LEARNING THE GEOMETRY OF DRIFTING MODELS: PERSISTENT REPRESENTATIONS FOR ONE-STEP GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Recently proposed Drifting Models shift iterative distribution refinement from inference to training, enabling effective one-step generation. However, their performance on complex image datasets depends strongly on the representation used to construct the drifting field: pixel-space drifting performs poorly, whereas pretrained feature spaces substantially improve sample quality for reasons that remain unclear. We trace this gap to the discriminative geometry of the representation, which determines sample weighting in kernel density estimation (KDE) and, consequently drift. We introduce *persistent representation learning*, which continuously learns a more discriminative representation geometry as the generator evolves across batches. We further establish a current-step gradient equivalence between the KDE ratio loss and drift regression loss under matched conditions, connecting density-ratio-based generator optimization to empirical drifting and motivating direct control of the drifting velocity. Across multiple datasets, our method learns effective discriminative representations directly from pixels and reduces FID by approximately $82 - 95\%$ over the original pixel-space Drifting Models, without pretrained encoders. Adapting pretrained representations and applying velocity clipping provide further gains.

1 INTRODUCTION

Diffusion models (Ho et al., 2020; Song et al., 2020) and flow-matching methods (Liu et al., 2022; Lipman et al., 2022) have achieved remarkable performance in image (Rombach et al., 2021; Esser et al., 2024) and video generation (Blattmann et al., 2023; Yang et al., 2024). However, high-quality synthesis typically relies on iterative sampling, whose computational cost grows with model scale, spatial resolution, and video length. This has motivated extensive research on high-quality one-step generation (Salimans & Ho, 2022; Liu et al., 2023; Song et al., 2023; Frans et al., 2025; Geng et al., 2025). Drifting Models (Deng et al., 2026) provide a distinct alternative by shifting iterative distribution refinement from inference to training. During training, they repeatedly construct a drifting field from real and generated samples and regress the generator toward the corresponding displaced targets. Once trained, they generate samples in a single inference step, without iterative sampling.

However, the success of Drifting Models on complex image datasets relies heavily on powerful pretrained encoders. Constructing the drifting field directly in pixel space often leads to poor sample quality, whereas constructing it in a pretrained feature space substantially improves performance (Deng et al., 2026; Zhang et al., 2026). This empirical contrast suggests that the representation space plays a central role in drifting optimization, yet how it shapes the optimization dynamics remains poorly understood. Why do pretrained representations induce more effective drifting fields? Are fixed external features essential, or can suitable representations emerge jointly with the generator through the drifting objective itself? Answering these questions is important both for understanding the mechanism underlying Drifting Models and for improving their training.

We study this problem by examining how the representation space shapes the drifting field. Representation-induced distances and local neighborhoods determine the kernel weights, which in turn govern the local density estimates and the resulting drift direction. When these relationships are defined by a fixed pixel-space geometry, the batch KDE density-ratio estimate has limited ability to

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

distinguish real from generated samples, leading to less informative supervision for the generator. Pretrained encoders instead induce a more discriminative representation geometry and empirically yield more effective drifting fields. This observation suggests that effective drifting requires a representation that captures sample relationships relevant to the evolving generative task.

Building on this insight, our method addresses both the learning of representation geometry and its effect on generator optimization. We introduce *persistent representation learning*, which continuously updates the representation throughout training while retaining information from successive discriminative updates in shared parameters. The geometry used to construct the drifting field can therefore evolve together with the generator, enabling Drifting Models to learn effective representations directly from pixels without relying on fixed pretrained features. To characterize how this learned geometry affects generator updates, we build on the Wasserstein-gradient-flow interpretation of drifting (Gretton et al., 2026) and the flow interpretation of divergence-based GAN training (Yi et al., 2023). We show that the spatial gradient of the KDE log-density-ratio score recovers the empirical drifting field up to scaling. Moreover, the corresponding density-ratio objective yields the same generator-parameter gradient as drift regression under matched conditions. This equivalence directly connects the representation-dependent density-ratio geometry to the generator update. Finally, because the drifting field explicitly defines the velocity of each generated sample, we introduce velocity clipping to bound its magnitude and stabilize training as the representation evolves.

Empirically, the resulting framework substantially improves one-step generation across both learned and pretrained representation settings. Taken together, our contributions are as follows:

- We introduce *persistent representation learning*, which accumulates discriminative updates across batches and progressively adapts the representation geometry to the evolving generator, supporting both learning directly from pixels and adapting pretrained feature encoders.
- We establish an algorithmic connection between density-ratio-based generator optimization and empirical drifting. For Gaussian kernels, we show a current-step gradient equivalence between the KDE ratio loss and the drift regression loss under matched conditions.
- We characterize how the local regularity of the density-ratio potential controls the induced drifting velocity, motivating direct velocity clipping to stabilize training as the representation evolves.
- Experiments show that our method reduces FID by approximately $82-95\%$ over the original pixel-space Drifting Models across multiple datasets without pretrained encoders, with further gains from adapting pretrained representations and applying velocity clipping. Diagnostic analysis further reveals how representation learning affects drift alignment and KDE discrimination.

2 Related Work

Drifting Models. Generative Modeling via Drifting (Deng et al., 2026) constructs a vector field through nonparametric interactions between real and generated samples, then regresses toward targets displaced by this field to absorb iterative distribution optimization into a one-step generator. Turan et al. (2026) connect Gaussian-kernel drifting fields to score differences between KDE-smoothed distributions. Lai et al. (2026) relate Gaussian- and Laplace-kernel drifting to score-based diffusion models through Tweedie's formula. Zhang et al. (2026) extend Drifting Models to pretrained representation spaces for one-step distillation and report that training from scratch without pretrained features remains open. W-Flow (Han et al., 2026) constructs a drifting model from the Wasserstein gradient flow of the Sinkhorn divergence and compresses the resulting particle evolution into a one-step generator. Other studies improve the computational efficiency of drifting or analyze its finite-particle convergence and stability (Falahati et al., 2026; Balasubramanian, 2026; Lee & Chun, 2026). Despite these advances, the role of representation in drifting remains underexplored: existing approaches typically rely on a fixed representation, leaving open how the discriminative representation can adapt during training.

Generative Adversarial Networks. GANs train a generator against a discriminator that distinguishes real from generated samples (Goodfellow et al., 2014). Variational formulations such as f-GAN show that the optimal discriminator encodes a density ratio through a prescribed transformation (Nowozin et al., 2016), while MonoFlow (Yi et al., 2023) interprets the resulting generator

direction through Wasserstein gradient flows. KDD-GAN and Projected GAN demonstrate the importance of the representation used for discrimination, using learned and fixed pretrained feature spaces, respectively (Lemkhenter et al., 2021; Sauer et al., 2021). Lipschitz constraints, implemented through methods such as gradient penalties and spectral normalization, further control the discriminator gradients received by the generator (Gulrajani et al., 2017; Miyato et al., 2018). These works highlight two recurring mechanisms in adversarial training: persistent discriminator learning determines how the discrepancy between real and generated distributions is represented, while regularity constraints control the spatial gradients through which this discrepancy acts on the generator.

Wasserstein Gradient Flows. Wasserstein gradient flows describe the steepest descent of an energy functional over probability distributions under the geometry of optimal transport (Santambrogio, 2017). In generative modeling, this perspective defines a velocity field that transports the generated distribution toward the real data distribution (Arjovsky et al., 2017). Gretton et al. (2026) connect Gaussian-kernel drifting to the limiting point of a Wasserstein gradient flow associated with the KL divergence between Parzen-smoothed distributions, and analyze extensions to other distributional objectives. These analyses characterize the distribution-level dynamics underlying drifting, but leave the finite-batch construction of the drifting field and the regularity of its induced generator update as separate algorithmic questions.

Our work connects these three lines of research by studying how the discriminative representation shapes the empirical drifting field and the resulting generator update. We introduce persistent representation learning to adapt the representation across batches instead of using a fixed or pretrained representation. We further show that the KDE density-ratio potential connects the drifting field to GAN generator updates, which motivates directly clipping the drifting velocity to stabilize training as the representation evolves.

3 BACKGROUND

Drifting Models. Drifting Models (Deng et al., 2026) construct a drifting field from attractive interactions with real samples and repulsive interactions with generated samples, and use this field to iteratively update the generator. Let p denote the real data distribution. A generator $G_\theta : \mathbb{R}^{d'} \to \mathbb{R}^d$ maps some latent z sampled from some easy-to-sample distribution μ (typically Gaussian) with $z \sim \mu$ to the generated data $\hat{x} = G_\theta(z)$. This process induces the generated data distribution $q_\theta = G_\theta \# \mu$. Let $k_\tau : \mathbb{R}^d \times \mathbb{R}^d \to (0, \infty)$ be a positive kernel measuring similarity between samples, with bandwidth parameter $\tau > 0$. We use the Gaussian kernel

$$k_\tau(x, y) = (\pi\tau)^{-d/2} e^{-\|x-y\|_2^2/\tau}. \tag{1}$$

where d is the sample dimension. Its normalization constant cancels in the normalized weights below. A *drifting field* is a vector field $V_{p,q_\theta} : \mathbb{R}^d \to \mathbb{R}^d$, determined by the pair (p, q_θ), that assigns to every point x a displacement pointing toward the real data and away from the generated data. It is formed as the difference of two kernel-weighted local means, one taken over real samples and one over generated samples, and in practice it is estimated from the samples in the current batch. Given real samples $\{\tilde{x}_j\}_{j=1}^{N^+}$ and generated samples $\{\hat{x}_i\}_{i=1}^{N^-}$, we have the empirical drifting field[1]:

$$\widehat{V}_{p,q_\theta}(x) = \frac{\sum_{j=1}^{N^+} k_\tau(x, \tilde{x}_j)(\tilde{x}_j - x)}{\sum_{j=1}^{N^+} k_\tau(x, \tilde{x}_j)} - \frac{\sum_{i=1}^{N^-} k_\tau(x, \hat{x}_i)(\hat{x}_i - x)}{\sum_{i=1}^{N^-} k_\tau(x, \hat{x}_i)}. \tag{2}$$

The first term is the attractive component pulling x toward nearby real samples, and the second the repulsive component pushing it away from nearby generated samples. The generator regresses toward the displaced stop-gradient target through the following drift regression loss

$$\mathcal{L}^{\text{Drift}}(\theta) = \mathbb{E}_{z\sim\mu}\left[\left\| G_\theta(z) - \text{sg}\Big(G_\theta(z) + \widehat{V}_{p,q_\theta}(G_\theta(z))\Big)\right\|_2^2\right], \tag{3}$$

where the field is constructed from the current batch and held fixed during the generator update.

