Structure-Preserving Generative Models: Extreme Values, Order Statistics, and Diffusion Fine-Tuning

December 18, 2025

Contents

List of Figures

List of Tables

Introduction

0.1 Context and Motivation

0.1.1 Structure in Statistical Data

The term *statistics* derives from the German *Statistik*, coined by Gottfried Achenwall in 1748 to denote the "science of the state": governing required organized information. Data is the raw material of statistics. A scientist measures temperature at several locations; a bank records daily returns of assets in a portfolio; a hospital collects patient vital signs. In each case, we have multiple observations, and each observation may consist of several related quantities. These quantities are organized in specific ways, governed by underlying regularities. Uncovering these regularities is the task of statistical science.

This process admits a clear mathematical formulation. We denote n observations by $X_1, \ldots, X_n$. Each X_i is a *vector*: a list of numbers representing the measured quantities. We assume these observations are governed by an underlying law. Mathematically, this law is a *probability distribution*, which we denote P. The distribution specifies how likely each possible outcome is. It is unknown; learning about it is the central problem.

The goal of *statistics* is to infer properties of P from the observations: estimate its parameters, test hypotheses about its form, quantify uncertainty. The goal of *generative modeling* is more ambitious: to produce new synthetic data that could plausibly have come from the same source, indistinguishable from genuine observations.

But data carries *structure*: dependencies between variables, constraints on their values, properties that define the phenomenon itself. A faithful generative model must respect this structure, not merely approximate the overall law.

0.1.2 Modern Generative Modeling

Given data, can we learn to generate more of it? Not just compute averages, but produce new data that look as if they came from the same source? This is the question of *generative modeling*.

When observations have many components (temperatures at hundreds of locations, returns of thousands of assets), directly describing the law P becomes impractical. Modern generative modeling takes a different path: rather than describing P explicitly, we learn

to produce data from it. The core idea is:

$$G(Z) \sim P.$$

Here Z is a source of randomness, like rolling dice or drawing from a bell curve. The transformation G turns this randomness into something that resembles the data. All the complexity of the unknown law P is encoded in G.

This echoes a classical idea. A statistician assumes the data follow a known family of laws (bell curves, exponentials, etc.) and estimates a few numbers to pin down which one. A generative modeler makes a similar bet: instead of assuming the law has a certain shape, we assume the transformation G does. When G is a neural network, we bet that transforming randomness through layers of computations can mimic the data.

The breakthrough of the past decade is that neural networks can learn G from examples, producing images indistinguishable from photographs, text that reads as human-written. But the catch: neural networks learn G by minimizing a global measure of error. This process has no mechanism to preserve specific structural properties.

0.1.3 The Problem: When Structure Matters

What does it mean for synthetic data to be *faithful* to the phenomenon it imitates?

Consider a bank generating fake crisis scenarios to test its investments. The generated scenarios look realistic on average, yet the worst losses occur in isolation: when stocks crash, bonds stay calm. The synthetic data capture each variable separately but miss how they move together. In a real crisis, losses are correlated. Or consider generating synthetic rankings: the third-best item sometimes scores higher than the first-best. The ranking is scrambled; the simulation is useless.

These failures reflect a mismatch between what generative models optimize and what applications require. Standard models reward getting the overall shape right but provide no guarantees on specific structural properties.

0.1.4 This Thesis: Theory-Guided Generative Methods

This manuscript starts from real-world problems where structure matters. We identify the key property that must be preserved (joint crashes, rankings, preferences), formulate it as a mathematical object, and analyze how well neural networks can approximate it. This is the core methodology: from applied problem, to mathematical abstraction, to theoretical analysis.

We study three structural regimes: tail dependence, order statistics, and reward tilting. For each, we design generative methods that *provably* respect the structure, providing guarantees on approximation accuracy.

0.2 Three Structural Regimes

We now describe the three structural regimes addressed in this thesis. For each regime, we present the real-world problem that motivates our work and formalize it mathematically.

0.2.1 Extreme Values and Tail Dependence

The problem: when extremes drive everything. In many domains, extreme events dominate outcomes despite their rarity. In portfolio management, a handful of large losses can wipe out years of steady gains: the distribution of returns is heavy-tailed, and risk is concentrated in the extremes (Rama Cont 2001). Value-at-Risk, stress testing, and regulatory capital requirements all hinge on the probability of rare but catastrophic losses. The 2008 financial crisis demonstrated that models calibrated to normal conditions fail when extremes occur (A. J. McNeil et al. 2015).

In insurance, extreme events are the core business. Reinsurers must price the risk of simultaneous catastrophes: a hurricane hitting Florida while an earthquake strikes California. The joint occurrence of such events, not their individual probabilities, determines solvency requirements. In climate science, compound extremes (concurrent heatwaves and droughts, sequential flooding events) pose the greatest risks to ecosystems and infrastructure (Jakob Zscheischler et al. 2018; Bevacqua et al. 2021). Climate models must capture not just the frequency of individual extremes, but their tendency to cluster in space and time.

The common thread is *tail dependence*: the tendency for extremes to occur together. Two random variables can have zero correlation under normal conditions yet exhibit strong tail dependence: when one takes an extreme value, the other is likely to as well (see Figure 2). For a portfolio manager, this means diversification fails precisely when it is needed most. For an insurer, it means correlated claims. For climate risk, it means compound disasters.

Two distinct challenges. Generating realistic extreme value data poses two fundamentally different challenges.

The first is *marginal heavy-tailedness*. A distribution is heavy-tailed when extreme values are far more likely than light-tailed models would predict. Figure 1 compares three distributions: the absolute Gaussian $|X|$, the exponential (both light-tailed), and the Pareto (heavy-tailed). Near zero they look similar, but their tails behave very differently. Table 1 shows the probability of exceeding various thresholds. At $t = 8$, the Gaussian assigns probability 10^{-15}, the exponential 10^{-4}, but the Pareto still gives 3.7%. Heavy-tailed phenomena produce extreme events that light-tailed models dismiss as virtually impossible.

The second challenge is *extremal dependence*: when multiple variables are involved, their joint behavior in the tails can be entirely different from their dependence in normal conditions. Figure 2 illustrates this: two samples with identical marginal distributions

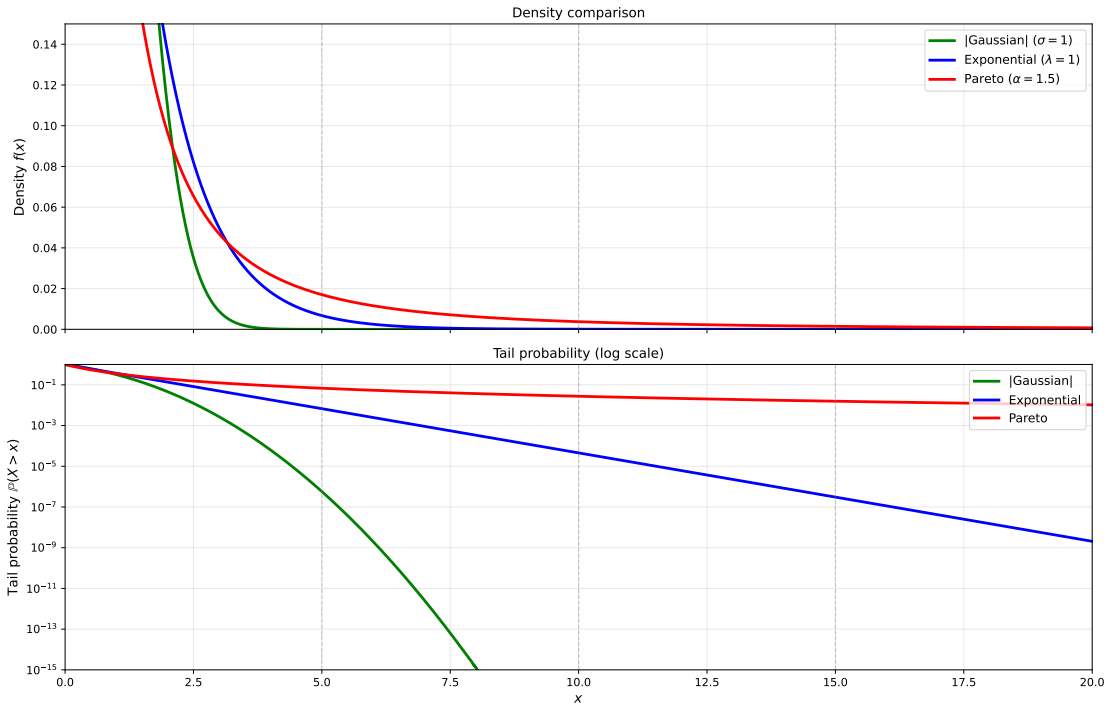


Figure 1: Light-tailed vs heavy-tailed distributions. Left: density comparison for |Gaussian|, exponential, and Pareto. Right: tail probability $\mathbb{P}(X > t)$ on log scale. The Gaussian and exponential tails decay rapidly; the Pareto tail decays as a power law, remaining substantial even for large t.

Table 1: Probability of exceeding threshold t for |Gaussian| ($\sigma = 1$), exponential ($\lambda = 1$), and Pareto ($\alpha = 1.5$).

t	|Gaussian|	Exponential	Pareto
3	2.7×10^{-3}	5.0×10^{-2}	1.3×10^{-1}
5	5.7×10^{-7}	6.7×10^{-3}	6.8×10^{-2}
8	1.3×10^{-15}	3.4×10^{-4}	3.7×10^{-2}

behave very differently in the tails depending on whether extremes occur together or independently.

As we shall see in Section 0.4.1, standard generative models fail on both counts.

Mathematical formulation: the stable tail dependence function. In classical statistics, *correlation* measures how two variables move together on average. Correlation summarizes dependence across the entire distribution but does not fully characterize tail behavior. Two variables can have zero correlation yet crash together in a crisis; conversely, variables with the same correlation can exhibit very different tail dependence structures.

The *stable tail dependence function* (stdf) plays for extremes the role that correlation plays for the bulk of the distribution. Just as correlation captures average co-movement, the stdf captures extreme co-movement: when one variable takes a very large value, how likely is the other to be large as well? For a random vector $\mathbf{X} = (X_1, \ldots, X_d)$, the stdf ℓ

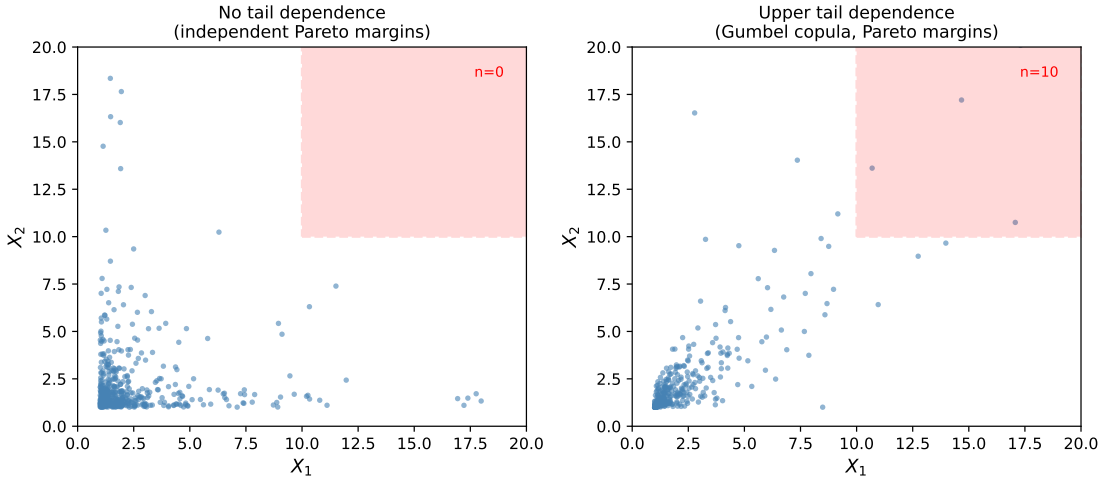


Figure 2: Tail dependence illustrated with Pareto margins ($\alpha = 1.5$). Left: independent margins; extreme values of X_1 and X_2 occur independently, so the upper tail region (red box) is sparse. Right: Gumbel copula ($\theta = 2.5$) with upper tail dependence; when X_1 is extreme, X_2 tends to be extreme as well, creating clustering in the upper-right corner. Both samples have identical marginal distributions, yet behave very differently in the tails.

is defined by

$$\ell(\mathbf{x}) = \lim_{t\to\infty} t\,\mathbb{P}\left(\bigcup_{j=1}^{d}\{X_j > tx_j\}\right). \tag{1}$$

This limit captures the asymptotic probability of joint exceedances as we move deeper into the tail. The stdf completely characterizes extremal dependence: two distributions with the same stdf exhibit identical behavior in extremes, regardless of how they behave in the bulk. Knowing the stdf is essential for understanding diversification when it matters most: during extreme events.

0.2.2 Order Statistics and Ranking Constraints

The problem: when ranking defines the data. Many phenomena are defined by their rank structure. In auctions, what matters is the distribution of the highest bid, or the gap between the two highest bids. In sports, we care about the performance of the top-ranked player, not the average. In extreme value statistics, the k largest observations are used to estimate tail behavior. In each case, the ordering is not incidental: it is the defining structure of the data.

Consider impact investing, where assets are ranked by environmental, social, or governance (ESG) scores (Lo and R. Zhang 2021). A fund manager backtesting an ESG strategy needs to simulate the returns of the best-ranked asset, the second-best, and so on. These are *induced order statistics*: the returns $\Theta_{[1:N]}, \ldots, \Theta_{[N:N]}$ associated with assets ranked by their ESG scores $X_{1:N} \leqslant \cdots \leqslant X_{N:N}$. Synthetic data must preserve both the marginal distribution at each rank and the joint dependence across ranks.

Mathematical formulation: order statistics and their joint density. Given a sample $X_1, \ldots, X_n$ from a continuous distribution F with density f, the *order statistics* are the sorted values $X_{1:n} \leqslant X_{2:n} \leqslant \cdots \leqslant X_{n:n}$. Their joint density is

$$f_{1:n,\ldots,n:n}(x_1, \ldots, x_n) = n! \prod_{i=1}^{n} f(x_i) \cdot \mathbf{1}\{x_1 \leqslant \cdots \leqslant x_n\}. \tag{2}$$

The factor $n!$ counts the number of arrangements, and the indicator $\mathbf{1}\{\cdot\}$ enforces the ordering constraint. Classical representations, such as those of Sukhatme and Schucany, decompose order statistics into tractable building blocks that have enabled exact simulation for decades. As we shall see in Section 0.5.2, these representations also provide the foundation for neural network approximation with provable complexity bounds.

0.2.3 Reward-Tilted Distributions and Model Fine-Tuning

The problem: aligning generative models with human preferences. A pre-trained generative model produces samples from a distribution learned from training data. But for many applications, we want samples that are not merely typical, but *good* according to some criterion: images that are aesthetically pleasing, text that is helpful and harmless, molecules that bind to a target protein. This is the *alignment problem*: adjusting a generative model to favor high-quality outputs while maintaining diversity and coherence.

Consider a text-to-image diffusion model trained on web-scraped images. The model generates realistic images, but many are poorly composed, contain artifacts, or depict undesirable content. We have access to a reward model (trained on human preferences) that scores images by aesthetic quality and safety. The goal is to fine-tune the diffusion model so that it preferentially generates high-reward images, without collapsing to a single mode or forgetting how to generate diverse content.

Mathematical formulation: exponential tilting. The natural mathematical formulation is *exponential tilting*. Given a base distribution p_0 (the pre-trained model) and a reward function $r : \mathcal{X} \to \mathbb{R}$, the target is the *tilted distribution*

$$p(x) \propto p_0(x) \exp(\lambda\, r(x)), \tag{3}$$

where $\lambda > 0$ controls the strength of the tilting. This distribution is the unique solution to the KL-regularized reward maximization problem:

$$p = \arg\max_{q} \left\{ \mathbb{E}_q[r(x)] - \frac{1}{\lambda} \mathrm{KL}(q \| p_0) \right\}.$$

The tilted distribution balances reward maximization against staying close to the pre-trained model, with λ controlling the trade-off. Large λ favors high reward; small λ stays close to p_0.

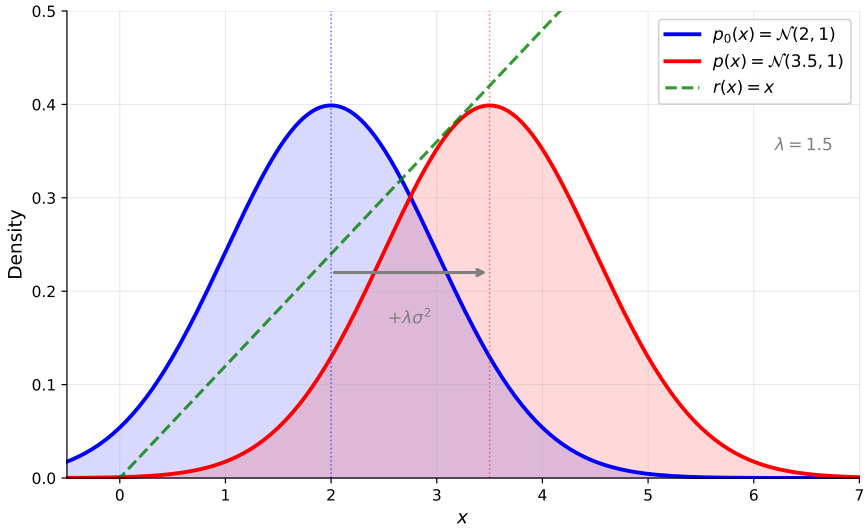


Figure 3: Reward tilting shifts the distribution toward high-reward regions. With $p_0 = \mathcal{N}(2, 1)$, linear reward $r(x) = x$, and tilting strength $\lambda = 1.5$, the tilted distribution $p(x) \propto p_0(x) \exp(\lambda r(x))$ is $\mathcal{N}(3.5, 1)$: the mean shifts by $\lambda\sigma^2 = 1.5$ toward high reward, while the variance is unchanged.

0.3 Generative Modeling Background

Section 0.1 introduced the goal of learning from data while respecting structure. Section 0.2 presented three regimes where structure matters: tail dependence, order statistics, and reward tilting. To address these challenges, we turn to generative modeling: the task of producing synthetic data that follows the same law as the observations. This section provides the technical background on generative models and neural network approximation theory necessary to understand the contributions of this thesis.

0.3.1 The Problem of Generative Modeling

The fundamental problem of generative modeling can be stated as follows: given samples $X_1, \ldots, X_n$ drawn independently from an unknown distribution P on $\mathbb{R}^d$, construct a procedure for generating new samples that are distributed according to P, or a close approximation thereof.

This problem has a long history in statistics and probability. Classical approaches include *density estimation* (Silverman 1986; Scott 2015), where one estimates the density $p = dP/d\lambda$ and then samples from the estimated density, and *Markov Chain Monte Carlo* (MCMC) (Metropolis et al. 1953; Hastings 1970; C. P. Robert and Casella 2004), where one constructs a Markov chain with P as its stationary distribution and samples by running the chain to equilibrium. Both approaches face the curse of dimensionality (Bellman 1961): density estimation becomes statistically intractable in high dimensions, while MCMC mixing times can grow exponentially with dimension.

The modern approach to generative modeling, enabled by deep learning, takes a different path. Rather than explicitly estimating the density, we learn a *transport map* G

that transforms samples from a simple reference distribution (typically standard Gaussian) into samples from the target distribution. If Z is Gaussian noise, then $G(Z)$ should be distributed approximately as P. The map G is parameterized by a neural network, and the parameters are learned from data.

This transport-based view unifies several modern generative paradigms. In *normalizing flows* (Rezende and Mohamed 2015; Dinh et al. 2017), the map G is constrained to be invertible with tractable Jacobian. In *variational autoencoders* (VAEs) (Diederik P. Kingma and Max Welling 2014), the map is learned jointly with an approximate inverse. In *generative adversarial networks* (GANs) (I. J. Goodfellow et al. 2014), the map is learned via an adversarial objective. In *diffusion models* (Ho, Jain, et al. 2020; Y. Song, Jascha Sohl-Dickstein, et al. 2021), the map is implicitly defined by the reverse of a noising process. We describe GANs and diffusion models in detail below, as they are the foundations of our contributions.

0.3.2 Neural Networks

A *feedforward neural network* with L hidden layers transforms an input x through successive layers (Figure 4). Denoting $h_0 = x$, the hidden representations are

$$h_\ell = \sigma(W_\ell h_{\ell-1} + b_\ell) \quad \text{for } \ell = 1, 2, \ldots, L, \tag{4}$$

where W_ℓ is a weight matrix, b_ℓ a bias vector, and σ a nonlinear activation function applied coordinate-wise. The output is $g(x) = W_{L+1}h_L + b_{L+1}$. Common activations include the sigmoid, tanh, and ReLU $\sigma(t) = \max(0, t)$, which dominates modern practice.

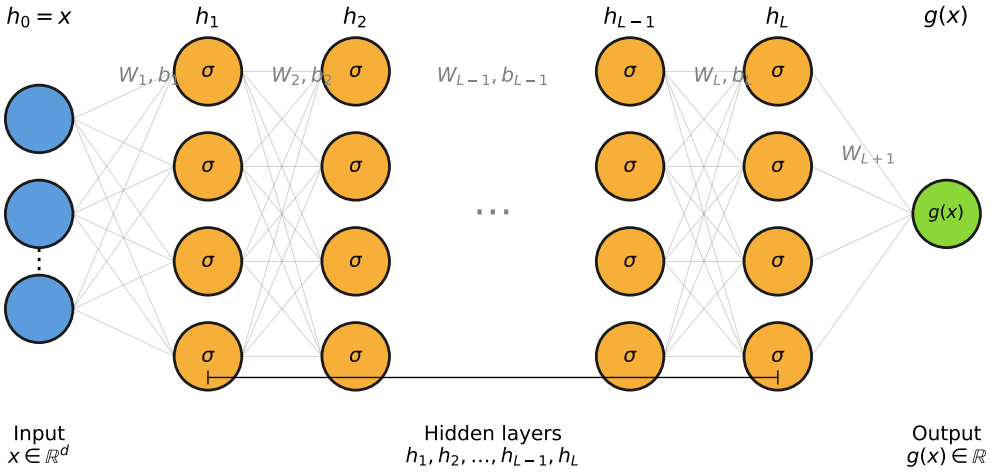


Figure 4: A feedforward neural network. The input $h_0 = x$ passes through L hidden layers, each applying an affine transformation followed by activation σ. The output $g(x)$ is a linear function of h_L.

0.3.3 Neural Network Approximation Theory

The *universal approximation theorem* (Cybenko 1989; Hornik et al. 1989) states that neural networks with a single hidden layer can approximate any continuous function on a compact domain to arbitrary accuracy, given sufficient width. This extends to ReLU networks (Leshno et al. 1993).

Quantitative bounds. Quantitative bounds relate approximation error to network size (Barron 1993; Yarotsky 2017). For Lipschitz functions on $[0, 1]^d$, achieving error ε requires $O(\varepsilon^{-d})$ parameters, exhibiting the curse of dimensionality. For smoother functions (Hölder class with smoothness s), networks with $O(\varepsilon^{-d/s})$ parameters suffice (Yarotsky 2017).

Limitations. These theorems concern pointwise function approximation, not preservation of distributional structure. A network approximating a transport map G uniformly may still fail to preserve tail behavior, ordering constraints, or dependence structure. Moreover, the theorems assume optimal parameters are known; in practice, gradient descent may not find them. These limitations motivate the structure-specific theory developed in this thesis.

0.3.4 Generative Adversarial Networks

Generative Adversarial Networks (I. J. Goodfellow et al. 2014) frame generative modeling as a two-player game. The intuition is best understood through an analogy: imagine a counterfeiter trying to produce fake banknotes, and a detective trying to detect them. The counterfeiter improves by studying which fakes get caught; the detective improves by studying the counterfeiter's latest techniques. Over time, both become more sophisticated, until eventually the counterfeiter produces notes so convincing that even the expert detective cannot tell them apart from genuine currency.

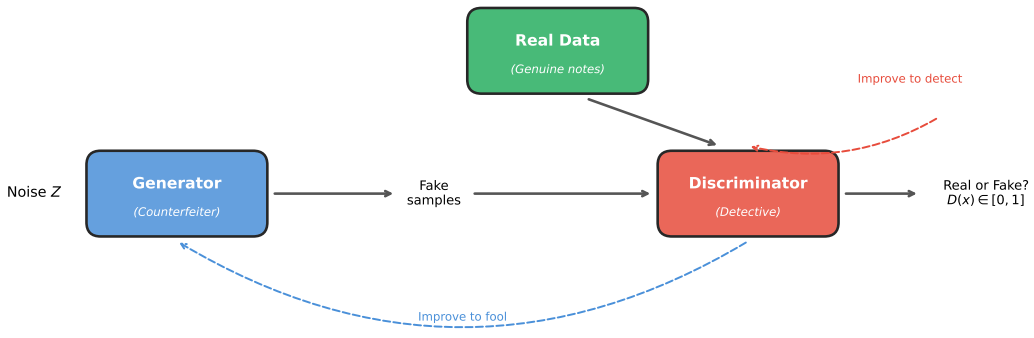


Figure 5: The GAN framework as an adversarial game. The generator (counterfeiter) transforms noise into fake samples; the discriminator (detective) tries to distinguish fakes from real data. Both networks improve through competition until equilibrium, when generated samples are indistinguishable from real data.

In mathematical terms, a *generator* $G_\theta : \mathcal{Z} \to \mathcal{X}$ maps latent noise $Z \sim p_Z$ to synthetic samples (the counterfeiter's output). A *discriminator* $D_\phi : \mathcal{X} \to [0,1]$ attempts to distinguish real samples from generated ones (the detective's verdict). The networks are trained by solving a minimax problem:

$$\min_\theta \max_\phi \; \mathbb{E}_{X\sim p_{\text{data}}}[\log D_\phi(X)] + \mathbb{E}_{Z\sim p_Z}[\log(1 - D_\phi(G_\theta(Z)))].$$

The generator minimizes this objective (trying to fool the discriminator), while the discriminator maximizes it (trying to correctly classify real vs. fake). At equilibrium, the generator produces samples from the data distribution, and the discriminator outputs $1/2$ for all inputs, unable to distinguish real from fake.

The Wasserstein GAN (WGAN) (Arjovsky et al. 2017) replaces the Jensen-Shannon divergence with the Wasserstein-1 distance:

$$W_1(p_{\text{data}}, p_\theta) = \sup_{\|f\|_L \leqslant 1} \mathbb{E}_{X\sim p_{\text{data}}}[f(X)] - \mathbb{E}_{Z\sim p_Z}[f(G_\theta(Z))],$$

where the supremum is over 1-Lipschitz functions. The WGAN with gradient penalty (WGAN-GP) (Gulrajani et al. 2017) enforces the Lipschitz constraint via a penalty term, improving training stability.

0.3.5 Diffusion and Score-Based Models

Diffusion models (Ho, Jain, et al. 2020; Y. Song, Jascha Sohl-Dickstein, et al. 2021) define a generative process by constructing a continuous path $(p_t)_{t\in[0,1]}$ between the data distribution p_0 and a tractable reference $p_1 = \mathrm{N}(0, \mathrm{I}_d)$. The path is defined via the *forward interpolation*

$$X_t = \alpha_t X_0 + \sigma_t X_1, \quad X_0 \sim p_0, \quad X_1 \sim \mathrm{N}(0, \mathrm{I}_d), \tag{5}$$

where (α_t, σ_t) are deterministic schedules with $(\alpha_0, \sigma_0) = (1, 0)$ and $(\alpha_1, \sigma_1) = (0, 1)$. At $t = 0$, we have clean data; at $t = 1$, pure noise. Standard choices include the variance-preserving schedule ($\alpha_t^2 + \sigma_t^2 = 1$) and the linear schedule ($\alpha_t = 1 - t$, $\sigma_t = t$).

To generate samples, we reverse this process. The key quantity is the *score function* $\nabla_x \log p_t$, which points toward regions of high density at each noise level. For the Gaussian forward kernel, the score satisfies $\nabla_x \log p_t(x_t) = -\hat{x}_1(x_t, t)/\sigma_t$, where $\hat{x}_1(x_t, t) = \mathbb{E}[X_1 \mid X_t = x_t]$ is the conditional expectation of the noise given the noised sample.

The score is learned via *denoising score matching* (Hyvärinen 2005; Vincent 2011): a network $s_\theta(x, t)$ is trained to predict the noise X_1 from the noised sample X_t, which is equivalent to learning $\nabla_x \log p_t$. Starting from $X_1 \sim \mathrm{N}(0, \mathrm{I}_d)$ and iteratively denoising using the learned score yields samples from p_0.

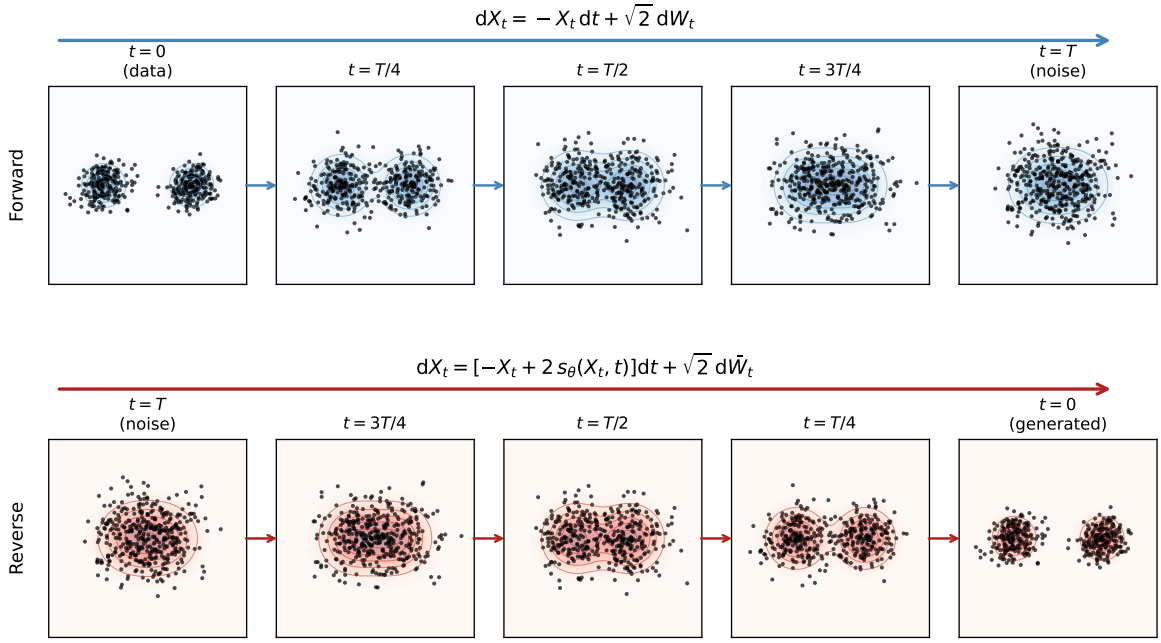


Figure 6: The diffusion model framework on a 2D Gaussian mixture. Forward process (top row): data from p_0 is progressively corrupted by adding noise until it becomes indistinguishable from the Gaussian reference p_1 at $t = 1$. Reverse process (bottom row): starting from pure noise, a learned score network s_θ guides the denoising, gradually recovering the bimodal structure of p_0 at $t = 0$.

0.4 State of the Art and Open Problems

We now review the state of the art for each structural regime, identify the gaps in the existing literature, and formulate the objectives of this thesis.

0.4.1 Extreme Value Generation

Classical extreme value theory. The statistical theory of multivariate extremes is well-developed (Paul Embrechts et al. 1997; L. d. Haan and Ferreira 2006; Sidney I. Resnick 1987). For univariate extremes, the Fisher-Tippett-Gnedenko theorem establishes that properly normalized maxima converge to one of three distributions: Gumbel, Fréchet, or Weibull, unified by the Generalized Extreme Value (GEV) distribution. In the multivariate setting, the stable tail dependence function (stdf) provides a complete characterization of extremal dependence, describing how the probability of joint exceedances scales as one moves into the tail. Numerous parametric models have been proposed: the Gumbel (logistic) copula (Gumbel 1960), the Hüsler-Reiss model arising from Gaussian processes (Hüsler and Reiss 1989a), and the extremal-t model generalizing the Hüsler-Reiss to Student-t margins (Nikoloulopoulos et al. 2009). Estimation methods include empirical likelihood (Einmahl, Krajina, et al. 2012), rank-based M-estimation (Einmahl, L. d. Haan, et al. 2016), and Bayesian approaches (Sabourin et al. 2013).

Why standard GANs fail. Standard generative models are fundamentally incompatible with heavy-tailed data. The reason is mathematical: (Wiese et al. 2019) proved that if the latent distribution has light tails (e.g., Gaussian) and the generator G is Lipschitz continuous, then the generated distribution $G(Z)$ necessarily has light tails. Specifically, if Z is sub-Gaussian, then $G(Z)$ is sub-Gaussian for any Lipschitz G. Since modern GANs enforce Lipschitz constraints for training stability (spectral normalization, gradient penalties), they *provably cannot* generate heavy-tailed outputs from Gaussian noise. This is not a failure of optimization or architecture; it is a fundamental impossibility.

Heavy-tailed generative models. Recent work has addressed this limitation through several approaches (Michaël Allouche, Emmanuel Gobet, et al. 2024). For GANs: *preprocessing methods* such as Quant-GAN (Wiese et al. 2019) and evtGAN (Youssef Boulaguiem et al. 2022) transform data to remove tail heaviness before training; *heavy-tailed latent* methods (Huster et al. 2021) replace Gaussian noise with Pareto distributions; *EVT-parameterized* approaches such as EV-GAN (Michaël Allouche, Girard, et al. 2022) design generators that explicitly encode extreme-value structure. For diffusion models, heavy-tailed extensions replace the Gaussian noise process with α-stable Lévy processes (Yoon et al. 2023), allowing direct generation of heavy-tailed samples. However, all these approaches focus on *marginal* tail behavior. None addresses the *dependence structure* in the tails, characterized by the stdf.

Gap 1. To our knowledge, no existing generative models provide provable guarantees on tail dependence approximation. Existing methods address marginal tail behavior but ignore the joint structure. We address this gap in Chapter 1, developing a GAN architecture that provably approximates any stdf.

0.4.2 Order Statistics Generation

The mathematical formulation of order statistics was given in Section 0.2.2. Classical representations (Sukhatme, Schucany) have enabled exact simulation for decades (H. A. David and H. N. Nagaraja 2003; Arnold et al. 2008). We now review the state of the art.

Machine learning approaches. The machine learning literature has addressed sorting from the perspective of learning-to-rank (T.-Y. Liu 2009), where the goal is to predict orderings rather than generate them. Differentiable sorting operators have been developed to enable gradient-based learning: (Grover et al. 2019) proposed NeuralSort, a continuous relaxation of the argsort operator based on the Sinkhorn algorithm; (Cuturi et al. 2019) developed optimal transport-based differentiable sorting; (Blondel et al. 2020) introduced fast differentiable sorting with $O(n \log n)$ complexity. These methods enable sorting within neural network pipelines but are designed for ranking tasks, not distribution matching.

Generative models and ordering. Standard generative models (GANs, VAEs, diffusion) produce unordered samples. When applied to ranked data, they face two problems.

First, generated samples are not sorted: a GAN trained on order statistics will produce vectors where coordinates are not monotonically ordered. Post-hoc sorting can restore order but introduces bias in the distribution. Second, standard preprocessing destroys ordering structure: per-coordinate normalization to $[0, 1]$, common in GAN training, normalizes each coordinate independently based on its marginal range, destroying the relative ordering. If the k-th order statistic is normalized by its own mean and variance, and the $(k+1)$-th by different values, there is no guarantee that normalized values remain sorted.

Gap 2. To our knowledge, no existing generative models produce order statistics with both correct marginal/joint distributions and guaranteed ordering constraints. Existing differentiable sorting methods address ranking prediction, not generative modeling. We address this gap in Chapter 2, developing an architecture that exploits classical representations to guarantee sortedness while matching target distributions.

0.4.3 Reward-Tilted Generation

The alignment problem (Section 0.2.3) requires sampling from the tilted distribution (3). We review existing approaches.

Reinforcement learning from human feedback. The dominant paradigm for alignment is reinforcement learning from human feedback (RLHF) (Christiano et al. 2017; Bai et al. 2022; Ouyang et al. 2022). A reward model is trained on human preference data (pairwise comparisons of outputs), then the generative model is fine-tuned to maximize expected reward using policy gradient methods (PPO, REINFORCE). RLHF has been transformative for large language models, enabling ChatGPT, Claude, and other conversational agents. However, RLHF is notoriously unstable: reward hacking, mode collapse, and catastrophic forgetting are common failure modes (Casper et al. 2023).

Diffusion model fine-tuning. For diffusion models, several fine-tuning approaches have emerged. DRaFT (Clark et al. 2024) backpropagates through the full denoising chain to maximize reward, but this requires storing activations across all timesteps, with memory scaling linearly in chain length. DPOK (Fan et al. 2023) combines policy gradients with KL regularization. ReFL (Jing Xu et al. 2024) uses reward feedback learning to fine-tune text-to-image diffusion models. Adjoint matching (Domingo-Enrich, Drozdzal, et al. 2025) provides a theoretically principled approach based on adjoint SDEs, achieving state-of-the-art results but still requiring reward gradients.

The gradient bottleneck. Gradient-based fine-tuning methods (DRaFT, adjoint matching) require backpropagating through the reward function, which creates practical limitations: storing activations for large reward models leads to prohibitive memory costs, and non-differentiable rewards (human feedback, discrete evaluations, simulator-based metrics) cannot be handled. RL-based approaches (RLHF, DPOK) avoid this by using only reward values, not gradients, but introduce other challenges: policy gradient estimators have high

variance, and careful hyperparameter tuning is needed to balance reward maximization against mode collapse. Neither approach provides distributional guarantees on the resulting fine-tuned model.

Gap 3. To our knowledge, no existing gradient-free methods fine-tune diffusion models to sample from reward-tilted distributions. Existing methods either require reward gradients or use reinforcement learning with high-variance estimators. We address this gap in Chapter 3, developing a method based on Fisher's identity that requires only forward evaluations of the reward function.

0.4.4 Objectives of This Thesis

Based on the identified gaps, we formulate three objectives:

1. **Extreme value generation.** Develop a generative model for multivariate extremes with provable guarantees on stdf approximation. The method should use heavy-tailed latent noise and provide explicit bounds on the approximation error in terms of network architecture.

2. **Order statistics generation.** Develop a generative model for order statistics that guarantees sortedness while matching the target distribution. The method should exploit classical representations and provide approximation bounds that scale favorably with dimension.

3. **Reward-tilted generation.** Develop a gradient-free method for fine-tuning diffusion models to reward-tilted distributions. The method should require only forward evaluations of the reward function.

0.5 Contributions of This Thesis

We now summarize the main contributions, organized by the three objectives.

0.5.1 Heavy-Tailed GANs for Extreme Value Generation

We develop Heavy-Tailed GANs (HTGAN) for generating data with correct tail dependence.

Theoretical contributions. We replace Gaussian latent noise with heavy-tailed (Pareto or Fréchet) noise. Because the generator is Lipschitz, the heavy-tail property propagates to the output. We prove that the stdf of the generated distribution is always discrete, a maximum of linear functions with finitely many atoms. Our main approximation theorem establishes that any stdf can be approximated within L^∞ error ε using latent dimension $N = O(\varepsilon^{-(d-1)})$ and a single hidden layer with ReLU activations. This appears to be the first approximation guarantee for tail dependence in the generative modeling literature.

Methodological contributions. The HTGAN algorithm proceeds in three steps: marginal transformation to unit Fréchet, GAN training with Pareto latent noise, and inverse transformation. This separation of margins and dependence is standard in copula modeling.

Experimental contributions. We evaluate on synthetic Gumbel copula data and S&P 500 returns, demonstrating that HTGAN correctly captures tail dependence while standard GANs fail by orders of magnitude.

0.5.2 Generative Neural Order Statistics

We develop Generative Neural Order Statistics (GENOS) for generating ranked data with guaranteed ordering.

Theoretical contributions. We combine classical statistics representations with neural network approximation theory (Yarotsky 2017) to derive non-asymptotic complexity bounds. Specifically, we adapt two classical representations of order statistics and analyze their neural network approximation complexity.

Schucany's representation (Schucany 1972) constructs order statistics recursively:

$$U_{n-r:n} = U_{n-r+1:n} \cdot U_r^{1/(n-r)}, \quad r = 1, \dots, n-1. \tag{6}$$

This requires n different transition functions. We prove that neural approximation of this scheme requires a number of parameters that is polynomial of degree n in ε^{-1}.

Sukhatme's representation (Sukhatme 1937) expresses uniform order statistics via exponential spacings:

$$U_{k:n} = F\left(\sum_{j=1}^{k} \frac{E_j}{n-j+1}\right), \quad E_j \overset{\text{iid}}{\sim} \text{Exp}(1), \quad F(z) = 1 - e^{-z}. \tag{7}$$

The cumulative sum has an exponential distribution; applying the exponential CDF F yields the k-th uniform order statistic $U_{k:n}$. This requires approximating only two universal functions (F and F^{-1}), independent of n. We prove that the Sukhatme-based approach achieves

$$\mathbb{E}|U_{k:n} - \hat{U}_{k:n}| \leqslant O(\varepsilon \log n) \quad \text{with} \quad \mathcal{C}_{\text{Sukhatme}}(\varepsilon) = O(\varepsilon^{-2} \log(1/\varepsilon)) \tag{8}$$

parameters. The complexity is quadratic in ε^{-1}, independent of n, versus polynomial of degree n for Schucany. This makes the Sukhatme-based approach scalable to large sample sizes.

Methodological contributions. We introduce global mean-scale normalization, which preserves ordering unlike per-coordinate normalization. We also propose an order statistics

penalty (OSP) that penalizes violations of the ordering constraint:

$$\mathcal{L}_{\text{OSP}} = \mathbb{E}_Z \left[\frac{1}{n-1} \sum_{k=1}^{n-1} \left(\max\{G(Z)_k - G(Z)_{k+1}, 0\} \right)^2 \right].$$

Experimental contributions. We evaluate on synthetic order statistics with varying dimension, comparing GAN, WGAN, and WGAN with OSP. Adding the OSP penalty improves sortedness metrics while maintaining competitive performance on Wasserstein distance.

0.5.3 Gradient-Free Fine-Tuning of Diffusion Models

We develop gradient-free methods for fine-tuning diffusion models to sample from reward-tilted distributions.

Theoretical contributions. Fisher's identity expresses the score of the tilted distribution as a covariance:

$$\nabla_x \log p(x) = \nabla_x \log p_0(x) + \lambda \cdot \text{Cov}_{q_{0|x}} \left[\nabla_{x_0} \log q(x_0|x),\, r(x_0) \right]. \tag{9}$$

This covariance can be estimated via Monte Carlo using only forward evaluations of r, without backpropagation through the reward model.

Methodological contributions. We propose an iterative algorithm that decomposes a large tilt into N small steps, estimating the score increment at each step via Fisher's identity. The algorithm applies even when r is non-differentiable or a black-box.

Experimental contributions. On 2D Gaussian mixtures with known ground truth, we validate that the method correctly recovers the tilted score and that increasing the number of steps N improves accuracy.

0.6 Organization of the Manuscript

Chapter 1 develops the theory and algorithms for heavy-tailed generative modeling. We present mathematical background on multivariate extreme value theory and the stdf, analyze why Lipschitz maps of Gaussian noise cannot produce heavy tails, characterize the discrete stdf induced by generators with Pareto noise, prove our main approximation theorem, and present experiments on synthetic and financial data.

Chapter 2 develops generative models for order statistics. We present background on order statistics and the Sukhatme and Schucany representations, prove approximation bounds for ReLU networks, introduce global mean-scale normalization and the WGAN-OSP objective, and present experiments on synthetic order statistics.

Chapter 3 develops gradient-free fine-tuning for diffusion models. We present background on exponential tilting and existing gradient-based methods, derive the Fisher identity-based score estimator, analyze the approximation error, present the iterative tilting algorithm, and validate on controlled experiments.

This thesis is partially based on collaborative work: Chapter 1 on heavy-tailed generative models, Chapter 2 on neural approximation of order statistics, and Chapter 3 on gradient-free diffusion fine-tuning. Specific venues and co-authors are indicated at the beginning of each chapter.

Introduction (Version française)

0.7 Contexte et motivation

0.7.1 La structure dans les données statistiques

Le terme *statistique* dérive de l'allemand *Statistik*, forgé par Gottfried Achenwall en 1748 pour désigner la « science de l'État » : gouverner nécessitait une information organisée. Les données sont la matière première de la statistique. Un scientifique mesure la température en plusieurs endroits ; une banque enregistre les rendements journaliers des actifs d'un portefeuille ; un hôpital collecte les signes vitaux des patients. Dans chaque cas, nous disposons de plusieurs observations, et chaque observation peut comprendre plusieurs quantités liées entre elles. Ces quantités sont organisées de manières spécifiques, régies par des régularités sous-jacentes. Découvrir ces régularités est la tâche de la science statistique. Ce processus admet une formulation mathématique claire. Nous notons n observations par $X_1, \ldots, X_n$. Chaque X_i est un *vecteur* : c'est à dire une liste de nombres représentant les quantités mesurées. Nous supposons que ces observations sont gouvernées par une loi sous-jacente. Mathématiquement, cette loi est une *distribution de probabilité*, que nous notons P. La distribution spécifie la probabilité de chaque résultat possible. Elle est inconnue ; apprendre à la connaître est le problème central.

L'objectif de la *statistique* est d'inférer des propriétés de P à partir des observations : estimer ses paramètres, tester des hypothèses sur sa forme, quantifier l'incertitude. L'objectif de la *modélisation générative* est plus ambitieux : produire de nouvelles données synthétiques qui auraient plausiblement pu provenir de la même source, indiscernables des observations authentiques.

Mais les données portent une *structure* : des dépendances entre variables, des contraintes sur leurs valeurs, des propriétés qui définissent le phénomène lui-même. Un modèle génératif fidèle doit respecter cette structure, et non simplement approximer la loi globale.

0.7.2 La modélisation générative moderne

À partir de données, pouvons-nous apprendre à en générer davantage ? Non pas simplement calculer des moyennes, mais produire de nouvelles données qui semblent provenir de la

même source ? C'est la question de la *modélisation générative*.

Lorsque les observations comportent de nombreuses composantes (températures en des centaines de lieux, rendements de milliers d'actifs), décrire directement la loi P devient impraticable. La modélisation générative moderne emprunte une voie différente : plutôt que de décrire P explicitement, nous apprenons à produire des données qui en sont issues. L'idée centrale est :

$$G(Z) \sim P.$$

Ici, Z est une source d'aléa, comme lancer des dés ou tirer selon une courbe en cloche. La transformation G convertit cet aléa en quelque chose qui ressemble aux données. Toute la complexité de la loi inconnue P est encodée dans G.

Cela fait écho à une idée classique. Un statisticien suppose que les données suivent une famille connue de lois (courbes en cloche, exponentielles, etc.) et estime quelques paramètres pour déterminer laquelle. Un modélisateur génératif fait un pari similaire : au lieu de supposer que la loi a une certaine forme, nous supposons que la transformation G en a une. Quand G est un réseau de neurones, nous parions que transformer de l'aléa à travers des couches de calculs peut imiter les données.

La percée de la dernière décennie est que les réseaux de neurones peuvent apprendre G à partir d'exemples, produisant des images indiscernables de photographies, du texte qui se lit comme écrit par un humain. Mais le revers de la médaille : les réseaux de neurones apprennent G en minimisant une mesure globale d'erreur. Ce processus n'a pas de mécanisme pour préserver des propriétés structurelles spécifiques.

0.7.3 Le problème : quand la structure compte

Que signifie pour des données synthétiques d'être *fidèles* au phénomène qu'elles imitent ?

Considérons une banque générant de faux scénarios de crise pour tester ses investissements. Les scénarios générés semblent réalistes en moyenne, mais les pires pertes surviennent isolément : quand les actions s'effondrent, les obligations restent calmes. Les données synthétiques capturent chaque variable séparément mais manquent la façon dont elles évoluent ensemble. Dans une vraie crise, les pertes sont corrélées. Ou considérons la génération de classements synthétiques : le troisième meilleur élément obtient parfois un score supérieur au premier. Le classement est brouillé ; la simulation est inutile.

Ces échecs reflètent un décalage entre ce que les modèles génératifs optimisent et ce que les applications requièrent. Les modèles standards récompensent l'obtention de la forme globale correcte mais ne fournissent aucune garantie sur des propriétés structurelles spécifiques.

0.7.4 Cette thèse : méthodes génératives guidées par la théorie

Ce manuscrit part de problèmes réels où la structure compte. Nous identifions la propriété clé qui doit être préservée (effondrements conjoints, classements, préférences), la formulons comme un objet mathématique, et analysons dans quelle mesure les réseaux de neurones

peuvent l'approximer. C'est la méthodologie centrale : du problème appliqué, à l'abstraction mathématique, à l'analyse théorique.

Nous étudions trois régimes structurels : la dépendance de queue, les statistiques d'ordre et l'inclinaison par récompense. Pour chacun, nous concevons des méthodes génératives qui respectent *prouvablement* la structure, fournissant des garanties sur la précision d'approximation.

0.8 Trois régimes structurels

Nous décrivons maintenant les trois régimes structurels abordés dans cette thèse. Pour chaque régime, nous présentons le problème réel qui motive notre travail et le formalisons mathématiquement.

0.8.1 Valeurs extrêmes et dépendance de queue

Le problème : quand les extrêmes gouvernent tout. Dans de nombreux domaines, les événements extrêmes dominent les résultats malgré leur rareté. En gestion de portefeuille, une poignée de grandes pertes peut effacer des années de gains réguliers : la distribution des rendements est à queue lourde, et le risque est concentré dans les extrêmes (Rama Cont 2001). La Value-at-Risk, les tests de stress et les exigences réglementaires en capital reposent tous sur la probabilité de pertes rares mais catastrophiques. La crise financière de 2008 a démontré que les modèles calibrés sur des conditions normales échouent lorsque des extrêmes surviennent (A. J. McNeil et al. 2015).

En assurance, les événements extrêmes sont le cœur de métier. Les réassureurs doivent tarifer le risque de catastrophes simultanées : un ouragan frappant la Floride pendant qu'un tremblement de terre frappe la Californie. C'est l'occurrence conjointe de tels événements, et non leurs probabilités individuelles, qui détermine les exigences de solvabilité. En science du climat, les extrêmes composés (vagues de chaleur et sécheresses simultanées, événements d'inondation séquentiels) posent les plus grands risques aux écosystèmes et aux infrastructures (Jakob Zscheischler et al. 2018 ; Bevacqua et al. 2021). Les modèles climatiques doivent capturer non seulement la fréquence des extrêmes individuels, mais leur tendance à se regrouper dans l'espace et le temps.

Le fil conducteur est la *dépendance de queue* : la tendance des extrêmes à survenir ensemble. Deux variables aléatoires peuvent avoir une corrélation nulle dans des conditions normales tout en exhibant une forte dépendance de queue : quand l'une prend une valeur extrême, l'autre a tendance à en faire autant (voir la Figure 8). Pour un gestionnaire de portefeuille, cela signifie que la diversification échoue précisément quand elle est le plus nécessaire. Pour un assureur, cela signifie des sinistres corrélés. Pour le risque climatique, cela signifie des catastrophes composées.

Deux défis distincts. Générer des données de valeurs extrêmes réalistes pose deux défis fondamentalement différents.

Le premier est la *lourdeur marginale des queues*. Une distribution est à queue lourde quand les valeurs extrêmes sont bien plus probables que ce que les modèles à queue légère prédiraient. La Figure 7 compare trois distributions : la gaussienne absolue $|X|$, l'exponentielle (toutes deux à queue légère) et la Pareto (à queue lourde). Près de zéro, elles se ressemblent, mais leurs queues se comportent très différemment. Le Tableau 2 montre la probabilité de dépasser différents seuils. À $t = 8$, la gaussienne attribue une probabilité de 10^{-15}, l'exponentielle 10^{-4}, mais la Pareto donne encore 3,7%. Les phénomènes à queue lourde produisent des événements extrêmes que les modèles à queue légère considèrent comme virtuellement impossibles.

