UNDERSTANDING AND STABILIZING GANS' TRAINING DYNAMICS WITH CONTROL THEORY

Anonymous authors
Paper under double-blind review

ABSTRACT

Generative adversarial networks (GANs) have made significant progress on realistic image generation but often suffer from instability during the training process. Most previous analyses mainly focus on the equilibrium that GANs achieve, whereas a gap exists between such theoretical analyses and practical implementations, where it is the training dynamics that plays a vital role in the convergence and stability of GANs. In this paper, we directly model the dynamics of GANs and adopt the control theory to understand and stabilize it. Specifically, we interpret the training process of various GANs as certain types of dynamics in a unified perspective of control theory which enables us to model the stability and convergence easily. Borrowed from control theory, we adopt the widely-used negative feedback control to stabilize the training dynamics, which can be considered as an $L2$ regularization on the output of the discriminator. We empirically verify our method on both synthetic data and natural image datasets. The results demonstrate that our method can stabilize the training dynamics as well as converge better than baselines.

1 INTRODUCTION

Generative adversarial networks (GANs) (Goodfellow et al., 2014) have shown promise in generating realistic natural images (Brock et al., 2018) and facilitating unsupervised and semi-supervised learning (Chen et al., 2016; Li et al., 2017; Donahue & Simonyan, 2019). In GANs, an implicit generator (G) is defined by mapping a noise distribution to the data space. Since no density function is defined for the implicit generator, a discriminator (D) is introduced to estimate the density ratio between the data distribution p_D and the generating distribution p_G by telling the real samples from fake ones. G aims to recover the data distribution by maximizing this ratio. This framework is formulated as a minimax optimization problem, which can be solved by optimizing G and D alternately. In practice, however, GANs can be extremely sensitive to hyperparameters (Lucic et al., 2018; Radford et al., 2015) and oscillations are often observed (Liang et al., 2018; Chavdarova & Fleuret, 2018), thereby suffering from the instability of training (Goodfellow, 2016).

There are some theoretical analyses aiming to understand and stabilize the training process of GANs, such as designing different objective functions using other statistical divergences (Nowozin et al., 2016; Nguyen et al., 2017; Du et al., 2018; Mao et al., 2017), and introducing auxiliary regularization terms (Gulrajani et al., 2017; Arjovsky et al., 2017). However, there is a gap between the theoretical analyses and practical implementations: most of previous work assumes that (1) D achieves optimal when training G (Goodfellow et al., 2014; Arjovsky et al., 2017; Gulrajani et al., 2017; Nowozin et al., 2016); and (2) the optimal D is a smooth function of G (Metz et al., 2016). These two assumptions are violated in most practical cases. First, GANs are optimized by alternating between k steps of optimizing D and one step of optimizing G, where k generally takes 1 or 5. It results in a sub-optimal D, especially in the early stage of the training process. Second, D is constant w.r.t. G when G is optimized, which results in missing gradients and an unstable training process (Metz et al., 2016).

Instead, to ensure practical convergence, it is the training dynamics of both G and D, i.e., how G (or D) changes given current G and D during the training process, that should be considered, where the previous two assumptions are no more required. There are some methods that focus on the dynamics of the parameters. Mescheder et al. (2017) propose to model the dynamics of the parameters using the vector field defined by its gradients. In such cases, the local convergence and stability are fully determined by the eigenvalues of the Jacobian matrix. Mescheder et al. (2018) further propose a regularization term to stabilize GANs by adjusting the eigenvalues of the Jacobian matrix. Gidel et al.

(2018) analyze the effect of momentum and propose to stabilize GANs using negative momentum. In these methods, the Jacobian matrix is the key to understand and stabilize the dynamics, whose calculation is, however, computationally expensive. It largely impedes us to generalize this analysis to complex models such as neural networks (LeCun et al., 2015).

In this paper, we argue that modeling the dynamics in the function space is more convenient for convergence analysis. Specifically, we directly model G and D as two dynamics whose output is considered as a signal of time t, i.e., representing the output of G and D as $G(t, z)$ and $D(t, x)$ respectively. The control theory (Kailath, 1980) provides a powerful tool to understand them and analyze their stability. Specifically, differential equations can be used to denote the dynamics, e.g., the dynamics of G can be denoted as $\frac{dG(t,z)}{dt} = f(G, D)$, which represents how G changes given current G and D. Then the stability and convergence can be easily modeled according the roots of the characteristic equation of the differential equation (Kailath, 1980) as introduced in Sec. 2.

Under the above perspective of control theory, we unify the dynamics of G and D as certain dynamics which are well-studied in control theory for various GANs, including Standard GAN (SGAN), WGAN and LSGAN (Mao et al., 2017), as illustrated in Sec. 3 and Appendix A. Specifically, the training process of GANs can be considered as a dynamic whose output is determined by its intrinsic properties (i.e., the objective function of G and D) and its input (i.e., the data distribution). In this perspective, the stability of GANs can be easily understood, and some previous methods (Mao et al., 2017; Mescheder et al., 2018; Gidel et al., 2018) can be interpreted as certain controlling methods which are commonly used in the control theory. Further, we propose to use the widely adopted controlling method, the negative feedback control (Åström & Hägglund, 1995), to stabilize GANs. This method acts as a regularization term which penalizes the $L2$ norm of the output of D as we described in Sec. 4. Our method is verified on the toy data such as Dirac GAN and Mixture of Gaussian, as well as the natural images such as CIFAR10 (Krizhevsky et al., 2009). The results demonstrate that our method can successfully stabilize the dynamics of GANs and provide better convergence results compared to baselines.

2 Preliminary

In this section, we provide the background and preliminary about Laplacian transformation (LT) and control theory. LT is a powerful tool to present an ordinary differential equation (ODE) as a rational fraction. Since most dynamics can be represented as an ODE, LT can largely simplify the analysis about stability by merely analyzing the properties of the rational fraction.

2.1 Laplacian Transformation

Laplacian transformation (LT) (Widder, 2015) can be considered as an extension of Fourier transformation (FT), which transforms a signal, i.e., a function of time, to a function of complex variables. Formally, the transformation and its inversion are given by:

$$\mathcal{F}(f)(s) = \int_0^{\infty} f(t)e^{-st}dt, \quad \mathcal{F}^{-1}(F)(t) = \frac{1}{2\pi i}\int_{c-\infty i}^{c+\infty i} e^{st}F(s)ds, \tag{1}$$

where f is a signal that is represented as a function over time t, c is a constant that ensures $F(s) \leq \infty$. $s \in \mathbb{C}$ is a complex number, i.e., $s = \sigma + \omega i$ with real numbers σ and ω and for each $s \in \mathbb{C}$, $F(s)$ denotes the components of certain frequency in the original signal f similar to FT.

