Drifting Model

Generative Modeling via Drifting
M Deng, H Li, T Li, Y Du, K He
arXiv preprint arXiv:2602.04770

74 *

2026

Generative Modeling via Drifting

Mingyang Deng, He Li, Tianhong Li, Yilun Du, Kaiming He

21 Jan 2026 (modified: 24 Jun 2026) Submitted to ICML 2026 Everyone Revisions BibTeX CC BY 4.0

TL;DR: A new generative modeling paradigm that evolves distributions during training, enabling high-quality one-step generation.

Abstract:
Generative modeling can be formulated as learning a mapping f such that its pushforward distribution matches the data distribution. The pushforward behavior can be carried out iteratively at inference time, e.g., in diffusion/flow-based models. In this paper, we propose a new paradigm called *Drifting Models*, which evolve the pushforward distribution during training and naturally admit one-step inference. We introduce a drifting field that governs the sample movement and achieves equilibrium when the distributions match. This leads to a training objective that allows the neural network optimizer to evolve the distribution. In experiments, our one-step generator achieves state-of-the-art results on ImageNet 256×256, with FID 1.54 in latent space and 1.73 in pixel space. We hope that our work opens up new opportunities for high-quality one-step generation.

Primary Area: Deep Learning->Generative Models and Autoencoders
Keywords: Generative models; Single-step generation; Distribution dynamics
Submission Number: 12536

Paper Decision

Decision by Program Chairs 30 Apr 2026, 11:59 (modified: 24 Jun 2026, 19:35) Everyone Revisions

Decision: Reject

Comment:

This paper proposes a generative modelling framework in which the outputs of a one-step generator are pushed to move in the direction of the current generated distribution evolved by a "drifting field", where the latter is defined in such a way that (under asymptotic conditions) the target distribution is a unique stationary point. This is distinct from typical formulations in which a divergence between the generated and target distributions is minimised directly: instead, the target for infinitesimal evolution of the generated distribution is defined explicitly. If the drifting field is the Wasserstein gradient of a divergence, the modelled distribution follows the divergence's gradient flow over the course of training.

The idea to use an MMD-based flow for learning generative models is old (for example, discussion in [Genevay et al., "Learning Generative Models with Sinkhorn Divergences", AISTATS'18] or MMD-GAN), but the use of such ideas in combination with the drifting field method is new and surprisingly effective, as shown in this paper's experiments. The paper also provides some theoretical analysis showing convergence under certain assumptions. The reviewers are mostly positive about the paper, especially the strong one-step generation results on a variety of problems, and about the presentation and clarity of the paper. (I also enjoyed reading the paper and share the positive assessment.)

However, multiple reviewers pointed out the following important weaknesses of the paper:

- Reliance on the feature encoder (could that be learnt jointly, and the method made adversarial?). More experiments, possibly in low-dimensional settings, are encouraged to stress-test the reliance on the encoder.
- Missed related work and placement in context. The addition of analogies and connections, not only contrasts, between the proposed method and existing work -- see the discussion in rebuttal -- would not weaken the paper's contribution, but rather increase its appeal and impact.

The above questions open some interesting directions, but the degree to which these questions begin to be explored in this submission is insufficient. The paper is near the borderline and I strongly encourage a resubmission with improvements along the axes described above.

The Philosophy Behind Generative Models
Also, the Moltivation of Drifting Models

- noise: x0 ~ p0 ### usually gaussian noise
  target: x1 ~ p1 ### whatever target we define; in this case, let's say different kitty pictures
- We believe that certain types of pictures have certain corresponding patterns, from which we can recognize them as "kitty" pictures rather than puppy pictures. These patterns can be represented, clustered, and learned in a high-dimensional space.
- We also believe that, "to generate kitty pictures" is equal to learning the target distribution p1, or learning the high-dimensional structure behind all those kitty pictures.
- In generative models, we aim to learn how to move samples from gaussian noise x0 ~ p0 to the target distribution x1 ~ p1.
- The goal of generative modeling is to find f such that

$$f_{\#}p_{\epsilon} \approx p_{\text{data}}$$

Background

- How different are p and q?
- How should samples move to reduce this difference?
- How can a neural network learn this movement?

