SCORE-BASED OUTLIER GENERATION WITHOUT RETRAINING VIA LIKELIHOOD REWEIGHTING

Anonymous authors
Paper under double-blind review

ABSTRACT

Outlier generation plays a fundamental role in stress testing, risk analysis, and trustworthy machine learning. Existing methods typically require specialized architectures or training objectives designed specifically for this task. Instead, we show that a score-based diffusion model trained only on regular data can be turned into a controllable outlier generator without retraining or fine-tuning. We adopt recent insights that formulate outlier generation as reweighting the one-dimensional distribution of log-likelihood values induced by the pretrained model, and introduce exponential tilting as a tractable realization of this idea. This yields an explicit scalar modification of the diffusion score that steers sampling toward lower-likelihood regions. We establish a quantitative relationship between the magnitude of the exponential tilting parameter and the resulting outlier strength, and show that the controlled sampling dynamics retain convergence toward the data manifold. Across nine real-world multivariate time-series datasets, our method ranks first or second in all main benchmark comparisons against specialized outlier-generation methods, while preserving marginal, temporal, and cross-feature structure. These results demonstrate that effective outlier generation can be achieved through likelihood reweighting alone, without training a specialized generative model.

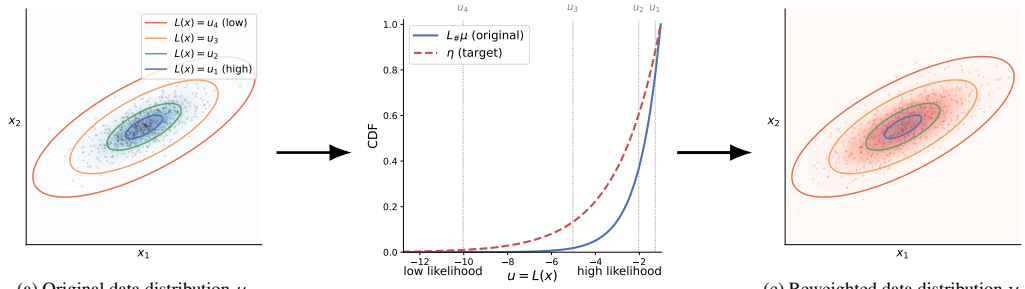
1 INTRODUCTION

Outliers play an important role in stress-testing algorithms and decision-making systems under unlikely but impactful conditions. For example, extreme financial scenarios are essential for evaluating portfolio risk under market shocks (Franses & Van Dijk, 2000). At the same time, synthetic patient records that reflect atypical disease profiles can facilitate the development of robust clinical decision-support systems (Hu et al., 2024). In general, since rare events are inherently underrepresented in training datasets, predictive models trained without consideration of outliers are often deployed in operating regimes that differ substantially from those encountered during training. Motivated by these applications, recent work has identified outlier and rare-event generation as an important problem in generative modelling (Gu et al., 2025).

Existing methods often provide limited control over the statistical properties of the generated outliers and are typically tailored to specific application domains. In particular, they commonly rely on heuristic notions of outlierness, such as reconstruction error, latent-space distance, classifier uncertainty, or adversarial perturbations (Tao et al., 2023; Du et al., 2023), and employ specialized architectures (Huster et al., 2021; Girard et al., 2025) or modified training objectives (Cont et al., 2026; Allouche et al., 2022; Li et al., 2023). This casts outlier generation as a separate learning problem, typically requiring model retraining. In this paper, we instead adopt a likelihood-based definition of outlierness and present an alternative approach based on steering the sampling dynamics of a pretrained generative model toward lower-likelihood regions.

Instead of modifying the high-dimensional data distribution directly, we follow Mukherjee et al. (2026b) and control a one-dimensional object: the distribution of log-likelihood values induced by the data distribution, which we call the *log-likelihood pushforward measure* L. We define outlier distributions through *likelihood reweighting*, requiring their likelihood pushforwards to shift toward lower values, and therefore toward samples that are less likely under the original distribution; the magnitude of this shift is defined by a prescribed Wasserstein-1 separation ρ from L. This viewpoint transforms outlier generation into a 1D distribution-steering problem. We construct such target

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



(a) Original data distribution μ. (b) Likelihood reweighting $L_{\#}\mu \to \eta$. (c) Reweighted data distribution ν.

Figure 1: Likelihood reweighting via disintegration. Colored contours denote likelihood level sets $l(\mathbf{x}) := \log p(\mathbf{x}) = u_i$. Reweighting $L_{\#}\mu$ toward lower likelihoods increases the mass assigned to outer (low-likelihood) level sets while preserving the conditional distributions along them. While μ appears probabilistically concentrated on $l(\mathbf{x}) \geq u_2$, ν appears more spread out and concentrated on $l(\mathbf{x}) \geq u_3$.

distributions using *exponential tilting*, where we reweight the probability of each log-likelihood value $u \sim L$ proportionally to $e^{-\lambda u}$ for a parameter $\lambda > 0$. We show that the resulting distributions shift monotonically toward lower likelihoods and that λ controls the Wasserstein-1 distance to the original log-likelihood distribution L. Figure 1 illustrates the effect of this process on a simple Gaussian distribution.

Having constructed a target outlier distribution through likelihood reweighting, we devise a method aimed at generating samples from the target using a diffusion model (DM) (Ho et al., 2020; Song et al., 2021b). DMs possess several properties that make them particularly attractive for controllable outlier generation. First, they admit tractable likelihood estimation of generated samples via the Fokker-Planck-Kolmogorov (FPK) equation (Lai et al., 2023). Second, the generation process is governed by continuous-time dynamics that can be controlled (Kawar et al., 2022; Chung et al., 2022a; He et al., 2024). We show that likelihood reweighting modifies the diffusion score through a scalar correction along the log-likelihood coordinate, and for exponential tilting, this correction is available in closed form. Substituting it into the reverse-time dynamics yields a simple inference-time controller requiring neither retraining nor fine-tuning. We further show that, under reasonable assumptions, the resulting controlled dynamics retain convergence toward the data manifold, indicating that likelihood steering preserves the model's tendency to generate realistic samples.

Finally, we evaluate the proposed framework on real-world time-series datasets spanning financial, environmental, traffic, and health-related domains. We compare our controlled DM approach against existing generative models designed specifically for extreme-event and outlier generation. The results show that the proposed method produces controllable shifts toward low-likelihood and tail events while remaining competitive in preserving the marginal, temporal, and cross-feature structure of the underlying data. Using a simple control form across all datasets, our method ranks first or second across all main benchmark comparisons, despite requiring no specialized adversarial training or retraining for outlier generation.

2 Related Work

Synthetic extreme-event generation for multivariate time series has received growing attention in finance, climate science, hydrology, and energy systems. The recent survey of Gu et al. (2025) categorizes existing approaches into GAN and VAE-based methods, many of which incorporate extreme value theory (EVT) through specialized latent distributions, sampling strategies, or training objectives. Representative examples include TailGAN (Cont et al., 2026), ParetoGAN (Huster et al., 2021), HTGAN (Girard et al., 2025), EV-GAN (Allouche et al., 2022), and C-DCGAN (Li et al., 2023). Unlike these approaches, our method uses a DM rather than a specialized GAN or VAE, and requires no retraining or outlier-specific training objective. Instead, a model trained only on regular data is converted into an outlier generator by modifying its sampling dynamics. Concurrent work by Mukherjee et al. (2026b) introduced the likelihood-space formulation and the notion of ρ-outlier distributions adopted in this work, but relied on a strong chi-square approximation of the log-likelihood distribution. We build on this formulation using exponential tilting, which applies to a broader class of likelihood distributions and yields both a closed-form score correction and

an interpretable relationship between the control parameter and outlier strength. Furthermore, our method applies to diffusion processes with non-uniform time scales, which are more commonly used in practice.

Recent work has also studied outlier synthesis for improving out-of-distribution (OOD) detection, including recent approaches based on DMs. Methods such as DreamOOD (Du et al., 2023), SONA (Yoon et al., 2025), and BOOD (Liao et al., 2025) use DMs to generate image outliers through text embeddings, semantic guidance, or boundary perturbations, while other methods rely on feature-space sampling or perturbations (Li & Zhang, 2025; Chen et al., 2024; Regmi, 2026). These approaches are designed specifically for OOD detection in image data and rely on image-specific representations and objectives. In contrast, our framework is domain-agnostic: it derives a simple inference-time controller directly from score-based DM theory and applies to any data modality for which a score-based DM is available.

3 Background

3.1 Score-Based Diffusion Models

Score-based DMs (Song et al., 2021b) generate samples by reversing a stochastic differential equation (SDE). In continuous time, the forward process is described by the Ornstein–Uhlenbeck (OU) SDE

$$d\mathbf{x}_t = -b(t)\mathbf{x}_t\,dt + \sqrt{2b(t)}\,d\mathbf{w}_t, \qquad \mathbf{x}_0 \sim p_{\mathrm{data}}, \tag{1}$$

where $b(t) > 0$ is a noise schedule and $\mathbf{w}_t$ denotes Brownian motion. As t increases, the distribution p_t gradually evolves from the data distribution toward a standard Gaussian. Sampling can be performed either by solving the reverse-time SDE (Anderson, 1982; Song et al., 2021b)

$$d\mathbf{x}_t = [-b(t)\mathbf{x}_t - 2b(t)\nabla \log p_t(\mathbf{x}_t)]\,dt + \sqrt{2b(t)}\,d\overline{\mathbf{w}}_t, \tag{2}$$

or equivalently through the probability flow ordinary differential equation (PF-ODE)

$$\dot{\mathbf{x}}_t = -b(t)\left(\mathbf{x}_t + \nabla \log p_t(\mathbf{x}_t)\right), \qquad \mathbf{x}_T \sim \mathcal{N}(0, I), \tag{3}$$

which induces the same marginal distributions while eliminating stochasticity. Due to its simplicity, we will work with the PF-ODE formulation in this paper. In practice, the score function $\nabla \log p_t(\mathbf{x})$ is approximated by a neural network $\mathbf{s}_\theta(\mathbf{x}, t)$ trained using score matching.

3.2 Density Evolution

Under the SDE (1), the evolution of the underlying density is governed by the Fokker–Planck–Kolmogorov (FPK) equation (Bogachev et al., 2015),

$$\frac{\partial p_t}{\partial t} = b(t)\nabla\cdot(\mathbf{x}p_t) + b(t)\Delta p_t. \tag{4}$$

Since our method controls the distribution of log-likelihood values, it is more convenient to work with the corresponding log-density $l_t(\mathbf{x}) := \log p_t(\mathbf{x}_t)$, the evolution of which is governed by yet another PDE (see Proposition 3.1 of Lai et al. (2023)),

$$\frac{\partial l_t}{\partial t} = b(t)\left(\mathbf{x}\cdot\nabla l_t + n + \Delta l_t + \|\nabla l_t\|^2\right). \tag{5}$$

The log-FPK equation will serve as the starting point for our analysis of likelihood evolution under the diffusion process.

4 Exponential Tilting for Outlier Generation

Outlierness admits several application-dependent definitions. In this work, we focus specifically on likelihood-based outliers: samples that occupy lower-density regions of a reference distribution. We build on the distributional formulation of likelihood-based outliers introduced by Mukherjee et al. (2026b), which characterizes outlier distributions through shifts of their induced log-likelihood distributions. We briefly review this formulation in Section 4.1 before introducing our main contribution: an exponential tilting mechanism for constructing such outlier distributions. We show that exponential tilting induces the desired stochastic ordering while providing explicit control over the magnitude of the likelihood shift.

4.1 Problem Formulation and Definition of Outlier

Our approach characterizes outliers through their likelihood under the data distribution. Let μ be a probability measure on $\mathbb{R}^n$ with density p, and define $l(\mathbf{x}) = \log p(\mathbf{x})$. Rather than directly perturbing samples in $\mathbb{R}^n$, we consider the one-dimensional distribution induced by $l(\mathbf{x})$ for $\mathbf{x} \sim \mu$, which we formally denote as $L = (l)_{\#}\mu$, the *log-likelihood pushforward measure*. This allows outlier generation to be formulated as reweighting likelihood levels: probability mass is shifted toward smaller values of $l(\mathbf{x})$. In contrast, the distribution of samples within each likelihood level is left unchanged.

Definition 1 (Likelihood-reweighted measure, Mukherjee et al. (2026b)) *Let η be a probability measure on $\mathbb{R}$ satisfying $\eta \ll L$, where the notation $\ll$ indicates absolute continuity. A probability measure ν on $\mathbb{R}^n$ is a likelihood-reweighted measure of μ with target η if it admits the disintegration*

$$\nu(A) = \int_{\mathbb{R}} \mu(A \mid l(\mathbf{x}) = u)\, \eta(du), \quad A \subset \mathbb{R}^n \text{ Borel},$$

where $\mu(\cdot \mid l(\mathbf{x}) = u)$ is a regular conditional probability, which exists for Borel probability measures on Polish spaces (Leao Jr et al., 2004).

By construction, $l_{\#}\nu = \eta$. Thus, likelihood reweighting changes the distribution with which different likelihood levels are sampled, without altering the geometry of the data within those levels. It remains to specify what properties of η correspond to generating outliers. We use first-order stochastic dominance to encode the direction of the likelihood shift.

Definition 2 (First-order stochastic dominance (FOSD, Shaked & Shanthikumar (2007))) *For probability measures μ and ν on $\mathbb{R}$, we write $\mu \leq_{st} \nu$ if*

$$\mu([x, \infty)) \leq \nu([x, \infty)) \quad \text{for all } x \in \mathbb{R}.$$

Equivalently, in terms of their CDFs,

$$\mu \leq_{st} \nu \quad \Longleftrightarrow \quad F_\mu(u) \geq F_\nu(u) \quad \text{for all } u \in \mathbb{R}.$$

For outlier generation, we therefore require $\eta \leq_{st} L$: samples from η are systematically biased toward lower log-likelihoods than samples from L. Following Mukherjee et al. (2026b), we additionally quantify the magnitude of this shift using the Wasserstein-1 distance.

Definition 3 (ρ-outlier measure) *Let $L = l_{\#}\mu$, and let ν be a likelihood-reweighted measure of μ with target η. We call ν a ρ-*outlier measure *if*

$$\text{(1) } \eta \leq_{st} L, \quad \text{and} \quad \text{(2) } W_1(L, \eta) \geq \rho,$$

This definition separates two properties of the target log-likelihood distribution. Stochastic dominance specifies the *direction* of the shift, while $W_1(L, \eta)$ measures its *magnitude*. To see this explicitly, let F_L^{-1} and F_η^{-1} denote the corresponding quantile functions. In one dimension (Villani, 2021),

$$W_1(L, \eta) = \int_0^1 \left| F_L^{-1}(w) - F_\eta^{-1}(w) \right| dw \underbrace{=}_{\eta \leq_{st} L} \int_0^1 \left(F_L^{-1}(w) - F_\eta^{-1}(w) \right) dw. \tag{6}$$

Hence, ρ provides a direct measure of the average downward displacement of the log-likelihood quantiles. This distributional definition is particularly suited to generative modeling, where the objective is to construct an outlier *distribution* rather than to classify individual samples as outliers.

