Multi View Approach to 3D Generative Models

Bryon Kucharski
University of Massachusetts Amherst
140 Governors Drive
Amherst, MA 01003-9264
bkucharski@umass.edu

Abstract

Generative models are a commonly studied topic in artifical intelligence that aim to understand the underlying structure of data by estimating a probability distribution of a dataset. Variational Autoencoders (VAE) and General Adverserial Networks (GAN) have become two of the most popular generative models. Promising results have been shown when combining VAEs and GANs by sharing the decoder network of a VAE and generator network of the GAN. Recent work has extended these architectures into the 3D realm. The 3D-GAN and 3D-VAE-GAN architectures are explored in this project and a new architecture is introduced to incorporate multiple views with the 3D-VAE-GAN for 3D object generation.

1. Introduction

The goal of many artificial intelligence researchers is to achieve human level intelligence in machines. Many algorithms are designed to learn and understand a given dataset to perform tasks such as classification. Humans have the ability to create things such as music, art, stories, etc. Thus, in order for machines to achieve human level intelligence, they must be able to create as well as understand. Generative models are a step towards teaching machines how to create. They aim to estimate the probability distribution of a dataset and novel data points could be sampled from this distribution.

Currently three main approach to generative models exist in machine learning: General Adverserial Networks (GANs) [2], Variational Autoencoeders (VAEs) [3], and Autoregressive models (such as PixelRNN [12]). GAN architectures use a two player adversial approach, where one model tries to generate realistic looking data and the other model determines which inputs are real and which are generated. A VAE encodes then decodes the input data and learns a latent distribution of the dataset. PixelRNN models a conditional distribution of each pixel given all previous pixels. [6] introduced a VAE-GAN architecture which combines the decoder of the VAE and the generator of the GAN to learn a generative model.

2. Related Work

While much progress has been made in generative approaches in 2D domains, there are also extensions of these models into the 3D domain. The work in [1] proposed a voxel based VAE and [11] proposed a mesh based VAE. [13] introduced the 3D General Adverserial network (3D-GAN) and is the main paper corresponding to this project. The main contribution of this work was the high-quality novel and realistic objects generated by leveraging advances in volumetric convolutional networks [14] and GANs. In addition to 3D-GAN, they also introduced a novel single image to 3D object algorithm named 3D-VAE-GAN. This algorithm combines the work from VAE-GAN with the 3D-GAN, where the decoder of the auto encoder network is shared with the generator of the GAN.

This project contains two main goals. First, an extension of the 3D-VAE-GAN architecture is explored to incorporate multiple views. Different methods of combining the latent representations of each view is explored. Secondly, an unofficial PyTorch implementation of the 3D-GAN work is found on GitHub [9], which is extended to contain 3D-VAE-GAN and the multi view 3D-VAE-GAN (MV-3D-VAE-GAN).

The extension of this project to a multi view 3D-VAE-GAN is based on the idea from the multi view convolution nerual networks [10]. In this work, the authors proposed a novel method for classifying 3D objects using multiple CNNs that operate independently of each other. To make a prediction on the class, each CNN is pooled together and passed through another CNN. In this project, each view is encoded into the latent vector, pooled together, and passed though the generator network to produce a 3D object.

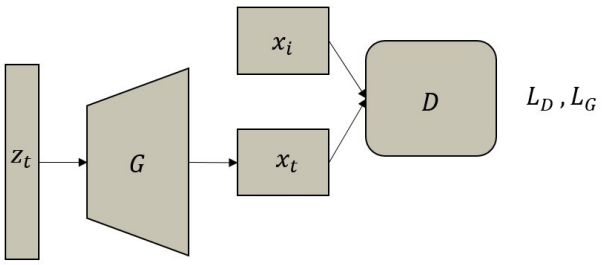

Figure 1. Architecture of GAN. Given a random latent vector z_t, the generator network G produces a 3D object x_t. The discriminator network D classifies which object is real and which is fake.

3. Method

3.1. Architectures

GANs can be described as a two player game. One player is a generator model which produces fake data points from a latent vector. The second player is a discriminator which is given both the generated data point and a real data point from the dataset and classifies which is real and which is generated. The goal is to have the generator successfully trick the discriminator into classifying the generated data point as real. This would mean the generator is producing data that is realistic. Given the data point x_i and a random vector z_t, the loss of the discriminator tries to maximize

$$L_D = \log D(x_i) + \log(1 - D(G(z_t)))$$

and is two-fold so that $\log D(x_i) \in (0, 1)$ is close to 1, and $\log(1-D(G(z_t))) \in (0, 1)$ is close to 0. The generator loss function tries to minimize

$$L_G = 1 - D(G(z_t))$$

so that $D(G(Z_t))$ is close to 1. Then, the total loss during training is combined as

$$L = L_D + L_G$$

3D-GAN architecture in Figure 1 follows the the original GAN architecture, but the output x_i is a 3D object instead of an image. It consists of a generator network that repeatedly upsamples with deconvolutional layers, starting from a one dimensional z_t vector to a 3D object.