For the Gaussian kernel in Eq. (1), the drift admits a score-difference interpretation through Parzen-smoothed densities. We define the smoothed densities of the real data distribution p and the generated data distribution q_θ as

$$p_\tau(x) = \mathbb{E}_{\tilde{x}\sim p}[k_\tau(x, \tilde{x})], \qquad q_\tau^\theta(x) = \mathbb{E}_{\hat{x}\sim q_\theta}[k_\tau(x, \hat{x})].$$

[1]For simplicity, we omit that same distribution KDE estimates exclude the evaluation point.

The population drift is their score difference $V_{p,q_\theta}(x) = \frac{\tau}{2}\left(\nabla \log p_\tau(x) - \nabla \log q_\tau^\theta(x)\right)$. This is a smoothed approximation of the Wasserstein gradient flow associated with $\mathrm{KL}(q_\theta \| p)$ (Gretton et al., 2026), whose velocity is $V_{p,q_\theta}^{\mathrm{KL}}(x) = \nabla \log p(x) - \nabla \log q_\theta(x)$.

4 METHODOLOGY

In Drifting Models, the representation induces the geometry used by the KDE: feature-space distances determine which real and generated samples receive high kernel weights and how they contribute to the drifting update. With a fixed representation, the KDE and drifting field still evolve with the generator distribution, but the geometry used to measure and localize discrepancies remains unchanged. We therefore train the encoder alongside the generator, retaining its parameters across batches while rebuilding the KDE from each current batch. We first characterize how the representation geometry shapes the drifting update, then introduce alternating encoder and generator updates, and finally apply velocity clipping to control the magnitude of the feature-space drift. Algorithm 1 summarizes the complete training procedure.

Algorithm 1: Persistent Representation for Drifting Training with Velocity Clipping

Input: Generator G_θ, encoder E_φ, real data distribution p, noise distribution μ, kernel k_τ, batch size B, encoder steps K, velocity threshold $V_{\max}$, and learning rates γ_E, γ_G

repeat
  for $k = 1, \dots, K$ **do**
    Sample fresh $\{z_i\}_{i=1}^B \sim \mu$ and $\{\tilde{x}_j\}_{j=1}^B \sim p$
    $\hat{x}_i \leftarrow \mathrm{sg}(G_\theta(z_i))$, $u_i \leftarrow E_\varphi(\hat{x}_i)$, $v_j \leftarrow E_\varphi(\tilde{x}_j)$
    Evaluate $s_\varphi^{\mathrm{KDE}}(\tilde{x}_j)$ and $s_\varphi^{\mathrm{KDE}}(\hat{x}_i)$
    Compute $\widehat{\mathcal{L}}_\varphi^D$ using Eq. (7) and update φ
  end for
  Sample fresh $\{z_i\}_{i=1}^B \sim \mu$ and $\{\tilde{x}_j\}_{j=1}^B \sim p$
  $u_i \leftarrow E_\varphi(G_\theta(z_i))$ and $v_j \leftarrow E_\varphi(\tilde{x}_j)$
  Construct $\widehat{V}_\varphi(u_i)$
  $\widehat{V}_{\varphi,\mathrm{clip}}(u_i) \leftarrow \widehat{V}_\varphi(u_i) / \max(1, \|\widehat{V}_\varphi(u_i)\|_2 / V_{\max})$
  $\widetilde{u}_i \leftarrow \mathrm{sg}(u_i + \widehat{V}_{\varphi,\mathrm{clip}}(u_i))$
  $\widehat{\mathcal{L}}_{\varphi,\theta}^{\mathrm{Drift}} \leftarrow B^{-1} \sum_{i=1}^B \|u_i - \widetilde{u}_i\|_2^2$ ▷ clipped target
  Update θ using this loss
until converged

4.1 KDE-RATIO AND DRIFTING UPDATES IN REPRESENTATION SPACE

We formulate the generator update directly in the representation space used to construct the drifting field, where the corresponding pixel-space update is recovered by taking E_φ to be the identity map. Let $E_\varphi : \mathbb{R}^d \to \mathbb{R}^{\tilde{d}}$ be an encoder. We use k_τ in the dimension of its arguments; in representation space, the Gaussian normalization in Eq. (1) is $(\pi\tau)^{-\tilde{d}/2}$. We set $N^+ = N^- = B$ below. For a batch of B real samples $\{\tilde{x}_j\}_{j=1}^B$ and generated samples $\{\hat{x}_i\}_{i=1}^B$, where $\hat{x}_i = G_\theta(z_i)$ and $z_i \sim \mu$, we hold φ fixed during the generator update. Their corresponding representations are $v_j = E_\varphi(\tilde{x}_j)$ and $u_i = E_\varphi(\hat{x}_i)$. We then construct batch KDEs in the representation space with $\widehat{p}_{\varphi,\tau}(v) = \frac{1}{B}\sum_{j=1}^B k_\tau(v, v_j)$ and $\widehat{q}_{\varphi,\tau}^\theta(v) = \frac{1}{B}\sum_{i=1}^B k_\tau(v, u_i)$.

For any input $x \in \mathbb{R}^d$, we define its KDE-based log-density-ratio score (Lemkhenter et al., 2021) in the representation space induced by E_φ as

$$s_\varphi^{\mathrm{KDE}}(x) := \log \widehat{p}_{\varphi,\tau}(E_\varphi(x)) - \log \widehat{q}_{\varphi,\tau}^\theta(E_\varphi(x)) \,. \tag{4}$$

This score can serve as a real-versus-generated classification logit under equal class priors, as detailed in Section 4.2. We define the KDE ratio loss $\widehat{\mathcal{L}}_{\varphi,\theta}^{\mathrm{KDE}} := -\frac{\tau}{B}\sum_{i=1}^B s_\varphi^{\mathrm{KDE}}(G_\theta(z_i))$. For a fixed encoder, the feature-space drifting field is

$$\widehat{V}_\varphi(u) = \frac{\sum_{j=1}^B k_\tau(u, v_j)(v_j - u)}{\sum_{j=1}^B k_\tau(u, v_j)} - \frac{\sum_{i=1}^B k_\tau(u, u_i)(u_i - u)}{\sum_{i=1}^B k_\tau(u, u_i)} \,, \tag{5}$$

Then the drift regression loss in the representation space is

$$\widehat{\mathcal{L}}_{\varphi,\theta}^{\mathrm{Drift}} := \frac{1}{B}\sum_{i=1}^B \left\| E_\varphi(G_\theta(z_i)) - \mathrm{sg}(u_i + \widehat{V}_\varphi(u_i)) \right\|_2^2 . \tag{6}$$

During the generator update, the encoder parameters φ are held fixed, while gradients are still propagated through E_φ w.r.t. its input. Consequently, the feature-space drifting direction is pulled back

to the generator parameters through the Jacobians of the encoder and generator. The following proposition makes this connection explicit by relating the generator gradient of the KDE-based log-density-ratio score to the feature-space drifting field.

Proposition 1 (Equivalence of representation-space generator gradients)**.** *Let $B \geq 1$ and $\tau > 0$. For each sampled input z_i, let $\hat{x}_i = G_\theta(z_i)$ and $u_i = E_\varphi(\hat{x}_i)$. Assume that G_θ is differentiable w.r.t. θ at each z_i and that E_φ is differentiable w.r.t. its input at each $\hat{x}_i$. Let $J_E(x) = \partial E_\varphi(x)/\partial x$ and $J_G(z) = \partial G_\theta(z)/\partial\theta$. Use the Gaussian kernel in Eq. (1), with the same bandwidth and feature anchors in the KDE ratio loss $\widehat{\mathcal{L}}^{KDE}_{\varphi,\theta}$ and the drift regression loss $\widehat{\mathcal{L}}^{Drift}_{\varphi,\theta}$ defined above. Hold φ and the feature anchors fixed, differentiating the KDE ratio loss only through the generated queries $E_\varphi(G_\theta(z_i))$. With no velocity clipping, the gradients evaluated at the current parameters used to construct the detached regression targets satisfy*

$$\nabla_\theta \widehat{\mathcal{L}}^{KDE}_{\varphi,\theta} = \nabla_\theta \widehat{\mathcal{L}}^{Drift}_{\varphi,\theta} = -\frac{2}{B}\sum_{i=1}^{B} J_G(z_i)^\top J_E(\hat{x}_i)^\top \widehat{V}_\varphi(u_i).$$

The proof is given in Appendix D.1. Pixel-space drifting is the special case $E_\varphi = \mathrm{Id}$; Appendix D.2 gives its original formulation and compares the two cases. Appendix D.3 discusses how this KDE-based classification score relates to density-ratio estimates induced by GAN critics (Yi et al., 2023).

Learning a discriminative geometry for generator supervision. Proposition 1 shows that the same KDE score used to distinguish real and generated samples also supplies the generator supervision in unclipped drifting. The encoder changes both the kernel weights defining this score and the gradients propagated to the generator. This motivates training E_φ to learn a discriminative geometry, rather than fixing the geometry throughout training. We optimize the KDE classification objective introduced next to expose current real–generated discrepancies, then update the generator using the corresponding drift. The gradient identity does not imply that every representation improves on pixels, or that stronger batch discrimination guarantees better generation; Figures 1 and 2 examine drift alignment, KDE discrimination, and generation quality empirically.