Background I:
Measuring Distribution Mismatch

- For two points, we can use distance (e.g. l2, l1) to describe their discrepancy.
- For distributions, we observe distributions through samples.

$$X = \{x_i\}_{i=1}^{N} \sim p, \qquad Y = \{y_j\}_{j=1}^{M} \sim q.$$

- However, there is usually no one-to-one correspondence between X and Y. Therefore, a single distance cannot describe the difference between the two distributions.

Background I:
Measuring Distribution Mismatch

- We ask the same question h to both distributions:

$$\mathbb{E}_{X\sim p}[h(X)] \quad \text{and} \quad \mathbb{E}_{Y\sim q}[h(Y)].$$

- Their difference is

$$\Delta_h(p,q) = \mathbb{E}_{X\sim p}[h(X)] - \mathbb{E}_{Y\sim q}[h(Y)].$$

- If $p = q$, then

$$\Delta_h(p,q) = 0$$

- The more function we use, the more accurate we can describe it.
- Thus, we use a family of functions: $h_1(x), h_2(x), \ldots, h_m(x).$
- We collect them into a feature representation:

$$\phi(x) = \begin{bmatrix} h_1(x) \\ h_2(x) \\ \vdots \\ h_m(x) \end{bmatrix} \in \mathbb{R}^m.$$

Background I:
Measuring Distribution Mismatch

- feature space: $\phi(x) = \begin{bmatrix} h_1(x) \\ h_2(x) \\ \vdots \end{bmatrix}$, we have $\mu_p = \mathbb{E}_p[\phi(X)], \qquad \mu_q = \mathbb{E}_q[\phi(Y)].$

- Maximum Mean Discrepancy (MMD): $D_k^2(p, q) = \|\mu_p - \mu_q\|^2.$

$k(x, y) = \langle \Phi(x), \Phi(y) \rangle_{\mathcal{H}}.$

- kernel: $k(x, y) = \langle \phi(x), \phi(y) \rangle;$, < > is inner product $k(x, y)$ large $\Rightarrow$ $\phi(x)$ and $\phi(y)$ are similar.
- Instead of manually designing one test function h, we represent each sample with many features and compare samples in that feature space.

$$x \xrightarrow{\phi} \text{semantic feature} \xrightarrow{k} \text{similarity}.$$

$$k_\phi(x, y) = \exp\left(-\frac{\|\phi(x) - \phi(y)\|_2}{\tau}\right).$$

Background II:
Transporting a Distribution with a Vector Field

- A vector field assigns a movement direction to every point in space. $v_t : \mathbb{R}^d \to \mathbb{R}^d, \qquad x \mapsto v_t(x).$
- We want to find a field v_t such that: $(\Phi_1)_{\#} p_{\text{noise}} = p_{\text{data}}.$
- Flow Matching trains a neural network to predict the time-variant velocity field. $\mathcal{L}_{\text{FM}} = \mathbb{E}_{t,x_0,x_1}\left[\|v_\theta(x_t,t) - u_t(x_t \mid x_0,x_1)\|_2^2\right]$

- Drifting Models instead use a field to evolve the generated distribution during training.

