Journal of Machine Learning Research 27 (2026) 1-49 Submitted 7/25; Revised 2/26; Published 3/26

Robust training of implicit generative models for multivariate and heavy-tailed distributions with an invariant statistical loss

José Manuel de Frutos[a,*] JOFRUTOS@ING.UC3M.ES
Manuel A. Vázquez[a] MAVAZQUE@ING.UC3M.ES
Pablo M. Olmos[a] PAMARTIN@ING.UC3M.ES
Joaquín Míguez[a] JOAQUIN.MIGUEZ@UC3M.ES
[a] *Department of Signal Theory and Communications, Universidad Carlos III de Madrid, Leganés 28911, Madrid, Spain*

Editor: Aapo Hyvärinen

Abstract

Implicit generative models are often trained adversarially, which can yield unstable dynamics and mode collapse. The *invariant statistical loss* (ISL) offers a fully sample-based alternative by comparing empirical ranks of real and generated samples. In this work, we formally characterize ISL as a proper divergence over continuous distributions and establish key regularity properties, showing that it is continuous and differentiable, thereby enabling stable gradient-based optimization without adversarial games. We further enhance ISL along two practical axes. First, to better model heavy-tailed data, where Gaussian latent priors can limit tail expressivity, we introduce Pareto-ISL, which replaces Gaussian noise with a generalized Pareto latent distribution to improve the representation of both typical and extreme events. Second, to handle multivariate data at scale, we propose ISL-slicing: a computationally efficient procedure that projects samples onto random one-dimensional subspaces, computes rank-based losses per projection, and averages them to capture high-dimensional structure. Experiments demonstrate improved tail fidelity with Pareto-ISL and show that ISL-slicing scales effectively to high dimensions. Specifically, in high dimensional settings we show that ISL can be used either as a standalone criterion or as a strong pretraining objective for subsequent adversarial fine-tuning.

Keywords: implicit generative models, deep generative models, mode collapse, heavy-tailed distributions, multivariate distributions

1. Introduction

1.1 Motivation

Generative modeling underpins many machine learning applications, including density estimation, data augmentation, and unsupervised representation learning (Kingma and Welling, 2014; Ho et al., 2020; Papamakarios et al., 2021; Shorten and Khoshgoftaar, 2019). A common distinction is between *prescribed* models, which optimize an explicit likelihood or a variational surrogate (e.g., Variational autoencoder and diffusion models), and *implicit* models, which define a sampler by transforming a simple latent prior through a neural network without

∗. Corresponding author.

requiring tractable density evaluation (Mohamed and Lakshminarayanan, 2016). Generative adversarial networks (GANs) are a prominent example of the latter (Goodfellow et al., 2014).

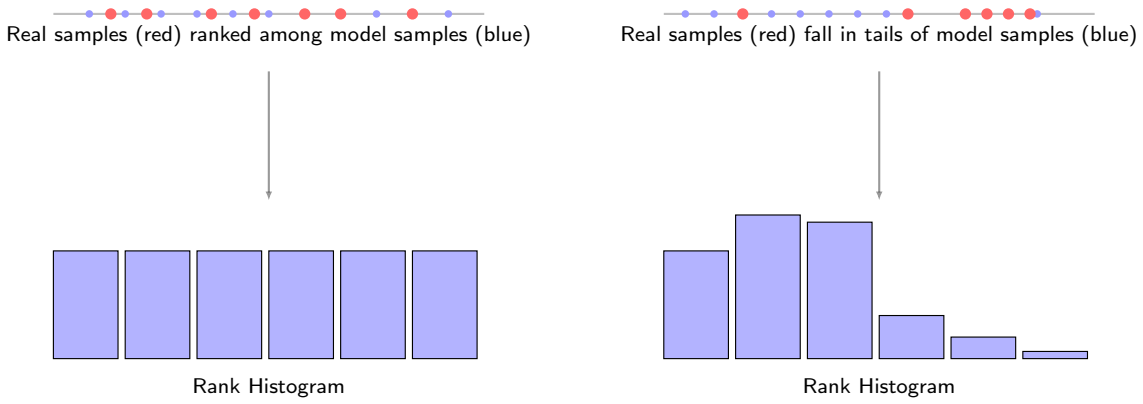
GANs can produce high-fidelity samples, but their training objective is a two-player game that is often difficult to optimize. Instabilities may arise from non-convex/non-concave dynamics and imperfect discriminator training, and generators may exhibit *mode collapse*, allocating mass to only a subset of the data distribution (Arora et al., 2018). A large body of work has therefore aimed to stabilize adversarial learning, either by modifying the discrepancy being optimized or by constraining the critic. For instance, Wasserstein GANs replace the Jensen–Shannon divergence with a Wasserstein-type objective that yields more informative gradients, provided the critic is (approximately) Lipschitz (Arjovsky et al., 2017). Practical implementations enforce this constraint through regularization mechanisms such as gradient penalties or spectral normalization (Gulrajani et al., 2017; Miyato et al., 2018). Architectural and training refinements further improve sample quality and robustness at scale (Brock et al., 2018; Karras et al., 2019). Overall, these approaches mitigate failure modes in many settings, but they still rely on adversarial min–max optimization and typically require careful tuning of optimization and regularization choices.

A separate challenge, particularly relevant beyond standard vision benchmarks, is *tail fidelity*. Many real-world distributions are heavy-tailed, meaning that rare but consequential events carry substantial probability mass and strongly influence downstream decisions (e.g., in finance, climate modeling, and anomaly detection) (Lu et al., 2023; Gong et al., 2024; Seo et al., 2024). In such regimes, implicit models can under-represent extremes, and adversarial training can become especially sensitive because the critic must resolve discrepancies driven by scarce tail samples while remaining stable. Recent methods have begun to explicitly target extremes, for example by emphasizing rare events in the objective or by injecting heavy-tailed latent noise (Bhatia et al., 2021; Huster et al., 2021); however, these approaches still inherit adversarial dynamics and do not by themselves provide a simple, critic-free discrepancy that scales gracefully beyond low-dimensional settings.

To address these issues, we build on the invariant statistical loss (ISL) framework of de Frutos et al. (2024), which replaces the discriminator with a sample-only discrepancy derived from invariance properties of rank statistics.

At the core of ISL lies the following observation: draw K i.i.d. samples $\tilde{y}_1, \ldots, \tilde{y}_K \sim \tilde{p}$ from the generative model and take an independent real data point $y \sim p$. Consider the *rank* of y among the K model samples, i.e., the number of indices i such that $\tilde{y}_i < y$. This rank is uniformly distributed on $\{0, \ldots, K\}$ if and only if $p = \tilde{p}$ (see Figure 1). Thus, the rank statistic provides an implicit, distribution-free certificate of equality between the data and model distributions without evaluating p.

Repeating this procedure over multiple data points yields an empirical rank histogram. ISL defines a loss by measuring how far this histogram deviates from uniformity, and optimizes it via a differentiable surrogate that mimics the rank histogram. The resulting objective is fully sample-based, avoids min–max games, and can be extended to address key limitations of current implicit methods, including poor tail fidelity and scalability to high-dimensional data.



(a) Ideal case: $p = \tilde{p}$ **(b) Model mismatch:** $p \neq \tilde{p}$

Figure 1: High-level intuition behind ISL. When the distribution $\tilde{p}$ (distribution associated to the generator neural network) matches the data distribution p, the ranks of real samples y among generated samples $\tilde{y}_1, \ldots, \tilde{y}_K$ are uniformly distributed (Figure (a)). If the distributions differ, the rank histogram becomes unbalanced (Figure (b)), revealing a difference that ISL uses to detect how well the model fits the data. Red points correspond to real data samples from p, while blue points represent samples from $\tilde{p}$ (model samples).

1.2 Contributions

In this work, we address each of the limitations outlined above. We first formally define the ISL as a discrepancy between two probability distributions, and then prove that it constitutes a valid divergence by establishing its continuity and (almost everywhere) differentiability with respect to the generator parameters. To overcome the tail modeling issue, we introduce *Pareto-ISL*, which replaces the latent noise with a Generalized Pareto distribution and leverages unbounded generators to capture the behavior of heavy-tailed distributions. For high-dimensional data, we propose *ISL-slicing*, a scalable extension that yields a multivariate ISL by averaging over random one-dimensional projections that preserve statistical dependencies while avoiding marginal-factorization or adversarial discriminators. Collectively, we believe that these advances make ISL a viable and competitive framework for modern generative modeling across univariate, multivariate, and heavy-tailed settings. A more detailed description of the contributions of the work is provided below.

1. Divergence guarantees. We establish that the ISL, denoted $d_K(p, \tilde{p})$, defines a valid statistical divergence. Moreover, when the generator model $g_\theta(z)$, with θ a vector of parameters and z a random input, satisfies mild regularity conditions (such as continuity and a uniform Lipschitz property) $d_K(p, \tilde{p}_\theta)$ becomes a Lipschitz-continuous function of θ, and hence differentiable almost everywhere. These properties justify the use of d_K as a theoretically sound and optimizable objective for training implicit generative models.

2. Pareto-ISL for heavy tailed data distributions. We extend ISL to model heavy-tailed distributions by replacing the standard Gaussian input with generalized Pareto noise, following principles from extreme value theory (Coles et al., 2001). This adaptation enables the generator to more accurately capture extreme-value behavior in the tails

while maintaining fidelity in the central region of the distribution. Empirical results show that Pareto-ISL models distributions exhibiting heavy-tailed behavior more faithfully, improving on both Gaussian-latent ISL and adversarial schemes.

3. Scalable multivariate training with ISL-slicing. We introduce ISL-slicing, a scalable divergence for high-dimensional data that computes one-dimensional ISL ranks along m random projections on the $(d-1)$-dimensional hypersphere (where d is the dimension of the data space) and averages the results. ISL-slicing avoids the need for marginal factorization or adversarial training, maintains sensitivity to joint structure, and runs in $\mathcal{O}(mNK)$ time (where m is the number of random projections, N is the batch size, and K is the number of fictitious samples), making it efficient for high-dimensional settings.

4. Empirical validation and GAN pretraining. We demonstrate the effectiveness of ISL methods across a range of benchmarks, showing performance on par with state-of-the-art implicit models. Furthermore, we show that ISL-slicing also serves as an effective pretraining objective for GANs, significantly reducing mode collapse and enhancing sample diversity. In our experiments, ISL pretraining enables simple GANs to outperform more sophisticated (and computationally costly) adversarial models.

1.3 Organization of the paper

In Section 2 we recall basic material on rank statistics from de Frutos et al. (2024), introduce the ISL divergence and its smooth surrogate, and show that it defines a valid statistical divergence that is differentiable in the generator parameters. Section 3 presents Pareto-ISL for heavy-tailed modeling. In Section 4 we define ISL-slicing (averaging 1D ISL over random projections) to extend the proposed divergence to multivariate settings and demonstrate its effectiveness across various benchmarks. Section 5 applies these techniques to time-series forecasting. Finally, Sections 6 and 7 discuss the limitations of the proposed approach, directions for future work, and concluding remarks.

2. Rank statistics and the invariant loss function

2.1 Discrete uniform rank statistics

Let $\tilde{y}_1, \ldots, \tilde{y}_K$ be a random sample from a univariate real distribution with pdf $\tilde{p}$ and let y be a single random sample independently drawn from another distribution with pdf p. We construct the set,

$$\mathcal{A}_K := \Big\{ \tilde{y} \in \{\tilde{y}_k\}_{k=1}^K : \tilde{y} \leq y \Big\},$$

and the rank statistic

$$A_K := |\mathcal{A}_K|, \tag{1}$$

i.e., A_K is the number of elements in $\mathcal{A}_K$. The statistic A_K is a discrete r.v. that takes values in the set $\{0, \ldots, K\}$; and we denote its pmf as $\mathbb{Q}_K : \{0, \ldots, K\} \mapsto [0, 1]$. This pmf satisfies the following key result.

Theorem 1 *If $p = \tilde{p}$ then $\mathbb{Q}_K(n) = \frac{1}{K+1}$ $\forall n \in \{0, \ldots, K\}$, i.e., A_K is a discrete uniform r.v. on the set $\{0, \ldots, K\}$.*

Proof See Elvira et al. (2016) for an explicit proof. This basic result appears under different forms in the literature, e.g., in Rosenblatt (1952), Djuric and Míguez (2010) or Elvira et al. (2021). ■

The following result generalizes Theorem 2 in de Frutos et al. (2024). It establishes the continuity of the rank statistic w.r.t. the L^1 norm.

Theorem 2 *If $\|p - \tilde{p}\|_{L^1(\mathbb{R})} \leq \epsilon$ then,*

$$\frac{1}{K+1} - \epsilon \leq \mathbb{Q}_K(n) \leq \frac{1}{K+1} + \epsilon, \ \forall n \in \{0, \ldots, K\}.$$

Proof See Appendix A. ■

Remark 3 *If p and $\tilde{p}$ have compact support $\mathcal{K} \subset \mathbb{R}$, we can generalize the previous result. By Hölder's inequality (Brézis, 2011, Theorem 4.6), we readily see that*

$$\|p - \tilde{p}\|_{L^1(\mathbb{R})} = \|\mathbb{I}_{\mathcal{K}}\,(p - \tilde{p})\|_{L^1(\mathbb{R})} \leq \|\mathbb{I}_{\mathcal{K}}\|_{L^q(\mathbb{R})} \|p - \tilde{p}\|_{L^{q'}(\mathbb{R})},$$

where $\frac{1}{q} + \frac{1}{q'} = 1$, $q \in [1, \infty]$, $\mathbb{I}_{\mathcal{K}}$ denotes the indicator function on $\mathcal{K}$, and $\|\mathbb{I}_{\mathcal{K}}\|_{L^q(\mathbb{R})} = \mathcal{L}(\mathcal{K})^{1/q}$, with $\mathcal{L}(\mathcal{K})$ being the Lebesgue measure of $\mathcal{K}$. Therefore, if $\|p - \tilde{p}\|_{L^{q'}(\mathbb{R})} \leq \epsilon / \mathcal{L}(\mathcal{K})^{1/q}$, then $\|p - \tilde{p}\|_{L^1(\mathbb{R})} \leq \epsilon$. This implies that Theorem 2 holds for any $L^{q'}$ with $q' \in [1, \infty]$ and not only for L^1, provided that both p and $\tilde{p}$ have compact support.

So far, we have shown that if the pdf of the generative model, $\tilde{p}$, is close to the target pdf, p, then the statistic A_K is close to uniform. A natural question to ask is whether A_K displaying a uniform distribution implies that $\tilde{p} = p$. This result is also established in de Frutos et al. (2024) and is reproduced here for convenience.

Theorem 4 *Let p and $\tilde{p}$ be pdf's of univariate real r.v.'s and let A_K be the rank statistic in Eq 1. If A_K has a discrete uniform distribution on $\{0, \ldots, K\}$ for every $K \in \mathbb{N}$ then $p = \tilde{p}$ almost everywhere (a.e.).*

Remark 5 *If p and $\tilde{p}$ are continuous functions then Theorem 4 implies that $p = \tilde{p}$ (everywhere).*

Figure 2 illustrates how the generator density and the empirical rank-statistic pmf $\mathbb{Q}_K(n)$ evolve over the course of a training process aimed at aligning $\tilde{p}$ with the target distribution p. As optimization proceeds, the generator density increasingly resembles the target, and the rank-statistic distribution transitions from a skewed shape to the uniform distribution expected when $\tilde{p} = p$.

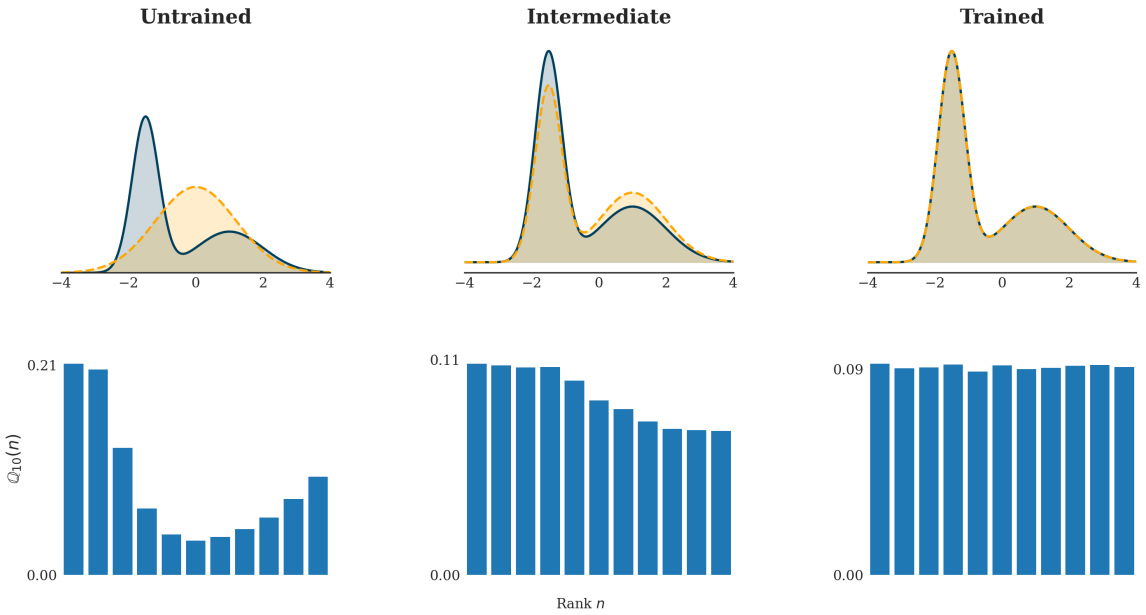


Figure 2: Evolution of generator density and rank-statistic pmf over the course of a training process aimed at aligning $\tilde{p}$ with the target distribution p. Columns (left to right) show the untrained, intermediate, and trained stages. *Top row:* target density (solid blue) versus model density (dashed orange). *Bottom row:* empirical pmf $\mathbb{Q}_K(n)$ of the rank statistic. As training proceeds, the model density aligns with the target and $\mathbb{Q}_K(n)$ converges to the uniform distribution.

2.2 The invariant statistical loss function

In this subsection we build upon previous results to introduce a new discrepancy function $d_K(p, \tilde{p}) : C(\mathcal{K}) \times C(\mathcal{K}) \mapsto [0, +\infty)$ between two continuous densities on a compact set $\mathcal{K} \subset \mathbb{R}$. This function measures the ℓ_1-norm of the difference between the pmf $\mathbb{Q}_K$ associated with the statistic A_K and the uniform pmf, i.e.,

$$d_K(p, \tilde{p}) = \frac{1}{K+1} \left\| \frac{1}{K+1} \mathbf{1}_{K+1} - \mathbb{Q}_K \right\|_{\ell_1} = \frac{1}{K+1} \sum_{n=0}^{K} \left| \frac{1}{K+1} - \mathbb{Q}_K(n) \right|.$$

It is clear that $d_K(p, \tilde{p}) \geq 0$ for any pair of pdf's p and $\tilde{p}$. By Theorems 2 and 4, $d_K(p, \tilde{p}) = 0$ for arbitrarily large K if and only if $p = \tilde{p}$. Thus, $\lim_{K\to\infty} d_K(p, \tilde{p})$ is a probability divergence (Chen et al., 2023; Sugiyama et al., 2013). Furthermore, Theorem 2 shows that d_K is continuous w.r.t. the $L^1(\mathbb{R})$ norm, i.e., whenever $\|p - \tilde{p}\|_{L^1(\mathbb{R})} \leq \epsilon$, one has $d_K(p, \tilde{p}) \leq \epsilon$. This discrepancy measure serves as the theoretical loss function on which we build up a training procedure for implicit generative models.

The following theorem identifies two key regularity properties—continuity and differentiability—of the divergence $d_K(p, \tilde{p}_\theta)$ w.r.t. the network parameters θ of g_θ.

Theorem 6 *Let $p : \mathcal{X} \to [0, \infty)$ be a pdf, where $\mathcal{X} \subseteq \mathbb{R}$. Let Z be a real r.v. taking values in $\mathcal{Z} \subseteq \mathbb{R}$ and choose a function*

$$g : \mathcal{Z} \times \mathbb{R}^d \to \mathcal{X},$$
$$(z, \theta) \to g_\theta(z).$$

Let $\tilde{p}_\theta$ denote the pdf of the r.v. $g_\theta(Z)$. Then,

1. *If g is continuous w.r.t. θ for almost every $z \in \mathcal{Z}$, then $d_K(p, \tilde{p}_\theta)$ is also continuous w.r.t. θ.*

2. *Assume that $g_\theta(z)$ satisfies the Lipschitz condition w.r.t. θ, i.e., $|g_\theta(z) - g_{\theta'}(z)| \leq L(z)\|\theta - \theta'\|$, and there is a constant $L_{\max} < +\infty$ such that $L(z) < L_{\max}$ for almost every $z \in \mathcal{Z}$. If $g_\theta(z)$ is differentiable w.r.t. z and there exists $m > 0$ such that $\inf_{(z,\theta)\in\mathcal{Z}\times\mathbb{R}^d} |g'_\theta(z)| \geq m$, then $d_K(p, \tilde{p}_\theta)$ is Lipschitz continuous w.r.t. θ, and consequently, it is differentiable a.e.*

Proof See Appendix B. ■

Theorem 6 shows that the discrepancy $d_K(p, \tilde{p}_\theta)$, which measures the difference between a fixed density p and a parametric family $\tilde{p}_\theta$ generated by $g_\theta(z)$, is continuous whenever $g_\theta(z)$ is continuous in θ. Additionally, if g_θ is Lipschitz continuous and monotonic, the discrepancy becomes differentiable a.e. However, since the dependence of the empirical distribution $\mathbb{Q}_K$ on θ is unknown, gradient-based methods cannot be directly used to minimize $d_K(p, \tilde{p}_\theta)$ w.r.t. θ.

