

¹ **Structure-Preserving Generative Models:**
² **Extreme Values, Order Statistics, and**
³ **Diffusion Fine-Tuning**

⁴ December 16, 2025

⁵ **Abstract**

6 Acknowledgments

7 Contents

8	Abstract	i
9	Acknowledgments	ii
10	List of Figures	ix
11	List of Tables	x
12	Introduction	1
13	0.1 Context and Motivation	1
14	0.1.1 Structure in Statistical Data	1
15	0.1.2 Modern Generative Modeling	1
16	0.1.3 The Problem: When Structure Matters	2
17	0.1.4 This Thesis: Theory-Guided Generative Methods	2
18	0.2 Three Structural Regimes	3
19	0.2.1 Extreme Values and Tail Dependence	3
20	0.2.2 Order Statistics and Ranking Constraints	5
21	0.2.3 Reward-Tilted Distributions and Model Fine-Tuning	6
22	0.3 Generative Modeling Background	7
23	0.3.1 The Problem of Generative Modeling	7
24	0.3.2 Neural Networks	8
25	0.3.3 Neural Network Approximation Theory	9
26	0.3.4 Generative Adversarial Networks	9
27	0.3.5 Diffusion and Score-Based Models	10
28	0.4 State of the Art and Open Problems	11
29	0.4.1 Extreme Value Generation	11
30	0.4.2 Order Statistics Generation	12
31	0.4.3 Reward-Tilted Generation	13
32	0.4.4 Objectives of This Thesis	14
33	0.5 Contributions of This Thesis	14
34	0.5.1 Heavy-Tailed GANs for Extreme Value Generation	14
35	0.5.2 Generative Neural Order Statistics	15
36	0.5.3 Gradient-Free Fine-Tuning of Diffusion Models	16
37	0.6 Organization of the Manuscript	16

38	Introduction (Version française)	18
39	0.7 Contexte et motivation	18
40	0.7.1 La structure dans les données statistiques	18
41	0.7.2 La modélisation générative moderne	18
42	0.7.3 Le problème : quand la structure compte	19
43	0.7.4 Cette thèse : méthodes génératives guidées par la théorie	19
44	0.8 Trois régimes structurels	20
45	0.8.1 Valeurs extrêmes et dépendance de queue	20
46	0.8.2 Statistiques d'ordre et contraintes de classement	23
47	0.8.3 Distributions inclinées par récompense et ajustement fin de modèles	23
48	0.9 Contexte de la modélisation générative	25
49	0.9.1 Le problème de la modélisation générative	25
50	0.9.2 Réseaux de neurones	26
51	0.9.3 Théorie d'approximation des réseaux de neurones	26
52	0.9.4 Réseaux antagonistes génératifs	27
53	0.9.5 Modèles de diffusion et basés sur le score	28
54	0.10 État de l'art et problèmes ouverts	28
55	0.10.1 Génération de valeurs extrêmes	28
56	0.10.2 Génération de statistiques d'ordre	30
57	0.10.3 Génération inclinée par récompense	31
58	0.10.4 Objectifs de cette thèse	32
59	0.11 Contributions de cette thèse	32
60	0.11.1 GAN à queue lourde pour la génération de valeurs extrêmes	32
61	0.11.2 Statistiques d'ordre neurales génératives	33
62	0.11.3 Ajustement fin sans gradient des modèles de diffusion	34
63	0.12 Organisation du manuscrit	35
64	1 Heavy-Tailed GANs for Extreme Value Generation	36
65	1.1 Introduction	36
66	1.2 Reproducing dependence in extreme regions with Generative Adversarial Networks: Theory	40
67	1.2.1 Extreme-value theory	40
68	1.2.2 Measuring dependence in extremes	41
69	1.2.3 Generative Adversarial Networks	43
70	1.2.4 New approximation results for the stdf	44
71	1.3 Numerical experiments	45
72	1.3.1 Algorithms and implementation	46
73	1.3.2 Experiments on simulated data	48
74	1.3.3 Experiments on real data	56
75	1.4 Conclusion	58
76	Appendix	60
77	3.A Theoretical background	60

79	3.B	Extreme-value theory	60
80	3.C	D-norms	61
81	3.D	Quantization	63
82	3.E	Copulas, measures of dependence	64
83	3.F	Proofs of claims	65
84	3.G	Proof of Proposition 1.4	66
85	3.H	Proof of Theorem 1.3	68
86	3.I	Additional experiments	70
87	3.J	Additional plots for experiments on Real data	70
88	2	Generative Neural Order Statistics	77
89	2.1	Introduction	77
90	3.A	Notations	80
91	2.2	A theoretical analysis of generative modeling of order statistics with Neural Networks	80
93	3.A	Representational results for order statistics	81
94	3.B	Generating order statistics: approximation results	81
95	2.3	Experiments	84
96	3.A	Synthetic data	84
97	3.B	Benchmarking on synthetic data	88
98	2.4	Conclusion	90
99	Appendix	92
100	2.A	Additional Figures	92
101	2.B	Some theory of concomitants	92
102	2.C	Concomitants of order statistics from a single distribution	95
103	2.D	Some elements on Functional approximation with Neural Network	98
104	2.E	Proofs	99
105	2.F	Proof of Theorem 2.2	102
106	2.G	Additional experimental details	105
107	2.H	Hyperparameters	106
108	2.I	Metrics	106
109	3	Gradient-Free Fine-Tuning of Diffusion Models	109
110	3.1	Introduction	109
111	3.2	Preliminaries on Denoising Diffusion Models and Fine-Tuning	110
112	3.A	Diffusion models	110
113	3.B	Fine-tuning formulations	113
114	3.3	Iterative Tilting Method for Fine-Tuning Diffusion Models	114
115	3.A	Score of the tilted distribution	115
116	3.B	Neural approximation and algorithm	117
117	3.4	Experiments	117
118	3.A	Gaussian data with quadratic rewards	117
119	3.B	Implementation and results	120

120	3.5 Conclusion	122
121	References	124

¹²² List of Figures

123	1	Light-tailed vs heavy-tailed distributions	4
124	2	Tail dependence with Pareto margins	5
125	3	Reward tilting of a Gaussian distribution	7
126	4	Feedforward neural network architecture	8
127	5	GAN framework as adversarial game	9
128	6	Diffusion model framework on a Gaussian mixture	11
129	7	Distributions à queue légère vs à queue lourde	21
130	8	Dépendance de queue avec marges de Pareto	22
131	9	Inclinaison par récompense d'une distribution gaussienne	24
132	10	Architecture d'un réseau de neurones feedforward	26
133	11	Le cadre GAN comme jeu adversarial	27
134	12	Cadre des modèles de diffusion sur un mélange de gaussiennes	29
135	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.	49
136	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.	50
137	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>i.e.</i> 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.	53

155	1.4 Illustration in a d -dimensional setting on simulated data of size $n_{\text{data}} =$	
156	10,000 from a Gumbel copula (with dependence parameter β) and Pareto	
157	margins (with tail parameter $\alpha = 2.5$), $d \in \{2, 5, 10, 20, 50\}$. See Figure 1.3	
158	for further details.	54
159	1.5 Estimated Kendall's τ for every pair of index considered. Left: every	
160	S&P500 ticker. On the top right: Tickers for the Financials sector. On	
161	the bottom right: Tickers for the Utilities sector. A comprehensive list of	
162	sectors and subsectors is given in Paragraph 1.3.3.	57
163	1.6 Results of the performance of HTGAN vs LTGAN on the Utilities ($d = 27$)	
164	and Financials subsectors ($d = 57$) of the S&P500 dataset. Results are	
165	aggregated with respect to the dimension d for plot. Each two groups of 2×3	
166	plots corresponds to one experimental setting, <i>i.e.</i> $\alpha = 1.5$ (top) and $\alpha = 2.5$	
167	(bottom). In each of these two groups: the first row plots the specified	
168	metric of HTGAN vs LTGAN in log scale for a given hyperparametrization	
169	configuration. The second row is a histogram of the difference of the log of	
170	the metric for HTGAN vs the metric for LTGAN for each parametrization.	
171	The legend corresponds to the proportion of cases where HTGAN performs	
172	better.	59
173	1.7 Illustration in a d -dimensional setting on simulated data of size $n_{\text{data}} =$	
174	10,000 from a Gaussian copula and Pareto margins, $d \in \{2, 5, 10, 20, 50\}$.	
175	Results obtained with the SWD metric. See Figure 1.3 for further details. .	71
176	1.8 Illustration in a d -dimensional setting on simulated data of size $n_{\text{data}} =$	
177	10,000 from a Gaussian copula and Pareto margins, $d \in \{2, 5, 10, 20, 50\}$.	
178	Results obtained with the AKE metric. See Figure 1.3 for further details. .	72
179	1.9 Illustration in a d -dimensional setting on simulated data of size $n_{\text{data}} =$	
180	10,000 from a Hüsler-ReiSS copula and Pareto margins, $d \in \{2, 5, 10, 20, 50\}$.	
181	Results obtained with the SWD metric. See Figure 1.3 for further details. .	73
182	1.10 Illustration in a d -dimensional setting on simulated data of size $n_{\text{data}} =$	
183	10,000 from a Hüsler-ReiSS copula and Pareto margins, $d \in \{2, 5, 10, 20, 50\}$.	
184	Results obtained with the AKE metric. See Figure 1.3 for further details. .	74
185	1.11 (a)-(d): histograms of real vs synthetic marginals across dimensions one to	
186	four. (e)-(f): Scatter plot AMF vs BEN and BK vs BLK tickers. (g)-(h):	
187	Histogram of θ after selecting largest values, at level 80% and 90%.	76
188	2.1 Real and generated data after min-max renormalization. <code>n_samples=1,000</code> . .	86
189	2.2 Comparison of real and generated 2D order statistics under different nor-	
190	malization schemes, with an additional overlay view. The red dashed lines	
191	indicate $x = y$; the meanscale normalization preserves the sorted structure	
192	(no line crossing), while minmax normalization does not.	87
193	2.3 Meanscale comparison across dimensions	92

194	2.4 Generated vs. real means for order statistics in dimension 10. Every di-	
195	dimension is scattered against one another. Real data points are in blue and	
196	generated data points are in orange.	93
197	3.1 Iterative tilting on the 2-D GMM. Each row shows fixed- N settings (top to	
198	bottom: $N = 20, 50, 100, 200$) at iterations $N/2$ (left) and N (right). The	
199	rightmost plot in each row corresponds to the target distribution.	121
200	3.2 Evolution of the score error during iterative tilting for different N . We plot	
201	RMSE ($\sqrt{\text{MSE}}$) versus tilt iteration for $N \in \{20, 50, 100, 200\}$	122

²⁰² List of Tables

203	1	Tail probabilities comparison	4
204	2	Comparaison des probabilités de queue	21
205	1.1	Results on simulated data from a Gumbel copula. Percentage (%) of parametriza-	
206		tions for which HTGAN is better than a LTGAN for the six considered	
207		metrics. Results are given with precision $\pm 0.1\%$. Values are averaged over	
208		data dimensions $d \in \{2, 5, 10, 20, 50\}$	52
209	1.2	Results on simulated data from a Gumbel copula. Statistics (mean, min,	
210		max) on three on the top 5% best performing runs of HTGAN (best perform-	
211		ing w.r.t. the swd_90 metric). Statistics are averaged over $\beta \in \{4/3, 2, 4\}$	55
212	1.3	Results on simulated data from Hüsler-ReiSS and Gaussian copulas. Per-	
213		centage (%) of parametrizations for which HTGAN is better than a LTGAN	
214		for the six considered metrics. Results are given with precision $\pm 0.1\%$. Val-	
215		ues are averaged over data dimensions $d \in \{2, 5, 10, 20, 50\}$	56
216	2.1	Hyperparameters used for training our vanilla model.	86
217	2.2	Summary of metric performances for various models across dimensions.	89
218	3.1	Running times and metrics for iterative tilting on the 2-D GMM. Runtime	
219		columns report total sampling and training time in seconds. Metrics report	
220		negative log-likelihood and mean squared error on the mean.	123

221 Introduction

222 0.1 Context and Motivation

223 0.1.1 Structure in Statistical Data

224 The term *statistics* derives from the German *Statistik*, coined by Gottfried Achenwall in
225 1748 to denote the “science of the state”: governing required organized information. Data
226 is the raw material of statistics. A scientist measures temperature at several locations; a
227 bank records daily returns of assets in a portfolio; a hospital collects patient vital signs.
228 In each case, we have multiple observations, and each observation may consist of several
229 related quantities. These quantities are organized in specific ways, governed by underlying
230 regularities. Uncovering these regularities is the task of statistical science.

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

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

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

243 0.1.2 Modern Generative Modeling

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

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

250 to produce data from it. The core idea is:

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

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

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

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

263 0.1.3 The Problem: When Structure Matters

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

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

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

274 0.1.4 This Thesis: Theory-Guided Generative Methods

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

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

283 0.2 Three Structural Regimes

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

286 0.2.1 Extreme Values and Tail Dependence

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

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

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

308 **Two distinct challenges.** Generating realistic extreme value data poses two fundamen-
309 tally different challenges.

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

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

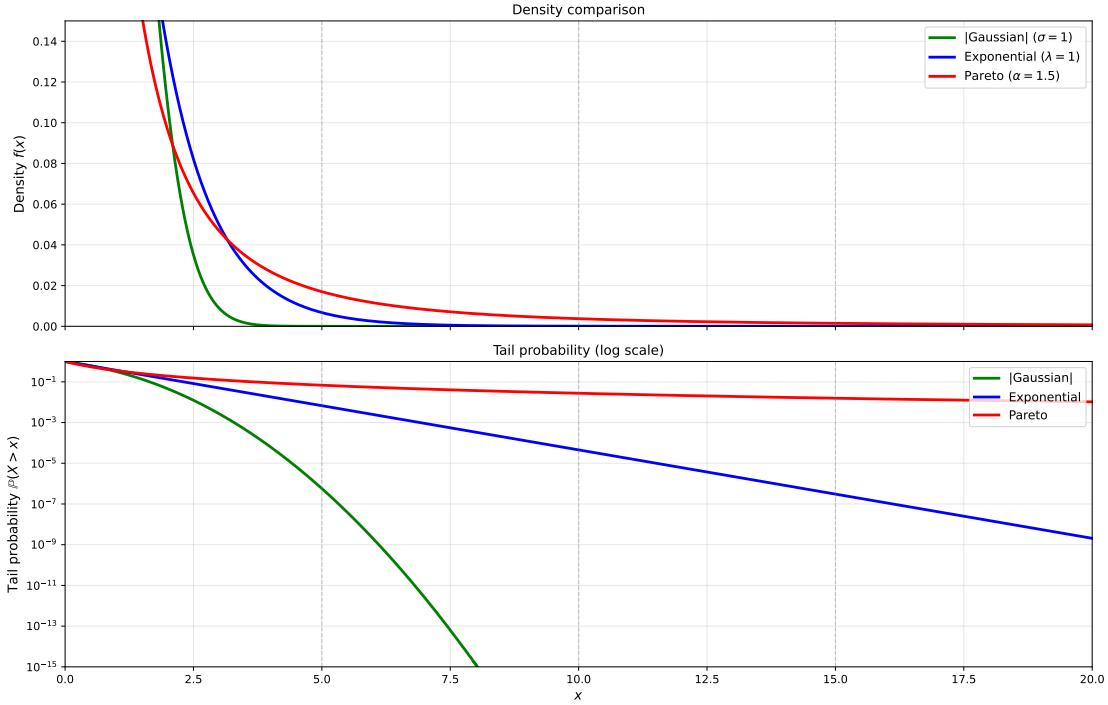


Figure 1: Light-tailed vs heavy-tailed distributions. Left: density comparison for $|Gaussian|$, exponential, and Pareto. Right: tail probability $\mathbb{P}(X > t)$ on log scale. The Gaussian and exponential tails decay rapidly; the Pareto tail decays as a power law, remaining substantial even for large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

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

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

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

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

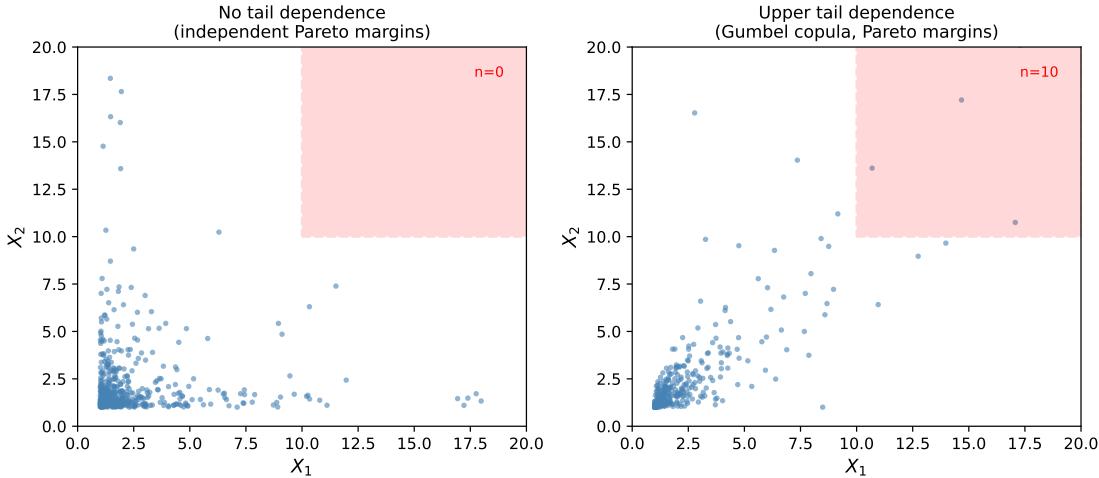


Figure 2: Tail dependence illustrated with Pareto margins ($\alpha = 1.5$). Left: independent margins; extreme values of X_1 and X_2 occur independently, so the upper tail region (red box) is sparse. Right: Gumbel copula ($\theta = 2.5$) with upper tail dependence; when X_1 is extreme, X_2 tends to be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

333 is defined by

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

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

339 0.2.2 Order Statistics and Ranking Constraints

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

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

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

$$f_{1:n, \dots, n:n}(x_1, \dots, x_n) = n! \prod_{i=1}^n f(x_i) \cdot \mathbf{1}\{x_1 \leq \dots \leq x_n\}. \quad (2)$$

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

360 0.2.3 Reward-Tilted Distributions and Model Fine-Tuning

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

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

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

$$p(x) \propto p_0(x) \exp(\lambda r(x)), \quad (3)$$

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

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

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

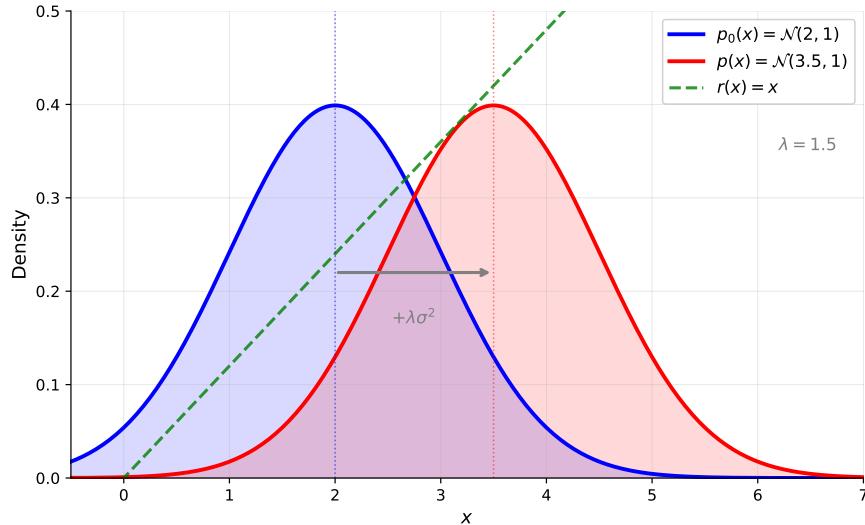


Figure 3: Reward tilting shifts the distribution toward high-reward regions. With $p_0 = \mathcal{N}(2, 1)$, linear reward $r(x) = x$, and tilting strength $\lambda = 1.5$, the tilted distribution $p(x) \propto p_0(x) \exp(\lambda r(x))$ is $\mathcal{N}(3.5, 1)$: the mean shifts by $\lambda\sigma^2 =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

381 0.3 Generative Modeling Background

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

388 0.3.1 The Problem of Generative Modeling

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

393 This problem has a long history in statistics and probability. Classical approaches
394 include *density estimation* (Silverman 1986; Scott 2015), where one estimates the density
395 $p = dP/d\lambda$ and then samples from the estimated density, and *Markov Chain Monte*
396 *Carlo* (MCMC) (Metropolis et al. 1953; Hastings 1970; C. P. Robert and Casella 2004),
397 where one constructs a Markov chain with P as its stationary distribution and samples
398 by running the chain to equilibrium. Both approaches face the curse of dimensionality
399 (Bellman 1961): density estimation becomes statistically intractable in high dimensions,
400 while MCMC mixing times can grow exponentially with dimension.

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

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

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

416 0.3.2 Neural Networks

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

$$419 \quad h_\ell = \sigma(W_\ell h_{\ell-1} + b_\ell) \quad \text{for } \ell = 1, 2, \dots, L, \quad (4)$$

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

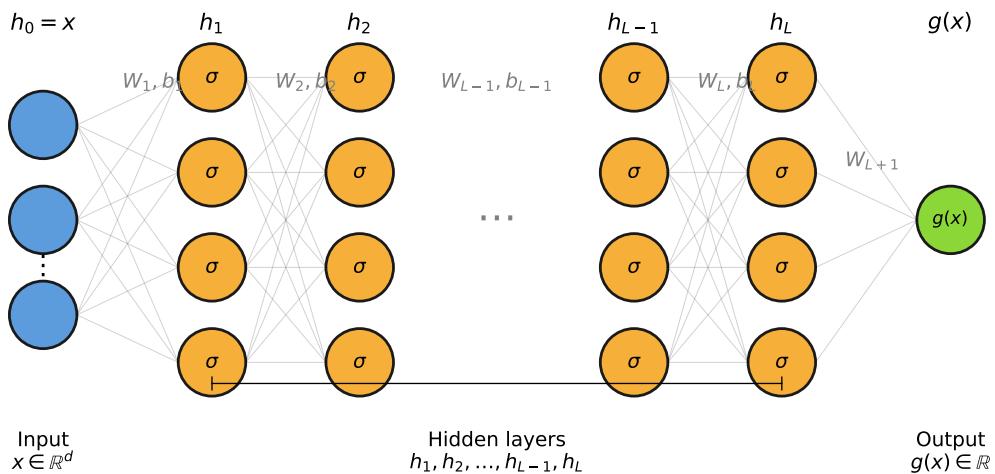


Figure 4: A feedforward neural network. The input $h_0 = x$ passes through L hidden layers, each applying an affine transformation followed by activation σ . The output $g(x)$ is a linear function of h_L .

422 0.3.3 Neural Network Approximation Theory

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

427 **Quantitative bounds.** Quantitative bounds relate approximation error to network size
 428 (Barron 1993; Yarotsky 2017). For Lipschitz functions on $[0, 1]^d$, achieving error ε requires
 429 $O(\varepsilon^{-d})$ parameters, exhibiting the curse of dimensionality. For smoother functions (Hölder
 430 class with smoothness s), networks with $O(\varepsilon^{-d/s})$ parameters suffice (Yarotsky 2017).

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

437 0.3.4 Generative Adversarial Networks

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

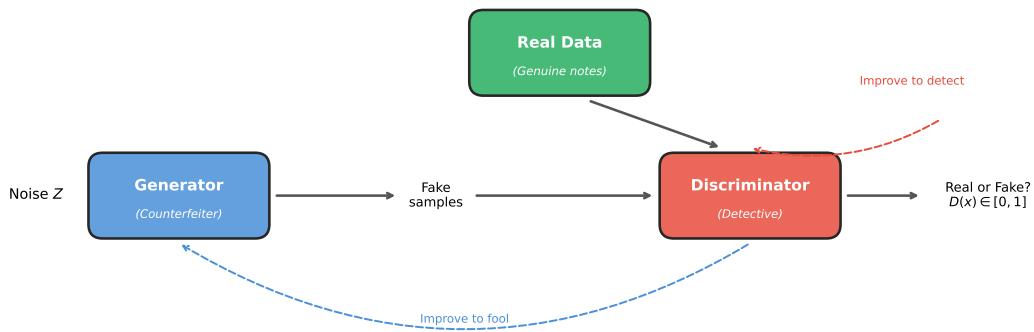


Figure 5: The GAN framework as an adversarial game. The generator (counterfeiter) transforms noise into fake samples; the discriminator (detective) tries to distinguish fakes from real data. Both networks improve through competition until equilibrium, when generated samples are indistinguishable from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

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

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

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

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

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

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

458 0.3.5 Diffusion and Score-Based Models

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

$$X_t = \alpha_t X_0 + \sigma_t X_1, \quad X_0 \sim p_0, \quad X_1 \sim N(0, I_d), \quad (5)$$

463 where (α_t, σ_t) are deterministic schedules with $(\alpha_0, \sigma_0) = (1, 0)$ and $(\alpha_1, \sigma_1) = (0, 1)$. At
464 $t = 0$, we have clean data; at $t = 1$, pure noise. Standard choices include the variance-
465 preserving schedule ($\alpha_t^2 + \sigma_t^2 = 1$) and the linear schedule ($\alpha_t = 1 - t$, $\sigma_t = t$).

466 To generate samples, we reverse this process. The key quantity is the *score function*
467 $\nabla_x \log p_t$, which points toward regions of high density at each noise level. For the Gaussian
468 forward kernel, the score satisfies $\nabla_x \log p_t(x_t) = -\hat{x}_1(x_t, t)/\sigma_t$, where $\hat{x}_1(x_t, t) = \mathbb{E}[X_1 |$
469 $X_t = x_t]$ is the conditional expectation of the noise given the noised sample.

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

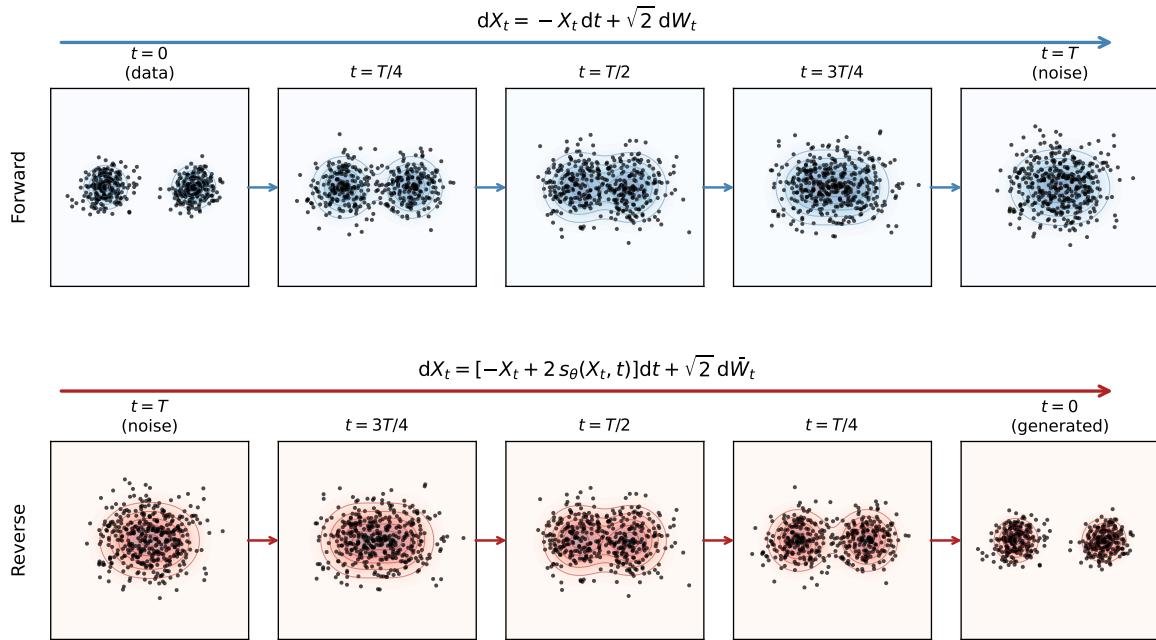


Figure 6: The diffusion model framework on a 2D Gaussian mixture. Forward process (top row): data from p_0 is progressively corrupted by adding noise until it becomes indistinguishable from the Gaussian reference p_1 at $t = 1$. Reverse process (bottom row): starting from pure noise, a learned score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

474 0.4 State of the Art and Open Problems

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

477 0.4.1 Extreme Value Generation

478 **Classical extreme value theory.** The statistical theory of multivariate extremes is
 479 well-developed (Paul Embrechts et al. 1997; L. d. Haan and Ferreira 2006; Sidney I. Resnick
 480 1987). For univariate extremes, the Fisher-Tippett-Gnedenko theorem establishes that
 481 properly normalized maxima converge to one of three distributions: Gumbel, Fréchet, or
 482 Weibull, unified by the Generalized Extreme Value (GEV) distribution. In the multivari-
 483 ate setting, the stable tail dependence function (stdf) provides a complete characterization
 484 of extremal dependence, describing how the probability of joint exceedances scales as one
 485 moves into the tail. Numerous parametric models have been proposed: the Gumbel (logis-
 486 tic) copula (Gumbel 1960), the Hüsler-Reiss model arising from Gaussian processes (Hüsler
 487 and Reiss 1989a), and the extremal- t model generalizing the Hüsler-Reiss to Student- t
 488 margins (Nikoloulopoulos et al. 2009). Estimation methods include empirical likelihood
 489 (Einmahl, Krajina, et al. 2012), rank-based M-estimation (Einmahl, L. d. Haan, et al.
 490 2016), and Bayesian approaches (Sabourin et al. 2013).

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

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

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

514 0.4.2 Order Statistics Generation

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

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

526 **Generative models and ordering.** Standard generative models (GANs, VAEs, diffu-
527 sion) produce unordered samples. When applied to ranked data, they face two problems.

528 First, generated samples are not sorted: a GAN trained on order statistics will produce
 529 vectors where coordinates are not monotonically ordered. Post-hoc sorting can restore
 530 order but introduces bias in the distribution. Second, standard preprocessing destroys
 531 ordering structure: per-coordinate normalization to $[0, 1]$, common in GAN training, nor-
 532 malizes each coordinate independently based on its marginal range, destroying the relative
 533 ordering. If the k -th order statistic is normalized by its own mean and variance, and the
 534 $(k + 1)$ -th by different values, there is no guarantee that normalized values remain sorted.

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

540 0.4.3 Reward-Tilted Generation

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

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

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

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

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

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

573 0.4.4 Objectives of This Thesis

574 Based on the identified gaps, we formulate three objectives:

- 575 1. **Extreme value generation.** Develop a generative model for multivariate extremes
 576 with provable guarantees on stdf approximation. The method should use heavy-
 577 tailed latent noise and provide explicit bounds on the approximation error in terms
 578 of network architecture.
- 579 2. **Order statistics generation.** Develop a generative model for order statistics that
 580 guarantees sortedness while matching the target distribution. The method should ex-
 581 ploit classical representations and provide approximation bounds that scale favorably
 582 with dimension.
- 583 3. **Reward-tilted generation.** Develop a gradient-free method for fine-tuning diffu-
 584 sion models to reward-tilted distributions. The method should require only forward
 585 evaluations of the reward function.

586 0.5 Contributions of This Thesis

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

588 0.5.1 Heavy-Tailed GANs for Extreme Value Generation

589 We develop Heavy-Tailed GANs (HTGAN) for generating data with correct tail depen-
 590 dence.

591 **Theoretical contributions.** We replace Gaussian latent noise with heavy-tailed (Pareto
 592 or Fréchet) noise. Because the generator is Lipschitz, the heavy-tail property propagates
 593 to the output. We prove that the stdf of the generated distribution is always discrete, a
 594 maximum of linear functions with finitely many atoms. Our main approximation theorem
 595 establishes that any stdf can be approximated within L^∞ error ε using latent dimension
 596 $N = O(\varepsilon^{-(d-1)})$ and a single hidden layer with ReLU activations. This appears to be the
 597 first approximation guarantee for tail dependence in the generative modeling literature.

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

601 **Experimental contributions.** We evaluate on synthetic Gumbel copula data and S&P
 602 500 returns, demonstrating that HTGAN correctly captures tail dependence while standard
 603 GANs fail by orders of magnitude.

604 0.5.2 Generative Neural Order Statistics

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

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

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

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

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

614 *Sukhatme's representation* (Sukhatme 1937) expresses uniform order statistics via ex-
 615 ponential spacings:

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

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

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

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

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

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

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

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

630 0.5.3 Gradient-Free Fine-Tuning of Diffusion Models

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

633 **Theoretical contributions.** Fisher’s identity expresses the score of the tilted distribu-
 634 tion as a covariance:

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

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

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

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

643 0.6 Organization of the Manuscript

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

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

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

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

661 **Introduction (Version 662 française)**

663 **0.7 Contexte et motivation**

664 **0.7.1 La structure dans les données statistiques**

665 Le terme *statistique* dérive de l'allemand *Statistik*, forgé par Gottfried Achenwall en 1748
666 pour désigner la science de l'État : gouverner nécessitait une information organisée. Les
667 données sont la matière première de la statistique. Un scientifique mesure la température
668 en plusieurs endroits ; une banque enregistre les rendements journaliers des actifs d'un por-
669 tefeuille ; un hôpital collecte les signes vitaux des patients. Dans chaque cas, nous disposons
670 de plusieurs observations, et chaque observation peut comprendre plusieurs quantités liées
671 entre elles. Ces quantités sont organisées de manières spécifiques, régies par des régularités
672 sous-jacentes. Découvrir ces régularités est la tâche de la science statistique.

673 Ce processus admet une formulation mathématique claire. Nous notons n observations
674 par X_1, \dots, X_n . Chaque X_i est un vecteur : une liste de nombres représentant les quantités
675 mesurées. Nous supposons que ces observations sont gouvernées par une loi sous-jacente.
676 Mathématiquement, cette loi est une *distribution de probabilité*, que nous notons P . La
677 distribution spécifie la probabilité de chaque résultat possible. Elle est inconnue ; apprendre
678 à la connaître est le problème central.

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

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

687 **0.7.2 La modélisation générative moderne**

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

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

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

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

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

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

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

710 0.7.3 Le problème : quand la structure compte

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

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

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

723 0.7.4 Cette thèse : méthodes génératives guidées par la théorie

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

⁷²⁷ peuvent l'approximer. C'est la méthodologie centrale : du problème appliqué, à l'abstraction
⁷²⁸ mathématique, à l'analyse théorique.

⁷²⁹ Nous étudions trois régimes structurels : la dépendance de queue, les statistiques d'ordre
⁷³⁰ et l'inclinaison par récompense. Pour chacun, nous concevons des méthodes génératives qui
⁷³¹ respectent *prouvablement* la structure, fournissant des garanties sur la précision d'approxi-
⁷³² mation.

⁷³³ 0.8 Trois régimes structurels

⁷³⁴ Nous décrivons maintenant les trois régimes structurels abordés dans cette thèse. Pour
⁷³⁵ chaque régime, nous présentons le problème réel qui motive notre travail et le formalisons
⁷³⁶ mathématiquement.

⁷³⁷ 0.8.1 Valeurs extrêmes et dépendance de queue

⁷³⁸ **Le problème : quand les extrêmes gouvernent tout.** Dans de nombreux domaines,
⁷³⁹ les événements extrêmes dominent les résultats malgré leur rareté. En gestion de por-
⁷⁴⁰ tefeuille, une poignée de grandes pertes peut effacer des années de gains réguliers : la
⁷⁴¹ distribution des rendements est à queue lourde, et le risque est concentré dans les extrêmes
⁷⁴² (Rama CONT 2001). La Value-at-Risk, les tests de stress et les exigences réglementaires en
⁷⁴³ capital reposent tous sur la probabilité de pertes rares mais catastrophiques. La crise fi-
⁷⁴⁴ nancière de 2008 a démontré que les modèles calibrés sur des conditions normales échouent
⁷⁴⁵ lorsque des extrêmes surviennent (A. J. MCNEIL et al. 2015).

⁷⁴⁶ En assurance, les événements extrêmes sont le cur de métier. Les réassureurs doivent
⁷⁴⁷ tarifer le risque de catastrophes simultanées : un ouragan frappant la Floride pendant
⁷⁴⁸ qu'un tremblement de terre frappe la Californie. C'est l'occurrence conjointe de tels évé-
⁷⁴⁹ nements, et non leurs probabilités individuelles, qui détermine les exigences de solvabilité.
⁷⁵⁰ En science du climat, les extrêmes composés (vagues de chaleur et sécheresses simultanées,
⁷⁵¹ événements d'inondation séquentiels) posent les plus grands risques aux écosystèmes et aux
⁷⁵² infrastructures (Jakob ZSCHEISCHLER et al. 2018 ; BEVACQUA et al. 2021). Les modèles
⁷⁵³ climatiques doivent capturer non seulement la fréquence des extrêmes individuels, mais
⁷⁵⁴ leur tendance à se regrouper dans l'espace et le temps.

⁷⁵⁵ Le fil conducteur est la *dépendance de queue* : la tendance des extrêmes à survenir
⁷⁵⁶ ensemble. Deux variables aléatoires peuvent avoir une corrélation nulle dans des conditions
⁷⁵⁷ normales tout en exhibant une forte dépendance de queue : quand l'une prend une valeur
⁷⁵⁸ extrême, l'autre a tendance à en faire autant (voir la Figure 8). Pour un gestionnaire de
⁷⁵⁹ portefeuille, cela signifie que la diversification échoue précisément quand elle est le plus
⁷⁶⁰ nécessaire. Pour un assureur, cela signifie des sinistres corrélés. Pour le risque climatique,
⁷⁶¹ cela signifie des catastrophes composées.

⁷⁶² **Deux défis distincts.** Générer des données de valeurs extrêmes réalistes pose deux défis
⁷⁶³ fondamentalement différents.

764 Le premier est la *lourdeur marginale des queues*. Une distribution est à queue lourde
 765 quand les valeurs extrêmes sont bien plus probables que ce que les modèles à queue légère
 766 prédiraient. La Figure 7 compare trois distributions : la gaussienne absolue $|X|$, l'expo-
 767 nentielle (toutes deux à queue légère) et la Pareto (à queue lourde). Près de zéro, elles se
 768 ressemblent, mais leurs queues se comportent très différemment. Le Tableau 2 montre la
 769 probabilité de dépasser différents seuils. À $t = 8$, la gaussienne attribue une probabilité de
 770 10^{-15} , l'exponentielle 10^{-4} , mais la Pareto donne encore 3,7%. Les phénomènes à queue
 771 lourde produisent des événements extrêmes que les modèles à queue légère considèrent
 772 comme virtuellement impossibles.