Background III:
Supervised Learning and Stop-Gradient Targets

- Standard supervised learning:
  - Given input z, label y, network output $\hat{y} = f_\theta(z)$.
  - We have a popular loss function: $\mathcal{L} = \|f_\theta(z) - y\|_2^2$.
  - Be careful here: In standard supervised learning, the label is fixed and independent of the model parameters, which means $\frac{\partial y}{\partial \theta} = 0$.
  - Gradient: $\nabla_\theta \mathcal{L} = 2J_f^\top (f_\theta(z) - y)$, here Jacobian matrix $J_f = \frac{\partial f_\theta(z)}{\partial \theta}$.
- What if the target depends on the model?
  - Let's say we have a pseudo-target $\tilde{y}_\theta = f_\theta(z) + \Delta_\theta$,which is dependent to parameters θ. We should have $\mathcal{L} = \|f_\theta(z) - \mathrm{sg}(\tilde{y}_\theta)\|_2^2$. so that the gradient is $\nabla_\theta \mathcal{L} = 2J_f^\top (f_\theta(z) - \tilde{y}_\theta)$.
  - Otherwise, if we don't put "stop-gradient", the loss would be $\mathcal{L} = \|f_\theta(z) - \tilde{y}_\theta\|_2^2$, and the gradient would be $\nabla_\theta \mathcal{L} = 2(J_f - J_{\tilde{y}})^\top (f_\theta(z) - \tilde{y}_\theta)$.

Drifting Model in Brief

- Target: $f_\theta : \epsilon \mapsto x, \qquad \epsilon \sim p_\epsilon.$ $\qquad x = f_\theta(\epsilon) \qquad f_{\#}p_{\boldsymbol{\epsilon}} \approx p_{\text{data}}$

Here, "$f_{\#}$" denotes the pushforward induced by f.

- Method: Iteratively train $q_i = [f_i]_{\#}p_{\boldsymbol{\epsilon}}$, using

$$x_{\text{target}} = x + V_{p,q_\theta}(x).$$

$$\mathcal{L} = \mathbb{E}_\epsilon \left[\| f_\theta(\epsilon) - \text{sg}\left(f_\theta(\epsilon) + V_{p,q_\theta}(f_\theta(\epsilon))\right) \|^2 \right]$$

Compare:	$k(x, y)$
Move:	$x \to x + V_{p,q}(x)$
Learn:	$f_\theta(\epsilon) \to \text{sg}\left(f_\theta(\epsilon) + V_{p,q}(f_\theta(\epsilon))\right)$

How to pick a Vpq

- Antisymmetry: $V_{p,q}(x) = -V_{q,p}(x).$
- So that when $p = q \implies V = 0$
- However, we don't have $V = 0 \implies p = q$

$$^{1}q{=}p \Rightarrow \mathbf{V}_{p,q}{=}\mathbf{V}_{q,p}{=}{-}\mathbf{V}_{p,q} \Rightarrow \mathbf{V}_{p,q}{=}\mathbf{0}$$

- Here, the author emphasizes that as long as Vpq satisfies the requirements above, it is trainable.

The field $\mathbf{V}_{p,q}$ depends on two distributions p and q. To obtain a computable formulation, we consider the form:

$$\mathbf{V}_{p,q}(\mathbf{x}) = \mathbb{E}_{\mathbf{y}^+\sim p}\mathbb{E}_{\mathbf{y}^-\sim q}[\mathcal{K}(x, \mathbf{y}^+, \mathbf{y}^-)], \qquad (7)$$

where $\mathcal{K}(\cdot,\cdot,\cdot)$ is a kernel-like function describing interactions among three sample points. $\mathcal{K}$ can optionally depend on p and q. Our framework supports a broad class of functions $\mathcal{K}$, as long as $\mathbf{V} = 0$ when $p = q$.

