Virtual GAN

Prateek Agarwal
University of Massachusetts Amherst
prateekagarw@umass.edu

Aidan Nuzum-Clark
University of Massachusetts Amherst
anuzumclark@umass.edu

Abstract

We propose Virtual GAN, an extension and combination of Multi-Agent Diverse Generative Adversarial Nets (MAD-GAN) and Conditional Generative Adversarial Nets (CGAN). Virtual GAN is a GAN architecture that virtualizes multiple generators on a single underlying hypervisor generator; this is achieved by conditioning on different GANs, similar to how CGANs utilize labels. Like MAD-GAN, the discriminator is designed to leverage multiple virtual GANs and force them to capture diverse modes, behaving as a mixture model over the true data distribution. This method aims to drastically reduce the number of parameters needed by multiple GANs, like in MAD-GAN, while still providing the user with control over which modes to generate from; this can lead to easier generation of balanced synthetic datasets. Since our method is unsupervised, the multiple virtual GANs can be further used to cluster the data. We perform multiple experiments exploring the properties of Virtual GANs, as well as comparing them to their counterparts, namely CGAN and generic GANs. On MNIST, we show that our Virtual GAN is capable of separating the 10 different classes of digits into 10 Virtual Generators, while having generative performance comparable to its counterparts, and with fewer parameters. We also show that our GAN is robust to changes in the number of generators (above and below the number of identifiable modes), and continues to seperate the data reasonably. Finally, we show that the new discriminator achieves good performance on the unsupervised representation task.

1. Introduction

Since their introduction in 2014, in [4], Generative Adversarial Networks, commonly known as GANs, have become a largely popular architecture for generating photo-realistic images. They work by pitting a generator network, G, whose goal is to generate realistic images, against a discriminator network, D, whose goal is to identify real images (in contrast to generated images), in a minimax game where G tries to fool D. There has since been significant research into improving the stability, convergence and control of GANs. However, although generic GANs have been shown to generate from the diverse modes of unlabeled datasets, it still remains difficult to generate, in a balanced manner, from desired modes of a dataset. Currently, this would require a significant amount of sampling and human supervision to accomplish.

To combat this issue, the authors of MAD-GAN (Multi-Agent Diverse GAN) [3] use multiple-generators, coupled with a discriminator and a new loss function to explicitly force the GANs to model diverse modes. In addition to identifying real data from generated data, the new loss function forces the discriminator to also identify which generator the image came from. The authors describe that in order for the discriminator to do a good job, it would have to force the generators to become distinct from one another, also avoiding mode collapse. The model achieved remarkable performance, outperforming competing methods, and demonstrated that, under the new loss, the generators collectively behaved as a mixture model, with each generator capturing a different mode of the dataset. Although some of the parameters between the generators can be shared, for diverse datasets, this approach can potentially lead to a large increase in the number of parameters and training time, limiting its usability. Since most of the datasets that were used, like MNIST, were originally labeled and achieved good performance using Conditional GANs [6], we knew that diverse modes can be generated in a controlled manner using only a modest number of parameters.

Our goal remained to develop a GAN architecture which would have the same abilities as multiple generators modeling seperate modes, but with a complexity closer to its conditional GAN counterparts. We designed the generator based on the idea of a hypervisor, which is designed to run multiple virtual machines on the same underlying hardware. In addition to taking in noise, the underlying generator is designed to also take in an integer representing which "Virtual" Generator to pass through. For each Virtual Generator, a unique embedding is concatenated to the noise, similar to a Conditional GAN, to maintain the illusion of a different generator modeling a unique mode. On the other end, we

use a single discriminator with the same loss function used by MAD-GAN, in order to force the different Virtual Generators to model diverse modes.
We perform experiments using MNIST, as a proof of concept, and showed that our Virtual GAN was able to seperate, unsupervised, the different modes of the dataset into different Virtual Generators, allowing for tightly controlled sampling. We also qualitatively compare the generative performance of our model to CGANs and generic DCGANs and observe comparable, if not better quality. In a more realistic setting where the number of modes is unknown, we show that our model is robust to different numbers of virtual generators, and provides reasonable clustering of the true distribution. Using the test dataset of MNIST, we further show that the discriminator learns useful unsupervised representations.

2. Related Work

There has been some prior work in creating new GAN architectures which allow generative control over different modes in the dataset. InfoGAN [1], in an unsupervised fashion, uses only a small number of noise variables as latent codes, and captures the diversity of the dataset by maximizing the mutual information between it and the real data. For GANs producing faces, [8] shows that an unconstrained GAN can be converted to a controllable GAN using a factorization of the latent space after training. They display the ability to change individual attributes such as age, gender etc of various faces.
There has also been some work in exploring the effectiveness of using multiple generators and discriminators in tandem, similar to MAD-GAN. Hoang *et al*. [5] utilize multiple generators, a discriminator and a classifier which determines which generator the image came from. Samples created from the generators are picked at random, one at a time, and selected as the output, similar to a mixture model. Durugkar *et al*. [2] uses multiple discriminators instead, taking the maximum output and using it for more consistent feedback for the generator. They explore different frameworks varying the harshness of the discriminator "teacher", and its resulting effect on the generator's training loop.