4.2 Learning Discriminative Geometry through Persistent Representations

Persistent Representation Learning. Standard drifting constructs sample relationships in a fixed representation, so the discriminative geometry cannot adapt during training. We now train E_φ and retain its parameters across batches throughout training. This encoder is used only to define the representation space for drifting, while G_θ continues to output images in pixel space. The KDE estimates and drifting field are reconstructed from each current batch. This allows earlier batch samples to shape subsequent discriminative geometry without storing them. Section 5.3 evaluates persistence in representation against maintaining persistence in a scalar critic or both. In our from-scratch experiments, $E_\varphi(x) := x + R_\varphi(x)$ with $R_\varphi : \mathbb{R}^d \to \mathbb{R}^d$ preserves the pixel dimension d, which allows E_φ to be initialized exactly as the identity mapping; R_φ is implemented as a lightweight variant of a Dilated Residual Network (DRN) (Yu et al., 2017), with architecture details in Appendix A.2.

Learning a Discriminative Representation. The representation determines the pairwise distances and kernel weights entering the KDE estimates, and therefore directly affects the density-ratio estimate from which the drifting field is subsequently constructed. We train E_φ so that the KDE comparison used by Drifting Models more clearly exposes differences between current real and generated samples. Before each generator update, we perform K encoder updates using fresh real and generated batches, with $\hat{x}_i = \mathrm{sg}(G_\theta(z_i))$. For these encoder updates, we use s^{KDE}_φ defined in Section 4.1. This score uses no additional trainable scalar head, and its gradient propagates through the KDE computation to the encoder.

To classify real and generated samples, we use equal class priors, since each classification batch contains equal numbers of equally weighted samples from the two classes. By Bayes' rule, the KDE-based posterior in representation space is

$$\widehat{P}(y = \mathrm{real} \mid u) = \frac{\widehat{p}_{\varphi,\tau}(u)}{\widehat{p}_{\varphi,\tau}(u) + \widehat{q}^{\theta}_{\varphi,\tau}(u)} = \sigma\big(s^{\mathrm{KDE}}_\varphi(x)\big)\,,$$

where $u = E_\varphi(x)$ and $\sigma(t) = (1 + e^{-t})^{-1}$. Thus, the score serves as a classification logit: positive values favor real samples and negative values favor generated samples. Using this posterior, we train

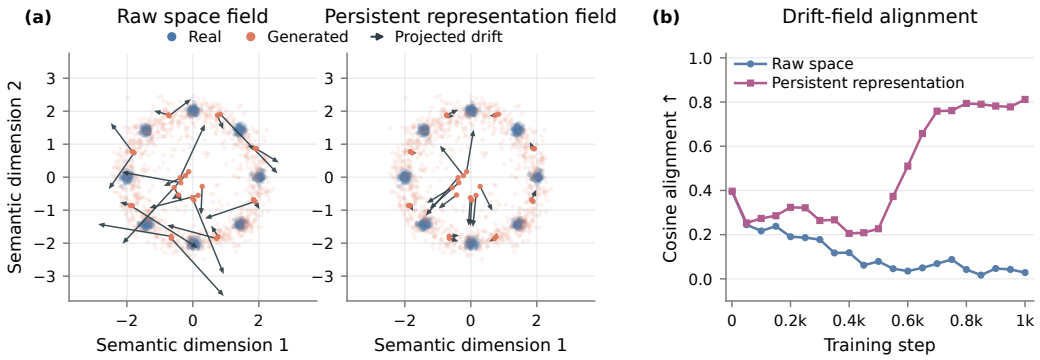

Figure 1: **Persistent Representation Learning Improves Drift Alignment.** **(a)** Drift field constructed in the raw space is disorganized by the nuisance coordinates, while the persistent representation field aligns with the ring shaped target structure. **(b)** Persistent representation space field reaches substantially higher alignment with the reference field than raw space as training progresses.

the encoder with binary cross-entropy, summed over the two class-wise averages:

$$\widehat{\mathcal{L}}_{\varphi}^{D} := -\frac{1}{B}\sum_{j=1}^{B}\log\sigma\big(s_{\varphi}^{\mathrm{KDE}}(\tilde{x}_j)\big) - \frac{1}{B}\sum_{i=1}^{B}\log\big(1-\sigma\big(s_{\varphi}^{\mathrm{KDE}}(\hat{x}_i)\big)\big)\,. \tag{7}$$

This objective encourages higher KDE scores for real samples and lower scores for generated samples, with the aim of learning a more discriminative geometry for constructing the drifting field.

This phase updates only the representation. The generator continues to learn from an explicitly constructed drifting field. After K representation updates, we freeze φ and draw fresh real samples and noise variables. We then construct the feature-space drift in Eq. (5) and update the generator using Eq. (6), with the velocity control described below.

Controlling the Representation-Induced Drift Velocity. The density-ratio connection further allows us to characterize the magnitude of the induced drift. In particular, the local regularity of the density-ratio potential bounds the corresponding drift velocity. We use this relation to motivate a direct constraint on the velocity applied in the generator update.

For the Gaussian kernel, let $\ell_{\varphi}(u) = \log\widehat{p}_{\varphi,\tau}(u) - \log\widehat{q}^{\theta}_{\varphi,\tau}(u)$ denote the feature-space log-density ratio, so that $s_{\varphi}^{\mathrm{KDE}}(x) = \ell_{\varphi}(E_{\varphi}(x))$. With the anchors fixed, the identity underlying Proposition 1 gives $\widehat{V}_{\varphi}(u) = \frac{\tau}{2}\nabla_u\ell_{\varphi}(u)$. Thus, learning the representation changes both the direction and magnitude of the drift. If $\|\nabla_u\ell_{\varphi}(u)\|_2 \le L$ in a region, then $\|\widehat{V}_{\varphi}(u)\|_2 \le \tau L/2$ there. This motivates directly bounding the feature-space displacement used in the drift regression target. The analogy with Lipschitz control of GAN critics is discussed in Appendix D.3.

This relation suggests two forms of control: spectral normalization of the encoder indirectly constrains local representation changes, whereas a direct constraint acts on the feature-space drift received by the generator. We use the latter and compare the two approaches in Section 5.3. Given a velocity budget $V_{\max} > 0$, we define

$$\widehat{V}_{\varphi,\mathrm{clip}}(u) = \widehat{V}_{\varphi}(u)/\max\Big(1, \|\widehat{V}_{\varphi}(u)\|_2/V_{\max}\Big)\,. \tag{8}$$

This radial projection maps zero drift to zero, preserves the direction of nonzero drift, and guarantees $\|\widehat{V}_{\varphi,\mathrm{clip}}(u)\|_2 \le V_{\max}$. The generator subsequently uses $\widetilde{u} = \mathrm{sg}(u + \widehat{V}_{\varphi,\mathrm{clip}}(u))$ as its regression target. This bound controls the feature-space displacement; the generator parameter gradient also depends on the encoder and generator Jacobians.

5 Experiments

To evaluate the proposed method and investigate the factors that shape its performance, we organize our experiments around three research questions. **RQ1:** How does representation geometry shape

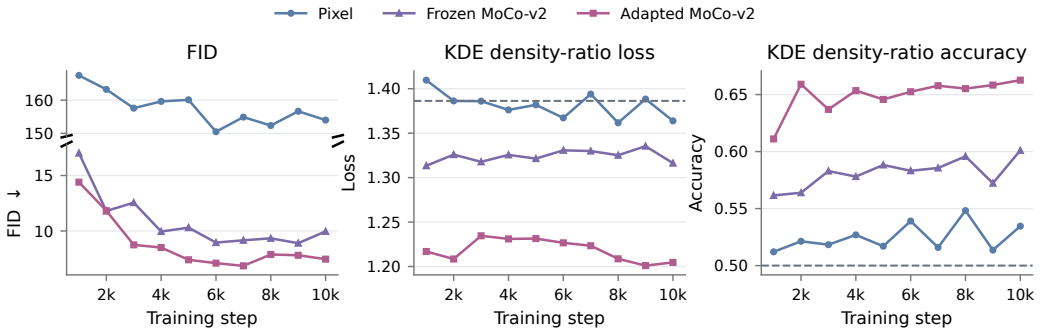

Figure 2: **Persistent Representations Improve Discrimination and Generation.** Pixel-space KDE discrimination remains near chance, while a frozen MoCo-v2 representation improves both discrimination and sample quality, and persistent adaptation improves them further. Dashed lines indicate the loss for indistinguishable distributions and chance-level accuracy, respectively.

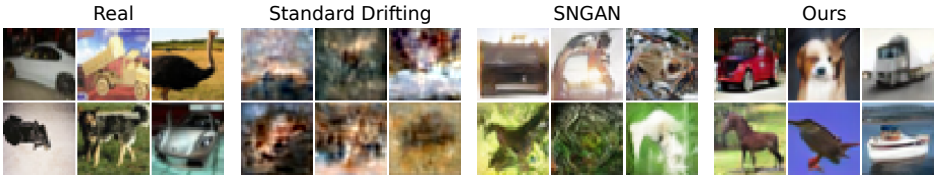

Figure 3: **Qualitative Comparisons on CIFAR-10.** Our method produces substantially clearer and more recognizable samples than standard drifting, with more coherent object structure than SNGAN.

drift estimation, and what are its consequences for KDE discrimination and sample quality? **RQ2:** Can persistent representation learning enable effective drifting without pretrained features? **RQ3:** How do persistence in different components and drift velocity control affect sample quality? We address these questions through mechanistic analysis, benchmark evaluations, and ablation studies.

Following DriftXpress (Falahati et al., 2026), we use U-Net (Ronneberger et al., 2015) as the generator and conduct experiments on SVHN (Netzer et al., 2011), CIFAR-10 (Krizhevsky & Hinton, 2009), and CIFAR-100 (Krizhevsky & Hinton, 2009). We report FID and Inception Score (IS) on 50K generated images, with detailed architectures, optimization settings, and training configurations provided in Appendix A. We use *standard drifting* to denote drifting directly in pixel space. Our experiments focus on isolating the effect of persistent representation learning relative to standard drifting, rather than reproducing computationally intensive large-scale results in Deng et al. (2026).

