A Step Beyond Generative Multi-adversarial Networks

Aman Singh(✉)

Department of Computing Science, University of Alberta,
Edmonton, AB T6G 2E8, Canada
amansing@ualberta.ca

Abstract. In this paper we modify the structure and introduce new formulation to improve the performance of the Generative adversarial networks (GANs). We achieve this based on the discriminating capability of the Generative Multi-Adversarial Network (GMAN), which is a variation of GANs. GANs in general has the advantage of accelerating training at the initial phase using the minimax objectives. On the other hand, GMAN can produce reliable training using the original dataset. We explored a number of improvement possibilities, including automatic regulations, boosting using Adaboost and a new Generative Adversarial Metric (GAM). In our design, the images generated from noisy samples are reused by the generator instead of adding new samples. Experimental results show that our image generation strategy produces better resolution and higher quality samples as compared to the standard GANs. Furthermore, the number of iterations and the required time for quantitative evaluation is greatly reduced using our method.

Keywords: Multi-discriminators · Generators · Adversarial Minimax · GAN · GMAN · GAM

1 Introduction

Generative adversarial networks (GANs) is a powerful subclass of generative modeling. It had received a tremendous amount of success when applied to image generation, editing, domain adaptation and semi-supervised learning [1]. GANs [2] provides a structure for the implementation of a generative model based on the concept of a two-player minimax game. There is player one (the generator), who tries to generate realistic images from the noisy samples. The model tries to learn a deterministic transformation G from a simple distribution p_z, with only one objective to match the data distribution p_d. A feedback is provided by the discriminator to the generator showing how realistic the produced samples are in comparison to another player. The discriminator has to determine whether the samples are produced by the generator, or drawn from the actual dataset.

In this study, our implementation is based on the Deep Convoluntional GANs (DCGAN), which is a variantion of the state-of-art GANs structure. We incorporated a new and improved formulation of Generative multiple-Adversarial Metric

A. Basu and S. Berretti (Eds.): ICSM 2018, LNCS 11010, pp. 345–356, 2018.
https://doi.org/10.1007/978-3-030-04375-9_29

(GMAN) [3]. GMAN uses multiple discriminators and is also a variation of GAN. We modified the structure of GMAN and used a pre-trained GMAN to get better image resolution and quality. Experiments were conducted using the CIFAR-10 dataset [4].

In the remaining paper, Sect. 2 reviews related work. In Sect. 3, we describe the structure of GANs. Section 4 presents our proposed structure and the new formulation. Sections 5 and 6 shows the experimental setup and results respectively. Finally, we conclude the paper in Sect. 7. Future work is given in Sect. 8.

Our main contribution in the research of GANs are: (a) the images generated from noise are used again by the generator instead of taking new samples; (b) A new methodology for GMAN, a multi-discriminator GAN framework with the reuse of images generated from generator; (c) A new a generative multi-adversarial metric (GMAM) formulation for separately trained frameworks to perform pairwise evaluation on them;

2 Related Work

There are ongoing challenges in the study of GANs, which include convergence properties [5] and stability in terms of optimization [6], but researchers believe that the most critical and difficult challenge is the quantitative evaluation of GANs. The most classical approach towards evaluating generative models is based on the likelihood of the models which in most cases is intractable. Additionally, the log-likelihood can be approximated for distributions on low-dimensional vectors but when it comes to the context of complex high-dimensional dataset the task becomes extremely overwhelming and challenging [7]. In [8] a sampling algorithm to estimate the hold-out log-likelihood was proposed, which had many drawbacks. The key drawback was the assumption of the Gaussian observation model which carried over all issues of kernel density estimation in high-dimensional spaces. A partial solution to this was provided by Theis et al. [9], in which the likelihood was higher but visual quality and latency was low. Furthermore, it was argued that using Parzen window density estimates as the likelihood estimation provides incorrect results, and thus ranking the models based on this estimation was highly discouraged.