4.2 Exponential Tilting of Likelihood Distributions

The notion of ρ-outlier in Definition 3 relies on a target log-likelihood distribution η that satisfies stochastic dominance and a prescribed separation from L_0 in the Wasserstein-1 distance. In this section, we construct η by applying exponential tilting to L_0.

Assumption 1 *Let L_0 be a log-likelihood-pushforward measure on $\mathbb{R}$ representing the distribution of log-likelihood values. Assume that L_0 admits finite exponential moments, i.e., there exists $\lambda_{max} > 0$ such that*

$$\int e^{-\lambda u}\, L_0(du) < \infty \quad \textit{for all } \lambda \in [0, \lambda_{\max}).$$

Definition 4 (Exponentially tilted distribution) *Let L_0 be a measure on $\mathbb{R}$ satisfying Assumption 1. Define the exponentially tilted measure L_0^λ of L_0 with tilting parameter λ by*

$$L_0^\lambda(du) := \frac{e^{-\lambda u}}{Z(\lambda)}\, L_0(du), \quad Z(\lambda) := \int e^{-\lambda v}\, L_0(dv). \tag{7}$$

The density ratio between L_0^λ and L_0 is therefore

$$z_0^\lambda(u) = \frac{dL_0^\lambda}{dL_0}(u) = \frac{e^{-\lambda u}}{Z(\lambda)}.$$

We now show that exponential tilting shifts the log-likelihood distribution toward lower values in the sense of stochastic dominance.

Theorem 1 (Stochastic ordering under exponential tilting) *Let L_0^λ be defined by* (7)*. Then, for all $\lambda \in [0, \lambda_{\max})$, $L_0^\lambda \leq_{\mathrm{st}} L_0$.*

The proof of this theorem is in Appendix B.1. The tilt parameter λ determines the magnitude of the controller, whereas Definition 3 quantifies outlier strength through the Wasserstein-1 threshold ρ. We now establish an explicit calibration between these two quantities. The key observation is that the curvature of $\rho(\lambda)$ is determined by the skewness of the tilted log-likelihood distribution.

Theorem 2 (Wasserstein calibration of outlier strength) *Let $U_\lambda \sim L_0^\lambda$, where L_0^λ is defined by* (7) *and define the outlier strength of L_0^λ as $\rho(\lambda) := W_1(L_0, L_0^\lambda)$. Then,*

$$\rho'(\lambda) = \mathrm{Var}(U_\lambda) \geq 0,$$

with strict inequality whenever L_0^λ is not a point mass. Correspondingly, $\rho(\lambda)$ is strictly increasing with respect to λ for non-degenerate log-likelihood distributions. Further, if the first three moments of L_0^λ are finite for $\lambda \in [0, \lambda_{\max})$, then,

$$\rho''(\lambda) = -\mathbb{E}\left[(U_\lambda - \mathbb{E}[U_\lambda])^3\right].$$

In particular, if

$$\mathbb{E}\left[(U_\lambda - \mathbb{E}[U_\lambda])^3\right] \leq 0, \quad \lambda \in [0, \lambda_{\max}),$$

then ρ is convex and

$$\rho(\lambda) \geq \lambda\, \mathrm{Var}(U_0).$$

Thus, any $\lambda \geq \frac{\rho_\star}{\mathrm{Var}(U_0)}$ guarantees that L_0^λ is at Wasserstein-1 distance at least $\rho_\star$ from L_0, provided that $\lambda < \lambda_{\max}$.

The proof of this theorem is in Appendix B.2 and uses the 1-dimensional representation of the W_1 distance under stochastic dominance Villani (2021).

Remark 1 *The negative-skewness assumption is satisfied by several canonical distributions supported on $(-\infty, C]$. For example, it is observed in the reflected chi-square distribution $-\chi_n^2$, reflected exponential distribution $-\mathrm{Exp}(\beta)$, and more generally the reflected Gamma distribution $-\Gamma(\alpha, \beta)$.*

5 Realizing Exponential Tilting in Diffusion Models

Section 4 showed that exponential tilting provides a tractable family of target log-likelihood distributions, and the tilting parameter directly controls outlier strength. We now address an algorithmic question: how can this likelihood reweighting be translated into a modification of the diffusion score?

We first reiterate results from Mukherjee et al. (2026b), characterizing the Radon–Nikodym (RN) structure of likelihood reweighting for a general target distribution η, and showing that its associated score differs from the pretrained score only through a scalar correction along the log-likelihood coordinate. We then specialize this characterization to exponential tilting and construct a time-dependent score controller.

5.1 Score-Based Likelihood Reweighting

We first characterize how likelihood reweighting in the one-dimensional log-likelihood space translates to a modification of the score in the original data space. This provides the key connection between the distributional formulation of Section 4.1 and score-based generative models.

Let μ_t denote the distribution of $\mathbf{x}_t$ under the OU SDE evolution (1), with density p_t, and define $l_t(\mathbf{x}) := \log p_t(\mathbf{x})$, $L_t := (l_t)_{\#}\mu_t$. Suppose that $\eta_t \ll L_t$ is a target distribution over log-likelihood values, and let ν_t denote the corresponding likelihood-reweighted measure.

Theorem 3 (Score in likelihood reweighting, Mukherjee et al. (2026b)) *Let* $z_t(u) := \frac{d\eta_t}{dL_t}(u)$. *Then* $\nu_t \ll \mu_t$ *and*

$$\frac{d\nu_t}{d\mu_t}(\mathbf{x}) = z_t(l_t(\mathbf{x})) = \frac{d\eta_t}{dL_t}(l_t(\mathbf{x})). \tag{8}$$

Moreover, if μ_t *and* ν_t *admit densities* p_t *and* q_t*, respectively, then*

$$q_t(\mathbf{x}) = z_t(l_t(\mathbf{x}))p_t(\mathbf{x}). \tag{9}$$

If, in addition, p_t *is* C^1 *and* z_t *is positive and differentiable, then*

$$\nabla \log q_t(\mathbf{x}) = [1 + c_t(l_t(\mathbf{x}))]\, \nabla \log p_t(\mathbf{x}), \tag{10}$$

where

$$c_t(u) := \partial_u \log z_t(u) = \partial_u \log \frac{d\eta_t}{dL_t}(u). \tag{11}$$

Theorem 3 reduces likelihood reweighting to a scalar modification of the original score. Specifically, the score of ν_t can be written in terms of the pretrained score $\nabla \log p_t$ and a scalar correction determined solely by the current log-likelihood $l_t(\mathbf{x})$. We refer to this scalar correction c_t as the *likelihood controller*.

5.2 Temporal Evolution of the Likelihood Controller

Theorem 3 provides, at each time t, the score associated with a prescribed likelihood-reweighted measure ν_t. This suggests modifying the reference PF-ODE by replacing the pretrained score with its likelihood-reweighted counterpart. The remaining design problem is to choose a time-dependent controller that agrees with the desired endpoint reweighting c_0 while vanishing as the reference diffusion approaches its Gaussian endpoint. To obtain a controller, we propose a dynamic exponential tilting schedule for η_t (see Definition 4). In this case, the controller admits a simple closed form.

Proposition 1 (Dynamic exponential tilt) *Denote* $B_t := \int_0^t b(s)ds$. *Let* $\lambda_t = \lambda e^{-B_t}$. *Let* $\eta_t := L_t^{\lambda_t}$ *denote the exponential tilting of the log-likelihood distribution* L_t *with tilting parameter* λ_t. *Then*

$$\log \frac{dL_t^{\lambda_t}}{dL_t}(u) = -\lambda_t u - \log Z_t(\lambda_t), \quad \textit{where} \quad Z_t(\lambda_t) := \int e^{-\lambda_t v}\, L_t(dv),$$

and therefore,

$$c_t(u) = \partial_u \log \frac{dL_t^{\lambda_t}}{dL_t}(u) = -\lambda_t.$$

Thus, exponential tilting induces the simplest possible likelihood controller: it is independent of the log-likelihood coordinate and depends on time only through the diffusion noise schedule. Our analysis in Section 4.2 establishes stochastic ordering and Wasserstein separation in this setting.

Table 1: Main benchmark results on time series datasets. Each dataset reports Marginal KS and Tail-MSLE$_{95}$. We report diffusion with $\lambda = 0.12$. Lower is better. Results are mean $\pm$ standard deviation over 5 seeds. Best results are bolded and second-best results are underlined.

Model	Weather		Traffic		Illness	
	KS	Tail	KS	Tail	KS	Tail
WGAN	0.240 ± 0.002	8.858 ± 0.028	$\underline{0.193 \pm 0.003}$	$\underline{0.384 \pm 0.002}$	0.502 ± 0.010	1.147 ± 0.087
Vanilla	0.268 ± 0.002	$\underline{8.653 \pm 0.025}$	0.437 ± 0.006	0.870 ± 0.042	0.220 ± 0.005	0.673 ± 0.004
TailGAN	0.670 ± 0.129	9.327 ± 1.297	0.537 ± 0.021	1.307 ± 0.075	0.358 ± 0.002	0.811 ± 0.006
ParetoGAN	0.602 ± 0.000	8.978 ± 0.010	0.561 ± 0.000	1.454 ± 0.000	0.613 ± 0.001	1.839 ± 0.007
HTGAN	$\mathbf{0.114 \pm 0.003}$	9.251 ± 0.031	0.538 ± 0.012	2.736 ± 0.313	$\mathbf{0.190 \pm 0.015}$	$\underline{0.398 \pm 0.047}$
EVGAN	0.940 ± 0.019	320.269 ± 13.950	0.812 ± 0.054	74.557 ± 98.109	0.682 ± 0.003	1.440 ± 0.057
CDCGAN	0.785 ± 0.029	9.516 ± 0.820	0.748 ± 0.031	2.763 ± 1.059	0.761 ± 0.005	1.722 ± 0.142
Diffusion	$\underline{0.214 \pm 0.001}$	$\mathbf{7.154 \pm 0.016}$	$\mathbf{0.130 \pm 0.001}$	$\mathbf{0.185 \pm 0.001}$	$\underline{0.209 \pm 0.003}$	$\mathbf{0.051 \pm 0.002}$
Model	ETTm2		ETTh2		Electricity	
	KS	Tail	KS	Tail	KS	Tail
WGAN	0.179 ± 0.003	$\mathbf{0.335 \pm 0.000}$	0.204 ± 0.005	$\mathbf{0.336 \pm 0.003}$	$\underline{0.224 \pm 0.001}$	$\underline{0.097 \pm 0.000}$
Vanilla	0.206 ± 0.001	0.632 ± 0.004	0.225 ± 0.005	0.632 ± 0.019	0.376 ± 0.026	0.222 ± 0.009
TailGAN	0.743 ± 0.149	1.588 ± 0.026	0.508 ± 0.008	1.321 ± 0.032	0.549 ± 0.005	0.393 ± 0.007
ParetoGAN	0.533 ± 0.007	2.999 ± 0.643	0.528 ± 0.000	2.908 ± 0.022	0.574 ± 0.000	0.445 ± 0.000
HTGAN	$\mathbf{0.076 \pm 0.001}$	0.552 ± 0.043	$\mathbf{0.111 \pm 0.006}$	0.581 ± 0.057	0.327 ± 0.011	0.202 ± 0.066
EVGAN	0.517 ± 0.001	1.861 ± 0.011	0.557 ± 0.071	2.373 ± 1.346	0.582 ± 0.016	56.279 ± 12.656
CDCGAN	0.792 ± 0.024	2.571 ± 0.930	0.804 ± 0.029	2.187 ± 1.471	0.746 ± 0.012	0.667 ± 0.071
Diffusion	$\underline{0.133 \pm 0.001}$	$\underline{0.419 \pm 0.021}$	$\underline{0.161 \pm 0.001}$	$\underline{0.363 \pm 0.023}$	$\mathbf{0.110 \pm 0.001}$	$\mathbf{0.018 \pm 0.000}$
Model	S&P500 Utilities		S&P500 Financials		Exchange Rate	
	KS	Tail	KS	Tail	KS	Tail
WGAN	0.363 ± 0.013	1.308 ± 0.014	0.278 ± 0.008	1.627 ± 0.010	0.211 ± 0.005	0.019 ± 0.001
Vanilla	$\underline{0.229 \pm 0.001}$	1.499 ± 0.004	$\underline{0.225 \pm 0.001}$	1.709 ± 0.008	0.255 ± 0.021	0.029 ± 0.004
TailGAN	0.300 ± 0.003	2.371 ± 0.079	0.286 ± 0.005	2.525 ± 0.103	0.416 ± 0.004	0.050 ± 0.001
ParetoGAN	0.517 ± 0.000	18.715 ± 0.147	0.510 ± 0.000	15.796 ± 0.066	0.540 ± 0.002	0.092 ± 0.002
HTGAN	0.267 ± 0.004	$\underline{1.187 \pm 0.028}$	0.310 ± 0.005	$\underline{1.582 \pm 0.051}$	$\mathbf{0.110 \pm 0.007}$	$\mathbf{0.002 \pm 0.000}$
EVGAN	0.998 ± 0.000	12.084 ± 0.042	0.996 ± 0.000	12.489 ± 0.014	0.592 ± 0.070	4.263 ± 9.436
CDCGAN	0.627 ± 0.008	7.958 ± 0.459	0.627 ± 0.006	8.604 ± 0.354	0.749 ± 0.042	0.122 ± 0.066
Diffusion	$\mathbf{0.054 \pm 0.001}$	$\mathbf{0.074 \pm 0.002}$	$\mathbf{0.049 \pm 0.001}$	$\mathbf{0.099 \pm 0.002}$	$\underline{0.178 \pm 0.003}$	$\underline{0.006 \pm 0.000}$

Furthermore, this choice of η_t satisfies the boundary conditions: $\eta_0 = L_0^\lambda$, and η_t converges to L_t as t increases, thereby converging to the Gaussian log-likelihood distribution. Combining Theorem 3 and Proposition 1 yields the controlled sampling rule used throughout the remainder of the paper:

$$\dot{\mathbf{x}}_t = -b(t)\left[\mathbf{x}_t + \left(1 - \lambda e^{-B_t}\right)\nabla \log p_t(\mathbf{x}_t)\right]. \tag{12}$$

The method therefore requires only a scalar rescaling of the pretrained score at each diffusion time, with no additional networks, likelihood estimation during sampling, or model retraining. Importantly, this modification does not sacrifice the convergence properties of the underlying diffusion sampler, as we show in Theorem 4.

Theorem 4 (Informal, see Appendix B.3 for details) *Let $\mathcal{M}$ denote the data manifold. Under standard manifold projection assumptions (Chung et al., 2022b) and a velocity bound on the projection, the controlled PF-ODE* (12) *converges to $\mathcal{M}$ for $\lambda \in [0, 1)$. In particular,* $\mathrm{dist}(\mathbf{x}_t, \mathcal{M}) = O\left((B_t)^{(1-\lambda)/2}\right)$.

