DRIFTXPRESS: FASTER DRIFTING MODELS VIA PROJECTED RKHS FIELDS

Anonymous authors
Paper under double-blind review

ABSTRACT

Drifting Models have emerged as a new paradigm for one-step generative modeling, achieving strong image quality without iterative inference. The premise is to replace the iterative denoising process in diffusion models with a single evaluation of a generator. However, this creates a different trade-off: drifting reduces inference cost by moving much of the computation into training. We introduce **DriftXpress**, an accelerated formulation of drifting models based on projected RKHS fields. DriftXpress approximates the drifting kernel in a low-rank feature space. This preserves the attraction-repulsion structure of the original drifting field while reducing the cost of field evaluation. Across image-generation benchmarks, DriftXpress achieves comparable FID to standard drifting while reducing wall-clock training cost. These results show that the training-inference trade-off of drifting models can be pushed further without giving up their one-step inference advantage.

1 INTRODUCTION

One-step generative models are appealing because they reduce sampling to a single forward pass. However, achieving this with high fidelity has proven difficult. Diffusion and flow-based generators (Sohl-Dickstein et al., 2015; Ho et al., 2020; Song et al., 2021; Lipman et al., 2023; Liu et al., 2023; Albergo et al., 2025) obtain strong sample quality by distributing generation across many iterative refinement steps, each individually simple. Collapsing these steps into one, whether by distillation (Salimans & Ho, 2022; Luo, 2023; Yin et al., 2024; Zhou et al., 2024) or by direct single-step training (Song et al., 2023; Frans et al., 2025; Geng et al., 2026; Boffi et al., 2024), typically requires either a pretrained multi-step teacher or approximations of the underlying ODE or SDE trajectories. Generative Adversarial Networks (GANs) (Goodfellow et al., 2014) are one-step, but rely on adversarial optimization and inherit its well-known instabilities (Mescheder et al., 2018; Arjovsky & Bottou, 2017; Salimans et al., 2016).

Drifting Models (Deng et al., 2026) provide an alternative to one-step generation by learning an attraction–repulsion field in feature space: generated samples are pulled toward the data distribution and pushed away from the current model distribution. Their premise is to replace diffusion-style iterative inference with a single generator evaluation. While drifting models are effective in practice, matching or surpassing strong one-step and multi-step baselines, they introduce a different trade-off: inference is cheap, but training is expensive because the model must repeatedly estimate a data-dependent vector field. This paper asks whether that trade-off can be pushed further:

> *"Can we reduce the training cost of drifting models while preserving their one-step inference advantage?"*

Addressing this challenge is important, as once inference is reduced to a single step, training becomes the primary computational bottleneck. We observe that in standard drifting, this bottleneck arises from estimating the field via kernel evaluations between generated and training samples. Since the field depends on these evaluations, we focus on reducing the cost of field estimation by replacing exact kernel interactions with projected kernel evaluations while preserving the field's form.

The reproducing kernel Hilbert space (RKHS) perspective provides the foundation for kernel methods. RKHSs are used because positive-definite kernels enable nonlinear estimators to be stud-

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.

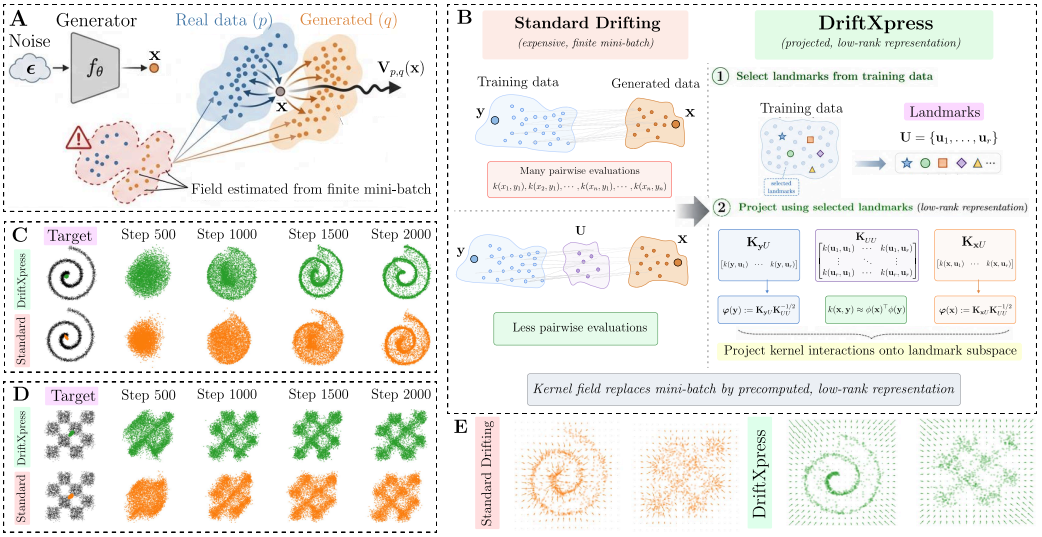

Figure 1: **Overview of DriftXpress.** (A) Drifting trains a generator by estimating an attraction–repulsion field from finite mini-batches. (B) DriftXpress replaces repeated exact attraction with a Nyström landmark projection. (C–D) Swissroll and checkerboard training trajectories show faster convergence to structured targets than standard drifting. (E) DriftXpress yields smoother vector fields that better follow the target geometry.

ied through linear geometry in function space, which underlies a broad class of methods, including support vector machines and kernel principal component analysis (Cortes & Vapnik, 1995; Schölkopf et al., 1998). The Nyström method is a standard tool for scaling such kernel-based algorithms Drineas & Mahoney (2005). Rather than working with the full kernel matrix, it approximates the kernel interaction structure using a subset of landmark columns, yielding a low-rank surrogate that preserves the kernel formulation and reducing computational cost (Williams & Seeger, 2001).

Motivated by this view, we introduce DriftXpress, a projected RKHS formulation of drifting models (Figure 1). Instead of recomputing exact attraction against the training support at every step, DriftXpress approximates these kernel interactions using a Nyström feature map built from landmark points selected before training. This turns repeated training-support interactions into fast query-landmark evaluations and cached summary multiplications, while preserving the attraction–repulsion form of the drifting field. Empirically, this yields two sources of acceleration: cheaper field evaluation per step and faster wall-clock convergence, since projected attraction uses a summary of the full positive support rather than only mini-batch interactions.

Our contributions are threefold. First, we recast drifting fields in the RKHS induced by the drifting kernel, enabling low-rank landmark-based evaluation. Second, we introduce DriftXpress, which replaces repeated exact attraction with cached summaries of training data and composes these summaries across shards for larger datasets. Third, we show that DriftXpress preserves comparable sample quality while substantially reducing training cost.

2 Background: Drifting Models

Drifting Models (Deng et al., 2026) provide a one-step perspective on generative modeling by constructing a transport rule from noise to the data distribution. We briefly review the Drifting Model.

Pushforward at training time. Let $G_\theta : \mathcal{Z} \to \mathcal{X}$ be the image generator, let $h : \mathcal{X} \to \mathbb{R}^D$ be a fixed feature encoder, and let $p_{\boldsymbol{e}}$ be the noise distribution on $\mathcal{Z}$. We write $\mathbf{x} = h(G_\theta(\boldsymbol{e})) \in \mathbb{R}^D$ for a generated feature and $\mathbf{y}^+ = h(\mathbf{a}^+)$ for a data feature, where $\mathbf{a}^+ \sim p_{\text{data}}$. As training iterates, the sequence of models $\{G_{\theta_i}\}$ induces a feature-space pushforward distribution $q_{\theta_i} = (h \circ G_{\theta_i})_{\#} p_{\boldsymbol{e}}$ and is intended to approach the pushforward data distribution $p = h_{\#} p_{\text{data}}$.

Drifting field. The evolution of samples at iteration i is governed by $\mathbf{V}_{p,q_i} : \mathbb{R}^D \to \mathbb{R}^D$:

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \mathbf{V}_{p,q_i}(\mathbf{x}_i). \tag{1}$$

For a generic model distribution q, the field satisfies the anti-symmetry property $\mathbf{V}_{p,q} = -\mathbf{V}_{q,p}$, which guarantees that $\mathbf{V}_{p,q}(\mathbf{x}) = 0$ for all $\mathbf{x}$ when $p = q$.

Attraction-repulsion instantiation. The drifting field decomposes into attraction toward data and repulsion from generated samples:

$$\mathbf{V}_{p,q}(\mathbf{x}) = \mathbf{V}_p^+(\mathbf{x}) - \mathbf{V}_q^-(\mathbf{x}), \tag{2}$$

where $\mathbf{y}^+ \sim p$ denotes a data sample and $\mathbf{y}^- \sim q$ denotes a generated sample. We have

$$\mathbf{V}_p^+(\mathbf{x}) = \frac{1}{Z_p(\mathbf{x})}\mathbf{E}_{\mathbf{y}^+\sim p}\big[k(\mathbf{x},\mathbf{y}^+)(\mathbf{y}^+ - \mathbf{x})\big], \qquad \mathbf{V}_q^-(\mathbf{x}) = \frac{1}{Z_q(\mathbf{x})}\mathbf{E}_{\mathbf{y}^-\sim q}\big[k(\mathbf{x},\mathbf{y}^-)(\mathbf{y}^- - \mathbf{x})\big], \tag{3}$$

with normalization factors

$$Z_p(\mathbf{x}) = \mathbf{E}_{\mathbf{y}^+\sim p}\big[k(\mathbf{x},\mathbf{y}^+)\big], \qquad Z_q(\mathbf{x}) = \mathbf{E}_{\mathbf{y}^-\sim q}\big[k(\mathbf{x},\mathbf{y}^-)\big], \tag{4}$$

where the default kernel k is the Laplace kernel.

Training objective. The training loss drives the encoded generator output toward the drifted feature target via a stop-gradient. With $\mathbf{x}_\theta(\boldsymbol{e}) = h(G_\theta(\boldsymbol{e}))$,

$$\mathcal{L} = \mathbf{E}_{\boldsymbol{e}\sim p_{\boldsymbol{e}}}\Big[\big\|\mathbf{x}_\theta(\boldsymbol{e}) - \texttt{stopgrad}\big(\mathbf{x}_\theta(\boldsymbol{e}) + \mathbf{V}_{p,q_\theta}(\mathbf{x}_\theta(\boldsymbol{e}))\big)\big\|^2\Big]. \tag{5}$$

The encoder parameters remain fixed, while gradients through h update G_θ. The expectations in $\mathbf{V}_{p,q}$ are estimated empirically during training. Repeated normalized kernel interactions between generated features and data features are the field-estimation cost targeted by DriftXpress.

3 DriftXpress: RKHS Projected Drifting Fields

We introduce DriftXpress, a projected RKHS formulation of drifting fields. The field is written as a difference of normalized kernel-weighted averages, and scalability comes from replacing exact kernel sections with projections onto a landmark-induced subspace. We implement this projection with the Nyström approximation, which expresses the kernel through evaluations against a small landmark set and yields the compact summaries used by DriftXpress.

3.1 RKHS Projection via Nyström Landmarks

Let $\mathcal{H}_k$ denote the RKHS associated with the kernel k, and let

$$\mathcal{H}_{\mathbf{U}} := \operatorname{span}\{k(\mathbf{u}_1,\cdot),\ldots,k(\mathbf{u}_r,\cdot)\} \subset \mathcal{H}_k$$

be the landmark-induced subspace associated with a set of landmarks $\mathbf{U} = \{\mathbf{u}_1,\ldots,\mathbf{u}_r\} \subset \mathbb{R}^D$. Landmarks are selected from the training samples. We sample landmarks uniformly at random within each class. The resulting landmarks are fixed before training and define the subspace $\mathcal{H}_{\mathbf{U}}$. We study alternative landmark-selection rules in Section 4.2.

DriftXpress approximates each kernel section $k(\mathbf{x},\cdot)$ by its projection onto $\mathcal{H}_{\mathbf{U}}$. In practice, this projection is implemented through the Nyström feature map. Define the landmark kernel matrix $\mathbf{K}_{UU} \in \mathbb{R}^{r\times r}$ with entries

$$[\mathbf{K}_{UU}]_{ab} = k(\mathbf{u}_a,\mathbf{u}_b).$$

For any point $\mathbf{z} \in \mathbb{R}^D$, define

$$\mathbf{K}_{\mathbf{z}U} = [k(\mathbf{z},\mathbf{u}_1),\ldots,k(\mathbf{z},\mathbf{u}_r)] \in \mathbb{R}^{1\times r}.$$

The Nyström feature map is[1]

$$\boldsymbol{\varphi}(\mathbf{z}) := \Big(\mathbf{K}_{\mathbf{z}U}\mathbf{K}_{UU}^{-1/2}\Big)^\top \in \mathbb{R}^r. \tag{6}$$

[1]For numerical stability, $\mathbf{K}_{UU}^{-1/2}$ is replaced by $(\mathbf{K}_{UU} + \lambda\mathbf{I})^{-1/2}$

For any two points $\mathbf{z}, \mathbf{z}' \in \mathbb{R}^D$, the Nyström features approximate their kernel similarity through the low-rank kernel based on landmarks

$$k_{\mathbf{U}}(\mathbf{z}, \mathbf{z}') := \boldsymbol{\varphi}(\mathbf{z})^\top \boldsymbol{\varphi}(\mathbf{z}') \approx k(\mathbf{z}, \mathbf{z}'). \tag{7}$$

Thus, instead of evaluating the full kernel k, DriftXpress works with the low-rank kernel $k_{\mathbf{U}}$ associated with the landmark subspace $\mathcal{H}_{\mathbf{U}}$.