In recent developed GAN frameworks, the algorithms between learners are carefully formulated so that Nash equilibrium should harmonize with appropriate set of criteria. Nash equilibrium is a system involving two participants interacting with each other and these participants cannot gain over the other by changing the strategies if the strategies of the other participant remain the same. Initially the main idea behind the development of GANs was on generating images (e.g., MNIST [10]) and CIFAR [11]. However, it has been proven that GANs can be applied on a variety of application domains, which include transferring of domain knowledge [12], imitation of the expert policies [13] for better understanding the system, censored representation learning system [14], learning of features [15] to evaluate models in a better manner, extending the work of GANs to semi-supervised learning [16] and improving the image generation [17] process and

methodology. All these fields of study have presented a lot of potential and successful results. Considering all these successes, GANs are still very tedious, critical and difficult to train, and thus have room to improve. So far research to improve the training techniques focuses on understanding and generalizing GANs with the aim of providing better results in terms of quality of generation instead of generation running time.

3 Overview of GAN

The logic behind GANs is a minimax game between the generator and the discriminator. In this minimax game, the following function is expected to be optimized:

$$\min_G \max_D V(D,G) = E_{x\sim P_d(x)}[\log(D(x))] + E_{z\sim P_z(z)}[\log(1 - D(G(z)))] \quad (1)$$

where the true (correct) data distribution is assumed to be $p_d(x)$ and simple distribution can be assumed to be $p_z(z)$, which can generally be fixed and is easy to draw samples [2,3]. The differentiation of the functional space of discriminator, D and the elements of the functional space is performed by using various differential methods. At the same time, it is assumed that for generator the distribution induced is $p_{G(x)}$ for the functional space $G_\theta(z)$ and lastly, D and G are the deep neural network which is usually the case in generative adversarial models.

In the introductory and initial work on GANs [2], it was anticipated that with the optimal discriminator $D^* = arg\max_D V(D,G)$ from oracle, with sufficient network capacities and capabilities, the minimization function (stochastic gradient descent) on $p_G(x)$ will provide the required solution, $p_G(x)$ equal to $p_d(x)$, which can be applied to match the generator distribution exactly with data distribution to produce the high quality results for the enhancement of the gradient signals and better approximation. In practice, it was found that it required to replace the second term of the equation, $\log(1 - D(G(z)))$ with $-\log(D(G(z)))$ at the very beginning of the game so that the game will no longer be considered as a zero-sum game. In the convergence and optimality proof using oracle, the provided $D*$, can give a minimization only over G:

$$\min_G V(D^*,G) = \min_G C(G) = -\log(4) + 2JSD(p_{data}||p_G) \quad (2)$$

where JSD stands for Jensen-Shannon divergence, C(G) is the function which is necessary to minimizes JSD. However, very little is known about D* when the minimization of $V(D,G)$ is attempted [2]. This insight leads to the development of a model that tries to minimize maximum mean discrepancy (MMD) for all of the moments of $p_G(x)$ with $p_d(x)$ [18]. In one of the other approaches known as EBGAN [19], the objective of the generator and discriminator is to take real-valued "energies" as the input to the functions instead of using the probabilities. Different perspectives of JSD have been extended for GANs to achieve more general divergences [20,21].

In a recent paper [22] on the training dynamics of generative adversarial networks, the authors described the problems and possible solutions in training GAN. In the training process of GAN, people have found that as the discriminator gets better, the updates to the generator get consistently worse; if we just train D until convergence, its error will go to zero. The authors argue that this is because of the discontinuous distributions and distribution supports lie on low dimensional manifolds. In the case of GANs, the distribution of current sample x is defined via sampling from a simple prior $z \sim p(z)$. If the dimensionality of z is less than the dimension of x (as is the typical case in GAN), then it is impossible for the distribution to be continuous. It is also proved that if the two distributions we care about have supports that are disjoint or lie on low dimensional manifolds, the optimal discriminator will be perfect and its gradient will be zero almost everywhere.