3. Preliminaries

This section is intended to give an overview of GANs, including model architectures we borrow heavily from. The learning problem for GANs, introduced by Goodfellow *et al*. [4], is one in which the generator G, optimizes its parameters θ_G, to generate samples, given noise z, whose resulting distribution p_G is close to the real data distribution p_{data}. To guide G to create realistic examples, the Discriminator D, aims to differentiate and identify samples from p_{data} and p_G. These two networks *G* and *D* then engage in a minimax game where G seeks to fool D. The objective function is as follows:

$$\min_G \max_D V(D,G) = \\ \mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}[\log(1 - D(G(\mathbf{z}))] \tag{1}$$

The Discriminator D, given an input **x** / G(**z**), outputs a probability of whether it is from p_{data}; D aims to maximize this probability over the real data, and minimize it over the generated data from G. On the other hand, G aims to maximize the output probability from D over its generated samples.

A logical extension to generic GANs, in the presence of labeled data, are Conditional GANs [6]. Introduced by Mirza and Osindero, these GANs utilize a generator G and discriminator D conditioned on the available labels **y**. The goal of G is to produce realistic images based on the class label. G takes both noise z and label **y** to generate the image, and D also takes in label **y** to seperate its criticism per the label. The two networks once again play a minimax game over the following objective function:

$$\min_G \max_D V(D,G) = \\ \mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}[\log D(\mathbf{x}|\mathbf{y})] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}[\log(1 - D(G(\mathbf{z}|\mathbf{y}))] \tag{2}$$

We borrow heavily from Multi-Agent Diverse Generative Adversarial Networks (MAD-GANs) by Ghosh *et al*. [3]. They use multiple generators G_i along with a new discriminator objective which encourages the different generators to model diverse modes. In addition to differentiating between images from p_{data} and p_G, the new objective forces the discriminator D to also identify which generator the image came from. Given k generators, the discriminator outputs $k+1$ softmax probabilities; the first k representing the probability that it came from the k^{th} generator, and the $k+1$ entry representing the probability that it came from p_{data}. Images from different generators and p_{data} are then put into the discriminator with their respective
labels $\in [1,\ k+1]$ ($k+1$ being the real data label) and D's objective is then to simply maximize the negative cross-entropy loss function.
Retaining the minimax framework, each of the generators G_i, $i \in [1, k]$ seek to minimize the original GAN objective, specifically:

$$\mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}[\log(1 - D(G_i(\mathbf{z}))] \tag{3}$$

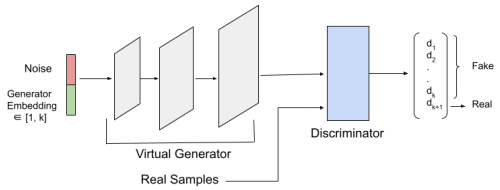

Figure 1. Virtual GAN architecture.

4. Virtual GAN

Using generic GANs, it can be especially difficult to sample from desired modes of the dataset in a balanced manner. To reduce the overhead of multiple Generators, we propose Virtual GAN which uses multiple "Virtual" Generators on a single underlying "Hypervisor" Generator, each of which capture a unique mode of the dataset. Our model works in an unsupervised setting, and allows for controlled sampling from different modes after training. To guide the training of multiple Virtual Generators, we utilize the discriminator from MAD-GAN.

4.1. Virtual GAN Architecture

The architecture for the Virtual GAN consists of a single underlying "hypervisor" generator and a single discriminator with a multi-generator loss function. In order to simulate multiple generators, the hypervisor generator takes both noise, and a label corresponding to which generator to simulate (the label can be a single number). Practically, the label is fed into an embedding layer, whose outputs are then concatenated with the input noise and fed through the hypervisor generator. The embeddings are the only unique parameters not shared by the different virtual generators, VG_i, i.e the entire rest of the generator is shared to reduce parameter cost. Similar to MAD-GAN, the objective forces the discriminator D to identify samples from p_{data}, as well as identify which virtual generator the image came from. In order to do well, the discriminator pushes the generators towards different modes of the data so that their outputs remain distinctive; meanwhile, the generators only need to make their outputs realistic.

