Adversarial Object Generation

YiDing Jiang
Department of EECS
University of California, Berkeley
Berkeley, CA 94709
yjiang9603@berkeley.edu

Pusong Li
Department of EECS
University of California, Berkeley
Berkeley, CA 94709
alanpusongli@berkeley.edu

Jeffrey Mahler
Department of EECS
University of California, Berkeley
Berkeley, CA 94709
jmahler@berkeley.edu

Ken Goldberg
Department of EECS
University of California, Berkeley
Berkeley, CA 94709
goldberg@berkeley.edu

Abstract

We propose an augmentation to the generative adversarial network framework to shift the generated distribution towards some desired property, which we represent using some not necessarily differentiable blackbox quality function. This quality function is multiplied to the generator objective to weight higher quality samples more heavily, motivated by policy gradient. We evaluate this strategy on the tasks of minimizing mean grasp quality (as determined by the Dex-Net system for computing robust grasp quality, after remeshing using marching cubes and smoothing with three iterations of Laplacian smoothing) of generated objects and minimizing the number of connected components in the remeshed generated objects. We present results, analyze the limitation of the strategy and suggest possible future directions.

1 Introduction

Adversarial data are important in the benchmarking and understanding the shortcomings of many systems; however, finding challenging inputs is often difficult because of the complexity of the system and coupling effects between everything involved. The adversarial set of Dex-Net 2.0 [1], for instance is hand-picked and the features considered adversarial geometry are chosen by intuition and not statistically proven to be adversarial.

Generative adversarial networks [2] have shown promise in data augmentation tasks, and such the technique has shown enough capacities that have been extended to generate data for challenging real world tasks [3, 4].

We propose a general method for using GANs to generate realistic distributions that minimize some arbitrary (not necessarily differentiable) loss, and apply this method to generating objects that minimize mean grasp quality as given by the Dex-Net 1.0 [5] system acting as a black-box grasp quality environment. We also apply this method to minimizing the number of connected components per sample in the generated distribution. We find that this method is able to shift the distribution to improve the quality function in both cases.

2 Related Works

Given an object, grasp planning techniques seek to find a gripper configuration that maximizes a quality metric. We focus on the analytic robust grasp planning approach detailed in Dex-Net 1.0 [5] using the Ferrari-Canny metric [6].

Discriminative and generative tasks with 3D data have also attracted attentions in recent years. [7, 8]. The work by JiaJun et al. [9] really showcases GAN's capability to generate objects in 3D spaces even though each generator in the work in limited to a single class. In this work, the architecture we use is very similar to the architecture in 3DGAN but we will use it to generate objects that are geometrically diverse and not pose-aligned.

There have also been works on incorporating particular structure into generative models that are trained on unconditioned data, or the structure is not included. Engel et al. [10] proposes to use an actor critic on the noise space of an variational autoencoder to generate data with conditioned property defined by some critic function from unlabeled data. Yu et al. [11] unrolls the sequence generation process and uses techniques akin to variance reduction to generate sequential data using SeqGAN. This type of tasks has very large practical value because collecting data with exact labels beforehand is in many cases impractical and labeling them afterwards is expensive. Although SeqGAN uses techniques from policy gradient, it does not have intermediate reward. To our knowledge, this paper is the first work where an external quality function is introduced in the setting of GAN.

3 Policy Gradient Generative Adversarial Nets

Generative adversarial networks, or GANs have shown promise in generating sharp and high quality samples based on the training data. To learn the generator distribution p_g, we transform the prior $z \sim p_z(z)$ to data through a mapping $G(z; \theta_g)$, where θ_g is the parameter of the generator, which in this work is the generator of a 3D-DCGAN. The discriminator $D(x; \theta_d)$ is also the discriminator of a 3D-DCGAN that outputs the probability that x comes from p_{data} rather than p_g. We train D to maximize the probability of correctly classifying the origin of x and simultaneously train G to minimize $log(1 - D(G(z)))$, which is the log probability of the discriminator classifying a generated example as coming from p_{data}. The non-saturating objective and gradient for the generator is formulated as:

$$J^{(G)} = \mathbb{E}_z[log(D(G(z)))] \tag{1}$$

$$\nabla_{\theta_g} J^{(G)} = \mathbb{E}_z[\nabla_{\theta_g} log(D(G(z)))] \tag{2}$$

