A GAME OF SNAKES AND GANS

Siddarth Asokan[1]*, Fatwir Sheikh Mohammed* [2]*, and Chandra Sekhar Seelamantula*[3]

[1] Robert Bosch Center for Cyber-Physical Systems, Indian Institute of Science, Bengaluru-12, India
[2] University of Washington, Seattle, WA 98195, United States
[3]Department of Electrical Engineering, Indian Institute of Science, Bengaluru-12, India
{[1]*siddartha,*[3]*css*}*@iisc.ac.in,* [2]*fatwir@uw.edu*

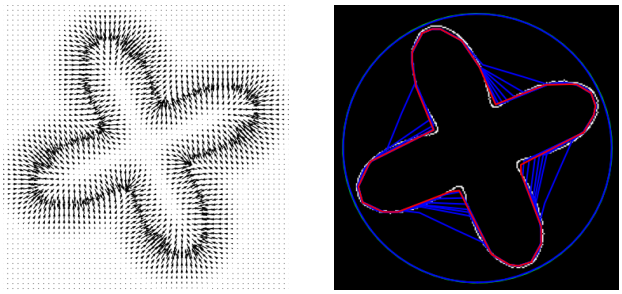
ABSTRACT

Generative adversarial networks (GANs) comprise generator and discriminator networks trained adversarially to learn the underlying distribution of a dataset. Recently, we have shown that the optimal GAN discriminator can be obtained in closed-form as the solution to the Poisson partial differential equation (PDE). While existing approaches either train a network or solve the PDE in closed-form, we propose training the generator through the gradient field of the optimal discriminator. In this paper, we establish a connection between active contour models (snakes) and GANs. We evolve a set of *snake* points over the gradient field of radial basis function (RBF) Coulomb GAN. The generator is then trained to follow the trajectory of the *snake*. The proposed approach benefits from both the sample diversity seen in flow-based approaches and the fast sampling capability of GANs. Experimental validation on 2-D synthetic data shows that the proposed approach leads to accelerated convergence, compared against the baseline approaches that either employ network or kernel-based discriminators.

***Index Terms*—** Generative adversarial networks, active contours models (snakes), Laplacian operator, optimal discriminator.

1. INTRODUCTION

Generative adversarial networks (GANs) [1] constitute a framework where two networks, namely the generator G, and the discriminator D, are trained in an adversarial manner to approximate the distribution of a target dataset p_d. The generator is a mapping that transforms noise input, typically Gaussian, into *fake* samples that follow a distribution p_g. The discriminator is tasked with learning a classifier between the samples drawn from p_d and p_g. The discriminator output is in turn used to improve the quality of generation of the fakes. In the *vanilla GAN* formulation of Goodfellow *et al.* [1], and subsequent variants such as the least-squares GAN (LSGAN) [2] or the f-GAN [3], the discriminator approximates a divergence metric between p_d and p_g, such as the Jensen-Shannon divergence in vanilla GAN. Essentially, the discriminator is trained to learn a divergence between p_g and p_d, while the generator is trained to minimize it. The numerical instability of divergence-minimizing GANs [4] gave rise to *integral probability metric* (IPM) minimizing GANs, such as the Wasserstein GAN (WGAN) [5]. In IPM-GANs, the discriminator approximates a distance metric, and the optimal generator minimizes the chosen distance. WGANs consider the Wasserstein-1 distance, which constrains the discriminator to be Lipschitz-1. Subsequent works [6–10] have shown that including first-order gradient-based regularization on the discriminator results in stabler training.



(a) (b)

Fig. 1: (a) The gradient vector field associated with an object boundary. The arrows point towards the edges of the object (the minimum-energy solution). (b) Evolution of a snake curve from the initial circle to the target contour that learns the object boundary.

GANs and PDEs: While most GAN algorithms aim to optimize a network for the discriminator, variants such as Sobolev GANs [11] or Coulomb GANs [12, 13] can be seen as being the solutions to partial differential equations (PDEs). In Sobolev GAN, the energy of the gradient in the discriminator is constrained, resulting in an optimal discriminator that solves a Fokker-Planck equation, while the Coulomb GAN discriminator corresponds to the solution to a Poisson PDE [13]. In Coulomb GANs [12], the discriminator network is trained to approximate the potential field induced by p_d and p_g. On the other hand, in radial basis function (RBF) Coulomb GANs [13], the optimal GAN discriminator is evaluated in closed-form, while the generator is trained using the standard WGAN loss. In this work, we consider the RBF-Coulomb GAN formulation [13], wherein the optimal discriminator can be expressed through an RBF expansion.