2.3 The surrogate invariant statistical loss function

Since the divergence $d_K(p, \tilde{p}_\theta)$ cannot be used in practice, we present a surrogate loss function that is tractable, in the sense that it can be optimized w.r.t. the network parameters θ using standard methods. The training dataset consists of a set of N i.i.d. samples, $y_1, \ldots, y_N$, from the true data distribution, p. For each y_n, we generate K i.i.d. samples from the generative model, denoted by $\tilde{\mathbf{y}} = [\tilde{y}_1, \ldots, \tilde{y}_K]^\top$, where each $\tilde{y}_i = g_\theta(z_i)$ with $z_i \sim \mathcal{N}(0, 1)$. From y_n we obtain one sample of the r.v. A_K, that we denote as $a_{K,n}$.

We replace $d_K(p, \tilde{p}_\theta)$ by a differentiable approximation and refer to this surrogate function as invariant statistical loss (ISL) (de Frutos et al., 2024). The ISL mimics the construction of a histogram from the statistics $a_{K,1}, a_{K,2}, \ldots, a_{K,N}$. Given a real data point y_n, we can tally how many of the K simulated samples in $\tilde{\mathbf{y}}$ are less than the n-th observation y_n. Specifically, one computes

$$\tilde{a}_{K,n}(y) = \sum_{i=1}^{K} \sigma_\alpha(y_n - \tilde{y}_i) = \sum_{i=1}^{K} \sigma_\alpha(y_n - g_\theta(z_i)),$$

where $z_i \sim p_z$ is a sample from a univariate distribution, and $\sigma_\alpha(x) := \sigma(\alpha x)$, with $\sigma(x) := 1/\left(1 + \exp(-x)\right)$ being the sigmoid function. As we can see, $\tilde{a}_{K,n}$ is a differentiable (w.r.t. θ) approximation of the actual statistic A_K for the observation y_n. The parameter α enables

us to adjust the slope of the sigmoid function to better approximate the (discrete) 'counting' in the construction of $\tilde{a}_{K,n}$.

A differentiable surrogate histogram is constructed from $\tilde{a}_{K,1}, \ldots, \tilde{a}_{K,n}$ by leveraging a sequence of differentiable functions. These functions are designed to mimic the bins around $k \in \{0, \ldots, K\}$, replacing sharp bin edges with functions that replicate bin values at k and smoothly decay outside the neighbourhood of k. In our particular case, we consider radial basis function (RBF) kernels $\{\psi_k\}_{k=0}^K$ centred at $k \in \{0, \ldots, K\}$ with length-scale ν^2, i.e., $\psi_k(a) = \exp(-(a-k)^2/2\nu^2)$. Thus, the approximate normalized histogram count at bin k is given by

$$q[k] = \frac{1}{N} \sum_{i=1}^{N} \psi_k(\tilde{a}_{K,i}(y_i)), \tag{2}$$

for $k = 0, \ldots, K$. The ISL is computed as the ℓ-norm distance between the uniform vector $\frac{1}{K+1}\mathbf{1}_{K+1}$ and the vector of empirical probabilities $\mathbf{q} = [q[0], q[1], \ldots, q[K]]^\top$, namely,

$$\mathcal{L}_{ISL}(\theta, K) := \left\| \frac{1}{K+1}\mathbf{1}_{K+1} - \mathbf{q} \right\|_{\ell_1}. \tag{3}$$

Remark 7 *The ISL is a sum and composition of $C^\infty(\mathbb{R})$ functions. It is smooth w.r.t. the fictitious samples $\{\tilde{y}_i\}_{i=1}^K$ and the data y_n. As a result, its regularity aligns with that of the neural network (as a function of both its parameters and the input noise).*

The hyperparameters for the ISL method include the number of samples K, which is tunable, the activation function σ_α, and the set of basis functions $\{\psi_k\}_{k=0}^K$, specified as radial basis function (RBF) kernels with a length scale of ν^2. These parameters control the flexibility and behavior of the model during learning.

To better illustrate the transition from discrete to differentiable rank-based histograms, Figure 3 provides a visual comparison. In Section C.1, we present a numerical study assessing how accurately the surrogate loss $\mathcal{L}_{\text{ISL}}$ approximates the true divergence $d_K(p, \tilde{p}_\theta)$. Finally, Figure 4 outlines the full 1D ISL training pipeline, summarizing the steps involved in computing and optimizing the surrogate loss.

2.4 Role and practical setting of the order K

Role of K (resolution, approximation quality, and cost) Recall that the ISL surrogate is built from rank statistics $A_K \in \{0, \ldots, K\}$, where K is the number of *fictitious* samples used to form the rank (equivalently, the number of bins of the rank histogram is $K+1$). Thus, K controls the *resolution* of the rank-based comparison: a larger K yields a more discriminative test and a finer approximation of the limiting divergence as $K \to \infty$, at the expense of increased computation. Concretely, for a batch of size N, computing the soft ranks and the associated histogram-based discrepancy scales as $O(NK)$ (and as $O(mNK)$ for the sliced variant with m projections), so runtime grows approximately linearly with K.

Practical setting of K: bias–variance trade-off While increasing K reduces discretisation bias, it also increases the effective dimension of the histogram and can raise the finite-sample variance when N is not large enough to populate the $K+1$ bins reliably.

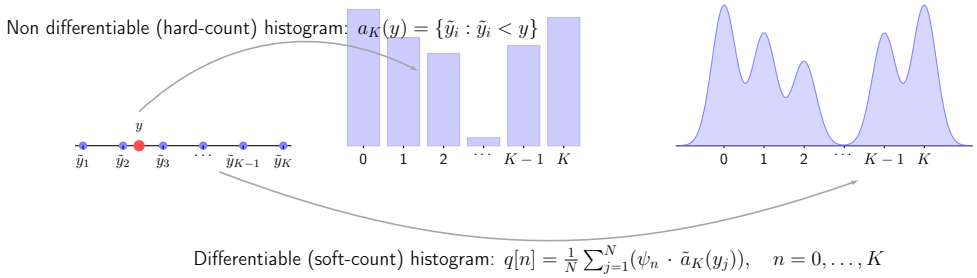


Figure 3: Hard- vs. Soft-Count Histograms via RBF-Binning. *Left:* Sorted model samples $\tilde{y}_1 \leq \cdots \leq \tilde{y}_K$, with the observed value y highlighted in red. *Middle:* Standard hard-count histogram: each bin i tallies the number of model samples below y, i.e. $a_K(y) = \big|\{\tilde{y}_j : \tilde{y}_j < y\}\big|$, yielding discrete, integer counts per bin. *Right:* Differentiable soft-count histogram obtained by RBF-binning, where each $\tilde{y}_j$ contributes fractionally to every bin according to $q[n] = \frac{1}{N}\sum_{j=1}^{N}\psi_n(\tilde{y}_j, y)$ for each $n \in \{0, \ldots, K\}$, giving a smooth, continuous pmf.

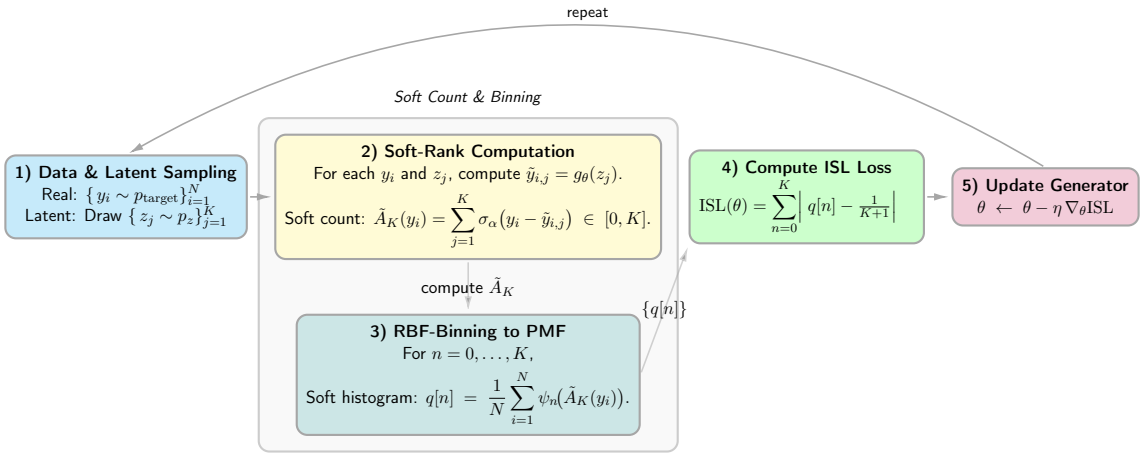


Figure 4: 1D ISL Training Pipeline. First, real data samples $y_i \sim p_{\text{target}}$ and latent inputs $z_j \sim p_z$ are drawn (Step 1). Next, for each pair (y_i, z_j), the network output $\tilde{y}_{i,j} = g_\theta(z_j)$ is compared to y_i via a smooth indicator function to produce *soft counts* $\tilde{A}_K(x_i) \in [0, K]$ (Step 2). These counts are then converted into a *continuous pseudo-PMF* $q[n]$ over ranks $n = 0, \ldots, K$ using RBF kernels (Step 3). The ISL loss is computed as the L^1-distance between $q[n]$ and the uniform distribution (Step 4). Finally, θ is updated by gradient descent on this loss (Step 5), and the process repeats. This pipeline yields a fully differentiable surrogate for the invariant statistical loss suitable for end-to-end training.

Appendix C.2 provides a systematic empirical evaluation of this trade-off. In particular, we observe (i) a consistent decay of the approximation error $|d_K - d_{K_{\text{ref}}}|$ with K, where K_{ref} denotes a large reference resolution used as a proxy for the $K \to \infty$ limit (Figure 14), (ii) rapid stabilization of the *gradient direction* already at moderate K (e.g., cosine similarity typically above 0.98 for $K \geq 32$ in Table 13), and (iii) a characteristic U-shaped finite-sample MSE curve in K (Figure 16), where MSE denotes the mean squared error of d_K relative to the high-resolution reference estimate $d_{K_{\text{ref}}}$, reflecting the bias–variance balance. These results support using *moderate* values of K as a robust default (e.g., $K \in [10, 64]$ for typical

Algorithm 1 Progressively training by increasing K

Input: generator g_θ; training data $\{y_i\}_{i=1}^M$; batch size N; number of epochs T; schedule $\{K^{(i)}\}_{i=1}^I$ with $K^{(I)} = K_{\max}$; test level α; evaluation period Δ.
Output: trained generator g_θ.
Initialize stage $i \leftarrow 1$ and $K \leftarrow K^{(1)}$.
for $t = 1, \dots, T$ **do**
  Sample minibatch $\{y_j\}_{j=1}^N$ and latent noise; form model samples $\{x_j\}_{j=1}^N$.
  Update θ by minimizing $\mathcal{L}_{\text{ISL}}(\theta; K)$ on the minibatch.
  if $t \bmod \Delta = 0$ **then**
    Compute rank statistics $\{a_{K,1}, \dots, a_{K,N}\}$ for the current minibatch (or a held-out minibatch).
    Perform Pearson χ^2 test of uniformity of A_K against $\mathbb{Q}_K$ at level α.
    if test accepts uniformity **and** $i < I$ **then**
      Set $i \leftarrow i + 1$ and $K \leftarrow K^{(i)}$.
    end if
  end if
end for
return g_θ.

batch sizes N in the 10^2–10^3 range), with diminishing returns for substantially larger K unless N is increased accordingly.

Progressive training (small-to-large K) The above considerations motivate a simple efficiency strategy: train in stages with increasing values of K (see de Frutos et al. (2024)). Specifically, choose a schedule $K^{(1)} < K^{(2)} < \cdots < K^{(I)}$, with $K^{(I)} = K_{\max}$. Early in training, when the generator is still far from the data distribution, a coarse surrogate (small K) provides sufficiently informative gradients at low cost. As training progresses, increasing K refines the surrogate only when needed, reducing overall compute while retaining the benefits of higher resolution near convergence. We use a simple criterion based on a Pearson χ^2 uniformity test on the rank histogram to decide when to move to the next stage: if the null hypothesis "A_K is uniform" is accepted and $K < K_{\max}$, we increase K. The resulting scheme is summarized in Algorithm 1. Empirically, Appendix C.3 shows that this progressive procedure can yield substantial wall-clock savings (often close to 50% in our 1D benchmarks) with comparable final performance.

3. Pareto-ISL

As shown by de Frutos et al. (2024), ISL outperforms other generative methods in learning the central regions of typical 1D distributions. However, Figure 5 indicates that when standard Gaussian noise is used as an input, NNs struggle to capture the tails of Cauchy mixtures, since compactly supported inputs cannot produce unbounded pdfs (which we will refer to as unbounded distributions). This issue can be addressed by using input noise from

a generalized Pareto distribution (GPD). In this section, we introduce Pareto-ISL, which utilizes a GPD for input noise, and demonstrate its effectiveness in learning heavy-tailed distributions.

3.1 Motivation and alternative approaches

Standard implicit generative models typically adopt a light–tailed latent prior, most often a standard Gaussian. This choice is computationally convenient and empirically effective on many benchmarks, but it can hinder learning when the target distribution exhibits substantial tail mass. Intuitively, if the latent variable is sub–Gaussian, then extreme latent draws are exponentially rare; matching rare, high-magnitude observations then requires the generator to "stretch" typical latent values into the far tails. In practice, this often yields poor tail fidelity even when the central region is well captured, since the training signal coming from extremes is both scarce and noisy.

A first line of work replaces the Gaussian latent with an explicitly heavy–tailed prior, such as Student's t or Cauchy distributions, or mixtures thereof (Sun et al., 2020; Zhang and Zhou, 2021). These alternatives increase the probability of generating large–magnitude samples and can improve coverage of extreme regions. However, they usually introduce tail parameters that are fixed or selected by hand (e.g., the degrees of freedom ν in a Student's t), effectively hard–wiring the tail behavior into the model. When the chosen tail index does not match the data, the generator must still compensate through complex warping, so the burden of tail adaptation is shifted, rather than removed, from the modeling pipeline.

A second family of approaches alters the training objective or architecture to explicitly emphasize extremes. For example, tail–adaptive losses can upweight rare, high–magnitude samples during training (Bhatia et al., 2021). While such reweighting can substantially improve tail coverage, these methods are often tied to adversarial objectives and additional heuristic design choices. In heavy–tailed regimes, where extreme observations are intrinsically scarce, these objectives can amplify gradient variance and exacerbate the already delicate stability of adversarial training.

A related line of work models heavy tails by decoupling marginal behavior from dependence (e.g., copula constructions), or by learning an explicit transformation from a light–tailed base distribution (e.g., normalizing flows) (Wiese et al., 2019; Papamakarios et al., 2021). In principle, these approaches can accommodate heavy–tailed marginals while retaining rich multivariate structure. However, they are typically implemented in a likelihood-based setting and rely on tractable density evaluation and/or invertible architectures, together with additional modeling components (marginal fits and dependence/transport parameterizations) and tuning. This added complexity is less aligned with our goal of keeping training fully sample-based and directly compatible with standard implicit pipelines.

In contrast, we use a generalized Pareto distribution (GPD) for the latent noise, motivated by both theoretical universality and practical simplicity. By the Pickands–Balkema–de Haan theorem (Balkema and De Haan, 1974; Coles et al., 2001), for a broad class of underlying distributions the law of exceedances above a high threshold converges to a GPD, making it a canonical parametric model for tail behavior. Crucially, the GPD is parameterized by a tail index ξ (and scale σ), which we treat as learnable rather than fixed. This decoupling lets the network parameters focus on modeling global geometry and fine–scale structure, while (ξ, σ)

adaptively control tail heaviness in a data–driven manner. Beyond tail expressiveness, this choice yields a simple and stable integration into implicit training: sampling from a GPD is straightforward, adds no auxiliary networks, and does not require density evaluation or invertible architectures. Unlike fixed heavy–tailed priors, the tail index is learned rather than selected by hand, reducing sensitivity to misspecification. Unlike tail–reweighted adversarial objectives, tail control is achieved at the level of the latent distribution, avoiding additional loss engineering and mitigating gradient variance induced by rare extremes.

3.2 Tail distributions and extreme value theory

The conditional excess distribution function $F_u(y)$ provides a key tool for analyzing the tail of a distribution by focusing on exceedances over a specified threshold u. By conditioning on large values, it isolates the behavior of the distribution in its tail, where extreme or rare events are more likely to occur. In the context of extreme value theory, for sufficiently high thresholds u, the conditional excess distribution function converges to the GPD. The GPD, parameterized by the tail index ξ and scaling parameter σ, provides a flexible model for the tail, allowing us to characterize its heaviness and the probability of extreme values.

The following definitions are taken from Huster et al. (2021) and provided here for convenience.

Definition 8 *The conditional excess distribution function with threshold $u \in \mathbb{R}$ is defined as*

$$F_u(y) = \mathbb{P}(X - u \leq y | X > u) = \frac{F(u+y) - F(u)}{1 - F(u)}.$$

Definition 9 *The GPD, parametrized by tail index $\xi \in \mathbb{R}$ and scaling parameter $\sigma \in \mathbb{R}^+$, has the following complementary cumulative distribution function (CCDF)*

$$S(z; \xi, \sigma) = \begin{cases} (1 + \xi z/\sigma)^{-1/\xi}, & \text{for } \xi \neq 0, \\ e^{-z/\sigma}, & \text{for } \xi = 0. \end{cases}$$

Lipschitz continuous functions map bounded distributions to bounded ones (Evans, 2018, Chapter 3), limiting the ability of NNs to model heavy-tailed distributions. To address this, unbounded input distributions are required.

To construct unbounded NN generators, recall that piecewise linear (PWL) functions, (which include operations like rectified linear unit (ReLU), leaky ReLU, linear layers, addition, and batch normalization), are closed under composition (Arora et al., 2016, Theorem 2.1) and are unbounded, making them ideal for constructing generators that approximate heavy-tailed distributions.

We then define a Pareto-ISL generator, g^{PWL}, as a piecewise linear generator driven by an input from a GPD with tail index ξ, and trained using ISL. Estimators such as Hill's (Resnick and Stărică, 1997) can be used to estimate ξ, aiding in the accurate modeling of heavy-tailed behavior.

Definition 10 (Pareto-ISL) *Let $z_\xi = (U^{-\xi} - 1)/\xi$, where $U \sim \mathcal{U}(0,1)$, be a GPD r.v. with tail index ξ and CCDF $S(x; \xi, 1)$. A Pareto-ISL generator is g_θ^{PWL}, with an input distribution z_ξ parameterized by ξ, and output distribution $y_\xi = g_\theta^{PWL}(z_\xi)$.*

The Pickands-Balkema-de Haan theorem (Balkema and De Haan, 1974) states that for a wide range of probability distributions, the conditional excess distribution function converges to the GPD as the threshold u increases. This applies to distributions like Gaussian, Laplacian, Cauchy, Lévy, Student-t, and Pareto. Building on this result and (Huster et al., 2021, Theorem 2), a generator constructed using g^{PWL} with a GPD input can effectively approximate the conditional excess distribution of heavy-tailed distributions. Specifically, if y_ξ has unbounded support, the conditional excess distribution $F_u(y)$ of y_ξ converges to $S(y; \xi, \sigma)$ for some $\sigma \in \mathbb{R}^+$. This indicates that ISL-Pareto is especially well-suited for these types of problems, outperforming other implicit methods, including ISL with non-Pareto noise, as we demonstrate in the following subsection.

3.3 Comparison of Pareto-ISL and standard ISL in learning a Cauchy mixture

In Figure 5, we compare Pareto-ISL against standard ISL where the data distribution is a two-component Cauchy mixture with locations at -1.0 and 1.0, and scales 0.7 and 0.85. All generators use a four-layer multilayer perceptron (MLP) with 35 units per layer and ReLU activation. Generators are trained with ISL using $K = 20$, $N = 1000$, and a learning rate of 10^{-3}. For Pareto-ISL, the tail parameter is set to $\xi = 1$, aligning GPD noise with the tail index of the Cauchy mixture. Introducing GPD noise improves tail approximation and enhances the modeling of the central part of the target distribution, as demonstrated in the logarithmic-scale (bottom row) and linear-scale (top row) of Figure 5.