Thus, exponential tilting weakens the normal restoring rate, but does not eliminate manifold attraction while $\lambda < 1$. This result provides an important theoretical safeguard for controllable outlier generation: increasing the rarity of generated samples does not require abandoning convergence toward the learned data geometry. Overall, our controller $c_t(u) = -\lambda e^{-B_t}$ is the exact score correction associated with the time-dependent exponential tilt L_t^λ, and preserves convergence toward the data manifold for $\lambda \in [0, 1)$.

6 Experiments

In this section, we evaluate whether exponential tilting generates increasingly unlikely and extreme samples while preserving distributional fidelity. We further study how λ controls this trade-off and validate the likelihood-space predictions of Theorems 1 and 2. We consider multivariate time-series datasets across multiple domains, and compare against prior generative models implemented in the `ExtremeGANs` package (Mukherjee, 2026), designed for either general-purpose synthesis or extreme-event generation. See Appendix E for more details on reproducibility.

Model Architecture. Following Peebles & Xie (2023), we implement $s_\theta(\mathbf{x}_t, t) \approx \nabla \log p_t(\mathbf{x}_t)$ using a Diffusion Transformer (DiT) architecture without the patchify and patch embedding components. More specifically, we use the implementation by Mukherjee et al. (2026a), which is already designed for time series prediction tasks. Furthermore, we equip the model with causal attention that enforces a strict temporal ordering, ensuring that every event only attends to itself and past events, which makes DiT a natural choice of a model architecture in time-series tasks.

Diffusion Schedule Details. We train the DM with the standard noise-prediction objective and use deterministic DDIM sampling at inference (Song et al., 2021a). We use the standard linear noise schedule implemented by the `Diffusers` HuggingFace library (von Platen et al., 2022). During training, we use 1,000 time steps, and during inference, we use 500 time steps to ensure faster generation. See Appendix C for a derivation of exponential tilting in the discrete time setting.

Datasets. We evaluate our method on real-world multivariate time-series datasets. In particular, we apply our method to data from two sectors of the S&P 500 index, namely Utilities and Financials, where abrupt market movements are important for developing robust trading strategies. We additionally consider the Traffic, National Illness, Electricity, Weather, Exchange-Rate, ETTh2, and ETTm2 benchmarks from the long-term forecasting literature (Wu et al., 2021; Zhou et al., 2021). These datasets span a wide range of temporal dynamics, including periodic patterns, non-stationary behaviour, and rare events, making them suitable benchmarks for evaluating outlier generation. A detailed description of preprocessing steps is provided in Appendix E.

Baselines. We compare against WGAN (Arjovsky et al., 2017), Vanilla GAN (Goodfellow et al., 2014), and the outlier generation methods TailGAN (Cont et al., 2026), ParetoGAN (Huster et al., 2021), HTGAN (Girard et al., 2025), EVGAN (Allouche et al., 2022), and CDCGAN (Li et al., 2023). We do not include diffusion-based outlier generation methods such as DreamOOD (Du et al., 2023), as existing approaches are tailored to image or text-to-image generation and rely on domain-specific components that do not directly extend to multivariate time series. A summary of each method is provided in Appendix D.

Ablation Studies. For each dataset, we evaluate our DM-based method on each metric across different values of the exponential tilting parameter $\lambda \in \{0.0, 0.04, 0.08, 0.12, 0.16, 0.2\}$, with $\lambda = 0.0$ corresponding to uncontrolled diffusion-based generation. As shown in Theorem 2, the outlier strength ρ increases at least proportionally to λ under the negative-skewness assumption. We report our detailed results in Appendix G. In the main results, we report a fixed intermediate value, $\lambda = 0.12$, chosen globally and used for every dataset.

Evaluation metrics. We report marginal Kolmogorov–Smirnov (Huster et al., 2021) for distributional fidelity and Tail-MSLE$_{95}$ (Allouche et al., 2022) for extreme-value quality. Additional metrics, including correlation/autocorrelation error and CCDF area, are reported in Appendices F, G and summarized in Appendix I.

6.1 Main Results

Validation of Theoretical Results. We first validate the theoretical predictions of Section 4. We compute the log-likelihood of each generated sample by numerically solving the log-FPK equation (5) (see Lai et al. (2023) for computational details). Figure 2 shows the resulting empirical CDFs for the S&P500 Financials dataset. As λ increases, the distribution shifts toward lower log-likelihood values, consistent with the stochastic ordering predicted by Theorem 1. The same behaviour is observed across all nine datasets; the corresponding CDFs for the remaining datasets are provided in Appendix H.

We next examine the quantitative relationship between the tilting parameter λ and the resulting outlier strength $\rho(\lambda)$. Figure 3 plots $\rho(\lambda)/\mathrm{Var}_{U\sim L_0}(U)$ against λ for all nine datasets. Theorem 2 predicts that, under the negative skewness assumption, every dataset should lie above the reference line $y = \lambda$. Figure 3 shows that this inequality holds empirically for all nine datasets over the tested range. Together, these results support both the predicted direction of the likelihood shift and the theoretical calibration between λ and outlier strength.

Comparison with Baselines. Table 1 reports the main benchmark results across nine real-world multivariate time-series datasets. Using a single hyperparameter $\lambda = 0.12$ across all datasets without

dataset-specific tuning, our method ranks first or second in all 18 dataset–metric comparisons and achieves the best Tail-MSLE$_{95}$ on six of nine datasets. Meanwhile, marginal KS remains comparable to existing methods, indicating that samples retain distributional fidelity while shifting toward lower-likelihood regions. These results demonstrate that inference-time likelihood control can achieve competitive outlier-generation performance without retraining, architectural changes, or task-specific objectives.

Finally, Appendices F and G provide complementary evaluations on the same datasets and generated samples using additional distributional and temporal metrics. Our exponential tilting method achieves state-of-the-art or second-best results across most datasets in autocorrelation error (ACF), correlation error, Fréchet Gaussian Distance (FGD), and complementary cumulative distribution function (CCDF) area. Collectively, these experiments demonstrate that exponential tilting not only generates controllable low-likelihood samples but also preserves the temporal and statistical structure of the original data.

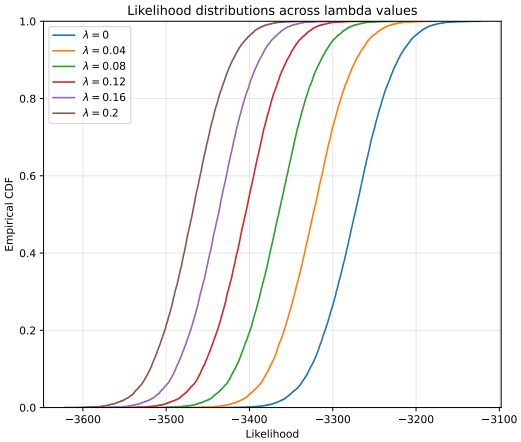


Figure 2: Empirical CDFs of log-likelihood values for different values of λ for the S&P500 financials dataset. Plots for the remaining datasets are in Figure 4

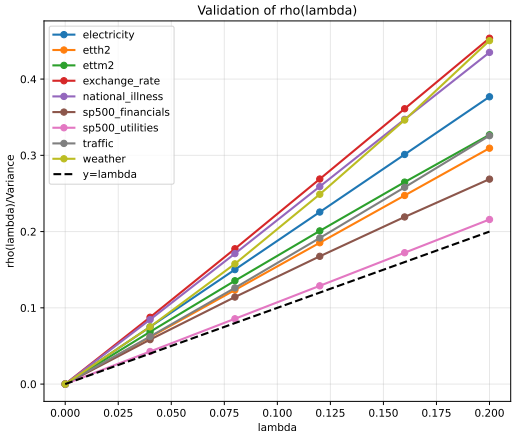


Figure 3: Computations of $\rho(\lambda)/Var_{U\sim L_0}(U)$

7 Conclusion

We introduced a likelihood-based framework for controllable outlier generation with pretrained score-based DMs. Rather than modifying the high-dimensional data distribution directly, we formulate outlier generation through the one-dimensional distribution of log-likelihood values induced by the reference model. Exponential tilting provides a simple mechanism for shifting this log-likelihood distribution toward lower values: the resulting distributions satisfy first-order stochastic dominance, while the tilting parameter λ admits an interpretable calibration of the outlier strength via the Wasserstein displacement of the log-likelihood distribution. The resulting likelihood reweighting induces a scalar correction of the pretrained diffusion score, yielding a simple inference-time controller that requires neither retraining nor an auxiliary model. Across nine multivariate time-series datasets, the proposed method produces shifts toward lower-likelihood regions while remaining competitive with specialized extreme-event generation methods in preserving marginal, temporal, and cross-feature structure. These results suggest that likelihood-space steering provides a useful alternative to treating outlier generation as a separate model-training problem. An important direction for future work is to use the generated outliers as targeted training data for robust decision-making models. In reinforcement learning, for example, controllably generating low-likelihood trajectories could expose policies to rare but consequential scenarios that are poorly represented in the training distribution, potentially improving robustness under distribution shift and extreme events. Additionally, although we establish manifold convergence rates and validate the likelihood shifts empirically, deriving rigorous guarantees on the convergence of the proposed sampling scheme to the target reweighted distribution under suitable distributional metrics remains an area for future research.

References

Michaël Allouche, Stéphane Girard, and Emmanuel Gobet. EV-GAN: Simulation of extreme events with ReLU neural networks. *Journal of Machine Learning Research*, 23(150):1–39, 2022.

Brian D.O. Anderson. Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326, 1982. ISSN 0304-4149. URL https://www.sciencedirect.com/science/article/pii/0304414982900515.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 214–223. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/arjovsky17a.html.

Vladimir I. Bogachev, Nicolai V. Krylov, Michael Röckner, and Stanislav V. Shaposhnikov. *Fokker–Planck–Kolmogorov Equations*, volume 207. Mathematical Surveys and Monographs, 2015.

Valentin De Bortoli. Convergence of denoising diffusion models under the manifold hypothesis. *Transactions on Machine Learning Research*, 2022. ISSN 2835-8856. URL https://openreview.net/forum?id=MhK5aXo3gB. Expert Certification.

Jiankang Chen, Ling Deng, Zhiyong Gan, Wei-Shi Zheng, and Ruixuan Wang. Fodfom: Fake outlier data by foundation models creates stronger visual out-of-distribution detector. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pp. 1981–1990, 2024.

Hyungjin Chung, Jeongsol Kim, Michael Thompson Mccann, Marc Louis Klasky, and Jong Chul Ye. Diffusion posterior sampling for general noisy inverse problems. In *The Eleventh International Conference on Learning Representations*, 2022a.

Hyungjin Chung, Byeongsu Sim, Dohoon Ryu, and Jong Chul Ye. Improving diffusion models for inverse problems using manifold constraints. *Advances in Neural Information Processing Systems*, 35:25683–25696, 2022b.

Rama Cont, Mihai Cucuringu, Renyuan Xu, and Chao Zhang. Tail-gan: Learning to simulate tail risk scenarios. *Management Science*, 72(4):2917–2936, 2026.

Xuefeng Du, Yiyou Sun, Jerry Zhu, and Yixuan Li. Dream the impossible: Outlier imagination with diffusion models. *Advances in Neural Information Processing Systems*, 36:60878–60901, 2023.

Philip Hans Franses and Dick Van Dijk. *Non-linear time series models in empirical finance*. Cambridge university press, 2000.

Stéphane Girard, Emmanuel Gobet, and Jean Pachebat. HTGAN: heavy-tail GAN for multivariate dependent extremes via latent-dimensional control. *International Journal of Computer Mathematics*, pp. 1–22, 2025.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014. URL https://proceedings.neurips.cc/paper_files/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf.

Jingyi Gu, Xuan Zhang, and Guiling Wang. Beyond the norm: A survey of synthetic data generation for rare events. *arXiv preprint arXiv:2506.06380*, 2025.

Yutong He, Naoki Murata, Chieh-Hsin Lai, Yuhta Takida, Toshimitsu Uesaka, Dongjun Kim, Wei-Hsiang Liao, Yuki Mitsufuji, J Zico Kolter, Ruslan Salakhutdinov, et al. Manifold preserving guided diffusion. *International Conference on Learning Representations*, 12, 2024.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/4c5bcfec8584af0d967f1ab10179ca4b-Paper.pdf.

Bing Hu, Ashish Saragadam, Anita Layton, and Helen Chen. Synthetic data from diffusion models improves drug discovery prediction. In *2024 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pp. 6278–6285. IEEE, 2024.

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*, pp. 4523–4532. PMLR, 2021.

Bahjat Kawar, Michael Elad, Stefano Ermon, and Jiaming Song. Denoising diffusion restoration models. *Advances in neural information processing systems*, 35:23593–23606, 2022.

Chieh-Hsin Lai, Yuhta Takida, Naoki Murata, Toshimitsu Uesaka, Yuki Mitsufuji, and Stefano Ermon. FP-Diffusion: Improving score-based diffusion models by enforcing the underlying score Fokker-Planck equation. In *Proceedings of the 40th International Conference on Machine Learning*, June 2023.

Dorival Leao Jr, Marcelo Fragoso, and Paulo Ruffino. Regular conditional probability, disintegration of probability and radon spaces. *Proyecciones (Antofagasta)*, 23(1):15–29, 2004.

Hengzhuang Li and Teng Zhang. Outlier synthesis via hamiltonian monte carlo for out-of-distribution detection. *arXiv preprint arXiv:2501.16718*, 2025.

Yanchun Li, Peng Li, Tianmeng Yang, Zelong Chen, Xuan Song, Yumin Zhao, Jicheng Liu, and Wei Feng. A C-DCGAN-based method for generating extreme risk scenarios of high percentage new energy systems. In *2023 3rd International Conference on Intelligent Power and Systems (ICIPS)*, pp. 588–593. IEEE, 2023.

Qilin Liao, Shuo Yang, Bo Zhao, Ping Luo, and Hengshuang Zhao. BOOD: Boundary-based out-of-distribution data generation. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 37573–37586. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/liao25g.html.

Amartya Mukherjee. ExtremeGANs: A library and benchmark for gan-based rare-event generation, 2026. URL https://github.com/amartyamukherjee/ExtremeGANs.

Amartya Mukherjee, Ruizhi Deng, He Zhao, Yuzhen Mao, Leonid Sigal, and Frederick Tung. ADiff4TPP: Asynchronous diffusion models for temporal point processes. *Transactions on Machine Learning Research*, 2026a. ISSN 2835-8856. URL https://openreview.net/forum?id=bwnZW4wXh4.

Amartya Mukherjee, Tristan Milne, Kry Yik Chau Lui, Stephanie Hazlewood, and Jun Liu. Score-based outlier generation via controlling the radon-nikodym derivative. In *Proceedings of the IEEE Conference on Decision and Control*, 2026b.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4195–4205, 2023.

Sudarshan Regmi. Image-based outlier synthesis with training data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 39338–39350, June 2026.

Moshe Shaked and J George Shanthikumar. *Stochastic orders*. Springer, 2007.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021a. URL https://openreview.net/forum?id=St1giarCHLP.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021b. URL https://openreview.net/forum?id=PxTIG12RRHS.

Leitian Tao, Xuefeng Du, Xiaojin Zhu, and Yixuan Li. Non-parametric outlier synthesis. *arXiv preprint arXiv:2303.02966*, 2023.