The 3D-VAE-GAN proposes an addition to the 3D-GAN architecture. In this variation, the decoder of the VAE is shared with the generator of the GAN seen in Figure 2. Given an image y_i and a corresponding 3D object x_i, the VAE learns a distribution with z_{mu} and z_σ. This distribution is sampled to receive z_e. Then, the generator of the GAN produces a 3D object by upsampling z_e. During training, reconstructed loss is calculated as the distance between the generated 3D object and input 3D object

$$L_{recon} = \|G(z_e) - x_i\|$$

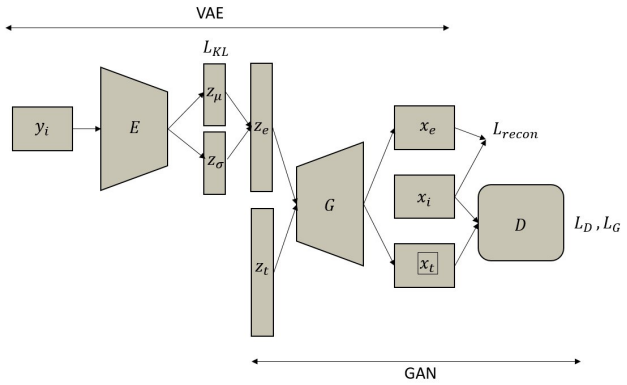

Figure 2. Architecture of 3D-VAE-GAN. The VAE is given a 2D image y_i and is encoded by E into a mean z_μ and variance z_σ. A latent vector z_e is sampled from this distribution and then passed through the generator G to get a 3D object x_e. A random latent vector z_t is passed through G to get the 3D object x_t. The discriminator D classifies if the real 3D object x_i or x_t is real or fake. The reconstruction loss is calculated between x_e and x_i.

During training, the KL Divergence [5] is calculated to ensure z_{mu} and z_σ are close to a normal distribution.

$$L_{KL} = -\frac{1}{2} * \sum(1 + z_{var} - z_\mu^2 - e^{z_{var}})$$

The loss for the generator is updated to include the reconstruction loss

$$L_G = 1 - D(G(z_t)) + L_{recon}$$

The total loss is the addition of all the separate loss functions

$$L = L_D + L_G + L_{KL}$$

The MV-3D-VAE-GAN is an extension of the 3D-VAE-GAN architecture. In this architecture, seen in Figure 3, the goal is to learn a better latent vector of the 2D image by merging multiple 2D images into a single representation. Each view is a 2D image $y_i \in Y$ of the same 3D object x_i and is encoded into a latent vector. Each view learns it's own distribution $z_{\mu_{y_i}}$ and $z_{\sigma y_i}$. Each is sampled to receive z_{ey_i} and then pooled together to receive a final representation of z_e. Then, a 3D object is generated and the reconstruction loss is calculated the same way as 3D-VAE-GAN. The only difference during training is the KL divergence is averaged across all views to ensure each $z_{\mu_{y_i}}$ and $z_{\sigma_{y_i}}$ is close to a normal distribution

$$L_{MVKL} = \frac{1}{N}(\sum_{i=0}^{N} -\frac{1}{2} * \sum(1 + z_{y_i} - z_{\mu_{y_i}}^2 - e^{z_{y_i}})$$

where N is the number of views. The total loss is updated to include the average KL divergence.

$$L = L_D + L_G + L_{MVKL}$$

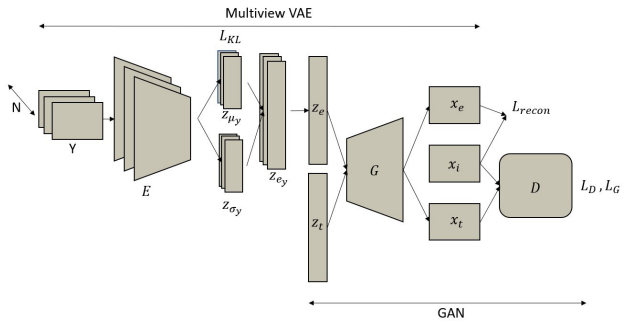

Figure 3. Architecture of MV-3D-VAE-GAN. The VAE is given multiple 2D images Y. Each image $y \in Y$ is encoded by E into their respective mean $z_{\mu y}$ and variance $z_{\sigma y}$. A latent vector z_{ey} is sampled from each distribution and then all combined using either max pooling or average pooling to receive a single dimension encoding z_e. Then, z_e is passed through the generator G to get a 3D object x_e. A random latent vector z_t is passed through G to get the 3D object x_t. The discriminator D classifies if the real 3D object x_i or x_t is real or fake. The reconstruction loss is calculated between x_e and x_i..