In Appendix C.6, we present a multidimensional heavy-tailed distribution and compare Pareto-ISL to ISL with Gaussian noise (results shown in Figure 21).

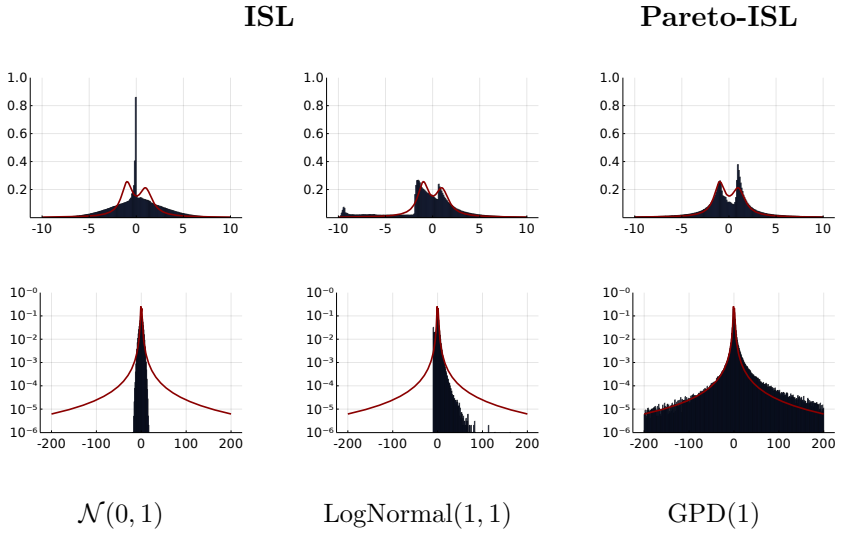


Figure 5: Pdfs approximated by generators with various tail behaviors. The noise input for each subplot is given by the subcaption of the corresponding column. Generators are trained on a mixture of Cauchy distributions, with the true density displayed in red. The top row presents the central region of the distributions on a linear scale, while the bottom row highlights the tails on a logarithmic scale.

3.4 Performance evaluation of Pareto-ISL compared to other implicit generative models

In a second experiment, we evaluate the performance of Pareto-ISL as compared to different GANs. For this comparison, we consider four data distributions, including a Cauchy distribution with location parameter 1 and scale parameter 2 (labelled Cauchy$(1,2)$), a Pareto distribution with scale parameter 1 and shape parameter 1 (labelled Pareto$(1,1)$) and two mixture distributions, labelled Model$_3$ and Model$_4$. Model$_3$ is a mixture of a $\mathcal{N}(-5,2)$ distribution and a Pareto$(5,1)$ distribution, while Model$_4$ is a mixture of a Cauchy$(-1,0.7)$ distribution and a Cauchy$(1,0.85)$ distribution.

As the generator network, we use a 4-layer MLP with $7, 13, 7$ and 1 units at the corresponding layers. As activation function we use a ReLU. We train each setting up to 10^3 epochs with 10^{-2} learning rate using Adam. We compare Pareto-ISL, GAN, Wasserstein GAN (WGAN) from (Arjovsky et al., 2017), and maximum mean discrepancy GAN (MMD-GAN) proposed in Li et al. (2017) using KSD (Kolmogorov-Smirnov distance), MAE (mean absolute error), and MSE (mean squared error) error metrics, defined as follows

$$\text{KSD} = \sup_{x\in\mathbb{R}} \left|F(x) - \tilde{F}(x)\right|, \quad \text{MAE} = \int_{\mathbb{R}} \left|g(z) - g_\theta(z)\right| p_z(z)\,\mathrm{d}z, \quad \text{MSE} = \int_{\mathbb{R}} \left(g(z) - g_\theta(z)\right)^2 p_z(z)\,\mathrm{d}z.$$

where F and $\tilde{F}$ represent, respectively, the cdfs of the data distribution and the r.v. generated by the neural network g_θ with input $z \sim p_z$. Moreover, g denotes the optimal transformation of the data distribution. The results are detailed in Table 1.

Target	**Pareto-ISL**			**GAN**			**WGAN**			**MMD-GAN**		
	KSD	MAE	MSE	KSD	MAE	MSE	KSD	MAE	MSE	KSD	MAE	MSE
Cauchy(1,2)	**1.90e-3**	**1.42**	**15.78**	0.08	8.96	8207.00	0.03	10.57	8127.00	0.03	9.68	57 975.00
Pareto(1,1)	**5.30e-3**	**1.16**	**2.41**	0.10	12.64	114 970.00	0.49	7.64	7062.00	0.50	9.02	10 674.00
Model$_3$	**0.02**	0.47	**0.33**	0.19	**0.45**	1.61	0.30	3.04	13.13	0.56	3.20	21.52
Model$_4$	**0.01**	**0.61**	**1.05**	0.02	0.77	2.75	0.03	2.63	66.06	0.02	0.66	2.60

Table 1: Comparison of Pareto-ISL results with vanilla GAN, WGAN, and MMD-GAN when input noise is a standard Gaussian, $K_{\max} = 10$, epochs=1000, and $N = 1000$. The best result for each metric is highlighted in bold.

3.5 Assessment of Pareto-ISL on heavy-tailed datasets

In this third experiment, we demonstrate the effectiveness of the Pareto-ISL scheme on two univariate heavy-tailed datasets:

- 136 million keystrokes (`Keystrokes`): this dataset includes inter-arrival times between keystrokes for a variety of users.
- Wikipedia web traffic (`Wiki Traffic`): this dataset includes the daily number of views of Wikipedia articles during 2015 and 2016.

Our assessment uses two metrics. First, we compute the KSD to compare the data distribution and the generated distribution. Then, we calculate the area A_{CCDF} between the log-log plots of the CCDFs of data and generated samples, indicating how well the tails of the

distributions match. For n real samples, we have

$$\mathrm{A}_{CCDF} = \sum_{i=1}^{n} \left[\log\left(F_p^{-1}\left(\frac{i}{n}\right)\right) - \log\left(\tilde{F}_{\tilde{p}}^{-1}\left(\frac{i}{n}\right)\right)\right] \log\left(\frac{i+1}{i}\right),$$

where F_p^{-1} and $\tilde{F}_{\tilde{p}}^{-1}$ are the inverse empirical CCDFs for the data distribution p and the generated distribution $\tilde{p}$, respectively.

We use a common network architecture and training procedure for all experiments. The network consists of 4 fully connected layers with 32 hidden units per layer and ReLU activations. Results are shown in Table 2.

Method	**Keystrokes**		**Wiki Traffic**	
	KS	A_{CCDF}	KS	A_{CCDF}
Uniform (ISL)	0.087	6.20	0.025	10.30
Normal (ISL)	0.090	2.70	0.023	8.60
Lognormal (ISL)	0.096	1.70	0.019	9.50
Pareto GAN (Huster et al., 2021)	0.013	21.10	0.017	4.50
Gamma–Weibull KDE (Markovich, 2016)	0.050	1.70	0.075	1.50
Pareto-ISL	**0.006**	**1.4**	**0.017**	**1.19**

Table 2: Pareto-ISL outperforms ISL variants (Uniform, Normal, Lognormal), Pareto GAN (Huster et al., 2021), and Gamma–Weibull KDE (Markovich, 2016) in tail estimation (A_{CCDF}), while also achieving lower KS distance.

4. ISL-slicing: A random projections-based approach

Machine learning datasets are often multi-dimensional. Building on Bonneel et al. (2015) and Kolouri et al. (2019), we extend the one-dimensional loss function $\mathcal{L}_{ISL}$ to a general metric for higher dimensions. We do this by randomly projecting high-dimensional data onto various 1D subspaces, specifically in all possible directions $s \in \mathbb{S}^d$, where $\mathbb{S}^d$ is the unit hypersphere in $d+1$-dimensional space.

Let x be a $(d+1)$-dimensional r.v. and let $s \in \mathbb{R}^{d+1}$ be a deterministic vector. We denote by $s\#p$ the pdf of the real r.v. $y = s^\top x$. Using this notation we define the *sliced ISL distance* between distributions with pdfs p and $\tilde{p}$ as

$$d_K^{\mathbb{S}^d}(p,\tilde{p}) := \int_{s\in\mathbb{S}^d} d_K^s(p,\tilde{p})\, ds,$$

where $d_K^s(p,\tilde{p}) = d_K(s\#p, s\#\tilde{p})$.

Since the expectation in the definition of the sliced ISL distance is computationally intractable, we approximate it using Monte Carlo sampling. Specifically, we choose a pdf q on $\mathbb{S}^d$ and sample directions $s_i \sim q$, for $i = 1,\dots,m$. Then, the Monte Carlo approximation of the sliced ISL distance is

$$\tilde{d}_K^{\mathbb{S}^d}(p,\tilde{p}) = \frac{1}{m}\sum_{i=1}^{m} d_K(s_i\#p, s_i\#\tilde{p}). \tag{4}$$

If $\tilde{p} \equiv \tilde{p}_\theta$ is the pdf of the r.v. $y = g_\theta(z)$, $z \sim p_z$, i.e., the output of a NN with random input z and parameters θ, then one can use $\tilde{d}_K^{\mathbb{S}^d}(p, \tilde{p}_\theta)$ as a loss function to train the NN.

Remark 11 *In practice, we use the surrogate loss (see Section 2.3) to approximate $d_K(s_i\#p, s_i\#\tilde{p}_\theta)$ in Eq. 4, and sample random vectors from the unit sphere. Randomly chosen vectors from the unit sphere in a high-dimensional space are typically almost orthogonal. More precisely, (see Gorban and Tyukin (2018))*

$$\mathbb{P}\left(\left|\frac{v_1^\top v_0}{\|v_1\|_2\|v_0\|_2}\right| < \epsilon\right) > 1 - 2\,e^{-\frac{1}{2}d\epsilon^2},$$

where v_0 and v_1 are uniformly distributed random vectors, and ϵ is a small positive constant.

The pseudocode for training implicit models using the proposed random projection method, referred to as the *ISL-slicing algorithm*, is provided in Algorithm 2.

Algorithm 2 ISL-slicing algorithm

Input Neural network g_θ; hyperparameter K; number of epochs N; batch size M; training data $\{y_i\}_{i=1}^N$; number of randomly chosen projections m.
Output Trained neural network g_θ.
For $t = 1, \dots,$ epochs **do**
  For iteration $= 1, \dots, N/M$ **do**
    $L = 0$
    Sample uniformly distributed random projection directions $\hat{\mathbb{S}}^d = \{s_{1:m}\}$
    Select M samples from $\{y_j\}_{j=1}^N$ at random
    For each $s \in \hat{\mathbb{S}}^d$ **do**
      $\{z_i\}_{i=1}^K \sim \mathcal{N}(\mathbf{0}, I)$
      $\mathbf{q} = \frac{1}{M}\sum_{j=1}^M \psi_k\Big(\sum_{i=1}^K \sigma_\alpha(s^\top y_j - s^\top g_\theta(z_i))\Big)$
      $\mathrm{L} \leftarrow \mathrm{L} + \nabla_\theta \left|\left| \frac{1}{K+1}\mathbf{1}_{K+1} - \mathbf{q}\right|\right|_2$
    Backpropagation$(g_\theta, \nabla_\theta \mathrm{L}\,/m)$
return g_θ

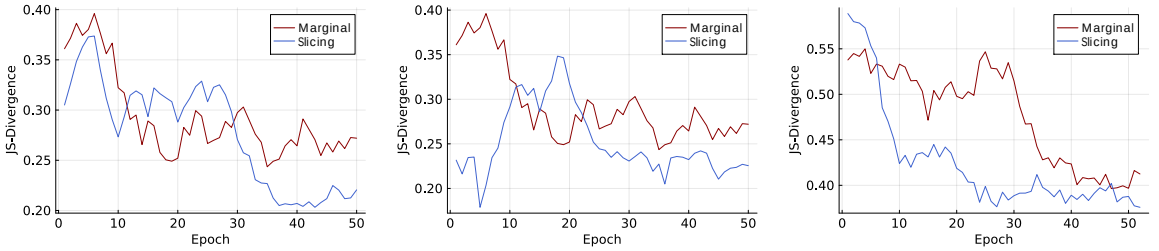
4.1 Random projections vs. marginals on high-dimensional data

Following Saatci and Wilson (2017), we conduct experiments on a multi-modal synthetic dataset. Specifically, we generate D-dimensional synthetic data from the model

$$\begin{cases} \boldsymbol{z} \sim \mathcal{N}(0, 10 \cdot I_d), \quad \boldsymbol{A} \sim \mathcal{N}(0, I_{D\times d}), \quad \epsilon \sim \mathcal{N}(0, 0.01 \cdot I_D), \\ \boldsymbol{x} = \boldsymbol{A}\boldsymbol{z} + \epsilon, \quad d \ll D. \end{cases}$$

In these experiments, we fit a standard GAN to a dataset where $D = 100$ and $d = 2$. The generator is a 3-layer fully connected neural network with 10, 1000, and 100 units, respectively, using ReLU activations throughout.

Figure 6 illustrates the performance of random projections (ISL-slicing) when compared with that of the marginal pdf-based method from de Frutos et al. (2024). Our results show that even with a limited number of projections, ISL-slicing achieves a lower Jensen-Shannon divergence (`JS-divergence`) w.r.t. the true distribution, as estimated by kernel density. This approach also significantly reduces computation time per iteration compared to ISL marginals, since the number of projections is a fraction of the total marginals. Specifically, it took 10x and 5x less execution time, respectively, to obtain the results shown in Figures 6a and 6b using the slicing method compared to the marginals method.



(a) $m = 10$ random projections. (b) $m = 20$ random projections. (c) $D = 500$, $m = 10$ random projections.

Figure 6: Performance evaluation of ISL-slicing vs ISL marginal methods on a synthetic dataset. Plots (a) and (b) correspond to $D = 100$ with different random projections ($m = 10$ and $m = 20$). Plot (c) corresponds to $D = 500$ and $m = 10$. The hyperparameters are $K = 10$, $N = 1000$ and learning rate of 10^{-4}.

4.2 Experiments on 2D distributions

We begin by examining simple 2D distributions characterized by different topological structures: one distribution with two modes, another with eight modes, and a third featuring two rings. Our objective is to assess the ability of the ISL-slicing method to fully capture the support of these distributions. We compare our approach to normalizing flows and GANs, using KL-divergence and visual assessment as metrics.

For GAN, WGAN, and ISL, we use a 4-layer MLP generator with a 2D input sampled from a standard normal distribution. Each layer has 32 units and uses the hyperbolic tangent activation. The discriminator is an MLP with 128 units per layer, using ReLU activations except for the final sigmoid layer. We used a batch size of 1000 and optimized the critic-to-generator update ratio over $\{1{:}1, 2{:}1, 3{:}1, 4{:}1, 5{:}1\}$ for GAN and WGAN. The learning rate was chosen from $\{10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}\}$. For ISL, we set $K = 10$, $N = 1000$, $m = 10$ random projections, and a learning rate of 10^{-3}. For the normalizing flow model, we used the RealNVP architecture from Dinh et al. (2016), with 4 layers of affine coupling blocks, parameterized by MLPs with two hidden layers of 32 units each. The learning rate was set to $5 \cdot 10^{-5}$, using the implementation from Stimper et al. (2023). All methods were trained for 1000 epochs, with optimization performed using the ADAM algorithm.

Figure 7 highlights the challenges of training GANs, particularly their susceptibility to mode collapse. In contrast, normalizing flow methods preserve topology via invertibility constraints but struggle to model complex structures, often forming a single connected component due to density filaments. Our method overcomes this by capturing the full

distribution support and distinguishing between connected components, as seen in the `Dual Moon` example. However, ISL can fill regions between modes, as seen in the `Circle of Gaussians`, an issue mitigated by increasing K. Alternatively, combining a pretrained network with ISL (trained for 100 epochs) and a GAN yielded the best results (method denoted as ISL+GAN), capturing full support while excluding zero-density regions. This approach is detailed further in Section 4.3. We also estimate the KL-divergences between the target and model distributions, as listed in Table 3. In all cases, the ISL method, and particularly the ISL+GAN approach, outperform the respective baselines.

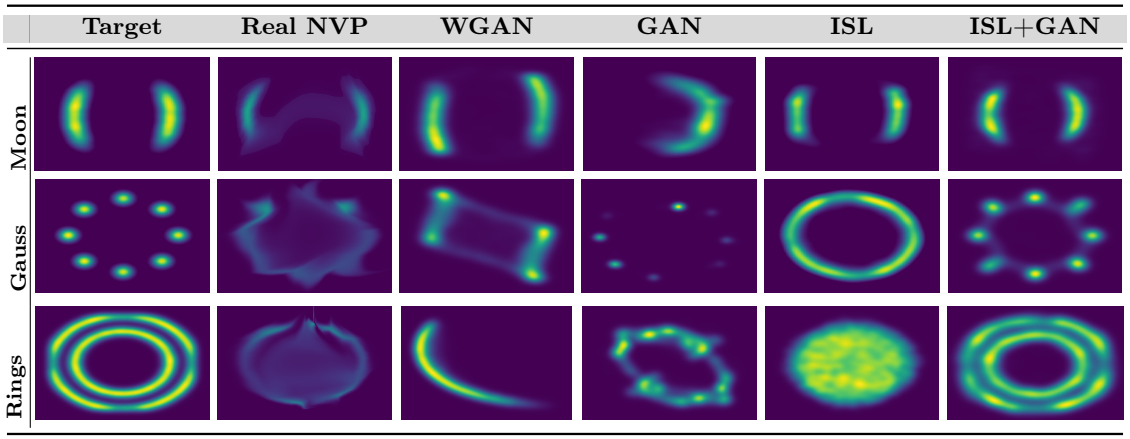


Figure 7: Comparison of the performance of various generative methods—Real NVP, WGAN, GAN, ISL-slicing, and GAN pretrained with ISL to approximate three complex 2D distributions: the `Moon`, `Gauss`, and `Rings` datasets— referred to as Dual Moon, Circle Gaussian, and Two Rings, respectively. For a detailed description of the dataset, refer to Stimper et al. (2023). The first column shows the target distributions, while the subsequent columns display the results generated by Real NVP, WGAN, GAN, ISL-slicing, and GAN pretrained with ISL, in that order.

Method / Dataset	Real NVP	GAN	WGAN	ISL	ISL+GAN
Dual moon	1.77	1.23	1.02	0.43	**0.30**
Circle of Gaussians	2.59	2.24	2.38	1.61	**0.46**
Two rings	2.69	1.46	2.74	0.56	**0.38**

Table 3: Performance comparison of generative methods on 2D distributions. The table summarizes KL-divergence results for Real NVP, GAN, WGAN, and ISL on `Dual Moon`, `Circle of Gaussians`, and `Two Rings`. Lower values indicate better performance.

4.3 ISL-pretrained GANs for robust mode coverage in 2D grids

In this experiment, we explore how ISL-pretrained GANs improve mode coverage on the `2D-Ring` and `2D-Grid` datasets, benchmarks commonly used in generative model evaluation. The `2D-Ring` dataset consists of eight Gaussian distributions arranged in a circle, while the `2D-Grid` dataset has twenty-five Gaussians on a grid. We first pretrain with ISL-slicing to ensure comprehensive mode coverage, then fine-tune with a GAN. Performance was compared

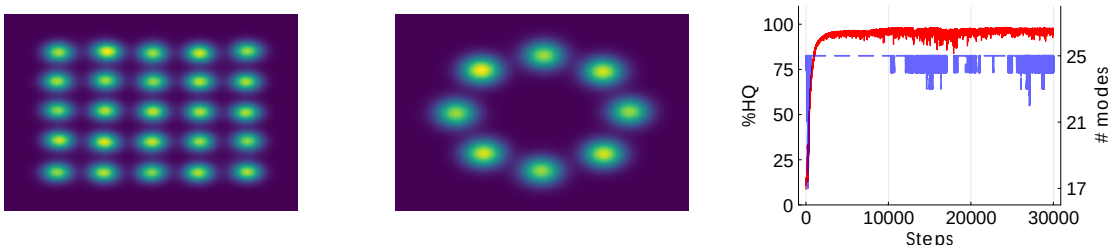
to other GANs using two metrics: the number of covered modes (`#modes`) and the percentage of high-quality samples (`%HQ`). A mode is covered if generated samples fall within three standard deviations of the Gaussian centre, which also defines high-quality samples. The second metric represents their proportion among all generated samples.