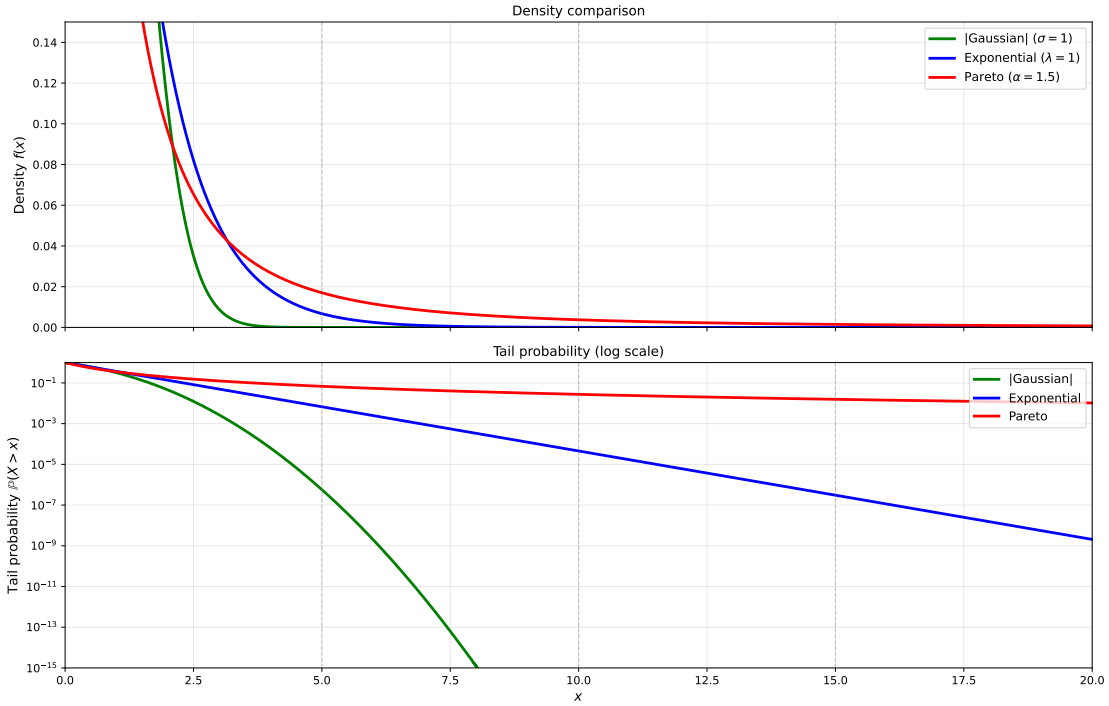


FIGURE 7 : Distributions à queue légère vs à queue lourde. Gauche : comparaison des densités pour |Gaussienne|, exponentielle et Pareto. Droite : probabilité de queue $\mathbb{P}(X > t)$ en échelle logarithmique. Les queues gaussienne et exponentielle décroissent rapidement ; la queue de Pareto décroît comme une loi de puissance, restant substantielle même pour de grandes valeurs de t.

TABLE 2 : Probabilité de dépasser le seuil t pour |Gaussienne| ($\sigma = 1$), exponentielle ($\lambda = 1$) et Pareto ($\alpha = 1{,}5$).

t	|Gaussienne|	Exponentielle	Pareto
3	$2{,}7 \times 10^{-3}$	$5{,}0 \times 10^{-2}$	$1{,}3 \times 10^{-1}$
5	$5{,}7 \times 10^{-7}$	$6{,}7 \times 10^{-3}$	$6{,}8 \times 10^{-2}$
8	$1{,}3 \times 10^{-15}$	$3{,}4 \times 10^{-4}$	$3{,}7 \times 10^{-2}$

Le second défi est la *dépendance extrémale* : quand plusieurs variables sont impliquées, leur comportement conjoint dans les queues peut être entièrement différent de leur dépendance dans des conditions normales. La Figure 8 illustre ceci : deux échantillons avec des

distributions marginales identiques se comportent très différemment dans les queues selon que les extrêmes surviennent ensemble ou indépendamment.

Comme nous le verrons dans la Section 0.10.1, les modèles génératifs standards échouent sur ces deux aspects.

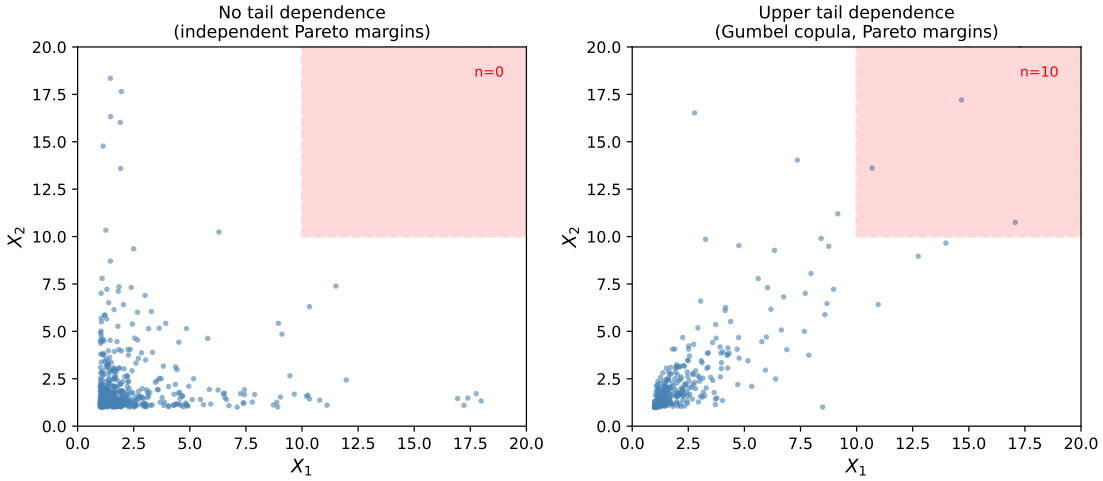


FIGURE 8 : Dépendance de queue illustrée avec des marges de Pareto ($\alpha = 1{,}5$). Gauche : marges indépendantes ; les valeurs extrêmes de X_1 et X_2 surviennent indépendamment, donc la région de queue supérieure (boîte rouge) est clairsemée. Droite : copule de Gumbel ($\theta = 2{,}5$) avec dépendance de queue supérieure ; quand X_1 est extrême, X_2 tend à être extrême également, créant un regroupement dans le coin supérieur droit. Les deux échantillons ont des distributions marginales identiques, mais se comportent très différemment dans les queues.

Formulation mathématique : la fonction de dépendance de queue stable. En statistique classique, la *corrélation* mesure comment deux variables évoluent ensemble en moyenne. La corrélation résume la dépendance sur l'ensemble de la distribution mais ne caractérise pas complètement le comportement des queues. Deux variables peuvent avoir une corrélation nulle tout en s'effondrant ensemble dans une crise ; inversement, des variables avec la même corrélation peuvent exhiber des structures de dépendance de queue très différentes.

La *fonction de dépendance de queue stable* (stdf, de l'anglais *stable tail dependence function*) joue pour les extrêmes le rôle que la corrélation joue pour le corps de la distribution. Tout comme la corrélation capture le co-mouvement moyen, la stdf capture le co-mouvement extrême : quand une variable prend une très grande valeur, quelle est la probabilité que l'autre soit grande également ? Pour un vecteur aléatoire $\mathbf{X} = (X_1, \ldots, X_d)$, la stdf ℓ est définie par

$$\ell(\mathbf{x}) = \lim_{t \to \infty} t\, \mathbb{P}\left(\bigcup_{j=1}^{d} \{X_j > t x_j\} \right). \tag{10}$$

Cette limite capture la probabilité asymptotique des dépassements conjoints lorsque l'on s'enfonce dans la queue. La stdf caractérise complètement la dépendance extrémale : deux distributions avec la même stdf exhibent un comportement identique dans les extrêmes,

indépendamment de leur comportement dans le corps de la distribution. Connaître la stdf est essentiel pour comprendre la diversification quand elle compte le plus : pendant les événements extrêmes.

0.8.2 Statistiques d'ordre et contraintes de classement

Le problème : quand le classement définit les données. De nombreux phénomènes sont définis par leur structure de rang. Dans les enchères, ce qui compte est la distribution de l'offre la plus élevée, ou l'écart entre les deux offres les plus élevées. En sport, nous nous intéressons à la performance du joueur le mieux classé, pas à la moyenne. En statistique des valeurs extrêmes, les k plus grandes observations sont utilisées pour estimer le comportement des queues. Dans chaque cas, l'ordonnancement n'est pas accessoire : c'est la structure définissante des données.

Considérons l'investissement à impact, où les actifs sont classés par scores environnementaux, sociaux ou de gouvernance (ESG) (Lo et R. Zhang 2021). Un gestionnaire de fonds testant rétrospectivement une stratégie ESG doit simuler les rendements du meilleur actif classé, du deuxième meilleur, et ainsi de suite. Ce sont des *statistiques d'ordre induites* : les rendements $\Theta_{[1:N]}, \ldots, \Theta_{[N:N]}$ associés aux actifs classés par leurs scores ESG $X_{1:N} \leqslant \cdots \leqslant X_{N:N}$. Les données synthétiques doivent préserver à la fois la distribution marginale à chaque rang et la dépendance conjointe entre les rangs.

Formulation mathématique : statistiques d'ordre et leur densité conjointe. Étant donné un échantillon $X_1, \ldots, X_n$ d'une distribution continue F de densité f, les *statistiques d'ordre* sont les valeurs triées $X_{1:n} \leqslant X_{2:n} \leqslant \cdots \leqslant X_{n:n}$. Leur densité conjointe est

$$f_{1:n,\ldots,n:n}(x_1, \ldots, x_n) = n! \prod_{i=1}^{n} f(x_i) \cdot \mathbf{1}\{x_1 \leqslant \cdots \leqslant x_n\}. \tag{11}$$

Le facteur $n!$ compte le nombre d'arrangements, et l'indicatrice $\mathbf{1}\{\cdot\}$ impose la contrainte d'ordonnancement. Les représentations classiques, telles que celles de Sukhatme et Schucany, décomposent les statistiques d'ordre en blocs élémentaires tractables qui ont permis la simulation exacte depuis des décennies. Comme nous le verrons dans la Section 0.11.2, ces représentations fournissent également les fondements pour l'approximation par réseaux de neurones avec des bornes de complexité démontrables.

0.8.3 Distributions inclinées par récompense et ajustement fin de modèles

Le problème : aligner les modèles génératifs avec les préférences humaines. Un modèle génératif pré-entraîné produit des échantillons d'une distribution apprise à partir des données d'entraînement. Mais pour de nombreuses applications, nous voulons des échantillons qui ne sont pas simplement typiques, mais *bons* selon un certain critère : des images esthétiquement plaisantes, du texte utile et inoffensif, des molécules qui se lient

à une protéine cible. C'est le *problème d'alignement* : ajuster un modèle génératif pour favoriser les sorties de haute qualité tout en maintenant la diversité et la cohérence.

Considérons un modèle de diffusion texte-vers-image entraîné sur des images collectées sur le web. Le modèle génère des images réalistes, mais beaucoup sont mal composées, contiennent des artefacts ou représentent du contenu indésirable. Nous avons accès à un modèle de récompense (entraîné sur les préférences humaines) qui note les images selon leur qualité esthétique et leur sûreté. L'objectif est d'affiner le modèle de diffusion pour qu'il génère préférentiellement des images à haute récompense, sans s'effondrer vers un mode unique ni oublier comment générer du contenu diversifié.

Formulation mathématique : inclinaison exponentielle. La formulation mathématique naturelle est l'*inclinaison exponentielle*. Étant donné une distribution de base p_0 (le modèle pré-entraîné) et une fonction de récompense $r : \mathcal{X} \to \mathbb{R}$, la cible est la *distribution inclinée*

$$p(x) \propto p_0(x) \exp(\lambda\, r(x)), \tag{12}$$

où $\lambda > 0$ contrôle l'intensité de l'inclinaison. Cette distribution est l'unique solution du problème de maximisation de récompense régularisé par KL :

$$p = \arg\max_q \left\{ \mathbb{E}_q[r(x)] - \frac{1}{\lambda} \mathrm{KL}(q \| p_0) \right\}.$$

La distribution inclinée équilibre la maximisation de la récompense contre le fait de rester proche du modèle pré-entraîné, avec λ contrôlant le compromis. Un grand λ favorise la haute récompense ; un petit λ reste proche de p_0.

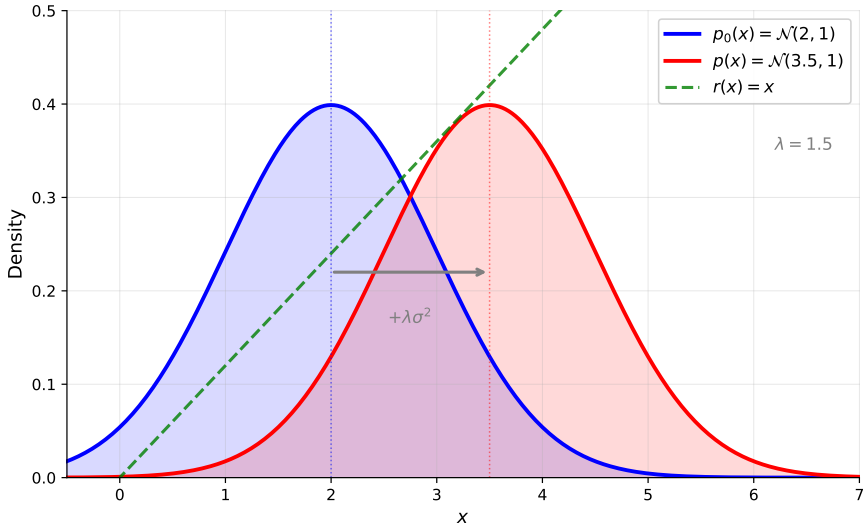


FIGURE 9 : L'inclinaison par récompense déplace la distribution vers les régions à haute récompense. Avec $p_0 = \mathcal{N}(2, 1)$, récompense linéaire $r(x) = x$, et intensité d'inclinaison $\lambda = 1{,}5$, la distribution inclinée $p(x) \propto p_0(x) \exp(\lambda r(x))$ est $\mathcal{N}(3{,}5, 1)$: la moyenne se décale de $\lambda\sigma^2 = 1{,}5$ vers la haute récompense, tandis que la variance reste inchangée.

0.9 Contexte de la modélisation générative

La Section 0.7 a introduit l'objectif d'apprendre à partir des données tout en respectant la structure. La Section 0.8 a présenté trois régimes où la structure compte : la dépendance de queue, les statistiques d'ordre et l'inclinaison par récompense. Pour relever ces défis, nous nous tournons vers la modélisation générative : la tâche de produire des données synthétiques qui suivent la même loi que les observations. Cette section fournit le contexte technique sur les modèles génératifs et la théorie d'approximation par réseaux de neurones nécessaire pour comprendre les contributions de cette thèse.

0.9.1 Le problème de la modélisation générative

Le problème fondamental de la modélisation générative peut être énoncé comme suit : étant donné des échantillons $X_1, \ldots, X_n$ tirés indépendamment d'une distribution inconnue P sur $\mathbb{R}^d$, construire une procédure pour générer de nouveaux échantillons qui sont distribués selon P, ou une approximation proche de celle-ci.

Ce problème a une longue histoire en statistique et probabilités. Les approches classiques incluent l'*estimation de densité* (Silverman 1986 ; Scott 2015), où l'on estime la densité $p = dP/d\lambda$ puis on échantillonne à partir de la densité estimée, et les *chaînes de Markov Monte Carlo* (MCMC) (Metropolis et al. 1953 ; Hastings 1970 ; C. P. Robert et Casella 2004), où l'on construit une chaîne de Markov ayant P comme distribution stationnaire et on échantillonne en faisant tourner la chaîne jusqu'à l'équilibre. Les deux approches font face à la malédiction de la dimension (Bellman 1961) : l'estimation de densité devient statistiquement intractable en haute dimension, tandis que les temps de mélange MCMC peuvent croître exponentiellement avec la dimension.

L'approche moderne de la modélisation générative, rendue possible par l'apprentissage profond, emprunte une voie différente. Plutôt qu'estimer explicitement la densité, nous apprenons une *application de transport* G qui transforme des échantillons d'une distribution de référence simple (typiquement gaussienne standard) en échantillons de la distribution cible. Si Z est un bruit gaussien, alors $G(Z)$ devrait être distribué approximativement comme P. L'application G est paramétrée par un réseau de neurones, et les paramètres sont appris à partir des données.

Cette vision basée sur le transport unifie plusieurs paradigmes génératifs modernes. Dans les *flots normalisants* (Rezende et Mohamed 2015 ; Dinh et al. 2017), l'application G est contrainte à être inversible avec un jacobien tractable. Dans les *autoencodeurs variationnels* (VAE) (Diederik P. Kingma et Max Welling 2014), l'application est apprise conjointement avec une inverse approximative. Dans les *réseaux antagonistes génératifs* (GAN) (I. J. Goodfellow et al. 2014), l'application est apprise via un objectif adversarial. Dans les *modèles de diffusion* (Ho, Jain et al. 2020 ; Y. Song, Jascha Sohl-Dickstein et al. 2021), l'application est implicitement définie par l'inverse d'un processus de bruitage. Nous décrivons les GAN et les modèles de diffusion en détail ci-dessous, car ils sont les fondements de nos contributions.

0.9.2 Réseaux de neurones

Un *réseau de neurones feedforward* à L couches cachées transforme une entrée x à travers des couches successives (Figure 10). En notant $h_0 = x$, les représentations cachées sont

$$h_\ell = \sigma(W_\ell h_{\ell-1} + b_\ell) \quad \text{pour } \ell = 1, 2, \ldots, L, \tag{13}$$

où W_ℓ est une matrice de poids, b_ℓ un vecteur de biais, et σ une fonction d'activation non linéaire appliquée coordonnée par coordonnée. La sortie est $g(x) = W_{L+1} h_L + b_{L+1}$. Les activations courantes incluent la sigmoïde, la tangente hyperbolique et la ReLU $\sigma(t) = \max(0, t)$, qui domine la pratique moderne.

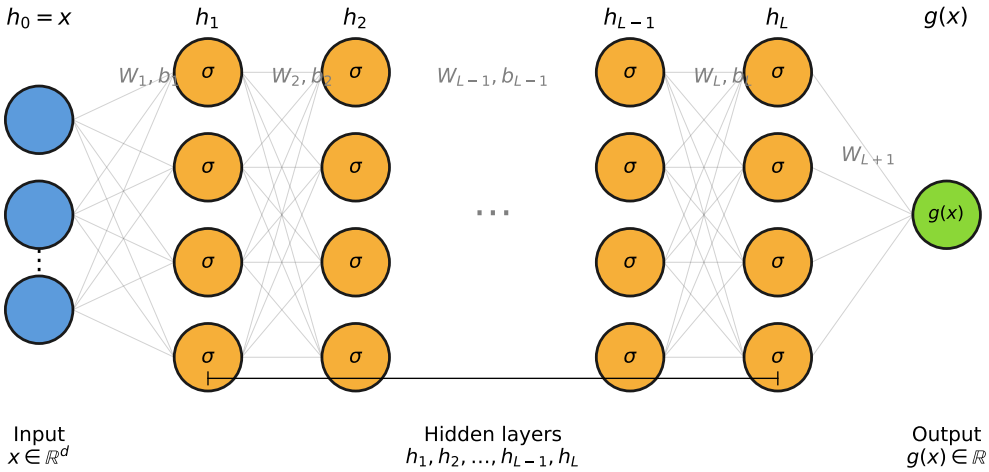


FIGURE 10 : Un réseau de neurones feedforward. L'entrée $h_0 = x$ passe à travers L couches cachées, chacune appliquant une transformation affine suivie de l'activation σ. La sortie $g(x)$ est une fonction linéaire de h_L.

0.9.3 Théorie d'approximation des réseaux de neurones

Le *théorème d'approximation universelle* (CYBENKO 1989 ; HORNIK et al. 1989) établit que les réseaux de neurones à une seule couche cachée peuvent approximer toute fonction continue sur un domaine compact avec une précision arbitraire, étant donnée une largeur suffisante. Cela s'étend aux réseaux ReLU (LESHNO et al. 1993).

Bornes quantitatives. Les bornes quantitatives relient l'erreur d'approximation à la taille du réseau (BARRON 1993 ; YAROTSKY 2017). Pour les fonctions lipschitziennes sur $[0, 1]^d$, atteindre une erreur ε nécessite $O(\varepsilon^{-d})$ paramètres, exhibant la malédiction de la dimension. Pour des fonctions plus régulières (classe de Hölder avec régularité s), des réseaux avec $O(\varepsilon^{-d/s})$ paramètres suffisent (YAROTSKY 2017).

Limitations. Ces théorèmes concernent l'approximation ponctuelle de fonctions, pas la préservation de structure distributionnelle. Un réseau approximant une application

de transport G uniformément peut néanmoins échouer à préserver le comportement des queues, les contraintes d'ordonnancement ou la structure de dépendance. De plus, les théorèmes supposent que les paramètres optimaux sont connus ; en pratique, la descente de gradient peut ne pas les trouver. Ces limitations motivent la théorie spécifique à la structure développée dans cette thèse.

0.9.4 Réseaux antagonistes génératifs

Les réseaux antagonistes génératifs (I. J. Goodfellow et al. 2014) formulent la modélisation générative comme un jeu à deux joueurs. L'intuition se comprend mieux à travers une analogie : imaginez un faussaire essayant de produire de faux billets de banque, et un détective essayant de les détecter. Le faussaire s'améliore en étudiant quels faux sont détectés ; le détective s'améliore en étudiant les dernières techniques du faussaire. Au fil du temps, les deux deviennent plus sophistiqués, jusqu'à ce que finalement le faussaire produise des billets si convaincants que même le détective expert ne peut les distinguer de la vraie monnaie.

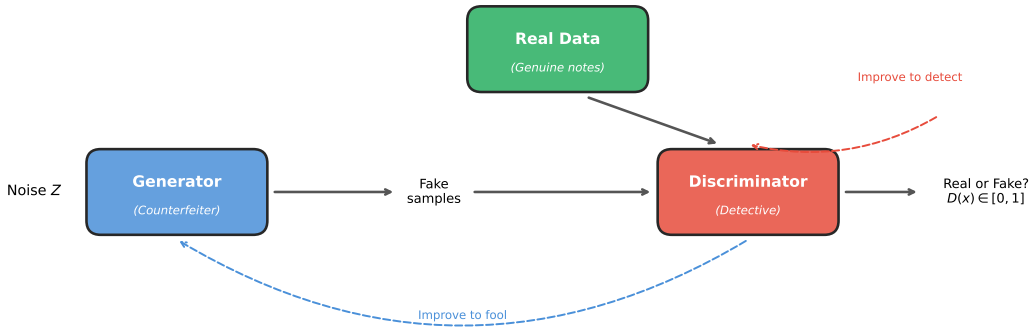


Figure 11 : Le cadre GAN comme jeu adversarial. Le générateur (faussaire) transforme du bruit en faux échantillons ; le discriminateur (détective) tente de distinguer les faux des vraies données. Les deux réseaux s'améliorent par compétition jusqu'à l'équilibre, quand les échantillons générés sont indiscernables des vraies données.

En termes mathématiques, un *générateur* $G_\theta : \mathcal{Z} \to \mathcal{X}$ transforme un bruit latent $Z \sim p_Z$ en échantillons synthétiques (la production du faussaire). Un *discriminateur* $D_\phi : \mathcal{X} \to [0, 1]$ tente de distinguer les échantillons réels des générés (le verdict du détective). Les réseaux sont entraînés en résolvant un problème minimax :

$$\min_\theta \max_\phi \; \mathbb{E}_{X \sim p_{\text{data}}}[\log D_\phi(X)] + \mathbb{E}_{Z \sim p_Z}[\log(1 - D_\phi(G_\theta(Z)))].$$

Le générateur minimise cet objectif (essayant de tromper le discriminateur), tandis que le discriminateur le maximise (essayant de classifier correctement réel vs. faux). À l'équilibre, le générateur produit des échantillons de la distribution des données, et le discriminateur sort $1/2$ pour toutes les entrées, incapable de distinguer le réel du faux.

Le GAN de Wasserstein (WGAN) (Arjovsky et al. 2017) remplace la divergence de

Jensen-Shannon par la distance de Wasserstein-1 :

$$W_1(p_{\text{data}}, p_\theta) = \sup_{\|f\|_L \leqslant 1} \mathbb{E}_{X \sim p_{\text{data}}}[f(X)] - \mathbb{E}_{Z \sim p_Z}[f(G_\theta(Z))],$$

où le supremum est pris sur les fonctions 1-lipschitziennes. Le WGAN avec pénalité de gradient (WGAN-GP) (Gulrajani et al. 2017) impose la contrainte de Lipschitz via un terme de pénalité, améliorant la stabilité de l'entraînement.

0.9.5 Modèles de diffusion et basés sur le score

Les modèles de diffusion (Ho, Jain et al. 2020 ; Y. Song, Jascha Sohl-Dickstein et al. 2021) définissent un processus génératif en construisant un chemin continu $(p_t)_{t\in[0,1]}$ entre la distribution des données p_0 et une référence tractable $p_1 = \mathrm{N}(0, \mathrm{I}_d)$. Le chemin est défini via l'*interpolation directe*

$$X_t = \alpha_t X_0 + \sigma_t X_1, \quad X_0 \sim p_0, \quad X_1 \sim \mathrm{N}(0, \mathrm{I}_d), \tag{14}$$

où (α_t, σ_t) sont des programmes déterministes avec $(\alpha_0, \sigma_0) = (1, 0)$ et $(\alpha_1, \sigma_1) = (0, 1)$. À $t = 0$, nous avons des données propres ; à $t = 1$, du pur bruit. Les choix standards incluent le programme préservant la variance ($\alpha_t^2 + \sigma_t^2 = 1$) et le programme linéaire ($\alpha_t = 1 - t$, $\sigma_t = t$).

Pour générer des échantillons, nous inversons ce processus. La quantité clé est la *fonction de score* $\nabla_x \log p_t$, qui pointe vers les régions de haute densité à chaque niveau de bruit. Pour le noyau direct gaussien, le score satisfait $\nabla_x \log p_t(x_t) = -\hat{x}_1(x_t, t)/\sigma_t$, où $\hat{x}_1(x_t, t) = \mathbb{E}[X_1 \mid X_t = x_t]$ est l'espérance conditionnelle du bruit sachant l'échantillon bruité.

Le score est appris via l'*appariement de score par débruitage* (Hyvärinen 2005 ; Vincent 2011) : un réseau $s_\theta(x, t)$ est entraîné à prédire le bruit X_1 à partir de l'échantillon bruité X_t, ce qui équivaut à apprendre $\nabla_x \log p_t$. En partant de $X_1 \sim \mathrm{N}(0, \mathrm{I}_d)$ et en débruitant itérativement en utilisant le score appris, on obtient des échantillons de p_0.

0.10 État de l'art et problèmes ouverts

Nous passons maintenant en revue l'état de l'art pour chaque régime structurel, identifions les lacunes dans la littérature existante et formulons les objectifs de cette thèse.

0.10.1 Génération de valeurs extrêmes

Théorie classique des valeurs extrêmes. La théorie statistique des extrêmes multivariés est bien développée (Paul Embrechts et al. 1997 ; L. d. Haan et Ferreira 2006 ; Sidney I. Resnick 1987). Pour les extrêmes univariés, le théorème de Fisher-Tippett-Gnedenko établit que les maxima correctement normalisés convergent vers l'une des trois

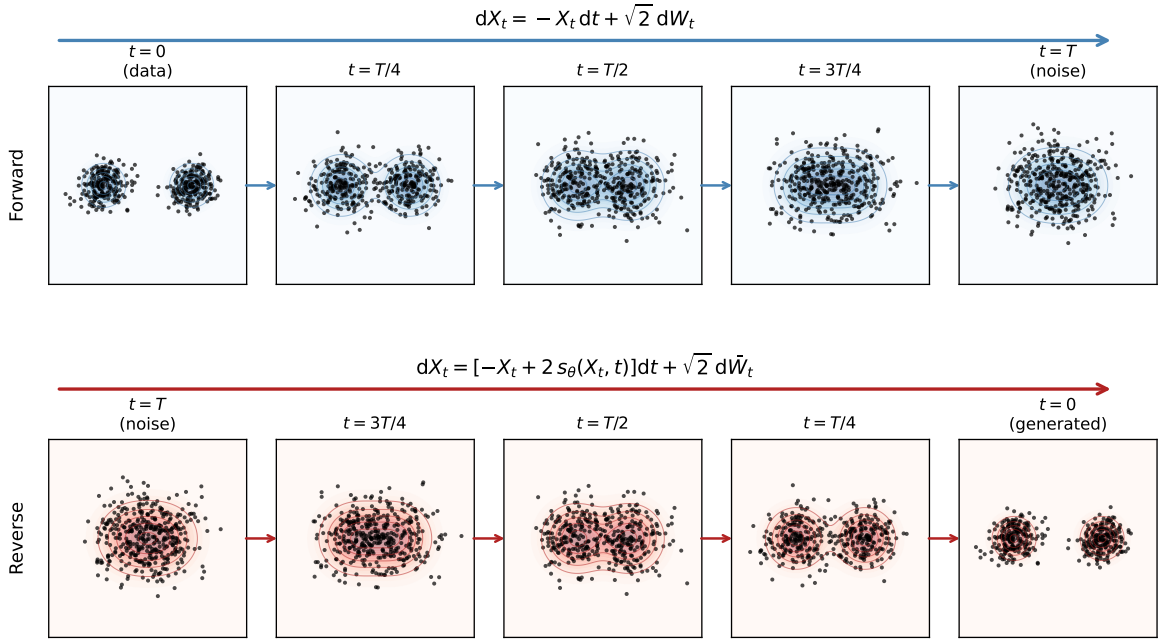


FIGURE 12 : Le cadre des modèles de diffusion sur un mélange de gaussiennes 2D. Processus direct (ligne du haut) : les données de p_0 sont progressivement corrompues par ajout de bruit jusqu'à devenir indiscernables de la référence gaussienne p_1 à $t = 1$. Processus inverse (ligne du bas) : partant du bruit pur, un réseau de score appris s_θ guide le débruitage, récupérant graduellement la structure bimodale de p_0 à $t = 0$.

distributions : Gumbel, Fréchet ou Weibull, unifiées par la distribution des valeurs extrêmes généralisée (GEV). Dans le cadre multivarié, la fonction de dépendance de queue stable (stdf) fournit une caractérisation complète de la dépendance extrémale, décrivant comment la probabilité de dépassements conjoints évolue lorsque l'on s'enfonce dans la queue. De nombreux modèles paramétriques ont été proposés : la copule de Gumbel (logistique) (GUMBEL 1960), le modèle de Hüsler-Reiss provenant des processus gaussiens (HÜSLER et REISS 1989a), et le modèle extrémal-t généralisant Hüsler-Reiss aux marges de Student (NIKOLOULOPOULOS et al. 2009). Les méthodes d'estimation incluent la vraisemblance empirique (EINMAHL, KRAJINA et al. 2012), la M-estimation basée sur les rangs (EINMAHL, L. d. HAAN et al. 2016) et les approches bayésiennes (SABOURIN et al. 2013).

Pourquoi les GAN standards échouent. Les modèles génératifs standards sont fondamentalement incompatibles avec les données à queue lourde. La raison est mathématique : (WIESE et al. 2019) ont prouvé que si la distribution latente a des queues légères (par exemple gaussienne) et que le générateur G est lipschitzien continu, alors la distribution générée $G(Z)$ a nécessairement des queues légères. Spécifiquement, si Z est sous-gaussien, alors $G(Z)$ est sous-gaussien pour tout G lipschitzien. Puisque les GAN modernes imposent des contraintes de Lipschitz pour la stabilité de l'entraînement (normalisation spectrale, pénalités de gradient), ils ne *peuvent prouvablement pas* générer des sorties à queue lourde à partir de bruit gaussien. Ce n'est pas un échec de l'optimisation ou de l'architecture ; c'est une impossibilité fondamentale.

Modèles génératifs à queue lourde. Des travaux récents ont abordé cette limitation à travers plusieurs approches (Michaël Allouche, Emmanuel Gobet et al. 2024). Pour les GAN : les *méthodes de prétraitement* comme Quant-GAN (Wiese et al. 2019) et evtGAN (Youssef Boulaguiem et al. 2022) transforment les données pour supprimer la lourdeur des queues avant l'entraînement ; les méthodes à *latent à queue lourde* (Huster et al. 2021) remplacent le bruit gaussien par des distributions de Pareto ; les approches *paramétrées par EVT* comme EV-GAN (Michaël Allouche, Girard et al. 2022) conçoivent des générateurs qui encodent explicitement la structure des valeurs extrêmes. Pour les modèles de diffusion, les extensions à queue lourde remplacent le processus de bruit gaussien par des processus de Lévy α-stables (Yoon et al. 2023), permettant la génération directe d'échantillons à queue lourde. Cependant, toutes ces approches se concentrent sur le comportement *marginal* des queues. Aucune n'aborde la *structure de dépendance* dans les queues, caractérisée par la stdf.

Lacune 1. À notre connaissance, aucun modèle génératif existant ne fournit de garanties prouvables sur l'approximation de la dépendance de queue. Les méthodes existantes traitent le comportement marginal des queues mais ignorent la structure conjointe. Nous comblons cette lacune dans le Chapitre 1, en développant une architecture GAN qui approxime prouvablement toute stdf.

0.10.2 Génération de statistiques d'ordre

La formulation mathématique des statistiques d'ordre a été donnée dans la Section 0.8.2. Les représentations classiques (Sukhatme, Schucany) ont permis la simulation exacte depuis des décennies (H. A. David et H. N. Nagaraja 2003 ; Arnold et al. 2008). Nous passons maintenant en revue l'état de l'art.

Approches d'apprentissage automatique. La littérature en apprentissage automatique a abordé le tri du point de vue de l'apprentissage à classer (T.-Y. Liu 2009), où l'objectif est de prédire des ordres plutôt que de les générer. Des opérateurs de tri différentiables ont été développés pour permettre l'apprentissage par gradient : (Grover et al. 2019) ont proposé NeuralSort, une relaxation continue de l'opérateur argsort basée sur l'algorithme de Sinkhorn ; (Cuturi et al. 2019) ont développé le tri différentiable basé sur le transport optimal ; (Blondel et al. 2020) ont introduit le tri différentiable rapide avec une complexité $O(n \log n)$. Ces méthodes permettent le tri dans les pipelines de réseaux de neurones mais sont conçues pour les tâches de classement, pas pour l'appariement de distributions.

Modèles génératifs et ordonnancement. Les modèles génératifs standards (GAN, VAE, diffusion) produisent des échantillons non ordonnés. Appliqués à des données classées, ils font face à deux problèmes. Premièrement, les échantillons générés ne sont pas triés : un GAN entraîné sur des statistiques d'ordre produira des vecteurs dont les coordonnées ne sont pas monotonement ordonnées. Le tri a posteriori peut restaurer l'ordre

mais introduit un biais dans la distribution. Deuxièmement, le prétraitement standard détruit la structure d'ordonnancement : la normalisation par coordonnée vers $[0, 1]$, courante dans l'entraînement des GAN, normalise chaque coordonnée indépendamment selon son étendue marginale, détruisant l'ordonnancement relatif. Si la k-ième statistique d'ordre est normalisée par sa propre moyenne et variance, et la $(k+1)$-ième par des valeurs différentes, il n'y a aucune garantie que les valeurs normalisées restent triées.

Lacune 2. À notre connaissance, aucun modèle génératif existant ne produit des statistiques d'ordre avec à la fois les distributions marginales/conjointes correctes et des contraintes d'ordonnancement garanties. Les méthodes de tri différentiable existantes traitent la prédiction de classement, pas la modélisation générative. Nous comblons cette lacune dans le Chapitre 2, en développant une architecture qui exploite les représentations classiques pour garantir l'ordonnancement tout en appariant les distributions cibles.

0.10.3 Génération inclinée par récompense

Le problème d'alignement (Section 0.8.3) requiert d'échantillonner à partir de la distribution inclinée (12). Nous passons en revue les approches existantes.

Apprentissage par renforcement à partir de feedback humain. Le paradigme dominant pour l'alignement est l'apprentissage par renforcement à partir de feedback humain (RLHF) (Christiano et al. 2017 ; Bai et al. 2022 ; Ouyang et al. 2022). Un modèle de récompense est entraîné sur des données de préférence humaine (comparaisons par paires de sorties), puis le modèle génératif est affiné pour maximiser la récompense espérée en utilisant des méthodes de gradient de politique (PPO, REINFORCE). Le RLHF a été transformateur pour les grands modèles de langage, permettant ChatGPT, Claude et d'autres agents conversationnels. Cependant, le RLHF est notoirement instable : le piratage de récompense, l'effondrement de mode et l'oubli catastrophique sont des modes d'échec courants (Casper et al. 2023).

Ajustement fin des modèles de diffusion. Pour les modèles de diffusion, plusieurs approches d'ajustement fin ont émergé. DRaFT (Clark et al. 2024) rétropropage à travers toute la chaîne de débruitage pour maximiser la récompense, mais cela nécessite de stocker les activations à travers tous les pas de temps, avec une mémoire croissant linéairement avec la longueur de la chaîne. DPOK (Fan et al. 2023) combine les gradients de politique avec la régularisation KL. ReFL (Jing Xu et al. 2024) utilise l'apprentissage par feedback de récompense pour affiner les modèles de diffusion texte-vers-image. L'appariement adjoint (Domingo-Enrich, Drozdzal et al. 2025) fournit une approche théoriquement fondée basée sur les EDS adjointes, atteignant des résultats état de l'art mais nécessitant toujours les gradients de récompense.

Le goulot d'étranglement du gradient. Les méthodes d'ajustement fin basées sur le gradient (DRaFT, appariement adjoint) nécessitent de rétropropager à travers la fonction

de récompense, ce qui crée des limitations pratiques : stocker les activations pour les grands modèles de récompense mène à des coûts mémoire prohibitifs, et les récompenses non différentiables (feedback humain, évaluations discrètes, métriques basées sur simulateur) ne peuvent pas être traitées. Les approches basées sur l'apprentissage par renforcement (RLHF, DPOK) évitent cela en utilisant uniquement les valeurs de récompense, pas les gradients, mais introduisent d'autres défis : les estimateurs de gradient de politique ont une haute variance, et un réglage fin des hyperparamètres est nécessaire pour équilibrer la maximisation de récompense contre l'effondrement de mode. Aucune approche ne fournit de garanties distributionnelles sur le modèle affiné résultant.

Lacune 3. À notre connaissance, aucune méthode existante sans gradient n'affine les modèles de diffusion pour échantillonner à partir de distributions inclinées par récompense. Les méthodes existantes nécessitent soit les gradients de récompense, soit utilisent l'apprentissage par renforcement avec des estimateurs à haute variance. Nous comblons cette lacune dans le Chapitre 3, en développant une méthode basée sur l'identité de Fisher qui ne requiert que des évaluations directes de la fonction de récompense.

0.10.4 Objectifs de cette thèse

Sur la base des lacunes identifiées, nous formulons trois objectifs :

1. **Génération de valeurs extrêmes.** Développer un modèle génératif pour les extrêmes multivariés avec des garanties prouvables sur l'approximation de la stdf. La méthode devrait utiliser un bruit latent à queue lourde et fournir des bornes explicites sur l'erreur d'approximation en termes d'architecture de réseau.

2. **Génération de statistiques d'ordre.** Développer un modèle génératif pour les statistiques d'ordre qui garantit l'ordonnancement tout en appariant la distribution cible. La méthode devrait exploiter les représentations classiques et fournir des bornes d'approximation qui évoluent favorablement avec la dimension.

3. **Génération inclinée par récompense.** Développer une méthode sans gradient pour affiner les modèles de diffusion vers des distributions inclinées par récompense. La méthode ne devrait requérir que des évaluations directes de la fonction de récompense.

0.11 Contributions de cette thèse

Nous résumons maintenant les contributions principales, organisées selon les trois objectifs.

0.11.1 GAN à queue lourde pour la génération de valeurs extrêmes

Nous développons les GAN à queue lourde (HTGAN) pour générer des données avec une dépendance de queue correcte.

Contributions théoriques. Nous remplaçons le bruit latent gaussien par un bruit à queue lourde (Pareto ou Fréchet). Comme le générateur est lipschitzien, la propriété de queue lourde se propage à la sortie. Nous prouvons que la stdf de la distribution générée est toujours discrète, un maximum de fonctions linéaires avec un nombre fini d'atomes. Notre théorème d'approximation principal établit que toute stdf peut être approximée dans l'erreur L^∞ ε en utilisant une dimension latente $N = O(\varepsilon^{-(d-1)})$ et une seule couche cachée avec des activations ReLU. Cela semble être la première garantie d'approximation pour la dépendance de queue dans la littérature de modélisation générative.

Contributions méthodologiques. L'algorithme HTGAN procède en trois étapes : transformation marginale vers Fréchet unitaire, entraînement GAN avec bruit latent de Pareto, et transformation inverse. Cette séparation des marges et de la dépendance est standard en modélisation par copules.

Contributions expérimentales. Nous évaluons sur des données synthétiques de copule de Gumbel et des rendements du S&P 500, démontrant que HTGAN capture correctement la dépendance de queue tandis que les GAN standards échouent de plusieurs ordres de grandeur.

0.11.2 Statistiques d'ordre neurales génératives

Nous développons les statistiques d'ordre neurales génératives (GENOS) pour générer des données classées avec un ordonnancement garanti.

Contributions théoriques. Nous combinons les représentations statistiques classiques avec la théorie d'approximation des réseaux de neurones (Yarotsky 2017) pour dériver des bornes de complexité non asymptotiques. Spécifiquement, nous adaptons deux représentations classiques des statistiques d'ordre et analysons leur complexité d'approximation par réseaux de neurones.

La représentation de Schucany (Schucany 1972) construit les statistiques d'ordre récursivement :

$$U_{n-r:n} = U_{n-r+1:n} \cdot U_r^{1/(n-r)}, \quad r = 1, \dots, n-1. \tag{15}$$

Cela nécessite n fonctions de transition différentes. Nous prouvons que l'approximation neurale de ce schéma requiert un nombre de paramètres qui est polynomial de degré n en ε^{-1}.

La représentation de Sukhatme (Sukhatme 1937) exprime les statistiques d'ordre uniformes via des espacements exponentiels :

$$U_{k:n} = F\left(\sum_{j=1}^{k} \frac{E_j}{n-j+1}\right), \quad E_j \overset{\text{iid}}{\sim} \text{Exp}(1), \quad F(z) = 1 - e^{-z}. \tag{16}$$

La somme cumulative a une distribution exponentielle ; appliquer la fonction de répartition exponentielle F donne la k-ième statistique d'ordre uniforme $U_{k:n}$. Cela ne requiert

d'approximer que deux fonctions universelles (F et F^{-1}), indépendamment de n. Nous prouvons que l'approche basée sur Sukhatme atteint

$$\mathbb{E}|U_{k:n} - \hat{U}_{k:n}| \leqslant O(\varepsilon \log n) \quad \text{avec} \quad \mathcal{C}_{\text{Sukhatme}}(\varepsilon) = O(\varepsilon^{-2} \log(1/\varepsilon)) \tag{17}$$

paramètres. La complexité est quadratique en ε^{-1}, indépendante de n, contre polynomiale de degré n pour Schucany. Cela rend l'approche basée sur Sukhatme scalable à de grandes tailles d'échantillon.

Contributions méthodologiques. Nous introduisons la normalisation globale moyenne-échelle, qui préserve l'ordonnancement contrairement à la normalisation par coordonnée. Nous proposons également une pénalité de statistiques d'ordre (OSP) qui pénalise les violations de la contrainte d'ordonnancement :

$$\mathcal{L}_{\text{OSP}} = \mathbb{E}_Z \left[\frac{1}{n-1} \sum_{k=1}^{n-1} \left(\max\{G(Z)_k - G(Z)_{k+1}, 0\} \right)^2 \right].$$

Contributions expérimentales. Nous évaluons sur des statistiques d'ordre synthétiques avec dimension variable, comparant GAN, WGAN et WGAN avec OSP. L'ajout de la pénalité OSP améliore les métriques d'ordonnancement tout en maintenant des performances compétitives sur la distance de Wasserstein.

0.11.3 Ajustement fin sans gradient des modèles de diffusion

Nous développons des méthodes sans gradient pour affiner les modèles de diffusion afin d'échantillonner à partir de distributions inclinées par récompense.

Contributions théoriques. L'identité de Fisher exprime le score de la distribution inclinée comme une covariance :

$$\nabla_x \log p(x) = \nabla_x \log p_0(x) + \lambda \cdot \text{Cov}_{q_{0|x}} \left[\nabla_{x_0} \log q(x_0|x),\, r(x_0) \right]. \tag{18}$$

Cette covariance peut être estimée par Monte Carlo en utilisant uniquement des évaluations directes de r, sans rétropropagation à travers le modèle de récompense.

Contributions méthodologiques. Nous proposons un algorithme itératif qui décompose une grande inclinaison en N petites étapes, estimant l'incrément de score à chaque étape via l'identité de Fisher. L'algorithme s'applique même lorsque r est non différentiable ou une boîte noire.

Contributions expérimentales. Sur des mélanges de gaussiennes 2D avec vérité terrain connue, nous validons que la méthode récupère correctement le score incliné et que l'augmentation du nombre d'étapes N améliore la précision.

0.12 Organisation du manuscrit

Le Chapitre 1 développe la théorie et les algorithmes pour la modélisation générative à queue lourde. Nous présentons le contexte mathématique sur la théorie des valeurs extrêmes multivariées et la stdf, analysons pourquoi les applications lipschitziennes de bruit gaussien ne peuvent pas produire de queues lourdes, caractérisons la stdf discrète induite par les générateurs avec bruit de Pareto, prouvons notre théorème d'approximation principal et présentons des expériences sur des données synthétiques et financières.

Le Chapitre 2 développe les modèles génératifs pour les statistiques d'ordre. Nous présentons le contexte sur les statistiques d'ordre et les représentations de Sukhatme et Schucany, prouvons des bornes d'approximation pour les réseaux ReLU, introduisons la normalisation globale moyenne-échelle et l'objectif WGAN-OSP, et présentons des expériences sur des statistiques d'ordre synthétiques.

Le Chapitre 3 développe l'ajustement fin sans gradient pour les modèles de diffusion. Nous présentons le contexte sur l'inclinaison exponentielle et les méthodes existantes basées sur le gradient, dérivons l'estimateur de score basé sur l'identité de Fisher, analysons l'erreur d'approximation, présentons l'algorithme d'inclinaison itératif et validons sur des expériences contrôlées.

Cette thèse est partiellement basée sur des travaux collaboratifs : le Chapitre 1 sur les modèles génératifs à queue lourde, le Chapitre 2 sur l'approximation neurale des statistiques d'ordre, et le Chapitre 3 sur l'ajustement fin sans gradient des modèles de diffusion. Les lieux de publication et co-auteurs spécifiques sont indiqués au début de chaque chapitre.

Chapter 1

Heavy-Tailed GANs for Extreme Value Generation

1.1 Introduction

Statement of the problem. Examining extreme events is a critical concern across various fields such as economics, engineering, and life sciences, with wide-ranging applications like actuarial and financial risks (Asmussen and Albrecher 2010, Chapters X and XIII)-(P. Embrechts et al. 1997, Chapters 1 and 6)-(A. McNeil et al. 2015, Chapters 5 and 16, Part III), communication network reliability (P. Robert 2003), and aircraft safety (Prandini and Watkins 2005). Extreme events play a crucial role also in the context of climate change (J. Zscheischler et al. 2020), with the occurrence of more and more severe weather events, or in the context of cyber-security (Cremer et al. 2022) with the increasing number of cyber-attacks of private companies or public entities. In recent decades, the importance of extreme event analysis has surged in financial risk management, which calls for an ever-increasing number of types of risk (market, credit, operational, reputational, cyber, climate, and so on and so forth) to be encompassed in order to measure their impacts on banking activity and the stability of the financial system. In particular, regulators (European Banking Authority 2014) are increasingly imposing stress tests to test the resilience of the banking system against different risk categories. These stress tests involve numerical simulations of unfavorable yet plausible extreme scenarios, serving as a primary means to evaluate the potential impact on risks (see (ECB Banking Supervision 2023) for the 2023 exercises in Europe). Often, extreme events are described as tail events of multivariate relevant random variables.

The challenge arises in efficiently obtaining relevant samples for assessing these tail risks. Informally, it writes as finding a generator G and a latent distribution μ_Z which one can easily sample from, such that:

$$Z \sim \mu_Z, \qquad G(Z) \overset{d}{=} \text{target distribution.} \tag{1.1}$$

A large family of numerical approaches relies on physics-based methods, and aims to efficiently sample the distribution tail with minimal computational effort and to avoid the inefficiency of a basic accept-rejection algorithm. These methods are specific to the model at hand: among these methods, we mention importance sampling (Bucklew 2004, Chapter 4), MCMC with splitting – (E. Gobet and G. Liu 2015), or interacting particles system – (Del Moral and Garnier 2005). Another family of simulation schemes is data-driven approaches which do not require the knowledge of the (supposedly existing) physical model behind the sample generation. These data-driven schemes take as input a data set of observations of given size n_{data} and design a generative model able to re-generate samples that mimic the empirical distribution of the observed data. This concept aligns with recent paradigms in Artificial Intelligence such as Generative Adversarial Networks (GANs) initiated by (I. Goodfellow et al. 2014) and Variational Autoencoders (VAEs) by (D. P. Kingma and M. Welling 2014), score-based diffusion models (J. Sohl-Dickstein et al. 2015), normalising flows (Papamakarios et al. 2021), etc.

Our work is in this vein for generating extreme samples. Unlike the classical uses of Deep Generative Models as mentioned above, where the number n_{data} of training data is generally colossal (millions), in the context of extreme events, n_{data} is small by nature (a few hundred at most). Standard training procedures are thus bound to have poor performance all the more so as, by construction, their design makes them unable to reproduce heavy and dependent tails (M. Allouche et al. 2022). The aim of this work is to design new efficient procedures taking advantage of the probabilistic structure behind extreme values.

State of the art. Very recently, the machine learning community became aware of the difficulties of sampling extremes using Deep Generative Models, requiring to appropriately adapt known generative methods to the extreme setting. Most of the newly designed algorithms were based on GANs, see an overview in (Michaël Allouche, Emmanuel Gobet, et al. 2024). Three directions have been mainly investigated in the literature. First, certain works apply some preprocessing to the data to get rid of the tail heaviness, see Quant-GAN (Wiese et al. 2020) and evtGAN (Y. Boulaguiem et al. 2022): essentially, it consists in suitably transforming each data coordinate in order to retrieve either Gaussian or uniform marginal distributions, then performing a usual GAN method and reverting to the original space by inverse transform. A second direction is to consider new latent variables Z with a heavy-tailed distribution. In (Feder et al. 2020; Huster et al. 2021), a Generalized Pareto Distribution is adopted for the latent distribution μ_Z, in contrast to original GANs which use light-tailed distributions such as uniform or Gaussian. This idea finds its origins in a crucial observation: neural networks are Lipschitz functions (Wiese et al. 2020). As such, they cannot transform a random variable with light-tailed to a distribution with heavier tail, this is the main result of (Liang et al. 2022). Using heavy-tailed noise however raises an issue with the training procedure: if the underlying distribution is too heavy-tailed, the gradient of the loss function may not exist. To alleviate this, in (Huster et al. 2021), the authors introduce a loss function designed to produce stable training. To be effective, these methods require to estimate accurately the tail-index parameters for each marginal

distribution, which is a difficult task because the number n_{data} of training data is usually small. In (Huster et al. 2021), the authors provide empirical evidence that the Pareto-GAN method is able to match heavy-tailed distributions embedded in lower dimensional manifolds in a high dimensional setup. However, they do not provide a quantification of the power of Pareto-GAN to reproduce asymptotic dependence in the data, which is our main scope of interest in this paper.