5.1 Mechanistic Analysis

Learning Discriminative Geometry through Persistent Representations. To directly evaluate how representation geometry affects drift estimation, we use a controlled toy example with eight Gaussian modes in a known two-dimensional semantic space and 30 Gaussian nuisance dimensions; full settings are given in Appendix A. Figure 1(a) compares drift fields computed on the same generated samples, with kernel weights determined in the raw space and learned representation space. The raw-space field varies sharply across nearby samples, whereas the learned representation produces a more coherent field around the target modes. Figure 1(b) further shows substantially higher cosine alignment with a reference KDE drift field computed using only the two semantic coordinates, where alignment is measured between the projected estimated drift field and the reference drift field at the same generated samples. These results illustrate how a persistent representation can improve batch drift estimation when raw distances are affected by nuisance variation.

Effect of Better Geometry for Discrimination with Persistent Representation. We next examine how representation geometry affects KDE discrimination and sample quality on image data. We compare three representations for constructing the drifting field on CIFAR-10: raw pixels, a frozen MoCo-v2 encoder (He et al., 2020; Chen et al., 2020), and the same encoder with its final residual

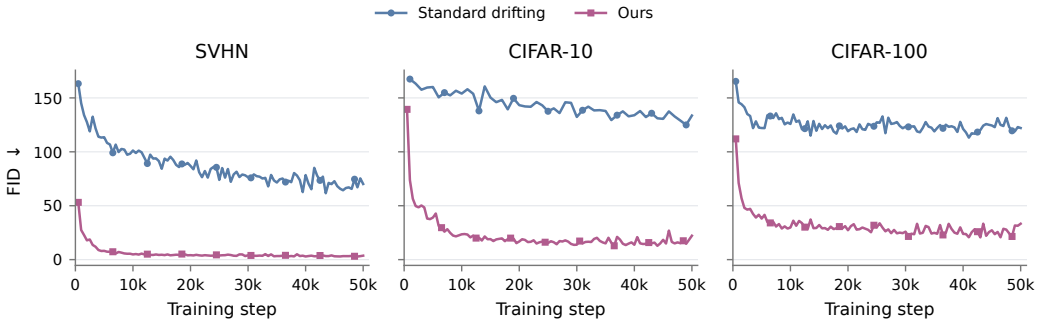

Figure 4: **FID over Training Steps.** Our method converges to substantially lower FID than standard drifting on SVHN, CIFAR-10, and CIFAR-100.

block adapted by persistent representation learning. Alongside FID, we report the encoder objective $\widehat{\mathcal{L}}_{\varphi}^{D}$ in Eq. (7) as the KDE classification loss and compute the corresponding classification accuracy by assigning positive scores to real samples and negative scores to generated samples.

Figure 2 shows that pixel-space drifting remains ineffective and KDE density-ratio accuracy close to chance; Appendix B.3 confirms that this is not due to an untuned kernel bandwidth. A frozen MoCo-v2 representation improves the FID, together with higher KDE density-ratio accuracy and lower loss, while adapting its final residual block further improves the FID and discrimination. These results show that the representation significantly affects both KDE discrimination and sample quality.

5.2 Main Benchmark Results

We compare our method with standard drifting to evaluate whether it enables effective drifting without pretrained features. We additionally include SNGAN (Miyato et al., 2018) as an adversarial reference, using a widely used open-source implementation.[2]

Quantitative Results. Our method consistently improves standard drifting across all three datasets. Figure 4 and Table 1 show substantial improvements in both FID and IS on SVHN, CIFAR-10, and CIFAR-100. Our method also achieves better performance than SNGAN across all three datasets.

Qualitative Results. Figure 3 shows that our method produces substantially clearer and more recognizable samples than standard drifting, with more coherent object structure than SNGAN. Appendix C provides additional SVHN samples, Appendix B.1 tracks the learned representation across checkpoints, and Appendix B.4 confirms the gain at equal training time.

Table 1: **Quantitative Comparisons.** Our method substantially improves both FID and IS over standard drifting on SVHN, CIFAR-10, and CIFAR-100, and also outperforms SNGAN.

	SVHN		CIFAR-10		CIFAR-100	
Method	IS ↑	FID ↓	IS ↑	FID ↓	IS ↑	FID ↓
Standard drifting	2.91	61.56	3.18	124.94	3.46	113.14
Ours	**3.24**	**2.88**	**8.69**	**12.81**	**8.52**	**20.61**
SNGAN	3.04	13.16	7.22	28.59	7.96	30.33

5.3 Ablation Studies

Effect of Persistence in Different Components. To study where cross-batch persistence is most effective, we compare four configurations: no persistent component, persistent representation only, persistent scalar critic only, and persistence in both the representation and the scalar critic. As shown in Figure 5, within 10k training steps on CIFAR-10 (Krizhevsky & Hinton, 2009), persistent repre-

[2]https://github.com/christiancosgrove/pytorch-spectral-normalization-gan

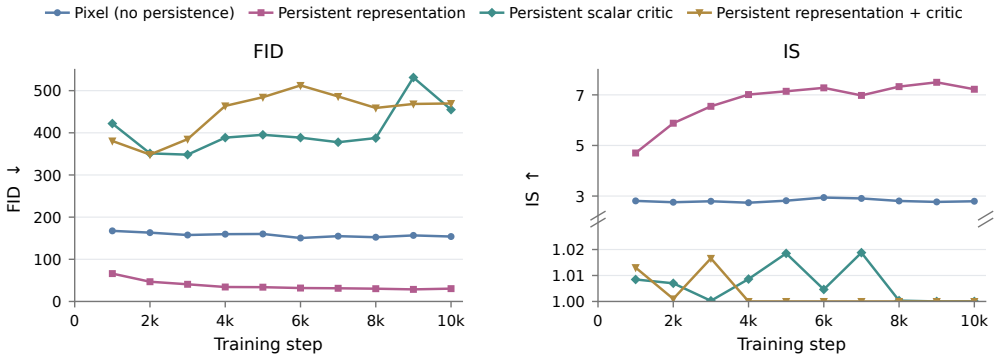

Figure 5: **Effect of Persistence in Different Components.** Among the four configurations, placing persistence only in the representation achieves substantially lower FID and higher IS than the other configurations, whereas introducing a persistent scalar critic degrades performance.

Table 2: **Effect of Lipschitz Regularization.** Velocity clipping outperforms spectral normalization.

Lipschitz regularization	IS ↑	FID ↓
Unconstrained	7.50	27.41
Encoder spectral normalization	7.23	31.82
Velocity clipping		
$V_{\max} = 1$	7.53	25.72
$V_{\max} = 0.5$	7.75	22.19
$V_{\max} = 0.05$	7.66	23.47
$V_{\max} = 0.005$	**7.80**	**21.57**
$V_{\max} = 0.0005$	7.67	22.85

Table 3: **Factorial Ablation.** Persistent representation provides most of the improvement, while velocity clipping further improves performance. Velocity clipping uses $V_{\max} = 1.5$ in pixel space.

Persistence representation	Velocity clipping	IS ↑	FID ↓
×	×	2.94	150.49
×	✓	3.44	126.33
✓	×	7.50	27.41
✓	✓	**7.80**	**21.57**

sentation substantially improves performance over no persistence, whereas introducing persistence in the scalar critic leads to markedly worse results.

Effect of Lipschitz Regularization. Finally, we study whether Lipschitz regularization should be applied to the trainable residual encoder or directly to the drifting velocity. The former uses spectral normalization for every convolution in the encoder, whereas the latter clips the velocity that updates the generator. Table 2 reports the best IS and FID within 10k steps on CIFAR-10. Spectral normalization degrades performance relative to the unconstrained model, whereas velocity clipping improves both metrics for all evaluated budgets, with $V_{\max} = 0.005$ achieving the best performance. These results support constraining the explicit velocity rather than the encoder; Appendix B.2 shows the pre-clipping velocity distribution and the locations of the evaluated budgets. Table 3 further separates the effects of persistent representation and velocity clipping. Persistent representation provides the dominant improvement, while velocity clipping yields additional gains.

6 Conclusion

We studied whether Drifting Models can learn the representation geometry required for effective drifting without relying on fixed pretrained features. Our analysis identifies a representation bottleneck: the geometry determines kernel weights and drift directions, while fixed pixel space provides weak discrimination performance. We addressed this bottleneck with persistent representation learning, which continuously updates the representation while retaining information from successive discriminative updates, so that the geometry used to construct the drifting field evolves together with the generator. We further connect the KDE density-ratio potential to the empirical drifting field. This connection relates the local regularity of the potential to the drifting velocity and motivates velocity clipping. Experiments show that our method improves sample quality significantly without pretrained encoders, and that adapting pretrained encoders and velocity clipping yield further gains.

AI USE STATEMENT

We used generative AI tools to assist with the implementation and editing of limited portions of the experimental code. All AI-assisted code was reviewed, verified, and tested by the authors before use. Generative AI tools were not used to generate synthetic datasets, develop theoretical models or conceptual frameworks, formulate or prove mathematical claims, design the research methodology or experiments, perform data analysis, or interpret the results.

We also used generative AI tools to improve the grammar, clarity, and phrasing of the paper. We reviewed all AI-assisted work and take responsibility for the final content of this work, including all text, claims, code, results, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work does not involve human subjects or release a new dataset; all experiments use previously released datasets. As a method for improving existing generative models, our approach is potentially dual-use and could be applied to generate misleading, harmful, or inappropriate content. It may also inherit or amplify biases, privacy risks, and other limitations of the underlying data and pretrained models. Appropriate safeguards and compliance with the licenses and intended-use conditions of the relevant models and datasets should therefore accompany its deployment.