We follow the experimental settings for GAN and WGAN as outlined by Luo and Yang (2024). The generator is a 4-layer MLP with 128 units per layer and ReLU activation, while the discriminator is an MLP with 128 units, using ReLU except for a sigmoid in the final layer. We use a batch size of 128, a 1:1 critic-to-generator update ratio, and learning rates of 10^{-4} for GAN and 10^{-5} for WGAN, both optimized using ADAM. ISL was run for 250 epochs with $K = 10$, $N = 1000$, and $m = 5$ random projections, capturing the full support of the distribution. The GAN is then trained for 30000 epochs. The results presented in Table 4 and Figure 8 compare our method with other techniques for addressing mode collapse in GANs. They demonstrate that our approach performs on par with more advanced methods, such as DynGAN (a semi-supervised technique, see Luo and Yang (2024)) and BourGAN (computationally intensive, see Xiao et al. (2018)), while offering greater simplicity.

Finally, we conduct a robustness experiment comparing the performance of a WGAN with suboptimal hyperparameters to the same WGAN after ISL pretraining. Both use the same generator and discriminator architectures described in Section 4.2. The WGAN hyperparameters include a batch size of 1000, a critic-to-generator update ratio optimized over $\{1:1, 2:1, 3:1, 4:1, 5:1\}$, and a learning rate of 10^{-5}. For ISL, we use $K = 10$, $N = 1000$, $m = 10$ random projections, and a learning rate of 10^{-3}. Numerical results are displayed in Table 5. Additional experiments with batch sizes $\{64, 128, 252, 512\}$ and optimized critic-to-generator ratios are shown in Table 6. ISL-pretrained WGAN consistently outperforms standard WGAN across all ratios and batch sizes, detecting all 8 modes and producing high-quality samples. This shows that ISL pretraining significantly enhances stability and performance, leading to more robust GAN training. Once all modes are captured by ISL, a low learning rate allows the generator to improve high-quality metrics while maintaining full distribution coverage.

Method	**2D-Ring**		**2D-Grid**	
	#modes	%HQ	#modes	%HQ
GAN (Goodfellow et al., 2014)	6.3	98.2	17.1	92.5
VEEGAN (Srivastava et al., 2017)	**8.0**	86.8	24.4	74.2
Pointwise (Zhong et al., 2019)	**8.0**	87.5	**25.0**	76.7
BourGAN (Xiao et al., 2018)	**8.0**	**99.9**	**25.0**	94.9
WGAN (Arjovsky et al., 2017)	7.7	86.4	24.8	83.7
DynGAN (Luo and Yang, 2024)	**8.0**	99.5	**25.0**	96.0
ISL + GAN	**8.0**	97.9	24.6	**96.8**
ISL + WGAN	**8.0**	97.4	**25.0**	96.0

Table 4: Quantitative results on synthetic 2D-mode benchmarks. #modes is the number of modes covered; %HQ is the percentage of high-quality samples (higher is better).



(a) ISL + GAN, 25 modes dataset (b) ISL + GAN, 8 modes dataset (c) 25 modes dataset, ISL + GAN

Figure 8: ISL + GAN on 25 and 8 modes datasets (plots (a) and (b), respectively) with $K = 10$, $N = 1000$, ISL learning rate of 10^{-3} for 250 epochs, and GAN with a batch size of 128, 1:1 critic-generator ratio, and learning rate of 10^{-4} for 30000 epochs. Plot (c) shows the coverage of the `2D-Grid` dataset, highlighting the number of modes covered (`#modes`, blue) and high-quality samples (`%HQ`, red) during GAN training after ISL pretraining.

Critics	**WGAN**			**ISL+WGAN**		
	#modes	%HQ	KL div	#modes	%HQ	KL div
1:1	3.70	40.99	33.99	**8.0**	**97.74**	**0.51**
1:2	7.00	0.07	59.58	**8.0**	**98.43**	**0.47**
1:3	8.00	0.63	57.05	**8.0**	**97.39**	**0.55**
1:4	8.00	0.44	59.18	**8.0**	**98.07**	**0.78**
1:5	8.00	0.19	56.83	**8.0**	**98.24**	**0.72**

Table 5: Comparison of WGAN and ISL + WGAN on the `2D-Ring` dataset.

Batch Size	**WGAN**			**ISL+WGAN**		
	#modes	%HQ	KL div	#modes	%HQ	KL div
64	6.10	30.01	40.52	**8.0**	**99.31**	**0.82**
128	4.50	19.94	47.29	**8.0**	**98.22**	**1.20**
252	3.10	19.98	46.57	**8.0**	**89.70**	**2.24**
512	3.50	29.97	41.19	**8.0**	**95.18**	**1.71**

Table 6: Comparison of WGAN and ISL + WGAN with different batch sizes.

4.4 Addressing mode collapse on MNIST and FMNIST

In this section, we assess the effectiveness of ISL-slicing for generating high-dimensional images. ISL-slicing approximates the multivariate objective by averaging rank-based losses over m random one-dimensional projections; as discussed and empirically quantified in Appendix C.4, m controls a bias/variance (and "slicing complexity") trade-off, and a larger m may be required to reliably capture highly localized discrepancies in high dimension. Moreover, as noted in the literature on sliced objectives (Nadjahi et al., 2020), sliced losses are often strongly *coverage-oriented* (high recall/diversity) but may under-emphasize fine, high-frequency perceptual details. For this reason we evaluate a two-stage pipeline where ISL-slicing is used for *pretraining* (to obtain a stable, well-covered initialization) and standard adversarial training is used for refinement. We integrate the ISL-slicing loss into the deep convolutional GAN (DCGAN) generator's training (Radford et al., 2015) and evaluate its performance on the MNIST and Fashion-MNIST benchmarks. Following Sajjadi et al. (2018), we characterize the quality and diversity of generated images by reporting precision (as a proxy for fidelity) and recall (as a proxy for diversity). We train each model for 40 epochs with a batch size of 128. For the pretrained variants, we first ran 20 epochs under the sliced dual-ISL objective (using 20 random projections) and then continued with 40 epochs of standard DCGAN training.

In Table 7, we compare our results to those of other implicit generative models. On MNIST, our straightforward ISL-based model matches the recall of five-discriminator GANs (Durugkar et al., 2016; Choi and Han, 2022) while using only one-third the number of parameters. Moreover, pretraining the generator with sliced-ISL and then fine-tuning under the standard adversarial loss delivers state-of-the-art precision and recall.

On Fashion-MNIST, our model almost ties the recall of MCL-GAN's and, although the recall of GMAN is higher, we outperform it in precision. This shows that despite relying on a simpler architecture, ISL can achieve competitive recall and precision across diverse image-generation benchmarks.

Dataset	**Method**	**F-score**		**P&R**	
		$F_{1/8}\uparrow$	$F_8\uparrow$	Precision↑	Recall↑
MNIST	ISL (m=20)	85.00 ± 0.32	95.17 ± 1.76	84.85 ± 1.20	95.35 ± 1.39
	ISL (m=50)	85.69 ± 0.29	95.81 ± 1.24	85.55 ± 1.11	96.23 ± 1.98
	DCGAN	93.58 ± 0.64	75.66 ± 1.46	93.85 ± 1.45	75.43 ± 2.56
	ISL + DCGAN	93.58 ± 0.84	95.82 ± 1.61	94.03 ± 1.82	96.68 ± 2.42
	GMAN	97.60 ± 0.70	96.81 ± 1.71	97.60 ± 1.82	96.80 ± 2.42
	MCL-GAN	$\mathbf{97.71 \pm 0.19}$	$\mathbf{98.49 \pm 1.57}$	$\mathbf{97.70 \pm 1.33}$	$\mathbf{98.50 \pm 2.15}$
FMNIST	ISL (m=20)	81.84 ± 0.11	91.08 ± 1.83	81.48 ± 1.43	91.49 ± 2.15
	ISL (m=50)	83.90 ± 0.09	91.18 ± 1.57	84.08 ± 1.31	92.92 ± 1.23
	DCGAN	86.14 ± 0.11	88.92 ± 1.51	86.60 ± 1.58	88.97 ± 1.33
	ISL + DCGAN	91.43 ± 0.19	91.87 ± 1.57	91.88 ± 1.35	92.42 ± 1.47
	GMAN	90.97 ± 0.09	$\mathbf{95.43 \pm 1.12}$	90.90 ± 1.33	$\mathbf{95.50 \pm 2.25}$
	MCL-GAN	$\mathbf{97.62 \pm 0.09}$	92.97 ± 1.28	$\mathbf{97.70 \pm 1.33}$	92.90 ± 2.31

Table 7: Quantitative results on MNIST and Fashion-MNIST (28×28), reporting $F_{1/8}$ and F_8 (β-weighted harmonic means of precision and recall), precision, and recall (mean±std, %). We compare ISL (m=20, 50), standard DCGAN (with and without ISL pretraining), GMAN, and MCL-GAN. Bold entries mark the best score per column; higher is better.

Figure 9 compares the class-frequency distributions produced by the sliced ISL model (trained for 40 epochs with $m = 50$ random projections) against those generated by a conventional DCGAN. Our model generates all ten digit classes with nearly uniform frequency whereas the DCGAN displays marked class imbalances. To estimate these frequencies, we sampled $10,000$ images from each model and labelled them using a pretrained digit classifier. Under a one-sample Kolmogorov–Smirnov test for uniformity on $10,000$ samples, the sliced-ISL model yielded a p-value $p = 0.062$, whereas the DCGAN produced $p = 0.742$—reflecting a much closer match to the ideal uniform distribution.

Figure 10 shows the precision and recall of the ISL model on MNIST in an ablation study with $m = 20$ and $m = 100$ random projections. With just $m = 20$ projections, the model delivers robust performance; increasing to $m = 100$ yields only marginal improvements in these metrics. Monte Carlo sampling error for the sliced-ISL estimator scales as $1/\sqrt{m}$, so initial increases in m deliver significant gains but further increases produce only marginal improvements, explaining the observed performance plateau.

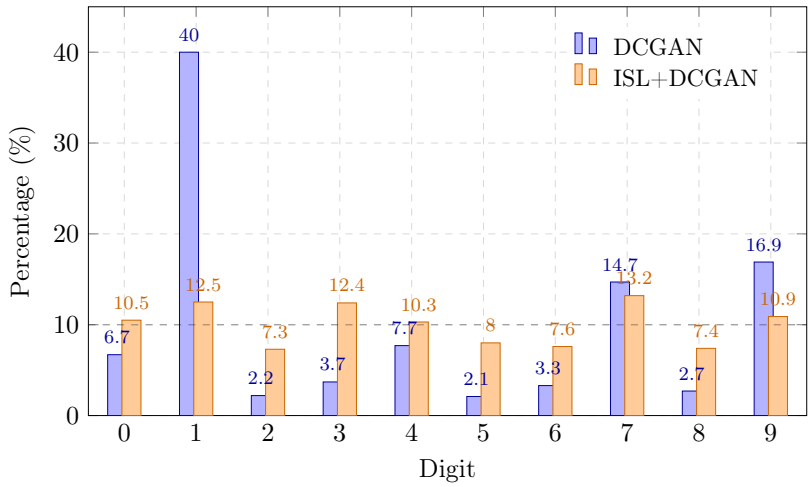


Figure 9: Digit-frequency comparison for DCGAN vs. ISL+DCGAN. The dashed line marks the ideal 10% for each class.

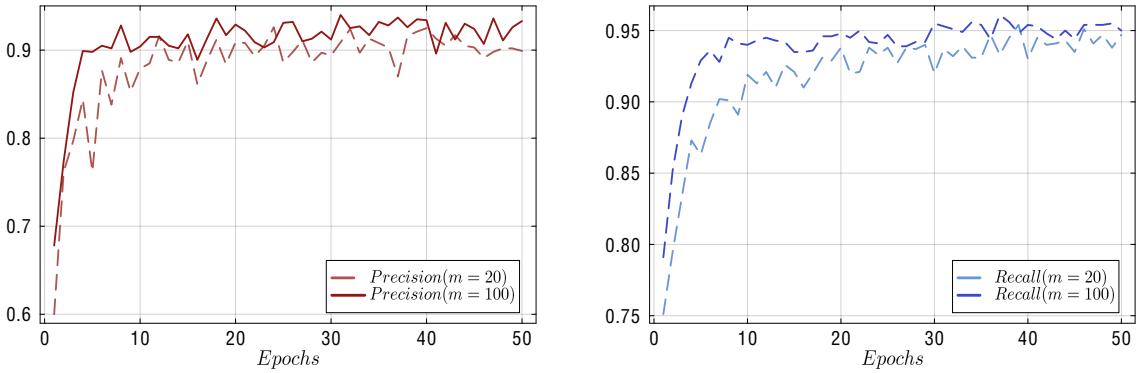


Figure 10: Precision (left) and recall (right) on the MNIST test set after 50 training epochs, using $m = 20$ and $m = 100$ random projections.

Finally, Figure 11 compares the sample diversity of a standard DCGAN Figure 11a against a DCGAN pretrained with our ISL method Figure 11b. In Figure 11a, the red boxes highlight multiple occurrences of the digit "1," a clear sign of mode collapse where the generator repeatedly produces the same class. After pretraining with ISL (Figure 11b), the generator exhibits far fewer repeated "1"s, instead producing more diverse handwritten digits. This visual evidence demonstrates that ISL pretraining effectively mitigates mode collapse and encourages the generator to cover a broader range of the MNIST data distribution.

4.5 Improving diversity on CelebA with ISL pretraining

In this section, we evaluate ISL pretraining on the CelebA dataset under a unified 50-epoch regime. We first pretrain a DCGAN with ISL for 10 epochs (learning rate 10^2, $K = 40$), then fine-tune it adversarially for 40 epochs (batch size 128, learning rate 2×10^{-4}). For a fair

(a) DCGAN (b) DCGAN pretrained with ISL

Figure 11: Generated samples from the MNIST dataset: plot (a) shows numbers generated by a DCGAN, while the plot (b) shows numbers generated by a DCGAN pretrained with ISL. Red squares around the repeated "1"'s.

comparison, we trained every baseline using 50 epochs, batch size 128, and a grid search over learning rates $\{1 \times 10^{-2}, 1 \times 10^{-3}, 2 \times 10^{-4}, 1 \times 10^{-4}\}$. The baselines we evaluated were

- Generative multi-adversarial networks (GMAN) (Durugkar et al., 2016): uses multiple discriminators to improve mode coverage.
- Least-squares DCGAN (LS-DCGAN) (Mao et al., 2017): swaps the usual cross-entropy loss for a least-squares objective.
- Wasserstein GAN with gradient penalty (W-DCGAN-GP) (Gulrajani et al., 2017): enforces the 1-Lipschitz constraint via a gradient penalty.
- Spectral normalization DCGAN (SN-DCGAN) (Miyato et al., 2018): stabilizes training by normalizing discriminator weights.
- Dynamic GAN (DynGAN) (Luo and Yang, 2024): detects collapsed modes during training and uses conditional generators to recover them.

From each model's single run, we report the best recall, Fréchet Inception Distance (FID), and precision scores. Full details are provided in Table 8.

Remarkably, our ISL-pretrained DCGAN achieves the highest recall of all models, despite its relative simplicity. It even outperforms multi-discriminator frameworks such as GMAN—while using three times fewer parameters—and specialized techniques like DynGAN, which introduce semi-supervised overhead to recover collapsed modes. Moreover, in terms of both FID and precision, the ISL-pretrained network remains highly competitive, despite

fitted only on 40 adversarial training epochs (10 fewer than every baseline). These results underscore the power of ISL pretraining to capture diverse image modes without extra architectural complexity.

Dataset	**Method**	*FID* ↓	*Precision* ↑	*Recall* ↑
CelebA	ISL + DCGAN	31.64	0.887	**0.954**
	DCGAN (Radford et al., 2015)	30.93	0.839	0.834
	LS-DCGAN (Mao et al., 2017)	**22.99**	**0.997**	0.324
	W-DCGAN-GP (Gulrajani et al., 2017)	80.30	0.982	0.374
	SN-DCGAN (Miyato et al., 2018)	32.94	0.974	0.887
	DynGAN (Luo and Yang, 2024)	48.06	0.955	0.718
	GMAN (Durugkar et al., 2016)	31.66	0.873	0.888

Table 8: Performance comparison on CelebA. We evaluate ISL-pretrained DCGAN (ISL+DCGAN), standard DCGAN, Least Squares DCGAN (LS-DCGAN), Spectral Normalization DCGAN (SN-DCGAN), DynGAN, and Generative Multi-Adversarial Networks (GMAN) using Fréchet Inception Distance (FID↓), precision (↑), and recall (↑). Lower FID and higher precision/recall indicate better sample quality and diversity.

Our two-stage training pipeline (first pretraining the generator by minimizing ISL, then fine-tuning it with standard adversarial updates) parallels the strategy of the sliced-Wasserstein generator (SWG) from Nadjahi et al. (2021). In SWG, the authors begin by minimizing a Monte Carlo estimate of the sliced-Wasserstein distance in feature space before introducing a discriminator (see Nadjahi et al. (2021, Sec. 3.2)). They motivate this design by noting that naive slicing in very high dimensions demands an impractically large number of random projections; adding a learned discriminator effectively reduces the problem's dimensionality to the most discriminative subspace, thereby cutting the projection requirement and restoring strong FID performance. We build on exactly this insight: by replacing the Monte Carlo sliced-Wasserstein objective with ISL during the initial phase, we still "warm up" the generator into a space close to the real-data manifold, but with a discrepancy measure that more directly reduces mode collapse. Under identical hyperparameter settings and evaluation protocols, ISL pretraining achieves substantially higher recall on every benchmark, while matching the precision and FID of SWG. These results confirm that ISL is a more effective statistical divergence than sliced-Wasserstein for preserving diversity without sacrificing sample fidelity. Table 9 provides the full comparison, highlighting the superior recall of ISL alongside equivalent precision and FID.

5. Time series prediction

Our approach applies to both univariate and multivariate time series. Hereafter we briefly describe the methodology for both types for series and conclude the section with experiments comparing these techniques on various datasets.

m	ISL + DCGAN			SWG			SWG-2		
	FID ↓	*Precision* ↑	*Recall* ↑	*FID* ↓	*Precision* ↑	*Recall* ↑	*FID* ↓	*Precision* ↑	*Recall* ↑
100	**31.64**	**0.89**	**0.95**	32.82	1.00	0.16	34.57	1.00	0.14
1 000	**30.41**	**0.91**	**0.94**	37.27	0.98	0.68	35.83	1.00	0.56
10 000	**30.33**	**0.92**	**0.93**	37.29	0.98	0.63	37.20	0.99	0.74

Table 9: Comparison for CelebA between ISL + DCGAN, the Sliced-Wasserstein Generator (SWG), and its Sliced-Wasserstein-2 variant (SWG-2) as a function of the number of random projections m. Lower FID and higher precision/recall indicate better sample quality and diversity.

5.1 Univariate time series prediction

Let $y[0], y[1], \ldots, y[T]$ represent a realization of a discrete-time random process. We assume that the process begins at $t = 0$, $Y[t]$ denotes the r.v. at time t, and $p_t = p(Y[t]|Y[0], \ldots, Y[t-1])$ is the unknown conditional distribution. Given the sequence $y[0], y[1], \ldots, y[t-1]$, we aim to train an autoregressive conditional implicit generator network, $g_\theta(z_t, \mathbf{h}[t])$, to approximate p_t. Here, z_t is a sequence of i.i.d. Gaussian r.v.s, and $\mathbf{h}[t]$ is an embedding of the sequence $y[0], \ldots, y[t-1]$ via a NN, such as a simple RNN connected to the generator (see Figure 12). At time t, the observation $y[t]$ is fed into the RNN, compressing the history into a hidden state $\mathbf{h}[t]$. The generator $g_\theta(z, \mathbf{h}[t])$ uses this hidden state and noise z to predict $\tilde{y}[t+1]$. During testing, $\tilde{y}[t+1]$ is fed back into the RNN for forecasting.

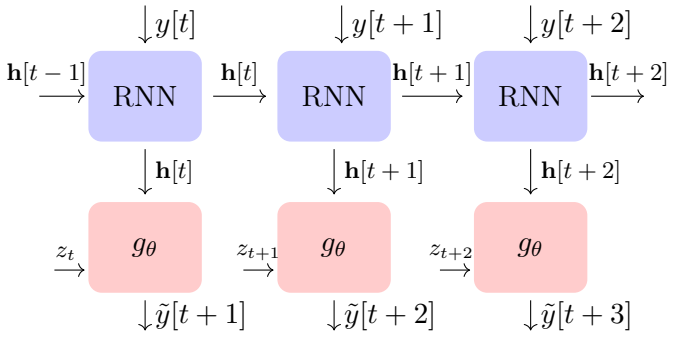


Figure 12: Conditional implicit generative model for time-series prediction.

As noted in de Frutos et al. (2024), all results from previous sections remain valid in this temporal setup. The sequence of observations $\mathbf{y} = [y[0], y[1], \ldots, y[T]]$ is used to construct a sequence of statistics $a_K[0], a_K[1], \ldots, a_K[T]$, whose empirical distribution should be approximately uniform if $\tilde{p}_{t,\theta} \approx p_t$ for $t = 0, \ldots, T$, where $\tilde{p}_{t,\theta}$ is the pdf of the output r.v. $\tilde{y}[t] = g_\theta(z_t, \mathbf{h}[t])$. To build the ISL, we follow the same procedure described in Section 2 to obtain a differentiable surrogate.