4 Backpropagation in Generative Adversarial Networks (GANs)

In order to address some of the deficiencies in GANS and inspire further improvements, we implemented a framework of backpropagation for the generated images based on two main variants of the GANs: (a) Deep Convolutional Generative Adversarial Network (DCGAN) [1]; (b) Generative Multi-Adversarial Network (GMAN) [3].

4.1 Deep Convolutional Generative Adversarial Network

We consider using Deep Convolutional Generative Adversarial Nets (DCGAN), a topologically constrained variant of conditional GAN, as our basic GAN structure and baseline. DCGAN is by far the most cited basic GAN structure and remains as a base line or reference architecture for other models. It has outstanding performance in dealing with image processing tasks. In our study, we trained the model on the CIFAR-10 dataset [4]. We used the backpropagation of generated image methodology to improve the results of current methods by changing the parameters and structure of the system. DCGAN pairs learnt a hierarchy of representations from object parts level to scene level in both the generator and discriminator [1]. We find that comparing to other models, DCGAN is stable to train and is useful to learn unsupervised image representations.

- Structure of DCGAN
  In the DCGAN model, we introduced the following changes to the CNN architecture [1].
  - Replace pooling layers with strided convolutions (discriminator) and fractional-strided convolutions (generator), which allows the network to learn its own spatial down-sampling.
  - Eliminate fully connected layers on top of convolutional features.

- Use batch normalization in both generator (all layers except output layer) and discriminator (all layers except input layer). This helps to resolve training problems that arise due to poor initialization and helps the gradient flow in deeper models.
- Use ReLU activation in generator for all layers except the output, which uses Tanh to break the linearity and keep the values between 1 and −1, Use LeakyReLU activation in the discriminator for all layers.

Here are some of the parameters that are used for DCGAN in our research:

- Minibatch with minibatch size of 64.
- Generated image with imageSize of 64.
- Slope of leak = 0.1 for LeakyReLU.
- Learning rate = 0.0001, $\beta 1 = 0.3$
- Weights initialized with 0 centered normal distribution with standard deviation = 0.01

4.2 Generative Multi-adversarial Networks

We propose an improvement to Generative Multi-Adversarial Networks (GMANs) [3] by providing a modified formulation. GMANs differ from the traditional GANs in that multiple Discriminators are used with the Generator to improve the output of the system [3]. In GMANs there are four important components: (1) Maximizing $V(D, G)$, where D and G are the discriminator and Generator respectively, (2) Boosting, (3) Soft-Discriminator and (4) Automating Regulation. This architecture improves the output of the system and reduces the turnaround time and throughput of the system. In GMANs, Boosting of N weak discriminators is done to enhance the output of the system. The Soft-Discriminators provide realistic samples so that the generator receives a positive feedback from the discriminator [3]. The Use of Automating Regulation allows the generator to reduce the performance of the discriminator when necessary and also encourages the generator to challenge itself to be more efficient towards more accurate adversaries. Comparing to other models, GMANs are able to generate better quality images and it can be trained on a wide variety of datasets when using restricted domains of images. But the proposed methodology [3] has some problems - the booster does not generate comparative results and Generative Adversarial Metric (GAM) type of evaluation does not consider the performance of intermediate discriminators.

To address the above issue, we consider two approaches for training of discriminator in GMAN as suggested in [3] and made improvement to the original model structure: (a) a larger discriminating D using AdaBoost (for the better approximation of $\max_D V(D, G)$ (b) a better formulation of GAM type metric.

We studied a multiple-discriminators variant [3], which attempts to do a better approximation on $\max_D V(D, G)$ by providing a tough and strong critic to the generator, so that the generator has an opportunity to do a better evaluations on the reused images.