Cédric Villani. *Topics in optimal transportation*, volume 58. American Mathematical Soc., 2021.

Patrick von Platen, Suraj Patil, Anton Lozhkov, Pedro Cuenca, Nathan Lambert, Kashif Rasul, Mishig Davaadorj, Dhruv Nair, Sayak Paul, William Berman, Yiyi Xu, Steven Liu, and Thomas Wolf. Diffusers: State-of-the-art diffusion models. `https://github.com/huggingface/diffusers`, 2022.

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.

Suhee Yoon, Sanghyu Yoon, Ye Seul Sim, Sungik Choi, Kyungeun Lee, Hye-Seung Cho, Hankook Lee, and Woohyung Lim. Diffusion-based semantic outlier generation via nuisance awareness for out-of-distribution detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39(12), pp. 13078–13086, 2025.

Guang Zhao, Xihaier Luo, Shinjae Yoo, and Nicole D Jackson. GAN-based extreme conditional distribution estimation for renewable energy systems. In *2024 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*, pp. 562–567. IEEE, 2024.

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(12), pp. 11106–11115, 2021.

A LLM USAGE

LLMs were used to aid and polish writing, but not to generate core research content.

B PROOFS

B.1 STOCHASTIC ORDERING UNDER EXPONENTIAL TILTING

Theorem 1 Let L_0^λ be defined by (7). Then, for all $\lambda \in [0, \lambda_{\max})$,

$$L_0^\lambda \leq_{\mathrm{st}} L_0.$$

Proof: Let $X \sim L_0$, and define

$$f_\lambda(u) := e^{-\lambda u}.$$

Then

$$L_0^\lambda(du) = \frac{f_\lambda(u)}{\mathbb{E}[f_\lambda(X)]}\, L_0(du).$$

First, we prove stochastic dominance. Let

$$g_x(u) := \mathbf{1}_{\{u \leq x\}}.$$

Both f_λ and g_x are decreasing functions of u. Let X' be an independent copy of X. Then

$$\big(f_\lambda(X) - f_\lambda(X')\big)\big(g_x(X) - g_x(X')\big) \geq 0$$

almost surely. Taking expectations and expanding gives

$$\mathbb{E}[f_\lambda(X) g_x(X)] \geq \mathbb{E}[f_\lambda(X)]\mathbb{E}[g_x(X)].$$

Therefore, the CDF of L_0^λ can be bounded as

$$F_\lambda(x) = L_0^\lambda((-\infty, x]) = \frac{\mathbb{E}[f_\lambda(X) g_x(X)]}{\mathbb{E}[f_\lambda(X)]} \geq \mathbb{E}[g_x(X)] = F(x).$$

Hence

$$F_\lambda(x) \geq F(x) \qquad \text{for all } x \in \mathbb{R},$$

which is equivalent to

$$L_0^\lambda \leq_{\mathrm{st}} L_0.$$

■

B.2 BOUNDING ρ GIVEN λ

Lemma 1 *Let L_0 be a probability measure on $\mathbb{R}$, and define its exponentially tilted measure by 4,*

$$dL_0^\lambda(u) = \frac{e^{-\lambda u}}{Z(\lambda)}\, dL_0(u), \qquad Z(\lambda) := \int_{\mathbb{R}} e^{-\lambda u}\, dL_0(u).$$

Let $f : \mathbb{R} \to \mathbb{R}$ be measurable and let $U_\lambda \sim L_0^\lambda$. Suppose that, for λ in an open interval I, the quantities

$$\int_{\mathbb{R}} \big(|f(u)| + |uf(u)| + |u|\big) e^{-\lambda u}\, dL_0(u)$$

are finite and locally uniformly bounded in $\lambda \in I$. Then

$$\frac{d}{d\lambda}\mathbb{E}[f(U_\lambda)] = -\operatorname{Cov}\big(f(U_\lambda), U_\lambda\big).$$

Proof: By definition of L_0^λ,

$$\mathbb{E}[f(U_\lambda)] = \frac{\int_{\mathbb{R}} f(u) e^{-\lambda u}\, dL_0(u)}{\int_{\mathbb{R}} e^{-\lambda u}\, dL_0(u)}.$$

Define

$$N_f(\lambda) := \int_{\mathbb{R}} f(u) e^{-\lambda u}\, dL_0(u).$$

The assumed integrability conditions justify differentiation under the integral sign, yielding

$$N_f'(\lambda) = -\int_{\mathbb{R}} u f(u) e^{-\lambda u}\, dL_0(u)$$

and

$$Z'(\lambda) = -\int_{\mathbb{R}} u e^{-\lambda u}\, dL_0(u).$$

Applying the quotient rule gives

$$\begin{aligned}
\frac{d}{d\lambda}\mathbb{E}[f(U_\lambda)] &= \frac{N_f'(\lambda)Z(\lambda) - N_f(\lambda)Z'(\lambda)}{Z(\lambda)^2} \\
&= -\frac{\int_{\mathbb{R}} u f(u) e^{-\lambda u}\, dL_0(u)}{Z(\lambda)} + \frac{\int_{\mathbb{R}} f(u) e^{-\lambda u}\, dL_0(u)}{Z(\lambda)} \frac{\int_{\mathbb{R}} u e^{-\lambda u}\, dL_0(u)}{Z(\lambda)} \\
&= -\mathbb{E}[U_\lambda f(U_\lambda)] + \mathbb{E}[f(U_\lambda)]\mathbb{E}[U_\lambda] \\
&= -\operatorname{Cov}\big(f(U_\lambda), U_\lambda\big).
\end{aligned}$$

■

Theorem 2 Let $U_\lambda \sim L_0^\lambda$, where L_0^λ is defined by (7) and define the outlier strength of L_0^λ as $\rho(\lambda) := W_1(L_0, L_0^\lambda)$. Then,

$$\rho'(\lambda) = \operatorname{Var}(U_\lambda) \geq 0,$$

with strict inequality whenever L_0^λ is non-degenerate. Correspondingly, $\rho(\lambda)$ is strictly increasing with respect to λ. Furthermore, assume that the first three moments of L_0^λ are finite for $\lambda \in [0, \lambda_{\max})$. Then,

$$\rho''(\lambda) = -\mathbb{E}\left[\big(U_\lambda - \mathbb{E}[U_\lambda]\big)^3\right].$$

In particular, if

$$\mathbb{E}\left[\big(U_\lambda - \mathbb{E}[U_\lambda]\big)^3\right] \leq 0, \qquad \lambda \in [0, \lambda_{\max}),$$

then ρ is convex and

$$\rho(\lambda) \geq \lambda \operatorname{Var}(U_0).$$

Thus, any $\lambda \geq \frac{\rho_\star}{\operatorname{Var}(U_0)}$ guarantees that L_0^λ is at Wasserstein-1 distance at least $\rho_\star$ from L_0, provided that $\lambda < \lambda_{\max}$.

Proof: Define

$$\rho(\lambda) := W_1(L_0, L_0^\lambda) = \mathbb{E}[U_0] - \mathbb{E}[U_\lambda],$$

and the expression for the Wasserstein-1 distance is taken from Equation (6). From Lemma 1, for any sufficiently integrable function f, differentiation under the tilted measure gives

$$\frac{d}{d\lambda}\mathbb{E}[f(U_\lambda)] = -\operatorname{Cov}\big(f(U_\lambda), U_\lambda\big).$$

Applying this identity to $f(u) = u$ yields

$$\rho'(\lambda) = -\frac{d}{d\lambda}\mathbb{E}_{U\sim L_0^\lambda}[U] = \operatorname{Var}_{U\sim L_0^\lambda}(U) \geq 0.$$

Therefore $\rho(\lambda)$ is nondecreasing in λ. If L_0 is not a point mass, then $\operatorname{Var}_{L_0^\lambda}(U)$ is strictly positive, and hence $\rho(\lambda)$ is strictly increasing. From Lemma 1, for any sufficiently integrable function f, differentiation under the tilted measure gives

$$\frac{d}{d\lambda}\mathbb{E}[f(U_\lambda)] = -\operatorname{Cov}\big(f(U_\lambda), U_\lambda\big).$$

Applying this identity to the first and second moments yields

$$\frac{d}{d\lambda}\operatorname{Var}(U_\lambda) = -\mathbb{E}\left[(U_\lambda - \mathbb{E}[U_\lambda])^3\right].$$

Thus,

$$\rho''(\lambda) = -\mathbb{E}\left[(U_\lambda - \mathbb{E}[U_\lambda])^3\right].$$

Under the stated negative-skewness condition, $\rho''(\lambda) \geq 0$, so ρ is convex. Since

$$\rho(0) = 0, \qquad \rho'(0) = \operatorname{Var}(U_0),$$

the tangent-line inequality for convex functions gives

$$\rho(\lambda) \geq \rho(0) + \lambda\rho'(0) = \lambda \operatorname{Var}(U_0).$$

■

B.3 Manifold Preservation under Exponential Tilting

Let p_{data} be supported on a compact k-dimensional submanifold $\mathcal{M} \subset \mathbb{R}^n$. Consider the OU forward process (1) and define

$$B(t) = \int_0^t b(s)\, ds, \qquad m_t = e^{-B(t)}, \qquad \sigma_t^2 = 1 - e^{-2B(t)}. \tag{13}$$

For $t > 0$, the marginal law p_t admits a smooth density. Following the manifold-hypothesis analysis of DMs (Bortoli, 2022; Chung et al., 2022b), the score $\nabla \log p_t$ may blow up to the order of σ_t^{-m} with $m < 2$ as $t \downarrow 0$, with the dominant singular component pointing toward the data manifold in the normal direction.

We study the exponentially tilted PF-ODE

$$\dot{\mathbf{x}}_t = -b(t)\left[\mathbf{x}_t + (1 - \lambda m_t)\nabla \log p_t(\mathbf{x}_t)\right], \qquad 0 \leq \lambda < 1, \tag{14}$$

run backward from T to 0. The goal of this section is to show that the exponential tilting controller slows down, but does not destroy, the tendency of the reverse dynamics to approach the data manifold.

Definition 5 (Tweedie's estimate) *For each $t > 0$, define the Tweedie map*

$$Q_t(\mathbf{x}) := \frac{\mathbf{x} + \sigma_t^2 \nabla \log p_t(\mathbf{x})}{m_t}. \tag{15}$$

Along a trajectory $\mathbf{x}_t$, we define the Tweedie's estimate $\mathbf{x}_{0|t}$ of the clean sample $\mathbf{x}_0$ conditioned on the noisy sample $\mathbf{x}_t$ under the OU SDE (1) *as*

$$\mathbf{x}_{0|t} := Q_t(\mathbf{x}_t) = \frac{\mathbf{x}_t + \sigma_t^2 \nabla \log p_t(\mathbf{x}_t)}{m_t}. \tag{16}$$

Under the setting of (Chung et al., 2022b; He et al., 2024), $\mathbf{x}_{0|t} \in \mathcal{M}$ for all t. Equivalently, the score function has the form

$$\nabla \log p_t(\mathbf{x}_t) = -\frac{\mathbf{x}_t - m_t \mathbf{x}_{0|t}}{\sigma_t^2}. \tag{17}$$

The map Q_t can be interpreted as a denoising map from the noisy sample $\mathbf{x}_t$ to a clean estimate on the data manifold. Under the ideal manifold setting, Proposition 2 of Chung et al. (2022b) states that the corresponding discrete Tweedie map sends noisy samples to $\mathcal{M}$, has Jacobian mapping into the tangent space, and behaves locally as an orthogonal projection onto the data manifold. Motivated by this result, we assume throughout this section that

$$\mathbf{x}_{0|t} = Q_t(\mathbf{x}_t) \in \mathcal{M} \qquad \text{for all } t.$$

Define the residual

$$\mathbf{r}_t := \mathbf{x}_t - m_t \mathbf{x}_{0|t}. \tag{18}$$

Then $\mathbf{r}_t$ measures the discrepancy between the current noisy state $\mathbf{x}_t$ and the scaled Tweedie reconstruction $m_t \mathbf{x}_{0|t}$. By (17), this is precisely the component responsible for the singular part of the score near the data manifold.

Assumption 2 (Tweedie retraction regularity) *For all t, the Tweedie map Q_t is a C^1 retraction onto $\mathcal{M}$ in a neighbourhood of the trajectory $\{\mathbf{x}_t\}$. In particular,*

$$Q_t(\mathbf{x}) \in \mathcal{M}, \qquad DQ_t(\mathbf{x}) : \mathbb{R}^n \to T_{Q_t(\mathbf{x})}\mathcal{M}.$$

For the stronger manifold convergence result, we additionally assume that the denoised trajectory varies at the natural OU time scale: there exists $C > 0$ such that

$$\|\dot{\mathbf{x}}_{0|t}\| \le C b(t) \tag{19}$$

for all t.

Remark 2 *The first part of Assumption 2 is the continuous-time analogue of the projection property established for the discrete Tweedie map in Chung et al. (2022b). The additional velocity bound* (19) *is only needed to control the full residual $\mathbf{r}_t$. Intuitively, it prevents the Tweedie reconstruction from moving along $\mathcal{M}$ faster than the natural time scale of the PF-ODE.*

Theorem 4 (Convergence to the data manifold) *Assume the Tweedie reconstruction satisfies the manifold constraint and the velocity bound in Assumption 2 for all t. Then the full residual satisfies*

$$\|\mathbf{r}_t\| = O\Big(B(t)^{\frac{1-\lambda}{2}}\Big) = O\big(\sigma_t^{1-\lambda}\big), \qquad t \downarrow 0. \tag{20}$$

Thus,

$$\operatorname{dist}(\mathbf{x}_t, \mathcal{M}) = O\Big(B(t)^{\frac{1-\lambda}{2}}\Big), \tag{21}$$

and hence

$$\lim_{t \downarrow 0} \operatorname{dist}(\mathbf{x}_t, \mathcal{M}) = 0.$$

Proof: Using (17), the tilted PF-ODE can be written as

$$\dot{\mathbf{x}}_t = -b(t) \left[\mathbf{x}_t - (1 - \lambda m_t) \frac{\mathbf{r}_t}{\sigma_t^2} \right]. \tag{22}$$

Differentiating $\mathbf{r}_t = \mathbf{x}_t - m_t \mathbf{x}_{0|t}$ gives us the reverse-time ODE

$$\begin{aligned}
\dot{\mathbf{r}}_t &= \dot{\mathbf{x}}_t - \dot{m}_t \mathbf{x}_{0|t} - m_t \dot{\mathbf{x}}_{0|t} \\
&= -b(t) \left[\mathbf{x}_t - (1 - \lambda m_t) \frac{\mathbf{r}_t}{\sigma_t^2} \right] + b(t) m_t \mathbf{x}_{0|t} - m_t \dot{\mathbf{x}}_{0|t} \\
&= -b(t) \left[\mathbf{x}_t - (1 - \lambda m_t) \frac{\mathbf{r}_t}{\sigma_t^2} \right] + b(t)[\mathbf{x}_t - \mathbf{r}_t] - m_t \dot{\mathbf{x}}_{0|t} \\
&= b(t) \left[\frac{1 - \lambda m_t}{\sigma_t^2} - 1 \right] \mathbf{r}_t - m_t \dot{\mathbf{x}}_{0|t}.
\end{aligned}$$

