D shapes with only one visible surface as input. By conditioning on the observed free space, DBM can generate a likely configuration of the unobserved space.

3 Results

3.1 3D convolution neural network (3D CNN)

The testing set performance (Table 2) of our 3D CNN classification model is 90.2% for accuracy (ACC) and 80.6% for mAP when no rotation is applied to dataset. The performance gets worse when rotation is applied to training or testing samples. The performance of other algorithms are also listed in Table 2.

Algorithm	40 class ACC	40 class mAP	10 class ACC	10 class mAP
3D-GAN(Wu et al. (2016))	83.3	X	91	X
VRN Ensemble(Brock et al. (2016))	**95.54**	X	**97.14**	X
ORION	X	X	93.8	X
FusionNet	90.8	X	93.11	X
Pairwise	90.7	X	92.8	X
MVCNN	90.1	X	X	X
GIFT	83.1	**81.94**	92.35	**91.12**
VoxNet	83	X	92	X
DeepPano	77.63	76.81	85.45	84.18
3DShapeNets(Wu et al. (2015))	77	49.2	83.5	68.3
Our 3D CNN (no rotate)	X	X	90.2	80.6
Our 3D CNN (train rotate)	X	X	87.3	73.1
Our 3D CNN (train&test rotate)	X	X	86.0	67.2

Table 2: ModelNet Benchmark Leaderboard

3.2 Variational Autoencoder (VAE)

In this project, we have built a VAE for 3D shapes which uses voxel to represent shapes. We did our first experiement using two categories in our shape data, dressers and desks, and trained 5 epochs to verify the model.

In Figure 6, we show the 2D representation of our data, dressers and desks, learned by the VAE and a reconstruction result random sampled from the data manifold (a desk).

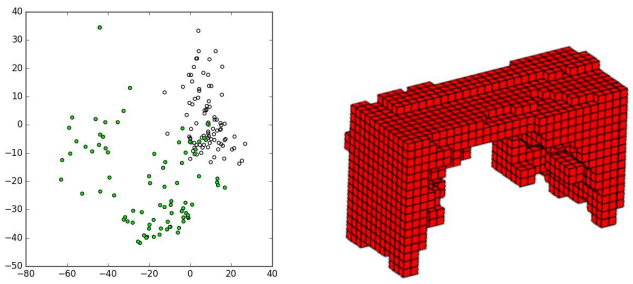

Figure 6: Latent space and generated table from VAE

3.3 3D-AAE

We would like to modify the AE code from this website (`https://blog.keras.io/building-autoencoders-in-keras.html`) into 3D-AAE. However, we found that it is very hard to extend our CNN and the AE code into 3D-AAE with keras because AAE is too complicated for keras to implement. Later, we found an AAE tutorial (`http://hjweide.github.io/adversarial-autoencoders`) with code that implements AAE for mnist dataset. However, the code use lasagne tool to implement but lasagne doesn't support 3D convolutional layers. Another tutorial (`https://swarbrickjones.wordpress.com/2016/01/24/generative-adversarial-autoencoders-in-theano/`) uses theano to implement AAE on a 2D face dataset. We originally would like to modify it to 3D version but later we found that the implementation is actually different from original AAE paper. After various failures of implementing 3D-AAE model, we eventually ran out of time to finish the implementation of 3D-AAE.

3.4 3D-GAN

Our 3D-GAN model takes about 2 hours to train 25 epochs on rotated ModelNet40 dataset with GTX 980 Ti GPU. This model trains really fast because the original torch implementation is well-optimized.

The randomly generated figures of our 3D-GAN model after epoch 25 are shown in Figure 7. We can observe that the generated 3D models exhibit the structures of a vase, sink and TV stand.

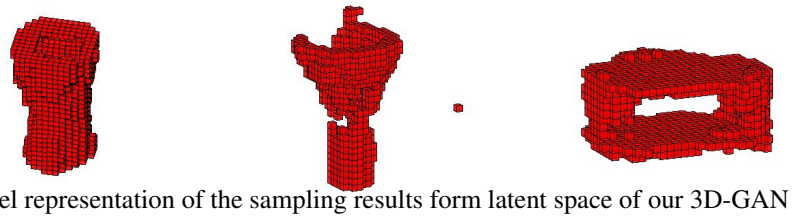

Figure 7: Voxel representation of the sampling results form latent space of our 3D-GAN model.
Left to right seems to be: vase, sink, TV stand

3.5 2.5D Reconstruction with DBM

We created a 3D-DBM with 2 hidden layers of 1000 units each and trained it on a two classes of the data (computer monitors and chairs). We ran a training procedure combining variational inference and stochastic approximation for several epochs. Because there are no standard libraries for Deep Boltzmann Machines yet, we wrote our machine from scratch.

Object generation in DBMs are typically done by performing several iterations of persistent contrastive divergence on random-noise input. Our 3D reconstruction was done similarly, except that we held the observed surface in the 2.5D depth map constant during generation. Hence a configuration is inferred in the unknown space, conditioned on the observed space.