With LT, the integration and derivation can be presented as a single operator, which is given by:

$$\mathcal{F}(\frac{df(t)}{dt}) = s\mathcal{F}(f), \quad \mathcal{F}(\int_0^t f(u)du) = \frac{1}{s}\mathcal{F}(f). \tag{2}$$

With this property, an ODE can be converted to an algebraic equation by substituting derivation as an operator s. A simple example is illustrated as follows:

$$\frac{d^2x}{dt^2} = -x + u \rightarrow s^2X = -X + U, \tag{3}$$

where u and x are the input and output signal correspondingly and U and X are their Laplacian transformation. This ODE can represent a spring dynamic, whose acceleration $\frac{d^2x}{dt^2}$ depends on the input u and the current position x according to Hooke's law. In this case, the ODE can be solved

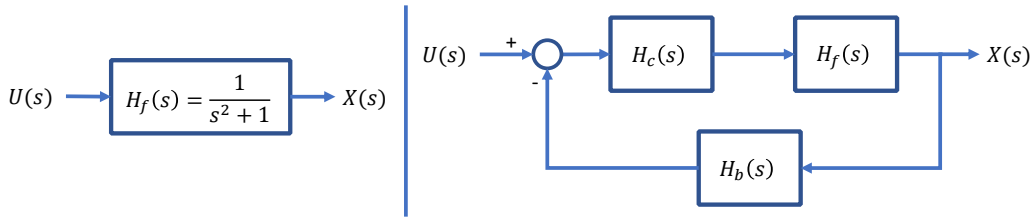


Figure 1: The diagram representations of the inputs and dynamics after LT. Left: the original diagram of the spring dynamic in Eqn. (3). Right: the diagram of a classical controlled dynamic with feedback H_b and controller H_c.

by simply applying the inverse Laplacian transformation to $X = \frac{1}{s^2+1}U = H_f(s)U(s)$. Here $H_f(s) = \frac{1}{s^2+1}$ is the transfer function of this dynamic which can also be represented as a diagram illustrated in Fig. 1. Since the input signal can be considered as summation of infinity Dirac signal at different positions, i.e., $f(t) = \int f(u)\delta(t-u)du$ where δ denotes the Dirac delta function (Dirac, 1981), we only need to analyze the output of the dynamics for Dirac signal whose LT is constant, i.e., $\mathcal{F}(\delta) = 1$. Therefore, the transfer function represents the intrinsic properties of the dynamics.

2.2 Control Theory and Negative Feedback

Given a dynamic, or equivalently a transfer function, the stability can be analyzed according to the roots of the denominator of the transfer function, e.g., $s^2 + 1 = 0$ for Eqn. (3). The roots are called the poles of the dynamics. Specifically, if all poles' real parts are negative, the dynamic is stable; if there are poles which are pure imaginary numbers, the dynamic will oscillate; if there is at least one pole with positive real parts, the dynamic will diverge (Kailath, 1980). In the above example, the poles of the dynamic are $0 \pm i$, which indicates that the object will oscillate around the original.

For those unstable dynamics, we need to stabilize it by adjusting the poles of the dynamics. One of the most commonly adopted methods here is introducing a negative feed back with an auxiliary controller (Kailath, 1980). The diagram is illustrated in Fig. 1 (right). The $H_b(s)$ denotes the transfer function of feedback and $H_c(s)$ denotes the transfer function of controller.

In this setting, the dynamic takes the error (i.e., $E(s) = U(s) - H_b(s)X(s)$) rather than $U(s)$ as input, resulting the fact that the transfer function of the whole controlled dynamic is given by:

$$E(s) = U(s) - H_b(s)X(s), X(s) = H_c(s)H_f(s)E(s) \rightarrow X = \frac{H_c(s)H_f(s)}{1 + H_c(s)H_f(s)H_b(s)}U, \quad (4)$$

which provides us an approach to adjust the poles of an unstable dynamic. For most cases, simply adopting the negative feedback, i.e., let $H_b = H_c = 1$, is enough to stabilize a dynamic. In the following, we provide the dynamics of GANs in the perspective of control theory, and introduce the negative feedback control to stabilize the training process of GANs.

3 Unifying the Dynamics of GANs in Control Theory

In this section, we provide a novel perspective on the dynamic of GANs. Unlike previous methods that mainly focus on the dynamics of the parameters, we directly model the output of two dynamics G and D in the function space. In this part, we mainly focus on the Dirac GAN and unregularized WGAN. This method can be extended to other GANs easily as we illustrated in Appendix A.

To model the dynamics of GANs, we first denote the output of G and D as a function with respect to time t, which is also known as a signal, i.e., $D = D(t, x)$ and $G = G(t, z)$. Then the training process can be represented as a dynamic using an ODE, and the equilibrium is equivalent to the final value of the dynamics (Oppenheim et al., 1998), i.e., the output of the dynamics with $t \rightarrow \infty$.

3.1 The Dynamics of Dirac GAN

In Dirac GAN, the data distribution is defined as $p_D(x) = \delta_c(x) = \delta(x - c)$ which is the density of an idealized point mass at point c. The generator distribution is defined as $p_G(x) = \delta_\theta(x)$, where θ is the learnable parameter of G. In this case, we use $G(t) = G(t, z)$ for simplicity as its output does not depend on z. The discriminator is a simple linear function $D(t, x) = \phi(t) \times x$. Therefore, the minimax optimization problem of Dirac GAN is formulated as:

$$\min_G \max_D \mathbb{E}_{p_D}[\phi x] - \mathbb{E}_{p_G}[\phi x] = c\phi - \theta\phi. \quad (5)$$

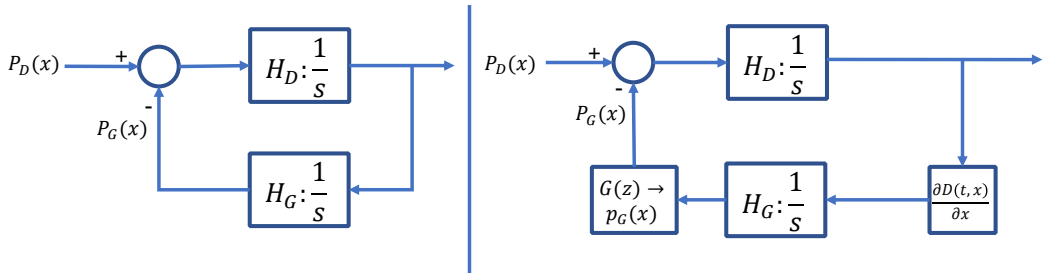


Figure 2: The diagram block of Dirac GAN and unregularized WGAN. Left: Dirac GAN, Right: unregularized WGAN.