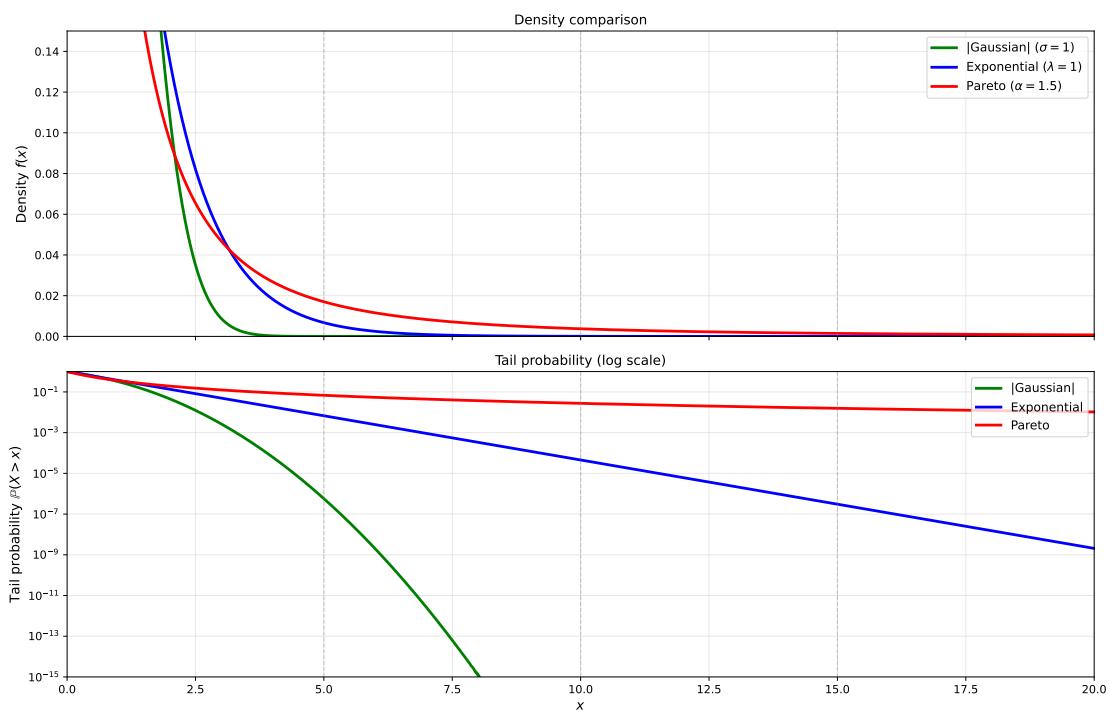


FIGURE 7 : Distributions à queue légère vs à queue lourde. Gauche : comparaison des densités pour |Gaussienne|, exponentielle et Pareto. Droite : probabilité de queue $\mathbb{P}(X > t)$ en échelle logarithmique. Les queues gaussienne et exponentielle décroissent rapidement ; la queue de Pareto décroît 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	$1,3 \times 10^{-15}$	$3,4 \times 10^{-4}$	$3,7 \times 10^{-2}$

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

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

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

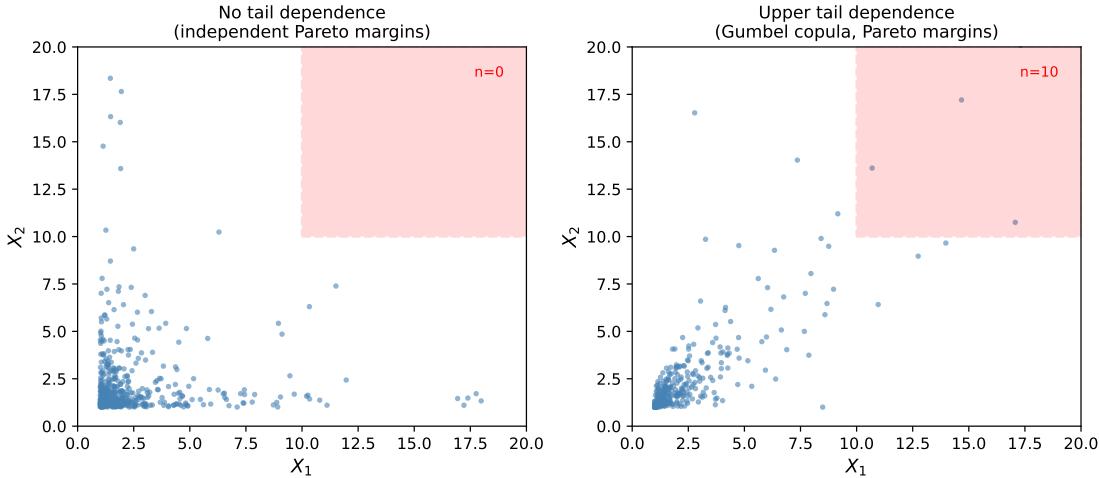


FIGURE 8 : Dépendance de queue illustrée avec des marges de Pareto ($\alpha = 1,5$). Gauche : marges indépendantes ; les valeurs extrêmes de X_1 et X_2 surviennent indépendamment, donc la région de queue supérieure (boîte rouge) est clairsemée. Droite : copule de Gumbel ($\theta = 2,5$) avec dépendance 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.

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

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

$$\ell(\mathbf{x}) = \lim_{t \rightarrow \infty} t \mathbb{P} \left(\bigcup_{j=1}^d \{X_j > tx_j\} \right). \quad (10)$$

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

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

799 0.8.2 Statistiques d'ordre et contraintes de classement

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

807 Considérons l'investissement à impact, où les actifs sont classés par scores environne-
 808 mentaux, sociaux ou de gouvernance (ESG) (LO et R. ZHANG 2021). Un gestionnaire de
 809 fonds testant rétrospectivement une stratégie ESG doit simuler les rendements du meilleur
 810 actif classé, du deuxième meilleur, et ainsi de suite. Ce sont des *statistiques d'ordre induites* : les rendements $\Theta_{[1:N]}, \dots, \Theta_{[N:N]}$ associés aux actifs classés par leurs scores ESG
 812 $X_{1:N} \leq \dots \leq X_{N:N}$. Les données synthétiques doivent préserver à la fois la distribution
 813 marginale à chaque rang et la dépendance conjointe entre les rangs.

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

$$f_{1:n, \dots, n:n}(x_1, \dots, x_n) = n! \prod_{i=1}^n f(x_i) \cdot \mathbf{1}\{x_1 \leq \dots \leq x_n\}. \quad (11)$$

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

824 0.8.3 Distributions inclinées par récompense et ajustement fin 825 de modèles

826 **Le problème : aligner les modèles génératifs avec les préférences humaines.**
 827 Un modèle génératif pré-entraîné produit des échantillons d'une distribution apprise à
 828 partir des données d'entraînement. Mais pour de nombreuses applications, nous voulons
 829 des échantillons qui ne sont pas simplement typiques, mais *bons* selon un certain critère :
 830 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 $r : \mathcal{X} \rightarrow \mathbb{R}$, la cible est la *distribution inclinée*

$$p(x) \propto p_0(x) \exp(\lambda r(x)), \quad (12)$$

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

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

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 .

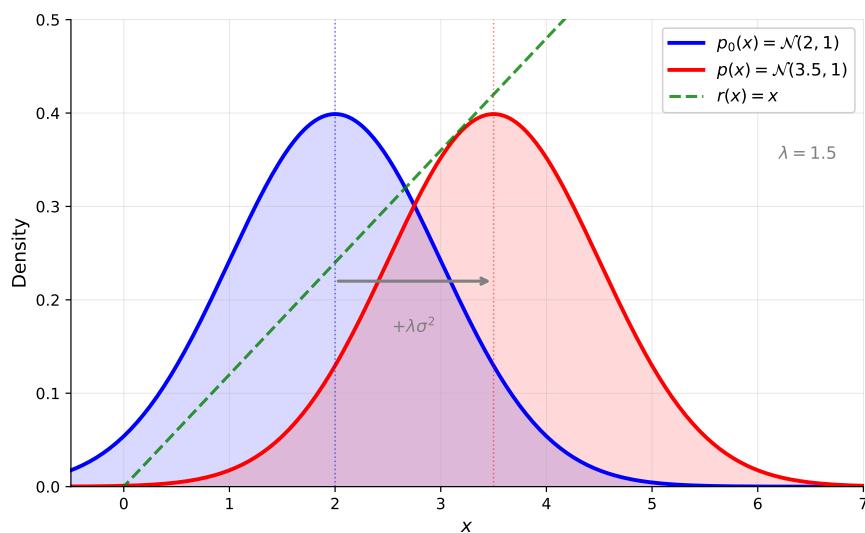


FIGURE 9 : L'inclinaison par récompense déplace la distribution vers les régions à haute récompense. Avec $p_0 = \mathcal{N}(2, 1)$, récompense linéaire $r(x) = x$, et intensité d'inclinaison $\lambda = 1,5$, la distribution inclinée $p(x) \propto p_0(x) \exp(\lambda r(x))$ est $\mathcal{N}(3,5, 1)$: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

849 0.9 Contexte de la modélisation générative

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

857 0.9.1 Le problème de la modélisation générative

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

862 Ce problème a une longue histoire en statistique et probabilités. Les approches clas-
 863 siques incluent l'*estimation de densité* (SILVERMAN 1986 ; SCOTT 2015), où l'on estime la
 864 densité $p = dP/d\lambda$ puis on échantillonne à partir de la densité estimée, et les *chaînes de*
 865 *Markov Monte Carlo* (MCMC) (METROPOLIS et al. 1953 ; HASTINGS 1970 ; C. P. ROBERT
 866 et CASELLA 2004), où l'on construit une chaîne de Markov ayant P comme distribution
 867 stationnaire et on échantillonne en faisant tourner la chaîne jusqu'à l'équilibre. Les deux
 868 approches font face à la malédiction de la dimension (BELLMAN 1961) : l'estimation de
 869 densité devient statistiquement intractable en haute dimension, tandis que les temps de
 870 mélange MCMC peuvent croître exponentiellement avec la dimension.

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

878 Cette vision basée sur le transport unifie plusieurs paradigmes génératifs modernes.
 879 Dans les *flocs normalisants* (REZENDE et MOHAMED 2015 ; DINH et al. 2017), l'appli-
 880 cation G est contrainte à être inversible avec un jacobien tractable. Dans les *autoencodeurs*
 881 *variationnels* (VAE) (Diederik P. KINGMA et Max WELLING 2014), l'application est ap-
 882 prise conjointement avec une inverse approximative. Dans les *réseaux antagonistes génér-
 883 ratifs* (GAN) (I. J. GOODFELLOW et al. 2014), l'application est apprise via un objectif
 884 adversarial. Dans les *modèles de diffusion* (HO, JAIN et al. 2020 ; Y. SONG, Jascha SOHL-
 885 DICKSTEIN et al. 2021), l'application est implicitement définie par l'inverse d'un processus
 886 de bruitage. Nous décrivons les GAN et les modèles de diffusion en détail ci-dessous, car
 887 ils sont les fondements de nos contributions.

888 0.9.2 Réseaux de neurones

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

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

891 où W_ℓ est une matrice de poids, b_ℓ un vecteur de biais, et σ une fonction d'activation non
 892 linéaire appliquée coordonnée par coordonnée. La sortie est $g(x) = W_{L+1}h_L + b_{L+1}$. Les
 893 activations courantes incluent la sigmoïde, la tangente hyperbolique et la ReLU $\sigma(t) =$
 894 $\max(0, t)$, qui domine la pratique moderne.

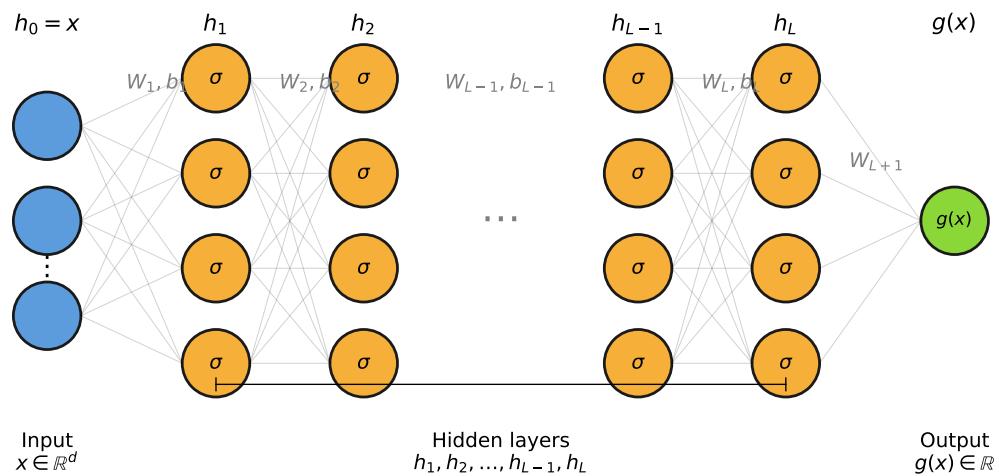


FIGURE 10 : Un réseau de neurones feedforward. L'entrée $h_0 = x$ passe à travers L couches cachées, chacune appliquant une transformation affine suivie de l'activation σ . La sortie $g(x)$ est une fonction linéaire de h_L .

895 0.9.3 Théorie d'approximation des réseaux de neurones

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

900 **Bornes quantitatives.** Les bornes quantitatives relient l'erreur d'approximation à la
 901 taille du réseau (BARRON 1993; YAROTSKY 2017). Pour les fonctions lipschitziennes sur
 902 $[0, 1]^d$, atteindre une erreur ε nécessite $O(\varepsilon^{-d})$ paramètres, exhibant la malédiction de
 903 la dimension. Pour des fonctions plus régulières (classe de Hölder avec régularité s), des
 904 réseaux avec $O(\varepsilon^{-d/s})$ paramètres suffisent (YAROTSKY 2017).

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

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

912 0.9.4 Réseaux antagonistes génératifs

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

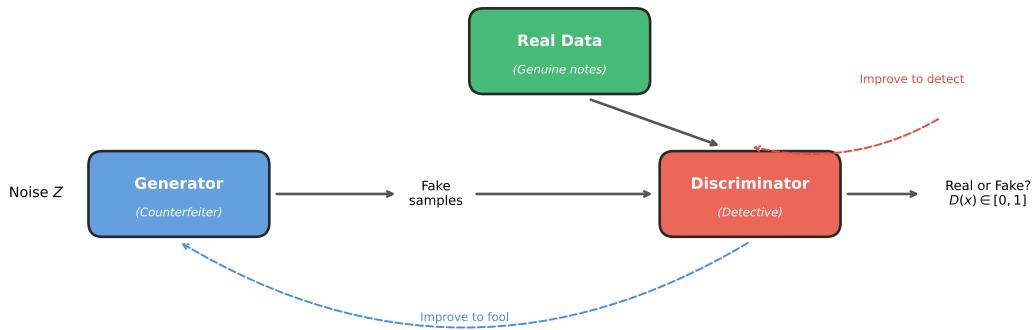


FIGURE 11 : Le cadre GAN comme jeu adversarial. Le générateur (faussaire) transforme du bruit en faux échantillons ; le discriminateur (détective) tente de distinguer les faux des vraies données. Les deux réseaux s'améliorent par compétition jusqu'à l'équilibre, quand les échantillons générés sont indiscernables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

921 En termes mathématiques, un *générateur* $G_\theta : \mathcal{Z} \rightarrow \mathcal{X}$ transforme un bruit latent
 922 $Z \sim p_Z$ en échantillons synthétiques (la production du faussaire). Un *discriminateur* $D_\phi : \mathcal{X} \rightarrow [0, 1]$:
 923 tente de distinguer les échantillons réels des générés (le verdict du détective).
 924 Les réseaux sont entraînés en résolvant un problème minimax :

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

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

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

930 Jensen-Shannon par la distance de Wasserstein-1 :

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

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

934 0.9.5 Modèles de diffusion et basés sur le score

935 Les modèles de diffusion (HO, JAIN et al. 2020 ; Y. SONG, Jascha SOHL-DICKSTEIN et al.
936 2021) définissent un processus génératif en construisant un chemin continu $(p_t)_{t \in [0,1]}$ entre
937 la distribution des données p_0 et une référence tractable $p_1 = N(0, I_d)$. Le chemin est défini
938 via l'*interpolation directe*

$$X_t = \alpha_t X_0 + \sigma_t X_1, \quad X_0 \sim p_0, \quad X_1 \sim N(0, I_d), \quad (14)$$

939 où (α_t, σ_t) sont des programmes déterministes avec $(\alpha_0, \sigma_0) = (1, 0)$ et $(\alpha_1, \sigma_1) = (0, 1)$. À
940 $t = 0$, nous avons des données propres ; à $t = 1$, du pur bruit. Les choix standards incluent
941 le programme préservant la variance ($\alpha_t^2 + \sigma_t^2 = 1$) et le programme linéaire ($\alpha_t = 1 - t$,
942 $\sigma_t = t$).

943 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
944 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
945 $\hat{x}_1(x_t, t) = \mathbb{E}[X_1 | X_t = x_t]$ est l'espérance conditionnelle du bruit sachant l'échantillon
946 bruité.

947 Le score est appris via l'*appariement de score par débruitage* (HYVÄRINEN 2005 ; VINCENT
948 2011) : un réseau $s_{\theta}(x, t)$ est entraîné à prédire le bruit X_1 à partir de l'échantillon bruité
949 X_t , ce qui équivaut à apprendre $\nabla_x \log p_t$. En partant de $X_1 \sim N(0, I_d)$ et en débruitant
950 itérativement en utilisant le score appris, on obtient des échantillons de p_0 .

952 0.10 État de l'art et problèmes ouverts

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

955 0.10.1 Génération de valeurs extrêmes

956 **Théorie classique des valeurs extrêmes.** La théorie statistique des extrêmes multi-
957 variés est bien développée (Paul EMBRECHTS et al. 1997 ; L. d. HAAN et FERREIRA 2006 ;
958 Sidney I. RESNICK 1987). Pour les extrêmes univariés, le théorème de Fisher-Tippett-
959 Gnedenko établit que les maxima correctement normalisés convergent vers l'une des trois

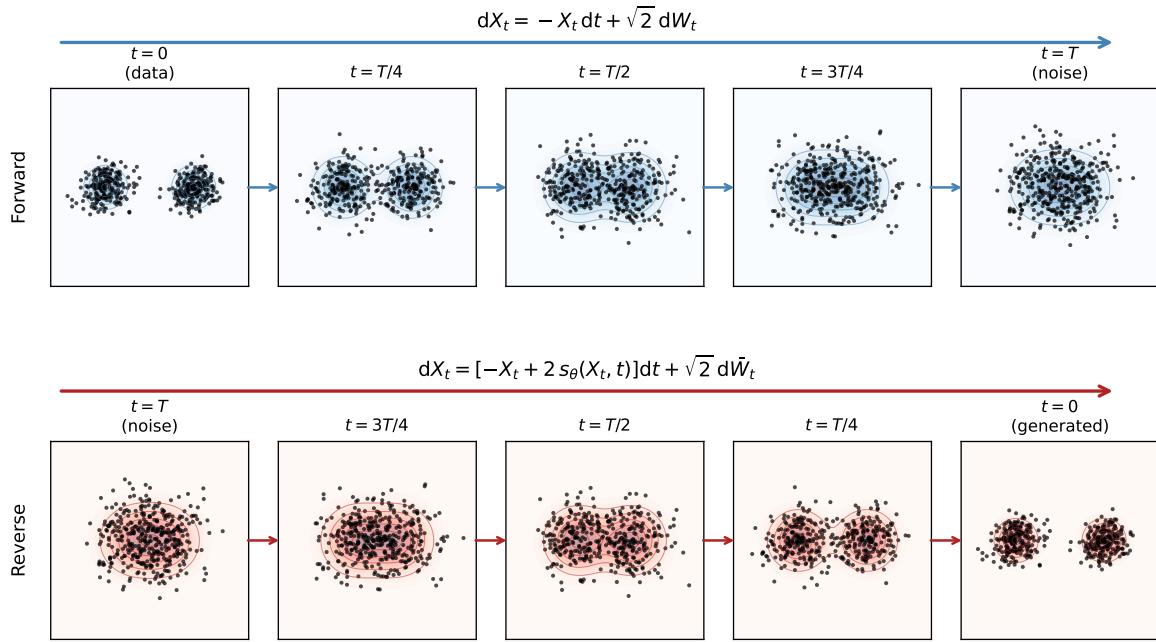


FIGURE 12 : Le cadre des modèles de diffusion sur un mélange de gaussiennes 2D. Processus direct (ligne du haut) : les données de p_0 sont progressivement corrompues par ajout de bruit jusqu'à devenir indiscernables de la référence gaussienne p_1 à $t = 1$. Processus inverse (ligne du bas) : partant 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 (NIKOLOUPOULOS 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

1018 mais introduit un biais dans la distribution. Deuxièmement, le prétraitement standard dé-
1019 truit la structure d'ordonnancement : la normalisation par coordonnée vers $[0, 1]$, courante
1020 dans l'entraînement des GAN, normalise chaque coordonnée indépendamment selon son
1021 étendue marginale, détruisant l'ordonnancement relatif. Si la k -ième statistique d'ordre est
1022 normalisée par sa propre moyenne et variance, et la $(k+1)$ -ième par des valeurs différentes,
1023 il n'y a aucune garantie que les valeurs normalisées restent triées.

1024 **Lacune 2.** À notre connaissance, aucun modèle génératif existant ne produit des sta-
1025 tistiques d'ordre avec à la fois les distributions marginales/conjointes correctes et des
1026 contraintes d'ordonnancement garanties. Les méthodes de tri différentiable existantes traitent
1027 la prédiction de classement, pas la modélisation générative. Nous comblons cette lacune
1028 dans le Chapitre 2, en développant une architecture qui exploite les représentations clas-
1029 siques pour garantir l'ordonnancement tout en appariant les distributions cibles.

1030 0.10.3 Génération inclinée par récompense

1031 Le problème d'alignement (Section 0.8.3) requiert d'échantillonner à partir de la distribu-
1032 tion inclinée (12). Nous passons en revue les approches existantes.

1033 **Apprentissage par renforcement à partir de feedback humain.** Le paradigme
1034 dominant pour l'alignement est l'apprentissage par renforcement à partir de feedback hu-
1035 main (RLHF) (CHRISTIANO et al. 2017; BAI et al. 2022; OUYANG et al. 2022). Un mo-
1036 dèle de récompense est entraîné sur des données de préférence humaine (comparaisons par
1037 paires de sorties), puis le modèle génératif est affiné pour maximiser la récompense espé-
1038 rée en utilisant des méthodes de gradient de politique (PPO, REINFORCE). Le RLHF
1039 a été transformateur pour les grands modèles de langage, permettant ChatGPT, Claude
1040 et d'autres agents conversationnels. Cependant, le RLHF est notoirement instable : le pi-
1041 ratage de récompense, l'effondrement de mode et l'oubli catastrophique sont des modes
1042 d'échec courants (CASPER et al. 2023).

1043 **Ajustement fin des modèles de diffusion.** Pour les modèles de diffusion, plusieurs
1044 approches d'ajustement fin ont émergé. DRaFT (CLARK et al. 2024) rétropropage à travers
1045 toute la chaîne de débruitage pour maximiser la récompense, mais cela nécessite de stocker
1046 les activations à travers tous les pas de temps, avec une mémoire croissant linéairement avec
1047 la longueur de la chaîne. DPOK (FAN et al. 2023) combine les gradients de politique avec
1048 la régularisation KL. ReFL (Jing XU et al. 2024) utilise l'apprentissage par feedback de
1049 récompense pour affiner les modèles de diffusion texte-vers-image. L'appariement adjoint
1050 (DOMINGO-ENRICH, DROZDZAL et al. 2025) fournit une approche théoriquement fondée
1051 basée sur les EDS adjointes, atteignant des résultats état de l'art mais nécessitant toujours
1052 les gradients de récompense.

1053 **Le goulot d'étranglement du gradient.** Les méthodes d'ajustement fin basées sur le
1054 gradient (DRaFT, appariement adjoint) nécessitent de rétropropager à travers la fonction

1055 de récompense, ce qui crée des limitations pratiques : stocker les activations pour les grands
 1056 modèles de récompense mène à des coûts mémoire prohibitifs, et les récompenses non
 1057 différentiables (feedback humain, évaluations discrètes, métriques basées sur simulateur)
 1058 ne peuvent pas être traitées. Les approches basées sur l'apprentissage par renforcement
 1059 (RLHF, DPOK) évitent cela en utilisant uniquement les valeurs de récompense, pas les
 1060 gradients, mais introduisent d'autres défis : les estimateurs de gradient de politique ont
 1061 une haute variance, et un réglage fin des hyperparamètres est nécessaire pour équilibrer la
 1062 maximisation de récompense contre l'effondrement de mode. Aucune approche ne fournit
 1063 de garanties distributionnelles sur le modèle affiné résultant.

1064 **Lacune 3.** À notre connaissance, aucune méthode existante sans gradient n'affine les mo-
 1065 dèles de diffusion pour échantillonner à partir de distributions inclinées par récompense.
 1066 Les méthodes existantes nécessitent soit les gradients de récompense, soit utilisent l'ap-
 1067 prentissage par renforcement avec des estimateurs à haute variance. Nous comblons cette
 1068 lacune dans le Chapitre 3, en développant une méthode basée sur l'identité de Fisher qui
 1069 ne requiert que des évaluations directes de la fonction de récompense.

1070 0.10.4 Objectifs de cette thèse

1071 Sur la base des lacunes identifiées, nous formulons trois objectifs :

- 1072 1. **Génération de valeurs extrêmes.** Développer un modèle génératif pour les ex-
 1073 trêmes multivariés avec des garanties prouvables sur l'approximation de la stdf. La
 1074 méthode devrait utiliser un bruit latent à queue lourde et fournir des bornes explicites
 1075 sur l'erreur d'approximation en termes d'architecture de réseau.
- 1076 2. **Génération de statistiques d'ordre.** Développer un modèle génératif pour les
 1077 statistiques d'ordre qui garantit l'ordonnancement tout en appariant la distribution
 1078 cible. La méthode devrait exploiter les représentations classiques et fournir des bornes
 1079 d'approximation qui évoluent favorablement avec la dimension.
- 1080 3. **Génération inclinée par récompense.** Développer une méthode sans gradient
 1081 pour affiner les modèles de diffusion vers des distributions inclinées par récompense. La
 1082 méthode ne devrait requérir que des évaluations directes de la fonction de récompense.

1083 0.11 Contributions de cette thèse

1084 Nous résumons maintenant les contributions principales, organisées selon les trois objectifs.

1085 0.11.1 GAN à queue lourde pour la génération de valeurs ex- 1086 trêmes

1087 Nous développons les GAN à queue lourde (HTGAN) pour générer des données avec une
 1088 dépendance de queue correcte.

1089 **Contributions théoriques.** Nous remplaçons le bruit latent gaussien par un bruit à
1090 queue lourde (Pareto ou Fréchet). Comme le générateur est lipschitzien, la propriété de
1091 queue lourde se propage à la sortie. Nous prouvons que la stdf de la distribution générée
1092 est toujours discrète, un maximum de fonctions linéaires avec un nombre fini d'atomes.
1093 Notre théorème d'approximation principal établit que toute stdf peut être approximée
1094 dans l'erreur $L^\infty \varepsilon$ en utilisant une dimension latente $N = O(\varepsilon^{-(d-1)})$ et une seule couche
1095 cachée avec des activations ReLU. Cela semble être la première garantie d'approximation
1096 pour la dépendance de queue dans la littérature de modélisation générative.

1097 **Contributions méthodologiques.** L'algorithme HTGAN procède en trois étapes :
1098 transformation marginale vers Fréchet unitaire, entraînement GAN avec bruit latent de
1099 Pareto, et transformation inverse. Cette séparation des marges et de la dépendance est
1100 standard en modélisation par copules.

1101 **Contributions expérimentales.** Nous évaluons sur des données synthétiques de copule
1102 de Gumbel et des rendements du S&P 500, démontrant que HTGAN capture correctement
1103 la dépendance de queue tandis que les GAN standards échouent de plusieurs ordres de
1104 grandeur.

1105 0.11.2 Statistiques d'ordre neurales génératives

1106 Nous développons les statistiques d'ordre neurales génératives (GENOS) pour générer des
1107 données classées avec un ordonnancement garanti.

1108 **Contributions théoriques.** Nous combinons les représentations statistiques classiques
1109 avec la théorie d'approximation des réseaux de neurones (YAROTSKY 2017) pour dériver
1110 des bornes de complexité non asymptotiques. Spécifiquement, nous adaptons deux repré-
1111 sentations classiques des statistiques d'ordre et analysons leur complexité d'approximation
1112 par réseaux de neurones.

1113 *La représentation de Schucany* (SCHUCANY 1972) construit les statistiques d'ordre
1114 récursivement :

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

1115 Cela nécessite n fonctions de transition différentes. Nous prouvons que l'approximation
1116 neurale de ce schéma requiert un nombre de paramètres qui est polynomial de degré n en
1117 ε^{-1} .

1118 *La représentation de Sukhatme* (SUKHATME 1937) exprime les statistiques d'ordre uni-
1119 formes via des espacements exponentiels :

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

1120 La somme cumulative a une distribution exponentielle ; appliquer la fonction de réparti-
1121 tion exponentielle F donne la k -ième statistique d'ordre uniforme $U_{k:n}$. Cela ne requiert

₁₁₂₂ d'approximer que deux fonctions universelles (F et F^{-1}), indépendamment de n . Nous
₁₁₂₃ prouvons que l'approche basée sur Sukhatme atteint

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

₁₁₂₄ paramètres. La complexité est quadratique en ε^{-1} , indépendante de n , contre polynomiale
₁₁₂₅ de degré n pour Schucany. Cela rend l'approche basée sur Sukhatme scalable à de grandes
₁₁₂₆ tailles d'échantillon.

₁₁₂₇ **Contributions méthodologiques.** Nous introduisons la normalisation globale moyenne-
₁₁₂₈ échelle, qui préserve l'ordonnancement contrairement à la normalisation par coordonnée.
₁₁₂₉ Nous proposons également une pénalité de statistiques d'ordre (OSP) qui pénalise les vio-
₁₁₃₀ lations de la contrainte d'ordonnancement :

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

₁₁₃₁ **Contributions expérimentales.** Nous évaluons sur des statistiques d'ordre synthé-
₁₁₃₂ tiques avec dimension variable, comparant GAN, WGAN et WGAN avec OSP. L'ajout de
₁₁₃₃ la pénalité OSP améliore les métriques d'ordonnancement tout en maintenant des perfor-
₁₁₃₄ mances compétitives sur la distance de Wasserstein.

₁₁₃₅ 0.11.3 Ajustement fin sans gradient des modèles de diffusion

₁₁₃₆ Nous développons des méthodes sans gradient pour affiner les modèles de diffusion afin
₁₁₃₇ d'échantillonner à partir de distributions inclinées par récompense.

₁₁₃₈ **Contributions théoriques.** L'identité de Fisher exprime le score de la distribution
₁₁₃₉ inclinée comme une covariance :

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

₁₁₄₀ Cette covariance peut être estimée par Monte Carlo en utilisant uniquement des évaluations
₁₁₄₁ directes de r , sans rétropropagation à travers le modèle de récompense.

₁₁₄₂ **Contributions méthodologiques.** Nous proposons un algorithme itératif qui décom-
₁₁₄₃ pose une grande inclinaison en N petites étapes, estimant l'incrément de score à chaque
₁₁₄₄ étape via l'identité de Fisher. L'algorithme s'applique même lorsque r est non différentiable
₁₁₄₅ ou une boîte noire.

₁₁₄₆ **Contributions expérimentales.** Sur des mélanges de gaussiennes 2D avec vérité ter-
₁₁₄₇ rain connue, nous validons que la méthode récupère correctement le score incliné et que
₁₁₄₈ l'augmentation du nombre d'étapes N améliore la précision.

1149 0.12 Organisation du manuscrit

1150 Le Chapitre 1 développe la théorie et les algorithmes pour la modélisation générative à
1151 queue lourde. Nous présentons le contexte mathématique sur la théorie des valeurs extrêmes
1152 multivariées et la stdf, analysons pourquoi les applications lipschitziennes de bruit gaussien
1153 ne peuvent pas produire de queues lourdes, caractérisons la stdf discrète induite par les
1154 générateurs avec bruit de Pareto, prouvons notre théorème d'approximation principal et
1155 présentons des expériences sur des données synthétiques et financières.

1156 Le Chapitre 2 développe les modèles génératifs pour les statistiques d'ordre. Nous
1157 présentons le contexte sur les statistiques d'ordre et les représentations de Sukhatme et
1158 Schucany, prouvons des bornes d'approximation pour les réseaux ReLU, introduisons la
1159 normalisation globale moyenne-échelle et l'objectif WGAN-OSP, et présentons des expé-
1160 riences sur des statistiques d'ordre synthétiques.

1161 Le Chapitre 3 développe l'ajustement fin sans gradient pour les modèles de diffusion.
1162 Nous présentons le contexte sur l'inclinaison exponentielle et les méthodes existantes basées
1163 sur le gradient, dérivons l'estimateur de score basé sur l'identité de Fisher, analysons
1164 l'erreur d'approximation, présentons l'algorithme d'inclinaison itératif et validons sur des
1165 expériences contrôlées.

1166 Cette thèse est partiellement basée sur des travaux collaboratifs : le Chapitre 1 sur les
1167 modèles génératifs à queue lourde, le Chapitre 2 sur l'approximation neurale des statistiques
1168 d'ordre, et le Chapitre 3 sur l'ajustement fin sans gradient des modèles de diffusion. Les
1169 lieux de publication et co-auteurs spécifiques sont indiqués au début de chaque chapitre.

₁₁₇₀ **Chapter 1**

₁₁₇₁ **Heavy-Tailed GANs for**
₁₁₇₂ **Extreme Value Generation**

₁₁₇₃ **1.1 Introduction**

₁₁₇₄ **Statement of the problem.** Examining extreme events is a critical concern across var-
₁₁₇₅ ious fields such as economics, engineering, and life sciences, with wide-ranging applications
₁₁₇₆ like actuarial and financial risks (Asmussen and Albrecher [2010](#), Chapters X and XIII)-(P.
₁₁₇₇ Embrechts et al. [1997](#), Chapters 1 and 6)-(A. McNeil et al. [2015](#), Chapters 5 and 16, Part
₁₁₇₈ III), communication network reliability (P. Robert [2003](#)), and aircraft safety (Prandini and
₁₁₇₉ Watkins [2005](#)). Extreme events play a crucial role also in the context of climate change
₁₁₈₀ (J. Zscheischler et al. [2020](#)), with the occurrence of more and more severe weather events,
₁₁₈₁ or in the context of cyber-security (Cremer et al. [2022](#)) with the increasing number of
₁₁₈₂ cyber-attacks of private companies or public entities. In recent decades, the importance
₁₁₈₃ of extreme event analysis has surged in financial risk management, which calls for an
₁₁₈₄ ever-increasing number of types of risk (market, credit, operational, reputational, cyber,
₁₁₈₅ climate, and so on and so forth) to be encompassed in order to measure their impacts on
₁₁₈₆ banking activity and the stability of the financial system. In particular, regulators (Euro-
₁₁₈₇ pean Banking Authority [2014](#)) are increasingly imposing stress tests to test the resilience
₁₁₈₈ of the banking system against different risk categories. These stress tests involve numerical
₁₁₈₉ simulations of unfavorable yet plausible extreme scenarios, serving as a primary means to
₁₁₉₀ evaluate the potential impact on risks (see (ECB Banking Supervision [2023](#)) for the 2023
₁₁₉₁ exercises in Europe). Often, extreme events are described as tail events of multivariate
₁₁₉₂ relevant random variables.

₁₁₉₃ The challenge arises in efficiently obtaining relevant samples for assessing these tail risks.
₁₁₉₄ Informally, it writes as finding a generator G and a latent distribution μ_Z which one can
₁₁₉₅ easily sample from, such that:

$$Z \sim \mu_Z, \quad G(Z) \stackrel{d}{=} \text{target distribution.} \quad (1.1)$$

1196 A large family of numerical approaches relies on physics-based methods, and aims to
 1197 efficiently sample the distribution tail with minimal computational effort and to avoid
 1198 the inefficiency of a basic accept-rejection algorithm. These methods are specific to the
 1199 model at hand: among these methods, we mention importance sampling (Bucklew 2004,
 1200 Chapter 4), MCMC with splitting – (E. Gobet and G. Liu 2015), or interacting particles
 1201 system – (Del Moral and Garnier 2005). Another family of simulation schemes is data-
 1202 driven approaches which do not require the knowledge of the (supposedly existing) physical
 1203 model behind the sample generation. These data-driven schemes take as input a data
 1204 set of observations of given size n_{data} and design a generative model able to re-generate
 1205 samples that mimic the empirical distribution of the observed data. This concept aligns
 1206 with recent paradigms in Artificial Intelligence such as Generative Adversarial Networks
 1207 (GANs) initiated by (I. Goodfellow et al. 2014) and Variational Autoencoders (VAEs) by
 1208 (D. P. Kingma and M. Welling 2014), score-based diffusion models (J. Sohl-Dickstein et al.
 1209 2015), normalising flows (Papamakarios et al. 2021), etc.

1210 Our work is in this vein for generating extreme samples. Unlike the classical uses of
 1211 Deep Generative Models as mentioned above, where the number n_{data} of training data is
 1212 generally colossal (millions), in the context of extreme events, n_{data} is small by nature (a few
 1213 hundred at most). Standard training procedures are thus bound to have poor performance
 1214 all the more so as, by construction, their design makes them unable to reproduce heavy
 1215 and dependent tails (M. Allouche et al. 2022). The aim of this work is to design new
 1216 efficient procedures taking advantage of the probabilistic structure behind extreme values.

1217 **State of the art.** Very recently, the machine learning community became aware of the
 1218 difficulties of sampling extremes using Deep Generative Models, requiring to appropriately
 1219 adapt known generative methods to the extreme setting. Most of the newly designed
 1220 algorithms were based on GANs, see an overview in (Michaël Allouche, Emmanuel Gobet,
 1221 et al. 2024). Three directions have been mainly investigated in the literature. First, certain
 1222 works apply some preprocessing to the data to get rid of the tail heaviness, see Quant-GAN
 1223 (Wiese et al. 2020) and evtGAN (Y. Boulaguiem et al. 2022): essentially, it consists in
 1224 suitably transforming each data coordinate in order to retrieve either Gaussian or uniform
 1225 marginal distributions, then performing a usual GAN method and reverting to the original
 1226 space by inverse transform. A second direction is to consider new latent variables Z with a
 1227 heavy-tailed distribution. In (Feder et al. 2020; Huster et al. 2021), a Generalized Pareto
 1228 Distribution is adopted for the latent distribution μ_Z , in contrast to original GANs which
 1229 use light-tailed distributions such as uniform or Gaussian. This idea finds its origins in a
 1230 crucial observation: neural networks are Lipschitz functions (Wiese et al. 2020). As such,
 1231 they cannot transform a random variable with light-tailed to a distribution with heavier
 1232 tail, this is the main result of (Liang et al. 2022). Using heavy-tailed noise however raises
 1233 an issue with the training procedure: if the underlying distribution is too heavy-tailed,
 1234 the gradient of the loss function may not exist. To alleviate this, in (Huster et al. 2021),
 1235 the authors introduce a loss function designed to produce stable training. To be effective,
 1236 these methods require to estimate accurately the tail-index parameters for each marginal

1237 distribution, which is a difficult task because the number n_{data} of training data is usually
1238 small. In (Huster et al. 2021), the authors provide empirical evidence that the Pareto-
1239 GAN method is able to match heavy-tailed distributions embedded in lower dimensional
1240 manifolds in a high dimensional setup. However, they do not provide a quantification of
1241 the power of Pareto-GAN to reproduce asymptotic dependence in the data, which is our
1242 main scope of interest in this paper.

1243 A third approach consists in suitably parameterize the generator to account for extreme-
1244 value theory. This approach has been developed in (M. Allouche et al. 2022) (EV-GAN)
1245 where the renormalized log-quantile (called tail index function by the authors) is learnt
1246 using ReLU neural networks. The authors provide error bounds in uniform norm according
1247 to the complexity of neural networks and the second-order conditions in extreme-value
1248 theory. The latter reference shows that EV-GAN largely outperforms usual GANs, and is
1249 able to accurately sample X in dimension up to 50. For higher dimension, the marginals
1250 of X are still accurately reproduced but the tail dependence structure is under-estimated.
1251 Our work tackles the problem of better learning the dependence between margins with
1252 GANs.

1253 The closest work to ours is presumably (Hasan et al. 2022). The authors endeavour to
1254 approximate the stable tail dependence function (stdf) of the vector X of interest, see Def-
1255 inition 1.1 later, which characterizes the dependence in the extremes, once the marginals
1256 have been standardized to a unit Fréchet scale. Recall that, once the stdf is known (even
1257 approximately), the vector X can be sampled using the spectral representation of a max-
1258 stable process (Hofert et al. 2018, Theorem 2.1). In (Hasan et al. 2022) two learning
1259 techniques of the stdf are introduced. On the one hand, a deterministic approach is pro-
1260 posed using a dedicated architecture of neural networks called “d-Max NN”. The Fréchet
1261 marginalised data are projected onto directions in the simplex leading to exponentially
1262 distributed data. An algorithm is then introduced to maximise the log-likelihood using
1263 randomised directions on the neural network parameters. On the other hand, a stochastic
1264 approach is designed, based on the stochastic representation of the stdf via the spectral
1265 measure (see Proposition 1.1). Note that this approach is calibrated using the previous
1266 deterministic method. At first glance, our approach may look similar to (Hasan et al. 2022)
1267 but it is actually quite different, even though we also work on unitary Fréchet marginals.
1268 Here, we do rely on the spectral representation of max-stable processes, but we rather
1269 directly design a generative model of the vector X using an unitary Fréchet latent noise.
1270 In our opinion, this is much simpler since our approach avoids truncating (at a poorly con-
1271 trolled rank) the max-stable representation. Then, in contrast to Pareto-GAN, we show
1272 a convergence result of the approximated stdf to the true one when the dimension of the
1273 latent noise increases.