A third approach consists in suitably parameterize the generator to account for extreme-value theory. This approach has been developed in (M. Allouche et al. 2022) (EV-GAN) where the renormalized log-quantile (called tail index function by the authors) is learnt using ReLU neural networks. The authors provide error bounds in uniform norm according to the complexity of neural networks and the second-order conditions in extreme-value theory. The latter reference shows that EV-GAN largely outperforms usual GANs, and is able to accurately sample X in dimension up to 50. For higher dimension, the marginals of X are still accurately reproduced but the tail dependence structure is under-estimated. Our work tackles the problem of better learning the dependence between margins with GANs.

The closest work to ours is presumably (Hasan et al. 2022). The authors endeavour to approximate the stable tail dependence function (stdf) of the vector X of interest, see Definition 1.1 later, which characterizes the dependence in the extremes, once the marginals have been standardized to a unit Fréchet scale. Recall that, once the stdf is known (even approximately), the vector X can be sampled using the spectral representation of a max-stable process (Hofert et al. 2018, Theorem 2.1). In (Hasan et al. 2022) two learning techniques of the stdf are introduced. On the one hand, a deterministic approach is proposed using a dedicated architecture of neural networks called "d-Max NN". The Fréchet marginalised data are projected onto directions in the simplex leading to exponentially distributed data. An algorithm is then introduced to maximise the log-likelihood using randomised directions on the neural network parameters. On the other hand, a stochastic approach is designed, based on the stochastic representation of the stdf via the spectral measure (see Proposition 1.1). Note that this approach is calibrated using the previous deterministic method. At first glance, our approach may look similar to (Hasan et al. 2022) but it is actually quite different, even though we also work on unitary Fréchet marginals. Here, we do rely on the spectral representation of max-stable processes, but we rather directly design a generative model of the vector X using an unitary Fréchet latent noise. In our opinion, this is much simpler since our approach avoids truncating (at a poorly controlled rank) the max-stable representation. Then, in contrast to Pareto-GAN, we show a convergence result of the approximated stdf to the true one when the dimension of the latent noise increases.

Last, we shall mention the Tail-GAN of (R. Cont et al. 2022) in a financial setting, where a usual GAN is performed but the loss function for learning depends on the risk-metrics of the scalar quantity of interest (e.g. Value-at-Risk and Expected Shorfall). It is a way to enforce the generative model to reproduce some specific metrics: note that this is different to sample a multidimensional extreme distribution.

On the VAE side, it is shown in (Lafon et al. 2023, Corollary 7) that a VAE built with piecewise linear activation functions and Gaussian distributions cannot reproduce heavy-tailed margins. It is moreover established that the spectral measure (see Section 1.2.2) associated with a classical VAE output is necessarily discrete (Lafon et al. 2023, Proposition 8). To overcome these problems, the authors consider a univariate heavy-tailed distribution to sample the radius, and, conditionally on the latter, an angle is sampled from a multivariate normal distribution. The product of the two yields the desired multivariate regularly-varying vector. The appropriate KL-divergences are derived leading to two objective functions: one for the radius VAE and one for the angular VAE. In the recent work (L. Zhang et al. 2024), the authors develop a VAE incorporating the max-id spatial model in order to sample spatial extremes with flexible and non-stationary dependence structures.

Our contributions. Focusing on GANs, we investigate the representation power of neural network based generative models with heavy-tailed input noise. In particular, we study the ability of such models to reproduce multivariate asymptotic dependence, which is quantified by the stdf. To this end, we first provide a modification to the original GAN training to accommodate the specificity of heavy-tailed input noise. Then, the quality of approximation of the stdf by the modified GAN is assessed thanks to the stdf representation in terms of D-norms (see Section 1.2.2). We show that the approximation error is decreasing when the dimension of the latent noise increases (see Theorem 1.3). Finally, we provide a set of experiments to illustrate the behaviour of the algorithm and perform a comparison with a benchmark method on both synthetic and real datasets.

Organization of the paper. Section 1.2 is devoted to the theory on the representation capability of the stdf with GANs based on heavy-tailed input noise. The main theoretical tools are introduced before the statement of our approximation results. Experiments on synthetic and real datasets are summarized in Section 1.3 to illustrate the generative power of heavy-tailed GANs. Some background results and the proofs of main results are postponed to the Supplementary (Appendix 3.A and Appendix 3.F respectively). Extra experiments are available in Appendix 3.I.

Notations.

- $[n] = \{i \in \mathbb{N} : \ 1 \leqslant i \leqslant n\}$ denotes the set of natural numbers from 1 to n. $\lceil \cdot \rceil$ denotes the ceil function.
- The vectors of the canonical basis of $\mathbb{R}^d$ are $\mathbf{e}_i = (0, \ldots, 0, 1, 0, \ldots, 0), \ i \in [d]$, with a 1 on the i^{th} coordinate.
- Vectors or matrices are denoted with bold symbols and scalar with roman ones. The i^{th} coordinate of a vector $\boldsymbol{x}$ is referred to as x_i. Random variables are denoted with capital letters and constant (deterministic) values with small letters. To fix ideas, x is a constant scalar, X is an $\mathbb{R}$-valued random variable, $\boldsymbol{x}$ is a constant vector and $\boldsymbol{X}$

is a random vector. Vectors with equal scalar coordinates are denoted in bold, such as $\mathbf{0} = (0, \dots, 0)$ and $\mathbf{1} = (1, \dots, 1)$. The dimension will be clear from the context.

- Without further specifications, operations such as multiplication, division, max, exponentiation and boolean operations on vectors are meant componentwise. For instance, for two vectors $\boldsymbol{x}, \mathbf{y} \in \mathbb{R}^d$, $\boldsymbol{x}\,\mathbf{y} := (x_1 y_1, \cdots, x_d y_d)$. Boolean operations are meant componentwise: $\boldsymbol{x} \leqslant \mathbf{y} \Leftrightarrow \forall i \in [d],\ x_i \leqslant y_i$. The scalar product is denoted with a dot: $\boldsymbol{x} \cdot \mathbf{y} = \sum_{i=1}^{d} x_i y_i$.
- The cumulative distribution function (cdf) of a random variable $\boldsymbol{X}$ is denoted by $F_{\boldsymbol{X}}$ or F according to the context. The cdf of the extreme-value distribution is denoted by G and the associated domain of attraction is denoted by $\mathrm{Dom}\,(G)$.
- $\mathrm{supp}(P)$ denotes the support of P.
- The Pareto distribution is defined by its cdf $1_{x \geqslant 1}\left(1 - x^{-1/\gamma}\right)$ with $\gamma > 0$. When introducing a Pareto distribution, we might also parametrize it with $\alpha = 1/\gamma$.
- $\#A$ denotes the cardinal of a set A.

1.2 Reproducing dependence in extreme regions with Generative Adversarial Networks: Theory

Some background on extreme-value theory is briefly given in Section 1.2.1. Section 1.2.2 provides the tools necessary to study dependence in extreme regions of a distribution. Finally, some key elements on Generative Adversarial Networks (GANs) are recalled in Section 1.2.3. Additional material on each topic is also given in the Supplementary (Appendix 3.A). Finally, our theoretical results are presented in Section 1.2.4: we provide a quantization bound on how well GANs with heavy-tailed noise (HTGANs) approximate the stdf of a target distribution.

1.2.1 Extreme-value theory

Extreme-value theory is a branch of statistics concerned with studying the upper tails of probability distributions. We refer to Appendix 3.B in the Supplementary for a brief introduction to this theory. Given a $\mathbb{R}^d$ valued random vector $\boldsymbol{X}$ with associated cdf $\boldsymbol{x} \in \mathbb{R}^d \mapsto F(\boldsymbol{x}) = \mathbb{P}\left(\boldsymbol{X} \leqslant \boldsymbol{x}\right)$, extreme-value theory establishes the asymptotic distribution of normalized maxima of realizations of $\boldsymbol{X}$. Whence normalized to unit Fréchet margins (1.28), the limiting distribution function $G_\star$ is a simple max-stable distribution, see Definition 1.6 and Equation (1.27). As illustrated in the next paragraph, $G_\star$ encodes the dependence structure in the tails of F.

1.2.2 Measuring dependence in extremes

We first recall the definition of the stable tail dependence function (stdf) associated with F.

Definition 1.1. (L. d. Haan and Ferreira 2006, Corollary 6.1.4) Let F be a continuous cdf on $\mathbb{R}^d$ belonging to the max-domain of attraction of some distribution. Then, one can define

$$\forall \boldsymbol{x} \in (0,\infty)^d : \ \ell_F(\boldsymbol{x}) = \lim_{t\to+\infty} t\,\mathbb{P}\left(1 - F_1(X_1) \leqslant \frac{x_1}{t} \ \text{ or } \ \ldots \ \text{ or } \ 1 - F_d(X_d) \leqslant \frac{x_d}{t}\right), \tag{1.2}$$

where F_j's are the marginal cdf of $\boldsymbol{X}$. The limit ℓ_F is the so-called *stable tail dependence function* (stdf) of F.

The stdf provides a "normalized" representation of the dependence structure of the distribution in the tails (as opposed to the copula function which globally models the dependence), in the sense that its marginals are normalized: the stdf does not carry any information on the marginal distributions of the base distribution of interest.

The following proposition gives the classical parametrization of the dependence structure in terms of spectral measure. A more recent and less known parametrization using D-norms is provided in Theorem 1.1 below.

Proposition 1.1 (Spectral representation of the stdf, (L. d. Haan and Ferreira 2006, Remark 6.1.16.))**.** *Consider a cdf F (as in Definition 1.1) and its stdf ℓ_F. Then, there exists a probability measure Λ on the simplex $\Delta_{d-1} := \left\{\boldsymbol{\omega} \geqslant \mathbf{0} : \ \sum_{i\in[d]} \omega_i = 1\right\}$ such that:*

$$\forall \boldsymbol{x} \in (0,\infty)^d : \ \ell_F(\boldsymbol{x}) = d \int_{\Delta_{d-1}} \max_{i\in[d]}(\omega_i x_i)\Lambda(\mathrm{d}\boldsymbol{\omega}), \tag{1.3}$$

with constraints: $\forall i \in [d] : \ \int_{\Delta_{d-1}} \omega_i \ \Lambda(\mathrm{d}\boldsymbol{\omega}) = 1/d$.

Quantifying dependence in extremes with D-norms. D-norms (Falk 2019) are a class of norms in vector space, very convenient for studying dependence in extremes.

Definition 1.2 (D-norms, (Falk 2019, Lemma 1.1.3))**.** Let $\boldsymbol{\Gamma} = (\boldsymbol{\Gamma}_1, \ldots, \boldsymbol{\Gamma}_d) \in \mathbb{R}^d$ be a random vector, whose components satisfy for all $i \in [d]$, $\boldsymbol{\Gamma}_i \geqslant 0$ a.s.and $\mathbb{E}[\boldsymbol{\Gamma}_i] = 1$. Then,

$$\|\cdot\|_{\boldsymbol{\Gamma}} : \ \boldsymbol{x} \mapsto \|\boldsymbol{x}\|_{\boldsymbol{\Gamma}} = \mathbb{E}\left[\max_{i\in[d]}(|x_i|\boldsymbol{\Gamma}_i)\right] \tag{1.4}$$

defines a norm on $\mathbb{R}^d$, called a D-norm, and $\boldsymbol{\Gamma}$ is called a generator of the D-norm.

Main properties of the D-norm are recalled in Appendix 3.C. To alleviate the role of the generator, we may simply denote the above norm by $\|\cdot\|_D$. The key role of D-norms is highlighted in the next result.

Theorem 1.1. *Let F be a continuous cdf on $\mathbb{R}^d$ belonging to the max-domain of some distribution. Then, there exists a D-norm that exactly represents the stdf of F:*

$$\forall \boldsymbol{x} \in (0,\infty)^d, \qquad \ell_F(\boldsymbol{x}) = \|\boldsymbol{x}\|_D \,. \tag{1.5}$$

This is a consequence of Theorem 1.6 in the Supplementary. Note that the two representations in (1.3) and in (1.4)–(1.5) may look similar at first sight, but actually they are substantially different. While both representations involve a probability measure (the spectral measure Λ on the one hand, and that of the generator $\boldsymbol{\Gamma}$ associated with the D-norm on the other hand), the second representation is more tractable for approximation purposes: Instead of designing a non-negative measure on the simplex Δ_{d-1}, one *simply* has to define a random vector $\boldsymbol{\Gamma} \in [0,+\infty)^d$ with unit expectation.

Definition 1.3. Let F be a cdf satisfying the assumptions of Theorem 1.1. The stdf ℓ_F is said to be *discrete*, with N atoms, when its D-norm is related to a generator $\boldsymbol{\Gamma}$ whose distribution is *discrete* and made of N atoms.

Example 1.1. Let us consider the special case where $d = N$. Note that it can be typically the case of the input of a generative model with a latent distribution of dimension N. Let F be a cdf in $\mathbb{R}^N$ with independent margins. Then, let $\ell_F(\boldsymbol{x}) = \sum_{i\in[N]} x_i, \quad \forall \boldsymbol{x} \in (0,\infty)^N$. This stdf is discrete with N atoms: indeed, one can easily check that, if $\boldsymbol{\Gamma}$ has the discrete uniform distribution on $\{N\,\mathbf{e}_1, \ldots, N\,\mathbf{e}_N\}$, then $\mathbb{E}\,[\boldsymbol{\Gamma}_i] = 1$ for any $i \in [N]$, and its D-norm is given by

$$\|\boldsymbol{x}\|_D = \mathbb{E}\left[\max_{i\in[N]}(x_i \boldsymbol{\Gamma}_i)\right] = \frac{1}{N}\sum_{i\in[N]} x_i\, N = \ell_F(\boldsymbol{x}), \qquad \forall \boldsymbol{x} \in (0,\infty)^N. \tag{1.6}$$

With similar arguments, it is easily checked that the spectral measure is the discrete uniform distribution on $\{\mathbf{e}_1, \ldots, \mathbf{e}_N\}$.

The concept of Definition 1.3 is at the core of our numerical scheme, since we will consider approximations based on discrete distributions for the generator $\boldsymbol{\Gamma}$ in Section 1.2.4.

Characterization of simple max-stable distribution using D-norms and homogeneous function. We aim at giving another criterion to establish the max-domain of attraction of a simple max-stable distribution. It will be crucially used in the proof of our main result (Theorem 1.3).

Definition 1.4 (1-homogeneous function)**.** A function $h : [0,\infty)^d \to [0,\infty)^{d'}$ is 1-homogeneous if it satisfies: $\forall \boldsymbol{x} \in [0,\infty)^d,\ \forall \lambda \geqslant 0 :\ h(\lambda \boldsymbol{x}) = \lambda h(\boldsymbol{x})$. The set of 1-homogeneous continuous functions from $\mathcal{X}$ to $\mathcal{Y}$ is denoted by $\mathbb{H}_1(\mathcal{X},\mathcal{Y})$.

The next result claims that identifying the stdf (or equivalently the D-norm) of a cdf F (with asymptotically simple max-stable distribution) is equivalent to analyze tails of 1-homogeneous transforms of the associated random vector.

Theorem 1.2 ((Falk and Fuller 2021, Theorem 4.4)). *Let $\boldsymbol{X}$ be a random vector in $[0,+\infty)^d$ with continuous cdf F and let $\|.\|_D$ be a D-norm in $\mathbb{R}^d$ generated by $\boldsymbol{\Gamma}$. The following equivalence holds:*

$$\forall \boldsymbol{x} \in (0,\infty)^d : \quad F^t(t\boldsymbol{x}) \underset{t\to\infty}{\longrightarrow} \exp - \left\| \frac{1}{\boldsymbol{x}} \right\|_D \tag{1.7}$$

$$\iff \qquad \forall h \in \mathbb{H}_1([0,\infty)^d, [0,\infty)) : \quad t\mathbb{P}\left(h(\boldsymbol{X}) > t\right) \underset{t\to\infty}{\longrightarrow} \mathbb{E}\left[h(\boldsymbol{\Gamma})\right]. \tag{1.8}$$

An indirect consequence of the above theorem is that the limiting value $\mathbb{E}\left[h(\boldsymbol{\Gamma})\right]$ does not depend on the chosen generator of the D-norm, when h is a continuous 1-homogeneous function, see (Falk and Fuller 2021, Theorem 3.5) for a direct proof.

1.2.3 Generative Adversarial Networks

Neural networks. We refer to (Murphy 2022, Chapter 13) for a thorough introduction on neural networks. Let us consider fully connected neural networks, also called Multi Layer Perceptron (MLP). These neural networks consist of a chain of L operations of the form:

$$\mathbf{z}_l = f_l\left(\mathbf{z}_{l-1}\right) = \varphi_l\left(\boldsymbol{b}_l + \mathbf{W}_l \mathbf{z}_{l-1}\right) \tag{1.9}$$

for $l \in [L]$, where $\mathbf{z}_l \in \mathbb{R}^{q_l}$, $\varphi_l : \mathbb{R}^{q_l} \to (\mathbb{R}^+)^{q_l}$ is an activation function, q_l is the number of neurons on the layer l, $\mathbf{W}_l$ is a matrix and $\boldsymbol{b}_l$ is a bias vector. Note that $\mathbf{z}_0$ corresponds to the q_0-dimensional input of the network, while $\mathbf{z}_L$ is the q_L-dimensional output. In this work, we focus on neural networks with ReLU activation functions: $\varphi_l(\mathbf{z}) = \max\{\mathbf{z}, \mathbf{0}\}$. The parameters of the network are:

$$\theta = \{\boldsymbol{b}_1, \ldots, \boldsymbol{b}_L, \mathbf{W}_1, \ldots, \mathbf{W}_L\} \in \Theta \tag{1.10}$$

(activation functions are fixed), where Θ is the space of parameters, the full Euclidean space without constraints.

Proposition 1.2. *A neural network parameterized by (1.9) with ReLU activation functions is a piecewise affine function. It is a continuous 1-homogeneous function when the bias vectors are zero.*

The proof is easy and left to the reader. Combining Proposition 1.2 with Theorem 1.2 shows that a distribution (with simple max-stable distribution) transformed by a ReLU neural network remains a heavy-tailed distribution with the same tail parameter γ.

Generative Adversarial Networks. Generative modeling is the task of approximating at best a distribution of interest, here written $\boldsymbol{X} \sim p_{\text{data}} \in \mathbb{R}^d$, and being able to draw samples from it. One way to do is to find a map $\mathcal{G}$ which takes as input a random variable $\boldsymbol{Z} \sim p_{\boldsymbol{Z}} \in \mathbb{R}^N$ which is easy to sample, such that:

$$\mathcal{G}(\boldsymbol{Z}) \overset{d}{\approx} \boldsymbol{X}. \tag{1.11}$$

The random vector $\boldsymbol{Z}$ is commonly referred as the latent noise and $p_{\boldsymbol{Z}}$ the latent distribution. Generative Adversarial Networks (I. Goodfellow et al. 2014) are widespread generative models. GANs consist of a generator $\mathcal{G}$ and a discriminator $\mathcal{D}$. GANs play an adversial game where the generator tries to mimic the true distribution generating the data whereas the discriminator tries to uncover fake data points simulated by the generator. Namely, GANs optimize for:

$$\min_{\mathcal{G}} \max_{\mathcal{D}} V(\mathcal{D}, \mathcal{G}) \quad \text{where} \quad V(\mathcal{D}, \mathcal{G}) = \mathbb{E}_{\boldsymbol{X} \sim p_{\text{data}}} \left[\log \mathcal{D}(\boldsymbol{X})\right] + \mathbb{E}_{\boldsymbol{Z} \sim p_{\boldsymbol{Z}}} \left[\log(1 - \mathcal{D}(\mathcal{G}(\boldsymbol{Z})))\right], \tag{1.12}$$

where p_{data} is the distribution of the data and $p_{\boldsymbol{Z}}$ is the latent distribution. Usually, p_{data} is chosen among the uniform or the Gaussian distribution. Both $\mathcal{G}$ and $\mathcal{D}$ are neural networks. In this work, we consider neural networks parameterized with MLPs (1.9), and, with the notations of (1.9), note that $q_0 = N$ and that $q_L = d$.

1.2.4 New approximation results for the stdf

Our goal is to assess how well the dependence in the extremes of a target distribution can be by approximated with a GAN based on heavy-tailed independent input noise. To this end, our first result establishes the nature of the stdf of a neural network output, when used with heavy-tailed input noise.

Proposition 1.3. *Consider a random vector $\boldsymbol{Z}$ with a discrete stdf $\ell_{\boldsymbol{Z}}$ with N atoms. Consider a neural network $G_{\theta,N}$, with ReLU activation functions and $L-1$ hidden layers, where θ is the parametrization of the network (1.10) and N is the dimension of the input noise ($q_0 = N$). Assume that all the matrix weights $(\mathbf{W}_1, \cdots, \mathbf{W}_L)$ are non-negative. Then, the stdf of the output $G_{\theta,N}(\boldsymbol{Z})$ is discrete with at most N atoms.*

See Section 3.F for the proof. It appears that the output of a neural network with heavy-tailed input noise has a discrete stdf. The next result states that it is possible to attain perfect reproduction of any discrete stdf with MLP neural networks.

Proposition 1.4. *Consider a heavy-tailed random vector $\boldsymbol{X}$ with a discrete stdf ℓ with N atoms. There exists a neural network $\mathcal{G}_{\theta^*,N}$ with parametrization $\theta^* \in \Theta$ and input noise $\boldsymbol{Z} \in \mathbb{R}^N$ with i.i.d.unit Fréchet margins such that:*

$$\ell = \ell_{\mathcal{G}_{\theta^*,N}(\boldsymbol{Z})} \tag{1.13}$$

where $\ell_{\mathcal{G}_{\theta^*,N}(\boldsymbol{Z})}$ is the stdf of $\mathcal{G}_{\theta^*,N}(\boldsymbol{Z})$.

In words, any discrete stdf can be perfectly approximated by a neural network with i.i.d.unit Fréchet margins as input noise with sufficiently large dimension.

When the stdf of interest is not discrete, our final result proves that it can be nevertheless approximated using a MLP neural network with an arbitrary precision.

Theorem 1.3. *Let $\boldsymbol{X}$ be a random vector in $\mathbb{R}^d$, with continuous cdf F, belonging to the max-domain of some distribution. Denote by ℓ_F its stdf that is related to a D-norm*

(Theorem 1.1). Consider using a MLP neural network $\mathcal{G}_{\theta,N}$ with a single layer (i.e.$L = 1$) and with as latent noise a random vector of N i.i.d.unit Fréchet margins, and denote by $\ell_{\mathcal{G}_{\theta,N}}$ its stdf. Then,

$$\inf_{\theta\in\Theta} \sup_{\boldsymbol{x}\in(0,\infty)^d} \frac{\left|\ell_F(\boldsymbol{x}) - \ell_{\mathcal{G}_{\theta,N}}(\boldsymbol{x})\right|}{\|\boldsymbol{x}\|_\infty} \leqslant \epsilon(N), \tag{1.14}$$

$$\epsilon(N) \sim C(d)N^{-1/(d-1)} \text{ as } N \to \infty, \tag{1.15}$$

where $C(d)$ is a constant depending on d and $\ell_F(\mathbf{1})$. The bound (1.14) is achieved with one layer neural networks.

The intuition behind the bounds (1.14)-(1.15) can be developed as follows. Approximating the stdf amounts to approximating a probability measure in dimension d using N atoms; it is thus natural to expect an error bound of order $N^{-1/d}$. However, the random variable having this probability measure is subject to a realization or moment constraint (see the proof for details), which reduces the degrees of freedom by one, leading to an approximation error of at most order $N^{-1/(d-1)}$.

In this previous theorem, note that the error bound is a worst case scenario. For example, when the stdf is discrete, the error is zero for N sufficiently large in virtue of Proposition 1.4. Let us highlight that, in view of the upper bound (1.15), the attainable precision of the neural network approximation increases with the dimension of the latent noise. However, the higher the dimension d, the slower is the convergence. One can then expect that accurate approximations of the dependence in high dimension would require high dimensional latent noise, which is consistent with the intuition. This will be illustrated in the Numerical Experiments section.

In other words, the above result states a denseness property. We would like to thank one of the referees to spot the analogy with the results from (Fougères and Mercadier 2013, Lemma 9, Theorem 3). As a difference, we use other arguments (see the proof based on quantization tools) and, consequently, we are able to quantify the effect of both latent and ambient dimensions N and d. Let us finally note that this denseness property is used in (JanSSen and Wan 2020) to interpret the proposed k-means procedure as an inference method for max-linear models.

1.3 Numerical experiments

First, we describe in Section 1.3.1 the proposed algorithms for training and sampling from a dataset presenting heavy-tailed characteristics. Second, experiments performed on synthetic datasets generated using the Gumbel copula are detailed in Section 1.3.2 to compare the proposed method with the standard GAN approach. Finally, experiments on the stock market index S&P500 are presented in Section 1.3.3.

1.3.1 Algorithms and implementation

Algorithms

We accommodate the original GAN algorithm (I. Goodfellow et al. 2014) to have a heavy-tailed latent noise. This idea originates from (Huster et al. 2021). Algorithm 1 describes the learning phase while Algorithm 2 summarizes the generation phase.

Algorithm 1: Generative modelling: learning algorithm

input :

- Dataset $\left\{\boldsymbol{X}^{(1)}, \cdots, \boldsymbol{X}^{(n_{\text{data}})}\right\}$ of n_{data} i.i.d. sample points in $\mathbb{R}^d$.
- Fully connected initialized generator $\mathcal{G}$ and discriminator $\mathcal{D}$.
- Estimators $\hat{F}_1, \ldots, \hat{F}_d$ of marginal cdf.
- Tail parameter $\gamma > 0$ for the renormalization step.
- GAN hyperparameters: Network architecture, optimization parameters and callbacks (early stopping, maximum number of iterations ...). The parameters are dependent on the chosen implementation. See Section 3 for more details.

begin

1. For any $j \in [d]$, estimate the marginal cdf of X_j on the training set $\left\{X_j^{(1)}, \ldots, X_j^{(n_{\text{data}})}\right\}$ by
$$x \in \mathbb{R} \mapsto \hat{F}_j(x) \in [0,1]. \tag{1.16}$$
2. Transform the marginals to Fréchet distributions with tail parameter γ:
$$\forall i \in [n_{\text{data}}], j \in [d] : \tilde{X}_j^{(i)} = \left(\frac{1}{1 - \hat{F}_j\left(X_j^{(i)}\right)}\right)^{\gamma}. \tag{1.17}$$
3. Train GAN over transformed data $\left\{\tilde{\boldsymbol{X}}^{(1)}, \ldots, \tilde{\boldsymbol{X}}^{(n_{\text{data}})}\right\}$: optimize (1.12) where p_{data} is a componentwise Pareto distribution with tail parameter γ and independent margins.

return *the optimal generator* $\mathcal{G}^\star$*, and the estimators* $\hat{F}_1, \ldots, \hat{F}_d$ *of the margins.*

end

Remarks about the marginals. In Subsection 1.3.3, which focuses on experiments with real data, the estimator $\hat{F}$ is taken to be the empirical cdf. While other choices are possible – since estimating a one-dimensional cdf is a classical problem in statistics – this simple choice suffices to run Algorithm 1 and obtain an approximation of the generator $\mathcal{G}^\star$. In doing so, we address the primary objective: reproducing the dependence structure in the extremes. It is important to note that Algorithm 2 requires access to the inverse cdf

Algorithm 2: Generative modelling: sampling algorithm

input :

- Number of data points to sample n_{sample}.
- Tail parameter $\gamma > 0$.
- Estimators $\hat{F}_1, \ldots, \hat{F}_d$ of the margins.
- Generator $\mathcal{G}^\star$.

begin

1. Sample n_{sample} points $\left(\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(\text{nsample})}\right)$ from p_{data}, a componentwise Pareto distribution with tail parameter γ and independent margins.
2. Transform the noise through the generator:

$$\forall i \in [n_{\text{sample}}] : \ \tilde{\boldsymbol{X}}^{(i)}_{\text{sample}} = \mathcal{G}^\star(\mathbf{z}^{(i)}). \tag{1.18}$$

3. Transform the sample points $\tilde{\boldsymbol{X}}_{\text{sample}}$ to the original scale with the inverse of the estimated margins:

$$\forall i \in [n_{\text{sample}}], j \in [d] : \ \hat{X}^{(i)}_j = \hat{F}_j^{-1}(\tilde{X}^{(i)}_j). \tag{1.19}$$

end

return *Sampled points* $\left\{\hat{\boldsymbol{X}}^{(1)}, \ldots, \hat{\boldsymbol{X}}^{\left(n_{\text{sample}}\right)}\right\}$.

(*i.e.*the quantile function). To ensure an accurate fit of the marginals – while preserving the learned dependence – we recommend using a quantile estimator that performs well across the entire distribution, both in the bulk and in the tails. A natural candidate is the estimator based on eLU-activated neural networks proposed by (M. Allouche et al. 2024).

Implementation

Code. The implementation is based on a publicly available GitHub repository[1] using the machine learning library PyTorch[2]. We adapt the source code so that both the generator and discriminator are fully connected MLPs (1.9). For writing a specific architecture of a neural network, we use a list that details all hidden layers, that is layers q_l for $l \in [L-1]$ in (1.9). As an example, a network parametrized by $[100, 200]$ means that it has 2 hidden layers, $L = 3$, $q_1 = 100$ and $q_2 = 200$. For the activation functions, we use LeakyReLu (Maas et al. 2013): note that our theoretical result (Theorem 1.3) still holds since a LeakyReLu is a difference of two ReLUs. For gradient descent, we use the optimizer Adam (D. P. Kingma and Ba 2014) and we do not tune the base parameters. Early stopping is used for regularization.

[1] https://github.com/eriklindernoren/PyTorch-GAN?tab=readme-ov-file#gan
[2] https://pytorch.org/

In these experiments, we allow the networks to have several hidden layers. The proof of Proposition 1.4 only involves networks of depth one to handle the dependence in the extremes, but higher depths may be necessary to reproduce the dependence in the tails or in the bulk of the distribution. In (1.49), it is seen that a one layer transformation produces a heavy-tail term (left term) and a bounded term. Our aim is that on one side, we manage to catch the behaviour of the distribution tail with the left unbounded, 1-homogenous term and, on the other side, we hope to capture the behaviour of the bulk with the right bounded term.

Hardware. All experiments were performed on the cluster Cholesky[3] of Ecole Polytechnique. Code was run on Intel Xeon CPU Gold 6230 20 cores @ 2.1 Ghz with 192 GB of memory.

1.3.2 Experiments on simulated data

We first propose in Section 1.3.2 a visual illustration of the behaviour of heavy-tailed latent noise models on a two dimensional dataset. Second, we investigate in Section 1.3.2 the performance of Algorithm 1 on higher dimensional data and perform a comparison with the baseline GAN algorithm using Gaussian input noise. Specifically, we study the model's ability to reproduce data with both given dependence degree and tail heaviness. To this end, the target distribution is obtained by combining Gumbel, Gaussian and Hüsler-ReiSS copulas with Pareto margins. We refer to Section 3.E for basic material on copulas.

Two-dimensional data: an illustration

Let us first consider a toy dataset of size $n_{\text{data}} = 10,000$ generated by a Gumbel copula in dimension $d = 2$ with identically distributed Pareto margins, see the left panel of Figure 1.1 for an illustration. It appears that, in the tail, data are evenly distributed along the horizontal axis $\mathbf{e}_1$ and the vertical one $\mathbf{e}_2$. As a comparison, the right panel of Figure 1.1 displays a plot of the input noise with independent Pareto random margins. In contrast to the Gumbel training data, the independent Pareto noise whence conditioned to being large in norm, is concentrated in the two directions $\mathbf{e}_1$ and $\mathbf{e}_2$. This is due to the fact that the spectral measure associated with independent Pareto random variable is discrete, with weights on $\mathbf{e}_1$ and $\mathbf{e}_2$.

The left panels of Figure 1.2 show simulated data generated by a GAN trained on the previously described dataset, using $N \in \{2, 3, 5, 10, 50\}$ independent Pareto random variables as input noise. The right panels display the histogram of the angles $\arccos(\hat{X}_1/\|\hat{\boldsymbol{X}}\|)$ for the 10% largest Euclidean norms $\|\hat{\boldsymbol{X}}\|$. For small latent dimensions ($N \in \{2, 3\}$), spikes are clearly visible in the tail of the distribution: the generated datasets exhibit as many spikes as the dimension of the latent noise. This is expected, since the spectral measure of i.i.d. Pareto noise is discrete, and, in accordance with Proposition 1.3, transforming N-dimensional independent Pareto noise (with a discrete spectral measure of N atoms)

[3] https://docs.idcs.mesocentre.ip-paris.fr/cholesky/hardware_description/

through a 1-homogeneous mapping yields a distribution whose spectral measure has at most N atoms. For larger dimensions ($N \in \{5, 10\}$), spikes become less pronounced, though still discernible, and the dependence structure appears visually closer to the target. For $N = 50$, the spikes are almost indistinguishable, and the scatter plot of the generated data closely resembles that of the real data (Figure 1.2-f). To sum up, these tests show that the larger N is, the closer the distribution gets to the target distribution, which supports the idea of convergence as N tends to infinity, in line with our theoretical results.

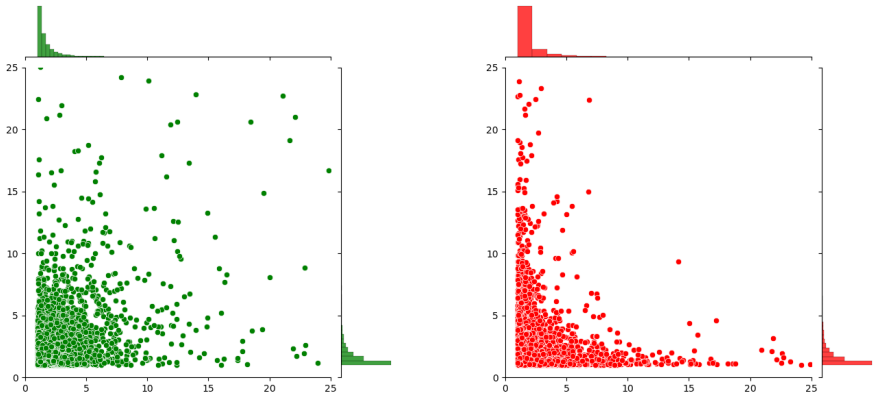


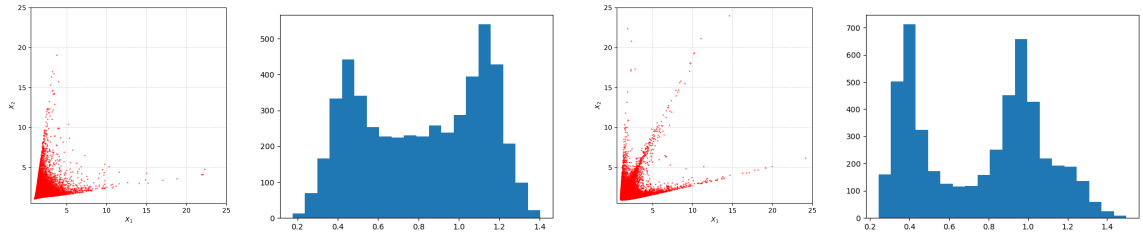
Figure 1.1: Illustration in a two-dimensional setting on simulated data of size $n_{\text{data}} = 10,000$ with Pareto margins and tail parameter $\alpha = 2.0$. Left panel: sample from a Gumbel copula with dependence parameter $\beta = 4/3$. Right panel: sample with two independent margins.

Higher dimensional experiments on Gumbel copula

After a thorough understanding of the behaviour of the model in a two dimensional setting, we explore its performance on higher dimensional datasets, still using the Gumbel copula, but with a variety of dependence and tail coefficients.

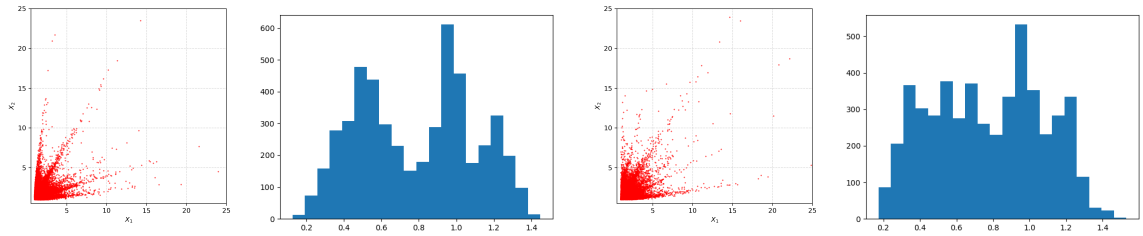
Experimental setup. The trained data are sampled from the d-dimensional Gumbel copula (see Section 3.E) with $d \in \{2, 5, 10, 20, 50\}$ and dependence parameter $\beta \in \{4/3, 2, 4\}$, leading to Kendall's $\tau \in \{1/4, 1/2, 3/4\}$. See (1.41) for a definition of Kendall's τ. The margins are chosen to be Pareto distributed with shape parameter $\alpha \in \{1.5, 2.0, 2.5\}$. In each of the $5 \times 3 \times 3 = 45$ considered situations, $n_{\text{data}} = 10,000$ data points are simulated for training and $20,000$ data points are considered for testing. Note that the testing set is larger than the training set in order to assess the ability of GANs to extrapolate to extreme regions unseen in training. In this synthetic case, we provide the tail-index $\gamma = 1/\alpha$ to Algorithm 1 and focus on the quality of reproduction of dependence in extreme regions.

Metrics. Two metrics are considered for assessing the performance of the GANS: the Absolute Kendall Error (AKE), see (M. Allouche et al. 2022, Section 3.2 and Appendix A)



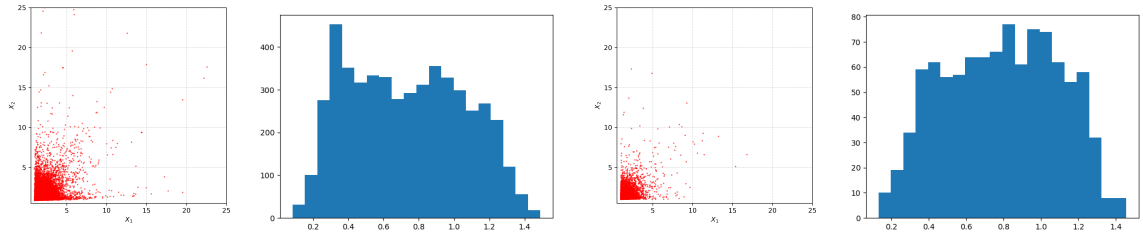
(a) $N = 2$

(b) $N = 3$



(c) $N = 5$

(d) $N = 10$



(e) $N = 50$

(f) Real data

Figure 1.2: Illustration of real and generated data in a two-dimensional setting for data from a Gumbel copula with $\beta = 4/3$ and Pareto margins ($\alpha = 2.0$). For each subfigure (a)-(f), on the left: scatter plot of data points; on the right: histogram of angles $\arccos(X_1/\sqrt{X_1^2 + X_2^2})$ for the 10% largest Euclidean norms. Subfigures (a)-(e): $n_{\text{gen}} = 50{,}000$ generated data after training with $n_{\text{data}} = 10{,}000$ points. Subfigure (f): real data.

for a definition and some background, and the Sliced Wasserstein Distance (SWD), see (1.42). Since the focus is on extreme regions, we compute the Euclidean norm of all data points and select data points whose norms are larger than the upper ξ-quantile, with $\xi \in \{90\%, 95\%, 99\%\}$. Both metrics are then computed (denoted by AKE_ξ and SWD_ξ) on the resulting points. This operation is performed both on the sample generated by GAN methods and on the dataset simulated from the true distribution. These new subsamples are then normalized on the Euclidean sphere, where the Sliced Wasserstein distance is estimated.

Preliminary hyperparameter search. We explore a number of possible parametrizations of GAN models within a selected range of hyperparameters. For both the generator and the discriminator, fully connected neural networks are used with the following 15 hidden dimensions: [50], [75], [100], [125], [200], [300], [400], [100, 100], [200, 200], [300, 300],

[400, 400], [100, 200, 100], [200, 400, 200], [100, 200, 200, 100] and [200, 400, 400, 200]. For the latent dimension, we explore values N ranging from 1 to 200. For the optimization parameter, values ranging (log-uniformly) from 10^{-6} to 10^{-1} are tested. We investigate the use of four batch sizes: $\{128, 256, 512, 1024\}$. A Bayesian search (see (J. Wu et al. 2019)) is implemented using weight and biases' sweep tool[4]. At first, some parametrizations are selected at random and, next, the probability for a configuration to be selected is updated with the use of a Bayesian rule. See the referred online documentation for more details on Bayesian hyperparemeter tuning. Considering the top performing models of this research, the following conclusions can be drawn on the choice of hyper parameters: Among tested models, best performing generators (80%+) have a light parametrization (fewer parameters) [100] or [200], while best performing discriminators (80%+) have a heavy parametrization (more parameters) [100, 200, 200, 100] or [200, 400, 400, 200]. Bigger batch sizes yield better performances. Good values for the learning rate include range $[5 \cdot 10^{-6}, 5 \cdot 10^{-4}]$.

Setting up models for comparison. After this hyperparameter exploration, two versions of GAN are compared: the base version of GAN with a Gaussian noise $\boldsymbol{Z} \sim \mathcal{N}(0, \boldsymbol{I}_{N\times N})$, referred to as LTGAN (LT for light-tailed), and our version, Algorithm 1, referred to as HTGAN (HT for heavy-tailed). Thanks to the previous insights on the preliminary hyperparameter search, we are able to choose a subset of evaluation hyperparameters to compare both models. Both models are run for each configuration. In order to have an exhaustive comparison, a grid search is performed on the following range of hyperparameters: $N \in \{2, 5, 10, 20, 50, 80\}$, hidden dimensions ranging in $\{[100, 200, 100], [200, 400, 200], [100, 200, 200, 100], [200, 400, 400, 200]\}$ for the discriminator, in $\{[100], [200]\}$ for the generator, $\{10^{-4}, 10^{-5}\}$ for the learning rate.

Results. It appears in Table 1.1 that HTGAN performs better than LTGAN with respect to all six metrics and all considered dependence coefficients $\beta \in \{4/3, 2, 4\}$ in the heavier tail setting $\alpha = 1.5$ (all percentages are above 50%). For $\alpha = 2.5$, it appears that HTGAN fails to outperform the baseline on the $\beta = 4$, $\alpha = 2.5$ setting. It suggests that in low tail-index settings, light-tailed outputs still manage to provide a good approximation of the target distribution, even thought the target is heavy-tailed. For almost all experiment settings, HTGAN is comparatively better than LTGAN for the highest tail coefficient. For the sliced Wasserstein distance, it appears that HTGAN gets comparatively better as both β and $\gamma = 1/\alpha$ increase. This result is coherent with our theoretical development: It is known that (piecewise) linear transformations of Gaussian variables cannot generate (asymptotic) dependence in the tails (Wiese et al. 2020), a case which is met when β gets larger and larger.

Figures 1.3 and 1.4 plot the performance of both methods for $\alpha = 1.5$ and $\alpha = 2.5$, with varying values of β. With $\alpha = 1.5$, the point cloud is well above the median and our

[4] https://docs.wandb.ai/guides/sweeps

method is better performing than the baseline. In the case $\alpha = 2.5$, the difference is not neat and no clear conclusion can be drawn.

Table 1.1: Results on simulated data from a Gumbel copula. Percentage (%) of parametrizations for which HTGAN is better than a LTGAN for the six considered metrics. Results are given with precision $\pm 0.1\%$. Values are averaged over data dimensions $d \in \{2, 5, 10, 20, 50\}$.

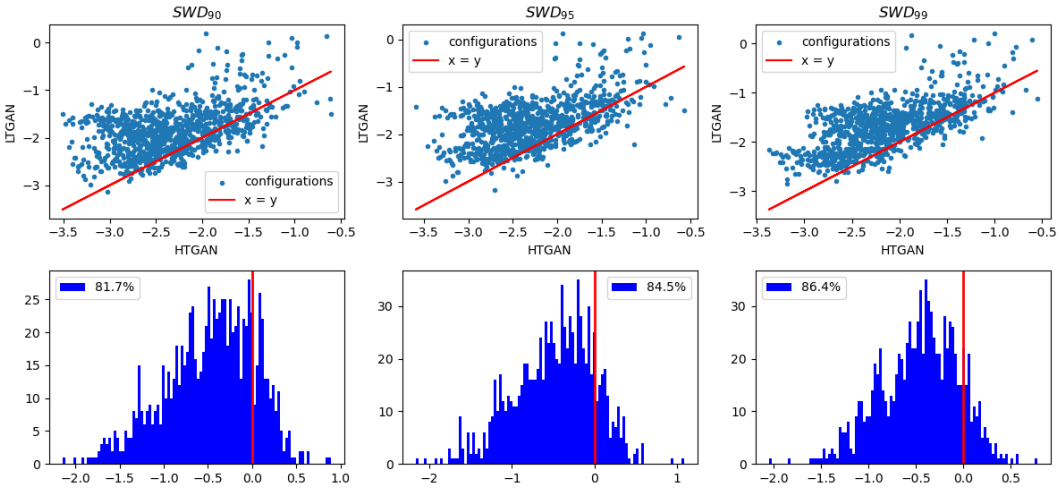
	$\beta = 4/3$			$\beta = 2$			$\beta = 4$		
α	1.5	2	2.5	1.5	2	2.5	1.5	2	2.5
AKE_90	65.8	68.1	65.6	80.4	78.4	66.7	88.7	86.6	72.6
AKE_95	65.6	69.5	66.4	79.7	79.4	69.2	87.8	87.2	75.1
AKE_99	62.5	67.4	65.1	77.0	80.1	72.7	86.8	88.0	78.4
SWD_90	81.7	69.1	45.9	76.7	67.7	52.6	63.7	54.7	41.4
SWD_95	84.5	74.3	51.1	76.9	66.7	54.3	62.0	52.2	39.5
SWD_99	86.4	81.5	64.8	77.5	59.5	49.6	56.9	48.3	34.4

Latent dimension. Equation (1.14) gives evidence that a higher value of the latent dimension N is necessary to obtain a better approximation of the dependence structure. For each experiment specification (*i.e.*a value for α and for d), we have performed a Bayesian hyperparameter search (J. Wu et al. 2019) with 400 runs, with the same hyperparameter ranges as for the previous experiment section. Table 1.2 summarizes statistics on the latent dimension, $\hat{\alpha}$ and SWD_90 for the top 5% run on each experiment. It appears that, empirically, the theoretical result is verified: generally, for the various values of α, as the dimension of the data increases, the optimal value of the latent dimension increases. We also note that, for each value of α, as the dimension of the data increases, the estimate $\hat{\alpha}$ increases accordingly, which means lower tails. However, this conclusion is to be interpreted with caution in view of the great variability associated with the statistics (reflected by the values taken by min and max). This may be a consequence of the instability in training generative models with heavy-tailed inputs (Hickling and Prangle 2024).

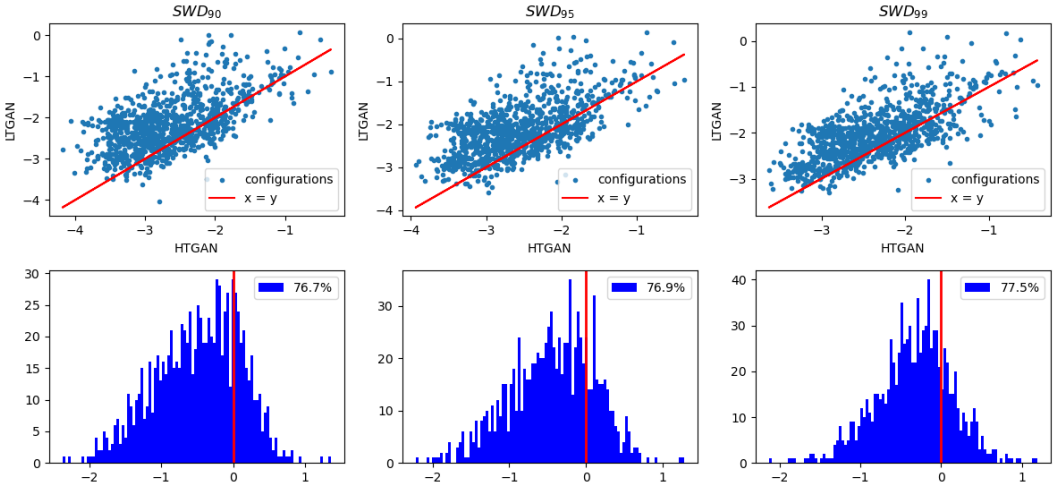
Other models: Gaussian and Hüsler-ReiSS copulas

We extend our evaluation to higher dimensions using two dependence structures: the multivariate Hüsler–ReiSS (HR) extreme-value copula and the Gaussian copula. These represent two qualitatively different tail regimes: HR exhibits non-trivial asymptotic tail dependence, whereas the Gaussian copula is asymptotically independent.