For the discriminator, the objective function is given by: $\max_\phi U(D) = c\phi - \theta\phi$, which indicates that the dynamics of D can be formulated as:

$$\frac{\partial U(D)}{\partial \phi} = c - \theta \rightarrow \frac{d\phi(t)}{dt} = c - \theta \rightarrow \frac{\partial D(t,x)}{\partial t} = x \times (c - \theta), \forall x. \tag{6}$$

By considering $c - \theta$ as the error signal, i.e., $e = c - \theta$, the dynamics of the discriminator can be modeled as an integral part $D(t,x) = x\int_0^t c - G(t)dt$. By applying LT to the dynamic, we have:

$$\mathcal{F}(D(t,x)) = \frac{x\mathcal{F}(e)}{s} = \frac{x}{s}E(s), \forall x. \tag{7}$$

For the generator, the objective function is given by $\max_G U(G) = \theta\phi$. Therefore the dynamics of G can be formulated as:

$$\frac{\partial U(G)}{\partial \theta} = \phi \rightarrow \frac{d\theta(t)}{dt} = \phi \rightarrow \frac{\partial G(t)}{\partial t} = \frac{\partial D(t,x)}{\partial x}. \tag{8}$$

Since $D(t,x) = x\int_0^t c - G(t)dt$, we have $\frac{\partial D(t,x)}{\partial x} = \int_0^t c - G(t)dt$. We therefore combine the x part in D's dynamic and the the partial derivation in G into an identity mapping. Then D can be considered as an integral dynamic and its transfer function is given by $H_D(s) = \frac{1}{s}$. Further, the generator also integrate the output of D, which is also $\frac{1}{s}$.

Hence, the block diagram of the Dirac GAN is illustrated in the left panel of Fig. 2. As we can see, D takes the difference between the real sample and fake sample, i.e., $c - \theta$ as input, and output the integration of the input signal. The generator takes D's output as input and output the generator distribution which is the integration of D's output.

It is worth noting that when treating G and D as dynamics, the inputs to the dynamics are not z and x. Instead, for D, the input is $p_D - p_G$, and the input of G is D's output signal $\frac{\partial D(t,x)}{\partial x}$.

3.2 The Dynamics of WGAN

In this subsection, we further model the dynamics of WGAN (Arjovsky et al., 2017) without regularization (i.e., no Lipschitz continuity constraints on the discriminator) in the function space. In this case, we consider the nonparametric setting, where D is a piece-wise linear function, G is a mapping from the noise space to the data space, and both of them are of infinity capacity. This analysis can be extended to other variants of GANs easily.

In the general case, the objective function of D is given by:

$$\max_D U(D) = \mathbb{E}_{p_D}[D(x)] - \mathbb{E}_{p_G}[D(x)] = \int (p_D(x) - p_G(x))D(x)dx. \tag{9}$$

According to the calculus of variations (Gelfand et al., 2000), the gradient of $U(D)$ with respect to the function D, and the dynamic of D are given by:

$$\frac{\partial U(D)}{\partial D} = p_D - p_G \rightarrow \frac{dD(t,x)}{dt} = p_D(x) - p_G(x). \tag{10}$$

Since the input signal is $p_D - p_G$, the dynamics of WGAN is similar to that of the Dirac GAN, whose transfer function can be denoted as $H_D(s) = \frac{1}{s}$.

For the generator, the objective function and its dynamics are given by:

$$\max_G U(G) = \mathbb{E}_{p_z(z)}[D(G(z))] = \int p_z(z)D(G(z))dz \rightarrow \frac{\partial U(G)}{\partial G} = p_z(z)\frac{\partial D(G(z))}{\partial G(z)}, \forall z. \quad (11)$$

It is worth noting that compared to Dirac GAN, this integral is taken in the function space, i.e., $G(t+\delta_t, z) = G(t,z) + \delta_t p_z(z)\frac{\partial D(G(z))}{\partial G(z)}$. Besides, the generator distribution p_G is not a Dirac distribution, resulting in that another operation, which converts the generated samples to the distribution, should be added after G to make the model consistent. The diagram blocks of WGAN is illustrated in Fig. 2 (right). Compared to Dirac-GAN, another two operations are introduced to the framework. We further provide the dynamics of other GANs, including standard GAN proposed by Goodfellow et al. (2014) and Least-Square GAN proposed by Mao et al. (2017) in Appendix A.

4 Understanding and Stabilizing GANs with Negative Feedback

In this section, we first provide the stability analysis of GANs from the perspective of control theory. Besides, many previous methods can be interpreted in this perspective, and we use the momentum as an example. Other further examples are given in Appendix C. We further propose to stabilize the dynamics of GANs using the nagetive feedback controlling methods, and an example of another controller is given in Appendix B.

4.1 Stability

In the dynamics of GANs, we argue that it is dominated by the two integral dynamics of G and D. We therefore ignore the two non-linear operations in GANs in the following analysis. Since both G and D can be considered as integral parts, we can formulate the transfer function of G and D as $H_G = H_D = \frac{1}{s}$. According to Eqn. (4), the transfer function of WGAN is given by:

$$H(s) = \frac{H_D}{1 + H_D H_G} = \frac{1/s}{1 + 1/s^2} = \frac{s}{1+s^2}. \quad (12)$$

It is worth noting that the poles of the transfer function of the whole dynamic are $0 \pm i$, which indicates that the output will oscillate around the equilibrium rather than converge to 0. This result is also consistent with the analysis in (Mescheder et al., 2018). This perspective can also partially explains why larger learning rate or adopting multiple update steps for D, where the transfer function of D can be approximated by $\frac{k}{s}$ with $k > 1$, cannot provide a principle solution for stabilizing GANs. In this case, the poles of the dynamics are adjusted to $0 \pm \sqrt{k}i$ which is still an oscillation dynamics.

4.1.1 The Affect of Momentum

Our method can also provide the stability analysis for other practical techniques used in training GANs. In this part we use the momentum as an examples. Gidel et al. (2018) provide some theoretical analysis of momentum in training GANs, which is formulated in the parameter space. In the following, we re-analyze the momentum in the function space under the perspective of control theory .

The momentum method (Qian, 1999) achieves great success in training deep neural networks. Previous methods such as SGD with momentum can help neural networks to escape from local optima. Its theoretical formulation is given by:

$$\tilde{\theta}_{t+1} = \beta\tilde{\theta}_t + (1-\beta)\nabla\theta_t, \ \theta_{t+1} = \theta_t + \eta\tilde{\theta}_{t+1}. \quad (13)$$

The β is the coefficient for the exponential decay of the gradients, which generally takes 0.9. However, things changed when it comes to training GANs. DCGAN (Radford et al., 2015) recommends a smaller momentum with $\beta = 0.5$. Furthermore, recent state-of-the-art models (Mescheder et al., 2018; Brock et al., 2018; Gulrajani et al., 2017) just remove the momentum.