However, without additional labels, the generator in a conventional GAN settings tries to generate samples that match the data distribution exactly. This may fall short in many real world applications, where we are interested in using generated examples with particular properties. Formally, let $q(x)$ be a quality function mapping x to a real number that defines how good example x is under some metrics. We are interested in generating examples that maximize $q(x)$ in addition to the likelihood that it comes from from p_{data}. One could run $q(x)$ on many examples from the real data and keep the ones with high quality but this approach is very inefficient and not scalable when $q(x)$ is a complex function and samples from data distribution are limited.

When $q(x)$ is a blackbox or a non-differentiable function, we cannot account for this function in conventional GAN training. This setting shares similarities with the use of policy gradient [12] in reinforcement learning where the agent has no knowledge of the environment other than the reward $r(x)$ it gets from the environment. Suppose $\pi_g(x)$ is some generator policy, the policy gradient formulation of the problem can be expressed as the following using the log trick:

$$J(\theta_g) = \mathbb{E}_{x \sim \pi_g(x)}[q(x)] \tag{3}$$

$$\nabla_{\theta_g} J(\theta_g) = \mathbb{E}_{x \sim \pi_g(x)}[q(x) \nabla_{\theta_g} log(\pi_g(x))] \tag{4}$$

We propose to combine the GAN with policy gradient so the generator can account for $q(x)$, which is the equivalent of $r(x)$, into the training. In a traditional policy gradient setting, the neural network outputs a policy distribution directly, but this is not the case for GAN, which samples from its underlying distribution implicitly. In fact, measuring the likelihood of samples of a generative model

is a hard problem. Models that are capable of directly modelling likelihood, such as the restricted Boltzman Machine [13] or deep belief networks [14], are usually not scalable or are hard to train.

To combine GAN and policy gradient, we use $D(x)$ as the proxy for $\pi_g(x)$:

$$\pi_g(x) \cong D(x) \tag{5}$$

This estimate can be justified under the assumption that the discriminator is optimal. From the original GAN paper, we know that for a fixed G, the optimal discriminator outputs:

$$D(x) = \frac{p_{data}(x)}{p_{data}(x) + p_g(x)} \tag{6}$$

The expression above can be interpreted as the probability that sample x comes from the distribution of real data rather than the distribution of generator. Therefore, when the discriminator is optimal for a fixed generator, its output can be interpreted as the likelihood of the sample under the real data distribution and generator distribution.

We then rewrite the generator objective as:

$$J^{(G)} = \mathbb{E}_z[log(D(G(z)))] \cong \mathbb{E}_{x\sim\pi_g(x)}[log(\pi_g(x))] \tag{7}$$

$$\nabla_{\theta_g} J^{(G)} = \mathbb{E}_z[\nabla_{\theta_g} log(D(G(z)))] \cong \mathbb{E}_{x\sim\pi_g(x)}[\nabla_{\theta_g} log(\pi_g(x))] \tag{8}$$

Notice that $eq(4)$ and $eq(8)$ are off by $q(x)$ and we can introduce it into the generator objective and arrive at policy gradient augmented GAN:

$$J^{(G)} = \mathbb{E}_z[q(G(z))log(D(G(z))] \tag{9}$$

$$\nabla_{\theta_g} J^{(G)} = \mathbb{E}_z[q(G(z))\nabla_{\theta_g} log(D(G(z)))] \tag{10}$$

The discriminator training is identical to the original setting in this work, but we suspect that adjusting the training scheme of discriminator may be beneficial as well, the details of which will be discussed in section 5.

