GANs Powered by Autoencoding — A Theoretic Reasoning

Zhifei Zhang [1] **Yang Song** [1] **Hairong Qi** [1]

Abstract

Many recent works integrate GANs with autoencoding (AE) networks to generate photo-realistic images conditioned on input images. It has been observed empirically that the addition of the AE network, besides generating images under certain condition, helps alleviate the mode missing and instability problems during training. The main objective of this work is not to propose new solutions, but rather we take the first attempt in providing theoretical reasoning on why the AE-based GAN structure is able to remedy the mode missing and instability issues. We further show that by adding an adaptive decay variable to the adversarial error, the instability issue caused by competition between the generator and discriminator is largely alleviated. Experimental results on compositional digits, natural images, and faces all show superior performance of autoencoding-based GAN in handling the mode missing and instability problems.

1. Introduction

Despite the great potential, the vanilla (non-conditional) GANs are still generally considered very difficult to train with undamped oscillations often occurring during the convergence process; and on top of that, they suffer from the so-called mode missing problem, where large volumes of probability mass tend to collapse onto a few modes (Che et al., 2016). In recent years, multiple works have been proposed to tackle the problem of mode missing and instability during training. Generally speaking, these methods can be divided into two categories: 1) strategy-based methods and 2) autoencoding-based methods.

The strategy-based methods preserve the basic structure of GANs but improve the objective or the way how the generator and/or discriminator is updated. For example, (Arjovsky & Bottou, 2016) suggested to add continuous noise to the input of discriminator, which could fix the instability and vanishing gradients issues. (Metz et al., 2016) unrolled the iterative updating procedure of GANs and approximated the discriminator toward optimal when updating the generator. (Durugkar et al., 2016) extended GANs to multiple discriminators and trained the generator against the best available discriminator. In this process, they also tried to obtain more optimal discriminator while updating the generator. (Salimans et al., 2016) introduced an extra layer into the discriminator to coordinate all samples with a mini batch, instead of considering those samples as independent like in the original GANs. This method avoided the collapse of the generator. Recently, LSGAN (Mao et al., 2016) and WGAN (Arjovsky et al., 2017) modified the original objective of GAN to improve stability.

In autoencoding-based methods, (Salimans et al., 2016) proposed feature matching that utilized an intermediate layer of the discriminator (behaves like an encoder) to minimize the feature distance between the real and generated data. (Larsen et al., 2016) combined GAN with VAE (Kingma & Welling, 2014), thus utilizing reconstruction error between the real and generated data to regulate the training. Similarly, (Donahue et al., 2016; Che et al., 2016) incorporated the autoencoding structure into GANs to avoid mode missing as well as stabilize the training procedure. (Li et al., 2015) proposed an alternative way of learning a generative model by maximum mean discrepancy (MMD). All of these works share the similar idea — directly matching the statistics of generated data to that of the real data by introducing an autoencoder-like architecture. In the original GANs, however, the statistics of generated data is matched to that specified by a discriminator (indirect estimate of the real data distribution), which has been demonstrated to cause mode missing (Metz et al., 2016; Che et al., 2016).

Compared to the strategy-based methods, which has strong theoretical support as in (Arjovsky & Bottou, 2016; Arjovsky et al., 2017), the autoencoding-based solutions tend to be heuristic-driven that lack theoretic reasoning. Although, AE-based solutions are capable of solving the mode missing problem and have the potential of stabilizing the training of GANs, the validation is only done through empirical study.

[1]University of Tennessee, Knoxville, TN USA. Correspondence to: Zhifei Zhang <zzhang61@vols.utk.edu>.

ICML 2017 Workshop on Implicit Models.

The contribution of this paper is thus two-fold. First, we provide theoretic reasoning on why AE-based GANs can remedy the mode missing problem. Second, we propose an adaptive decay mechanism that effectively stabilizes the training of AE-based GANs. The effectiveness is validated through, again, theoretic reasoning. We emphasize that the contribution of the paper is not to propose the AE-based GANs, but rather, we aim for the reasoning of this structure in its effective handling of the mode missing problem and the instability problem.

The rest of the paper is organized as follows. Section 2 elaborates on the theoretical reasoning on why incorporating an AE to GAN could solve the mode missing problem. A simple but effective method of decaying the undamped oscillation as the generator approaching optimum is proposed in section 3 to further stabilize the training process. Experimental result are shown in section 4. Finally, section 5 concludes the paper.