5.2 Multivariate time series prediction

For a multivariate time series the data has the form $\mathbf{y} = [\mathbf{y}[0], \ldots, \mathbf{y}[T]]$ where each element of the series is an N-dimensional vector, i.e., $\mathbf{y}[t] = [y_1[t], \ldots, y_N[t]]^\top$. A NN $g_\theta(\cdot, \mathbf{h}[t])$ can

be trained using the same scheme as in Figure 12 and an ISL loss function constructed from the N marginals $p(y_i[t]|\mathbf{y}[0], \ldots, \mathbf{y}[t-1])$, as suggested in de Frutos et al. (2024), or using the ISL-slicing (Algorithm 2) method introduced in Section 4.

5.3 Experiments

This subsection presents the results of multivariate long-sequence time-series forecasting on the ETTh2, ETTm1, and ETTm2 datasets using Autoformer (Wu et al., 2021), Informer(Zhou et al., 2021), LogTrans(Nie et al., 2022) and various ISL-based methods. The Electricity Transformer Temperature (ETT) dataset comprises four distinct subsets: two with hourly resolutions (ETTh) and two with 15-minute resolutions (ETTm), each containing seven features related to oil and load characteristics of electricity transformers collected between July 2016 and July 2018. For more details on the methods and datasets, refer to Zeng et al. (2023). The evaluation spans multiple forecasting horizons (τ) and is assessed using two key metrics: mean squared error (MSE) and mean absolute error (MAE). Detailed results are provided in Table 10.

For these experiments, we use a 1-layer RNN with 5 units, followed by batch normalization and a 2-layer MLP with 10 units per layer. The MLP has ReLU in the first layer, identity activation in the last, and 5% dropout in the first layer. Despite this simple architecture, ISL outperforms state-of-the-art transformers. We also observe that the ISL-slicing method generally performs better than both the marginal fitting technique and other state-of-the-art methods, with improvements as the number of projections increases. Notably, the ISL models (with $\approx$ 25K parameters), using an RNN and a simple MLP, achieves better forecasting accuracy than many transformer models with millions of parameters (Zeng et al., 2023).

DB	τ	**Autoformer**		**Informer**		**LogTrans**		**ISL-M**		**ISL-S7**		**ISL-S10**		**ISL-S20**	
		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh2	96	0.36	**0.40**	3.76	1.53	2.12	1.20	0.42	0.49	0.31	0.45	0.28	0.43	**0.26**	0.41
	192	**0.46**	**0.45**	5.60	1.93	4.32	1.63	0.68	0.69	0.61	0.64	0.54	0.57	0.55	0.60
	336	0.48	0.49	4.72	1.84	1.12	1.60	0.61	0.64	0.41	0.55	**0.33**	**0.43**	0.44	0.51
	720	**0.52**	**0.51**	3.65	1.63	3.19	1.54	0.60	0.62	0.75	0.68	0.56	0.57	0.65	0.64
ETTm1	96	0.67	0.57	0.54	0.51	0.60	0.55	0.41	0.51	0.43	0.55	0.28	0.43	**0.19**	**0.37**
	192	0.80	0.67	0.56	0.54	0.84	0.70	0.57	0.62	0.67	0.65	0.35	**0.36**	**0.30**	0.49
	336	1.21	0.87	0.75	0.66	1.12	0.83	0.70	0.67	0.79	0.78	0.67	0.68	**0.54**	**0.61**
	720	1.17	0.82	0.91	0.72	1.15	0.82	0.77	**0.69**	0.92	0.73	0.87	0.77	**0.76**	0.74
ETTm2	96	0.26	**0.34**	0.37	0.45	0.77	0.64	0.28	0.44	**0.21**	0.38	0.22	0.37	0.22	0.36
	192	0.28	0.34	0.53	0.56	0.99	0.76	0.34	0.46	0.33	0.43	0.26	0.40	**0.21**	**0.34**
	336	0.34	0.37	1.36	0.89	1.33	0.87	0.31	0.46	0.37	0.47	0.31	0.45	**0.15**	**0.31**
	720	0.43	0.43	3.38	1.34	3.05	1.33	0.43	0.55	0.37	0.50	0.36	0.49	**0.24**	**0.40**

Table 10: Forecasting results for multivariate time-series on ETTh2, ETTm1, and ETTm2 datasets using Autoformer, Informer, LogTrans and ISL. ISL-M denotes the marginal approach, and ISL-Sn is the ISL-slicing with n random projections.

6. Limitations and future work

While ISL and its sliced variant typically yield stable optimization and good distributional coverage, they do not yet match the best adversarial methods in terms of perceptual quality on high–dimensional image benchmarks.

This performance gap can be attributed to three primary factors. First, sliced objectives are only approximate proxies of the underlying multivariate divergence: using a finite number of projections m and a finite discretisation level K induces a non–negligible bias–variance trade–off. Specifically, K governs the approximation error of the one-dimensional density estimates (the bias), while m controls the Monte Carlo variance of the manifold-to-line projection. The value of the pair (m, K) required to closely approximate the target divergence in very high dimensions can be computationally demanding, as the cost grows essentially linearly in both parameters. Moreover, as reported in prior work on sliced–Wasserstein generative models (Deshpande et al., 2018; Nadjahi et al., 2020, 2021), while sliced objectives are computationally attractive and coverage–oriented, they can be challenging to optimise when the goal is state–of–the–art perceptual quality on large–scale image data. This is often due to the difficulty of capturing narrow, high-frequency features of the data manifold when limited by a fixed projection budget. Second, our current experiments optimize a purely pixel–space divergence, which is not perfectly aligned with perceptual metrics computed in deep feature spaces (e.g., FID, precision/recall), and thus may under–emphasize global structure and high–frequency texture compared with specialized GAN losses. Third, the method remains sensitive to a small set of hyperparameters (notably, m and K), for which we only provide empirical guidelines in the present version.

We see several promising directions for improvement.

- More informative projection schemes: To address the aforementioned accuracy–cost trade-off, one can move beyond i.i.d. random directions. Recent work on generalized, max–sliced, and learned sliced–Wasserstein distances (Kolouri et al., 2019; Deshpande et al., 2019; Paty and Cuturi, 2019) shows that adapting the projection distribution to the data geometry can substantially increase the information carried by each slice. Analogous adaptive strategies for ISL, such as learning projection families jointly with the generator or biasing slices toward directions with large rank/discrepancy signals, could bridge the gap between random slicing and full high–dimensional divergences. Additionally, exploring how specific slicing directions can better capture the extreme-value dependence structures inherent in the heavy-tailed data handled by Pareto-ISL remains an important open question.

- Developing principled rules or adaptive schedules for choosing (m, K) under a given computational budget, supported by sharper approximation and generalization guarantees: Beyond heuristic tuning, future research should aim to establish finite–sample error bounds that characterize the total approximation gap $\left|\tilde{d}_K^{S_d}(p, \tilde{p}_\theta) - d_K^{S_d}(p, \tilde{p}_\theta)\right|$, representing the discrepancy between the Monte Carlo estimator and the true multivariate divergence, as well as the resulting generalization error of the generator. These bounds should explicitly account for the interaction between (m, K), sample size, and dimensionality. Such a theoretical foundation would, in turn, yield principled prescriptions for practical implementation, such as determining the optimal number of slices per batch or designing an adaptive schedule for growing K as training progresses.

- Feature-space ISL variants computed on learned representations (e.g., embeddings from a pre-trained ResNet or Vision Transformer (He et al., 2016; Wang et al., 2021)): Such variants could leverage the low-dimensional manifold structure inherent in deep features, allowing ISL to capture semantic consistency and global coherence that pixel-wise divergences often overlook.

- Finally, a complementary direction is to develop continuous-time gradient flows in Wasserstein space driven by ISL-slicing, in the spirit of sliced-Wasserstein flows for implicit generative modeling (Liutkus et al., 2019). In that framework, one minimizes a functional of the form "sliced divergence + entropy" over probability measures and follows its Wasserstein gradient flow, which can be implemented via a McKean–Vlasov SDE and an interacting particle system with finite-time error bounds. Replacing the sliced Wasserstein cost by ISL-slicing would lead to nonparametric particle algorithms whose drift is expressed in terms of one-dimensional ISL statistics, and could serve either as standalone generators or as preconditioning / initialization mechanisms for training neural implicit models. Analyzing well-posedness, convergence rates, and the interaction between such ISL flows and finite-dimensional generators remains an open but promising avenue.

7. Conclusions

We have investigated the construction of loss functions based on rank statistics with an invariant distribution in order to train implicit generative models that can reproduce heavy-tailed and multivariate data distributions. As a result, we have introduced two novel methodologies, Pareto-ISL and ISL-slicing. Pareto-ISL is used to fit a NN with a Pareto-distributed random input. The resulting model can represent accurately both the central features and the tails of heavy-tailed data distributions. ISL-slicing can be used to train NN with a random input and a high-dimensional output. We have shown through simulations with several data sets (including time series data) that the proposed methods are competitive with (and often outperform) state of the art models. The computational cost of training generative models with ISL-based methods is usually low compared to that of state of the art schemes.

Acknowledgments

This work was partially supported by the Office of Naval Research (award N00014-22-1-2647) and by Spain's Agencia Estatal de Investigación (ref. PID2024-158181NB-I00 NISA, PID2024-157856NB-I00 CARTESIAN, and PID2021-123182OB-I00 EPiCENTER), funded by MCIN/AEI/10.13039/501100011033 and by the ERDF ("A way of making Europe"). Pablo M. Olmos was also supported by the Comunidad de Madrid (IND2024/TIC-34728), the IDEA-CM project (TEC-2024/COM-89), the ELLIS Unit Madrid (European Laboratory for Learning and Intelligent Systems), and the 2024 Leonardo Grant for Scientific Research and Cultural Creation from the BBVA Foundation, and has also received funding from the EU Horizon Europe research and innovation programme under the Marie Skłodowska-Curie Grant Agreement No. 101226456 (MLCARE: Machine Learning Computational Advancements

for peRsonalized mEdicine). The views and opinions expressed are, however, those of the author(s) only and do not necessarily reflect those of the EU or the European Commission (EC). Neither the EU nor the EC can be held responsible for them.

Appendix A. Proof of Theorem 2

Proof For clarity, we introduce the following notation: for a real function $f : \mathbb{R} \to \mathbb{R}$ and the pdf p of a univariate r.v. we define

$$(f, p) := \int_{\mathbb{R}} f(x)p(x)\, dx.$$

Let $B_1(\mathbb{R}) := \{f : \mathbb{R} \to \mathbb{R} \mid \sup_{x\in\mathbb{R}} |f(x)| \leq 1\}$ be the set of real functions bounded by 1. We note that the assumption $\sup_{f\in B_1(\mathbb{R})} |(f, p) - (f, \tilde{p})|$ in (de Frutos et al., 2024, Theorem 2) is equivalent to the assumption of Theorem 4. Indeed,

$$\begin{aligned}\|p - \tilde{p}\|_{L^1(\mathbb{R})} = \int_{\mathbb{R}} |p(x) - \tilde{p}(x)|\, dx &= \int_{\mathbb{R}} \operatorname{sgn}(p(x) - \tilde{p}(x))(p(x) - \tilde{p}(x))\, dx \\ &= \sup_{f\in B_1(\mathbb{R})} |(f, p) - (f, \tilde{p})|\,.\end{aligned}$$

The remainder of the proof follows directly from (de Frutos et al., 2024, Theorem 2). ■

Appendix B. Proof of Theorem 6

B.1 Preliminary results

Let, $\tilde{F}_\theta(y_0)$ represents the cdf of the transformed r.v. $y = g_\theta(z)$, evaluated at y_0, where g_θ is a function parameterized by θ. The r.v. y has a pdf denoted $\tilde{p}_\theta$ and $z \sim p_z$.

Let $\tilde{y}_1, \ldots, \tilde{y}_k$ be K i.i.d. draws from $\tilde{p}_\theta$. Using $\tilde{F}_\theta(y_0)$, we define the probability of observing exactly n successes in K independent Bernoulli trials. The i-th trial is considered a success when $\tilde{y}_i \leq y_0$, hence the success probability is $\tilde{F}_\theta(y_0)$. We write of the resulting binomial distribution as

$$h_{n,\theta}(y_0) = \binom{K}{n} \left[\tilde{F}_\theta(y_0)\right]^n \left[1 - \tilde{F}_\theta(y_0)\right]^{K-n}, \quad \text{for } n \in \{0, \ldots K\}. \tag{B.1}$$

Using B.1, the pmf of the rank statistic $A_K = \left|\left\{\tilde{y} \in \{\tilde{y}_i\}_{i=1}^K : \tilde{y} \leq y_0\right\}\right|$ when $y_0 \sim \tilde{p}_\theta$ can be constructed as

$$\mathbb{Q}_{K,\tilde{p}_\theta}(n) = \int_{\mathbb{R}} h_{n,\theta}(y)\, \tilde{p}_\theta(y)\, dy, \quad \text{for } n \in \{0, \ldots, K\}. \tag{B.2}$$

Lemma 1 *The cdf $\tilde{F}_\theta(y_0) = \mathbb{P}(g_\theta(z) \leq y_0)$ is continuous in θ, for every fixed $y_0 \in \mathbb{R}$.*

Proof The cdf $\tilde{F}_\theta(y_0)$ can be expressed as

$$\tilde{F}_\theta(y_0) = \mathbb{P}(y \leq y_0) = \mathbb{P}(g_\theta(z) \leq y_0) = \int_{\mathbb{R}} \mathbb{I}_{S_{y_0,\theta}}(z)\, p_z(z)\, dz,$$

where p_z is the pdf of the input noise, $S_{y_0,\theta} = \{u \in \mathbb{R} : g_\theta(u) \leq y_0\}$ and $\mathbb{I}_{S_{y_0,\theta}}(z)$ is the indicator function. We need to prove that for any sequence $\{\theta_m\}_{m=1}^\infty$ such that $\theta_m \to \theta$ as $m \to \infty$, we have $\lim_{m\to\infty} \left|\tilde{F}_{\theta_m}(y_0) - \tilde{F}_\theta(y_0)\right| = 0$. We can write

$$\left|\tilde{F}_{\theta_m}(y_0) - \tilde{F}_\theta(y_0)\right| \leq \int_{\mathbb{R}} \left|\mathbb{I}_{S_{y_0,\theta_m}}(z) - \mathbb{I}_{S_{y_0,\theta}}(z)\right| p_z(z)\, dz.$$

If we define the set $A_m = \{u \in \mathbb{R} : (g_\theta(u) \le y_0 < g_{\theta_m}(u)) \text{ or } (g_{\theta_m}(u) \le y_0 < g_\theta(u))\}$ then is clear that

$$\left|\mathbb{I}_{S_{y_0,\theta_m}}(z) - \mathbb{I}_{S_{y_0,\theta}}(z)\right| = \begin{cases} 1 & \text{if } z \in A_m, \\ 0 & \text{otherwise.} \end{cases}$$

Hence, we can write

$$\left|\tilde{F}_{\theta_m}(y_0) - \tilde{F}_\theta(y_0)\right| = \int_{\mathbb{R}} \mathbb{I}_{A_m}(z)\, p_z(z)\, dz = \int_{A_m} p_z(z)\, dz, \tag{B.3}$$

and there is a constant $C < \infty$ such that $\int_{A_m} p_z\, dz \le C \int_{A_m} dz$. However $g_\theta(z)$ is continuous in θ for almost every $z \in \mathbb{R}$, i.e., $\lim_{m\to\infty} g_{\theta_m}(z) = g_\theta(z)$, which implies that $\lim_{m\to\infty} \int_{A_m} dz = 0$. Therefore $\lim_{m\to\infty} \left|\tilde{F}_{\theta_m}(y_0) - \tilde{F}_\theta(y_0)\right| \le C \int_{A_m} dz = 0$. ■

Lemma 2 *For each $n \in \{0, 1, \ldots, K\}$, there is a constant $C_{n,K} < \infty$ such that*

$$\left|h_{n,\theta}(y_0) - h_{n,\theta'}(y_0)\right| \le \binom{K}{n} C_{n,K} \left|\tilde{F}_\theta(y_0) - \tilde{F}_{\theta'}(y_0)\right|.$$

Proof We can bound the difference between $h_{n,\theta}(y_0)$ and $h_{n,\theta'}(y_0)$ using B.1 as

$$\left|h_{n,\theta}(y_0) - h_{n,\theta'}(y_0)\right| \le \binom{K}{n} \left|\tilde{F}_\theta(y_0)^n (1 - \tilde{F}_\theta(y_0))^{K-n} - \tilde{F}_{\theta'}(y_0)^n (1 - \tilde{F}_{\theta'}(y_0))^{K-n}\right|.$$

Let us define $f(q) = q^n (1-q)^{K-n}$, which is continuously differentiable for $q \in [0, 1]$. By the mean value theorem, for some value $\tilde{F}_{\theta^*}(y_0) \in \left[\tilde{F}_\theta(y_0) \wedge \tilde{F}_{\theta'}(y_0), \tilde{F}_\theta(y_0) \vee \tilde{F}_{\theta'}(y_0)\right]$, we have

$$\left|f(\tilde{F}_\theta(y_0)) - f(\tilde{F}_{\theta'}(y_0))\right| \le \left|f'(\tilde{F}_{\theta^*}(y_0))\right| \left|\tilde{F}_\theta(y_0) - \tilde{F}_{\theta'}(y_0)\right|.$$

Since $f(q) = q^n(1-q)^{K-n}$ is a polynomial in q, its derivative $f'(q)$ is continuous and bounded on the interval $q \in [0, 1]$. Given that $\tilde{F}_{\theta^*}(y_0) \in [0, 1]$ (as it is a cdf), there exists a constant C_n such that, $\left|f'(\tilde{F}_{\theta^*}(y_0))\right| \le C_n$ for any $\tilde{F}_{\theta^*}(y_0)$. ■

Lemma 3 *Let $\tilde{F}_\theta(y_0)$ and $\tilde{F}_{\theta'}(y_0)$ be the cdfs of the transformed r.v.s $y = g_\theta(z)$ and $y' = g_{\theta'}(z)$, respectively, where $g_\theta(z)$ is differentiable w.r.t. z and satisfies the Lipschitz condition $|g_\theta(z) - g_{\theta'}(z)| \le L_{\max}\|\theta - \theta'\|$ for some Lipschitz constant $L_{\max} < \infty$, and there exists $m > 0$ such that $\inf_{(z,\theta)} |g'_\theta(z)| \ge m$. Then*

$$\left|\tilde{F}_\theta(y_0) - \tilde{F}_{\theta'}(y_0)\right| \le L_1 \|\theta - \theta'\|,$$

where $L_1 = \|p_z\|_{L^\infty(\mathbb{R})} \dfrac{2L_{\max}}{m}$ and $p_z(z)$ is the pdf of the input variable z.

Proof Let $A = \{u \in \mathbb{R} : (g_\theta(u) \leq y_0 < g_{\theta'}(u)) \text{ or } (g_{\theta'}(u) \leq y_0 < g_\theta(u))\}$. By the same argument as in the proof of Lemma 1 (see Eq. B.3) we have

$$\left|\tilde{F}_\theta(y_0) - \tilde{F}_{\theta'}(y_0)\right| \leq \int_A p_z(z)\, dz. \tag{B.4}$$

We now prove that $A \subseteq B = \left\{u \in \mathbb{R} \;\middle|\; |g_\theta(u) - y_0| \leq L_{\max} \|\theta - \theta'\|\right\}$. For any $z \in A$, there are two possible cases to consider: either $g_\theta(z) \leq y_0 < g_{\theta'}(z)$ or $g_{\theta'}(z) \leq y_0 < g_\theta(z)$. In the first case, we can see that $|g_\theta(z) - y_0| \leq |g_\theta(z) - g_{\theta'}(z)| \leq L_{\max} \|\theta - \theta'\|$ by the Lipschitz assumption. An analogous argument holds for the second case. Thus, we have shown that for any $z \in A$, $|g_\theta(z) - y_0| \leq L_{\max} \|\theta - \theta'\|$ and, therefore, $A \subseteq B$.