GANs and Gradient Flow: Along a parallel vertical, diffusion models [14–16] and flow-based generative learning techniques [17–19] replace training a generator network with updating an initial set of points trough the gradient fields induced by the score (gradient of the logarithm of the density) of the data distribution. While these approaches lead to superior image quality, they are slower to sample, requiring hundreds to thousands of steps of Langevin flow to generate an image [16]. Flow-based update schemes have also been explored in GAN models in *Sobolev descent* [20, 21], where the gradient field of a Sobolev critic is used to update the noise particles. FlowGANs [22] link GANs to normalizing flows through an invertible generator that maximizes the likelihood of the generated samples.

[1] Corresponding Author. Siddarth Asokan is funded by the Qualcomm Innovation Fellowship, and the Robert Bosch Center for Cyber-Physical Systems Ph.D. Fellowship.

[2] Work done during internship at Spectrum Lab, Department of Electrical Engineering, Indian Institute of Science, Bangalore.

1.1. Our Contributions

We present a new approach, wherein the gradient field of the optimal discriminator is leveraged to train a generator. We consider the RBF-Coulomb GAN discriminator [12, 13], and derive the gradient field associated with it. We propose a novel GAN training paradigm, which alternates between generator network optimization and gradient vector flow with snakes. Given the generator and data distributions, we iterate a snake model in the neighborhood of the generated samples. The flow is then used to train a generator to learn the trajectory that minimizes the GAN cost. As the snake contour is represented by an ordered set of points, we present analysis for 2-D Gaussian learning, and demonstrate that, in the proposed approach, the ordering of points can be traded-off for slower convergence. Through experiments on 2-D Gaussian learning and latent-space learning on image data, we show that the proposed approach performs on par with WGAN and Wasserstein autoencoder (WAE) [23] variants, that involve either discriminator network training or closed-form computations, with an order of magnitude faster convergence. The source code is available at `https://github.com/DarthSid95/SnakeGANs`.

2. GRADIENT-REGULARIZED GANS

Arjovsky *et al.* [5] introduced the Wasserstein GAN min-max *optimal transport* cost. Through the Kantorovich-Rubinstein duality, they defined the WGAN discriminator and generator loss functions as

$$\mathcal{L}_D^W = \mathbb{E}_{\boldsymbol{x}\sim p_g}[D(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{x}\sim p_d}[D(\boldsymbol{x})] \text{ and } \mathcal{L}_G^W = -\mathcal{L}_D^W,$$

respectively, where $\boldsymbol{x} \in \mathbb{R}^n$ and $D(\boldsymbol{x})$ is Lipschitz-1. Gulrajani *et al.* [6] consider the gradient penalty $\mathbb{E}_{\boldsymbol{x}\sim p_{\text{int}}}\left[(\|\nabla D\left(\boldsymbol{x}\right)\|_2 - 1)^2\right]$, where $\nabla D(\boldsymbol{x})$ denotes the gradient vector, with the τ^{th} entry given by $\frac{\partial D(\boldsymbol{x})}{\partial x_\tau}$ and p_{int} denotes the interpolated distribution $\eta p_d + (1-\eta)p_g$, $\eta \in [0, 1]$. Subsequent works [7–11] have replaced p_{int} with alternative gradient constraints that result in improved training. While these GANs are trained via stochastic gradient-descent, various works [24–27] have shown that learning the optimal discriminator function improves generator convergence. Coulomb GANs consider a discriminator based on the difference of charge potentials between the real and fake samples, and a network is trained to approximate the potential in the least-squares sense [12]. We adopt the RBF-Coulomb GAN discriminator formulation [13], where $\mathcal{L}_D = \mathcal{L}_D^W + \lambda_d \int_{\mathcal{X}} \|\nabla D(\boldsymbol{x})\|_2^2 \, \mathrm{d}\boldsymbol{x}$, λ_d is the Lagrange multiplier, and $\mathcal{X}$ is the convex hull of the supports of p_d and p_g. Asokan and Seelamantula [13] showed that the optimal discriminator satisfies the Poisson PDE whose solution can be approximated via the sample estimate:

$$D^*(\boldsymbol{x}) = \frac{1}{2N\lambda_d} \sum_{\boldsymbol{g}^j \sim p_g} \psi(\boldsymbol{x} - \boldsymbol{g}^j) - \frac{1}{2N\lambda_d} \sum_{\boldsymbol{d}^i \sim p_d} \psi(\boldsymbol{x} - \boldsymbol{d}^i), \quad (1)$$

where $\{\boldsymbol{g}^j\}$ and $\{\boldsymbol{d}^i\}$ denote the set of generated and target centers, respectively, and the kernel function ψ_n is given by

$$\psi(\boldsymbol{x} - \boldsymbol{y}) = \begin{cases} |\boldsymbol{x} - \boldsymbol{y}| & \text{for } n = 1, \\ -\ln\left(\|\boldsymbol{x} - \boldsymbol{y}\|\right) & \text{for } n = 2, \\ \|\boldsymbol{x} - \boldsymbol{y}\|^{2-n} & \text{for } n = 3, 4, 5, \ldots. \end{cases} \quad (2)$$