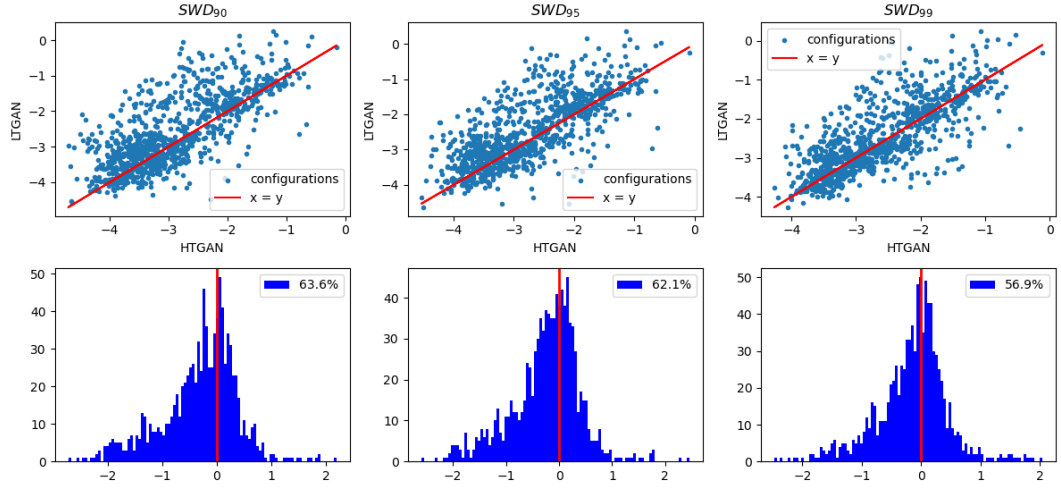
Background: multivariate Hüsler–ReiSS. The HR model is a max-stable (extreme-value) dependence model that arises as the limit of componentwise maxima of *log-Gaussian* vectors, see (Hüsler and Reiss 1989b). In dimension d, the dependence is encoded by a matrix $(\lambda_{ij})_{1 \le i,j \le d}$ with $\lambda_{ii} = 0$ and $\lambda_{ij} > 0$ for $i \neq j$. Smaller λ_{ij} means stronger extremal dependence between components i and j; larger λ_{ij} weakens it. In the bivariate case, the



(a) $\alpha = 1.5$, $\beta = 4/3$

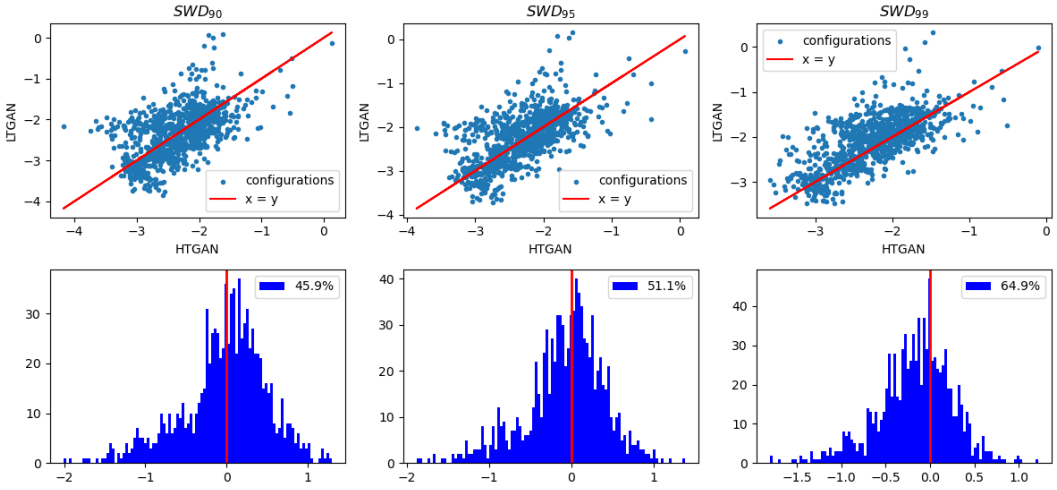


(b) $\alpha = 1.5$, $\beta = 2.0$

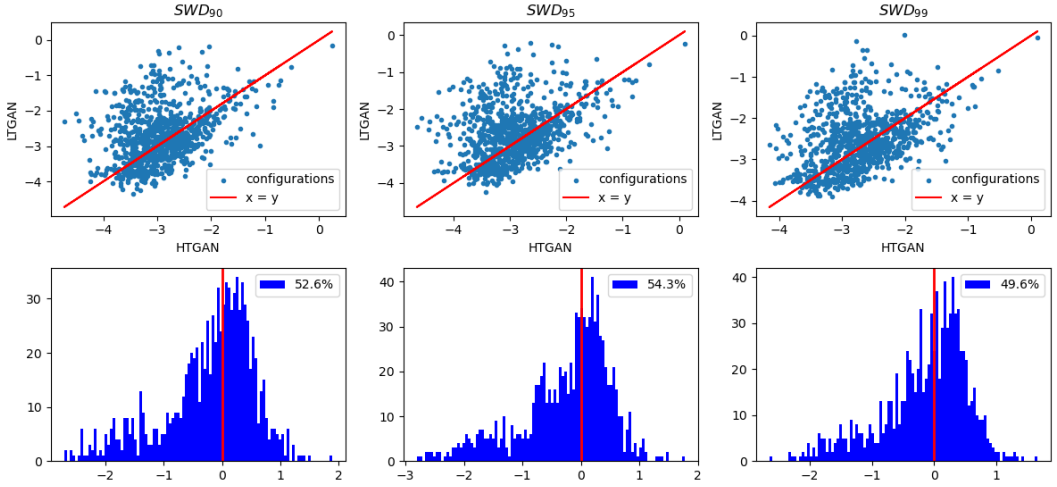


(c) $\alpha = 1.5$, $\beta = 4.0$

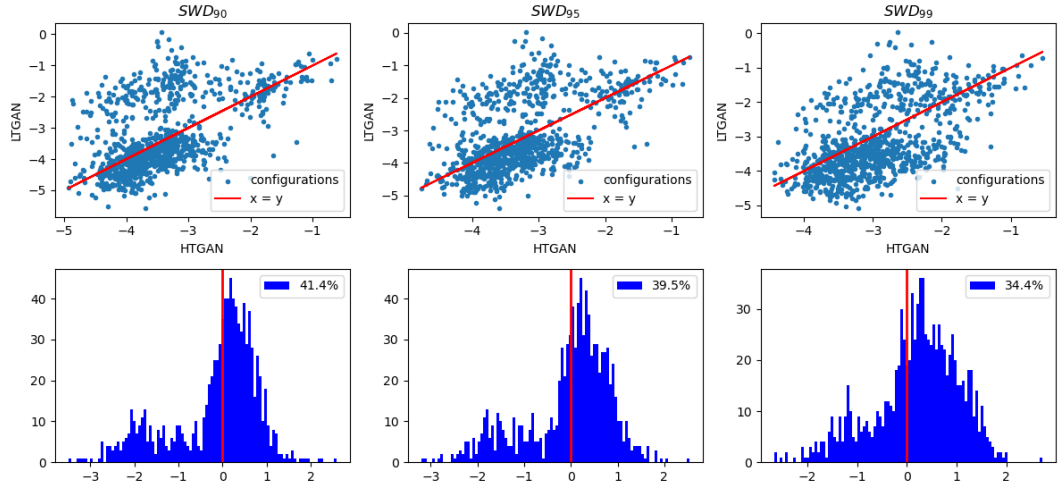
Figure 1.3: Illustration in a d-dimensional setting on simulated data of size $n_{\text{data}} = 10{,}000$ from a Gumbel copula (with dependence parameter β) and Pareto margins (with tail parameter $\alpha = 1.5$). Each group of 2×3 plots corresponds to one experimental setting, *i.e.*a specification of α and β. Figures are averaged over dimensions $d \in \{2, 5, 10, 20, 50\}$. In each of these three groups: the first row plots the specified metric of HTGAN vs LTGAN in log scale for a given hyperparameter configuration. The second row is a histogram of the difference of the log of the metric for HTGAN vs LHTGAN for each parametrization. The legend corresponds to the proportion of cases where HTGAN performs better.



(a) $\alpha = 2.5$, $\beta = 4/3$



(b) $\alpha = 2.5$, $\beta = 2.0$



(c) $\alpha = 2.5$, $\beta = 4.0$

Figure 1.4: Illustration in a d-dimensional setting on simulated data of size $n_{\text{data}} = 10,000$ from a Gumbel copula (with dependence parameter β) and Pareto margins (with tail parameter $\alpha = 2.5$), $d \in \{2, 5, 10, 20, 50\}$. See Figure 1.3 for further details.

Table 1.2: Results on simulated data from a Gumbel copula. Statistics (mean, min, max) on three on the top 5% best performing runs of HTGAN (best performing w.r.t. the SWD_90 metric). Statistics are averaged over $\beta \in \{4/3, 2, 4\}$.

		N (latent dimension)			$\hat{\alpha}$			SWD_90		
α	d	mean	min	max	mean	min	max	mean	min	max
1.5	2	75.96	10	195	1.53	1.16	1.72	0.03	0.01	0.06
	5	91.07	17	197	1.61	1.20	1.96	0.04	0.01	0.06
	10	90.69	21	197	1.72	1.30	2.23	0.04	0.01	0.07
	20	91.33	26	182	1.77	1.23	2.64	0.03	0.01	0.07
	50	107.98	22	188	2.05	1.31	4.00	0.02	0.01	0.04
2.0	2	40.71	5	122	2.07	0.70	3.44	0.04	0.01	0.08
	5	55.96	11	164	2.41	1.63	3.68	0.04	0.01	0.12
	10	57.36	10	187	2.83	1.49	4.66	0.04	0.01	0.10
	20	64.60	10	191	3.03	1.06	4.88	0.03	0.01	0.07
	50	69.02	14	187	3.42	1.63	5.27	0.02	0.01	0.04
2.5	2	39.24	2	154	2.82	1.27	4.92	0.04	0.01	0.10
	5	50.20	7	177	3.54	1.33	7.63	0.05	0.01	0.13
	10	57.40	9	174	4.17	1.31	7.40	0.04	0.01	0.09
	20	72.98	12	175	4.32	0.97	8.79	0.03	0.01	0.06
	50	58.02	9	169	4.44	1.07	7.97	0.02	0.01	0.04

upper tail dependence coefficient is

$$\chi_{ij} = 2\{1 - \Phi(\lambda_{ij})\}, \tag{1.20}$$

where Φ is the cumulative distribution function of the standard Gaussian distribution. Clearly, $\chi_{ij} \downarrow 0$ as $\lambda_{ij} \uparrow \infty$. A convenient spectral construction is via a centered Gaussian vector $\boldsymbol{Z} \in \mathbb{R}^d$ with covariance Σ derived from (λ_{ij}) (Brown–Resnick representation): Set $\boldsymbol{Y} = \exp \boldsymbol{Z}$, normalize $\boldsymbol{S} = \boldsymbol{Y}/\sum_k Y_k$, draw $E \sim \text{Exp}(1)$, and form a unit-Fréchet HR vector $\boldsymbol{X}^{\text{Fr}} = \boldsymbol{S}/E$. Arbitrary continuous margins are then obtained by monotone transforms.

Background: Gaussian copula. Let $\boldsymbol{Z} \sim \mathcal{N}_d(0, \Sigma)$ with equicorrelation $\rho \in (-1/(d-1), 1)$, set $U_i = \Phi(Z_i)$ and apply marginal transforms to obtain X from a Gaussian copula. For $|\rho| < 1$ the Gaussian copula is asymptotically independent ($\chi = 0$), but larger ρ increases sub-asymptotic dependence.

More details on the specific experimental design for these two models are given in Subsection 3.I.

Qualitative summary. We summarize results in Table 1.3. Across both datasets, the trends mirror those observed for the Gumbel copula: heavier tails (lower α) consistently amplify the relative advantage of HTGAN over LTGAN. Therefore, both in asymptotic dependence and in independence settings, HTGAN outperforms LTGAN. Additional figures are provided in Appendix 3.I.

Table 1.3: Results on simulated data from Hüsler-ReiSS and Gaussian copulas. Percentage (%) of parametrizations for which HTGAN is better than a LTGAN for the six considered metrics. Results are given with precision ±0.1%. Values are averaged over data dimensions $d \in \{2, 5, 10, 20, 50\}$.

	Hüsler-ReiSS			Gaussian		
α	1.5	2	2.5	1.5	2	2.5
AKE_90	78.3	81.9	70.8	62.3	57.0	42.9
AKE_95	76.7	78.6	67.4	59.2	56.4	47.2
AKE_99	65.6	73.3	60.2	58.3	53.8	46.2
SWD_90	78.6	63.3	34.4	96.9	96.7	78.3
SWD_95	75.8	68.6	45.6	94.7	93.6	79.4
SWD_99	48.3	44.2	34.2	92.2	90.0	77.8

1.3.3 Experiments on real data

The proposed HTGAN method is tested on the publicly available dataset[5] consisting of daily data for companies of the S&P500. The S&P500 is a stock market index tracking of the 500 most valuable companies in the United States of America. The signal of interest is the absolute value of the normalized marginal returns for each stock: $r_t = |(X_{t+1} - X_t)/X_t|$, which are bounded below by zero and only have an upper tail. The dataset consists of $n_{\text{data}} = 1259$ normalized marginal returns for each ticker. Companies are organized using the Global Industry Classification Standard (GICS). This classification divides companies into 11 sectors: Energy, Materials, Industrials, Consumer Discretionary, Consumer Staples, Health Care, Health Care, Financials, Information Technology, Communication Services, Utilities and Real Estate. Each sector is divided further into industry groups, industry and sub-industry. The returns of the S&P500 present a particular correlation structure illustrated in Figure 1.5 by a heatmap of the table consisting of estimated Kendall's tau for every pair of stock ticker, see the Appendix for more details on Kendall's tau coefficient. It appears that returns are more correlated within each sector, especially in both the Financials and the Utilities sectors, than between sectors. Therefore, in the following, the performance of LTGAN and HTGAN are assessed on the two subsets associated with sectors Financials and Utilities, having both strong correlations (see Figure 1.5). Note that the number of tickers for the Financials and Utilities sector are not the same: $d = 27$ for the Financial sector, and $d = 57$ for the Utilities sectors.

Before moving to the results, let us mention that on the application side of portfolio management, it is important to model the dependence of extreme asset returns well, as shown in (Mainik et al. 2015), in order to manage risks more accurately. This supports the use of HTGAN compared to LTGAN. Their study compares the traditional Markowitz approach with the Extreme Risk Index – a method based on multivariate EVT – to optimize a portfolio of S&P 500 stocks: their results show the superiority of a model that better accounts for dependencies in the tails, compared to the one based only on covariances.

[5] https://www.kaggle.com/datasets/camnugent/sandp500

Sectors and subsectors. The subsectors for the financial sector are: Asset Management & Custody Banks, Consumer Finance, Diversified Banks, Financial Exchanges & Data, Insurance Brokers, Investment Banking & Brokerage, Life & Health Insurance, Multi-Sector Holdings, Multi-line Insurance, Casualty Insurance, Regional Banks and Processing Services. For the Utilities sector, Subsectors are: Electric Utilities, Gas Utilities, Independent Power Producers & Energy Traders, Multi-Utilities and Water Utilities.

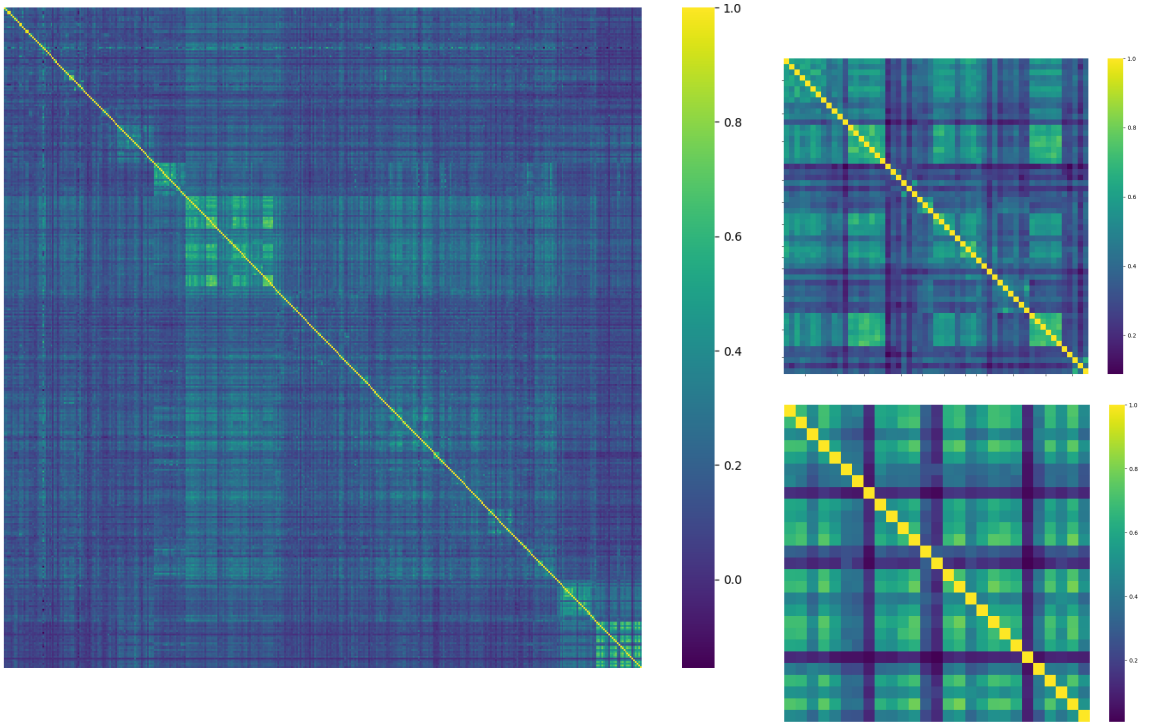


Figure 1.5: Estimated Kendall's τ for every pair of index considered. Left: every S&P500 ticker. On the top right: Tickers for the Financials sector. On the bottom right: Tickers for the Utilities sector. A comprehensive list of sectors and subsectors is given in Paragraph 1.3.3.

Data normalization. In the normalizing step (1.17) of Algorithm 1, two values of $\gamma = 1/\alpha$ are investigated: $\alpha = 1.5$ and $\alpha = 2.5$. The cumulative distribution function is estimated using its empirical counterpart:

$$\widehat{F}_j(x) = \frac{1}{n_{\text{data}} + 1} \sum_{i=1}^{n_{\text{data}}} \mathbb{I}\left\{X_j^{(i)} \leqslant x\right\}. \tag{1.21}$$

This transformation is not invertible, and therefore it is not possible to proceed to step (1.19) in the generation step of Algorithm 2. Therefore, we skip this step in the generation process. In our evaluation, we solely focus on the good reproduction of dependence in extreme regions (*i.e.*the difficult task). Our criteria do not assess how well the marginals are fitted.

Results. Figure 1.6 presents the results for experiments run on the S&P500 dataset. The results argue in favor of better performance of the heavy-tailed noise setting when data is

renormalized to Pareto margins with a larger tail index (case $\alpha = 1.5$). This can be seen in the proportion of cases in which HTGAN performs better than LTGAN: 62.5%, 64.3% and 63.7% respectively for SWD_50, SWD_80 and SWD_90 metrics. In the case of a lighter tail, $\alpha = 2.5$, the performance of HTGAN is degraded, with relative performances of 38.7%, 30.4% and 34.5% respectively for SWD_50, SWD_80 and SWD_90 metrics. Therefore, for better accuracy in extreme regions, we argue in favor of a smaller α in the renormalization step. In summary, HTGAN is generally more effective than LTGAN on this dataset. In addition to this conclusion, we provide visual illustrations of the performance of HTGAN on the Financials dataset in Figure 1.11 of the supplementary material.

1.4 Conclusion

We have provided a theoretical framework to analyse how a HTGAN behaves in extreme regions. We have established a bound on the quality of approximation of the stable tail dependence function of the target distribution. This analysis builds upon the deep connections between Extreme Value Theory and the theory of D-norms.

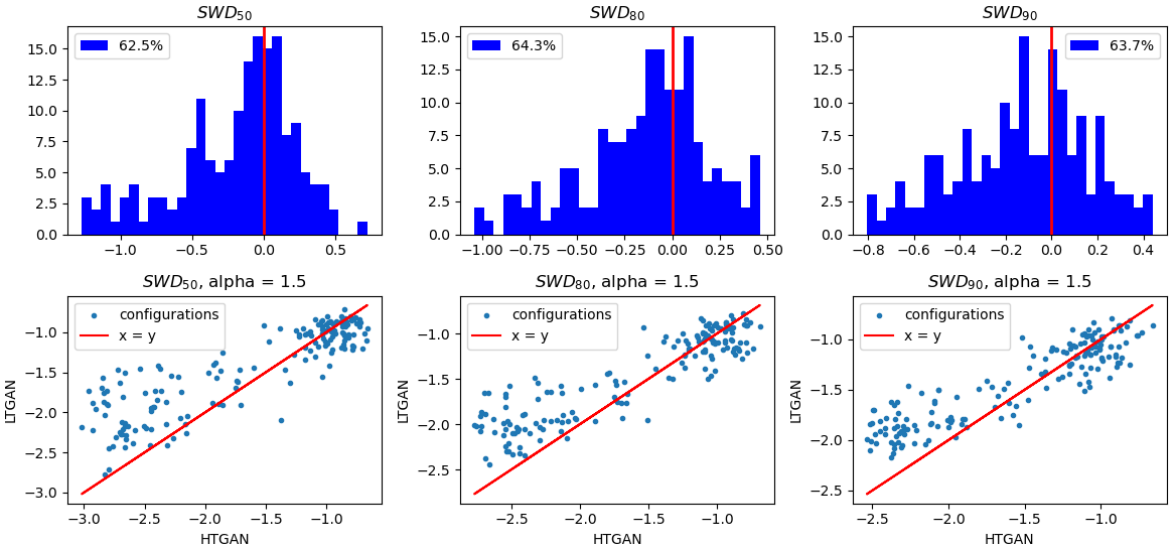
Our numerical experiments support the theoretical results. They show that HTGAN significantly outperforms a standard LTGAN in capturing complex dependencies in the extremes, both on synthetic heavy-tailed datasets with varying dependence structures and on real-world data. We have also demonstrated the key role of the latent dimension in enhancing the model's ability to reproduce such extreme dependencies.

This work highlights, both theoretically and empirically, how introducing a large latent dimension into generative models can improve the modelling of complex dependence structures in extreme regions. While our application focus here has been on financial data, extending this methodology to other domains – such as image generation – remains an exciting direction for future work.

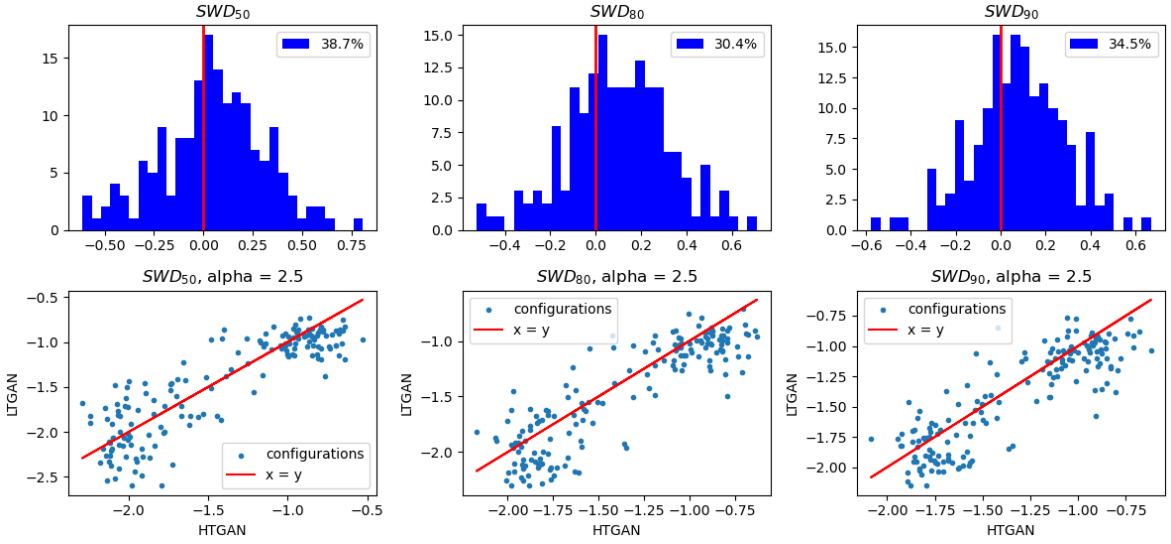
Some important challenges are left open. In particular, we have not addressed the training instability that may arise when working with heavy-tailed inputs. Furthermore, there is ambiguity in selecting the tail parameter α for the marginal normalization step. We suggest treating α as a hyperparameter to be tuned during training, depending on the application and the data at hand.

Acknowledgements

The authors thank the referees for their constructive remarks which improve the manuscript. They acknowledge the support of the Chair "Stress Test, Risk Management and Financial Steering", led by the French Ecole Polytechnique and its Foundation and sponsored by BNP Paribas. S. Girard also gratefully acknowledges support from the French National Research Agency under the grant ANR-23-CE40-0009.



(a) $\alpha = 1.5$



(b) $\alpha = 2.5$

Figure 1.6: Results of the performance of HTGAN vs LTGAN on the Utilities ($d = 27$) and Financials subsectors ($d = 57$) of the S&P500 dataset. Results are aggregated with respect to the dimension d for plot. Each two groups of 2×3 plots corresponds to one experimental setting, *i.e.*$\alpha = 1.5$ (top) and $\alpha = 2.5$ (bottom). In each of these two groups: the first row plots the specified metric of HTGAN vs LTGAN in log scale for a given hyperparametrization configuration. The second row is a histogram of the difference of the log of the metric for HTGAN vs the metric for LTGAN for each parametrization. The legend corresponds to the proportion of cases where HTGAN performs better.

Appendix

3.A Theoretical background

Section 3.B provides additional background on extreme-value theory to complement Section 1.2.1. Section 3.C is devoted to D-norms, introduced in Section 1.2.2. Section 3.D focuses on the theory of quantization. Finally, Section 3.E provides some details on copulas and measures of dependence.

3.B Extreme-value theory

General set-up

Consider a $\mathbb{R}^d$ valued random vector $\boldsymbol{X}$ with associated cdf $\boldsymbol{x} \in \mathbb{R}^d \mapsto F(\boldsymbol{x}) = \mathbb{P}(\boldsymbol{X} \leqslant \boldsymbol{x})$. The primary goal of extreme-value theory is to establish the asymptotic distribution of well normalized maxima of realizations of $\boldsymbol{X}$, see (S. I. Resnick 1987; Beirlant et al. 2004; L. d. Haan and Ferreira 2006) for reference textbooks. Let $(\boldsymbol{X}_i)_{i\in[n]}$ be an i.i.d.sample from F and consider $\boldsymbol{M}_n := \frac{\max_{i\in[n]} \boldsymbol{X}_i - \boldsymbol{b}_n}{\boldsymbol{a}_n}$, where $\boldsymbol{a}_n > \boldsymbol{0}$ and $\boldsymbol{b}_n$ are normalizing sequences in $\mathbb{R}^d$. Clearly, the cdf of $\boldsymbol{M}_n$ is given by $F_{\boldsymbol{M}_n}(\boldsymbol{x}) = F^n(\boldsymbol{a}_n\boldsymbol{x} + \boldsymbol{b}_n)$. This remark gives rise to the following definition:

Definition 1.5 (Max-Domain of Attraction (MDA), (L. d. Haan and Ferreira 2006, Section 6.1.2))**.** A cdf F on $\mathbb{R}^d$ is said to be in the max-domain of attraction of a cdf G, denoted by $F \in \mathrm{Dom}(G)$, if there exist sequences $\boldsymbol{a}_n > 0$ and $\boldsymbol{b}_n$, $n \in \mathbb{N}$ such that:

$$\forall \boldsymbol{x} \in \mathbb{R}^d : \; F^n(\boldsymbol{a}_n\boldsymbol{x} + \boldsymbol{b}_n) \underset{n\to\infty}{\longrightarrow} G(\boldsymbol{x}). \tag{1.22}$$

In other words, the normalized maximum of a sample from F converges in distribution to a random vector with cdf G. It can be shown that the limiting distribution function is necessarily max-stable, as defined below:

Definition 1.6 (Max-stable distribution, (L. d. Haan and Ferreira 2006, Section 6.1.2))**.** A cdf G on $\mathbb{R}^d$ is called max-stable if, for any $n \in \mathbb{N}$, there exist sequences $\boldsymbol{a}_n > 0$ and $\boldsymbol{b}_n \in \mathbb{R}^d$ such that

$$\forall \boldsymbol{x} \in \mathbb{R}^d : \; G^n(\boldsymbol{a}_n\boldsymbol{x} + \boldsymbol{b}_n) = G(\boldsymbol{x}). \tag{1.23}$$

Besides, convergence (1.22) can be extended to a continuous counterpart; one has $F \in \mathrm{Dom}(G)$ if and only if there exist two functions $\boldsymbol{a}_t > 0 \in \mathbb{R}^d$ and $\boldsymbol{\theta} \in \mathbb{R}^d$ such that:

$$\forall \boldsymbol{x} \in \mathbb{R}^d : \; F^t(\boldsymbol{a}_t\boldsymbol{x} + \boldsymbol{\theta}) \underset{t\to\infty}{\longrightarrow} G(\boldsymbol{x}). \tag{1.24}$$

Margins

Let us highlight that, if F fulfills (1.22) or (1.24), then its margins F_j satisfy

$$\forall x \in \mathbb{R} : \; F_j^n(a_{j,n}x + b_{j,n}) \underset{n\to\infty}{\longrightarrow} G_j(x), \tag{1.25}$$

where G_j denotes the jth margin of G, $j \in [d]$. The *univariate* extreme-value theorem (see for instance (Fisher and Tippett 1928; Gnedenko 1943)) then provides a parametric form for the cdf G_js which generalizes the parametrizations of Fréchet ($\gamma > 0$), Gumbel ($\gamma = 0$) and reverse-Weibull ($\gamma < 0$):

Theorem 1.4 ((L. d. Haan and Ferreira 2006, Theorem 1.1.3))**.** *The $\mathbb{R}$-valued cdf's satisfying (1.22) are of the form $G_\gamma(a \cdot + b)$ with $a > 0$, $b \in \mathbb{R}$ and $\gamma \in \mathbb{R}$, where G_γ is a Generalized Extreme Value Distribution (GEVD) defined as*

$$\forall x \in \mathbb{R},\ 1 + \gamma x > 0 :\ G_\gamma(x) = \exp - (1 + \gamma x)^{-1/\gamma}. \tag{1.26}$$

When $\gamma = 0$, G_0 is interpreted as the pointwise limit in (1.26): $G_0(x) = \exp -e^{-x}$.

Therefore, each margin G_j of G can be written as $G_j = G_{\gamma_j}$ following (1.26) with $\gamma_j \in \mathbb{R}$, $j \in [d]$. Let us note that, for each max-stable cdf G, one can consider $G_\star$, the so-called associated simple max-stable distribution, which is defined as:

$$\forall \boldsymbol{x} \in \mathbb{R}^d :\ G_\star(\boldsymbol{x}) = G\left(\frac{\boldsymbol{x}^{\boldsymbol{\gamma}} - \mathbf{1}}{\boldsymbol{\gamma}}\right), \tag{1.27}$$

with $\boldsymbol{\gamma} = (\gamma_j)_{j \in [d]} \in \mathbb{R}^d$. The margins of $G_{\star,j}$ of $G_\star$ are unit Fréchet distributed *i.e.*

$$\forall x > 0,\ G_{\star,j}(x) = \exp -1/x. \tag{1.28}$$

As a consequence, it is possible to characterize a GEVD with two components: On the one hand, the marginal tail indices $\boldsymbol{\gamma}$, which characterize its margins and, on the other hand, its dependence structure encoded in $G_\star$.

Dependence structure

The following result establishes a strong link between the stable tail dependence function (stdf) of G (as introduced in Definition 1.1) and its simple max-stable cdf $G_\star$.

Proposition 1.5 ((L. d. Haan and Ferreira 2006, Section 6.1.5))**.** *Consider a max-stable cdf G and its associated simple max-stable cdf $G_\star$ defined in (1.27). The stdf of G is given by*

$$\forall \boldsymbol{x} \in (0, \infty)^d :\ \ell_G(\boldsymbol{x}) = -\log G_\star(1/\boldsymbol{x}). \tag{1.29}$$

We show in the next paragraph that both ℓ_G and $G_\star$ can be interpreted in terms of D-norms, which is a key property for our analysis.

3.C D-norms

Let us first recall classical results from multivariate extreme-value theory through the lens of D-norms. The reference monograph on the matter is (Falk 2019).

Definition and basic properties

We start with basic definitions.

Definition 1.7 (Norm and D-norms)**.** A function $\|\cdot\| : \mathbb{R}^d \to \mathbb{R}^+$ is a norm on $\mathbb{R}^d$ if it verifies three conditions: a) *Positive definiteness*, b) *Absolute Homogeneity*, c) *Triangle inequality.*

A subset of norms is the set of D-norms defined by

$$\|\cdot\|_{\mathbf{\Gamma}} : \ \boldsymbol{x} \in \mathbb{R}^d \mapsto \|\boldsymbol{x}\|_{\mathbf{\Gamma}} = \mathbb{E}\left[\max_{i\in[d]}(|x_i|\mathbf{\Gamma}_i)\right] \tag{1.30}$$

for a non-negative random variable $\mathbf{\Gamma} \in \mathbb{R}^d$ with unit expectation (Definition 1.2), $\mathbf{\Gamma}$ is also known as the generator of the D-norm.

Let us give a few remarks. First, since D-norms are monotone and radially symmetric, not all norms are D-norms (see (Falk 2019, Chapter 1)). Second, usual norms (such as the L_p-norms for $p \in [1, \infty]$) are D-norms. Last, observe that there are infinitely many generators $\mathbf{\Gamma}$ leading to the same D-norm: multiply $\mathbf{\Gamma}$ by an independent positive random variable $\boldsymbol{U}$ with unit expectation, it readily gives $\|\cdot\|_{\mathbf{\Gamma}\boldsymbol{U}} = \|\cdot\|_{\mathbf{\Gamma}}$. Hence, to ease notations, sometimes we may prefer to write the above norm by $\|\cdot\|_D$ to focus less on the generator $\mathbf{\Gamma}$; in such a situation, we refer to the generator $\mathbf{\Gamma}$ associated with the D-norm by writing $\mathbf{\Gamma} \triangleleft \|\cdot\|_D$, which reads "$\mathbf{\Gamma}$ generates $\|\cdot\|_D$". Despite the multiple generators associated with the same D-norm, the following theorem provides a uniqueness result on the choice of $\mathbf{\Gamma}$ when the norm of the random variable $\mathbf{\Gamma}$ is constrained.

Theorem 1.5 (Normed Generators, (Falk 2019, Theorem 1.7.1))**.** *Let $\|\cdot\|$ be an arbitrary norm on $\mathbb{R}^d$. For any D-norm $\|\cdot\|_D$ on $\mathbb{R}^d$, there exists a generator $\mathbf{\Gamma} \triangleleft \|\cdot\|_D$ and a constant c with the additional property $\mathbb{P}(\|\mathbf{\Gamma}\| = c) = 1$. The distribution of the generator is uniquely defined.*

Let us remark that taking $\|\cdot\| = \|\cdot\|_1$ as the L_1-norm, the constant c is necessarily d: indeed, $\|\mathbf{\Gamma}\|_1 = c$ implies $c = \mathbb{E}[\|\mathbf{\Gamma}\|_1] = \sum_{i\in[d]} \mathbb{E}[\mathbf{\Gamma}_i] = d$.

Relation with stdf

The next result is fundamental to connect a stdf to a D-norm.

Theorem 1.6 (Representation of simple max stable distributions)**.** *A cdf $G_\star$ on $\mathbb{R}^d$ is simple max-stable if and only if there exists a D-norm $\|\cdot\|_D$ on $\mathbb{R}^d$ such that:*

$$\forall \boldsymbol{x} \in (0,\infty)^d : \ G_\star(\boldsymbol{x}) = \exp - \left\|\frac{\mathbf{1}}{\boldsymbol{x}}\right\|_D . \tag{1.31}$$

In particular, if G is a max-stable cdf on $\mathbb{R}^d$, then its stdf is given by

$$\ell_G(\boldsymbol{x}) = \|\boldsymbol{x}\|_D . \tag{1.32}$$

The first statement (1.31) is taken from (Falk 2019, Theorem 2.3.3) or (Falk and Fuller 2021, Theorem 4.1), while the second one (1.32) follows from Proposition 1.5.

3.D Quantization

The quantization problem. The quantization problem is concerned with finding the best approximation of a random vector $\boldsymbol{X} \in \mathbb{R}^d$ with a quantized version, *i.e.*a version that takes a discrete number of values. For a general reference on quantization, see (Graf and Luschgy 2000). Finding the best approximation of a random vector depends on the chosen measure. Define the quantization problem (Graf and Luschgy 2000, Section 10.1) of a probability measure P, for a given norm $\|\cdot\|$ (for example the Euclidean norm), as:

$$e_{N,\infty}(P) := \inf_{f:\mathbb{R}^d \to \mathbb{R}^d} \left\{ P\text{-esssup} \, \|\boldsymbol{X} - f(\boldsymbol{X})\| : \#f(\mathbb{R}^d) \leqslant N \right\} \tag{1.33}$$

$$= \inf_{\substack{\boldsymbol{\alpha} \subset \mathbb{R}^d \\ \#\boldsymbol{\alpha} \leqslant N}} \sup_{\boldsymbol{x} \in \mathrm{supp}(P)} \min_{\boldsymbol{a} \in \boldsymbol{\alpha}} \|\boldsymbol{x} - \boldsymbol{a}\|, \tag{1.34}$$

where $f : \mathbb{R}^d \to \mathbb{R}^d$ is a quantizer. Since $e_{N,\infty}(P)$ depends on the probability measure P only through its support, sometimes (see Lemma 1.1 below) we will write $e_{N,\infty}(A)$ as a function of a set $A \subset \mathbb{R}^d$ (that can be taken as the support of P).

Then, if $\mathrm{supp}(P)$ is compact, Jordan measurable with positive volume (Graf and Luschgy 2000, p.3), the following limit exists:

$$Q_\infty\left(\mathrm{supp}(P)\right) = \lim_{N \to \infty} N^{1/d} e_{N,\infty}(P). \tag{1.35}$$

Note that $Q_\infty\left(\mathrm{supp}(P)\right)$ does not depend on the law of P but only on its support. Therefore, we will interchangeably use a probability distribution or a set as input of $Q_\infty(\cdot)$. The limit $Q_\infty\left(\mathrm{supp}(P)\right)$ is called the covering coefficient or quantization coefficient of order ∞ and can be expressed in terms of the covering coefficient $Q_\infty\left([0,1]^d\right)$. Moreover, with a further constraint, we have the following theorem:

Theorem 1.7 ((Graf and Luschgy 2000, Theorem 10.7))**.** *Let $A \subset \mathbb{R}^d$ be a non empty compact set with $\lambda^d(\partial A) = 0$. Let $Q_\infty\left([0,1]^d\right)$ be the quantization coefficient of $[0,1]^d$ defined in (1.35). Then $Q_\infty\left([0,1]^d\right) > 0$ and:*

$$\lim_{N \to \infty} N^{1/d} e_{N,\infty}(A) = Q_\infty\left([0,1]^d\right) \lambda^d(A)^{1/d}, \tag{1.36}$$

where λ^d is the Lebesgue measure in dimension d.

We state the following technical lemma, which will be helpful for the proof of Theorem 1.3:

Lemma 1.1 ((Graf and Luschgy 2000, Lemma 10.6))**.**

1. *Let $A, B \subset \mathbb{R}^d$ be nonempty compact sets with $A \subset B$. Then,*

$$\forall N \in \mathbb{N} : \; e_{N,\infty}(A) \leqslant e_{N,\infty}(B). \tag{1.37}$$

2. *Let N, m and $\{N_k\}_{k\in[m]}$ be positive integers such that $\sum_{k\in[m]} N_k \leqslant N$ and consider m nonempty compact sets $\{A_k\}_{k\in[m]}$. Then,*

$$e_{N,\infty}\Big(\bigcup_{k\in[m]} A_k\Big) \leqslant \max_{k\in[m]} e_{N_k,\infty}\left(A_k\right). \tag{1.38}$$

Here is a particular case of (1.38) when $N \geqslant 2m$:

$$e_{N,\infty}\Big(\bigcup_{k\in[m]} A_k\Big) \leqslant \max_{k\in[m]} e_{\lceil N/m\rceil-1,\infty}\left(A_k\right). \tag{1.39}$$

Proof. The proof of (1.38) can be found in (Graf and Luschgy 2000, Lemma 10.6). Equation (1.39) is deduced from Equation (1.38) with $N_k = \lceil N/m\rceil - 1$ for all $k \in [m]$. □

3.E Copulas, measures of dependence

Copulas. Let us consider a d-variate random vector $\boldsymbol{X}$ from a cdf F with continuous margins F_j. Sklar's Theorem (Sklar 1959) states that there exists a unique function $C : [0,1]^d \to [0,1]$ such that, for all $\boldsymbol{x} \in \mathbb{R}^d$, $F(x_1,\dots,x_d) = C(F_1(x_1),\dots,F_d(x_d))$.

Introducing for any $j \in [d]$ the uniformly distributed random variable $U_j = F_j(x_j)$, the copula C is the cdf of the joint random vector $(U_1,\dots,U_d)$. Copulas are a tool for studying dependence of probability objects, independently from the margins. See (Nelsen 2006) for a thorough overview. As an example, the Gumbel copula is given by (see (Nelsen 2006, Equation (2.4.2.))):

$$\forall \boldsymbol{u} \in [0,1]^d : \; C_\beta(\boldsymbol{u}) = \exp -\Big(\sum_{i\in[d]} (-\log(u_i)^\beta\Big)^{1/\beta}, \tag{1.40}$$

where $\beta \geqslant 1$.

Definition 1.8 (Kendall's τ, (Nelsen 2006, Theorem 5.1.3.))**.** Let C be a bivariate copula. Its associated Kendall's τ is:

$$\tau = 4\mathbb{E}\left[C(U_1,U_2)\right] - 1, \tag{1.41}$$

where $(U_1,U_2) \sim C$.

Kendall's τ is a common measure of dependence for copulas, see (Joe 2014, Section 6.8) for an estimation procedure. Note that Kendall's τ for the Gumbel copula is given by $\tau = 1 - 1/\beta$, $\beta \geqslant 1$.

Wasserstein distance. Let μ and ν be two measures on $\mathbb{R}^d$. Considering a cost function $c : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^+$ and $p \geqslant 1$, the p - Wasserstein distance associated with the cost c

between μ and ν is defined as (Villani 2009, Chap. 6):

$$\mathcal{W}_p(\mu,\nu) = \left(\inf_{\gamma\in\Gamma(\mu,\nu)} \mathbb{E}_{(\boldsymbol{X},\boldsymbol{Y})\sim\gamma}\left[c(\boldsymbol{X},\boldsymbol{Y})^p\right] \right)^{1/p},$$

where $\Gamma(\mu,\nu)$ is the set of *couplings* of μ and ν, *i.e.*the set of distributions on $\mathbb{R}^d\times\mathbb{R}^d$ such that the marginals are respectively μ and ν. The Wasserstein distance suffers from the fact that there exists no closed formula to compute it, nor very accessible computable estimates in dimensions higher that one. In order to perform estimation, we therefore rely on the sliced-Wasserstein distance, which is an efficient proxy, much used for computations in experimental settings (Nadjahi 2021). It is computed by randomly projecting the two compared measures on axes defined by directions $\mathbf{v}$ and averaging over the projections. Specifically, the sliced-Wasserstein distance is defined as:

$$\mathrm{SWD}_p(\mu,\nu) = \mathop{\mathbb{E}}_{\mathbf{v}\sim\mathcal{U}\left(\mathbb{S}^{d-1}\right)} \left(\mathcal{W}_p^p\left(\mathbf{v}_{\#}\mu,\mathbf{v}_{\#}\nu\right)\right)^{1/p}, \tag{1.42}$$

where $\mathbb{S}^{d-1}$ is the Euclidean unit sphere, $\mathbf{v}_{\#}\mu$ (respectively $\mathbf{v}_{\#}\nu$) stands for the pushforward of the projection $\boldsymbol{X}\in\mathbb{R}^d\mapsto\langle\boldsymbol{X},\mathbf{v}\rangle\in\mathbb{R}$, where $\boldsymbol{X}\sim\mu$ (respectively ν).

3.F Proofs of claims

Proof of Proposition 1.3

The following states an interesting corollary of Theorem 1.2, which we are the first to prove formally as far as we know.

Corollary 1.1 (of Theorem 1.2)**.** *Let $\boldsymbol{Z}\in\mathbb{R}^N$ be a positive random vector with continuous cdf $F_{\boldsymbol{Z}}\in\mathrm{Dom}\left(G_{\boldsymbol{\Gamma}_{\boldsymbol{Z}}}\right)$ where $G_{\boldsymbol{\Gamma}_{\boldsymbol{Z}}}$ is a simple max-stable distribution i.e.$F_{\boldsymbol{Z}}$ satisfies (1.7) and (1.8). Consider $\Phi\in\mathbb{H}_1\left([0,\infty)^N,[0,\infty)^d\right)$, such that:*

$$\mathbb{E}\left[\Phi\left(\boldsymbol{\Gamma}_{\boldsymbol{Z}}\right)\right] > \mathbf{0}. \tag{1.43}$$

Let $\boldsymbol{X}=\Phi(\boldsymbol{Z})$. Then, $\Phi\left(\boldsymbol{\Gamma}_{\boldsymbol{Z}}\right)/\mathbb{E}\left[\Phi\left(\boldsymbol{\Gamma}_{\boldsymbol{Z}}\right)\right] =: \boldsymbol{\Gamma}_{\boldsymbol{X}}$ is a valid generator of a D-norm and $F_{\boldsymbol{X}}\in\mathrm{Dom}\left(G_{\boldsymbol{\Gamma}_{\boldsymbol{X}}}\right)$ where $G_{\boldsymbol{\Gamma}_{\boldsymbol{X}}}$ is a simple max-stable distribution. Moreover, the normalizing constants to ensure convergence in (1.24) are $\boldsymbol{a}_t = t\,\mathbb{E}\left[\Phi\left(\boldsymbol{\Gamma}_{\boldsymbol{Z}}\right)\right]$ and $\boldsymbol{\theta}=\mathbf{0}$.

Proof. In view of the characterization of Theorem 1.2, consider any $h\in\mathbb{H}_1\left([0,\infty)^d,[0,\infty)\right)$ and

$$\Psi_h : \left\{ \begin{array}{ccc} [0,\infty)^N & \to & [0,\infty), \\ \mathbf{z} & \mapsto & h\left(\frac{\Phi(\mathbf{z})}{\mathbb{E}[\Phi(\boldsymbol{\Gamma}_{\boldsymbol{Z}})]}\right). \end{array} \right. \tag{1.44}$$

Clearly, Ψ_h is continuous 1-homogeneous as composition of the continuous 1-homogeneous functions Φ and h. Since $F_{\boldsymbol{Z}}$ verifies (1.8), it follows that:

$$t\,\mathbb{P}\left(h\left(\frac{\Phi(\boldsymbol{Z})}{\mathbb{E}\left[\Phi(\boldsymbol{\Gamma}_{\boldsymbol{Z}})\right]}\right)>t\right) = t\,\mathbb{P}\left(\Psi_h(\boldsymbol{Z})>t\right) \underset{t\to\infty}{\longrightarrow} \mathbb{E}\left[\Psi_h(\boldsymbol{\Gamma}_{\boldsymbol{Z}})\right] = \mathbb{E}\left[h\left(\frac{\Phi(\boldsymbol{\Gamma}_{\boldsymbol{Z}})}{\mathbb{E}\left[\Phi(\boldsymbol{\Gamma}_{\boldsymbol{Z}})\right]}\right)\right]. \tag{1.45}$$

Introducing $\boldsymbol{Y} = \Phi(\boldsymbol{Z})/\mathbb{E}\left[\Phi(\boldsymbol{\Gamma_Z})\right] = \boldsymbol{X}/\mathbb{E}\left[\Phi(\boldsymbol{\Gamma_Z})\right]$ and its corresponding distribution function $F_{\boldsymbol{Y}}$, the previous convergence can be rewritten as:

$$\forall h \in \mathbb{H}_1\left([0,\infty)^d, [0,\infty)\right): \quad t\,\mathbb{P}\left(h\left(\boldsymbol{Y}\right) > t\right) \underset{t\to\infty}{\longrightarrow} \mathbb{E}\left[h\left(\boldsymbol{\Gamma_X}\right)\right], \tag{1.46}$$

so that $\boldsymbol{Y}$ satisfies characterization (1.8) in Theorem 1.2. Moreover, since $\mathbb{E}\left[\boldsymbol{\Gamma_X}\right] = \mathbf{1}$ and the coordinates of $\boldsymbol{\Gamma_X}$ are non-negative, $\boldsymbol{\Gamma_X}$ is indeed the generator of a D-norm. Therefore:

$$\forall \boldsymbol{x} \in (0,\infty)^d: \quad F_{\boldsymbol{Y}}^t(t\boldsymbol{x}) \underset{t\to\infty}{\longrightarrow} \exp - \left\|\frac{1}{\boldsymbol{x}}\right\|_{\boldsymbol{\Gamma_X}}. \tag{1.47}$$

Noting that $F_{\boldsymbol{Y}}(\boldsymbol{x}) = F_{\boldsymbol{X}}(\mathbb{E}\left[\Phi(\boldsymbol{\Gamma_Z})\right]\boldsymbol{x})$ finally proves

$$\forall \boldsymbol{x} \in (0,\infty)^d: \quad F_{\boldsymbol{X}}^t(t\,\mathbb{E}\left[\Phi(\boldsymbol{\Gamma_Z})\right]\boldsymbol{x}) \underset{t\to\infty}{\longrightarrow} \exp - \left\|\frac{1}{\boldsymbol{x}}\right\|_{\boldsymbol{\Gamma_X}} =: G_{\boldsymbol{\Gamma_X}}(\boldsymbol{x}), \tag{1.48}$$

and the result is proved. □

We now complete the proof of Proposition 1.3. The previous result holds for continuous 1-homogeneous function Φ, which is not exactly the situation with a neural network with ReLU activation functions because of the bias terms (see Proposition 1.2). However, we claim that, asymptotically, these bias terms do not play any role on the stdf. To see this, let us investigate the behavior of a ReLU-neural network when going from a layer to the next one.

Denoting by σ the ReLU activation function, and supposing that $\varphi_l = \sigma$, $\forall l \in [L]$, we have

$$\begin{aligned} \boldsymbol{Z}_l &= \sigma\left(\mathbf{W}_l \boldsymbol{Z}_{l-1} + \boldsymbol{b}_l\right) \\ &= \sigma\left(\mathbf{W}_l \boldsymbol{Z}_{l-1}\right) + \left[\sigma\left(\mathbf{W}_l \boldsymbol{Z}_{l-1} + \boldsymbol{b}_l\right) - \sigma\left(\mathbf{W}_l \boldsymbol{Z}_{l-1}\right)\right]. \end{aligned} \tag{1.49}$$

On the one hand, $\sigma\left(\mathbf{W}_l \boldsymbol{Z}_{l-1}\right)$ is a 1-homogeneous function of $\boldsymbol{Z}_{l-1}$ and is thus a heavy-tailed random variable if $\mathbf{W}_l \neq \mathbf{0}$ and $\boldsymbol{Z}_{l-1}$ is heavy-tailed itself. On the other hand, the remaining term $\left[\cdots\right]$ in (1.49) is a bounded random variable (bounded by $|\boldsymbol{b}_l|$); the stdf of the sum of a heavy-tailed random variable and a bounded one is the stdf of the heavy-tailed random variable. The stdf of (1.49) is therefore the stdf of the transformation without the bias, and by induction the stdf of the output is the stdf of the input after iterative applications of $\sigma\left(\mathbf{W}_l\,\cdot\right)$ for $l \in [L]$, which is 1-homogeneous continuous. Let denote by Φ the output of the L compositions of $\sigma\left(\mathbf{W}_l\,\cdot\right)$: since the matrix weights $\mathbf{W}_l$ are non-negative, $\Phi \in \mathbb{H}_1\left([0,\infty)^N, [0,\infty)^d\right)$. Therefore, Corollary 1.1 applies and so the stdf associated with the output of the network is $\boldsymbol{\Gamma}_{\mathcal{G}_Z} = \Phi(\boldsymbol{\Gamma_Z})/\mathbb{E}\Phi(\boldsymbol{\Gamma_Z})$, which has at most N atoms. Thus, the proof is concluded.

3.G Proof of Proposition 1.4

The next lemma can be interpreted in the following way. If a discrete random vector generates a D-norm, then it admits a discrete counterpart which can be any other discrete

distribution with modified atoms: it is also a generator of the concerned D-norm.

Lemma 1.2. *Let* $\mathbf{\Gamma} \triangleleft \|\cdot\|_D$ *where* $\mathbf{\Gamma} \in [0,\infty)^d$ *is discrete, that is* $\mathbf{\Gamma} \sim \mu_{\mathbf{\Gamma}}$ *where*

$$\mu_{\mathbf{\Gamma}} = \sum_{i\in[N]} p_i \delta_{\gamma^{(i)}}. \tag{1.50}$$

Consider a new discrete probability measure with positive weights $(q_i)_{i\in[N]}$ *and a new random variable* $\widetilde{\mathbf{\Gamma}} \sim \mu_{\widetilde{\mathbf{\Gamma}}}$*, where*

$$\mu_{\widetilde{\mathbf{\Gamma}}} := \sum_{i\in[N]} q_i \delta_{\widetilde{\gamma}^{(i)}}, \tag{1.51}$$

with $\widetilde{\gamma}^{(i)} = \frac{p_i}{q_i}\gamma^{(i)}$*,* $i \in [N]$*. Then,* $\widetilde{\mathbf{\Gamma}}$ *is a valid generator and* $\widetilde{\mathbf{\Gamma}} \triangleleft \|\cdot\|_D$*.*

Proof. By definition of the D-norm, one has $\forall \boldsymbol{x} \in \mathbb{R}^N$:

$$\begin{aligned}
\|\boldsymbol{x}\|_D &= \mathbb{E}_{\mathbf{\Gamma}}\left[\max_{j\in[N]}\{|x_j|\mathbf{\Gamma}_j\}\right] \\
&= \sum_{i\in[N]} p_i \max_{j\in[N]}\left\{|x_j|\gamma_j^{(i)}\right\} \\
&= \sum_{i\in[N]} q_i \max_{j\in[N]}\left\{|x_j|\widetilde{\gamma}_j^{(i)}\right\} = \mathbb{E}_{\widetilde{\mathbf{\Gamma}}}\left[\max_{j\in[N]}\left\{|x_j|\widetilde{\mathbf{\Gamma}}_j\right\}\right].
\end{aligned}$$

Since $\mathbb{E}_{\widetilde{\mathbf{\Gamma}}}\left[\widetilde{\mathbf{\Gamma}}\right] = \mathbb{E}_{\mathbf{\Gamma}}\left[\mathbf{\Gamma}\right] = \mathbf{1}$ and $\widetilde{\mathbf{\Gamma}} \in [0,\infty)^d$, $\widetilde{\mathbf{\Gamma}}$ is a valid generator. □

In particular, it is possible to choose uniform weights, $q_i = 1/N$, $i \in [N]$. In other words, if a D-norm admits a generator which admits a finite number of values, it also admits a generator with the same number of finite values, with uniform probability. This result is important as it means that the class of generators of D-norms which take a finite number of values and with uniform weights is as rich as the class of generator which take finite values with arbitrary weights.