2. Autoencoding Remedies Mode Missing

The mode missing problem is mainly caused by the insufficient punishment on the condition of $p_g(x) < p_x(x)$, where $p_g(x)$ indicates the probability that a sample x is generated by the generator, and $p_x(x)$ indicates the probability that a sample x appears in the real data. It indicates that a real sample is with lower probability to be generated. To solve this problem, an extra penalty could be added to emphasize the cost on mode missing. A possible solution is to minimize $KL(p_x\|p_g)$, which is an asymmetric measurement, punishing more on mode missing than on unreality (i.e., $p_g(x) > p_x(x)$). We will derive that minimizing $KL(p_x\|p_g)$ is equivalent to minimizing the reconstruction error in autoencoding. From the KL divergence,

$$\begin{aligned} KL(p_x\|p_g) = &\int_{\Omega_x} p_x(x) \log \frac{p_x(x)}{p_g(x)} dx \\ =&\mathbb{E}_{x\sim p_x}\left[\log p_x(x)\right] - \mathbb{E}_{x\sim p_x}\left[\log p_g(x)\right], \end{aligned}$$

where $\mathbb{E}_{x\sim p_x}\left[\log p_x(x)\right]$ could be considered as a constant because the unknown data distribution is fixed. We need to update G to maximize $\mathbb{E}_{x\sim p_x}\left[\log p_g(x)\right]$, which fits p_g to p_x without mode missing. However, we only have observations drawn from p_x and p_g instead of analytical expression, so direct comparison between p_x and p_g is intractable. An intuitive way of comparing two unknown distributions would then be the Monte Carlo method, which estimates distance of two arbitrary distributions by repeated random sampling. A direct measurement is maximum mean discrepancy (MMD), which measures the kernel-based distance of every sample pair within and between p_x and p_g. In other word, MMD blindly computes the distance of all sample pairs. Yet a more effective measurement is to compare the sample pairs assigned by the Hungarian method, where the average distance between non-repetitive pairs is minimized. Assume two large but finite sample sets $\mathbb{X}_x$ and $\mathbb{X}_g$, of the same size, n, randomly drawn from p_x and p_g, respectively. Suppose the Hungarian assignment function is $\mathcal{H} : \mathbb{X}_x \to \mathbb{X}_g$ based on a distance metric $\mathcal{L}(x, \mathcal{H}(x))$, $x \in \mathbb{X}_x$ and $\mathcal{H}(x) \in \mathbb{X}_g$, such that $\mathbb{E}_{x\in\mathbb{X}_x}[\mathcal{L}(x, \mathcal{H}(x))]$ is minimized. Then the distance between p_x and p_g can be measured by

$$\mathbb{E}_{x\in\mathbb{X}_x}[\mathcal{L}(x, \mathcal{H}(x))]. \tag{1}$$

Ideally, if $p_x = p_g$, Eq. 1 achieves its minimum. In practice, the training dataset could be considered as $\mathbb{X}_x$, and $\mathbb{X}_g$ consists of the generated samples. In GANs, $G(z)$ generates $\mathbb{X}_g$ from random sampling, $z \sim p_z$. However, in mini-batch learning, the size of $\mathbb{X}_x$ is limited (i.e., 50 or 100), which cannot sufficiently represent the true distribution of p_x. Such limited number of random samples would result in larger average distance from Eq. 1. Furthermore, the computational complexity of Hungarian matching is $O(n^3)$. Therefore, it will be more efficient if the Hungarian assignment function $\mathcal{H} : \mathbb{X}_x \to \mathbb{X}_g$ is learned through a network.

In many existing works, $\mathcal{H}(x)$ is interpreted as an encoding-decoding (AE) structure where the input-output pairs would result in minimum average distance, and $\mathcal{L}(x, \mathcal{H}(x))$ may be interpreted as the reconstruction error. For example, (Larsen et al., 2016; Che et al., 2016) concatenated an encoder to the generator, which together forms an AE that is equivalent to the functionality of $\mathcal{H}(x)$. (Salimans et al., 2016) reused the discriminator network as an encoder to compare higher level feature of the real and generated data, in which the discriminator also performed like $\mathcal{L}(x, \mathcal{H}(x))$. Although many related works have verified the effectiveness of incorporating AE to the GAN, the discussion is rather heuristic and empirical. Here, we provide theoretic reasoning of the effectiveness from the perspective of avoiding mode missing.