1274 Last, we shall mention the Tail-GAN of (R. Cont et al. 2022) in a financial setting,
1275 where a usual GAN is performed but the loss function for learning depends on the risk-
1276 metrics of the scalar quantity of interest (e.g. Value-at-Risk and Expected Shortfall). It is
1277 a way to enforce the generative model to reproduce some specific metrics: note that this
1278 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.

- $[n] = \{i \in \mathbb{N} : 1 \leq i \leq n\}$ 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 $\mathbf{e}_i = (0, \dots, 0, 1, 0, \dots, 0)$, $i \in [d]$, 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 \mathbf{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, \mathbf{x} is a constant vector and \mathbf{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 $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$, $\mathbf{x}\mathbf{y} := (x_1y_1, \dots, x_dy_d)$. Boolean operations are meant componentwise: $\mathbf{x} \leq \mathbf{y} \Leftrightarrow \forall i \in [d], x_i \leq y_i$. The scalar product is denoted with a dot: $\mathbf{x} \cdot \mathbf{y} = \sum_{i=1}^d x_i y_i$.
- The cumulative distribution function (cdf) of a random variable \mathbf{X} is denoted by $F_{\mathbf{X}}$ 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 $\text{Dom}(G)$.
- $\text{supp}(P)$ denotes the support of P .
- The Pareto distribution is defined by its cdf $1_{x \geq 1} (1 - x^{-1/\gamma})$ 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 \mathbf{X} with associated cdf $\mathbf{x} \in \mathbb{R}^d \mapsto F(\mathbf{x}) = \mathbb{P}(\mathbf{X} \leq \mathbf{x})$, extreme-value theory establishes the asymptotic distribution of normalized maxima of realizations of \mathbf{X} . Whence normalized to unit Fréchet margins (1.28), the limiting distribution function G_* is a simple max-stable distribution, see Definition 1.6 and Equation (1.27). As illustrated in the next paragraph, G_* encodes the dependence structure in the tails of F .

1349 1.2.2 Measuring dependence in extremes

1350 We first recall the definition of the stable tail dependence function (stdf) associated with
 1351 F .

1352 **Definition 1.1.** (L. d. Haan and Ferreira 2006, Corollary 6.1.4) Let F be a continuous
 1353 cdf on \mathbb{R}^d belonging to the max-domain of attraction of some distribution. Then, one can
 1354 define

$$\forall \mathbf{x} \in (0, \infty)^d : \ell_F(\mathbf{x}) = \lim_{t \rightarrow +\infty} t \mathbb{P} \left(1 - F_1(X_1) \leq \frac{x_1}{t} \text{ or } \dots \text{ or } 1 - F_d(X_d) \leq \frac{x_d}{t} \right), \quad (1.2)$$

1355 where F_j 's are the marginal cdf of \mathbf{X} . The limit ℓ_F is the so-called *stable tail dependence*
 1356 *function* (stdf) of F .

1357 The stdf provides a “normalized” representation of the dependence structure of the
 1358 distribution in the tails (as opposed to the copula function which globally models the
 1359 dependence), in the sense that its marginals are normalized: the stdf does not carry any
 1360 information on the marginal distributions of the base distribution of interest.

1361 The following proposition gives the classical parametrization of the dependence struc-
 1362 ture in terms of spectral measure. A more recent and less known parametrization using
 1363 D-norms is provided in Theorem 1.1 below.

1364 **Proposition 1.1** (Spectral representation of the stdf, (L. d. Haan and Ferreira 2006,
 1365 Remark 6.1.16.)). Consider a cdf F (as in Definition 1.1) and its stdf ℓ_F . Then, there
 1366 exists a probability measure Λ on the simplex $\Delta_{d-1} := \{\boldsymbol{\omega} \geq \mathbf{0} : \sum_{i \in [d]} \omega_i = 1\}$ such that:

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

1367 with constraints: $\forall i \in [d] : \int_{\Delta_{d-1}} \omega_i \Lambda(d\boldsymbol{\omega}) = 1/d$.

1368 **Quantifying dependence in extremes with D-norms.** D-norms (Falk 2019) are a
 1369 class of norms in vector space, very convenient for studying dependence in extremes.

1370 **Definition 1.2** (D-norms, (Falk 2019, Lemma 1.1.3)). Let $\boldsymbol{\Gamma} = (\boldsymbol{\Gamma}_1, \dots, \boldsymbol{\Gamma}_d) \in \mathbb{R}^d$ be a
 1371 random vector, whose components satisfy for all $i \in [d]$, $\boldsymbol{\Gamma}_i \geq 0$ a.s. and $\mathbb{E}[\boldsymbol{\Gamma}_i] = 1$. Then,

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

1372 defines a norm on \mathbb{R}^d , called a D-norm, and $\boldsymbol{\Gamma}$ is called a generator of the D-norm.

1373 Main properties of the D-norm are recalled in Appendix 3.C. To alleviate the role of
 1374 the generator, we may simply denote the above norm by $\|\cdot\|_D$. The key role of D-norms is
 1375 highlighted in the next result.

1376 **Theorem 1.1.** Let F be a continuous cdf on \mathbb{R}^d belonging to the max-domain of some
1377 distribution. Then, there exists a D-norm that exactly represents the stdf of F :

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

1378 This is a consequence of Theorem 1.6 in the Supplementary. Note that the two rep-
1379 resentations in (1.3) and in (1.4)–(1.5) may look similar at first sight, but actually they
1380 are substantially different. While both representations involve a probability measure (the
1381 spectral measure Λ on the one hand, and that of the generator Γ associated with the
1382 D-norm on the other hand), the second representation is more tractable for approximation
1383 purposes: Instead of designing a non-negative measure on the simplex Δ_{d-1} , one *simply*
1384 has to define a random vector $\Gamma \in [0, +\infty)^d$ with unit expectation.

1385 **Definition 1.3.** Let F be a cdf satisfying the assumptions of Theorem 1.1. The stdf ℓ_F
1386 is said to be *discrete*, with N atoms, when its D-norm is related to a generator Γ whose
1387 distribution is *discrete* and made of N atoms.

1388 **Example 1.1.** Let us consider the special case where $d = N$. Note that it can be typically
1389 the case of the input of a generative model with a latent distribution of dimension N . Let F
1390 be a cdf in \mathbb{R}^N with independent margins. Then, let $\ell_F(\mathbf{x}) = \sum_{i \in [N]} x_i, \quad \forall \mathbf{x} \in (0, \infty)^N$.
1391 This stdf is discrete with N atoms: indeed, one can easily check that, if Γ has the discrete
1392 uniform distribution on $\{N \mathbf{e}_1, \dots, N \mathbf{e}_N\}$, then $\mathbb{E}[\Gamma_i] = 1$ for any $i \in [N]$, and its D-norm
1393 is given by

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

1394 With similar arguments, it is easily checked that the spectral measure is the discrete
1395 uniform distribution on $\{\mathbf{e}_1, \dots, \mathbf{e}_N\}$.

1396 The concept of Definition 1.3 is at the core of our numerical scheme, since we will
1397 consider approximations based on discrete distributions for the generator Γ in Section 1.2.4.

1398 **Characterization of simple max-stable distribution using D-norms and homo-**
1399 **geneous function.** We aim at giving another criterion to establish the max-domain of
1400 attraction of a simple max-stable distribution. It will be crucially used in the proof of our
1401 main result (Theorem 1.3).

1402 **Definition 1.4** (1-homogeneous function). A function $h : [0, \infty)^d \rightarrow [0, \infty)^{d'}$ is 1-
1403 homogeneous if it satisfies: $\forall \mathbf{x} \in [0, \infty)^d, \forall \lambda \geq 0 : h(\lambda \mathbf{x}) = \lambda h(\mathbf{x})$. The set of 1-
1404 homogeneous continuous functions from \mathcal{X} to \mathcal{Y} is denoted by $\mathbb{H}_1(\mathcal{X}, \mathcal{Y})$.

1405 The next result claims that identifying the stdf (or equivalently the D-norm) of a cdf
1406 F (with asymptotically simple max-stable distribution) is equivalent to analyze tails of
1407 1-homogeneous transforms of the associated random vector.

1408 **Theorem 1.2** ((Falk and Fuller 2021, Theorem 4.4)). Let \mathbf{X} be a random vector in
1409 $[0, +\infty)^d$ with continuous cdf F and let $\|\cdot\|_D$ be a D-norm in \mathbb{R}^d generated by $\boldsymbol{\Gamma}$. The
1410 following equivalence holds:

$$\forall \mathbf{x} \in (0, \infty)^d : F^t(t\mathbf{x}) \xrightarrow[t \rightarrow \infty]{} \exp - \left\| \frac{1}{\mathbf{x}} \right\|_D \quad (1.7)$$

$$\iff \forall h \in \mathbb{H}_1([0, \infty)^d, [0, \infty)) : t\mathbb{P}(h(\mathbf{X}) > t) \xrightarrow[t \rightarrow \infty]{} \mathbb{E}[h(\boldsymbol{\Gamma})]. \quad (1.8)$$

1411 An indirect consequence of the above theorem is that the limiting value $\mathbb{E}[h(\boldsymbol{\Gamma})]$ does
1412 not depend on the chosen generator of the D-norm, when h is a continuous 1-homogeneous
1413 function, see (Falk and Fuller 2021, Theorem 3.5) for a direct proof.

1414 1.2.3 Generative Adversarial Networks

1415 **Neural networks.** We refer to (Murphy 2022, Chapter 13) for a thorough introduction
1416 on neural networks. Let us consider fully connected neural networks, also called Multi
1417 Layer Perceptron (MLP). These neural networks consist of a chain of L operations of the
1418 form:

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

1419 for $l \in [L]$, where $\mathbf{z}_l \in \mathbb{R}^{q_l}$, $\varphi_l : \mathbb{R}^{q_l} \rightarrow (\mathbb{R}^+)^{q_l}$ is an activation function, q_l is the number
1420 of neurons on the layer l , \mathbf{W}_l is a matrix and \mathbf{b}_l is a bias vector. Note that \mathbf{z}_0 corresponds
1421 to the q_0 -dimensional input of the network, while \mathbf{z}_L is the q_L -dimensional output. In this
1422 work, we focus on neural networks with ReLU activation functions: $\varphi_l(\mathbf{z}) = \max\{\mathbf{z}, \mathbf{0}\}$.
1423 The parameters of the network are:

$$\theta = \{\mathbf{b}_1, \dots, \mathbf{b}_L, \mathbf{W}_1, \dots, \mathbf{W}_L\} \in \Theta \quad (1.10)$$

1424 (activation functions are fixed), where Θ is the space of parameters, the full Euclidean
1425 space without constraints.

1426 **Proposition 1.2.** A neural network parameterized by (1.9) with ReLU activation functions
1427 is a piecewise affine function. It is a continuous 1-homogeneous function when the bias
1428 vectors are zero.

1429 The proof is easy and left to the reader. Combining Proposition 1.2 with Theorem 1.2
1430 shows that a distribution (with simple max-stable distribution) transformed by a ReLU
1431 neural network remains a heavy-tailed distribution with the same tail parameter γ .

1432 **Generative Adversarial Networks.** Generative modeling is the task of approximating
1433 at best a distribution of interest, here written $\mathbf{X} \sim p_{\text{data}} \in \mathbb{R}^d$, and being able to draw
1434 samples from it. One way to do is to find a map \mathcal{G} which takes as input a random variable
1435 $\mathbf{Z} \sim p_{\mathbf{Z}} \in \mathbb{R}^N$ which is easy to sample, such that:

$$\mathcal{G}(\mathbf{Z}) \stackrel{d}{\approx} \mathbf{X}. \quad (1.11)$$

The random vector \mathbf{Z} is commonly referred as the latent noise and $p_{\mathbf{Z}}$ 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 adversarial 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:

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

where p_{data} is the distribution of the data and $p_{\mathbf{Z}}$ 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 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 \mathbf{Z} with a discrete stdf $\ell_{\mathbf{Z}}$ 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 $(\mathbf{W}_1, \dots, \mathbf{W}_L)$ are non-negative. Then, the stdf of the output $G_{\theta, N}(\mathbf{Z})$ 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 \mathbf{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 $\mathbf{Z} \in \mathbb{R}^N$ with i.i.d. unit Fréchet margins such that:*

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

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

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 \mathbf{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*

1469 (*Theorem 1.1*). Consider using a MLP neural network $\mathcal{G}_{\theta,N}$ with a single layer (i.e. $L = 1$)
1470 and with as latent noise a random vector of N i.i.d. unit Fréchet margins, and denote by
1471 $\ell_{\mathcal{G}_{\theta,N}}$ its stdf. Then,

$$\inf_{\theta \in \Theta} \sup_{\mathbf{x} \in (0, \infty)^d} \frac{|\ell_F(\mathbf{x}) - \ell_{\mathcal{G}_{\theta,N}}(\mathbf{x})|}{\|\mathbf{x}\|_\infty} \leq \epsilon(N), \quad (1.14)$$

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

1472 where $C(d)$ is a constant depending on d and $\ell_F(\mathbf{1})$. The bound (1.14) is achieved with
1473 one layer neural networks.

1474 The intuition behind the bounds (1.14)-(1.15) can be developed as follows. Approximating
1475 the stdf amounts to approximating a probability measure in dimension d using N
1476 atoms; it is thus natural to expect an error bound of order $N^{-1/d}$. However, the random
1477 variable having this probability measure is subject to a realization or moment constraint
1478 (see the proof for details), which reduces the degrees of freedom by one, leading to an
1479 approximation error of at most order $N^{-1/(d-1)}$.

1480 In this previous theorem, note that the error bound is a worst case scenario. For
1481 example, when the stdf is discrete, the error is zero for N sufficiently large in virtue of
1482 Proposition 1.4. Let us highlight that, in view of the upper bound (1.15), the attainable
1483 precision of the neural network approximation increases with the dimension of the latent
1484 noise. However, the higher the dimension d , the slower is the convergence. One can then
1485 expect that accurate approximations of the dependence in high dimension would require
1486 high dimensional latent noise, which is consistent with the intuition. This will be illustrated
1487 in the Numerical Experiments section.

1488 In other words, the above result states a denseness property. We would like to thank
1489 one of the referees to spot the analogy with the results from (Fougères and Mercadier 2013,
1490 Lemma 9, Theorem 3). As a difference, we use other arguments (see the proof based on
1491 quantization tools) and, consequently, we are able to quantify the effect of both latent
1492 and ambient dimensions N and d . Let us finally note that this denseness property is used
1493 in (JanSSen and Wan 2020) to interpret the proposed k -means procedure as an inference
1494 method for max-linear models.

1495 1.3 Numerical experiments

1496 First, we describe in Section 1.3.1 the proposed algorithms for training and sampling
1497 from a dataset presenting heavy-tailed characteristics. Second, experiments performed
1498 on synthetic datasets generated using the Gumbel copula are detailed in Section 1.3.2 to
1499 compare the proposed method with the standard GAN approach. Finally, experiments on
1500 the stock market index S&P500 are presented in Section 1.3.3.

1501 1.3.1 Algorithms and implementation

1502 **Algorithms**

1503 We accommodate the original GAN algorithm (I. Goodfellow et al. 2014) to have a heavy-
 1504 tailed latent noise. This idea originates from (Huster et al. 2021). Algorithm 1 describes
 1505 the learning phase while Algorithm 2 summarizes the generation phase.

Algorithm 1: Generative modelling: learning algorithm

input :

- Dataset $\{\mathbf{X}^{(1)}, \dots, \mathbf{X}^{(n_{\text{data}})}\}$ of n_{data} i.i.d. sample points in \mathbb{R}^d .
- Fully connected initialized generator \mathcal{G} and discriminator \mathcal{D} .
- Estimators $\hat{F}_1, \dots, \hat{F}_d$ 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.

1 begin

1. For any $j \in [d]$, estimate the marginal cdf of X_j on the training set
 $\{X_j^{(1)}, \dots, X_j^{(n_{\text{data}})}\}$ by

$$x \in \mathbb{R} \mapsto \hat{F}_j(x) \in [0, 1]. \quad (1.16)$$

2. Transform the marginals to Fréchet distributions with tail parameter γ :

$$\forall i \in [n_{\text{data}}], j \in [d] : \tilde{X}_j^{(i)} = \left(\frac{1}{1 - \hat{F}_j(X_j^{(i)})} \right)^\gamma. \quad (1.17)$$

3. Train GAN over transformed data $\{\tilde{\mathbf{X}}^{(1)}, \dots, \tilde{\mathbf{X}}^{(n_{\text{data}})}\}$: optimize (1.12) where p_{data} is a componentwise Pareto distribution with tail parameter γ and independent margins.

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

2 end

1506 **Remarks about the marginals.** In Subsection 1.3.3, which focuses on experiments
 1507 with real data, the estimator \hat{F} is taken to be the empirical cdf. While other choices are
 1508 possible – since estimating a one-dimensional cdf is a classical problem in statistics – this
 1509 simple choice suffices to run Algorithm 1 and obtain an approximation of the generator
 1510 \mathcal{G}^* . In doing so, we address the primary objective: reproducing the dependence structure
 1511 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 $\hat{F}_1, \dots, \hat{F}_d$ of the margins.
- Generator \mathcal{G}^* .

1 begin

1. Sample n_{sample} points $(\mathbf{z}^{(1)}, \dots, \mathbf{z}^{(n_{\text{sample}})})$ from p_{data} , a componentwise Pareto distribution with tail parameter γ and independent margins.

2. Transform the noise through the generator:

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

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

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

2 end**3 return** Sampled points $\left\{ \hat{\mathbf{X}}^{(1)}, \dots, \hat{\mathbf{X}}^{(n_{\text{sample}})} \right\}$.

1512 (*i.e.*the quantile function). To ensure an accurate fit of the marginals – while preserving
1513 the learned dependence – we recommend using a quantile estimator that performs well
1514 across the entire distribution, both in the bulk and in the tails. A natural candidate is the
1515 estimator based on eLU-activated neural networks proposed by (M. Allouche et al. 2024).

1516 **Implementation**

1517 **Code.** The implementation is based on a publicly available GitHub repository¹ using the
1518 machine learning library PyTorch². We adapt the source code so that both the generator
1519 and discriminator are fully connected MLPs (1.9). For writing a specific architecture of a
1520 neural network, we use a list that details all hidden layers, that is layers q_l for $l \in [L - 1]$
1521 in (1.9). As an example, a network parametrized by $[100, 200]$ means that it has 2 hidden
1522 layers, $L = 3$, $q_1 = 100$ and $q_2 = 200$. For the activation functions, we use LeakyReLU
1523 (Maas et al. 2013): note that our theoretical result (Theorem 1.3) still holds since a
1524 LeakyReLU is a difference of two ReLUs. For gradient descent, we use the optimizer Adam
1525 (D. P. Kingma and Ba 2014) and we do not tune the base parameters. Early stopping is
1526 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 Cholesky³ 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.

1.546 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 $\arccos(\hat{\mathbf{X}}_1 / \|\hat{\mathbf{X}}\|)$ for the 10% largest Euclidean norms $\|\hat{\mathbf{X}}\|$. 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)

³https://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.

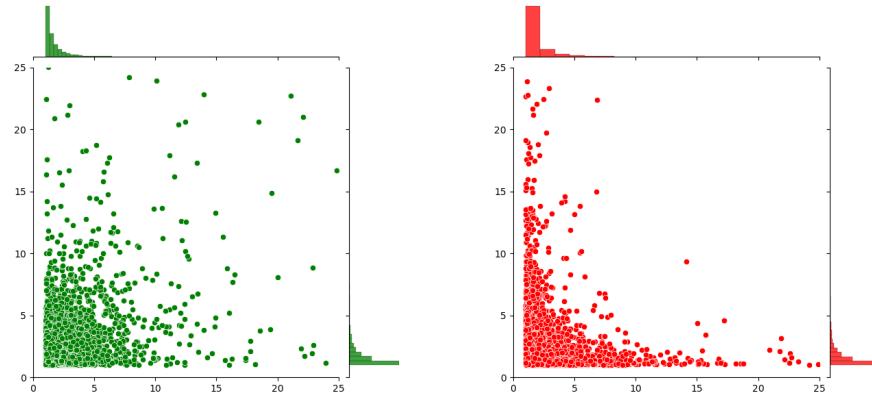


Figure 1.1: Illustration in a two-dimensional setting on simulated data of size $n_{\text{data}} = 10,000$ with Pareto margins and tail parameter $\alpha = 2.0$. Left panel: sample from a Gumbel copula with dependence parameter $\beta = 4/3$. Right panel: sample with two independent margins.

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)

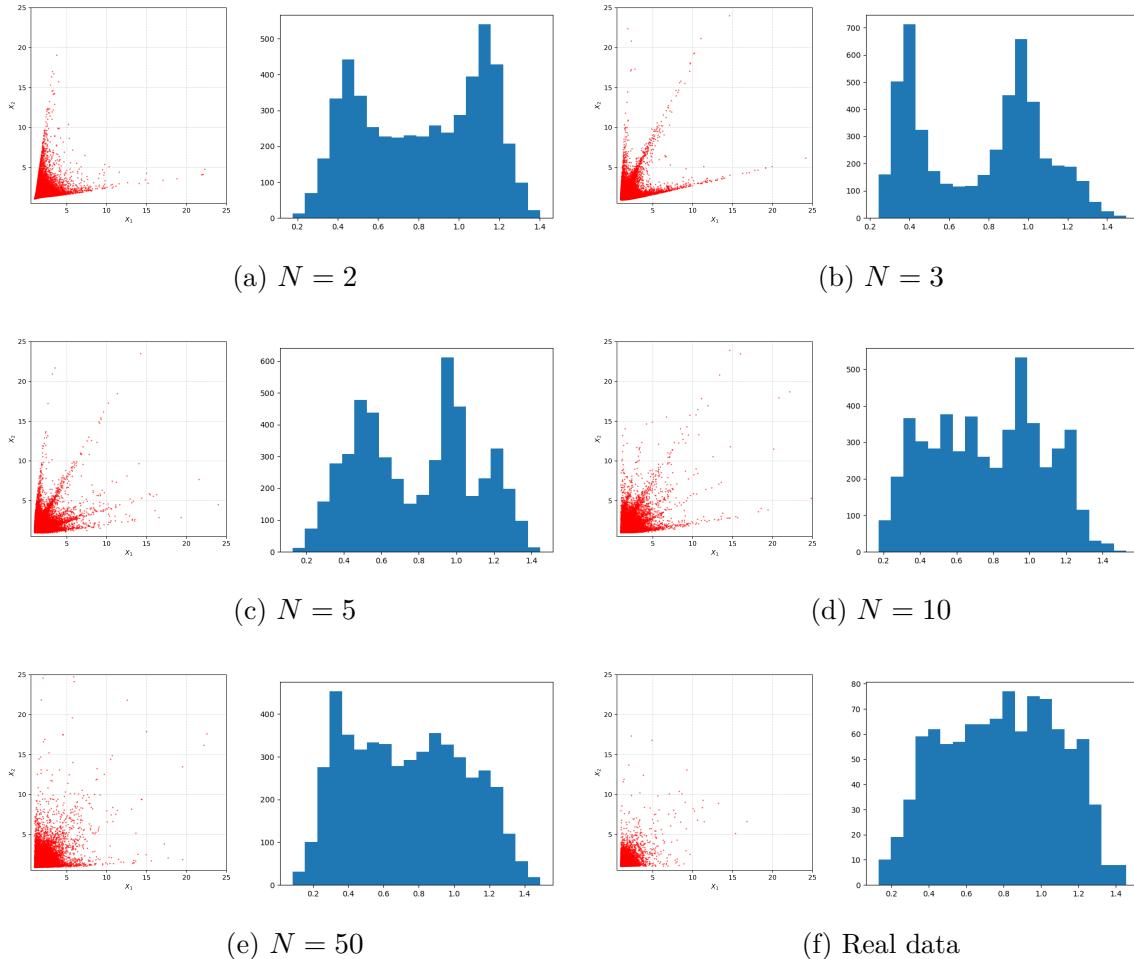


Figure 1.2: Illustration of real and generated data in a two-dimensional setting for data from a Gumbel copula with $\beta = 4/3$ and Pareto margins ($\alpha = 2.0$). For each sub-figure (a)-(f), on the left: scatter plot of data points; on the right: histogram of angles $\arccos(X_1/\sqrt{X_1^2 + X_2^2})$ 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.

1596 Preliminary hyperparameter search. We explore a number of possible parametrizations
1597 of GAN models within a selected range of hyperparameters. For both the generator
1598 and the discriminator, fully connected neural networks are used with the following 15 hid-
1599 den 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 tool⁴. 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 hyperparameter 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 $\mathbf{Z} \sim \mathcal{N}(0, \mathbf{I}_{N \times N})$, 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 though 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>

1638 method is better performing than the baseline. In the case $\alpha = 2.5$, the difference is not
1639 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 $d \in \{2, 5, 10, 20, 50\}$.

α	$\beta = 4/3$			$\beta = 2$			$\beta = 4$		
	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

1640 **Latent dimension.** Equation (1.14) gives evidence that a higher value of the latent di-
1641 mension N is necessary to obtain a better approximation of the dependence structure. For
1642 each experiment specification (*i.e.*a value for α and for d), we have performed a Bayesian
1643 hyperparameter search (J. Wu et al. 2019) with 400 runs, with the same hyperparameter
1644 ranges as for the previous experiment section. Table 1.2 summarizes statistics on the la-
1645 tent dimension, $\hat{\alpha}$ and SWD_90 for the top 5% run on each experiment. It appears that,
1646 empirically, the theoretical result is verified: generally, for the various values of α , as the
1647 dimension of the data increases, the optimal value of the latent dimension increases. We
1648 also note that, for each value of α , as the dimension of the data increases, the estimate
1649 $\hat{\alpha}$ increases accordingly, which means lower tails. However, this conclusion is to be inter-
1650 preted with caution in view of the great variability associated with the statistics (reflected
1651 by the values taken by min and max). This may be a consequence of the instability in
1652 training generative models with heavy-tailed inputs (Hickling and Prangle 2024).

1653 Other models: Gaussian and Hüsler-ReiSS copulas

1654 We extend our evaluation to higher dimensions using two dependence structures: the
1655 multivariate Hüsler–ReiSS (HR) extreme-value copula and the Gaussian copula. These
1656 represent two qualitatively different tail regimes: HR exhibits non-trivial asymptotic tail
1657 dependence, whereas the Gaussian copula is asymptotically independent.

1658 **Background: multivariate Hüsler–ReiSS.** The HR model is a max-stable (extreme-
1659 value) dependence model that arises as the limit of componentwise maxima of *log-Gaussian*
1660 vectors, see (Hüsler and Reiss 1989b). In dimension d , the dependence is encoded by a
1661 matrix $(\lambda_{ij})_{1 \leq i,j \leq d}$ with $\lambda_{ii} = 0$ and $\lambda_{ij} > 0$ for $i \neq j$. Smaller λ_{ij} means stronger extremal
1662 dependence between components i and j ; larger λ_{ij} weakens it. In the bivariate case, the

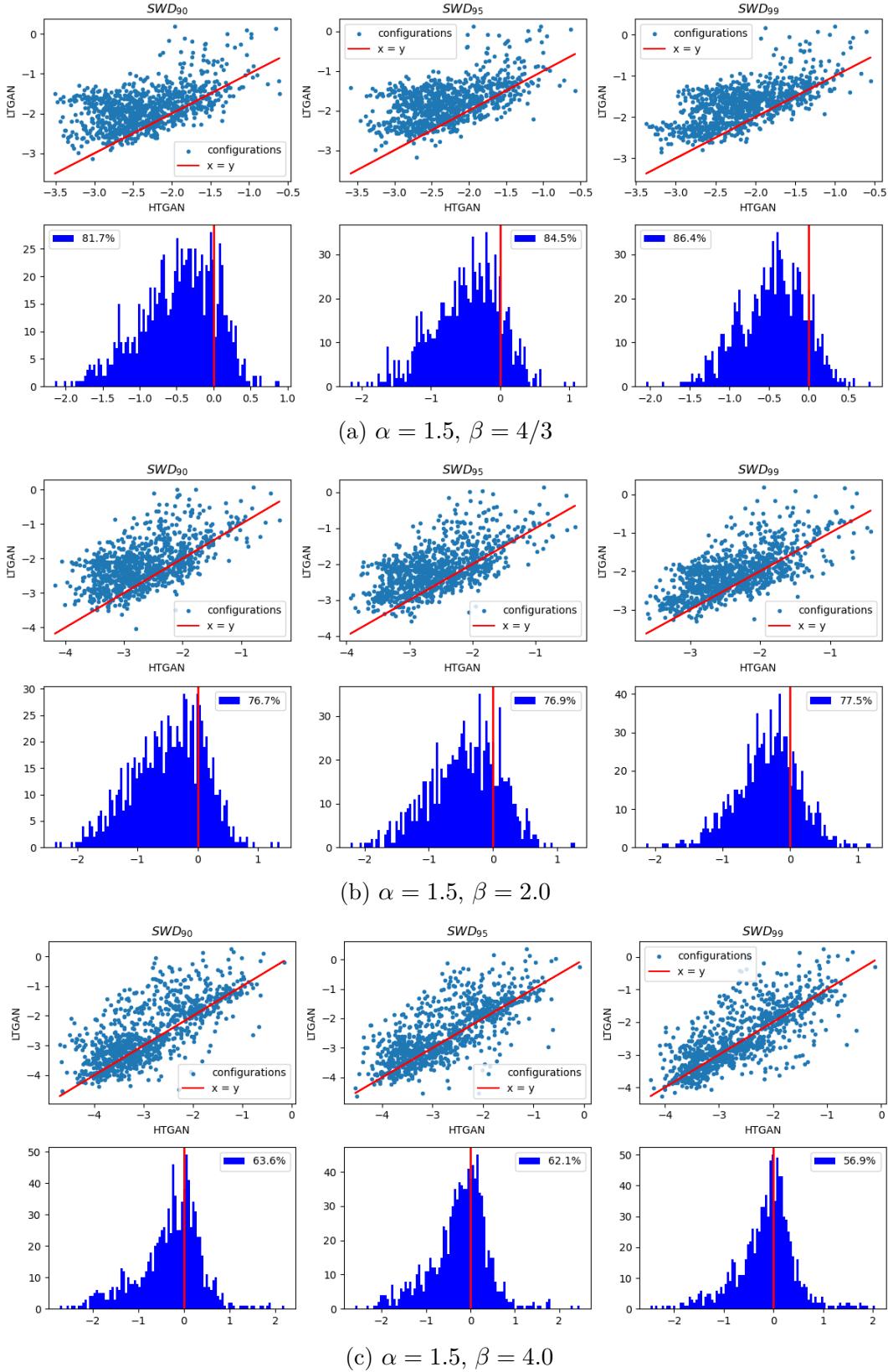


Figure 1.3: Illustration in a d -dimensional setting on simulated data of size $n_{\text{data}} = 10,000$ from a Gumbel copula (with dependence parameter β) and Pareto margins (with tail parameter $\alpha = 1.5$). Each group of 2×3 plots corresponds to one experimental setting, *i.e.* a specification 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.

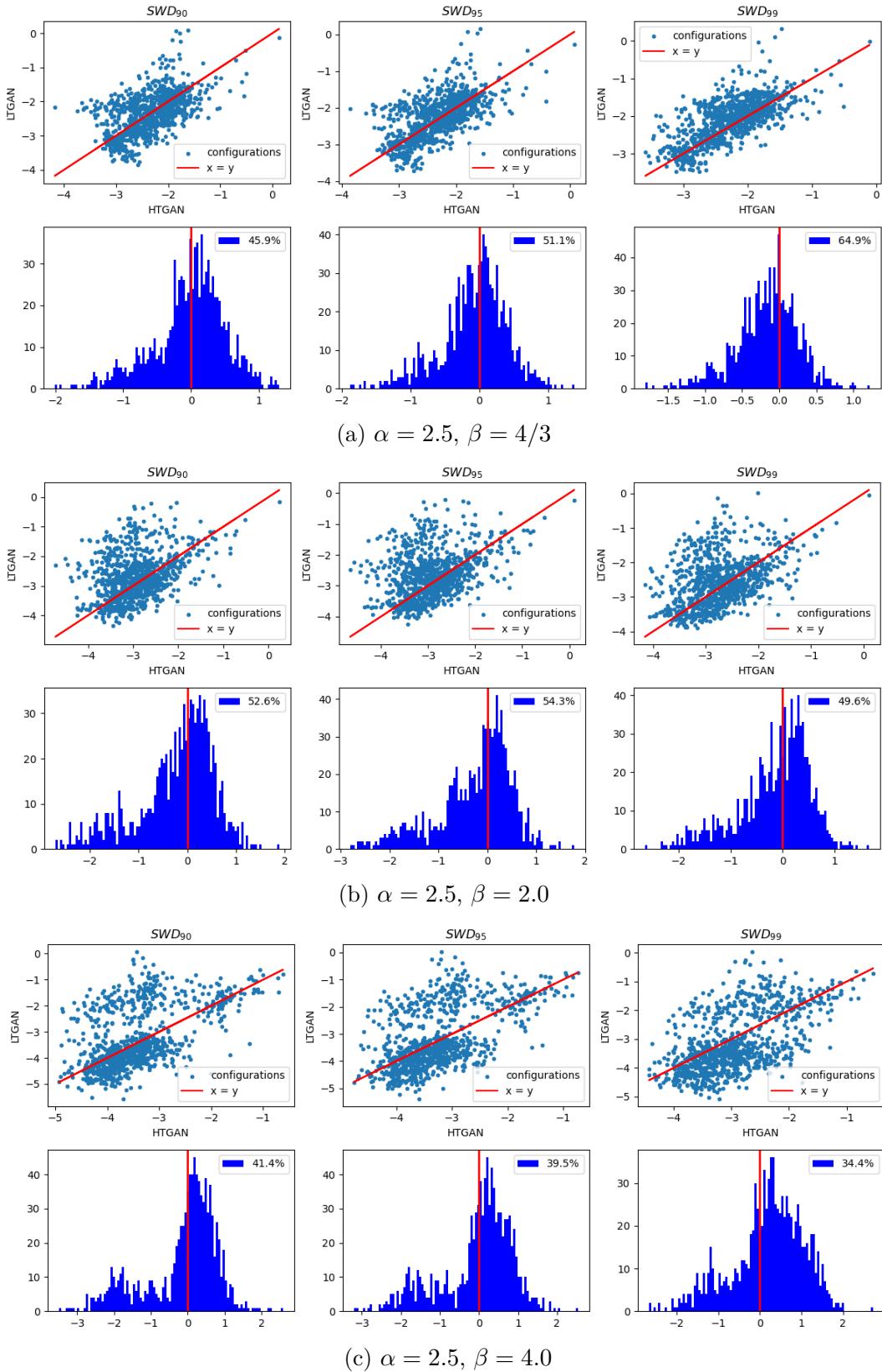


Figure 1.4: Illustration in a d -dimensional setting on simulated data of size $n_{\text{data}} = 10,000$ from a Gumbel copula (with dependence parameter β) and Pareto margins (with tail parameter $\alpha = 2.5$), $d \in \{2, 5, 10, 20, 50\}$. See Figure 1.3 for further details.

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 $\beta \in \{4/3, 2, 4\}$.

α	d	N (latent dimension)			$\hat{\alpha}$			SWD_90		
		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

1663 upper tail dependence coefficient is

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

1664 where Φ is the cumulative distribution function of the standard Gaussian distribution.
1665 Clearly, $\chi_{ij} \downarrow 0$ as $\lambda_{ij} \uparrow \infty$. A convenient spectral construction is via a centered Gaussian
1666 vector $\mathbf{Z} \in \mathbb{R}^d$ with covariance Σ derived from (λ_{ij}) (Brown–Resnick representation): Set
1667 $\mathbf{Y} = \exp \mathbf{Z}$, normalize $\mathbf{S} = \mathbf{Y} / \sum_k Y_k$, draw $E \sim \text{Exp}(1)$, and form a unit-Fréchet
1668 HR vector $\mathbf{X}^{\text{Fr}} = \mathbf{S}/E$. Arbitrary continuous margins are then obtained by monotone
1669 transforms.

1670 **Background: Gaussian copula.** Let $\mathbf{Z} \sim \mathcal{N}_d(0, \Sigma)$ 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.
1671 For $|\rho| < 1$ the Gaussian copula is asymptotically independent ($\chi = 0$), but larger ρ
1672 increases sub-asymptotic dependence.

1673 More details on the specific experimental design for these two models are given in
1674 Subsection 3.I.

1676 **Qualitative summary.** We summarize results in Table 1.3. Across both datasets, the
1677 trends mirror those observed for the Gumbel copula: heavier tails (lower α) consistently
1678 amplify the relative advantage of HTGAN over LTGAN. Therefore, both in asymptotic de-
1679 pendence and in independence settings, HTGAN outperforms LTGAN. Additional figures
1680 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 $\pm 0.1\%$. Values are averaged over data dimensions $d \in \{2, 5, 10, 20, 50\}$.

α	Hüsler-ReiSS			Gaussian		
	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

1681 1.3.3 Experiments on real data

1682 The proposed HTGAN method is tested on the publicly available dataset⁵ consisting of
 1683 daily data for companies of the S&P500. The S&P500 is a stock market index track-
 1684 ing of the 500 most valuable companies in the United States of America. The sig-
 1685 nal of interest is the absolute value of the normalized marginal returns for each stock:
 1686 $r_t = |(X_{t+1} - X_t) / X_t|$, which are bounded below by zero and only have an upper tail. The
 1687 dataset consists of $n_{\text{data}} = 1259$ normalized marginal returns for each ticker. Companies
 1688 are organized using the Global Industry Classification Standard (GICS). This classification
 1689 divides companies into 11 sectors: Energy, Materials, Industrials, Consumer Discretionary,
 1690 Consumer Staples, Health Care, Health Care, Financials, Information Technology, Com-
 1691 munication Services, Utilities and Real Estate. Each sector is divided further into industry
 1692 groups, industry and sub-industry. The returns of the S&P500 present a particular corre-
 1693 lation structure illustrated in Figure 1.5 by a heatmap of the table consisting of estimated
 1694 Kendall's tau for every pair of stock ticker, see the Appendix for more details on Kendall's
 1695 tau coefficient. It appears that returns are more correlated within each sector, especially
 1696 in both the Financials and the Utilities sectors, than between sectors. Therefore, in the
 1697 following, the performance of LTGAN and HTGAN are assessed on the two subsets associ-
 1698 ated with sectors Financials and Utilities, having both strong correlations (see Figure 1.5).
 1699 Note that the number of tickers for the Financials and Utilities sector are not the same:
 1700 $d = 27$ for the Financial sector, and $d = 57$ for the Utilities sectors.

1701 Before moving to the results, let us mention that on the application side of portfolio
 1702 management, it is important to model the dependence of extreme asset returns well, as
 1703 shown in (Mainik et al. 2015), in order to manage risks more accurately. This supports
 1704 the use of HTGAN compared to LTGAN. Their study compares the traditional Markowitz
 1705 approach with the Extreme Risk Index – a method based on multivariate EVT – to optimize
 1706 a portfolio of S&P 500 stocks: their results show the superiority of a model that better
 1707 accounts for dependencies in the tails, compared to the one based only on covariances.