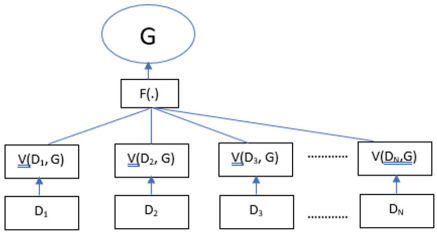


Fig. 1. (GMAN) The generator is trained using feedback from multiple discriminators and the image is being reused by the generator.

Figure 1 shows the basic system structure of GMAN. As we mentioned before, we use DCGAN as our basic GAN structure. Thus, deep convolution networks for Discriminators and Generator are used instead of the normal neural networks. Initially, from the noise signal a random image is generated which is given to the discriminators along with the images from the dataset for evaluation. The discriminators provide feedback to the generator to improve results. In the second cycle, instead of generating a new set of noisy images, the generator uses the image created in the initial step and tries to improve the created images which are then provided again to the discriminators.

Boosting Using AdaBoost. The boosted discriminator takes the maximum value from the N discriminators. In this process the discriminator is given a sample to predict whether the sample comes from the generator or the dataset [3]. The booster takes the reference of N weaker discriminator predictions and then makes a prediction of its own to evaluate the dataset. In our experiment we trained the weak discriminators using boosting and then for optimality we used *AdaBoost.OL* [23]. To get the comparative images for the boosted discriminators, the generator needs to have a mechanism to automatically decrease the performance of the discriminator whenever necessary. This is because there can be problems keeping both the discriminator and generator in the balance, e.g., unstable dynamics, oscillatory behavior and generator collapse [3]. Although the generator dampens the discriminator's performance, the dampening is restricted so that the generator still drives itself towards more accurate adversaries [3]. As described in [3], we use the following equation:

$$\min_{G,\lambda>0} F_G(V_i) - l(\lambda) \tag{3}$$

Here $l(\lambda)$ is monotonically increasing in λ. In our experiments, $l(\lambda) = k\lambda$ where k is a constant which in our case is initially set to 0.005. In order to compete against the best available adversary produced by the boosted algorithm and increase λ at the same time, the generator is given a relaxation to reduce its objectives [3]. In our work, the boosting produced quite impressive results on the image generation tasks.

GMAM Metric. The authors in [3,17] introduced the Generative Adversarial Metric (GAM) for the comparison of independently trained GAN models by allowing a pairwise comparison between them. They believed that the generator and discriminator pairs (G_1, D_1) and (G_2, D_2) system should be capable to understand and learn the relative performance by evaluating the generator under the reference of the opponent's discriminator. However, there are two issues: (a) No evaluation of intermediate results derived from the GMAM metric. (b) $V \leq 0$ does not always hold for all evaluations. We suggest to modify the formulation of the Generative Multi-Adversarial Metric (GMAM). Our modified metric is adaptable for multiple discriminator training and can overcome some existing shortcomings.

$$GMAM = \log\left(\frac{F^b_{G_a}(V^b_i)}{F^a_{G_b}(V^b_i)} \Big/ \frac{F^b_{G_b}(V^a_i)}{F^a_{G_a}(V^b_i)}\right) \tag{4}$$

where a and b represent the two different variants of GMAN. The idea behind this method is that if generator 1 performs better than generator 2 with respect to both discriminator 1 and 2, then GMAM ≥ 0.5 and ≤ 1. If generator 1 performs better in both cases, then GMAN ≥ 0 and ≤ 0.5. This way, intermediate cases are taken into consideration.