Author's pick for Vpq: mean-shift method

- Attraction from real data y+ ~ p

$$V_p^+(x) = \frac{\mathbb{E}_{y^+\sim p}[k(x,y^+)y^+]}{\mathbb{E}_{y^+\sim p}[k(x,y^+)]} - x \quad = \mu_p(x) - x.$$

- Repulsion from generated data y- ~ q

$$V_q^-(x) = \frac{\mathbb{E}_{y^-\sim q}[k(x,y^-)(y^- - x)]}{Z_q(x)} = \mu_q(x) - x.$$

- Define Vpq = Vp – Vq = $\mu_p(x) - \mu_q(x)$

$$k_\phi(x,y) = \exp\left(-\frac{\|\phi(x) - \phi(y)\|_2}{\tau}\right). \quad \tau \in \{0.02, 0.05, 0.2\}.$$

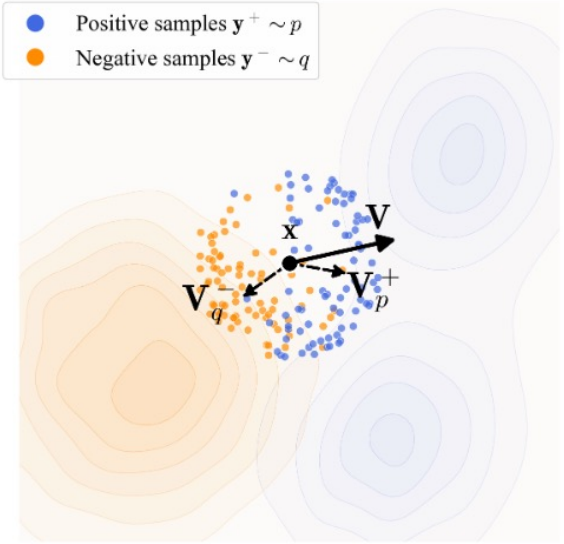

Figure 2. **Illustration of drifting a sample.** A generated sample $\mathbf{x}$ (black) drifts according to a vector: $\mathbf{V} = \mathbf{V}_p^+ - \mathbf{V}_q^-$. Here, $\mathbf{V}_p^+$ is the mean-shift vector of the positive samples (blue) and $\mathbf{V}_q^-$ is the mean-shift vector of the negative samples (orange): see Eq. (8). $\mathbf{x}$ is attracted by $\mathbf{V}_p^+$ and repulsed by $\mathbf{V}_q^-$.

Training Process

- 1. $$\epsilon_i \sim p_\epsilon,$$ $$x_i = f_\theta(\epsilon_i, c, \alpha).$$
- 2. $y_j^+ \sim p_{\text{data}}(\cdot \mid c).$ $y_k^- = x_k.$
- 3. compute ϕ and kernel

$$x, \quad y^+, \quad y^-,$$

$$\phi_j(\cdot).$$

$$k_{ij}^+ = \exp\left(-\frac{\|\phi(x_i) - \phi(y_j^+)\|}{\tau}\right),$$
$$k_{ik}^- = \exp\left(-\frac{\|\phi(x_i) - \phi(y_k^-)\|}{\tau}\right).$$