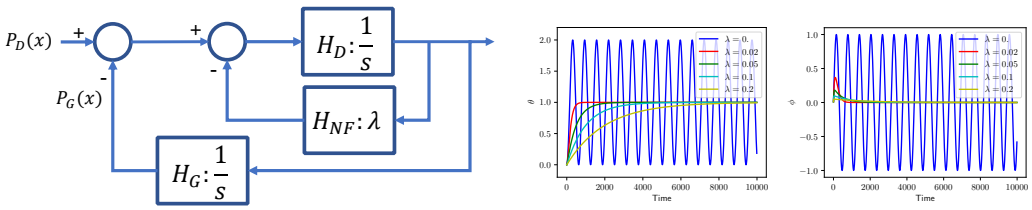
In control theory, the momentum is equivalent to adding an exponential decay to the input of the dynamics (An et al., 2018), and the LT of an exponential decay dynamic is $\frac{1}{s+\tau}$. $\tau > 0$ denotes the decay coefficient which depends on β. Therefore, the transfer function of the D with momentum is $H_{mD}(s) = \frac{1}{s(s+\tau)}$, and the transfer function of the WGAN with momentum is given by:

$$H(s) = \frac{H_{mD}}{1 + H_{mD}H_G} = \frac{1/(s(s+\tau))}{1 + 1/(s^2(s+\tau))} = \frac{s}{s^3 + \tau s^2 + 1}. \quad (14)$$

By letting $(s+a)(s+b)(s+c) = s^3 + \tau s^2 + 1$, we have that $a+b+c = 0$ and $ab+ac+bc = \tau > 0$, which indicates that there are at least one pole of this dynamic whose real part is larger than 0, indicating the instability of the dynamics for GANs with momentum.

Algorithm 1 Negative Feedback GAN

Input: Buffer size N_b, feedback coefficient λ, batch size N, initialized θ and ϕ, learning rate η.
repeat
  Initialize B_r and B_f for real samples and fake samples respectively.
  Sample a batch of $\{x_r\} \sim p_D$, $\{x_f\} \sim p_G$ of N samples.
  Update B_r with $\{x_r\}$. Update B_f with $\{x_f\}$.
  Sample a batch of $x'_r \sim B_r$, $x'_f \sim B_f$ of N samples respectively.
  Estimate the objective of D:
  $U(D) = \frac{1}{N}[\sum_{x\in\{x_r\}} D(x) - \sum_{x\in\{x_f\}} D(x)] - \frac{\lambda}{2N}[\sum_{x\in\{x'_r\}} D^2(x) + \sum_{x\in\{x'_f\}} D^2(x)]$.
  Update ϕ to maximize $U(D)$ with learning rate η.
  Estimate the objective of G: $U(G) = \frac{1}{N}\sum_{x\in\{x_f\}} D(x)$.
  Update θ to maximize $U(G)$ with learning rate η.
until Convergence



(a) The diagram of GANs with negative feedback.

(b) The dynamics of GANs with negative feedback.

Figure 3: The diagram and its corresponding dynamics of GANs using negative feedback.

4.2 Stabilize GANs

According to the above analysis, for both the unregularized WGAN and the GAN with momentum, the equilibrium point (i.e., $p_G = p_D$) is not a stable point. In the following, we propose to stabilize GANs using negative feedback control, which is the most widely-used method in control theory.

In this setting, we set $H_{NF} = \lambda$ which is the transfer function of the negative feedback for D. After applying the negative feedback to D, The diagram of the controlled dynamics is given in Fig. 3 (a). According to Eqn. (4), the transfer function of D and the whole dynamic can be represented as:

$$H'_D(s) = \frac{1/s}{1+\lambda/s} = \frac{1}{s+\lambda} \rightarrow H(s) = \frac{H'_D(s)}{1+H'_D(s)H_G(s)} = \frac{s}{s^2+\lambda s+1}. \tag{15}$$

Therefore, for a positive and small λ, the real part of the two poles are $-\frac{\lambda}{2} < 0$, which indicates that the dynamic is stable. The dynamics of the Dirac GAN using different λ is illustrated in Fig. 3 (b). As we can see, when no feedback is applied (i.e., $\lambda = 0$), the dynamic will oscillate around the equilibrium. Applying negative feedback can significant improve the stability of the dynamics.

4.2.1 Practical Implementation

The most direct way to add negative feed is to introduce a regularization term $R_D = \frac{\lambda}{2}\int D^2(x)dx$ to the objective function of D. However, directly integrating over the input space is computationally infeasible. Instead, we give an approximation to R_D. We assume that the area where $|D(x)|$ is large mainly concentrates on its training samples, i.e., previous samples used to train D. Therefore, we keep two buffers B_r and B_f of size N_b to store the old real samples and fake samples respectively. Then R_D is evaluated on these two buffers to regularize D, the two buffers are updated the with replacement at each step. The training procedure is illustrated in Alg. 1. In the following, we use NF-GAN $(\cdot)$ to denote our proposed method with the hyperparameters $\lambda/2$ denoted in the parentheses.

5 Related Work

GANs are known for its unstable training process. Radford et al. (2015) provide several practical techniques to stabilize the training, such as designing suitable architecture of neural networks and the hyperparameters selection. The Unrolled GAN (Metz et al., 2016) proposes to make D as a function of G by unrolling the training process of the discriminator where the gap between the theoretical analysis and implementation can be reduced. Arjovsky et al. (2017) provide a theoretical analysis of the gradient vanishing problem and further propose to optimize GANs using Wasserstein distance.

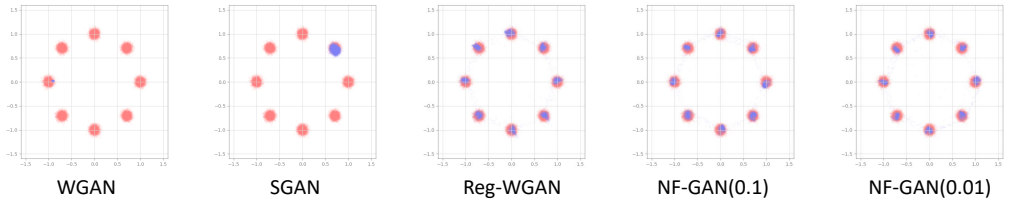


Figure 4: The generated samples for mixture of gaussian distribution. The red points demonstrate the location of data distribution and the blue points are generated samples. Each distribution is plotted using kernel density estimation with 50,000 samples.

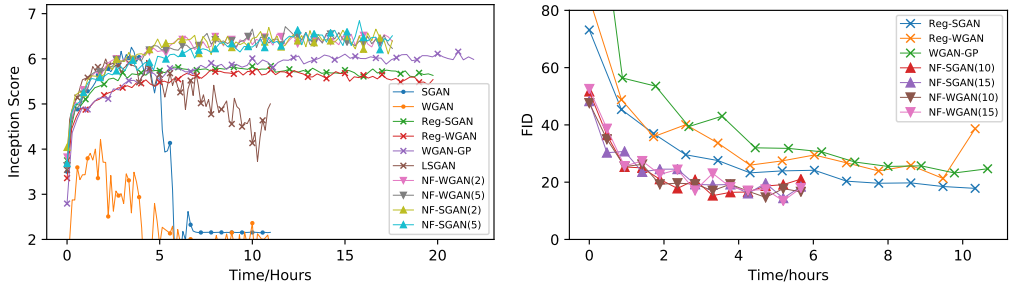


Figure 5: The learning curve of baselines and our proposed methods. Left: The Inception Score of CIFAR10. Right: The FID score for CelebA. We plot the curve with respect to time for better representation of the computational cost and convergence speed.

Gulrajani et al. (2017) use the gradient penalty to ensure the Lipschitz continuity in discriminator, which is required by WGAN. However, GANs still suffer from instability in the training process.