⁵<https://www.kaggle.com/datasets/camnugent/sandp500>

1708 **Sectors and subsectors.** The subsectors for the financial sector are: Asset Management
1709 & Custody Banks, Consumer Finance, Diversified Banks, Financial Exchanges & Data, In-
1710 surance Brokers, Investment Banking & Brokerage, Life & Health Insurance, Multi-Sector
1711 Holdings, Multi-line Insurance, Casualty Insurance, Regional Banks and Processing Ser-
1712 vices. For the Utilities sector, Subsectors are: Electric Utilities, Gas Utilities, Independent
1713 Power Producers & Energy Traders, Multi-Utilities and Water Utilities.

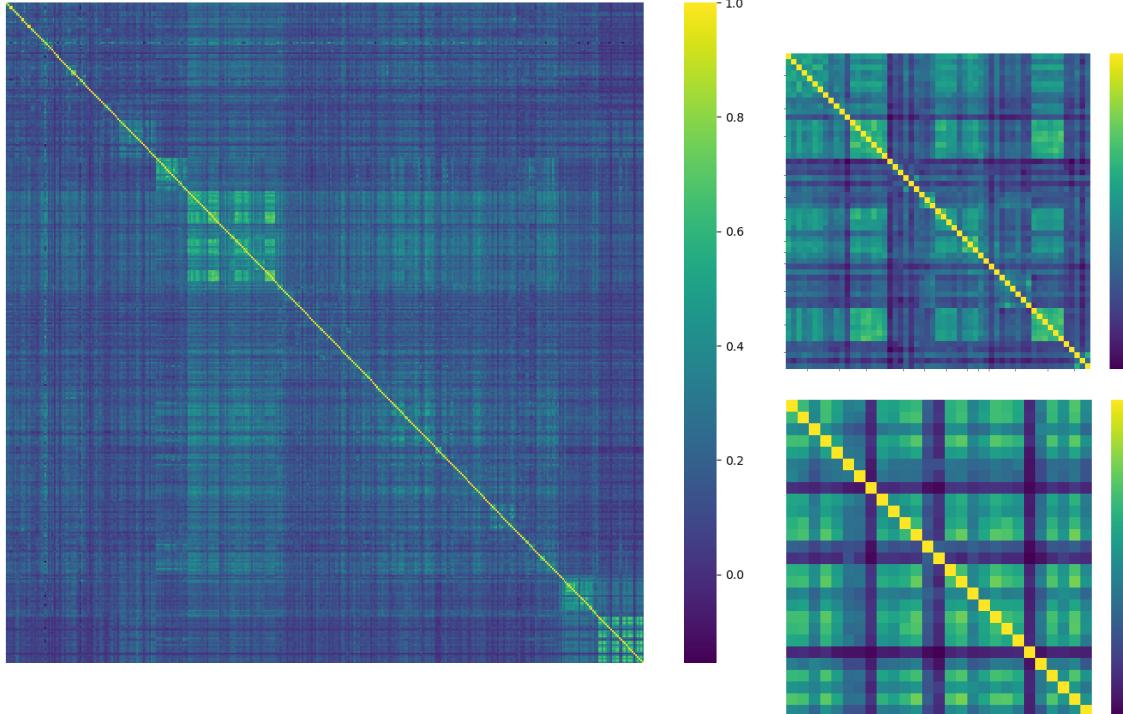


Figure 1.5: Estimated Kendall’s τ for every pair of index considered. Left: every S&P500 ticker. On the top right: Tickers for the Financials sector. On the bottom right: Tickers for the Utilities sector. A comprehensive list of sectors and subsectors is given in Paragraph 1.3.3.

1714 **Data normalization.** In the normalizing step (1.17) of Algorithm 1, two values of $\gamma =$
1715 $1/\alpha$ are investigated: $\alpha = 1.5$ and $\alpha = 2.5$. The cumulative distribution function is
1716 estimated using its empirical counterpart:

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

1717 This transformation is not invertible, and therefore it is not possible to proceed to step (1.19)
1718 in the generation step of Algorithm 2. Therefore, we skip this step in the generation pro-
1719 cess. In our evaluation, we solely focus on the good reproduction of dependence in extreme
1720 regions (*i.e.*the difficult task). Our criteria do not assess how well the marginals are fitted.

1721 **Results.** Figure 1.6 presents the results for experiments run on the S&P500 dataset. The
1722 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 models 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.

1752 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.

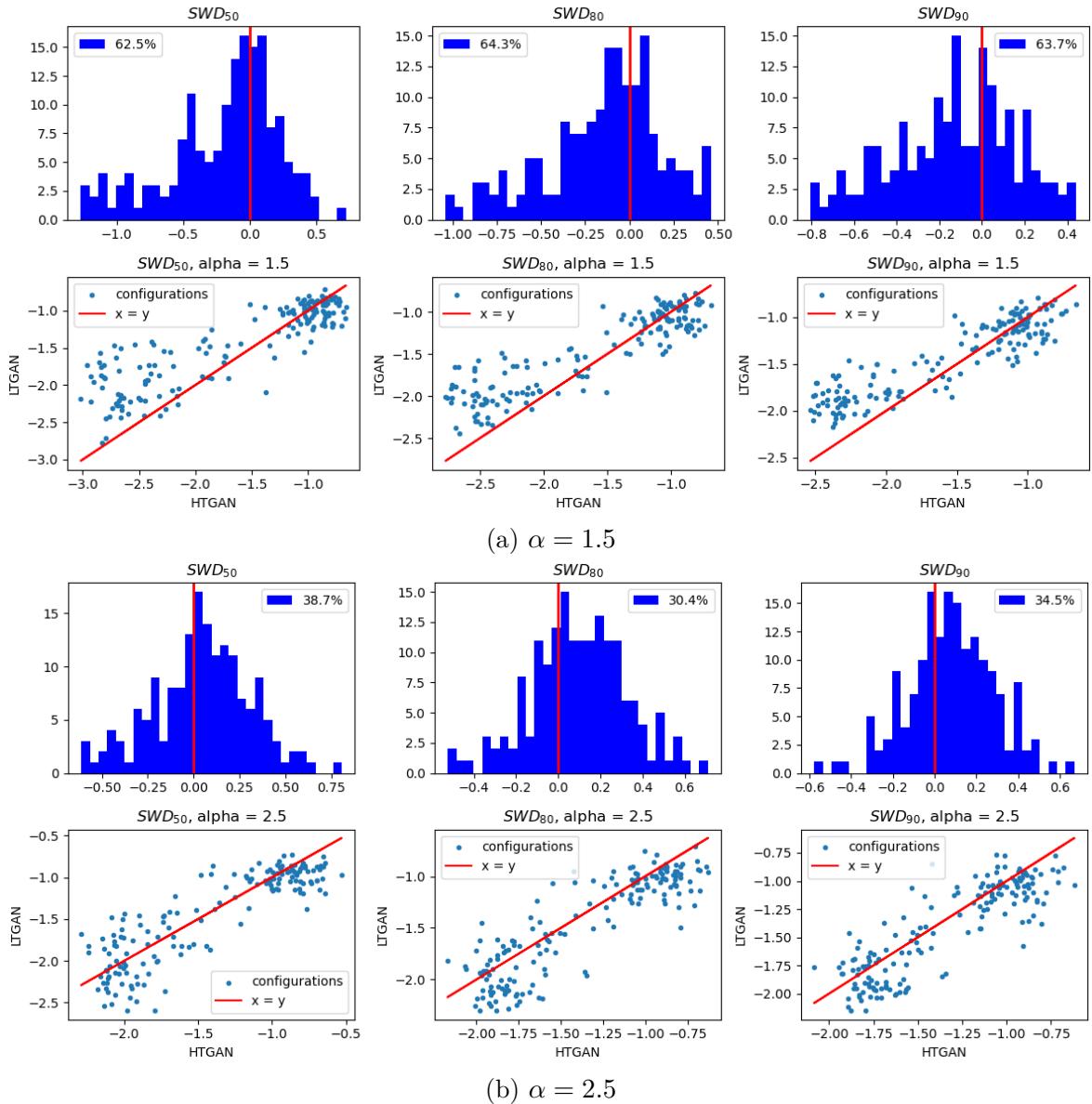


Figure 1.6: Results of the performance of HTGAN vs LTGAN on the Utilities ($d = 27$) and Financials subsectors ($d = 57$) of the S&P500 dataset. Results are aggregated with respect to the dimension d for plot. Each two groups of 2×3 plots corresponds to one experimental setting, *i.e.* $\alpha = 1.5$ (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.

1758 **Appendix**

1759 **3.A Theoretical background**

1760 Section 3.B provides additional background on extreme-value theory to complement Sec-
 1761 tion 1.2.1. Section 3.C is devoted to D-norms, introduced in Section 1.2.2. Section 3.D
 1762 focuses on the theory of quantization. Finally, Section 3.E provides some details on copulas
 1763 and measures of dependence.

1764 **3.B Extreme-value theory**

1765 **General set-up**

1766 Consider a \mathbb{R}^d valued random vector \mathbf{X} with associated cdf $\mathbf{x} \in \mathbb{R}^d \mapsto F(\mathbf{x}) = \mathbb{P}(\mathbf{X} \leq \mathbf{x})$.
 1767 The primary goal of extreme-value theory is to establish the asymptotic distribution of
 1768 well normalized maxima of realizations of \mathbf{X} , see (S. I. Resnick 1987; Beirlant et al. 2004;
 1769 L. d. Haan and Ferreira 2006) for reference textbooks. Let $(\mathbf{X}_i)_{i \in [n]}$ be an i.i.d.sample from
 1770 F and consider $\mathbf{M}_n := \frac{\max_{i \in [n]} \mathbf{X}_i - \mathbf{b}_n}{\mathbf{a}_n}$, where $\mathbf{a}_n > \mathbf{0}$ and \mathbf{b}_n are normalizing sequences in
 1771 \mathbb{R}^d . Clearly, the cdf of \mathbf{M}_n is given by $F_{\mathbf{M}_n}(\mathbf{x}) = F^n(\mathbf{a}_n \mathbf{x} + \mathbf{b}_n)$. This remark gives rise
 1772 to the following definition:

1773 **Definition 1.5** (Max-Domain of Attraction (MDA), (L. d. Haan and Ferreira 2006, Section
 1774 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
 1775 by $F \in \text{Dom}(G)$, if there exist sequences $\mathbf{a}_n > 0$ and \mathbf{b}_n , $n \in \mathbb{N}$ such that:

$$\forall \mathbf{x} \in \mathbb{R}^d : F^n(\mathbf{a}_n \mathbf{x} + \mathbf{b}_n) \xrightarrow[n \rightarrow \infty]{} G(\mathbf{x}). \quad (1.22)$$

1776 In other words, the normalized maximum of a sample from F converges in distribution
 1777 to a random vector with cdf G . It can be shown that the limiting distribution function is
 1778 necessarily max-stable, as defined below:

1779 **Definition 1.6** (Max-stable distribution, (L. d. Haan and Ferreira 2006, Section 6.1.2)).
 1780 A cdf G on \mathbb{R}^d is called max-stable if, for any $n \in \mathbb{N}$, there exist sequences $\mathbf{a}_n > 0$ and
 1781 $\mathbf{b}_n \in \mathbb{R}^d$ such that

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

1782 Besides, convergence (1.22) can be extended to a continuous counterpart; one has
 1783 $F \in \text{Dom}(G)$ if and only if there exist two functions $\mathbf{a}_t > 0 \in \mathbb{R}^d$ and $\boldsymbol{\theta} \in \mathbb{R}^d$ such that:

$$\forall \mathbf{x} \in \mathbb{R}^d : F^t(\mathbf{a}_t \mathbf{x} + \boldsymbol{\theta}) \xrightarrow[t \rightarrow \infty]{} G(\mathbf{x}). \quad (1.24)$$

1784 **Margins**

1785 Let us highlight that, if F fulfills (1.22) or (1.24), then its margins F_j satisfy

$$\forall x \in \mathbb{R} : F_j^n(a_{j,n}x + b_{j,n}) \xrightarrow[n \rightarrow \infty]{} G_j(x), \quad (1.25)$$

where G_j denotes the j th 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_j s 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*

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

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_* , the so-called associated simple max-stable distribution, which is defined as:

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

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

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

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_* .

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_* .

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_* defined in (1.27). The stdf of G is given by*

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

We show in the next paragraph that both ℓ_G and G_* 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).

1812 **Definition and basic properties**

1813 We start with basic definitions.

1814 **Definition 1.7** (Norm and D-norms). A function $\|\cdot\| : \mathbb{R}^d \rightarrow \mathbb{R}^+$ is a norm on \mathbb{R}^d if it
1815 verifies three conditions: a) *Positive definiteness*, b) *Absolute Homogeneity*, c) *Triangle*
1816 *inequality*.

1817 A subset of norms is the set of D-norms defined by

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

1818 for a non-negative random variable $\boldsymbol{\Gamma} \in \mathbb{R}^d$ with unit expectation (Definition 1.2), $\boldsymbol{\Gamma}$ is
1819 also known as the generator of the D-norm.

1820 Let us give a few remarks. First, since D-norms are monotone and radially symmetric,
1821 not all norms are D-norms (see (Falk 2019, Chapter 1)). Second, usual norms (such as
1822 the L_p -norms for $p \in [1, \infty]$) are D-norms. Last, observe that there are infinitely many
1823 generators $\boldsymbol{\Gamma}$ leading to the same D-norm: multiply $\boldsymbol{\Gamma}$ by an independent positive random
1824 variable \mathbf{U} with unit expectation, it readily gives $\|\cdot\|_{\boldsymbol{\Gamma}\mathbf{U}} = \|\cdot\|_{\boldsymbol{\Gamma}}$. Hence, to ease notations,
1825 sometimes we may prefer to write the above norm by $\|\cdot\|_D$ to focus less on the generator
1826 $\boldsymbol{\Gamma}$; in such a situation, we refer to the generator $\boldsymbol{\Gamma}$ associated with the D-norm by writing
1827 $\boldsymbol{\Gamma} \triangleleft \|\cdot\|_D$, which reads “ $\boldsymbol{\Gamma}$ generates $\|\cdot\|_D$ ”. Despite the multiple generators associated with
1828 the same D-norm, the following theorem provides a uniqueness result on the choice of $\boldsymbol{\Gamma}$
1829 when the norm of the random variable $\boldsymbol{\Gamma}$ is constrained.

1830 **Theorem 1.5** (Normed Generators, (Falk 2019, Theorem 1.7.1)). *Let $\|\cdot\|$ be an arbitrary
1831 norm on \mathbb{R}^d . For any D-norm $\|\cdot\|_D$ on \mathbb{R}^d , there exists a generator $\boldsymbol{\Gamma} \triangleleft \|\cdot\|_D$ and a constant c
1832 with the additional property $\mathbb{P}(\|\boldsymbol{\Gamma}\| = c) = 1$. The distribution of the generator is uniquely
1833 defined.*

1834 Let us remark that taking $\|\cdot\| = \|\cdot\|_1$ as the L_1 -norm, the constant c is necessarily d :
1835 indeed, $\|\boldsymbol{\Gamma}\|_1 = c$ implies $c = \mathbb{E}[\|\boldsymbol{\Gamma}\|_1] = \sum_{i \in [d]} \mathbb{E}[\boldsymbol{\Gamma}_i] = d$.

1836 **Relation with stdf**

1837 The next result is fundamental to connect a stdf to a D-norm.

1838 **Theorem 1.6** (Representation of simple max stable distributions). *A cdf G_* on \mathbb{R}^d is
1839 simple max-stable if and only if there exists a D-norm $\|\cdot\|_D$ on \mathbb{R}^d such that:*

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

1840 In particular, if G is a max-stable cdf on \mathbb{R}^d , then its stdf is given by

$$\ell_G(\mathbf{x}) = \|\mathbf{x}\|_D. \quad (1.32)$$

1841 The first statement (1.31) is taken from (Falk 2019, Theorem 2.3.3) or (Falk and Fuller
1842 2021, Theorem 4.1), while the second one (1.32) follows from Proposition 1.5.

1843 **3.D Quantization**

1844 **The quantization problem.** The quantization problem is concerned with finding the
1845 best approximation of a random vector $\mathbf{X} \in \mathbb{R}^d$ with a quantized version, *i.e.*a version
1846 that takes a discrete number of values. For a general reference on quantization, see (Graf
1847 and Luschgy 2000). Finding the best approximation of a random vector depends on the
1848 chosen measure. Define the quantization problem (Graf and Luschgy 2000, Section 10.1)
1849 of a probability measure P , for a given norm $\|\cdot\|$ (for example the Euclidean norm), as:

$$e_{N,\infty}(P) := \inf_{f: \mathbb{R}^d \rightarrow \mathbb{R}^d} \{P\text{-esssup } \|f(\mathbf{X}) - f(\mathbf{X})\| : \#f(\mathbb{R}^d) \leq N\} \quad (1.33)$$

$$\begin{aligned} &= \inf_{\boldsymbol{\alpha} \subset \mathbb{R}^d} \sup_{\mathbf{x} \in \text{supp}(P)} \min_{\mathbf{a} \in \boldsymbol{\alpha}} \|\mathbf{x} - \mathbf{a}\|, \\ &\# \boldsymbol{\alpha} \leq N \end{aligned} \quad (1.34)$$

1850 where $f : \mathbb{R}^d \rightarrow \mathbb{R}^d$ is a quantizer. Since $e_{N,\infty}(P)$ depends on the probability measure P
1851 only through its support, sometimes (see Lemma 1.1 below) we will write $e_{N,\infty}(A)$ as a
1852 function of a set $A \subset \mathbb{R}^d$ (that can be taken as the support of P).

1853 Then, if $\text{supp}(P)$ is compact, Jordan measurable with positive volume (Graf and
1854 Luschgy 2000, p.3), the following limit exists:

$$Q_\infty(\text{supp}(P)) = \lim_{N \rightarrow \infty} N^{1/d} e_{N,\infty}(P). \quad (1.35)$$

1855 Note that $Q_\infty(\text{supp}(P))$ does not depend on the law of P but only on its support. There-
1856 fore, we will interchangeably use a probability distribution or a set as input of $Q_\infty(\cdot)$. The
1857 limit $Q_\infty(\text{supp}(P))$ is called the covering coefficient or quantization coefficient of order
1858 ∞ and can be expressed in terms of the covering coefficient $Q_\infty([0, 1]^d)$. Moreover, with
1859 a further constraint, we have the following theorem:

1860 **Theorem 1.7** ((Graf and Luschgy 2000, Theorem 10.7)). *Let $A \subset \mathbb{R}^d$ be a non empty
1861 compact set with $\lambda^d(\partial A) = 0$. Let $Q_\infty([0, 1]^d)$ be the quantization coefficient of $[0, 1]^d$
1862 defined in (1.35). Then $Q_\infty([0, 1]^d) > 0$ and:*

$$\lim_{N \rightarrow \infty} N^{1/d} e_{N,\infty}(A) = Q_\infty([0, 1]^d) \lambda^d(A)^{1/d}, \quad (1.36)$$

1863 where λ^d is the Lebesgue measure in dimension d .

1864 We state the following technical lemma, which will be helpful for the proof of Theo-
1865 rem 1.3:

1866 **Lemma 1.1** ((Graf and Luschgy 2000, Lemma 10.6)).

1867 1. Let $A, B \subset \mathbb{R}^d$ be nonempty compact sets with $A \subset B$. Then,

$$\forall N \in \mathbb{N} : e_{N,\infty}(A) \leq e_{N,\infty}(B). \quad (1.37)$$

1868 2. Let N, m and $\{N_k\}_{k \in [m]}$ be positive integers such that $\sum_{k \in [m]} N_k \leq N$ and consider
1869 m nonempty compact sets $\{A_k\}_{k \in [m]}$. Then,

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

1870 Here is a particular case of (1.38) when $N \geq 2m$:

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

1871 Proof. The proof of (1.38) can be found in (Graf and Luschgy 2000, Lemma 10.6). Equation
1872 (1.39) is deduced from Equation (1.38) with $N_k = \lceil N/m \rceil - 1$ for all $k \in [m]$. \square

1873 3.E Copulas, measures of dependence

1874 **Copulas.** Let us consider a d -variate random vector \mathbf{X} from a cdf F with continuous
1875 margins F_j . Sklar's Theorem (Sklar 1959) states that there exists a unique function $C : [0, 1]^d \rightarrow [0, 1]$ such that, for all $\mathbf{x} \in \mathbb{R}^d$, $F(x_1, \dots, x_d) = C(F_1(x_1), \dots, F_d(x_d))$.

1877 Introducing for any $j \in [d]$ the uniformly distributed random variable $U_j = F_j(x_j)$,
1878 the copula C is the cdf of the joint random vector (U_1, \dots, U_d) . Copulas are a tool for
1879 studying dependence of probability objects, independently from the margins. See (Nelsen
1880 2006) for a thorough overview. As an example, the Gumbel copula is given by (see (Nelsen
1881 2006, Equation (2.4.2.))):

$$\forall \mathbf{u} \in [0, 1]^d : C_\beta(\mathbf{u}) = \exp - \left(\sum_{i \in [d]} (-\log(u_i))^\beta \right)^{1/\beta}, \quad (1.40)$$

1882 where $\beta \geq 1$.

1883 **Definition 1.8** (Kendall's τ , (Nelsen 2006, Theorem 5.1.3.)). Let C be a bivariate copula.
1884 Its associated Kendall's τ is:

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

1885 where $(U_1, U_2) \sim C$.

1886 Kendall's τ is a common measure of dependence for copulas, see (Joe 2014, Section 6.8)
1887 for an estimation procedure. Note that Kendall's τ for the Gumbel copula is given by
1888 $\tau = 1 - 1/\beta$, $\beta \geq 1$.

1889 **Wasserstein distance.** Let μ and ν be two measures on \mathbb{R}^d . Considering a cost function
1890 $c : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}^+$ and $p \geq 1$, the p - Wasserstein distance associated with the cost c

1891 between μ and ν is defined as (Villani 2009, Chap. 6):

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

1892 where $\Gamma(\mu, \nu)$ is the set of *couplings* of μ and ν , *i.e.* the set of distributions on $\mathbb{R}^d \times \mathbb{R}^d$
1893 such that the marginals are respectively μ and ν . The Wasserstein distance suffers from
1894 the fact that there exists no closed formula to compute it, nor very accessible computable
1895 estimates in dimensions higher than one. In order to perform estimation, we therefore rely
1896 on the sliced-Wasserstein distance, which is an efficient proxy, much used for computations
1897 in experimental settings (Nadjahi 2021). It is computed by randomly projecting the two
1898 compared measures on axes defined by directions \mathbf{v} and averaging over the projections.
1899 Specifically, the sliced-Wasserstein distance is defined as:

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

1900 where \mathbb{S}^{d-1} is the Euclidean unit sphere, $\mathbf{v} \# \mu$ (respectively $\mathbf{v} \# \nu$) stands for the pushforward
1901 of the projection $\mathbf{X} \in \mathbb{R}^d \mapsto \langle \mathbf{X}, \mathbf{v} \rangle \in \mathbb{R}$, where $\mathbf{X} \sim \mu$ (respectively ν).

1902 3.F Proofs of claims

1903 Proof of Proposition 1.3

1904 The following states an interesting corollary of Theorem 1.2, which we are the first to prove
1905 formally as far as we know.

1906 **Corollary 1.1** (of Theorem 1.2). *Let $\mathbf{Z} \in \mathbb{R}^N$ be a positive random vector with continuous
1907 cdf $F_{\mathbf{Z}} \in \text{Dom}(G_{\Gamma_{\mathbf{Z}}})$ where $G_{\Gamma_{\mathbf{Z}}}$ is a simple max-stable distribution *i.e.* $F_{\mathbf{Z}}$ satisfies (1.7)
1908 and (1.8). Consider $\Phi \in \mathbb{H}_1([0, \infty)^N, [0, \infty)^d)$, such that:*

$$\mathbb{E}[\Phi(\Gamma_{\mathbf{Z}})] > 0. \quad (1.43)$$

1909 Let $\mathbf{X} = \Phi(\mathbf{Z})$. Then, $\Phi(\Gamma_{\mathbf{Z}})/\mathbb{E}[\Phi(\Gamma_{\mathbf{Z}})] =: \Gamma_{\mathbf{X}}$ is a valid generator of a D-norm and
1910 $F_{\mathbf{X}} \in \text{Dom}(G_{\Gamma_{\mathbf{X}}})$ where $G_{\Gamma_{\mathbf{X}}}$ is a simple max-stable distribution. Moreover, the normalizing
1911 constants to ensure convergence in (1.24) are $\mathbf{a}_t = t \mathbb{E}[\Phi(\Gamma_{\mathbf{Z}})]$ and $\boldsymbol{\theta} = \mathbf{0}$.

1912 *Proof.* In view of the characterization of Theorem 1.2, consider any $h \in \mathbb{H}_1([0, \infty)^d, [0, \infty))$
1913 and

$$\Psi_h : \begin{cases} [0, \infty)^N & \rightarrow [0, \infty), \\ \mathbf{z} & \mapsto h\left(\frac{\Phi(\mathbf{z})}{\mathbb{E}[\Phi(\Gamma_{\mathbf{Z}})]}\right). \end{cases} \quad (1.44)$$

1914 Clearly, Ψ_h is continuous 1-homogeneous as composition of the continuous 1-homogeneous
1915 functions Φ and h . Since $F_{\mathbf{Z}}$ verifies (1.8), it follows that:

$$t \mathbb{P}\left(h\left(\frac{\Phi(\mathbf{Z})}{\mathbb{E}[\Phi(\Gamma_{\mathbf{Z}})]}\right) > t\right) = t \mathbb{P}(\Psi_h(\mathbf{Z}) > t) \xrightarrow{t \rightarrow \infty} \mathbb{E}[\Psi_h(\Gamma_{\mathbf{Z}})] = \mathbb{E}\left[h\left(\frac{\Phi(\Gamma_{\mathbf{Z}})}{\mathbb{E}[\Phi(\Gamma_{\mathbf{Z}})]}\right)\right]. \quad (1.45)$$

1916 Introducing $\mathbf{Y} = \Phi(\mathbf{Z})/\mathbb{E}[\Phi(\mathbf{\Gamma}_Z)] = \mathbf{X}/\mathbb{E}[\Phi(\mathbf{\Gamma}_Z)]$ and its corresponding distribution
1917 function F_Y , the previous convergence can be rewritten as:

$$\forall h \in \mathbb{H}_1([0, \infty)^d, [0, \infty)) : t \mathbb{P}(h(\mathbf{Y}) > t) \xrightarrow[t \rightarrow \infty]{} \mathbb{E}[h(\mathbf{\Gamma}_X)], \quad (1.46)$$

1918 so that \mathbf{Y} satisfies characterization (1.8) in Theorem 1.2. Moreover, since $\mathbb{E}[\mathbf{\Gamma}_X] = \mathbf{1}$
1919 and the coordinates of $\mathbf{\Gamma}_X$ are non-negative, $\mathbf{\Gamma}_X$ is indeed the generator of a D-norm.
1920 Therefore:

$$\forall \mathbf{x} \in (0, \infty)^d : F_Y^t(t\mathbf{x}) \xrightarrow[t \rightarrow \infty]{} \exp - \left\| \frac{1}{\mathbf{x}} \right\|_{\mathbf{\Gamma}_X}. \quad (1.47)$$

1921 Noting that $F_Y(\mathbf{x}) = F_X(\mathbb{E}[\Phi(\mathbf{\Gamma}_Z)] \mathbf{x})$ finally proves

$$\forall \mathbf{x} \in (0, \infty)^d : F_X^t(t \mathbb{E}[\Phi(\mathbf{\Gamma}_Z)] \mathbf{x}) \xrightarrow[t \rightarrow \infty]{} \exp - \left\| \frac{1}{\mathbf{x}} \right\|_{\mathbf{\Gamma}_X} =: G_{\mathbf{\Gamma}_X}(\mathbf{x}), \quad (1.48)$$

1922 and the result is proved. \square

1923 We now complete the proof of Proposition 1.3. The previous result holds for continuous
1924 1-homogeneous function Φ , which is not exactly the situation with a neural network with
1925 ReLU activation functions because of the bias terms (see Proposition 1.2). However, we
1926 claim that, asymptotically, these bias terms do not play any role on the stdf. To see this,
1927 let us investigate the behavior of a ReLU-neural network when going from a layer to the
1928 next one.

1929 Denoting by σ the ReLU activation function, and supposing that $\varphi_l = \sigma$, $\forall l \in [L]$, we
1930 have

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

1931 On the one hand, $\sigma(\mathbf{W}_l \mathbf{Z}_{l-1})$ is a 1-homogeneous function of \mathbf{Z}_{l-1} and is thus a heavy-tailed random variable if $\mathbf{W}_l \neq \mathbf{0}$ and \mathbf{Z}_{l-1} is heavy-tailed itself. On the other hand,
1932 the remaining term $[\dots]$ in (1.49) is a bounded random variable (bounded by $|\mathbf{b}_l|$); the
1933 stdf of the sum of a heavy-tailed random variable and a bounded one is the stdf of the
1934 heavy-tailed random variable. The stdf of (1.49) is therefore the stdf of the transformation
1935 without the bias, and by induction the stdf of the output is the stdf of the input after
1936 iterative applications of $\sigma(\mathbf{W}_l \cdot)$ for $l \in [L]$, which is 1-homogeneous continuous. Let
1937 denote by Φ the output of the L compositions of $\sigma(\mathbf{W}_l \cdot)$: since the matrix weights \mathbf{W}_l
1939 are non-negative, $\Phi \in \mathbb{H}_1([0, \infty)^N, [0, \infty)^d)$. Therefore, Corollary 1.1 applies and so the
1940 stdf associated with the output of the network is $\mathbf{\Gamma}_{G_Z} = \Phi(\mathbf{\Gamma}_Z)/\mathbb{E}\Phi(\mathbf{\Gamma}_Z)$, which has at
1941 most N atoms. Thus, the proof is concluded.

1942 3.G Proof of Proposition 1.4

1943 The next lemma can be interpreted in the following way. If a discrete random vector
1944 generates a D-norm, then it admits a discrete counterpart which can be any other discrete

¹⁹⁴⁵ distribution with modified atoms: it is also a generator of the concerned D-norm.

¹⁹⁴⁶ **Lemma 1.2.** *Let $\Gamma \triangleleft \|\cdot\|_D$ where $\Gamma \in [0, \infty)^d$ is discrete, that is $\Gamma \sim \mu_\Gamma$ where*

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

¹⁹⁴⁷ Consider a new discrete probability measure with positive weights $(q_i)_{i \in [N]}$ and a new ran-
¹⁹⁴⁸ dom variable $\tilde{\Gamma} \sim \mu_{\tilde{\Gamma}}$, where

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

¹⁹⁴⁹ with $\tilde{\gamma}^{(i)} = \frac{p_i}{q_i} \gamma^{(i)}$, $i \in [N]$. Then, $\tilde{\Gamma}$ is a valid generator and $\tilde{\Gamma} \triangleleft \|\cdot\|_D$.

¹⁹⁵⁰ *Proof.* By definition of the D-norm, one has $\forall \mathbf{x} \in \mathbb{R}^N$:

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

¹⁹⁵¹ Since $\mathbb{E}_{\tilde{\Gamma}} [\tilde{\Gamma}] = \mathbb{E}_\Gamma [\Gamma] = \mathbf{1}$ and $\tilde{\Gamma} \in [0, \infty)^d$, $\tilde{\Gamma}$ is a valid generator. □

¹⁹⁵² In particular, it is possible to choose uniform weights, $q_i = 1/N$, $i \in [N]$. In other
¹⁹⁵³ words, if a D-norm admits a generator which admits a finite number of values, it also
¹⁹⁵⁴ admits a generator with the same number of finite values, with uniform probability. This
¹⁹⁵⁵ result is important as it means that the class of generators of D-norms which take a finite
¹⁹⁵⁶ number of values and with uniform weights is as rich as the class of generator which take
¹⁹⁵⁷ finite values with arbitrary weights.

¹⁹⁵⁸ **Proposition 1.6.** *Consider a generator Γ of a D-norm in \mathbb{R}^d . Suppose that it is discrete
¹⁹⁵⁹ uniform with N atoms $(\gamma^{(i)})_{i \in [N]}$:*

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

¹⁹⁶⁰ Consider moreover a random vector $\mathbf{Z} \in \mathbb{R}^N$ with i.i.d. unit Fréchet margins: $\mathbf{Z} = (Z_1, \dots, Z_N)$.
¹⁹⁶¹ Then, there exists a linear mapping $\Phi : \mathbb{R}^N \rightarrow \mathbb{R}^d$ such that:

$$F_{\Phi(\mathbf{Z})} \in \text{Dom}(G_\Gamma). \quad (1.53)$$

¹⁹⁶² *Proof.* Let us first remark that $F_{\mathbf{Z}} \in \text{Dom}(G_{\Gamma_Z})$ with:

$$\mu_{\Gamma_Z} = \frac{1}{N} \sum_{i \in [N]} \delta_{N \mathbf{e}_i},$$

1963 see Example 1.1. Consider $\boldsymbol{\Gamma} \in \mathbb{R}^N$ a valid generator of a D-norm with a discrete uniform
1964 distribution (1.52). There exists $\Phi : \mathbb{R}^N \rightarrow \mathbb{R}^d$ 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.
1965 Moreover, remark that $\forall i \in [N] : \gamma^{(i)} \in [0, \infty)^d$ implies $\Phi([0, \infty)^N) \subset \Phi([0, \infty)^d)$ and
1966 thus $\Phi(\mathbf{Z}) \in [0, \infty)^d$. Corollary 1.1 ensures that

$$F_{\Phi(\mathbf{Z})} \in \text{Dom}(G_{\Phi(\boldsymbol{\Gamma}_{\mathbf{Z}})/\mathbb{E}(\Phi(\boldsymbol{\Gamma}_{\mathbf{Z}}))}) = \text{Dom}(G_{\boldsymbol{\Gamma}}) \quad (1.54)$$

1968 since, as $\Phi(\mathbf{e}_i) = \gamma^{(i)}$, $\Phi(\boldsymbol{\Gamma}_{\mathbf{Z}})$ is equal in distribution to $\boldsymbol{\Gamma}$ (note that $\mathbb{E}[\Phi(\boldsymbol{\Gamma}_{\mathbf{Z}})] = \mathbf{1}$).
1969 The result is thus proved. \square

1970 *Remark 1.1.* This result ensures that, given any random vector of interest \mathbf{X} for which the
1971 associated D-norm generator $\boldsymbol{\Gamma}_{\mathbf{X}}$ has a uniform distribution with N atoms, one can find a
1972 linear map Φ transforming a noise with unit Fréchet margins with at least N atoms to a
1973 distribution whose D-norm generator is the desired one.

1974 Lemma 1.2 and Proposition 1.6 together prove Proposition 1.4. \square

1975 3.H Proof of Theorem 1.3

1976 The following technical lemma will be useful to prove the result:

1977 **Lemma 1.3.** $\forall \mathbf{a}, \mathbf{b}, \mathbf{c} \in \mathbb{R}^d$:

$$\left| \max_{i \in [d]} a_i b_i - \max_{i \in [d]} a_i c_i \right| \leq \|\mathbf{a}\|_{\infty} \max_{i \in [d]} |b_i - c_i|. \quad (1.55)$$

1978 *Proof.* Without loss of generality, suppose that $\max_{i \in [d]} a_i b_i \geq \max_{i \in [d]} a_i c_i$, the argument
1979 being symmetric. One has:

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

1980 and the result is proved. \square

1981 We seek to find a upper bound on:

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

1982 In view of the above, and remembering that the stdf of the output of a neural network
1983 is discrete (Proposition 1.3), our goal is to find the best approximation of the D-norm
1984 $\|\cdot\|_{\boldsymbol{\Gamma}}$ with a discretized generator with N atoms; we will leverage results from quantization

1985 (Section 3.D). Take $\boldsymbol{\Gamma}$ an arbitrary generator of the D-norm related to ℓ_F . One has:

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

$$\leq \min_{f: \mathbb{R}^d \rightarrow \mathbb{R}^d, \#f(\mathbb{R}^d)=N} \| \mathbf{x} \|_{\infty} \mathbb{E}_{\boldsymbol{\Gamma}} \left[\max_{i \in [d]} |\boldsymbol{\Gamma}_i - f(\boldsymbol{\Gamma}_i)| \right] \quad (1.58)$$

$$\leq \| \mathbf{x} \|_{\infty} e_{N,\infty}(\boldsymbol{\Gamma}), \quad (1.59)$$

1986 from Lemma 1.3. To bound (1.59), we use a quantizer which has a constant sup-norm,
1987 which existence is guaranteed by (Theorem 1.5):

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

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

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

1988 Now, we are in a position to derive a bound on $e_{N,\infty}(\boldsymbol{\Gamma})$.

1989 **Definition 1.9.** For any norm $\|\cdot\|$, let us denote by $\mathcal{B}_{\|\cdot\|}$ the associated unit ball. Besides,
1990 let $\mathcal{B}_{\|\cdot\|}^+ = \mathcal{B}_{\|\cdot\|} \cap [0, \infty)^d$ be the part of the unit ball in the positive orthant.

1991 Owing to (1.60) and Definition 1.2, $\text{supp}(\boldsymbol{\Gamma}) \subset c \cdot [0, 1]^d$. Equation (1.36) shows that
1992 one can get an error bound for (1.59) with rate $N^{-1/d}$. However, it is possible to improve
1993 this result by remarking that $\boldsymbol{\Gamma}$ takes values on the boundary of a hypercube (see (1.60)).
1994 Indeed, now note that:

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

1995 where, for all $i \in [d]$,

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

1996 *Remark 1.2.* For all $i \in [d]$, $\mathcal{B}_{\|\cdot\|_{\infty}}^+(i)$ is canonically homeomorphic to $[0, 1]^{d-1}$ in the topo-
1997 logical space \mathbb{R}^{d-1} endowed with the topology of $\|\cdot\|_{\infty}$.

1998 Using Lemma 1.1 (inequality (1.37) with (1.61) from first to second term and from
1999 second to third and inequality (1.38) from third to fourth), we get

$$e_{N,\infty}(\boldsymbol{\Gamma}) \leq c \cdot e_{N,\infty}(\mathcal{B}_{\|\cdot\|_{\infty}}^+) \leq c \cdot e_{\lceil N/d \rceil - 1, \infty}(\mathcal{B}_{\|\cdot\|_{\infty}}^+(1)) = c \cdot e_{\lceil N/d \rceil - 1, \infty}([0, 1]^{d-1}). \quad (1.63)$$

2000 Gathering previous arguments, we have proved the following:

2001 **Lemma 1.4.** Consider $\boldsymbol{\Gamma} \in \mathbb{R}^d$ a generator of a D-norm such that $\| \boldsymbol{\Gamma} \|_{\infty} = c$ a.s.. Then,

2002 the following holds true:

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

2003 From (1.36), we get that:

$$e_{\lceil N/d \rceil - 1, \infty} ([0, 1]^{d-1}) \xrightarrow{N \rightarrow \infty} (\lceil N/d \rceil - 1)^{-1/(d-1)} Q_{\infty} ([0, 1]^{d-1}) \lambda([0, 1]^{d-1})^{1/(d-1)} \quad (1.65)$$

$$\xrightarrow{N \rightarrow \infty} d^{1/(d-1)} N^{-1/(d-1)} Q_{\infty} ([0, 1]^{d-1}) \quad (1.66)$$

2004 which proves (1.14), where $C(d) = c \cdot d^{1/(d-1)} Q_{\infty} ([0, 1]^{d-1})$ in (1.15). The proof of
2005 Theorem 1.3 is thus complete. \square

2006 Observe that the quantization upper bound (1.59) is a bit rough since the right-hand
2007 side of (1.58) refers to the L_1 -quantization that we bound using L_{∞} -quantization. One
2008 could use L_1 -quantization estimates but they would depend on the distribution of Γ and we
2009 prefer to provide worst-case estimates. Instead of using the L_{∞} and using L_1 quantization
2010 on (1.58), (Graf and Luschgy 2000, Th. 6.2.) gives that the first order error of the error
2011 is proportional to the $L_{(d-1)/d}$ norm of the density of the generator on the boundaries of
2012 $\mathcal{B}_{\|\cdot\|_{\infty}}^+ (i)$ times $N^{-1/(d-1)}$. In particular, if the generator Γ has a null density with respect
2013 to the Lebesgue measure (e.g. if the distribution is discrete), the multiplicative factor of the
2014 $N^{-1/(d-1)}$ term is null and the convergence of the bound can be quicker than $N^{-1/(d-1)}$.