Next, we estimate the Lebesgue measure $\mathcal{L}(B)$ of the set B. By assumption, $g_\theta(z)$ is differentiable w.r.t. z, with $\inf_{(z,\theta)\in\mathcal{Z}\times\mathbb{R}^d} |g'_\theta(z)| \geq m > 0$. Hence, we can perform a change of variable $y = g_\theta(z)$, and obtain $z = g_\theta^{-1}(y)$, where g_θ^{-1} is the local inverse function. We note that $z \in B$ when $y \in [y_0 - \delta, y_0 + \delta]$, where $\delta = L_{\max} \|\theta - \theta'\|$. The Lebesgue measure of B can be upper bounded as

$$\mathcal{L}(B) = \int_B dz \leq \int_{y_0-\delta}^{y_0+\delta} \left|\frac{dz}{dy}\right| dy \leq \frac{1}{m} \int_{y_0-\delta}^{y_0+\delta} dy = \frac{2\delta}{m} = \frac{2L_{\max}}{m} \|\theta - \theta'\|,$$

since $\frac{dz}{dy} = \frac{1}{g'_\theta(z)}$ and, by assumption, $\inf_{(z,\theta)} |g'_\theta(z)| \geq m$, hence $\left|\frac{dz}{dy}\right| = \frac{1}{|g'_\theta(z)|} \leq \frac{1}{m}$.
Finally, we proceed to bound the absolute difference between the cdfs. In particular,

$$\left|\tilde{F}_\theta(y_0) - \tilde{F}_{\theta'}(y_0)\right| \leq \int_A p_z(z)\, dz \leq \int_B p_z(z)\, dz \leq \|p_z\|_{L^\infty(\mathbb{R})}\, \mathcal{L}(B) \leq L_1 \left\|\theta - \theta'\right\|,$$

the first inequality follows from B.4, the second is given by $A \subseteq B$, and the last inequality follow from the upper bound on $\mathcal{L}(B)$ with $L_1 = \|p_z\|_{L^\infty(\mathbb{R})}\, \dfrac{2L_{\max}}{m}$. ■

B.2 Proof of Theorem 6

Proof Continuity (Part 1)

To prove that $d_K(p, \tilde{p}_\theta)$ is continuous w.r.t. θ, we need to show that

$$\lim_{m\to\infty} |d_K(p, \tilde{p}_{\theta_m}) - d_K(p, \tilde{p}_\theta)| = 0$$

for any sequence $\{\theta_m\}_{m=1}^\infty$ such that $\theta_m \to \theta$.

We begin by examining the difference $|d_K(p,\tilde{p}_{\theta_m}) - d_K(p,\tilde{p}_\theta)|$. Let us denote $\mathbf{q}_{K,\tilde{p}_\theta} = [\mathbb{Q}_{K,\tilde{p}_\theta}(0),\ldots,\mathbb{Q}_{K,\tilde{p}_\theta}(K)]^\top$. We readily arrive at the bound

$$\begin{aligned}
|d_K(p,\tilde{p}_{\theta_m}) - d_K(p,\tilde{p}_\theta)| &= \frac{1}{K+1}\left|\left\|\frac{1}{K+1}\mathbf{1}_{K+1} - \mathbf{q}_{K,\tilde{p}_{\theta_m}}\right\|_{\ell_1} - \left\|\frac{1}{K+1}\mathbf{1}_{K+1} - \mathbf{q}_{K,\tilde{p}_\theta}\right\|_{\ell_1}\right| \\
&\leq \frac{1}{K+1}\left\|\mathbf{q}_{K,\tilde{p}_{\theta_m}} - \mathbf{q}_{K,\tilde{p}_\theta}\right\|_{\ell_1} \qquad \text{(B.5)} \\
&\leq \frac{1}{K+1}\sum_{n=0}^{K}\int_{\mathbb{R}} |h_{n,\theta_m}(y) - h_{n,\theta}(y)|\,\max\left\{\tilde{p}_{\theta_m}(y),\tilde{p}_\theta(y)\right\}\,dy, \qquad \text{(B.6)}
\end{aligned}$$

where (B.5) follows from the reverse triangle inequality and (B.6) is obtained from the construction of the pmf $\mathbb{Q}_{K,\tilde{p}_\theta}(n)$ in Eq. (B.2). It is easy to see that

$$|h_{n,\theta_m}(y) - h_{n,\theta}(y)|\max\{\tilde{p}_{\theta_m}(y),\tilde{p}_\theta(y)\} \leq 2\,(\tilde{p}_\theta(y) + \tilde{p}_{\theta_m}(y)),$$

hence the dominated convergence theorem yields

$$\lim_{m\to\infty}|d_K(p,\tilde{p}_{\theta_m}) - d_K(p,\tilde{p}_\theta)| \leq \sum_{n=0}^{K}\int_{\mathbb{R}}\lim_{m\to\infty}|h_{n,\theta_m}(y) - h_{n,\theta}(y)|\,\max\left\{\tilde{p}_{\theta_m}(y),\tilde{p}_\theta(y)\right\}\,dy. \tag{B.7}$$

However, $h_{n,\theta_m}(y)$ is continuous in θ because it depends continuously on the cdf $\tilde{F}_{\theta_m}(y)$ which, in turn, is continuous by Lemma 1. Therefore,

$$\lim_{m\to\infty}|h_{n,\theta_m}(y) - h_{n,\theta}(y)| = 0 \tag{B.8}$$

for any sequence $\{\theta_m\}_{m=1}^{\infty}$ such that $\theta_m \to \theta$. Combining (B.8) with the inequality (B.7) yields $\lim_{m\to\infty}|d_K(p,\tilde{p}_{\theta_m}) - d_K(p,\tilde{p}_\theta)| = 0$ whenever $\theta_m \to \theta$ and completes the proof of Part 1. ■

Proof Differentiability (Part 2).

By Rademacher's theorem (see (Evans, 2018, Theorem 3.2)), if $d_K(p,\tilde{p}_\theta)$ is Lipschitz then it is differentiable almost everywhere. Hence, we aim to prove that $d_K(p,\tilde{p}_\theta)$ is Lipschitz continuous w.r.t. θ.

Lemma 2 yields the upper bound

$$\left|h_{n,\theta}(y_0) - h_{n,\theta'}(y_0)\right| \leq \binom{K}{n} C_{n,K}\,\left|\tilde{F}_\theta(y_0) - \tilde{F}_{\theta'}(y_0)\right|, \tag{B.9}$$

where $C_{n,K} < \infty$ is a constant that depends on n and K. Combining (B.9) with Lemma 3 yields the Lipschitz continuity of $h_{n,\theta}(y_0)$, namely

$$\left|h_{n,\theta}(y_0) - h_{n,\theta'}(y_0)\right| \leq \binom{K}{n} C_{n,K}\,L_1\left\|\theta - \theta'\right\|, \tag{B.10}$$

where $L_1 < \infty$ is a constant. Moreover, we have a bound for the pdf $\tilde{p}_\theta$ of the form

$$\sup_{\theta,y} \tilde{p}_\theta(y) = \sup_{\theta,y} \frac{p_z(g_\theta^{-1}(y))}{\left|g'_\theta\left(g_\theta^{-1}(y)\right)\right|} \leq \frac{\|p_z\|_{L^\infty(\mathbb{R})}}{m}, \tag{B.11}$$

where $g_\theta^{-1}(y)$ is the local inverse of $g_\theta(y)$ and we have used the assumption $\inf_{z,\theta} |g'_\theta(z)| \geq m > 0$. Substituting the upper bounds (B.10) and (B.11) back into (B.6) (with $\theta_m = \theta'$) yields

$$|d_K(p, \tilde{p}_\theta) - d_K(p, \tilde{p}_{\theta'})| \leq \frac{1}{K+1} \frac{\|p_z\|_{L^\infty(\mathbb{R})}}{m} \sum_{n=0}^{K} \binom{K}{n} C_{n,K} L_1 \|\theta - \theta'\|.$$

Finally, we note that $\sum_{n=0}^{K} \binom{K}{n} = 2^K$ to obtain

$$|d_K(p, \tilde{p}_\theta) - d_K(p, \tilde{p}_{\theta'})| \leq \frac{\|p_z\|_{L^\infty(\mathbb{R})}}{m\,(K+1)}\, 2^K \left(\max_{0\leq n\leq K} C_{n,K}\right) L_1 \|\theta - \theta'\|.$$

and complete the proof. ■

Appendix C. Experimental results

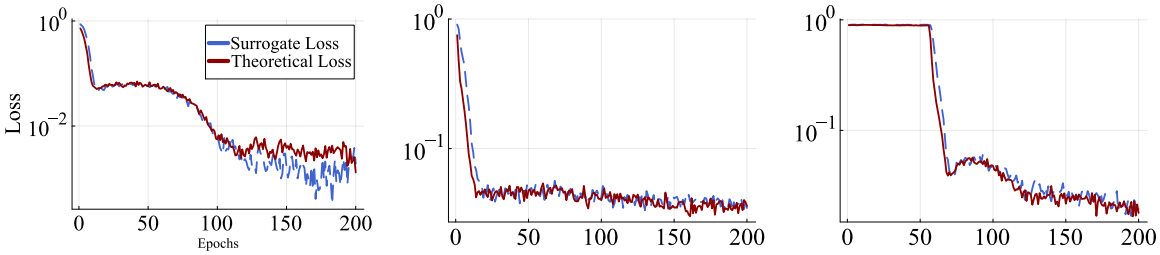
C.1 Comparison of the surrogate and theoretical loss functions

Table 11 compares the error rates between the surrogate and theoretical losses. The metrics displayed include the L_1 and L_∞ norms of the values obtained from the surrogate loss and the theoretical loss at each epoch, along with their respective percentage errors. In this setup the noise is drawn from a standard Gaussian, $\mathcal{N}(0,1)$, the number of training epochs is 1000, $K = 10$, the sample size is $N = 1000$, and the results are averaged over 100 trials. The final three rows in the table describe mixture models with equal component weights: Model 1 is a mixture of $\mathcal{N}(5,2)$ and $\mathcal{N}(-1,1)$, Model 2 is a mixture of $\mathcal{N}(5,2)$, $\mathcal{N}(-1,1)$, and $\mathcal{N}(-10,3)$, and Model 3 is a mixture of $\mathcal{N}(-5,2)$ and Pareto$(5,1)$.

Target	L_1	L_∞	% error L_1	% error L_∞
$\mathcal{N}(4,2)$	0.005307 ± 0.001505	0.254805 ± 0.034452	0.019625	0.930903
$\mathcal{U}(-2,2)$	0.015454 ± 0.001450	0.244568 ± 0.028237	0.015454	0.734944
Cauchy(1,2)	0.005192 ± 0.001329	0.242361 ± 0.028494	0.015636	0.729832
Pareto(1,1)	0.003290 ± 0.004527	0.137377 ± 0.183803	0.000547	0.022837
Model_1	0.004701 ± 0.002198	0.175290 ± 0.025778	0.003925	1.471159
Model_2	0.004991 ± 0.001895	0.173241 ± 0.030297	0.033870	0.117551
Model_3	0.009817 ± 0.002167	0.348440 ± 0.068293	0.016706	0.592932

Table 11: Comparison of error between the surrogate and theoretical losses. Noise is $\sim \mathcal{N}(0,1)$, $K = 10$, epochs $= 1000$, and $N = 1000$. Entries are mean $\pm$ std over 100 trials; last two columns show percentage error.

Figure 13 illustrates the surrogate loss versus the theoretical one across several distributions, presented in log-scale to highlight performance differences.



(a) $\mathcal{N}(4,2)$ (b) Pareto$(1,1)$ (c) Model$_3$

Figure 13: Comparison of the surrogate loss and the theoretical one during training for different distributions. Only the first 200 epochs are shown, and the scale in the vertical axis is logarithmic.

In conclusion, the surrogate loss performs well for different target distributions, as shown in Table 11 and Figure 13. The small L_1 and L_∞ norms with minimal percentage errors indicate its close approximation of the theoretical loss. Even for complex distributions like the mixture models, it maintains low error rates, demonstrating robustness and reliability.

C.2 Empirical evaluation of the order K

This Section empirically studies how the histogram order K affects the ISL surrogate, highlighting the trade-off between discretisation bias at small K and finite-sample variance at large K. We report ablations on (i) loss convergence to a high-resolution reference, (ii) gradient-direction stability, and (iii) finite-sample MSE across representative 1D benchmarks.

Convergence of the surrogate loss. We first evaluate the discretisation bias induced by the order K by comparing the surrogate loss d_K to a high-resolution reference value computed at $K_{\text{ref}} = 256$. We introduce two additional 1D benchmarks:

- *Gaussian mean shift:* the model is $\mathcal{N}(0,1)$ and the target is $\mathcal{N}(\theta,1)$, equivalently $x_{\text{fake}} = \theta + \varepsilon$ with $\varepsilon \sim \mathcal{N}(0,1)$. In the discretisation study we fix $\theta = \theta_0$ (e.g., $\theta_0 = 1$ in our experiments).

- *Symmetric 2-GMM (2-GMM):* the target is $\mu_{\delta_{\text{true}}} = \frac{1}{2}\mathcal{N}(-\delta_{\text{true}},1) + \frac{1}{2}\mathcal{N}(+\delta_{\text{true}},1)$, while the model family is $\mu_\theta = \frac{1}{2}\mathcal{N}(-\theta,1) + \frac{1}{2}\mathcal{N}(+\theta,1)$, where θ controls the component separation. In our experiments we set $\theta = 1$ and $\delta_{\text{true}} = 1.5$.

For all remaining benchmarks, we fix the model to $\mathcal{N}(0,1)$ and vary the target distribution; the corresponding targets are listed in Table 12. A full specification of these targets is provided in Section C.1.

To isolate the deterministic discretization error, we use a large reference sample size ($N_{\text{ref}} = 50{,}000$), which largely suppresses statistical variance and highlights the effect of histogram resolution. As shown in Table 12 and Figure 14, the absolute error $|d_K - d_{K_{\text{ref}}}|$ decays as K increases and empirically follows an approximately $O(K^{-1})$ rate across all benchmarks. This confirms that K acts as a resolution parameter for the underlying rank statistic: increasing K reduces discretisation bias in a predictable manner, at the expense of increased per-iteration computation.

Target	$K = 2$	$K = 4$	$K = 8$	$K = 16$	$K = 32$	$K = 64$	$K = 128$
Gaussian mean	3.35×10^{-1}	1.78×10^{-1}	8.91×10^{-2}	4.33×10^{-2}	2.02×10^{-2}	8.66×10^{-3}	2.89×10^{-3}
2-GMM	1.14×10^{-2}	9.02×10^{-2}	4.37×10^{-2}	2.13×10^{-2}	9.99×10^{-3}	4.28×10^{-3}	1.43×10^{-3}
$N(4, 2)$	4.92×10^{-1}	3.63×10^{-1}	2.05×10^{-1}	9.85×10^{-2}	4.57×10^{-2}	1.96×10^{-2}	6.52×10^{-3}
$\mathcal{U}(-2, 2)$	7.54×10^{-2}	7.43×10^{-2}	3.66×10^{-2}	1.87×10^{-2}	8.67×10^{-3}	3.71×10^{-3}	1.24×10^{-3}
Cauchy$(1, 2)$	1.51×10^{-1}	1.63×10^{-1}	1.10×10^{-1}	5.29×10^{-2}	2.49×10^{-2}	1.06×10^{-2}	3.55×10^{-3}
Pareto$(1, 1)$	4.94×10^{-1}	3.69×10^{-1}	1.81×10^{-1}	8.98×10^{-2}	4.17×10^{-2}	1.79×10^{-2}	5.98×10^{-3}
Model 1	6.92×10^{-2}	1.66×10^{-1}	1.14×10^{-1}	5.41×10^{-2}	2.54×10^{-2}	1.09×10^{-2}	3.64×10^{-3}
Model 2	1.17×10^{-1}	1.87×10^{-1}	1.34×10^{-1}	6.42×10^{-2}	3.05×10^{-2}	1.31×10^{-2}	4.35×10^{-3}
Model 3	6.67×10^{-4}	2.44×10^{-1}	1.80×10^{-1}	8.66×10^{-2}	4.11×10^{-2}	1.76×10^{-2}	5.87×10^{-3}

Table 12: Absolute error $|d_K - d_{K_{\text{ref}}}|$ between the surrogate loss with order K and a high-resolution reference $K_{\text{ref}} = 256$ for all 1D experiments.

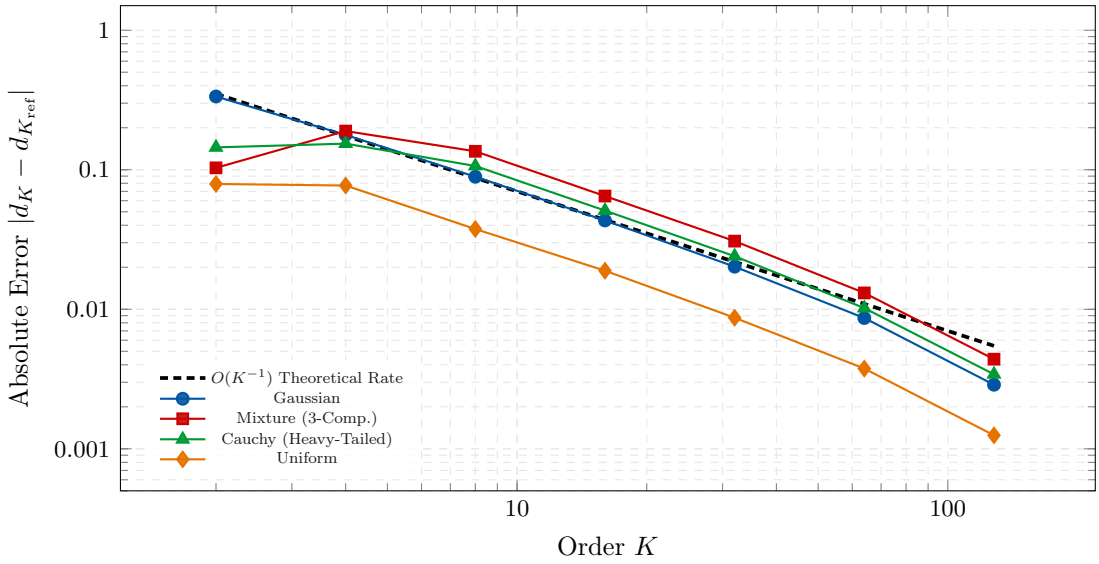


Figure 14: Convergence of ISL across different benchmarks. The alignment with the $O(K^{-1})$ slope (dashed black) demonstrates that K serves as an effective resolution parameter for the estimator, providing consistent convergence properties regardless of the distribution geometry or tail thickness.

Gradient alignment and directional stability. In gradient-based optimization, the surrogate is useful only insofar as it provides an accurate update direction. Accordingly, we focus on the gradient $g_K := \nabla_\theta d_K(\theta)$ rather than on the absolute value of $d_K(\theta)$. In the same setting as the previous experiment, we quantify gradient fidelity by reporting (i) the ℓ_2 error $\|g_K - g_{K_{\text{ref}}}\|_2$ and (ii) the cosine similarity between g_K and a high-resolution reference gradient $g_{K_{\text{ref}}}$, computed using $K_{\text{ref}} = 256$.

As shown in Tables 13–14 and Figure 15, although the divergence estimate can be biased at low resolutions, the *gradient direction* becomes stable at moderate orders: cosine similarities are typically very close to 1 already by $K = 32$ and essentially indistinguishable from the reference by $K = 64$. This indicates that beyond a modest resolution threshold, increasing K primarily rescales the update magnitude without materially changing its direction, so reliable

optimization is compatible with relatively small K even when the scalar loss estimate is still biased.

In the context of gradient-based optimization, the utility of the surrogate is determined by the fidelity of the signal g_K rather than the absolute value of the loss. To assess the reliability of this optimization signal, we measure the L_2 error and the cosine similarity, a scale-invariant metric defined as the normalized inner product between the surrogate gradient g_K and the reference $g_{K_{\text{ref}}}$.

Target	$K=2$	4	8	16	32	64	128
Gaussian mean	0.9090	0.9896	0.9954	0.9979	0.9997	0.9998	1.0000
2-GMM	0.6268	0.9810	0.9926	0.9971	0.9999	0.9999	1.0000
$N(4,2)$	0.8849	0.9260	0.9921	0.9980	0.9998	0.9999	0.9999
$U(-2,2)$	0.9018	0.9767	0.9829	0.9985	0.9983	0.9998	0.9998
Cauchy(1,2)	0.8595	0.9645	0.9942	0.9973	0.9994	0.9999	1.0000
Pareto(1,1)	0.5610	0.8027	0.7694	0.9363	0.9776	0.9987	0.9982
Model 1	0.1208	0.7785	0.9970	0.9956	0.9993	0.9998	0.9999
Model 2	0.4732	0.8118	0.9762	0.9982	0.9996	0.9995	0.9999
Model 3	−0.3793	0.5981	0.9893	0.9943	0.9967	0.9997	0.9998

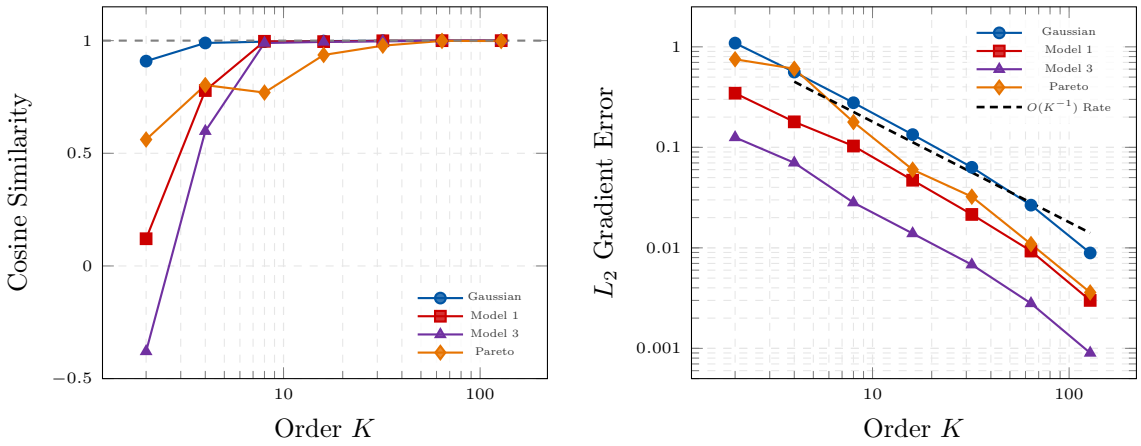
Table 13: Cosine similarity between $\nabla_\theta d_K$ and the reference gradient with $K_{\text{ref}} = 256$ for all 1D experiments. Values close to 1 indicate strong directional alignment even for moderate K.