Since $\sigma_t^2 = 1 - m_t^2$, this becomes

$$\dot{\mathbf{r}}_t = a(t)\mathbf{r}_t - m_t\dot{\mathbf{x}}_{0|t}, \qquad a(t) := b(t)\frac{m_t^2 - \lambda m_t}{1 - m_t^2}. \tag{23}$$

We first solve the homogeneous part of the ODE,

$$\dot{\widehat{\mathbf{r}}}_t = a(t)\widehat{\mathbf{r}}_t. \tag{24}$$

Using $m_t = e^{-B(t)}$ and $dm_t = -b(t)m_t\,dt$, we obtain that

$$\begin{aligned}\int a(t)\,dt &= \int b(t)\frac{m_t^2 - \lambda m_t}{1 - m_t^2}\,dt \\ &= \frac{1}{2}\log(1 - m_t^2) + \lambda\,\operatorname{arctanh}(m_t) + C.\end{aligned} \tag{25}$$

Solving the scalar linear equation (24) gives

$$\widehat{\mathbf{r}}_t = C_T(1 - m_t^2)^{1/2}\left(\frac{1 + m_t}{1 - m_t}\right)^{\lambda/2}, \tag{26}$$

where $C_T = e^C$ depends on the terminal condition. We return to the non-homogeneous equation

$$\dot{\mathbf{r}}_t = a(t)\mathbf{r}_t - m_t\dot{\mathbf{x}}_{0|t}, \qquad a(t) = b(t)\frac{m_t^2 - \lambda m_t}{1 - m_t^2}.$$

Let $\Phi(t)$ denote the homogeneous solution factor obtained in (26):

$$\Phi(t) := (1 - m_t^2)^{1/2}\left(\frac{1 + m_t}{1 - m_t}\right)^{\lambda/2}. \tag{27}$$

Then $\Phi(t) = \Theta\Big(B(t)^{\frac{1-\lambda}{2}}\Big)$. By variation of constants,

$$\mathbf{r}_t = \Phi(t)\left[\frac{\mathbf{r}_T}{\Phi(T)} - \int_t^T \Phi(s)^{-1}m_s\dot{\mathbf{x}}_{0|s}\,ds\right]. \tag{28}$$

Using $m_s \le 1$ and the velocity bound $\|\dot{\mathbf{x}}_{0|s}\| \le Cb(s)$, we obtain

$$\|\mathbf{r}_t\| \le \Phi(t)\left\|\frac{\mathbf{r}_T}{\Phi(T)}\right\| + C\Phi(t)\int_t^T \Phi(s)^{-1}b(s)\,ds. \tag{29}$$

For all s,

$$\Phi(s)^{-1} = \Theta\Big(B(s)^{-\frac{1-\lambda}{2}}\Big).$$

Using the change of variables $u = B(s)$, $du = b(s)\,ds$, we get

$$\int_0^T \Phi(s)^{-1}b(s)\,ds \lesssim \int_0^{B(T)} u^{-\frac{1-\lambda}{2}}\,du < \infty,$$

because $0 \le \lambda < 1$. Therefore

$$\|\mathbf{r}_t\| = O(\Phi(t)) = O\Big(B(t)^{\frac{1-\lambda}{2}}\Big).$$

Finally,

$$\mathbf{x}_t = m_t\mathbf{x}_{0|t} + \mathbf{r}_t, \qquad \mathbf{x}_{0|t} \in \mathcal{M}.$$

Since $\mathcal{M}$ is compact, there exists $R > 0$ such that $\|\mathbf{x}\| \le R$ for all $\mathbf{x} \in \mathcal{M}$. Hence

$$\begin{aligned}\operatorname{dist}(\mathbf{x}_t, \mathcal{M}) &\le \|\mathbf{x}_t - \mathbf{x}_{0|t}\| \\ &\le |1 - m_t|\|\mathbf{x}_{0|t}\| + \|\mathbf{r}_t\| \\ &\le R|1 - m_t| + O\Big(B(t)^{\frac{1-\lambda}{2}}\Big).\end{aligned}$$

Since $1 - m_t = 1 - e^{-B(t)} = O(B(t))$, and

$$B(t) = o\Big(B(t)^{\frac{1-\lambda}{2}}\Big) \qquad (0 \le \lambda < 1),$$

we conclude

$$\operatorname{dist}(\mathbf{x}_t, \mathcal{M}) = O\Big(B(t)^{\frac{1-\lambda}{2}}\Big).$$

Thus $\operatorname{dist}(\mathbf{x}_t, \mathcal{M}) \to 0$. If $\mathbf{x}_t \to \mathbf{x}_0$ as $t \downarrow 0$, then by closedness of the compact manifold $\mathcal{M}$, we have $\mathbf{x}_0 \in \mathcal{M}$. ■

Remark 3 *The restriction $\lambda < 1$ is exactly the condition ensuring that the manifold-attracting singularity of the score remains strong enough near $t = 0$. Larger λ weakens the normal restoring force induced by the score, while $\lambda < 1$ preserves a positive decay exponent $(1 - \lambda)/2$.*

C Outlier Generation with DDIM Sampling

This section relates the continuous-time probability-flow formulation to DDIM and shows that exponential-tilting-based outlier generation requires only a scalar modification of the standard DDIM noise prediction. In particular, the proposed controller can be implemented without retraining the DM.

C.1 Continuous-time and DDIM correspondence

Consider the variance-preserving Ornstein–Uhlenbeck forward process

$$dx_t = -b(t)x_t\,dt + \sqrt{2b(t)}\,dw_t, \qquad b(t) \ge 0. \tag{30}$$

Define

$$B(t) := \int_0^t b(s)\,ds.$$

Its conditional marginal distribution is

$$x_t|x_0 \sim \mathcal{N}\left(e^{-B(t)}x_0, \left(1 - e^{-2B(t)}\right) I\right). \tag{31}$$

The associated probability-flow ODE is

$$\dot{x}_t = -b(t)\left[x_t + \nabla \log p_t(x_t)\right]. \tag{32}$$

The DDIM forward marginal is written as

$$x_\tau = \sqrt{\bar{\alpha}_\tau}\,x_0 + \sqrt{1 - \bar{\alpha}_\tau}\,\epsilon, \qquad \epsilon \sim \mathcal{N}(0, I), \tag{33}$$

where $\tau \in \{0, \dots, N\}$. Comparing (31) and (33) gives

$$\sqrt{\bar{\alpha}_\tau} = e^{-B(t)} = \exp\left(-\int_0^t b(s)\,ds\right), \tag{34}$$

where τ is the discrete index corresponding to continuous time t. The noise-prediction and score parameterizations are related by

$$\epsilon_\theta(x_\tau, \tau) \approx -\sqrt{1 - \bar{\alpha}_\tau}\,\nabla \log p_\tau(x_\tau). \tag{35}$$

Indeed, differentiating the Gaussian conditional density in (33) gives

$$\nabla_{x_\tau} \log p(x_\tau|x_0) = -\frac{x_\tau - \sqrt{\bar{\alpha}_\tau}x_0}{1 - \bar{\alpha}_\tau} = -\frac{\epsilon}{\sqrt{1 - \bar{\alpha}_\tau}}.$$

The deterministic DDIM update is

$$x_{\tau-1} = \sqrt{\bar{\alpha}_{\tau-1}}\,\frac{x_\tau - \sqrt{1 - \bar{\alpha}_\tau}\,\epsilon_\theta(x_\tau, \tau)}{\sqrt{\bar{\alpha}_\tau}} + \sqrt{1 - \bar{\alpha}_{\tau-1}}\,\epsilon_\theta(x_\tau, \tau). \tag{36}$$

Using (35), this can equivalently be expressed in score form as

$$x_{\tau-1} = \sqrt{\frac{\bar{\alpha}_{\tau-1}}{\bar{\alpha}_\tau}}x_\tau + \left[\sqrt{\frac{\bar{\alpha}_{\tau-1}}{\bar{\alpha}_\tau}}(1 - \bar{\alpha}_\tau) - \sqrt{(1 - \bar{\alpha}_{\tau-1})(1 - \bar{\alpha}_\tau)}\right]\nabla \log p_\tau(x_\tau). \tag{37}$$

C.2 EXPONENTIAL-TILTING CONTROL

Let

$$\ell_0(x) := \log p_0(x), \qquad L_0 := (\ell_0)_{\#} p_0$$

be the log-likelihood pushforward distribution. For $\lambda \geq 0$, define its exponential tilt by

$$L_0^{\lambda}(du) = \frac{e^{-\lambda u}}{Z(\lambda)} L_0(du), \qquad Z(\lambda) = \int e^{-\lambda v} L_0(dv). \tag{38}$$

The corresponding Radon–Nikodym derivative is

$$z_0^{\lambda}(u) = \frac{dL_0^{\lambda}}{dL_0}(u) = \frac{e^{-\lambda u}}{Z(\lambda)}.$$

Consequently, the terminal control coefficient is constant:

$$c_0(u) := \partial_u \log z_0^{\lambda}(u) = -\lambda. \tag{39}$$

As established in the main text, the time-dependent controller is obtained by the exponentially interpolated coefficient

$$c_t(u) \approx \exp\left(-\int_0^t b(s)\, ds\right) c_0(u). \tag{40}$$

Combining (34), (39), and (40) gives the discrete-time control

$$c_\tau(u) \approx -\lambda \sqrt{\bar{\alpha}_\tau}. \tag{41}$$

The controlled score is therefore

$$\nabla \log q_\tau(x_\tau) \approx \left(1 - \lambda \sqrt{\bar{\alpha}_\tau}\right) \nabla \log p_\tau(x_\tau). \tag{42}$$

Using the score–noise correspondence (35), this is equivalent to replacing the usual noise prediction by

$$\epsilon_\theta^{\text{out}}(x_\tau, \tau) := \left(1 - \lambda \sqrt{\bar{\alpha}_\tau}\right) \epsilon_\theta(x_\tau, \tau). \tag{43}$$

Thus, outlier generation can be incorporated into an existing DDIM sampler by multiplying its predicted noise by a deterministic scalar at each reverse step. No auxiliary network, likelihood estimator, or model retraining is required.

Substituting (43) into the standard DDIM update (36) gives

$$x_{\tau-1} = \sqrt{\bar{\alpha}_{\tau-1}} \frac{x_\tau - \sqrt{1-\bar{\alpha}_\tau}\, \left(1 - \lambda \sqrt{\bar{\alpha}_\tau}\right) \epsilon_\theta(x_\tau, \tau)}{\sqrt{\bar{\alpha}_\tau}} + \sqrt{1-\bar{\alpha}_{\tau-1}}\, \left(1 - \lambda \sqrt{\bar{\alpha}_\tau}\right) \epsilon_\theta(x_\tau, \tau). \tag{44}$$

Equivalently, in score form,

$$x_{\tau-1} = \sqrt{\frac{\bar{\alpha}_{\tau-1}}{\bar{\alpha}_\tau}} x_\tau + \left[\sqrt{\frac{\bar{\alpha}_{\tau-1}}{\bar{\alpha}_\tau}}(1-\bar{\alpha}_\tau) - \sqrt{(1-\bar{\alpha}_{\tau-1})(1-\bar{\alpha}_\tau)}\right] \left(1 - \lambda \sqrt{\bar{\alpha}_\tau}\right) \nabla \log p_\tau(x_\tau). \tag{45}$$

Equation (43) is the most direct implementation of the proposed method: a standard deterministic DDIM sampler is converted into an outlier generator through the single-line modification

$$\epsilon_\theta \leftarrow \left(1 - \lambda \sqrt{\bar{\alpha}_\tau}\right) \epsilon_\theta.$$

The parameter λ controls the strength of the likelihood tilt and therefore the degree to which sampling is steered toward low-likelihood regions.

D SUMMARY OF BASELINE MODELS

We demonstrate strong performance against a wide range of benchmarks. The authors of `ExtremeGANs` (Mukherjee, 2026) provide implementations and evaluations for widely cited GAN-based models in the field.

TailGAN (Cont et al., 2026) is a modification of the Vanilla GAN architecture (Goodfellow et al., 2014). It trains the discriminator network to predict the α-value at risk (VaR) and α-expected shortfall (ES) of the generated data rather than distributional similarity. This tail score loss is equivalently used to train the generator.

ParetoGAN (Huster et al., 2021) is a discriminator-free modification of the Vanilla GAN architecture (Goodfellow et al., 2014). The generator takes noise sampled from the generalized Pareto distribution (GPD) rather than the Gaussian distribution. The generator is trained via an energy distance with respect to the training data.

Heavy-Tail GAN (HTGAN, Girard et al. (2025)) pre-processes the training data by transforming the empirical coordinate-wise marginals of the training set into Fréchet distributions. It then trains a Vanilla GAN (Goodfellow et al., 2014) with these modified samples and with input noise sampled from the GPD.

Extreme-Value GAN (EV-GAN, Allouche et al. (2022)) takes uniform distribution noise as input and modifies the generator to approximate a tail-index function (TIF) in logarithmic space. The discriminator is trained identically to the Vanilla GAN (Goodfellow et al., 2014), but also in logarithmic space.

Conditional Deep Convolutional Generative Adversarial Network (CDCGAN, Li et al. (2023)) assigns each sample a measure of their "extremeness", often computed as the mean of all their components. It estimates quantiles of the extremeness scores and uses the quantile as conditioning in the networks.

Finally, standard GANs such as Vanilla GAN (Goodfellow et al., 2014) and Wasserstein-GAN (WGAN) (Arjovsky et al., 2017) are included in the package as well.

E Reproducibility Details

This section provides additional implementation and evaluation details required to reproduce the experiments reported in the paper. Unless otherwise stated, all models are trained using the same preprocessing pipeline and evaluated using the same number of generated samples and random seeds.

E.1 Dataset Preprocessing

All time-series datasets are converted into fixed-length windows of $T = 64$ time steps. We construct the windows using overlapping segments of the original time series, so that consecutive windows may contain shared observations. Each resulting training example therefore has dimension $64 \times d$, where d denotes the number of features in the corresponding dataset.

We do not partition the datasets into separate training, validation, and test sets. Instead, all windows constructed from each dataset are used for model training. The same collection of real windows is subsequently used as the reference distribution when computing the evaluation metrics. This choice is motivated by our objective of evaluating how accurately each generative model can learn and reproduce the empirical distribution, including its relatively rare tail observations, rather than forecasting unseen future observations.

Before model training, each feature is independently standardized using z-score normalization. The same normalization parameters are used for all windows from a given dataset. Models are trained, and samples are generated in this normalized space. Before evaluation, every generated sample is transformed back to the original data scale. This ensures all distributional, dependence, and tail metrics reported in the paper are computed in the original, unnormalized data space.