3. SNAKES

We now briefly recall the active contour *snake* formulation, which will be used within the GAN optimization. Active contour models, or *snakes* are a framework developed by Kass *et al.* [28] aimed at delineating the boundaries of objects in images (usually in the presence of noise). Active contour methods have found great success in image processing applications such as segmentation and boundary detection [29, 30]. Figure 1 presents a prototypical snake-based edge detection problem, where a snake is initialized as a random contour on the plane of the target, and iterated to minimize the gradient energy, thereby learning the object boundary. Improvements such as gradient vector flow (GVF) [31] or the balloon [32] models have been shown to improve convergence of the snake.

We consider the snake formulation in the standard 2-D setting for illustration. We show in Section 5 that ordering of the points does not affect the GAN formulation with snakes, and an unordered set of points could also be used, potentially allowing for extensions to n-D. Consider the active contour snake $\boldsymbol{s}(\xi) \in \mathbb{R}^2$ parametrized by $\xi \in [0, 1]$, which *navigates* through an image by minimizing the energy functional [28]: $\int_0^1 \frac{1}{2}\left(\alpha\|\boldsymbol{s}'(\xi)\|^2 + \beta\|\boldsymbol{s}''(\xi)\|^2\right) + \gamma E_{\text{ext}}(\boldsymbol{s}(\xi))\,\mathrm{d}\xi$, where α and β are parameters that control the snake's tension and rigidity, respectively; $\boldsymbol{s}'$ and $\boldsymbol{s}''$ denote the first- and second-order derivatives of $\boldsymbol{s}$; and γ is the weight assigned to the external energy. Constraints on the snake's internal energy (first- and second-order gradients) translate to the smoothness and curvature of the snake. The external energy E_{ext} is a function of the target shape being learnt, and governs the movement of the snake as the iterations progress. While, in the context of images I, the external energy is typically the negative of the gradient field of the image $E_{\text{ext}} = -\|\nabla \mathrm{I}\|^2$ [31] (cf. Figure 1), in GAN learning, we will use the gradient field of the discriminator as the external energy to be minimized (cf. Section 4).

The *optimal* snake is the one that minimizes the GVF energy function, and can be derived through the *Calculus of Variations* [28]. However, solving for the snake in the functional space is impractical, and the optimum is obtained by discretizing the solution. Consider the snake parametrized by the time index $\boldsymbol{s}_t(\xi)$, and subsequently discretized to an ordered set of points $\mathcal{S}^t = \left\{\boldsymbol{s}_t(\xi^k) |\ k = 1, 2, \ldots, N_s\right\}$. The ordering affects the accuracy of the finite-difference approximation. Then, the snake points in $\mathcal{S}$ can be updated through the following gradient descent:

$$\boldsymbol{s}_t(\xi^k) = [(\mathbb{I} - \gamma\mathbf{A})^{-1}]_k \left(\boldsymbol{s}_{t-1}(\xi^k) + \gamma\, \nabla E_{\text{ext}}\left(\boldsymbol{s}_{t-1}(\xi^k)\right)\right), \quad (3)$$

where $\boldsymbol{s}_t(\xi^k)$ denotes the k^{th} snake point at time instant t, ∇E_{ext} is the gradient vector associated with the external energy, and the subscript in $[(\mathbb{I} - \gamma\mathbf{A})^{-1}]_k$ denotes the k^{th} row of the cyclic penta-diagonal banded matrix $\mathbf{A}$, obtained by zero-padding and rolling the row vector $[-\beta, \alpha + 4\beta, -2\alpha - 6\beta, \alpha + 4\beta, -\beta]$ to form a matrix in $\mathbb{R}^{N_s \times N_s}$. The elements of $\mathbf{A}$ represent finite-differences associated with the second- and fourth-order derivatives. We show in Section 4 how the above snake update algorithm can be used to train a GAN.