For the case of AE, $\mathcal{H}(x) = G(E(x))$, where $E(x) = z$ denotes an encoder. Rewriting Eq. 1, the equivalent tractable objective is

$$\mathbb{E}_{x\sim p_x}\left[\mathcal{L}(x, G(E(x)))\right]. \tag{2}$$

Adding Eq. 2 to the original formulation of GANs (Goodfellow et al., 2014), we obtain the new objective function that penalizes both unrealisticness and mode missing,

$$\mathbb{E}_{x\sim p_x}[\log\left(D(x)(1 - D(\mathcal{H}(x)))\right) + \lambda\mathcal{L}(x, \mathcal{H}(x))], \tag{3}$$

where λ balances the effect of reconstruction error, and $\mathcal{H}(x) = G(E(x))$.

3. Adaptive Decay Stabilizes the Training

Compared to the objective of GANs, Eq. 3 appends an extra regularization $\mathbb{E}_{x\sim p_x}[\mathcal{L}(x, \mathcal{H}(x))]$, whose gradient with

respect to the generator is non-zero if we do not consider overfitting. Therefore, the problem of gradient vanishing (analyzed in supplementary) is naturally resolved. For the problem of instability, it tends to occur when G approaches optima, which makes $\|\mathbb{E}_{x\sim p_g}[D(x)-1]\|_2$ approach 0, resulting in gradient explosion. In addition, even though the discriminator and generator both achieve optimum, the loss of G is not zero, causing the undamped oscillation. To solve this problem, we introduce a variable $\alpha_{\mathcal{L}} \leq 1$ whose value will reduce as G approaches its optimum, i.e., $\mathbb{E}_{x\sim p_x}[\mathcal{L}(x,\mathcal{H}(x))]$ approaches zero. Then, Eq. 3 can be rewritten as

$$\mathbb{E}_{x\sim p_x}[\log(D(x)(1-\alpha_{\mathcal{L}}D(\mathcal{H}(x))))+\lambda\mathcal{L}(x,\mathcal{H}(x))],$$

where $\alpha_{\mathcal{L}} = \min\{\alpha\mathbb{E}_{x\sim p_x}[\mathcal{L}(x,\mathcal{H}(x))],1\}$, $\alpha \in [0,\infty)$. Intuitively, if $\alpha = 0$, the effect of discriminator is diminished, then the network functions as an AE. If α approaches infinity, it becomes a normal GAN+AE.

When updating the generator, $\alpha_{\mathcal{L}}$ is considered a constant computed before the update. Then, the gradient of generator with parameter θ is

$$\Delta\theta = \mathbb{E}_{x\sim p_x}\left[\frac{\alpha_{\mathcal{L}}\nabla_\theta D(\mathcal{H}(x))}{\alpha_{\mathcal{L}}D(\mathcal{H}(x))-1}+\lambda\nabla_\theta\mathcal{L}(x,\mathcal{H}(x))\right].$$

For differentiable D and G, $\nabla_\theta D(\mathcal{H}(x)) = C_1$ is finite, and $D(\mathcal{H}(x)) = C_2 \in [0,1]$. Although the extreme case $D(\mathcal{H}(x)) = 1$ yields infinite $\Delta\theta$, an $\alpha_{\mathcal{L}} < 1$ could suppress the unstable update driven by the discriminator.

The more general instability scenario is the undamped oscillation when both D and G approach their optima, then $D(x) \approx D(\mathcal{H}(x)) \approx 1/2$. In addition, $\nabla_\theta\mathcal{L}(x,\mathcal{H}(x)) = \epsilon$, which is a small value. We thus have

$$\Delta\theta|_{\theta=\theta^*} = \mathbb{E}_{x\sim p_x}\left[\frac{\alpha_{\mathcal{L}}C_1}{\alpha_{\mathcal{L}}C_2-1}+\epsilon\right]. \tag{4}$$

In vanilla GAN or GAN+AE ($\alpha_{\mathcal{L}} = 1$), $\Delta\theta \neq 0$, although $p_x = p_g$. Then, the training process will arrive at the Nash equilibrium and present undamped oscillation on $\Delta\theta$. On the contrary, by incorporating $\alpha_{\mathcal{L}}$, which may decrease exponentially with $\mathcal{L}(x,\mathcal{H}(x))$, the gradient flows from the discriminator will be significantly suppressed towards zero.

For the update of discriminator with parameter ϕ,

$$\Delta\phi = \mathbb{E}_{x\sim p_x}\left[\frac{\nabla_\phi D(x)}{D(x)}+\frac{\alpha_{\mathcal{L}}\nabla_\phi D(\mathcal{H}(x))}{\alpha_{\mathcal{L}}D(\mathcal{H}(x))-1}\right], \tag{5}$$

which approaches infinity if $D(x)|_{x\sim p_x} \to 0$. This would effectively discourage the strategy of training the generator for a long time without updating the discriminator. To ensure symmetric objective, let $\alpha_{\mathcal{L}} = 1$ ($\alpha \to \infty$) when updating the discriminator.