REPRODUCIBILITY STATEMENT

We provide detailed experimental configurations for reproducibility in Appendix A. Appendix D provides the derivation of the matched KDE ratio and drift regression loss gradient identity and discusses the GAN connection. Appendix E provides the generator-update pseudocode.

REFERENCES

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein GAN. *arXiv preprint arXiv:1701.07875*, December 2017. doi: 10.48550/arXiv.1701.07875.

Krishnakumar Balasubramanian. Finite-Particle Convergence Rates for Conservative and Non-Conservative Drifting Models. *arXiv preprint arXiv:2605.22795*, May 2026. doi: 10.48550/arXiv.2605.22795.

Andreas Blattmann, Tim Dockhorn, Sumith Kulal, Daniel Mendelevitch, Maciej Kilian, Dominik Lorenz, Yam Levi, Zion English, Vikram Voleti, Adam Letts, Varun Jampani, and Robin Rombach. Stable Video Diffusion: Scaling Latent Video Diffusion Models to Large Datasets. *arXiv preprint arXiv:2311.15127*, November 2023. doi: 10.48550/arXiv.2311.15127.

Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved Baselines with Momentum Contrastive Learning. *arXiv preprint arXiv:2003.04297*, March 2020. doi: 10.48550/arXiv.2003.04297.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, June 2009. doi: 10.1109/CVPR.2009.5206848.

Mingyang Deng, He Li, Tianhong Li, Yilun Du, and Kaiming He. Generative Modeling via Drifting. *arXiv preprint arXiv:2602.04770*, February 2026. doi: 10.48550/arXiv.2602.04770.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *arXiv preprint arXiv:2010.11929*, June 2021. doi: 10.48550/arXiv.2010.11929.

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, Dustin Podell, Tim Dockhorn, Zion English, and Robin Rombach. Scaling Rectified Flow Transformers for High-Resolution Image Synthesis. In *Forty-First International Conference on Machine Learning*, June 2024.

Ali Falahati, Elliot Creager, Gautam Kamath, and Shubhankar Mohapatra. DriftXpress: Faster Drifting Models via Projected RKHS Fields. *arXiv preprint arXiv:2605.12183*, May 2026. doi: 10.48550/arXiv.2605.12183.

Kevin Frans, Danijar Hafner, Sergey Levine, and Pieter Abbeel. One Step Diffusion via Shortcut Models. *arXiv preprint arXiv:2410.12557*, June 2025. doi: 10.48550/arXiv.2410.12557.

Zhengyang Geng, Mingyang Deng, Xingjian Bai, J. Zico Kolter, and Kaiming He. Mean Flows for One-step Generative Modeling. *arXiv preprint arXiv:2505.13447*, May 2025. doi: 10.48550/arXiv.2505.13447.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Networks. *arXiv preprint arXiv:1406.2661*, June 2014. doi: 10.48550/arXiv.1406.2661.

Arthur Gretton, Li Kevin Wenliang, Alexandre Galashov, James Thornton, Valentin De Bortoli, and Arnaud Doucet. On the Wasserstein Gradient Flow Interpretation of Drifting Models. *arXiv preprint arXiv:2605.05118v2*, May 2026. doi: 10.48550/arXiv.2605.05118.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved Training of Wasserstein GANs. *arXiv preprint arXiv:1704.00028*, December 2017. doi: 10.48550/arXiv.1704.00028.

Jiaqi Han, Puheng Li, Qiushan Guo, Renyuan Xu, Stefano Ermon, and Emmanuel J. Candès. One-Step Generative Modeling via Wasserstein Gradient Flows. *arXiv preprint arXiv:2605.11755*, May 2026. doi: 10.48550/arXiv.2605.11755.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, June 2016. doi: 10.1109/CVPR.2016.90.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum Contrast for Unsupervised Visual Representation Learning. *arXiv preprint arXiv:1911.05722*, March 2020. doi: 10.48550/arXiv.1911.05722.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising Diffusion Probabilistic Models. In *Advances in Neural Information Processing Systems*, volume 33, pp. 6840–6851. Curran Associates, Inc., June 2020.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of Neural Network Representations Revisited. In *Proceedings of the 36th International Conference on Machine Learning*, pp. 3519–3529. PMLR, May 2019.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

Chieh-Hsin Lai, Bac Nguyen, Naoki Murata, Yuhta Takida, Toshimitsu Uesaka, Yuki Mitsufuji, Stefano Ermon, and Molei Tao. A Unified View of Score-Based and Drifting Models. *arXiv preprint arXiv:2603.07514*, May 2026. doi: 10.48550/arXiv.2603.07514.

HakGeun Lee and Hyonho Chun. Identifiability and Stability of Generative Drifting with Companion-Elliptic Kernel Families. *arXiv preprint arXiv:2604.24196*, June 2026. doi: 10.48550/arXiv.2604.24196.

Abdelhak Lemkhenter, Adam Bielski, Alp Eren Sari, and Paolo Favaro. Generative Adversarial Learning via Kernel Density Discrimination. *arXiv preprint arXiv:2107.06197*, July 2021. doi: 10.48550/arXiv.2107.06197.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow Matching for Generative Modeling. *arXiv preprint arXiv:2210.02747*, October 2022.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow Straight and Fast: Learning to Generate and Transfer Data with Rectified Flow. *arXiv preprint arXiv:2209.03003*, September 2022.

Xingchao Liu, Xiwen Zhang, Jianzhu Ma, Jian Peng, and Qiang Liu. InstaFlow: One Step is Enough for High-Quality Diffusion-Based Text-to-Image Generation. *arXiv preprint arXiv:2309.06380*, September 2023.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral Normalization for Generative Adversarial Networks. *arXiv preprint arXiv:1802.05957*, February 2018. doi: 10.48550/arXiv.1802.05957.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Baolin Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, volume 2011, pp. 4. Granada, 2011.

Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. F-GAN: Training Generative Neural Samplers using Variational Divergence Minimization. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-Resolution Image Synthesis With Latent Diffusion Models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10684–10695, December 2021.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi (eds.), *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pp. 234–241, Cham, 2015. Springer International Publishing. ISBN 978-3-319-24574-4. doi: 10.1007/978-3-319-24574-4_28.

Tim Salimans and Jonathan Ho. Progressive Distillation for Fast Sampling of Diffusion Models. *arXiv preprint arXiv:2202.00512*, June 2022. doi: 10.48550/arXiv.2202.00512.

Filippo Santambrogio. {Euclidean, metric, and Wasserstein} gradient flows: An overview. *Bulletin of Mathematical Sciences*, 7(1):87–154, April 2017. ISSN 1664-3615. doi: 10.1007/s13373-017-0101-1.

Axel Sauer, Kashyap Chitta, Jens Müller, and Andreas Geiger. Projected GANs Converge Faster. *arXiv preprint arXiv:2111.01007*, November 2021. doi: 10.48550/arXiv.2111.01007.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising Diffusion Implicit Models. *arXiv preprint arXiv:2010.02502v4*, October 2020. doi: 10.48550/arXiv.2010.02502.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency Models. *arXiv preprint arXiv:2303.01469*, May 2023. doi: 10.48550/arXiv.2303.01469.

Erkan Turan, Nicolas Dufour, and Maks Ovsjanikov. Generative Drifting is Secretly Score Matching: A Spectral and Variational Perspective. *arXiv preprint arXiv:2603.09936*, May 2026. doi: 10.48550/arXiv.2603.09936.

Zhuoyi Yang, Jiayan Teng, Wendi Zheng, Ming Ding, Shiyu Huang, Jiazheng Xu, Yuanming Yang, Wenyi Hong, Xiaohan Zhang, Guanyu Feng, Da Yin, Xiaotao Gu, Yuxuan Zhang, Weihan Wang, Yean Cheng, Ting Liu, Bin Xu, Yuxiao Dong, and Jie Tang. CogVideoX: Text-to-Video Diffusion Models with An Expert Transformer. *arXiv preprint arXiv:2408.06072*, October 2024. doi: 10.48550/arXiv.2408.06072.

Mingxuan Yi, Zhanxing Zhu, and Song Liu. MonoFlow: Rethinking Divergence GANs via the Perspective of Wasserstein Gradient Flows. *arXiv preprint arXiv:2302.01075*, August 2023. doi: 10.48550/arXiv.2302.01075.

Fisher Yu, Vladlen Koltun, and Thomas Funkhouser. Dilated Residual Networks. *arXiv preprint arXiv:1705.09914*, May 2017. doi: 10.48550/arXiv.1705.09914.

Jiawei Zhang, Mengfei Xia, Gen Li, and Yuantao Gu. Distilling Drifting Transformers with Representation Autoencoders. *arXiv preprint arXiv:2606.15553*, August 2026. doi: 10.48550/arXiv.2606.15553.

APPENDIX

A IMPLEMENTATION DETAILS

Table 4 summarizes the configurations and hyperparameters for the main benchmark (Table 1) and the ablation studies (Section 5.3).