3.2 RKHS Projected Approximation of the Drifting Field

For a fixed positive feature support $\{\mathbf{y}_j^+\}_{j=1}^{N^+}$, define the stabilized empirical attractive mean with offset $\varepsilon > 0$:

$$\mu_p(\mathbf{x}) := \frac{\sum_{j=1}^{N^+} k(\mathbf{x}, \mathbf{y}_j^+)\, \mathbf{y}_j^+}{\sum_{j=1}^{N^+} k(\mathbf{x}, \mathbf{y}_j^+) + \varepsilon}, \qquad \mathbf{V}_p^+(\mathbf{x}) = \mu_p(\mathbf{x}) - \mathbf{x}. \tag{8}$$

For $\varepsilon = 0$, this is the empirical version of the normalized attractive mean in Section 2. DriftXpress replaces k by $k_{\mathbf{U}}$ while retaining the same offset:

$$\mu_p^{\mathbf{U}}(\mathbf{x}) := \frac{\sum_{j=1}^{N^+} \boldsymbol{\varphi}(\mathbf{x})^\top \boldsymbol{\varphi}(\mathbf{y}_j^+)\, \mathbf{y}_j^+}{\sum_{j=1}^{N^+} \boldsymbol{\varphi}(\mathbf{x})^\top \boldsymbol{\varphi}(\mathbf{y}_j^+) + \varepsilon}, \qquad \mathbf{V}_{p,\mathbf{U}}^+(\mathbf{x}) = \mu_p^{\mathbf{U}}(\mathbf{x}) - \mathbf{x}. \tag{9}$$

Since $\boldsymbol{\varphi}(\mathbf{x})^\top$ does not depend on j, we can factor it out and precompute the following *summaries*:

$$\mathbf{A}_p := \sum_{j=1}^{N^+} \boldsymbol{\varphi}(\mathbf{y}_j^+)\, \mathbf{y}_j^{+\top} \in \mathbb{R}^{r \times D}, \qquad \mathbf{b}_p := \sum_{j=1}^{N^+} \boldsymbol{\varphi}(\mathbf{y}_j^+) \in \mathbb{R}^r. \tag{10}$$

This gives the compact expression

$$\mu_p^{\mathbf{U}}(\mathbf{x}) = \frac{\mathbf{A}_p^\top \boldsymbol{\varphi}(\mathbf{x})}{\boldsymbol{\varphi}(\mathbf{x})^\top \mathbf{b}_p + \varepsilon}. \tag{11}$$

For the repulsive component, DriftXpress keeps exact kernel interactions:

$$\mu_q(\mathbf{x}) := \frac{\sum_{j=1}^{N^-} k(\mathbf{x}, \mathbf{y}_j^-)\mathbf{y}_j^-}{\sum_{j=1}^{N^-} k(\mathbf{x}, \mathbf{y}_j^-) + \varepsilon}, \qquad \mathbf{V}_q^-(\mathbf{x}) = \mu_q(\mathbf{x}) - \mathbf{x}. \tag{12}$$

Here $\{\mathbf{y}_j^-\}_{j=1}^{N^-}$ denotes the generated samples from the current model distribution. In practice, this set is the generated batch at the current training step, with self-interactions removed when the query point is also part of the repulsive support. Therefore, the DriftXpress drifting field uses projected attraction and exact repulsion:

$$\mathbf{V}_{p,q}^{\mathbf{U}}(\mathbf{x}) := \mu_p^{\mathbf{U}}(\mathbf{x}) - \mu_q(\mathbf{x}) = \underbrace{\frac{\mathbf{A}_p^\top \boldsymbol{\varphi}(\mathbf{x})}{\boldsymbol{\varphi}(\mathbf{x})^\top \mathbf{b}_p + \varepsilon}}_{\mu_p^{\mathbf{U}}(\mathbf{x})} - \underbrace{\frac{\sum_{j=1}^{N^-} k(\mathbf{x}, \mathbf{y}_j^-)\mathbf{y}_j^-}{\sum_{j=1}^{N^-} k(\mathbf{x}, \mathbf{y}_j^-) + \varepsilon}}_{\mu_q(\mathbf{x})}. \tag{13}$$

As in the exact formulation, the $\mathbf{x}$ terms in $\mathbf{V}_{p,\mathbf{U}}^+(\mathbf{x}) = \mu_p^{\mathbf{U}}(\mathbf{x}) - \mathbf{x}$ and $\mathbf{V}_q^-(\mathbf{x}) = \mu_q(\mathbf{x}) - \mathbf{x}$ cancel exactly, so the implemented field is determined by a projected attractive field and an exact repulsive field. DriftXpress replaces the attraction term it with reusable summaries $\mathbf{A}_p$ and $\mathbf{b}_p$. These summaries can now also be computed on the entire dataset prior to training, and, as we will see in Section 4, they help DriftXpress achieve a holistic view of the data, resulting in a much faster convergence rate. The repulsive term is computed on the current generated batch, so it cannot be precomputed before training, unlike the attractive training-data summaries. Since approximating it would not remove the main offline-reusable bottleneck, we keep this term exact.

3.3 Compositional Summaries

A single landmark matrix can be too large at high class counts. We partition the positive support into S shards and construct a separate Nyström map and summary for each shard.

Let shard $s \in \{1, \dots, S\}$ have a landmark set $\mathbf{U}^{(s)}$ and feature map $\boldsymbol{\varphi}^{(s)}(\mathbf{x})$. We partition the positive training samples across shards. Let $\mathcal{I}_p^{(s)}$ denote the indices of positive samples assigned to shard s. The shard-local attractive summaries are

$$\mathbf{A}_p^{(s)} := \sum_{j \in \mathcal{I}_p^{(s)}} \boldsymbol{\varphi}^{(s)}(\mathbf{y}_j^+)\mathbf{y}_j^{+\top}, \qquad \mathbf{b}_p^{(s)} := \sum_{j \in \mathcal{I}_p^{(s)}} \boldsymbol{\varphi}^{(s)}(\mathbf{y}_j^+). \tag{14}$$

The projected attractive mean is obtained by summing shard contributions:

$$\mu_{p,\mathrm{comp}}(\mathbf{x}) = \frac{\sum_{s=1}^{S} \mathbf{A}_p^{(s)\top} \boldsymbol{\varphi}^{(s)}(\mathbf{x})}{\sum_{s=1}^{S} \boldsymbol{\varphi}^{(s)}(\mathbf{x})^\top \mathbf{b}_p^{(s)} + \varepsilon}. \tag{15}$$

Each shard contributes an unnormalized numerator and denominator, which can be accumulated sequentially. The resulting DriftXpress field is

$$\mathbf{V}_{p,q}^{\mathrm{comp}}(\mathbf{x}) := \mu_{p,\mathrm{comp}}(\mathbf{x}) - \mu_q(\mathbf{x}). \tag{16}$$

An advantage of this formulation is that both the numerator and denominator decompose additively across shards. Therefore, the field can be evaluated without materializing all summaries on the device at once: shard contributions can be accumulated sequentially, so memory scales with the largest shard rather than the full support.

Theorem 3.1 (Compositionality of attractive summaries)**.** *Let* $\bar{\boldsymbol{\varphi}}(\mathbf{x}) := \left[\boldsymbol{\varphi}^{(1)}(\mathbf{x}); \dots; \boldsymbol{\varphi}^{(S)}(\mathbf{x})\right]$ *denote the concatenation of the shard feature maps, and let*

$$\bar{\mathbf{A}}_p := \begin{bmatrix} \mathbf{A}_p^{(1)} \\ \vdots \\ \mathbf{A}_p^{(S)} \end{bmatrix}, \qquad \bar{\mathbf{b}}_p := \begin{bmatrix} \mathbf{b}_p^{(1)} \\ \vdots \\ \mathbf{b}_p^{(S)} \end{bmatrix},$$

denote the concatenated attractive summaries. Then the compositional attractive mean satisfies

$$\mu_{p,\mathrm{comp}}(\mathbf{x}) = \frac{\bar{\mathbf{A}}_p^\top \bar{\boldsymbol{\varphi}}(\mathbf{x})}{\bar{\boldsymbol{\varphi}}(\mathbf{x})^\top \bar{\mathbf{b}}_p + \varepsilon}.$$

Consequently, given the same repulsive mean $\mu_q(\mathbf{x})$*, the compositional field* $\mathbf{V}_{p,q}^{\mathrm{comp}}(\mathbf{x}) = \mu_{p,\mathrm{comp}}(\mathbf{x}) - \mu_q(\mathbf{x})$ *is identical to the field obtained using a single attractive summary.*

We next bound the error in the attractive mean on a fixed positive support. Proofs are in App. D. In the following statements, we set the offset value according to the feature support $\alpha := \varepsilon / N^+$.

Theorem 3.2 (Local distortion of the projected attractive field)**.** *Let* $\{\mathbf{y}_j^+\}_{j=1}^{N^+} \subset \mathbb{R}^D$ *satisfy* $\|\mathbf{y}_j^+\| \le R$*, and let* k *be a nonnegative kernel. Let* $k_{\mathbf{U}}$ *be the projected kernel induced by landmarks* $\mathbf{U}$*. For a query* $\mathbf{x}$*, define the kernel residual vector*

$$\mathbf{r}_{\mathbf{U}}(\mathbf{x}) := \left(k(\mathbf{x}, \mathbf{y}_1^+) - k_{\mathbf{U}}(\mathbf{x}, \mathbf{y}_1^+), \dots, k(\mathbf{x}, \mathbf{y}_{N^+}^+) - k_{\mathbf{U}}(\mathbf{x}, \mathbf{y}_{N^+}^+)\right)^\top \in \mathbb{R}^{N^+},$$

and the exact positive normalizer $d_p(\mathbf{x}) := \frac{1}{N^+} \sum_{j=1}^{N^+} k(\mathbf{x}, \mathbf{y}_j^+)$*. Assume* $d_p(\mathbf{x}) > 0$*. If* $\|\mathbf{r}_{\mathbf{U}}(\mathbf{x})\|_2 \le \frac{\sqrt{N^+}\left(d_p(\mathbf{x})+\alpha\right)}{2}$*, then*

$$\left\|\mathbf{V}_{p,\mathbf{U}}^+(\mathbf{x}) - \mathbf{V}_p^+(\mathbf{x})\right\| \le \frac{4R}{\sqrt{N^+}\left(d_p(\mathbf{x})+\alpha\right)} \|\mathbf{r}_{\mathbf{U}}(\mathbf{x})\|_2.$$

Corollary 3.3 (On-support distortion)**.** *Let* $\mathbf{K}, \mathbf{K}_{\mathbf{U}} \in \mathbb{R}^{N^+ \times N^+}$ *be the exact and projected Gram matrices on* $\{\mathbf{y}_j^+\}_{j=1}^{N^+}$*, with entries* $[\mathbf{K}]_{ij} = k(\mathbf{y}_i^+, \mathbf{y}_j^+)$ *and* $[\mathbf{K}_{\mathbf{U}}]_{ij} = k_{\mathbf{U}}(\mathbf{y}_i^+, \mathbf{y}_j^+)$*. Define* $\kappa_{\min} := \min_{1 \le i \le N^+} \frac{1}{N^+} \sum_{j=1}^{N^+} k(\mathbf{y}_i^+, \mathbf{y}_j^+)$*. If* $\|\mathbf{K} - \mathbf{K}_{\mathbf{U}}\|_{2,\infty} \le \frac{\sqrt{N^+}\left(\kappa_{\min}+\alpha\right)}{2}$*, then*

$$\frac{1}{N^+} \sum_{i=1}^{N^+} \left\|\mathbf{V}_{p,\mathbf{U}}^+(\mathbf{y}_i^+) - \mathbf{V}_p^+(\mathbf{y}_i^+)\right\|^2 \le \frac{16R^2}{{N^+}^2\left(\kappa_{\min}+\alpha\right)^2} \|\mathbf{K} - \mathbf{K}_{\mathbf{U}}\|_F^2.$$

Table 1: **Quality and training throughput.** Best FID is the lowest FID observed during training. Throughput is aggregate generated images per second across eight NVIDIA H100 GPUs (global batch size 12,000; 1,500 per GPU). Speedup is relative to standard drifting on the same dataset.

Dataset	Method	Landmarks / class	Best FID ↓	Throughput ↑	VRAM (GB) ↓	Speedup ↑
SVHN	Standard Drift	-	**2.94 ± 0.21**	2,307 img/s	**59.4**	
SVHN	DriftXpress (Non-Sharded)	500	3.11 ± 0.12	**15,402 img/s**	73.1	**6.68**×
SVHN	DriftXpress (Sharded)	500	3.15 ± 0.14	14,713 img/s	62.9	6.38×
CIFAR10	Standard Drift	-	5.64 ± 0.16	2,463 img/s	**71.5**	
CIFAR10	DriftXpress (Non-Sharded)	500	5.52 ± 0.06	**16,342 img/s**	74.6	**6.63**×
CIFAR10	DriftXpress (Sharded)	500	**5.14 ± 0.11**	14,985 img/s	72.3	6.08×
CIFAR100	Standard Drift	-	6.24 ± 0.11	2,703 img/s	**63.4**	
CIFAR100	DriftXpress (Non-Sharded)	70	9.65 ± 0.10	**10,143 img/s**	74.5	**3.75**×
CIFAR100	DriftXpress (Sharded)	200	**6.15 ± 0.09**	7,979 img/s	71.1	2.95×
ImageNet	Standard Drift	-	**8.83 ± 0.13**	2,058 img/s	**65.7**	
ImageNet	DriftXpress (Non-Sharded)	> 20	-	-	OOM	-
ImageNet	DriftXpress (Sharded)	150	9.21 ± 0.16	**5,453 img/s**	78.1	**2.64**×

Table 2: **Landmark selection ablation on CIFAR10.** Best FID over 10k training steps and average selection time for different landmark selection methods. Most density-aware methods perform competitively, while k-center performs substantially worse.

Method	Global			Per-class		
	Best FID ↓ (1280)	Best FID ↓ (5120)	Time (s) ↓	Best FID ↓ (128/class)	Best FID ↓ (512/class)	Time (s) ↓
Random	11.80 ± 0.71	11.34 ± 2.24	**0.02**	**11.82 ± 0.60**	**9.60 ± 0.95**	**0.06**
k-means	**11.26 ± 0.89**	**9.37 ± 0.37**	7.39	12.91 ± 0.25	10.52 ± 0.13	2.70
Facility location	12.26 ± 1.43	9.49 ± 0.84	32.70	11.83 ± 1.11	10.57 ± 0.67	1.21
Weighted k-center	–	–	–	11.94 ± 1.09	10.31 ± 1.40	17.91
k-center	28.65 ± 2.69	23.78 ± 2.40	170.52	23.87 ± 1.11	13.91 ± 1.01	17.79

Theorem 3.2 and Corollary 3.3 show that the error introduced by projecting the attractive field is controlled by two quantities: the landmark kernel residual $\|\mathbf{r}_{\mathbf{U}}(\mathbf{x})\|_2$ and the local kernel mass $d_p(\mathbf{x})$. In particular, the local distortion scales as $\|\mathbf{r}_{\mathbf{U}}(\mathbf{x})\|_2/\big(d_p(\mathbf{x})+\alpha\big)$. This also quantifies when $q = p$, DriftXpress evaluates to $\mu_p^{\mathbf{U}}(\mathbf{x}) - \mu_p(\mathbf{x})$ rather than zero, and the theorem bounds exactly this deviation. Thus, the projected field is accurate when the landmark kernel approximates the exact kernel well for queries sufficiently similar to the data. The corollary also explains why landmark selection matters. On the training support, the average squared field distortion is controlled by the Gram approximation error $\|\mathbf{K} - \mathbf{K}_{\mathbf{U}}\|_F^2$. Landmark selection methods that represent high-density regions, such as global k-means and facility location, should therefore yield better field estimates than methods that optimize worst-case geometric spread. This matches the results in Table 2.

4 Experiments

We evaluate DriftXpress along four axes. First, we compare its quality and efficiency against standard drifting across several image datasets. Second, we study wall-clock convergence to determine whether the projected field improves not only per-step speed, but also the rate at which FID decreases during training. Third, we isolate the effect of batch size on DriftXpress and the standard method. Finally, we ablate the landmark construction with respect to both the landmark selection strategy and the landmark ratio. The appendix provides additional experiments and implementation details. Specifically, we ablate exact versus projected repulsion (App. C.3), verify that DriftXpress maintains one-step inference efficiency (App. C.4), profile runtime and preprocessing gains (App. C.5), and evaluate the empirical and theoretical fidelity of the projected field (App. C.7).

4.1 Experimental setup

We compare DriftXpress against standard drifting by isolating the field estimator, keeping the generator architecture and objective identical. We evaluate sample quality using Best FID, image throughput, wall-clock runtime, peak GPU memory usage, and relative speedup.

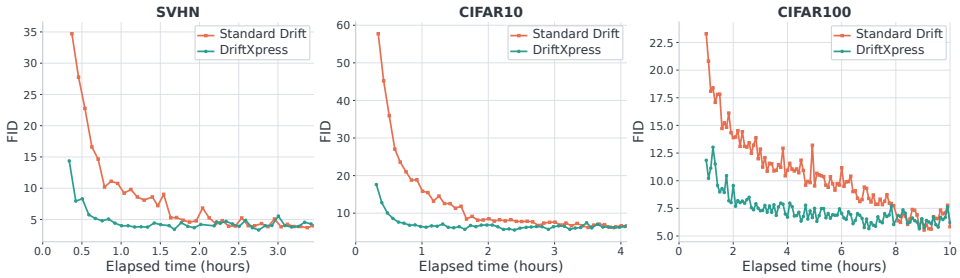

Figure 2: **FID over wall-clock training time.** FID trajectories for standard drifting and DriftXpress on SVHN, CIFAR10, and CIFAR100 and it reaches low-FID regimes faster in wall-clock time.