4. Experimental Evaluation

We claim that incorporating AE is a more appealing way to stabilize the training of GANs and avoid mode missing. Table 1 displays the methods in comparison.

Table 1. Notation of methods in comparison

Notation	Method
GAN	Vanilla GAN (Goodfellow et al., 2014)
LSGAN	Least square GAN (Mao et al., 2016)
GMMN	Minimize MMD (Li et al., 2015)
GAN+MMD	Incorporate MMD as loss function
GAN+MB	Mini-batch (Salimans et al., 2016)
GAN+UR	Unrolled GAN (Metz et al., 2016)
GAN+AE	Incorporate AE to GAN

For fair comparison, we implement each method with the same architecture of convolution and deconvolution networks (detailed in the supplementary). Note that the network is not delicately designed to achieve the best performance, since the ultimate goal is to demonstrate the stability by adding AE to GAN. The parameter $\lambda = 100$ (Eq. 3) in all experiments. We first compare the convergence speed on MNIST (LeCun et al., 1998), then estimate the number of missing modes of each method on the compositional MNIST with 1000 modes. In addition, CIFAR10 and three face datasets (i.e., Morph (Ricanek & Tesafaye, 2006) and FGNET (Lanitis et al., 2002), CelebA (Liu et al., 2015)) are used to visualize the performance on generating natural images. Finally, we verify the proposed training strategy in decaying the undamped oscillation problem.

4.1. MNIST

We apply all the methods in Table 1 on the MNIST dataset, and the interpolation results between two random digits at epochs 1 and 10 are shown in Fig. 1. Obviously, GMMN cannot generate clear digits, GAN+UR converges very slowly, and GAN+AE presents the best image quality visually and quantitatively with the highest inception score.

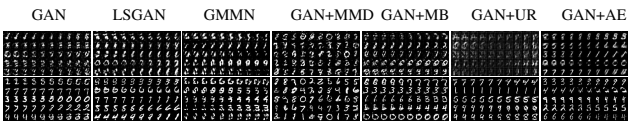

Figure 1. Comparison of different methods on MNIST dataset. Top: 1 epoch. Bottom: 10 epochs.

4.2. Compositional MNIST

The compositional MNIST dataset is constructed by randomly picking three digits from MNIST and form a three-digit number from 000 to 999 by stitching them horizon-

Table 2. Number of missed modes on the compositional MNIST

Epoch	GAN	LSGAN	GMMN	GAN+MMD	GAN+MB	GAN+UR	GAN+AE
20	154	142	409	301	516	136	**52**
50	82	69	408	76	79	41	**39**

tally. Fig. 2 shows the results of each method at epoch 20. Again, GAN+AE presents more appealing results.

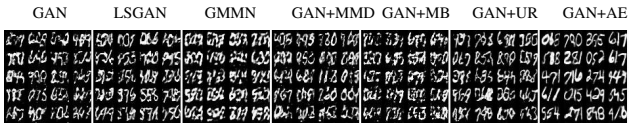

Figure 2. Comparison of different methods on the compositional MNIST dataset with 1000 modes. The samples are obtained at epoch 20.

To count the number of missed modes, we employ the method inspired by (Che et al., 2016), i.e., testing the optimal discriminator by real data. If the expectation of discriminator outputs from certain mode is close to 1 (e.g., larger than 0.8), the corresponding input mode is considered as missed mode. From Table 2, we observe that GAN+AE converges the fastest. After 50 epochs, GAN+MB, GAN+UR, and GAN+AE all perform similar. However, GMMN gains nothing from more training iterations. GAN+MMD eventually outperforms both GAN and GMMN. In this experiment, GAN+AE shows superior performance in aspects of convergence rate and the capability in avoiding missing mode.

4.3. Natural Images and Faces

In the experiment, we compare all the methods on natural images and faces. The results are shown in Fig. 3. Generally speaking, the methods without AE, i.e., GAN, GMMN, GAN+MB, and GAN+UR, tend to generate images looking more like mixture from multiple objects, missing the real models especially when learning natural images. Compared to AE which generates blurry images, GAN+AE preserves more texture and prevents the mixing effect, while avoiding mode missing. Similarly, GANs without AE yields distorted faces as compared to GAN+AE.