Table 4: **Configurations for the Main Benchmark and Ablation Studies.**

	Ablation default	Standard drifting	Ours
Generator			
Architecture		U-Net	
Base channels, mult.		128, (1, 2, 2, 2)	
Res. blocks / attn. res.		2 per stage; 16×16	
Generator Optimizer			
Optimizer		AdamW	
Betas		$(0.9, 0.999)$	
Weight decay		0	
Learning rate		2×10^{-4}	
Batch size (real & generated)		1,500 per GPU	
Encoder			
Architecture	DRN-like (App. A.2)	–	DRN-like
Encoder Optimizer			
Optimizer	AdamW	–	AdamW
Betas	$(0.9, 0.999)$	–	$(0.9, 0.999)$
Weight decay	0	–	0
Learning rate	10^{-4}	–	10^{-4}
Batch size (real & generated)	1,500 per GPU	–	1,500 per GPU
Update schedule	1 step, per generator step	–	1 step, per generator step
Loss Computation			
Mechanism		stateless batch KDE	
KDE temperature		0.05	
Regularization			
$V_{\max}$	0.005	–	0.005
Training			
GPUs		4× NVIDIA GH200	
Training steps	10,000	50,000	50,000

A.1 ADDITIONAL CONFIGURATION SETTINGS

Toy Experiment Setting. The semantic variable in the nuisance-dimension toy experiment (Section 5.1) follows a ring of eight isotropic Gaussians with radius 2.0 and standard deviation 0.08. Each sample is paired with an independent 30-dimensional standard Gaussian nuisance vector, scaled by 1.0 and concatenated with the two semantic dimensions to form the 32-dimensional observation used to construct the drift field. The generator is a three-hidden-layer MLP with 256 hidden units and SiLU activations and produces 32-dimensional samples from Gaussian noise. The persistent representation uses an identity-initialized residual MLP with one 64-unit hidden layer, following the same residual parameterization as the DRN encoder. The generator and the representation both use AdamW with a learning rate of 10^{-3}, weight decay 0, and a batch size of 256, updating the representation once per generator step over 1,000 steps with a KDE temperature of 0.5. For the alignment diagnostic in Figure 1(b), we project the estimated drift onto the two-dimensional semantic space and compare its direction with a reference field constructed from independent sets of 4,096 real and generated samples in that space.

Persistent Scalar Critic Setting. The persistent critic uses the SNGAN discriminator architecture (Miyato et al., 2018) with 128 base channels, trained with AdamW using a logistic objective over real and generated samples, a learning rate of 10^{-4}, a batch size of $4 \times 1{,}500$, one update per generator step, and gradients clipped to a norm of 1. Its scaled output supplies the drift for the drift regression loss through the gradient with respect to the input, following the same construction used throughout the paper. In the joint configuration, the critic is applied to the same DRN-style encoder described in Appendix A.2, trained jointly under the same optimizer settings.

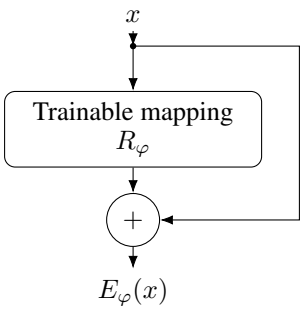

Figure 6: **Persistent Representation Learning with a Residual Parameterization.**

MoCo-v2 Adaptation Setting. We initialize a ResNet-50 backbone from the official 800-epoch MoCo v2 checkpoint (He et al., 2020; Chen et al., 2020) pretrained on ImageNet (Deng et al., 2009), resizing inputs to 112×112 with standard ImageNet normalization. The frozen configuration keeps the backbone fixed, while the adapted configuration unfreezes only the final residual block, updated with the same persistent schedule as the DRN encoder using AdamW with a learning rate of 10^{-6}, weight decay 0, a batch size of $4 \times 1{,}500$, and gradients clipped to a norm of 1.

A.2 Persistent Representation Architecture

Persistent representation learning accommodates encoder architectures tailored to the data modality. We explored several common choices, including ResNet-style convolutional networks (He et al., 2016), U-Nets (Ronneberger et al., 2015), and Vision Transformers (Dosovitskiy et al., 2021). For the main experiments, we use a lightweight DRN variant. Although simple, this architecture is sufficient to provide an effective learned representation for drifting.

Our DRN-style encoder preserves the input resolution throughout the network and uses the residual parameterization $E_\varphi(x) = x + R_\varphi(x)$, illustrated in Figure 6. The residual branch begins with a 3×3 convolution from three input channels to 128 hidden channels, followed by four constant-width residual blocks with dilation rates $(1, 2, 4, 1)$ and SiLU activations. Each block contains two 3×3 convolutions and an identity skip connection. A final 3×3 projection maps the hidden features back to three channels. We use neither spatial downsampling nor normalization layers. Initializing the final projection to zero gives $E_{\varphi_0}(x) = x$ exactly, while subsequent training modifies the representation through the learned residual. As in Drifting Models (Deng et al., 2026; Falahati et al., 2026), the drifting objective can also extend to multiple feature groups, which we extract from the input, the stem output, each residual-block output, and the final encoded image.

B Additional Analyses

B.1 Evolution of the Learned Representation Geometry

To study the evolution of the learned representation geometry, we analyze checkpoints of the persistent encoder trained from scratch on CIFAR-10 (Krizhevsky & Hinton, 2009). We compare their representations using linear centered kernel alignment (CKA) (Kornblith et al., 2019). CKA measures scale-invariant similarity between the pairwise sample relations induced by two encoders, with higher values indicating greater similarity.

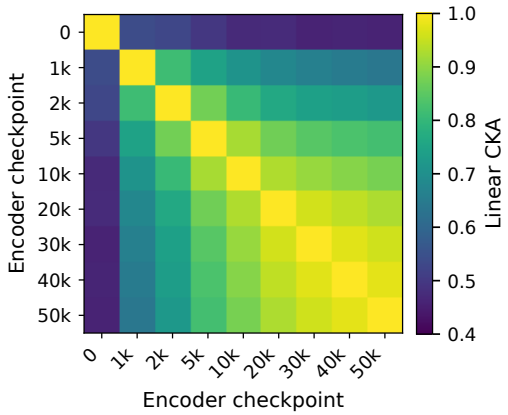

Figure 7: **Representation Similarity across Training.** Linear CKA between encoder checkpoints shows a gradual departure from the identity initialization that stabilizes among late-training checkpoints.

As shown in Figure 7, persistent representation learning produces a gradual change in the geometry used to construct the drifting field. The learned representation progressively departs from its identity initialization, with CKA against the initial encoder decreasing to 0.46 at 50k steps, while neighboring late-training checkpoints remain highly similar, with CKA above 0.95 from 30k onward.

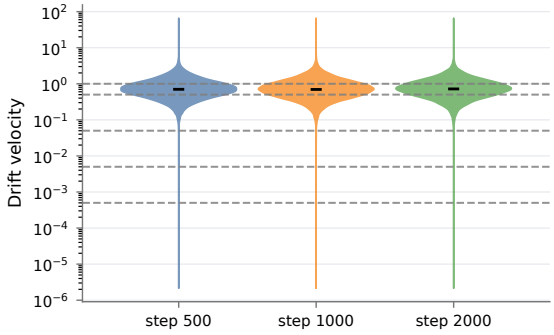

Figure 8: **Velocity Budgets Relative to Drift Magnitudes.** Distribution of $\|\widehat{V}_\varphi(u_i)\|_2$ at steps 500, 1000, and 2000 of the $V_{\max} = 0.005$ run. Dashed lines mark the five $V_{\max}$ budgets compared in Table 2.

B.2 Drift-Magnitude Distributions and Velocity Budgets

Figure 8 presents the distribution of the drift magnitude $\|\widehat{V}_\varphi(u_i)\|_2$ (Eq. (8)) immediately before clipping, measured at training steps 500, 1000, and 2000 of the $V_{\max} = 0.005$ run. The dashed lines indicate the five velocity budgets evaluated in Table 2. The three distributions have similar shapes, with medians remaining approximately 0.70 across steps 500, 1000, and 2000. The $V_{\max} = 1$ and

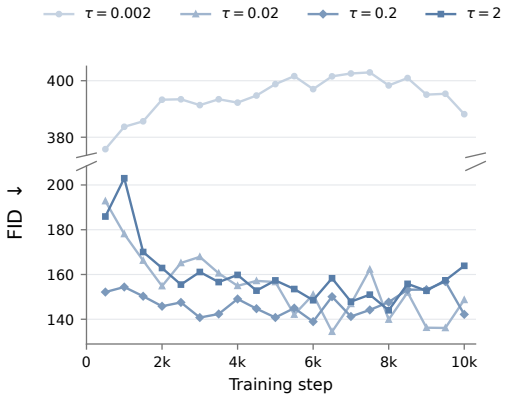

Figure 9: **Effect of Kernel Bandwidth on Pixel-space Drifting.** FID over training steps on CIFAR-10 for four KDE temperatures $\tau \in \{0.002, 0.02, 0.2, 2\}$ in pixel space. None approaches the 27.41 FID achieved by persistent representation learning within the same 10k-step budget.

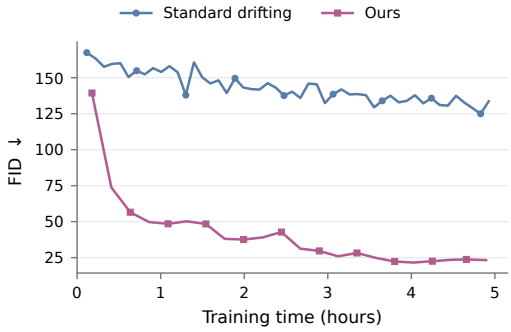

Figure 10: **FID under the Same Training Time on CIFAR-10.** FID against training time within 5 hours.

$V_{\max} = 0.5$ lines intersect the displayed distributions, whereas the budgets at or below 0.05 lie beneath their main mass.

B.3 Kernel Bandwidth in Pixel Space

Changing the KDE temperature alone cannot make Drifting Models train well in pixel space. We sweep the temperature τ over three orders of magnitude (0.002 to 2) in pixel space, training each configuration for 10k steps on CIFAR-10. As shown in Figure 9, FID remains between 134 and 376 throughout training across the sweep, far above the 27.41 achieved by persistent representation learning within the same step budget. This indicates that the ineffectiveness of pixel-space drifting reflects the representation geometry itself, not merely a poorly tuned kernel bandwidth.

B.4 Training Time Cost Comparison

We compare standard drifting and our method under the same training time budget. Persistent representation learning adds an encoder update and the encoder forward and backward passes to each training step. Figure 10 reports FID against training time on CIFAR-10 (Krizhevsky & Hinton, 2009), and with four GH200 GPUs, our method reaches an FID of 21.57, compared with 124.94 for standard drifting.

C ADDITIONAL QUALITATIVE RESULTS

Figure 11 provides additional samples from our method on SVHN (Netzer et al., 2011).