5 Experimental Setup

The experiments for GMAN are performed under similar architecture as DCGAN [1]. For G a convolutional transpose layer and for D strided convolutions are being used with the exception for the input of G and the final layer of D. In this experimentation single step gradient method [20] is being used and for each of the generated layers we used the batch normalization. The training on various discriminators were performed with different dropout rates ranging between [0.2, 0.8]. We used two different ways to effect variations in the discriminators. First, the architecture was changed by changing the number of filters in the discriminator layers which was reduced by the factors of 2, 4 and so on, and at the same time changing the dropout rates. Second, the training samples of the discriminator were decorrelated by splitting the minibatch across the discriminators. The code has been written in Pytorch and run on Nvidia GTX 1060 GPUs. The details about MNIST architecture and training process are as follows:

- The architecture of Generator transpose layers: $(4, 4, 128), (8, 8, 64), (16, 16, 32), (32, 32, 1)$.
- The architecture of Discriminator: $(32, 32, 1), (16, 16, 32), (8, 8, 64), (4, 4, 128)$..
- Slope of leak = 0.1 for LeakyReLU.
- Learning rate = 0.0001, $\beta 1 = 0.3$
- Weights initialized with 0 centered normal distribution with standard deviation = 0.01

- In the variants tests are performed by removing either convolution 3 $(4, 4, 128)$ or by dividing all the filter sizes by 2 or 4 that is $(32, 32, 1)$, $(16, 16, 16)$, $(8, 8, 32)$, $(4, 4, 64)$ or $(32, 32, 1)$, $(16, 16, 8)$, $(8, 8, 16)$, $(4, 4, 32)$.
- ReLu activation, Tanh activation and Sigmoid activation is taken respectively for all the hidden units, output units of the generator and output of the discriminator.
- The training of MNIST was performed for 25 epochs with a minibatch of size 64.
- CIFAR dataset was trained over 25000 iteration with a minibatch of size 64.

6 Results

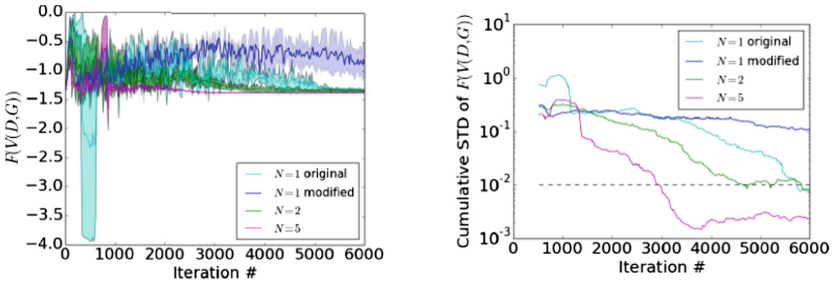
Note that in the original work [2], the authors reported that log likelihood estimates from Gaussian Parzen windows does not perform properly in high dimensions and has high variance. To evaluate the performance of our new GMAN structure in addressing this issue, we conducted two experiments on the MNIST [10] and CIFAR-10 [11] datasets. Tables 1 and 2 compare the performance of our approach with the standard GMAN (Figs. 2, 3, 4, 5 and 6).

Table 1. The selection of the models on MNIST with standard deviation for pairwise GMAM. The positive values for GMAN for each column represent the better performance, similarly the degraded performance is represented by negative values. These scores are obtained by doing the summation of each variant's column.

Score [3]	Score (ours)	Variant	GMAN*	GMAN-0	GMAN-max	Mod-GAN
0.127	**0.124**	GMAN*	-	-0.019 ± 0.008	−0.027 ± 0.018	−0.078 ± 0.035
0.007	**0.006**	GMAN-0	0.019 ± 0.008	-	−0.01 ± 0.014	−0.015 ± 0.026
−0.03	**−0.027**	GMAN-max	0.027 ± 0.018	0.01 ± 0.014	-	−0.01 ± 0.023
−0.122	−0.13	Mod-GAN	0.078 ± 0.035	0.015 ± 0.026	0.01 ± 0.023	-

Table 2. Results for 5 iterations on MNIST for the pairwise evaluation of GMAN/stddev (GMAN) considered for GMAN -λ and GMAN* (λ*)