On the CIFAR-10 dataset, the generated images from GAN, LSGAN, and GAN+UR look more reasonable. However, they are not realistic images, rather, they look like mixtures formed from multiple objects or backgrounds. Similarly, GMMN cannot compete with other methods. GAN+MB shows unstable output for this dataset. GAN+AE shows more noise (or details) as compared to GAN, LSGAN, and GAN+UR, but it prevents the mixing issue. We also provide the result of AE, which yields blurry images but there is no mixing effect. This advantage of GAN+AE effectively facilitates the generation of more discriminative images. Same effect can be observed from the results on the face dataset.

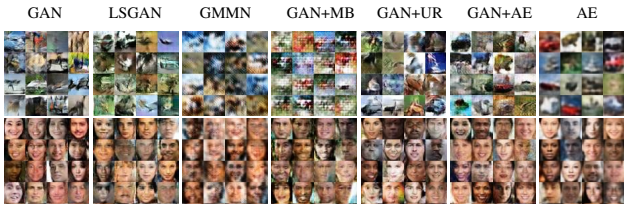

Figure 3. Comparison of different methods on CIFAR-10 (top) and face (bottom) datasets.

4.4. Instability Evaluation

We claim that the variable $\alpha_{\mathcal{L}}$ can alleviate the undamped oscillation problem caused by competition between the generator and discriminator. In this experiment, we set $\alpha = 0.1$ and compare GAN+AE and GAN+AE+$\alpha_{\mathcal{L}}$ on the face dataset. Fig. 4 shows the comparison results. We clearly observe the oscillation is drastically suppressed by adding the variable $\alpha_{\mathcal{L}}$. And as a result, the generated images are more photo-realistic and sharp.

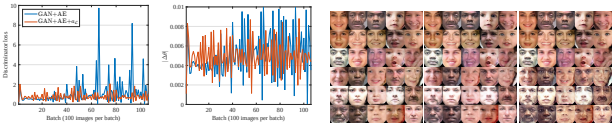

Figure 4. Left: comparison on discriminator loss (average output of the discriminator). Middle: absolute gradient from the discriminator. Right: generated images after 10, 50, and 100 batches from GAN+AE (top) and AE+GAN+$\alpha_{\mathcal{L}}$ (bottom) based on a pre-trained GAN+AE model, which can generate realistic faces.

5. Conclusion

In this paper, we made the initial attempt to theoretically reasoning the effectiveness of autoencoding-based GANs. We showed that with the incorporation of an AE, the reconstruction error serves the purpose of penalizing the missing modes. Also, we showed that by adding a simple decay variable in the objective function, the vanishing gradient, gradient explosion, and the undamped oscillation problems can be effectively resolved.

References

Arjovsky, M. and Bottou, L. Towards principled methods for training generative adversarial networks. In *NIPS 2016 Workshop on Adversarial Training. In review for ICLR*, 2016.

Arjovsky, Martin, Chintala, Soumith, and Bottou, Léon. Wasserstein GAN. *arXiv preprint arXiv:1701.07875*, 2017.

Che, T., Li, Y., Jacob, A. P., Bengio, Y., and Li, W. Mode regularized generative adversarial networks. *arXiv preprint arXiv:1612.02136*, 2016.

Donahue, J., Krähenbühl, P., and Darrell, T. Adversarial feature learning. *arXiv preprint arXiv:1605.09782*, 2016.

Durugkar, I., Gemp, I., and Mahadevan, S. Generative multi-adversarial networks. *arXiv preprint arXiv:1611.01673*, 2016.

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

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. In *The International Conference on Learning Representations (ICLR)*, 2014.

Lanitis, Andreas, Taylor, Christopher J., and Cootes, Timothy F. Toward automatic simulation of aging effects on face images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(4):442–455, 2002.

Larsen, A. B. L., Sønderby, S. K., and Winther, O. Autoencoding beyond pixels using a learned similarity metric. *Proceedings of The 33rd International Conference on Machine Learning (ICML)*, 2016.

LeCun, Yann, Bottou, Léon, Bengio, Yoshua, and Haffner, Patrick. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Li, Y., Swersky, K., and Zemel, R. Generative moment matching networks. In *International Conference on Machine Learning (ICML)*, pp. 1718–1727, 2015.

Liu, Ziwei, Luo, Ping, Wang, Xiaogang, and Tang, Xiaoou. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, 12 2015.

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

Mao, Xudong, Li, Qing, Xie, Haoran, Lau, Raymond YK, and Wang, Zhen. Least squares generative adversarial networks. *arXiv preprint ArXiv:1611.04076*, 2016.

Metz, L., Poole, B., Pfau, D., and Sohl-Dickstein, J. Unrolled generative adversarial networks. *arXiv preprint arXiv:1611.02163*, 2016.