All experiments employ a 38.3M-parameter UNet generator (Ronneberger et al., 2015) and a frozen DINOv3 ViT-B/16 (Siméoni et al., 2025) feature encoder (required only during training). Results are reported as mean $\pm$ standard deviation across three seeds. The main benchmarks and ablations are run on 8 NVIDIA H100 GPUs and 1 NVIDIA H100 GPU, respectively. Rather than targeting computationally expensive state-of-the-art reproduction (Deng et al., 2026), our controlled comparisons specifically evaluate whether RKHS-projected field estimation improves efficiency and scalability under a fixed protocol.

4.2 Main Benchmark Results

We evaluate DriftXpress's training efficiency against standard drifting under a fixed 8-hour budget and global batch size of 12,000 (1500/GPU). Table 1 details sample quality and efficiency across diverse datasets: SVHN (Netzer et al., 2011), CIFAR10/100 (Krizhevsky et al., 2009), and ImageNet (Deng et al., 2009) (statistics in Table 3). We observe three key findings:

First, DriftXpress increases throughput while preserving sample quality comparable to that of standard drifting. On SVHN, DriftXpress accelerates throughput by $6.68\times$ (from 2,307 to 15,402 images per second) while maintaining a comparable best FID of 3.11 ± 0.12 versus 2.94 ± 0.21. Similarly, on CIFAR10, we observe a $6.63\times$ throughput speedup, along with a slight improvement in the best FID (5.52 ± 0.06 versus 5.64 ± 0.16). For datasets with larger class counts, the sharded variant of DriftXpress continues to deliver strong gains: yielding a $2.95\times$ speedup on CIFAR100 (FID 6.15 ± 0.09 vs. 6.24 ± 0.11) and a $2.64\times$ speedup on ImageNet (FID 9.21 ± 0.16 vs. 8.83 ± 0.13).

These measurements compare two attractive-field estimators under a shared generator and training protocol. Standard drifting computes pairwise kernel interactions against sampled positive mini-batches at every step, whereas DriftXpress evaluates Nyström attraction using cached summaries of the entire training set. Thus the methods differ both in kernel approximation and in positive-support sampling. We provide an online and offline cost breakdown in App. C.5.

Second, sharding is essential for ensuring that projected summaries remain memory-feasible at larger class counts. Because DriftXpress stores class-wise landmark summaries for the positive training support, unsharded memory costs scale greatly with both class count and landmark budget. We address this by defining one shard per class (10 shards for SVHN/CIFAR10, 100 for CIFAR100, and 1000 for ImageNet). During training, the numerator and denominator of the attractive field are accumulated shard-wise. This prevents materializing a full, unsharded positive summary, thereby reducing the size of the summary used in each projection step while keeping all shards on the GPU.

Third, we observe that the sharding approach introduces a trade-off between memory and throughput. Without sharding, CIFAR100 runs out of memory for landmark budgets exceeding 70 per class, and ImageNet exceeds the capacity of an 80GB GPU across all tested configurations. Sharding resolves these bottlenecks by evaluating class-wise summaries sequentially (see App. B.2). However, when memory permits both variants, the unsharded approach is faster as it evaluates a single projected summary. For instance, sharding on SVHN reduces peak VRAM from 73.1GB to 62.9GB, but drops throughput from 15,402 to 14,713 images/sec. We observe similar trends on CIFAR10 (74.6GB to 72.3GB; 16,342 $\rightarrow$ 14,985 images/sec) and CIFAR100 (74.5GB to 71.1GB; 10,143 $\rightarrow$ 7,979 images/sec). We conclude that while DriftXpress requires more overall mem-

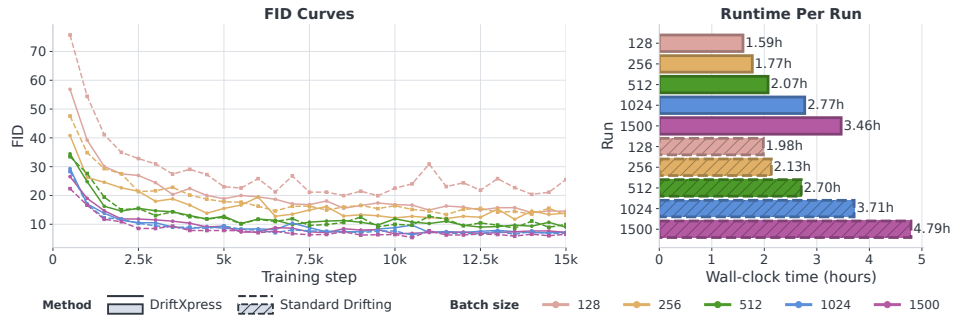

Figure 3: **CIFAR10 batch-size sweep.** (left): FID trajectories for DriftXpress and standard drifting across batch sizes 128–1500. (right): wall-clock runtime for each run. DriftXpress outperforms standard drifting across all batch sizes. For larger batch sizes, the primary benefit is runtime, whereas for small batches, the advantage is faster convergence and improved training trajectories.

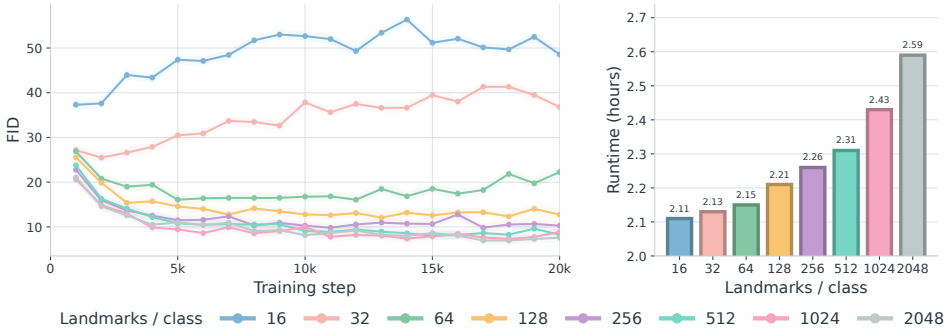

Figure 4: **Landmark ratio ablation.** (left) Training FID for different numbers of landmarks per class. (right) total training time. While higher landmark ratios improve FID, diminishing returns in sample quality against increasing runtimes suggest an optimal balance between 0.02 and 0.1.

ory than standard drifting to store landmarks and projected summaries, sharding provides a flexible mechanism to strictly bound this overhead.

Wall-Clock Convergence. Figure 2 plots FID against elapsed training time for SVHN, CIFAR10 (both unsharded), and CIFAR100 (sharded). Across all three datasets, DriftXpress (green) improves the early and mid-training FID trajectories relative to standard drifting (orange). This is because the projected kernel summary uses a larger effective support set than the mini-batches used by standard drifting, yielding a smoother and more stable field estimate. Consequently, DriftXpress reaches low-FID regimes faster in wall-clock time. The effect is most visible on SVHN and CIFAR10, while CIFAR100 has a more gradual trajectory due to its higher dataset complexity and use of the sharded variant. Overall, these trajectories support the main efficiency claim: the projected field not only reduces per-step cost, but also improves the rate at which drifting reaches competitive sample quality under a fixed compute budget. CIFAR10 samples at matched training checkpoints are shown in App. C.6.

Batch-Size Sweep. Figure 3 compares DriftXpress (solid lines) to standard drifting (dashed lines) across batch sizes from 128 to 1500. The left panel tracks CIFAR10 FID trajectories over 15,000 steps, while the right panel reports end-to-end wall-clock runtimes.

DriftXpress consistently outperforms standard drifting, with the specific advantage varying by regime. In large-batch regimes (typical of GPUs with larger memory), the primary benefit is computational efficiency. Because large batches stabilize field estimation, both methods yield comparable trajectories, but DriftXpress is much faster: runtime drops by 25.3% (0.94 hours) at batch size 1024 (in blue), and by 27.7% (1.33 hours) at 1500 (in purple). This efficiency gap accounts for the major speedups observed in Table 1. Conversely, in memory-constrained small-batch regimes, the primary advantage is faster convergence. For batch sizes of 128 (in red) and 256 (in orange), DriftXpress

yields strictly better FID trajectories alongside moderate runtime reductions of 19.7% and 17.0%. This superior convergence occurs because DriftXpress evaluates its attractive field against precomputed global summaries of the entire training set, maintaining a stable view of the landscape rather than relying on the limited, noisy positives of a small mini-batch.

Landmark Selection Strategies. Table 2 compares landmark selection strategies on CIFAR10, evaluated over 10k training steps with a batch size of 1024. We report the best FID scores for four primary methods: random sampling, k-means, k-center, and facility location (see App. C.2 for details). We evaluate these strategies across two variants: global (selected from the full training set) and per-class (selected equally within each class). These are tested at matched budgets of 1280 total landmarks (128 per class) and 5120 total landmarks (512 per class). Weighted k-center is evaluated per-class exclusively, as a global version would confound density weighting with class imbalance.

We observe that most methods perform similarly (FID between $9-12$) with the notable exception of k-center, which significantly underperforms (FID between $14-29$). The overall best FID is achieved by global k-means, scoring 11.26 ± 0.89 and 9.37 ± 0.37 at the low and high budgets, respectively, while maintaining moderate computational cost. Facility location is competitive under the higher 5120-budget (9.49 ± 0.84 globally) but is computationally expensive. Meanwhile, random selection is substantially cheaper yet remains highly competitive, particularly in the per-class setting, where it achieves an FID of 9.60 ± 0.95 with 512 landmarks per class. This aligns with prior Nyström approximation findings by (Kumar et al., 2009), who demonstrated that uniform sampling without replacement provides effective approximations with minimal time and memory overhead. Because random selection offers the best balance of computational speed and competitive sample quality, we adopt random per-class landmarks as the default for all main experiments.

Landmark Ratio. Figure 4 evaluates the effect of landmark ratio in DriftXpress on CIFAR10 using random per-class landmark selection. Since CIFAR10 has 5000 training samples per class, varying the number of landmarks per class from 16 to 2048 corresponds to landmark ratios from 0.0032 to 0.4096. The left panel reports FID trajectories over 20k training steps and the right panel reports the corresponding end-to-end wall-clock runtime. We observe a clear quality-runtime trade-off. At the lowest ratio (0.0032), training is fast (2.11 hours) but unstable, ending with a poor FID of 48.57. Conversely, the highest ratio (0.4096) yields a strong 7.57 FID but increases the runtime to 2.59 hours. This reflects the increased computational cost of query-landmark features and projected barycenters required for a larger, more accurate Nyström approximation. However, quality gains show diminishing returns. Increasing the ratio to the 0.0256–0.1024 range significantly improves FID, but further increases yield only marginal benefits given the computational overhead. Once the basis captures the distribution's dominant geometry, extra landmarks inflate the runtime. Consequently, the optimal trade-off lies between ratios of 0.0256 and 0.1024 (128–512 landmarks/class).

5 Conclusion

Drifting models replace iterative sampling with one-step generation, shifting computational burden to training. The main cost is field estimation: standard drifting repeatedly computes kernel interactions between generated samples and the support defining the attraction–repulsion field. DriftXpress addresses this by projecting attraction onto an RKHS subspace, thereby replacing repeated interactions with cached summaries. This provides a twofold speedup. First, wall-clock convergence improves because each projected step uses a full-dataset cached summary rather than mini-batch positive samples, yielding a smoother field that reaches low-FID regions faster. Second, it reduces the per-step field-estimation cost by replacing exact dataset attraction with query-landmark kernels and low-rank summary-multiplication. Across SVHN, CIFAR10, CIFAR100, and ImageNet, DriftXpress achieves comparable sample quality to standard drifting while reducing training time.

The method has several limitations. Projecting attraction while keeping repulsion exact improves stability, but breaks exact anti-symmetry and leaves a quadratic batch-size dependence in the repulsive term. Our theory bounds the resulting projected-attraction error through the landmark kernel residual and local kernel mass, but safely approximating repulsion remains open. DriftXpress also assumes a fixed feature encoder, since changing the encoder would invalidate the cached summaries. These limitations point to generated-sample-aware repulsion approximations and more memory-efficient projected summaries as next steps.

AI USE STATEMENT

We used generative AI tools to assist with coding, mathematical proofs, literature review, and drafting and editing the manuscript. All research ideas and contributions originated with the authors. The authors reviewed the AI-assisted material and take responsibility for the final paper, including its proofs, citations, and code.

REFERENCES

Michael Albergo, Nicholas M Boffi, and Eric Vanden-Eijnden. Stochastic interpolants: A unifying framework for flows and diffusions. *Journal of Machine Learning Research*, 26(209):1–80, 2025.

Martín Arjovsky and Léon Bottou. Towards principled methods for training generative adversarial networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

Nicholas M Boffi, Michael S Albergo, and Eric Vanden-Eijnden. Flow map matching with stochastic interpolants: A mathematical framework for consistency models. *arXiv preprint arXiv:2406.07507*, 2024.

Jiarui Cao, Zixuan Wei, and Yuxin Liu. Gradient flow drifting: Generative modeling via wasserstein gradient flows of kde-approximated divergences. *arXiv preprint arXiv:2603.10592*, 2026.

Antoine Chatalic, Marco Letizia, Nicolas Schreuder, and Lorenzo Rosasco. A scalable nystrom-based kernel two-sample test with permutations. *arXiv preprint arXiv:2502.13570*, 2025.

Yizong Cheng. Mean shift, mode seeking, and clustering. *IEEE Transactions on pattern analysis and machine intelligence*, 17(8):790–799, 1995.

Kacper Chwialkowski, Heiko Strathmann, and Arthur Gretton. A kernel test of goodness of fit. In *International conference on machine learning*, pp. 2606–2615. PMLR, 2016.

Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on pattern analysis and machine intelligence*, 24(5):603–619, 2002.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

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. Ieee, 2009.

Mingyang Deng, He Li, Tianhong Li, Yilun Du, and Kaiming He. Generative modeling via drifting. *arXiv preprint arXiv:2602.04770*, 2026.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

Petros Drineas and Michael W. Mahoney. On the nyström method for approximating a gram matrix for improved kernel-based learning. *Journal of Machine Learning Research*, 6:2153–2175, 2005.

Gintare Karolina Dziugaite, Daniel M Roy, and Zoubin Ghahramani. Training generative neural networks via maximum mean discrepancy optimization. *arXiv preprint arXiv:1505.03906*, 2015.

Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 12873–12883, 2021.

Kevin Frans, Danijar Hafner, Sergey Levine, and Pieter Abbeel. One step diffusion via shortcut models. In *International Conference on Learning Representations*, volume 2025, pp. 34668–34684, 2025.

Leonard T Franz, Sebastian Hoffmann, Tim Weiland, Bernhard Schölkopf, and Georg Martius. Drifting fields are not conservative. *arXiv preprint arXiv:2604.06333*, 2026.

Zhengyang Geng, Mingyang Deng, Xingjian Bai, Zico Kolter, and Kaiming He. Mean flows for one-step generative modeling. *Advances in Neural Information Processing Systems*, 38:75460–75482, 2026.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, volume 27, 2014.

Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *The journal of machine learning research*, 13(1):723–773, 2012.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, pp. 6840–6851, 2020.