Algorithm 1 Training PG-GAN

Train (D, G) like a regular GAN to convergence to stabilize the gradients
for $i \leftarrow 1$ to number of iterations **do**
  Sample a minibatch of noises z, $\{z^{(1)}, z^{(2)}, \ldots, z^{(m)}\}$ from $p_z(z)$
  Sample a minibatch of real example x_d, $\{x_d^{(1)}, x_d^{(2)}, \ldots, x_d^{(m)}\}$ from $p_{data}(x)$
  Generate example x_g, $\{x_g^{(1)}, x_g^{(2)}, \ldots, x_g^{(m)}\}$ where $x_g^{(i)} = G(z^{(i)})$
  Compute quality q, $\{q^{(1)}, q^{(2)}, \ldots, q^{(m)}\}$ where $q^{(i)} = q(x_g^{(i)})$
  Update G with $\frac{1}{m}\sum_i^m q^{(i)} \nabla_{\theta_g} log(D(x_g^{(i)}))$ with gradient ascent
  Update D with $\nabla_{\theta_d} \frac{1}{m}\sum_i^m log(D(x_d^{(i)})) + log(1 - D(x_g^{(i)}))$ with gradient ascent
end for
(Any gradient based optimization can be used to perform the gradient ascent)

4 Experiments

We evaluated this framework on the task of minimizing mean grasp quality of the generated objects as determined by the Dex-Net system and the task of minimizing number of connected components in the generated objects. In each case, 567 samples are generated from the final trained GANs in order to evaluate the shift in generator distribution. The model used for all experiments are the same:

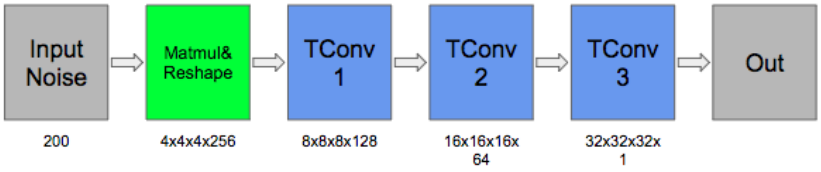


Figure 1: **Generator**: Noise is drawn uniform distribution in $[-1, 1]$, Matmul&Reshape converts $\mathbb{R}^{200}$ to $4 \times 4 \times 4 \times 256$ tensor, every transposed 3D convolution has $4 \times 4 \times 4$ kernels with stride 2 with batchnorm, out applies $sigmoid$

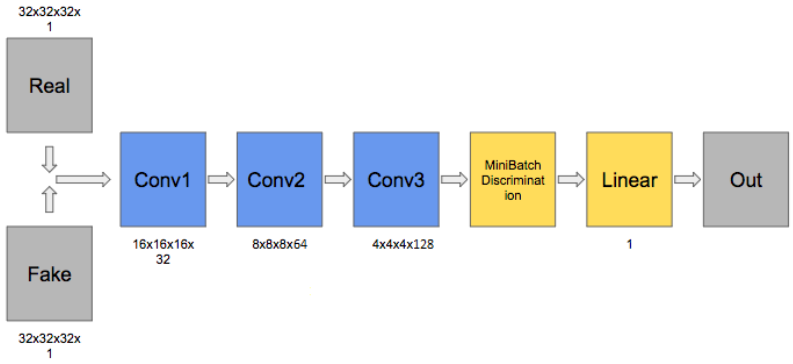


Figure 2: **Discriminator**: Every conv is $4 \times 4 \times 4$ 3D convolution with stride 2 with batchnorm, Minibatch discrimination is implemented as [15], linear layer converts the vector to dimension 1 and out applies $sigmoid$.