4. TRAINING GANS WITH SNAKES

Within the proposed framework, we do away with the discriminator altogether, and use only its gradient field to predict the optimal trajectory that the generator points must take to minimize the WGAN loss. Consider the optimal discriminator given through Eq. (1) and (2). Then, its gradient vector $\nabla D^*(\boldsymbol{x})$ has entries given by

$$\frac{\partial D^*(\boldsymbol{x})}{\partial x_\tau} = \mathrm{C}_n \sum_{\boldsymbol{g}^j \sim p_g} \frac{x_\tau - g_\tau^j}{\|\boldsymbol{x} - \boldsymbol{g}^j\|^n} - \mathrm{C}_n \sum_{\boldsymbol{d}^i \sim p_d} \frac{x_\tau - d_\tau^i}{\|\boldsymbol{x} - \boldsymbol{d}^i\|^n}, \quad (4)$$

where x_τ denotes the τ^{th} entry in $\boldsymbol{x}$, $\boldsymbol{x} \in \mathbb{R}^n$, and C_n is a positive constant. As opposed to training a network with the WGAN generator loss, we compute the trajectory that the generated points must

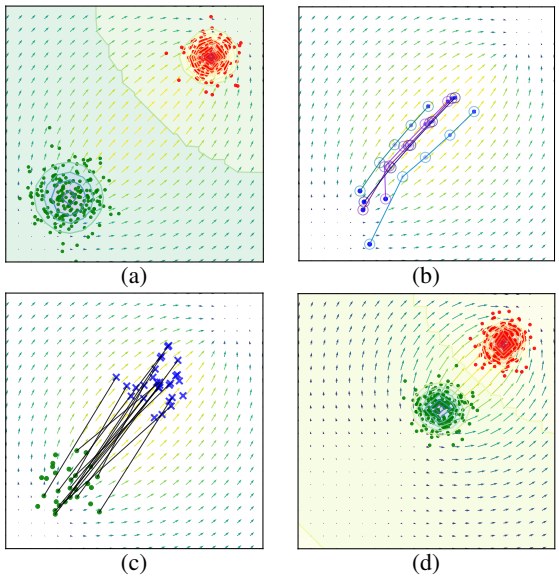


Fig. 2: (Color online; Best viewed in the *.pdf* format) The training algorithm in SnakeGANs. The dataset samples, generator outputs and snake iterates are plotted against the level sets of the generator and data distributions and the optimal discriminator's gradient field (quiver plots). (a) Based on the data and current generator distributions, the optimal discriminator (Eq. (1)) and its gradient field (Eq. (4)) are computed. (b) Snake contours (represented by a curve and its corresponding centroid) are initialized at the generated sample, and iterated (Eq. (3)) on the discriminator field for T steps. (c) The generator is updated based on the least-squares loss between the generated samples and the corresponding target snake centroids at T (Eq. (5)). (d) The new generator distribution approximates the distribution of the snake centroids at T, and the algorithm is repeated.

take, to minimize the GAN loss, as given by the snake updates in Eq. (3). While the snake algorithm defined in Section 3 computes the trajectory of any closed contour $\boldsymbol{s} \in \mathbb{R}^2$, such an ordered set of points within the GAN formulation is lacking. Therefore, given a batch of N generator samples $\mathcal{D} = \{\boldsymbol{x}_\ell | \boldsymbol{x}_\ell = G(\boldsymbol{z}) \sim p_g\}$, in 2-D, with $\boldsymbol{z}$ being Gaussian noise, we define closed contours around each sample $\boldsymbol{x}_\ell$, given by the set $\mathcal{S}_\ell^t$ at $t = 0$, consisting of N_ℓ points each. Intuitively, $\mathcal{S}_\ell^0$ represents an ordered set of points in the neighborhood of $\boldsymbol{x}_\ell$ that flow on the discriminator gradient field. Each contour is updated for T steps based on the snake algorithm in Eq. (3). The target of a generator point $\boldsymbol{x}_\ell$ is defined as the mean of the points in the set $\mathcal{S}_\ell^T$, given by $\boldsymbol{\mu}_\ell = \frac{1}{N_\ell} \sum_{\boldsymbol{y}_j \in \mathcal{S}_\ell^T} \boldsymbol{y}_j$, $\ell = 1, 2, \ldots, N$, and the generator is updated with a least-squares loss computed between the generated points $\boldsymbol{x}_\ell$ and their targets $\boldsymbol{\mu}_\ell$:

$$\mathcal{L}_G = \frac{1}{N_\ell} \sum_{\boldsymbol{x}_\ell \in \mathcal{D}} \|\boldsymbol{x}_\ell - \boldsymbol{\mu}_\ell\|_2^2, \text{ where } \boldsymbol{x}_\ell = G(\boldsymbol{z}) \sim p_g. \quad (5)$$

Owing to the *snake-based* approach to exploring the *discriminator* landscape, the proposed approach is called SnakeGAN. An illustrative algorithm for training SnakeGANs is provided in Figure 2.