Florian Kalinke, Zoltán Szabó, and Bharath Sriperumbudur. Nyström kernel stein discrepancy. In *International Conference on Artificial Intelligence and Statistics*, pp. 388–396. PMLR, 2025.

Arkadii Kazanskii, Tatiana Petrova, Konstantin Bagrianskii, Aleksandr Puzikov, and Radu State. Attraction, repulsion, and friction: Introducing dmf, a friction-augmented drifting model. *arXiv preprint arXiv:2604.18194*, 2026.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

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

Sanjiv Kumar, Mehryar Mohri, and Ameet Talwalkar. Sampling techniques for the nystrom method. In *Artificial intelligence and statistics*, pp. 304–311. PMLR, 2009.

HakGeun Lee and Hyonho Chun. Identifiability and stability of generative drifting with companion-elliptic kernel families. *arXiv preprint arXiv:2604.24196*, 2026.

Yujia Li, Kevin Swersky, and Rich Zemel. Generative moment matching networks. In *International conference on machine learning*, pp. 1718–1727. PMLR, 2015.

Yaron Lipman, Ricky T Q Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023.

Qiang Liu, Jason Lee, and Michael Jordan. A kernelized stein discrepancy for goodness-of-fit tests. In *International conference on machine learning*, pp. 276–284. PMLR, 2016.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *International Conference on Learning Representations*, 2023.

Weijian Luo. A comprehensive survey on knowledge distillation of diffusion models. *arXiv preprint arXiv:2304.04262*, 2023.

Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for gans do actually converge? In *International conference on machine learning*, pp. 3481–3490. PMLR, 2018.

Krikamol Muandet, Kenji Fukumizu, Bharath Sriperumbudur, and Bernhard Schölkopf. Kernel mean embedding of distributions: A review and beyond. *arXiv preprint arXiv:1605.09522*, 2016.

Elizbar A Nadaraya. On estimating regression. *Theory of Probability & Its Applications*, 9(1): 141–142, 1964.

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, 2011.

Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International conference on machine learning*, pp. 1530–1538. PMLR, 2015.

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, 2022.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241. Springer, 2015.

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. *International Conference on Learning Representations*, 2022.

Tim Salimans, Ian J. Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pp. 2226–2234, 2016.

Bernhard Schölkopf, Alexander J. Smola, and Klaus-Robert Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5):1299–1319, 1998.

Oriane Siméoni, Huy V Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, et al. Dinov3. *arXiv preprint arXiv:2508.10104*, 2025.

Alex Smola, Arthur Gretton, Le Song, and Bernhard Scholkopf. A hilbert space embedding for distributions. *Algorithmic Learning Theory*, pp. 13–31, 2007.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pp. 2256–2265, 2015.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *International Conference on Machine Learning*, pp. 32211–32252. PMLR, 2023.

Bharath K Sriperumbudur, Arthur Gretton, Kenji Fukumizu, Bernhard Schölkopf, and Gert RG Lanckriet. Hilbert space embeddings and metrics on probability measures. *The Journal of Machine Learning Research*, 11:1517–1561, 2010.

Erkan Turan, Nicolas Dufour, and Maks Ovsjanikov. Generative drifting is secretly score matching: a spectral and variational perspective. *arXiv preprint arXiv:2603.09936*, 2026.

Geoffrey S Watson. Smooth regression analysis. *Sankhyā: The Indian Journal of Statistics, Series A*, pp. 359–372, 1964.

Christopher K. I. Williams and Matthias Seeger. Using the nyström method to speed up kernel machines. In Todd K. Leen, Thomas G. Dietterich, and Volker Tresp (eds.), *Advances in Neural Information Processing Systems 13*, pp. 682–688. MIT Press, 2001.

Tianwei Yin, Michaël Gharbi, Richard Zhang, Eli Shechtman, Frédo Durand, William T Freeman, and Taesung Park. One-step diffusion with distribution matching distillation. In *Computer Vision and Pattern Recognition*, 2024.

Linqi Zhou, Stefano Ermon, and Jiaming Song. Inductive moment matching. In *International Conference on Machine Learning*, pp. 78651–78686. PMLR, 2025.

Mingyuan Zhou, Huangjie Zheng, Zhendong Wang, Mingzhang Yin, and Hai Huang. Score identity distillation: Exponentially fast distillation of pretrained diffusion models for one-step generation. In *Forty-first International Conference on Machine Learning*, 2024.

APPENDIX CONTENTS

A Related Work **13**

B Algorithm **15**

B.1 Complexity Analysis 15

B.2 Memory and Sharding 17

C Additional Experiments **18**

C.1 Experimental Setup 18

C.2 Landmark Selection Algorithms 20

C.3 Exact and Projected Repulsion 21

C.4 Inference-time comparison 21

C.5 DriftXpress Cost Profile 22

C.6 Early Training Snapshots 25

C.7 Field Approximation Fidelity 25

D Proofs **27**

A RELATED WORK

Diffusion and flow-based generative models. Diffusion models and score-based generative models learn to transform noise into data by reversing a gradual noising process or by solving a reverse-time stochastic differential equation (Sohl-Dickstein et al., 2015; Ho et al., 2020; Song et al., 2021). Flow-based generative models and stochastic interpolants instead learn deterministic or stochastic transport dynamics between simple and data distributions, often through ordinary differential equations or velocity fields (Lipman et al., 2023; Liu et al., 2023; Albergo et al., 2025). These methods have achieved strong sample quality, but their generation procedure requires many function evaluations. At inference time, generation is therefore tied to an iterative solver that repeatedly updates a sample along a learned trajectory.

One-step and few-step generation. A large body of work has aimed to reduce the sampling cost of diffusion and flow-based models. Distillation-based methods compress a pretrained multi-step teacher into a generator that uses fewer sampling steps (Salimans & Ho, 2022; Yin et al., 2024; Zhou et al., 2024). Consistency models train mappings that are consistent along the probability-flow trajectory and can generate in one or a few steps, either by distillation or standalone training (Song et al., 2023). More recent approaches train one-step or few-step models from scratch by modifying the underlying flow or diffusion objective, for example, through shortcut models, mean flows, flow map matching, or moment-matching self-distillation (Frans et al., 2025; Geng et al., 2026; Boffi et al., 2024; Zhou et al., 2025). These methods remain closely tied to diffusion or flow trajectories: they either distill a pretrained trajectory, enforce consistency along one, or learn a modified velocity/flow representation.

Generative adversarial networks. Generative adversarial networks (GANs) train a generator and discriminator through a minimax objective (Goodfellow et al., 2014). Like GANs, Drifting Models use a generator that maps noise to samples in a single forward pass. However, the training signal is fundamentally different. GANs learn through adversarial discrimination, which can introduce optimization instability, mode collapse, and sensitivity to training dynamics (Salimans et al., 2016; Arjovsky & Bottou, 2017; Mescheder et al., 2018). Drifting Models avoid an adversarial discriminator and instead define an attraction–repulsion field whose equilibrium is reached when the generated and data distributions match (Deng et al., 2026).

Variational autoencoders and normalizing flows. Variational autoencoders (VAEs) optimize an evidence lower bound with an encoder-decoder architecture and a latent prior (Kingma & Welling, 2013). They provide efficient ancestral sampling from the prior, but their objective and inductive biases differ from drift-based training. Modern generative systems also often use autoencoding models as latent tokenizers or compression modules for diffusion or autoregressive models rather than as standalone high-fidelity generators (Esser et al., 2021; Rombach et al., 2022). Normalizing flows learn invertible transformations and optimize exact likelihood through change-of-variables formulas (Rezende & Mohamed, 2015; Dinh et al., 2016). They admit exact sampling and likelihood evaluation, but require architectural invertibility and tractable Jacobian determinants.

Moment matching and kernel generative models. Kernel moment-matching methods compare distributions through discrepancies between kernel mean embeddings. Maximum Mean Discrepancy (MMD) measures the largest difference in expectations over the unit ball of an RKHS (Gretton et al., 2012), and generative moment matching networks use MMD as a training objective for feed-forward generators (Dziugaite et al., 2015; Li et al., 2015). More recent work has revisited moment matching for fast generative modeling, including one-step and few-step models (Zhou et al., 2025). DriftXpress is related in its use of kernels and positive/negative samples, but the role of the kernel is different. Moment-matching methods usually define a scalar discrepancy between distributions. Drifting Models instead define a local vector field that moves each generated sample through attraction toward positive samples and repulsion from negative samples.

Kernel smoothing and mean-shift structure. The normalized attraction term in Drifting Models is closely related to classical kernel smoothing. In particular, the map

$$\boldsymbol{\mu}_p(\mathbf{x}) = \frac{\sum_j k(\mathbf{x}, \mathbf{y}_j^+)\mathbf{y}_j^+}{\sum_j k(\mathbf{x}, \mathbf{y}_j^+)}$$

has the form of a Nadaraya–Watson kernel regression estimator, which predicts a locally weighted average using normalized kernel weights (Nadaraya, 1964; Watson, 1964). In the drifting setting, the response values are the data features themselves, so this estimator defines a local barycenter that pulls a generated sample toward nearby data. This also connects drifting to mean shift, where kernel-weighted local means induce updates that move points toward modes of a kernel density estimate (Cheng, 1995; Comaniciu & Meer, 2002). The distinction is that Drifting Models do not perform pure density-mode seeking: they combine attraction toward the data distribution with repulsion from the generated distribution, yielding a signed attraction–repulsion field.

RKHS embeddings, Nyström approximation, and scalable kernel discrepancies. RKHS methods provide a classical way to study nonlinear learning problems through linear geometry in function space. This perspective underlies support vector machines, kernel PCA, kernel mean embeddings, MMD, and related nonparametric estimators (Cortes & Vapnik, 1995; Schölkopf et al., 1998; Smola et al., 2007; Gretton et al., 2012; Muandet et al., 2016). Kernel mean embeddings represent probability distributions as RKHS elements and support two-sample testing, independence testing, conditional mean embeddings, and distributional learning (Smola et al., 2007; Sriperumbudur et al., 2010; Muandet et al., 2016). Since exact kernel methods often require large pairwise kernel matrices, a long line of work studies scalable approximations. The Nyström method approximates a kernel matrix or operator using a subset of landmark columns (Williams & Seeger, 2001; Drineas & Mahoney, 2005), and related ideas have been used to accelerate kernel discrepancy methods such as MMD and Kernel Stein Discrepancy (Gretton et al., 2012; Chatalic et al., 2025; Liu et al., 2016; Chwialkowski et al., 2016; Kalinke et al., 2025). DriftXpress is closest in spirit to these scalable kernel methods, but differs in its object of approximation: rather than accelerating a kernel classifier, eigendecomposition, or scalar discrepancy, it projects drifting kernel sections onto a landmark-induced RKHS subspace. This yields finite-dimensional features that allow the attractive field to be written using precomputed summaries over the training set, preserving the normalized kernel-field form of Drifting Models while reducing repeated exact interactions with all positive samples.

Recent work on Drifting Models. Drifting Models (Deng et al., 2026) have already prompted several follow-up studies that analyze, reinterpret, or extend the drifting field. A first line of work studies the mathematical structure of the field itself. (Turan et al., 2026) shows that, under a Gaussian kernel, the drift operator can be written as a score difference between smoothed distributions,

connecting drifting to score matching, spectral convergence, and Wasserstein gradient-flow interpretations. Similarly, (Cao et al., 2026) proves an equivalence between drifting and the Wasserstein gradient flow of a KDE-approximated forward KL divergence, and further relates MMD-based generators to the same broader family of KDE-approximated divergence flows. (Franz et al., 2026) study whether drifting corresponds to optimizing a scalar potential, showing that drift fields are generally non-conservative due to the position-dependent normalization, while Gaussian kernels form a special conservative case. Complementary theoretical work studies identifiability and stability: (Lee & Chun, 2026) analyzes companion-elliptic kernel families, including the Laplace kernel, and (Kazanskii et al., 2026) introduces a friction-augmented variant that addresses local instability in simplified drifting dynamics.

B ALGORITHM

We provide the algorithms for DriftXpress preprocessing and training in Algorithms 1 and 2 respectively.

Algorithm 1 DriftXpress: Preprocessing

Require: Training data $\{\mathbf{y}_j^+\}_{j=1}^{N^+}$, landmark budget r, kernel k, regularizer λ

Ensure: Landmarks $\mathbf{U}$, Nyström transform $\mathbf{W} \in \mathbb{R}^{r\times r}$, attractive summaries $\mathbf{A}_p \in \mathbb{R}^{r\times D}$, $\mathbf{b}_p \in \mathbb{R}^r$

// Landmark selection
Sample landmarks $\mathbf{U} = \{\mathbf{u}_1, \dots, \mathbf{u}_r\}$ from $\{\mathbf{y}_j^+\}_{j=1}^{N^+}$ {e.g., random per-class sampling}
// Nyström transform
Compute landmark Gram matrix $[\mathbf{K}_{UU}]_{ab} \leftarrow k(\mathbf{u}_a, \mathbf{u}_b)$ for $a, b \in [r]$
Compute $\mathbf{W} \leftarrow (\mathbf{K}_{UU} + \lambda \mathbf{I})^{-1/2}$ {Stored once; used throughout training}
// Precompute attractive summaries over the full training set
Initialize $\mathbf{A}_p \leftarrow \mathbf{0}^{r\times D}$, $\mathbf{b}_p \leftarrow \mathbf{0}^r$
for $j = 1, \dots, N^+$ **do**
  $\mathbf{K}_{\mathbf{y}_j^+ U} \leftarrow [k(\mathbf{y}_j^+, \mathbf{u}_1), \dots, k(\mathbf{y}_j^+, \mathbf{u}_r)] \in \mathbb{R}^{1\times r}$
  $\boldsymbol{\varphi}(\mathbf{y}_j^+) \leftarrow \mathbf{W}^\top \mathbf{K}_{\mathbf{y}_j^+ U}^\top \in \mathbb{R}^r$
  $\mathbf{A}_p \leftarrow \mathbf{A}_p + \boldsymbol{\varphi}(\mathbf{y}_j^+)\, \mathbf{y}_j^{+\top}$
  $\mathbf{b}_p \leftarrow \mathbf{b}_p + \boldsymbol{\varphi}(\mathbf{y}_j^+)$
end for

return $\mathbf{U},\ \mathbf{W},\ \mathbf{A}_p,\ \mathbf{b}_p$

B.1 COMPLEXITY ANALYSIS

We analyze the cost of estimating the drifting field. Let B be the number of generated query samples per step, N^+ the number of training samples used for attraction, N^- the number of generated samples used for repulsion, D the feature dimension, and r the number of landmarks.