Proposition 1.6. *Consider a generator* $\mathbf{\Gamma}$ *of a D-norm in* $\mathbb{R}^d$*. Suppose that it is discrete uniform with* N *atoms* $(\gamma^{(i)})_{i\in[N]}$*:*

$$\mu_{\mathbf{\Gamma}} = \frac{1}{N}\sum_{i\in[N]} \delta_{\gamma^{(i)}}. \tag{1.52}$$

Consider morever a random vector $\boldsymbol{Z} \in \mathbb{R}^N$ *with i.i.d.unit Fréchet margins:* $\boldsymbol{Z} = (Z_1,\ldots,Z_N)$*. Then, there exists a linear mapping* $\Phi : \mathbb{R}^N \to \mathbb{R}^d$ *such that:*

$$F_{\Phi(\boldsymbol{Z})} \in \mathrm{Dom}\left(G_{\mathbf{\Gamma}}\right). \tag{1.53}$$

Proof. Let us first remark that $F_{\boldsymbol{Z}} \in \mathrm{Dom}\left(G_{\mathbf{\Gamma}_{\boldsymbol{Z}}}\right)$ with:

$$\mu_{\mathbf{\Gamma}_{\boldsymbol{Z}}} = \frac{1}{N}\sum_{i\in[N]} \delta_{N\mathbf{e}_i},$$

see Example 1.1. Consider $\mathbf{\Gamma} \in \mathbb{R}^N$ a valid generator of a D-norm with a discrete uniform distribution (1.52). There exists $\Phi : \mathbb{R}^N \to \mathbb{R}^d$ a linear function such that $\forall i \in [N] : \Phi(N\mathbf{e}_i) = \gamma^{(i)}$. Let us note that Φ is continuous $1-$homogeneous because it is linear. Moreover, remark that $\forall i \in [N] : \gamma^{(i)} \in [0,\infty)^d$ implies $\Phi\left([0,\infty)^N\right) \subset \Phi\left([0,\infty)^d\right)$ and thus $\Phi(\boldsymbol{Z}) \in [0,\infty)^d$. Corollary 1.1 ensures that

$$F_{\Phi(\boldsymbol{Z})} \in \mathrm{Dom}\left(G_{\Phi(\mathbf{\Gamma}_{\boldsymbol{Z}})/\mathbb{E}(\Phi(\mathbf{\Gamma}_{\boldsymbol{Z}}))}\right) = \mathrm{Dom}\left(G_{\mathbf{\Gamma}}\right) \tag{1.54}$$

since, as $\Phi\left(\boldsymbol{e}_i\right) = \gamma^{(i)}$, $\Phi(\mathbf{\Gamma}_{\boldsymbol{Z}})$ is equal in distribution to $\mathbf{\Gamma}$ (note that $\mathbb{E}\left[\Phi(\mathbf{\Gamma}_{\boldsymbol{Z}})\right] = \mathbf{1}$). The result is thus proved. □

Remark 1.1. This result ensures that, given any random vector of interest $\boldsymbol{X}$ for which the associated D-norm generator $\mathbf{\Gamma}_{\boldsymbol{X}}$ has a uniform distribution with N atoms, one can find a linear map Φ transforming a noise with unit Fréchet margins with at least N atoms to a distribution whose D-norm generator is the desired one.

Lemma 1.2 and Proposition 1.6 together prove Proposition 1.4. □

3.H Proof of Theorem 1.3

The following technical lemma will be useful to prove the result:

Lemma 1.3. *$\forall \boldsymbol{a}, \boldsymbol{b}, \mathbf{c} \in \mathbb{R}^d$:*

$$\left|\max_{i\in[d]} a_i b_i - \max_{i\in[d]} a_i c_i\right| \leqslant \|\boldsymbol{a}\|_\infty \max_{i\in[d]} |b_i - c_i|. \tag{1.55}$$

Proof. Without loss of generality, suppose that $\max_{i\in[d]} a_i b_i \geqslant \max_{i\in[d]} a_i c_i$, the argument being symmetric. One has:

$$\begin{aligned}\left|\max_{i\in[d]} a_i b_i - \max_{i\in[d]} a_i c_i\right| &= \max_{i\in[d]}\left\{a_i c_i + a_i b_i - a_i c_i\right\} - \max_{i\in[d]} a_i c_i \\ &\leqslant \max_{i\in[d]} a_i c_i + \max_{i\in[d]}\left\{a_i b_i - a_i c_i\right\} - \max_{i\in[d]} a_i c_i \\ &\leqslant \max_{i\in[d]} |a_i| \max_{i\in[d]} |b_i - c_i|\end{aligned}$$

and the result is proved. □

We seek to find a upper bound on:

$$\inf_{\theta\in\Theta} \sup_{x\in(0,\infty)^d} |\ell_F(\boldsymbol{x}) - \ell_{G_{\theta,N}}(\boldsymbol{x})| / \|\boldsymbol{x}\|_\infty . \tag{1.56}$$

In view of the above, and remembering that the stdf of the output of a neural network is discrete (Proposition 1.3), our goal is to find the best approximation of the D-norm $\|.\|_{\mathbf{\Gamma}}$ with a discretized generator with N atoms; we will leverage results from quantization

(Section 3.D). Take $\mathbf{\Gamma}$ an arbitrary generator of the D-norm related to ℓ_F. One has:

$$\begin{aligned}
&\min_{f:\mathbb{R}^d\to\mathbb{R}^d,\#f(\mathbb{R}^d)=N} \left| \|\boldsymbol{x}\|_{\mathbf{\Gamma}} - \|\boldsymbol{x}\|_{f(\mathbf{\Gamma})} \right| \\
&= \min_{f:\mathbb{R}^d\to\mathbb{R}^d,\#f(\mathbb{R}^d)=N} \left| \mathbb{E}_{\mathbf{\Gamma}} \left[\max_{i\in[d]}(|x_i|\mathbf{\Gamma}_i) - \max_{i\in[d]}(|x_i|f(\mathbf{\Gamma}_i)) \right] \right|, \qquad (1.57) \\
&\leqslant \min_{f:\mathbb{R}^d\to\mathbb{R}^d,\#f(\mathbb{R}^d)=N} \|\boldsymbol{x}\|_\infty \, \mathbb{E}_{\mathbf{\Gamma}} \left[\max_{i\in[d]} |\mathbf{\Gamma}_i - f(\mathbf{\Gamma}_i)| \right] \qquad (1.58) \\
&\leqslant \|\boldsymbol{x}\|_\infty \, e_{N,\infty}(\mathbf{\Gamma}), \qquad (1.59)
\end{aligned}$$

from Lemma 1.3. To bound (1.59), we use a quantizer which has a constant sup-norm, which existence is guaranteed by (Theorem 1.5):

$$\|\mathbf{\Gamma}\|_\infty = c \quad a.s.. \qquad (1.60)$$

Observe that the constant c must be related to the stdf using Definition 1.2:

$$\ell_F(1) = \mathbb{E}\left[\|\mathbf{\Gamma}\|_\infty\right] = c.$$

Now, we are in a position to derive a bound on $e_{N,\infty}(\mathbf{\Gamma})$.

Definition 1.9. For any norm $\|\cdot\|$, let us denote by $\mathcal{B}_{\|\cdot\|}$ the associated unit ball. Besides, let $\mathcal{B}^+_{\|\cdot\|} = \mathcal{B}_{\|\cdot\|} \cap [0,\infty)^d$ be the part of the unit ball in the positive orthant.

Owing to (1.60) and Definition 1.2, $\operatorname{supp}(\mathbf{\Gamma}) \subset c\cdot[0,1]^d$. Equation (1.36) shows that one can get an error bound for (1.59) with rate $N^{-1/d}$. However, it is possible to improve this result by remarking that $\mathbf{\Gamma}$ takes values on the boundary of a hypercube (see (1.60)). Indeed, now note that:

$$\operatorname{supp}(\mathbf{\Gamma}) \subset c\cdot\mathcal{B}^+_{\|\cdot\|_\infty} \subset \bigcup_{i\in[d]} c\cdot\mathcal{B}^+_{\|\cdot\|_\infty}(i), \qquad (1.61)$$

where, for all $i\in[d]$,

$$\mathcal{B}^+_{\|\cdot\|_\infty}(i) = \left\{(x_1,\ldots,x_{i-1},1,x_{i+1},\ldots,x_d) : (x_1,\ldots,x_{i-1},x_{i+1},\ldots,x_d)\in[0,1]^{d-1}\right\}. \qquad (1.62)$$

Remark 1.2. For all $i\in[d]$, $\mathcal{B}^+_{\|\cdot\|_\infty}(i)$ is canonically homeomorph to $[0,1]^{d-1}$ in the topological space $\mathbb{R}^{d-1}$ endowed with the topology of $\|\cdot\|_\infty$.

Using Lemma 1.1 (inequality (1.37) with (1.61) from first to second term and from second to third and inequality (1.38) from third to fourth), we get

$$e_{N,\infty}(\mathbf{\Gamma}) \leqslant c\cdot e_{N,\infty}\left(\mathcal{B}^+_{\|\cdot\|_\infty}\right) \leqslant c\cdot e_{\lceil N/d\rceil-1,\infty}\left(\mathcal{B}^+_{\|\cdot\|_\infty}(1)\right) = c\cdot e_{\lceil N/d\rceil-1,\infty}\left([0,1]^{d-1}\right). \qquad (1.63)$$

Gathering previous arguments, we have proved the following:

Lemma 1.4. *Consider $\mathbf{\Gamma}\in\mathbb{R}^d$ a generator of a D-norm such that $\|\mathbf{\Gamma}\|_\infty = c$ a.s.. Then,*

the following holds true:

$$\forall \boldsymbol{x} \in \mathbb{R}^d : \min_{f:\mathbb{R}^d \to \mathbb{R}^d, \#f(\mathbb{R}^d)=N} \left| \|\boldsymbol{x}\|_{\boldsymbol{\Gamma}} - \|\boldsymbol{x}\|_{f(\boldsymbol{\Gamma})} \right| \leqslant c \cdot \|\boldsymbol{x}\|_{\infty} e_{\lceil N/d \rceil - 1, \infty} \left([0,1]^{d-1} \right). \quad (1.64)$$

From (1.36), we get that:

$$e_{\lceil N/d \rceil - 1, \infty} \left([0,1]^{d-1} \right) \overset{N \to \infty}{\sim} \left(\lceil N/d \rceil - 1 \right)^{-1/(d-1)} Q_{\infty} \left([0,1]^{d-1} \right) \lambda([0,1]^{d-1})^{1/(d-1)} \quad (1.65)$$

$$\overset{N \to \infty}{\sim} d^{1/(d-1)} N^{-1/(d-1)} Q_{\infty} \left([0,1]^{d-1} \right) \quad (1.66)$$

which proves (1.14), where $C(d) = c \cdot d^{1/(d-1)} Q_{\infty} \left([0,1]^{d-1} \right)$ in (1.15). The proof of Theorem 1.3 is thus complete. □

Observe that the quantization upper bound (1.59) is a bit rough since the right-hand side of (1.58) refers to the L_1-quantization that we bound using L_∞-quantization. One could use L_1-quantization estimates but they would depend on the distribution of $\boldsymbol{\Gamma}$ and we prefer to provide worst-case estimates. Instead of using the L_∞ and using L_1 quantization on (1.58), (Graf and Luschgy 2000, Th. 6.2.) gives that the first order error of the error is proportional to the $L_{(d-1)/d}$ norm of the density of the generator on the boundaries of $\mathcal{B}^+_{\|\cdot\|_\infty}(i)$ times $N^{-1/(d-1)}$. In particular, if the generator $\boldsymbol{\Gamma}$ has a null density with respect to the Lebesgue measure (e.g.if the distribution is discrete), the multiplicative factor of the $N^{-1/(d-1)}$ term is null and the convergence of the bound can be quicker than $N^{-1/(d-1)}$.

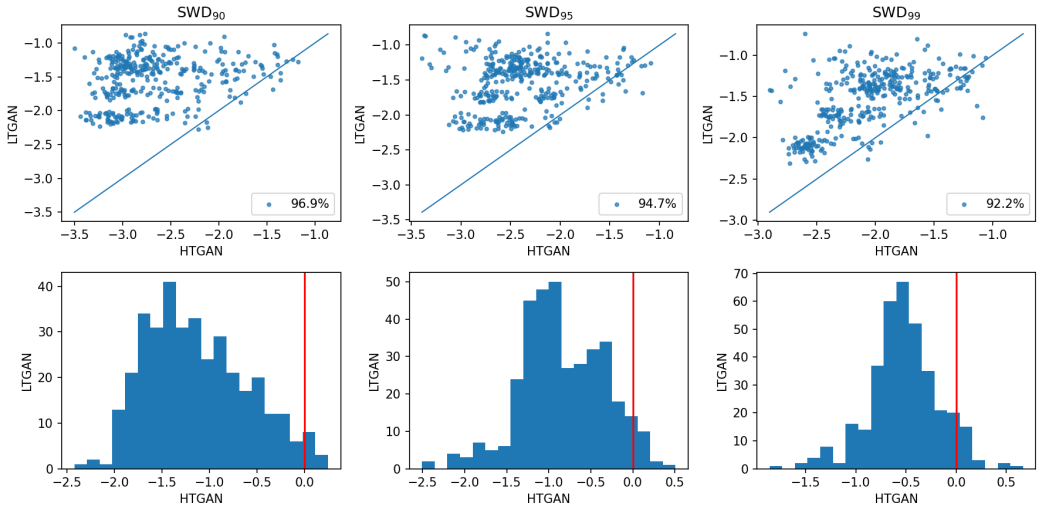
3.I Additional experiments

Experiments on Hüsler-ReiSS and Gaussian models

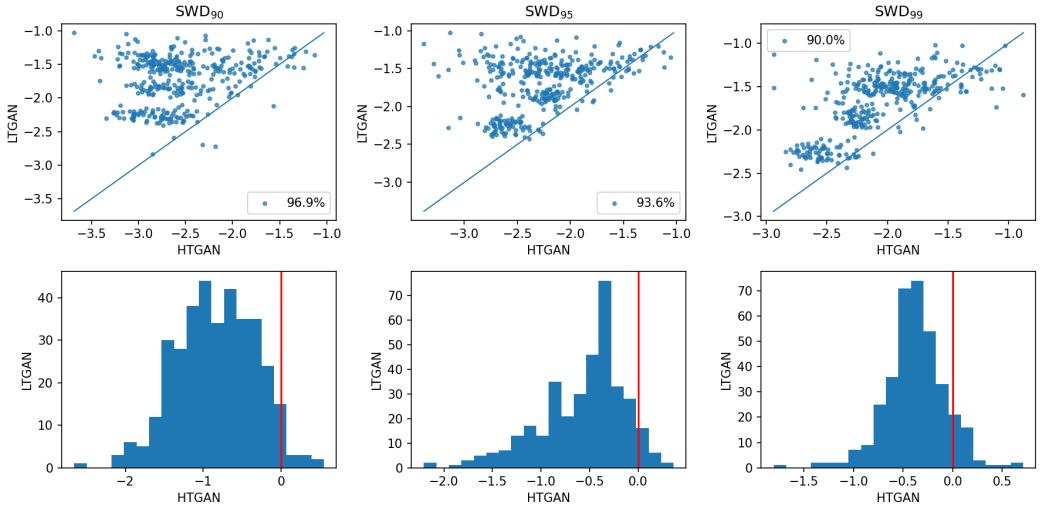
Experimental setup. We consider dimensions $d \in \{2, 5, 10, 20, 50\}$. In the Gaussian copula model, the equicorrelation parameter is fixed at $\rho = 0.5$. In the Hüsler-ReiSS case, the equi-λ model is adopted with $\lambda = 1.0$ (*i.e.*$\lambda_{ij} = 1$ for $i \neq j$). In both datasets, the margins are Pareto distributed with tail index $\alpha \in \{1.5, 2.0, 2.5\}$ (implemented by inverse-cdf transforms after simulating the copula). For each pair (d, α) and dataset type, we generate 10,000 training and 2,000 validation points. Results are provided in Table 1.3 of the main paper and Figures 1.7, 1.8, 1.9 and 1.10.

3.J Additional plots for experiments on Real data

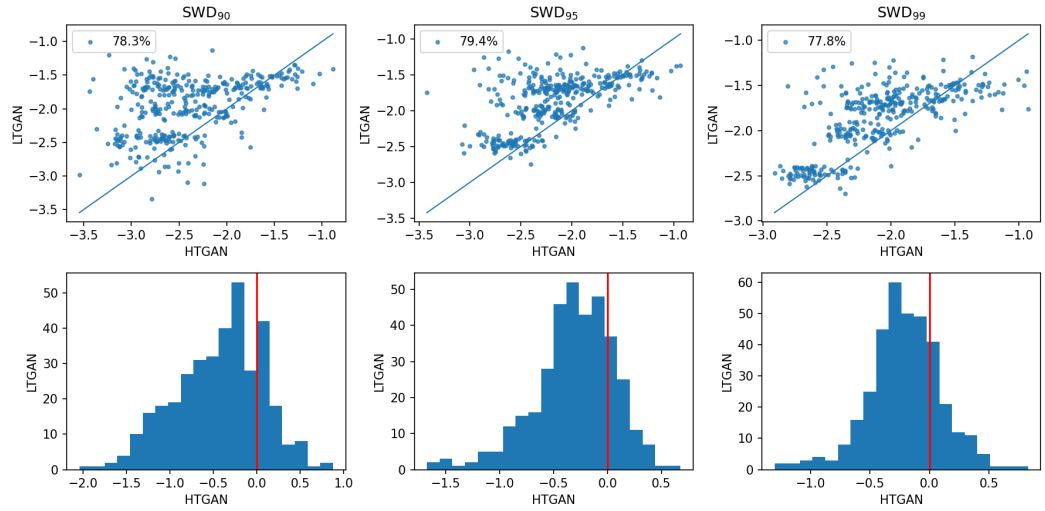
We provide additional illustrations for assessing the performance of HTGAN for real data, on the Financials dataset in Figure 1.11. Real vs generated data are plotted with: (i) histograms of marginal densities for four tickers, (ii) scatter plots for two tickers and (iii) histograms of angles of large data points after projection for two tickers. For scatter plots and angles, two tickers are selected such that they exhibit quite different values of Kendall's τ, chosen on the relevant heatmap in Figure 1.5. The selected securities are: Ameriprise Financial (AMP), Franklin Ressources (BEN), BNY Mellon (BK) and BlackRock (BLK). This contrast in dependence is reflected in the corresponding scatter plots and angular



(a) $\alpha = 1.5$

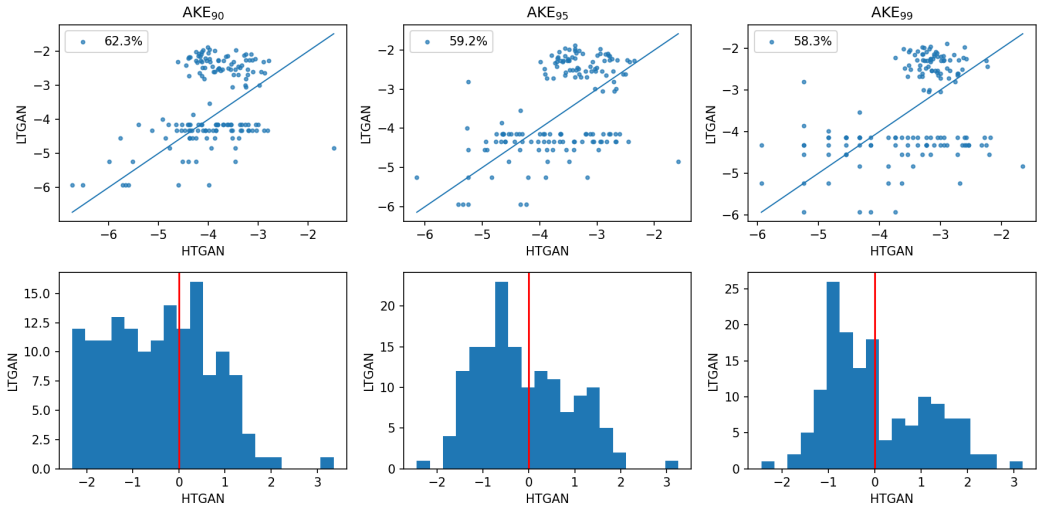


(b) $\alpha = 2.0$

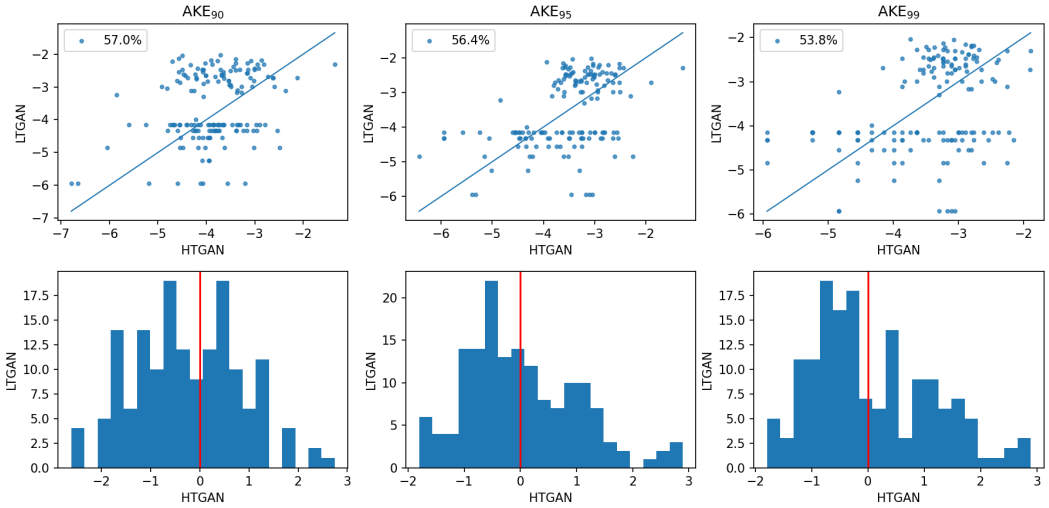


(c) $\alpha = 2.5$

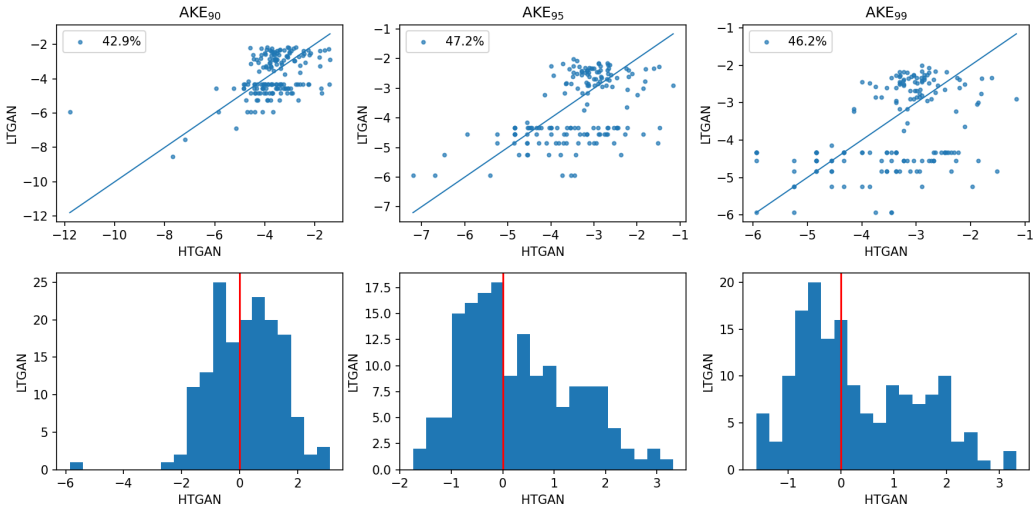
Figure 1.7: Illustration in a d-dimensional setting on simulated data of size $n_{\text{data}} = 10,000$ from a Gaussian copula and Pareto margins, $d \in \{2, 5, 10, 20, 50\}$. Results obtained with the SWD metric. See Figure 1.3 for further details.



(a) $\alpha = 1.5$

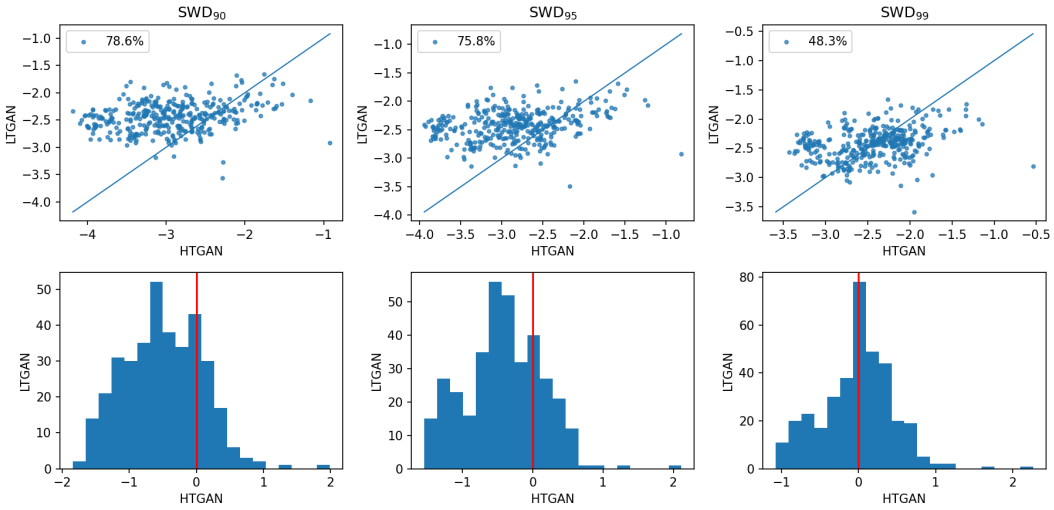


(b) $\alpha = 2.0$

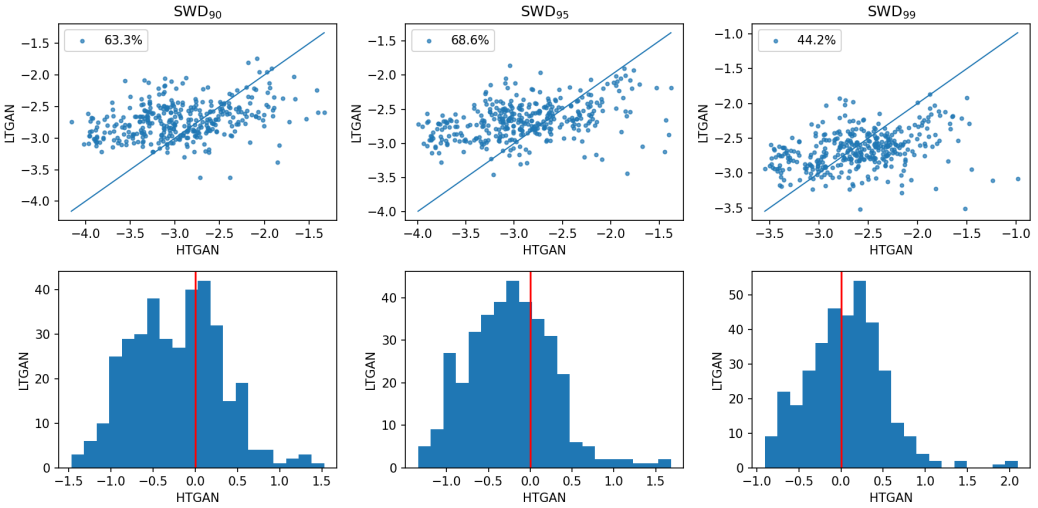


(c) $\alpha = 2.5$

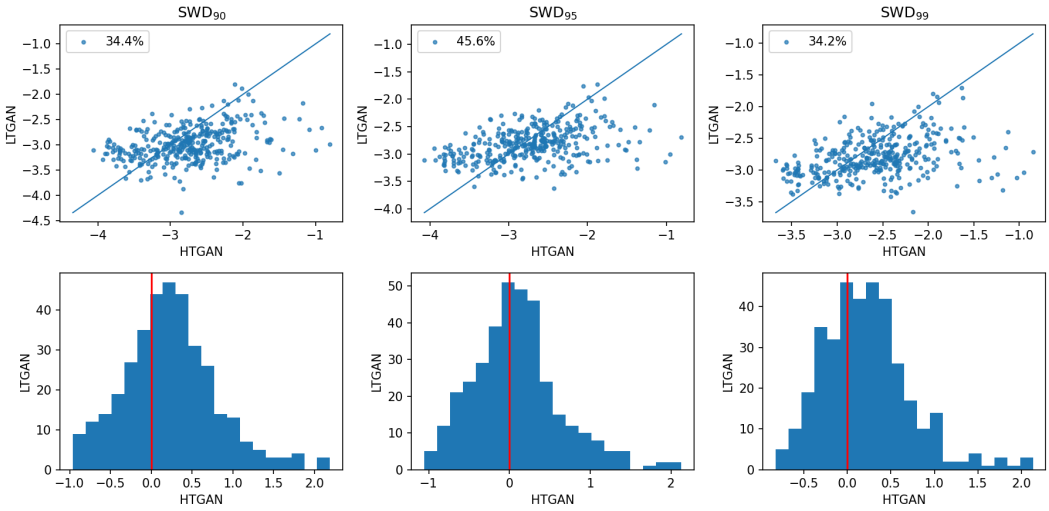
Figure 1.8: Illustration in a d-dimensional setting on simulated data of size $n_{\text{data}} = 10,000$ from a Gaussian copula and Pareto margins, $d \in \{2, 5, 10, 20, 50\}$. Results obtained with the AKE metric. See Figure 1.3 for further details.



(a) $\alpha = 1.5$

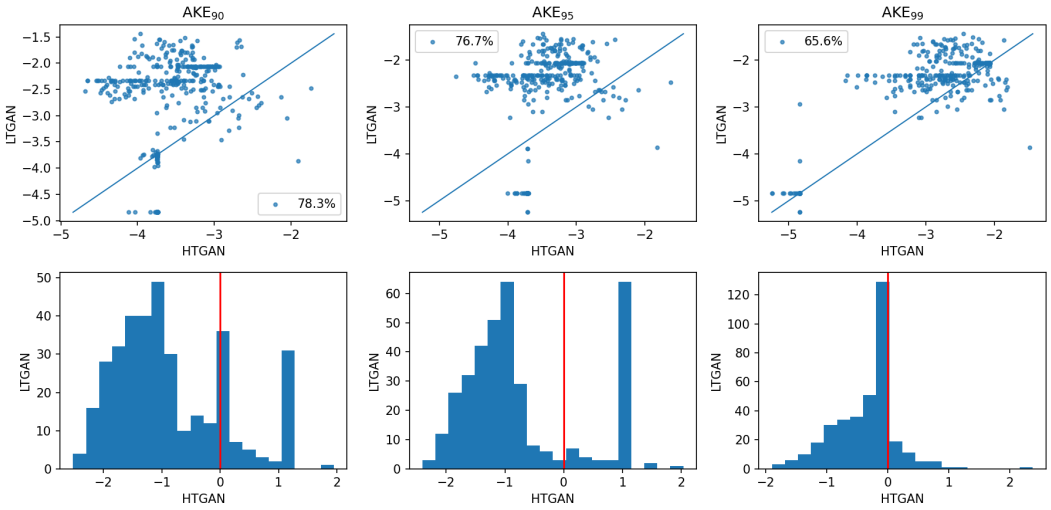


(b) $\alpha = 2.0$

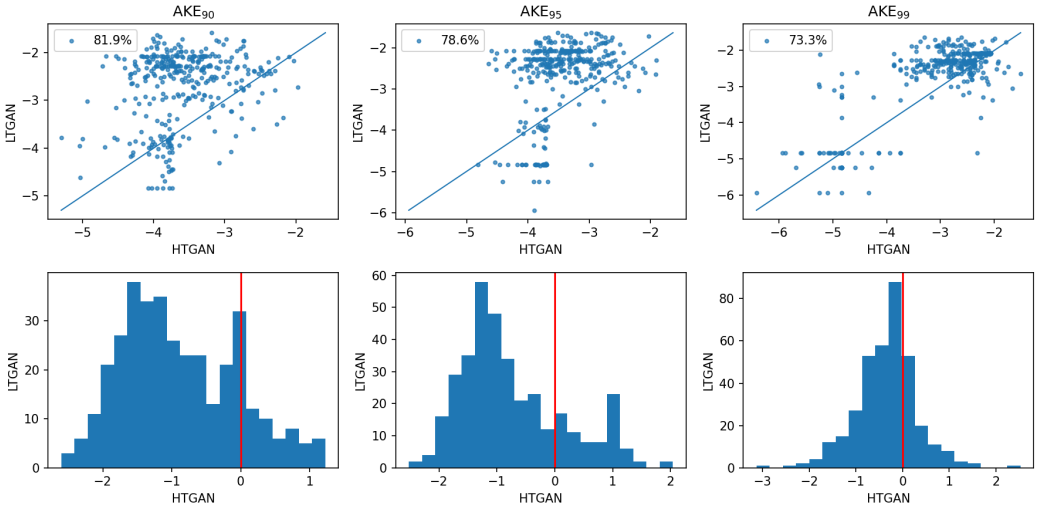


(c) $\alpha = 2.5$

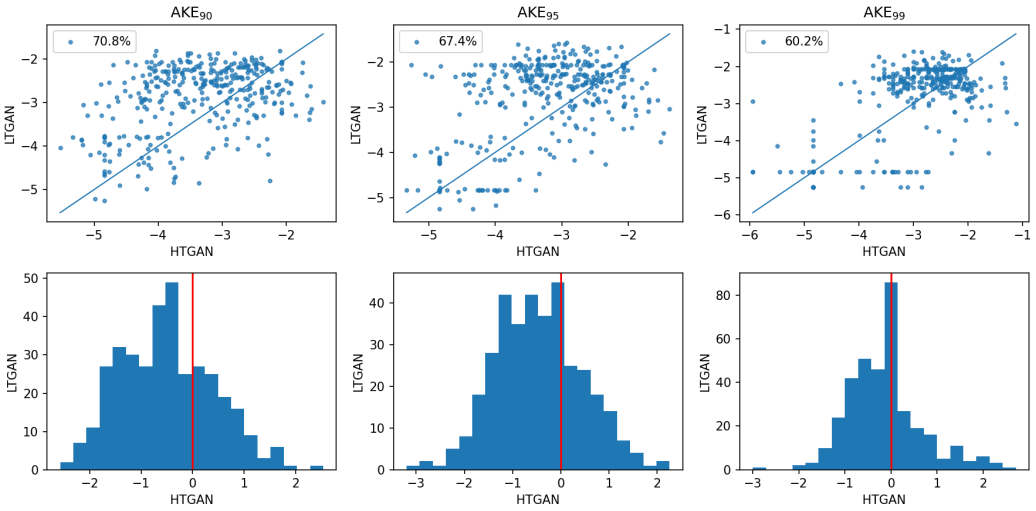
Figure 1.9: Illustration in a d-dimensional setting on simulated data of size $n_{\text{data}} = 10,000$ from a Hüsler-ReiSS copula and Pareto margins, $d \in \{2, 5, 10, 20, 50\}$. Results obtained with the SWD metric. See Figure 1.3 for further details.



(a) $\alpha = 1.5$



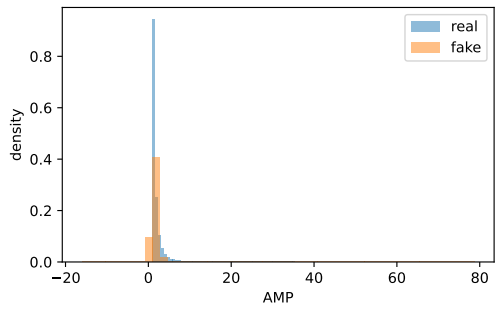
(b) $\alpha = 2.0$



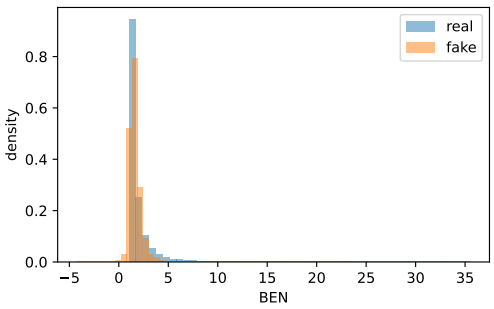
(c) $\alpha = 2.5$

Figure 1.10: Illustration in a d-dimensional setting on simulated data of size $n_{\text{data}} = 10,000$ from a Hüsler-ReiSS copula and Pareto margins, $d \in \{2, 5, 10, 20, 50\}$. Results obtained with the AKE metric. See Figure 1.3 for further details.

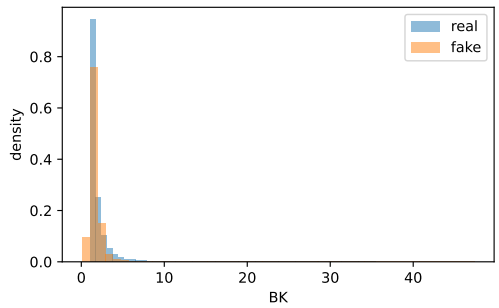
distributions. These illustrations highlight HTGAN's capacity to capture various and complex asymptotic dependence structures - even if not perfectly - despite the challenges posed by the high-dimensional setting and the limited sample size.



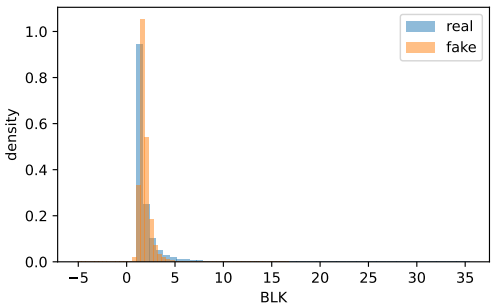
(a)



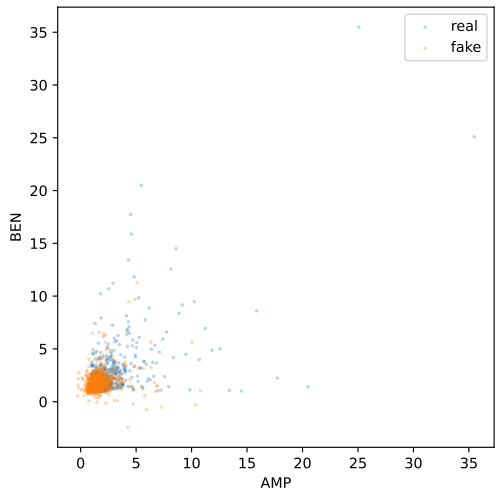
(b)



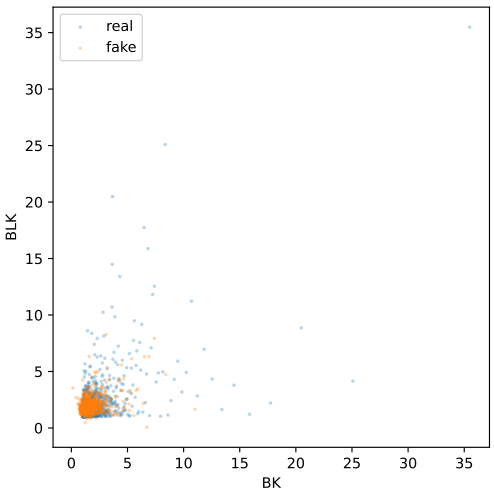
(c)



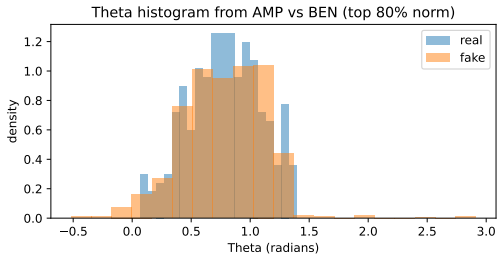
(d)



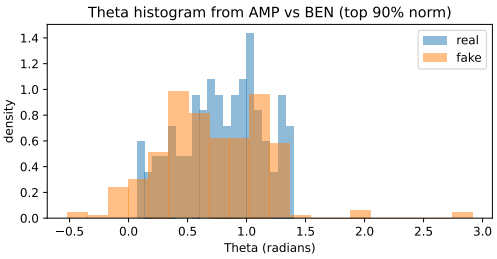
(e)



(f)



(g)



(h)

Figure 1.11: (a)-(d): histograms of real vs synthetic marginals across dimensions one to four. (e)-(f): Scatter plot AMF vs BEN and BK vs BLK tickers. (g)-(h): Histogram of θ after selecting largest values, at level 80% and 90%.

Chapter 2

Generative Neural Order Statistics

This chapter develops the theory and algorithms for generating order statistics using neural networks. We analyze the representational capabilities of neural network-based generative models for sorted data, deriving approximation bounds based on the Sukhatme and Schucany representations.

2.1 Introduction

Sustainable investing. Sustainable finance refers to the process of taking environmental, social and governance (ESG) considerations into account when making investment decisions in the financial sector, leading to more long-term investments in sustainable economic activities and projects, as defined by the European Commission (2024a). Within sustainable finance, a particular quantitative framework, impact investing, is broadly defined as investments that consider not only financial objectives but also other goals such as supporting certain social priorities and agendas (Lo and R. Zhang 2021). Impact investing seeks to optimize both financial objectives and metrics that quantify the compliance level of assets with sustainable goals. For example, ratings on ESG factors provide information about the sustainability performance of a company or a financial instrument, by assessing its exposure to sustainability risks and/or its impact on people and the environment (European Commission 2024b).

Impact investing, order statistics and induced order statistics. Recent developments in Mathematical finance and economics have tackled the challenging task of translating sustainable investing into a mathematical framework. Among those, the work in (Lo and R. Zhang 2021) introduces a general framework for quantifying the impact of impact investing. Namely, authors propose to formalize impact investing as the sorting and selection of an investment universe of N securities based on an *impact factor*, typically the ESG score, $\boldsymbol{X} = (X_1, \ldots, X_N)^T$ and a *performance metric*, typically the return associated

with an asset, $\boldsymbol{\Theta} = (\Theta_1, \ldots, \Theta_N)$. In this framework, $\boldsymbol{X}$ and $\boldsymbol{\Theta}$ are time series evolving in time. It is a common assumption to suppose that both the performance metric and the impact factor are identically distributed throughout time and we will hold this assumption in this work. In this case, the training data consists of the sorted realizations of the time series. Suppose that $\boldsymbol{X}$ varies with time, denoted $\boldsymbol{X}_t$. Under the assumption of identical distributions over time, we drop the time index and treat each observation as a vector. Typically, we are interested in time scales where there are significant changes in the values taken by the impact factor, i.e., periods corresponding to months or years. This excludes high-frequency trading. In this framework, we rank assets by the impact factor and study the associated returns ordered accordingly. This operation fits into the framework of *order statistics* and *induced order statistics*. Formally: suppose that we have a dataset consisting of i.i.d.realizations of the investment universe $\{(X_1, \Theta_1), \ldots, (X_N, \Theta_N)\}$, the order statistics of the impact factor $\boldsymbol{X}$ is defined as a random permutation σ of its elements satisfying:

$$X_{\sigma(1)} \leqslant \ldots \leqslant X_{\sigma(N)}. \tag{2.1}$$

Recall that throughout the paper, we suppose that the realizations $X_1, \ldots, X_N$ are i.i.d.. Note that if the distribution of $\boldsymbol{X}$ is continuous, such a permutation is unique *almost surely*. Also note that the permutation is random and that each instance depends on the realization $X_1, \ldots, X_N$. We therefore denote, for any $i \in [N]$, $X_{i:N} = X_{\sigma(i)}$, the i^{th} *order statistic*. Furthermore, note the marginal law of $X_{k:N}$ is different of the marginal law of X_k for each k. The associated *induced order statistics* (IOS) are defined as:

$$\forall i \in [N] : \ \Theta_{[i:N]} = \Theta_{\sigma(i)}. \tag{2.2}$$

An *impact portfolio strategy*, as defined in (Lo and R. Zhang 2021), is a strategy that considers both the performance metric $\boldsymbol{\Theta}$ and the impact factor $\boldsymbol{X}$. Typically, an impact driven investor is willing to invest in stocks that have high scores for the impact factor (high values of $\boldsymbol{X}$) but also have good values for the performance metric Θ. In this view, it is crucial to perform accurate estimation of the joint distribution of order statistics and the associated induced order statistics counterparts. Modeling order statistics and induced order statistics is an rich, ongoing research field. A thorough treatment of theory of order statistics is given in (H. David and H. Nagaraja 2004). The work (Lo and R. Zhang 2021) proposes a model-based assumption of the impact factor and the performance metric. This approach, combined with a representation theorem of the data, allows for powerful representation theorems for the new portfolio.

Machine Learning and Generative Modeling. Financial datasets are typically scarce and very high dimensional. In a typical setting, $\boldsymbol{X}$ and $\boldsymbol{\Theta}$ are observed over a period of 30 years. If the case where daily data are observable, it amounts to roughly around 30 days $\times$ 12 months $\times$ 30 years $\approx 10,000$ data pointsdata points. Note that it usually makes sense to study data at the granularity of a day for the performance metric (for

example, considering daily stocks returns), but not necessarily for the impact factor. For example, the scale of change for the ESG score is of month or even years. On the other side, the data is of size N, where N corresponds to the size of the investment universe. In the case of the $S\&P500$ index, the size amounts to roughly 500. Such characteristics make it difficult to build predictive models or perform statistical estimation without a calibration model. To cope with this difficulty, a line of research has focused on augmenting the data available with accurate simulations. Historically, for financial applications, approaches for data augmentation include bootstrap methods, Monte Carlo simulations and parametric models. In portfolio management, Monte Carlo methods are widely used by practitioners (Glasserman 2013). Generally speaking, the simulation approach for data augmentation can be framed as mapping a source of randomness to the data of interest. In our impact investing framework, the problem is defined as finding a generator function $\mathcal{G}$ associated to a source of randomness $\boldsymbol{Z}$ such that:

$$\mathcal{G}(\boldsymbol{Z}) \overset{d}{=} \left(\left(X_{1:N}, \Theta_{[1:N]}\right), \ldots, \left(X_{N:N}, \Theta_{[N:N]}\right)\right) \tag{2.3}$$

where $\overset{d}{=}$ stands for equality in distribution. The problem of the existence of tractable models achieving exactly (2.3) is an open question. (Lo and R. Zhang 2021, Theorem 1) provides a representation theorem that maps a source of randomness to the data distribution.

Problem statement. It is not known how real (understand finite, approximate) machine learning models are able to correctly approximate the link function Φ. Our analysis is therefore oriented towards finding non-asymptotic guarantees on the quality of approximation of Φ. We build on work from the theory of function approximation with neural network (Cybenko 1989; Hornik et al. 1989; Yarotsky 2017) and properties such as the Markovian property (H. David and H. Nagaraja 2004, Chapter 2, Section 5) of order statistics and induced order statistics. Moreover, the problem of modeling impact portfolios lies in its high dimensional nature. Typically, the number of asset in the universe is large (eg: $N = 500$ for the S&P 500 index and $N = 1,000$ for the Russell 1000 index) and we have only few data at disposition. Typically, 20 to 30 years of data are available. Considering monthly datasets, that yields a dataset with less than 400 data points. It is neither theoretically nor practically feasible to learn a generative model with so few data points. However, in view of impact investing, we are solely interested in modeling assets with best performing performance metrics.

We are focused on modeling best performing assets in regard of the performance metric, in descending order:

$$\mathcal{G}(\boldsymbol{Z}) \overset{d}{=} \left(\left(X_{N:N}, \Theta_{[N:N]}\right), \ldots, \left(X_{N-d+1:N}, \Theta_{[N-K:N]}\right)\right), \tag{2.4}$$

where typically the dimension of datapoints $d = \lceil \eta N \rceil$, η is a thresholding value, e.g. $5\% = 0.05$.

3.A Notations

We summarize the main notations used throughout the paper.

- **Indices and sets:** For an integer $N \geq 1$, we write $[N] \stackrel{\text{def}}{=} \{1, 2, \dots, N\}$. The symbol $k \in [N]$ denotes an index.
- **Scalars, vectors, matrices:** Lowercase letters $(x, y, u, \dots)$ denote scalars. Bold $(\boldsymbol{x}, \boldsymbol{X}, \mathbf{y}, \boldsymbol{Y}, \dots)$ denote column vectors. The i-th component of $\boldsymbol{x}$ is x_i.
- **Probability:** $\mathbb{P}$ and $\mathbb{E}$ denote probability and expectation. Equality in distribution is written $X \stackrel{d}{=} Y$. The indicator of an event A is $\mathbb{I}\{A\}$. The cumulative distribution of X is commonly written F_X, except if it is explicitly stated otherwise.
- **Spaces and functions:** $\mathbb{R}$ is the set of real numbers, $\mathbb{N}$ the set of positive integers. For a cumulative distribution function (cdf) F, F^{-1} denotes the (left-continuous) quantile function. We use $\ln(\cdot)$ for the natural logarithm.
- **Order statistics and induced order statistics (OS/IOS):** For i.i.d.$X_1, \dots, X_N \in \mathbb{R}$ with cdf F_X, the order statistics are $X_{1:N} \leq \cdots \leq X_{N:N}$, where $X_{k:N}$ is the k-th order statistic. For i.i.d.pairs (X_i, Θ_i), the concomitant of $X_{r:N}$ is $\Theta_{[r:N]}$, *i.e.*, the Θ attached to the observation whose X-value ranks rth; see Definition 2.1.
- **Uniforms and exponentials:** U denotes a standard uniform random variable on $(0, 1)$. $\mathcal{E}(1)$ denotes the unit exponential law with density $f(z) = e^{-z}\,\mathbb{I}\{z \geq 0\}$. We write $Z \sim \mathcal{E}(1)$ and use $F(z) = 1 - e^{-z}$, $F^{-1}(u) = -\ln(1-u)$.
- **Neural networks:** G_{NN} denotes a neural network generator mapping i.i.d.inputs (e.g., uniforms) to outputs. When a family of networks is used, superscripts indicate roles, e.g., F_{NN} and F^{-1}_{NN} for approximations of F and F^{-1}.
- **Asymptotics:** $a_N \sim b_N$ means $\lim_{N\to\infty} a_N/b_N = 1$. Big-$\mathcal{O}$ notation is standard.

2.2 A theoretical analysis of generative modeling of order statistics with Neural Networks

Throughout this section, we let N denote the dimension of the vector to be sorted for obtaining order statistics. For $k \in [N]$, the k^{th} order statistic is denoted by $X_{k:N}$. We leverage these representations to analyze the capacity of neural networks to model order statistics (OS) and induced order statistics (IOS). Formally, we seek a neural network G_{NN} such that

$$G_{NN}(U_1, \dots, U_N) \stackrel{d}{=} (U_{1:N}, \dots, U_{N:N}), \tag{2.5}$$

where $U_1, \dots, U_N$ are independent standard uniform random variables. To address this, we consider several possible constructions. We first recall the following result:

Proposition 2.1 (H. David and H. Nagaraja (2004), eq. (2.3.7), p. 15). Let $X \in \mathbb{R}$ be a random variable with arbitrary distribution function F_X. Let U be a standard uniform random variable. Then:

$$(X_{1:N}, \ldots, X_{N:N}) \stackrel{d}{=} \left(F_X^{-1}(U_{1:N}), \ldots, F_X^{-1}(U_{N:N})\right), \tag{2.6}$$

This proposition shows that, up to transformation by the inverse cdf, the joint distribution of order statistics from any i.i.d. sample is determined by the order statistics of the uniform distribution. Therefore, in this section, we focus solely on approximation results for the order statistics of the uniform distribution.

3.A Representational results for order statistics

We summarize two key sampling schemes for order statistics, with detailed proofs provided in Section 2.B.