2015 3.1 Additional experiments

2016 Experiments on Hüsler-ReiSS and Gaussian models

2017 **Experimental setup.** We consider dimensions $d \in \{2, 5, 10, 20, 50\}$. In the Gaussian
2018 copula model, the equicorrelation parameter is fixed at $\rho = 0.5$. In the Hüsler-ReiSS case,
2019 the equi- λ model is adopted with $\lambda = 1.0$ (*i.e.* $\lambda_{ij} = 1$ for $i \neq j$). In both datasets,
2020 the margins are Pareto distributed with tail index $\alpha \in \{1.5, 2.0, 2.5\}$ (implemented by
2021 inverse-cdf transforms after simulating the copula). For each pair (d, α) and dataset type,
2022 we generate 10,000 training and 2,000 validation points. Results are provided in Table 1.3
2023 of the main paper and Figures 1.7, 1.8, 1.9 and 1.10.

2024 3.3 Additional plots for experiments on Real data

2025 We provide additional illustrations for assessing the performance of HTGAN for real data,
2026 on the Financials dataset in Figure 1.11. Real vs generated data are plotted with: (i)
2027 histograms of marginal densities for four tickers, (ii) scatter plots for two tickers and (iii)
2028 histograms of angles of large data points after projection for two tickers. For scatter
2029 plots and angles, two tickers are selected such that they exhibit quite different values
2030 of Kendall's τ , chosen on the relevant heatmap in Figure 1.5. The selected securities
2031 are: Ameriprise Financial (AMP), Franklin Ressources (BEN), BNY Mellon (BK) and
2032 BlackRock (BLK). This contrast in dependence is reflected in the corresponding scatter

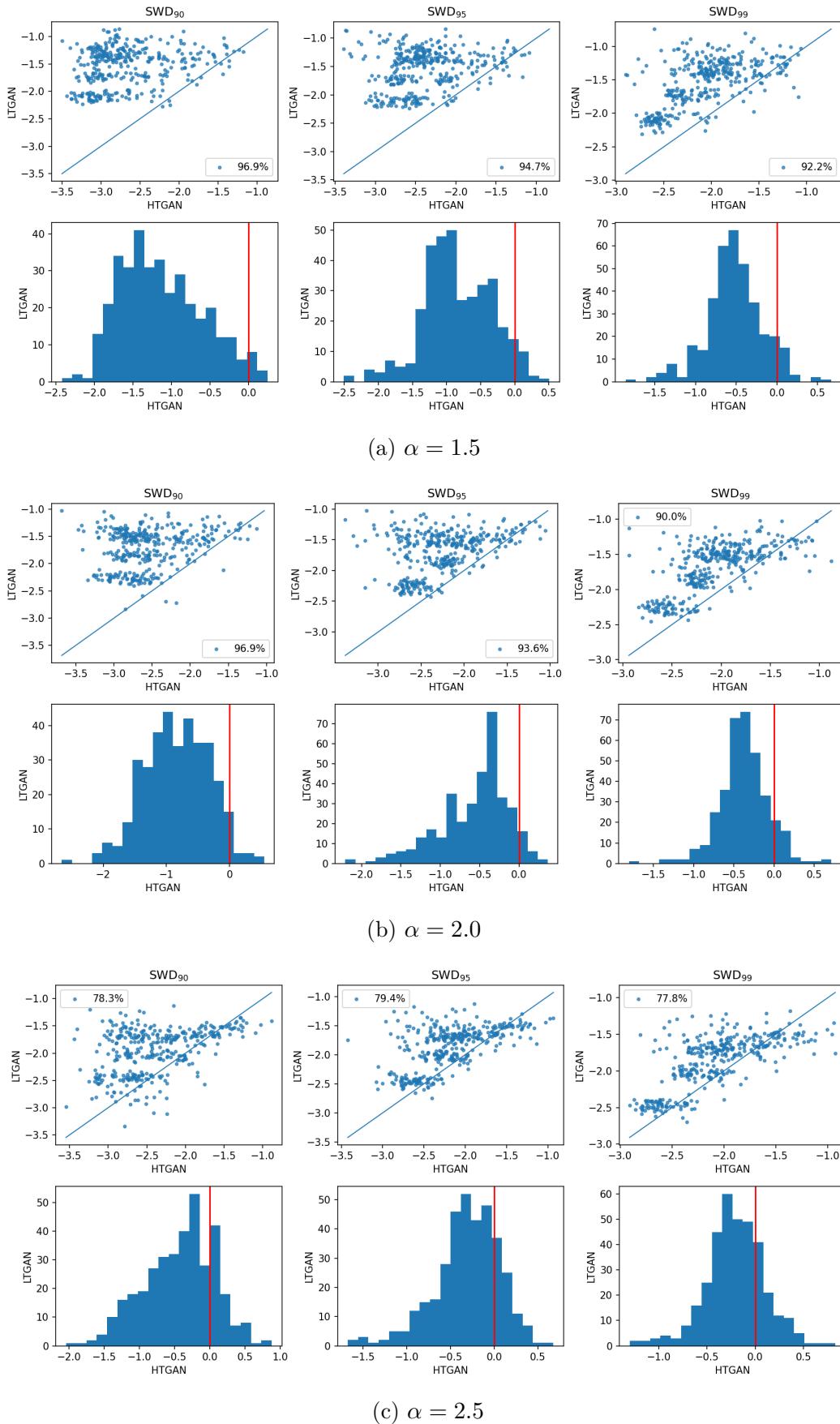


Figure 1.7: Illustration in a d -dimensional setting on simulated data of size $n_{\text{data}} = 10,000$ from a Gaussian copula and Pareto margins, $d \in \{2, 5, 10, 20, 50\}$. Results obtained with the SWD metric. See Figure 1.3 for further details.

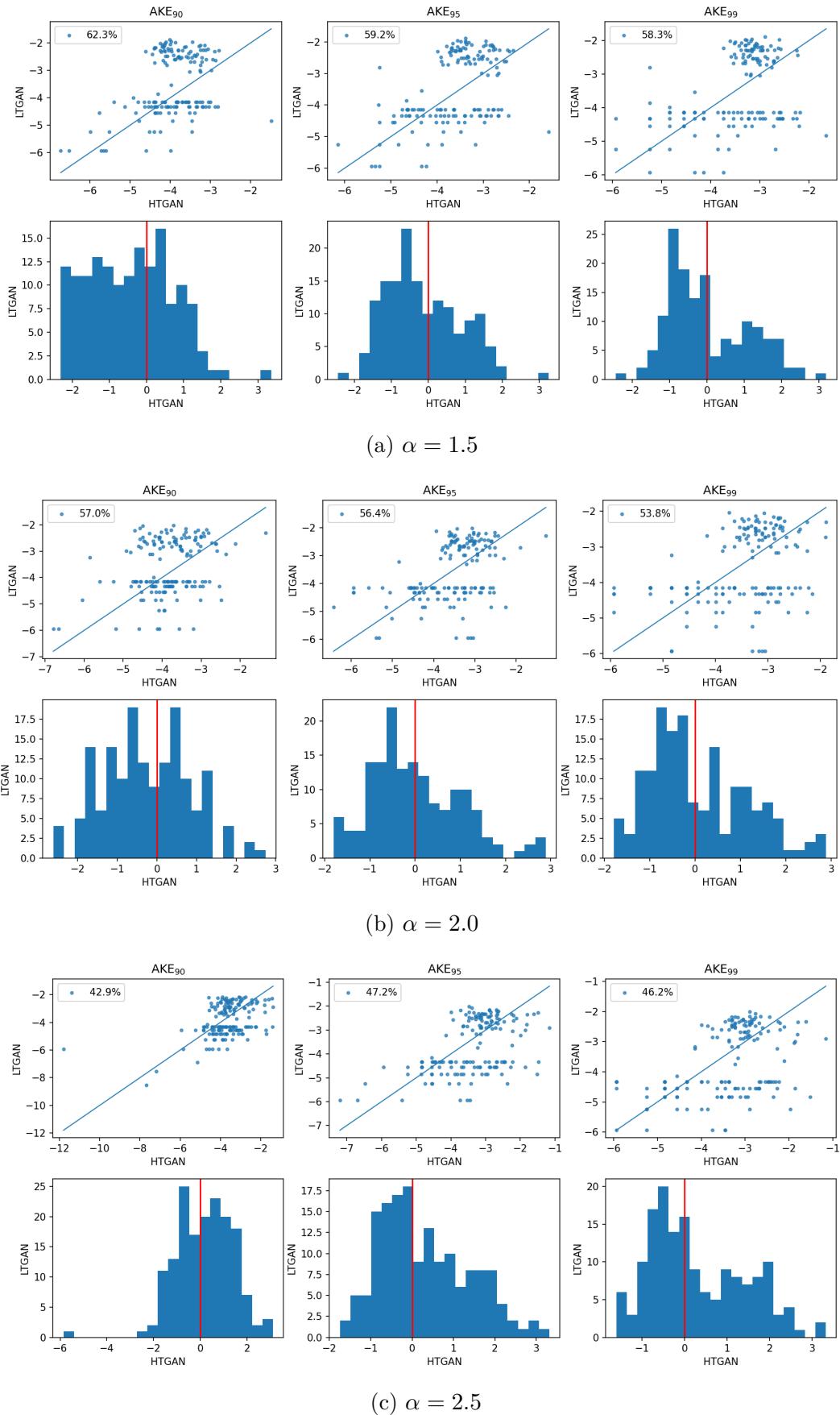


Figure 1.8: Illustration in a d -dimensional setting on simulated data of size $n_{\text{data}} = 10,000$ from a Gaussian copula and Pareto margins, $d \in \{2, 5, 10, 20, 50\}$. Results obtained with the AKE metric. See Figure 1.3 for further details.

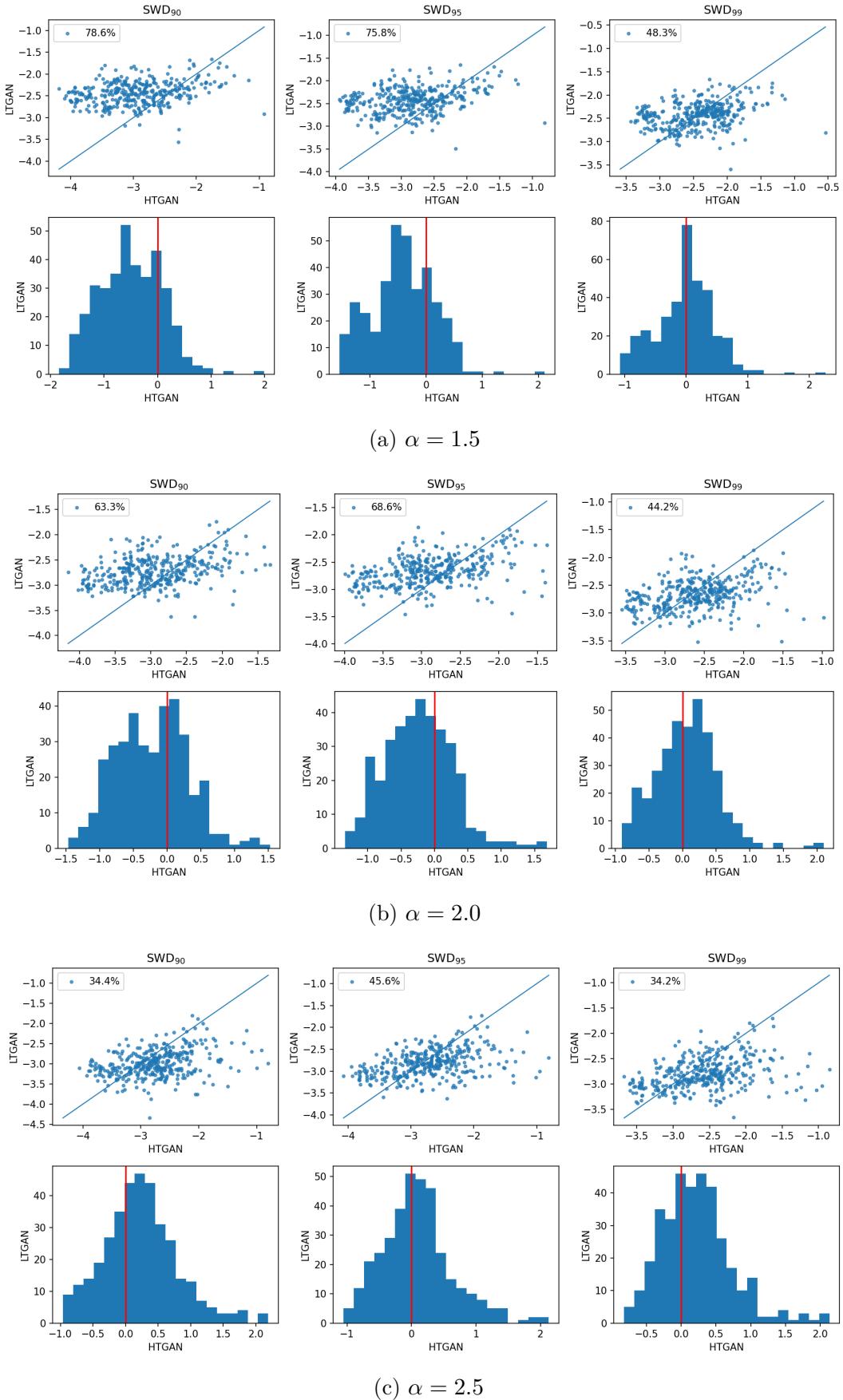


Figure 1.9: Illustration in a d -dimensional setting on simulated data of size $n_{\text{data}} = 10,000$ from a Hüsler-ReiSS copula and Pareto margins, $d \in \{2, 5, 10, 20, 50\}$. Results obtained with the SWD metric. See Figure 1.3 for further details.

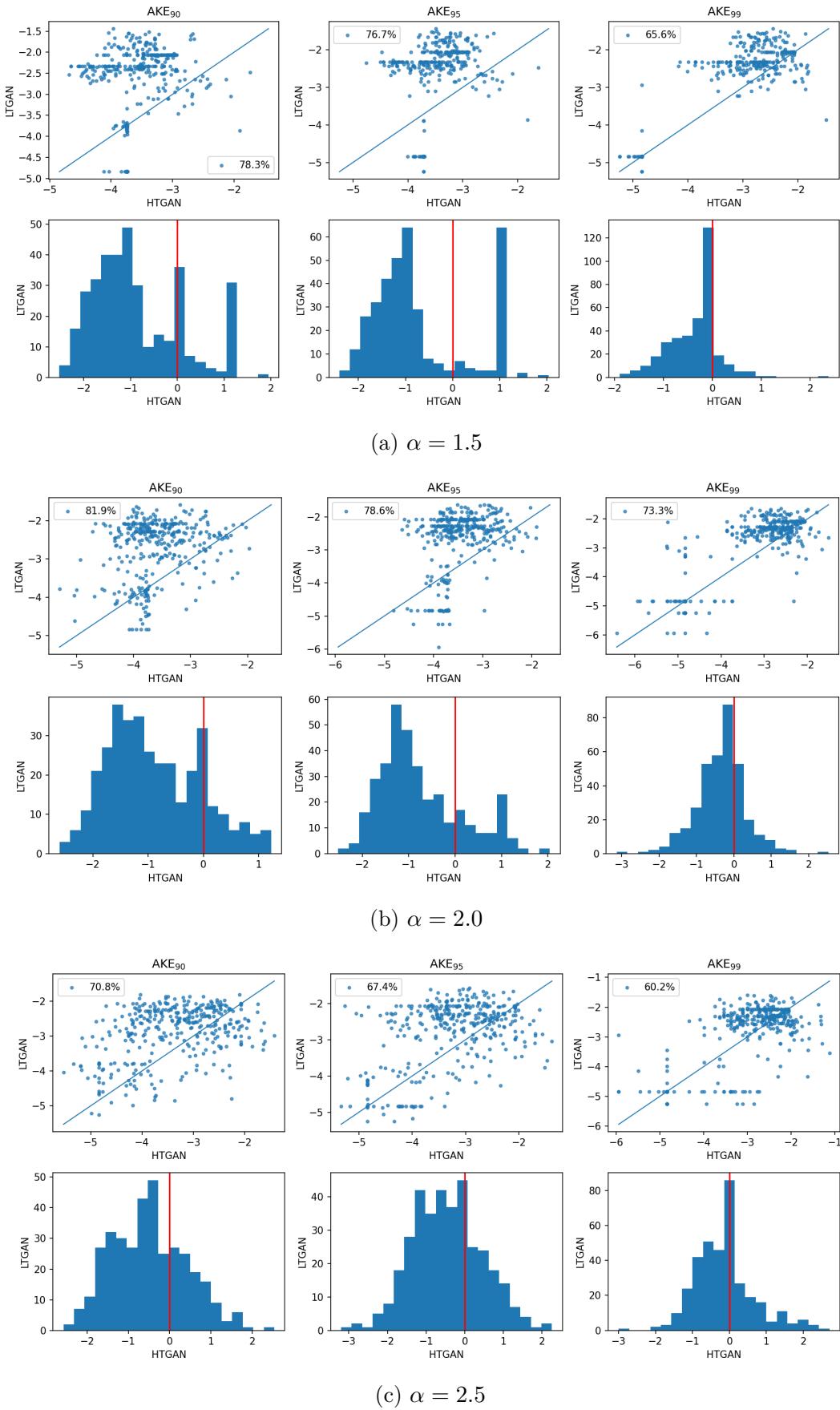


Figure 1.10: Illustration in a d -dimensional setting on simulated data of size $n_{\text{data}} = 10,000$ from a Hüsler-ReiSS copula and Pareto margins, $d \in \{2, 5, 10, 20, 50\}$. Results obtained with the AKE metric. See Figure 1.3 for further details.

2033 plots and angular distributions. These illustrations highlight HTGANs capacity to capture
2034 various and complex asymptotic dependence structures - even if not perfectly - despite the
2035 challenges posed by the high-dimensional setting and the limited sample size.

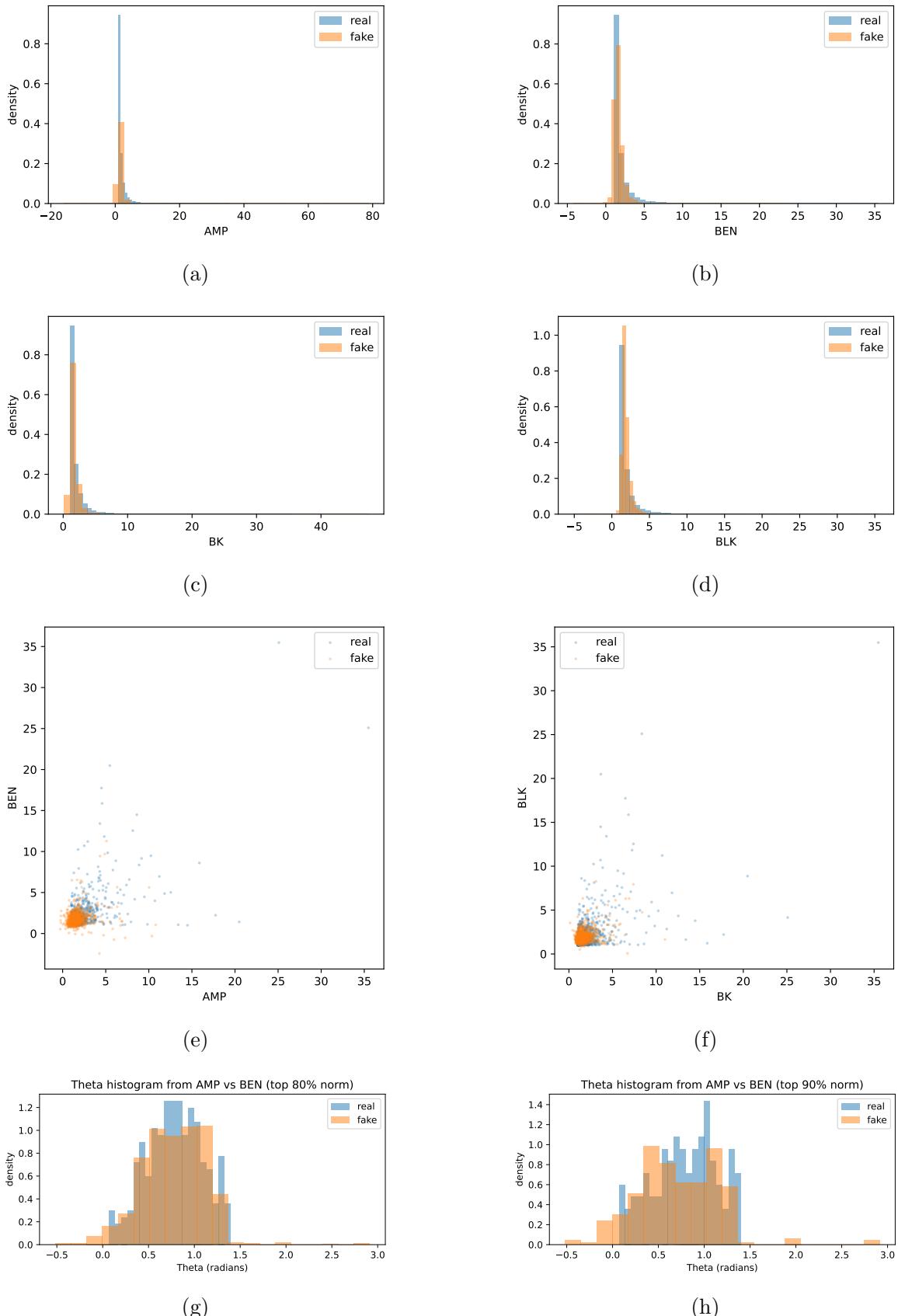


Figure 1.11: (a)-(d): histograms of real vs synthetic marginals across dimensions one to four. (e)-(f): Scatter plot AMF vs BEN and BK vs BLK tickers. (g)-(h): Histogram of θ after selecting largest values, at level 80% and 90%.

2036 **Chapter 2**

2037 **Generative Neural Order
Statistics**

2039 This chapter develops the theory and algorithms for generating order statistics using neural
2040 networks. We analyze the representational capabilities of neural network-based generative
2041 models for sorted data, deriving approximation bounds based on the Sukhatme and Schu-
2042 cany representations.

2043 **2.1 Introduction**

2044 **Sustainable investing.** Sustainable finance refers to the process of taking environmen-
2045 tal, social and governance (ESG) considerations into account when making investment
2046 decisions in the financial sector, leading to more long-term investments in sustainable eco-
2047 nomic activities and projects, as defined by the European Commission ([2024a](#)). Within
2048 sustainable finance, a particular quantitative framework, impact investing, is broadly de-
2049 fined as investments that consider not only financial objectives but also other goals such as
2050 supporting certain social priorities and agendas (Lo and R. Zhang [2021](#)). Impact investing
2051 seeks to optimize both financial objectives and metrics that quantify the compliance level
2052 of assets with sustainable goals. For example, ratings on ESG factors provide information
2053 about the sustainability performance of a company or a financial instrument, by assess-
2054 ing its exposure to sustainability risks and/or its impact on people and the environment
2055 (European Commission [2024b](#)).

2056 **Impact investing, order statistics and induced order statistics.** Recent develop-
2057 ments in Mathematical finance and economics have tackled the challenging task of trans-
2058 lating sustainable investing into a mathematical framework. Among those, the work in
2059 (Lo and R. Zhang [2021](#)) introduces a general framework for quantifying [the](#) impact of im-
2060 pact investing. Namely, authors propose to formalize impact investing as the sorting and
2061 selection of an investment universe of N securities based on an *impact factor*, typically the
2062 ESG score, $\mathbf{X} = (X_1, \dots, X_N)^T$ and a *performance metric*, typically the return associated

with an asset, $\Theta = (\Theta_1, \dots, \Theta_N)$. In this framework, \mathbf{X} and Θ 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 \mathbf{X} varies with time, denoted \mathbf{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 $\{(X_1, \Theta_1), \dots, (X_N, \Theta_N)\}$, the order statistics of the impact factor \mathbf{X} is defined as a random permutation σ of its elements satisfying:

$$X_{\sigma(1)} \leq \dots \leq X_{\sigma(N)}. \quad (2.1)$$

Recall that throughout the paper, we suppose that the realizations X_1, \dots, X_N are **i.i.d.**. Note that if the distribution of \mathbf{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, \dots, 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:

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

An *impact portfolio strategy*, as defined in (Lo and R. Zhang 2021), is a strategy that considers both the performance metric Θ and the impact factor \mathbf{X} . Typically, an impact driven investor is willing to invest in stocks that have high scores for the impact factor (high values of \mathbf{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, \mathbf{X} and Θ are observed over a period of 30 years. If the case where daily data are observable, it amounts to roughly around $30 \text{ days} \times 12 \text{ months} \times 30 \text{ years} \approx 10,000$ data 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 \mathbf{Z} such that:

$$\mathcal{G}(\mathbf{Z}) \stackrel{d}{=} ((X_{1:N}, \Theta_{[1:N]}), \dots, (X_{N:N}, \Theta_{[N:N]})) \quad (2.3)$$

where $\stackrel{d}{=}$ 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:

$$\mathcal{G}(\mathbf{Z}) \stackrel{d}{=} ((X_{N:N}, \Theta_{[N:N]}), \dots, (X_{N-d+1:N}, \Theta_{[N-K:N]})), \quad (2.4)$$

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

2134 3.A Notations

2135 We summarize the main notations used throughout the paper.

- 2136 • **Indices and sets:** For an integer $N \geq 1$, we write $[N] \stackrel{\text{def}}{=} \{1, 2, \dots, N\}$. The symbol
2137 $k \in [N]$ denotes an index.
- 2138 • **Scalars, vectors, matrices:** Lowercase letters (x, y, u, \dots) denote scalars. Bold
2139 $(\mathbf{x}, \mathbf{X}, \mathbf{y}, \mathbf{Y}, \dots)$ denote column vectors. The i -th component of \mathbf{x} is x_i .
- 2140 • **Probability:** \mathbb{P} and \mathbb{E} denote probability and expectation. Equality in distribution
2141 is written $X \stackrel{d}{=} Y$. The indicator of an event A is $\mathbb{I}\{A\}$. The cumulative distribution
2142 of X is commonly written F_X , except if it is explicitly stated otherwise.
- 2143 • **Spaces and functions:** \mathbb{R} is the set of real numbers, \mathbb{N} the set of positive integers.
2144 For a cumulative distribution function (cdf) F , F^{-1} denotes the (left-continuous)
2145 quantile function. We use $\ln(\cdot)$ for the natural logarithm.
- 2146 • **Order statistics and induced order statistics (OS/IOS):** For i.i.d. $X_1, \dots, X_N \in$
2147 \mathbb{R} with cdf F_X , the order statistics are $X_{1:N} \leq \dots \leq X_{N:N}$, where $X_{k:N}$ is the k -th
2148 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
2149 attached to the observation whose X -value ranks r th; see Definition 2.1.
- 2150 • **Uniforms and exponentials:** U denotes a standard uniform random variable on
2151 $(0, 1)$. $\mathcal{E}(1)$ denotes the unit exponential law with density $f(z) = e^{-z} \mathbb{I}\{z \geq 0\}$. We
2152 write $Z \sim \mathcal{E}(1)$ and use $F(z) = 1 - e^{-z}$, $F^{-1}(u) = -\ln(1 - u)$.
- 2153 • **Neural networks:** G_{NN} denotes a neural network generator mapping i.i.d. inputs
2154 (e.g., uniforms) to outputs. When a family of networks is used, superscripts indicate
2155 roles, e.g., F_{NN} and F_{NN}^{-1} for approximations of F and F^{-1} .
- 2156 • **Asymptotics:** $a_N \sim b_N$ means $\lim_{N \rightarrow \infty} a_N/b_N = 1$. Big- \mathcal{O} notation is standard.

2157 2.2 A theoretical analysis of generative modeling of 2158 order statistics with Neural Networks

2159 Throughout this section, we let N denote the dimension of the vector to be sorted for
2160 obtaining order statistics. For $k \in [N]$, the k^{th} order statistic is denoted by $X_{k:N}$. We
2161 leverage these representations to analyze the capacity of neural networks to model order
2162 statistics (OS) and induced order statistics (IOS). Formally, we seek a neural network G_{NN}
2163 such that

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

2164 where U_1, \dots, U_N are independent standard uniform random variables. To address this,
2165 we consider several possible constructions. We first recall the following result:

2166 *Proposition 2.1* (H. David and H. Nagaraja (2004), eq. (2.3.7), p. 15). Let $X \in \mathbb{R}$ be a
2167 random variable with arbitrary distribution function F_X . Let U be a standard uniform
2168 random variable. Then:

$$(X_{1:N}, \dots, X_{N:N}) \stackrel{d}{=} (F_X^{-1}(U_{1:N}), \dots, F_X^{-1}(U_{N:N})), \quad (2.6)$$

2169 This proposition shows that, up to transformation by the inverse cdf, the joint distri-
2170 bution of order statistics from any i.i.d. sample is determined by the order statistics of the
2171 uniform distribution. Therefore, in this section, we focus solely on approximation results
2172 for the order statistics of the uniform distribution.

2173 3.A Representational results for order statistics

2174 We summarize two key sampling schemes for order statistics, with detailed proofs provided
2175 in Section 2.B.

2176 **Schucany's scheme.** Order statistics exhibit a Markov property. Let U_1, \dots, U_N be
2177 i.i.d. standard uniform random variables. Following (Schucany 1972), the order statistics
2178 of the uniform distribution can be recursively represented as

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

2179 **Sukhatme's representation.** Another important representation, due to (Sukhatme
2180 1937), also exhibits a Markov chain-like structure. Let Z_1, \dots, Z_N be i.i.d. standard ex-
2181 ponential random variables, with density

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

2182 Let $F(z) = 1 - e^{-z}$ denote the associated cdf. Then, for all $r = 1, \dots, N$,

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

2183 where Y_1, \dots, Y_N are i.i.d. standard exponential random variables and $Z_{0:N} = 0$ by con-
2184 vention. Applying Proposition 2.1, and noting that $F^{-1}(u) = -\ln(1-u)$ for $u \in (0, 1)$,
2185 we have

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

2186 3.B Generating order statistics: approximation results

2187 To analyze how well neural networks can approximate the mapping from independent
2188 uniforms to order statistics, we establish bounds for the uniform case.

2189 **Learning to generate order statistics based on Schucany's scheme.** We investi-
2190 giate the approximation quality of neural network-based generative schemes for uniform
2191 order statistics, using Schucany's recursive representation. The algorithm iteratively ap-
2192 plies (2.7), with neural networks $\hat{G}_{r:N}^{NN}$ approximating the functions

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

2193 Thus, (2.7) can be rewritten as

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

Algorithm 3: Sampling from order statistics $U_{1:N} \leq \dots \leq U_{N:N}$.

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

U_1, \dots, U_N

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

2194 1 Set $\hat{U}_{N:N} = U_N^{1/N}$;
2 **for** $r \leftarrow 1$ **to** N **do**
3 $\hat{U}_{N-r:N} \leftarrow \hat{G}_{N-r:N}^{NN}(\hat{U}_{N-r+1:N}, U_r);$
 // In parallel: $U_{N-r:N} \leftarrow U_{N-r+1:N} U_r^{1/(N-r)}$.
4 **return** $\hat{U}_{r:N}$ and $U_{r:N}$, for $1 \leq r \leq N$.

2195 With this procedure, we obtain the following result:

2196 **Theorem 2.1.** Let $\varepsilon > 0$. There exists a family of *ReLU Neural Networks* $G_{N-r:N}^{NN}$ for
2197 $1 \leq r \leq N$ such that, using the sampling procedure of Algorithm 3 and setting

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

2198 we have $\mathbb{E}|\Delta V_r| < \varepsilon$, where the total number of neurons in the family $\{G_{N-r:N}^{NN}\}_{1 \leq r \leq N}$ is

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

2199

2200 The proof is given in Section 2.E. This result is pessimistic: the error is asymptotically
2201 polynomial of degree N in ε^{-1} .

2202 **Learning with Sukhatme's representation.** Next, we consider the Sukhatme rep-
2203 resentation for generating order statistics. Let F denote the cdf of the unit exponential
2204 distribution, $F(z) = 1 - e^{-z}$ for $z > 0$, and $F^{-1}(u) = -\ln(1-u)$ for $u \in (0, 1)$. Let F_{NN}
2205 and F_{NN}^{-1} be neural network approximations of F and F^{-1} , respectively. The sampling
2206 procedure is detailed in Algorithm 4:

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

Input : Learned function \hat{G}^{NN} ; N uniform samples U_1, \dots, U_N

Output: $\hat{Z}_{1:N}, \dots, \hat{Z}_{N:N}$

1 Set $\hat{U}_{0:N} = U_{0:N} = 0$;

2207 2 **for** $r \leftarrow 1$ **to** N **do**

3 $\hat{Z}_{r:N} \leftarrow \hat{Z}_{r-1:N} + \frac{F_{NN}^{-1}(U_r)}{N-r+1};$

 // In parallel: $Z_{r:N} \leftarrow Z_{r-1:N} + \frac{F^{-1}(U_r)}{N-r+1}.$

4 **return** $\hat{U}_{r:N} = F_{NN}(\hat{Z}_{r:N})$ and $U_{r:N} = F(Z_{r:N})$, for $r \in \{1, \dots, N\}$.

2208 **Theorem 2.2.** *With the sampling strategy of Algorithm 4, there exist ReLU Neural Net-*
 2209 *works F_{NN} and F_{NN}^{-1} such that the error in approximating the uniform order statistics*
 2210 *satisfies, in expectation,*

$$\mathbb{E}|U_{r:N} - \hat{U}_{r:N}| \leq \varepsilon \left[1 + 2 \sum_{i=1}^r \frac{1}{N - \textcolor{blue}{i} + 1} \right], \quad (2.15)$$

2211 *with at most*

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

2212 *neurons involved, for some constant c .*

2213 The proof is provided in Section 2.F. For $r = N$ (the worst case), the error term admits
 2214 the asymptotics, as $N \rightarrow \infty$:

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

2215 and the complexity term satisfies the following asymptotic results, as $\varepsilon \rightarrow 0$:

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

2216 This representation is more advantageous than that of Theorem 2.1, as the network com-
 2217 plexity scales asymptotically as a polynomial of order 2 in ε^{-1} times a logarithmic term.

2218 **Discussion.** The two results are not strictly comparable: one yields an error of order
 2219 ε , while the other yields an error of order $\varepsilon \ln(N)$. However, Algorithm 4 presents a key
 2220 advantage: it exploits a structural property linking order statistics to linear combinations
 2221 of i.i.d.exponential variables. As a result, the main complexity in neural network approx-
 2222 imation is concentrated in accurately simulating unit exponential random variables from
 2223 standard uniform inputs. Moreover, it enables computational reuse: we only need to ap-
 2224 proximate the inverse cdf function F^{-1} , where $Z \sim \mathcal{E}(1)$, rather than approximating N
 2225 separate functions as required by Algorithm 3.

2226 2.3 Experiments

2227 In this section, we present a suite of synthetic experiments designed to evaluate the performance
 2228 of generative models in producing samples of order statistics. All results reported
 2229 here are obtained using synthetic data, allowing us to assess model capabilities under
 2230 controlled conditions where the ground truth distribution is known. Our experimental
 2231 framework is specifically tailored to test how well different neural network-based genera-
 2232 tive models—notably various GAN architectures—can approximate the distributional and
 2233 structural properties of order statistics.

2234 We systematically compare several GAN variants, including classic GANs, Wasserstein
 2235 GANs (WGAN), and WGANs with gradient penalty (WGAN-GP). These models are eval-
 2236 uated exclusively on synthetic datasets, which enables precise quantitative measurement of
 2237 model performance. The data used in our experiments are constructed to exhibit the char-
 2238 acteristics of order statistics, as described earlier, providing a rigorous testbed for model
 2239 assessment.

2240 Our experimental design incorporates a range of evaluation metrics that capture both
 2241 standard distributional similarity and the sortedness and dependence properties intrinsic
 2242 to order statistics. This approach allows us to identify model strengths and limitations.

2243 The following subsections detail our experimental protocols, model configurations, and
 2244 results, focusing on synthetic data benchmarks.

2245 3.A Synthetic data

2246 We begin by recalling a setup for a GAN to learn simple nonlinearities across different
 2247 synthetic datasets, and then extend this setup to the case of order statistics. This vanilla
 2248 test case illustrates some design choices for implementation, especially data normalization
 2249 during training and sampling.

2250 We provide evidence that these models can learn simple multivariate distributions,
 2251 commonly used for sanity checks in the literature. We then test whether this architecture
 2252 can learn order statistics. To do so, we quantify performance with respect to two aspects:
 2253 first, the ability of the model to reproduce the marginal distribution of the components;
 2254 second, the ability to reproduce dependence among coordinates. A key structural property
 2255 of order statistics is that they should be ordered. We verify whether the components of the
 2256 model output are sorted by computing a mean 0/1 value, and then proceed to computation
 2257 of a softer metric that quantifies the excess.

2258 GANs, variants and hyperparameters

2259 Generative Adversarial Networks, as introduced by I. J. Goodfellow et al. (2014), are
 2260 trained by playing a minimax game between a generator network \mathcal{G} and a discriminator
 2261 network \mathcal{D} (dependence on parameters is implicit). Namely, both networks optimize the

2262 common objective:

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

2263 where p_{data} is the underlying distribution of the data and $p_{\mathbf{Z}}$ is the distribution of the
 2264 latent variable, a distribution that is chosen to be easy to sample from, typically Gaussian
 2265 or Uniform. Throughout this work, we use a uniformly distributed latent variable. More
 2266 details on GANs and variants are given in Section 2.G. Regarding architecture, we consider
 2267 both networks to be simple [MLP models](#).

2268 **Hyperparameters.** The hyperparameters to define in an implementation include the
 2269 batch size, the latent dimension, the learning rate, the number of epochs, and of course the
 2270 architectures of both neural networks (generator and discriminator). Moreover, a common
 2271 practice is, at each epoch, to train the generator for `g_step` steps and the discriminator
 2272 for `d_step`. The following choices are common practice: `g_step=1` and `d_steps=5`. We
 2273 provide more details on [hyperparameters in Section 2.H](#).

2274 How accurate is a GAN at approximating order statistics?

2275 In this section, we seek to find the limitations of the representation power of GANs for
 2276 modeling order statistics. For this purpose, we choose a classic MLP architecture for the
 2277 networks, illustrate that it can fit a classical benchmark dataset properly, and then assess
 2278 visually up to which dimension [the model can](#) generate proper order statistics. We also
 2279 use this toy-study to discuss the special scaling used.

2280 **Fitting a benchmark dataset.** A simple fitting test can be done on the banana-shaped
 2281 distribution of (Haario et al. 1999), defined as follow: consider $(x_1, x_2) \sim \mathcal{N}(0, I)$ and:

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

2282 where $b \in \mathbb{R}$ controls the curvature of the distribution. An illustration of the GAN
 2283 performance is given in Figure 2.1.

2284 Fitting naive model on order statistics

2285 We now take the model obtained from the benchmarking on the banana-shaped distribution
 2286 to fit order statistics. We sample *large* order-statistics:

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

2287 **Normalization scheme.** Per-coordinate minmax normalization, while common in GANs,
 2288 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.

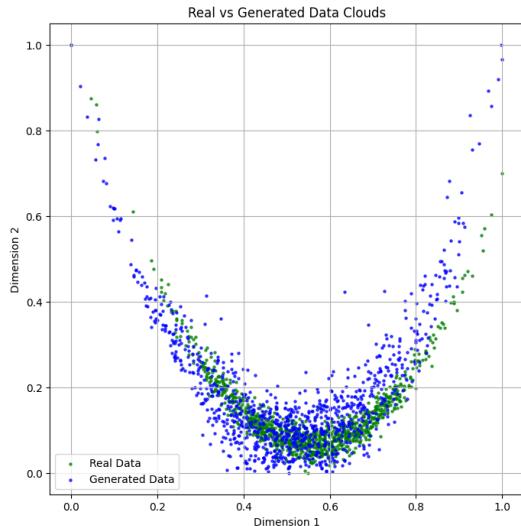


Figure 2.1: Real and generated data after min-max renormalization. n_samples=1,000.

