MonoFlow: Rethinking Divergence GANs via the Perspective of Wasserstein Gradient Flows

Mingxuan Yi [1] **Zhanxing Zhu** [2 3] **Song Liu** [1]

Abstract

The conventional understanding of adversarial training in generative adversarial networks (GANs) is that the discriminator is trained to estimate a divergence, and the generator learns to minimize this divergence. We argue that despite the fact that many variants of GANs were developed following this paradigm, the current theoretical understanding of GANs and their practical algorithms are inconsistent. In this paper, we leverage Wasserstein gradient flows which characterize the evolution of particles in the sample space, to gain theoretical insights and algorithmic inspiration for GANs. We introduce a unified generative modeling framework – MonoFlow: the particle evolution is rescaled via a monotonically increasing mapping of the log density ratio. Under our framework, adversarial training can be viewed as a procedure first obtaining MonoFlow's vector field via training the discriminator and the generator learns to draw the particle flow defined by the corresponding vector field. We also reveal the fundamental difference between variational divergence minimization and adversarial training. This analysis helps us to identify what types of generator loss functions can lead to the successful training of GANs and suggest that GANs may have more loss designs beyond the literature (e.g., non-saturated loss), as long as they realize MonoFlow. Consistent empirical studies are included to validate the effectiveness of our framework.

1. Introduction

Generative adversarial nets (GANs) (Goodfellow et al., 2014; Jabbar et al., 2021) are a powerful generative modeling framework that has gained tremendous attention in recent years. GANs have achieved significant successes in applications, especially in high-dimensional image processing such as high-fidelity image generation (Brock et al., 2018; Karras et al., 2019), super-resolution (Ledig et al., 2017) and domain adaption (Zhang et al., 2017).

In the GAN framework, a discriminator d and a generator g play a minmax game. The discriminator is trained to distinguish real and fake samples and the generator is trained to generate fake samples to fool the discriminator. The equilibrium of the vanilla GAN is defined by[1]

$$\min_g \max_d V(g,d) = \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}\big\{\log \sigma[d(\mathbf{x})]\big\} + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}}\big\{\log\big(1-\sigma[d(g(\mathbf{z}))]\big)\big\} \tag{1}$$

The elementary optimization approach to solve the minmax game is adversarial training. Previous perspectives explained it as first estimating Jensen-Shannon divergence and the generator learns to minimize this divergence. Several variants of GANs have been developed based on this point of view for other probability divergences, e.g., χ^2 divergence (Mao et al., 2017), Kullback-Leibler (KL) divergence (Arbel et al., 2021) and general f-divergences (Nowozin et al., 2016; Uehara et al., 2016), while others are developed with Integral Probability Metrics (Arjovsky et al., 2017; Dziugaite et al., 2015; Mroueh et al., 2018a). However, we emphasize that the traditional perspective on GANs is inconsistent and we present three non-negligible facts which are commonly associated with adversarial training, making it markedly different from the standard variational divergence minimization (VDM) problem:

1. The estimated divergence is computed from the discriminator $d(\mathbf{x})$. $d(\mathbf{x})$ is trained using samples $\mathbf{x}$ only such that it cannot capture the variability of the generator's distribution p_g (Metz et al., 2017; Franceschi et al., 2022). However, the optimal discriminator in the adversarial game by Goodfellow et al. (2014) requires p_g to be a functional variable such that the dependency between the optimal discriminator and p_g exists, i.e.,

[1]University of Bristol. [2]Changping National Laboratory, China. [3]Peking University. Correspondence to: Mingxuan Yi <mingxuan.yi@bristol.ac.uk>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

[1]We use a slightly different notation: $d(\mathbf{x})$ is the logit output of the classifier and $\sigma(\cdot)$ is the Sigmoid activation.

the discriminator is a function $d(\mathbf{x}, g)$ taking as input generator's parameter as well.

2. The generator minimizes a divergence with a missing term, e.g., the vanilla GAN only minimizes the second term of the Jensen-Shannon divergence $-\mathbb{E}_{\mathbf{z}\sim p_\mathbf{z}}\left\{-\log\left(1-\sigma[d(g(\mathbf{z}))]\right)\right\}$ which is, however, a KL divergence up to a constant, see Eq. (5) of Goodfellow et al. (2014).

3. Practical algorithms are inconsistent with the theory, a heuristic trick "non-saturated loss" is commonly adopted to mitigate the gradient vanishing problem, but it still lacks a rigorous mathematical understanding. For example, the generator loss of the non-saturated GAN is $-\mathbb{E}_{\mathbf{z}\sim p_\mathbf{z}}\left\{\log\sigma[d(g(\mathbf{z}))]\right\}$. We can even modify the generator loss to the logit loss $-\mathbb{E}_{\mathbf{z}\sim p_\mathbf{z}}\left\{d(g(\mathbf{z}))\right\}$ or the $\operatorname{arcsinh}$ loss $-\mathbb{E}_{\mathbf{z}\sim p_\mathbf{z}}\left\{\operatorname{arcsinh}\left(d(g(\mathbf{z}))\right)\right\}$, the generator still learns the data distribution, as shown in Figure 1.

Figure 1. Generated Celeb-A faces (Liu et al., 2015) with the logit loss and the $\operatorname{arcsinh}$ loss.

All of the above generator losses satisfy

$$-\mathbb{E}_{\mathbf{z}\sim p_\mathbf{z}}\left\{h[d(g(\mathbf{z}))]\right\}, \tag{2}$$

where $h\colon \mathbb{R}\to\mathbb{R}$ is a strictly monotonically increasing function with $h'(\cdot)>0$. It is known the logit output $d(\mathbf{x})$ of a binary classifier in Eq. (1) is the logarithm density ratio estimator between two distributions (Qin, 1998; Sugiyama et al., 2012). To gain a deeper understanding of divergence GANs, we study the Wasserstein gradient flow of the KL divergence which characterizes a Euclidean particle flow ordinary differential equation (ODE). This ODE is also known as the "probability flow ODE" (Song et al., 2021) of Langevin dynamics, with its vector field defined by the gradient of the log density ratio. Based on this ODE, we propose the **MonoFlow** framework – transforming the log density ratio by a strictly increasing mapping such that the vector field of the ODE is rescaled along the same direction. Consequently, learning to simulate MonoFlow is identical to training divergence GANs. All variants of divergence GANs are a subclass of our framework. We reveal that the discriminator loss and generator loss do not need to follow the same objective which is contradictory to the adversarial game (Goodfellow et al., 2014). The discriminator maximizes an objective to obtain a bijection of the log density ratio. Then the generator loss can be any strictly increasing mapping of this learned log ratio. Our contributions can be summarized as follows:

- A novel generative modeling framework has been developed, which unifies divergence GANs and provides a new understanding of their training dynamics. This framework not only provides a new theoretical perspective but also ensures practical consistency.
- We reveal the fundamental difference between VDM and adversarial training, which indicates that the previous analysis of GANs based on the perspective of VDM might not provide benefits, and instead we should treat GANs as a particle flow method similar to diffusion models (Ho et al., 2020; Song et al., 2021).
- An analysis of what types of generator losses can practically lead to the success of training GAN. Our framework explains why and how non-saturated loss works.
- An algorithmic inspiration where GANs may have more variants of generative losses than we already know.

2. Wasserstein Gradient Flows

In this section, we review the definition of gradient flows in Wasserstein space $(\mathcal{P}(\mathbb{R}^n), W_2)$, the space of Borel probability measures $\mathcal{P}(\mathbb{R}^n)$ defined on $\mathbb{R}^n$ with finite second moments and equipped with the Wasserstein-2 metric. An absolutely continuous curve of probability measures $\{q_t\}_{t\geq 0}\in\mathcal{P}(\mathbb{R}^n)$ is a Wasserstein gradient flow if it satisfies the following continuity equation (Ambrosio et al., 2008),

$$\frac{\partial q_t}{\partial t} = \operatorname{div}\big(q_t\nabla_{W_2}\mathcal{F}(q_t)\big), \tag{3}$$

where $\nabla_{W_2}\mathcal{F}(q_t)$ is called the Wasserstein gradient of the functional $\mathcal{F}\colon \mathcal{P}(\mathbb{R}^n)\to\mathbb{R}$.

The Wasserstein gradient is given as $\nabla_\mathbf{x}(\delta\mathcal{F}/\delta q_t)$, i.e. the Euclidean gradient of the functional's first variation $\delta\mathcal{F}/\delta q_t$. Specifically, for the KL divergence $\mathcal{F}(q_t)=\int\log(q_t/p)\mathrm{d}q_t$, where p is a fixed target probability measure, we have $\delta\mathcal{F}/\delta q_t=\log q_t-\log p+1$. Hence, the Wasserstein gradient flow of the KL divergence reads the Fokker-Planck equation (Risken & Risken, 1996),

$$\frac{\partial q_t}{\partial t} = \operatorname{div}\big(q_t(\nabla_\mathbf{x}\log q_t-\nabla_\mathbf{x}\log p)\big). \tag{4}$$

If we denote the Euclidean path of random variables as $\{\mathbf{x}_t\}_{t\geq 0}\in\mathbb{R}^n$ with the initial condition $\mathbf{x}_0\sim q_0$, we can