Target	$K=2$	$K=4$	$K=8$	$K=16$	$K=32$	$K=64$	$K=128$
Gaussian mean	1.09×10^{0}	5.61×10^{-1}	2.78×10^{-1}	1.34×10^{-1}	6.34×10^{-2}	2.66×10^{-2}	8.93×10^{-3}
2-GMM	1.24×10^{-1}	4.59×10^{-1}	2.52×10^{-1}	1.20×10^{-1}	5.64×10^{-2}	2.41×10^{-2}	8.03×10^{-3}
$N(4,2)$	4.62×10^{-1}	2.88×10^{-1}	1.30×10^{-1}	6.19×10^{-2}	2.97×10^{-2}	1.26×10^{-2}	4.22×10^{-3}
$U(-2,2)$	7.68×10^{-1}	3.99×10^{-1}	1.89×10^{-1}	8.37×10^{-2}	3.99×10^{-2}	1.71×10^{-2}	5.70×10^{-3}
Cauchy(1,2)	2.96×10^{-1}	1.19×10^{-1}	5.30×10^{-2}	2.92×10^{-2}	1.30×10^{-2}	5.59×10^{-3}	1.87×10^{-3}
Pareto(1,1)	7.52×10^{-1}	6.06×10^{-1}	1.78×10^{-1}	6.01×10^{-2}	3.24×10^{-2}	1.10×10^{-2}	3.69×10^{-3}
Model_1	3.46×10^{-1}	1.79×10^{-1}	1.03×10^{-1}	4.71×10^{-2}	2.16×10^{-2}	9.35×10^{-3}	3.09×10^{-3}
Model_2	2.52×10^{-1}	1.12×10^{-1}	6.26×10^{-2}	2.93×10^{-2}	1.38×10^{-2}	5.85×10^{-3}	1.96×10^{-3}
Model_3	1.26×10^{-1}	7.02×10^{-2}	2.82×10^{-2}	1.39×10^{-2}	6.89×10^{-3}	2.87×10^{-3}	9.66×10^{-4}

Table 14: Gradient L_2 error $\|g_K - g_{K_{\text{ref}}}\|_2$ between $g_K = \nabla_\theta d_K$ and the reference gradient $g_{\text{ref}} = \nabla_\theta d_{K_{\text{ref}}}$ with $K_{\text{ref}} = 256$. Each row corresponds to one 1D toy experiment; columns vary the surrogate resolution K.

Finite-sample bias–variance trade-off and optimal order. The previous analysis isolates the deterministic discretisation error induced by K at a large sample size N_{ref}. To understand how K should be chosen in practice for finite N, we perform a bias–variance ablation where we treat d_K as a Monte Carlo estimator. For each $N \in \{10^2, 2\cdot10^2, 5\cdot10^2, 10^3, 2\cdot10^3, 5\cdot10^3, 10^4, 2\cdot10^4\}$ and $K \in \{2,4,8,16,32,64,128\}$ we repeat the following 20 times: draw two independent batches of size N, $(x_{\text{real}})_{i=0}^N \sim p$ and $(x_{\text{fake}})_{i=0}^N \sim \tilde{p}$, and compute the corresponding loss d_K. We then estimate, for each pair (N,K), the empirical mean, variance, and mean–squared error (MSE) with respect to a high-resolution reference $d_{K_{\text{ref}}}$ computed with $(N_{\text{ref}}, K_{\text{ref}}) = (5\cdot10^4, 256)$. This experiment is repeated across all 1D benchmarks (Gaussian mean shift, the two Gaussian mixtures and the heavy-tailed Model 3).

The empirical results presented in Figure 16 and Figure 17 reveal a fundamental bias–variance trade-off that dictates the optimal resolution of the ISL estimator. As shown in



(a) Directional Alignment

(b) Magnitude Convergence

Figure 15: Gradient convergence analysis of the ISL surrogate across five distinct benchmarks. (a) Cosine similarity shows rapid directional alignment. (b) The L_2 error norm decay follows the theoretical $O(K^{-1})$ rate.

Figure 16, the MSE curves are consistently U-shaped in K, characterizing the transition between two distinct error regimes. At low values of K, the total error is dominated by the discretisation bias, which follows the $O(K^{-1})$ power-law decay observed in our deterministic analysis. However, as the discretisation becomes increasingly fine, the finite sample size N becomes insufficient to populate the growing number of bins reliably. This causes the Monte Carlo variance to dominate, driving the total error back up. The well-defined minima in these plots represent the optimal order K^* that balances these competing forces for a given sample budget.

This statistical behavior is further illuminated by the stability of the $\sqrt{N}$-scaled standard deviation shown in Figure 17. The fact that these curves remain horizontal across several orders of magnitude of N confirms that for any fixed K, the estimator achieves the standard Monte Carlo convergence rate of $O(N^{-1/2})$. This is important for stochastic training: it shows that, for fixed K, the estimator exhibits standard Monte Carlo scaling $\mathrm{std}(\widehat{d}_K) \propto N^{-1/2}$. Hence increasing the batch size reduces the stochastic fluctuations in a predictable way, and the variance does not display pathological growth as N changes. Furthermore, Figure 16 shows that, as (N) increases, the MSE-minimizing value of (K) shifts toward larger orders. This indicates that larger datasets can sustain higher-resolution surrogates, reducing discretization bias without incurring prohibitive variance. In typical settings with batch sizes on the order of 10^2–10^3, choosing $(K \in [10, 64])$ consistently yields the smallest total error while preserving a highly informative gradient signal. We therefore recommend this range as a robust default in practice, offering a favorable trade-off between computational cost and update-direction fidelity.

C.3 Efficiency gains from progressive K training vs. fixed K

In Table 15, we compare the outcomes of progressively increasing K during training against a fixed K value. After running 100 trials and averaging the results, we found that progressively

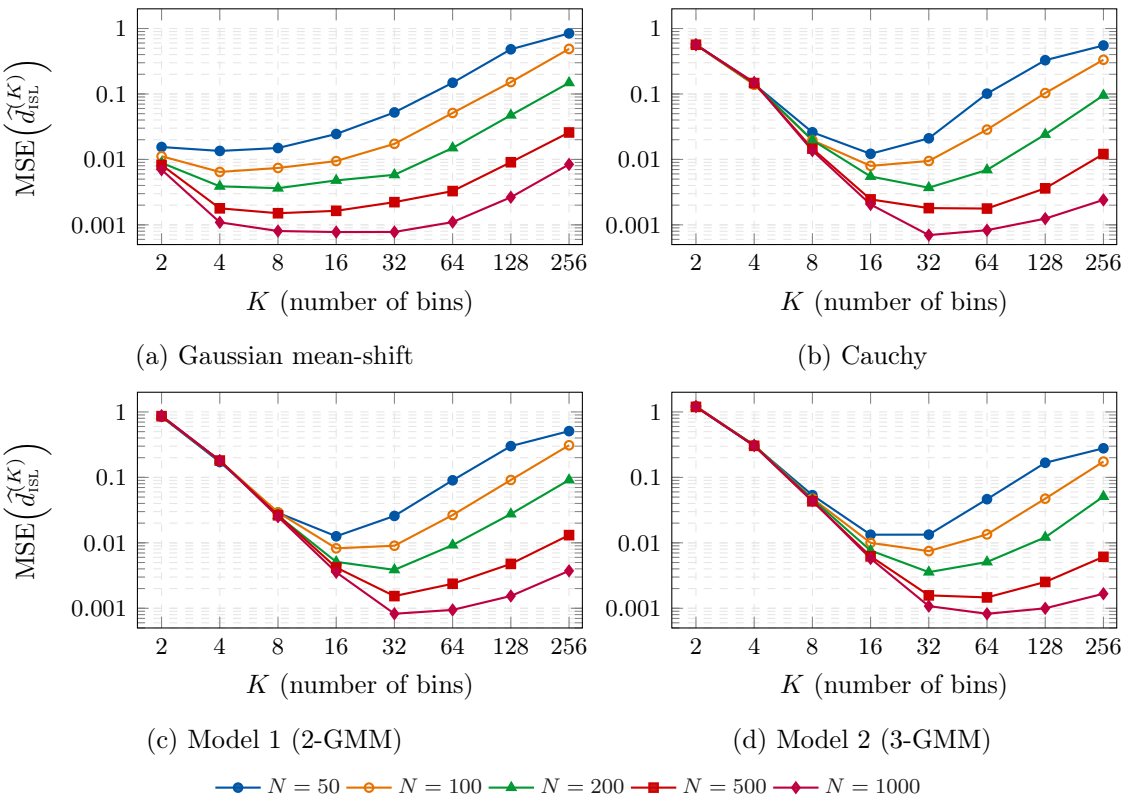
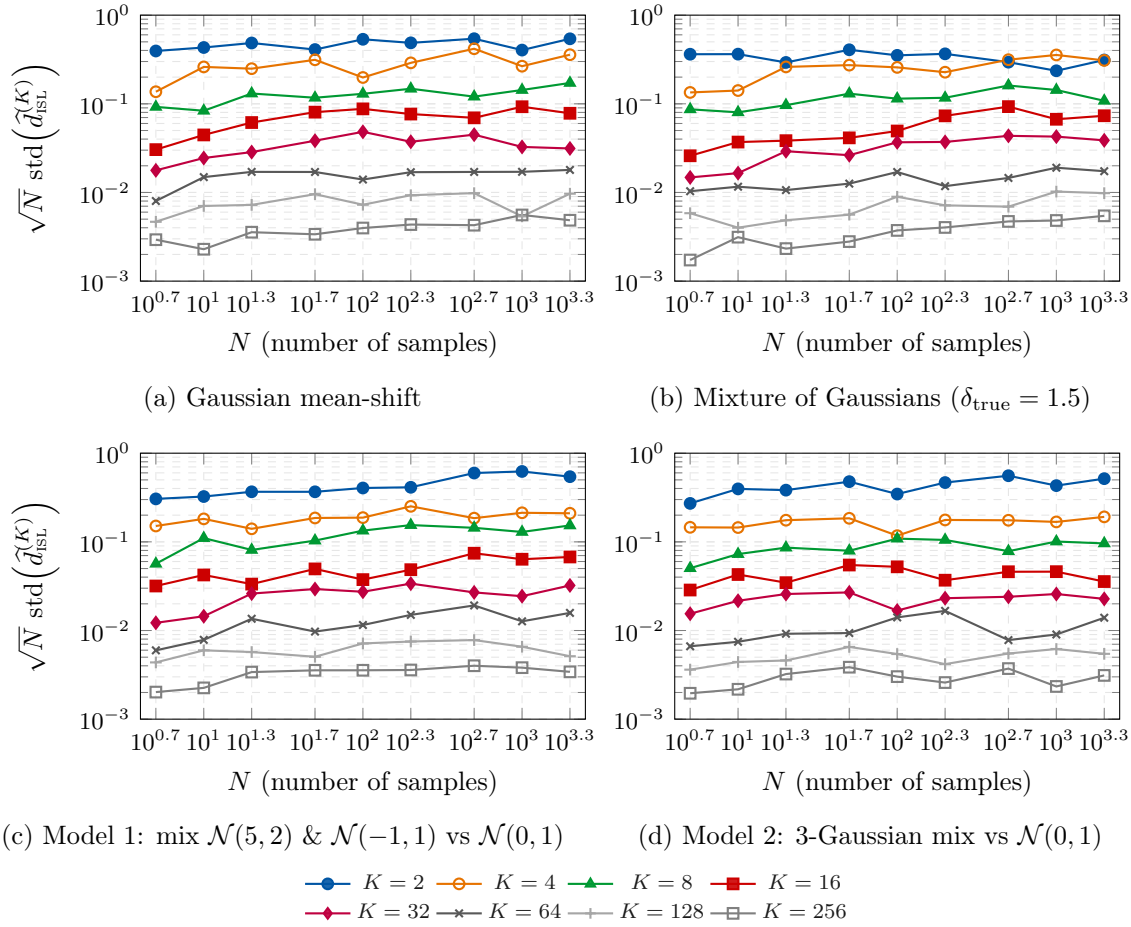


Figure 16: MSE of the paper-like 1D ISL estimator as a function of the number of bins K (log–log scale) across benchmarks: Gaussian mean-shift, Cauchy, Model 1 (2-GMM), and Model 2 (3-GMM).

increasing K reduces training time by up to 50% without sacrificing accuracy. Additionally, the KSD values are comparable or slightly better, indicating that this strategy offers a more efficient balance between time savings and model performance.

	Progressive K		$K = 10$		**Time Imp. %**
	KSD	Time	KSD	Time	Time
$\mathcal{N}(4,2)$	0.00831 ± 0.00268	4149	0.00894 ± 0.00276	9016	53.98
$\mathcal{U}(-2,2)$	0.01431 ± 0.00158	8721	0.01373 ± 0.00152	11745	25.74
Cauchy(1,2)	0.01084 ± 0.00146	3700	0.01190 ± 0.00161	8002	53.76
Pareto(1,1)	0.08344 ± 0.00334	4978	0.24238 ± 0.17257	10800	53.90
Model_1	0.01001 ± 0.00134	8000	0.01175 ± 0.00336	11044	27.56
Model_2	0.01067 ± 0.00270	11042	0.00921 ± 0.00187	8039	27.19
Model_3	0.18525 ± 0.00406	8700	0.18053 ± 0.04512	11207	22.37

Table 15: Comparison of progressive-K training versus fixed $K = 10$. Noise is $\sim \mathcal{N}(0,1)$, $K_{\max} = 10$ (progressive only), epochs $= 200$, and $N = 1000$. Times are in seconds; "KSD" entries are mean $\pm$ std.



(a) Gaussian mean-shift

(b) Mixture of Gaussians ($\delta_{\mathrm{true}} = 1.5$)

(c) Model 1: mix $\mathcal{N}(5,2)$ & $\mathcal{N}(-1,1)$ vs $\mathcal{N}(0,1)$

(d) Model 2: 3-Gaussian mix vs $\mathcal{N}(0,1)$

Figure 17: $\sqrt{N}$-scaled standard deviation of the 1D ISL estimator vs N (log–log). Flat curves indicate std $\propto N^{-1/2}$.

C.4 Empirical evaluation of the slicing complexity m

A central practical question for sliced objectives is how many random projections (slices) are needed for the Monte Carlo approximation in Eq. (4) to be a faithful proxy of its infinite–slicing counterpart. In this section we empirically study the *slicing complexity* as a function of the ambient dimension d, by estimating the minimal number of slices $m_{\min}(d)$ required so that the sliced ISL (computed with m random directions) is within a prescribed relative tolerance of a high–precision reference computed with a large number of directions.

Protocol and definition of $m_{\min}(d)$. Fix a discrepancy scenario (choice of p and $\tilde{p}$), as well as the hyperparameter K and batch size N. We repeat the following procedure for each *ambient* dimension d. For a fixed discrepancy scenario, let p and $\tilde{p}$ denote the resulting distributions on $\mathbb{R}^d$. We then draw two independent batches of size N: $\{x_i\}_{i=1}^N \sim p$ and $\{\tilde{x}_i\}_{i=1}^N \sim \tilde{p}$. We then sample m_{ref} directions $s_1, \dots, s_{m_{\mathrm{ref}}}$ uniformly on the unit sphere, compute the one–dimensional surrogate ISL along each direction once, and define the reference

sliced loss as the average over all m_{ref} directions,

$$\widehat{d}_{(m_{\text{ref}},K)}(p,\tilde{p}) \;:=\; \frac{1}{m_{\text{ref}}}\sum_{i=1}^{m_{\text{ref}}} \widehat{d}_K\big(s_{i\#}p,\; s_{i\#}\tilde{p}\big),$$

where $\widehat{d}_K(\cdot,\cdot)$ denotes the one–dimensional surrogate loss from Section 2.3 applied to the projected samples. For a candidate $m \le m_{\text{ref}}$, we emulate the m–slice estimator by subsampling m directions without replacement and averaging their precomputed one–dimensional values. Repeating this subsampling procedure over multiple trials produces a distribution of relative errors with respect to the reference.

To avoid instability due to occasional unlucky subsamples, we adopt a *quantile criterion*: for a fixed tolerance $\varepsilon > 0$ and quantile level $q \in (0,1)$ (e.g. $q = 0.9$), we define

$$m_{\min}(d) \;:=\; \min\Big\{m \in \{1,\dots,m_{\text{ref}}\} :\; \text{Quantile}_q\Big(\frac{\big|\widehat{d}_{(m,K)}(p,\tilde{p}) - \widehat{d}_{(m_{\text{ref}},K)}(p,\tilde{p})\big|}{\big|\widehat{d}_{(m_{\text{ref}},K)}(p,\tilde{p})\big| + 10^{-12}}\Big) \le \varepsilon\Big\}.$$

In practice, we search m over a dense log–spaced integer grid in $[1, m_{\text{ref}}]$, which yields a reliable estimate of $m_{\min}(d)$ with modest compute.

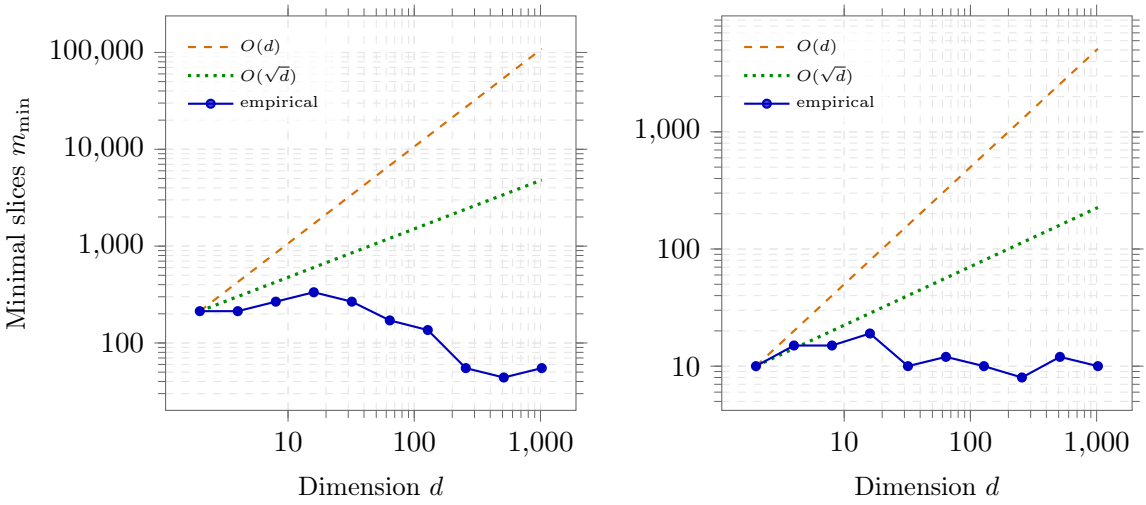
Global features: mean shift and tail heaviness. Figure 18 reports $m_{\min}(d)$ for two global discrepancies. In Fig. 18(a), we consider a global location difference, $p = \mathcal{N}(0, I_d)$ and $\tilde{p} = \mathcal{N}(\delta, I_d)$ with $\|\delta\|_2 = 1$ (implemented as $\delta = (1/\sqrt{d}, \dots, 1/\sqrt{d})$). In Fig. 18(b), we consider a global tail discrepancy with matched first– and second–order moments: $p = \mathcal{N}(0, I_d)$ and $\tilde{p}$ a coordinate–wise Student-t distribution with $\nu = 5$ degrees of freedom, rescaled to have unit variance per coordinate. In both cases, the discrepancy is visible in *most* directions, so the integrand $s \mapsto \widehat{d}_K(s_{\#}p, s_{\#}\tilde{p})$ does not concentrate on an exponentially small subset of the sphere. Empirically, $m_{\min}(d)$ remains modest across dimensions and stays far below linear growth benchmarks such as $O(d)$.