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., minmax 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 $s > 0$,

$$\tilde{\mathbf{x}} = s(\mathbf{x} - \bar{\mathbf{x}}), \quad \bar{\mathbf{x}} \stackrel{\text{def}}{=} \frac{1}{Nd} \sum_{n=1}^N \sum_{j=1}^d x_j^{(n)}. \quad (2.22)$$

This meanscale normalization stabilizes optimization *and* exactly preserves the ordering constraint: for any sample \mathbf{x} and indices $i < j$, $\text{sign}(\tilde{x}_i - \tilde{x}_j) = \text{sign}(x_i - x_j)$. 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 minmax model yields visually plausible marginals, yet many generated points cross the diagonal, signaling broken sortedness. With meanscale 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.

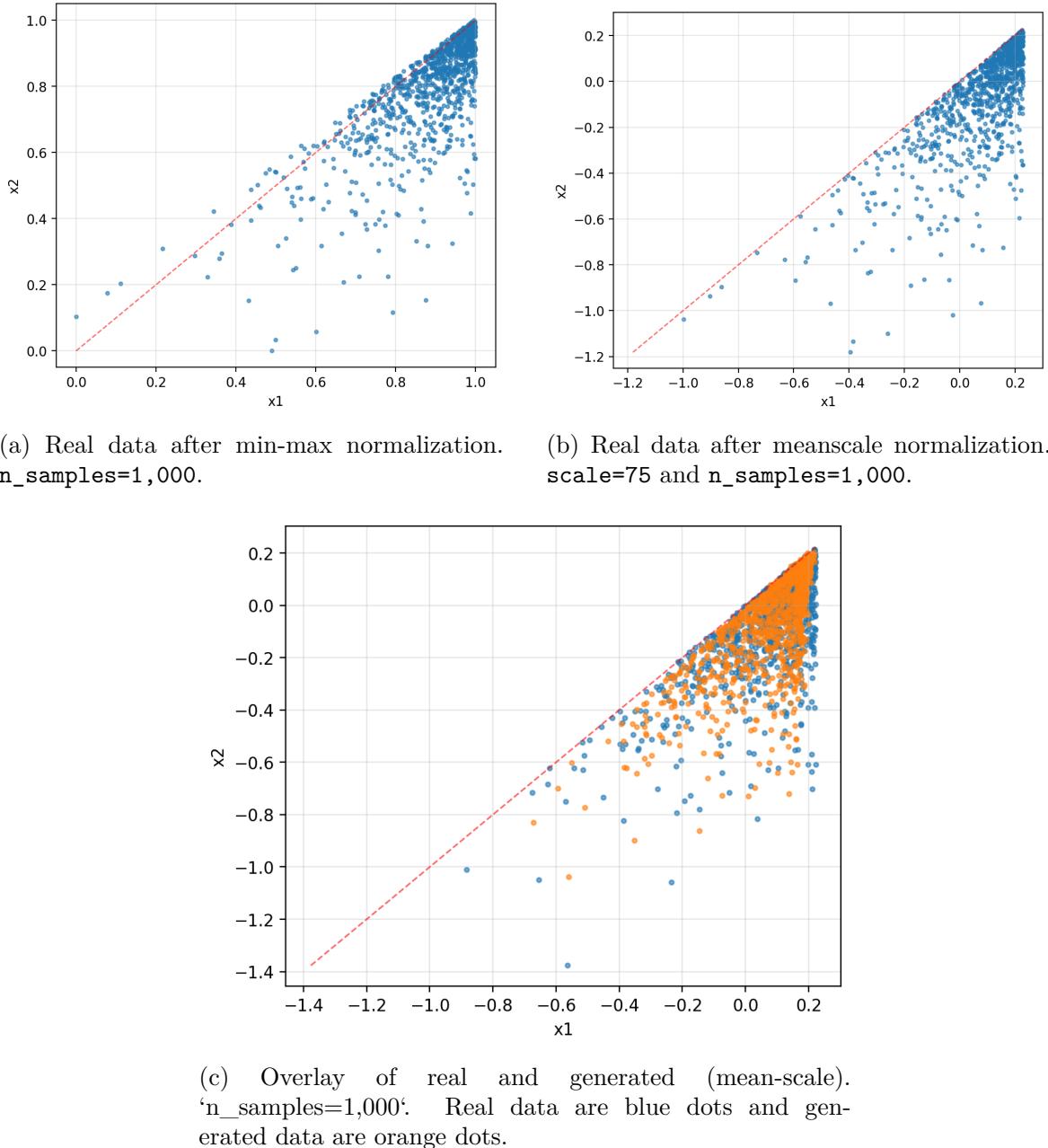


Figure 2.2: Comparison of real and generated 2D order statistics under different normalization schemes, with an additional overlay view. The red dashed lines indicate $x = y$; the meanscale normalization preserves the sorted structure (no line crossing), while minmax normalization does not.

2307 3.B Benchmarking on synthetic data

2308 In this section, we detail our benchmarking study of GAN architectures to model order
 2309 statistics. We consider three baseline variants of GANs: vanilla GAN (I. J. Goodfellow
 2310 et al. 2014), Wasserstein-GAN (WGAN) (Arjovsky et al. 2017), and WGAN with gradient
 2311 penalty (WGAN-GP) (Wei et al. 2018). We evaluate each of the three models over different
 2312 dimension settings. Once the best model is selected, we enhance it with an auxiliary
 2313 regularization term that proxies the quality of the order-statistics property of the produced
 2314 data points. We then reevaluate the enhanced model against its corresponding base version.

2315 **Evaluating the best model out of the three base models**

2316 **Scope of the study.** We explore the performance of each model in different settings.
 2317 We proceed to replicate the distribution of (2.21), with $N = 1000$ and various values of k .
 2318 Namely, we first explore dimensions $d = 2, 3, 5, 10, 20$ and 50 for assessing the best model
 2319 out of the three base models. We choose to use a data set of size $n_{\text{data}} = 32,000$, fixed
 2320 across dimensions. For each variant of implementation, we perform $R = 10$ replications
 2321 and compute, for each replicated model, six metrics on generated datasets.

2322 **Metrics.** Among these six metrics, two are based on a Wasserstein criterion: the Wasser-
 2323 stein 1D (abbr. W1D) and Sliced-Wasserstein Distance (SWD). Two are general de-
 2324 pendence metrics: the Spearman’s ρ coefficient difference (Spearman) and the Absolute
 2325 Kendall Error (AKE). Finally, two are specifically designed to assess the order-statistics
 2326 property of the generated samples, the Sortedness and Softsortedness metric. Precise def-
 2327 initions and background on the metrics are detailed in Section 2.I.

2328 **Sweeping with Optuna.** For exploring the best hyperparameters, we use Optuna (Ak-
 2329 iba et al. 2019), a hyperparameter optimization framework. We use the default TPE
 2330 sampler, which models the objective function using a tree-structured Parzen estimator.
 2331 We define a grid of hyperparameters over which we perform optimization. Details on hy-
 2332 perparameters are given in Section 2.H. For each pair of model and dimension, we perform
 2333 200 trials. The target metric for TPE sampling is the W1D criteria. We provide both the
 2334 hyperparameter grid and the hyperparameters selected by the sweep in Section 2.H.

2335 **Replications.** Once the sweep done, we perform $R = 10$ replications for each pair of
 2336 model and dimension. We compute each of the six metrics for each of the replications and
 2337 average over model and dimension. Results are provided in Table 2.2. Best values are in
 2338 bold.

2339 **Results analysis.** The results of Table 2.2 provides us the following insights on the
 2340 performance of models: The WGAN-GP underperforms by a wide margin compared to
 2341 the two other models. On Wasserstein-based metrics, marginal (W1D) and multivariate
 2342 (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 models 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 $d \in \{20, 50\}$. Furthermore, 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 interpret 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	dim	GAN	WGAN	WGAN OSP
2	0.341	0.328	0.486	2	0.336	0.332	0.475
5	0.22	0.424	0.251	5	0.204	0.396	0.251
10	0.513	0.363	0.488	10	0.473	0.356	0.478
20	1.1	0.55	0.839	20	1.03	0.526	0.815
50	2.33	1.13	1.1	50	2.16	1.1	1.02

(a) W1D				(b) SWD			
dim	GAN	WGAN	WGAN OSP	dim	GAN	WGAN	WGAN OSP
2	1.53	1.18	1.17	2	94.8%	88.7%	98.3%
5	0.315	0.506	0.896	5	89.2%	76.6%	80.3%
10	0.452	0.989	0.93	10	56.3%	56.2%	99.4%
20	0.391	0.935	1.33	20	4.0%	60.0%	93.5%
50	0.713	2.02	1.65	50	0.0%	88.3%	74.9%

(c) Spearman				(d) Sortedness			
dim	GAN	WGAN	WGAN OSP	dim	GAN	WGAN	WGAN OSP
2	1.79	1.84	1.92	2	0.128	0.364	0.019
5	2.64	2.71	2.67	5	0.033	0.19	0.193
10	3.59	3.56	3.66	10	0.263	0.28	0.000692
20	4.95	5.05	4.93	20	1.04	0.105	0.0102
50	7.94	8.08	7.95	50	3.21	0.0807	0.0832

(e) AKE				(f) Softsortedness			
dim	GAN	WGAN	WGAN OSP	dim	GAN	WGAN	WGAN OSP
2	1.79	1.84	1.92	2	0.128	0.364	0.019
5	2.64	2.71	2.67	5	0.033	0.19	0.193
10	3.59	3.56	3.66	10	0.263	0.28	0.000692
20	4.95	5.05	4.93	20	1.04	0.105	0.0102
50	7.94	8.08	7.95	50	3.21	0.0807	0.0832

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 auxiliary regularization term, favoring order statistics samples.

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

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

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 training. 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. Results 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 consistently 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 topdown recursion and Sukhatme's exponential spacings) can be leveraged to design neural samplers with explicit capacity/error tradeoffs. Concentrating approximation into one scalar inversecdf block (with Sukhatme's representation) yields a parameter growth of order $\mathcal{O}(\varepsilon^{-2} \log(\varepsilon^{-1}))$ and a coordinatewise 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 meanscale 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 orderviolation penalty further tightens structureaware 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

2395 impact factor and paired with returns. This calls for learning generative IOS models
2396 with the same geometrypreserving principles. Second, a systematic, controlled empirical
2397 study that instantiates architectures mirroring the Schucany and Sukhatme constructions
2398 would enable direct validation of the predicted scalings (e.g., $\varepsilon \log N$ error growth and
2399 $\varepsilon^{-2} \log(\varepsilon^{-1})$ parameter counts). While feasible, such sweeps are tedious and we leave
2400 them to future work. Finally, applying these ideas to real impactinvesting datasets will
2401 require robust preprocessing, careful evaluation protocols, and domainspecific stress tests
2402 in which IOS simulators can meaningfully inform portfolio construction.

2403 Taken together, our analysis and experiments show that a small amount of struc-
2404 turecodified by probabilistic representations and simple normalizationgoes a long way for
2405 learning on the ordered manifold. We hope this bridges classical orderstatistics theory and
2406 modern deep generative modeling, and serves as a foundation for IOSaware simulators in
2407 sustainable finance.

2408 Appendix

2409 2.A Additional Figures

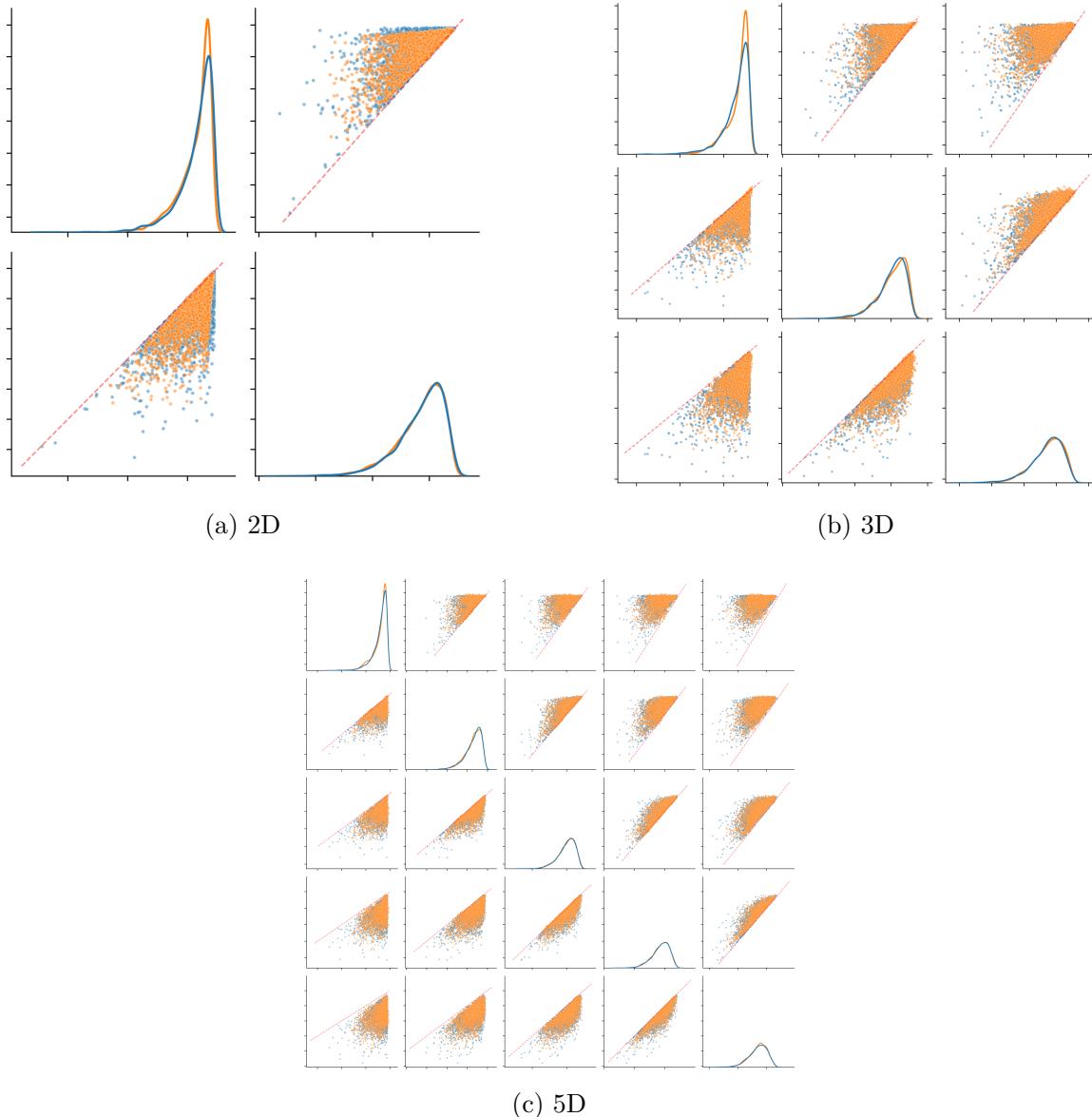


Figure 2.3: Generated vs. real means for order statistics in various dimensions: (a) 2D, (b) 3D, (c) 5D. Every dimension is scattered against one another. Real data points are in blue and generated data points are in orange.

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:

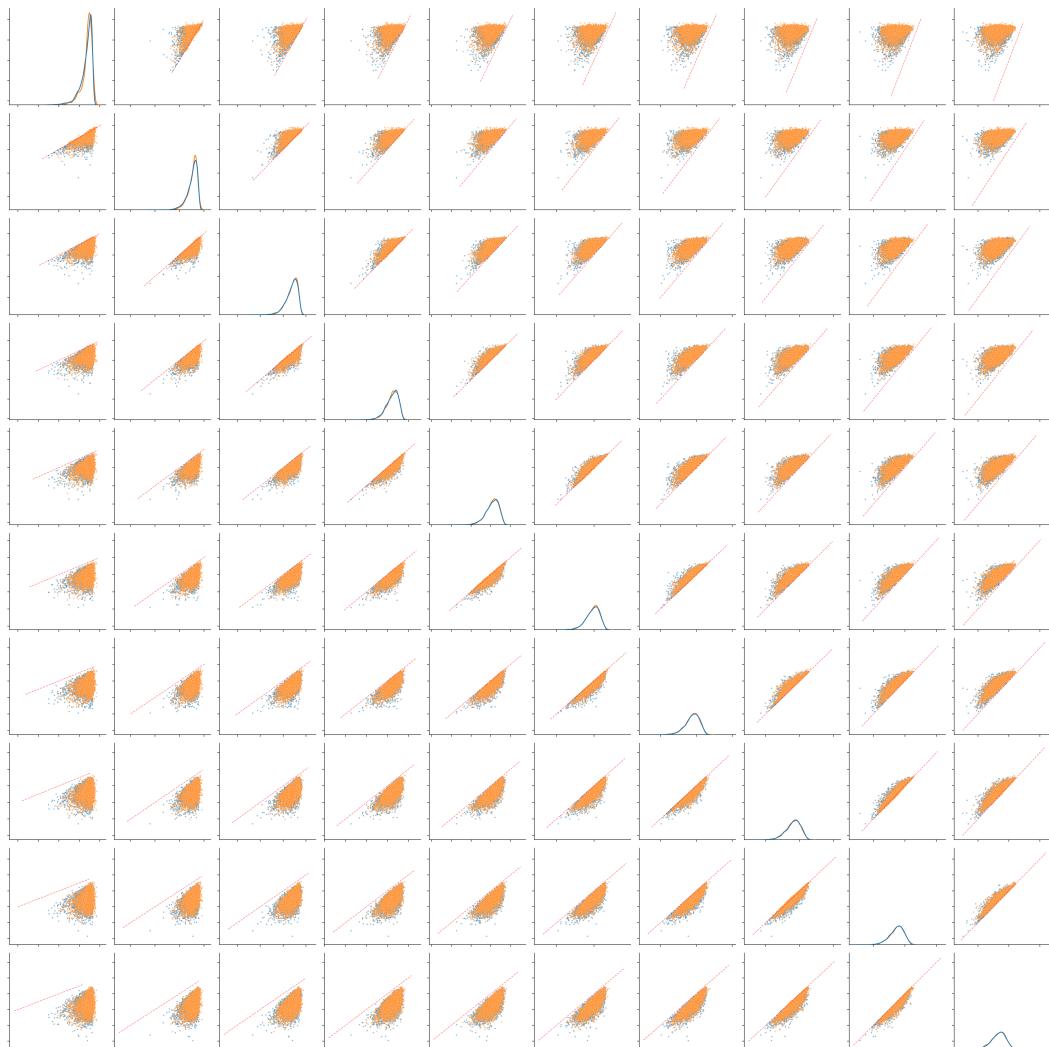


Figure 2.4: Generated vs. real means for order statistics in dimension 10. Every dimension is scattered against one another. Real data points are in blue and generated data points are in orange.

$$X_{k:N} := \max_{1 \leq i_1 < \dots < i_k \leq N} \min \{X_{i_1}, \dots, X_{i_k}\}. \quad (2.24)$$

2415 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:

$$\max\{a, b\} = a + (b - a)_+, \quad (2.25)$$

$$\min\{a, b\} = -\max\{-a, -b\}. \quad (2.26)$$

2417 Moreover, it is possible to compute min and max over more than two variables by 2418 computing recursively:

$$\max\{a_1, \dots, a_N\} = \max \{\max\{a_1, \dots, a_{N-1}\}, a_N\}. \quad (2.27)$$

2419 This formula has the benefit, conversely to further representations in this section, of 2420 being fully deterministic. However, this comes at a cost: in practice, it is not reasonably 2421 feasible to perform computation of (2.24), as it involves, for the full sorting of vector 2422 $(X_{1:N}, \dots, X_{N:N})$, 2^N operations.

2423 **Order statistics as a Markov chain (Schucany form).** Order statistics admit a 2424 Markovian structure (H. David and H. Nagaraja 2004, Chapter 1, Section 5). For the 2425 uniform case, the chain is especially simple in the *topdown* direction. Conditionally on 2426 $U_{m+1:N} = x$, the distribution of $U_{m:N}$ on $[0, x]$ is the maximum of m i.i.d. uniforms on 2427 $[0, x]$, hence

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

2428 Equivalently, if $(U_m)_{m=1}^N$ are i.i.d. $\text{Unif}(0, 1)$ independent of $U_{m+1:N}$, then

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

2429 This is precisely Schucany's recursive sampler for uniform order statistics and matches 2430 Equation (2.7) (identify $m = N - r$ and $U_m \equiv U_r$). Defining the bivariate update

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

2431 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 2432 formulation used for our neural approximations in Algorithm 3.

2433 **Using standard exponential distributions.** Another method for modeling order statistics stems from Sukhatme's representation (H. David and H. Nagaraja 2004, p.17). Let 2434 $Z_{1:N} \leq \dots \leq Z_{N:N}$ be the order statistics from the standard exponential distribution. 2435 Then the joint distribution of the $Z_{i:N}$ is :

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

²⁴³⁷ which may be rewritten as (Sukhatme 1937):

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

²⁴³⁸ with convention $z_{0:N} = 0$. Making the transformation

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

²⁴³⁹ and noting that the range of each y_i is $(0, \infty)$, we see that the Y_i are statistically
²⁴⁴⁰ independent variables, each with pdf (2.8). The relation (2.33) allow $Z_{r:N}$ to be expressed
²⁴⁴¹ as:

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

²⁴⁴² Following the probability integral transformation, we can sample from the standard expo-
²⁴⁴³ nential distribution by sampling from an uniform distribution and applying the inverse cdf
²⁴⁴⁴ $F^{-1} : u \mapsto -\ln(1-u)$:

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

²⁴⁴⁵ 2.C Concomitants of order statistics from a single distribution

²⁴⁴⁶ Definition

²⁴⁴⁷ Let $(X_i, Y_i)_{i=1,\dots,n}$ be a random sample from a bivariate distribution with cdf $F(x, y)$. Let
²⁴⁴⁸ us denote

$$X_{r:n} \quad (2.36)$$

²⁴⁴⁹ the r -th order statistic (where data have ordered increasingly), for any $r = 1, \dots, n$:

$$X_{1:n} < \dots < X_{r:n} < \dots < X_{n:n}. \quad (2.37)$$

²⁴⁵⁰ When the marginal cdf of X is continuous, this ordering exists and is almost surely
²⁴⁵¹ unique.

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

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

²⁴⁵² 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

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

When the $(\varepsilon_i)_{1 \leq i \leq n}$ are i.i.d. then the $(\varepsilon_{[r:n]})_{1 \leq r \leq n}$ 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 $X = (X_i)_{1 \leq i \leq n}$ are *associated* if:

$$\text{Cov}(f(X), g(X)) \geq 0 \quad (2.39)$$

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 $(Y_{[r:n]})_{1 \leq r \leq n}$ 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 $(Y_{[r:n]})_{1 \leq r \leq n}$ 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*

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

2480 where $U_1, \dots, U_n, V_1, \dots, V_n$ are independent random variables uniformly distributed on
2481 $[0, 1]$. The function g is given explicitly by the formula

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

2482 Note that in the above theorem, C is uniquely defined since the marginal cdf are
2483 continuous.

2484 **Representation theorem without the continuity assumption**

2485 We now state a variant avoiding the continuity assumption of the cdf of X and Y . We
2486 follow the arguments of (Barrera et al. 2025) to well define the quantity under study.

2487 We denote by $\mathcal{B}_{\mathbb{R}}$ the Borel sigma algebra on \mathbb{R} . We fix a conditional distribution
2488 function $\mu : \mathcal{B}_{\mathbb{R}} \times \mathcal{B}_{\mathbb{R}} \rightarrow [0, 1]$ of Y given X (Kallenberg 2002, theorem 5.3, p.84), and we
2489 assume that the function $\mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$ defined by $(x, y) \mapsto \mu(x, (-\infty, y])$ is $(\mathcal{B}_{\mathbb{R}} \otimes \mathcal{B}_{\mathbb{R}})/\mathcal{B}_{\mathbb{R}}$
2490 (i.e. Borel)-measurable. With these conventions, we will use implicitly the corresponding
2491 version

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

2492 The conditional quantile of Y given $X = x$ is defined by

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

2493 **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.*

2494 *Proof.* Given $t \in \mathbb{R}$, since

$$\{q_{Y|X}(x, v) \leq t\} = \{\mu(x, (-\infty, t]) \geq v\}, \quad (2.44)$$

2495 we get the result. □

2496 **Theorem 2.5.** *Denote by F_X and F_Y the respective cdf of X and Y (they can be discontinuous). Then*

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

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

2498 where $U_1, \dots, U_n, V_1, \dots, V_n$ are independent random variables uniformly distributed on
2499 $[0, 1]$.

2500 *Proof.* The Rosenblatt transformation writes

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

2501 Because the quantile F_X^{-1} is increasing,

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

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

2502 Since the V_i 's are i.i.d. we have

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

2503 In addition, since they are independent from the U_i 's, we get

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

2504

□

2505 **The case of independent (X_i, Y_i) but with different cdfs $F_i(x, y)$**

2506 In that case, the Rosenblatt transformation write

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

2507 **The intermediate case of independent (X_i, Y_i) but with same marginal X_i**

2508 In that case, the Rosenblatt transformation writes

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

2509 **2.D Some elements on Functional approximation with Neural Network**

2511 Section 2.B provides different computational methodologies for obtaining order statistics.
2512 A natural question is how well a Neural Network-based method can approximate the
2513 obtained functions. For quantifying this, we rely on results from the theory of function
2514 approximation with Neural Networks. (Yarotsky 2017) provides several results on how well
2515 neural networks are able to approximate smooth functions.

2516 **Lemma 2.2** (Yarotsky (2017), Proposition 3). *Given $M > 0$ and $\varepsilon \in (0, 1)$, there exists
2517 a ReLU network η with two input units that implements a function $\tilde{x} : \mathbb{R}^2 \rightarrow \mathbb{R}$ so that:*

- 2518 • for any input x, y , if $|x| < M$ and $|y| < M$, then $|\tilde{x}(x, y) - xy| \leq \varepsilon$
- 2519 • if $x = 0$ or $y = 0$ then $\tilde{x}(x, y) = 0$
- 2520 • the depth and the number of weights and computation units in η is no greater than
2521 $c_1 \ln(1/\varepsilon) + c_2$ 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:

$$\|f\|_{\mathcal{W}^{n,\infty}([0,1]^d)} = \max_{\mathbf{n}:|\mathbf{n}| \leq n} \text{ess}_{\mathbf{x} \in [0,1]^d} |D^{\mathbf{n}} f(\mathbf{x})| \quad (2.54)$$

and its associated unit-norm class is given by:

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

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 $\Delta V_r = \hat{U}_{N-r:N} - U_{N-r:N}$.

$$\Delta V_0 = 0, \quad (2.56)$$

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

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

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

On the one side:

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

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

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

On the other:

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

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

2537 Thus for $1 \leq r \leq N$:

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

Lemma 2.3.

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

2538 with convention $\varepsilon_0 = 0$.

2539 *Proof.* The proof goes by induction:

2540 • $r = 0$: (2.65) writes:

$$|\Delta V_1| \leq |\Delta V_0| U_1^{1/N} + \varepsilon_1 = \varepsilon_1, \quad (2.67)$$

2541 as $\Delta V_0 = 0$, which is the announced result.

2542 • Suppose that (2.66) holds true for $0 \leq r \leq N-1$. Then:

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

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

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

2543 The result is thus proved. □

2544

2545 A direct computation yields $\mathbb{E}[U^{1/(N-j)}] = \frac{N-j}{N-j+1}$ and thus

$$\mathbb{E} |\Delta V_r| \leq \sum_{i=0}^r \varepsilon_i \prod_{j=i+1}^r \frac{N-j}{N-j+1} \quad (2.71)$$

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

2546 Let's investigate ε_i . We are looking to the best approximation of the function $G_{N-r:N}(u_1, u_2) =$
2547 $u_1 \cdot u_2^{1/(N-r)}$. On the one side, the function

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

2548 is $\zeta_r := 1/(N-r)$ Holder continuous, (the space of Holder continuous functions writes
2549 $C^{0,\zeta_r}([0, 1]))$, with Holder constant 1.

2550 Define the ReLU function as $\sigma^R(x) := \max(0, x)$. Then denote the triangular function
 2551 by:

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

2552 Note that the triangular function can be approximated by a **ReLU** Neural Network with
 2553 three neurons, see (Michaël Allouche, Girard, et al. 2025). We also have the following
 2554 result:

2555 **Lemma 2.4** (Michaël Allouche, Girard, et al. (2025), Lemma 8). *Let $\widehat{\sigma}^R$ be a triangular
 2556 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 $h = \tau/M$
 2557 and $u_j = jh$ for $j \in \{0, \dots, M\}$. Then,*

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

2558 where H_f is the Hölder constant associated with f . This construction involves $3(M+1)$
 2559 neurons.

2560 This result thus applies to functions ϕ_r , with $\tau = 1$ and we denote the approximation
 2561 of ϕ_r by $\widehat{\phi}_r$. To obtain the minimal number of neuron sufficient for a given precision, let's
 2562 write the error ε :

$$\varepsilon \leq (1/M)^{\zeta_r} \Rightarrow M \geq 1/\varepsilon^{N-r} \Rightarrow M = \lceil 1/\varepsilon^{N-r} \rceil. \quad (2.74)$$

2563 Thus a neural network with

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

2564 neurons is able to approximate the function ϕ_r with precision ε and a neural network with
 2565 $\lceil (2/\varepsilon)^{N-r} \rceil + 1$ neurons is able to approximate the function ϕ_r with precision $\varepsilon/2$. Now,
 2566 for the multiplicative term, we invoke Lemma 2.2, which writes as follows in this context:

2567 A neural network η_r with $c_1 \ln(8/\varepsilon) + c_2$ weights with an absolute constant c_1 and a
 2568 constant $c_2 = c_2(M)$, with $M = 1$ here, is able to learn $\tilde{x}_r(x, v) = x \cdot v$ with precision
 2569 $\varepsilon/8$.

2570 Now consider

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

2571 Then:

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

$$\leq \left| \tilde{x}_r(u_1, \phi_r(u_2)) - \tilde{x}_r(u_1, u_2^{1/(N-r)}) \right| \quad (2.78)$$

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

2572 The second term is $\leq \varepsilon/8$ by construction of \tilde{x}_r . Let's look at the first:

$$\left| \tilde{x}_r(u_1, \phi_r(u_2)) - \tilde{x}_r(u_1, u_2^{1/(N-r)}) \right| \leq \left| \tilde{x}_r(u_1, \phi_r(u_2)) - u_1 \cdot \phi_r(u_2) \right| \quad (2.80)$$

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

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

$$\leq 3\varepsilon/4, \quad (2.83)$$

2573 where (2.80), (2.82) $\leq \varepsilon/8$ by construction of \tilde{x} , (2.81) $\leq \varepsilon/2$ by construction of ϕ_r
2574 and because $0 \leq x \leq 1$. Together with (2.78):

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

2575 **Lemma 2.5.** Let $\varepsilon > 0$, $N \in \mathbb{N}^*$ and $r \in [N]$. A feedforward ReLU Neural Network
2576 $\hat{G}_{N-r:N}^{NN}$ with:

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

2577 neurons is able to approximate $G_{N-r:N}$ with precision ε .

2578 Back to (2.72): choosing for $i \in [r]$, $\varepsilon > 0$, $\varepsilon_i = \frac{(N-r)}{r(N-i)}\varepsilon$, we get $\mathbb{E}|\Delta V_r| \leq \varepsilon$ and thus
2579 we obtain Theorem 2.1 by summing over i .

2. F Proof of Theorem 2.2

2581 Let's investigate the quality of approximation of the scheme in Algorithm 4. For a starter,
2582 let's look into the quality of approximation of the function $F^{-1} : u \mapsto -\ln(1-u)$ with a
2583 neural network. For this sake, consider $\varepsilon \in [0, 1]$ and introduce the function :

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

2584 The function $\varepsilon \cdot \tilde{F}_\varepsilon^{-1}$ is in $F_{1,1}$. Indeed, $\forall u \in [0, 1]$:

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

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

2585 Therefore, the assumptions of Theorem 2.6 are met. We apply it to $\varepsilon \tilde{F}_\varepsilon^{-1}$ with precision
 2586 ε^2 , which writes:

2587 There exists a ReLU network η_{NN} that:

- 2588 • is capable of approximating $\varepsilon \tilde{F}_\varepsilon^{-1}$ with error ε^2 , i.e.:

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

- 2589 • has depth at most $c(\ln(1/\varepsilon^2 + 1))$ and at most

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

2590 weights and computation units with some constant $c = c(1, 1)$.

2591 Thus, multiplying the obtained neural network by $1/\varepsilon$, we get that:

$$\max_{u \in [0, 1-\varepsilon]} |\tilde{F}_\varepsilon^{-1}(u) - 1/\varepsilon \cdot \eta_{NN}(u)| \leq \varepsilon \quad (2.91)$$

2592 And thus the following result holds:

2593 There exists a ReLU network architecture η_{NN} (we reuse the notation η_{NN}) such that:

- 2594 • is capable of approximating $\tilde{F}_\varepsilon^{-1}$ with error ε ,
 2595 • has depth at most $c(\ln(1/\varepsilon^2) + 1)$ and at most

$$c(\varepsilon^2)^{-1} (\ln(1/\varepsilon^2) + 1) = c\varepsilon^{-2}(2\ln(1/\varepsilon) + 1) \quad (2.92)$$

2596 weights and computation units with some constant $c = c(1, 1)$.

2597 Note that the Neural Network η_{NN} is defined on $[0, 1 - \varepsilon]$. We artificially set its value
 2598 on the interval $[1 - \varepsilon, 1]$ to $\eta_{NN}(u) = F^{-1}(1 - \varepsilon) = -\ln(\varepsilon)$. Now consider a standard
 2599 uniform variable U . We investigate the mean error between $Z = F^{-1}(U)$ (which is $\mathcal{E}(1)$)
 2600 and its approximation $\hat{Z} = \eta_{NN}(U)$. We have that:

$$\mathbb{E}_U |Z - \hat{Z}| = \int_0^1 |F_Z^{-1}(u) - \eta_{NN}(u)| du \quad (2.93)$$

$$= \int_0^{1-\varepsilon} |F_Z^{-1}(u) - \eta_{NN}(u)| du + \int_{1-\varepsilon}^1 |F_Z^{-1}(u) - \eta_{NN}(u)| du. \quad (2.94)$$

2601 First the LHS of (2.94):

$$\int_0^{1-\varepsilon} |F_Z^{-1}(u) - \eta_{NN}(u)| du \leq \int_0^{1-\varepsilon} \|F_Z^{-1} - \eta_{NN}\|_\infty du \leq \varepsilon, \quad (2.95)$$

2602 by construction of η_{NN} and Theorem 2.6 and second the RHS of (2.94):

$$\int_{1-\varepsilon}^1 |F_Z^{-1}(u) - \eta_{NN}(u)| du = \int_{1-\varepsilon}^1 |- \ln(1-u) + \ln(\varepsilon)| du \quad (2.96)$$

$$= \int_{1-\varepsilon}^1 [\ln(\varepsilon) - \ln(1-u)] du \quad (2.97)$$

$$= \varepsilon \ln(\varepsilon) - \int_{1-\varepsilon}^1 \ln(1-u) du = \varepsilon, \text{ after integration.} \quad (2.98)$$

2603 Together,

$$\mathbb{E}_U |Z - \hat{Z}| \leq 2\varepsilon. \quad (2.99)$$

2604 Therefore,

$$\mathbb{E} |Z_{r:N} - \hat{Z}_{r:N}| = \mathbb{E} \left| \sum_{i=1}^r \frac{Y_i}{N-r+1} - \sum_{i=1}^r \frac{\hat{Y}_i}{N-r+1} \right| \quad (2.100)$$

$$\leq \sum_{i=1}^r \frac{\mathbb{E}|Y_i - \hat{Y}_i|}{N-r+1} \leq 2\varepsilon \sum_{i=1}^r 1/(N-r+1). \quad (2.101)$$

2605 We now consider approximation of the function F . As for the inverse \tilde{F}_ε , we will construct a suitable function to which we can apply Yarotsky's result. For F , the construction 2606 is a bit more subtle: Let $\varepsilon \in [0, 1]$. We consider its restriction to F to $[0, \ln(1/\varepsilon)]$, and 2607 therefore define the function \tilde{F}_ε as follows:

$$\text{for } x \geq 0 : \tilde{F}_\varepsilon(x) = \frac{F(\ln(1/\varepsilon)x)}{\ln(1/\varepsilon)}. \quad (2.102)$$

2609 The function \tilde{F}_ε is in $F_{1,1}$. Indeed $\tilde{F}_\varepsilon(x) \in [0, 1]$ and $\tilde{F}_\varepsilon(x) = e^{-\ln(1/\varepsilon)x} \in [0, 1]$. 2610 Therefore, Theorem 2.6 applies to \tilde{F}_ε . We apply it to F_ε with error $\varepsilon/\ln(1/\varepsilon)$, which 2611 writes out:

2612 There exists a ReLU Neural Network η_{NN} (we reuse the notation η_{NN}) such that:

- 2613 • is capable of approximating \tilde{F}_ε with error $\varepsilon/\ln(1/\varepsilon)$,
- 2614 • has depth at most $c \ln\left(\frac{\ln(1/\varepsilon)}{\varepsilon} + 1\right)$ and at most

$$c(\varepsilon/\ln(\varepsilon))^{-1} \left(\ln\left(\frac{\ln(1/\varepsilon)}{\varepsilon}\right) + 1 \right) = c\varepsilon^{-1} \ln(1/\varepsilon) (\ln(1/\varepsilon) + \ln(\ln(1/\varepsilon)) + 1) \quad (2.103)$$

2615 weights and computational units, for some constant $c = c(1, 1)$.

2616 Finally, doing the converse operation of rescaling on the neural network η_{NN} , that is 2617 considering (we reuse the notation η_{NN}):

$$\eta_{NN}(x) = \ln(1/\varepsilon) \eta_{NN}^{\text{old}} \left(\frac{x}{\ln(1/\varepsilon)} \right) \quad (2.104)$$

2618, we get that, $\forall x > 0$:

$$|F(x) - \ln(1/\varepsilon)\eta_{NN}^{\text{old}}(x/\ln(1/\varepsilon))| = |F(x) - \eta_{NN}(x)| \leq \varepsilon. \quad (2.105)$$

2619 Now, noting $U_{r:N} = F(Z_{r:N})$ and $\hat{U}_{r:N} = \eta_{NN}(\hat{Z}_{r:N})$:

$$|U_{r:N} - \hat{U}_{r:N}| = |F(Z_{r:N}) - \eta_{NN}(\hat{Z}_{r:N})| \leq |\eta_{NN}(\hat{Z}_{r:N}) - F(\hat{Z}_{r:N})| + |F(\hat{Z}_{r:N}) - F(Z_{r:N})| \quad (2.106)$$

$$\leq \varepsilon + |\hat{Z}_{r:N} - Z_{r:N}| \quad (2.107)$$

2620 where the LHS holds in virtue of (2.105) and the RHS because F is a 1-Lipschitz function.

2621 Taking expectations on both sides:

$$\mathbb{E}|U_{r:N} - \hat{U}_{r:N}| \leq \varepsilon + \mathbb{E}|Z_{r:N} - \hat{Z}_{r:N}| \leq \varepsilon + 2\varepsilon \sum_{i=1}^r \frac{1}{N-r+1} \quad (2.108)$$

$$= \varepsilon \left[1 + 2 \sum_{i=1}^r \frac{1}{N-r+1} \right]. \quad (2.109)$$

2622 The total number of neurons involved for the computation of $\hat{U}_{r:N}$ with this precision
2623 is :

$$c\varepsilon^{-2}(2\ln(1/\varepsilon) + 1) + c\varepsilon^{-1}\ln(1/\varepsilon)(\ln(1/\varepsilon) + \ln(\ln(1/\varepsilon)) + 1), \quad (2.110)$$

2624 which achieves the proof of Theorem 2.2.