Ricanek, Karl and Tesafaye, Tamirat. Morph: A longitudinal image database of normal adult age-progression. In *7th International Conference on Automatic Face and Gesture Recognition (FGR06)*, pp. 341–345. IEEE, 2006.

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. Improved techniques for training GANs. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 2226–2234, 2016.

Appendix

A. Instability

Besides the mode missing problem, GANs also suffer from the instability problem. Here, we discuss and tackle one component of the general instability issue of GANs, i.e., the unstable update of the generator during training, namely, vanishing or infinite gradient, and undamped oscillation as GANs approach the Nash equilibrium. Fixing the discriminator $D(x)$ which is differentiable, the gradient of generator $G(z)$ with respect to its parameter θ is

$$\begin{aligned}\nabla_\theta \mathcal{L}_G =& \nabla_\theta \mathbb{E}_{z\sim p_z}\left[\log(1-D(G_\theta(z)))\right] \\ =& \mathbb{E}_{z\sim p_z}\left[\frac{1}{D(G_\theta(z))-1}\frac{\partial D(G_\theta(z))}{\partial G_\theta(z)}\frac{\partial G_\theta(z)}{\partial \theta}\right] \\ =& \mathbb{E}_{x\sim p_g}\left[\frac{\nabla_x D(x)\nabla_\theta G_\theta}{D(x)-1}\right],\end{aligned}$$

where $\|\mathbb{E}_{x\sim p_g}[\nabla_\theta G_\theta]\|_2$ is bounded for a differentiable generator. If D is a perfect discriminator $D^\dagger$, $D^\dagger(x)|_{x\sim p_x}=1$ and $D^\dagger(x)|_{x\sim p_g}=0$. Note that a perfect discriminator is an optimal discriminator, but an optimal discriminator is not necessarily a perfect one. When D is approaching $D^\dagger$ and $x\sim p_g$, $\lim_{D\to D^\dagger} D(x)=0$, and $\lim_{D\to D^\dagger}\nabla_x D(x)=0$. Therefore,

$$\lim_{D\to D^\dagger}\mathbb{E}_{x\sim p_g}\left[\frac{\nabla_x D(x)\nabla_\theta G_\theta}{D(x)-1}\right]=0, \tag{6}$$

which corresponds to the condition of *gradient vanishing*, where the generator is trapped into a bad state without update. On the contrary, if the generator achieves optimum G^* given $D^\dagger$, $D^\dagger(G^*(z))=1$. Then, $\lim_{G\to G^*} D(x)=1$, and $\|\mathbb{E}_{x\sim p_g}[\nabla_x D(x)]\|_2>0$[1]. Thus,

$$\lim_{G\to G^*}\mathbb{E}_{x\sim p_g}\left[\frac{\nabla_x D(x)\nabla_\theta G_\theta}{D(x)-1}\right]=\infty, \tag{7}$$

which indicates the phenomenon of *gradient explosion* when G is approaching the optimum[2]. In practice, D and G are updated alternatively based on mini-batch learning. Therefore, the discriminator is more likely optimal rather than perfect. For a mini-batch, however, the discriminator may be locally perfect, then the behavior of instability would deteriorate the generator. Therefore, starting to train the generator after training a perfect discriminator causes gradient explosion, which partly explains why alternative update of D and G is suggested in (Goodfellow et al., 2014). Now, assume $D=D^*$ and $G=G^*$ after epochs of training, then $p_g=p_x$ and $D(x)=1/2$. Under this condition, we expect $\nabla_\theta\mathcal{L}_G=0$. However, $\mathbb{E}_{x\sim p_g}[D(x)]=1/2$ and $\|\mathbb{E}_{x\sim p_g}[\nabla_x D(x)]\|_2>0$. Therefore, $\nabla_\theta\mathcal{L}_G\neq 0$ that forces the generator to continuously being updated towards $D(x)|_{x\sim p_g}=1$. Simultaneously, the discriminator tries to make $D(x)|_{x\sim p_g}=0$. In this game, $\mathbb{E}_{x\sim p_g}[D(x)]$ will present undamped oscillation.

[1] During training, $D(x)$ is updated under two conditions; either $D(x)>0$ for $x\sim p_g$ or $D(x)<1$ for $x\sim p_x$. In either case, $\nabla_x D(x)$ needs to be non-zero to update D.

[2] The objective is to find θ that makes $D(G_\theta)=1$, but not to optimize G_θ. Therefore, G_θ doesnt have to be local optimum, i.e., $\nabla_\theta G_\theta=0$. In fact, it is extremely rare case that G_θ is the local optimum.