- 4. $$V_i = \mu_p(x_i) - \mu_q(x_i).$$

$$t_i = \text{sg}\left(\phi(x_i) + V_i\right).$$

$$\mathcal{L} = \sum_{i,j} \|\phi_j(x_i) - t_{i,j}\|^2.$$

- 5. $q_\theta \to q_{\theta'}.$

Experiments

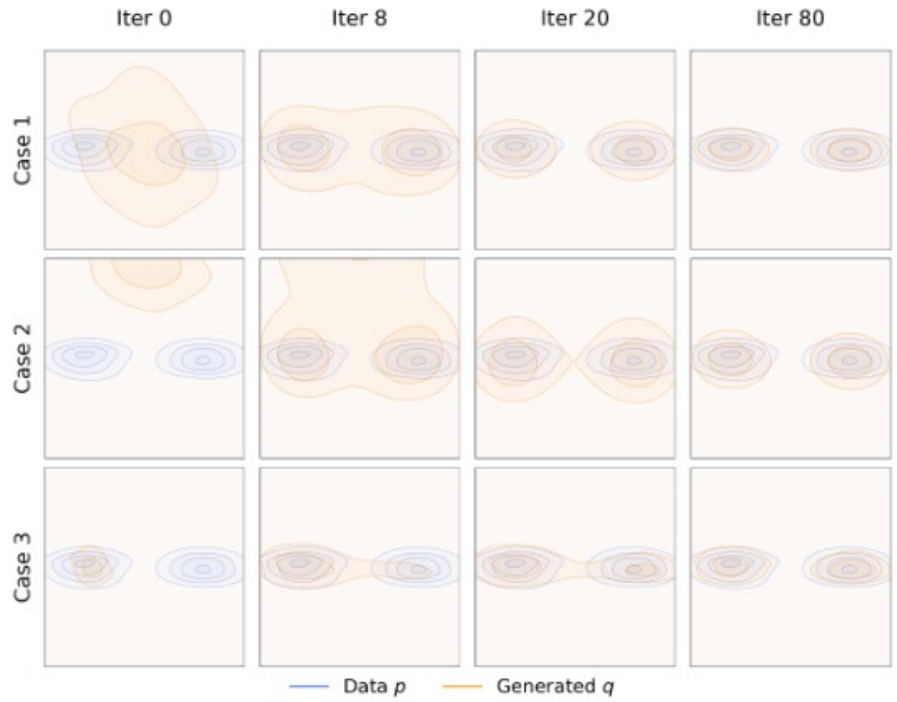

Figure 3. **Evolution of the generated distribution.** The distribution q (orange) evolves toward a bimodal target p (blue) during training. We show three initializations of q: **(top)**: initialized between the two modes; **(middle)**: initialized far from both modes; **(bottom)**: initialized collapsed onto one mode. Across all initializations, our method approximates the target distribution without mode collapse.

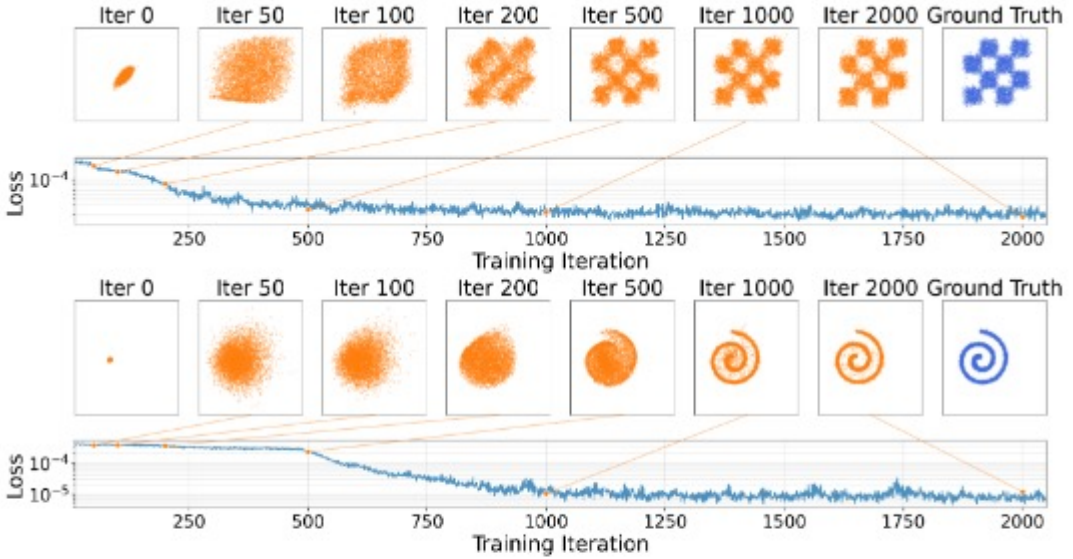

Figure 4. **Evolution of samples.** We show generated points sampled at different training iterations, along with their loss values. The loss (whose value equals $\|V\|^2$) decreases as the distribution converges to the target. (y-axis is log-scale.)

Experiments

Figure 7. **Uncurated samples from our latent-L/2 model with CFG = 1.0 (page 1/4).** FID = 1.54, IS = 258.9.