Standard Drift. Standard drifting computes both attraction and repulsion through exact kernel evaluations. For each query $\mathbf{x}_b$, the attractive component requires the weights

$$k(\mathbf{x}_b, \mathbf{y}_j^+), \qquad j = 1, \dots, N^+,$$

and the repulsive component requires the weights

$$k(\mathbf{x}_b, \mathbf{y}_\ell^-), \qquad \ell = 1, \dots, N^-.$$

Thus, for B query samples, standard drifting must form BN^+ attraction kernels and BN^- repulsion kernels at every training step. Since each kernel evaluation requires computing a distance in D-dimensional feature space, the per-step field-estimation cost is

$$\mathcal{O}\big(BN^+D + BN^-D\big),$$

with memory

$$\mathcal{O}\big(BN^+ + BN^-\big)$$

Algorithm 2 DriftXpress: Training

Require: Generator G_θ, fixed feature encoder h, noise distribution $p_{\boldsymbol{e}}$, precomputed $\mathbf{U}, \mathbf{W}, \mathbf{A}_p, \mathbf{b}_p$, kernel k, positive normalization offset $\varepsilon > 0$, learning rate η, training steps T
Ensure: Trained generator G_θ

for $i = 1, \dots, T$ **do**
  // Sample generated batch
  Sample $\{\boldsymbol{e}_b\}_{b=1}^{B} \sim p_{\boldsymbol{e}}$
  $\mathbf{a}_b \leftarrow G_\theta(\boldsymbol{e}_b)$, $\mathbf{x}_b \leftarrow h(\mathbf{a}_b)$ for $b \in [B]$ {Generated features}
  // Compute projected attraction from cached training-data summaries
  for $b = 1, \dots, B$ **do**
    $\mathbf{K}_{\mathbf{x}_b U} \leftarrow [k(\mathbf{x}_b, \mathbf{u}_1), \dots, k(\mathbf{x}_b, \mathbf{u}_r)] \in \mathbb{R}^{1\times r}$
    $\boldsymbol{\varphi}_b \leftarrow \mathbf{W}^\top \mathbf{K}_{\mathbf{x}_b U}^\top \in \mathbb{R}^r$
    $\mu_p^U(\mathbf{x}_b) \leftarrow \dfrac{\mathbf{A}_p^\top \boldsymbol{\varphi}_b}{\boldsymbol{\varphi}_b^\top \mathbf{b}_p + \varepsilon}$ {Projected attractive mean in $\mathbb{R}^D$}
  end for
  // Compute exact repulsion over the current generated batch
  for $b = 1, \dots, B$ **do**
    $K^-_{b\ell} \leftarrow k(\mathbf{x}_b, \mathbf{x}_\ell)$ for $\ell \in [B]$
    $K^-_{bb} \leftarrow 0$ {Remove self-interaction}
    $\mu_q(\mathbf{x}_b) \leftarrow \dfrac{\sum_{\ell=1}^{B} K^-_{b\ell}\mathbf{x}_\ell}{\sum_{\ell=1}^{B} K^-_{b\ell} + \varepsilon}$ {Exact repulsive mean}
  end for
  // Assemble the DriftXpress field
  for $b = 1, \dots, B$ **do**
    $\mathbf{V}^{\mathbf{U}}_{p,q}(\mathbf{x}_b) \leftarrow \mu_p^U(\mathbf{x}_b) - \mu_q(\mathbf{x}_b)$
  end for
  // Compute loss and update
  $\mathcal{L} \leftarrow \dfrac{1}{B}\sum_{b=1}^{B} \left\|\mathbf{x}_b - \mathrm{stopgrad}\big(\mathbf{x}_b + \mathbf{V}^{\mathbf{U}}_{p,q}(\mathbf{x}_b)\big)\right\|_2^2$
  $\theta \leftarrow \theta - \eta\nabla_\theta\mathcal{L}$
end for

return G_θ

DriftXpress. DriftXpress replaces the exact attractive-side kernel evaluations with a Nyström approximation. Given landmarks

$$\mathbf{U} = \{\mathbf{u}_1, \dots, \mathbf{u}_r\} \subset \mathbb{R}^D,$$

we form

$$[\mathbf{K}_{UU}]_{ab} = k(\mathbf{u}_a, \mathbf{u}_b), \qquad \mathbf{W} = (\mathbf{K}_{UU} + \lambda\mathbf{I})^{-1/2}.$$

This preprocessing costs

$$\mathcal{O}(r^2 D + r^3),$$

where $r^2 D$ forms the landmark Gram matrix and r^3 computes the inverse square root. For any point $\mathbf{z}$, the Nyström feature is

$$\boldsymbol{\varphi}(\mathbf{z}) = (\mathbf{K}_{\mathbf{z}U}\mathbf{W})^\top \in \mathbb{R}^r, \qquad \mathbf{K}_{\mathbf{z}U} = [k(\mathbf{z}, \mathbf{u}_1), \dots, k(\mathbf{z}, \mathbf{u}_r)].$$

Computing one feature costs $\mathcal{O}(rD + r^2)$.

The attractive summaries are precomputed once:

$$\mathbf{A}_p = \sum_{j=1}^{N^+} \boldsymbol{\varphi}(\mathbf{y}_j^+)\mathbf{y}_j^{+\top} \in \mathbb{R}^{r\times D}, \qquad \mathbf{b}_p = \sum_{j=1}^{N^+} \boldsymbol{\varphi}(\mathbf{y}_j^+) \in \mathbb{R}^r.$$

The one-time attractive-side preprocessing cost is

$$\mathcal{O}\big(r^2 D + r^3 + N^+(rD + r^2)\big).$$

At training time, for B generated queries, DriftXpress computes their projected features and evaluates

$$\boldsymbol{\mu}_p^{\mathbf{U}}(\mathbf{x}) = \frac{\mathbf{A}_p^\top \boldsymbol{\varphi}(\mathbf{x})}{\boldsymbol{\varphi}(\mathbf{x})^\top \mathbf{b}_p + \varepsilon}.$$

This costs

$$\mathcal{O}\big(B(rD + r^2)\big)$$

for the projected attractive component.

The repulsive component is unchanged and is computed exactly:

$$\boldsymbol{\mu}_q(\mathbf{x}) = \frac{\sum_{\ell=1}^{N^-} k(\mathbf{x}, \mathbf{y}_\ell^-)\mathbf{y}_\ell^-}{\sum_{\ell=1}^{N^-} k(\mathbf{x}, \mathbf{y}_\ell^-) + \varepsilon}.$$

Thus exact repulsion costs

$$\mathcal{O}(BN^-D)$$

per step. Overall, DriftXpress has per-step field-estimation cost

$$\mathcal{O}\big(B(rD + r^2) + BN^-D\big)\,,$$

compared with

$$\mathcal{O}\big(BN^+D + BN^-D\big)$$

for standard drifting. These bounds describe the explicit-feature evaluation in Algorithms 1–2. In the reported comparison, standard drifting draws positive minibatches at each step, whereas DriftXpress caches the full positive training support. Thus N^+ in the standard-drifting bound denotes its per-step positive support, while the preprocessing bound uses the full training-set size. The gain depends on the sizes of those supports, the landmark budget, and the fraction of iteration time spent computing attraction.

B.2 Memory and Sharding

Treating the number of feature groups and temperatures as constants, standard drifting materializes exact attraction and repulsion weights of sizes $B \times N^+$ and $B \times N^-$. Its dominant batchwise memory is therefore

$$\mathcal{O}(BN^+ + BN^-).$$

DriftXpress replaces the exact attractive kernel with a Nyström cache. With r landmarks, the attractive summaries store

$$\mathbf{A}_p \in \mathbb{R}^{r\times D}, \qquad \mathbf{W} \in \mathbb{R}^{r\times r},$$

and each step materializes query features of size $B \times r$. Thus the dominant memory is

$$\mathcal{O}(rD + r^2 + Br + BN^-),$$

or more precisely $\mathcal{O}(T(rD + r^2) + Br + BN^-)$ when the number of temperatures T is not treated as constant.

For large landmark banks, we split the attractive cache into S shards with r_s landmarks per shard, where $r = \sum_{s=1}^{S} r_s$. Sharding is possible because the attractive numerator and denominator decompose additively across shards. The total resident attractive cache becomes

$$\mathcal{O}\left(\sum_{s=1}^{S}(r_sD + r_s^2)\right) = \mathcal{O}\left(rD + \sum_{s=1}^{S} r_s^2\right).$$

Under sequential shard evaluation, the active query-feature workspace is reduced from $\mathcal{O}(Br)$ to

$$\mathcal{O}\left(\max_{s\in[S]} Br_s\right).$$

Therefore, when all shards remain resident on GPU, the peak memory is

$$\mathcal{O}\left(rD + \sum_{s=1}^{S} r_s^2 + \max_{s\in[S]} Br_s + BN^-\right),$$

up to common output and accumulator buffers. If shards are streamed one at a time from CPU or off-rank storage, the peak memory can instead be reduced to

$$\mathcal{O}\left(\max_{s\in[S]}(r_sD + r_s^2 + Br_s) + BN^-\right).$$

Scaling limits. With C class shards and m landmarks per class, the total landmark count is $r = Cm$. An unsharded Nyström construction stores an $r \times r$ landmark kernel matrix and its factor; class-wise sharding instead stores C matrices of size $m \times m$. This reduces their combined storage from $\mathcal{O}(C^2m^2)$ to $\mathcal{O}(Cm^2)$. In the reported implementation, all shards remain GPU-resident, so the attractive summaries still contain $\mathcal{O}(CmD)$ entries. Sequential evaluation reduces the active query-feature workspace from $\mathcal{O}(BCm)$ to $\mathcal{O}(Bm)$, but queries are still evaluated against all Cm landmarks and contributions must be accumulated across shards. These costs explain the memory–throughput trade-off of sharding. Sharding also results in a certain sharding speed overhead. Splitting work into shards adds accumulation costs that increase with the class count C.

Exact repulsion costs $\mathcal{O}(B^2D)$ when the generated batch serves as both queries and negative support. Its share of the iteration cost can therefore increase with the local batch size. However, the single-GPU batch-size sweep ends at $B = 1500$, near the memory limit of the evaluated GPU configuration; our measurements do not reach a larger local-batch regime in which to locate the point where repulsion dominates. GPUs with more memory could support larger local batches, where the quadratic repulsion cost may reduce the speedup from approximating attraction.

Scaling strategies. As discussed above, the main bottleneck is the growth of the total landmarks with the number of classes. Although class-wise sharding removes the quadratic dependence across classes, the linear term in the total landmark count remains and dominates, which is why the speedup decreases from smaller datasets like CIFAR10 to larger datasets like ImageNet. This degradation of throughout is expected and will continue as the class count increases.

A couple of things that we were not able to experiment with but can be done for further scaling include: (i) using a global bank with class-conditional summaries, so that r decouples from C and the dominant rD term stops growing with the label space. Table 2 supports this direction, as global k-means outperforms per-class selection at matched landmark budgets; (ii) distributing shards across ranks, which Theorem 3.1 licenses, giving per-device memory set by the largest shard rather than the full landmark bank; and (iii) increasing the number of devices (scaling vertically), allowing larger landmark budgets while having a fixed per-device workload.

C Additional Experiments

C.1 Experimental Setup

We use the same generator, feature encoder, optimizer, and evaluation protocol for all reported runs unless otherwise stated. The goal of the experimental protocol is to isolate the effect of the field estimator: standard drifting and DriftXpress use the same generator architecture, the same drifting objective, the same frozen feature encoder, and the same optimization hyperparameters. The only difference is that standard drifting estimates attraction with exact kernel interactions, whereas DriftXpress replaces the attractive component with a Nyström-projected RKHS field and cached positive summaries.

Datasets. We evaluate on SVHN, CIFAR10, CIFAR100, and ImageNet. CIFAR10 and CIFAR100 contain $50{,}000$ training images, with 10 and 100 classes, respectively. SVHN contains $73{,}257$ training images and 10 imbalanced classes. ImageNet contains $1{,}281{,}167$ training images and $1{,}000$ imbalanced classes. All datasets are used with their standard training splits for training and reference statistics.

Generator architecture. Every reported run uses the same U-Net generator. The generator has base channel width 128, channel multipliers $(1, 2, 2, 2)$, two residual blocks per resolution, self-attention at spatial resolution 16×16, dropout 0.1, and four attention heads. For RGB datasets, this architecture has 38.3M trainable parameters. Since DriftXpress changes only the training-time field estimator, the generator architecture and one-step inference procedure are identical to standard drifting.

Feature encoder. The drifting field is computed in the representation space of a frozen DINOv3 feature encoder. We use input resolution 112×112 and pooled spatial size 4. The encoder has 85.6M frozen parameters. The encoder is used only during training to define the feature-space

drifting objective. It is not used at inference time, where generation requires only a single forward pass through the trained U-Net generator.

Optimization. All methods are trained with AdamW using learning rate 2×10^{-4}, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and zero weight decay. We use EMA with decay 0.9999, gradient clipping at norm 2.0, fused optimizer updates, and bfloat16 autocast. Both the generator and the frozen feature encoder are compiled in the reported runs.

Drifting field configuration. Standard drifting computes both attraction and repulsion using exact kernel-weighted barycenters. In contrast, DriftXpress uses a Nyström approximation only for the attractive component and keeps the repulsive component exact. Landmarks are selected from the training data before training begins. Unless otherwise stated, we use random per-class landmark selection.

Sharding. For datasets with many classes or large landmark banks, we use the sharded variant of DriftXpress. Sharding partitions the positive support by class, constructs a separate Nyström map for each shard, and accumulates the attractive numerator and denominator sequentially. In the main benchmarks, we use one shard per class: 10 shards for SVHN and CIFAR10, 100 shards for CIFAR100, and 1000 shards for ImageNet. This reduces active query-feature memory for the specified shard-local estimator; it is not equivalent to an unsharded Nyström approximation using all landmarks in one Gram matrix.

Compute. Main benchmark experiments are run on 8 NVIDIA H100 GPUs. Ablations and profiling experiments are run on a single NVIDIA H100 GPU unless otherwise specified. The main benchmark uses a fixed wall-clock budget of 8 hours and a global batch size of $12{,}000$, corresponding to 1500 samples per GPU. We report iteration speed, image throughput, wall-clock runtime, peak GPU memory, and relative speedup against standard drifting under the same dataset and training protocol.

Random seeds and error bars. All main quantitative results are reported as mean $\pm$ standard deviation across three independent seeds. The seeds affect model initialization, data ordering, noise sampling, and landmark selection when random landmarks are used. For each run, we track FID throughout training and report *Best FID*, defined as the lowest FID achieved during the run.

Kernel and temperature. All drifting fields use the Laplace kernel in the frozen feature space. For feature vectors $x, y \in \mathbb{R}^D$, we define

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

where $\tau > 0$ is the kernel temperature. We use $\tau = 0.05$ for all reported experiments. The same kernel and temperature are used for standard drifting and DriftXpress.

FID evaluation. We evaluate sample quality using Fréchet Inception Distance (FID) (Heusel et al., 2017). FID is computed with the standard Inception-v3 feature extractor. For each evaluation checkpoint, we generate samples with the one-step generator, extract Inception features for generated and real images, fit Gaussian statistics to both feature sets, and report the resulting FID. We use the same FID code, preprocessing, reference set, number of generated samples, and evaluation schedule for standard drifting and DriftXpress.

For CIFAR10, CIFAR100, and SVHN, FID is computed using a fixed $50{,}000$-image real reference set from the corresponding dataset and $50{,}000$ generated samples. The reported *Best FID* is the lowest FID observed over the evaluation checkpoints of a training run. For ImageNet, FID is computed against the $50{,}000$ images from the ILSVRC2012 validation split as the real reference distribution. We generate $50{,}000$ samples from the trained generator. For class-conditional ImageNet generation, labels are sampled uniformly over the 1000 classes, corresponding to 50 generated images per class.

Table 3: Dataset statistics.

Dataset	Training samples	Classes	Class Balance	Samples per class
CIFAR10	50,000	10	Balanced	5,000
CIFAR100	50,000	100	Balanced	500
SVHN	73,257	10	Imbalanced	4,659–13,861
ImageNet	1,281,167	1,000	Imbalanced	732–1,300

C.2 Landmark Selection Algorithms

We compare four landmark-selection strategies: random sampling, k-means, k-center, and facility location. These methods differ in whether they prioritize simplicity, density-aware representativeness, worst-case geometric coverage, or average-case similarity coverage. The main paper reports the resulting FID and landmark-selection time for CIFAR10, where global k-means gives the strongest performance among the tested methods.