Schucany's scheme. Order statistics exhibit a Markov property. Let $U_1, \ldots, U_N$ be i.i.d. standard uniform random variables. Following (Schucany 1972), the order statistics of the uniform distribution can be recursively represented as

$$\begin{cases} U_{N:N} = U_1^{1/N}, \\ U_{N-r:N} = U_{N-r+1:N} \cdot U_r^{1/(N-r)} \quad \forall r \in \{1, \ldots, N\}. \end{cases} \tag{2.7}$$

Sukhatme's representation. Another important representation, due to (Sukhatme 1937), also exhibits a Markov chain-like structure. Let $Z_1, \ldots, Z_N$ be i.i.d. standard exponential random variables, with density

$$f(z) = e^{-z}, \quad z \geqslant 0. \tag{2.8}$$

Let $F(z) = 1 - e^{-z}$ denote the associated cdf. Then, for all $r = 1, \ldots, N$,

$$Z_{r:N} \stackrel{d}{=} \sum_{i=1}^{r} (Z_{i:N} - Z_{i-1:N}) = \sum_{i=1}^{r} \frac{Y_i}{N - i + 1}, \tag{2.9}$$

where $Y_1, \ldots, Y_N$ are i.i.d. standard exponential random variables and $Z_{0:N} = 0$ by convention. Applying Proposition 2.1, and noting that $F^{-1}(u) = -\ln(1-u)$ for $u \in (0,1)$, we have

$$U_{r:N} \stackrel{d}{=} F(Z_{r:N}) = 1 - e^{-Z_{r:N}}. \tag{2.10}$$

3.B Generating order statistics: approximation results

To analyze how well neural networks can approximate the mapping from independent uniforms to order statistics, we establish bounds for the uniform case.

Learning to generate order statistics based on Schucany's scheme. We investigate the approximation quality of neural network-based generative schemes for uniform order statistics, using Schucany's recursive representation. The algorithm iteratively applies (2.7), with neural networks $\hat{G}_{r:N}^{NN}$ approximating the functions

$$G_{m:N}(u_1, u_2) = u_1 \cdot u_2^{1/m}. \tag{2.11}$$

Thus, (2.7) can be rewritten as

$$U_{N-r:N} = G_{N-r:N}(U_{N-r+1:N}, U_r). \tag{2.12}$$

Algorithm 3: Sampling from order statistics $U_{1:N} \leqslant \ldots \leqslant U_{N:N}$.

Input : Learned functions $\hat{G}_{r:N}^{NN} : [0,1] \to \mathbb{R}$ for $r \in [N]$; N uniform samples $U_1, \ldots, U_N$

Output: $\hat{U}_{1:N}, \ldots, \hat{U}_{N:N}$

Set $\hat{U}_{N:N} = U_N^{1/N}$;

for $r \leftarrow 1$ **to** N **do**

  $\hat{U}_{N-r:N} \leftarrow \hat{G}_{N-r:N}^{NN}(\hat{U}_{N-r+1:N}, U_r)$;

  `// In parallel:` $U_{N-r:N} \leftarrow U_{N-r+1:N} U_r^{1/(N-r)}$.

return $\hat{U}_{r:N}$ *and* $U_{r:N}$, *for* $1 \leqslant r \leqslant N$.

With this procedure, we obtain the following result:

Theorem 2.1. *Let $\varepsilon > 0$. There exists a family of ReLU Neural Networks $G_{N-r:N}^{NN}$ for $1 \leqslant r \leqslant N$ such that, using the sampling procedure of Algorithm 3 and setting*

$$\Delta V_r = |\hat{U}_{N-r:N} - U_{N-r:N}|, \tag{2.13}$$

we have $\mathbb{E}|\Delta V_r| < \varepsilon$, where the total number of neurons in the family $\{G_{N-r:N}^{NN}\}_{1 \leqslant r \leqslant N}$ is

$$\mathcal{C}_{Schucany}(\varepsilon) = \sum_{i=1}^{r} 3 \left(\left\lceil \left(\frac{2r(N-i)}{(N-r)\varepsilon} \right)^{N-i} \right\rceil + 1 \right) + c_1 \ln \left(\frac{8r(N-i)}{(N-r)\varepsilon} \right) + c_2. \tag{2.14}$$

The proof is given in Section 2.E. This result is pessimistic: the error is asymptotically polynomial of degree N in ε^{-1}.

Learning with Sukhatme's representation. Next, we consider the Sukhatme representation for generating order statistics. Let F denote the cdf of the unit exponential distribution, $F(z) = 1 - e^{-z}$ for $z > 0$, and $F^{-1}(u) = -\ln(1-u)$ for $u \in (0,1)$. Let F_{NN} and F_{NN}^{-1} be neural network approximations of F and F^{-1}, respectively. The sampling procedure is detailed in Algorithm 4:

Algorithm 4: Sampling from order statistics $U_{1:N} \leqslant \ldots \leqslant U_{N:N}$ based on (2.9)

Input : Learned function $\hat{G}^{NN}$; N uniform samples $U_1, \ldots, U_N$
Output: $\hat{Z}_{1:N}, \ldots, \hat{Z}_{N:N}$

Set $\hat{U}_{0:N} = U_{0:N} = 0$;
for $r \leftarrow 1$ **to** N **do**
  $\hat{Z}_{r:N} \leftarrow \hat{Z}_{r-1:N} + \frac{F_{NN}^{-1}(U_r)}{N-r+1}$;
  `// In parallel:` $Z_{r:N} \leftarrow Z_{r-1:N} + \frac{F^{-1}(U_r)}{N-r+1}$.
return $\hat{U}_{r:N} = F_{NN}(\hat{Z}_{r:N})$ *and* $U_{r:N} = F(Z_{r:N})$, *for* $r \in \{1, \ldots, N\}$.

Theorem 2.2. *With the sampling strategy of Algorithm 4, there exist ReLU Neural Networks F_{NN} and F_{NN}^{-1} such that the error in approximating the uniform order statistics satisfies, in expectation,*

$$\mathbb{E}|U_{r:N} - \hat{U}_{r:N}| \leqslant \varepsilon \left[1 + 2\sum_{i=1}^{r} \frac{1}{N-i+1}\right], \tag{2.15}$$

with at most

$$\mathcal{C}_{Sukhatme}(\varepsilon) \leqslant c\varepsilon^{-2}(2\ln(1/\varepsilon) + 1) + c\varepsilon^{-1}\ln(1/\varepsilon)(\ln(1/\varepsilon) + \ln(\ln(1/\varepsilon)) + 1) \tag{2.16}$$

neurons involved, for some constant c.

The proof is provided in Section 2.F. For $r = N$ (the worst case), the error term admits the asymptotics, as $N \to \infty$:

$$\varepsilon \left[1 + 2\sum_{i=1}^{N} \frac{1}{N-i+1}\right] \sim \varepsilon \ln(N), \tag{2.17}$$

and the complexity term satisfies the following asymptotic results, as $\varepsilon \to 0$:

$$\mathcal{C}_{\text{Sukhatme}}(\varepsilon) \sim 2c\varepsilon^{-2}\ln(\varepsilon^{-1}). \tag{2.18}$$

This representation is more advantageous than that of Theorem 2.1, as the network complexity scales asymptotically as a polynomial of order 2 in ε^{-1} times a logarithmic term.

Discussion. The two results are not strictly comparable: one yields an error of order ε, while the other yields an error of order $\varepsilon \ln(N)$. However, Algorithm 4 presents a key advantage: it exploits a structural property linking order statistics to linear combinations of i.i.d.exponential variables. As a result, the main complexity in neural network approximation is concentrated in accurately simulating unit exponential random variables from standard uniform inputs. Moreover, it enables computational reuse: we only need to approximate the inverse cdf function F^{-1}, where $Z \sim \mathcal{E}(1)$, rather than approximating N separate functions as required by Algorithm 3.

2.3 Experiments

In this section, we present a suite of synthetic experiments designed to evaluate the performance of generative models in producing samples of order statistics. All results reported here are obtained using synthetic data, allowing us to assess model capabilities under controlled conditions where the ground truth distribution is known. Our experimental framework is specifically tailored to test how well different neural network-based generative models—notably various GAN architectures—can approximate the distributional and structural properties of order statistics.

We systematically compare several GAN variants, including classic GANs, Wasserstein GANs (WGAN), and WGANs with gradient penalty (WGAN-GP). These models are evaluated exclusively on synthetic datasets, which enables precise quantitative measurement of model performance. The data used in our experiments are constructed to exhibit the characteristics of order statistics, as described earlier, providing a rigorous testbed for model assessment.

Our experimental design incorporates a range of evaluation metrics that capture both standard distributional similarity and the sortedness and dependence properties intrinsic to order statistics. This approach allows us to identify model strengths and limitations.

The following subsections detail our experimental protocols, model configurations, and results, focusing on synthetic data benchmarks.

3.A Synthetic data

We begin by recalling a setup for a GAN to learn simple nonlinearities across different synthetic datasets, and then extend this setup to the case of order statistics. This vanilla test case illustrates some design choices for implementation, especially data normalization during training and sampling.

We provide evidence that these models can learn simple multivariate distributions, commonly used for sanity checks in the literature. We then test whether this architecture can learn order statistics. To do so, we quantify performance with respect to two aspects: first, the ability of the model to reproduce the marginal distribution of the components; second, the ability to reproduce dependence among coordinates. A key structural property of order statistics is that they should be ordered. We verify whether the components of the model output are sorted by computing a mean 0/1 value, and then proceed to computation of a softer metric that quantifies the excess.

GANs, variants and hyperparameters

Generative Adversarial Networks, as introduced by I. J. Goodfellow et al. (2014), are trained by playing a minimax game between a generator network $\mathcal{G}$ and a discriminator network $\mathcal{D}$ (dependence on parameters is implicit). Namely, both networks optimize the

common objective:

$$\min_{\mathcal{G}} \max_{\mathcal{D}} V(\mathcal{D}, \mathcal{G}) \quad \text{where} \quad V(\mathcal{D}, \mathcal{G}) = \mathbb{E}_{\boldsymbol{X} \sim p_{\text{data}}} \left[\log \mathcal{D}(\boldsymbol{X})\right] + \mathbb{E}_{\boldsymbol{Z} \sim p_{\boldsymbol{Z}}} \left[\log(1 - \mathcal{D}(\mathcal{G}(\boldsymbol{Z})))\right], \tag{2.19}$$

where p_{data} is the underlying distribution of the data and $p_{\boldsymbol{Z}}$ is the distribution of the latent variable, a distribution that is chosen to be easy to sample from, typically Gaussian or Uniform. Throughout this work, we use a uniformly distributed latent variable. More details on GANs and variants are given in Section 2.G. Regarding architecture, we consider both networks to be simple MLP models.

Hyperparameters. The hyperparameters to define in an implementation include the batch size, the latent dimension, the learning rate, the number of epochs, and of course the architectures of both neural networks (generator and discriminator). Moreover, a common practice is, at each epoch, to train the generator for `g_step` steps and the discriminator for `d_step`. The following choices are common practice: `g_step=1` and `d_steps=5`. We provide more details on hyperparameters in Section 2.H.

How accurate is a GAN at approximating order statistics?

In this section, we seek to find the limitations of the representation power of GANs for modeling order statistics. For this purpose, we choose a classic MLP architecture for the networks, illustrate that it can fit a classical benchmark dataset properly, and then assess visually up to which dimension the model can generate proper order statistics. We also use this toy-study to discuss the special scaling used.

Fitting a benchmark dataset. A simple fitting test can be done on the banana-shaped distribution of (Haario et al. 1999), defined as follow: consider $(x_1, x_2) \sim \mathcal{N}(0, I)$ and:

$$\begin{cases} y_1 = x_1 \\ y_2 = x_2 + b(x_1^2 - 1) \end{cases} \tag{2.20}$$

where $b \in \mathbb{R}$ controls the curvature of the distribution. An illustration of the GAN performance is given in Figure 2.1.

Fitting naive model on order statistics

We now take the model obtained from the benchmarking on the banana-shaped distribution to fit order statistics. We sample *large* order-statistics:

$$(X_{N,N}, \dots, X_{N-d+1:N}). \tag{2.21}$$

Normalization scheme. Per-coordinate min–max normalization, while common in GANs, applies different affine maps to each coordinate; real data then leave the sorted manifold,

Hyperparameter	Value
Latent dimension (ℓ)	5
Output dimension (d)	4
Batch size	256
Learning rate	$1 \cdot 10^{-4}$
Epochs	5,000
Discriminator steps (n_D)	5
Generator steps (n_G)	1
Normalization scale factor	150
Optimizer	Adam
Activation	LeakyReLU

Table 2.1: Hyperparameters used for training our vanilla model.

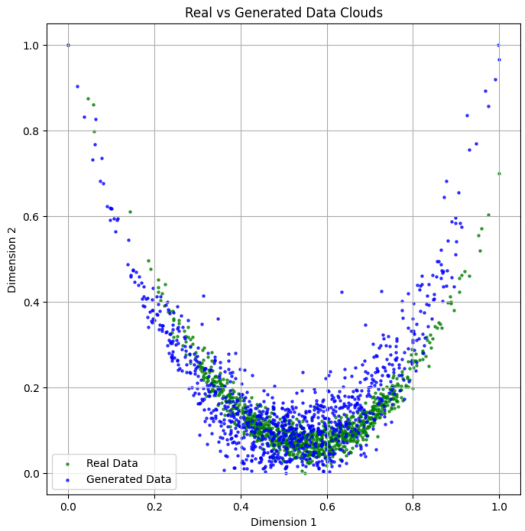


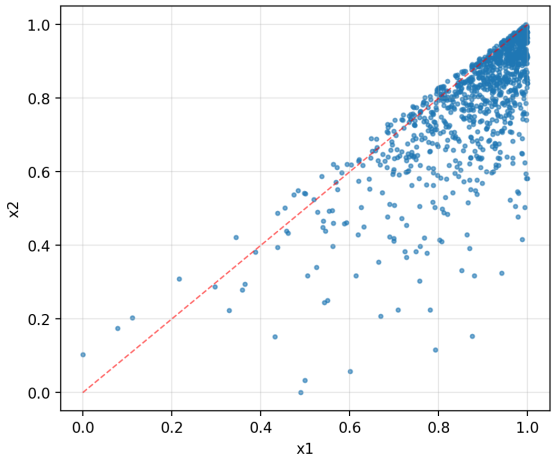
Figure 2.1: Real and generated data after min-max renormalization. `n_samples=1,000`.

and the generator has no incentive to produce ordered outputs. To respect the combinatorial structure of order statistics during learning, we avoid per-coordinate normalizations (e.g., min–max applied separately to each dimension), which apply different affine maps to each coordinate and can flip pairwise differences within a sample. Instead, we use a single *global* affine transform applied jointly to all coordinates: we subtract the global mean computed over the entire training set and rescale by a positive factor $s > 0$,

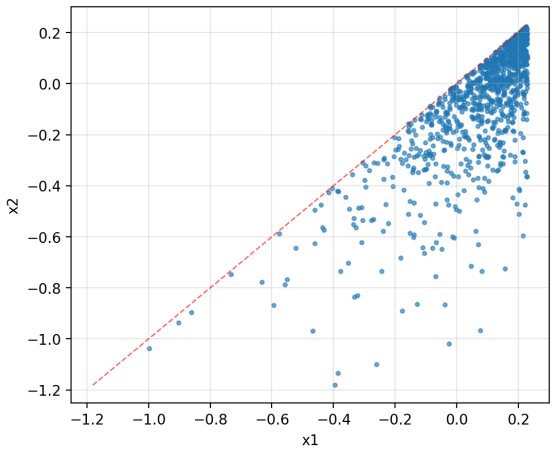
$$\tilde{\boldsymbol{x}} \;=\; s\,(\boldsymbol{x} - \bar{x}), \qquad \bar{x} \overset{\text{def}}{=} \frac{1}{Nd} \sum_{n=1}^{N} \sum_{j=1}^{d} x_j^{(n)}. \tag{2.22}$$

This mean–scale normalization stabilizes optimization *and* exactly preserves the ordering constraint: for any sample $\boldsymbol{x}$ and indices $i < j$, $\operatorname{sign}(\tilde{x}_i - \tilde{x}_j) = \operatorname{sign}(x_i - x_j)$. The scale s is a hyperparameter chosen for numerical convenience and visualization; adding a constant offset after training can further improve plot readability without affecting sortedness. We apply the same transform to real and generated data for a fair comparison. An illustration of the normalization scheme is provided in Figure 2.2. Empirically, the min–max model yields visually plausible marginals, yet many generated points cross the diagonal, signaling broken sortedness. With mean–scale normalization, generated samples concentrate strictly below the diagonal. Structural metrics corroborate this observation: the proportion of sorted samples is close to 1.0. See Figure 2.2 for a side-by-side comparison.

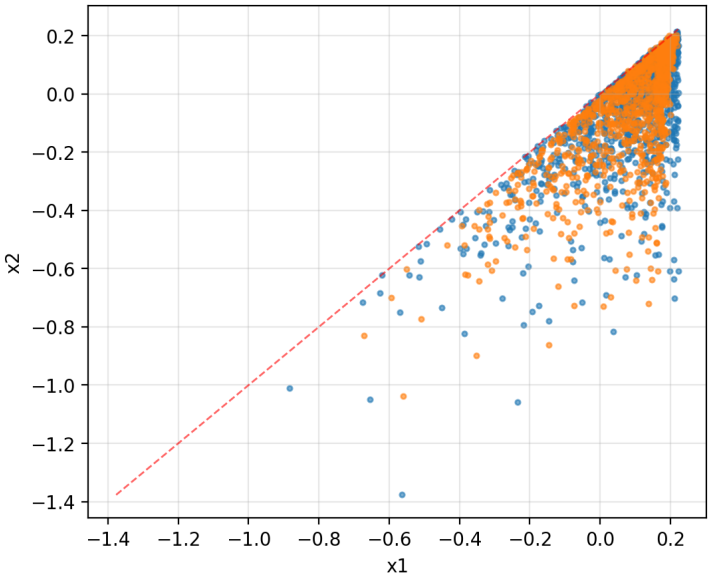
Ablation study: $d = 2$. This case targets the top two order statistics $(X_{N:N}, X_{N-1:N})$ with N=500. An illustration of the generated data cloud is provided in Figure 2.2.



(a) Real data after min-max normalization. `n_samples=1,000`.



(b) Real data after mean–scale normalization. `scale=75` and `n_samples=1,000`.



(c) Overlay of real and generated (mean-scale). 'n_samples=1,000'. Real data are blue dots and generated data are orange dots.

Figure 2.2: Comparison of real and generated 2D order statistics under different normalization schemes, with an additional overlay view. The red dashed lines indicate $x = y$; the mean–scale normalization preserves the sorted structure (no line crossing), while min–max normalization does not.

3.B Benchmarking on synthetic data

In this section, we detail our benchmarking study of GAN architectures to model order statistics. We consider three baseline variants of GANs: vanilla GAN (I. J. Goodfellow et al. 2014), Wasserstein-GAN (WGAN) (Arjovsky et al. 2017), and WGAN with gradient penalty (WGAN-GP) (Wei et al. 2018). We evaluate each of the three models over different dimension settings. Once the best model is selected, we enhance it with an auxiliary regularization term that proxies the quality of the order-statistics property of the produced data points. We then reevaluate the enhanced model against its corresponding base version.

Evaluating the best model out of the three base models

Scope of the study. We explore the performance of each model in different settings. We proceed to replicate the distribution of (2.21), with $N = 1000$ and various values of k. Namely, we first explore dimensions $d = 2, 3, 5, 10, 20$ and 50 for assessing the best model out of the three base models. We choose to use a data set of size $n_{\text{data}} = 32,000$, fixed across dimensions. For each variant of implementation, we perform $R = 10$ replications and compute, for each replicated model, six metrics on generated datasets.

Metrics. Among these six metrics, two are based on a Wasserstein criterion: the Wasserstein 1D (abbr. W1D) and Sliced-Wasserstein Distance (SWD). Two are general dependence metrics: the Spearman's ρ coefficient difference (Spearman) and the Absolute Kendall Error (AKE). Finally, two are specifically designed to assess the order-statistics property of the generated samples, the Sortedness and Softsortedness metric. Precise definitions and background on the metrics are detailed in Section 2.I.

Sweeping with Optuna. For exploring the best hyperparameters, we use Optuna (Akiba et al. 2019), a hyperparameter optimization framework. We use the default TPE sampler, which models the objective function using a tree-structured Parzen estimator. We define a grid of hyperparameters over which we perform optimization. Details on hyperparameters are given in Section 2.H. For each pair of model and dimension, we perform 200 trials. The target metric for TPE sampling is the W1D criteria. We provide both the hyperparamter grid and the hyperparameters selected by the sweep in Section 2.H.

Replications. Once the sweep done, we perform $R = 10$ replications for each pair of model and dimension. We compute each of the six metrics for each of the replications and average over model and dimension. Results are provided in Table 2.2. Best values are in bold.

Results analysis. The results of Table 2.2 provides us the following insights on the performance of models: The WGAN-GP underperforms by a wide margin compared to the two other models. On Wasserstein-based metrics, marginal (W1D) and multivariate (SWD), the WGAN outperforms the GAN, except in the vanilla 2 dimensional case. This is

expected, as the minimization of the Wasserstein distance is embedded in the model's loss. Looking at classical metrics of dependence, namely Spearman and AKE, the GAN seems to outperform the WGAN, except for $d = 3$ for the Spearman metric and $d = 50$ for the AKE metric. However, looking at the specifically designed Sortedness and Softsortedness metrics, GAN performs better in low dimensional settings $d \in \{2, 3, 5, 10\}$, but WGAN is better for higher dimensional settings $d \in \{20, 50\}$. Furthermost, for higher values of d ($d \in \{20, 50\}$), performances of GAN is especially poor (values of Sortedness of 4.7% and 0% respectively), whereas values for WGAN stay higher (51.9% and 50.7% respectively). Qualitatively, the pairwise projections in Figures 2.3 and 2.4 in Section 2.A exhibit tight alignment between real and generated samples across dimensions. For this reason, we interprete WGAN to be more performant at respecting the order statistics structure in higher dimensions. We henceforth choose WGAN as the best performing model and candidate for enhancement in the latter.

dim	GAN	WGAN	WGAN_OSP
2	0.341	**0.328**	0.486
5	**0.22**	0.424	0.251
10	0.513	**0.363**	0.488
20	1.1	**0.55**	0.839
50	2.33	1.13	**1.1**

(a) W1D

dim	GAN	WGAN	WGAN_OSP
2	0.336	**0.332**	0.475
5	**0.204**	0.396	0.251
10	0.473	**0.356**	0.478
20	1.03	**0.526**	0.815
50	2.16	1.1	**1.02**

(b) SWD

dim	GAN	WGAN	WGAN_OSP
2	1.53	1.18	**1.17**
5	**0.315**	0.506	0.896
10	**0.452**	0.989	0.93
20	**0.391**	0.935	1.33
50	**0.713**	2.02	1.65

(c) Spearman

dim	GAN	WGAN	WGAN_OSP
2	94.8%	88.7%	**98.3%**
5	**89.2%**	76.6%	80.3%
10	56.3%	56.2%	**99.4%**
20	4.0%	60.0%	**93.5%**
50	0.0%	**88.3%**	74.9%

(d) Sortedness

dim	GAN	WGAN	WGAN_OSP
2	**1.79**	1.84	1.92
5	**2.64**	2.71	2.67
10	3.59	**3.56**	3.66
20	4.95	5.05	**4.93**
50	**7.94**	8.08	7.95

(e) AKE

dim	GAN	WGAN	WGAN_OSP
2	0.128	0.364	**0.019**
5	**0.033**	0.19	0.193
10	0.263	0.28	**0.000692**
20	1.04	0.105	**0.0102**
50	3.21	**0.0807**	0.0832

(f) Softsortedness

Table 2.2: Summary of metric performances for various models across dimensions.

Order-statistics penalization

In this section, we enforce WGAN to perform better quality samples with an auxilary regularization term, favoring order statistics samples.

Order statistics penalization (OSP). We define the following penalization term:

$$\mathcal{L}_{\mathrm{OSP}}(G) = \mathbb{E}_{z\sim p(z)}\left[\frac{1}{d-1}\sum_{j=1}^{d-1}\left(\max\left\{G(z)_j - G(z)_{j+1} + \varepsilon,\, 0\right\}\right)^2\right] \tag{2.23}$$

where $G(z) \in \mathbb{R}^d$ is the generator output, $\varepsilon > 0$ is a small offset constant, and the sum is over consecutive coordinate pairs. We include this loss in the loss of the generator during traininer. We name this new version WGAN_OSP.

Results. We use the same settings as in the previous section, with the same hyperparameter grid, and we perform a sweep with Optuna to find the best hyperparameters for WGAN_OSP. Resuts for WGAN_OSP are given along other models in Table 2.2. We observe that in regard of the Softsortedness metric, WGAN_OSP outperforms WGAN in three out of the five dimensional settings. This results also in better performance for the Sortedness metric for four out of the five dimensional settings. Regarding W1D and SWD distances, WGAN outperforms for three dimensions, which is expected, as WGAN is only optimizing for the Wasserstein distance. Regarding the classical dependence metrics, namely Spearman's ρ and Kendall's τ, WGAN is consistenly better than WGAN_OSP, but GAN is the overall best model with respectively four and three dimensions in which it is best performing. Overall, this study shows that adding a Softsortedness regularization term to WGAN improves its performance on metrics related to actually producing order statistics, while still maintaining competitive results across the board.

2.4 Conclusion

This work develops a route to generative modeling of ordered statistics. Starting from classical probability identities, we showed how two representations of order statistics (Schucany's top–down recursion and Sukhatme's exponential spacings) can be leveraged to design neural samplers with explicit capacity/error tradeoffs. Concentrating approximation into one scalar inverse–cdf block (with Sukhatme's representation) yields a parameter growth of order $\mathcal{O}(\varepsilon^{-2}\log(\varepsilon^{-1}))$ and a coordinate-wise expected L_1 error that grows only logarithmically with dimension through a harmonic factor. The theory clarifies what must be learned (and what does not) when the target distribution lives on the ordered manifold.

On the practical side, we argued that respecting the geometry of order statistics during training is as important as the network architecture. A simple global mean–scale normalization preserves pairwise order within each sample and proved crucial to maintain the OS structure under adversarial training. Experiments on synthetic OS benchmarks across dimensions demonstrate that standard GAN/WGAN baselines equipped with this normalization achieve competitive performance on classical metrics while improving sortedness; an additional soft order-violation penalty further tightens structure-aware metrics with small impact on Wasserstein distances.

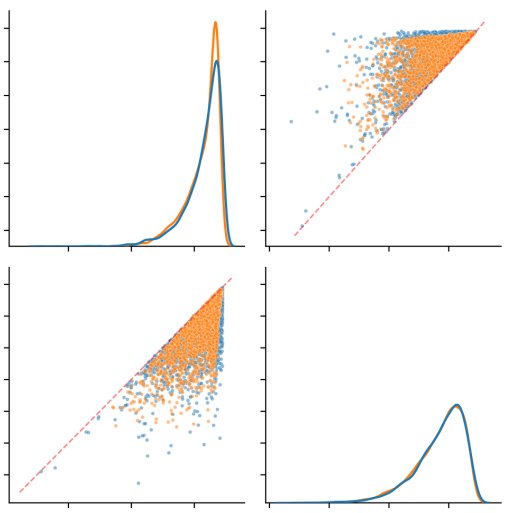
There are several avenues for future work. First, moving from OS to induced order statistics (IOS) is essential for financial applications in which assets are ranked by an

impact factor and paired with returns. This calls for learning generative IOS models with the same geometry-preserving principles. Second, a systematic, controlled empirical study that instantiates architectures mirroring the Schucany and Sukhatme constructions would enable direct validation of the predicted scalings (e.g., $\varepsilon \log N$ error growth and $\varepsilon^{-2} \log(\varepsilon^{-1})$ parameter counts). While feasible, such sweeps are tedious and we leave them to future work. Finally, applying these ideas to real impact-investing datasets will require robust preprocessing, careful evaluation protocols, and domain-specific stress tests in which IOS simulators can meaningfully inform portfolio construction.

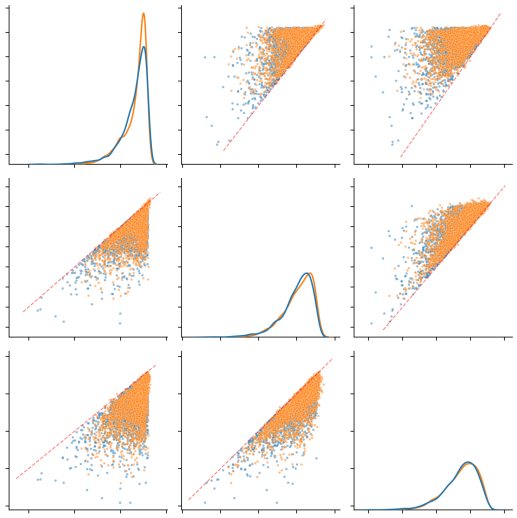
Taken together, our analysis and experiments show that a small amount of structure—codified by probabilistic representations and simple normalization—goes a long way for learning on the ordered manifold. We hope this bridges classical order-statistics theory and modern deep generative modeling, and serves as a foundation for IOS-aware simulators in sustainable finance.

Appendix

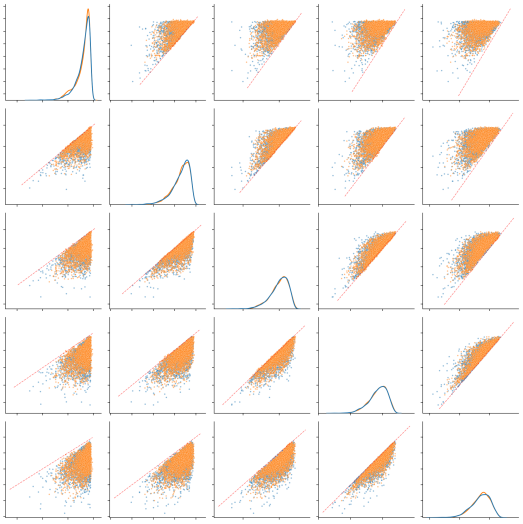
2.A Additional Figures



(a) 2D



(b) 3D



(c) 5D

Figure 2.3: Generated vs. real means for order statistics in various dimensions: (a) 2D, (b) 3D, (c) 5D. Every dimension is scattered against one another. Real data points are in blue and generated data points are in orange.

2.B Some theory of concomitants

Representation results for order statistics

Writing order statistics as min and max. Following the results in (Banner and Ghomrasni 2008), order statistics can be written as a combination of min and max. The following results holds:

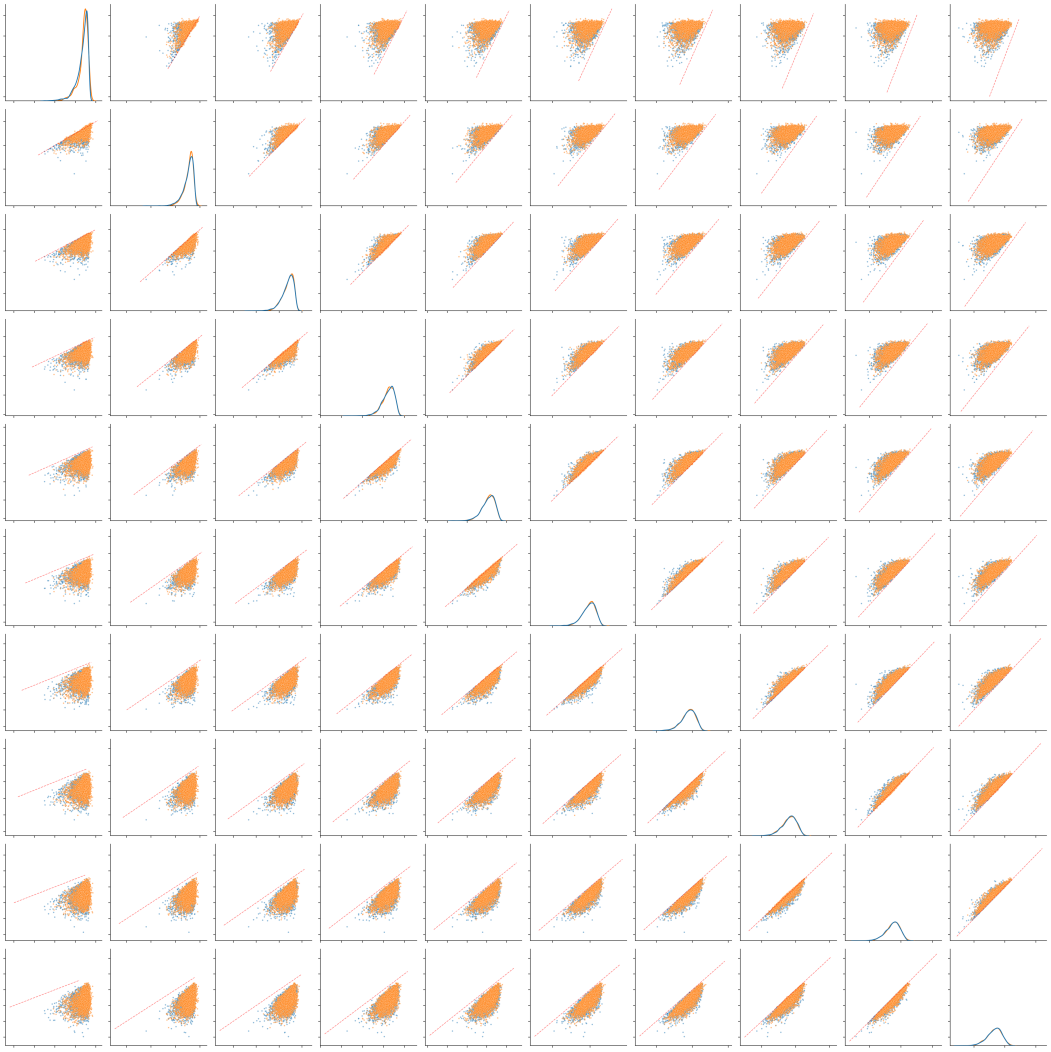


Figure 2.4: Generated vs. real means for order statistics in dimension 10. Every dimension is scattered against one another. Real data points are in blue and generated data points are in orange.

$$X_{k:N} := \max_{1 \leqslant i_1 < \cdots < i_k \leqslant N} \min \{X_{i_1}, \ldots, X_{i_k}\}. \tag{2.24}$$

It is possible to represent min and max with neural network, considering ReLU activation function, that is $\forall x \in \mathbb{R} : \ \sigma(x) = (x)_+ = \max\{0, x\}$, by writing:

$$\max\{a, b\} = a + (b - a)_+, \tag{2.25}$$
$$\min\{a, b\} = -\max\{-a, -b\}. \tag{2.26}$$

Moreover, it is possible to compute min and max over more than two variables by computing recursively:

$$\max\{a_1, \ldots, a_N\} = \max\{\max\{a_1, \ldots, a_{N-1}\}, a_N\}. \tag{2.27}$$

This formula has the benefit, conversely to further representations in this section, of being fully deterministic. However, this comes at a cost: in practice, it is not reasonably feasible to perform computation of (2.24), as it involves, for the full sorting of vector $(X_{1:N}, \ldots, X_{N:N})$, 2^N operations.

Order statistics as a Markov chain (Schucany form). Order statistics admit a Markovian structure (H. David and H. Nagaraja 2004, Chapter 1, Section 5). For the uniform case, the chain is especially simple in the *top–down* direction. Conditionally on $U_{m+1:N} = x$, the distribution of $U_{m:N}$ on $[0, x]$ is the maximum of m i.i.d. uniforms on $[0, x]$, hence

$$\forall y \in [0, x] : \quad F_{U_{m:N} | U_{m+1:N} = x}(y) \ = \ \left(\frac{y}{x}\right)^m, \qquad m = 1, \ldots, N-1. \tag{2.28}$$

Equivalently, if $(U_m)_{m=1}^N$ are i.i.d. $\mathrm{Unif}(0, 1)$ independent of $U_{m+1:N}$, then

$$U_{N:N} \ = \ U_N^{1/N}, \qquad U_{m:N} \ = \ U_{m+1:N}\, U_m^{1/m} \quad (m = N-1, \ldots, 1). \tag{2.29}$$

This is precisely Schucany's recursive sampler for uniform order statistics and matches Equation (2.7) (identify $m = N - r$ and $U_m \equiv U_r$). Defining the bivariate update

$$G_{m:N}(u, w) \stackrel{\text{def}}{=} u\, w^{1/m}, \quad m \in [N], \tag{2.30}$$

the chain reads $U_{m:N} = G_{m:N}(U_{m+1:N}, U_m)$ with $U_{N:N} = G_{N:N}(1, U_N)$, which is the formulation used for our neural approximations in Algorithm 3.

Using standard exponential distributions. Another method for modeling order statistics stems from Sukhatme's representation (H. David and H. Nagaraja 2004, p.17). Let $Z_{1:N} \leqslant \ldots \leqslant Z_{N:N}$ be the order statistics from the standard exponential distribution. Then the joint distribution of the $Z_{i:N}$ is :

$$N! \exp\left(-\sum_{i=1}^{N} z_i\right), \tag{2.31}$$

which may be rewritten as (Sukhatme 1937):

$$N! \exp\left(-\sum_{i=1}^{N} (N-i+1)(z_{i:N} - z_{i-1:N})\right), \tag{2.32}$$

with convention $z_{0:N} = 0$. Making the transformation

$$y_i = (N-i+1)(z_{i:N} - z_{i+1:N}), \tag{2.33}$$

and noting that the range of each y_i is $(0, \infty)$, we see that the Y_i are statistically independent variables, each with pdf (2.8). The relation (2.33) allow $Z_{r:N}$ to be expressed as:

$$Z_{r:N} = \sum_{i=1}^{r} (Z_{i:N} - Z_{i-1:N}) = \sum_{i=1}^{r} \frac{Y_i}{N-i+1}. \tag{2.34}$$

Following the probability integral transformation, we can sample from the standard exponential distribution by sampling from an uniform distribution and applying the inverse cdf $F^{-1} : u \mapsto -\ln(1-u)$:

$$Z_{r:N} = -\ln(1 - U_{r:N}). \tag{2.35}$$

2.C Concomitants of order statistics from a single distribution

Definition

Let $(X_i, Y_i)_{i=1,\dots,n}$ be a random sample from a bivariate distribution with cdf $F(x, y)$. Let us denote

$$X_{r:n} \tag{2.36}$$

the r-th order statistic (where data have ordered increasingly), for any $r = 1, \dots, n$:

$$X_{1:n} < \cdots < X_{r:n} < \cdots < X_{n:n}. \tag{2.37}$$

When the marginal cdf of X is continuous, this ordering exists and is almost surely unique.

Definition 2.1. If $X_{r:n}$ corresponds to the data X_i, its *concomitant* is denoted by

$$Y_{[r:n]}$$

and corresponds to Y_i.

This notion was first introduced of (David, Herbert A 1973) and was called *Induced Order Statistics* (IOS) of (Bhattacharya 1974). See (H. David and H. Nagaraja 2004, Section 6.8) for an introduction.

An important use of concomitants corresponds to select $k < n$ objects Y according to the ranking of their X-values.

Standard properties

When $Y_i = \mathcal{R}(X_i, \varepsilon_i)$ like in a general regression model where X_i and ε_i are independent (see the discussion by H. David and H. Nagaraja (2004, p. 145)), we have

$$Y_{[r:n]} = \mathcal{R}(X_{r:n}, \varepsilon_{[r:n]}). \tag{2.38}$$

When the $(\varepsilon_i)_{1 \leqslant i \leqslant n}$ are i.i.d. then the $(\varepsilon_{[r:n]})_{1 \leqslant r \leqslant n}$ are i.i.d. too, see (Kim and H. David 1990, Section 4).

We recall the definition of associated random variables.

Definition 2.2 (Kim and H. David (1990, Definition 3.1))**.** The random variables $X = (X_i)_{1 \leqslant i \leqslant n}$ are *associated* if:

$$\mathbb{C}ov(f(X), g(X)) \geqslant 0 \tag{2.39}$$

for all non-decreasing (or non-increasing) functions f and g for which the covariance exists.

In particular, if X is square integrable, all pairs (X_i, X_j) are non-negatively correlated.

It turns out that the $(Y_{[r:n]})_{1 \leqslant r \leqslant n}$ in (2.38) are associated under some mild conditions.

Theorem 2.3 (Kim and H. David (1990), definition 4.1)**.** *Assume that $\mathcal{R}$ in (2.38) is a non-decreasing (or non-increasing) function of both arguments, then $(Y_{[r:n]})_{1 \leqslant r \leqslant n}$ are associated.*

Representation theorem under the continuity assumption

Getting the function $\mathcal{R}$ can be achieved through the Rosenblatt transformation (Rosenblatt 1952). (Lo and R. Zhang 2021, Theorem 1) gives an explicit formula for the $\mathcal{R}$. Their representation theorem formula goes beyond (2.38) by representing all the concomitants using order statistics of uniformly distributed random variables.

Theorem 2.4. *If X and Y have continuous cdf F_X and F_Y, with a copula C having a density, we have*

$$(Y_{[1:n]}, \ldots, Y_{[r:n]}, \ldots, Y_{[r:n]}) \stackrel{d}{=} (g(U_{1:n}, V_1), \ldots, g(U_{r:n}, V_r), \ldots, g(U_{n:n}, V_n)) \tag{2.40}$$

where $U_1, \dots, U_n, V_1, \dots, V_n$ are independent random variables uniformly distributed on $[0,1]$. The function g is given explicitly by the formula

$$g(u,v) = F_Y^{-1}\left([\partial_u C(u,\cdot)^{-1}(v)]\right). \tag{2.41}$$

Note that in the above theorem, C is uniquely defined since the marginal cdf are continuous.

Representation theorem without the continuity assumption

We now state a variant avoiding the continuity assumption of the cdf of X and Y. We follow the arguments of (Barrera et al. 2025) to well define the quantity under study.

We denote by $\mathcal{B}_{\mathbb{R}}$ the Borel sigma algebra on $\mathbb{R}$. We fix a conditional distribution function $\mu : \mathcal{B}_{\mathbb{R}} \times \mathcal{B}_{\mathbb{R}} \to [0,1]$ of Y given X (Kallenberg 2002, theorem 5.3, p.84), and we assume that the function $\mathbb{R} \times \mathbb{R} \to \mathbb{R}$ defined by $(x,y) \mapsto \mu(x,(-\infty,y])$ is $(\mathcal{B}_{\mathbb{R}} \otimes \mathcal{B}_{\mathbb{R}})/\mathcal{B}_{\mathbb{R}}$ (i.e. Borel)-measurable. With these conventions, we will use implicitly the corresponding version

$$\mathbb{P}Y \in \cdot | X \overset{\text{def}}{=} \mu(X, \cdot). \tag{2.42}$$

The conditional quantile of Y given $X = x$ is defined by

$$q_{Y|X}(x,v) \overset{\text{def}}{=} \inf\{y : \mu(x,(-\infty,y]) \geqslant v\}, \qquad v \in [0,1]. \tag{2.43}$$

Lemma 2.1. *The function $(x,v) \in \mathbb{R}\times[0,1] \mapsto q_{Y|X}(x,v) \in \mathbb{R}$ is $(\mathcal{B}_{\mathbb{R}} \otimes \mathcal{B}_{[0,1]})$-measurable.*

Proof. Given $t \in \mathbb{R}$, since

$$\{q_{Y|X}(x,v) \leqslant t\} = \{\mu(x,(-\infty,t]) \geqslant v\}, \tag{2.44}$$

we get the result. □

Theorem 2.5. *Denote by F_X and F_Y the respective cdf of X and Y (they can be discontinuous). Then*

$$(Y_{[1:n]}, \dots, Y_{[r:n]}, \dots, Y_{[n:n]}) \tag{2.45}$$

$$\overset{d}{=} (q_{Y|X}(F_X^{-1}(U_{1:n}), V_1), \dots, q_{Y|X}(F_X^{-1}(U_{r:n}), V_r), \dots, q_{Y|X}(F_X^{-1}(U_{n:n}), V_n)) \tag{2.46}$$

where $U_1, \dots, U_n, V_1, \dots, V_n$ are independent random variables uniformly distributed on $[0,1]$.

Proof. The Rosenblatt transformation writes

$$(X_i, Y_i : 1 \leqslant i \leqslant n) \overset{d}{=} (F_X^{-1}(U_i), q_{Y|X}(X_i, V_i) : 1 \leqslant i \leqslant n). \tag{2.47}$$

Because the quantile F_X^{-1} is increasing,

$$(X_{r:n}, Y_{[r,n]} : 1 \leqslant r \leqslant n) \stackrel{d}{=} (F_X^{-1}(U_{r:n}), q_{Y|X}(X_{r:n}, V_{[r,n]}) : 1 \leqslant r \leqslant n) \tag{2.48}$$

$$\stackrel{d}{=} (F_X^{-1}(U_{r:n}), q_{Y|X}(F_X^{-1}(U_{r:n}), V_{[r,n]}) : 1 \leqslant r \leqslant n) \tag{2.49}$$

Since the V_i's are i.i.d. we have

$$(V_{[r,n]} : 1 \leqslant r \leqslant n) \stackrel{d}{=} (V_r : 1 \leqslant r \leqslant n). \tag{2.50}$$

In addition, since they are independent from the U_i's, we get

$$(X_{r:n}, Y_{[r,n]} : 1 \leqslant r \leqslant n) \stackrel{d}{=} (F_X^{-1}(U_{r:n}), q_{Y|X}(F_X^{-1}(U_{r:n}), V_r) : 1 \leqslant r \leqslant n). \tag{2.51}$$

□

The case of independent (X_i, Y_i) but with different cdfs $F_i(x, y)$

In that case, the Rosenblatt transformation write

$$(X_i, Y_i : 1 \leqslant i \leqslant n) \stackrel{d}{=} (F_{X_i}^{-1}(U_i), q_{Y_i|X_i}(X_i, V_i) : 1 \leqslant i \leqslant n). \tag{2.52}$$

The intermediate case of independent (X_i, Y_i) but with same marginal X_i

In that case, the Rosenblatt transformation writes

$$(X_i, Y_i : 1 \leqslant i \leqslant n) \stackrel{d}{=} (F_X^{-1}(U_i), q_{Y_i|X_i}(X_i, V_i) : 1 \leqslant i \leqslant n). \tag{2.53}$$

2.D Some elements on Functional approximation with Neural Network

Section 2.B provides different computational methodologies for obtaining order statistics. A natural question is how well a Neural Network-based method can approximate the obtained functions. For quantifying this, we rely on results from the theory of function approximation with Neural Networks. (Yarotsky 2017) provides several results on how well neural networks are able to approximate smooth functions.

Lemma 2.2 (Yarotsky (2017), Proposition 3)**.** *Given $M > 0$ and $\varepsilon \in (0, 1)$, there exists a ReLU network η with two input units that implements a function $\tilde{\times} : \mathbb{R}^2 \to \mathbb{R}$ so that:*

- *for any input x, y, if $|x| < M$ and $|y| < M$, then $|\tilde{\times}(x, y) - xy| \leqslant \varepsilon$*
- *if $x = 0$ or $y = 0$ then $\tilde{\times}(x, y) = 0$*
- *the depth and the number of weights and computation units in η is no greater than $c_1 \ln(1/\varepsilon) + c_2$ with an absolut constant c_1 and a constant $c_2 = c_2(M)$.*

The second proposition gives a result on approximating Sobolev functions with neural network. For a function with sufficient conditions, the Sobolev norm is defined as:

$$\|f\|_{\mathcal{W}^{n,\infty}\left([0,1]^d\right)} = \max_{\mathbf{n}:|\mathbf{n}|\leqslant n} \operatorname{ess}_{\mathbf{x}\in[0,1]^d} \left|D^{\mathbf{n}} f(\mathbf{x})\right| \tag{2.54}$$

and its associated unit-norm class is given by:

$$F_{n,d} = \left\{ f \in \mathcal{W}^{n,\infty}\left([0,1]^d\right) : \|f\|_{\mathcal{W}^{n,\infty}\left([0,1]^d\right)} \leqslant 1 \right\}. \tag{2.55}$$

The core result of (Yarotsky 2017) is the following:

Theorem 2.6 (Yarotsky (2017), Theorem 1)**.** *For any d, n integers and $\varepsilon \in (0,1)$, there exists a ReLU network architecture that:*

- *is capable of approximating any function from $F_{d,n}$ with error ε,*
- *has depth at most $c\ln(1/\varepsilon + 1)$ and at most $c\varepsilon^{-d/n}(\ln(1/\varepsilon) + 1)$ weights and computation units with some constant $c = c(d,n)$.*

2.E Proofs

Proof of Theorem 2.1

Let's investigate the error induced by the sampling procedure of Algorithm 3. Let $r \in [N]$. We are looking to bound the quantity $\Delta V_r = \hat{U}_{N-r:N} - U_{N-r:N}$.

$$\Delta V_0 = 0, \tag{2.56}$$

$$\Delta V_r = \hat{U}_{N-r:N} - U_{N-r:N} = \hat{G}^{NN}_{N-r:N}\left(\hat{U}_{N-r+1:N}, U_r\right) - G_{N-r:N}\left(U_{N-r+1:N}, U_r\right) \tag{2.57}$$

$$= \left[\hat{G}^{NN}_{N-r:N}\left(\hat{U}_{N-r+1:N}, U_r\right) - G_{N-r:N}\left(\hat{U}_{N-r+1:N}, U_r\right)\right] \tag{2.58}$$

$$+ \left[G_{N-r:N}\left(\hat{U}_{N-r+1:N}, U_r\right) - G_{N-r+1:N}\left(U_{N-r+1:N}, U_r\right)\right]. \tag{2.59}$$

On the one side:

$$G_{N-r:N}\left(\hat{U}_{N-r+1:N}, U_r\right) - G_{N-r:N}\left(U_{N-r+1:N}, U_r\right) \tag{2.60}$$

$$= \left(\hat{U}_{N-r+1:N} - U_{N-r+1:N}\right) U_r^{1/(N-r)} \tag{2.61}$$

$$= \Delta V_{r-1} U_r^{1/(N-r)}. \tag{2.62}$$

On the other:

$$\left|\hat{G}^{NN}_{N-r:N}\big(\hat{U}_{N-r+1:N}, U_r\big) - G_{N-r:N}\big(\hat{U}_{N-r+1:N}, U_r\big)\right| \tag{2.63}$$

$$\leqslant \varepsilon_r := \max_{u_1,u_2\in[0,1]} \left|\hat{G}^{NN}_{N-r:N}\big(u_1,u_2\big) - G_{N-r:N}\big(u_1,u_2\big)\right|. \tag{2.64}$$

Thus for $1 \leqslant r \leqslant N$:

$$|\Delta V_r| \leqslant |\Delta V_{r-1}|\, U_r^{1/(N-r)} + \varepsilon_r. \tag{2.65}$$

Lemma 2.3.

$$\textit{For}\ \ 1 \leqslant r \leqslant N:\ |\Delta V_r| \leqslant \sum_{i=0}^{r} \varepsilon_i \prod_{j=i+1}^{r} U_j^{1/(N-j)}, \tag{2.66}$$

with convention $\varepsilon_0 = 0$.

Proof. The proof goes by induction:

- $r = 0$: (2.65) writes:

$$|\Delta V_1| \leqslant |\Delta V_0|\, U_1^{1/N} + \varepsilon_1 = \varepsilon_1, \tag{2.67}$$

  as $\Delta V_0 = 0$, which is the announced result.

- Suppose that (2.66) holds true for $0 \leqslant r \leqslant N-1$. Then:

$$|\Delta V_{r+1}| \leqslant |\Delta V_r|\, U_{r+1}^{1/(N-r-1)} + \varepsilon_{r+1} \tag{2.68}$$

$$\leqslant \left(\sum_{i=0}^{r} \varepsilon_i \prod_{j=i+1}^{r} U_j^{1/(N-j)}\right) U_{r+1}^{1/(N-r-1)} + \varepsilon_{r+1} \tag{2.69}$$

$$= \sum_{i=0}^{r+1} \varepsilon_i \prod_{j=i+1}^{r+1} U_j^{1/(N-j)}. \tag{2.70}$$

  The result is thus proved.