Figure 11: **Additional Qualitative Results on SVHN.** Samples generated by our method at the checkpoint attaining the best FID of 2.88.

D KDE RATIO LOSS, DRIFT REGRESSION LOSS, AND THE GAN LOSS CONNECTION

D.1 PROOF OF PROPOSITION 1

Proposition 1 (Equivalence of representation-space generator gradients, restated)**.** *Let $B \geq 1$ and $\tau > 0$. For each sampled input z_i, let $\hat{x}_i = G_\theta(z_i)$ and $u_i = E_\varphi(\hat{x}_i)$. Assume that G_θ is differentiable with respect to θ at each z_i and that E_φ is differentiable with respect to its input at each $\hat{x}_i$. Write $J_E(x) = \partial E_\varphi(x)/\partial x$ and $J_G(z) = \partial G_\theta(z)/\partial \theta$. Use the Gaussian kernel in Eq. (1), with the same bandwidth and feature anchors in the KDE ratio loss $\widehat{\mathcal{L}}^{KDE}_{\varphi,\theta}$ and the drift regression loss $\widehat{\mathcal{L}}^{Drift}_{\varphi,\theta}$ defined in Section 4.1. Hold φ and the feature anchors fixed, differentiating the KDE ratio loss only through the generated queries $E_\varphi(G_\theta(z_i))$. With no velocity clipping, the gradients evaluated at the current parameters used to construct the detached regression targets satisfy*

$$\nabla_\theta \widehat{\mathcal{L}}^{KDE}_{\varphi,\theta} = \nabla_\theta \widehat{\mathcal{L}}^{Drift}_{\varphi,\theta} = -\frac{2}{B}\sum_{i=1}^{B} J_G(z_i)^\top J_E(\hat{x}_i)^\top \widehat{V}_\varphi(u_i).$$

Proof. For $\tau > 0$, both representation-space Gaussian KDEs are positive. Let $v_j = E_\varphi(\tilde{x}_j)$ denote the real feature anchors. Hold the anchor values $\{u_r\}_{r=1}^B$ and $\{v_j\}_{j=1}^B$ fixed; the generated query $u_i = E_\varphi(G_\theta(z_i))$ remains differentiable through G_θ and E_φ. Differentiating with respect to a feature query u gives

$$\nabla_u \log \widehat{p}_{\varphi,\tau}(u) = \frac{2}{\tau}\frac{\sum_j k_\tau(u, v_j)(v_j - u)}{\sum_j k_\tau(u, v_j)},$$
$$\nabla_u \log \widehat{q}^{\theta}_{\varphi,\tau}(u) = \frac{2}{\tau}\frac{\sum_r k_\tau(u, u_r)(u_r - u)}{\sum_r k_\tau(u, u_r)}.$$

Consequently,

$$\tau \nabla_u \log \frac{\widehat{p}_{\varphi,\tau}(u)}{\widehat{q}^{\theta}_{\varphi,\tau}(u)} = 2\widehat{V}_\varphi(u). \tag{9}$$

With the Jacobians defined in the proposition, $\partial u_i/\partial\theta = J_E(\hat{x}_i)J_G(z_i)$. Applying the chain rule to the KDE ratio loss and using Eq. (9) yields

$$\nabla_\theta \widehat{\mathcal{L}}^{\text{KDE}}_{\varphi,\theta} = -\frac{2}{B}\sum_{i=1}^{B} J_G(z_i)^\top J_E(\hat{x}_i)^\top \widehat{V}_\varphi(u_i).$$

For the detached target $\widetilde{u}_i = \text{sg}(u_i + \widehat{V}_\varphi(u_i))$, the target is held constant during differentiation. At the current parameters used to construct it, the residual is $u_i - \widetilde{u}_i = -\widehat{V}_\varphi(u_i)$. Differentiating the drift regression loss therefore gives

$$\nabla_\theta \widehat{\mathcal{L}}^{\text{Drift}}_{\varphi,\theta} = \frac{2}{B}\sum_{i=1}^{B} J_G(z_i)^\top J_E(\hat{x}_i)^\top (u_i - \widetilde{u}_i) = \nabla_\theta \widehat{\mathcal{L}}^{\text{KDE}}_{\varphi,\theta}.$$

This proves the claimed identity under the stated differentiability assumptions. Repeated anchors do not invalidate the argument because the KDEs remain positive. The encoder parameters and anchors are fixed only for this generator-gradient comparison; the encoder-learning phase differentiates through the KDE to update φ. □

Scope of the identity. The two objectives must use the same anchors, bandwidth, and treatment of self-interactions. Differentiating through generated anchors introduces additional terms in the KDE ratio loss. Clipping changes the drift regression loss gradient and generally breaks equality with the KDE ratio loss gradient. The proposition concerns one current-step gradient, and does not assert an equality of losses or entire training trajectories.

D.2 Pixel-Space Special Case and Comparison

Setting $E_\varphi = \mathrm{Id}$ gives $u_i = \hat{x}_i$, $v_j = \tilde{x}_j$, and $J_E = I$. We retain the original pixel-space notation below. We denote the batch size by B. For each batch, let $\{\tilde{x}_j\}_{j=1}^B$ denote the real samples and $\{\hat{x}_i\}_{i=1}^B$ the generated samples, where $\hat{x}_i = G_\theta(z_i), z_i \sim \mu, i \in [B]$. The KDEs estimated from the current batch:

$$\widehat{p}_\tau(x) = \frac{1}{B}\sum_{j=1}^{B} k_\tau(x, \tilde{x}_j), \qquad \widehat{q}_\tau^\theta(x) = \frac{1}{B}\sum_{i=1}^{B} k_\tau(x, \hat{x}_i).$$

The dependence on the current generator is implicit. The corresponding density ratio and scalar potential are

$$\widehat{r}(x) = \frac{\widehat{p}_\tau(x)}{\widehat{q}_\tau^\theta(x)}, \qquad \widehat{U}(x) = \tau \log \widehat{r}(x).$$

In this case, $\tau s_\varphi^{\mathrm{KDE}}(x) = \widehat{U}(x)$. We retain the loss notation from Section 4.1, with $E_\varphi = \mathrm{Id}$ throughout this pixel-space specialization. The KDE ratio loss and drift regression loss therefore become

$$\widehat{\mathcal{L}}_{\varphi,\theta}^{\mathrm{KDE}} = -\frac{1}{B}\sum_{i=1}^{B} \widehat{U}(G_\theta(z_i)), \quad \widehat{\mathcal{L}}_{\varphi,\theta}^{\mathrm{Drift}} = \frac{1}{B}\sum_{i=1}^{B} \left\| G_\theta(z_i) - \mathrm{sg}\Big(\hat{x}_i + \widehat{V}_{p,q_\theta}(\hat{x}_i)\Big) \right\|_2^2.$$

Let $J_i = \partial G_\theta(z_i)/\partial\theta$. For $\tau > 0$, the Gaussian KDEs are positive, so their log gradients are well defined. Holding the anchors fixed and differentiating the query gives

$$\nabla_x \log \widehat{p}_\tau(x) = \frac{2}{\tau}\frac{\sum_{j=1}^B k_\tau(x, \tilde{x}_j)(\tilde{x}_j - x)}{\sum_{j=1}^B k_\tau(x, \tilde{x}_j)},$$
$$\nabla_x \log \widehat{q}_\tau^\theta(x) = \frac{2}{\tau}\frac{\sum_{i=1}^B k_\tau(x, \hat{x}_i)(\hat{x}_i - x)}{\sum_{i=1}^B k_\tau(x, \hat{x}_i)}.$$

Subtracting these expressions and using the definitions of $\widehat{U}$ and $\widehat{V}_{p,q_\theta}$ yields

$$\nabla_x \widehat{U}(x) = \tau\left(\nabla_x \log \widehat{p}_\tau(x) - \nabla_x \log \widehat{q}_\tau^\theta(x)\right) = 2\widehat{V}_{p,q_\theta}(x). \tag{10}$$

Although $\widehat{q}_\tau^\theta$ records the generator that produced the anchors, those anchors remain constant in the differentiation below. The chain rule therefore gives

$$\nabla_\theta \widehat{\mathcal{L}}_{\varphi,\theta}^{\mathrm{KDE}} = -\frac{1}{B}\sum_{i=1}^{B} J_i^\top \nabla_x \widehat{U}(\hat{x}_i) = -\frac{2}{B}\sum_{i=1}^{B} J_i^\top \widehat{V}_{p,q_\theta}(\hat{x}_i).$$

For the drift regression loss, the stop-gradient target is constant. At the current parameters, its residual is $G_\theta(z_i) - \mathrm{sg}(\hat{x}_i + \widehat{V}_{p,q_\theta}(\hat{x}_i)) = -\widehat{V}_{p,q_\theta}(\hat{x}_i)$. Thus

$$\nabla_\theta \widehat{\mathcal{L}}_{\varphi,\theta}^{\mathrm{Drift}} = -\frac{2}{B}\sum_{i=1}^{B} J_i^\top \widehat{V}_{p,q_\theta}(\hat{x}_i) = \nabla_\theta \widehat{\mathcal{L}}_{\varphi,\theta}^{\mathrm{KDE}},$$

which recovers the original pixel-space gradient identity. The argument also covers coincident anchors, since the Gaussian KDEs remain strictly positive.

What changes with the representation? Pixel-space kernels use $\|x-y\|_2^2$, whereas representation-space kernels use $\|E_\varphi(x) - E_\varphi(y)\|_2^2$. Thus an encoder changes which anchors contribute to the KDE scores and drift. In addition, the feature direction is pulled back through $J_E^\top$ before reaching the generator; this factor is the identity in pixel space. These are the two mechanisms by which learning the representation changes generator supervision. The comparison establishes no universal ordering of the two geometries: an encoder that discards relevant differences can also remove useful supervision. The benefits of the learned geometry are evaluated empirically in Figures 1 and 2.