Random sampling. Random sampling selects M landmarks uniformly without replacement:

$$\mathbf{U} \sim \mathrm{Unif}\{\mathbf{U} \subseteq \mathcal{Y} : |\mathbf{U}| = M\}.$$

It is computationally cheap and unbiased with respect to the empirical distribution, but it does not optimize coverage or reduce redundancy.

k-means. k-means first solves

$$\min_{\mathbf{c}_1,\dots,\mathbf{c}_M} \sum_{i=1}^{N} \min_{m\in[M]} \|\mathbf{y}_i - \mathbf{c}_m\|_2^2,$$

and then selects the nearest real feature to each centroid:

$$\mathbf{u}_m = \arg\min_{\mathbf{y}_i\in\mathcal{Y}} \|\mathbf{y}_i - \mathbf{c}_m\|_2.$$

This gives a medoid-like landmark set with $\mathbf{U} \subseteq \mathcal{Y}$. Because the objective is density-weighted, k-means places more landmarks in high-mass regions of the feature distribution. This is well aligned with DriftXpress, where the field is determined by normalized kernel averages rather than worst-case coverage.

k-center. k-center approximates the minimax covering objective

$$\min_{\mathbf{U}\subseteq\mathcal{Y},\ |\mathbf{U}|=M} \max_{\mathbf{y}_i\in\mathcal{Y}} \min_{\mathbf{u}\in\mathbf{U}} \|\mathbf{y}_i - \mathbf{u}\|_2.$$

The greedy farthest-point rule adds

$$\mathbf{u}_{t+1} = \arg\max_{\mathbf{y}_i\in\mathcal{Y}} \min_{\mathbf{u}\in\mathbf{U}_t} \|\mathbf{y}_i - \mathbf{u}\|_2.$$

This prioritizes worst-case geometric spread. While useful for covering radius, it can overallocate landmarks to isolated or low-density points. For DriftXpress, this is not necessarily desirable because the projected field depends on typical kernel-weighted neighborhoods, not only on the worst-covered feature.

Facility location. Facility location maximizes a similarity-based coverage objective:

$$F(\mathbf{U}) = \sum_{i=1}^{N} \max_{\mathbf{u}\in\mathbf{U}} s(\mathbf{y}_i, \mathbf{u}), \qquad |\mathbf{U}| = M,$$

where $s(\cdot,\cdot)$ is a nonnegative similarity. In our implementation, we use an exponential Euclidean-distance similarity

$$s(x,u) = \exp\left(-\frac{\|x-u\|_2}{\tau}\right),$$

Table 4: Ablation of exact versus projected repulsion. Exact repulsion is the default DriftXpress implementation. Projecting the repulsive term causes severe collapse, showing that repulsion is substantially more sensitive to approximation error than attraction.

Dataset	Repulsion estimator	Best FID ↓
SVHN	Exact	3.11 ± 0.12
SVHN	Nyström	347.6 ± 21.4
CIFAR10	Exact	5.52 ± 0.06
CIFAR10	Nyström	368.3 ± 18.7
CIFAR100	Exact	6.15 ± 0.09
CIFAR100	Nyström	382.9 ± 16.2
ImageNet	Exact	9.21 ± 0.16
ImageNet	Nyström	394.5 ± 13.8

where τ is the temperature-scaled bandwidth. Given a candidate set C and an evaluation set E, the facility-location objective is

$$F(S) = \sum_{x \in E} \max_{u \in S} s(x, u).$$

The greedy update is

$$u_t = \arg \max_{u \in C \setminus S_{t-1}} \sum_{x \in E} \left[s(x, u) - \max_{v \in S_{t-1}} s(x, v) \right]_+ ,$$

and we set $S_t = S_{t-1} \cup \{u_t\}$. Here, the gain measures the additional coverage obtained by adding u beyond the best landmark already selected.

Unlike k-center, facility location optimizes average-case similarity coverage. It is therefore more density-aware, but usually more expensive because marginal gains require similarities to many training points.

C.3 Exact and Projected Repulsion

DriftXpress approximates the attractive component with a projected RKHS field, but keeps the repulsive component exact. We ablate this design choice by also replacing the repulsive term with the same Nyström approximation used for attraction. Table 4 reports the resulting FID.

The result is clear: the method is highly sensitive to the repulsive component. While projecting the attractive term preserves quality and improves speed, a direct reuse of training data landmarks for repulsion caused severe collapse in our experiments. This suggests that repulsion is more sensitive to approximation error and that safely approximating it may require landmarks adapted to the generated distribution. The attractive term mainly pulls generated samples toward the data manifold, and its approximation can be stabilized by precomputing a large training-data summary. The repulsive term, however, controls local diversity among generated samples. Small errors in this term can remove or distort the forces that separate nearby generated samples, causing them to concentrate in a small region of feature space.

This ablation motivates our default design: DriftXpress applies the Nyström approximation only to attraction and keeps repulsion exact. This removes the main training-data bottleneck from the inner loop while preserving the stabilizing effect of exact generated-sample repulsion.

C.4 Inference-time comparison

Table 5 compares the inference-time latency of standard drifting and DriftXpress. The two methods have essentially identical latency across all datasets, with all measurements around 0.177 ms per generated sample. This is expected: DriftXpress changes the field-estimation procedure used during training, but it does not change the generator architecture or the one-step sampling procedure used at inference time.

Table 5: Inference-time benchmark. We report latency in milliseconds per generated sample from the final trained checkpoint. The inference times are similar as there both methods use the same one-step image generation.

Dataset	**Method**	**Inference time (ms/sample)** $\downarrow$
CIFAR10	Standard drifting	0.177512 ± 0.000062
CIFAR10	DriftXpress	0.177348 ± 0.000055
CIFAR100	Standard drifting	0.177442 ± 0.000124
CIFAR100	DriftXpress	0.177417 ± 0.000119
SVHN	Standard drifting	0.177393 ± 0.000004
SVHN	DriftXpress	0.177426 ± 0.000018
ImageNet	Standard drifting	0.177264 ± 0.000027
ImageNet	DriftXpress	0.177285 ± 0.000022

Table 6: Per-iteration field-estimation profile on CIFAR10 using random per-class landmark selection. Entries report mean $\pm$ standard deviation over three seeds. Lower total field-estimation and full-iteration times are bolded.

Component	Standard drifting	DriftXpress (sharded)	DriftXpress (unsharded)
Exact attraction kernels $k(\mathbf{x}_b, \mathbf{y}_j^+)$	157.8 ± 11.7 ms	–	–
Nyström query kernels $K(\mathbf{x}_b, \tilde{\mathbf{U}})$	–	40.1 ± 0.1 ms	$\mathbf{33.7 \pm 0.0}$ ms
Projected attraction summary products	–	20.8 ± 0.0 ms	$\mathbf{15.4 \pm 0.0}$ ms
Remaining field reduction / repulsion / aggregation	111.6 ± 0.1 ms	24.4 ± 0.1 ms	$\mathbf{24.0 \pm 0.1}$ ms
Total field estimation	269.4 ± 11.8 ms	85.3 ± 0.2 ms	$\mathbf{73.2 \pm 0.2}$ ms
Full training iteration	760.3 ± 12.1 ms	576.6 ± 0.9 ms	$\mathbf{564.5 \pm 0.1}$ ms

This confirms the intended trade-off. DriftXpress reduces the wall-clock cost of training while preserving the inference-time advantage of drifting models. Once training is complete, both standard drifting and DriftXpress generate samples with the same single generator evaluation.

C.5 DriftXpress Cost Profile

We profile the online and offline costs of DriftXpress separately. For the online profile, we use the CIFAR10 configuration matching the main benchmark setting. Landmarks are selected by random per-class sampling, and all times are averaged over three seeds. We report both class-sharded and unsharded DriftXpress. For the offline profile, we measure the one-time preprocessing pipeline on SVHN, CIFAR10, and CIFAR100.

C.5.1 Online Cost Profile

Table 6 reports the per-iteration profile and distinguishes field-estimation time from full-iteration time. Standard drifting spends 269.4 ± 11.8 ms on field estimation. DriftXpress reduces this to 85.3 ± 0.2 ms in the sharded configuration and 73.2 ± 0.2 ms in the unsharded configuration, corresponding to reductions of 68.3% and 72.8%, respectively. End-to-end iteration time decreases from 760.3 ± 12.1 ms for standard drifting to 576.6 ± 0.9 ms for sharded DriftXpress and 564.5 ± 0.1 ms for unsharded DriftXpress, corresponding to full-iteration reductions of 24.2% and 25.8%. For unsharded CIFAR10, these are $3.68\times$ and $1.35\times$ speedups in field estimation and full-iteration time, respectively. Table 1 instead reports image throughput from eight GPUs with a global batch of 12,000; its distributed throughput ratio cannot be inferred from the single-GPU step times.

The profile also identifies where the savings occur. Standard drifting spends 157.8 ± 11.7 ms on exact attraction kernels and 111.6 ± 0.1 ms on repulsion, field normalization, and aggregation. DriftXpress removes the exact attraction block and replaces it with Nyström query kernels plus projected attraction. These two blocks cost 40.1 ± 0.1 ms and 20.8 ± 0.0 ms in the sharded variant, and 33.7 ± 0.0 ms and 15.4 ± 0.0 ms in the unsharded variant. The remaining repulsion and aggregation cost is 24.4 ± 0.1 ms for sharded DriftXpress and 24.0 ± 0.1 ms for unsharded DriftXpress. The unsharded

Table 7: One-time DriftXpress preprocessing cost using random per-class landmark selection. Entries report mean ± standard deviation over three seeds. The largest component for each dataset is bolded.

Dataset	Feature extraction	Landmark selection	$\mathbf{K}_{UU}$ build + $\mathbf{K}_{UU}^{-1/2}$	Summary construction	Total
CIFAR10	**19.7 ± 0.4 s**	2.4 ± 0.1 s	7.4 ± 0.1 s	4.6 ± 0.0 s	37.5 ± 0.5 s
CIFAR100	19.6 ± 0.2 s	2.5 ± 0.0 s	**25.2 ± 0.2 s**	6.6 ± 0.1 s	59.6 ± 0.3 s
SVHN	**28.1 ± 2.6 s**	3.7 ± 0.6 s	19.1 ± 0.1 s	9.9 ± 1.8 s	63.0 ± 5.0 s

variant is faster because it avoids per-shard projection and accumulation overhead. The sharded variant is still useful when the full summary does not fit in memory or when classwise partitioning is needed for larger settings.

Total field estimation measures only the cost of computing the drifting field: attraction, repulsion, normalization, and aggregation. Full training iteration measures the entire optimization step, including field estimation as well as the generator forward pass, feature computation, loss evaluation, backpropagation, and optimizer update.

C.5.2 Computational steps in Table 6

Let $\{\mathbf{x}_b\}_{b=1}^{B}$ denote the generated query batch, let $\{\mathbf{y}_j^+\}_{j=1}^{N^+}$ denote the positive reference samples used for attraction, and let $\{\mathbf{y}_\ell^-\}_{\ell=1}^{N^-}$ denote the generated samples used for repulsion. All quantities are evaluated in the pre-trained feature representation space that serves as the input domain of the drifting kernel.

Exact attraction kernels $k(\mathbf{x}_b, \mathbf{y}_j^+)$. In standard drifting, the attractive component forms the pairwise kernel matrix

$$\mathbf{K}_{bj}^+ = k(\mathbf{x}_b, \mathbf{y}_j^+), \qquad b = 1, \dots, B, \quad j = 1, \dots, N^+.$$

These weights define the exact attractive barycenter

$$\boldsymbol{\mu}_p(\mathbf{x}_b) = \frac{\sum_{j=1}^{N^+} \mathbf{K}_{bj}^+ \mathbf{y}_j^+}{\sum_{j=1}^{N^+} \mathbf{K}_{bj}^+ + \varepsilon}.$$

This row measures the time spent forming the exact attractive kernel weights between the generated batch and the positive reference samples. This is the repeated training-data interaction that DriftXpress removes from the online loop.

Nyström query kernels $K(\mathbf{x}_b, \mathbf{U})$. DriftXpress replaces the exact attractive kernel matrix with query-landmark kernels. Given landmarks $\mathbf{U} = \{\mathbf{u}_1, \dots, \mathbf{u}_r\}$, it forms

$$\mathbf{K}_{BU}[b, a] = k(\mathbf{x}_b, \mathbf{u}_a), \qquad b = 1, \dots, B, \quad a = 1, \dots, r.$$

With the precomputed transform

$$\mathbf{W} = (\mathbf{K}_{UU} + \lambda \mathbf{I})^{-1/2},$$

the query features are

$$\boldsymbol{\Phi}_B = \mathbf{K}_{BU}\mathbf{W} \in \mathbb{R}^{B \times r}, \qquad [\boldsymbol{\Phi}_B]_{b,:} = \boldsymbol{\varphi}(\mathbf{x}_b)^\top.$$

This row measures the online cost of computing the Nyström features for the generated queries.

Projected attraction $K_B\tilde{\mathbf{A}}_p, K_B\tilde{\mathbf{b}}_p$. The positive summaries are precomputed offline as

$$\mathbf{A}_p = \sum_{j=1}^{N^+} \boldsymbol{\varphi}(\mathbf{y}_j^+)\mathbf{y}_j^{+\top} \in \mathbb{R}^{r \times D}, \qquad \mathbf{b}_p = \sum_{j=1}^{N^+} \boldsymbol{\varphi}(\mathbf{y}_j^+) \in \mathbb{R}^r.$$

Equivalently, after folding the Nyström transform into the summaries, we denote the stored tensors by $\tilde{\mathbf{A}}_p$ and $\tilde{\mathbf{b}}_p$. The projected attractive numerator and denominator for the full batch are computed as

$$\mathbf{N}_p^{\mathbf{U}} = \mathbf{\Phi}_B \mathbf{A}_p \in \mathbb{R}^{B\times D}, \qquad \mathbf{d}_p^{\mathbf{U}} = \mathbf{\Phi}_B \mathbf{b}_p \in \mathbb{R}^B.$$

For each query,

$$\boldsymbol{\mu}_p^{\mathbf{U}}(\mathbf{x}_b) = \frac{[\mathbf{N}_p^{\mathbf{U}}]_{b,:}}{[\mathbf{d}_p^{\mathbf{U}}]_b + \varepsilon}.$$

This row measures the low-rank matrix products used to recover the attractive barycenter from the precomputed summaries.

Remaining field reduction / repulsion / aggregation. This row collects the remaining online field computation after the attraction-specific block has been evaluated. For standard drifting, it includes the coupled dense exact-field reduction after forming exact attraction kernels, together with the repulsive computation and field aggregation. For DriftXpress, attraction has already been summarized by the projected blocks above, so this row consists of exact batch repulsion and lightweight aggregation.