Localized discrepancies and exponential slicing complexity. The situation changes drastically when the discrepancy is *localized* in the sense that only a vanishingly small subset of projection directions on the unit sphere reveals a difference. In Figure 19 we illustrate this phenomenon with a synthetic example in which the discrepancy is detectable only for a small spherical cap of directions. We fix a reference direction $v_\star \in \mathbb{S}^{d-1}$ and a small opening angle $\theta > 0$, and define a spherical cap

$$\mathcal{C}_\theta \;=\; \big\{u \in \mathbb{S}^{d-1} : \langle u, v_\star\rangle \;\ge\; \cos\theta\big\}.$$

Two high-dimensional distributions p and $\tilde{p}$ are then chosen so that their one-dimensional projections coincide for all directions $u \notin \mathcal{C}_\theta$, while they differ in a controlled way only for $u \in \mathcal{C}_\theta$. In other words, the sliced discrepancy is concentrated on the cap $\mathcal{C}_\theta$: outside this tiny region of the sphere, random projections are essentially blind to the difference between p and $\tilde{p}$.

In high dimension, the surface measure of a fixed cap $\mathcal{C}_\theta$ on $\mathbb{S}^{d-1}$ decays exponentially fast with d. Consequently, a sliced estimator that samples directions uniformly at random will, with overwhelming probability, miss the informative cap unless the number of slices m grows exponentially in d. Our experiment makes this effect explicit: for each dimension d, we



(a) Gaussian Mean-Shift (Location)

(b) Tail-Heaviness (Student-t vs. Gaussian)

Figure 18: Empirical scaling of minimal slices $m_{\min}$ for two types of global features. (a) Mean-shift detection and (b) tail-heaviness detection (Student-t with $\nu = 5$).

numerically estimate the minimal number of slices $m_{\min}(d)$ needed for the sliced discrepancy to reliably detect the difference between p and $\tilde{p}$ (e.g., to exceed a fixed threshold with high probability). We observe an exponential scaling $m_{\min}(d) \propto 1.1^d$ over the tested range, consistent with the fact that only a small spherical cap of directions is informative, and illustrating that localized discrepancies can induce exponential slicing complexity.

These highlight that the slicing complexity depends strongly on the *geometry* of the discrepancy: for global shifts or tail differences, a relatively small number of random projections can suffice to stabilize the sliced estimator, whereas for highly localized features the required number of slices can grow exponentially with dimension, reflecting a curse of dimensionality inherent to random projections.

C.5 ISL-Regularized StyleGAN2-ADA on CelebA-64

We train StyleGAN2-ADA (Karras et al., 2020) on 64×64 images from CelebA with horizontal mirroring and ADA enabled, and modify only the generator objective by adding an invariant statistical loss (ISL). Concretely, ISL is evaluated on *downsampled pixel vectors*: we resize each 64×64 image to 32×32 using area downsampling, flatten it, and compare the real and generated batches with a sliced ISL estimator using m projection directions and rank/histogram resolution K.

Because the adversarial and ISL objectives can differ substantially in scale, we combine them using an exponential moving average-based normalization: we track exponential moving averages of the two loss magnitudes and rescale the ISL term accordingly, then mix the normalized terms so that ISL contributes a fraction β of the generator update. Figure 20 (results are averaged over three independent runs with different random seeds.) compares ISL-regularized training to an adversarial-only baseline on CelebA-64. The adversarial run attains strong early fidelity, but later exhibits a less favorable coverage–fidelity trade-off:

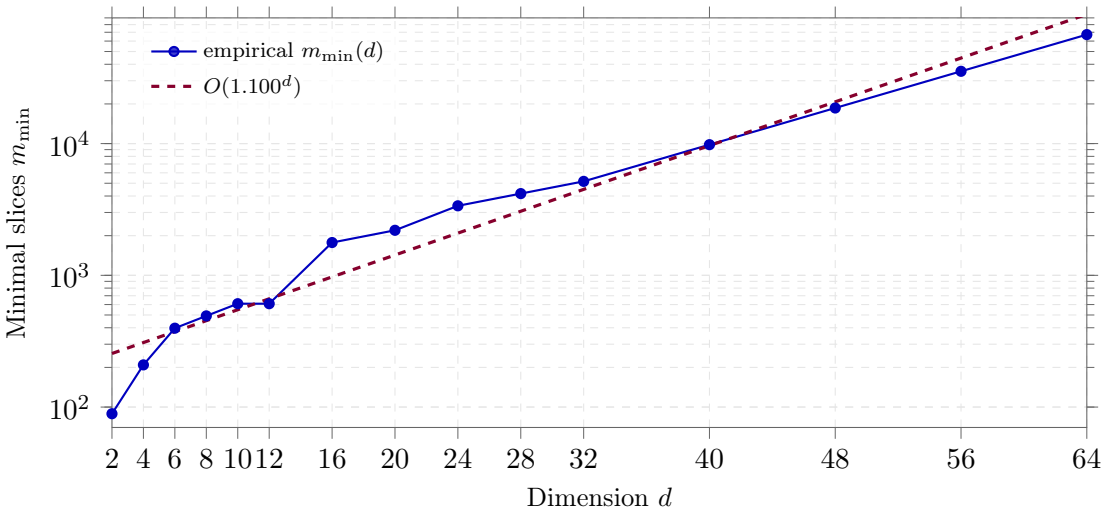
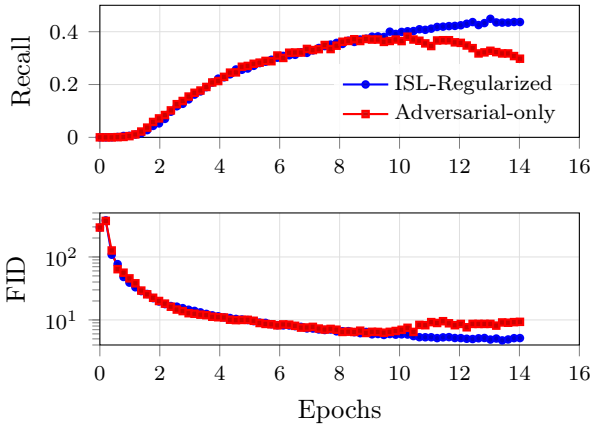


Figure 19: Scaling of $m_{\min}$ for a *localized* discrepancy. Unlike global features, detecting localized discrepancies requires a number of slices that grows exponentially with dimension ($m_{\min} \propto 1.1^d$). This illustrates the curse of dimensionality inherent in random projections when the feature of interest is not aligned with the coordinate axes or is concentrated in a small fraction of the unit sphere.

recall stagnates (and partially decreases) while FID becomes more variable. In contrast, adding ISL (rank resolution $K = 64$ with $m = 1000$ projection directions) yields smoother dynamics, sustaining higher recall and avoiding pronounced FID degradation. Overall, this suggests that the recall-oriented ISL term helps counter coverage loss and stabilizes training.

(a) Generated samples.



(b) Recall/FID dynamics.

Figure 20: CelebA-64 qualitative and quantitative comparison (curves averaged over three seeds). (a) Generated samples. (b) Recall (top) and FID (bottom) versus epoch for ISL-regularized training (K=64, m=1000) and an adversarial-only baseline.

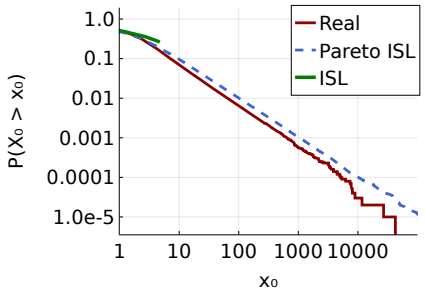
C.6 Comparing ISL and Pareto-ISL for approximating heavy-tailed multi-dimensional distributions

We define multidimensional distributions with heavy-tailed characteristics and train a generator to approximate them. Specifically, we introduce a joint distribution on $[X_0, X_1]$ with the following component definitions

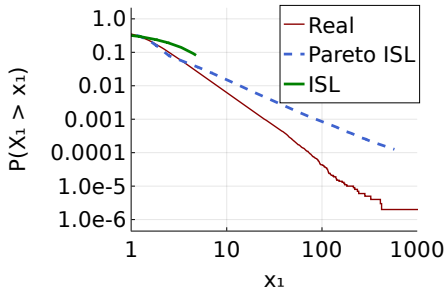
$$X_0 = A + B, \qquad X_1 = \text{sign}(A - B)\,|A - B|^{1/2},$$

where A and B are independent Cauchy r.vs. with location 0.5 and scale 1.0. Note that X_0 and X_1 have different tail indices (1 and $1/2$, respectively) and are not independent.

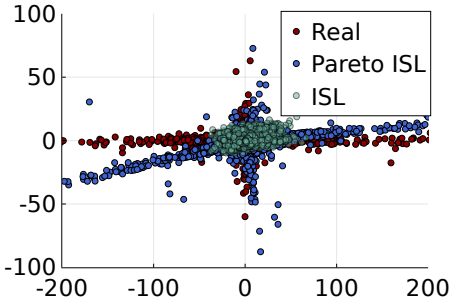
We trained a Pareto-ISL model on this distribution using a NN with an input layer of 2 features, followed by three hidden layers of 256 neurons each with ReLU activations. The input was a mixture of GPD with tail indices of 1 and 0.5. As shown in Figure 21, the marginals and joint distributions closely match the target. We also compared these results to those obtained using ISL with multivariate normal input noise, characterized by a zero mean and identity covariance matrix.



(a) Marginal Distribution along X_0



(b) Marginal Distribution along X_1



(c) Joint Distribution Scatter Plot: Pareto-ISL vs. Gaussian Noise ISL

Figure 21: Plots (a) and (b) show the marginal distributions along the X_0 and X_1 axes for the Pareto-ISL model and ISL with Gaussian noise. Plot (c) presents a scatter plot of 10000 samples from both models, illustrating the joint distribution between the two dimensions.

Appendix D. Experimental Setup

All experiments were run on a MacBook Pro (macOS 13.2.1) with an Apple M1 Pro CPU and 16 GB RAM. For experiments requiring GPU acceleration, we used a single NVIDIA TITAN Xp (12 GB VRAM). The hyperparameter settings for each experiment are reported in the corresponding sections. Our code is publicly available at `https://github.com/josemanuel22/isl-implicit-generative-models`.

References

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International Conference on Machine Learning*, pages 214–223. PMLR, 2017.

Raman Arora, Amitabh Basu, Poorya Mianjy, and Anirbit Mukherjee. Understanding deep neural networks with rectified linear units. *arXiv preprint arXiv:1611.01491*, 2016.

Sanjeev Arora, Rong Ge, Yingyu Liang, Tengyu Ma, and Yi Zhang. Towards principled methods for training generative adversarial networks. In *International Conference on Learning Representations (ICLR)*, 2018.

August A Balkema and Laurens De Haan. Residual life time at great age. *The Annals of Probability*, 2(5):792–804, 1974.

Siddharth Bhatia, Arjit Jain, and Bryan Hooi. Exgan: Adversarial generation of extreme samples. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 6750–6758, 2021.

Nicolas Bonneel, Julien Rabin, Gabriel Peyré, and Hanspeter Pfister. Sliced and Radon Wasserstein barycenters of measures. *Journal of Mathematical Imaging and Vision*, 51: 22–45, 2015.

Haim Brézis. *Functional Analysis, Sobolev Spaces and Partial Differential Equations*, volume 2. Springer, 2011.

Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

Ziyu Chen, Markos Katsoulakis, Luc Rey-Bellet, and Wei Zhu. Sample complexity of probability divergences under group symmetry. In *International Conference on Machine Learning*, pages 4713–4734. PMLR, 2023.

Jinyoung Choi and Bohyung Han. MCL-GAN: Generative adversarial networks with multiple specialized discriminators. *Advances in Neural Information Processing Systems*, 35:29597–29609, 2022.

Stuart Coles, Joanna Bawa, Lesley Trenner, and Pat Dorazio. *An introduction to statistical modeling of extreme values*, volume 208. Springer, 2001.

José Manuel de Frutos, Pablo Olmos, Manuel A. Vázquez, and Joaquín Míguez. Training implicit generative models via an invariant statistical loss. In *International Conference on Artificial Intelligence and Statistics*, pages 2026–2034. PMLR, 2024.

Ishan Deshpande, Ziyu Zhang, and Alexander G Schwing. Generative modeling using the sliced Wasserstein distance. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3483–3491, 2018.

Ishan Deshpande, Yuan-Ting Hu, Ruoyu Sun, Ayis Pyrros, Nasir Siddiqui, Sanmi Koyejo, Zhizhen Zhao, David Forsyth, and Alexander G Schwing. Max-sliced Wasserstein distance and its use for GANs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10648–10656, 2019.

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

Petar M Djuric and Joaquín Míguez. Assessment of nonlinear dynamic models by Kolmogorov–Smirnov statistics. *IEEE transactions on signal processing*, 58(10):5069–5079, 2010.

Ishan Durugkar, Ian Gemp, and Sridhar Mahadevan. Generative multi-adversarial networks. *arXiv preprint arXiv:1611.01673*, 2016.

Víctor Elvira, Joaquín Míguez, and Petar M Djurić. Adapting the number of particles in sequential Monte Carlo methods through an online scheme for convergence assessment. *IEEE Transactions on Signal Processing*, 65(7):1781–1794, 2016.

Víctor Elvira, Joaquín Míguez, and Petar M Djurić. On the performance of particle filters with adaptive number of particles. *Statistics and Computing*, 31:1–18, 2021.

Lawrence Craig Evans. *Measure Theory and Fine Properties of Functions*. Routledge, 2018.

Yanxiang Gong, Zhiwei Xie, Mei Xie, and Xin Ma. Testing generated distributions in GANs to penalize mode collapse. In *International Conference on Artificial Intelligence and Statistics*, pages 442–450. PMLR, 2024.

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

Alexander N Gorban and Ivan Yu Tyukin. Blessing of dimensionality: mathematical foundations of the statistical physics of data. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 376(2118):20170237, 2018.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of Wasserstein GANs. *Advances in Neural Information Processing Systems*, 30, 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

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

Todd Huster, Jeremy Cohen, Zinan Lin, Kevin Chan, Charles Kamhoua, Nandi O Leslie, Cho-Yu Jason Chiang, and Vyas Sekar. Pareto GAN: Extending the representational power of GANs to heavy-tailed distributions. In *International Conference on Machine Learning*, pages 4523–4532. PMLR, 2021.

Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4401–4410, 2019.

Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Training generative adversarial networks with limited data. *Advances in Neural Information Processing Systems*, 33:12104–12114, 2020.

Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. In *International Conference on Learning Representations*, 2014.

Soheil Kolouri, Kimia Nadjahi, Umut Simsekli, Roland Badeau, and Gustavo Rohde. Generalized sliced Wasserstein distances. *Advances in Neural Information Processing Systems*, 32, 2019.

Chun-Liang Li, Wei-Cheng Chang, Yu Cheng, Yiming Yang, and Barnabás Póczos. MMD GAN: Towards deeper understanding of moment matching network. *Advances in Neural Information Processing Systems*, 30, 2017.

Antoine Liutkus, Umut Simsekli, Szymon Majewski, Alain Durmus, and Fabian-Robert Stöter. Sliced-Wasserstein flows: Nonparametric generative modeling via optimal transport and diffusions. In *International Conference on Machine Learning*, pages 4104–4113. PMLR, 2019.

Haoye Lu, Yiwei Lu, Dihong Jiang, Spencer Ryan Szabados, Sun Sun, and Yaoliang Yu. CM-GAN: Stabilizing GAN training with consistency models. In *ICML 2023 Workshop on Structured Probabilistic Inference and Generative Modeling*, 2023.

Yixin Luo and Zhouwang Yang. Dyngan: Solving mode collapse in GANs with dynamic clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2794–2802, 2017.

L Markovich. Light-and heavy-tailed density estimation by gamma-Weibull kernel. In *Conference of the International Society for Non-Parametric Statistics*, pages 145–158. Springer, 2016.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

Shakir Mohamed and Balaji Lakshminarayanan. Learning in implicit generative models. *arXiv preprint arXiv:1610.03483*, 2016.

Kimia Nadjahi, Alain Durmus, Lénaïc Chizat, Soheil Kolouri, Shahin Shahrampour, and Umut Simsekli. Statistical and topological properties of sliced probability divergences. *Advances in Neural Information Processing Systems*, 33:20802–20812, 2020.

Kimia Nadjahi, Alain Durmus, Pierre E Jacob, Roland Badeau, and Umut Simsekli. Fast approximation of the sliced-Wasserstein distance using concentration of random projections. *Advances in Neural Information Processing Systems*, 34:12411–12424, 2021.

Xingqing Nie, Xiaogen Zhou, Zhiqiang Li, Luoyan Wang, Xingtao Lin, and Tong Tong. Logtrans: Providing efficient local-global fusion with transformer and cnn parallel network for biomedical image segmentation. In *2022 IEEE 24th Int Conf on High Performance Computing & Communications; 8th Int Conf on Data Science & Systems; 20th Int Conf on Smart City; 8th Int Conf on Dependability in Sensor, Cloud & Big Data Systems & Application (HPCC/DSS/SmartCity/DependSys)*, pages 769–776. IEEE, 2022.

George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, 22(57):1–64, 2021.

François-Pierre Paty and Marco Cuturi. Subspace robust Wasserstein distances. In *International Conference on Machine Learning*, pages 5072–5081. PMLR, 2019.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

Sidney Resnick and Cătălin Stărică. Smoothing the hill estimator. *Advances in Applied Probability*, 29(1):271–293, 1997.

Murray Rosenblatt. Remarks on a multivariate transformation. *The annals of mathematical statistics*, 23(3):470–472, 1952.

Yunus Saatci and Andrew G Wilson. Bayesian GAN. *Advances in Neural Information Processing Systems*, 30, 2017.

Mehdi SM Sajjadi, Olivier Bachem, Mario Lucic, Olivier Bousquet, and Sylvain Gelly. Assessing generative models via precision and recall. *Advances in Neural Information Processing Systems*, 31, 2018.

Jangwon Seo, Hyo-Seok Hwang, Minhyeok Lee, and Junhee Seok. Stabilized GAN models training with kernel-histogram transformation and probability mass function distance. *Applied Soft Computing*, 164:112003, 2024.

Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of big data*, 6(1):1–48, 2019.

Akash Srivastava, Lazar Valkov, Chris Russell, Michael U Gutmann, and Charles Sutton. Veegan: Reducing mode collapse in GANs using implicit variational learning. *Advances in Neural Information Processing Systems*, 30, 2017.

Vincent Stimper, David Liu, Andrew Campbell, Vincent Berenz, Lukas Ryll, Bernhard Schölkopf, and José Miguel Hernández-Lobato. normflows: A pytorch package for normalizing flows. *Journal of Open Source Software*, 8(86):5361, 2023. doi: 10.21105/joss.05361. URL `https://doi.org/10.21105/joss.05361`.

Masashi Sugiyama, Song Liu, Marthinus Christoffel du Plessis, Masao Yamanaka, Makoto Yamada, Taiji Suzuki, and Takafumi Kanamori. Direct divergence approximation between probability distributions and its applications in machine learning. *Journal of Computing Science and Engineering*, 7(2):99–111, 2013.

Jinxuan Sun, Guoqiang Zhong, Yang Chen, Yongbin Liu, Tao Li, and Kaizhu Huang. Generative adversarial networks with mixture of t-distributions noise for diverse image generation. *Neural Networks*, 122:374–381, 2020.

Yulin Wang, Rui Huang, Shiji Song, Zeyi Huang, and Gao Huang. Not all images are worth 16x16 words: Dynamic transformers for efficient image recognition. *Advances in Neural Information Processing Systems*, 34:11960–11973, 2021.

Magnus Wiese, Robert Knobloch, and Ralf Korn. Copula & marginal flows: Disentangling the marginal from its joint. *arXiv preprint arXiv:1907.03361*, 2019.

Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in Neural Information Processing Systems*, 34:22419–22430, 2021.

Chang Xiao, Peilin Zhong, and Changxi Zheng. Bourgan: Generative networks with metric embeddings. *Advances in Neural Information Processing Systems*, 31, 2018.

Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 11121–11128, 2023.

Xiaoyu Zhang and Jinglin Zhou. A heavy-tailed distribution data generation method based on generative adversarial network. In *2021 IEEE 10th Data Driven Control and Learning Systems Conference (DDCLS)*, pages 535–540. IEEE, 2021.

Peilin Zhong, Yuchen Mo, Chang Xiao, Pengyu Chen, and Changxi Zheng. Rethinking generative mode coverage: A pointwise guaranteed approach. *Advances in Neural Information Processing Systems*, 32, 2019.

Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 11106–11115, 2021.