□

A direct computation yields $\mathbb{E}\left[U^{1/(N-j)}\right] = \frac{N-j}{N-j+1}$ and thus

$$\mathbb{E}\,|\Delta V_r| \leqslant \sum_{i=0}^{r} \varepsilon_i \prod_{j=i+1}^{r} \frac{N-j}{N-j+1} \tag{2.71}$$

$$= \sum_{i=0}^{r} \varepsilon_i \frac{N-r}{N-i}. \tag{2.72}$$

Let's investigate ε_i. We are looking to the best appoximation of the function $G_{N-r:N}(u_1,u_2) = u_1 \cdot u_2^{1/(N-r)}$. On the one side, the function

$$\phi_r : u \in [0,1] \mapsto u^{1/(N-r)} \tag{2.73}$$

is $\zeta_r := 1/(N-r)$ Holder continuous, (the space of Holder continuous functions writes $C^{0,\zeta_r}([0,1])$), with Holder constant 1.

Define the ReLU function as $\sigma^R(x) := \max(0, x)$. Then denote the triangular function by:

$$\widehat{\sigma}^R(x) := \sigma^R(x+1) - 2\sigma^R(x) + \sigma^R(x-1) = \begin{cases} 1+x, & \text{if } -1 \leq x \leq 0, \\ 1-x, & \text{if } 0 \leq x \leq 1, \\ 0, & \text{otherwise.} \end{cases}$$

Note that the triangular function can be approximated by a ReLU Neural Network with three neurons, see (Michaël Allouche, Girard, et al. 2025). We also have the following result:

Lemma 2.4 (Michaël Allouche, Girard, et al. (2025), Lemma 8)**.** *Let $\widehat{\sigma}^R$ be a triangular function, $\tau \in (0,1)$ and $f \in C^{0,\xi}([0,\tau])$ with $\xi \in (0,1]$. For all $M \in \mathbb{N}\setminus\{0\}$, let $h = \tau/M$ and $u_j = jh$ for $j \in \{0,\dots,M\}$. Then,*

$$\sup_{u\in[0,\tau]} \left| f(u) - \sum_{j=0}^{M} f(u_j)\widehat{\sigma}^R\left(\frac{u-u_j}{h}\right)\right| \leqslant H_f \left(\frac{\tau}{M}\right)^{\xi},$$

where H_f is the Hölder constant associated with f. This contruction involves $3(M+1)$ neurons.

This result thus applies to functions ϕ_r, with $\tau = 1$ and we denote the approximation of ϕ_r by $\hat{\phi}_r$. To obtain the minimal number of neuron sufficient for a given precision, let's write the error ε:

$$\varepsilon \leqslant (1/M)^{\zeta_r} \;\Rightarrow\; M \geqslant 1/\varepsilon^{N-r} \;\Rightarrow\; M = \left\lceil 1/\varepsilon^{N-r} \right\rceil. \tag{2.74}$$

Thus a neural network with

$$3(M+1) = 3\left(\left\lceil 1/\varepsilon^{N-r}\right\rceil + 1\right) \tag{2.75}$$

neurons is able to approximate the function ϕ_r with precision ε and a neural network with $\left\lceil (2/\varepsilon)^{N-r}\right\rceil + 1$ neurons is able to approximate the function ϕ_r with precision $\varepsilon/2$. Now, for the multiplicative term, we invoke Lemma 2.2, which writes as follows in this context:

A neural network η_r with $c_1 \ln(8/\varepsilon) + c_2$ weights with an absolute constant c_1 and a constant $c_2 = c_2(M)$, with $M = 1$ here, is able to learn $\tilde{\times}_r(x,v) = x \cdot v$ with precision $\varepsilon/8$.

Now consider

$$\hat{G}^{NN}_{N-r:N}(u_1,u_2) = \tilde{\times}_r(u_1, \phi_r(u_2)). \tag{2.76}$$

Then:

$$\left|\hat{G}^{NN}_{N-r:N}(u_1,u_2) - G_{N-r:N}(u_1,u_2)\right| = \left|u_1 \cdot u_2^{1/(N-r)} - \tilde{\times}_r(u_1,\phi_r(u_2))\right| \tag{2.77}$$

$$\leqslant \left|\tilde{\times}_r(u_1,\phi_r(u_2)) - \tilde{\times}_r(u_1,u_2^{1/(N-r)})\right| \tag{2.78}$$

$$+ \left|\tilde{\times}_r(u_1,u_2^{1/(N-r)}) - u_1 \cdot u_2^{1/(N-r)}\right|. \tag{2.79}$$

The second term is $\leqslant \varepsilon/8$ by construction of $\tilde{\times}_r$. Let's look at the first:

$$\left|\tilde{\times}_r\left(u_1,\phi_r(u_2)\right) - \tilde{\times}_r\left(u_1,u_2^{1/(N-r)}\right)\right| \leqslant \left|\tilde{\times}_r\left(u_1,\phi_r(u_2)\right) - u_1 \cdot \phi_r(u_2)\right| \tag{2.80}$$

$$+ \left|u_1\phi_r(u_2) - u_1 \cdot u_2^{1/(N-r)}\right| \tag{2.81}$$

$$+ \left|u_1 \cdot u_2^{1/(N-r)} - \tilde{\times}_r(u_1,u_2^{1/(N-r)})\right| \tag{2.82}$$

$$\leqslant 3\varepsilon/4, \tag{2.83}$$

where (2.80), (2.82) $\leqslant \varepsilon/8$ by construction of $\tilde{\times}$, (2.81) $\leqslant \varepsilon/2$ by construction of ϕ_r and because $0 \leqslant x \leqslant 1$. Together with (2.78):

$$\forall u_1,u_2 \in [0,1]: \left|\hat{G}^{NN}_{N-r:N}(u_1,u_2) - G_{N-r:N}(u_1,u_2)\right| \leqslant 3\varepsilon/4 + \varepsilon/8 = 7\varepsilon/8 < \varepsilon. \tag{2.84}$$

Lemma 2.5. *Let $\varepsilon > 0$, $N \in \mathbb{N}^*$ and $r \in [N]$. A feedforward ReLU Neural Network $\hat{G}^{NN}_{N-r:N}$ with:*

$$3(\lceil (2/\varepsilon)^{N-r} \rceil + 1) + c_1 \ln(8/\varepsilon) + c_2 \tag{2.85}$$

neurons is able to approximate $G_{N-r:N}$ with precision ε.

Back to (2.72): choosing for $i \in [r], \varepsilon > 0$, $\varepsilon_i = \frac{(N-r)}{r(N-i)}\varepsilon$, we get $\mathbb{E}|\Delta V_r| \leqslant \varepsilon$ and thus we obtain Theorem 2.1 by summing over i.

2.F Proof of Theorem 2.2

Let's investigate the quality of approximation of the scheme in Algorithm 4. For a starter, let's look into the quality of approximation of the function $F^{-1} : u \mapsto -\ln(1-u)$ with a neural network. For this sake, consider $\varepsilon \in [0,1]$ and introduce the function :

$$\tilde{F}^{-1}_{\varepsilon} : \begin{cases} -\ln(1-u) \text{ if } u \in [0,1-\varepsilon], \\ F^{-1}(1-\varepsilon) = -\ln(\varepsilon) = \ln(1/\varepsilon) \text{ if } u \in [1-\varepsilon,1]. \end{cases} \tag{2.86}$$

The function $\varepsilon \cdot \tilde{F}^{-1}_{\varepsilon}$ is in $F_{1,1}$. Indeed, $\forall u \in [0,1]$:

$$|\varepsilon\tilde{F}^{-1}_{\varepsilon}(u)| \leqslant \varepsilon\tilde{F}^{-1}_{\varepsilon}(1-\varepsilon) = \varepsilon\ln(1/\varepsilon) \leqslant \varepsilon\left(\frac{1}{\varepsilon} - 1\right) = 1-\varepsilon, \tag{2.87}$$

$$|(\varepsilon\tilde{F}^{-1}_{\varepsilon})'(u)| = |\varepsilon\tilde{F}'_{\varepsilon}(u)| I\{u \in [0,1-\varepsilon]\} = \varepsilon/(1-u) I\{u \in [0,1-\varepsilon]\} \leqslant 1. \tag{2.88}$$

Therefore, the assumptions of Theorem 2.6 are met. We apply it to $\varepsilon\tilde{F}_{\varepsilon}^{-1}$ with precision ε^2, which writes:

There exists a ReLU network η_{NN} that:

- is capable of approximating $\varepsilon\tilde{F}_{\epsilon}$ with error ε^2, *i.e.*:

$$\max_{u\in[0,1-\varepsilon]} \left|\varepsilon\tilde{F}_{\varepsilon}^{-1}(u) - \eta_{NN}\right| \leqslant \varepsilon^2 \tag{2.89}$$

- has depth at most $c\left(\ln(1/\varepsilon^2+1)\right)$ and at most

$$c\left(\varepsilon^2\right)^{-1}\left(\ln(1/\varepsilon^2)+1\right) = c\varepsilon^{-2}(2\ln(1/\varepsilon)+1) \tag{2.90}$$

  weights and computation units with some constant $c = c(1,1)$.

Thus, multiplying the obtained neural network by $1/\varepsilon$, we get that:

$$\max_{u\in[0,1-\varepsilon]} \left|\tilde{F}_{\varepsilon}^{-1}(u) - 1/\varepsilon\cdot\eta_{NN}(u)\right| \leqslant \varepsilon \tag{2.91}$$

And thus the following result holds:

There exists a ReLU network architecture η_{NN} (we reuse the notation η_{NN}) such that:

- is capable of approximating $\tilde{F}_{\varepsilon}^{-1}$ with error ε,
- has depth at most $c(\ln(1/\varepsilon^2)+1)$ and at most

$$c\left(\varepsilon^2\right)^{-1}\left(\ln(1/\varepsilon^2)+1\right) = c\varepsilon^{-2}(2\ln(1/\varepsilon)+1) \tag{2.92}$$

  weights and computation units with some constant $c = c(1,1)$.

Note that the Neural Network η_{NN} is defined on $[0, 1-\varepsilon]$. We artificially set its value on the interval $[1-\varepsilon, 1]$ to $\eta_{NN}(u) = F^{-1}(1-\varepsilon) = -\ln(\varepsilon)$. Now consider a standard uniform variable U. We investigate the mean error between $Z = F^{-1}(U)$ (which is $\mathcal{E}(1)$) and its approximation $\hat{Z} = \eta_{NN}(U)$. We have that:

$$\mathbb{E}_U\left|Z-\hat{Z}\right| = \int_0^1 \left|F_Z^{-1}(u) - \eta_{NN}(u)\right| du \tag{2.93}$$

$$= \int_0^{1-\varepsilon} \left|F_Z^{-1}(u) - \eta_{NN}(u)\right| du + \int_{1-\varepsilon}^1 \left|F_Z^{-1}(u) - \eta_{NN}(u)\right| du. \tag{2.94}$$

First the LHS of (2.94):

$$\int_0^{1-\varepsilon} \left|F_Z^{-1}(u) - \eta_{NN}(u)\right| du \leqslant \int_0^{1-\varepsilon} \left\|F_Z^{-1} - \eta_{NN}\right\|_\infty du \leqslant \varepsilon, \tag{2.95}$$

by construction of η_{NN} and Theorem 2.6 and second the RHS of (2.94):

$$\int_{1-\varepsilon}^{1} \left|F_Z^{-1}(u) - \eta_{NN}(u)\right| du = \int_{1-\varepsilon}^{1} \left|-\ln(1-u) + \ln(\varepsilon)\right| du \tag{2.96}$$

$$= \int_{1-\varepsilon}^{1} \left[\ln(\varepsilon) - \ln(1-u)\right] du \tag{2.97}$$

$$= \varepsilon \ln(\varepsilon) - \int_{1-\varepsilon}^{1} \ln(1-u)\, du = \varepsilon, \text{ after integration.} \tag{2.98}$$

Together,

$$\mathbb{E}_U \left|Z - \hat{Z}\right| \leqslant 2\varepsilon. \tag{2.99}$$

Therefore,

$$\mathbb{E}\left|Z_{r:N} - \hat{Z}_{r:N}\right| = \mathbb{E}\left|\sum_{i=1}^{r} \frac{Y_i}{N-r+1} - \sum_{i=1}^{r} \frac{\hat{Y}_i}{N-r+1}\right| \tag{2.100}$$

$$\leqslant \sum_{i=1}^{r} \frac{\mathbb{E}|Y_i - \hat{Y}_i|}{N-r+1} \leqslant 2\varepsilon \sum_{i=1}^{r} 1/(N-r+1). \tag{2.101}$$

We now consider approximation of the function F. As for the inverse $\tilde{F}_\varepsilon$, we will construct a suitable function to which we can apply Yarotsky's result. For F, the construction is a bit more subtle: Let $\varepsilon \in [0,1]$. We consider its restriction to F to $[0, \ln(1/\varepsilon)]$, and therefore define the function $\tilde{F}_\varepsilon$ as follows:

$$\text{for } x \geqslant 0: \ \tilde{F}_\varepsilon(x) = \frac{F\left(\ln(1/\varepsilon)x\right)}{\ln\left(1/\varepsilon\right)}. \tag{2.102}$$

The function $\tilde{F}_\varepsilon$ is in $F_{1,1}$. Indeed $\tilde{F}_\varepsilon(x) \in [0,1]$ and $\tilde{F}_\varepsilon(x) = e^{-\ln(1/\varepsilon)x} \in [0,1]$. Therefore, Theorem 2.6 applies to $\tilde{F}_\varepsilon$. We apply it to F_ε with error $\varepsilon/\ln(1/\varepsilon)$, which writes out:
There exists a ReLU Neural Network η_{NN} (we reuse the notation η_{NN}) such that:

- is capable of approximating $\tilde{F}_\varepsilon$ with error $\varepsilon/\ln(1/\varepsilon)$,
- has depth at most $c \ln\left(\frac{\ln(1/\varepsilon)}{\varepsilon} + 1\right)$ and at most

$$c\left(\varepsilon/\ln(\varepsilon)\right)^{-1}\left(\ln\left(\frac{\ln(1/\varepsilon)}{\varepsilon}\right) + 1\right) = c\varepsilon^{-1}\ln(1/\varepsilon)\big(\ln(1/\varepsilon) + \ln(\ln(1/\varepsilon)) + 1\big) \tag{2.103}$$

  weights and computational units, for some constant $c = c(1,1)$.

Finally, doing the converse operation of rescaling on the neural network η_{NN}, that is considering (we reuse the notation η_{NN}):

$$\eta_{NN}(x) = \ln(1/\varepsilon)\eta_{NN}^{\text{old}}\left(\frac{x}{\ln(1/\varepsilon)}\right) \tag{2.104}$$

, we get that, $\forall x > 0$:

$$|F(x) - \ln(1/\varepsilon)\eta_{NN}^{\text{old}}(x/\ln(1/\varepsilon))\,| = |F(x) - \eta_{NN}(x)| \leqslant \varepsilon. \tag{2.105}$$

Now, noting $U_{r:N} = F(Z_{r:N})$ and $\hat{U}_{r:N} = \eta_{NN}(\hat{Z}_{r:N})$:

$$|U_{r:N} - \hat{U}_{r:N}| = |F(Z_{r:N}) - \eta_{NN}(\hat{Z}_{r:N})| \leqslant |\eta_{NN}(\hat{Z}_{r:N}) - F(\hat{Z}_{r:N})| + |F(\hat{Z}_{r:N}) - F\,(Z_{r:N})\,| \tag{2.106}$$

$$\leqslant \varepsilon + |\hat{Z}_{r:N} - Z_{r:N}| \tag{2.107}$$

where the LHS holds in virtue of (2.105) and the RHS because F is a 1-Lipschitz function. Taking expectations on both sides:

$$\mathbb{E}|U_{r:N} - \hat{U}_{r:N}| \leqslant \varepsilon + \mathbb{E}|Z_{r:N} - \hat{Z}_{r:N}| \leqslant \varepsilon + 2\varepsilon \sum_{i=1}^{r} \frac{1}{N-r+1} \tag{2.108}$$

$$= \varepsilon \left[1 + 2\sum_{i=1}^{r} \frac{1}{N-r+1}\right]. \tag{2.109}$$

The total number of neurons involved for the computation of $\hat{U}_{r:N}$ with this precision is :

$$c\varepsilon^{-2}(2\ln(1/\varepsilon) + 1) + c\varepsilon^{-1}\ln(1/\varepsilon)(\ln(1/\varepsilon) + \ln(\ln(1/\varepsilon)) + 1), \tag{2.110}$$

which achieves the proof of Theorem 2.2.

2.G Additional experimental details

More background on GANs

This appendix summarizes the adversarial training objectives used in our experiments and highlights practical details for stability.

GAN objective. Given a data distribution p_{data} and a latent prior $p_{\boldsymbol{Z}}$, the vanilla GAN objective (see Equation (1.12)) defines a two-player minimax game between a discriminator $\mathcal{D}$ and a generator $\mathcal{G}$. In practice, the non-saturating generator loss is often used to obtain stronger gradients early in training:

$$\min_{\mathcal{G}} \; \mathbb{E}_{\boldsymbol{Z}\sim p_{\boldsymbol{Z}}}\big[-\log \mathcal{D}(\mathcal{G}(\boldsymbol{Z}))\big]. \tag{2.111}$$

At equilibrium with idealized capacity, the generator recovers p_{data} and the discriminator outputs $1/2$ (I. J. Goodfellow et al. 2014). Training instabilities and mode collapse are common; Section Section 2.H records our simple, robust choices.

Wasserstein GAN (WGAN). The WGAN replaces the Jensen–Shannon divergence implicit in the vanilla objective with the Wasserstein-1 distance. Using the Kantorovich–Rubinstein duality, the critic $\mathcal{D}$ is constrained to be 1-Lipschitz and optimized via

$$\max_{\|\mathcal{D}\|_{\mathrm{Lip}}\le 1} \mathbb{E}_{\boldsymbol{X}\sim p_{\mathrm{data}}}[\mathcal{D}(\boldsymbol{X})] - \mathbb{E}_{\boldsymbol{Z}\sim p_{\boldsymbol{Z}}}[\mathcal{D}(\mathcal{G}(\boldsymbol{Z}))], \quad \min_{\mathcal{G}} -\mathbb{E}_{\boldsymbol{Z}\sim p_{\boldsymbol{Z}}}[\mathcal{D}(\mathcal{G}(\boldsymbol{Z}))]. \tag{2.112}$$

The resulting gradients are more informative when supports are disjoint, improving convergence (Arjovsky et al. 2017). The original implementation enforced the Lipschitz constraint by weight clipping, which can harm capacity and lead to optimization issues.

WGAN with Gradient Penalty (WGAN-GP). To avoid weight clipping, Wei et al. (2018) propose penalizing the deviation of the gradient norm from 1 along lines interpolating between real and generated samples. Writing $\tilde{\boldsymbol{X}} = \mathcal{G}(\boldsymbol{Z})$, $\hat{\boldsymbol{X}} = \epsilon\,\boldsymbol{X} + (1-\epsilon)\,\tilde{\boldsymbol{X}}$ with $\epsilon \sim \mathcal{U}[0,1]$, the critic maximizes

$$\mathbb{E}_{\boldsymbol{X}}[\mathcal{D}(\boldsymbol{X})] - \mathbb{E}_{\boldsymbol{Z}}[\mathcal{D}(\mathcal{G}(\boldsymbol{Z}))] - \lambda\,\mathbb{E}_{\hat{\boldsymbol{X}}}\Big(\|\nabla_{\hat{\boldsymbol{X}}}\mathcal{D}(\hat{\boldsymbol{X}})\|_2 - 1\Big)^2. \tag{2.113}$$

This penalty enforces the 1-Lipschitz property more effectively and stabilizes training across architectures and datasets.

2.H Hyperparameters

- Update ratio: several critic steps per generator step (e.g., n_D=5, n_G=1). - Optimizers: Adam with small learning rate and $\beta_1 \in [0, 0.5]$ works well for adversarial training. - Regularization: gradient penalty (WGAN-GP) improves stability; spectral normalization is an alternative 1-Lipschitz control. - Activations: LeakyReLU for critic; generator activations as in Section Section 2.H. - Evaluation: rely on distributional metrics (e.g., W1D, SWD) and structure-aware scores (e.g., sortedness) discussed in Section Section 2.I.

See Appendix Section 2.G for a concise primer on GANs, WGAN, and WGAN-GP, including objectives, constraints, and training tips.

2.I Metrics

We evaluate generative fidelity and structure with six metrics. Throughout, let real samples be $\{\boldsymbol{x}^{(n)}\}_{n=1}^N \subset \mathbb{R}^d$ and generated samples $\{\mathbf{y}^{(n)}\}_{n=1}^N \subset \mathbb{R}^d$. Unless stated otherwise, we adopt a *descending* order convention when focusing on the largest order statistics.

Wasserstein 1D (W1D). For each coordinate $j \in [d]$, we compute the empirical 1-Wasserstein distance between the marginal distributions by matching order statistics. Let $x_j^{n:N}$ and $y_j^{n:N}$ denote the n-th order statistics (ascending) of $\{x_j^{(n)}\}_{n=1}^N$ and $\{y_j^{(n)}\}_{n=1}^N$,

respectively. The reported score is the average over coordinates:

$$\widehat{\mathrm{W1D}} \;=\; \frac{1}{Nd}\sum_{j=1}^{d}\sum_{n=1}^{N}\big|x_j^{n:N} - y_j^{n:N}\big|. \tag{2.114}$$

This equals the empirical 1D Wasserstein distance per marginal, averaged across dimensions; it is invariant to permutations within each marginal sample.

Sliced-Wasserstein Distance (SWD). We approximate the 1-Wasserstein distance between the real and generated joint distributions via random 1D projections. Draw unit vectors $\mathbf{v}_\ell \in \mathbb{S}^{d-1}$, $\ell = 1, \ldots, L$, project $u_\ell^{(n)} = \langle \boldsymbol{x}^{(n)}, \mathbf{v}_\ell \rangle$ and $v_\ell^{(n)} = \langle \mathbf{y}^{(n)}, \mathbf{v}_\ell \rangle$, sort each set to obtain order statistics $u_\ell^{n:N}$ and $v_\ell^{n:N}$, and average the 1D distances:

$$\widehat{\mathrm{SWD}} \;=\; \frac{1}{LN}\sum_{\ell=1}^{L}\sum_{n=1}^{N}\big|u_\ell^{n:N} - v_\ell^{n:N}\big|. \tag{2.115}$$

As L increases, this Monte Carlo estimator tightens; SWD is sensitive to global geometry while remaining computationally light.

Pairwise Spearman difference. To assess monotone dependence, we compare pairwise Spearman rank correlations between real and generated data. For each pair (i, j), let $\rho_{ij}^{\mathrm{real}}$ and ρ_{ij}^{gen} denote Spearman's ρ. The metric is the mean absolute discrepancy over all pairs:

$$\text{Spearman diff} \;=\; \frac{2}{d(d-1)}\sum_{1\le i<j\le d}\big|\rho_{ij}^{\mathrm{real}} - \rho_{ij}^{\mathrm{gen}}\big|. \tag{2.116}$$

Being rank-based, it is invariant to strictly monotone marginal transforms.

Sortedness. Order statistics must be sorted. We report the proportion of generated samples that satisfy the coordinate-wise descending order $y_1^{(n)} \ge y_2^{(n)} \ge \cdots \ge y_d^{(n)}$:

$$\text{Sortedness} \;=\; \frac{1}{N}\sum_{n=1}^{N}\mathbb{I}\big\{y_k^{(n)} - y_{k+1}^{(n)} \ge 0,\ \forall k \in [d-1]\big\}. \tag{2.117}$$

This hard score equals 1 only if all generated vectors respect the ordering constraint.

Absolute Kendall Error (AKE). To compare distributions of order statistics, we sort each sample vector in descending order and compute the mean absolute deviation between the resulting order-statistic vectors:

$$\mathrm{AKE} \;=\; \frac{1}{Nd}\sum_{n=1}^{N}\sum_{k=1}^{d}\big|\mathrm{sort}_\downarrow(\boldsymbol{x}^{(n)})_k - \mathrm{sort}_\downarrow(\mathbf{y}^{(n)})_k\big|. \tag{2.118}$$

This captures alignment of empirical order-statistic marginals; lower is better.

Softsortedness. As a differentiable proxy for the sortedness constraint, we average the positive violations of the descending order:

$$\text{SoftSorted} \;=\; \frac{1}{N(d-1)} \sum_{n=1}^{N} \sum_{k=1}^{d-1} \big[\, y_{k+1}^{(n)} - y_{k}^{(n)} \,\big]_{+}, \quad [t]_{+} \stackrel{\text{def}}{=} \max\{0, t\}. \tag{2.119}$$

This penalty is zero if and only if every generated vector is sorted in descending order and increases linearly with the magnitude of violations.

Chapter 3

Gradient-Free Fine-Tuning of Diffusion Models

This chapter presents a gradient-free approach to fine-tuning diffusion models for reward-based generation. Using Fisher's identity, we derive a score estimator that requires only forward evaluations of the reward function, avoiding costly backpropagation through the reward model.

3.1 Introduction

Diffusion models are now a standard tool for generative modelling in images, audio, and related domains (Jascha Sohl-Dickstein et al. 2015; Ho, Jain, et al. 2020; Y. Song, Jascha Sohl-Dickstein, et al. 2021). They learn a time-dependent score (or denoiser) that inverts a simple noising process and produces new samples from noise. Following their success, it has been proposed to fine-tune pretrained diffusion models, taking into account new data preferences and/or constraints by simply shifting the learnt weights accordingly.

To achieve this, ideas from language-model alignment have recently started to be applied to diffusion: Either shifting the pre-trained weights using methods based on reward models (Jiazheng Xu et al. 2023) or reinforcement learning policies (Black et al. 2024; Fan et al. 2023), or adapting the architecture using lightweight adapters such as LoRA (Hu et al. 2021). In addition, diffusion-specific techniques like classifier-free guidance (Ho and Salimans 2022) or improved noise schedulers (Alex Nichol and Dhariwal 2021) provide complementary control mechanisms.

In this chapter we introduce *iterative tilting*, a gradient-free method for reward-based fine-tuning that shifts a diffusion model toward a reward-tilted density without differentiating through the reward function. During the preparation of this work, a concurrent paper was submitted to ICLR (Anonymous 2025). Their method is essentially the same as ours but formulated using the stochastic interpolants framework and developed in continuous time. In contrast, our approach is based on a diffusion framework and is derived in discrete time.

Section 3.2 recalls the theoretical framework of Denoising Diffusion Models (DDM) and reviews existing reward fine-tuning formulations, Section 3.3 develops the Iterative Tilting method from a theoretical standpoint, and Section 1.3 reports controlled experiments on a Gaussian synthetic fine-tuning task, before we conclude on the work of this chapter.

3.2 Preliminaries on Denoising Diffusion Models and Fine-Tuning

3.A Diffusion models

Denoising diffusion models (DDMs) (Jascha Sohl-Dickstein et al. 2015; Y. Song and Ermon 2019; Ho, Jain, et al. 2020) define a generative process for a data distribution p_0 by constructing a continuous probability path $(p_t)_{t\in[0,1]}$ of distributions between p_0 and a tractable reference $p_1 := \mathrm{N}(0, \mathrm{I}_d)$.

Forward noising process

The path is defined via the marginals $p_t = \mathrm{Law}(X_t)$, where

$$X_t = \alpha_t X_0 + \sigma_t X_1, \qquad X_0 \sim p_0, \quad X_1 \sim \mathrm{N}(0, \mathrm{I}_d). \tag{3.1}$$

Here X_0 and X_1 are independent, and $(\alpha_t)_{t\in[0,1]}$ and $(\sigma_t)_{t\in[0,1]}$ are deterministic schedules such that α_t is non-increasing, σ_t is non-decreasing, with boundary conditions $(\alpha_0, \sigma_0) = (1, 0)$ and $(\alpha_1, \sigma_1) = (0, 1)$. The path $(p_t)_{t\in[0,1]}$ thus defines an interpolation that gradually transforms the clean data distribution p_0 into the Gaussian reference p_1.

From (3.1), the conditional distribution of X_t given $X_0 = x_0$, denoted $q_{t|0}$, is Gaussian:

$$q_{t|0}(x_t \mid x_0) = \mathrm{N}(x_t;\, \alpha_t x_0,\, \sigma_t^2 \mathrm{I}_d). \tag{3.2}$$

Regarding the choices of schedules (α_t, σ_t), two standard choices are the *variance-preserving (VP)* schedule with $\alpha_t^2 + \sigma_t^2 = 1$ (Ho, Jain, et al. 2020; Dhariwal and Alexander Nichol 2021), which ensures $\mathrm{Var}(X_t) = \mathrm{Var}(X_0)$ when X_0 has unit variance, and the *linear (flow matching)* schedule with $(\alpha_t, \sigma_t) = (1 - t, t)$ (Lipman et al. 2023; Esser et al. 2024), corresponding to a straight-line interpolation between data and noise.

Reverse process and DDIM transitions

To generate new samples, DDMs simulate a time-reversed Markov chain. Given a increasing sequence $(t_k)_{k=0}^K$ of time steps with $t_K = 1$ and $t_0 = 0$, reverse transitions are iteratively applied to map a sample from $p_{t_{k+1}}$ to one from p_{t_k}, progressively denoising until convergence to p_0.

DDIM framework. The DDIM framework (J. Song et al. 2021) introduces a general family of reverse transitions. It defines a schedule $(\eta_t)_{t\in[0,1]}$ satisfying $\eta_t \leqslant \sigma_t$ for all

$t \in [0,1]$, along with transition densities given for $s < t$ by

$$p^{\eta}_{s|t}(x_s \mid x_t) = \mathbb{E}\left[q^{\eta}_{s|0,1}(x_s \mid X_0, X_1) \,\middle|\, X_t = x_t\right], \tag{3.3}$$

where

$$q^{\eta}_{s|0,1}(x_s \mid x_0, x_1) := \mathrm{N}\left(x_s;\, \alpha_s x_0 + \sqrt{\sigma_s^2 - \eta_s^2}\, x_1,\, \eta_s^2 \mathrm{I}_d\right) \tag{3.4}$$

and the random variables (X_0, X_t, X_1) are defined as in (3.1). By construction, this family satisfies the marginalisation property $p_s(x_s) = \int p^{\eta}_{s|t}(x_s \mid x_t)\, p_t(x_t)\, \mathrm{d}x_t$: by the law of total expectation, the right-hand side equals $\mathbb{E}_{(X_0,X_1)}[q^{\eta}_{s|0,1}(x_s \mid X_0, X_1)]$, and since $X_1 \sim \mathrm{N}(0, \mathrm{I}_d)$ is independent of X_0, sampling from the bridge (3.4) yields $X_s = \alpha_s X_0 + \sqrt{\sigma_s^2 - \eta_s^2}\, X_1 + \eta_s \epsilon$ with $\epsilon \sim \mathrm{N}(0, \mathrm{I}_d)$; the sum $\sqrt{\sigma_s^2 - \eta_s^2}\, X_1 + \eta_s \epsilon$ has variance $\sigma_s^2 \mathrm{I}_d$, so $X_s \overset{d}{=} \alpha_s X_0 + \sigma_s \tilde{X}_1$ with $\tilde{X}_1 \sim \mathrm{N}(0, \mathrm{I}_d)$, matching the definition of p_s. This ensures that $(p^{\eta}_{t_k|t_{k+1}})_{k=0}^{K-1}$ defines a consistent set of reverse transitions.

Denoiser. The transitions (3.3) are intractable and have to be approximated. To this end, it has been proposed to replace X_0 and X_1 by their conditional expectations given X_t. Let $\hat{x}_0^{\theta}(\cdot, t)$ and $\hat{x}_1^{\theta}(\cdot, t)$ denote parametric estimators of

$$\hat{x}_0(x_t, t) := \mathbb{E}[X_0 \mid X_t = x_t], \qquad \hat{x}_1(x_t, t) := \mathbb{E}[X_1 \mid X_t = x_t]. \tag{3.5}$$

From the forward interpolation (3.1), these two quantities are related by $\hat{x}_1(x_t, t) = (x_t - \alpha_t \hat{x}_0(x_t, t))/\sigma_t$, yielding the parametric relationship

$$\hat{x}_1^{\theta}(x_t, t) = \frac{x_t - \alpha_t \hat{x}_0^{\theta}(x_t, t)}{\sigma_t}, \qquad \hat{x}_0^{\theta}(x_t, t) = \frac{x_t - \sigma_t \hat{x}_1^{\theta}(x_t, t)}{\alpha_t}. \tag{3.6}$$

The parametric DDIM transition then becomes

$$p^{\eta,\theta}_{t_k|t_{k+1}}(x_{t_k} \mid x_{t_{k+1}}) := q^{\eta}_{t_k|0,1}\big(x_{t_k} \mid \hat{x}_0^{\theta}(x_{t_{k+1}}, t_{k+1}),\, \hat{x}_1^{\theta}(x_{t_{k+1}}, t_{k+1})\big)\,. \tag{3.7}$$

For $k = 0$, the transition $p^{\eta,\theta}_{0|t_1}(\cdot \mid x_{t_1})$ is simply defined as the Dirac mass at $\hat{x}_0^{\theta}(x_{t_1}, t_1)$.

Sampling procedure. Given timesteps $(t_k)_{k=0}^{K}$ with $t_K = 1$ and $t_0 = 0$, DDIM sampling is summarised in Algorithm 5. Setting $\eta_t = 0$ yields deterministic sampling; setting $\eta_t = \sigma_t \sqrt{1 - \alpha_t^2/\alpha_s^2}$ (for the VP schedule) recovers the stochastic DDPM sampler (Ho, Jain, et al. 2020).

Score function and training

Score and denoiser relationship. The *score* of the noised distribution is $\nabla \log p_t(\cdot)$. For the Gaussian forward kernel (3.2), the score admits a closed-form expression in terms of

Algorithm 5: DDIM Sampling

Require: Denoiser $\hat{x}_1^\theta$, timesteps $(t_k)_{k=0}^K$ with $t_K = 1$, $t_0 = 0$, variance schedule $(\eta_{t_k})_{k=0}^{K-1}$

Sample $x_{t_K} \sim \mathrm{N}(0, \mathrm{I}_d)$ **for** $k = K-1, \dots, 1$ **do**

$\hat{x}_1 \leftarrow \hat{x}_1^\theta(x_{t_{k+1}}, t_{k+1})$ ▷ Predict X_1

$\hat{x}_0 \leftarrow (x_{t_{k+1}} - \sigma_{t_{k+1}} \hat{x}_1)/\alpha_{t_{k+1}}$ ▷ Estimate clean sample

$\mu_{t_k} \leftarrow \alpha_{t_k} \hat{x}_0 + \sqrt{\sigma_{t_k}^2 - \eta_{t_k}^2}\, \hat{x}_1$

Sample $z \sim \mathrm{N}(0, \mathrm{I}_d)$

$x_{t_k} \leftarrow \mu_{t_k} + \eta_{t_k} z$

$x_0 \leftarrow (x_{t_1} - \sigma_{t_1} \hat{x}_1^\theta(x_{t_1}, t_1))/\alpha_{t_1}$ ▷ Final denoising step

return x_0

the denoiser. Exchanging expectation and gradient under standard regularity assumptions:

$$\nabla_{x_t} \log p_t(x_t) = \mathbb{E}\left[\nabla_{x_t} \log q_{t|0}(x_t \mid X_0) \,\middle|\, X_t = x_t\right] = -\frac{\hat{x}_1(x_t, t)}{\sigma_t}. \tag{3.8}$$

Thus, the denoiser $\hat{x}_1^\theta$ directly provides a score estimator $s_\theta(x_t, t) = -\hat{x}_1^\theta(x_t, t)/\sigma_t$, and conversely $\hat{x}_1^\theta(x_t, t) = -\sigma_t s_\theta(x_t, t)$.

Training objective. The denoiser can be trained by regressing either X_0 or X_1 from the noised sample $X_t = \alpha_t X_0 + \sigma_t X_1$. The X_0-prediction loss reads

$$\mathrm{L}_{X_0}(\theta) = \int_0^1 \mathbb{E}_{X_0 \sim p_0, X_1 \sim \mathrm{N}(0, \mathrm{I}_d)} \left\| \hat{x}_0^\theta(\alpha_t X_0 + \sigma_t X_1, t) - X_0 \right\|^2 \mathrm{d}t, \tag{3.9}$$

while the X_1-prediction loss is

$$\mathrm{L}_{X_1}(\theta) = \int_0^1 \mathbb{E}_{X_0 \sim p_0, X_1 \sim \mathrm{N}(0, \mathrm{I}_d)} \left\| \hat{x}_1^\theta(\alpha_t X_0 + \sigma_t X_1, t) - X_1 \right\|^2 \mathrm{d}t. \tag{3.10}$$

Since $\hat{x}_1^\theta = -\sigma_t s_\theta$ by (3.8), the X_1-prediction loss is equivalent to denoising score matching (Hyvärinen 2005; Vincent 2011): minimising (3.10) amounts to learning the score $\nabla \log p_t$. In practice, the integral is approximated via Monte Carlo: sample $t \sim \mathrm{Unif}[0, 1]$, $x_0 \sim p_0$, $x_1 \sim \mathrm{N}(0, \mathrm{I}_d)$, form $x_t = \alpha_t x_0 + \sigma_t x_1$, and regress either x_0 or x_1.

Connection to SDEs

The interpolation (3.1) can be viewed as discretising a continuous-time process. Given a *noise schedule* $\beta(t) > 0$, consider the Itô SDE

$$\mathrm{d}X_t = -\tfrac{1}{2}\beta(t) X_t \,\mathrm{d}t + \sqrt{\beta(t)}\,\mathrm{d}W_t, \qquad X_0 \sim p_0, \tag{3.11}$$

where $(W_t)_{t \geqslant 0}$ is a standard Brownian motion in $\mathbb{R}^d$. For the VP schedule, setting $\alpha_t = \exp(-(1/2)\int_0^t \beta(s)\,\mathrm{d}s)$ and $\sigma_t = \sqrt{1 - \alpha_t^2}$, the marginal $\mathrm{Law}(X_t)$ coincides with p_t from

(3.1). The time-reversed process satisfies (Anderson 1982; Y. Song, Jascha Sohl-Dickstein, et al. 2021)

$$\mathrm{d}\bar{X}_t = \left[-\tfrac{1}{2}\beta(t)\bar{X}_t + \beta(t)\nabla_x \log p_t(\bar{X}_t)\right]\mathrm{d}t + \sqrt{\beta(t)}\,\mathrm{d}\bar{W}_t, \tag{3.12}$$

where $\bar{X}_t := X_{1-t}$ and $\bar{W}_t$ is a standard Brownian motion. The deterministic *probability-flow ODE* (Y. Song, Jascha Sohl-Dickstein, et al. 2021) with identical marginals reads

$$\mathrm{d}x_t = \left[-\tfrac{1}{2}\beta(t)x_t + \tfrac{1}{2}\beta(t)\nabla_x \log p_t(x_t)\right]\mathrm{d}t. \tag{3.13}$$

DDIM with $\eta_t = 0$ can be seen as a discretisation of (3.13), while DDPM corresponds to (3.12).

3.B Fine-tuning formulations

Given a positive reward function r and a pretrained diffusion model sampling from the base distribution p_0^{b}, we aim to fine-tune the model to sample from the *tilted distribution*, defined for $x \in \mathbb{R}^d$:

$$p_0^{\lambda r}(x) = Z_{\lambda r}^{-1} \exp(\lambda r(x))\, p_0^{\mathrm{b}}(x), \tag{3.14}$$

where $\lambda > 0$ is the tilt strength and $Z_{\lambda r} = \int \exp(\lambda r(x))\, p_0^{\mathrm{b}}(x)\,\mathrm{d}x$ is the normalising constant. This tilted distribution emphasises regions with high reward. Several formulations have been developed to tackle this goal.

Direct reward alignment (DRaFT). DRaFT (Clark et al. 2024) treats the diffusion sampler as a differentiable computation graph. For network parameters θ, the reverse sampling process is a deterministic function of θ and the i.i.d. Gaussian noises $\xi = (Z_K, \dots, Z_1)$, denoted $(\theta, \xi) \mapsto \psi_0(\theta, \xi)$, *i.e.*, $\psi_0(\theta, \xi) = \hat{X}_0$ where $\hat{X}_0$ is the output of Algorithm 5.

DRaFT maximises the expected reward:

$$J(\theta) = \mathbb{E}_\xi\left[r(\psi_0(\theta, \xi))\right] \tag{3.15}$$

by backpropagating the reward gradient through the entire chain of reverse steps:

$$\nabla_\theta J(\theta) = \mathbb{E}_\xi\left[\left(\frac{\partial \psi_0(\theta, \xi)}{\partial \theta}\right)^\top \nabla_x r(\psi_0(\theta, \xi))\right]. \tag{3.16}$$

Full backpropagation is memory-intensive. DRaFT-K reduces cost by differentiating only through the final K steps (typically $K \in \{1, 4\}$), detaching earlier gradients. DRaFT-LV further refines $K = 1$ with a control-variate correction for lower variance while remaining unbiased.

Stochastic optimal control and Adjoint Matching. An alternative approach casts fine-tuning as stochastic optimal control (SOC) (Uehara et al. 2024; Tang 2024; Domingo-Enrich, Drozdzal, et al. 2024). Rather than backpropagating through sampling (as in

DRaFT), SOC methods add a drift control u_t to shift the generative process toward high-reward regions. For a controlled SDE $\mathrm{d}X_t = [b(X_t, t) + g(t)u_t(X_t)]\,\mathrm{d}t + g(t)\,\mathrm{d}W_t$, where $g(t)$ denotes the diffusion coefficient (distinct from the noise schedule σ_t in (3.1)), the objective balances control effort against terminal reward:

$$\min_u \mathbb{E}\left[\tfrac{1}{2}\int_0^1 \|u_t(X_t)\|^2\,\mathrm{d}t - \lambda\, r(X_1)\right]. \tag{3.17}$$

By Girsanov's theorem, the quadratic control cost equals the KL divergence between controlled and base processes. However, naïvely solving (3.17) yields $p^*(X_0, X_1) \propto p^{\text{base}}(X_0, X_1)\exp(\lambda r(X_1)$ $V(X_0, 0))$, where V is the value function. This does not marginalize to the tilted distribution (3.14) unless the initial noise X_0 and generated sample X_1 are independent.

(Domingo-Enrich, Drozdzal, et al. 2024) show that a *memoryless* diffusion coefficient, one that ensures $X_0 \perp X_1$, removes the value function bias. Under this schedule, the optimal control is $u_t^\star(x) = g(t)^\top \tilde{a}_t(x)$, where the *lean adjoint* $\tilde{a}_t$ satisfies the backward ODE

$$\tfrac{\mathrm{d}}{\mathrm{d}t}\tilde{a}_t = -[\nabla_x b(x,t)]^\top \tilde{a}_t, \qquad \tilde{a}_1 = \lambda \nabla_x r(X_1). \tag{3.18}$$

Adjoint Matching learns a parametric control u_ϕ by minimizing:

$$\mathrm{L}_{\mathrm{AM}}(\phi) = \mathbb{E}\left[\int_0^1 \|u_\phi(X_t, t) + g(t)^\top \tilde{a}_t(X_t)\|^2\,\mathrm{d}t\right]. \tag{3.19}$$

After training, the learned control is absorbed into the drift, yielding a fine-tuned score or velocity field. Sampling can then use any diffusion coefficient, including deterministic sampling with $g(t) = 0$.

3.3 Iterative Tilting Method for Fine-Tuning Diffusion Models

The methods reviewed in Section 3.B share a common limitation: they require differentiating through the reward function r. In DRaFT, reward gradients appear explicitly in (3.16) via backpropagation through the sampling chain. In stochastic optimal control, they appear in the adjoint terminal condition $a_0 = \lambda \nabla_x r(X_0)$. When r is a large pretrained model (such as a vision transformer or human preference classifier trained on millions of examples), computing these gradients becomes prohibitively expensive, both in memory and compute. Moreover, many reward functions of practical interest are non-differentiable or defined only through black-box evaluations.

We introduce *iterative tilting*, a method that targets the reward-tilted distribution (3.14) using only forward evaluations of r, without any gradient computation. The key idea is to construct a path of intermediate distributions $(p_0^k)_{k=0}^N$ connecting the base p_0^{b} to the tilted $p_0^{\lambda r}$, each representing a gentle reweighting of the previous one. At each step, we reweight samples from the current distribution toward higher rewards, gradually approaching the target along this path. This sequential approach trades off a single expensive

gradient-based update for multiple cheaper gradient-free updates.

This section develops the method in continuous time, which makes the structure of the approximation explicit and connects naturally to the score-based and control perspectives introduced in Section 3.2.

3.A Score of the tilted distribution

Suppose we have a data distribution $X_0 \sim p_0^{\mathrm{b}}$ (the base distribution) and wish to sample from the tilted distribution (3.14). We use the forward noising kernel $q_{t|0}$ from (3.2), whose score is available in closed form:

$$\nabla_{x_t} \log q_{t|0}(x_t \mid x_0) = \frac{\alpha_t x_0 - x_t}{\sigma_t^2}. \tag{3.20}$$

Noising the tilted distribution, the marginal at time t is:

$$p_t^{\lambda r}(x_t) = \int q_{t|0}(x_t \mid x_0)\, p_0^{\lambda r}(x_0)\, \mathrm{d}x_0. \tag{3.21}$$

For any prior density p_0° (e.g., base p_0^{b} or tilted $p_0^{\lambda r}$) and the fixed forward noising kernel $q_{t|0}(x_t \mid x_0)$, we write $p_{0|t}^\circ(x_0 \mid x_t)$ for the posterior (inference) density of the clean sample X_0 given its noised version $X_t = x_t$ at time t:

$$p_{0|t}^\circ(x_0 \mid x_t) \propto p_0^\circ(x_0)\, q_{t|0}(x_t \mid x_0). \tag{3.22}$$

Proposition 3.1 (Score of the tilted distribution). Assume that r has at most linear growth, i.e., $\forall x \in \mathbb{R}^d :\ |r(x)| \leqslant C(1 + \|x\|)$ for some $C > 0$, and that $q_{t|0}$ is the Gaussian forward kernel (3.2). Then for all $t \in (0, 1]$ and $x_t \in \mathbb{R}^d$,

$$\nabla_{x_t} \log p_t^{\lambda r}(x_t) = \frac{\mathbb{E}_{X_0 \sim p_{0|t}^{\mathrm{b}}(\cdot|x_t)}\left[\nabla_{x_t} \log q_{t|0}(x_t \mid X_0)\, \exp(\lambda r(X_0))\right]}{\mathbb{E}_{X_0 \sim p_{0|t}^{\mathrm{b}}(\cdot|x_t)}[\exp(\lambda r(X_0))]}. \tag{3.23}$$

Proof. Exchanging differentiation and integration in (3.21) (justified by dominated convergence under the stated assumptions) yields

$$\nabla_{x_t} p_t^{\lambda r}(x_t) = \int p_0^{\lambda r}(x_0)\, \nabla_{x_t} q_{t|0}(x_t \mid x_0)\, \mathrm{d}x_0. \tag{3.24}$$

Dividing by $p_t^{\lambda r}(x_t)$ and using $\nabla_{x_t} p_t^{\lambda r}(x_t) = p_t^{\lambda r}(x_t) \nabla_{x_t} \log p_t^{\lambda r}(x_t)$, we obtain Fisher's identity:

$$\nabla_{x_t} \log p_t^{\lambda r}(x_t) = \mathbb{E}_{X_0 \sim p_{0|t}^{\lambda r}(\cdot|x_t)}\left[\nabla_{x_t} \log q_{t|0}(x_t \mid X_0)\right]. \tag{3.25}$$

The key observation is that the tilted posterior can be written as a reweighting of the base

posterior:

$$p_{0|t}^{\lambda r}(x_0 \mid x_t) = \frac{p_0^{\lambda r}(x_0)\, q_{t|0}(x_t \mid x_0)}{p_t^{\lambda r}(x_t)} = \frac{Z_{\lambda r}^{-1} \exp(\lambda r(x_0))\, p_0^{\mathrm{b}}(x_0)\, q_{t|0}(x_t \mid x_0)}{\int p_0^{\lambda r}(x_0')\, q_{t|0}(x_t \mid x_0')\, \mathrm{d}x_0'} = \frac{\exp(\lambda r(x_0))\, p_{0|t}^{\mathrm{b}}(x_0 \mid x_t)}{\mathbb{E}_{X_0' \sim p_{0|t}^{\mathrm{b}}(\cdot|x_t)}[\exp(\lambda r(X_0'))]}, \tag{3.26}$$

where in the last equality, the marginal $p_t^{\mathrm{b}}(x_t)$ cancels out. The tilted posterior $p_{0|t}^{\lambda r}(\cdot \mid x_t)$ is thus an exponential tilt of the base posterior $p_{0|t}^{\mathrm{b}}(\cdot \mid x_t)$. Substituting (3.26) into (3.25) gives (3.23). □

Iterative tilts. To efficiently approximate the score of the tilted distribution, we decompose the full tilt into a sequence of smaller tilts. Fix $N \in \mathbb{N}^*$ (typically large) and define, for $k = 0, \dots, N$,

$$p_0^k(x) := Z_k^{-1} \exp\big(\tfrac{k}{N} \lambda r(x)\big)\, p_0^{\mathrm{b}}(x), \tag{3.27}$$

where Z_k is the normalising constant. We have $p_0^0 = p_0^{\mathrm{b}}$ and $p_0^N = p_0^{\lambda r}$. The key observation is that consecutive distributions are related by a small tilt: $p_0^k(x) \propto \exp(\lambda r(x)/N)\, p_0^{k-1}(x)$.

Corollary 3.1 (First-order score approximation)**.** *Under the assumptions of Proposition 3.1, for all $x_t \in \mathbb{R}^d$ and $t \in (0, 1]$, define the posterior $\pi_{k-1}(\cdot) := p_{0|t}^{k-1}(\cdot \mid x_t)$ and the covariance*

$$C_{k-1}(x_t, t) := \mathrm{Cov}_{X_0 \sim \pi_{k-1}}\big(\nabla_{x_t} \log q_{t|0}(x_t \mid X_0),\, r(X_0)\big)\,. \tag{3.28}$$

Then, for large N,

$$\nabla_{x_t} \log p_t^k(x_t) = \nabla_{x_t} \log p_t^{k-1}(x_t) + \frac{\lambda}{N} C_{k-1}(x_t, t) + O\left(\frac{1}{N^2}\right). \tag{3.29}$$

Proof. Applying Proposition 3.1 with base p_0^{k-1} and tilt strength λ/N:

$$\nabla_{x_t} \log p_t^k(x_t) = \frac{\mathbb{E}_{\pi_{k-1}}\big[g(X_0)\, \exp(\frac{\lambda}{N} r(X_0))\big]}{\mathbb{E}_{\pi_{k-1}}\big[\exp(\frac{\lambda}{N} r(X_0))\big]}, \tag{3.30}$$

where $g(X_0) := \nabla_{x_t} \log q_{t|0}(x_t \mid X_0)$. Taylor-expanding $\exp(\lambda r/N) = 1 + \lambda r/N + O(N^{-2})$ in both numerator and denominator:

$$\begin{aligned} \text{Numerator} &= \mathbb{E}_{\pi_{k-1}}[g] + \tfrac{\lambda}{N}\, \mathbb{E}_{\pi_{k-1}}[g\, r] + O(N^{-2}), \\ \text{Denominator} &= 1 + \tfrac{\lambda}{N}\, \mathbb{E}_{\pi_{k-1}}[r] + O(N^{-2}). \end{aligned}$$

Dividing and keeping first-order terms yields $\mathbb{E}_{\pi_{k-1}}[g] + (\lambda/N)\, C_{k-1}(x_t, t) + O(N^{-2})$. By Fisher's identity, $\mathbb{E}_{\pi_{k-1}}[g] = \nabla_{x_t} \log p_t^{k-1}(x_t)$. □

3.B Neural approximation and algorithm

Corollary 3.1 motivates a recursive algorithm for estimating the sequence of score functions $\{\nabla_{x_t} \log p_t^k\}_{k=1}^N$ starting from $k = 0$, for which $p_t^0 = p_t^{\mathrm{b}}$. Suppose that at iteration k, we have access to an approximation for $\nabla_{x_t} \log p_t^k$, denoted by $s_{\theta^k}^k$. The resulting generative models associated to $s_{\theta^k}^k$ is called the teacher at step k. Corollary 3.1 provides a natural procedure to obtain an approximation for $\nabla_{x_t} \log p_t^{k+1}$: we train a new network s_θ^{k+1} with weights θ, so that it matches the target score from (3.29). The updated parameter θ^{k+1} is thus obtained as an approximate minimizer of the loss

$$\mathrm{L}_{\mathrm{IT}}(\theta; k+1) = \int_0^1 \int_{\mathcal{X}} w(t) \left\| s_\theta(x_t, t) - \left(s_{\theta^k}^k(x_t, t) + \frac{\lambda}{N} C_k(x_t, t) \right) \right\|^2 p_t^k(x_t) \, \mathrm{d}x_t \, \mathrm{d}t \, . \tag{3.31}$$

In practice, we make three simplifications. First, we use a Monte Carlo estimate of the loss using a teacher-generated dataset $\{X_0^i\}_{i+1}^N$ at the beginning of each step k. Second, for $X_t^i \sim q_{t|0}(\cdot|X_0^i)$, the covariance $C_k(X_t^i, t)$ is approximated by a single-sample estimate $g^i = r(X_0^i)(\nabla_{x_t} \log q_{t|0}(X_t^i \mid X_0^i) - s_{\theta^k}(X_t^i, t))$, where $\nabla_{x_t} \log q_{t|0}$ is computed via (3.20). Finally, we multiply by σ_t^2 to stabilize training across noise levels. This leads to the following empirical loss for a single sample time t and pair (X_0^i, X_1^i): setting $\delta = \lambda/N$,

$$\mathrm{L}_{\mathrm{prac}}(\theta; k+1) = \sum_{i=1}^N \sigma_t^2 \left\| s_\theta^{k+1}(X_t^i, t) - \sigma_t^2 \left(s_{\theta^k}^k(X_t^i, t) + \delta \, r(X_0^i) \left(g^i - s_{\theta^k}^k(X_t^i, t) \right) \right) \right\|^2 \, , \tag{3.32}$$

Finally, since we expect that $\nabla_{x_t} \log p_t^k$ to be close to $\nabla_{x_t} \log p_t^{k+1}$, the score network at step $k+1$, s_θ^{k+1} is initialized at $s_{\theta^k}^k$. The iterative tilting procedure is summarised in Algorithm 6.