The above model is implemented in Tensorflow [16] and optimized with the ADAM Optimizer [17] at batchsize of 64. We have also used non-saturating heuristic loss and label smoothing to 0.8 to improve the stability and quality of training [18]

4.1 Minimizing mean grasp quality

Each 32x32x32 voxel sample was first remeshed using Lewiner marching cubes [19, 20], then smoothed with three iterations of Laplacian smoothing [21] to decrease the number of sharp corners, as the Dex-Net grasp quality model works best on relatively smooth surfaces. 10 grasps were sampled over the surface of each mesh using the rejection sampling method for antipodal point pairs from Dex-Net 1.0, and for each grasp the expected quality under perturbations in object and gripper pose is evaluated using Monte-Carlo sampling. These expected qualities are then averaged to produce the final mean grasp score for the object. This mean grasp score is linearly standardized and then centered around 1 and clipped to be larger than 0 and smaller than 2. This was done because the quality function is multiplied to the generator loss, so it needs to be non-negative (a negative quality would flip the gradient direction) and should be centered around 1 (multiplicative identity).

We trained this network on a synthetic prior consisting of three cylinders randomly merged together through constructive solid geometry, as we felt that a synthetic prior would provide a balance between having geometry that can allow for a wide range of grasp qualities and being a uniform enough set to allow the 3D-GAN to function. In addition, recent developments have shown that procedurally generated sets are as good as or better than ShapeNet when training for grasping [22].

We found that the final sample after 2000 initial GAN training iterations and 500 PG-GAN training iterations had mean grasp quality of 0.000886 compared to 0.001013 for a sample from a GAN trained for 500 more standard iterations after resuming from the 2000-iteration checkpoint. The standard deviation of mean grasp quality was 0.000315 before and 0.000263 after training. This means that the mean grasp quality decreased by 0.5 standard deviations. The distributions of grasp quality with and without PG-GAN training are shown in Figure 3.

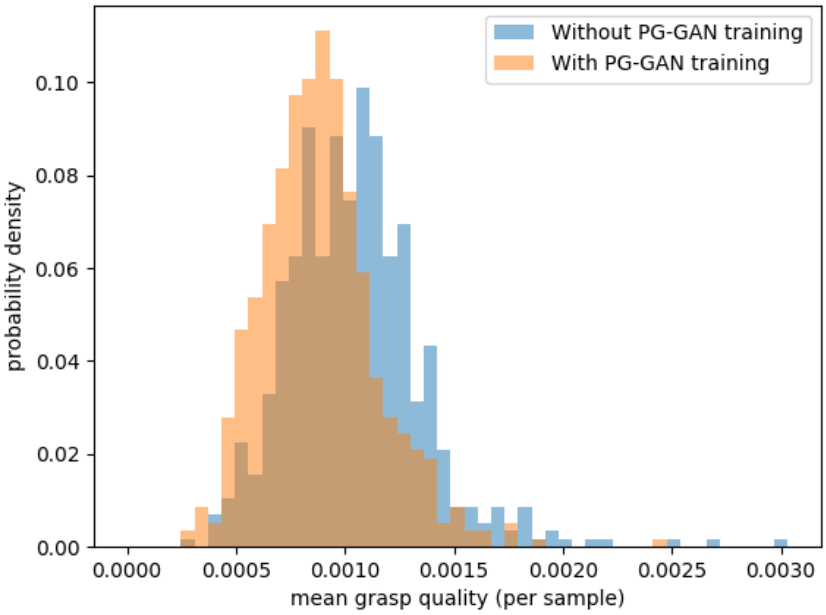


Figure 3: Histogram of grasp quality with and without PG-GAN training

Inspecting a random sample from the generated objects does not reveal any immediately noticeable change in geometry that would explain the shift in mean grasp quality.

(a) Without PG-GAN training

(b) With PG-GAN training

Figure 4: Objects sampled from the GANs with and without PG-GAN training. The 32x32x32 voxel grid outputs were remeshed with marching cubes, then smoothed with 3 iterations of Laplacian smoothing, which is the same processing done before they are sent to the Dex-Net system

4.2 Minimizing number of connected components

Each 32x32x32 voxel sample was remeshed using marching cubes, then the number of connected components was counted. This was chosen as a test of the PG-GAN as having only one connected component in a mesh is desirable quality, and because the number of connected components is a relatively nonlinear and nontrivial function.