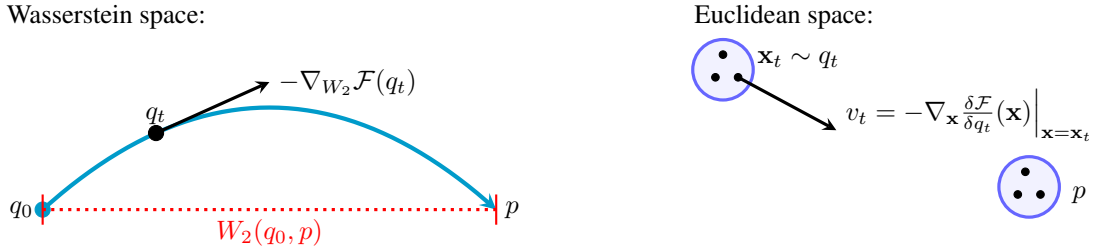


Figure 2. The illustration of a Wasserstein gradient flow and its particle evolution. In Wasserstein space, the blue curve is a gradient flow and the red dotted line is a geodesic. q_t evolves along a curve whose tangent vector is given by $-\nabla_{W_2}\mathcal{F}(q_t)$ such that the functional is always decreasing with time. Correspondingly, particles evolve in Euclidean space towards the target measure p with the vector field $-\nabla_{\mathbf{x}} \frac{\delta \mathcal{F}}{\delta q_t}(\mathbf{x})$. Note that directly minimizing the Wasserstein-2 metric $W_2(q_t, p)$ instead yields a path $\{q_t\}_{t\geq 0}$ along the geodesic connecting q_0 and p over Wasserstein space.

define an ordinary differential equation (ODE) to describe the evolution of particles in $\mathbb{R}^n$,

$$\mathrm{d}\mathbf{x}_t = \big(\nabla_{\mathbf{x}} \log p(\mathbf{x}_t) - \nabla_{\mathbf{x}} \log q_t(\mathbf{x}_t)\big)\mathrm{d}t := v_t(\mathbf{x}_t)\mathrm{d}t, \tag{5}$$

where the vector field v_t of these particles is the negative Euclidean gradient of the functional's first variation. As shown in Figure 2, Wasserstein gradient flows establish a connection between the probability evolution in Wasserstein space and its associated particle evolution in Euclidean space.

Applying Itô integral to Langevin dynamics $\mathrm{d}\mathbf{x}_t = \nabla_{\mathbf{x}} \log p(\mathbf{x}_t)\mathrm{d}t + \sqrt{2}\mathrm{d}\mathbf{w}_t$ where $\mathrm{d}\mathbf{w}_t$ is a Wiener process, we obtain the same Fokker-Planck equation in Eq. (4). This indicates that the deterministic particle evolution by the ODE can be approximated via a stochastic differential equation (SDE). Langevin dynamics admits the same marginal probability measure q_t as Eq. (5), this relation of SDE and its corresponding ODE was also studied in score-based diffusion models (Song et al., 2021). Langevin dynamics was first interpreted as the Wasserstein gradient flow of the KL divergence by Jordan et al. (1998); Otto (2001). It plays an important role in generative modeling as a sampling scheme. In order to transform noises into the target data distribution by Langevin dynamics, an essential step is to fit the data distribution using energy-based models (Song & Kingma, 2021) or to directly estimate its scores with score-matching techniques (Hyvärinen & Dayan, 2005; Vincent, 2011; Song & Ermon, 2019).

3. MonoFlow: A Unified Generative Modeling Framework

This section presents our main contribution that connects gradient flows and divergence GANs. We first introduce **MonoFlow** where the ODE evolution is rescaled via a monotonically increasing function. Consequently, learning to simulate and draw the rescaled particle flow recovers the bi-level optimization dynamics of training divergence GANs. This gives us a novel understanding of the hidden mechanism of adversarial training.

3.1. MonoFlow

We consider the ODE in Eq. (5) with a fixed target measure p, e.g., a data distribution in a generative modeling scenario. Assume that we have a time-dependent log density ratio function as $\log r_t(\mathbf{x}) = \log\big[p(\mathbf{x})/q_t(\mathbf{x})\big]$, the ODE can be rewritten as

$$\mathrm{d}\mathbf{x}_t = \nabla_{\mathbf{x}} \log r_t(\mathbf{x}_t)\mathrm{d}t, \quad \mathbf{x}_0 \sim q_0. \tag{6}$$

This is a gradient flow in Euclidean space where its vector field is the gradient of the log density ratio. With a strictly monotonically increasing and differentiable mapping $h\colon \mathbb{R} \to \mathbb{R}$, we can define another ODE:

$$\begin{aligned}\mathrm{d}\mathbf{x}_t &= \nabla_{\mathbf{x}} h\big(\log r_t(\mathbf{x}_t)\big)\mathrm{d}t \\ &= h'\big(\log r_t(\mathbf{x}_t)\big)\nabla_{\mathbf{x}} \log r_t(\mathbf{x}_t)\mathrm{d}t, \quad \mathbf{x}_0 \sim q_0\end{aligned} \tag{7}$$

By transforming the time-dependent log density ratio under the mapping h, its first-order derivative rescales the vector field of the original particle flows defined in Eq. (6). We call Eq. (7) as **MonoFlow**. MonoFlow defines a different family of vector fields $\{v_t\}_{t\geq 0}$ for the particle evolution where $v_t(\mathbf{x}_t) = h'\big(\log r_t(\mathbf{x}_t)\big)\nabla_{\mathbf{x}} \log r_t(\mathbf{x}_t)$. Conversely, the vector fields $\{v_t\}_{t\geq 0}$ also determine an absolutely continuous curve $\{q_t\}_{t\geq 0}$ in Wasserstein space by the continuity equation (see Theorem 4.6 of Ambrosio et al. (2008)),

$$\frac{\partial q_t}{\partial t} = -\mathrm{div}(q_t v_t), \tag{8}$$

under mild regularity conditions. Hence the probability evolution of MonoFlow is described by

$$\frac{\partial q_t}{\partial t} = \mathrm{div}\big(D_t \nabla_{\mathbf{x}} q_t\big) - \mathrm{div}\big(\zeta_t^{-1} q_t \nabla_{\mathbf{x}} \log p\big), \tag{9}$$

where $D_t = \zeta_t^{-1} = h'(\log r_t)$. Eq. (9) is a special case of convection-diffusion equations where D_t is called the diffusion coefficient and ζ_t^{-1} is called mobility. MonoFlow defines a positive diffusion coefficient. This has a physical interpretation that particles diffuse to spread probability mass over the target measure other than concentrate. Next, we study the properties of MonoFlow. Proofs are provided in Appendix A.

Table 1. Different types of divergence GANs. f is a convex function and $\tilde{f}$ is the convex conjugate by $\tilde{f}(d) = \sup_{r \in \mathrm{dom} f}\{rd - f(r)\}$.

	$\phi(d)$	$\psi(d)$	$d^*(\mathbf{x})$	$h_{\mathcal{T}}(d)$
Vanilla GAN	$\log \sigma(d)$	$\log(1-\sigma(d))$	$\log r(\mathbf{x})$	$-\log(1-\sigma(d))$
Non-saturated GAN	$\log \sigma(d)$	$\log(1-\sigma(d))$	$\log r(\mathbf{x})$	$\log \sigma(d)$
f-GAN	d	$-\tilde{f}(d)$	$f'(r(\mathbf{x}))$	d
b-GAN	$f'(d)$	$f(d) - df'(d)$	$r(\mathbf{x})$	$df'(d) - f(d)$
Least-square GAN	$-(d-1)^2$	$-d^2$	$\frac{r(\mathbf{x})}{1+r(\mathbf{x})}$	$-(d-1)^2$
Generalized EBM (KL)	$-(d+\lambda)$	$-\exp(-d-\lambda)$	$-\log r(\mathbf{x}) - \lambda$	$\exp(-d-\lambda)$

Theorem 3.1. *If $h'(\cdot) > 0$, the dissipation rate $\partial \mathcal{F}(q_t)/\partial t$ for the KL divergence $\mathcal{F}(q_t) = \int \log\big(q_t/p\big) dq_t$ satisfies*

$$\frac{\partial \mathcal{F}(q_t)}{\partial t} \leq 0, \tag{10}$$

the equality is achieved if and only if $q_t = p$ and the marginal probability q_t of MonoFlow evolves to p as $t \to \infty$.

Theorem 3.1 shows that MonoFlow does not disturb the stationary measure of Eq. (4). The negative dissipation rate ensures that the curve $\{q_t\}_{t\geq 0}$ of MonoFlow always decreases the KL divergence with time. It is obvious that the marginal probability q_t finally evolves to the target p with time since the KL divergence converges to zero if $t \to \infty$, by the monotone convergence theorem. Note that we do not assume the target probability measure p is log-concave, the rate of convergence is not studied in this paper.

MonoFlow is obtained by transforming the log density ratio which arises from the Wasserstein gradient flow of the KL divergence. We can also formulate different deterministic particle evolution by considering Wasserstein gradient flows of general f-divergences,

$$\mathcal{D}_f(p||q) = \int f\left(\frac{p}{q}\right) dq, \tag{11}$$

where $f\colon \mathbb{R}^+ \to \mathbb{R}$ is a twice differentiable convex function with $f(1) = 0$.

Theorem 3.2. *The Wasserstein gradient flow of an f-divergence characterizes the evolution of particles in $\mathbb{R}^n$ by*

$$\mathrm{d}\mathbf{x}_t = r(\mathbf{x}_t)^2 f''\big(r(\mathbf{x}_t)\big) \nabla_{\mathbf{x}} \log r_t(\mathbf{x}_t) \mathrm{d}t, \quad \mathbf{x}_0 \sim q_0. \tag{12}$$

A similar result can also be derived with the reversed f-divergences $\mathcal{D}_f(q||p)$ used by Johnson & Zhang (2018); Gao et al. (2019); Ansari et al. (2021). Theorem 3.2 shows that the particle evolution of the Wasserstein gradient flow of f-divergences is a special instance of MonoFlow if a stronger condition than convexity is holding, i.e., $f''(\cdot) > 0$ which implies f is a strictly convex function. The rescaling factor is given by $h'(\log r) = r^2 f''(r) > 0$, this indicates once a curve $\{q_t\}_{t\geq 0}$ evolves with the time t in Wasserstein space to decrease an f-divergence whose $f''(\cdot) > 0$, it simultaneously decreases the KL divergence as well since the dissipation rate of MonoFlow is negative.