4.2. Objective and Loss

Given k virtual generators, VG_i ; $i \in [1, k]$, the discriminator, D, outputs $k + 1$ softmax probabilities; the first k representing the probability that it came from the k^{th} virtual generator, and the $k + 1$ entry representing the probability that it came from p_{data}. Images from the different generators and p_{data} are then put into the discriminator with their respective labels $\in [1, k+1]$ ($k+1$ being the real data label) and D's objective remains to simply maximize the negative cross-entropy loss function.
As in other instances of GANs, VG_i and D are engaged in a minimax game in which each generator seeks to minimize the following equation:

$$\mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}[\log(1 - D(VG_i(\mathbf{z}))] \tag{4}$$

From the perspective of a conditional GAN, the hypervisor generator can be rephrased as a GAN conditioning on k unique labels, $\mathbf{y}_i$; $i \in [1, k]$, one for each generator. A virtual generators minimization objective is then:

$$\mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}[\log(1 - D(G(\mathbf{z}|\mathbf{y}_i))] \tag{5}$$

In practice the virtual generators work to maximize: $\mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}[\log(D(G(\mathbf{z}|\mathbf{y}_i)]$ as it is equivalent.

4.3. Comparison to Other Methods

Our architecture aims to leverage the demonstrated effectiveness of Conditional GANs along with the diversity modeled by MAD-GANs. In comparison to MAD-GAN, our approach aims to require much fewer parameters, since we use a single underlying generator, and the unique embeddings employed by each virtual generator do not add significant overhead. We obtain sufficient capacity in each of our virtual generators using this simple technique and capture the diverse modes of the dataset. The learned hypervisor generator can then be used to generate from any desired mode and be used for balanced data generation etc. In comparison to Conditional GANs [6], it is notable that we disentangle the modes and learn to generate from different classes in an unsupervised fashion. In essence, Virtual GAN automatically clusters the data and seperates it into different virtual generators. Additionally, there is little change and overhead required to transform generic GANs into Virtual GANs, while providing significantly more control and insight over the generation process.

5. Experiments

We present an extensive analysis of the Virtual GAN on the MNIST dataset (due to time and compute constraints). Additionally, every Virtual GAN was built on top of a DCGAN backbone, as they have proven performance [7]. All of our results and comparisons are evaluated using qualitative assessment since we could not find another suitable metric for image generation.
As a proof of concept, since we know that there are 10 classes in MNIST, we evaluate the performance of the unsupervised Virtual GAN with 10 virtual generators. For our model to be successful, we expect each generator to capture a different mode/class. We also compare the images generated, with similar model parameter counts, from Virtual GAN, classic GANs and CGANs; as a baseline, we expect our model to be as good as CGANs since CGANs use the

Figure 2. Result of training a normal GAN on MNIST. Each row is from the same GAN.

Figure 3. Result of training a virtual GAN with 10 virtual generators on MNIST. Each row corresponds to the output of a different virtual generator.

labels we are implicitly trying to learn. We also investigate the robustness of the Virtual GAN architecture when the number of modes in the dataset are not know, as is the general case in an unsupervised setting; we analyse the case when 5 or 15 virtual generators are used on MNIST. Lastly, we investigate the quality of the discriminator by evaluating it on the unsupervised representation task.

5.1. Generation Based on Known Number of Classes

When the number of classes is know, 10 in the case of MNIST, we find that the unsupervised Virtual GAN is capable of successfully disentangling the modes into the different virtual generators. We train a Virtual GAN with 10 generators on MNIST and the results can be seen in Fig. 3; each row contains 10 outputs from a single virtual generator. It is important to note that this clustering/implicit labels were learned in an unsupervised fashion during the learning process. Each virtual generator learns a different mode/class, in this case a digit between 0 and 9, and can be reliably used for controlled generation of images. Qualitatively, we find that each generator is capable of generating diverse and sharp intra-class images.

5.1.1 Comparison to CGAN and GAN

We qualitatively compare the generated images from Virtual GAN (Fig. 3) , trained with 10 virtual generators, with Conditional GANs (Fig. 4) and normal GANs (Fig. 2) on the MNIST dataset. Qualitatively, we can observe that the results of normal GAN are inferior to those of Virtual GAN. On the other hand, Conditional GANs and Virtual GANs have comparable performance. As seen in Tab. 1 the total number of parameters between the different vGANs are roughly equivalent and do not factor into the analysis. Both Conditional GANs and Virtual GANs allow for controlled generation from distinct classes, although Virtual GAN was able to do this unsupervised. We can also observe that the intra-class quality and variety is roughly equivalent between the two architectures. To summarize, our Virtual GAN, at no added complexity, is able to produce output of comparable quality to conditional GANs, without label supervision.