Besides, there is some work that directly focuses on the stability of the training process. Except for the works mentioned in Sec. 1, Peng et al. (2019) also directly model the dynamics of the parameters and adjust the update rules to stabilize GANs. Gidel et al. (2016) propose to find the saddle point of the optimization problem using Frank-wolfe algorithms. However, both of them only consider the bilinear games, and fail to generalize their methods to complex datasets such as natural images.

6 Experiments

In this section, we empirically verify our proposed method on both synthetic data and the widely-adopted CIFAR10 (Krizhevsky et al., 2009) and CelebA (Liu et al., 2015) datasets. Our model is implemented in Pytorch library (Paszke et al., 2017). For synthetic data, we adopt two-layer MLPs for both the generator and the discriminator whose hidden units are 128 and 512 respectively. For natural images, we use the resnet (He et al., 2016) for both the generator and the discriminator following Mescheder et al. (2018) and adopt the ReLU activation (Glorot et al., 2011). The batch size is 512 for synthetic data and 64 for natural images, and the buffer size is set to 100 times of the batch size for all settings. We use rmsprop (Hinton et al., 2012) optimizer with learning rate as 0.0001 and $\alpha = 0.99$ for all models. We refer the regularization proposed in Mescheder et al. (2018) as Reg-GAN, which is the direct baseline of our method. Our code is provided here.

6.1 Synthetic Data

In this section, we evaluate our proposed method on a mixture of Gaussian on the two dimensions. The data distribution consists of 8 2D isotropic Gaussian distributions arranged in a ring, where the radius of the ring is 1, and the deviation of each component Gaussian distribution is 0.05. For the coefficient $\lambda/2$, we follow the theoretical analysis and set it among $\lambda/2 \in \{0.01, 0.05, 0.1\}$.

The generated results are illustrated in Fig. 4 and we further provide the dynamics of the generator distribution in Fig. 11 in Appendix D. As we can see, the unregularized WGAN and SGAN suffer from severe model collapse problem and cannot cover the whole data distribution. Besides, the oscillation can be observed during the training process of WGAN: the generator distribution oscillates among the modes of data distribution. Our method can successfully cover all modes compared to the WGAN and SGAN and is comparable to Reg-GAN.

6.2 CIFAR10 & CelebA

Besides synthetic data, we also evaluate our method on two natural image datasets CIFAR10 and CelebA. Since for large dataset, the non-linearity cannot be ignored, which results in the gap between

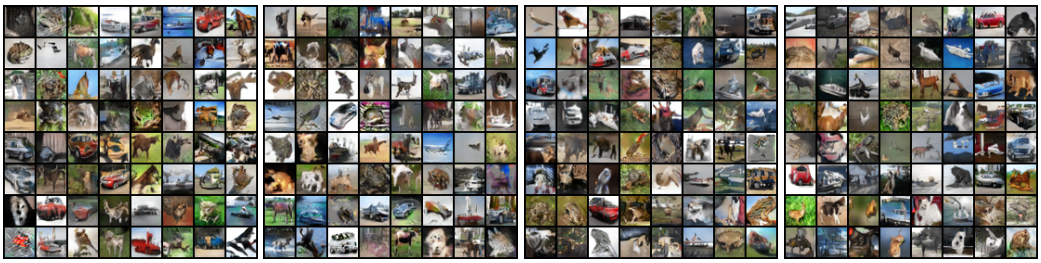


Figure 6: The generated results of CIFAR10 dataset. From left to right: WGAN-GP, Reg-WGAN, NF-WGAN(5), NF-SGAN(5).

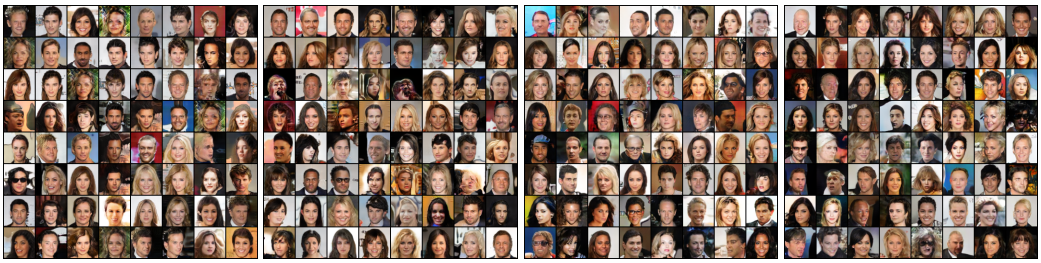


Figure 7: The generated results of CelebA dataset. From left to right: WGAN-GP, Reg-WGAN, NF-WGAN(15), NF-SGAN(15).

the simulation of Dirac GAN and our methods. A small $\lambda/2$ such as 0.01, which is suitable for Dirac GAN is not enough for stabilize the large models. We therefore select the coefficient $\lambda/2$ among $\{1, 2, 5, 10, 15, 20\}$. In this part, we only report representative results. Though our analysis mainly focuses on the WGAN framework, NF can also easily generalize to SGAN whose stability can also be improved. In this part, we also evaluate NF-SGAN with different coefficient $\lambda/2$.

We use the Inception Score (IS) (Salimans et al., 2016) to evaluate the image quality on CIFAR10 and FID score (Gulrajani et al., 2017) on both CIFAR10 and CelebA. The quantitative results are given in Table 1 and the learning curve is reported in Fig. 5. Our method gives an improvement to the FID and IS for both WGAN and SGAN. Specifically, for unregularized methods, the models demonstrate the unstable training process and finally diverge away from the data distribution. Compared with Reg-GAN, our method maintains the stability and provides better convergence results as also shown in Table 1. Besides, our method is more computational efficient, since regularizing the output of D need less computation compared to regularizing $||\frac{\partial D(x)}{\partial x}||$ and this gap is significant for large dataset. Specifically, on Geforce 1080Ti, our method can conduct approximate $12/8$ iterations per second of training for CIFAR10/CelebA, whereas Reg-GAN can only conduct $10/4$ iterations per second. What's more, our method is robust to the choice of the hyperparameter λ. We present the generated images in Fig. 6 and Fig. 7.

Table 1: The FID Score on CIFAR10. The results reported here are the best results over the training process.

Regularization	WGAN	SGAN
No regularization	105.21	28.51
Reg-GAN	30.43	28.39
Gradient Penalty	28.20	–
NF-GAN(2) (ours)	21.90	22.35
NF-GAN(5) (ours)	22.31	**21.61**
NF-GAN(10) (ours)	**21.19**	23.34

7 Conclusion

In this paper, we propose a novel method to analyze the dynamics of GANs. Instead of focusing on the equilibrium, our method directly models the dynamic of the discriminator and the generator during the training process. Using Laplacian transformation and control theory, the stability can be easily understood according to the poles of the dynamics. Besides, the negative feedback control is proposed to stabilize the dynamics of GANs, and the empirical results demonstrate that our method can successfully stabilize GANs and provide better convergence results.