2625 2.G Additional experimental details

2626 More background on GANs

2627 This appendix summarizes the adversarial training objectives used in our experiments and
2628 highlights practical details for stability.

2629 **GAN objective.** Given a data distribution p_{data} and a latent prior $p_{\mathbf{Z}}$, the vanilla GAN
2630 objective (see Equation (1.12)) defines a two-player minimax game between a discriminator
2631 \mathcal{D} and a generator \mathcal{G} . In practice, the non-saturating generator loss is often used to obtain
2632 stronger gradients early in training:

$$\min_{\mathcal{G}} \mathbb{E}_{\mathbf{Z} \sim p_{\mathbf{Z}}} [-\log \mathcal{D}(\mathcal{G}(\mathbf{Z}))]. \quad (2.111)$$

2633 At equilibrium with idealized capacity, the generator recovers p_{data} and the discriminator
2634 outputs 1/2 (I. J. Goodfellow et al. 2014). Training instabilities and mode collapse are
2635 common; Section Section 2.H records our simple, robust choices.

2636 **Wasserstein GAN (WGAN).** The WGAN replaces the JensenShannon divergence
2637 implicit in the vanilla objective with the Wasserstein-1 distance. Using the KantorovichRu-
2638 binstein duality, the critic \mathcal{D} is constrained to be 1-Lipschitz and optimized via

$$\max_{\|\mathcal{D}\|_{\text{Lip}} \leq 1} \mathbb{E}_{\mathbf{X} \sim p_{\text{data}}} [\mathcal{D}(\mathbf{X})] - \mathbb{E}_{\mathbf{Z} \sim p_{\mathbf{Z}}} [\mathcal{D}(\mathcal{G}(\mathbf{Z}))], \quad \min_{\mathcal{G}} -\mathbb{E}_{\mathbf{Z} \sim p_{\mathbf{Z}}} [\mathcal{D}(\mathcal{G}(\mathbf{Z}))]. \quad (2.112)$$

2639 The resulting gradients are more informative when supports are disjoint, improving conver-
2640 gence (Arjovsky et al. 2017). The original implementation enforced the Lipschitz constraint
2641 by weight clipping, which can harm capacity and lead to optimization issues.

2642 **WGAN with Gradient Penalty (WGAN-GP).** To avoid weight clipping, Wei et al.
2643 (2018) propose penalizing the deviation of the gradient norm from 1 along lines interpolat-
2644 ing between real and generated samples. Writing $\tilde{\mathbf{X}} = \mathcal{G}(\mathbf{Z})$, $\hat{\mathbf{X}} = \epsilon \mathbf{X} + (1 - \epsilon) \tilde{\mathbf{X}}$ with
2645 $\epsilon \sim \mathcal{U}[0, 1]$, the critic maximizes

$$\mathbb{E}_{\mathbf{X}} [\mathcal{D}(\mathbf{X})] - \mathbb{E}_{\mathbf{Z}} [\mathcal{D}(\mathcal{G}(\mathbf{Z}))] - \lambda \mathbb{E}_{\hat{\mathbf{X}}} \left(\|\nabla_{\hat{\mathbf{X}}} \mathcal{D}(\hat{\mathbf{X}})\|_2 - 1 \right)^2. \quad (2.113)$$

2646 This penalty enforces the 1-Lipschitz property more effectively and stabilizes training across
2647 architectures and datasets.

2648 2.H Hyperparameters

2649 - Update ratio: several critic steps per generator step (e.g., $n_D=5$, $n_G=1$). - Optimizers:
2650 Adam with small learning rate and $\beta_1 \in [0, 0.5]$ works well for adversarial training. - Reg-
2651 ularization: gradient penalty (WGAN-GP) improves stability; spectral normalization is an
2652 alternative 1-Lipschitz control. - Activations: LeakyReLU for critic; generator activations
2653 as in Section Section 2.H. - Evaluation: rely on distributional metrics (e.g., W1D, SWD)
2654 and structure-aware scores (e.g., sortedness) discussed in Section Section 2.I.

2655 See Appendix Section 2.G for a concise primer on GANs, WGAN, and WGAN-GP,
2656 including objectives, constraints, and training tips.

2657 2.I Metrics

2658 We evaluate generative fidelity and structure with six metrics. Throughout, let real samples
2659 be $\{\mathbf{x}^{(n)}\}_{n=1}^N \subset \mathbb{R}^d$ and generated samples $\{\mathbf{y}^{(n)}\}_{n=1}^N \subset \mathbb{R}^d$. Unless stated otherwise, we
2660 adopt a *descending* order convention when focusing on the largest order statistics.

2661 **Wasserstein 1D (W1D).** For each coordinate $j \in [d]$, we compute the empirical 1-
2662 Wasserstein distance between the marginal distributions by matching order statistics. Let
2663 $x_j^{n:N}$ and $y_j^{n:N}$ denote the n -th order statistics (ascending) of $\{x_j^{(n)}\}_{n=1}^N$ and $\{y_j^{(n)}\}_{n=1}^N$,

2664 respectively. The reported score is the average over coordinates:

$$\widehat{\text{W1D}} = \frac{1}{Nd} \sum_{j=1}^d \sum_{n=1}^N |x_j^{n:N} - y_j^{n:N}|. \quad (2.114)$$

2665 This equals the empirical 1D Wasserstein distance per marginal, averaged across dimensions; it is invariant to permutations within each marginal sample.

2667 **Sliced-Wasserstein Distance (SWD).** We approximate the 1-Wasserstein distance
2668 between the real and generated joint distributions via random 1D projections. Draw unit
2669 vectors $\mathbf{v}_\ell \in \mathbb{S}^{d-1}$, $\ell = 1, \dots, L$, project $u_\ell^{(n)} = \langle \mathbf{x}^{(n)}, \mathbf{v}_\ell \rangle$ and $v_\ell^{(n)} = \langle \mathbf{y}^{(n)}, \mathbf{v}_\ell \rangle$, sort each
2670 set to obtain order statistics $u_\ell^{n:N}$ and $v_\ell^{n:N}$, and average the 1D distances:

$$\widehat{\text{SWD}} = \frac{1}{LN} \sum_{\ell=1}^L \sum_{n=1}^N |u_\ell^{n:N} - v_\ell^{n:N}|. \quad (2.115)$$

2671 As L increases, this Monte Carlo estimator tightens; SWD is sensitive to global geometry
2672 while remaining computationally light.

2673 **Pairwise Spearman difference.** To assess monotone dependence, we compare pairwise
2674 Spearman rank correlations between real and generated data. For each pair (i, j) , let ρ_{ij}^{real}
2675 and ρ_{ij}^{gen} denote Spearman's ρ . The metric is the mean absolute discrepancy over all pairs:

$$\text{Spearman diff} = \frac{2}{d(d-1)} \sum_{1 \leq i < j \leq d} |\rho_{ij}^{\text{real}} - \rho_{ij}^{\text{gen}}|. \quad (2.116)$$

2676 Being rank-based, it is invariant to strictly monotone marginal transforms.

2677 **Sortedness.** Order statistics must be sorted. We report the proportion of generated
2678 samples that satisfy the coordinate-wise descending order $y_1^{(n)} \geq y_2^{(n)} \geq \dots \geq y_d^{(n)}$:

$$\text{Sortedness} = \frac{1}{N} \sum_{n=1}^N \mathbb{I}\{y_k^{(n)} - y_{k+1}^{(n)} \geq 0, \forall k \in [d-1]\}. \quad (2.117)$$

2679 This hard score equals 1 only if all generated vectors respect the ordering constraint.

2680 **Absolute Kendall Error (AKE).** To compare distributions of order statistics, we sort
2681 each sample vector in descending order and compute the mean absolute deviation between
2682 the resulting order-statistic vectors:

$$\text{AKE} = \frac{1}{Nd} \sum_{n=1}^N \sum_{k=1}^d |\text{sort}_{\downarrow}(\mathbf{x}^{(n)})_k - \text{sort}_{\downarrow}(\mathbf{y}^{(n)})_k|. \quad (2.118)$$

2683 This captures alignment of empirical order-statistic marginals; lower is better.

2684 **Softsortedness.** As a differentiable proxy for the sortedness constraint, we average the
2685 positive violations of the descending order:

$$\text{SoftSorted} = \frac{1}{N(d-1)} \sum_{n=1}^N \sum_{k=1}^{d-1} [y_{k+1}^{(n)} - y_k^{(n)}]_+, \quad [t]_+ \stackrel{\text{def}}{=} \max\{0, t\}. \quad (2.119)$$

2686 This penalty is zero if and only if every generated vector is sorted in descending order and
2687 increases linearly with the magnitude of violations.

2688 **Chapter 3**

2689 **Gradient-Free Fine-Tuning of**
2690 **Diffusion Models**

2691 This chapter presents a gradient-free approach to fine-tuning diffusion models for reward-
2692 based generation. Using Fisher’s identity, we derive a score estimator that requires only
2693 forward evaluations of the reward function, avoiding costly backpropagation through the
2694 reward model.

2695 **3.1 Introduction**

2696 Diffusion models are now a standard tool for generative modelling in images, audio, and
2697 related domains (Jascha Sohl-Dickstein et al. 2015; Ho, Jain, et al. 2020; Y. Song, Jascha
2698 Sohl-Dickstein, et al. 2021). They learn a time-dependent score (or denoiser) that inverts
2699 a simple noising process and produces new samples from noise. Following their success, it
2700 has been proposed to fine-tune pretrained diffusion models, taking into account new data
2701 preferences and/or constraints by simply shifting the learnt weights accordingly.

2702 To achieve this, ideas from language-model alignment have recently started to be ap-
2703 plied to diffusion: Either shifting the pre-trained weights using methods based on reward
2704 models (Jiazheng Xu et al. 2023) or reinforcement learning policies (Black et al. 2024;
2705 Fan et al. 2023), or adapting the architecture using lightweight adapters such as LoRA
2706 (Hu et al. 2021). In addition, diffusion-specific techniques like classifier-free guidance (Ho
2707 and Salimans 2022) or improved noise schedulers (Alex Nichol and Dhariwal 2021) provide
2708 complementary control mechanisms.

2709 In this chapter we introduce *iterative tilting*, a gradient-free method for reward-based
2710 fine-tuning that shifts a diffusion model toward a reward-tilted density without differen-
2711 tiating through the reward function. During the preparation of this work, a concurrent
2712 paper was submitted to ICLR (Anonymous 2025). Their method is essentially the same
2713 as ours but formulated using the stochastic interpolants framework and developed in con-
2714 tinuous time. In contrast, our approach is based on a diffusion framework and is derived
2715 in discrete time.

2716 Section 3.2 recalls the theoretical framework of Denoising Diffusion Models (DDM) and
 2717 reviews existing reward fine-tuning formulations, Section 3.3 develops the Iterative Tilting
 2718 method from a theoretical standpoint, and Section 1.3 reports controlled experiments on
 2719 a Gaussian synthetic fine-tuning task, before we conclude on the work of this chapter.

2720

3.2 Preliminaries on Denoising Diffusion Models and

 2721

Fine-Tuning

2722

3.3 Diffusion models

2723 Denoising diffusion models (DDMs) (Jascha Sohl-Dickstein et al. 2015; Y. Song and Ermon
 2724 2019; Ho, Jain, et al. 2020) define a generative process for a data distribution p_0 by
 2725 constructing a continuous probability path $(p_t)_{t \in [0,1]}$ of distributions between p_0 and a
 2726 tractable reference $p_1 := N(0, I_d)$.

2727

Forward noising process

2728 The path is defined via the marginals $p_t = \text{Law}(X_t)$, where

$$X_t = \alpha_t X_0 + \sigma_t X_1, \quad X_0 \sim p_0, \quad X_1 \sim N(0, I_d). \quad (3.1)$$

2729 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
 2730 such that α_t is non-increasing, σ_t is non-decreasing, with boundary conditions $(\alpha_0, \sigma_0) =$
 2731 $(1, 0)$ and $(\alpha_1, \sigma_1) = (0, 1)$. The path $(p_t)_{t \in [0,1]}$ thus defines an interpolation that gradually
 2732 transforms the clean data distribution p_0 into the Gaussian reference p_1 .

2733 From (3.1), the conditional distribution of X_t given $X_0 = x_0$, denoted $q_{t|0}$, is Gaussian:

$$q_{t|0}(x_t | x_0) = N(x_t; \alpha_t x_0, \sigma_t^2 I_d). \quad (3.2)$$

2734 Regarding the choices of schedules (α_t, σ_t) , two standard choices are the *variance-*
 2735 *preserving (VP)* schedule with $\alpha_t^2 + \sigma_t^2 = 1$ (Ho, Jain, et al. 2020; Dhariwal and Alexander
 2736 Nichol 2021), which ensures $\text{Var}(X_t) = \text{Var}(X_0)$ when X_0 has unit variance, and the *linear*
 2737 (*flow matching*) schedule with $(\alpha_t, \sigma_t) = (1 - t, t)$ (Lipman et al. 2023; Esser et al. 2024),
 2738 corresponding to a straight-line interpolation between data and noise.

2739

Reverse process and DDIM transitions

2740 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
 2741 iteratively applied to map a sample from $p_{t_{k+1}}$ to one from p_{t_k} , progressively denoising
 2742 until convergence to p_0 .

2744

DDIM framework.

 The DDIM framework (J. Song et al. 2021) introduces a general
 2745 family of reverse transitions. It defines a schedule $(\eta_t)_{t \in [0,1]}$ satisfying $\eta_t \leq \sigma_t$ for all

2746 $t \in [0, 1]$, along with transition densities given for $s < t$ by

$$p_{s|t}^\eta(x_s | x_t) = \mathbb{E} \left[q_{s|0,1}^\eta(x_s | X_0, X_1) \mid X_t = x_t \right], \quad (3.3)$$

2747 where

$$q_{s|0,1}^\eta(x_s | x_0, x_1) := N\left(x_s; \alpha_s x_0 + \sqrt{\sigma_s^2 - \eta_s^2} x_1, \eta_s^2 I_d\right) \quad (3.4)$$

2748 and the random variables (X_0, X_t, X_1) are defined as in (3.1). By construction, this
2749 family satisfies the marginalisation property $p_s(x_s) = \int p_{s|t}^\eta(x_s | x_t) p_t(x_t) dx_t$: by the
2750 law of total expectation, the right-hand side equals $\mathbb{E}_{(X_0, X_1)}[q_{s|0,1}^\eta(x_s | X_0, X_1)]$, and since
2751 $X_1 \sim N(0, I_d)$ is independent of X_0 , sampling from the bridge (3.4) yields $X_s = \alpha_s X_0 +$
2752 $\sqrt{\sigma_s^2 - \eta_s^2} X_1 + \eta_s \epsilon$ with $\epsilon \sim N(0, I_d)$; the sum $\sqrt{\sigma_s^2 - \eta_s^2} X_1 + \eta_s \epsilon$ has variance $\sigma_s^2 I_d$, so
2753 $X_s \stackrel{d}{=} \alpha_s X_0 + \sigma_s \tilde{X}_1$ with $\tilde{X}_1 \sim N(0, I_d)$, matching the definition of p_s . This ensures that
2754 $(p_{t_k|t_{k+1}}^\eta)_{k=0}^{K-1}$ defines a consistent set of reverse transitions.

2755 **Denoiser.** The transitions (3.3) are intractable and have to be approximated. To this
2756 end, it has been proposed to replace X_0 and X_1 by their conditional expectations given
2757 X_t . Let $\hat{x}_0^\theta(\cdot, t)$ and $\hat{x}_1^\theta(\cdot, t)$ denote parametric estimators of

$$\hat{x}_0(x_t, t) := \mathbb{E}[X_0 | X_t = x_t], \quad \hat{x}_1(x_t, t) := \mathbb{E}[X_1 | X_t = x_t]. \quad (3.5)$$

2758 From the forward interpolation (3.1), these two quantities are related by $\hat{x}_1(x_t, t) = (x_t -$
2759 $\alpha_t \hat{x}_0(x_t, t)) / \sigma_t$, yielding the parametric relationship

$$\hat{x}_1^\theta(x_t, t) = \frac{x_t - \alpha_t \hat{x}_0^\theta(x_t, t)}{\sigma_t}, \quad \hat{x}_0^\theta(x_t, t) = \frac{x_t - \sigma_t \hat{x}_1^\theta(x_t, t)}{\alpha_t}. \quad (3.6)$$

2760 The parametric DDIM transition then becomes

$$p_{t_k|t_{k+1}}^{\eta, \theta}(x_{t_k} | x_{t_{k+1}}) := q_{t_k|0,1}^\eta(x_{t_k} | \hat{x}_0^\theta(x_{t_{k+1}}, t_{k+1}), \hat{x}_1^\theta(x_{t_{k+1}}, t_{k+1})). \quad (3.7)$$

2761 For $k = 0$, the transition $p_{0|t_1}^{\eta, \theta}(\cdot | x_{t_1})$ is simply defined as the Dirac mass at $\hat{x}_0^\theta(x_{t_1}, t_1)$.

2762 **Sampling procedure.** Given timesteps $(t_k)_{k=0}^K$ with $t_K = 1$ and $t_0 = 0$, DDIM sampling
2763 is summarised in Algorithm 5. Setting $\eta_t = 0$ yields deterministic sampling; setting
2764 $\eta_t = \sigma_t \sqrt{1 - \alpha_t^2 / \alpha_s^2}$ (for the VP schedule) recovers the stochastic DDPM sampler (Ho,
2765 Jain, et al. 2020).

2766 Score function and training

2767 **Score and denoiser relationship.** The *score* of the noised distribution is $\nabla \log p_t(\cdot)$.
2768 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^θ , timesteps $(t_k)_{k=0}^K$ with $t_K = 1$, $t_0 = 0$, variance schedule $(\eta_{t_k})_{k=0}^{K-1}$

- 1: Sample $x_{t_K} \sim N(0, I_d)$ **for** $k = K - 1, \dots, 1$ **do**
- 2: $\hat{x}_1 \leftarrow \hat{x}_1^\theta(x_{t_{k+1}}, t_{k+1})$ ▷ Predict X_1
- 3: $\hat{x}_0 \leftarrow (x_{t_{k+1}} - \sigma_{t_{k+1}} \hat{x}_1) / \alpha_{t_{k+1}}$ ▷ Estimate clean sample
- 4: $\mu_{t_k} \leftarrow \alpha_{t_k} \hat{x}_0 + \sqrt{\sigma_{t_k}^2 - \eta_{t_k}^2} \hat{x}_1$
- 5: Sample $z \sim N(0, I_d)$
- 6: $x_{t_k} \leftarrow \mu_{t_k} + \eta_{t_k} z$
- 7:
- 8: $x_0 \leftarrow (x_{t_1} - \sigma_{t_1} \hat{x}_1^\theta(x_{t_1}, t_1)) / \alpha_{t_1}$ ▷ Final denoising step
- 9: **return** x_0

2769 the denoiser. Exchanging expectation and gradient under standard regularity assumptions:

$$\nabla_{x_t} \log p_t(x_t) = \mathbb{E} [\nabla_{x_t} \log q_{t|0}(x_t | X_0) \mid X_t = x_t] = -\frac{\hat{x}_1(x_t, t)}{\sigma_t}. \quad (3.8)$$

2770 Thus, the denoiser \hat{x}_1^θ directly provides a score estimator $s_\theta(x_t, t) = -\hat{x}_1^\theta(x_t, t)/\sigma_t$, and
2771 conversely $\hat{x}_1^\theta(x_t, t) = -\sigma_t s_\theta(x_t, t)$.

2772 **Training objective.** The denoiser can be trained by regressing either X_0 or X_1 from
2773 the noised sample $X_t = \alpha_t X_0 + \sigma_t X_1$. The X_0 -prediction loss reads

$$L_{X_0}(\theta) = \int_0^1 \mathbb{E}_{X_0 \sim p_0, X_1 \sim N(0, I_d)} \|\hat{x}_0^\theta(\alpha_t X_0 + \sigma_t X_1, t) - X_0\|^2 dt, \quad (3.9)$$

2774 while the X_1 -prediction loss is

$$L_{X_1}(\theta) = \int_0^1 \mathbb{E}_{X_0 \sim p_0, X_1 \sim N(0, I_d)} \|\hat{x}_1^\theta(\alpha_t X_0 + \sigma_t X_1, t) - X_1\|^2 dt. \quad (3.10)$$

2775 Since $\hat{x}_1^\theta = -\sigma_t s_\theta$ by (3.8), the X_1 -prediction loss is equivalent to denoising score matching
2776 (Hyvärinen 2005; Vincent 2011): minimising (3.10) amounts to learning the score $\nabla \log p_t$.
2777 In practice, the integral is approximated via Monte Carlo: sample $t \sim \text{Unif}[0, 1]$, $x_0 \sim p_0$,
2778 $x_1 \sim N(0, I_d)$, form $x_t = \alpha_t x_0 + \sigma_t x_1$, and regress either x_0 or x_1 .

2779 Connection to SDEs

2780 The interpolation (3.1) can be viewed as discretising a continuous-time process. Given a
2781 noise schedule $\beta(t) > 0$, consider the Itô SDE

$$dX_t = -\frac{1}{2}\beta(t)X_t dt + \sqrt{\beta(t)} dW_t, \quad X_0 \sim p_0, \quad (3.11)$$

2782 where $(W_t)_{t \geq 0}$ is a standard Brownian motion in \mathbb{R}^d . For the VP schedule, setting $\alpha_t =$
2783 $\exp(-(1/2) \int_0^t \beta(s) ds)$ and $\sigma_t = \sqrt{1 - \alpha_t^2}$, the marginal Law(X_t) coincides with p_t from

2784 (3.1). The time-reversed process satisfies (Anderson 1982; Y. Song, Jascha Sohl-Dickstein,
2785 et al. 2021)

$$d\bar{X}_t = \left[-\frac{1}{2}\beta(t)\bar{X}_t + \beta(t)\nabla_x \log p_t(\bar{X}_t) \right] dt + \sqrt{\beta(t)} d\bar{W}_t, \quad (3.12)$$

2786 where $\bar{X}_t := X_{1-t}$ and \bar{W}_t is a standard Brownian motion. The deterministic *probability-*
2787 *flow ODE* (Y. Song, Jascha Sohl-Dickstein, et al. 2021) with identical marginals reads

$$dx_t = \left[-\frac{1}{2}\beta(t)x_t + \frac{1}{2}\beta(t)\nabla_x \log p_t(x_t) \right] dt. \quad (3.13)$$

2788 DDIM with $\eta_t = 0$ can be seen as a discretisation of (3.13), while DDPM corresponds to
2789 (3.12).

2790 3.B Fine-tuning formulations

2791 Given a positive reward function r and a pretrained diffusion model sampling from the
2792 base distribution p_0^b , we aim to fine-tune the model to sample from the *tilted distribution*,
2793 defined for $x \in \mathbb{R}^d$:

$$p_0^{\lambda r}(x) = Z_{\lambda r}^{-1} \exp(\lambda r(x)) p_0^b(x), \quad (3.14)$$

2794 where $\lambda > 0$ is the tilt strength and $Z_{\lambda r} = \int \exp(\lambda r(x)) p_0^b(x) dx$ is the normalising con-
2795 stant. This tilted distribution emphasises regions with high reward. Several formulations
2796 have been developed to tackle this goal.

2797 **Direct reward alignment (DRaFT).** DRaFT (Clark et al. 2024) treats the diffu-
2798 sion sampler as a differentiable computation graph. For network parameters θ , the re-
2799 verse sampling process is a deterministic function of θ and the i.i.d. Gaussian noises
2800 $\xi = (Z_K, \dots, Z_1)$, denoted $(\theta, \xi) \mapsto \psi_0(\theta, \xi)$, i.e., $\psi_0(\theta, \xi) = \hat{X}_0$ where \hat{X}_0 is the out-
2801 put of Algorithm 5.

2802 DRaFT maximises the expected reward:

$$J(\theta) = \mathbb{E}_{\xi} [r(\psi_0(\theta, \xi))] \quad (3.15)$$

2803 by backpropagating the reward gradient through the entire chain of reverse steps:

$$\nabla_{\theta} J(\theta) = \mathbb{E}_{\xi} \left[\left(\frac{\partial \psi_0(\theta, \xi)}{\partial \theta} \right)^{\top} \nabla_x r(\psi_0(\theta, \xi)) \right]. \quad (3.16)$$

2804 Full backpropagation is memory-intensive. DRaFT- K reduces cost by differentiat-
2805 ing only through the final K steps (typically $K \in \{1, 4\}$), detaching earlier gradients.
2806 DRaFT-LV further refines $K = 1$ with a control-variate correction for lower variance
2807 while remaining unbiased.

2808 **Stochastic optimal control and Adjoint Matching.** An alternative approach casts
2809 fine-tuning as stochastic optimal control (SOC) (Uehara et al. 2024; Tang 2024; Domingo-
2810 Enrich, Drozdza, et al. 2024). Rather than backpropagating through sampling (as in

2811 DRaFT), SOC methods add a drift control u_t to shift the generative process toward high-
2812 reward regions. For a controlled SDE $dX_t = [b(X_t, t) + g(t)u_t(X_t)] dt + g(t) dW_t$, where
2813 $g(t)$ denotes the diffusion coefficient (distinct from the noise schedule σ_t in (3.1)), the
2814 objective balances control effort against terminal reward:

$$\min_u \mathbb{E} \left[\frac{1}{2} \int_0^1 \|u_t(X_t)\|^2 dt - \lambda r(X_1) \right]. \quad (3.17)$$

2815 By Girsanov’s theorem, the quadratic control cost equals the KL divergence between con-
2816 trolled and base processes. However, naïvely solving (3.17) yields $p^*(X_0, X_1) \propto p^{\text{base}}(X_0, X_1) \exp(\lambda r(X_1))$
2817 $V(X_0, 0)$, where V is the value function. This does not marginalize to the tilted distri-
2818 bution (3.14) unless the initial noise X_0 and generated sample X_1 are independent.

2819 (Domingo-Enrich, Drozdzal, et al. 2024) show that a *memoryless* diffusion coefficient,
2820 one that ensures $X_0 \perp X_1$, removes the value function bias. Under this schedule, the
2821 optimal control is $u_t^*(x) = g(t)^\top \tilde{a}_t(x)$, where the *lean adjoint* \tilde{a}_t satisfies the backward
2822 ODE

$$\frac{d}{dt} \tilde{a}_t = -[\nabla_x b(x, t)]^\top \tilde{a}_t, \quad \tilde{a}_1 = \lambda \nabla_x r(X_1). \quad (3.18)$$

2823 Adjoint Matching learns a parametric control u_ϕ by minimizing:

$$L_{\text{AM}}(\phi) = \mathbb{E} \left[\int_0^1 \|u_\phi(X_t, t) + g(t)^\top \tilde{a}_t(X_t)\|^2 dt \right]. \quad (3.19)$$

2824 After training, the learned control is absorbed into the drift, yielding a fine-tuned score
2825 or velocity field. Sampling can then use any diffusion coefficient, including deterministic
2826 sampling with $g(t) = 0$.

2827 3.3 Iterative Tilting Method for Fine-Tuning Diffu- 2828 sion Models

2829 The methods reviewed in Section 3.B share a common limitation: they require differen-
2830 tiating through the reward function r . In DRaFT, reward gradients appear explicitly in
2831 (3.16) via backpropagation through the sampling chain. In stochastic optimal control, they
2832 appear in the adjoint terminal condition $a_0 = \lambda \nabla_x r(X_0)$. When r is a large pretrained
2833 model (such as a vision transformer or human preference classifier trained on millions of
2834 examples), computing these gradients becomes prohibitively expensive, both in memory
2835 and compute. Moreover, many reward functions of practical interest are non-differentiable
2836 or defined only through black-box evaluations.

2837 We introduce *iterative tilting*, a method that targets the reward-tilted distribution
2838 (3.14) using only forward evaluations of r , without any gradient computation. The key
2839 idea is to construct a path of intermediate distributions $(p_0^k)_{k=0}^N$ connecting the base p_0^b to
2840 the tilted $p_0^{\lambda r}$, each representing a gentle reweighting of the previous one. At each step,
2841 we reweight samples from the current distribution toward higher rewards, gradually ap-
2842 proaching the target along this path. This sequential approach trades off a single expensive

2843 gradient-based update for multiple cheaper gradient-free updates.

2844 This section develops the method in continuous time, which makes the structure of the
2845 approximation explicit and connects naturally to the score-based and control perspectives
2846 introduced in Section 3.2.

2847 3.A Score of the tilted distribution

2848 Suppose we have a data distribution $X_0 \sim p_0^b$ (the base distribution) and wish to sample
2849 from the tilted distribution (3.14). We use the forward noising kernel $q_{t|0}$ from (3.2), whose
2850 score is available in closed form:

$$\nabla_{x_t} \log q_{t|0}(x_t | x_0) = \frac{\alpha_t x_0 - x_t}{\sigma_t^2}. \quad (3.20)$$

2851 Noising the tilted distribution, the marginal at time t is:

$$p_t^{\lambda r}(x_t) = \int q_{t|0}(x_t | x_0) p_0^{\lambda r}(x_0) dx_0. \quad (3.21)$$

2852 For any prior density p_0° (e.g., base p_0^b or tilted $p_0^{\lambda r}$) and the fixed forward noising kernel
2853 $q_{t|0}(x_t | x_0)$, we write $p_{0|t}^\circ(x_0 | x_t)$ for the posterior (inference) density of the clean sample
2854 X_0 given its noised version $X_t = x_t$ at time t :

$$p_{0|t}^\circ(x_0 | x_t) \propto p_0^\circ(x_0) q_{t|0}(x_t | x_0). \quad (3.22)$$

2855 *Proposition 3.1* (Score of the tilted distribution). Assume that r has at most linear growth,
2856 i.e., $\forall x \in \mathbb{R}^d : |r(x)| \leq C(1 + \|x\|)$ for some $C > 0$, and that $q_{t|0}$ is the Gaussian forward
2857 kernel (3.2). Then for all $t \in (0, 1]$ and $x_t \in \mathbb{R}^d$,

$$\nabla_{x_t} \log p_t^{\lambda r}(x_t) = \frac{\mathbb{E}_{X_0 \sim p_{0|t}^b(\cdot|x_t)} [\nabla_{x_t} \log q_{t|0}(x_t | X_0) \exp(\lambda r(X_0))]}{\mathbb{E}_{X_0 \sim p_{0|t}^b(\cdot|x_t)} [\exp(\lambda r(X_0))]} . \quad (3.23)$$

2858 *Proof.* Exchanging differentiation and integration in (3.21) (justified by dominated convergence under the stated assumptions) yields

$$\nabla_{x_t} p_t^{\lambda r}(x_t) = \int p_0^{\lambda r}(x_0) \nabla_{x_t} q_{t|0}(x_t | x_0) dx_0. \quad (3.24)$$

2860 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
2861 identity:

$$\nabla_{x_t} \log p_t^{\lambda r}(x_t) = \mathbb{E}_{X_0 \sim p_{0|t}^{\lambda r}(\cdot|x_t)} [\nabla_{x_t} \log q_{t|0}(x_t | X_0)] . \quad (3.25)$$

2862 The key observation is that the tilted posterior can be written as a reweighting of the base

2863 posterior:

$$\begin{aligned} p_{0|t}^{\lambda r}(x_0 | x_t) &= \frac{p_0^{\lambda r}(x_0) q_{t|0}(x_t | x_0)}{p_t^{\lambda r}(x_t)} = \frac{Z_{\lambda r}^{-1} \exp(\lambda r(x_0)) p_0^b(x_0) q_{t|0}(x_t | x_0)}{\int p_0^{\lambda r}(x'_0) q_{t|0}(x_t | x'_0) dx'_0} \\ &= \frac{\exp(\lambda r(x_0)) p_{0|t}^b(x_0 | x_t)}{\mathbb{E}_{X'_0 \sim p_{0|t}^b(\cdot | x_t)} [\exp(\lambda r(X'_0))]}, \end{aligned} \quad (3.26)$$

2864 where in the last equality, the marginal $p_t^b(x_t)$ cancels out. The tilted posterior $p_{0|t}^{\lambda r}(\cdot | x_t)$
2865 is thus an exponential tilt of the base posterior $p_{0|t}^b(\cdot | x_t)$. Substituting (3.26) into (3.25)
2866 gives (3.23). \square

2867 **Iterative tilts.** To efficiently approximate the score of the tilted distribution, we decom-
2868 pose the full tilt into a sequence of smaller tilts. Fix $N \in \mathbb{N}^*$ (typically large) and define,
2869 for $k = 0, \dots, N$,

$$p_0^k(x) := Z_k^{-1} \exp\left(\frac{k}{N} \lambda r(x)\right) p_0^b(x), \quad (3.27)$$

2870 where Z_k is the normalising constant. We have $p_0^0 = p_0^b$ and $p_0^N = p_0^{\lambda r}$. The key observation
2871 is that consecutive distributions are related by a small tilt: $p_0^k(x) \propto \exp(\lambda r(x)/N) p_0^{k-1}(x)$.

2872 **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 $\pi_{k-1}(\cdot) := p_{0|t}^{k-1}(\cdot | x_t)$ and the covariance*

$$C_{k-1}(x_t, t) := \text{Cov}_{X_0 \sim \pi_{k-1}} (\nabla_{x_t} \log q_{t|0}(x_t | X_0), r(X_0)). \quad (3.28)$$

2875 Then, for large N ,

$$\nabla_{x_t} \log p_t^k(x_t) = \nabla_{x_t} \log p_t^{k-1}(x_t) + \frac{\lambda}{N} C_{k-1}(x_t, t) + O\left(\frac{1}{N^2}\right). \quad (3.29)$$

2876 *Proof.* Applying Proposition 3.1 with base p_0^{k-1} and tilt strength λ/N :

$$\nabla_{x_t} \log p_t^k(x_t) = \frac{\mathbb{E}_{\pi_{k-1}} [g(X_0) \exp(\frac{\lambda}{N} r(X_0))]}{\mathbb{E}_{\pi_{k-1}} [\exp(\frac{\lambda}{N} r(X_0))]}, \quad (3.30)$$

2877 where $g(X_0) := \nabla_{x_t} \log q_{t|0}(x_t | X_0)$. Taylor-expanding $\exp(\lambda r/N) = 1 + \lambda r/N + O(N^{-2})$
2878 in both numerator and denominator:

$$\begin{aligned} \text{Numerator} &= \mathbb{E}_{\pi_{k-1}} [g] + \frac{\lambda}{N} \mathbb{E}_{\pi_{k-1}} [g r] + O(N^{-2}), \\ \text{Denominator} &= 1 + \frac{\lambda}{N} \mathbb{E}_{\pi_{k-1}} [r] + O(N^{-2}). \end{aligned}$$

2879 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
2880 Fisher's identity, $\mathbb{E}_{\pi_{k-1}} [g] = \nabla_{x_t} \log p_t^{k-1}(x_t)$. \square

2881 3.B Neural approximation and algorithm

2882 Corollary 3.1 motivates a recursive algorithm for estimating the sequence of score functions
 2883 $\{\nabla_{x_t} \log p_t^k\}_{k=1}^N$ starting from $k = 0$, for which $p_t^0 = p_t^b$. Suppose that at iteration k , we
 2884 have access to an approximation for $\nabla_{x_t} \log p_t^k$, denoted by $s_{\theta^k}^k$. The resulting generative
 2885 models associated to $s_{\theta^k}^k$ is called the teacher at step k . Corollary 3.1 provides a natural
 2886 procedure to obtain an approximation for $\nabla_{x_t} \log p_t^{k+1}$: we train a new network s_{θ}^{k+1} with
 2887 weights θ , so that it matches the target score from (3.29). The updated parameter θ^{k+1} is
 2888 thus obtained as an approximate minimizer of the loss

$$2889 L_{\text{IT}}(\theta; k+1) = \int_0^1 \int_{\mathcal{X}} w(t) \left\| s_{\theta}(x_t, t) - \left(s_{\theta^k}^k(x_t, t) + \frac{\lambda}{N} C_k(x_t, t) \right) \right\|^2 p_t^k(x_t) dx_t dt. \quad (3.31)$$

2889 In practice, we make three simplifications. First, we use a Monte Carlo estimate of the
 2890 loss using a teacher-generated dataset $\{X_0^i\}_{i=1}^N$ at the beginning of each step k . Second,
 2891 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
 2892 $g^i = r(X_0^i)(\nabla_{x_t} \log q_{t|0}(X_t^i | X_0^i) - s_{\theta^k}(X_t^i, t))$, where $\nabla_{x_t} \log q_{t|0}$ is computed via (3.20).
 2893 Finally, we multiply by σ_t^2 to stabilize training across noise levels. This leads to the
 2894 following empirical loss for a single sample time t and pair (X_0^i, X_1^i) : setting $\delta = \lambda/N$,

$$2895 L_{\text{prac}}(\theta; k+1) = \sum_{i=1}^N \sigma_t^2 \left\| s_{\theta}^{k+1}(X_t^i, t) - \sigma_t^2 (s_{\theta^k}^k(X_t^i, t) + \delta r(X_0^i) (g^i - s_{\theta^k}^k(X_t^i, t))) \right\|^2, \quad (3.32)$$

2895 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
 2896 at step $k+1$, s_{θ}^{k+1} is initialized at $s_{\theta^k}^k$. The iterative tilting procedure is summarised in
 2897 Algorithm 6.

2898 3.4 Experiments

2899 In this section, we validate the iterative tilting method on a two-dimensional Gaussian
 2900 mixture with a reward function. This setup has a key advantage: tilting a Gaussian (or
 2901 mixture of Gaussians) with a quadratic reward (including linear as a special case) yields
 2902 another Gaussian (or mixture), for which we can compute the exact tilted density and
 2903 score analytically.