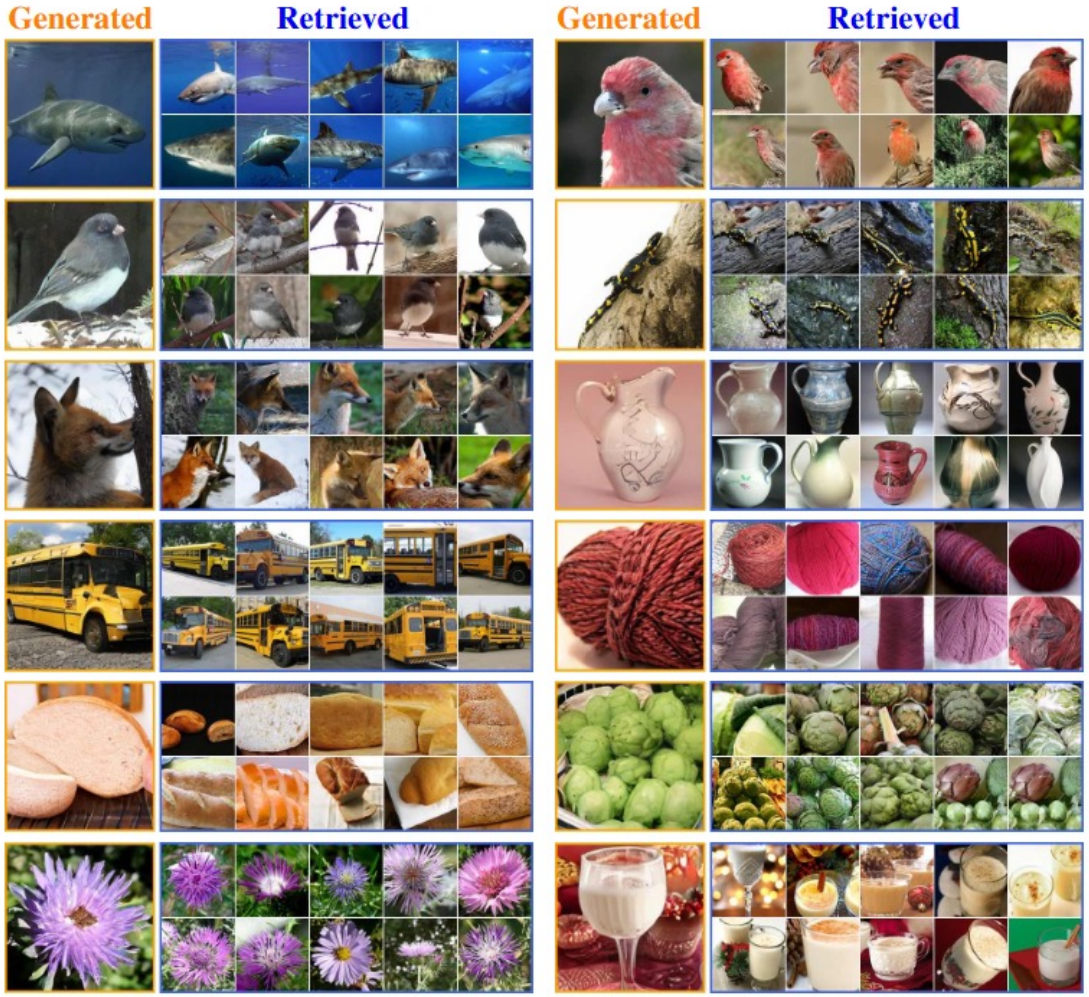

Figure 6. **Nearest neighbor analysis.** Each panel shows a generated sample together with its top-10 nearest real images. The nearest neighbors are retrieved from the ImageNet training set based on the cosine similarity using a CLIP encoder (Radford et al., 2021). Our method generates novel images that are visually distinct from their nearest neighbors.