The SnakeGAN framework with ordered points (SnakeGAN(o)) is computationally expensive, and can not be scaled to high-dimensional data, as ordering is ambiguous is n-D. We therefore consider an alternative with unordered points (SnakeGAN(uo)), wherein the set $\mathcal{D} = \{\boldsymbol{x}_\ell | \boldsymbol{x}_\ell = G(\boldsymbol{z}) \sim p_g\}$ itself is treated as the set of snake initializations $\mathcal{S}_\ell^t$ at $t = 0$.

5. EXPERIMENTATION

We validate the proposed SnakeGAN approach on 2-D Gaussian learning. We report comparisons against the baseline gradient-regularized WGAN variants such as WGAN-GP [6], WGAN with the Lipschitz penalty (WGAN-LP) [7], WGAN-R$_d$ and WGAN-R$_g$ [9]. We also consider the closed-form discriminator approaches with Coulomb GAN [12] and RBF-Coulomb GAN [13]. To compare against other generative models that employ a closed-form discriminative approach, we consider the generative moment matching networks (GMMNs) [33] with the inverse multi-quadric (IMQ) kernel.

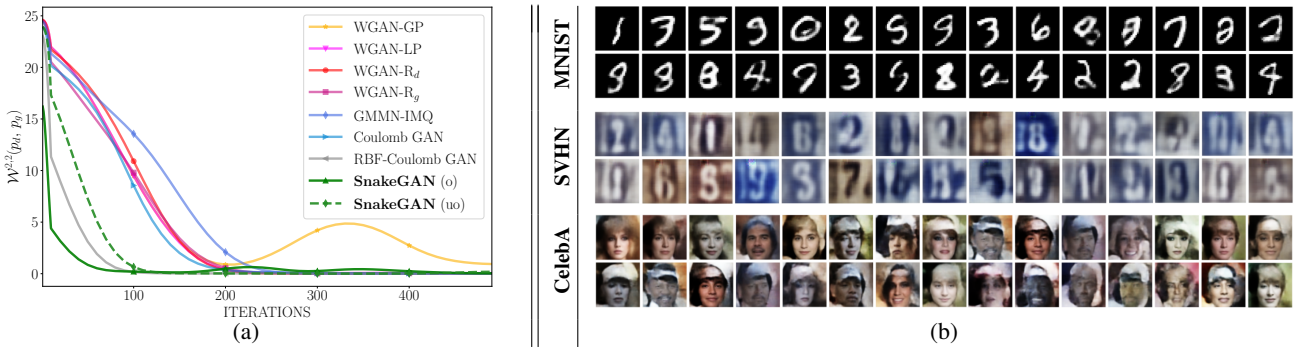
Unlike in RBF-Coulomb GANs, the SnakeGAN generator update does not involve the discriminator explicitly, but is trained on the least-squares loss between the generated points $\boldsymbol{x}_\ell$ and their corresponding targets $\boldsymbol{\mu}_\ell$. Consequently, in SnakeGAN, the gradient field of the discriminator and the snake updates need to be computed only once for every ten generator updates. To set the parameters of the snake, we performed initial experiments on learning the discriminator surface though active contours. Setting $\alpha > \beta$ leads to improved training of the generator. Empirically, we found that setting $\alpha = 0.5, \beta = 10^{-3}$ and $\gamma = 0.25$ results in a favorable performance of the generator. Each snake update involves ten iterations of Eq. (3).

The target Gaussian is $\mathcal{N}(5\mathbf{1}_2, 1.5\mathbb{I}_2)$, where $\mathbf{1}_2$ is the 2-D vector with both entries equal to one, and $\mathbb{I}_2$ is the 2×2 identity matrix. The generator in all GANs considered is an affine transformation of the noise $\boldsymbol{z}$, given by $\boldsymbol{x} = \mathbf{W}\boldsymbol{z} + \boldsymbol{b}$. While SnakeGAN does not involve a discriminator, the baselines use a three-layer fully-connected discriminator network with Leaky ReLU activations, having twenty, ten, and one node(s) in the first, second, and third layers, respectively. The Adam optimizer [34] is used to train the networks, with a learning rate of 0.02 for the generator and 0.075 for the discriminator. The batch size is 500. The models are evaluated using the Wasserstein-2 distance. Figure 3 (a) presents the Wasserstein-2 distance between p_d and p_g, as a function of the iterations. SnakeGAN outperforms all baseline variants with a trainable discriminator, and performs on par with the closed-form RBF-Coulomb GAN approach, while requiring fewer evaluations of the discriminator's gradient overall.