S&P500 Datasets For the S&P500 experiments, we use daily returns from the Utilities and Financials sectors over the period from February 8, 2013 to February 6, 2018. The Utilities and Financials datasets contain 10 and 42 features, respectively. We construct overlapping windows

Hyperparameter	Value
Depth	7
Number of heads*	16
MLP Ratio	4.0
Batch size	4
Number of epochs	100
Learning Rate	1×10^{-4}

Table 2: Training and architectural hyperparameters shared across datasets. *The number of attention heads is set to 42 for the Traffic dataset.

Dataset	Number of features	Hidden Size
S&P500 Utilities	10	304
S&P500 Financials	42	672
Weather	20	320
Traffic	861	1722
Illness	6	96
Exchange Rate	7	112
ETTm2	6	96
ETTh2	6	96
Electricity	320	640

Table 3: Dataset dimensionalities and hidden sizes used for the DiT architecture. The hidden size is chosen to be compatible with both the number of input features and the number of attention heads.

of length 64 from the return series and apply the normalization procedure described above. No additional transformations or feature engineering are performed prior to training.

CSV Datasets For the remaining time-series datasets, we load the observations from their corresponding CSV files. We remove the columns `Date` and `OT`, when present, and retain the remaining columns as the multivariate features used for training. We then construct overlapping windows of length 64 and apply feature-wise z-score normalization as described above. Generated samples are transformed back to the original scale before being compared with the corresponding real windows.

E.2 Diffusion Transformer

We implement the DM and all training procedures in PyTorch. The denoising network is parameterized by a Diffusion Transformer (DiT), whose architecture is adapted to the dimensionality of each dataset. All models are trained on a single NVIDIA H100 GPU.

We optimize the DiT parameters using Adam with a fixed learning rate of 1×10^{-4} and a batch size of 4. Each model is trained for 100 epochs. We use a transformer depth of 7, an MLP ratio of 4.0, and 16 attention heads for all datasets except Traffic, for which we use 42 attention heads. The hidden dimension is selected separately for each dataset so that it is compatible with both the feature dimension and the number of attention heads. The common training hyperparameters and dataset-specific architectural choices are summarized in Tables 2 and 3, respectively.

Training time varies substantially with the dimensionality and size of the dataset. On a single NVIDIA H100 GPU, training takes approximately 1 hour for the smallest models, such as ETTh2, and approximately 8 hours for the largest dataset, Traffic.

E.3 Sample Generation and Evaluation

For each dataset and generation method, we generate 10,000 synthetic time-series windows for evaluation. Each generated sample has the same shape as a real training window, namely $64 \times d$. We repeat the generation and evaluation procedure using five random seeds, $0 - 4$. Thus, each reported experimental configuration is evaluated using 10,000 independently generated windows to compute the metrics for each seed.

After generation, synthetic windows are transformed from the normalized model space back to the original data scale using the feature-wise means and standard deviations computed during preprocessing. All evaluation metrics are then computed between these unnormalized generated samples and the real dataset.

For each seed, we independently compute all distributional, dependence, and outlier metrics described in Section I. The results reported in the main text are then aggregated across the five generation seeds. This evaluation protocol is kept fixed across the proposed method and all baseline methods to ensure that differences in the reported metrics are not attributable to the number of generated samples or the choice of random seed.

E.4 ExtremeGANs Baselines

We compare against the generative models provided by the `ExtremeGANs` benchmark. To ensure a consistent comparison, all `ExtremeGANs` baselines are trained on exactly the same preprocessed windows used to train our DM. In particular, they use the same overlapping windows of length 64 and the same feature-wise z-score normalization.

We use the default model architectures, optimization settings, and training hyperparameters provided by the `ExtremeGANs` package. We do not perform additional dataset-specific hyperparameter tuning of the baselines. This protocol is intended to provide a reproducible comparison using the standard configurations supplied by their implementation while holding the input data and evaluation procedure fixed.

For evaluation, each `ExtremeGANs` baseline generates 10,000 synthetic windows for each of the same five random seeds, $0 - 4$. As with our method, generated samples are transformed back to the original data scale before evaluation. We then apply exactly the same metric implementations to the baseline and diffusion-generated samples. Hence, all methods are evaluated using identical real training windows, the same number of generated samples per seed, the same set of evaluation seeds, and the same metric implementations.

F Benchmark Results

This section provides additional quantitative comparisons with the baseline methods considered in Section 6. We report results across the full collection of datasets and evaluation metrics used in our experiments in Tables 4,5, and 6. These results complement the main-text evaluation by providing a more comprehensive view of the relative performance of the proposed likelihood control method across different notions of distributional fidelity and extreme-event generation.

Table 4: Distributional similarity and outlier-generation metrics on S&P500 Utilities and Financials. GAN baselines are compared with diffusion at $\lambda = 0$ and $\lambda = 0.12$. Best results are shown in bold and second-best results are underlined.

Dataset	Model	Marginal KS ↓	FID ↓	Tail-MSLE$_{95}$ ↓	CCDF Area ↓
Utilities	WGAN	0.363 ± 0.013	0.062 ± 0.001	1.308 ± 0.014	20.496 ± 0.129
Utilities	Vanilla	0.229 ± 0.001	0.049 ± 0.000	1.499 ± 0.004	22.622 ± 0.073
Utilities	TailGAN	0.300 ± 0.003	0.051 ± 0.000	2.371 ± 0.079	26.322 ± 0.386
Utilities	ParetoGAN	0.517 ± 0.000	0.062 ± 0.000	18.715 ± 0.147	60.002 ± 0.457
Utilities	HTGAN	0.267 ± 0.004	0.056 ± 0.000	1.187 ± 0.028	19.034 ± 0.372
Utilities	EVGAN	0.998 ± 0.000	998.389 ± 599.300	12.084 ± 0.042	67.598 ± 0.932
Utilities	CDCGAN	0.627 ± 0.008	0.070 ± 0.001	7.958 ± 0.459	22.867 ± 0.810
Utilities	Diffusion ($\lambda = 0$)	0.064 ± 0.000	0.009 ± 0.000	0.120 ± 0.002	2.878 ± 0.166
Utilities	Diffusion ($\lambda = 0.12$)	$\mathbf{0.054 \pm 0.001}$	$\mathbf{0.007 \pm 0.000}$	$\mathbf{0.074 \pm 0.002}$	$\mathbf{2.045 \pm 0.153}$
Financials	WGAN	0.278 ± 0.008	0.070 ± 0.001	1.627 ± 0.010	23.161 ± 0.081
Financials	Vanilla	0.225 ± 0.001	0.062 ± 0.000	1.709 ± 0.008	24.473 ± 0.035
Financials	TailGAN	0.286 ± 0.005	0.065 ± 0.000	2.525 ± 0.103	27.111 ± 0.523
Financials	ParetoGAN	0.510 ± 0.000	0.078 ± 0.000	15.796 ± 0.066	56.404 ± 0.256
Financials	HTGAN	0.310 ± 0.005	0.070 ± 0.000	1.582 ± 0.051	19.315 ± 0.463
Financials	EVGAN	0.996 ± 0.000	1176.464 ± 326.140	12.489 ± 0.014	79.048 ± 0.394
Financials	CDCGAN	0.627 ± 0.006	0.087 ± 0.002	8.604 ± 0.354	23.812 ± 0.986
Financials	Diffusion ($\lambda = 0$)	0.052 ± 0.001	0.013 ± 0.000	0.140 ± 0.003	3.942 ± 0.055
Financials	Diffusion ($\lambda = 0.12$)	$\mathbf{0.049 \pm 0.001}$	$\mathbf{0.011 \pm 0.000}$	$\mathbf{0.099 \pm 0.002}$	$\mathbf{3.286 \pm 0.040}$

Table 5: Weather, traffic, national illness, and exchange rate datasets. GAN baselines are compared with diffusion at $\lambda = 0$ and $\lambda = 0.12$. Best results are shown in bold and second-best results are underlined.

Dataset	Model	KS ↓	Corr. Error ↓	ACF Error ↓	Tail-MSLE$_{95}$ ↓	CCDF Area ↓
Weather	WGAN	0.240 ± 0.002	0.038 ± 0.008	0.281 ± 0.010	8.858 ± 0.028	15.862 ± 0.021
Weather	Vanilla	0.268 ± 0.002	0.125 ± 0.006	0.428 ± 0.008	8.653 ± 0.025	15.794 ± 0.009
Weather	TailGAN	0.670 ± 0.129	0.451 ± 0.029	0.666 ± 0.012	9.327 ± 1.297	16.693 ± 0.098
Weather	ParetoGAN	0.602 ± 0.000	0.429 ± 0.035	0.233 ± 0.022	8.978 ± 0.010	16.469 ± 0.007
Weather	HTGAN	$\mathbf{0.114 \pm 0.003}$	0.102 ± 0.018	0.410 ± 0.005	9.251 ± 0.031	8.798 ± 0.439
Weather	EVGAN	0.940 ± 0.019	0.450 ± 0.000	0.659 ± 0.001	320.269 ± 13.950	383.634 ± 1.621
Weather	CDCGAN	0.785 ± 0.029	0.385 ± 0.047	0.434 ± 0.019	9.516 ± 0.820	16.091 ± 1.105
Weather	Diffusion ($\lambda = 0$)	0.219 ± 0.001	$\mathbf{0.035 \pm 0.002}$	$\mathbf{0.069 \pm 0.000}$	$\mathbf{6.769 \pm 0.014}$	$\mathbf{2.532 \pm 0.313}$
Weather	Diffusion ($\lambda = 0.12$)	0.214 ± 0.001	0.040 ± 0.002	0.102 ± 0.001	7.154 ± 0.016	2.780 ± 0.296
Traffic	WGAN	0.193 ± 0.003	0.213 ± 0.003	0.056 ± 0.005	0.384 ± 0.002	11.471 ± 0.040
Traffic	Vanilla	0.437 ± 0.006	0.351 ± 0.055	0.235 ± 0.028	0.870 ± 0.042	13.796 ± 0.234
Traffic	TailGAN	0.537 ± 0.021	0.497 ± 0.044	0.375 ± 0.006	1.307 ± 0.075	19.812 ± 1.529
Traffic	ParetoGAN	0.561 ± 0.000	0.317 ± 0.002	0.554 ± 0.003	1.454 ± 0.000	21.963 ± 0.005
Traffic	HTGAN	0.538 ± 0.012	$\mathbf{0.124 \pm 0.002}$	0.114 ± 0.011	2.736 ± 0.313	14.776 ± 0.783
Traffic	EVGAN	0.812 ± 0.054	0.592 ± 0.000	0.360 ± 0.000	74.557 ± 98.109	234.968 ± 211.379
Traffic	CDCGAN	0.748 ± 0.031	0.150 ± 0.046	0.091 ± 0.029	2.763 ± 1.059	8.474 ± 2.003
Traffic	Diffusion ($\lambda = 0$)	$\mathbf{0.125 \pm 0.001}$	0.157 ± 0.000	0.048 ± 0.000	0.189 ± 0.002	$\mathbf{6.275 \pm 0.281}$
Traffic	Diffusion ($\lambda = 0.12$)	0.130 ± 0.001	0.150 ± 0.000	$\mathbf{0.043 \pm 0.000}$	$\mathbf{0.185 \pm 0.001}$	6.517 ± 0.279
Illness	WGAN	0.502 ± 0.010	0.700 ± 0.040	0.577 ± 0.010	1.147 ± 0.087	10.832 ± 0.186
Illness	Vanilla	0.220 ± 0.005	0.374 ± 0.023	0.343 ± 0.023	0.673 ± 0.004	10.960 ± 0.037
Illness	TailGAN	0.358 ± 0.002	0.708 ± 0.004	0.607 ± 0.000	0.811 ± 0.006	12.123 ± 0.095
Illness	ParetoGAN	0.613 ± 0.001	0.574 ± 0.051	0.443 ± 0.039	1.839 ± 0.007	18.166 ± 0.053
Illness	HTGAN	$\mathbf{0.190 \pm 0.015}$	0.207 ± 0.023	$\mathbf{0.194 \pm 0.024}$	0.398 ± 0.047	3.789 ± 0.584
Illness	EVGAN	0.682 ± 0.003	0.736 ± 0.000	0.586 ± 0.002	1.440 ± 0.057	31.256 ± 2.471
Illness	CDCGAN	0.761 ± 0.005	0.696 ± 0.028	0.549 ± 0.032	1.722 ± 0.142	9.717 ± 0.586
Illness	Diffusion ($\lambda = 0$)	0.200 ± 0.003	$\mathbf{0.030 \pm 0.001}$	0.359 ± 0.004	0.075 ± 0.003	3.039 ± 0.110
Illness	Diffusion ($\lambda = 0.12$)	0.209 ± 0.003	0.037 ± 0.002	0.388 ± 0.004	$\mathbf{0.051 \pm 0.002}$	$\mathbf{2.775 \pm 0.098}$
Exchange	WGAN	0.211 ± 0.005	0.090 ± 0.020	0.581 ± 0.009	0.019 ± 0.001	1.635 ± 0.003
Exchange	Vanilla	0.255 ± 0.021	0.320 ± 0.084	0.501 ± 0.012	0.029 ± 0.004	1.784 ± 0.114
Exchange	TailGAN	0.416 ± 0.004	0.490 ± 0.005	0.617 ± 0.003	0.050 ± 0.001	2.154 ± 0.008
Exchange	ParetoGAN	0.540 ± 0.002	0.559 ± 0.087	0.355 ± 0.102	0.092 ± 0.002	2.863 ± 0.007
Exchange	HTGAN	$\mathbf{0.110 \pm 0.007}$	0.227 ± 0.015	0.600 ± 0.009	$\mathbf{0.002 \pm 0.000}$	0.930 ± 0.091
Exchange	EVGAN	0.592 ± 0.070	0.530 ± 0.018	0.573 ± 0.015	4.263 ± 9.436	49.464 ± 94.413
Exchange	CDCGAN	0.749 ± 0.042	0.519 ± 0.034	0.610 ± 0.011	0.122 ± 0.066	2.853 ± 1.033
Exchange	Diffusion ($\lambda = 0$)	0.179 ± 0.004	0.094 ± 0.002	$\mathbf{0.063 \pm 0.001}$	0.006 ± 0.000	0.594 ± 0.007
Exchange	Diffusion ($\lambda = 0.12$)	0.178 ± 0.003	$\mathbf{0.081 \pm 0.001}$	0.092 ± 0.001	0.006 ± 0.000	$\mathbf{0.547 \pm 0.004}$

Table 6: ETT benchmark results. GAN baselines are compared with diffusion at $\lambda = 0$ and $\lambda = 0.12$. Lower is better for all metrics. Best results are shown in bold and second-best results are underlined.