Though negative feedback shows promise results, understanding and stabilizing GANs are far from solved. For one thing, our analysis mainly focuses on the continuous cases, where the practical implementation optimizes both G and D in discrete time steps, where the Z-transformation provides a better tools for discrete time dynamics. For another, we simply ignore the non-linear operations for WGAN. These operations can be further analyzed using modern control theory and nonlinear control methods (Khalil, 2002) to provide better results. Indeed, our method does not provide a perfect solution to GANs' dynamic. It does provide a new perspective to understand and stabilize GANs.

REFERENCES

Wangpeng An, Haoqian Wang, Qingyun Sun, Jun Xu, Qionghai Dai, and Lei Zhang. A pid controller approach for stochastic optimization of deep networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8522–8531, 2018.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pp. 214–223, 2017.

Karl Johan Åström and Tore Hägglund. *PID controllers: theory, design, and tuning*, volume 2. Instrument society of America Research Triangle Park, NC, 1995.

Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

Tatjana Chavdarova and François Fleuret. Sgan: An alternative training of generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9407–9415, 2018.

Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in neural information processing systems*, pp. 2172–2180, 2016.

Paul Adrien Maurice Dirac. *The principles of quantum mechanics*. Number 27. Oxford university press, 1981.

Jeff Donahue and Karen Simonyan. Large scale adversarial representation learning. *arXiv preprint arXiv:1907.02544*, 2019.

Chao Du, Kun Xu, Chongxuan Li, Jun Zhu, and Bo Zhang. Learning implicit generative models by teaching explicit ones. *arXiv preprint arXiv:1807.03870*, 2018.

Izrail Moiseevitch Gelfand, Richard A Silverman, et al. *Calculus of variations*. Courier Corporation, 2000.

Gauthier Gidel, Tony Jebara, and Simon Lacoste-Julien. Frank-wolfe algorithms for saddle point problems. *arXiv preprint arXiv:1610.07797*, 2016.

Gauthier Gidel, Reyhane Askari Hemmat, Mohammad Pezeshki, Remi Lepriol, Gabriel Huang, Simon Lacoste-Julien, and Ioannis Mitliagkas. Negative momentum for improved game dynamics. *arXiv preprint arXiv:1807.04740*, 2018.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 315–323, 2011.

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

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*, pp. 2672–2680, 2014.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Advances in neural information processing systems*, pp. 5767–5777, 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. *Cited on*, 14:8, 2012.

Rie Johnson and Tong Zhang. Composite functional gradient learning of generative adversarial models. *arXiv preprint arXiv:1801.06309*, 2018.

Thomas Kailath. *Linear systems*, volume 156. Prentice-Hall Englewood Cliffs, NJ, 1980.

Hassan K Khalil. Nonlinear systems. *Upper Saddle River*, 2002.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.

Chongxuan Li, Taufik Xu, Jun Zhu, and Bo Zhang. Triple generative adversarial nets. In *Advances in neural information processing systems*, pp. 4088–4098, 2017.

Kevin J Liang, Chunyuan Li, Guoyin Wang, and Lawrence Carin. Generative adversarial network training is a continual learning problem. *arXiv preprint arXiv:1811.11083*, 2018.

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

Mario Lucic, Karol Kurach, Marcin Michalski, Sylvain Gelly, and Olivier Bousquet. Are gans created equal? a large-scale study. In *Advances in neural information processing systems*, pp. 700–709, 2018.

Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2794–2802, 2017.

Lars Mescheder, Sebastian Nowozin, and Andreas Geiger. The numerics of gans. In *Advances in Neural Information Processing Systems*, pp. 1825–1835, 2017.

Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for gans do actually converge? *arXiv preprint arXiv:1801.04406*, 2018.

Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. Unrolled generative adversarial networks. *ArXiv*, abs/1611.02163, 2016.

Tu Nguyen, Trung Le, Hung Vu, and Dinh Phung. Dual discriminator generative adversarial nets. In *Advances in Neural Information Processing Systems*, pp. 2670–2680, 2017.

Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-gan: Training generative neural samplers using variational divergence minimization. In *Advances in neural information processing systems*, pp. 271–279, 2016.

Alan V Oppenheim, Alan S Willsky, and S Hamid Nawab. *Señales y sistemas*. Pearson Educación, 1998.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *NIPS Autodiff Workshop*, 2017.

Wei Peng, Yuhong Dai, Hui Zhang, and Lizhi Cheng. Training gans with centripetal acceleration. *arXiv preprint arXiv:1902.08949*, 2019.

Ning Qian. On the momentum term in gradient descent learning algorithms. *Neural networks*, 12(1): 145–151, 1999.

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

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in neural information processing systems*, pp. 2234–2242, 2016.

David Vernon Widder. *Laplace transform (PMS-6)*. Princeton university press, 2015.

A GANs AND THEIR CORRESPONDING DYNAMICS

In this section, we provide an illustration for different kind of GANs and their corresponding dynamics.

A.1 VANILLA GAN AND NON-SATURATION GAN

The GANs proposed by (Goodfellow et al., 2014) are formulated as follows:

$$\min_G \max_D \mathbb{E}_{p_D}[\log \sigma(D(x))] + \mathbb{E}_{p_G}[\log(1-\sigma(D(x)))].$$

Therefore, the objective function and the corresponding dynamics of D is given by:

- The objective function:

$$\begin{aligned}\max_D \quad U(D) =& \mathbb{E}_{p_D}[\log \sigma(D(x))] + \mathbb{E}_{p_G}[\log(1-\sigma(D(x)))] \\ =& \int p_D(x) \log \frac{1}{1+\exp(-D(x)) + p_G(x) \log \frac{\exp(-D(x))}{1+\exp(-D(x))}} dx \\ =& -\int p_D(x) \log(1+\exp(-D(x))) + p_G(x) \log(1+\exp(D(x)))dx.\end{aligned}$$

- Therefore, given current G, the gradient of $U(D)$ with respect to D at certain x is given by:

$$\begin{aligned}\frac{\partial U(D)}{dD(x)} &= p_D \frac{\exp(-D(x))}{1+\exp(-D(x))} - p_G \frac{\exp(D(x))}{1+\exp(D(x))} \\ &= p_D(1-\sigma(D(x))) - p_G \sigma(D(x)) \\ &\rightarrow \frac{dD_t(x)}{dt} = \frac{\partial U(D(x))}{dD(x)} = p_D(1-\sigma(D(x))) - p_G\sigma(D(x)).\end{aligned}$$

The objective function and the corresponding dynamics of G are given by:

- The objective function:

$$\begin{aligned}\min_G \quad U(G) =& \mathbb{E}_{p_G}[\log(1-\sigma(D(x)))] = \mathbb{E}_{p_z}[\log(1-\sigma(D(G(z))))] \\ =& \int p_z(z) \log \frac{\exp(-D(G(z)))}{1+\exp(-D(G(z)))} dz \\ =& -1 \times \int p_z(z) \log(1+\exp(D(G(z))))dz.\end{aligned}$$

- Since G is optimized using reparameterization, we rewrite $G(z)$ as x_z for simplicity. Given current D, the gradient of $U(G)$ w.r.t. G at a certain z is given by:

$$\begin{aligned}\frac{\partial U(G)}{\partial x_z} &= -1 \times p(z) \frac{\exp(D(x_z))}{1+\exp(D(x_z))} \frac{\partial D(x_z)}{\partial x_z} = -1 \times p(z)\sigma(D(x_z)) \frac{\partial D(x_z)}{\partial x_z} \\ &\rightarrow \frac{dG_t(z)}{dt} = p(z)\sigma(D(x_z)) \frac{\partial D(x_z)}{\partial x_z}\end{aligned}$$

For a well-trained D, which can successfully tell real samples from generated samples with high confidence, i.e., $D(x_z) \approx 0$ for all z, the gradient for G will tend to zero almost everywhere, which will result in gradient vanishing. This analysis is consistent with previous literature (Johnson & Zhang, 2018; Arjovsky et al., 2017).