Extension to Autoencoding GANs: To extend the SnakeGAN(uo) approach to n-dimensional data, we consider an approach similar to that advocated for in latent diffusion models [35]. We train a deep convolutional auto-encoder to learn the 16-, 32- and 64-dimensional latent-space representations of MNIST, SVHN, and CelebA datasets, respectively. Subsequently, the SnakeGAN generator is trained to learn a mapping from its input to the latent space of the data. We compare against the baseline Wasserstein Autoencoder [23] in terms of the Fréchet inception distance (FID) [36]. Figure 3(b) shows the images generated by decoding the samples output by SnakeGAN(uo). SnakeGAN converges in 10^4 iterations on SVHN and CelebA (an order of magnitude faster than the baselines) with performance on par with the baseline – An FID of 18.48 for SnakeGAN, compared to 17.95 for the baseline on MNIST; An FID of 45.32 for SnakeGAN, compared to to 45.083 for the baseline on SVHN; and an FID of 48.064 on CelebA, compared to 45.88 for the baseline.

5.1. Ablation Study

We consider ablation experiments on the number of snake updates and the number of points required in each snake contour. To assess the performance of SnakeGAN with and without ordered snake points, we consider two scenarios: (a) SnakeGAN(o), consisting of *ordered* sets of ten points each, in the neighborhood of every generator sample. The points are selected equidistant on a circle of radius 0.1 centered



(a) (b)

Fig. 3: (Color online) (a) Wasserstein-2 distance versus iterations on learning a 2-D Gaussian with various SnakeGAN, and baseline GAN variants. SnakeGAN(o), with *ordered* snake points, outperforms the baselines with a trainable discriminator, although the discriminator gradient is evaluated only once every ten iterations. SnakeGAN(uo) computes the snake over *unordered* points, with a poorer approximation of the finite difference. SnakeGAN(uo) performs on par with the closed-form RBF-Coulomb GAN, while being computationally more efficient. (b) Images generated by SnakeGAN(uo) when trained to learn the latent-space distribution of various standard datasets. SnakeGAN converges in 10^4 iterations on SVHN and CelebA, an order of magnitude faster than the baseline GAN generators.

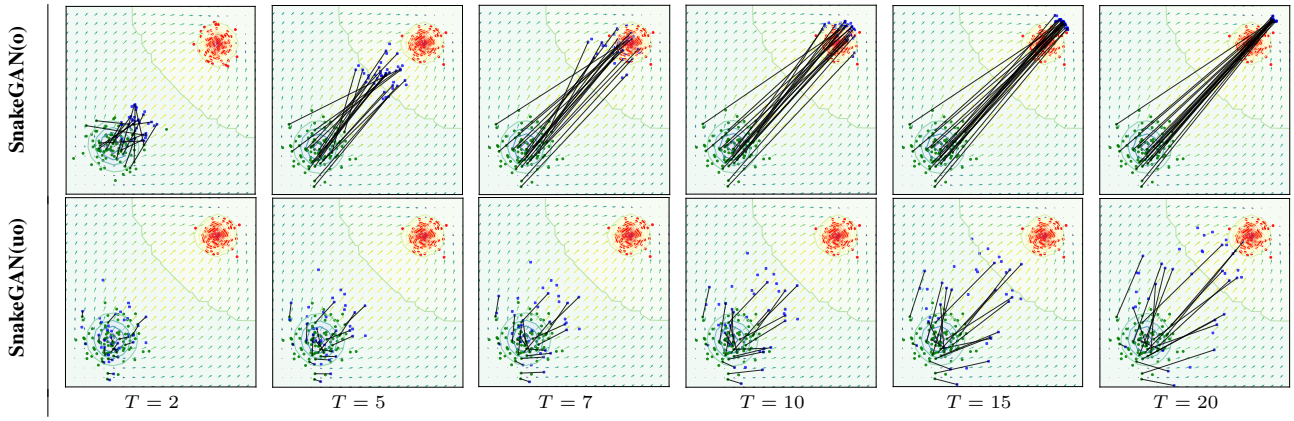


Fig. 4: (Color online) The dataset samples, generator outputs and converged snake points in SnakeGAN, considering both ordered and unordered snake points, for multiple choices of snake iterations T. The level-sets represent the generator and data distribution, while the quiver plots represent the optimal discriminator's gradient field. The **black** lines indicate point-wise correspondences between generated samples and their target snake location. Setting $T > 15$ results in poor performance, as the ordered snake points *mode-collapse*, while the unordered snake points diverge due to the error in estimating finite differences. Setting $T \approx 10$ was found to be a viable compromise.