Dataset	Model	KS ↓	ACF Error ↓	Corr. Error ↓	Tail-MSLE$_{95}$ ↓	CCDF Area ↓
ETTm2	WGAN	0.179 ± 0.003	0.094 ± 0.019	0.135 ± 0.000	$\mathbf{0.335 \pm 0.000}$	6.154 ± 0.005
ETTm2	Vanilla	0.206 ± 0.001	$\mathbf{0.074 \pm 0.007}$	0.278 ± 0.007	0.632 ± 0.004	6.617 ± 0.009
ETTm2	TailGAN	0.743 ± 0.149	0.355 ± 0.034	0.491 ± 0.124	1.588 ± 0.026	12.300 ± 2.674
ETTm2	ParetoGAN	0.533 ± 0.007	0.344 ± 0.041	0.421 ± 0.046	2.999 ± 0.643	10.237 ± 0.231
ETTm2	HTGAN	$\mathbf{0.076 \pm 0.001}$	0.088 ± 0.007	0.220 ± 0.008	0.552 ± 0.043	$\mathbf{0.714 \pm 0.027}$
ETTm2	EVGAN	0.517 ± 0.001	0.357 ± 0.017	0.397 ± 0.012	1.861 ± 0.011	5.792 ± 0.288
ETTm2	CDCGAN	0.792 ± 0.024	0.288 ± 0.059	0.389 ± 0.024	2.571 ± 0.930	6.667 ± 0.836
ETTm2	Diffusion ($\lambda = 0$)	0.145 ± 0.001	0.100 ± 0.001	0.103 ± 0.006	0.492 ± 0.022	1.596 ± 0.101
ETTm2	Diffusion ($\lambda = 0.12$)	0.133 ± 0.001	0.129 ± 0.001	$\mathbf{0.087 \pm 0.006}$	0.419 ± 0.021	1.413 ± 0.077
ETTh2	WGAN	0.204 ± 0.005	0.099 ± 0.022	0.115 ± 0.005	$\mathbf{0.336 \pm 0.003}$	6.240 ± 0.012
ETTh2	Vanilla	0.225 ± 0.005	$\mathbf{0.064 \pm 0.004}$	0.242 ± 0.016	0.632 ± 0.019	6.524 ± 0.011
ETTh2	TailGAN	0.508 ± 0.008	0.220 ± 0.004	0.351 ± 0.010	1.321 ± 0.032	8.513 ± 0.183
ETTh2	ParetoGAN	0.528 ± 0.000	0.296 ± 0.115	0.376 ± 0.065	2.908 ± 0.022	10.274 ± 0.011
ETTh2	HTGAN	$\mathbf{0.111 \pm 0.006}$	0.091 ± 0.006	0.252 ± 0.011	0.581 ± 0.057	$\mathbf{0.503 \pm 0.041}$
ETTh2	EVGAN	0.557 ± 0.071	0.214 ± 0.016	0.406 ± 0.001	2.373 ± 1.346	42.179 ± 84.105
ETTh2	CDCGAN	0.804 ± 0.029	0.231 ± 0.030	0.405 ± 0.043	2.187 ± 1.471	6.545 ± 1.950
ETTh2	Diffusion ($\lambda = 0$)	0.173 ± 0.001	0.132 ± 0.002	0.077 ± 0.005	0.435 ± 0.022	1.530 ± 0.066
ETTh2	Diffusion ($\lambda = 0.12$)	0.161 ± 0.001	0.121 ± 0.001	$\mathbf{0.070 \pm 0.004}$	0.363 ± 0.023	1.311 ± 0.060
Electricity	WGAN	0.224 ± 0.001	$\mathbf{0.035 \pm 0.001}$	0.069 ± 0.002	0.097 ± 0.000	7.623 ± 0.026
Electricity	Vanilla	0.376 ± 0.026	0.123 ± 0.023	0.133 ± 0.010	0.222 ± 0.009	8.118 ± 0.135
Electricity	TailGAN	0.549 ± 0.005	0.409 ± 0.002	0.517 ± 0.007	0.393 ± 0.007	12.909 ± 0.307
Electricity	ParetoGAN	0.574 ± 0.000	0.182 ± 0.006	0.326 ± 0.007	0.445 ± 0.000	14.923 ± 0.003
Electricity	HTGAN	0.327 ± 0.011	0.100 ± 0.007	0.109 ± 0.008	0.202 ± 0.066	3.183 ± 0.257
Electricity	EVGAN	0.582 ± 0.016	0.410 ± 0.000	0.564 ± 0.000	56.279 ± 12.656	372.121 ± 7.936
Electricity	CDCGAN	0.746 ± 0.012	0.076 ± 0.010	0.182 ± 0.017	0.667 ± 0.071	5.562 ± 1.394
Electricity	Diffusion ($\lambda = 0$)	$\mathbf{0.104 \pm 0.001}$	0.040 ± 0.000	$\mathbf{0.043 \pm 0.001}$	0.019 ± 0.001	$\mathbf{1.999 \pm 0.114}$
Electricity	Diffusion ($\lambda = 0.12$)	0.110 ± 0.001	0.040 ± 0.000	0.044 ± 0.001	$\mathbf{0.018 \pm 0.000}$	2.085 ± 0.111

G Ablation Studies

This section provides additional results illustrating the effect of the control strength λ on the generated samples. Recall that $\lambda = 0$ corresponds to standard sampling from the pretrained DM, while increasing λ progressively shifts the log-likelihood distribution toward lower-likelihood regions. Thus, λ provides a direct mechanism for controlling the extremeness of the generated samples without retraining the underlying generative model. We report results across the full collection of datasets and evaluation metrics used in our experiments in Tables 7,8, and 9.

Table 7: Distributional similarity and outlier-generation metrics on S&P500 Utilities and Financials. Best results are shown in bold and second-best results are underlined.

Dataset	λ	Marginal KS ↓	FGD ↓	Tail-MSLE$_{95}$ ↓	CCDF Area ↓
Utilities	0.0	0.064 ± 0.000	0.009 ± 0.000	0.120 ± 0.002	2.878 ± 0.166
Utilities	0.04	0.060 ± 0.000	0.008 ± 0.000	0.102 ± 0.002	2.540 ± 0.161
Utilities	0.08	0.056 ± 0.000	0.008 ± 0.000	0.088 ± 0.002	2.276 ± 0.157
Utilities	0.12	0.054 ± 0.001	0.007 ± 0.000	0.074 ± 0.002	2.045 ± 0.153
Utilities	0.16	0.052 ± 0.001	0.007 ± 0.000	0.064 ± 0.001	1.894 ± 0.147
Utilities	0.2	**0.051 ± 0.001**	**0.007 ± 0.000**	**0.054 ± 0.001**	**1.792 ± 0.141**
Financials	0.0	0.052 ± 0.001	0.013 ± 0.000	0.140 ± 0.003	3.942 ± 0.055
Financials	0.04	0.050 ± 0.001	0.012 ± 0.000	0.125 ± 0.003	3.686 ± 0.048
Financials	0.08	**0.049 ± 0.001**	0.011 ± 0.000	0.111 ± 0.002	3.478 ± 0.043
Financials	0.12	0.049 ± 0.001	0.011 ± 0.000	0.099 ± 0.002	3.286 ± 0.040
Financials	0.16	0.050 ± 0.001	0.011 ± 0.000	0.088 ± 0.002	3.131 ± 0.038
Financials	0.2	0.051 ± 0.001	**0.010 ± 0.000**	**0.078 ± 0.002**	**2.987 ± 0.036**

Table 8: Weather, traffic, national illness, and exchange rate datasets. Lower is better. Best results are shown in bold and second-best results are underlined.

Dataset	λ	KS ↓	Corr. Error ↓	ACF Error ↓	Tail-MSLE$_{95}$ ↓	CCDF Area ↓
Weather	0.0	0.219 ± 0.001	**0.035 ± 0.002**	**0.069 ± 0.000**	**6.769 ± 0.014**	**2.532 ± 0.313**
Weather	0.04	0.217 ± 0.001	0.037 ± 0.002	0.079 ± 0.001	6.858 ± 0.020	2.620 ± 0.315
Weather	0.08	0.216 ± 0.001	0.038 ± 0.002	0.089 ± 0.001	6.990 ± 0.020	2.703 ± 0.305
Weather	0.12	0.214 ± 0.001	0.040 ± 0.002	0.102 ± 0.001	7.154 ± 0.016	2.780 ± 0.296
Weather	0.16	0.214 ± 0.001	0.042 ± 0.002	0.115 ± 0.001	7.352 ± 0.014	2.851 ± 0.298
Weather	0.2	**0.213 ± 0.001**	0.044 ± 0.001	0.131 ± 0.001	7.670 ± 0.017	2.926 ± 0.301
Traffic	0.0	**0.125 ± 0.001**	0.157 ± 0.000	0.048 ± 0.000	0.189 ± 0.002	**6.275 ± 0.281**
Traffic	0.04	0.127 ± 0.001	0.155 ± 0.000	0.046 ± 0.000	0.188 ± 0.002	6.364 ± 0.280
Traffic	0.08	0.128 ± 0.001	0.153 ± 0.000	0.044 ± 0.000	0.186 ± 0.002	6.441 ± 0.279
Traffic	0.12	0.130 ± 0.001	0.150 ± 0.000	0.043 ± 0.000	0.185 ± 0.001	6.517 ± 0.279
Traffic	0.16	0.131 ± 0.001	0.147 ± 0.000	0.042 ± 0.000	0.183 ± 0.001	6.582 ± 0.278
Traffic	0.2	0.133 ± 0.001	**0.144 ± 0.000**	**0.041 ± 0.000**	**0.181 ± 0.001**	6.645 ± 0.278
Illness	0.0	**0.200 ± 0.003**	**0.030 ± 0.001**	**0.359 ± 0.004**	0.075 ± 0.003	3.039 ± 0.110
Illness	0.04	0.203 ± 0.003	0.032 ± 0.002	0.370 ± 0.004	0.065 ± 0.002	2.932 ± 0.103
Illness	0.08	0.206 ± 0.003	0.035 ± 0.002	0.379 ± 0.004	0.057 ± 0.002	2.849 ± 0.100
Illness	0.12	0.209 ± 0.003	0.037 ± 0.002	0.388 ± 0.004	0.051 ± 0.002	2.775 ± 0.098
Illness	0.16	0.211 ± 0.003	0.040 ± 0.002	0.396 ± 0.004	0.046 ± 0.002	2.717 ± 0.097
Illness	0.2	0.213 ± 0.003	0.043 ± 0.002	0.403 ± 0.003	**0.042 ± 0.002**	**2.666 ± 0.097**
Exchange	0.0	0.179 ± 0.004	0.094 ± 0.002	**0.063 ± 0.001**	0.006 ± 0.000	0.594 ± 0.007
Exchange	0.04	0.179 ± 0.004	0.089 ± 0.002	0.067 ± 0.001	0.006 ± 0.000	0.574 ± 0.004
Exchange	0.08	0.178 ± 0.004	0.085 ± 0.002	0.077 ± 0.001	0.006 ± 0.000	0.558 ± 0.003
Exchange	0.12	0.178 ± 0.003	0.081 ± 0.001	0.092 ± 0.001	0.006 ± 0.000	0.547 ± 0.004
Exchange	0.16	0.178 ± 0.003	0.077 ± 0.001	0.110 ± 0.001	0.006 ± 0.000	0.538 ± 0.005
Exchange	0.2	**0.177 ± 0.003**	**0.074 ± 0.001**	0.130 ± 0.001	**0.006 ± 0.000**	**0.531 ± 0.007**

Table 9: ETT benchmark results. Lower is better for all metrics. Best results are shown in bold and second-best results are underlined.

Dataset	λ	KS ↓	ACF Error ↓	Corr. Error ↓	Tail-MSLE$_{95}$ ↓	CCDF Area ↓
ETTm2	0.0	0.145 ± 0.001	$\mathbf{0.100 \pm 0.001}$	0.103 ± 0.006	0.492 ± 0.022	1.596 ± 0.101
ETTm2	0.04	0.141 ± 0.001	0.104 ± 0.001	0.097 ± 0.006	0.466 ± 0.022	1.528 ± 0.094
ETTm2	0.08	0.137 ± 0.001	0.114 ± 0.001	0.092 ± 0.006	0.443 ± 0.021	1.470 ± 0.087
ETTm2	0.12	0.133 ± 0.001	0.129 ± 0.001	0.087 ± 0.006	0.419 ± 0.021	1.413 ± 0.077
ETTm2	0.16	0.130 ± 0.001	0.143 ± 0.001	0.084 ± 0.005	0.397 ± 0.021	1.362 ± 0.069
ETTm2	0.2	$\mathbf{0.127 \pm 0.001}$	0.157 ± 0.001	$\mathbf{0.082 \pm 0.005}$	$\mathbf{0.375 \pm 0.021}$	$\mathbf{1.309 \pm 0.062}$
ETTh2	0.0	0.173 ± 0.001	0.132 ± 0.002	0.077 ± 0.005	0.435 ± 0.022	1.530 ± 0.066
ETTh2	0.04	0.168 ± 0.002	0.126 ± 0.001	0.074 ± 0.005	0.411 ± 0.023	1.453 ± 0.063
ETTh2	0.08	0.164 ± 0.002	0.123 ± 0.001	0.071 ± 0.005	0.388 ± 0.023	1.384 ± 0.062
ETTh2	0.12	0.161 ± 0.001	0.121 ± 0.001	0.070 ± 0.004	0.363 ± 0.023	1.311 ± 0.060
ETTh2	0.16	0.157 ± 0.001	$\mathbf{0.120 \pm 0.001}$	$\mathbf{0.069 \pm 0.004}$	0.342 ± 0.023	1.245 ± 0.057
ETTh2	0.2	$\mathbf{0.154 \pm 0.001}$	0.120 ± 0.001	0.070 ± 0.004	$\mathbf{0.319 \pm 0.023}$	$\mathbf{1.178 \pm 0.056}$
Electricity	0.0	$\mathbf{0.104 \pm 0.001}$	$\mathbf{0.040 \pm 0.000}$	$\mathbf{0.043 \pm 0.001}$	0.019 ± 0.001	$\mathbf{1.999 \pm 0.114}$
Electricity	0.04	0.106 ± 0.001	0.040 ± 0.000	0.043 ± 0.001	0.018 ± 0.001	2.032 ± 0.113
Electricity	0.08	0.108 ± 0.001	0.040 ± 0.000	0.043 ± 0.001	0.018 ± 0.000	2.059 ± 0.112
Electricity	0.12	0.110 ± 0.001	0.040 ± 0.000	0.044 ± 0.001	0.018 ± 0.000	2.085 ± 0.111
Electricity	0.16	0.111 ± 0.001	0.041 ± 0.000	0.043 ± 0.001	0.018 ± 0.000	2.107 ± 0.109
Electricity	0.2	0.113 ± 0.001	0.041 ± 0.000	0.043 ± 0.001	$\mathbf{0.018 \pm 0.000}$	2.127 ± 0.107