Experiments

Table 1. **Importance of anti-symmetry:** breaking the anti-symmetry leads to failure. Here, the anti-symmetric case is defined in Eq. (10) and Eq. (11); other destructive cases are defined in similar ways. (Setting: B/2 model, 100 epochs)

case	drifting field $\mathbf{V}$	FID
anti-symmetry (default)	$\mathbf{V}^+ - \mathbf{V}^-$	**8.46**
1.5× attraction	$1.5\mathbf{V}^+ - \mathbf{V}^-$	41.05
1.5× repulsion	$\mathbf{V}^+ - 1.5\mathbf{V}^-$	46.28
2.0× attraction	$2\mathbf{V}^+ - \mathbf{V}^-$	86.16
2.0× repulsion	$\mathbf{V}^+ - 2\mathbf{V}^-$	112.84
attraction-only	$\mathbf{V}^+$	177.14

Table 2. **Allocation of positive and negative samples.** In both sub-tables, we control the total compute by fixing the epochs (100) and the batch size $B = N_c \times N_{pos}$ (4096). Here, N_c is for class labels. Under the same budget, increasing positive samples (**left**) and negative samples (**right**) improves generation quality. (Setting: B/2 model, 100 epochs)

N_c	N_{pos}	N_{neg}	B	FID
64	1	64	4096	20.43
64	16	64	4096	10.39
64	32	64	4096	8.97
64	**64**	64	4096	**8.46**

N_c	N_{pos}	N_{neg}	B	FID
512	8	8	4096	11.82
256	16	16	4096	10.16
128	32	32	4096	9.32
64	64	**64**	4096	**8.46**

Experiments

Table 3. **Feature space for drifting.** We compare self-supervised learning (SSL) encoders. Standard SimCLR and MoCo encoders achieve competitive results, whereas our customized latent-MAE performs best and benefits from increased width and longer training. (Generator setting: B/2 model, 100 epochs)

SSL method	feature encoder (ϕ)				FID
	arch	block	width	SSL ep.	
SimCLR	ResNet	bottleneck	256	800	11.05
MoCo-v2	ResNet	bottleneck	256	800	8.41
latent-MAE (default)	ResNet	basic	256	192	8.46
latent-MAE	ResNet	basic	384	192	7.26
latent-MAE	ResNet	basic	512	192	6.49
latent-MAE	ResNet	basic	640	192	6.30
latent-MAE	ResNet	basic	640	1280	4.28
latent-MAE + cls ft	ResNet	basic	640	1280	**3.36**

Table 5. **System-level comparison: ImageNet 256×256 generation in latent space.** FID is on 50K images, all reported with CFG if applicable. The parameter numbers are "generator + decoder". All generators are trained from scratch (*i.e.*, not distilled).

method	space	params	NFE	FID↓	IS↑
Multi-step Diffusion/Flows					
DiT-XL/2 (Peebles & Xie, 2023)	SD-VAE	675M+49M	250×2	2.27	278.2
SiT-XL/2 (Ma et al., 2024)	SD-VAE	675M+49M	250×2	2.06	270.3
SiT-XL/2+REPA (Yu et al., 2024)	SD-VAE	675M+49M	250×2	1.42	305.7
LightningDiT-XL/2 (Yao et al., 2025)	VA-VAE	675M+70M	250×2	1.35	295.3
RAE+DiT^{DH}-XL/2 (Zheng et al., 2025)	RAE	839M+415M	50×2	**1.13**	262.6
Single-step Diffusion/Flows					
iCT-XL/2 (Song & Dhariwal, 2023)	SD-VAE	675M+49M	1	34.24	–
Shortcut-XL/2 (Frans et al., 2024)	SD-VAE	675M+49M	1	10.60	–
MeanFlow-XL/2 (Geng et al., 2025a)	SD-VAE	676M+49M	1	3.43	–
AdvFlow-XL/2 (Lin et al., 2025)	SD-VAE	673M+49M	1	2.38	284.2
iMeanFlow-XL/2 (Geng et al., 2025b)	SD-VAE	610M+49M	1	1.72	282.0
Drifting Models					
Drifting Model, B/2	SD-VAE	133M+49M	1	1.75	263.2
Drifting Model, L/2	SD-VAE	463M+49M	1	**1.54**	258.9