Furthermore, a corollary of Theorem 3.2 is that MonoFlow implicitly defines Wasserstein gradient flows of f-divergences via the increasing function h without specifying any strictly convex functions f.

Corollary 3.3. *For any continuously differentiable $h\colon \mathbb{R} \to \mathbb{R}$ with $h'(\cdot) > 0$, there exists a strictly convex and twice differentiable function $f\colon \mathbb{R}^+ \to \mathbb{R}$ with $f(1) = 0$ satisfying*

$$h(\log r) = rf'(r) - f(r), \tag{13}$$

MonoFlow associated with this increasing function h is the Wasserstein gradient flow of the functional $\mathcal{F}(q) = \mathcal{D}_f(p||q)$.

3.2. Practical Approximations of Density Ratios

We first discretize the ODE in Eq. (7) by the forward Euler method such that we obtain standard gradient ascent iterations with step size α and the index of the discretized time step k [2]:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha \nabla_{\mathbf{x}} h\big(\log r_k(\mathbf{x}_k)\big), \quad t_{k+1} = t_k + \alpha. \tag{14}$$

Therefore, we can sample initial particles $\mathbf{x}_0 \sim q_0$ and perform gradient ascent iterations by estimating the density ratio $r_k(\mathbf{x}) = p(\mathbf{x})/q_k(\mathbf{x})$ using samples from q_k and p. In order to enable a practical algorithm to obtain the time-dependent density ratio, we introduce a general framework that solves the following optimization problem,

$$\max_{d\in\mathcal{H}} \big\{\mathbb{E}_{\mathbf{x}\sim p}\left[\phi\big(d(\mathbf{x})\big)\right] + \mathbb{E}_{\mathbf{x}\sim q_k}\left[\psi\big(d(\mathbf{x})\big)\right]\big\}, \tag{15}$$

where $d\colon \mathbb{R}^n \to \mathbb{R}$ is a discriminator and $\mathcal{H}$ is a class of all measurable functions. ϕ and ψ are differentiable scalar functions upon design later. Similar to (Moustakides & Basioti, 2019), we show that if ϕ and ψ satisfy conditions

[2] For the sake of simplicity, we briefly replace t_k by its index k, though it is not rigorous.

in Lemma 3.4, the optimal d^* is a bijection of the density ratio between p and q_k.

Lemma 3.4. *Define* $\mathcal{T}(d(\mathbf{x})) := -\frac{\psi'(d(\mathbf{x}))}{\phi'(d(\mathbf{x}))}$. *If* ϕ *and* ψ *satisfy either of*

1. ϕ *is concave,* ψ *is strictly concave and the mapping* $\mathcal{T}$ *is a bijection.*
2. $\phi'(\cdot) > 0$ *and the mapping* $\mathcal{T}$ *is strictly increasing (also a bijection).*

Solving Eq. (15), the optimal d^* *satisfies*

$$d^*(\mathbf{x}) = \mathcal{T}^{-1}(r(\mathbf{x})), \quad r(\mathbf{x}) = \frac{p(\mathbf{x})}{q_k(\mathbf{x})}, \tag{16}$$

Remark: Note that two-sample density ratio estimations discard the density information from q_k. The functions $d(\mathbf{x})$, $r(\mathbf{x})$ only depend on $\mathbf{x}$ and they cannot capture the variability of q_k.

To this end, we can train d to solve the optimization problem in Eq. (15) and the density ratio is approximated by $\mathcal{T}(d(\mathbf{x}))$. For example, in a binary classification problem where $\phi(d) = \log\sigma(d)$ and $\psi(d) = \log(1-\sigma(d))$, we have $d^*(\mathbf{x}) = \log r(\mathbf{x})$ where its post-Sigmoid output $\sigma(d^*(\mathbf{x})) = r(\mathbf{x})/\big(1+r(\mathbf{x})\big) = p(\mathbf{x})/\big(p(\mathbf{x})+q_k(\mathbf{x})\big)$ is aligned with the Proposition 1 of Goodfellow et al. (2014). Other types of density ratio estimation can be found in Table 1 as they have been already used in GAN variants where q_k refers to the generator's distribution p_g. Specifically, f-GAN (Nowozin et al., 2016), Least-square GAN(Mao et al., 2017), Generalized EBM (Arbel et al., 2021) satisfy the condition 1 and b-GAN (Uehara et al., 2016) satisfies the condition 2 in Lemma 3.4.

In practice, since the change of $\mathbf{x}_k$ is sufficiently small at every step k, we can use a single discriminator $d(\mathbf{x})$ and perform a few gradient updates to solve Eq. (15) per iteration to approximate the time-dependent density ratio $r_k(\mathbf{x})$, which is identical to the GAN training.

3.3. Parameterization of the Discretized MonoFlow

The previous method directly pushes particles in the Euclidean space towards the target measure. We can use a neural network generator to mimic the distribution of these particles, i.e., train the generator to learn to draw samples.

We parameterize particles with a neural network generator g_{θ_k} that takes as input random noises $\mathbf{z} \sim p_{\mathbf{z}}$ and output particles $\mathbf{x}_k = g_{\theta_k}(\mathbf{z})$, we next move particles along the vector field of MonoFlow,

$$\mathbf{x}_{k+1} = g_{\theta_k}(\mathbf{z}) + \alpha\nabla_{\mathbf{x}} h\big(\log r_k(g_{\theta_k}(\mathbf{z}))\big) \tag{17}$$

Similar to (Wang & Liu, 2017), in order to encourage the generator to draw particles more similar to $\mathbf{x}_{k+1}$, we use one-step gradient descent to approximately solve $\min_\theta \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}}||g_\theta(\mathbf{z}) - \mathbf{x}_{k+1}||^2$, such that the generator's parameter is updated with learning rate β via

$$\theta_{k+1} = \theta_k + \beta\nabla_\theta \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}}\big[h\big(\log r_k(g_{\theta_k}(\mathbf{z}))\big)\big]. \tag{18}$$

In the continuous-time evolution, the associated infinitesimal change of the generator's parameter can be written as

$$\frac{\mathrm{d}\theta_t}{\mathrm{d}t} = \int \frac{\partial g_{\theta_t}(\mathbf{z})}{\partial\theta_t}\nabla_{\mathbf{x}} h\big(\log r_t(\mathbf{x}_t)\big)p_{\mathbf{z}}(\mathbf{z})\mathrm{d}\mathbf{z}, \tag{19}$$

where $\frac{\partial g_{\theta_t}(\mathbf{z})}{\partial\theta_t}$ is the Jacobian of the neural network generator. Consequently, if particles are generated via $\mathbf{x}_t = g_{\theta_t}(\mathbf{z})$, we have $\mathrm{d}\mathbf{x}_t = \frac{\partial g_{\theta_t}(\mathbf{z})}{\partial\theta_t}\mathrm{d}\theta_t$ by the chain rule, replace $\mathrm{d}\theta_t$ with Eq. (19), we obtain

$$\mathrm{d}\mathbf{x}_t = \mathbb{E}_{\mathbf{z}'\sim p_{\mathbf{z}}}\left[K_g^t(\mathbf{z},\mathbf{z}')\nabla_{\mathbf{x}} h\big(\log r_t(\mathbf{x}_t)\big)\right]\mathrm{d}t \tag{20}$$

where $K_g^t(\mathbf{z},\mathbf{z}') = \langle\frac{\partial g_{\theta_t}(\mathbf{z})}{\partial\theta_t}, \frac{\partial g_{\theta_t}(\mathbf{z}')}{\partial\theta_t}\rangle$ is the neural tangent kernel (NTK) (Jacot et al., 2018) defined by the generator. Eq. (20) realizes *Stein Variational Gradient Descent* (Liu & Wang, 2016; Franceschi et al., 2022) if h is an identity mapping.

3.4. A Unified Formulation of Divergence GANs

Based on the above derivation, we propose a general formulation for divergence GANs. We clarify that GANs can be treated with different objective functions for training discriminators and generators. All of these variants are algorithmic instantiations of the parameterized MonoFlow. The unified framework is summarized as: given a discriminator d and a generator g, the discriminator d learns to maximize

$$\mathbb{E}_{\mathbf{x}\sim p_{\mathrm{data}}}\left[\phi\big(d(\mathbf{x})\big)\right] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}}\left[\psi\big(d(g(\mathbf{z}))\big)\right], \tag{21}$$

where p_{data} refers to the data distribution. Next, we train the generator g to minimize

$$-\mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}}\left[h_{\mathcal{T}}\big(d(g(\mathbf{z}))\big)\right]. \tag{22}$$

where $h_{\mathcal{T}}(d) = h\big(\log(\mathcal{T}(d))\big)$ and h can be any strictly increasing function with $h'(\cdot) > 0$. We summarize some typical GAN variants in Table 1. We view adversarial training as maximizing Eq. (21) to obtain the density ratio estimator which suggests the vector field for MonoFlow and minimizing Eq. (22) as learning to parameterize MonoFlow corresponding to Eq. (18).

4. Understanding Adversarial Training via MonoFlow

The dominating understanding of adversarial training over GANs is that the generator learns to minimize the divergence

estimated from the discriminator. However, as pointed out in Section 1, the theoretical explanation of GANs and the practical algorithms are inconsistent. In this section, through the lens of MonoFlow, we will explain why this inconsistency does not prevent divergence GANs from achieving decent results and how it differs from a variational divergence minimization (VDM) problem.