This network was trained on a voxelized (using trimesh [23]) version of the KIT Cat50 [24] 3D model database, which contains three (of 112 total objects) very thin examples that resulted in vox-

elized versions with detached components and holes. These three were intentionally left in the set as bad data to increase the number of connected components in the outputs of the GAN without PG-GAN training. Some examples of objects from KIT are shown in Figure 5

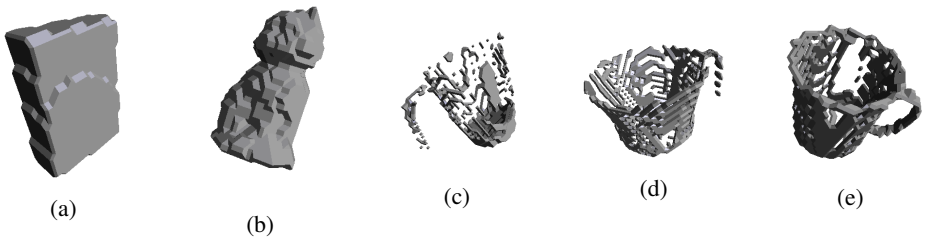


Figure 5: Some objects from KIT. (a) and (b) are examples of normal remeshed KIT objects, and (c), (d), and (e) are the three bad data points

We found that the final sample after 1000 initial GAN training iterations and 500 PG-GAN training iterations had mean 5.13 connected components, down from 14.86 for a sample from a GAN trained for 500 more standard iterations after resuming from the 1000-iteration checkpoint. The mode also moved to 1 connected component from 12. The distributions of connected component count with and without PG-GAN training are shown in Figure 6.

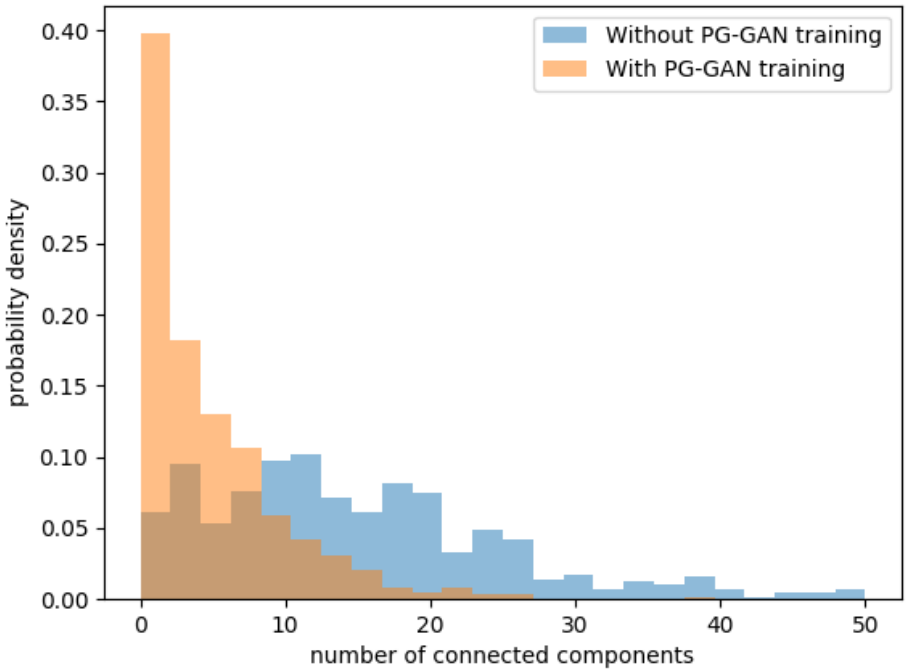
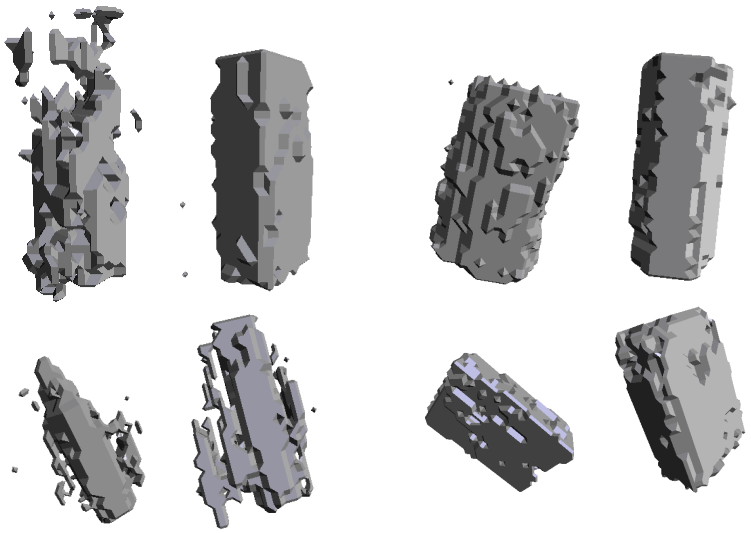