The exact repulsive component forms the generated-sample kernel matrix

$$\mathbf{K}^-_{b\ell} = k(\mathbf{x}_b, \mathbf{y}^-_\ell), \qquad b = 1,\dots,B, \quad \ell = 1,\dots,N^-.$$

In practice, when the query batch and repulsive support are the same generated batch, the diagonal self-interactions are masked before normalization. The exact repulsive barycenter is

$$\boldsymbol{\mu}_q(\mathbf{x}_b) = \frac{\sum_{\ell=1}^{N^-} \mathbf{K}^-_{b\ell}\mathbf{y}^-_\ell}{\sum_{\ell=1}^{N^-} \mathbf{K}^-_{b\ell}}.$$

The final field is then assembled as

$$\mathbf{V}_{p,q}(\mathbf{x}_b) = \boldsymbol{\mu}_p(\mathbf{x}_b) - \boldsymbol{\mu}_q(\mathbf{x}_b)$$

for standard drifting, and as

$$\mathbf{V}^{\mathrm{DX}}_{p,q}(\mathbf{x}_b) = \boldsymbol{\mu}_p^{\mathbf{U}}(\mathbf{x}_b) - \boldsymbol{\mu}_q(\mathbf{x}_b)$$

for DriftXpress. When multiple temperatures, feature groups, or resolutions are used, this step also includes per-group normalization and aggregation, e.g.

$$\mathbf{V}(\mathbf{x}_b) = \sum_{g\in\mathcal{G}} \alpha_g \frac{\mathbf{V}_g(\mathbf{x}_b)}{\|\mathbf{V}_g(\mathbf{x}_b)\| + \varepsilon}.$$

Thus, this row should not be interpreted as an isolated timing of the negative barycenter alone. It measures the remaining field-reduction, exact-repulsion, normalization, and aggregation work after the method-specific attraction computation has been accounted for.

Total field estimation. This is the total time spent computing the drifting field used to construct the training target. For standard drifting, it includes exact attraction and the remaining field-reduction, repulsion, and aggregation block:

$$T^{\mathrm{std}}_{\mathrm{field}} = T^{\mathrm{exact}}_{\mathrm{attr}} + T^{\mathrm{std}}_{\mathrm{rem}}.$$

For DriftXpress, it includes Nyström query kernels, projected attraction, and the remaining exact-repulsion and aggregation block:

$$T^{\mathrm{DX}}_{\mathrm{field}} = T_{\text{query-Nys}} + T^{\mathrm{proj}}_{\mathrm{attr}} + T^{\mathrm{DX}}_{\mathrm{rem}}.$$

This quantity isolates the cost of estimating the drifting field itself: attraction, repulsion, normalization, and aggregation.

Full training iteration. This is the end-to-end time for one optimization step. It includes field estimation as well as all other training-loop components:

$$T_{\mathrm{iter}} = T_{\mathrm{encoder}} + T_{\mathrm{field}} + T_{\mathrm{loss/backward}} + T_{\mathrm{optimizer}}.$$

Thus, total field estimation measures only the drifting-field computation, while full training iteration measures the complete optimization step, including encoder forward passes, construction of the drifted target, generator loss evaluation, backpropagation, and optimizer update. This is why a large reduction in field-estimation time translates into a smaller, but still substantial, reduction in end-to-end iteration time.



Figure 5: **CIFAR10 samples during training.** Columns $t = 1, \ldots, 7$ show matched checkpoints ordered from earliest to latest. In the displayed samples, recognizable object structure appears at earlier checkpoints with DriftXpress than with standard drifting.

C.5.3 Offline Cost Profile

Table 7 reports one-time preprocessing costs with random per-class landmarks. On CIFAR10, total preprocessing takes 37.5 ± 0.5 s. Feature extraction is the largest component at 19.7 ± 0.4 s, followed by the combined $\mathbf{K}_{UU}$ construction and $\mathbf{K}_{UU}^{-1/2}$ factorization stage at 7.4 ± 0.1 s. On CIFAR100, total preprocessing increases to 59.6 ± 0.3 s, with the $\mathbf{K}_{UU}$ stage becoming the largest component at 25.2 ± 0.2 s. On SVHN, preprocessing takes 63.0 ± 5.0 s, dominated by feature extraction at 28.1 ± 2.6 s and the $\mathbf{K}_{UU}$ stage at 19.1 ± 0.1 s. Random landmark selection is cheap across these datasets, taking only 2.4–3.7 s.

Takeaway. DriftXpress moves the training-data interactions to a one-time preprocessing stage, after which attraction is evaluated through Nyström query kernels and low-rank matrix products. With random per-class landmarks, preprocessing takes under one minute on CIFAR10 and CIFAR100 and about one minute on SVHN. The main offline costs are feature extraction and $\mathbf{K}_{UU}$ construction/factorization, while landmark selection is negligible.

These measurements are from a single-GPU profile. In multi-GPU training, the benefit of removing exact training-data interactions from the inner loop can be larger, since each device evaluates the drifting field for its local generated batch. Standard drifting repeats attraction against the training support on every device and every step, whereas DriftXpress reuses the same precomputed summaries and only evaluates query-landmark kernels plus low-rank products locally. Thus, the one-time preprocessing cost is amortized across devices and training steps, while the per-iteration savings scale with the number of generated batches evaluated during training.

C.6 Early Training Snapshots

Figure 5 compares CIFAR10 samples from DriftXpress and standard drifting at seven matched training checkpoints.

C.7 Field Approximation Fidelity

We next isolate the quality of the projected field approximation, independent of generator training. On CIFAR10, we compare the exact attractive field $V_p^+(\mathbf{x})$ with the projected attractive field $V_{p,U}^+(\mathbf{x})$ induced by different landmark budgets. We use random per-class landmark selection with 25, 50, 100, 250, 500, 1000, and 2000 landmarks per class. Since CIFAR10 has 5000 training samples per class, these budgets correspond to 0.5%, 1%, 2%, 5%, 10%, 20%, and 40% of each class support, respectively. For each budget, we report three fidelity metrics. Cosine similarity measures directional agreement between the exact and projected drift vectors:

$$\frac{1}{B}\sum_{b=1}^{B}\frac{\langle V_p^+(\mathbf{x}_b), V_{p,U}^+(\mathbf{x}_b)\rangle}{\|V_p^+(\mathbf{x}_b)\|_2\,\|V_{p,U}^+(\mathbf{x}_b)\|_2}.$$

Table 8: Field approximation fidelity on CIFAR10.

Landmarks / class	Total landmarks	Cosine similarity ↑	Relative ℓ_2 error ↓	Target MSE ↓
25	250	0.7426 ± 0.0002	0.8042 ± 0.0019	0.6468 ± 0.0031
50	500	0.8166 ± 0.0010	0.6820 ± 0.0043	0.4651 ± 0.0058
100	1000	0.8785 ± 0.0006	0.5674 ± 0.0035	0.3219 ± 0.0039
250	2500	0.9408 ± 0.0005	0.4235 ± 0.0039	0.1794 ± 0.0033
500	5000	0.9683 ± 0.0003	0.3250 ± 0.0067	0.1057 ± 0.0044
1000	10000	0.9840 ± 0.0007	0.2456 ± 0.0182	0.0607 ± 0.0088
2000	20000	$\mathbf{0.9930 \pm 0.0007}$	$\mathbf{0.1758 \pm 0.0273}$	$\mathbf{0.0317 \pm 0.0096}$

Relative ℓ_2 error measures the size of the approximation error normalized by the size of the exact field:

$$\frac{\|V^+_{p,U}(\mathbf{X}) - V^+_p(\mathbf{X})\|_F}{\|V^+_p(\mathbf{X})\|_F}.$$

Target MSE measures the mean squared error between the exact drift target and the projected drift target, where $\mathbf{t}_b = \mathbf{x}_b + V^+_p(\mathbf{x}_b)$ and $\mathbf{t}_{b,U} = \mathbf{x}_b + V^+_{p,U}(\mathbf{x}_b)$:

$$\frac{1}{BD}\sum_{b=1}^{B} \|\mathbf{t}_{b,U} - \mathbf{t}_b\|_2^2.$$

Thus, cosine similarity evaluates whether the projected field points in the right direction, relative ℓ_2 error evaluates the normalized field-distortion magnitude, and target MSE evaluates the error in the actual regression target used by the drifting loss.

Table 8 shows that increasing the landmark budget consistently improves the fidelity of the projected field. With only 25 landmarks per class, the projected field has cosine similarity 0.7426 ± 0.0002 and relative ℓ_2 error 0.8042 ± 0.0019, indicating that the landmark basis is too small to accurately reproduce the exact field. At 500 landmarks per class, corresponding to 10% of each class support, cosine similarity increases to 0.9683 ± 0.0003 and relative ℓ_2 error drops to 0.3250 ± 0.0067. At the largest budget, 2000 landmarks per class, the projected field is very close to the exact field, reaching cosine similarity 0.9930 ± 0.0007, relative ℓ_2 error 0.1758 ± 0.0273, and target MSE 0.0317 ± 0.0096.

This trend directly matches the approximation guarantees in Theorem 3.2 and Corollary 3.3. Theorem 3.2 shows that the query-wise distortion of the projected attractive field is controlled by the landmark kernel residual $\|r_U(\mathbf{x})\|_2$ normalized by the local kernel mass $d_p(\mathbf{x})$. Corollary 3.3 gives the corresponding on-support statement: the average squared field distortion is controlled by the Gram approximation error $\|\mathbf{K} - \mathbf{K}_U\|_F^2$. Increasing the number of landmarks improves the Nyström approximation of the training-support kernel matrix, reducing $\|\mathbf{K} - \mathbf{K}_U\|_F^2$ and therefore reducing the field error predicted by the theory. The empirical decrease in relative ℓ_2 error and target MSE in Table 8 is consistent with this prediction. It also explains why DriftXpress can preserve sample quality in the main benchmark: once the landmark basis captures the dominant geometry of the training distribution, the projected attractive field closely matches the exact attractive field while being much cheaper to evaluate during training.

Figure 6 compares class-conditional samples from DriftXpress and standard drifting on CIFAR10. Figure 7 provides the same comparison for 10 randomly selected CIFAR100 classes. These qualitative results are intended to complement the FID scores in Section 4: while DriftXpress changes the estimator used during training, it does not change the generator architecture, the one-step sampling procedure, or the target image domain. All profiled models correspond to the same checkpoints reported in Table 1.

Across both datasets, the samples produced by DriftXpress are visually comparable to those produced by standard drifting. The class identity is generally preserved, and we do not observe an obvious qualitative degradation from replacing exact attraction with projected summaries. This supports the main empirical claim that the computational gains of DriftXpress do not come from a visibly weaker generator, but from reducing the cost of estimating the attractive component of the drifting field. The CIFAR100 comparison is particularly useful because the larger number of classes makes the attraction summaries more demanding; nevertheless, the projected field still produces samples that are qualitatively close to the standard drifting baseline.



Figure 6: **CIFAR10 samples generated by DriftXpress and standard drifting.** For each class, the top row shows samples from DriftXpress and the bottom row shows samples from standard drifting. Both methods produce visually comparable samples across all CIFAR10 categories.

D PROOFS

Theorem D.1 (Compositionality of Nyström attractive summaries)**.** *Let*

$$\bar{\varphi}(\mathbf{x}) := \left[\varphi^{(1)}(\mathbf{x}); \ldots; \varphi^{(S)}(\mathbf{x})\right]$$

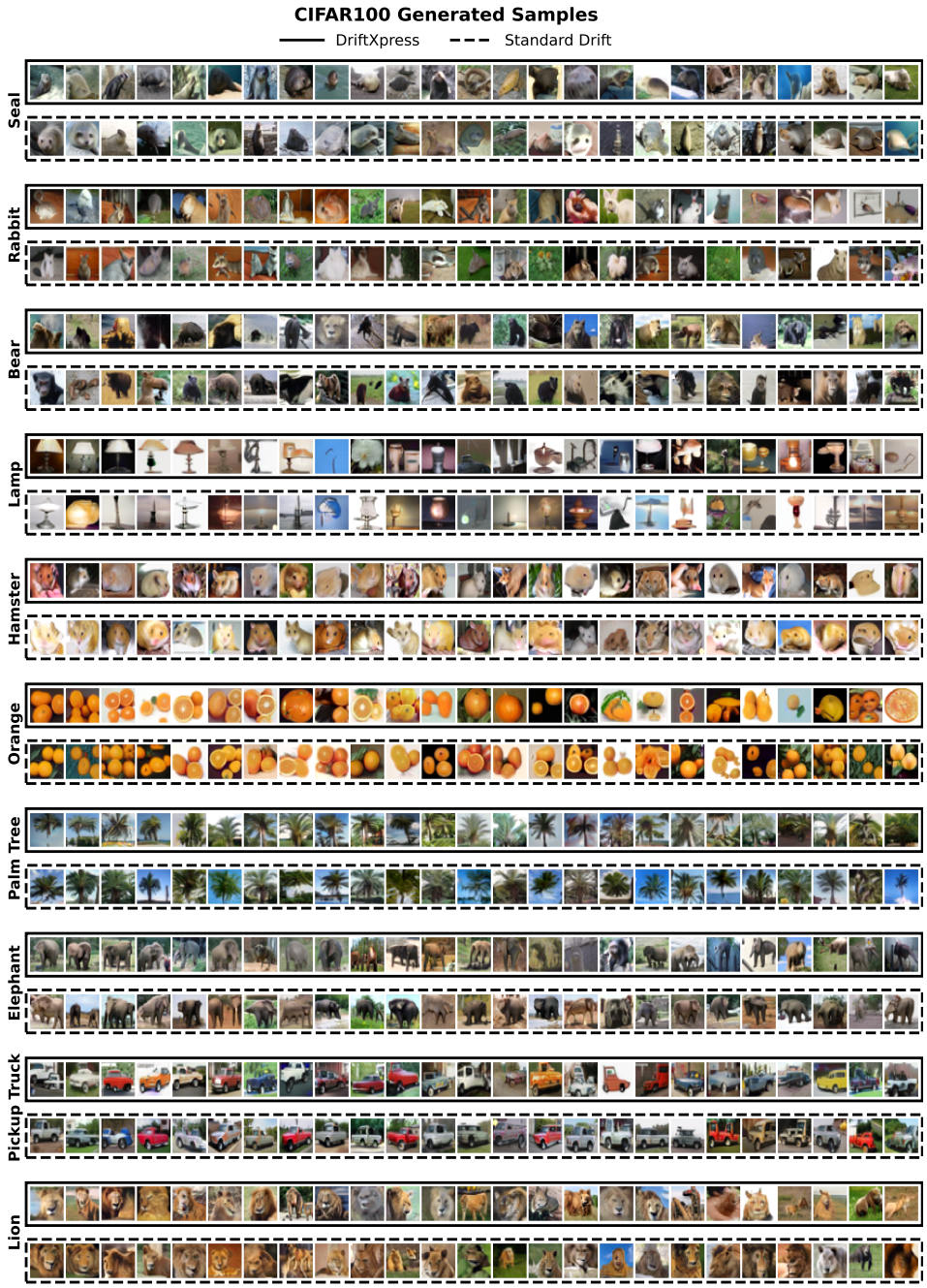

Figure 7: **CIFAR100 samples generated by DriftXpress and standard drifting.** We show 10 randomly selected CIFAR100 classes. For each class, the top row shows samples from DriftXpress and the bottom row shows samples from standard drifting. Both methods produce visually comparable samples across the selected CIFAR100 categories.

denote the concatenation of the shard feature maps, and define the concatenated attractive summaries

$$\bar{\mathbf{A}}_p := \begin{bmatrix} \mathbf{A}_p^{(1)} \\ \vdots \\ \mathbf{A}_p^{(S)} \end{bmatrix}, \qquad \bar{\mathbf{b}}_p := \begin{bmatrix} \mathbf{b}_p^{(1)} \\ \vdots \\ \mathbf{b}_p^{(S)} \end{bmatrix}.$$

Then the compositional attractive mean satisfies

$$\mu_{p,\mathrm{comp}}(\mathbf{x}) = \frac{\bar{\boldsymbol{\varphi}}(\mathbf{x})^\top \bar{\mathbf{A}}_p}{\bar{\boldsymbol{\varphi}}(\mathbf{x})^\top \bar{\mathbf{b}}_p + \varepsilon}.$$

Consequently, when the same exact repulsive mean $\mu_q(\mathbf{x})$ is used, the compositional field

$$\mathbf{V}_{p,q}^{\mathrm{comp}}(\mathbf{x}) := \mu_{p,\mathrm{comp}}(\mathbf{x}) - \mu_q(\mathbf{x})$$

is exactly equal to the field obtained from a single concatenated attractive summary and the same exact repulsive term.