4.1. Why the Adversarial Game Works?

In an adversarial game, the discriminator is trained to maximize the lower bound of f-divergences. This lower bound can be derived via the dual representation of f-divergences (Nguyen et al., 2010) between p_{data} and p_g,

$$\mathcal{D}_f(p_{\text{data}}||p_g) = \max_{d\in\mathcal{H}} \{ \underbrace{\mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}\big[d(\mathbf{x})\big] - \mathbb{E}_{\mathbf{x}\sim p_g}\big[\tilde{f}\big(d(\mathbf{x})\big)\big]}_{\text{lower bound}} \}, \tag{23}$$

where $r(\mathbf{x}) = p_{\text{data}}(\mathbf{x})/p_g(\mathbf{x})$ and $\tilde{f}(d) = \sup_{r\in\text{dom}f}\{rd - f(r)\}$ is the convex conjugate of $f(r)$. Note that for binary classification problems where we design specific ϕ and ψ, the corresponding optimization problem in Eq. (21) can be translated into an equivalent formulation as the above dual representation (Nowozin et al., 2016). Since the first term of the lower bound in Eq. (23) is irrelevant to p_g, the generator actually only learns to minimize the second term (vanilla loss),

$$\min_g -\mathbb{E}_{\mathbf{x}\sim p_g}\big[\tilde{f}\big(d(\mathbf{x})\big)\big] \tag{24}$$

Meanwhile, the generator can also alternatively minimize the heuristic non-saturated loss $-\mathbb{E}_{\mathbf{x}\sim p_g}\big[d(\mathbf{x})\big]$, which has been proven to work well in practice (Goodfellow et al., 2014; Nowozin et al., 2016). By the Fenchel duality, the optimal d^* is given by

$$d^* = f'(r) \tag{25}$$

with the equality $\tilde{f}(d^*) = rf'(r) - f(r)$. Fortunately, it can be simply verified that $f'(r)$ and $rf'(r) - f(r)$ are both strictly increasing functions of the density ratio (as well as the log density ratio) with positive derivatives if $f''(\cdot) > 0$ which implies strict convexity of f. Hence, adversarial training with the vanilla loss and the non-saturated loss both fall into the framework of MonoFlow which has theoretical guarantees.

4.2. Difference between Adversarial Training and Variational Divergence Minimization

In this part, we show that VDM differs from GAN training because it relies on the dependence between the discriminator (or the bijective density ratio) and the generator's distribution p_g. Metz et al. (2017) and Franceschi et al. (2022) also noticed that this dependence is discarded during the practical algorithms of GANs. We provide a further discussion of how this issue results in the difference between adversarial training and VDM as elaborated in the following.

The generator of GANs is a black box sampler without defining an explicit density function. However, in a VDM problem, the generator's output $\mathbf{x}$ can be reparameterized, e.g., as a Gaussian random variable where θ are its mean and scale, such that the generator g_θ defines a distribution via an explicit density function $p_g(\mathbf{x};\theta)$. In VDM, we are interested in minimizing an f-divergence with respect to p_g,

$$\min_{p_g} \mathcal{D}_f(p_{\text{data}}||p_g). \tag{26}$$

With the explicit density function $p_g(\mathbf{x};\theta)$, the density ratio $r(\mathbf{x},\theta) = p_{\text{data}}(\mathbf{x})/p_g(\mathbf{x};\theta)$ is a function depending on $\mathbf{x}$ as well as the generator's parameter θ to capture the variability of p_g. After integrating out $\mathbf{x}$, the f-divergence can be written as a cost function of θ.

$$\mathcal{D}_f(p_{\text{data}}||p_g) = \mathbb{E}_{\mathbf{x}\sim p_g}\big[f\big(r(\mathbf{x},\theta)\big)\big] = \text{Cost}(\theta) \tag{27}$$

Hence, the optimization over the functional space of p_g can be achieved by optimizing the parameter of the generator,

$$\min_{p_g} \mathcal{D}_f(p_{\text{data}}||p_g) \Longleftrightarrow \min_\theta \text{Cost}(\theta). \tag{28}$$

Since f is convex, by Jensen's inequality this cost is minimized at zero where $r(\mathbf{x},\theta)$ is a constant for each $\mathbf{x}$, meaning $p_g = p_{\text{data}}$ (see details in Appendix B). Similarly, we can rewrite the f-divergence under Fenchel-duality as

$$\text{Cost}(\theta) = \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}\big[d^*(\mathbf{x},\theta)\big] - \mathbb{E}_{\mathbf{x}\sim p_g}\big[\tilde{f}\big(d^*(\mathbf{x},\theta)\big)\big], \tag{29}$$

where $d^*(\mathbf{x},\theta) = f'(r(\mathbf{x},\theta))$.

The cost function in Eq. (29) is different from the objective of practical adversarial training in Eq. (23) since the first term of Eq. (29) has a dependence on the generator's parameter θ. This dependence is required in the theoretical adversarial game, see Eq. (4) of Goodfellow et al. (2014) as a special case of Eq. (29). However, in the practical algorithm, the density ratio estimator $r(\mathbf{x})$ or its bijection $d(\mathbf{x})$ are only functions of the sample $\mathbf{x}$. Plugging $r(\mathbf{x})$ or $d(\mathbf{x})$ into the f-divergence to replace $r(\mathbf{x},\theta)$ or $d^*(\mathbf{x},\theta)$, we can recover the approximated f-divergences but the approximated divergences can never be viewed as a cost function of θ anymore. Dropping out this dependence, the generator of GANs only minimizes the second term of the dual form of f-divergences or the non-saturated loss heuristically. This is the major disconnection between the theory and the practical algorithm over GANs.

Table 2. Comparisons on three density ratio models using different f and h: "✓" means the generator learns the data distribution and "✗" means it does not work. The evaluation is based on whether or not the parameter of the generator can finally approximate the target (μ_0, s_0), with visualizations as the complement. Visualization results are included in Appendix C.2. Code is available at `https://github.com/YiMX/MonoFlow`.

	if f convex	if h increases	$r(\mathbf{x}, \theta)$	$r(\mathbf{x}, \theta_{\text{de}})$	$r_{\text{GAN}}(\mathbf{x})$
KL	Yes	Yes	✓	✓	✓
Forward KL	Yes	No	✓	✗	✗
Chi-Square	Yes	No	✓	✗	✗
Hellinger	Yes	No	✓	✗	✗
Jensen-Shannon	Yes	No	✓	✗	✗
Exp	No	Yes	✗	✓	✓

4.3. Empirical Study of 2D Gaussians

We show how the dependence between the ratio model and the generator's parameter practically affects matching p_g to p_{data} on toy data sets. Let the data distribution be a Gaussian $p_{\text{data}} = N(\mu_0, \Sigma_0)$ where $\Sigma_0 = s_0^T s_0$. We start from the simplest form of a generator (reparameterization),

$$p_g : \mathbf{x}_\theta(\mathbf{z}) = g_\theta(\mathbf{z}) = \mu + s \cdot \mathbf{z}, \quad \mathbf{z} \sim N(0, I),$$

where $\theta = (\mu, s)$, μ is the mean and s is the scale matrix.

By assuming the generator and data distributions are Gaussians, we can define three density ratio models. The first model is $r(\mathbf{x}, \theta) = p_{\text{data}}(\mathbf{x})/p_g(\mathbf{x}; \theta)$, where the density ratio function depends on $\mathbf{x}$ and θ simultaneously. The second model is $r(\mathbf{x}, \theta_{\text{de}}) = p_{\text{data}}(\mathbf{x})/p_g(\mathbf{x}; \theta_{\text{de}})$, where θ_{de} means we detach the gradient of θ such that the second model cannot reflect the variability of p_g, i.e., the dependence between the ratio model and p_g is discarded (Metz et al., 2017; Franceschi et al., 2022). The third model is $r_{\text{GAN}}(\mathbf{x})$ where the density ratio is obtained by performing a single gradient update for the binary classification in standard GAN training. Note that $r(\mathbf{x}, \theta_{\text{de}})$ and $r_{\text{GAN}}(\mathbf{x})$ are only differentiable with $\mathbf{x}$.

We train the generator to minimize the following loss function with the above three density ratio models respectively (for $r_{\text{GAN}}(\mathbf{x})$, we use the standard bi-level optimization),

$$\min_\theta \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}}\big[f(r)\big]^3 \text{ or equivalently } \min_\theta -\mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}}\big[h(\log r)\big] \tag{30}$$

Given $f(r)$ we can rewrite it as a function of log density ratio $h(\log r) = -f(r)$. In this experiment, we consider five types of f-divergences with $f''(\cdot) > 0$ (expressions summarized in Appendix C.1). In addition, we study a strictly increasing function with $h'(\cdot) > 0$ given by $h(\log r) = \exp\big(1.5 \log r\big) = r^{1.5}$ where its $f(r) = -r^{1.5}$ is concave. The results are summarized in Table 2, which are consistent with our analysis that VDM is a convex problem requiring the ratio model $r(\mathbf{x}, \theta)$ to have a dependence on the generator that allows for the functional optimization, whereas $r(\mathbf{x}, \theta_{\text{de}})$ and $r_{\text{GAN}}(\mathbf{x})$ works with increasing functions with $h'(\cdot) > 0$ following the framework of MonoFlow.

The difference between $r(\mathbf{x}, \theta)$ and $r(\mathbf{x}, \theta_{\text{de}})$ is that they result in different gradient estimations. Using the ratio model $r(\mathbf{x}, \theta)$, the gradient to the parameter θ is evaluated by

$$\begin{aligned}\text{grad}(\theta) &= \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}}\big[\nabla_\theta f(r(\mathbf{x}_\theta(\mathbf{z}), \theta))\big] \\ &= \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}}\left[f'(r(\mathbf{x}_\theta(\mathbf{z}), \theta))\left(\frac{\partial r}{\partial \mathbf{x}_\theta}\frac{\partial \mathbf{x}_\theta}{\partial \theta} + \frac{\partial r}{\partial \theta}\right)\right],\end{aligned} \tag{31}$$

where backpropagation is applied to both $\mathbf{x}_\theta(\mathbf{z})$ and θ. However, to obtain the gradient estimation for the ratio model $r(\mathbf{x}, \theta_{\text{de}})$ or $r_{\text{GAN}}(\mathbf{x})$, backpropagation is only applied to the reparameterized sample $\mathbf{x}_\theta(\mathbf{z})$,

$$\begin{aligned}\text{grad}(\theta) &= \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}}\big[\nabla_\theta f(r(\mathbf{x}_\theta(\mathbf{z}), \theta_{\text{de}}))\big] \\ &= \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}}\left[f'(r(\mathbf{x}_\theta(\mathbf{z}), \theta_{\text{de}}))\left(\frac{\partial r}{\partial \mathbf{x}_\theta}\frac{\partial \mathbf{x}_\theta}{\partial \theta}\right)\right].\end{aligned} \tag{32}$$

Eq. (32) is also compatible with $r_{\text{GAN}}(\mathbf{x})$.