3.4 Experiments

In this section, we validate the iterative tilting method on a two-dimensional Gaussian mixture with a reward function. This setup has a key advantage: tilting a Gaussian (or mixture of Gaussians) with a quadratic reward (including linear as a special case) yields another Gaussian (or mixture), for which we can compute the exact tilted density and score analytically.

3.A Gaussian data with quadratic rewards

One Gaussian. Suppose the base density is Gaussian:

$$p_0^{\mathrm{b}}(x) = \frac{1}{(2\pi)^{d/2} |\Sigma|^{1/2}} \exp\left\{ -\tfrac{1}{2} (x-\mu)^T \Sigma^{-1} (x-\mu) \right\} = \mathrm{N}(x; \mu, \Sigma). \tag{3.33}$$

and the reward function r is quadratic, i.e.:

$$r(x) = \tfrac{1}{2} x^T A x + b^T x + c. \tag{3.34}$$

Algorithm 6: Iterative Tilting

Require: Pretrained model s_{θ^0}, number of tilts N, tilt strength λ, samples per tilt S, reverse steps n, batch size B, epochs E, schedule (α_t, σ_t)

Set step size $\delta \leftarrow \lambda/N$ **for** $k = 1$ *to* N **do**

  Generate dataset D_k of S samples: $x_0 \leftarrow \mathrm{DDIM}(s_{\theta^k}, n)$

Initialise $\theta \leftarrow \theta^k$ and freeze teacher $s^k_{\theta^k}$ **for** *epoch* $= 1$ *to* E **do**

  Sample mini-batch $\{X_0^{(i)}\}_{i=1}^B \subset \mathrm{D}_k$, $\{t_i\}_{i=1}^B \overset{\text{iid}}{\sim} \mathrm{Unif}[0,1]$, $\{Z_i\}_{i=1}^B \overset{\text{iid}}{\sim} \mathrm{N}(0, \mathrm{I}_d)$ **for** $i = 1$ *to* B **do**

    Noise: $x_{t_i}^{(i)} = \alpha_{t_i} X_0^{(i)} + \sigma_{t_i} Z_i$

  Compute: $g_i = (\alpha_{t_i} X_0^{(i)} - X_{t_i}^{(i)})/\sigma_{t_i}^2$

  Compute: $s_i^{\text{old}} = s^k_{\theta^k}(X_{t_i}^{(i)}, t_i)$

  Target: $\text{target}_i = s_i^{\text{old}} + \delta\, r(X_0^{(i)})(g_i - s_i^{\text{old}})$

  Update θ by one step of gradient descent on

$$\frac{1}{B}\sum_{i=1}^{B} \left\| \sigma_{t_i}^2 s_\theta(X_{t_i}^{(i)}, t_i) - \sigma_{t_i}^2 \text{target}_i \right\|^2$$

Set $\theta^k \leftarrow \theta$

return s_{θ^N}

Then the product $\exp(r(x))p_0^{\mathrm{b}}(x)$ writes:

$$\frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left\{-\tfrac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu) + \left(\tfrac{1}{2}x^TAx + b^Tx + c\right)\right\}. \tag{3.35}$$

Focusing on the term in the exponent:

$$\begin{aligned} &-\tfrac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu) + \left(\tfrac{1}{2}x^TAx + b^Tx + c\right) \\ &= -\tfrac{1}{2}\left[x^T(\Sigma^{-1}-A)x - 2(\Sigma^{-1}\mu + b)^Tx\right] + c - \tfrac{1}{2}\mu^T\Sigma^{-1}\mu. \end{aligned} \tag{3.36}$$

With $\Sigma' = (\Sigma^{-1} - A)^{-1}$ and $\mu' = \Sigma'(\Sigma^{-1}\mu + b)$ (we assume $\Sigma^{-1} - A$ is positive definite so that the tilt is normalizable), the exponential can be rewritten as

$$\exp(r(x))p_0^{\mathrm{b}}(x) = \left(\frac{|\Sigma'|}{|\Sigma|}\right)^{1/2} \exp\left\{\tfrac{1}{2}\left[\mu'^T\Sigma'^{-1}\mu' - \mu^T\Sigma^{-1}\mu\right] + c\right\} \mathrm{N}(x;\mu',\Sigma'). \tag{3.37}$$

Consequently,

$$\int \exp(r(x))p_0^{\mathrm{b}}(x)\,dx = \left(\frac{|\Sigma'|}{|\Sigma|}\right)^{1/2} \exp\left\{\tfrac{1}{2}\left[\mu'^T\Sigma'^{-1}\mu' - \mu^T\Sigma^{-1}\mu\right] + c\right\}. \tag{3.38}$$

Mixture of Gaussians. Suppose that we now have a mixture of Gaussians with $(\omega_1, \ldots, \omega_K) \in \Delta_{K-1}$ where

$$\Delta_{K-1} = \{(\omega_1, \ldots, \omega_K) \in \mathbb{R}_+^K : \sum_{i\in[K]} \omega_i = 1\} \tag{3.39}$$

is the simplex and $(\mu_1, \Sigma_1), \ldots, (\mu_K, \Sigma_K)$ corresponding means and covariances.

$$p_0^{\mathrm{b}}(x) = \sum_{i\in[K]} \omega_i\, \mathrm{N}(x;\mu_i,\Sigma_i). \tag{3.40}$$

Then, we have:

$$\begin{aligned} \exp(r(x))p_0^{\mathrm{b}}(x) &= \sum_{i\in[K]} \omega_i\, \exp(r(x))\, \mathrm{N}(x;\mu_i,\Sigma_i) \\ &= \sum_{i\in[K]} \omega_i'\, \mathrm{N}(x;\mu_i',\Sigma_i'), \end{aligned} \tag{3.41}$$

where the parameters of the tilted Gaussians are

$$\begin{cases} \Sigma_i' = (\Sigma_i^{-1} - A)^{-1}, \\ \mu_i' = \Sigma_i'(\Sigma_i^{-1}\mu_i + b), \\ \omega_i' = \omega_i \left(\dfrac{|\Sigma_i'|}{|\Sigma_i|}\right)^{1/2} \exp\left\{\tfrac{1}{2}\left[\mu_i'^T\Sigma_i'^{-1}\mu_i' - \mu_i^T\Sigma_i^{-1}\mu_i\right] + c\right\}. \end{cases} \tag{3.42}$$

We consider the following mixture of two Gaussians in dimension $d = 2$ as base distribution:

$$\begin{cases} K = 2 \\ \mu_1 = (-2, 0);\ \mu_2 = (2, 0) \\ \Sigma_1 = \Sigma_2 = \sigma^2 \mathbf{I};\ \sigma^2 = 1/2 \\ \omega_1 = \omega_2 = 1/2 \end{cases} \tag{3.43}$$

and we consider the linear reward $r(x) = b^T x$ with $b = (0, 4)$ (corresponding to $A = 0$, $c = 0$ in the general quadratic form). In this case, the resulting distribution is a mixture of two Gaussians with parameters given by Section 3.A:

$$\begin{cases} K' = 2 \\ \mu'_1 = (-2, 2);\ \mu'_2 = (2, 2) \\ \Sigma'_1 = \Sigma'_2 = \sigma^2 \mathbf{I};\ \sigma^2 = 1/2 \\ \omega'_1 = \omega'_2 = 1/2. \end{cases} \tag{3.44}$$

3.B Implementation and results

Experimental setup. We first train a diffusion model on the base distribution using the denoising score matching objective (3.10) with the cosine noise schedule of (Alex Nichol and Dhariwal 2021). The architecture of the base model is a MLP model with depth 5 of width 256. We embed the time t with Fourier embeding (Tancik et al. 2020; Y. Song, Jascha Sohl-Dickstein, et al. 2021) of dimension 128. We train the model on a dataset of $30,000$ samples from the base distribution for 200 epochs with early stopping (patience 50). The base model achieves a test negative log-likelihood of 3.278 nats (computed on $5,000$ data points) and a mean squared error on the mean of 4.3×10^{-3}.

We then fine-tune the model using the iterative tilting method (Algorithm 6) with $N \in \{20, 50, 100, 200\}$ tilts, $S = 1,000$ samples per tilt, $J = 200$ reverse steps with stochastic DDIM ($\eta = 1.0$), batch size $B = 64$ and $E = 100$ epochs per tilt.

Results. We provide in Figure 3.1 samples obtained with the fine-tuned model at different stages of the tilting process for $N \in \{20, 50, 100, 200\}$.

At each iteration of the tilting procedure, we know the closed form of the score of p_0^k thanks to Section 3.A, and we can thus compute the mean squared error between the learned score and the true score of the tilted distribution. Thus, we monitor the RMSE (Root Mean Squared Error) between the learned score and the true score at each iteration $k = 1, \dots, N$:

$$\mathrm{RMSE}_k = \mathbb{E}_{t \sim \mathrm{Unif}[0,1], X_t \sim p_t^k} \left[\| s_{\theta^k}(X_t, t) - \nabla_x \log p_t^k(X_t) \|^2 \right]^{1/2}. \tag{3.45}$$

We estimate this RMSE using $5,000$ samples from p_t^k obtained with 200 stochastic DDIM reverse steps. We provide in Figure 3.2 the evolution of this RMSE during the tilting procedure for different values of N.

We report running times—total sampling time (s) and total training time (s)—and evaluation metrics (negative log-likelihood and mean squared mean error) in Table 3.1.

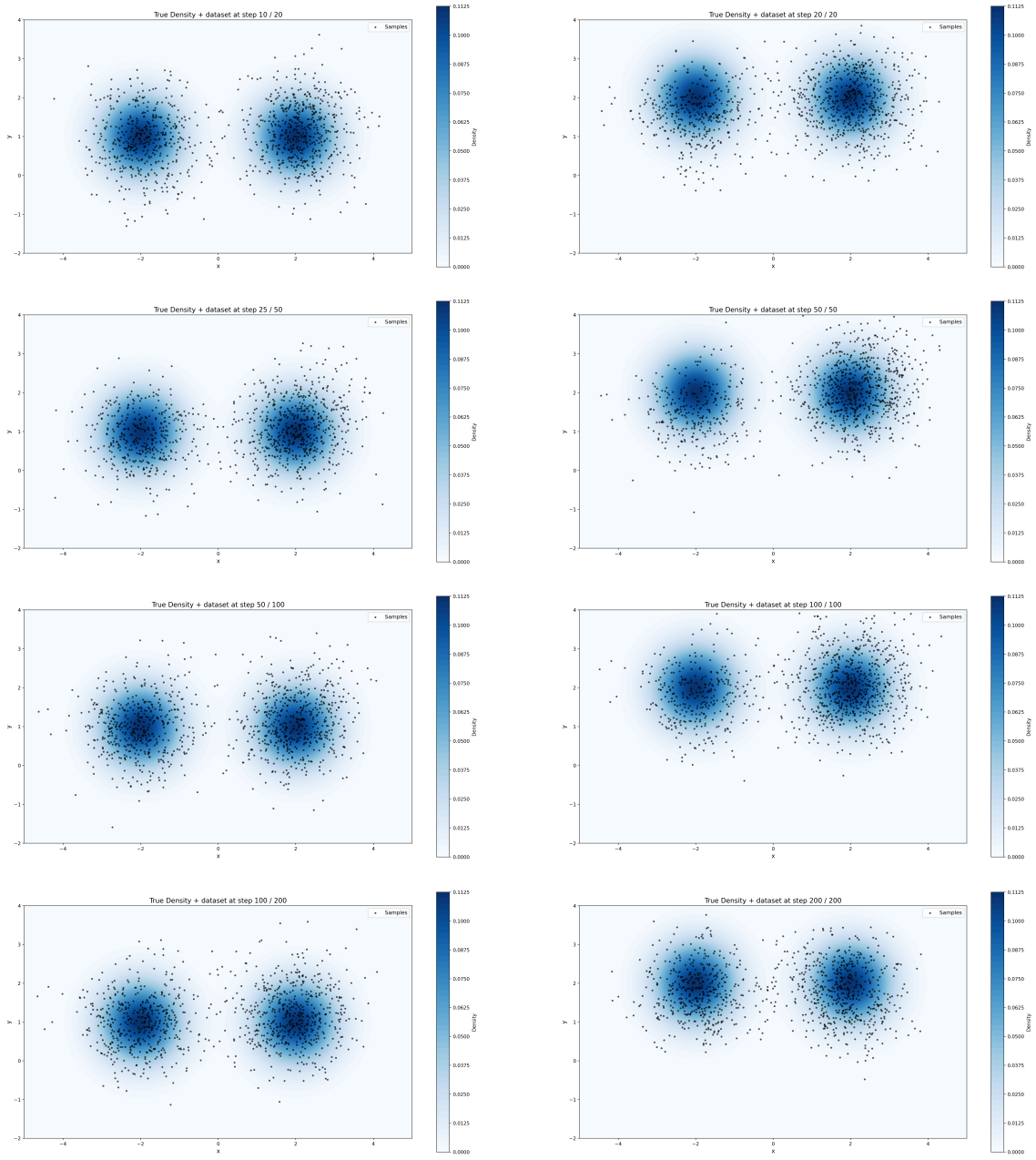


Figure 3.1: Iterative tilting on the 2-D GMM. Each row shows fixed-N settings (top to bottom: $N = 20, 50, 100, 200$) at iterations $N/2$ (left) and N (right). The rightmost plot in each row corresponds to the target distribution.

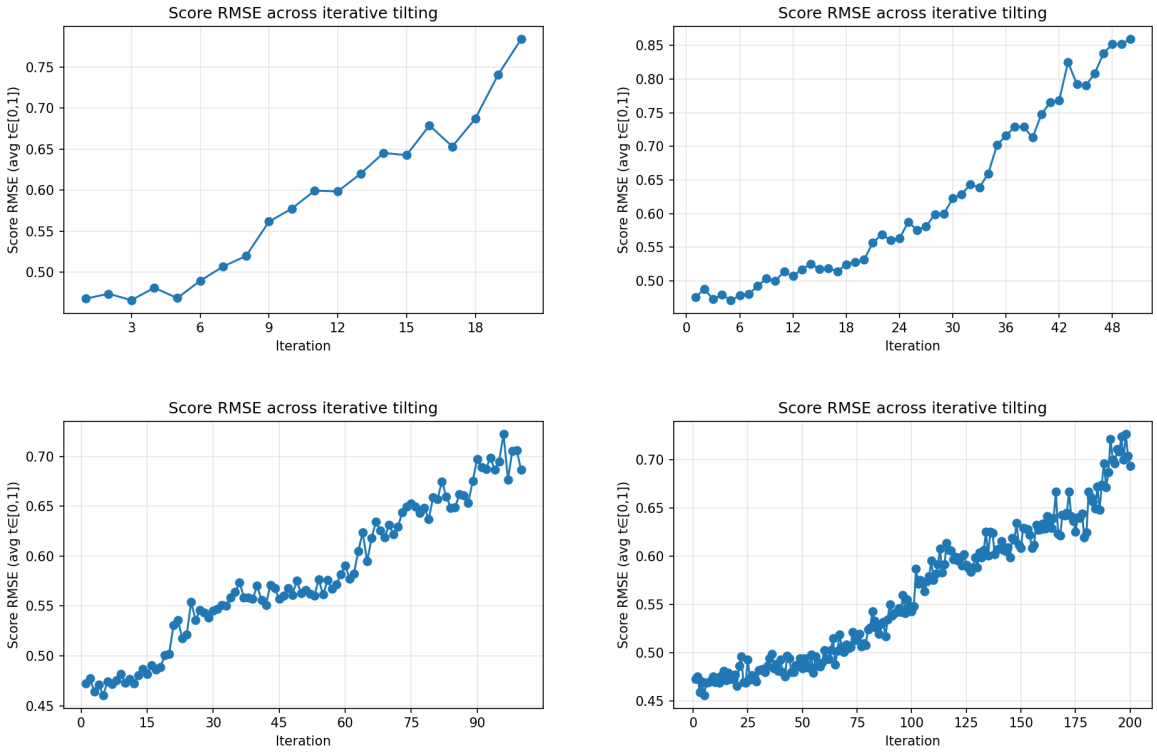


Figure 3.2: Evolution of the score error during iterative tilting for different N. We plot RMSE ($\sqrt{\text{MSE}}$) versus tilt iteration for $N \in \{20, 50, 100, 200\}$.

These controlled experiments validate the mechanics of iterative tilting in a setting where the exact score is observable. They also quantify the computational footprint of increasing the number of tilts, providing practical guidance for further studies on higher-dimensional datasets and richer reward models.

3.5 Conclusion

This chapter introduced iterative tilting, a gradient-free method for fine-tuning diffusion models toward reward-tilted distributions. The core idea is to decompose a single large tilt $\exp(\lambda r)$ into N sequential tilts $\exp(\lambda r/N)$, each producing a tractable score update. At iteration k, the score is approximated by

$$\nabla_x \log p_t^k(x) \approx \nabla_x \log p_t^{k-1}(x) + \frac{\lambda}{N} \operatorname{Cov}_{X_0 \sim p_{0|t}^{k-1}(\cdot|x)} \big(\nabla_x \log q_{t|0}(x \mid X_0),\, r(X_0) \big) ,$$

valid for small λ/N via first-order Taylor expansion. Crucially, the method requires only forward evaluations of r, avoiding the cost of backpropagating through sampling chains or solving adjoint equations.

We validated the approach on a two-dimensional Gaussian mixture with linear reward, where the exact tilted distribution is available in closed form. The experiments confirmed that the method successfully recovers the target distribution for $N \in \{20, 50, 100, 200\}$, with score error (RMSE) converging at each tilt and runtime scaling linearly with N.

	Runtime (s)		Metrics	
Number of tilts N	Sampling (s)	Training (s)	Neg. log-lik.	MSE (mean)
20	28.77	34.71	6.48	1.67
50	71.83	85.18	6.94	2.20
100	149.14	182.54	7.23	2.16
200	289.61	355.76	6.34	1.59

Table 3.1: Running times and metrics for iterative tilting on the 2-D GMM. Runtime columns report total sampling and training time in seconds. Metrics report negative log-likelihood and mean squared error on the mean.

Compared to DRaFT (Clark et al. 2024) and stochastic optimal control (Domingo-Enrich, Han, et al. 2024), which both require differentiating through the reward, iterative tilting trades a single gradient-based update for a sequence of gradient-free updates. This makes it well suited to settings where r is a large pretrained model or is only available through black-box evaluations.

Several directions remain open: reducing variance in the single-sample covariance approximation, establishing principled criteria for selecting N, extending to high-dimensional domains with learned reward models, and combining with parameter-efficient methods such as LoRA (Hu et al. 2021).

References

Akiba, Takuya, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama (July 2019). *Optuna: A Next-generation Hyperparameter Optimization Framework*. Comment: 10 pages, Accepted at KDD 2019 Applied Data Science track. arXiv: 1907.10902 [cs]. (Visited on 08/03/2025).

Allouche, M., S. Girard, and E. Gobet (2022). "EV-GAN: Simulation of extreme events with ReLU neural networks". In: *Journal of Machine Learning Research* 23.150, pp. 1–39. ISSN: 1533-7928. URL: http://jmlr.org/papers/v23/21-0663.html (visited on 05/16/2023).

— (2024). "Estimation of extreme quantiles from heavy-tailed distributions with neural networks". In: *Statistics and Computing* 34, p. 12.

Allouche, Michaël, Stéphane Girard, and Emmanuel Gobet (2022). *Robust Estimation for Extremal Dependence*. arXiv: 2206.10876.

— (Feb. 2025). "Learning extreme expected shortfall and conditional tail moments with neural networks. Application to cryptocurrency data". In: *Neural Networks* 182, p. 106903. ISSN: 08936080. DOI: 10.1016/j.neunet.2024.106903. (Visited on 02/21/2025).

Allouche, Michaël, Emmanuel Gobet, and Stéphane Girard (2024). "Extreme Event Simulation Using Generative Adversarial Networks". In: *Handbook of Statistics*. Chapter on extreme value GAN methods. Elsevier.

Anderson, Brian D.O. (1982). "Reverse-time diffusion equation models". In: *Stochastic Processes and their Applications* 12.3, pp. 313–326. DOI: 10.1016/0304-4149(82)90051-5.

Anonymous (2025). "Tilt Matching for Scalable Sampling and Fine-Tuning". In: *Submitted to The Fourteenth International Conference on Learning Representations*. under review. URL: https://openreview.net/forum?id=tT7CXL3I9C.

Arjovsky, Martin, Soumith Chintala, and Léon Bottou (2017). "Wasserstein Generative Adversarial Networks". In: *Proceedings of the 34th International Conference on Machine Learning*. Vol. 70. Proceedings of Machine Learning Research. PMLR, pp. 214–223.

Arnold, Barry C., Narayanaswamy Balakrishnan, and Haikady N. Nagaraja (2008). *A First Course in Order Statistics*. Classics in Applied Mathematics. Philadelphia: SIAM. DOI: 10.1137/1.9780898719062.

Asmussen, S. and H. Albrecher (2010). *Ruin probabilities*. second. Advanced Series on Statistical Science & Applied Probability, 14. World Scientific Publishing Co. Pte. Ltd., Hackensack, NJ.

Bai, Yuntao et al. (2022). *Training a Helpful and Harmless Assistant with Reinforcement Learning from Human Feedback*. arXiv: 2204.05862.

Banner, Adrian D. and Raouf Ghomrasni (July 2008). "Local times of ranked continuous semimartingales". In: *Stochastic Processes and their Applications* 118.7, pp. 1244–1253. ISSN: 03044149. DOI: 10.1016/j.spa.2007.08.001. (Visited on 12/12/2024).

Barrera, D., S. Crépey, E. Gobet, Hoang-Dung Nguyen, and B. Saadeddine (2025). "Statistical Learning of Value-at-Risk and Expected Shortfall". In: *Mathematical Finance, to appear*. arXiv: 2209.06476 [q-fin, stat]. URL: http://arxiv.org/abs/2209.06476 (visited on 09/20/2024).

Barron, Andrew R. (1993). "Universal approximation bounds for superpositions of a sigmoidal function". In: *IEEE Transactions on Information Theory* 39.3, pp. 930–945. DOI: 10.1109/18.256500.

Beirlant, J., Y. Goegebeur, J. Teugels, and J. Segers (2004). *Statistics of extremes: theory and applications*. en. Wiley series in probability and statistics. Hoboken, NJ: Wiley. ISBN: 978-0-471-97647-9.

Bellman, Richard (1961). *Adaptive Control Processes: A Guided Tour*. Princeton, NJ: Princeton University Press.

Bevacqua, Emanuele, Carlo De Michele, Colin Manning, Anais Couasnon, Andreia F. S. Ribeiro, Alexandre M. Ramos, Edoardo Vignotto, Ana Bastos, Suzana Blesic, Fabrizio Durante, et al. (2021). "Guidelines for studying diverse types of compound weather and climate events". In: *Earth's Future* 9.11, e2021EF002340. DOI: 10.1029/2021EF002340.

Bhattacharya, P. K. (1974). "Convergence of Sample Paths of Normalized Sums of Induced Order Statistics". In: *The Annals of Statistics* 2.5. DOI: 10.1214/aos/1176342823.

Black, Kevin, Michael Janner, Yilun Du, Ilya Kostrikov, and Sergey Levine (Jan. 2024). *Training Diffusion Models with Reinforcement Learning*. DOI: 10.48550/arXiv.2305.13301. arXiv: 2305.13301 [cs]. (Visited on 09/18/2025).

Blondel, Mathieu, Olivier Teboul, Quentin Berthet, and Josip Djolonga (2020). "Fast Differentiable Sorting and Ranking". In: *Proceedings of the 37th International Conference on Machine Learning*. PMLR, pp. 950–959.

Boulaguiem, Y., J. Zscheischler, E. Vignotto, K. van der Wiel, and S. Engelke (2022). "Modeling and simulating spatial extremes by combining extreme value theory with generative adversarial networks". In: *Environmental Data Science* 1, e5.

Boulaguiem, Youssef, Jakob Zscheischler, Edoardo Vignotto, Karin van der Wiel, and Sebastian Engelke (2022). "Modeling and simulating spatial extremes by combining extreme value theory with generative adversarial networks". In: *Environmental Data Science* 1, e5. DOI: 10.1017/eds.2022.4.

Bucklew, J. A. (2004). *Introduction to rare event simulation*. Springer Series in Statistics. Springer-Verlag, New York.

Casper, Stephen, Xander Davies, Claudia Shi, Thomas Krendl Gilbert, Jérémy Scheurer, Javier Rando, Rachel Freedman, Tomasz Korbak, David Lindner, Pedro Freire, et al. (2023). *Open Problems and Fundamental Limitations of Reinforcement Learning from Human Feedback*. arXiv: 2307.15217.

Christiano, Paul F., Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei (2017). *Deep Reinforcement Learning from Human Preferences*. NeurIPS 2017. arXiv: 1706.03741.

Clark, Kevin, Paul Vicol, Kevin Swersky, and David J. Fleet (2024). *Directly Fine-Tuning Diffusion Models on Differentiable Rewards*. ICLR 2024. arXiv: 2309.17400.

Cont, R., M. Cucuringu, R. Xu, and C. Zhang (2022). "Tail-GAN: Learning to Simulate Tail Risk Scenarios". In: *SSRN*.

Cont, Rama (2001). "Empirical Properties of Asset Returns: Stylized Facts and Statistical Issues". In: *Quantitative Finance* 1.2, pp. 223–236. DOI: 10.1080/713665670.

Cremer, F., B. Sheehan, M. Fortmann, A.N. Kia, M. Mullins, F. Murphy, and S. Materne (2022). "Cyber risk and cybersecurity: a systematic review of data availability". In: *The Geneva Papers on risk and insurance-Issues and practice* 47.3, pp. 698–736.

Cuturi, Marco, Olivier Teboul, and Jean-Philippe Vert (2019). "Differentiable Ranking and Sorting using Optimal Transport". In: *Advances in Neural Information Processing Systems*. Vol. 32. Curran Associates, Inc.

Cybenko, George (1989). "Approximation by superpositions of a sigmoidal function". In: *Mathematics of Control, Signals, and Systems* 2.4, pp. 303–314. DOI: 10.1007/BF02551274.

David, H.A. and H.N. Nagaraja (2004). *Order statistics*. Wiley series in probability and statistics. Wiley. ISBN: 978-0-471-65401-8. URL: https://books.google.fr/books?id=bdhzFXg6xFkC.

David, Herbert A. and Haikady N. Nagaraja (2003). *Order Statistics*. 3rd. Wiley Series in Probability and Statistics. Hoboken, NJ: Wiley-Interscience. ISBN: 978-0-471-38926-2.

David, Herbert A (1973). "Concomitants of order statistics". In: *Bull. Inst. Internat. Statist.* 45.1, pp. 295–300.

Del Moral, P. and J. Garnier (2005). "Genealogical particle analysis of rare events". In: *The Annals of Applied Probability* 15.4, pp. 2496–2534.

Dhariwal, Prafulla and Alexander Nichol (2021). "Diffusion Models Beat GANs on Image Synthesis". In: *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan. Vol. 34. Curran Associates, Inc., pp. 8780–8794.

Dinh, Laurent, Jascha Sohl-Dickstein, and Samy Bengio (2017). "Density estimation using Real-NVP". In: *International Conference on Learning Representations*.

Domingo-Enrich, Carles, Michal Drozdzal, Brian Karrer, and Ricky T. Q. Chen (Jan. 2024). *Adjoint Matching: Fine-tuning Flow and Diffusion Generative Models with Memoryless Stochastic Optimal Control*. DOI: 10.48550/arXiv.2409.08861. arXiv: 2409.08861 [cs]. (Visited on 09/22/2025).

Domingo-Enrich, Carles, Michal Drozdzal, Brian Karrer, and Ricky T. Q. Chen (2025). *Adjoint Matching: Fine-tuning Flow and Diffusion Generative Models with Memoryless Stochastic Optimal Control*. arXiv: 2409.08861.

Domingo-Enrich, Carles, Jiequn Han, Brandon Amos, Joan Bruna, and Ricky T. Q. Chen (Oct. 2024). *Stochastic Optimal Control Matching*. DOI: 10.48550/arXiv.2312.02027. arXiv: 2312.02027 [math]. (Visited on 09/22/2025).

ECB Banking Supervision (2023). "2023 stress test of euro area banks". In: *Available at https://www.bankingsupervision.europa.eu/ecb/pub/pdf/ssm.Report_2023_Stress_Test~96bb5a3af8.en.pdf* ECB.

Einmahl, John H. J., Laurens de Haan, and Chen Zhou (2016). "Statistics of heteroscedastic extremes". In: *Journal of the Royal Statistical Society: Series B* 78.1, pp. 31–51. DOI: 10.1111/rssb.12099.

Einmahl, John H. J., Andrea Krajina, and Johan Segers (2012). "An M-estimator for tail dependence in arbitrary dimensions". In: *The Annals of Statistics* 40.3, pp. 1764–1793. DOI: 10.1214/12-AOS1023.

Embrechts, P., C. Klüppelberg, and T. Mikosch (1997). *Modelling Extremal Events for Insurance and Finance*. Springer-Verlag, Berlin.

Embrechts, Paul, Claudia Klüppelberg, and Thomas Mikosch (1997). *Modelling Extremal Events*. Berlin, Heidelberg: Springer. DOI: 10.1007/978-3-642-33483-2.

Esser, Patrick, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, Dustin Podell, Tim Dockhorn, Zion English, Kyle Lacey, Alex Goodwin, Yannik Marek, and Robin Rombach (Mar. 2024). *Scaling Rectified Flow Transformers for High-Resolution Image Synthesis*. DOI: 10.48550/arXiv.2403.03206. arXiv: 2403.03206 [cs]. (Visited on 09/22/2025).

European Banking Authority (2014). "Guidelines on the revised common procedures and methodologies for the supervisory review and evaluation process (SREP) and supervisory stress testing". In: *Available at https://eba.europa.eu/regulation-and-policy/supervisory-review-and-evaluation-srep-and-pillar-2/guidelines-for-common-procedures-and-methodologies-for-the-supervisory-review-and-evaluation-process-srep-and-supervisory-stress-testing* EBA/GL/2014/13.

European Commission (2024a). URL: https://finance.ec.europa.eu/sustainable-finance/tools-and-standards/esg-rating-activities_en.

— (2024b). URL: https://finance.ec.europa.eu/sustainable-finance/overview-sustainable-finance_en.

Falk, M. (2019). *Multivariate Extreme Value Theory and D-Norms*. en. Springer Series in Operations Research and Financial Engineering. Cham: Springer International Publishing. ISBN: 978-3-030-03818-2. DOI: 10.1007/978-3-030-03819-9. URL: http://link.springer.com/10.1007/978-3-030-03819-9 (visited on 05/16/2023).

Falk, M. and T. Fuller (2021). "New characterizations of multivariate Max-domain of attraction and D-Norms". en. In: *Extremes* 24.4, pp. 849–879. ISSN: 1386-1999, 1572-

915X. DOI: 10.1007/s10687-021-00416-4. URL: https://link.springer.com/10.1007/s10687-021-00416-4 (visited on 05/16/2023).

Fan, Ying, Olivia Watkins, Yuqing Du, Hao Liu, Moonkyung Ryu, Craig Boutilier, Pieter Abbeel, Mohammad Ghavamzadeh, Kangwook Lee, and Kimin Lee (2023). *DPOK: Reinforcement Learning for Fine-tuning Text-to-Image Diffusion Models*. NeurIPS 2023. arXiv: 2305.16381.

Feder, R. M., P. Berger, and G. Stein (2020). "Nonlinear 3D cosmic web simulation with heavy-tailed generative adversarial networks". In: *Physical Review D.* 102.10, pp. 103504, 18.

Fisher, R. A. and L. H. C. Tippett (1928). "Limiting forms of the frequency distribution of the largest or smallest member of a sample". In: *Mathematical Proceedings of the Cambridge Philosophical Society* 24.2, pp. 180–190. DOI: 10.1017/S0305004100015681.

Fougères, Anne-Laure and Cécile Mercadier (2013). "Multivariate extremes and max-linear models". In: *ESAIM: Probability and Statistics* 17, pp. 109–123.

Glasserman, P. (2013). *Monte carlo methods in financial engineering*. Stochastic modelling and applied probability. Springer New York. ISBN: 978-0-387-21617-1. URL: https://books.google.fr/books?id=aeAlBQAAQBAJ.

Gnedenko, B. (1943). "Sur La Distribution Limite Du Terme Maximum D'Une Série Aléatoire". In: *Annals of Mathematics* 44.3, pp. 423–453. ISSN: 0003486X, 19398980. URL: http://www.jstor.org/stable/1968974 (visited on 06/25/2024).

Gobet, E. and G. Liu (2015). "Rare event simulation using reversible shaking transformations". In: *SIAM Journal on Scientific Computing* 37.5, A2295–A2316.

Goodfellow, I., J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio (2014). "Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems*. Ed. by Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger. Vol. 27, pp. 2672–2680.

Goodfellow, Ian J., Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio (2014). "Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems*. Ed. by Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger. Vol. 27. Curran Associates, Inc.

Graf, S. and H. Luschgy (2000). *Foundations of quantization for probability distributions*. en. Lecture Notes in Mathematics 1730. Berlin; New York: Springer. ISBN: 978-3-540-67394-1.

Grover, Aditya, Eric Wang, Aaron Zweig, and Stefano Ermon (2019). "Stochastic Optimization of Sorting Networks via Continuous Relaxations". In: *International Conference on Learning Representations*.

Gulrajani, Ishaan, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville (2017). "Improved Training of Wasserstein GANs". In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc.

Gumbel, Emil J. (1960). "Bivariate exponential distributions". In: *Journal of the American Statistical Association* 55.292, pp. 698–707. DOI: 10.1080/01621459.1960.10483368.

Haan, L. de and Ana Ferreira (2006). *Extreme value theory: an introduction*. en. Springer Series in Operations Research. New York ; London: Springer. ISBN: 978-0-387-23946-0.

Haan, Laurens de and Ana Ferreira (2006). *Extreme Value Theory: An Introduction*. Springer Series in Operations Research. New York: Springer. ISBN: 978-0-387-23946-0.

Haario, Heikki, Eero Saksman, and Johanna Tamminen (Sept. 1999). "Adaptive proposal distribution for random walk Metropolis algorithm". In: *Computational Statistics* 14.3, pp. 375–395. ISSN: 0943-4062, 1613-9658. DOI: 10.1007/s001800050022. (Visited on 07/08/2025).

Hasan, A., K. Elkhalil, Y. Ng, J.M. Pereira, S. Farsiu, J. Blanchet, and V. Tarokh (2022). "Modeling extremes with d-max-decreasing neural networks". In: *Uncertainty in Artificial Intelligence*. PMLR, pp. 759–768.

Hastings, W. K. (1970). "Monte Carlo Sampling Methods Using Markov Chains and Their Applications". In: *Biometrika* 57.1, pp. 97–109. DOI: 10.1093/biomet/57.1.97.

Hickling, T. and D. Prangle (2024). *Flexible Tails for Normalizing Flows*. en. arXiv:2406.16971. URL: http://arxiv.org/abs/2406.16971 (visited on 06/27/2024).

Ho, Jonathan, Ajay Jain, and Pieter Abbeel (2020). *Denoising Diffusion Probabilistic Models*. arXiv: 2006.11239 [cs.LG].

Ho, Jonathan and Tim Salimans (July 2022). *Classifier-Free Diffusion Guidance*. DOI: 10.48550/arXiv.2207.12598. arXiv: 2207.12598 [cs]. (Visited on 09/22/2025).

Hofert, M., R. Huser, and A. Prasad (2018). "Hierarchical archimax copulas". In: *Journal of Multivariate Analysis* 167, pp. 195–211.

Hornik, Kurt, Maxwell Stinchcombe, and Halbert White (1989). "Multilayer feedforward networks are universal approximators". In: *Neural Networks* 2.5, pp. 359–366. DOI: 10.1016/0893-6080(89)90020-8.

Hu, Edward J., Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen (Oct. 2021). *LoRA: Low-Rank Adaptation of Large Language Models*. DOI: 10.48550/arXiv.2106.09685. arXiv: 2106.09685 [cs]. (Visited on 09/22/2025).

Hüsler, Jürg and Rolf-Dieter Reiss (1989a). "Maxima of normal random vectors: between independence and complete dependence". In: *Statistics & Probability Letters* 7.4, pp. 283–286. DOI: 10.1016/0167-7152(89)90106-5.

— (1989b). "Maxima of normal random vectors: between independence and complete dependence". In: *Statistics & Probability Letters* 7.4, pp. 283–286.

Huster, Todd, Jeremy Cohen, Zinan Lin, Kevin Chan, Charles Kamhoua, Nandi Kundu, Kevin Kwiat, and Noel Leslie (2021). *Pareto GAN: Extending the Representational Power of GANs to Heavy-Tailed Distributions*. arXiv: 2101.09113.

Hyvärinen, Aapo (2005). "Estimation of Non-Normalized Statistical Models by Score Matching". In: *Journal of Machine Learning Research* 6.24, pp. 695–709.

JanSSen, Anja and Phyllis Wan (2020). "K-means clustering of extremes". In: *Electronic Journal of Statistics* 14.1, pp. 1211–1233.

Joe, H. (2014). *Dependence modeling with copulas*. CRC press.

Kallenberg, O. (2002). *Foundations of modern probability*. Probability and its applications. Springer New York. ISBN: 978-0-387-95313-7. URL: https://books.google.fr/books?id=L6fhXh13OyMC.

Kim, S.H. and H.A. David (Mar. 1990). "On the dependence structure of order statistics and concomitants of order statistics". In: *Journal of Statistical Planning and Inference* 24.3, pp. 363–368. ISSN: 0378-3758. DOI: 10.1016/0378-3758(90)90055-y. (Visited on 07/21/2025).

Kingma, D. P. and J. Ba (2014). *Adam: A Method for Stochastic Optimization*. arXiv:1412.6980.

Kingma, D. P. and M. Welling (2014). "Auto-Encoding Variational Bayes". In: *2nd International Conference on Learning Representation, ICLR*.

Kingma, Diederik P. and Max Welling (2014). *Auto-Encoding Variational Bayes*. arXiv: 1312.6114.

Lafon, Marco, Philippe Naveau, and Anthony Leblois (2023). "On the use of variational autoencoders for extreme values modelling". In: *arXiv preprint arXiv:2303.11536*.

Leshno, Moshe, Vladimir Ya. Lin, Allan Pinkus, and Shimon Schocken (1993). "Multilayer feedforward networks with a nonpolynomial activation function can approximate any function". In: *Neural Networks* 6.6, pp. 861–867. DOI: 10.1016/S0893-6080(05)80131-5.

Liang, F., M. Mahoney, and L. Hodgkinson (2022). "FatTailed Variational Inference with Anisotropic Tail Adaptive Flows". en. In: *Proceedings of the 39th International Conference on Machine Learning*. PMLR, pp. 13257–13270. URL: https://proceedings.mlr.press/v162/liang22a.html (visited on 01/11/2024).

Lipman, Yaron, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le (Feb. 2023). *Flow Matching for Generative Modeling*. DOI: 10.48550/arXiv.2210.02747. arXiv: 2210.02747 [cs]. (Visited on 09/22/2025).

Liu, Tie-Yan (2009). *Learning to Rank for Information Retrieval*. Berlin, Heidelberg: Springer. DOI: 10.1007/978-3-642-14267-3.

Lo, Andrew W. and Ruixun Zhang (2021). "Quantifying the Impact of Impact Investing". In: *Management Science*. DOI: 10.2139/ssrn.3944367.

Maas, A.L., A.Y. Hannun, and A.Y. Ng (2013). "Rectier Nonlinearities Improve Neural Network Acoustic Models". en. In: *Proceedings of the International Conference on Machine Learning (ICML)*.

Mainik, Georg, Georgi Mitov, and Ludger Rüschendorf (2015). "Portfolio optimization for heavy-tailed assets: Extreme Risk Index vs. Markowitz". In: *Journal of Empirical Finance* 32, pp. 115–134.

McNeil, A.J., R. Frey, and P. Embrechts (2015). *Quantitative risk management*. Princeton Series in Finance. Princeton University Press, Princeton, NJ, pp. xix+699. ISBN: 978-0-691-16627-8.

McNeil, Alexander J., Rüdiger Frey, and Paul Embrechts (2015). *Quantitative Risk Management: Concepts, Techniques and Tools*. Revised. Princeton Series in Finance. Princeton, NJ: Princeton University Press.

Metropolis, Nicholas, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller (1953). "Equation of State Calculations by Fast Computing Machines". In: *The Journal of Chemical Physics* 21.6, pp. 1087–1092. DOI: 10.1063/1.1699114.

Murphy, K. P. (2022). *Probabilistic Machine Learning: An introduction*. MIT Press. URL: probml.ai.

Nadjahi, K. (Nov. 2021). "Sliced-Wasserstein distance for large-scale machine learning : theory, methodology and extensions". PhD thesis. Institut Polytechnique de Paris. URL: https://theses.hal.science/tel-03533097.

Nelsen, R.B. (2006). *An introduction to copulas*. en. second. Springer series in statistics. New York Berlin Heidelberg: Springer. ISBN: 978-0-387-28659-4.

Nichol, Alex and Prafulla Dhariwal (Feb. 2021). *Improved Denoising Diffusion Probabilistic Models*. DOI: 10.48550/arXiv.2102.09672. arXiv: 2102.09672 [cs]. (Visited on 10/27/2025).

Nikoloulopoulos, Aristidis K., Harry Joe, and Haijun Li (2009). "Extreme value properties of multivariate t copulas". In: *Extremes* 12.2, pp. 129–148. DOI: 10.1007/s10687-008-0072-4.

Ouyang, Long, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. (2022). *Training language models to follow instructions with human feedback*. NeurIPS 2022. arXiv: 2203.02155.

Papamakarios, G., E. Nalisnick, D. J. Rezende, S. Mohamed, and B. Lakshminarayanan (2021). "Normalizing flows for probabilistic modeling and inference". In: *The Journal of Machine Learning Research* 22.1, pp. 2617–2680.

Prandini, M. and O. J. Watkins (2005). "Probabilistic Aircraft Conflict Detection". In: *HYBRIDGE WP3: Reachability analysis for probabilistic hybrid systems*.

Resnick, S. I. (1987). *Extreme Values, Regular Variation and Point Processes*. en. Springer Series in Operations Research and Financial Engineering. New York, NY: Springer. ISBN: 978-0-387-75952-4. DOI: 10.1007/978-0-387-75953-1. URL: http://link.springer.com/10.1007/978-0-387-75953-1 (visited on 05/16/2023).

Resnick, Sidney I. (1987). *Extreme Values, Regular Variation and Point Processes*. Springer Series in Operations Research and Financial Engineering. New York: Springer. DOI: 10.1007/978-0-387-75953-1.

Rezende, Danilo Jimenez and Shakir Mohamed (2015). "Variational Inference with Normalizing Flows". In: *Proceedings of the 32nd International Conference on Machine Learning*. PMLR, pp. 1530–1538.

Robert, Christian P. and George Casella (2004). *Monte Carlo Statistical Methods*. 2nd. New York: Springer. DOI: 10.1007/978-1-4757-4145-2.

Robert, P. (2003). *Stochastic networks and queues*. Vol. 52. Applications of Mathematics. Springer-Verlag, Berlin.

Rosenblatt, Murray (Sept. 1952). "Remarks on a Multivariate Transformation". In: *The Annals of Mathematical Statistics* 23.3, pp. 470–472. ISSN: 0003-4851. DOI: 10.1214/aoms/1177729394. (Visited on 07/21/2025).

Sabourin, Anne, Philippe Naveau, and Anne-Laure Fougères (2013). "Bayesian Dirichlet mixture model for multivariate extremes: A re-parametrization". In: *Computational Statistics & Data Analysis* 71, pp. 542–567. DOI: 10.1016/j.csda.2013.04.021.

Schucany, William R. (1972). "Order statistics in simulation". In: *Journal of Statistical Computation and Simulation* 1.3, pp. 281–286. DOI: 10.1080/00949657208810024.

Scott, David W. (2015). *Multivariate Density Estimation: Theory, Practice, and Visualization*. 2nd. Hoboken, NJ: Wiley. DOI: 10.1002/9781118575574.

Silverman, Bernard W. (1986). *Density Estimation for Statistics and Data Analysis*. London: Chapman and Hall. DOI: 10.1007/978-1-4899-3324-9.

Sklar, M. (1959). "Fonctions de répartition à N dimensions et leurs marges". fr. In: *Publications de lInstitut Statistique de lUniversité de Paris* 8, pp. 229–231.

Sohl-Dickstein, J., E. Weiss, N. Maheswaranathan, and S. Ganguli (2015). "Deep unsupervised learning using nonequilibrium thermodynamics". In: *International conference on machine learning*. PMLR, pp. 2256–2265.

Sohl-Dickstein, Jascha, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli (Nov. 2015). *Deep Unsupervised Learning using Nonequilibrium Thermodynamics*. DOI: 10.48550/arXiv.1503.03585. arXiv: 1503.03585 [cs]. (Visited on 09/22/2025).

Song, Jiaming, Chenlin Meng, and Stefano Ermon (2021). "Denoising Diffusion Implicit Models". In: *International Conference on Learning Representations*.

Song, Yang and Stefano Ermon (2019). "Generative Modeling by Estimating Gradients of the Data Distribution". In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett. Vol. 32. Curran Associates, Inc.

Song, Yang, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole (2021). "Score-Based Generative Modeling through Stochastic Differential Equations". In: *International Conference on Learning Representations*.

Sukhatme, Pandurang V. (1937). "Tests of significance for samples of the χ^2-population with two degrees of freedom". In: *Annals of Eugenics* 8.1, pp. 52–56.

Tancik, Matthew, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng (June 2020). *Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains*. DOI: 10.48550/arXiv.2006.10739. arXiv: 2006.10739 [cs]. (Visited on 10/27/2025).

Tang, Wenpin (Mar. 2024). *Fine-tuning of diffusion models via stochastic control: entropy regularization and beyond*. DOI: 10.48550/arXiv.2403.06279. arXiv: 2403.06279 [math]. (Visited on 09/22/2025).

Uehara, Masatoshi, Yulai Zhao, Kevin Black, Ehsan Hajiramezanali, Gabriele Scalia, Nathaniel Lee Diamant, Alex M. Tseng, Tommaso Biancalani, and Sergey Levine (Feb. 2024).

Fine-Tuning of Continuous-Time Diffusion Models as Entropy-Regularized Control. DOI: 10.48550/arXiv.2402.15194. arXiv: 2402.15194 [cs]. (Visited on 09/22/2025).

Villani, C. (2009). *Optimal transport: old and new*. Vol. 338. Springer.

Vincent, Pascal (2011). "A Connection Between Score Matching and Denoising Autoencoders". In: *Neural Computation* 23.7, pp. 1661–1674. DOI: 10.1162/NECO_a_00142.

Wei, Xiang, Zixia Liu, Liqiang Wang, and Boqing Gong (2018). "Improving the improved training of wasserstein GANs". In: *International conference on learning representations*. Comment: NIPS camera-ready. URL: https://openreview.net/forum?id=SJx9GQb0-.

Wiese, Magnus, Robert Knobloch, Ralf Korn, and Peter Kretschmer (2019). *Deep Hedging: Learning to Simulate Equity Option Markets*. arXiv: 1911.01700.

— (2020). "Quant GANs: Deep generation of financial time series". In: *Quantitative Finance* 20.9, pp. 1419–1440.

Wu, J., X.-Y. Chen, H. Zhang, L.-D. Xiong, H. Lei, and S.-H. Deng (2019). "Hyperparameter Optimization for Machine Learning Models Based on Bayesian Optimization". en. In: 17.1, pp. 26–40.

Xu, Jiazheng, Xiao Liu, Yuchen Wu, Yuxuan Tong, Qinkai Li, Ming Ding, Jie Tang, and Yuxiao Dong (Dec. 2023). *ImageReward: Learning and Evaluating Human Preferences for Text-to-Image Generation*. DOI: 10.48550/arXiv.2304.05977. arXiv: 2304.05977 [cs]. (Visited on 04/28/2025).

Xu, Jing, Andrew Liu, Andrea Wu, Parisa Saeedi, Jennifer Neville, Mona Diab, and Asli Celikyilmaz (2024). *ReFL: Reflective Feedback Learning for Language Model Alignment*. arXiv: 2403.01673.

Yarotsky, Dmitry (2017). "Error bounds for approximations with deep ReLU networks". In: *Neural Networks* 94, pp. 103–114. DOI: 10.1016/j.neunet.2017.07.002.

Yoon, Sungbin, Juho Park, Jaehyeon Lee, and Gunhee Kim (2023). *Score-based Generative Modeling through Stochastic Evolution Equations in Hilbert Spaces*. Heavy-tailed diffusion models using Lévy processes. arXiv: 2305.19241.

Zhang, Likun, Andrew Bray, and Joseph Guinness (2024). "Flexible and efficient spatial extremes emulation via variational autoencoders". In: *arXiv preprint arXiv:2407.02868*.

Zscheischler, J., O. Martius, S. Westra, E. Bevacqua, C. Raymond, R. M. Horton, B. van den Hurk, A. AghaKouchak, A. Jézéquel, M. D. Mahecha, D. Maraun, A. M. Ramos, N. N. Ridder, W. Thiery, and E. Vignotto (2020). "A typology of compound weather and climate events". In: *Nature Reviews Earth & Environment* 1.7, pp. 333–347.

Zscheischler, Jakob, Seth Westra, Bart J. J. M. van den Hurk, Sonia I. Seneviratne, Philip J. Ward, Andy Pitman, Amir AghaKouchak, David N. Bresch, Michael Leonard, Thomas Wahl, and Xuebin Zhang (2018). "Future climate risk from compound events". In: *Nature Climate Change* 8, pp. 469–477. DOI: 10.1038/s41558-018-0156-3.

Supplementary Material

.1 Additional Figures

.2 Additional Tables

.3 Research Instruments