about each sample $\boldsymbol{x}_\ell$; and (b) SnakeGAN(uo) with an *unordered* set of points, where the set of generator points at a given iteration are equal to the snake points, retaining the order. Figure 3 (a) compares the performance of these two variants against the baselines. While SnakeGAN(o) is superior to all baselines, SnakeGAN(uo) is on par with the convergence of the RBF-Coulomb GAN, which computes the closed-form discriminator at each time-step. SnakeGAN(uo) computes the gradient field once every ten generator updates.

We also compare the SnakeGAN performance as a function of T, the maximum number of snake updates. The dataset, generator and converged snake points, for various T, are shown in Figure 4. The correspondence between a few selected points is indicated using a **black** line. The convergence is superior in the case of SnakeGAN(o) as, with ordered points, the finite-difference approximations are more reliable. Fewer than five snake iterations result in insufficient convergence of the snake, and the generator training is slow, while training for more than 15 steps causes the errors in estimating the snake derivatives to accumulate. This is prominent in SnakeGAN(uo) for $T = 20$ (cf. Figure 4), where some snake points move away from the dataset samples. Across both variants, training for 20 or more iterations results in poorer performance. Choosing $T \approx 10$ is a compromise between generator convergence and accuracy of the snake updates.

6. DISCUSSIONS AND CONCLUSIONS

We proposed a novel approach to training GANs using active contours. The gradient field associated with the optimal discriminator in GANs, given a generator, is used as the external energy in a snake-update algorithm. The neighborhood of each fake sample is initialized with a snake, whose trajectory over the discriminator field indicated the optimal path for the generator update. Unlike existing generative modeling approaches that either train a generator adversarially or evolve particles through Langevin dynamics [14, 15], we update a generator through particle flow. The proposed WGAN with snake-based discriminator outperforms GAN variants with both trainable and closed-form discriminators on synthetic Gaussian learning tasks.

One can explore extensions of SnakeGAN in generating high-resolution images, similar to Sobolev descent [20] and normalizing flows [18]. While we considered the basic active contour framework [28], variants such as GVF snakes [31] could also be leveraged to improve the discriminator's gradient field. This paper establishes fundamental connections between GANs and active-contour flow-based models. Improving the performance of SnakeGANs on high-dimensional data involving real-world applications is a promising direction for future research.

7. REFERENCES

[1] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Info. Processing Systems 27*. 2014.

[2] X. Mao, Q. Li, H. Xie, R. Y. K. Lau, Z. Wang, and S. P. Smolley, "Least squares generative adversarial networks," in *Proc. of Intl. Conf. on Computer Vision*, 2017.

[3] S. Nowozin, B. Cseke, and R. Tomioka, "f-GAN: Training generative neural samplers using variational divergence minimization," in *Advances in Neural Info. Processing Systems 29*, pp. 271–279. 2016.

[4] M. Arjovsky and L. Bottou, "Towards principled methods for training generative adversarial networks," *arXiv preprints, arXiv:1701.04862*, 2017.

[5] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *Proc. of the 34th Intl. Conf. on Machine Learning*, 2017.

[6] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of Wasserstein GANs," in *Advances in Neural Info. Processing Systems 30*. 2017.

[7] H. Petzka, A. Fischer, and D. Lukovnikov, "On the regularization of Wasserstein GANs," in *Proc. of the 6th Intl. Conf. on Learning Representations*, 2018.

[8] D. Terjék, "Adversarial Lipschitz regularization," in *Proc. of the 8th Intl. Conf. on Learning Representations*, 2020.

[9] L. Mescheder, A. Geiger, and S. Nowozin, "Which training methods for GANs do actually converge?," in *Proc. of the 35th Intl. Conf. on Machine Learning*, 2018.

[10] N. Kodali, J. D. Abernethy, J. Hays, and Z. Kira, "On convergence and stability of GANs," *arXiv preprint, arXiv:1705.07215*, May 2017.

[11] Y. Mroueh, C. Li, T. Sercu, A. Raj, and Y. Cheng, "Sobolev GAN," in *Proc. of the 6th Intl. Conf. on Learning Representations*, 2018.