Sinkhorn-drifting generative models [PDF] arxiv.org

P He, O Khangaonkar, H Pirsiavash, Y Bai... - arXiv preprint arXiv ..., 2026 - arxiv.org

... **Sinkhorn drifting** reduces sensitivity to kernel temperature and improves one-step **generative** ... lowest temperature setting we evaluate, **Sinkhorn drifting** reduces mean FID from 187.7 to ...

☆ Save 99 Cite Cited by 13 Related articles All 2 versions »

Sinkhorn-Drifting Generative Models

Ping He[1] **Om Khangaonkar**[2]
Hamed Pirsiavash[2] **Yikun Bai**[3,‡] **Soheil Kolouri**[1,‡]

[1]Department of Computer Science, Vanderbilt University
[2]Department of Computer Science, University of California, Davis
[3]Department of Computer Science, Purdue University
ping.he@vanderbilt.edu, omkhanga@ucdavis.edu, hpirsiav@ucdavis.edu
bai195@purdue.edu, soheil.kolouri@vanderbilt.edu

Motivation & Method

- Repulsion: $V_q^-(x) = \sum_j w_j(x)(y_j^- - x).$ $w_j(x) \propto \exp\left(-\frac{\|x - y_j^-\|^2}{\tau}\right),$
  - For each generated sample x, the sum of the weights assigned to y is 1. $\sum_j W_{ij} = 1.$
  - When τ is small, weights are determined by the nearest y- (even x itself)
  - So that $y_i^- = x_i.$ $w_i \approx 1, \quad y_i^- - x_i = 0,$ $V_q^-(x_i) \approx 0.$

- Skinhorn:
  - the weight matrix satisfies both row and column marginal constraints

$$\sum_j W_{ij} = 1, \qquad \sum_i W_{ij} = 1.$$

  - For a small temperature: $W \approx \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}.$

On the Wasserstein Gradient Flow Interpretation of Drifting Models [PDF] arxiv.org

A Gretton, LK Wenliang, A Galashov, J Thornton, V De Bortoli, A Doucet

arXiv preprint arXiv:2605.05118, 2026 · arxiv.org

Recently, Deng et al. (2026) proposed Generative Modeling via Drifting (GMD), a novel framework for generative tasks. This note presents an analysis of GMD through the lens of Wasserstein Gradient Flows (WGF), i.e., the path of steepest descent for a functional in the space of probability measures, equipped with the geometry of optimal transport. Unlike previous WGF-based contributions, GMD can be thought of as directly targeting a fixed point of a specific WGF flow. We demonstrate three main results: first, that one algorithm

SHOW MORE

On the Wasserstein Gradient Flow Interpretation of Drifting Models

Arthur Gretton, Li Kevin Wenliang, Alexandre Galashov, James Thornton,
Valentin De Bortoli, Arnaud Doucet

Google DeepMind

Conclusion

- Simple drifting approximates the fixed point of the KL Wasserstein gradient flow using kernel-smoothed score estimates.

$$V_{p,q}(x) \propto \nabla \log p_\tau(x) - \nabla \log q_\tau(x),$$

- The practical drifting field is not the gradient of a well-defined distributional loss

- Because Gaussian kernels are local, drifting may fail to transfer probability mass between distant modes.

Other recent works

- Kernel choice determines which distribution frequencies can be learned efficiently.
- Drifting is nonparametric score matching.
- Using kernel gradients (instead of y-x) makes drifting a general score-based framework beyond Euclidean continuous data.