The generator network for each model included five transpose 3D layers with 3D batch normalization and a ReLU activation functions to map a single dimension vector into a 32x32x32 voxelized 3D object. The discriminator network for each model includes five 3D convolutional layers with 3D batch normalization, leaky ReLU activation functions with a sigmoid layer at the end to map a 3D object into a single scalar value representing the probability of being a generated object. The image encoding networks for the 3D-VAE-GAN and MV-3D-VAE-GAN models are a five layer 2D convolution network with 2D batch normalization and ReLU activation functions to map a the 2D image into a single latent dimension. All models where optimized using the ADAM optimizer with a learning rate of 0.0025, 0.0001, 0.001 for the generator, image encoder, and discriminator network.

3.2. Data and Preprocessing

The 3D-GAN architecture requires only a 3D object while the 3D-VAE-GAN architecture requires a 3D object and a corresponding 2D image of the object. The MV-3D-VAE-GAN architecture requires a 3D object and multiple 2D images of the 3D object at different views. The 'modelnet40v1png' dataset from the MVCNN paper [10] provides all the necessary images and 3D objects for all architectures. For this project, the models are trained using only the chair class from the ModelNet dataset.

The 3D objects were provided in .OFF file format and require a voxelization step as preprocessing. To achieve this, the binvox software developed by Patrick Min [8] was used to convert the .OFF files into .binvox files. Then, during the loading of the dataset in PyTorch, the 'binvox-rw-py' [7] script by Daniel Maturana is used to convert the binvox files into 3D arrays.

4. Experiments

In general, different GAN models can be compared as how well its distribution matches the distribution of the data. This can be achieved by a number of ways including visual comparison, reconstruction errors, or geometric metrics. GAN models typically do not have a reconstruction error because of the lack of an autoencoding architecture. However, since this work deals largely with the addition of VAEs to the GAN architecture, this provides an evaluation metric to compare models.

Two experiments were performed to answer two questions. The first experiment aims to answer the question if incorporating the autoencoding architectures help learn better parameters for generating novel and realistic looking 3D objects from random latent vectors. An empirical evaluation of Figure 4 reveals a few interesting aspects. During training, each algorithm was given a random latent vector z_t at epoch 250, 1000, and 2000, and the generated 3D objects are shown in the figure. While it may be difficult to concretely say which algorithm has the best generated 3D object, it is definite which models had the worst. In the early epochs and later epochs, the 3D-GAN algorithm was not able to produce an object that resembles a chair, meaning it took longer to train and at some point the generator diverged. MV-3D-VAE-GAN with max pooling suffered from this problem as well in the later epochs of training. The 3D-VAE-GAN on epoch 2000 and the MV-3D-VAE-GAN with mean pooling on epoch 250 had the most realistic and novel looking chairs. This may imply that MV-3D-VAE-GAN with mean pooling can learn in less epochs than all other algorithms.

One important distinction between the algorithms is the computation time to train. There is a slight computation increase between the 3D-GAN and the 3D-VA-GAN due to the autoencoding step, but there is a massive increase in computation time between the 3D-VAE-GAN and both MV-3D-VAE-GAN algorithms. This is due to the fact that each view is encoded separately and not vectorized. This leaves an direction of future work for the multi view algorithms. Thus, if computation time is a factor of interest, the 3D-VAE-GAN model would be the choice of algorithm for this experiment.

The second experiment aims to answer the question of which autoencoding algorithm produces the best 2D image to 3D reconstruction. The 3D-GAN algorithm is not included in this experiment due to the lack of 2D to 3D capabilities. To answer this question, each autoencoding algorithm is trained for 1000 epochs and ran on a test set of 3D objects and corresponding images, and the reconstruction error calculated. Based on Table 1, the MV-3D-VAE-GAN