[12] T. Unterthiner, B. Nessler, C. Seward, G. Klambauer, M. Heusel, H. Ramsauer, and S. Hochreiter, "Coulomb GANs: Provably optimal Nash equilibria via potential fields," in *Proc. of the 6th Intl. Conf. on Learning Representations*, 2018.

[13] S. Asokan and C. S. Seelamantula, "Bridging the gap between Coulomb GAN and gradient-regularized WGAN," in *Proceedings on "The Symbiosis of Deep Learning and Differential Equations - II" at NeurIPS Workshops*, 2022.

[14] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *arXiv preprint, arXiv:2006.11239*, vol. abs/2006.11239, 2020.

[15] Y. Song and S. Ermon, "Improved techniques for training score-based generative models," in *Advances in Neural Info. Processing Systems 33*, 2020.

[16] Z. Xiao, K. Kreis, and A. Vahdat, "Tackling the generative learning trilemma with denoising diffusion GANs," in *Proc. of the 10th Intl. Conf. on Learning Representations*, 2022.

[17] J. Ho, X. Chen, A. Srinivas, Y. Duan, and P. Abbeel, "Flow++: Improving flow-based generative models with variational dequantization and architecture design," in *Proc. of the 36th Intl. Conf. on Machine Learning*, 2019.

[18] G. Papamakarios, E. Nalisnick, D. J. Rezende, S. Mohamed, and B. Lakshminarayanan, "Normalizing flows for probabilistic modeling and inference," *J. of Machine Learning Research*, 2021.

[19] P. Glaser, M. Arbel, and A. Gretton, "KALE flow: A relaxed KL gradient flow for probabilities with disjoint support," in *Advances in Neural Info. Processing Systems 34*, 2021.

[20] Y. Mroueh, T. Sercu, and A. Raj, "Sobolev descent," in *Proc. of the 22nd Intl. Conf. on Artificial Intelligence and Statistics*, Apr 2019.

[21] Y. Mroueh and M. Rigotti, "Unbalanced Sobolev descent," in *Advances in Neural Info. Processing Systems*, 2020, vol. 33.

[22] A. Grover, M. Dhar, and S. Ermon, "Flow-GAN: Combining maximum likelihood and adversarial learning in generative models," *Proc. of the AAAI Conf. on Artificial Intelligence*, vol. 32, Apr. 2018.

[23] I. O. Tolstikhin, O. Bousquet, S. Gelly, and B. Schölkopf, "Wasserstein auto-encoders," in *Proc. of the 6th Intl. Conf. on Learning Representations*, 2018.

[24] J. Li, A. Madry, J. Peebles, and L. Schmidt, "Towards understanding the dynamics of generative adversarial networks.," *arXiv preprints, arXiv:1706.09884*, 2017.

[25] P. Zhang, Q. Liu, D. Zhou, T. Xu, and X. He, "On the discrimination-generalization trade-off in GANs," in *Proc. of the 6th Intl. Conf. on Learning Representations*, 2018.

[26] C. Daskalakis, A. Ilyas, V. Syrgkanis, and H. Zeng, "Training GANs with optimism," in *Proc. of the 6th Intl. Conf. on Learning Representations*, 2018.

[27] K. Wu, G. W. Ding, R. Huang, and Y. Yu, "On minimax optimality of GANs for robust mean estimation," in *Proc. of the 23rd Intl. Conf. on Artificial Intelligence and Statistics*, 2020.

[28] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," *Intl. J. of Computer Vision*, 1988.

[29] F. Leymarie and M. D. Levine, "Tracking deformable objects in the plane using an active contour model," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 1993.

[30] R. Ďurikovič, K. Kaneda, and H. Yamashita, "Dynamic contour: A texture approach and contour operations," *The Visual Computer*, vol. 11, no. 6, pp. 277–289, June 1995.

[31] C. Xu and J. Prince, "Snakes, shapes, and gradient vector flow," *IEEE Trans. on Image Processing*, 04 1998.

[32] L. D. Cohen, "On active contour models and balloons," *Computer Vision and Image Understanding*, 1991.

[33] Y. Li, K. Swersky, and R. Zemel, "Generative moment matching networks," in *Proc. of the 32nd Intl. Conf. on Machine Learning*, Jul 2015.

[34] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. of the 3rd Intl. Conf. on Learning Representations*, 2015.

[35] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, 2022.

[36] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "GANs trained by a two time-scale update rule converge to a local Nash equilibrium," *arXiv preprints, arXiv:1706.08500*, 2018.