Remark: $r(\mathbf{x}, \theta_{\text{de}})$ can recover the true f-divergence, but minimizing this f-divergence has no effects except for KL divergence. Roeder et al. (2017) showed that the obtained gradient estimation under KL divergence is still unbiased if detaching gradient operator is applied.

5. Algorithimic Insights: Alternatives of Generator Loss

5.1. Effectiveness of Generator Losses via Vector Field Rescaling

In this part, we analyze the practical effectiveness of different types of generator losses. We provide a study for the discriminator trained under the binary classification problem since it outputs the log density ratio $d(\mathbf{x}) = \log r(\mathbf{x})$ (see Table 1). We consider five generator losses which are monotonically increasing functions of the log density ratio: **1).**

[3]The Monte Carlo objective is resampled from $p_{\mathbf{z}}$ via the reparameterization trick (Kingma & Welling, 2014; Rezende et al., 2014).

Vanilla loss: $h(d) = -\log(1-\sigma(d))$; **2). Non-saturated (NS) loss**: $h(d) = \log(\sigma(d))$. **3). Maximum likelihood estimation (MLE)**: $h(d) = \exp(d)$. **4). Logit loss**: $h(d) = d$. **5). Arcsinh loss**: $h(d) = \text{arcsinh}(d)$

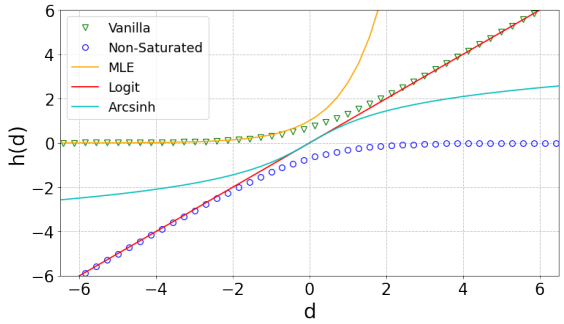


Figure 3. The plot of different generator losses as a function of d.

The plot of these functions is shown in Figure 3. It is known that the vanilla loss and the MLE loss suffer from the gradient vanishing problem in practice (Goodfellow, 2016). At the initial training steps, the generator is weak which means the associated p_g is far away from p_{data}. If the discriminator is too good, the estimated log ratio can be extremely small, i.e., $d(\mathbf{x}) = \log\left[p_{\text{data}}(\mathbf{x})/p_g(\mathbf{x})\right] \ll 0$, for $\mathbf{x} \sim p_g(\mathbf{x})$. We may observe in Figure 3, the curves of the vanilla loss and the MLE loss are fairly flat when $d(\mathbf{x}) \ll 0$, which means the derivative $h'(\cdot)$ is nearly zero. According to Eq. (7), such a rescaling scheme yields extremely small vector fields, resulting in the generator being trapped at the initial steps as the infinitesimal change of particles $\mathrm{d}\mathbf{x}_t \approx 0$. However, we may observe that the derivative of the vanilla loss and the ML loss deviates from zero if $d(x)$ is near zero. This suggests that these losses can work if the initial p_g is close to p_{data} where the estimated log ratio $d(\mathbf{x}) = \log\left[p_{\text{data}}(\mathbf{x})/p_g(\mathbf{x})\right]$ is not so small.

The NS loss, the logit loss and the arcsinh loss avoid gradient vanishing simply because they have non nearly zero derivatives when $d(\mathbf{x}) < 0$ despite that the NS loss is flat when $d(\mathbf{x}) > 0$. Since $p_{\text{data}}(\mathbf{x})/p_g(\mathbf{x}) \approx 0$ for $\mathbf{x} \sim p_g(\mathbf{x})$ at the beginning, the log ratio $d(\mathbf{x})$ gradually increases from a negative value to zero during the training. When $d(\mathbf{x})$ approaches zero, it means $p_g \approx p_{\text{data}}$ such that the generator has learned the data distribution.

5.2. An Embarrassingly Simple Trick to Fix Vanilla GAN on MNIST Generation

We have justified that MonoFlow can work with any strictly increasing mappings of the log density ratio and this mapping's derivative should deviate from zero when the $d(\mathbf{x}) < 0$ to better avoid too small rescaled vector fields (gradient vanishing). We show the effects of shifting the generator loss of the vanilla GAN left by adding a constant C to the Sigmoid function,

$$h(d) = -\log(1-\sigma(d+C)) \tag{33}$$

By adding a constant, we can obtain an increasing function whose derivative deviates from zero significantly, see Figure 4. The neural network architecture used here is DCGAN (Radford et al., 2015) and we follow the vanilla GAN framework where the log density ratio is obtained by logit output from the binary classifier and the model is trained with 15 epochs. The generated samples are shown in Figure 5. We observe that when $C = 3$ and $C = 5$, the generator losses in Eq. (33) start to work, i.e., generators output plausible fake images.

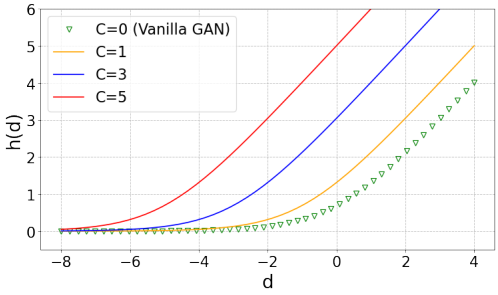
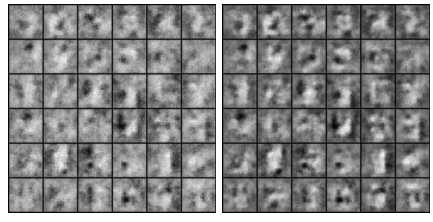


Figure 4. The plot of the vanilla losses by adding different Cs.



(a) $C = 0$ (b) $C = 1$

(c) $C = 3$ (d) $C = 5$

Figure 5. Generated samples with different Cs.

6. Related Works

Gradient Flow: Wasserstein gradient flows of f-divergences have been previously studied in deep generative modeling as a refinement approach to improve sample quality (Ansari et al., 2021). A close work to ours is (Gao et al., 2019) where the authors proposed to use gradient flows of f-divergences to refine fake samples output by the generator and the generator learns to minimize the

squared distance between the refined samples and the original fake samples. However, neither of the above reveals the equivalence between gradient flows and divergence GANs. Furthermore, MonoFlow is a more generalized framework to cover existing gradient flows of f-divergences and our method also applies to traditional loss designs as well as many other types of monotonically increasing functions.

IPM GANs: Our framework unifies divergence GANs since estimating a probability divergence is naturally related to density ratio estimation (Sugiyama et al., 2012). However, some variants of GANs are developed with Integral Probability Metric (IPM) (Sriperumbudur et al., 2009). For example, WGANs (Arjovsky et al., 2017; Gulrajani et al., 2017) estimate the Wasserstein-1 metric and then minimize this metric. While MonoFlow is associated with Wasserstein-2 metric, minimizing a functional in $\mathcal{P}(\mathbb{R}^d)$ naturally decreases Wasserstein-2 metric as well. Other types of IPM GANs are MMD GAN (Dziugaite et al., 2015) and Sobolev GAN (Mroueh et al., 2018a). Both of them have been interpreted as gradient flow approaches (Mroueh & Nguyen, 2021; Mroueh et al., 2018b) but associated with different vector fields. Franceschi et al. (2022) studied the NTK view on GANs given a vector field specified by a loss function of IPM but lacks connections to divergence GANs.

Diffusion Models: diffusion models (Ho et al., 2020; Song et al., 2021; Luo, 2022) are another line of generative modeling framework. This framework first perturbs data by adding noises with different scales to create a path $\{q_t\}_{t\geq 0}$ interpolating the data distribution and the noise distribution. Subsequently, the generative modeling is to reverse $\{q_t\}_{t\geq 0}$ as denoising. The similarity between MonoFlow and diffusion models is that they both involve particle evolution associated with different paths of marginal probabilities. However, the vector field of MonoFlow is obtained with the log density ratio that must be corrected per iteration by gradient update, whereas diffusion models directly estimate vector fields by time-dependent neural networks and they are straightforward particle methods.

7. Conclusions

MonoFlow provides a unified framework to explain why and how adversarial training of divergence GANs works. The mechanism of adversarial training may not be as adversarial as we used to think. It instead simulates an ODE system. The bi-level step of adversarial can be regarded as first estimating the vector field, then updating the generator as learning to draw particles of the ODE, a process we call parameterizing MonoFlow. All divergence GANs discussed in this paper are unified under our framework. They all are different methods of estimating the bijection of the log density ratio and then mapping the log density ratio by different monotonically increasing functions. , The methodological development closely matches our theoretical framework. The limitation of this paper is that our framework does not cover IPM GANs since these variants give a vector field that is different from the gradient of log density ratios. We leave it as a future work.

References

Ambrosio, L., Gigli, N., and Savaré, G. *Gradient flows: in metric spaces and in the space of probability measures*. Springer Science & Business Media, 2008.

Ansari, A. F., Ang, M. L., and Soh, H. Refining deep generative models via discriminator gradient flow. *In ICLR*, 2021.

Arbel, M., Zhou, L., and Gretton, A. Generalized energy based models. *In ICLR*, 2021.