D.3 Connection to Density-Ratio GAN Objectives

Divergence-Based GAN Objectives. Divergence-based GANs (Nowozin et al., 2016; Yi et al., 2023) formulate distribution matching as a minimax problem between a generator G_θ and a discriminator $D : \mathbb{R}^d \to \mathcal{I}$, where $\mathcal{I} \subseteq \mathbb{R}$ is the range of discriminator outputs. We use a variational representation of a divergence between p and q_θ, and let $\phi, \psi : \mathcal{I} \to \mathbb{R}$ be differentiable scalar functions specifying the discriminator's objective contributions for real and generated samples, respectively. For example, the standard logistic GAN uses $\phi(t) = \log \sigma(t)$ and $\psi(t) = \log(1 - \sigma(t))$, where $\sigma(t) = (1 + e^{-t})^{-1}$. The corresponding minimax objective is

$$\min_\theta \max_D \left\{ \mathbb{E}_{\tilde{x}\sim p}[\phi(D(\tilde{x}))] + \mathbb{E}_{z\sim\mu}[\psi(D(G_\theta(z)))] \right\}.$$

For a fixed generator G_θ, let D^* denote a maximizer of the inner problem. In practice, the generator can also be trained with an alternative loss against a fixed discriminator $\min_\theta -\mathbb{E}_{z\sim\mu}[g(D(G_\theta(z)))]$, where $g : \mathcal{I} \to \mathbb{R}$ is a differentiable increasing scalar function. Choosing $g = -\psi$ recovers the generator update of the minimax objective. For the logistic discriminator, $g(t) = \log \sigma(t)$ gives the non-saturating generator loss.

For an unrestricted discriminator class, an interior optimum D^* satisfies the pointwise first-order condition, yielding the following identity wherever $q_\theta(x) > 0$ and $\phi'(D^*(x)) \neq 0$:

$$-\psi'(D^*(x))/\phi'(D^*(x)) = p(x)/q_\theta(x). \tag{11}$$

Thus, the optimal discriminator output can be transformed into the density ratio between the real and generated data distributions. During the generator update, D^* is held fixed, and the input gradient of $D^*(G_\theta(z))$ supplies a distribution-matching direction.

The optimality condition in Eq. (11) shows that the output of a discriminator D induces the density-ratio estimate $r_D(x) := -\psi'(D(x))/\phi'(D(x))$. At the function-space optimum, $r_{D^*}(x) = p(x)/q_\theta(x)$. For a divergence parameterization satisfying $r_D(x) > 0$, define the scalar potential $U_D(x) = h(\log r_D(x))$, where $h : \mathbb{R} \to \mathbb{R}$ is differentiable and strictly increasing. At the optimum,

$$U_{D^*}(x) = h\left(\log \frac{p(x)}{q_\theta(x)}\right), \qquad \nabla_x U_{D^*}(x) = h'\left(\log \frac{p(x)}{q_\theta(x)}\right)(\nabla \log p(x) - \nabla \log q_\theta(x)).$$

The potential $U_D(x)$ assigns higher values to regions where the estimated data density is large relative to the current model density. Holding D fixed, we train the generator to increase this value at its outputs by minimizing $\mathcal{L}_G(\theta; D) = -\mathbb{E}_{z\sim\mu}[U_D(G_\theta(z))]$. The gradient $\nabla_x U_D(x)$ is backpropagated through G_θ to update its parameters, encouraging the generator to allocate more probability mass to regions that are underrepresented relative to the data.

Choosing $h(a) = \tau a$ and replacing the population ratio with the Gaussian KDE ratio $\widehat{r}$ gives the potential $\widehat{U}$ in Appendix D.2. Proposition 1 and its pixel-space specialization establish the exact matched-KDE gradient identity. For a general h, the factor $h'(\log r_D(x))$ changes the local magnitude; a learned discriminator also need not equal the empirical KDE ratio. The proposition therefore does not assert equivalence with arbitrary GAN generator losses. The matched batch is used to compare the two generator objectives, while Algorithm 1 uses fresh batches for the encoder and generator phases.

Connection to velocity control. Spectral normalization controls a discriminator's Lipschitz constant by constraining the spectral norm of each layer and is widely used to stabilize GAN training (Miyato et al., 2018). Motivated by this principle, we examine the corresponding controlled quantity in Drifting Models. The Gaussian-kernel KDE correspondence established in Section 4.1 relates the input gradient of a scalar potential to the explicit drifting velocity. Under the logistic density-ratio parameterization, $r_D(x) = e^{D(x)}$. Taking $h(u) = \tau u$ gives $U_D(x) = \tau D(x)$; within this specialization, the associated drifting velocity satisfies $V(x) = \frac{1}{2}\nabla_x U_D(x) = \frac{\tau}{2}\nabla_x D(x)$. If D is L_D-Lipschitz with respect to the input norm, then $\|\nabla_x D(x)\|_2 \leq L_D$ almost everywhere and consequently $\|V(x)\|_2 \leq \tau L_D/2$. Thus, a Lipschitz constraint on the discriminator corresponds to a bound on particle velocity in the associated drifting formulation.

E A Detailed Comparison of Generator Updates

The following algorithms make explicit the optimization sequences compared in Section 4.1. Drifting directly constructs a vector field and uses the drift regression loss to fit a detached target. A divergence-based GAN first optimizes a discriminator and then updates the generator through the density-ratio potential induced by the fixed discriminator. Here the KDE ratio loss $\widehat{\mathcal{L}}^{\mathrm{KDE}}_{\varphi,\theta}$ and drift regression loss $\widehat{\mathcal{L}}^{\mathrm{Drift}}_{\varphi,\theta}$ use the definitions in Section 4.1, specialized to $E_\varphi = \mathrm{Id}$ as in Appendix D.2. We write $J_i = J_G(z_i) = \partial G_\theta(z_i)/\partial\theta$ in both algorithms. The general GAN objective $\widehat{\mathcal{L}}_G(D^*, G_\theta)$ is denoted separately from the KDE ratio loss.

Algorithm 2: Drifting Models Training

Input: Generator G_θ, real data distribution p, latent distribution μ, kernel k_τ, batch size B, and learning rate γ_G

repeat
  Sample $\{z_i\}_{i=1}^B \sim \mu$ and $\{\tilde{x}_j\}_{j=1}^B \sim p$
  $\hat{x}_i \leftarrow G_\theta(z_i)$ for all i
  Construct $\widehat{V}_{p,q_\theta}(\hat{x}_i)$ using Eq. (2)
  $x_i^{\mathrm{target}} \leftarrow \mathrm{sg}(\hat{x}_i + \widehat{V}_{p,q_\theta}(\hat{x}_i))$
  $\widehat{\mathcal{L}}^{\mathrm{Drift}}_{\varphi,\theta} \leftarrow B^{-1}\sum_{i=1}^B \|\hat{x}_i - x_i^{\mathrm{target}}\|_2^2$
  $\nabla_\theta \widehat{\mathcal{L}}^{\mathrm{Drift}}_{\varphi,\theta} = -2B^{-1}\sum_{i=1}^B J_i^\top \widehat{V}_{p,q_\theta}(\hat{x}_i)$
  $\theta \leftarrow \theta - \gamma_G \nabla_\theta \widehat{\mathcal{L}}^{\mathrm{Drift}}_{\varphi,\theta}$
until converged

Algorithm 3: Divergence-Based GAN and Matched KDE Update

Input: Generator G_θ, discriminator D, real data distribution p, latent distribution μ, objectives ϕ, ψ, h, kernel k_τ, batch size B, discriminator steps K_D, and learning rates γ_D, γ_G

repeat
  for $k = 1, \ldots, K_D$ **do**
    Sample $\{z_i\}_{i=1}^B \sim \mu$ and $\{\tilde{x}_j\}_{j=1}^B \sim p$
    $\hat{x}_i \leftarrow \mathrm{sg}(G_\theta(z_i))$ for all i
    $\widehat{\mathcal{V}}(D, G_\theta) \leftarrow \frac{1}{B}\sum_{j=1}^B \phi(D(\tilde{x}_j)) + \frac{1}{B}\sum_{i=1}^B \psi(D(\hat{x}_i))$
    $D \leftarrow D + \gamma_D \nabla_D \widehat{\mathcal{V}}(D, G_\theta)$
  end for
  Reuse $\{z_i\}$ and $\{\tilde{x}_j\}$ of Algorithm 2; $\hat{x}_i \leftarrow G_\theta(z_i)$
  Estimate $r_{D^*} = p/q_\theta$ by $\widehat{r} = \widehat{p}_\tau/\widehat{q}_\tau^\theta$ with fixed anchors
  $\widehat{\mathcal{L}}_G(D^*, G_\theta) \leftarrow -B^{-1}\sum_i h(\log \widehat{r}(\hat{x}_i)) = \widehat{\mathcal{L}}^{\mathrm{KDE}}_{\varphi,\theta}$
  $\nabla_\theta \widehat{\mathcal{L}}^{\mathrm{KDE}}_{\varphi,\theta} = -2B^{-1}\sum_i J_i^\top \widehat{V}_{p,q_\theta}(\hat{x}_i) = \nabla_\theta \widehat{\mathcal{L}}^{\mathrm{Drift}}_{\varphi,\theta}$
  $\theta \leftarrow \theta - \gamma_G \nabla_\theta \widehat{\mathcal{L}}_G(D^*, G_\theta)$
until converged

The matched KDE comparison uses the same real and generated batch, fixed anchors, $h(a) = \tau a$, and no velocity clipping; the gradient identity is derived in Appendix D.2. The connection to a GAN critic is discussed separately in Appendix D.3. This comparison isolates the current-step gradients, whereas Algorithm 1 uses fresh batches for the representation and generator phases.

F Limitations

Our study has two limitations. Although representations learned from scratch substantially improve standard drifting, their performance still trails that obtained with pretrained encoders, indicating that training dynamics alone do not yet recover all of the useful structure provided by external pretrain-

ing. In addition, owing to computational constraints, the effectiveness of persistent representation learning on larger-scale datasets remains to be established.