Proof. By the definition of block concatenation,

$$\bar{\boldsymbol{\varphi}}(\mathbf{x})^\top \bar{\mathbf{A}}_p = \sum_{s=1}^{S} \boldsymbol{\varphi}^{(s)}(\mathbf{x})^\top \mathbf{A}_p^{(s)}, \qquad \bar{\boldsymbol{\varphi}}(\mathbf{x})^\top \bar{\mathbf{b}}_p = \sum_{s=1}^{S} \boldsymbol{\varphi}^{(s)}(\mathbf{x})^\top \mathbf{b}_p^{(s)}.$$

Substituting these identities into the single-summary attractive mean gives

$$\frac{\bar{\boldsymbol{\varphi}}(\mathbf{x})^\top \bar{\mathbf{A}}_p}{\bar{\boldsymbol{\varphi}}(\mathbf{x})^\top \bar{\mathbf{b}}_p + \varepsilon} = \frac{\sum_{s=1}^{S} \boldsymbol{\varphi}^{(s)}(\mathbf{x})^\top \mathbf{A}_p^{(s)}}{\sum_{s=1}^{S} \boldsymbol{\varphi}^{(s)}(\mathbf{x})^\top \mathbf{b}_p^{(s)} + \varepsilon} = \mu_{p,\mathrm{comp}}(\mathbf{x}).$$

The repulsive mean $\mu_q(\mathbf{x})$ is computed exactly over the current generated batch and is unchanged by the sharding of the positive summaries. Therefore,

$$\mathbf{V}_{p,q}^{\mathrm{comp}}(\mathbf{x}) = \mu_{p,\mathrm{comp}}(\mathbf{x}) - \mu_q(\mathbf{x})$$

is exactly the field obtained by evaluating the concatenated attractive summary and subtracting the same exact repulsive mean. □

For the following results, set $\alpha := \varepsilon / N^+$.

Theorem D.2 (Distortion of the projected attractive field)**.** *Let $\{\mathbf{y}_j^+\}_{j=1}^{N^+} \subset \mathbb{R}^D$ satisfy*

$$\|\mathbf{y}_j^+\| \le R, \qquad j = 1, \dots, N^+,$$

and let k be a nonnegative kernel. Let $k_{\mathbf{U}}$ be any projected kernel induced by a landmark set $\mathbf{U}$, for example the Nyström kernel from Section 3. For any query $\mathbf{x} \in \mathbb{R}^D$, define

$$\boldsymbol{\mu}_p(\mathbf{x}) := \frac{\frac{1}{N^+}\sum_{j=1}^{N^+} k(\mathbf{x}, \mathbf{y}_j^+)\mathbf{y}_j^+}{\frac{1}{N^+}\sum_{j=1}^{N^+} k(\mathbf{x}, \mathbf{y}_j^+) + \alpha}, \qquad \boldsymbol{\mu}_p^{\mathbf{U}}(\mathbf{x}) := \frac{\frac{1}{N^+}\sum_{j=1}^{N^+} k_{\mathbf{U}}(\mathbf{x}, \mathbf{y}_j^+)\mathbf{y}_j^+}{\frac{1}{N^+}\sum_{j=1}^{N^+} k_{\mathbf{U}}(\mathbf{x}, \mathbf{y}_j^+) + \alpha},$$

and

$$\mathbf{V}_p^+(\mathbf{x}) := \boldsymbol{\mu}_p(\mathbf{x}) - \mathbf{x}, \qquad \mathbf{V}_{p,\mathbf{U}}^+(\mathbf{x}) := \boldsymbol{\mu}_p^{\mathbf{U}}(\mathbf{x}) - \mathbf{x}.$$

Define the query-wise kernel residual vector

$$\mathbf{r}_{\mathbf{U}}(\mathbf{x}) := \big(k(\mathbf{x}, \mathbf{y}_1^+) - k_{\mathbf{U}}(\mathbf{x}, \mathbf{y}_1^+), \dots, k(\mathbf{x}, \mathbf{y}_{N^+}^+) - k_{\mathbf{U}}(\mathbf{x}, \mathbf{y}_{N^+}^+)\big)^\top \in \mathbb{R}^{N^+},$$

and the exact positive normalizer

$$d_p(\mathbf{x}) := \frac{1}{N^+}\sum_{j=1}^{N^+} k(\mathbf{x}, \mathbf{y}_j^+).$$

Assume $d_p(\mathbf{x}) > 0$. If

$$\|\mathbf{r}_{\mathbf{U}}(\mathbf{x})\|_2 \le \frac{\sqrt{N^+}\,(d_p(\mathbf{x}) + \alpha)}{2}, \tag{17}$$

then

$$\big\|\mathbf{V}_{p,\mathbf{U}}^+(\mathbf{x}) - \mathbf{V}_p^+(\mathbf{x})\big\| \le \frac{4R}{\sqrt{N^+}\,(d_p(\mathbf{x}) + \alpha)} \|\mathbf{r}_{\mathbf{U}}(\mathbf{x})\|_2. \tag{18}$$

Moreover, for the asymmetric DriftXpress field that uses projected attraction and exact repulsion,

$$\mathbf{V}_{p,p}^{\mathbf{U}}(\mathbf{x}) = \boldsymbol{\mu}_p^{\mathbf{U}}(\mathbf{x}) - \boldsymbol{\mu}_p(\mathbf{x}),$$

and therefore the same bound controls the equilibrium residual $\|\mathbf{V}_{p,p}^{\mathbf{U}}(\mathbf{x})\|$.

Proof. Fix a query $\mathbf{x}$. Define

$$\mathbf{n}_p(\mathbf{x}) := \frac{1}{N^+}\sum_{j=1}^{N^+} k(\mathbf{x}, \mathbf{y}_j^+)\mathbf{y}_j^+, \qquad d_p(\mathbf{x}) := \frac{1}{N^+}\sum_{j=1}^{N^+} k(\mathbf{x}, \mathbf{y}_j^+),$$

and similarly

$$\mathbf{n}_{p,\mathbf{U}}(\mathbf{x}) := \frac{1}{N^+}\sum_{j=1}^{N^+} k_{\mathbf{U}}(\mathbf{x}, \mathbf{y}_j^+)\mathbf{y}_j^+, \qquad d_{p,\mathbf{U}}(\mathbf{x}) := \frac{1}{N^+}\sum_{j=1}^{N^+} k_{\mathbf{U}}(\mathbf{x}, \mathbf{y}_j^+).$$

Then

$$\boldsymbol{\mu}_p(\mathbf{x}) = \frac{\mathbf{n}_p(\mathbf{x})}{d_p(\mathbf{x})+\alpha}, \qquad \boldsymbol{\mu}_p^{\mathbf{U}}(\mathbf{x}) = \frac{\mathbf{n}_{p,\mathbf{U}}(\mathbf{x})}{d_{p,\mathbf{U}}(\mathbf{x})+\alpha}.$$

Since the $\mathbf{x}$ terms cancel,

$$\mathbf{V}_{p,\mathbf{U}}^+(\mathbf{x}) - \mathbf{V}_p^+(\mathbf{x}) = \boldsymbol{\mu}_p^{\mathbf{U}}(\mathbf{x}) - \boldsymbol{\mu}_p(\mathbf{x}) = \frac{\mathbf{n}_{p,\mathbf{U}}(\mathbf{x})}{d_{p,\mathbf{U}}(\mathbf{x})+\alpha} - \frac{\mathbf{n}_p(\mathbf{x})}{d_p(\mathbf{x})+\alpha}.$$

By definition of $\mathbf{r}_{\mathbf{U}}(\mathbf{x})$,

$$\mathbf{n}_{p,\mathbf{U}}(\mathbf{x}) - \mathbf{n}_p(\mathbf{x}) = -\frac{1}{N^+}\sum_{j=1}^{N^+}[\mathbf{r}_{\mathbf{U}}(\mathbf{x})]_j\,\mathbf{y}_j^+,$$

and

$$d_{p,\mathbf{U}}(\mathbf{x}) - d_p(\mathbf{x}) = -\frac{1}{N^+}\sum_{j=1}^{N^+}[\mathbf{r}_{\mathbf{U}}(\mathbf{x})]_j.$$

Using $\|\mathbf{y}_j^+\| \le R$,

$$\|\mathbf{n}_{p,\mathbf{U}}(\mathbf{x}) - \mathbf{n}_p(\mathbf{x})\| \le \frac{R}{N^+}\|\mathbf{r}_{\mathbf{U}}(\mathbf{x})\|_1 \le \frac{R}{\sqrt{N^+}}\|\mathbf{r}_{\mathbf{U}}(\mathbf{x})\|_2,$$

and

$$|d_{p,\mathbf{U}}(\mathbf{x}) - d_p(\mathbf{x})| \le \frac{1}{N^+}\|\mathbf{r}_{\mathbf{U}}(\mathbf{x})\|_1 \le \frac{1}{\sqrt{N^+}}\|\mathbf{r}_{\mathbf{U}}(\mathbf{x})\|_2.$$

Now add and subtract $\mathbf{n}_p(\mathbf{x})/(d_{p,\mathbf{U}}(\mathbf{x})+\alpha)$:

$$\left\|\frac{\mathbf{n}_{p,\mathbf{U}}(\mathbf{x})}{d_{p,\mathbf{U}}(\mathbf{x})+\alpha} - \frac{\mathbf{n}_p(\mathbf{x})}{d_p(\mathbf{x})+\alpha}\right\| \le \frac{\|\mathbf{n}_{p,\mathbf{U}}(\mathbf{x}) - \mathbf{n}_p(\mathbf{x})\| + \left\|\frac{\mathbf{n}_p(\mathbf{x})}{d_p(\mathbf{x})+\alpha}\right\| |d_{p,\mathbf{U}}(\mathbf{x}) - d_p(\mathbf{x})|}{d_{p,\mathbf{U}}(\mathbf{x})+\alpha}.$$

Since k is nonnegative and $d_p(\mathbf{x}) > 0$, $\mathbf{n}_p(\mathbf{x})/(d_p(\mathbf{x}) + \alpha)$ is a nonnegative weighted average of $\{\mathbf{y}_j^+\}_{j=1}^{N^+}$ with total weight at most one. Therefore,

$$\left\|\frac{\mathbf{n}_p(\mathbf{x})}{d_p(\mathbf{x})+\alpha}\right\| \le R.$$

Moreover,

$$d_{p,\mathbf{U}}(\mathbf{x})+\alpha \ge d_p(\mathbf{x})+\alpha - |d_{p,\mathbf{U}}(\mathbf{x}) - d_p(\mathbf{x})| \ge d_p(\mathbf{x})+\alpha - \frac{1}{\sqrt{N^+}}\|\mathbf{r}_{\mathbf{U}}(\mathbf{x})\|_2.$$

Under equation 17,

$$d_{p,\mathbf{U}}(\mathbf{x})+\alpha \ge \frac{d_p(\mathbf{x})+\alpha}{2}.$$

Substituting these bounds gives

$$\left\|\frac{\mathbf{n}_{p,\mathbf{U}}(\mathbf{x})}{d_{p,\mathbf{U}}(\mathbf{x})+\alpha} - \frac{\mathbf{n}_p(\mathbf{x})}{d_p(\mathbf{x})+\alpha}\right\| \le \frac{\frac{R}{\sqrt{N^+}}\|\mathbf{r}_{\mathbf{U}}(\mathbf{x})\|_2 + R\frac{1}{\sqrt{N^+}}\|\mathbf{r}_{\mathbf{U}}(\mathbf{x})\|_2}{(d_p(\mathbf{x})+\alpha)/2} = \frac{4R}{\sqrt{N^+}\,(d_p(\mathbf{x})+\alpha)}\|\mathbf{r}_{\mathbf{U}}(\mathbf{x})\|_2.$$

This proves equation 18.

For the final claim, when the model distribution matches the positive distribution, the implemented asymmetric field uses projected attraction and exact repulsion:

$$\mathbf{V}^{\mathbf{U}}_{p,p}(\mathbf{x}) = \boldsymbol{\mu}^{\mathbf{U}}_p(\mathbf{x}) - \boldsymbol{\mu}_p(\mathbf{x}).$$

Thus,

$$\|\mathbf{V}^{\mathbf{U}}_{p,p}(\mathbf{x})\| = \|\boldsymbol{\mu}^{\mathbf{U}}_p(\mathbf{x}) - \boldsymbol{\mu}_p(\mathbf{x})\|,$$

which is exactly the quantity bounded above. □

Corollary D.3 (On-support distortion)**.** *Let* $\mathbf{K}, \mathbf{K}_{\mathbf{U}} \in \mathbb{R}^{N^+ \times N^+}$ *be the exact and projected Gram matrices on* $\{\mathbf{y}^+_j\}_{j=1}^{N^+}$*, with entries*

$$[\mathbf{K}]_{ij} = k(\mathbf{y}^+_i, \mathbf{y}^+_j), \qquad [\mathbf{K}_{\mathbf{U}}]_{ij} = k_{\mathbf{U}}(\mathbf{y}^+_i, \mathbf{y}^+_j).$$

Define

$$\kappa_{\min} := \min_{1 \le i \le N^+} \frac{1}{N^+} \sum_{j=1}^{N^+} k(\mathbf{y}^+_i, \mathbf{y}^+_j),$$

and assume

$$\|\mathbf{K} - \mathbf{K}_{\mathbf{U}}\|_{2,\infty} := \max_{1 \le i \le N^+} \|(\mathbf{K} - \mathbf{K}_{\mathbf{U}})_{i,:}\|_2 \le \frac{\sqrt{N^+}\,(\kappa_{\min} + \alpha)}{2}.$$

Then

$$\frac{1}{N^+} \sum_{i=1}^{N^+} \left\| \mathbf{V}^+_{p,\mathbf{U}}(\mathbf{y}^+_i) - \mathbf{V}^+_p(\mathbf{y}^+_i) \right\|^2 \le \frac{16R^2}{{N^+}^2(\kappa_{\min} + \alpha)^2} \|\mathbf{K} - \mathbf{K}_{\mathbf{U}}\|_F^2.$$

Proof. Apply Theorem D.2 with $\mathbf{x} = \mathbf{y}^+_i$. Then the residual vector $\mathbf{r}(\mathbf{y}^+_i)$ is exactly the ith row of $\mathbf{K} - \mathbf{K}_{\mathbf{U}}$, and $d(\mathbf{y}^+_i) \ge \kappa_{\min}$. Thus, for every i,

$$\left\| \mathbf{V}^+_{p,\mathbf{U}}(\mathbf{y}^+_i) - \mathbf{V}^+_p(\mathbf{y}^+_i) \right\| \le \frac{4R}{\sqrt{N^+}\,(\kappa_{\min} + \alpha)} \|(\mathbf{K} - \mathbf{K}_{\mathbf{U}})_{i,:}\|_2.$$

Square and average over i, then use

$$\sum_{i=1}^{N^+} \|(\mathbf{K} - \mathbf{K}_{\mathbf{U}})_{i,:}\|_2^2 = \|\mathbf{K} - \mathbf{K}_{\mathbf{U}}\|_F^2.$$

□