For non-saturation GAN, the objective function and dynamics of G are given by:

- The objective is given by:

$$\begin{aligned}\max_G \quad U(G) =& \mathbb{E}_{p_G}[\log \sigma(D(x))] = \mathbb{E}_{p_z}[\log \sigma(D(G(z)))] \\ =& \int p_z(z) \log \frac{\exp(D(G(z)))}{1+\exp(D(G(z)))} dz \\ =& \int p_z(z)(D(G(z)) - \log(1+\exp(D(G(z)))))\end{aligned}$$

- We rewrite $G(z)$ as x_z and provide the gradient of $U(G)$ w.r.t. x_z as follows:

$$\begin{aligned}\frac{\partial U(G)}{\partial x_z} &= p_z(z)(1 - \frac{\exp(D(x_z))}{1+\exp(D(x_z))})\frac{\partial D(x_z)}{\partial x_z} \\ &= p_z(z)(1-\sigma(D(x_z)))\frac{\partial D(x_z)}{\partial x_z} \\ \rightarrow \frac{dG_t(z)}{dt} &= p(z)(1-\sigma(D(x_z)))\frac{\partial D(x_z)}{\partial x_z}\end{aligned}$$

A.2 LEAST-SQUARE GAN

(Mao et al., 2017) propose to use the least square loss to train both the discriminator and the generator. The objective for the D and G are selected differently according to its theoretical analysis. Specifically, the objective function of D and G are given by:

$$\begin{aligned}\min_D U(D) &= \frac{1}{2}\mathbb{E}_{p_D}[(D(x)-1)^2] + \frac{1}{2}\mathbb{E}_{p_G}[(D(x))^2]; \\ \min_G U(G) &= \frac{1}{2}\mathbb{E}_{p_z}[(D(G(z))-1)^2].\end{aligned}$$

Therefore, the dynamics of the D and G are given by:

- The dynamic of D:

$$\begin{aligned}\frac{\partial U(D)}{\partial D(x)} &= \frac{1}{2}\frac{\partial}{\partial D(x)}(p_D(D(x)-1)^2 + p_G D(x)^2) \\ &= p_D(D(x)-1) + p_G D(x). \\ \rightarrow \frac{dD_t(x)}{dt} &= p_D - (p_D + p_G)D_t(x)\end{aligned} \tag{16}$$

- The dynamic of G:

$$\begin{aligned}\frac{\partial U(G)}{\partial x_z} &= p_z(D(x_z)-1)\frac{\partial D(x_z)}{\partial x_z} \\ \rightarrow \frac{dG_t(z)}{dt} &= p_z(z)(1-D(x_z))\frac{\partial D(x_z)}{\partial x_z}\end{aligned}$$

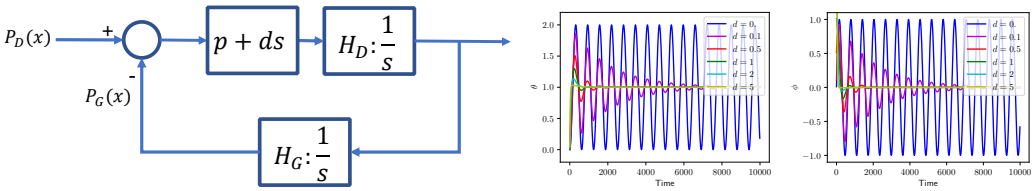
It is worth noting that if the G is fixed, the dynamics of D can be considered as an exponential decay which is the same as the D with negative feedback control. However, since G is changing during the training process, the dynamics of D is actually coupling with the dynamics of G, which generally results in the unstable dynamics Kailath (1980). The experimental results on CIFAR10 also verify this argument. It also explains why we use two buffers of old samples to approximate the regularization term instead of directly regularizing the output of D using current samples.

B FURTHER DISCUSSION: PD CONTROLLER

Besides negative feedback, the most commonly used controller is the PID controller, which consists of three parts: the partial (P), integration (I) and derivation (D). Specifically, the PID controller takes the error signal as input, and output the summation of the above three parts, whose formal definition is defined as follows:

$$c(t) = K_p e(t) + K_i \int_{u=0}^{t} e(u)du + K_d \frac{de(t)}{dt} \rightarrow H_c(s) = \frac{K_p s + K_i + K_d s^2}{s}. \tag{17}$$

K_p, K_i, K_d denotes the corresponding coefficients for PID controller. $e(t)$ denotes the error signal and c is the output of the controller, which is also the actual input of the dynamic. Generally, the I part can reduce the static error, and the D part can make the dynamic stable and accelerate the convergence.



(a) The diagram of GANs with PD Controller.

(b) The dynamics of GANs with with PD Controller.

Figure 8: The diagram and its corresponding dynamics of GANs using PD controller.

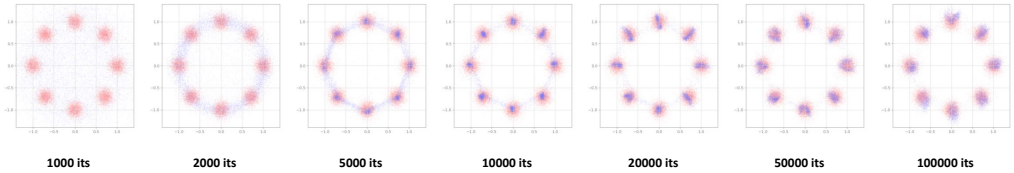


Figure 9: The generated results of PD controller with $d = 5$.

The PID controller is introduced to adjust the poles of the dynamic. Since the static error, i.e., the difference between the stable point and targeted point, is zero, the integral part is not necessary, and we only employ the PD controller in the following. By letting $H_c(s) = p + ds$, where p and d is the corresponding coefficient for the P and D part, the transfer function of the whole dynamic is given by:

$$H(s) = \frac{H_c(s)H_D(s)}{1 + H_c(s)H_G(s)H_D(s)} = \frac{s(p + ds)}{s^2 + ds + p}. \tag{18}$$

As we can see, given positive p and d, the PD controller can successfully adjust the real parts poles of the dynamic to negative. In the following analysis and implementation, we fix $p = 1$ for simplicity.