Score [3]	Score (ours)	λ	$\lambda*$	$\lambda = 1$	$\lambda = 0$
0.028	**0.024**	$\lambda*$	-	−0.007/±0.008	−0.018/±0.009
0.001	**0.0005**	$\lambda = 1$	0.007/± 0.008	-	−0.007/±0.009
−0.025	**−0.020**	$\lambda = 0$	0.018/±0.009	0.007/±0.009	-



(a) Stdev over runs (b) Stdev over time

Fig. 2. Plot (a) shows the plot for *stdev* over runs. It shows $V(D, G)$ average for iterations on MNIST. It can be seen with the increase in number of discriminators the acceleration of $V(D, G)$ converges to a steady state (solid line) and the variance is reduced to σ^2 (filled shadow $\pm\sigma$). Part (b) of the figure provides an alternative proof of $GMAN^*$'s accelerated convergence. Plot (b) shows the plot for *stdev* over time. In this plot *stdev*, σ, of the generator objective over a sliding window of 500 iterations is taken. It can be seen that lower values shows a more steady-state. $GMAN^*$ achieves a steady-state with $N = 5$ at a speed equivalent of 2.2x in comparison to the speed of GAN for $(N = 1)$.

Fig. 3. Training result on MNIST dataset

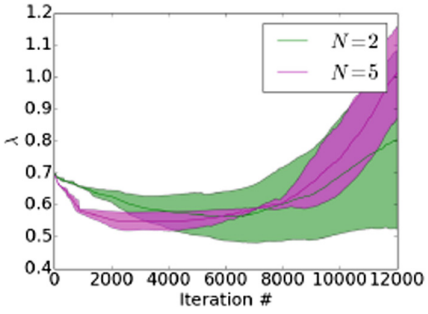


Fig. 4. The difficulty of the game is regulated by the adjustment of λ for $GMAN^*$. At first, G tends to reduce the value of λ to ease out the learning process and then continuously increases the value of λ to make the environment more challenging for the learning process.

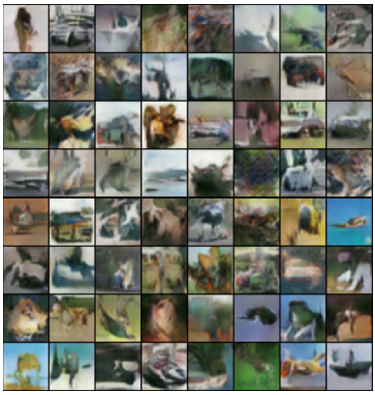
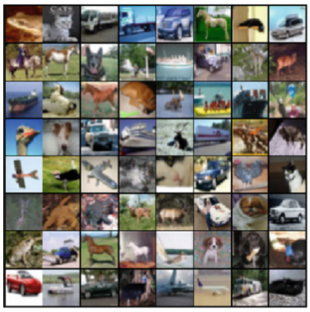
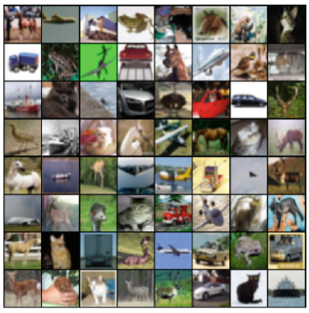


Fig. 5. Training result on CIFAR-10 dataset on DCGAN.



(a) Cifar-10 on GMAN



(b) Real data sample

Fig. 6. Generated images trained on Cifar-10 dataset for GMAN-variant

7 Conclusion

In this paper we propose to use a new formulation of GAM type metric for Generative Multi-Adversarial Network and further explored many roles and methods for the implementation of discriminators to find the optimal results. The implementation of boosting with automatic tunning of the generator for the generation of images totally outperformed the GANs with only one discriminator on MNIST, faster convergence was achieved without any milkiness or blurriness at a stable state on various processes with respect to the measurement performed by a GAM-type metric (GMAM) at variation of GMAN.