Arjovsky, M., Chintala, S., and Bottou, L. Wasserstein generative adversarial networks. *In ICML*, 2017.

Brock, A., Donahue, J., and Simonyan, K. Large scale gan training for high fidelity natural image synthesis. *In ICLR*, 2018.

Dziugaite, G. K., Roy, D. M., and Ghahramani, Z. Training generative neural networks via maximum mean discrepancy optimization. *arXiv preprint arXiv:1505.03906*, 2015.

Franceschi, J.-Y., De Bézenac, E., Ayed, I., Chen, M., Lamprier, S., and Gallinari, P. A neural tangent kernel perspective of gans. *In ICML*, 2022.

Gao, Y., Jiao, Y., Wang, Y., Wang, Y., Yang, C., and Zhang, S. Deep generative learning via variational gradient flow. *In ICML*, 2019.

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

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. *In NeurIPS*, 2014.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. Improved training of wasserstein gans. *In NeurIPS*, 2017.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *In NeurIPS*, 2017.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. *In NeurIPS*, 2020.

Hyvärinen, A. and Dayan, P. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(4), 2005.

Jabbar, A., Li, X., and Omar, B. A survey on generative adversarial networks: Variants, applications, and training. *ACM Computing Surveys (CSUR)*, 54(8):1–49, 2021.

Jacot, A., Gabriel, F., and Hongler, C. Neural tangent kernel: Convergence and generalization in neural networks. *In NeurIPS*, 2018.

Johnson, R. and Zhang, T. Composite functional gradient learning of generative adversarial models. In *ICML*, 2018.

Jordan, R., Kinderlehrer, D., and Otto, F. The variational formulation of the fokker–planck equation. *SIAM journal on mathematical analysis*, 29(1):1–17, 1998.

Karras, T., Laine, S., and Aila, T. A style-based generator architecture for generative adversarial networks. *In CVPR*, 2019.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *In ICLR*, 2014.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

Ledig, C., Theis, L., Huszár, F., Caballero, J., Cunningham, A., Acosta, A., Aitken, A., Tejani, A., Totz, J., Wang, Z., et al. Photo-realistic single image super-resolution using a generative adversarial network. *In CVPR*, 2017.

Liu, Q. and Wang, D. Stein variational gradient descent: A general purpose bayesian inference algorithm. *In NeurIPS*, 2016.

Liu, Z., Luo, P., Wang, X., and Tang, X. Deep learning face attributes in the wild. *In ICCV*, 2015.

Luo, C. Understanding diffusion models: A unified perspective. *arXiv preprint arXiv:2208.11970*, 2022.

Mao, X., Li, Q., Xie, H., Lau, R. Y., Wang, Z., and Paul Smolley, S. Least squares generative adversarial networks. In *ICCV*, 2017.

Metz, L., Poole, B., Pfau, D., and Sohl-Dickstein, J. Unrolled generative adversarial networks. *In ICLR*, 2017.

Mohamed, S., Rosca, M., Figurnov, M., and Mnih, A. Monte carlo gradient estimation in machine learning. arxiv e-prints, page. *arXiv preprint arXiv:1906.10652*, 2019.

Moustakides, G. V. and Basioti, K. Training neural networks for likelihood/density ratio estimation. *arXiv preprint arXiv:1911.00405*, 2019.

Mroueh, Y. and Nguyen, T. On the convergence of gradient descent in gans: Mmd gan as a gradient flow. *In AISTATS*, 2021.

Mroueh, Y., Li, C.-L., Sercu, T., Raj, A., and Cheng, Y. Sobolev gan. *In ICLR*, 2018a.

Mroueh, Y., Sercu, T., and Raj, A. Sobolev descent: Variational transport of distributions via advection. *Private communication. Apr*, 2018b.

Nguyen, X., Wainwright, M. J., and Jordan, M. I. Estimating divergence functionals and the likelihood ratio by convex risk minimization. *IEEE Transactions on Information Theory*, 56(11):5847–5861, 2010.

Nowozin, S., Cseke, B., and Tomioka, R. f-gan: Training generative neural samplers using variational divergence minimization. *In NeurIPS*, 2016.

Otto, F. The geometry of dissipative evolution equations: the porous medium equation. *Communications in Partial Differential Equations*, 26:101–174, 2001.

Qin, J. Inferences for case-control and semiparametric two-sample density ratio models. *Biometrika*, 85(3):619–630, 1998.

Radford, A., Metz, L., and Chintala, S. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

Rezende, D. J., Mohamed, S., and Wierstra, D. Stochastic backpropagation and approximate inference in deep generative models. *In ICML*, 2014.

Risken, H. and Risken, H. *Fokker-planck equation*. Springer, 1996.

Roeder, G., Wu, Y., and Duvenaud, D. K. Sticking the landing: Simple, lower-variance gradient estimators for variational inference. *In NeurIPS*, 2017.

Song, Y. and Ermon, S. Generative modeling by estimating gradients of the data distribution. *In NeurIPS*, 2019.

Song, Y. and Kingma, D. P. How to train your energy-based models. *arXiv preprint arXiv:2101.03288*, 2021.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. Score-based generative modeling through stochastic differential equations. *In ICLR*, 2021.

Sriperumbudur, B. K., Fukumizu, K., Gretton, A., Schölkopf, B., and Lanckriet, G. R. On integral probability metrics, ϕ-divergences and binary classification. *arXiv preprint arXiv:0901.2698*, 2009.

Sugiyama, M., Suzuki, T., and Kanamori, T. *Density ratio estimation in machine learning*. Cambridge University Press, 2012.

Uehara, M., Sato, I., Suzuki, M., Nakayama, K., and Matsuo, Y. Generative adversarial nets from a density ratio estimation perspective. *arXiv preprint arXiv:1610.02920*, 2016.

Vincent, P. A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674, 2011.

Wang, D. and Liu, Q. Learning to draw samples: With application to amortized mle for generative adversarial learning. *In ICLR*, 2017.

Zhang, H., Xu, T., Li, H., Zhang, S., Wang, X., Huang, X., and Metaxas, D. N. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. *In ICCV*, 2017.

A. Appendix

A.1. Proof of Theorem 3.1

The dissipation rate: For any curve $\{q_t\}_{t\geq 0}$ evolving according to the vector field $\{v_t\}_{t\geq 0}$, the dissipation rate of the functional (Ambrosio et al., 2008) is given as

$$\frac{\partial \mathcal{F}(q_t)}{\partial t} = \int \langle \nabla_{W_2} \mathcal{F}(q_t), v_t \rangle \mathrm{d}q_t. \tag{34}$$

If the functional is the KL divergence, we have the associated Wasserstein gradient $\nabla_{W_2}\mathcal{F}(q_t) = \nabla_{\mathbf{x}} \log(q_t/p)$. Recall that the vector filed of MonoFlow in Eq. (7) is

$$v_t = h'(\log r_t)\nabla_{\mathbf{x}} \log r_t, \quad r_t = \log(p/q_t) \tag{35}$$

Therefore, if $h'(\cdot) > 0$, MonoFlow dissipates the KL divergence with the rate

$$\frac{\partial \mathcal{F}(q_t)}{\partial t} = \mathbb{E}_{\mathbf{x}\sim q_t}\left[-h'\big(\log r_t(\mathbf{x})\big) \left\| \nabla_{\mathbf{x}} \log \frac{q_t(\mathbf{x})}{p(\mathbf{x})} \right\|^2 \right] \leq 0. \tag{36}$$

It is obvious that $\partial \mathcal{F}(q_t)/\partial t = 0$ implies

$$\mathbb{E}_{\mathbf{x}\sim q_t}\left[\left\| \nabla_{\mathbf{x}} \log q_t(\mathbf{x}) - \nabla_{\mathbf{x}} \log p(\mathbf{x}) \right\|^2 \right] = 0, \tag{37}$$

where the left hand side is the Fisher divergence $\mathcal{D}_{\mathrm{FI}}(q_t||p)$. If p is a well defined proper probability measure on $\mathbb{R}^n$, $\mathcal{D}_{\mathrm{FI}}(q_t||p)$ attains zero if and only if $q_t = p$.

Hence, MonoFlow always decreases the KL divergence with the time when $q_t \neq p$. By the monotone convergence theorem, i.e., any decreasing sequence converges to its infimum, thus the KL divergence converges to zero if $t \to \infty$, which indicates q_t evolves to p.

A.2. Proof of Theorem 3.2

Define the functional $\mathcal{F}(q)$ of f-divergences as

$$\mathcal{F}(q) = \mathcal{D}_f(p||q) = \int f\left(\frac{p}{q}\right)(\mathbf{x}) q(\mathbf{x}) \mathrm{d}\mathbf{x}. \tag{38}$$

where $f\colon \mathbb{R}^+ \to \mathbb{R}$ is a convex function and we may further assume that f is twice differentiable.