Figure 6: Histogram of grasp quality with and without PG-GAN training

Inspecting random samples from the generated objects with and without PG-GAN training clearly shows this difference.



(a) Without PG-GAN training

(b) With PG-GAN training

Figure 7: Objects sampled from the GANs with and without PG-GAN training.

5 Discussion

While not all of the experiments above worked as we predicted, we have made some interesting observations and problems. Two of the major difficulties are a as follows:

5.1 Effect of 2 Player Game

We learned that although one can establish the equivalence between policy gradients and the generator, the addition of a discriminator adds complication to the picture. When the generator tries to shift the distribution in response to the signal provided by the quality function, the discriminator will catch the shift and penalize the samples from the skewed distribution more. Originally, we hypothesized that the game would reach a new equilibrium through the quality factor but some of the results suggest that the dynamic is more intricate and requires further study. This may suggest that applying weighted gradients in the generator alone is not enough to achieve the goal. One potential solution would be to add weighted gradient into the discriminator as well to reach a new equilibrium. This procedure can be formalized via important sampling through introducing a uniform distribution over the union of generated and real objects.

5.2 Variance of Quality Estimate

In a vanilla policy gradient, various variance reduction techniques are required to obtain a more accurate reward estimation of trajectories. Many of these techniques deal with accumulation of errors over multiple time steps. Because our object generation task is done over a single time step, it does not suffer from accumulation of error; however, we still have the problem of inherent variance of the quality function. While functions like connected components are deterministic, functions such as grasp qualities do not have deterministic values or even are not well-defined. To estimate the grasp quality of an object, we used the Dex-Net 1.0 which use Monte Carlo sampling over the mesh surface to find valid grasps. This is inherently a stochastic process that has very large variance. It is computationally impractical or even intractable to sample many grasps. Even parallelized operation offers only marginally better efficiency improvement of sampling.

Furthermore, the smoothness and sensitivity of the quality function with respect to the parameters of the generator affects the performance of the network. For example, connected components would be a very sensitive function with respect to the parameters of the generator. If a noisy gradient causes the generator to produce a sample with small number of voxels outside of the main body of

the object (e.g. along the boundaries of the bounding box) to have values above 0.5, then it will drastically change the number of connected components even if the changes in the parameter space is very small. This also introduces additional variance into the framework.

6 Conclusions and Future Works

We have introduced a new class of GAN that leverages policy gradient to generate objects that improve some quality function. Through experiments we have shown that the PG-GAN can manipulate the underlying distribution of 3D objects learned by a regular GAN. We believe that this framework is general enough that it may see applications in other tasks.

We find that this new GAN framework has a weakness in that the discriminator can learn to differentiate between the real and generated distributions using the induced shift. In order to address this, future work can try to weight the discriminator as well.

The distribution shift in the adversarial objects for grasping experiment is small, and the distributions have significant overlap, making seeing what makes the objects adversarial difficult. This may be due to the previous problem with the discriminator, or it may also be due to the high variance of the Dex-Net system. Future work can try a neural net baseline for the Dex-Net system to reduce variance. Application of TRPO is also a possibility as it is the state of art policy gradient algorithm [25].