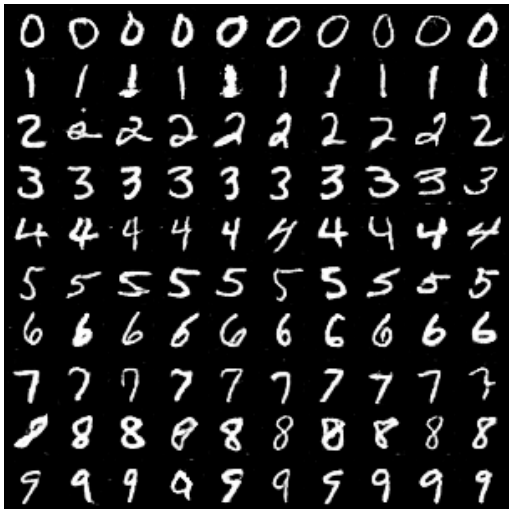

Figure 4. Result of training a conditional GAN on MNIST. Each row corresponds to output from the generator when conditioned on the corresponding class label.

Type of GAN	Total Parameters
Original GAN	945,314
Conditional GAN	1,481,570
Virtual GAN	816,776

Table 1. Each GAN architecture has a similar number of parameters.

Figure 5. Result of Virtual GAN with 5 generators trained on MNIST. Each row corresponds to output from a single virtual generator.

5.2. Robustness to Number of Generators

In most unsupervised settings, there is little estimation of the number of modes in the dataset. We evaluate the robustness of Virtual GAN when the number of generators is not equivalent to the number of modes in the dataset, namely MNIST with its 10 modes. We first examine the impact of having fewer generators than there are modes; we train 5 virtual generators on MNIST. We find that the 10 original modes cleanly separate into the 5 virtual generators, 2 modes per generator Fig. 5. 0s and 6s, 4s and 9s, 7s and 1s, 3s and 5s, 8s and 2s cleanly cluster themselves into the different generators. Interestingly, on multiple runs of this experiment, different pairs of digits were grouped together, all along intuitively reasonable lines.

We also examined the impact of having more generators than there are modes, and train 15 virtual generators on MNIST, whose results of which can be seen in Fig. 6. We see that 10 of the 15 generators produce solely a single digit from MNIST. Notably, each digit 0-9 is covered by these 10 generators. The remaining 5 generators each produce combination of several digits, similar to the 5-generator Virtual GAN. This indicates that in a domain where the true number of modes is unknown or ambiguous, Virtual GAN is robust and performs reasonably well, separating the modes along intuitive lines. Visual inspection of the generators could also lead to intuition about the number of real modes in a dataset.

5.3. Unsupervised Representation Learning

Although the main focus of virtual GAN is on analyzing its impressive generative power, we also investigate the quality of the discriminator through an unsupervised representation task. We extracted the feature representations produced by the discriminator on a held out test-set (not used for GAN training either) of MNIST, and trained a SVM to classify the labels of the digits. On a held out portion of the test-set, our Virtual GAN with 10 generators and a normal GAN had accuracies of 98.9% and 98.1% respectively, showing that our discriminator learned useful representations. Since the performance difference between the discriminators is not as large as the generators, this leads us to believe that the main benefit of Virtual GANs is providing extra self-supervision to the generator.

Figure 6. Result of Virtual GAN with 15 generators trained on MNIST. Each row corresponds to output from a single virtual generator.

6. Conclusion: Discussion and Future Work

We have presented an intuitive extension of GANs, combining ideas from both Conditional GANs and Multi-Agent Diverse GANs. We leveraged MAD-GANs ability to find diverse modes, while reducing the number of parameters needed, using the proven performance of Condition GANs. We demonstrate Virtual GAN's ability to capture inter-class and intra-class variation in the dataset, without the need for labeled data. In the future we would like to test the behaviour of Virtual GANs on color datasets, and other challenging datasets with unclear modes.

References

[1] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. *Advances in neural information processing systems*, 29, 2016. 2

[2] Ishan Durugkar, Ian Gemp, and Sridhar Mahadevan. Generative multi-adversarial networks. *arXiv preprint arXiv:1611.01673*, 2016. 2

[3] Arnab Ghosh, Viveka Kulharia, Vinay P Namboodiri, Philip HS Torr, and Puneet K Dokania. Multi-agent diverse generative adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8513–8521, 2018. 1, 2

[4] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020. 1, 2

[5] Quan Hoang, Tu Dinh Nguyen, Trung Le, and Dinh Phung. Mgan: Training generative adversarial nets with multiple generators. In *International conference on learning representations*, 2018. 2

[6] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014. 1, 2, 3

[7] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015. 3

[8] Yujun Shen, Jinjin Gu, Xiaoou Tang, and Bolei Zhou. Interpreting the latent space of gans for semantic face editing. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9243–9252, 2020. 2