Let $\phi \in \mathcal{P}(\mathbb{R}^n)$ be a test function, the first variation (functional derivative) $\frac{\delta \mathcal{F}}{\delta q}$ is defined as

$$\begin{aligned}
\int \frac{\delta \mathcal{F}}{\delta q}(\mathbf{x})\phi(\mathbf{x})\mathrm{d}\mathbf{x} &= \lim_{\epsilon\to 0} \frac{\mathcal{F}(q+\epsilon\phi) - \mathcal{F}(q)}{\epsilon} \\
&= \frac{d}{d\epsilon}\mathcal{F}(q+\epsilon\phi)\Big|_{\epsilon=0} \\
&= \frac{d}{d\epsilon}\int f\left(\frac{p}{q+\epsilon\phi}\right)(\mathbf{x})\big(q(\mathbf{x}) + \epsilon\phi(\mathbf{x})\big)\mathrm{d}\mathbf{x}\Big|_{\epsilon=0} \\
&= \int \left\{ f\left(\frac{p}{q+\epsilon\phi}\right)(\mathbf{x})\phi(\mathbf{x}) - f'\left(\frac{p}{q+\epsilon\phi}\right)(\mathbf{x}) \frac{p(\mathbf{x})\phi(\mathbf{x})}{q(\mathbf{x}) + \epsilon\phi(\mathbf{x})} \right\} \mathrm{d}\mathbf{x}\Big|_{\epsilon=0} \\
&= \int \left\{ f\left(\frac{p}{q}\right) - f'\left(\frac{p}{q}\right)\frac{p}{q} \right\}(\mathbf{x})\phi(\mathbf{x})\mathrm{d}\mathbf{x}.
\end{aligned} \tag{39}$$

Thus,

$$\frac{\delta \mathcal{F}}{\delta q} = f(r) - r f'(r), \quad \text{where} \quad r = \frac{p}{q}. \tag{40}$$

Recall that the Wasserstein gradient of $\mathcal{F}(q)$ is the Euclidean gradient of the first variation, we have

$$\nabla_{W_2}\mathcal{F}(q) = \nabla_{\mathbf{x}}\frac{\delta\mathcal{F}}{\delta q} = -rf''(r)\nabla_{\mathbf{x}}r. \tag{41}$$

The corresponding vector field is given by the negative Euclidean gradient, see Section 3, therefore the particle flow ODE of f-divergences can be written as

$$\mathrm{d}\mathbf{x} = -\nabla_{\mathbf{x}}\frac{\delta\mathcal{F}}{\delta q}(\mathbf{x})\mathrm{d}t = r(\mathbf{x})f''(r(\mathbf{x}))\nabla_{\mathbf{x}}r(\mathbf{x})\mathrm{d}t = r(\mathbf{x})^2 f''(r(\mathbf{x}))\nabla_{\mathbf{x}}\log r(\mathbf{x})\mathrm{d}t. \tag{42}$$

A.3. Proof of Corollary 3.3

According to the existence theorem of primitive functions (antiderivative), any continuous scalar function must have a primitive function and this primitive function is also continuous. Hence, if $f''(r) = h'(\log r)/r^2$ where h is continuously differentiable, we have $f''(r)$ is continuous such that $f'(r)$ and $f(r)$ both exist. We also have $h'(\log r) > 0 \Longrightarrow f''(r) > 0$, this indicates f is strictly convex.

Hence, given a differentiable h with $h'(\cdot) > 0$, there must exist a strictly convex function f such that $f(1) = 0$ (primitive functions differ in constants) where its second derivative is specified by $f''(r) = h'(\log r)/r^2$.

We can let $h(\log r) = rf'(r) - f(r) + C$, apparently $h(\log r) = rf'(r) - f(r) + C \Longleftrightarrow h'(\log r) = r^2 f''(r)$, this defines the particle evolution of Wasserstein gradient flows,

$$\mathrm{d}\mathbf{x}_t = h'\big(\log r_t(\mathbf{x}_t)\big)\nabla_{\mathbf{x}}\log r_t(\mathbf{x}_t)\mathrm{d}t = r(\mathbf{x}_t)^2 f''\big(r(\mathbf{x}_t)\big)\nabla_{\mathbf{x}}\log r_t(\mathbf{x}_t)\mathrm{d}t. \tag{43}$$

Without the loss of generality, we can let $h(\log r) = rf'(r) - f(r)$.

A.4. Proof of Lemma 3.4

This proof is adapted from Lemma 1 and 2 of Moustakides & Basioti (2019). Given the optimization problem

$$\max_{d\in\mathcal{H}} \mathbb{E}_{\mathbf{x}\sim p}\left[\phi\big(d(\mathbf{x})\big)\right] + \mathbb{E}_{\mathbf{x}\sim q}\left[\psi\big(d(\mathbf{x})\big)\right], \tag{44}$$

where $\mathcal{H}$ is a class of all measurable functions. We rewrite it as

$$\begin{aligned}&\max_{d\in\mathcal{H}} \mathbb{E}_{\mathbf{x}\sim q}\left[\frac{p(\mathbf{x})}{q(\mathbf{x})}\phi\big(d(\mathbf{x})\big) + \psi\big(d(\mathbf{x})\big)\right]\\ &= \mathbb{E}_{\mathbf{x}\sim q}\left[\max_{d\in\mathcal{H}}\left\{\frac{p(\mathbf{x})}{q(\mathbf{x})}\phi\big(d(\mathbf{x})\big) + \psi\big(d(\mathbf{x})\big)\right\}\right],\end{aligned} \tag{45}$$

we apply the interchange of maximum and integral because the integral operator is independent of d. Since the maximum is holding for every fixed $\mathbf{x}$, thus we let the derivative $\frac{\partial}{\partial d(\mathbf{x})}\left[\frac{p(\mathbf{x})}{q(\mathbf{x})}\phi\big(d(\mathbf{x})\big) + \psi\big(d(\mathbf{x})\big)\right] = 0$, we have the optimal d^*, the abbreviation of $d^*(\mathbf{x})$, to satisfy

$$r\phi'(d^*) + \psi'(d^*) = 0, \quad r(\mathbf{x}) = \frac{p(\mathbf{x})}{q(\mathbf{x})} > 0 \tag{46}$$

Furthermore, we need to discuss under what sufficient conditions, d^* is the unique maximizer for the above problem. Denote $l(d) = r\phi(d) + \psi(d)$, in order to ensure that d^* is the unique maximizer, $l'(d)$ should satisfy

$$l'(d) > 0, \forall d < d^* \text{ and } l'(d) < 0, \forall d > d^*. \tag{47}$$

We define the mapping $\mathcal{T}(d) := -\frac{\psi'(d)}{\phi'(d)}$ and summarize two **sufficient conditions** as:

1. ϕ is concave, ψ is strictly concave and the resulting mapping $\mathcal{T}$ is a bijection.
2. $\phi'(\cdot) > 0$ and the resulting mapping $\mathcal{T}$ is a strictly increasing mapping (also a bijection).

It is obvious that if $\mathcal{T}$ is a bijection, $d^* = \mathcal{T}^{-1}(r)$ is the root of Eq. (46).

For condition 1, since ϕ is concave and ψ is strictly concave, the linear combination $l(d)$ is strictly concave which satisfies Eq. (47). Therefore, d^* is the unique maximizer.

For condition 2, we can write $l'(d) = [\mathcal{T}(d^*) - \mathcal{T}(d)]\phi'(d)$. Since $\mathcal{T}$ is a strictly increasing mapping and d^* is the maximizer, we have $\mathcal{T}(d^*) - \mathcal{T}(d) > 0$ for $d < d^*$ and $\mathcal{T}(d^*) - \mathcal{T}(d) < 0$ for $d > d^*$. Hence $l'(d)$ satisfies the condition stated in Eq. (47).

In Table 2, b-gan satisfies condition 2 and the rest of the divergence GANs satisfy condition 1.

Some examples:

- For binary classification, $\phi(d) = \log \sigma(d)$ and $\psi(d) = \log(1 - \sigma(d))$, $r(\mathbf{x}) = \exp(d^*(\mathbf{x}))$.
- Fenchel-duality, $\phi(d) = d$, $\psi(d) = -\tilde{f}(d)$, $r(\mathbf{x}) = \tilde{f}'\big(d^*(\mathbf{x})\big)$ where the convex conjugate is $\tilde{f}(d) = \sup_{r \in \mathrm{dom} f}\{rd - f(r)\}$
- For least-square GAN, $\phi(d) = -(d-1)^2, \psi(d) = -d^2$, $r(\mathbf{x}) = \frac{d^*(\mathbf{x})}{1 - d^*(\mathbf{x})}$

B. Variational Divergence Minimization

Given an f-divergence $\mathcal{D}_f(p||q)$ where p is the fixed target distribution, variational divergence minimization finds an approximating distribution q via the functional optimization

$$\min_q \mathcal{D}_f(p||q). \tag{48}$$

If q is represented by a parametric model with an explicit density function $q(\mathbf{x}; \theta)$, the f-divergence can be written as

$$\text{Cost}(\theta) = \mathcal{D}_f(p||q) = \int f\big(r(\mathbf{x}, \theta)\big) q(\mathbf{x}; \theta) \mathrm{d}\mathbf{x} \tag{49}$$

where $r(\mathbf{x}, \theta) = p(\mathbf{x})/q(\mathbf{x}; \theta)$ explicitly depends on $\mathbf{x}$ and θ. The f-divergence becomes a cost function of the parameter θ because $\mathbf{x}$ is integrated out. A typical example is in standard variational inference where q is a parametric Gaussian distribution such that we know the exact density function $q(\mathbf{x}; \theta)$.

Hence, the functional optimization problem degenerates into an optimization problem over the parameter space,

$$\min_q \mathcal{D}_f(p||q) \Longleftrightarrow \min_\theta \text{Cost}(\theta). \tag{50}$$

Since f is convex, we can apply Jensen's inequality,

$$\mathcal{D}_f(p||q) = \int f\left(\frac{p(\mathbf{x})}{q(\mathbf{x}; \theta)}\right) q(\mathbf{x}; \theta) \mathrm{d}\mathbf{x} \geq f\left(\int \frac{p(\mathbf{x})}{q(\mathbf{x}; \theta)} q(\mathbf{x}; \theta) \mathrm{d}\mathbf{x}\right) = f(1) = 0, \tag{51}$$

Jensen's inequality indicates that $\mathcal{D}_f(p||q) = 0$ if and only if $p(\mathbf{x})/q(\mathbf{x}; \theta)$ is a constant, such that we have $q = p$.