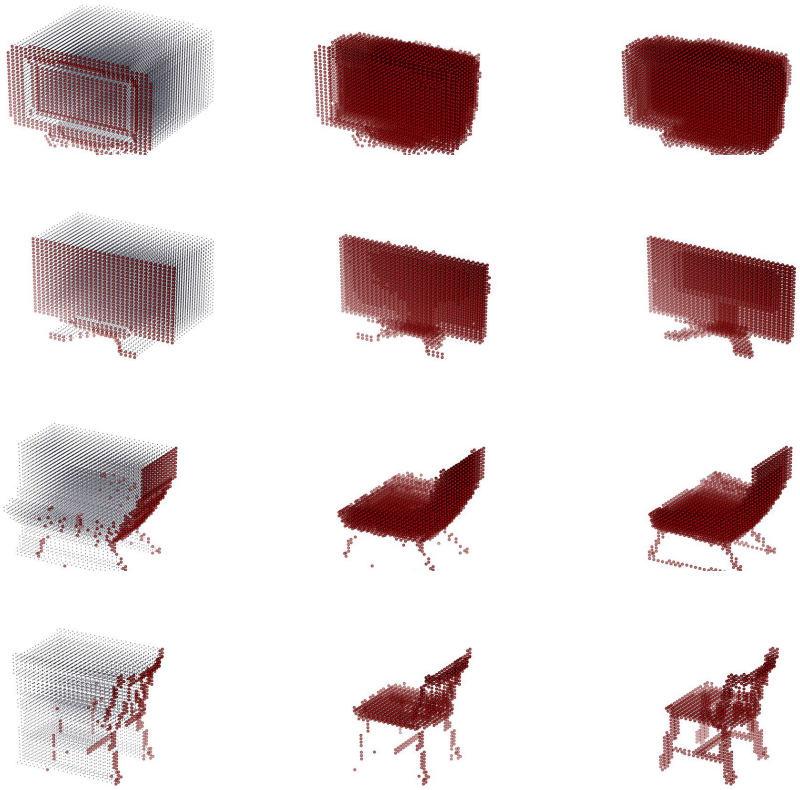

Figure 8: Sample results from 3D-DBM shape reconstruction.
Left to right: 2.5D view (gray area unknown), reconstructed 3D object, original 3D object

To reconstruct 3D shapes, we created a 2.5D depth map for each object and tasked our 3D-DBM with inferring the unknown obstructed space. This is done with 100 Note that these are images that have

not been observed by the machine before. We achieved an average voxel precision and recall of 0.673 and 0.806 on the monitor test data and 0.697 and 0.550 on the chair test data.

4 Discussion

In this project we have reviewed various deep learning models applied to 3D shape data. It has been shown that voxel representation is suitable for training deep learning models. The learned shape representation is able to perform classification and reconstruction given 2D or 2.5D images. However, how to ensure generated shapes are valid is still an unsolved problem in this field.

We implemented 3D-CNN for classification, 3D-VAE and 3D-GAN for shape generation, and 3D-DBM for reconstructing 3D shapes from 2.5D data. Our ultimate goal is to combine these approaches to build a 3D-AAE and modify the loss function to add requlerized term in order to generate more smooth and plausible shapes. Unfortunately, we could not obtain satisfactory results by the deadline. We failed to capture surface detail of the shapes, but we still achieve promising results in all three tasks we mentioned above.

References

Brock, A., Lim, T., Ritchie, J., and Weston, N. (2016). Generative and discriminative voxel modeling with convolutional neural networks. *arXiv preprint arXiv:1608.04236*.

Chollet, F. (2015). keras. `https://github.com/fchollet/keras`.

Doersch, C. (2016). Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*.

Eslami, S. M. A., Heess, N., and Winn, J. (2012). The shape boltzmann machine: a strong model of object shape. In *IEEE Conference on Computer Vision and Pattern Recognition 2012*.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680.

Gregor, K., Danihelka, I., Graves, A., Rezende, D. J., and Wierstra, D. (2015). Draw: A recurrent neural network for image generation. *arXiv preprint arXiv:1502.04623*.

Huang, H., Kalogerakis, E., and Marlin, B. (2015). Analysis and synthesis of 3d shape families via deep-learned generative models of surfaces. In *Computer Graphics Forum*, volume 34, pages 25–38. Wiley Online Library.

Kingma, D. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Lee, H., Battle, A., Raina, R., and Ng, A. Y. (2006). Efficient sparse coding algorithms. In *Advances in neural information processing systems*, pages 801–808.

Makhzani, A., Shlens, J., Jaitly, N., and Goodfellow, I. (2015). Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*.

Maturana, D. and Scherer, S. (2015). Voxnet: A 3d convolutional neural network for real-time object recognition. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 922–928. IEEE.

Mohamed, A.-r., Dahl, G. E., and Hinton, G. (2012). Acoustic modeling using deep belief networks. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(1):14–22.

Radford, A., Metz, L., and Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*.

Shilane, P., Min, P., Kazhdan, M., and Funkhouser, T. (2004). The princeton shape benchmark. In *Shape modeling applications, 2004. Proceedings*, pages 167–178. IEEE.

Su, H., Maji, S., Kalogerakis, E., and Learned-Miller, E. (2015). Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 945–953.

Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.-A. (2008). Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. ACM.

Wu, J., Zhang, C., Xue, T., Freeman, W. T., and Tenenbaum, J. B. (2016). Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. *arXiv preprint arXiv:1610.07584*.

Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., and Xiao, J. (2015). 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1912–1920.

Zhang, C., Wu, J., Xue, T., Freeman, B., and Tenenbaum, J. (2016). Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *Advances In Neural Information Processing Systems*, pages 82–90.