B. Effect of Adding AE and $\alpha_{\mathcal{L}}$

The effect of incorporating AE to GAN is illustrated in Fig. 5, and Fig. 6 demonstrates the effect of $\alpha_{\mathcal{L}}$.

C. The Training strategy of GAN+AE+$\alpha_{\mathcal{L}}$

We present the training strategy for the proposed GAN+AE+$\alpha_{\mathcal{L}}$ in Algorithm 1. With the auxiliary of AE (i.e., reconstruction loss), the generator can fit p_g to p_x without the participation of the discriminator. In addition, the training of AE is well-known to be relatively stable compared to that of GANs. Therefore, we first pre-train AE (i.e., encoder E and generator G) only by the reconstruction loss. After G is well initialized, the gradient from the discriminator D is incorporated. Such two-step training strategy could effectively enhance the stability of the training process. On the contrary, updating G by both reconstruction and adversarial losses from the very beginning, where the kernel parameters are randomly initialized, may trap into local optimal, bad equilibrium, or saturation area because of the activation functions, e.g., sigmoid and tanh. The training strategy is detailed in Algorithm 1.

D. Network Structure

For fair comparison, we implement each method with the same architecture of convolution and deconvolution networks as given in Table 3 We aim to evaluate the performance of different methods. In order to illustrate differences caused by different methods rather than different setups, we implement each method by the same architecture and parameter setting as shown in Table. 3. In the training, the networks are updated through ADAM with the learning rate 0.0002, and the batch size is 100. The kernel size is 5×5. The parameter $\lambda=100$ in all experiments. We set the prior distribution of z as uniform distribution. An extra discriminator on z is adopted like in AAE (Makhzani et al., 2015) to achieve generative modeling. The reconstruction error is measured by ℓ_2-norm. Note that the network is not delicately designed to achieve the best performance, since the ultimate goal is to demonstrate the stability by adding AE to GAN compared to the other improved methods.

Note that the inception score of GAN+AE on the CIFAR-10 dataset (Table 4) ranks lower than those of GAN, LSGAN, and GAN+UR. In addition, all standard devi-

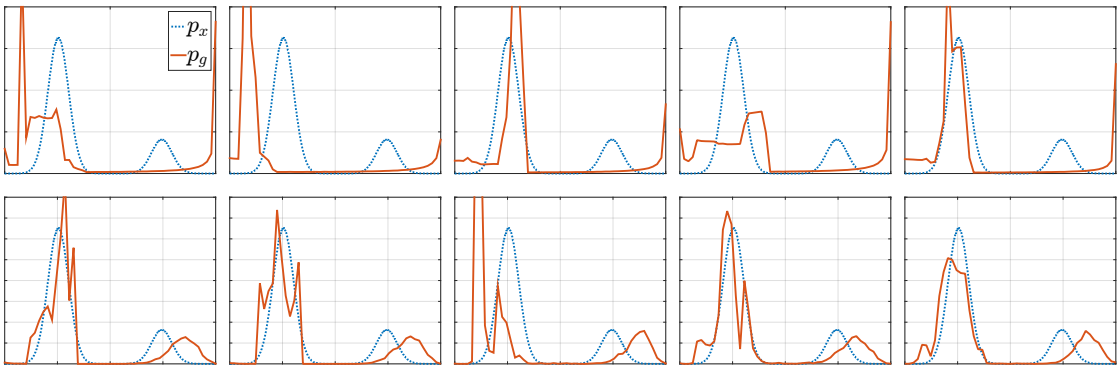

Figure 5. Comparison of GAN (top) and GAN + AE (bottom). From left to right: results after training by 200, 400, 600, 800, and 1000 batches. The real data distribution (blue dotted) is mixed Gaussian with two modes. The red curve shows the distribution of generated data.

Table 3. Details of convolution and deconvolution networks (a is the spatial dimension of images)

Conv	Size	Deconv	Size
Input (gray/color image)	$a \times a \times 1$ or 3	Input	50
Conv, ReLU, BN	$a/2 \times a/2 \times 64$	FC, ReLU, BN	1024
Conv, ReLU, BN	$a/4 \times a/4 \times 128$	FC, ReLU, BN, reshape	$a/4 \times a/4 \times 128$
reshape, FC, ReLU, BN	1024	Deconv, ReLU, BN	$a/2 \times a/2 \times 64$
FC, Sigmoid	50 or 1 (for D)	Deconv, Sigmoid	$a \times a \times 1$ or 3