Therefore, we can minimize the cost function $\text{Cost}(\theta)$ to approximate p with q. Solving the optimization problem in Eq. (50) requires Monte Carlo gradient estimation, we can apply the reparameterization trick (Kingma & Welling, 2014; Rezende et al., 2014) to solve

$$\min_\theta \text{Cost}(\theta) = \min_\theta \mathbb{E}_{\mathbf{z} \sim p_\mathbf{z}}\big[f\big(r(\mathbf{x}_\theta(\mathbf{z}), \theta)\big)\big], \text{ where } \mathbf{x}_\theta(\mathbf{z}) = g_\theta(\mathbf{z}). \tag{52}$$

g_θ is the Gaussian generator parameterized by θ. The associated gradient estimation is obtained by the chain rule,

$$\mathbb{E}_{\mathbf{z} \sim p_\mathbf{z}}\big[\nabla_\theta f(r(\mathbf{x}_\theta(\mathbf{z}), \theta))\big] = \mathbb{E}_{\mathbf{z} \sim p_\mathbf{z}}\big[f'(r(\mathbf{x}_\theta(\mathbf{z}), \theta))(\frac{\partial r}{\partial \mathbf{x}_\theta}\frac{\partial \mathbf{x}_\theta}{\partial \theta} + \frac{\partial r}{\partial \theta})\big] \tag{53}$$

if we detach the gradient of θ in the ratio model, the gradient estimation is distorted,

$$\mathbb{E}_{\mathbf{z} \sim p_\mathbf{z}}\big[\nabla_\theta f(r(\mathbf{x}_\theta(\mathbf{z}), \theta_{\text{de}}))\big] = \mathbb{E}_{\mathbf{z} \sim p_\mathbf{z}}\big[f'(r(\mathbf{x}_\theta(\mathbf{z}), \theta_{\text{de}}))(\frac{\partial r}{\partial \mathbf{x}_\theta}\frac{\partial \mathbf{x}_\theta}{\partial \theta})\big]. \tag{54}$$

Alternatively, we can apply score function gradient estimation, see (Mohamed et al., 2019) for more details.

C. Experiments

All codes are available at `https://github.com/YiMX/MonoFlow`.

C.1. Experiment Details for Section 4.3

Table 3. Explicit forms of f and h

	$f(r)$	$h(u), u = \log r$
KL	$-\log r$	u
Forward KL	$r \log r$	$-u \exp(u)$
Chi-Square	$(r-1)^2$	$-(\exp(u)-1)^2$
Hellinger	$(\sqrt{r}-1)^2$	$-(\sqrt{\exp(u)}-1)^2$
Jensen-Shannon (GAN)	$r \log \frac{2r}{1+r} + \log \frac{2}{1+r}$	$-\exp(u) \log \frac{2\exp(u)}{1+\exp(u)} - \log \frac{2}{1+\exp(u)}$
Exp	$-\exp(1.5 \log r)$	$\exp(1.5u)$

In Figure 6, we can observe that the Exp function is concave under $f(r)$. KL and Exp are increasing functions under $h(u)$.

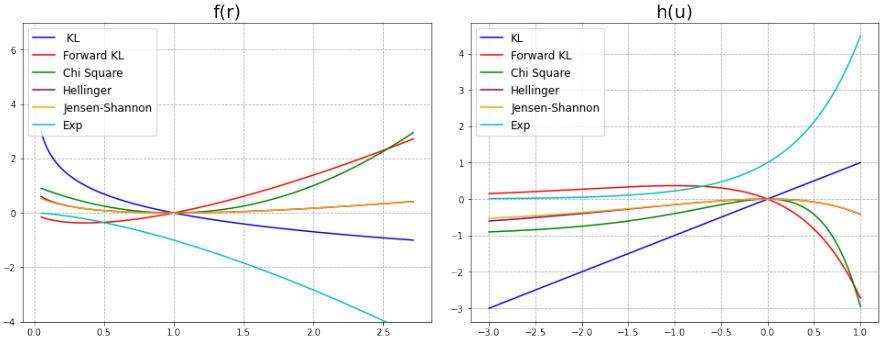
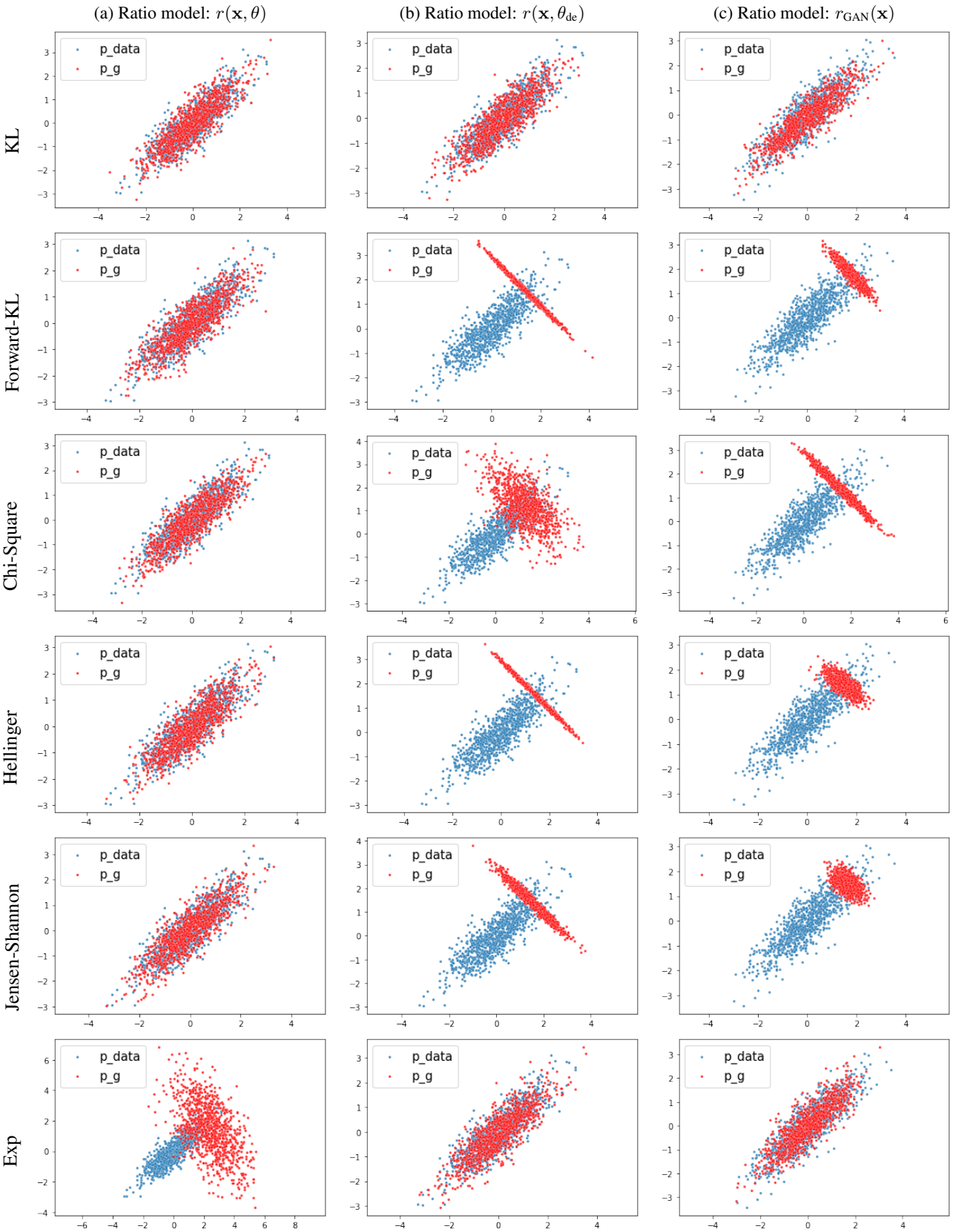


Figure 6. Function plots of $f(r)$ and $h(u)$

The generator is initialized at: $N\left[\begin{pmatrix}1.0\\1.0\end{pmatrix}, \begin{pmatrix}1.00 & 0.00\\0.00 & 1.00\end{pmatrix}\right]$ and the target distribution is : $N\left[\begin{pmatrix}0.0\\0.0\end{pmatrix}, \begin{pmatrix}1.00 & 0.80\\0.80 & 0.89\end{pmatrix}\right]$

$r_{\text{GAN}}(\mathbf{x})$ uses a simple 2-layer discriminator with Leaky ReLU activation that has logit output as the log density ratio.

C.2. Visualization Results for Section 4.3



C.3. FID Scores of Different Generator Losses

In this section, we demonstrate that different generator losses can achieve equal performances on image generations by evaluating the FID scores (Heusel et al., 2017). The discriminator are trained with the original GAN objective (Goodfellow et al., 2014) where the optimal $d^*(\mathbf{x}) = \log r(\mathbf{x})$ and the Least-square GAN (Mao et al., 2017) objective where the optimal $d^*(\mathbf{x}) = r(\mathbf{x})/(1 + r(\mathbf{x}))$, see Table 2. We use MNIST, CIFAR-10 (Krizhevsky et al., 2009) and Celeb-A (Liu et al., 2015) datasets in this experiment. Models are trained using the training sets and the FID scores are evaluated on the test sets as shown in Table 4 and Table 5. The neural network structures are modified from (Radford et al., 2015).

Table 4. FID scores with different generator losses where $d^*(\mathbf{x}) = \log r(\mathbf{x})$

$h(d(\mathbf{x}))$	MNIST	CIFAR-10	Celeb-A
$\log \sigma(d(\mathbf{x}))$:	4.3309	21.2980	20.6109
$d(\mathbf{x})$:	4.4631	20.7969	21.2240
$\arcsin(d(\mathbf{x}))$:	4.4893	21.2533	21.0077

Table 5. FID scores with different generator losses where $d^*(\mathbf{x}) = \frac{r(\mathbf{x})}{1+r(\mathbf{x})}$

$h(d(\mathbf{x}))$	MNIST	CIFAR-10	Celeb-A
$-(d(\mathbf{x})-1)^2$:	5.0808	23.8330	21.6787
$d(\mathbf{x})$:	4.6000	22.7969	20.5231
$\arcsin(d(\mathbf{x}))$:	4.5525	23.4698	22.1024