Additionally, GANS are not the only generative model we can leverage to generate adversarial objects. Other strategies, such as the variational autoencoder, can also be used and do not have the 2-player game effects.

References

[1] Jeffrey Mahler, Jacky Liang, Sherdil Niyaz, Michael Laskey, Richard Doan, Xinyu Liu, Juan Aparicio Ojea, and Ken Goldberg. Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. *arXiv preprint arXiv:1703.09312*, 2017.

[2] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[3] Konstantinos Bousmalis, Nathan Silberman, David Dohan, Dumitru Erhan, and Dilip Krishnan. Unsupervised pixel-level domain adaptation with generative adversarial networks. *CoRR*, abs/1612.05424, 2016.

[4] Xinyue Zhu, Yifan Liu, Zengchang Qin, and Jiahong Li. Data augmentation in emotion classification using generative adversarial networks. *CoRR*, abs/1711.00648, 2017.

[5] Jeffrey Mahler, Florian T Pokorny, Brian Hou, Melrose Roderick, Michael Laskey, Mathieu Aubry, Kai Kohlhoff, Torsten Kröger, James Kuffner, and Ken Goldberg. Dex-net 1.0: A cloud-based network of 3d objects for robust grasp planning using a multi-armed bandit model with correlated rewards. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 1957–1964. IEEE, 2016.

[6] Carlo Ferrari and John Canny. Planning optimal grasps. In *Robotics and Automation, 1992. Proceedings., 1992 IEEE International Conference on*, pages 2290–2295. IEEE, 1992.

[7] Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *CoRR*, abs/1612.00593, 2016.

[8] Angel X. Chang, Thomas A. Funkhouser, Leonidas J. Guibas, Pat Hanrahan, Qi-Xing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. Shapenet: An information-rich 3d model repository. *CoRR*, abs/1512.03012, 2015.

[9] 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 *Advances in Neural Information Processing Systems*, pages 82–90, 2016.

[10] Jesse Engel, Matthew Hoffman, and Adam Roberts. Latent constraints: Learning to generate conditionally from unconditional generative models, 2017.

[11] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. Seqgan: Sequence generative adversarial nets with policy gradient. *CoRR*, abs/1609.05473, 2016.

[12] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063, 2000.

[13] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. Restricted boltzmann machines for collaborative filtering. In *Proceedings of the 24th international conference on Machine learning*, pages 791–798. ACM, 2007.

[14] Geoffrey E Hinton. Deep belief networks. *Scholarpedia*, 4(5):5947, 2009.

[15] Tim Salimans, Ian J. Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *CoRR*, abs/1606.03498, 2016.

[16] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

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

[18] Ian Goodfellow. Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*, 2016.

[19] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *ACM siggraph computer graphics*, volume 21, pages 163–169. ACM, 1987.

[20] Thomas Lewiner, Hélio Lopes, Antônio Wilson Vieira, and Geovan Tavares. Efficient implementation of marching cubes' cases with topological guarantees. *Journal of graphics tools*, 8(2):1–15, 2003.

[21] David A Field. Laplacian smoothing and delaunay triangulations. *International Journal for Numerical Methods in Biomedical Engineering*, 4(6):709–712, 1988.

[22] Konstantinos Bousmalis, Alex Irpan, Paul Wohlhart, Yunfei Bai, Matthew Kelcey, Mrinal Kalakrishnan, Laura Downs, Julian Ibarz, Peter Pastor, Kurt Konolige, et al. Using simulation and domain adaptation to improve efficiency of deep robotic grasping. *arXiv preprint arXiv:1709.07857*, 2017.

[23] Michael Dawson-Haggerty. Trimesh. `https://github.com/mikedh/trimesh`, 2017.

[24] Alexander Kasper, Zhixing Xue, and Rüdiger Dillmann. The kit object models database: An object model database for object recognition, localization and manipulation in service robotics. *The International Journal of Robotics Research*, 31(8):927–934, 2012.

[25] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 1889–1897, 2015.