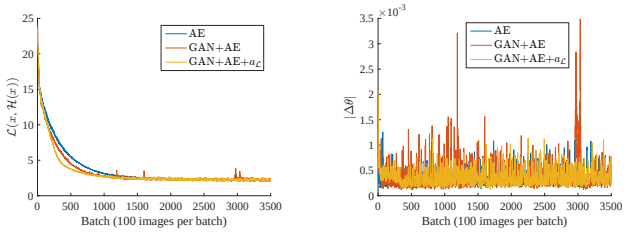

Figure 6. Comparison of AE ($\alpha_{\mathcal{L}} = 0$), GAN+AE ($\alpha_{\mathcal{L}} = 1$), and GAN+AE+$\alpha_{\mathcal{L}}$ (with dynamic $\alpha_{\mathcal{L}}$) on the MNIST dataset. They start with the same initial parameters. In GAN+AE+$\alpha_{\mathcal{L}}$, $\alpha_{\mathcal{L}} = 0.1$. AE converges slower than GAN+AE, while $\alpha_{\mathcal{L}}$ effectively boosts the convergence. In addition, $\alpha_{\mathcal{L}}$ stabilizes the update of generator (Right).

ation are significantly higher than those on MNIST and MNIST1000. This may be caused by the inherent design of inception score whose performance highly depends on that of the classifier. And we use a pre-trained model on ImageNet instead of a state-of-the-art model specifically trained on CIFAR-10.

Table 4. Inception scores of each method tested with 10,000 samples on three labeled datasets

Method	MNIST	MNIST1000	CIFAR10
GAN	$2.22 \pm .009$	$2.18 \pm .017$	$\mathbf{2.53 \pm 2.34}$
LSGAN	$2.26 \pm .008$	$2.17 \pm .019$	1.98 ± 2.01
GMMN	$2.00 \pm .014$	$2.05 \pm .028$	1.48 ± 1.44
GAN+MMD	$2.22 \pm .008$	$2.13 \pm .023$	1.37 ± 1.19
GAN+MB	$2.19 \pm .005$	$2.04 \pm .049$	1.41 ± 1.26
GAN+UR	$2.17 \pm .010$	$2.07 \pm .024$	2.39 ± 2.16
GAN+AE	$\mathbf{2.28 \pm .003}$	$\mathbf{2.29 \pm .028}$	1.89 ± 1.73

Algorithm 1 Training strategy GAN+AE+$\alpha_{\mathcal{L}}$

Input: the training dataset $\mathbb{X}_x$, weight of reconstruction loss λ, and decay rate α.
Initialize the encoder E, generator G, and discriminator D, which are parameterized by θ, ϕ, and φ, respectively.

Stage 1: pre-train E and G
repeat
  unrepeated samples $\{x_1, x_2, \cdots, x_m\} \subset \mathbb{X}_x$
  update the encoder E and generator G by SGD with gradient descent:

$$\Delta\theta, \Delta\varphi = \nabla_{\theta,\varphi} \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(x_i - G(E(x_i)))$$

  update the discriminator D by SGD with gradient ascent:

$$\Delta\phi = \nabla_{\phi} \frac{1}{m} \sum_{i=1}^{m} [\log(D(x_i)(1 - D(G(E(x_i)))))]$$

until $\frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(x_i, G(E(x_i))) < \frac{1}{\alpha}$

Stage 2: sharpen the generated data
repeat
  unrepeated samples $\{x_1, x_2, \cdots, x_m\} \subset \mathbb{X}_x$
  $\alpha_{\mathcal{L}} = \min \left\{ \frac{\alpha}{m} \sum_{i=1}^{m} \mathcal{L}(x_i, G(E(x_i))), 1 \right\}$
  update the encoder E and generator G by SGD with gradient descent:

$$\Delta\theta, \Delta\varphi = \nabla_{\theta,\varphi} \frac{\lambda}{m} \sum_{i=1}^{m} \mathcal{L}(x_i - G(E(x_i)))$$

  update the generator G by SGD with gradient descent:

$$\Delta\theta = \nabla_{\theta} \frac{1}{m} \sum_{i=1}^{m} [\log(1 - \alpha_{\mathcal{L}} D(G(E(x_i))) + \lambda \mathcal{L}(x_i - G(E(x_i)))]$$

  update the discriminator D by SGD with gradient ascend:

$$\Delta\phi = \nabla_{\phi} \frac{1}{m} \sum_{i=1}^{m} [\log(D(x_i)(1 - D(G(E(x_i)))))]$$

until terminating criterion is met