8 Future Work

The experiment results shown in this paper demonstrate the potential of GMAN method. In future work, considering the difficulty in training generator against discriminator, it is natural to choose multiple generator as new working direction.

Moreover, we will look into more sophisticated models like conditional GANs and GANs using different loss evaluation functions; we will also test our method on other dataset and provide detailed analysis of the performance of this method.

References

1. Radford, A., Metz, L., Chintala, S.: Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434 (2015)
2. Goodfellow, I., et al.: Generative adversarial nets. In: Advances in Neural Information Processing Systems, pp. 2672–2680 (2014)
3. Durugkar, I., Gemp, I., Mahadevan, S.: Generative multi-adversarial networks. arXiv preprint arXiv:1611.01673 (2016)
4. Krizhevsky, A., Nair, V., Hinton, G.: The CIFAR-10 dataset (2014). http://www.cs.toronto.edu/kriz/cifar.html
5. Arora, S., Ge, R., Liang, Y., Ma, T., Zhang, Y.: Generalization and equilibrium in generative adversarial nets (GANs). arXiv preprint arXiv:1703.00573 (2017)
6. Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., Chen, X.: Improved techniques for training GANs. In: Advances in Neural Information Processing Systems, pp. 2234–2242 (2016)
7. Lucic, M., Kurach, K., Michalski, M., Gelly, S., Bousquet, O.: Are GANs created equal. A Large-Scale Study. ArXiv e-prints (2017)
8. Wu, Y., Burda, Y., Salakhutdinov, R., Grosse, R.: On the quantitative analysis of decoder-based generative models. arXiv preprint arXiv:1611.04273 (2016)
9. Theis, L., Oord, A., Bethge, M.: A note on the evaluation of generative models. arXiv preprint arXiv:1511.01844 (2015)
10. LeCun, Y.: The MNIST database of handwritten digits (1998). http://yann.lecun.com/exdb/mnist/
11. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images (2009)
12. Yoo, D., Kim, N., Park, S., Paek, A.S., Kweon, I.S.: Pixel-level domain transfer. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9912, pp. 517–532. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46484-8_31
13. Ho, J., Ermon, S.: Generative adversarial imitation learning. In: Advances in Neural Information Processing Systems, pp. 4565–4573 (2016)
14. Edwards, H., Storkey, A.: Censoring representations with an adversary. arXiv preprint arXiv:1511.05897 (2015)
15. Donahue, J., Krähenbühl, P., Darrell, T.: Adversarial feature learning. arXiv preprint arXiv:1605.09782 (2016)
16. Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., Abbeel, P.: InfoGAN: interpretable representation learning by information maximizing generative adversarial nets. In: Advances in Neural Information Processing Systems, pp. 2172–2180 (2016)
17. Im, D.J., Kim, C.D., Jiang, H., Memisevic, R.: Generating images with recurrent adversarial networks. arXiv preprint arXiv:1602.05110 (2016)
18. Li, Y., Swersky, K., Zemel, R.: Generative moment matching networks. In: International Conference on Machine Learning, pp. 1718–1727 (2015)
19. Zhao, J., Mathieu, M., LeCun, Y.: Energy-based generative adversarial network. arXiv preprint arXiv:1609.03126 (2016)

20. Nowozin, S., Cseke, B., Tomioka, R.: f-GAN: training generative neural samplers using variational divergence minimization. In: Advances in Neural Information Processing Systems, pp. 271–279 (2016)
21. Uehara, M., Sato, I., Suzuki, M., Nakayama, K., Matsuo, Y.: Generative adversarial nets from a density ratio estimation perspective. arXiv preprint arXiv:1610.02920 (2016)
22. Arjovsky, M., Chintala, S., Bottou, L.: Wasserstein GAN. arXiv preprint arXiv:1701.07875 (2007)
23. Beygelzimer, A., Kale, S., Luo, H.: Optimal and adaptive algorithms for online boosting. In: International Conference on Machine Learning, pp. 2323–2331 (2015)