H Likelihood Computation and Validation of ρ-outliers

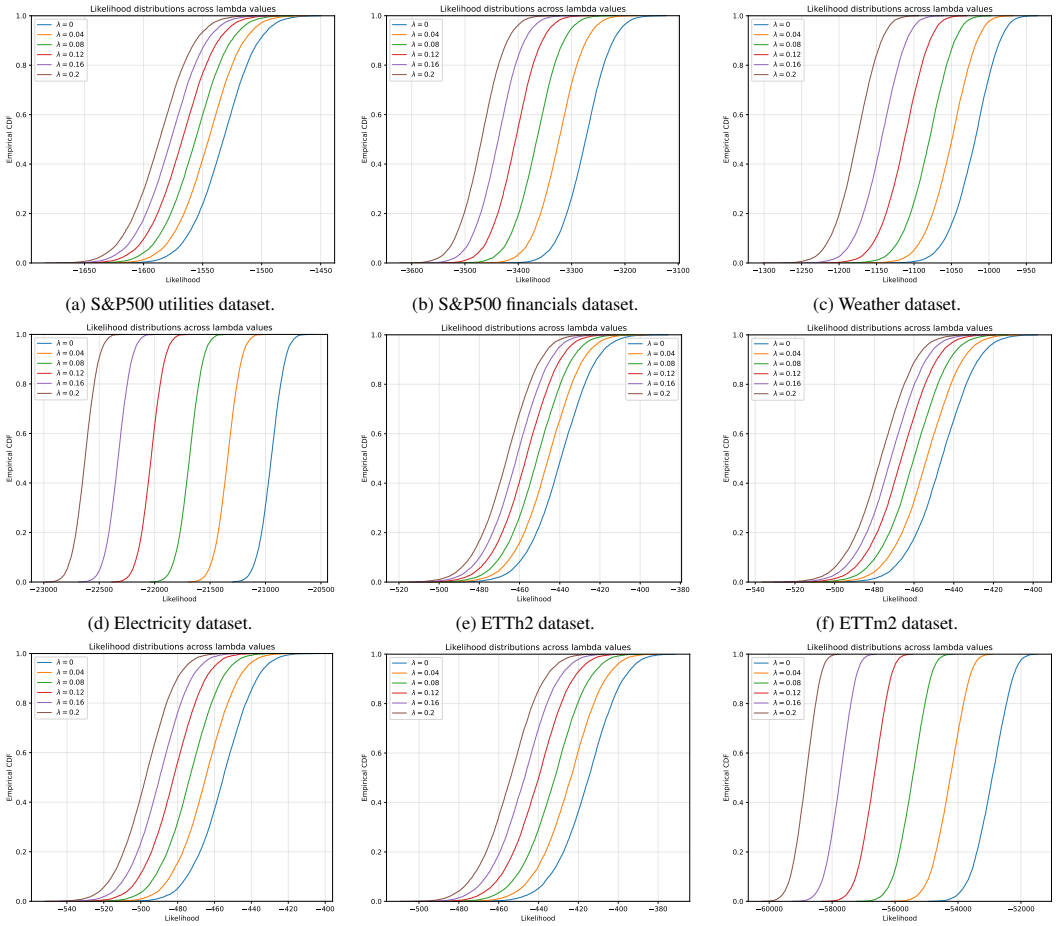
This section provides likelihood-space diagnostics supporting the theoretical properties of exponential tilting. We first examine empirical stochastic ordering of the generated log-likelihood distributions and then report the corresponding Wasserstein displacements used to validate the theoretical results in Section 4.

H.1 Validating stochastic dominance

We first empirically validate the stochastic ordering predicted by Theorem 1. Figure 4 shows the empirical CDFs of the computed log-likelihoods of samples generated using different values of the control strength λ. As expected, increasing λ progressively shifts the log-likelihood distribution toward more negative values, corresponding to samples that are less likely under the reference data distribution. This is consistent with the first-order stochastic dominance established for exponentially tilted log-likelihood distributions. These results demonstrate that λ provides a direct and monotonic mechanism for controlling the likelihood, and hence the degree of extremeness, of the generated samples.

H.2 Validating Theorem 2

We next empirically validate the lower-bound relationship between the control strength λ and the displacement of the log-likelihood distribution established in Theorem 2. For each dataset and value of λ, we compute the Wasserstein-1 distance between the log-likelihood distribution obtained under likelihood control and the uncontrolled log-likelihood distribution corresponding to $\lambda = 0$. We compute $\rho(\lambda)$ using the `wasserstein_distance` function from SciPy. Table 10 reports the raw Wasserstein displacement $\rho(\lambda)$, whose monotonicity follows from Theorem 1. To validate the stronger calibration result of Theorem 2, Figure 3 reports the normalized quantity $\rho(\lambda)/\mathrm{Var}(U_0)$.



(a) S&P500 utilities dataset. (b) S&P500 financials dataset. (c) Weather dataset.

(d) Electricity dataset. (e) ETTh2 dataset. (f) ETTm2 dataset.

(g) Exchange rate dataset. (h) Illness dataset. (i) Traffic dataset.

Figure 4: Empirical CDFs of log-likelihood values for different values of λ

Table 10: Empirical values of $\rho(\lambda)$ across datasets. The entry at $\lambda = 0$ is excluded because $\rho(0)$ is zero by definition.

Dataset	$\lambda = 0.04$	$\lambda = 0.08$	$\lambda = 0.12$	$\lambda = 0.16$	$\lambda = 0.2$
Electricity	393.103	787.064	1182.616	1578.314	1974.968
Etth2	6.339	12.710	19.094	25.481	31.865
Ettm2	7.589	15.007	22.247	29.331	36.205
Exchange Rate	9.559	19.365	29.345	39.402	49.497
National Illness	8.929	18.072	27.356	36.664	45.934
Sp500 Financials	49.806	97.469	143.075	187.158	229.507
Sp500 Utilities	12.580	25.235	37.969	50.754	63.591
Traffic	1330.759	2693.022	4090.082	5508.667	6953.858
Weather	30.725	64.541	101.883	141.730	184.344

I Evaluation Metrics

In this section, we summarize the evaluation metrics used throughout the experiments. The metrics are grouped into two categories: (i) distributional similarity metrics, and (ii) outlier and tail-focused metrics. Lower values indicate better agreement between the generated and real distributions.

I.1 Distributional Similarity Metrics

These metrics evaluate how closely the generated distribution matches the overall empirical distribution of the real data.

I.1.1 Marginal Kolmogorov–Smirnov (KS) Mean

The marginal KS mean (Huster et al., 2021) evaluates how well the generated data reproduce the marginal distribution at each time-feature coordinate. Suppose the real and generated datasets each consist of N time-series windows of length T with d features, and let $X_{t,j}^{(i)}$ and $Y_{t,j}^{(i)}$ denote feature j at time t in the ith real and generated windows, respectively. Each window is viewed as a vector in $\mathbb{R}^{Td}$, so that every coordinate corresponds to a particular time-feature pair (t, j).

For each (t, j), we construct empirical CDFs across the N windows,

$$F_{t,j}(x) = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}\left(X_{t,j}^{(i)} \le x\right), \qquad G_{t,j}(x) = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}\left(Y_{t,j}^{(i)} \le x\right). \tag{46}$$

The one-dimensional KS statistic for coordinate (t, j) is then

$$D_{t,j} = \sup_x |F_{t,j}(x) - G_{t,j}(x)|. \tag{47}$$

The marginal KS mean averages these discrepancies over all Td time-feature coordinates:

$$\mathrm{KSMean}(X, Y) = \frac{1}{Td}\sum_{t=1}^{T}\sum_{j=1}^{d} D_{t,j}. \tag{48}$$

Thus, the metric first compares the real and generated distributions across windows at each fixed time and feature, and then averages the resulting KS statistics over both the time and feature dimensions. Smaller values indicate closer agreement between the coordinate-wise marginal distributions of the real and generated data.

The metric is closely related to the Dimension-wise KS Summation (DKSS) used by Zhao et al. (2024), but averages rather than sums the coordinate-wise KS statistics. The marginal KS mean is normalized with respect to the dimensionality of the flattened time series, facilitating comparisons across datasets with different sequence lengths and numbers of features.

I.1.2 FRÉCHET GAUSSIAN DISTANCE (FGD)

We additionally compute a FGD between the generated and real feature distributions (Li et al., 2023). Given empirical means μ_r, μ_g and covariance matrices Σ_r, Σ_g for the real and generated data respectively, the metric is defined as

$$\mathrm{FGD}(X, Y) = \|\mu_r - \mu_g\|_2^2 + \mathrm{Tr}\left(\Sigma_r + \Sigma_g - 2(\Sigma_r \Sigma_g)^{1/2}\right). \tag{49}$$

Smaller values indicate closer agreement between the first and second-order moments of the distributions.

I.1.3 AUTOCORRELATION ERROR

Temporal dependence within each feature is evaluated using autocorrelation functions (ACFs) (Cont et al., 2026). Suppose the dataset contains N time-series windows, each of length T with d features, and let $X_{t,j}^{(i)}$ denote feature j at time t in window i. For each lag $\ell \in 1, \ldots, L$ and feature $j \in 1, \ldots, d$, we first compute the Pearson correlation between the two lagged sequences

$$\left(X_{1,j}^{(i)}, \ldots, X_{T-\ell,j}^{(i)}\right) \quad \text{and} \quad \left(X_{1+\ell,j}^{(i)}, \ldots, X_{T,j}^{(i)}\right)$$

within each window i. Denoting this window-level autocorrelation by $\rho_j^{(i)}(\ell)$, we then average over the N windows:

$$\rho_j(\ell) = \frac{1}{N} \sum_{i=1}^{N} \rho_j^{(i)}(\ell). \tag{50}$$

This procedure is applied separately to the real and generated datasets, yielding $\rho_{r,j}(\ell)$ and $\rho_{g,j}(\ell)$, respectively.

The autocorrelation error is then defined as the mean absolute difference between the real and generated ACFs, averaged over both features and lags:

$$\mathrm{ACFErr} = \frac{1}{Ld} \sum_{\ell=1}^{L} \sum_{j=1}^{d} \left|\rho_{r,j}(\ell) - \rho_{g,j}(\ell)\right|. \tag{51}$$

Thus, the metric measures how well the generated data reproduce feature-wise temporal dependence across lags $1, \ldots, L$, while averaging equally across features and lags.

I.1.4 PEARSON CORRELATION ERROR

Cross-feature dependence is evaluated using Pearson correlation matrices (Cont et al., 2026). Unlike the autocorrelation metric, which computes temporal correlations separately within each time-series window, we pool the window and time dimensions when computing cross-feature correlations. Specifically, the N windows of length T are reshaped into NT observations in $\mathbb{R}^d$,

$$\widetilde{X} = \left(X_t^{(i)} : i = 1, \ldots, N, ; t = 1, \ldots, T\right), \tag{52}$$

so that each window–time pair (i, t) is treated as one observation of the d features.

We then compute the $d \times d$ Pearson correlation matrix across features from these pooled observations. Let C_r and C_g denote the resulting correlation matrices for the real and generated datasets, respectively, where $(C_r)_{jk}$ and $(C_g)_{jk}$ measure the Pearson correlation between features j and k after pooling over all windows and time points. The Pearson correlation error is defined as

$$\mathrm{CorrErr} = \frac{1}{d} \left|C_r - C_g\right|_F = \sqrt{\frac{1}{d^2} \sum_{j=1}^{d} \sum_{k=1}^{d} \left((C_r)_{jk} - (C_g)_{jk}\right)^2}. \tag{53}$$

Hence, CorrErr is the root-mean-square discrepancy between the entries of the real and generated feature correlation matrices. Smaller values indicate better preservation of cross-feature dependence.

I.2 Outlier and Tail Metrics

While the preceding metrics evaluate overall distributional and dependence structure, we additionally consider metrics that focus specifically on the ability of the generated data to reproduce rare, large-magnitude observations and heavy-tail behaviour. Since extreme deviations may occur in either direction, both metrics are applied to the absolute values of the observations.

I.2.1 Tail Mean Squared Logarithmic Error

Following Allouche et al. (2022), we evaluate discrepancies between the upper tails of the real and generated distributions using a mean squared logarithmic error. Suppose the real and generated datasets each contain N time-series windows of length T with d features. As in the marginal KS metric, each window is viewed as a vector in $\mathbb{R}^{Td}$, so that each coordinate corresponds to a particular time-feature pair (t, j).

For a tail threshold $\alpha \in (0, 1)$, let

$$K = \lceil (1 - \alpha)N \rceil \tag{54}$$

denote the number of observations retained in the upper tail at each coordinate. For each (t, j), we take the K largest values of $|X_{t,j}^{(i)}|_{i=1}^{N}$ and $|Y_{t,j}^{(i)}|_{i=1}^{N}$ and arrange them in increasing order. Denote these ordered tail observations by

$$R_{t,j}^{(1)} \leq \cdots \leq R_{t,j}^{(K)}, \qquad G_{t,j}^{(1)} \leq \cdots \leq G_{t,j}^{(K)}. \tag{55}$$

The TailMSLE is then defined as

$$\text{TailMSLE} = \frac{1}{TdK} \sum_{t=1}^{T} \sum_{j=1}^{d} \sum_{k=1}^{K} \left[\log\Big(R_{t,j}^{(k)} + \epsilon\Big) - \log\Big(G_{t,j}^{(k)} + \epsilon\Big) \right]^2, \tag{56}$$

where $\epsilon > 0$ is a small constant for numerical stability.

Thus, rather than comparing only a single quantile, TailMSLE compares the real and generated observations throughout the upper $(1 - \alpha)$ fraction of the absolute-value distribution at each time-feature coordinate. Matching observations by their within-tail rank provides a quantile-based comparison of tail magnitudes, while the logarithmic transformation emphasizes relative rather than absolute discrepancies. The resulting errors are averaged over tail ranks, time points, and features.

I.2.2 Log-Log CCDF Area

Following Huster et al. (2021), we additionally compare the overall tail distributions of the real and generated data through their complementary cumulative distribution functions (CCDFs). In contrast to TailMSLE, which evaluates each time-feature coordinate separately, this metric pools the window, time, and feature dimensions. Specifically, all finite entries of the dataset are collected into a single scalar sample of absolute magnitudes,

$$\mathcal{R} = \Big(|X_{t,j}^{(i)}| : i = 1, \ldots, N, ; t = 1, \ldots, T, ; j = 1, \ldots, d \Big), \tag{57}$$

with $\mathcal{G}$ defined analogously for the generated data. Values smaller than a numerical tolerance ϵ are excluded before computing the metric.

Let n denote the smaller number of retained real and generated observations, and let

$$R_{(1)} \geq R_{(2)} \geq \cdots \geq R_{(n)}, \qquad G_{(1)} \geq G_{(2)} \geq \cdots \geq G_{(n)} \tag{58}$$

denote the corresponding observations sorted from largest to smallest. The kth largest observation corresponds to an empirical exceedance probability of approximately k/n. The log-log CCDF area is computed as

$$\text{CCDFArea} = \sum_{k=1}^{n} \big| \log\big(R_{(k)} + \epsilon\big) - \log\big(G_{(k)} + \epsilon\big) \big| \log\left(\frac{k+1}{k} \right). \tag{59}$$

The factor $\log((k + 1)/k)$ corresponds to the spacing between consecutive empirical exceedance probabilities on a logarithmic scale. Hence, the above sum approximates the area between the

inverse empirical CCDFs of the real and generated absolute magnitudes when both magnitude and exceedance probability are represented on logarithmic axes.

Unlike TailMSLE, which preserves the time-feature coordinates and focuses on a prescribed upper-tail fraction, $\mathrm{CCDFArea}$ pools all observations and compares the overall distribution of extreme magnitudes across the dataset. Smaller values indicate closer agreement between the real and generated heavy-tail behaviour.