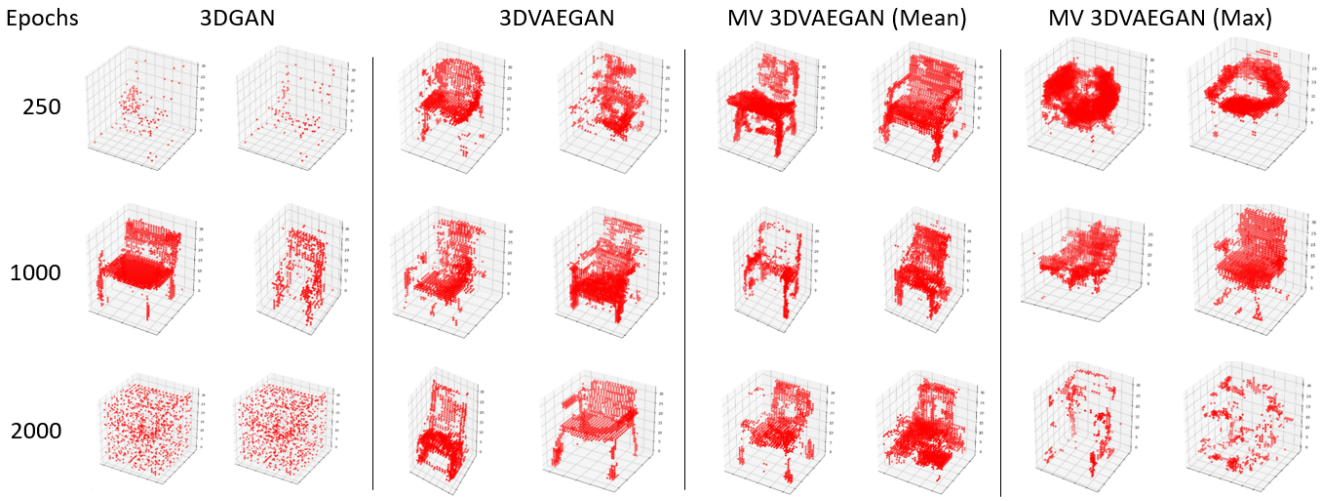

Figure 4. Examples of 3D objects generated by each algorithm from a random latent z_t vector at different epochs in training.

with max pooling algorithm had the lowest reconstruction error.

The authors of [13] used the the IKEA dataset which contains images of the 3D IKEA objects. The images in this dataset are captured in the wild and have heavy occlusion, and they where still able to provide quality results on the 2D image to 3D object task using the 3D-VAE-GAN algorithm. In the dataset I used, all of the images are not in the wild and have no occlusions. This may contribute to the success of the 3D-VAE-GAN on my dataset. A single image not in the wild with no occlusion may be enough to learn the parameters. I hypothesize that the MV-3D-VAE-GAN algorithms would excel on datasets in the wild with occlusion because more information will be extracts. Although there may be an increase in performance, it would be unlikely to achieve multiple views of the same object if they are in the wild.

Algorithm	Average Reconstruction Loss
3D-VAE-GAN	822
MV-3D-VAE-GAN (Mean)	861
MV-3D-VAE-GAN (Max)	**739**

Table 1. Average Reconstruction Error on Test Set

5. Conclusion

3D-GANs are a architecture introduced by [13] and combine previous works from 2D GANs and volumetric convolutional networks. 3D-VAE-GANs were also introduced and provide a way to perform single 2D image to 3D object generation. This project extends the 3D-VAE-GAN by incorporating the idea of view pooling from [10] to introduce a multi view 3DVAEGAN. Using the 'modelnet40v1png' dataset from the MVCNN project, the 3D-GAN,3D-VAE-GAN, and MV-3D-VAE-GAN with both average pooling and max pooling are compared on two experiments. 3D-VAE-GAN and MV-3D-VAE-GAN with average pooling where empirically the best model on the first experiment to test the learned parameters of the generator network. MV-3D-VAE-GAN with max pooling performed best on the reconstruction of test data points. In addition the GitHub repository was updated to include a PyTorch implementation of 3D-VAE-GAN and MV-3D-VAE-GAN and are available at [4]. Future work on this project would include a vectorized version of the multi view code and a new dataset of multi view objects in the wild.

References

[1] Andrew Brock, Theodore Lim, James M Ritchie, and Nick Weston. Generative and discriminative voxel modeling with convolutional neural networks. *arXiv preprint arXiv:1608.04236*, 2016.

[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] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[4] Bryon Kucharski. Multiview-3d-vae-gan. `https://github.com/bryonkucharski/Multiview-3D-VAE-GAN`, 2019.

[5] Solomon Kullback. *Information Theory and Statistics*. Wiley, New York, 1959.

[6] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. *arXiv preprint arXiv:1512.09300*, 2015.

[7] Daniel Maturana. binvox-rw-py. `https://github.com/dimatura/binvox-rw-py`, 2016.

[8] Fakir S. Nooruddin and Greg Turk. Simplification and repair of polygonal models using volumetric techniques.

IEEE Transactions on Visualization and Computer Graphics, 9(2):191–205, 2003.

[9] rimchang. 3dgan-pytorch. `https://github.com/rimchang/3DGAN-Pytorch`, 2017.

[10] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik G. Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *Proc. ICCV*, 2015.

[11] Nobuyuki Umetani. Exploring generative 3d shapes using autoencoder networks. In *SIGGRAPH Asia 2017 Technical Briefs*, page 24. ACM, 2017.

[12] Aaron Van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. Conditional image generation with pixelcnn decoders. In *Advances in neural information processing systems*, pages 4790–4798, 2016.

[13] 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.

[14] 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*, pages 1912–1920, 2015.