The dynamics of Dirac GANs using PD controller is given in Fig. 8 (b). With a PD controller, the dynamic can be successfully stabilized. For general GAN, the objective of D at time t is given by:

$$\begin{aligned} U(D, t) = \mathbb{E}_{p_D(t,x)}[D(x)] - \mathbb{E}_{p_G(t,x)}[D(x)] + \\ \lambda\{\mathbb{E}_{p_D(t,x)}[D(x)] - \mathbb{E}_{p_G(t,x)}[D(x)] - \mathbb{E}_{p_D(t-\delta t,x)}[D(x)] + \mathbb{E}_{p_G(t-\delta t,x)}[D(x)]\}. \end{aligned} \tag{19}$$

The regularization is actually minimizing the loss of D in the current step as well as maximizing the loss of D for previous step.

B.1 Experimental Results

The results of the synthetic data are illustrated in Fig. 9. WGAN with PD controller can successfully cover the whole data distribution and be stable at the equilibrium.

However, we fail to generalize PD controller to the natural image dataset such as CIFAR10. We argue that the main reason is the overshooting for the dynamics of D, as illustrated in Fig. 8 (b), where the D gives a large derivation from the equilibrium at the beginning. Because of the highly-nonlinearity in neural networks, this overshooting will result in unexpected behavior of neural networks. In contrast, for the negative feedback, the overshooting is much smaller, which enables it to generalize to the non-linear neural networks.

C Unifying Previous Methods in Control Theory

In this section, we also provide some analysis that interprets previous methods in the perspective of control theory.

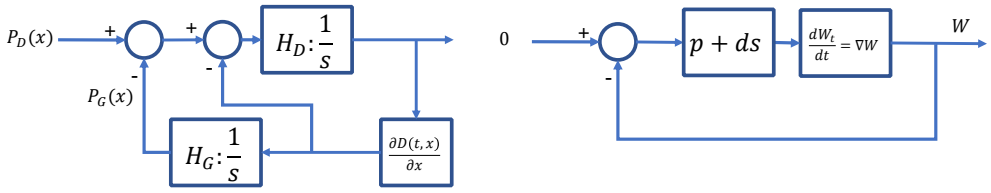


Figure 10: The diagram of previous methods. Left: the diagram of Reg-GAN which takes the $||\frac{\partial D}{\partial x}||$ as regularization. Right: the diagram of negative momentum together with weight decay.

C.1 Reg-GAN

Mescheder et al. (2018) propose a regularization which is added on D's gradient of x, i.e., $R_d(x) = ||\frac{\partial D(x)}{\partial x}||_2^2$. Therefore, it can be considered as an other form of negative feedback, which is illustrated in Fig. 10.

Compare to our analysis, Reg-GAN provides more accurate modeling of the dynamics and directly regularizes on the input of the dynamic of G. Therefore, Reg-GAN can also successfully stabilize the dynamic of GANs, which is consistent with our experimental results. Compared to Reg-GAN, our method is more computationally efficient.

C.2 Weight Decay and Negative Momentum

Gidel et al. (2018) analyze the effect of momentum in the training of GANs, and we already interpret this analysis in the context of control theory. Besides, the authors also propose to use the negative momentum to stabilize GANs, which is illustrated as follows:

$$W_{t+1} = W_t + \eta \nabla W + \lambda(W_t - W_{t-1}). \tag{20}$$

Together with weight decay, the controlled dynamics of W is given:

$$W_{t+1} = W_t + \eta \nabla W + \lambda_1(W_t - W_{t-1}) - \lambda_2 W_t. \tag{21}$$

In this setting, Eqn. (21) is equivalent to the PD controller with negative feedback in the parameter space whose diagram can be illustrated in Fig. 10. The $\lambda_1(W_t - W_{t-1})$ denotes the D part and $\lambda_2 W_t$ denotes the P part.

D Dynamics of Synthetic Data

In this section, we provide the training dynamics of several baselines and NF-GAN on the synthetic data. The result is illustrated in Fig. 11.

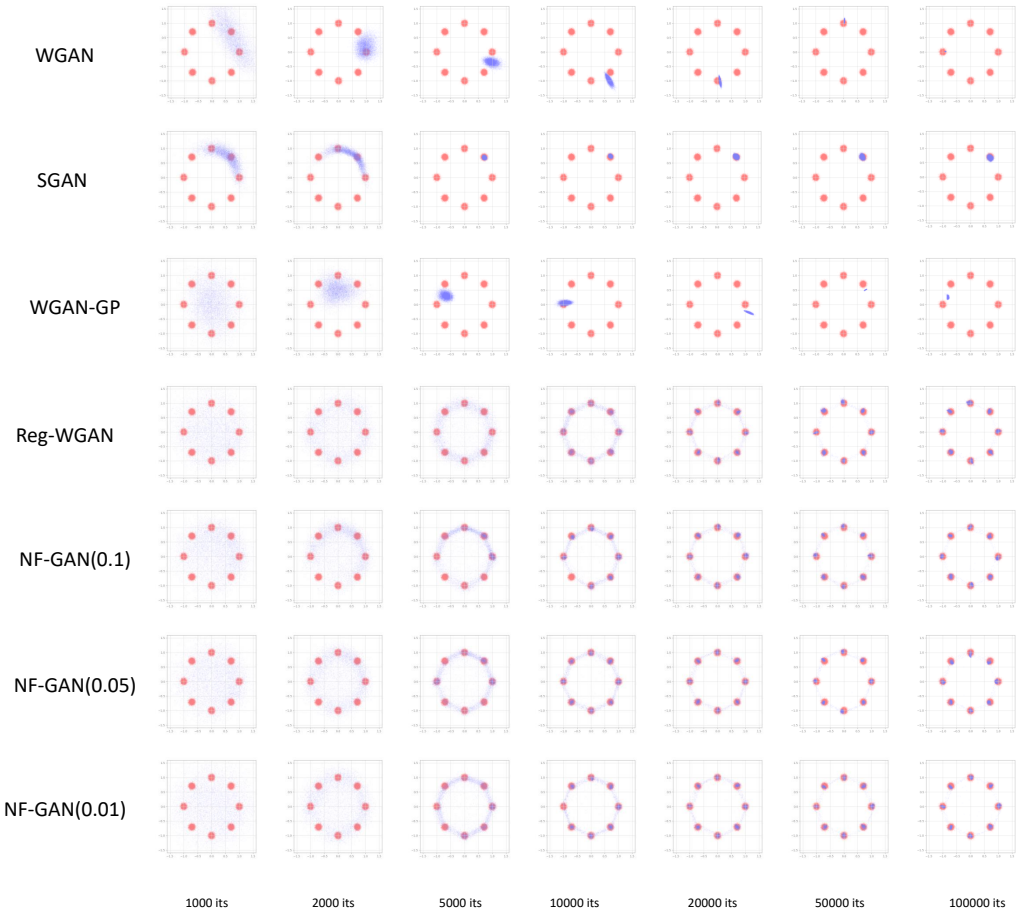


Figure 11: The training dynamics of various GANs on synthetic data.