2904 3.A Gaussian data with quadratic rewards

2905 **One Gaussian.** Suppose the base density is Gaussian:

$$2906 p_0^b(x) = \frac{1}{(2\pi)^{d/2} |\Sigma|^{1/2}} \exp \left\{ -\frac{1}{2} (x - \mu)^T \Sigma^{-1} (x - \mu) \right\} = N(x; \mu, \Sigma). \quad (3.33)$$

2906 and the reward function r is quadratic, i.e.:

$$2907 r(x) = \frac{1}{2} x^T A x + b^T x + c. \quad (3.34)$$

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)

- 1: Set step size $\delta \leftarrow \lambda/N$ **for** $k = 1$ to N **do**
- 2: Generate dataset D_k of S samples: $x_0 \leftarrow \text{DDIM}(s_{\theta^k}, n)$
- 3: Initialise $\theta \leftarrow \theta^k$ and freeze teacher s_{θ^k} **for** epoch = 1 to E **do**
- 4: Sample mini-batch $\{X_0^{(i)}\}_{i=1}^B \subset D_k$, $\{t_i\}_{i=1}^B \stackrel{\text{iid}}{\sim} \text{Unif}[0, 1]$, $\{Z_i\}_{i=1}^B \stackrel{\text{iid}}{\sim} N(0, I_d)$ **for**
- 5: *i* = 1 to B **do**
- 6: Noise: $x_{t_i}^{(i)} = \alpha_{t_i} X_0^{(i)} + \sigma_{t_i} Z_i$
- 7: Compute: $g_i = (\alpha_{t_i} X_0^{(i)} - X_{t_i}^{(i)}) / \sigma_{t_i}^2$
- 8: Compute: $s_i^{\text{old}} = s_{\theta^k}(X_{t_i}^{(i)}, t_i)$
- 9: Target: $\text{target}_i = s_i^{\text{old}} + \delta r(X_0^{(i)}) (g_i - s_i^{\text{old}})$
- 10: Update θ by one step of gradient descent on
$$\frac{1}{B} \sum_{i=1}^B \left\| \sigma_{t_i}^2 s_{\theta}(X_{t_i}^{(i)}, t_i) - \sigma_{t_i}^2 \text{target}_i \right\|^2$$
- 11:
- 12: Set $\theta^k \leftarrow \theta$
- 13:
- 14: **return** s_{θ^N}

²⁹⁰⁷ Then the product $\exp(r(x))p_0^b(x)$ writes:

$$\frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp \left\{ -\frac{1}{2}(x - \mu)^T \Sigma^{-1}(x - \mu) + \left(\frac{1}{2}x^T Ax + b^T x + c \right) \right\}. \quad (3.35)$$

²⁹⁰⁸ Focusing on the term in the exponent:

$$\begin{aligned} & -\frac{1}{2}(x - \mu)^T \Sigma^{-1}(x - \mu) + \left(\frac{1}{2}x^T Ax + b^T x + c \right) \\ & = -\frac{1}{2} [x^T (\Sigma^{-1} - A)x - 2(\Sigma^{-1}\mu + b)^T x] + c - \frac{1}{2}\mu^T \Sigma^{-1}\mu. \end{aligned} \quad (3.36)$$

²⁹⁰⁹ With $\Sigma' = (\Sigma^{-1} - A)^{-1}$ and $\mu' = \Sigma'(\Sigma^{-1}\mu + b)$ (we assume $\Sigma^{-1} - A$ is positive definite
²⁹¹⁰ so that the tilt is normalizable), the exponential can be rewritten as

$$\exp(r(x))p_0^b(x) = \left(\frac{|\Sigma'|}{|\Sigma|} \right)^{1/2} \exp \left\{ \frac{1}{2} [\mu'^T \Sigma'^{-1} \mu' - \mu^T \Sigma^{-1} \mu] + c \right\} N(x; \mu', \Sigma'). \quad (3.37)$$

²⁹¹¹ Consequently,

$$\int \exp(r(x))p_0^b(x) dx = \left(\frac{|\Sigma'|}{|\Sigma|} \right)^{1/2} \exp \left\{ \frac{1}{2} [\mu'^T \Sigma'^{-1} \mu' - \mu^T \Sigma^{-1} \mu] + c \right\}. \quad (3.38)$$

²⁹¹² **Mixture of Gaussians.** Suppose that we now have a mixture of Gaussians with $(\omega_1, \dots, \omega_K) \in$
²⁹¹³ Δ_{K-1} where

$$\Delta_{K-1} = \{(\omega_1, \dots, \omega_K) \in \mathbb{R}_+^K : \sum_{i \in [K]} \omega_i = 1\} \quad (3.39)$$

²⁹¹⁴ is the simplex and $(\mu_1, \Sigma_1), \dots, (\mu_K, \Sigma_K)$ corresponding means and covariances.

$$p_0^b(x) = \sum_{i \in [K]} \omega_i N(x; \mu_i, \Sigma_i). \quad (3.40)$$

²⁹¹⁵ Then, we have:

$$\begin{aligned} \exp(r(x))p_0^b(x) &= \sum_{i \in [K]} \omega_i \exp(r(x)) N(x; \mu_i, \Sigma_i) \\ &= \sum_{i \in [K]} \omega'_i N(x; \mu'_i, \Sigma'_i), \end{aligned} \quad (3.41)$$

²⁹¹⁶ where the parameters of the tilted Gaussians are

$$\begin{cases} \Sigma'_i = (\Sigma_i^{-1} - A)^{-1}, \\ \mu'_i = \Sigma'_i(\Sigma_i^{-1}\mu_i + b), \\ \omega'_i = \omega_i \left(\frac{|\Sigma'_i|}{|\Sigma_i|} \right)^{1/2} \exp \left\{ \frac{1}{2} [\mu'^T \Sigma'^{-1} \mu' - \mu^T \Sigma^{-1} \mu] + c \right\}. \end{cases} \quad (3.42)$$

²⁹¹⁷ We consider the following mixture of two Gaussians in dimension $d = 2$ as base distribution:
²⁹¹⁸

$$\begin{cases} K = 2 \\ \mu_1 = (-2, 0); \mu_2 = (2, 0) \\ \Sigma_1 = \Sigma_2 = \sigma^2 \mathbf{I}; \sigma^2 = 1/2 \\ \omega_1 = \omega_2 = 1/2 \end{cases} \quad (3.43)$$

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:

$$\begin{cases} K' = 2 \\ \mu'_1 = (-2, 2); \mu'_2 = (2, 2) \\ \Sigma'_1 = \Sigma'_2 = \sigma^2 \mathbf{I}; \sigma^2 = 1/2 \\ \omega'_1 = \omega'_2 = 1/2. \end{cases} \quad (3.44)$$

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 embedding (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 $k = 1, \dots, N$:

$$\text{RMSE}_k = \mathbb{E}_{t \sim \text{Unif}[0,1], X_t \sim p_t^k} [\|s_{\theta^k}(X_t, t) - \nabla_x \log p_t^k(X_t)\|^2]^{1/2}. \quad (3.45)$$

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 timestamp total sampling time (s) and total training time (s) and evaluation metrics (negative log-likelihood and mean squared mean error) in Table 3.1.

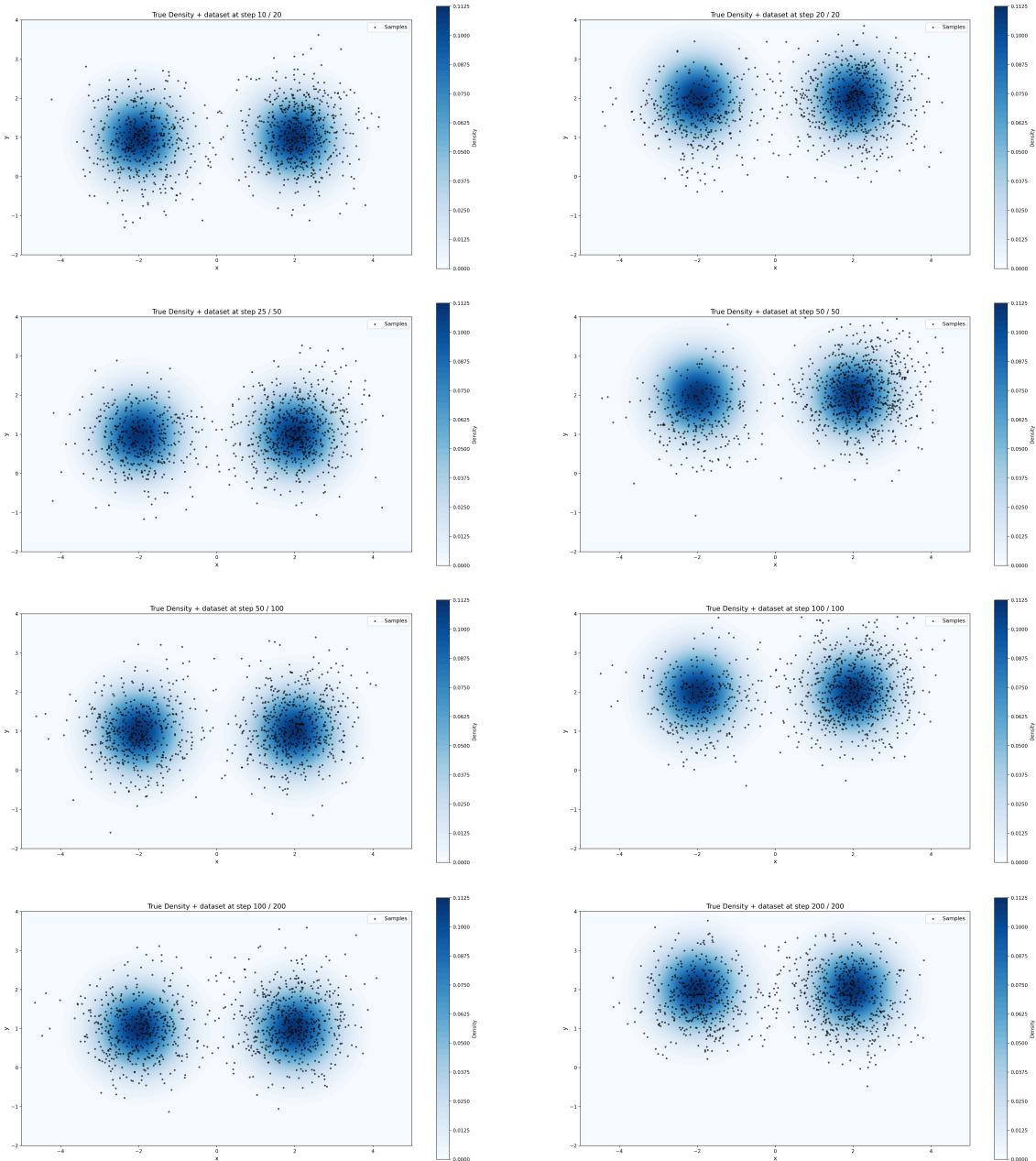


Figure 3.1: Iterative tilting on the 2-D GMM. Each row shows fixed- N settings (top to bottom: $N = 20, 50, 100, 200$) at iterations $N/2$ (left) and N (right). The rightmost plot in each row corresponds to the target distribution.

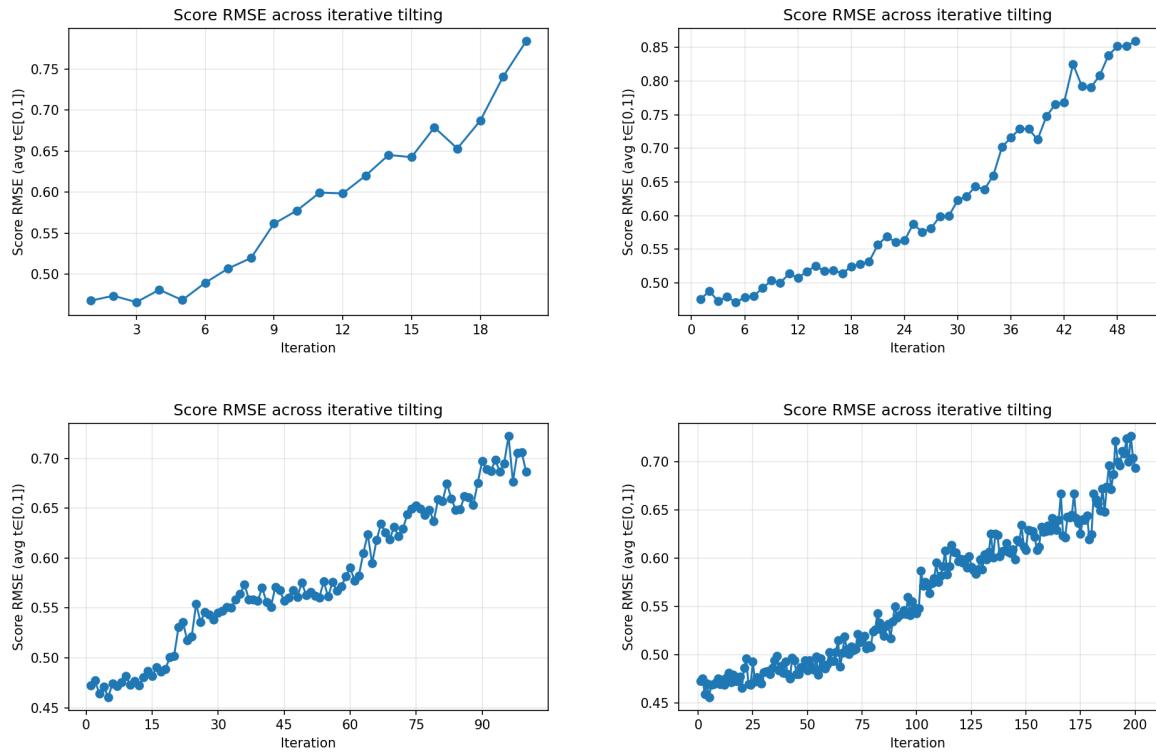


Figure 3.2: Evolution of the score error during iterative tilting for different N . We plot RMSE ($\sqrt{\text{MSE}}$) versus tilt iteration for $N \in \{20, 50, 100, 200\}$.

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

$$\nabla_x \log p_t^k(x) \approx \nabla_x \log p_t^{k-1}(x) + \frac{\lambda}{N} \text{Cov}_{X_0 \sim p_{0|t}^{k-1}(\cdot|x)} (\nabla_x \log q_{t|0}(x | X_0), r(X_0)),$$

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 .

Number of tilts N	Runtime (s)		Metrics	
	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).

2971 References

- 2972 Akiba, Takuya, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama (July
2973 2019). *Optuna: A Next-generation Hyperparameter Optimization Framework*. Comment:
2974 10 pages, Accepted at KDD 2019 Applied Data Science track. arXiv: [1907.10902](https://arxiv.org/abs/1907.10902)
2975 [[cs](#)]. (Visited on 08/03/2025).
- 2976 Allouche, M., S. Girard, and E. Gobet (2022). “EV-GAN: Simulation of extreme events
2977 with ReLU neural networks”. In: *Journal of Machine Learning Research* 23.150, pp. 1–
2978 39. ISSN: 1533-7928. URL: <http://jmlr.org/papers/v23/21-0663.html> (visited on
2979 05/16/2023).
- 2980 — (2024). “Estimation of extreme quantiles from heavy-tailed distributions with neural
2981 networks”. In: *Statistics and Computing* 34, p. 12.
- 2982 Allouche, Michaël, Stéphane Girard, and Emmanuel Gobet (2022). *Robust Estimation for
2983 Extremal Dependence*. arXiv: [2206.10876](https://arxiv.org/abs/2206.10876).
- 2984 — (Feb. 2025). “Learning extreme expected shortfall and conditional tail moments with
2985 neural networks. Application to cryptocurrency data”. In: *Neural Networks* 182, p. 106903.
2986 ISSN: 08936080. DOI: [10.1016/j.neunet.2024.106903](https://doi.org/10.1016/j.neunet.2024.106903). (Visited on 02/21/2025).
- 2987 Allouche, Michaël, Emmanuel Gobet, and Stéphane Girard (2024). “Extreme Event Simu-
2988 lation Using Generative Adversarial Networks”. In: *Handbook of Statistics*. Chapter on
2989 extreme value GAN methods. Elsevier.
- 2990 Anderson, Brian D.O. (1982). “Reverse-time diffusion equation models”. In: *Stochastic
2991 Processes and their Applications* 12.3, pp. 313–326. DOI: [10.1016/0304-4149\(82\)
2992 90051-5](https://doi.org/10.1016/0304-4149(82)90051-5).
- 2993 Anonymous (2025). “Tilt Matching for Scalable Sampling and Fine-Tuning”. In: *Submitted
2994 to The Fourteenth International Conference on Learning Representations*. under review.
2995 URL: <https://openreview.net/forum?id=tT7CXL3I9C>.
- 2996 Arjovsky, Martin, Soumith Chintala, and Léon Bottou (2017). “Wasserstein Generative
2997 Adversarial Networks”. In: *Proceedings of the 34th International Conference on Ma-
2998 chine Learning*. Vol. 70. Proceedings of Machine Learning Research. PMLR, pp. 214–
2999 223.
- 3000 Arnold, Barry C., Narayanaswamy Balakrishnan, and Haikady N. Nagaraja (2008). *A First
3001 Course in Order Statistics*. Classics in Applied Mathematics. Philadelphia: SIAM. DOI:
3002 [10.1137/1.9780898719062](https://doi.org/10.1137/1.9780898719062).

- 3003 Asmussen, S. and H. Albrecher (2010). *Ruin probabilities*. second. Advanced Series on
3004 Statistical Science & Applied Probability, 14. World Scientific Publishing Co. Pte. Ltd.,
3005 Hackensack, NJ.
- 3006 Bai, Yuntao et al. (2022). *Training a Helpful and Harmless Assistant with Reinforcement*
3007 *Learning from Human Feedback*. arXiv: [2204.05862](https://arxiv.org/abs/2204.05862).
- 3008 Banner, Adrian D. and Raouf Ghomrasni (July 2008). “Local times of ranked continuous
3009 semimartingales”. In: *Stochastic Processes and their Applications* 118.7, pp. 1244–1253.
3010 ISSN: 03044149. DOI: [10.1016/j.spa.2007.08.001](https://doi.org/10.1016/j.spa.2007.08.001). (Visited on 12/12/2024).
- 3011 Barrera, D., S. Crépey, E. Gobet, Hoang-Dung Nguyen, and B. Saadeddine (2025). “Statisti-
3012 cal Learning of Value-at-Risk and Expected Shortfall”. In: *Mathematical Finance, to*
3013 *appear*. arXiv: [2209.06476 \[q-fin, stat\]](https://arxiv.org/abs/2209.06476). URL: <http://arxiv.org/abs/2209.06476>
3014 (visited on 09/20/2024).
- 3015 Barron, Andrew R. (1993). “Universal approximation bounds for superpositions of a sig-
3016 moidal function”. In: *IEEE Transactions on Information Theory* 39.3, pp. 930–945.
3017 DOI: [10.1109/18.256500](https://doi.org/10.1109/18.256500).
- 3018 Beirlant, J., Y. Goegebeur, J. Teugels, and J. Segers (2004). *Statistics of extremes: theory*
3019 *and applications*. en. Wiley series in probability and statistics. Hoboken, NJ: Wiley.
3020 ISBN: 978-0-471-97647-9.
- 3021 Bellman, Richard (1961). *Adaptive Control Processes: A Guided Tour*. Princeton, NJ:
3022 Princeton University Press.
- 3023 Bevacqua, Emanuele, Carlo De Michele, Colin Manning, Anais Couasnon, Andreia F. S.
3024 Ribeiro, Alexandre M. Ramos, Edoardo Vignotto, Ana Bastos, Suzana Blesic, Fabrizio
3025 Durante, et al. (2021). “Guidelines for studying diverse types of compound weather and
3026 climate events”. In: *Earth's Future* 9.11, e2021EF002340. DOI: [10.1029/2021EF002340](https://doi.org/10.1029/2021EF002340).
- 3027 Bhattacharya, P. K. (1974). “Convergence of Sample Paths of Normalized Sums of Induced
3028 Order Statistics”. In: *The Annals of Statistics* 2.5. DOI: [10.1214/aos/1176342823](https://doi.org/10.1214/aos/1176342823).
- 3029 Black, Kevin, Michael Janner, Yilun Du, Ilya Kostrikov, and Sergey Levine (Jan. 2024).
3030 *Training Diffusion Models with Reinforcement Learning*. DOI: [10.48550/arXiv.2305.13301](https://doi.org/10.48550/arXiv.2305.13301). arXiv: [2305.13301 \[cs\]](https://arxiv.org/abs/2305.13301). (Visited on 09/18/2025).
- 3032 Blondel, Mathieu, Olivier Teboul, Quentin Berthet, and Josip Djolonga (2020). “Fast Dif-
3033 ferentiable Sorting and Ranking”. In: *Proceedings of the 37th International Conference*
3034 *on Machine Learning*. PMLR, pp. 950–959.
- 3035 Boulaguiem, Y., J. Zscheischler, E. Vignotto, K. van der Wiel, and S. Engelke (2022).
3036 “Modeling and simulating spatial extremes by combining extreme value theory with
3037 generative adversarial networks”. In: *Environmental Data Science* 1, e5.
- 3038 Boulaguiem, Youssef, Jakob Zscheischler, Edoardo Vignotto, Karin van der Wiel, and
3039 Sebastian Engelke (2022). “Modeling and simulating spatial extremes by combining
3040 extreme value theory with generative adversarial networks”. In: *Environmental Data*
3041 *Science* 1, e5. DOI: [10.1017/eds.2022.4](https://doi.org/10.1017/eds.2022.4).
- 3042 Bucklew, J. A. (2004). *Introduction to rare event simulation*. Springer Series in Statistics.
3043 Springer-Verlag, New York.

- 3044 Casper, Stephen, Xander Davies, Claudia Shi, Thomas Krendl Gilbert, Jérémie Scheurer,
3045 Javier Rando, Rachel Freedman, Tomasz Korbak, David Lindner, Pedro Freire, et al.
3046 (2023). *Open Problems and Fundamental Limitations of Reinforcement Learning from*
3047 *Human Feedback*. arXiv: [2307.15217](https://arxiv.org/abs/2307.15217).
- 3048 Christiano, Paul F., Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario
3049 Amodei (2017). *Deep Reinforcement Learning from Human Preferences*. NeurIPS 2017.
3050 arXiv: [1706.03741](https://arxiv.org/abs/1706.03741).
- 3051 Clark, Kevin, Paul Vicol, Kevin Swersky, and David J. Fleet (2024). *Directly Fine-Tuning*
3052 *Diffusion Models on Differentiable Rewards*. ICLR 2024. arXiv: [2309.17400](https://arxiv.org/abs/2309.17400).
- 3053 Cont, R., M. Cucuringu, R. Xu, and C. Zhang (2022). “Tail-GAN: Learning to Simulate
3054 Tail Risk Scenarios”. In: *SSRN*.
- 3055 Cont, Rama (2001). “Empirical Properties of Asset Returns: Stylized Facts and Statistical
3056 Issues”. In: *Quantitative Finance* 1.2, pp. 223–236. DOI: [10.1080/713665670](https://doi.org/10.1080/713665670).
- 3057 Cremer, F., B. Sheehan, M. Fortmann, A.N. Kia, M. Mullins, F. Murphy, and S. Materne
3058 (2022). “Cyber risk and cybersecurity: a systematic review of data availability”. In:
3059 *The Geneva Papers on risk and insurance-Issues and practice* 47.3, pp. 698–736.
- 3060 Cuturi, Marco, Olivier Teboul, and Jean-Philippe Vert (2019). “Differentiable Ranking
3061 and Sorting using Optimal Transport”. In: *Advances in Neural Information Processing*
3062 *Systems*. Vol. 32. Curran Associates, Inc.
- 3063 Cybenko, George (1989). “Approximation by superpositions of a sigmoidal function”.
3064 In: *Mathematics of Control, Signals, and Systems* 2.4, pp. 303–314. DOI: [10.1007/BF02551274](https://doi.org/10.1007/BF02551274).
- 3065 David, H.A. and H.N. Nagaraja (2004). *Order statistics*. Wiley series in probability and
3066 statistics. Wiley. ISBN: 978-0-471-65401-8. URL: <https://books.google.fr/books?id=bdhzFXg6xFkC>.
- 3067 David, Herbert A. and Haikady N. Nagaraja (2003). *Order Statistics*. 3rd. Wiley Series in
3068 Probability and Statistics. Hoboken, NJ: Wiley-Interscience. ISBN: 978-0-471-38926-2.
- 3069 David, Herbert A (1973). “Concomitants of order statistics”. In: *Bull. Inst. Internat.*
3070 *Statist.* 45.1, pp. 295–300.
- 3071 Del Moral, P. and J. Garnier (2005). “Genealogical particle analysis of rare events”. In:
3072 *The Annals of Applied Probability* 15.4, pp. 2496–2534.
- 3073 Dhariwal, Prafulla and Alexander Nichol (2021). “Diffusion Models Beat GANs on Image
3074 Synthesis”. In: *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato,
3075 A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan. Vol. 34. Curran
3076 Associates, Inc., pp. 8780–8794.
- 3077 Dinh, Laurent, Jascha Sohl-Dickstein, and Samy Bengio (2017). “Density estimation using
3078 Real-NVP”. In: *International Conference on Learning Representations*.
- 3079 Domingo-Enrich, Carles, Michal Drozdzal, Brian Karrer, and Ricky T. Q. Chen (Jan. 2024).
3080 *Adjoint Matching: Fine-tuning Flow and Diffusion Generative Models with Memoryless*
3081 *Stochastic Optimal Control*. DOI: [10.48550/arXiv.2409.08861](https://doi.org/10.48550/arXiv.2409.08861). arXiv: [2409.08861](https://arxiv.org/abs/2409.08861)
3082 [cs]. (Visited on 09/22/2025).

- 3085 Domingo-Enrich, Carles, Michal Drozdzal, Brian Karrer, and Ricky T. Q. Chen (2025).
3086 *Adjoint Matching: Fine-tuning Flow and Diffusion Generative Models with Memoryless*
3087 *Stochastic Optimal Control*. arXiv: [2409.08861](https://arxiv.org/abs/2409.08861).
- 3088 Domingo-Enrich, Carles, Jiequn Han, Brandon Amos, Joan Bruna, and Ricky T. Q. Chen
3089 (Oct. 2024). *Stochastic Optimal Control Matching*. DOI: [10.48550/arXiv.2312.02027](https://doi.org/10.48550/arXiv.2312.02027).
3090 arXiv: [2312.02027 \[math\]](https://arxiv.org/abs/2312.02027). (Visited on 09/22/2025).
- 3091 ECB Banking Supervision (2023). “2023 stress test of euro area banks”. In: *Available at*
3092 https://www.bankingsupervision.europa.eu/ecb/pub/pdf/ssm_Report_2023_Stress_Test~96bb5a3af8.en.pdf ECB.
- 3094 Einmahl, John H. J., Laurens de Haan, and Chen Zhou (2016). “Statistics of heteroscedastic
3095 extremes”. In: *Journal of the Royal Statistical Society: Series B* 78.1, pp. 31–51. DOI:
3096 [10.1111/rssb.12099](https://doi.org/10.1111/rssb.12099).
- 3097 Einmahl, John H. J., Andrea Krajina, and Johan Segers (2012). “An M-estimator for tail
3098 dependence in arbitrary dimensions”. In: *The Annals of Statistics* 40.3, pp. 1764–1793.
3099 DOI: [10.1214/12-AOS1023](https://doi.org/10.1214/12-AOS1023).
- 3100 Embrechts, P., C. Klüppelberg, and T. Mikosch (1997). *Modelling Extremal Events for*
3101 *Insurance and Finance*. Springer-Verlag, Berlin.
- 3102 Embrechts, Paul, Claudia Klüppelberg, and Thomas Mikosch (1997). *Modelling Extremal
3103 Events*. Berlin, Heidelberg: Springer. DOI: [10.1007/978-3-642-33483-2](https://doi.org/10.1007/978-3-642-33483-2).
- 3104 Esser, Patrick, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry
3105 Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, Dustin Podell, Tim
3106 Dockhorn, Zion English, Kyle Lacey, Alex Goodwin, Yannik Marek, and Robin Rom-
3107 bach (Mar. 2024). *Scaling Rectified Flow Transformers for High-Resolution Image Syn-
3108 thesis*. DOI: [10.48550/arXiv.2403.03206](https://arxiv.org/abs/2403.03206). arXiv: [2403.03206 \[cs\]](https://arxiv.org/abs/2403.03206). (Visited on
3109 09/22/2025).
- 3110 European Banking Authority (2014). “Guidelines on the revised common procedures and
3111 methodologies for the supervisory review and evaluation process (SREP) and super-
3112 visory 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>
3116 EBA/GL/2014/13.
- 3117 European Commission (2024a). URL: https://finance.ec.europa.eu/sustainable-finance/tools-and-standards/esg-rating-activities_en.
- 3118 — (2024b). URL: https://finance.ec.europa.eu/sustainable-finance/overview-sustainable-finance_en.
- 3121 Falk, M. (2019). *Multivariate Extreme Value Theory and D-Norms*. en. Springer Series in
3122 Operations Research and Financial Engineering. Cham: Springer International Publishing.
3123 ISBN: 978-3-030-03818-2. DOI: [10.1007/978-3-030-03819-9](https://doi.org/10.1007/978-3-030-03819-9). URL: <http://link.springer.com/10.1007/978-3-030-03819-9> (visited on 05/16/2023).
- 3125 Falk, M. and T. Fuller (2021). “New characterizations of multivariate Max-domain of
3126 attraction and D-Norms”. en. In: *Extremes* 24.4, pp. 849–879. ISSN: 1386-1999, 1572-

- 3127 915X. DOI: [10.1007/s10687-021-00416-4](https://doi.org/10.1007/s10687-021-00416-4). URL: <https://link.springer.com/10.1007/s10687-021-00416-4> (visited on 05/16/2023).
- 3128 Fan, Ying, Olivia Watkins, Yuqing Du, Hao Liu, Moonkyung Ryu, Craig Boutilier, Pieter
3129 Abbeel, Mohammad Ghavamzadeh, Kangwook Lee, and Kimin Lee (2023). *DPOK: Reinforcement Learning for Fine-tuning Text-to-Image Diffusion Models*. NeurIPS 2023.
3130 arXiv: [2305.16381](https://arxiv.org/abs/2305.16381).
- 3131 Feder, R. M., P. Berger, and G. Stein (2020). “Nonlinear 3D cosmic web simulation
3132 with heavy-tailed generative adversarial networks”. In: *Physical Review D*. 102.10,
3133 pp. 103504, 18.
- 3134 Fisher, R. A. and L. H. C. Tippett (1928). “Limiting forms of the frequency distribution
3135 of the largest or smallest member of a sample”. In: *Mathematical Proceedings of the
3136 Cambridge Philosophical Society* 24.2, pp. 180–190. DOI: [10.1017/S0305004100015681](https://doi.org/10.1017/S0305004100015681).
- 3137 Fougères, Anne-Laure and Cécile Mercadier (2013). “Multivariate extremes and max-linear
3138 models”. In: *ESAIM: Probability and Statistics* 17, pp. 109–123.
- 3139 Glasserman, P. (2013). *Monte carlo methods in financial engineering*. Stochastic modelling
3140 and applied probability. Springer New York. ISBN: 978-0-387-21617-1. URL: <https://books.google.fr/books?id=aeAlBQAAQBAJ>.
- 3141 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:
3142 <http://www.jstor.org/stable/1968974> (visited on 06/25/2024).
- 3143 Gobet, E. and G. Liu (2015). “Rare event simulation using reversible shaking transformations”. In: *SIAM Journal on Scientific Computing* 37.5, A2295–A2316.
- 3144 Goodfellow, I., J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville,
3145 and Y. Bengio (2014). “Generative Adversarial Nets”. In: *Advances in Neural Information
3146 Processing Systems*. Ed. by Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence,
3147 and K.Q. Weinberger. Vol. 27, pp. 2672–2680.
- 3148 Goodfellow, Ian J., Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley,
3149 Sherjil Ozair, Aaron Courville, and Yoshua Bengio (2014). “Generative Adversarial
3150 Nets”. In: *Advances in Neural Information Processing Systems*. Ed. by Z. Ghahramani,
3151 M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger. Vol. 27. Curran Associates,
3152 Inc.
- 3153 Graf, S. and H. Luschgy (2000). *Foundations of quantization for probability distributions*.
3154 en. Lecture Notes in Mathematics 1730. Berlin; New York: Springer. ISBN: 978-3-540-
3155 67394-1.
- 3156 Grover, Aditya, Eric Wang, Aaron Zweig, and Stefano Ermon (2019). “Stochastic Optimiza-
3157 tion of Sorting Networks via Continuous Relaxations”. In: *International Conference on
3158 Learning Representations*.
- 3159 Gulrajani, Ishaan, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville
3160 (2017). “Improved Training of Wasserstein GANs”. In: *Advances in Neural Information
3161 Processing Systems*. Vol. 30. Curran Associates, Inc.
- 3162 Gumbel, Emil J. (1960). “Bivariate exponential distributions”. In: *Journal of the American
3163 Statistical Association* 55.292, pp. 698–707. DOI: [10.1080/01621459.1960.10483368](https://doi.org/10.1080/01621459.1960.10483368).
- 3164
- 3165
- 3166
- 3167
- 3168

- 3169 Haan, L. de and Ana Ferreira (2006). *Extreme value theory: an introduction*. en. Springer
3170 Series in Operations Research. New York ; London: Springer. ISBN: 978-0-387-23946-0.
- 3171 Haan, Laurens de and Ana Ferreira (2006). *Extreme Value Theory: An Introduction*. Springer
3172 Series in Operations Research. New York: Springer. ISBN: 978-0-387-23946-0.
- 3173 Haario, Heikki, Eero Saksman, and Johanna Tammisen (Sept. 1999). “Adaptive proposal
3174 distribution for random walk Metropolis algorithm”. In: *Computational Statistics* 14.3,
3175 pp. 375–395. ISSN: 0943-4062, 1613-9658. DOI: [10.1007/s001800050022](https://doi.org/10.1007/s001800050022). (Visited on
3176 07/08/2025).
- 3177 Hasan, A., K. Elkhilil, Y. Ng, J.M. Pereira, S. Farsiu, J. Blanchet, and V. Tarokh (2022).
3178 “Modeling extremes with d -max-decreasing neural networks”. In: *Uncertainty in Arti-
3179 ficial Intelligence*. PMLR, pp. 759–768.
- 3180 Hastings, W. K. (1970). “Monte Carlo Sampling Methods Using Markov Chains and Their
3181 Applications”. In: *Biometrika* 57.1, pp. 97–109. DOI: [10.1093/biomet/57.1.97](https://doi.org/10.1093/biomet/57.1.97).
- 3182 Hickling, T. and D. Prangle (2024). *Flexible Tails for Normalizing Flows*. en. arXiv:2406.16971.
3183 URL: <http://arxiv.org/abs/2406.16971> (visited on 06/27/2024).
- 3184 Ho, Jonathan, Ajay Jain, and Pieter Abbeel (2020). *Denoising Diffusion Probabilistic Mod-
3185 els*. arXiv: [2006.11239 \[cs.LG\]](https://arxiv.org/abs/2006.11239).
- 3186 Ho, Jonathan and Tim Salimans (July 2022). *Classifier-Free Diffusion Guidance*. DOI:
3187 [10.48550/arXiv.2207.12598 \[cs\]](https://doi.org/10.48550/arXiv.2207.12598). arXiv: [2207.12598 \[cs\]](https://arxiv.org/abs/2207.12598). (Visited on 09/22/2025).
- 3188 Hofert, M., R. Huser, and A. Prasad (2018). “Hierarchical archimax copulas”. In: *Journal
3189 of Multivariate Analysis* 167, pp. 195–211.
- 3190 Hornik, Kurt, Maxwell Stinchcombe, and Halbert White (1989). “Multilayer feedforward
3191 networks are universal approximators”. In: *Neural Networks* 2.5, pp. 359–366. DOI:
3192 [10.1016/0893-6080\(89\)90020-8](https://doi.org/10.1016/0893-6080(89)90020-8).
- 3193 Hu, Edward J., Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang,
3194 Lu Wang, and Weizhu Chen (Oct. 2021). *LoRA: Low-Rank Adaptation of Large Lan-
3195 guage Models*. DOI: [10.48550/arXiv.2106.09685 \[cs\]](https://doi.org/10.48550/arXiv.2106.09685). (Visited
3196 on 09/22/2025).
- 3197 Hüsler, Jürg and Rolf-Dieter Reiss (1989a). “Maxima of normal random vectors: be-
3198 tween independence and complete dependence”. In: *Statistics & Probability Letters*
3199 7.4, pp. 283–286. DOI: [10.1016/0167-7152\(89\)90106-5](https://doi.org/10.1016/0167-7152(89)90106-5).
- 3200 — (1989b). “Maxima of normal random vectors: between independence and complete de-
3201 pendence”. In: *Statistics & Probability Letters* 7.4, pp. 283–286.
- 3202 Huster, Todd, Jeremy Cohen, Zinan Lin, Kevin Chan, Charles Kamhoua, Nandi Kundu,
3203 Kevin Kwiat, and Noel Leslie (2021). *Pareto GAN: Extending the Representational
3204 Power of GANs to Heavy-Tailed Distributions*. arXiv: [2101.09113](https://arxiv.org/abs/2101.09113).
- 3205 Hyvärinen, Aapo (2005). “Estimation of Non-Normalized Statistical Models by Score Match-
3206 ing”. In: *Journal of Machine Learning Research* 6.24, pp. 695–709.
- 3207 JanSSen, Anja and Phyllis Wan (2020). “K-means clustering of extremes”. In: *Electronic
3208 Journal of Statistics* 14.1, pp. 1211–1233.
- 3209 Joe, H. (2014). *Dependence modeling with copulas*. CRC press.

- 3210 Kallenberg, O. (2002). *Foundations of modern probability*. Probability and its applications.
3211 Springer New York. ISBN: 978-0-387-95313-7. URL: <https://books.google.fr/books?id=L6fhXh130yMC>.
- 3212 Kim, S.H. and H.A. David (Mar. 1990). “On the dependence structure of order statistics
3213 and concomitants of order statistics”. In: *Journal of Statistical Planning and Inference*
3214 24.3, pp. 363–368. ISSN: 0378-3758. DOI: [10.1016/0378-3758\(90\)90055-y](https://doi.org/10.1016/0378-3758(90)90055-y). (Visited
3215 on 07/21/2025).
- 3216 Kingma, D. P. and J. Ba (2014). *Adam: A Method for Stochastic Optimization*. arXiv:1412.6980.
- 3217 Kingma, D. P. and M. Welling (2014). “Auto-Encoding Variational Bayes”. In: *2nd Inter-*
3218 *national Conference on Learning Representation, ICLR*.
- 3219 Kingma, Diederik P. and Max Welling (2014). *Auto-Encoding Variational Bayes*. arXiv:
3220 [1312.6114](https://arxiv.org/abs/1312.6114).
- 3221 Lafon, Marco, Philippe Naveau, and Anthony Leblois (2023). “On the use of variational
3222 autoencoders for extreme values modelling”. In: *arXiv preprint arXiv:2303.11536*.
- 3223 Leshno, Moshe, Vladimir Ya. Lin, Allan Pinkus, and Shimon Schocken (1993). “Multilayer
3224 feedforward networks with a nonpolynomial activation function can approximate any
3225 function”. In: *Neural Networks* 6.6, pp. 861–867. DOI: [10.1016/S0893-6080\(05\)80131-5](https://doi.org/10.1016/S0893-6080(05)80131-5).
- 3226 Liang, F., M. Mahoney, and L. Hodgkinson (2022). “FatTailed Variational Inference with
3227 Anisotropic Tail Adaptive Flows”. en. In: *Proceedings of the 39th International Con-*
3228 *ference on Machine Learning*. PMLR, pp. 13257–13270. URL: <https://proceedings.mlr.press/v162/liang22a.html> (visited on 01/11/2024).
- 3229 Lipman, Yaron, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le (Feb.
3230 2023). *Flow Matching for Generative Modeling*. DOI: [10.48550/arXiv.2210.02747](https://doi.org/10.48550/arXiv.2210.02747).
3231 arXiv: [2210.02747 \[cs\]](https://arxiv.org/abs/2210.02747). (Visited on 09/22/2025).
- 3232 Liu, Tie-Yan (2009). *Learning to Rank for Information Retrieval*. Berlin, Heidelberg: Springer.
3233 DOI: [10.1007/978-3-642-14267-3](https://doi.org/10.1007/978-3-642-14267-3).
- 3234 Lo, Andrew W. and Ruixun Zhang (2021). “Quantifying the Impact of Impact Investing”.
3235 In: *Management Science*. DOI: [10.2139/ssrn.3944367](https://doi.org/10.2139/ssrn.3944367).
- 3236 Maas, A.L., A.Y. Hannun, and A.Y. Ng (2013). “Rectifier Nonlinearities Improve Neural
3237 Network Acoustic Models”. en. In: *Proceedings of the International Conference on Ma-*
3238 *chine Learning (ICML)*.
- 3239 Mainik, Georg, Georgi Mitov, and Ludger Rüschendorf (2015). “Portfolio optimization
3240 for heavy-tailed assets: Extreme Risk Index vs. Markowitz”. In: *Journal of Empirical*
3241 *Finance* 32, pp. 115–134.
- 3242 McNeil, A.J., R. Frey, and P. Embrechts (2015). *Quantitative risk management*. Princeton
3243 Series in Finance. Princeton University Press, Princeton, NJ, pp. xix+699. ISBN: 978-
3244 0-691-16627-8.
- 3245 McNeil, Alexander J., Rüdiger Frey, and Paul Embrechts (2015). *Quantitative Risk Man-*
3246 *agement: Concepts, Techniques and Tools*. Revised. Princeton Series in Finance. Prince-
3247 ton, NJ: Princeton University Press.
- 3248

- 3251 Metropolis, Nicholas, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller,
3252 and Edward Teller (1953). “Equation of State Calculations by Fast Computing Ma-
3253 chines”. In: *The Journal of Chemical Physics* 21.6, pp. 1087–1092. DOI: [10.1063/1.1699114](https://doi.org/10.1063/1.1699114).
- 3255 Murphy, K. P. (2022). *Probabilistic Machine Learning: An introduction*. MIT Press. URL:
3256 probml.ai.
- 3257 Nadjahi, K. (Nov. 2021). “Sliced-Wasserstein distance for large-scale machine learning
3258 : theory, methodology and extensions”. PhD thesis. Institut Polytechnique de Paris.
3259 URL: <https://theses.hal.science/tel-03533097>.
- 3260 Nelsen, R.B. (2006). *An introduction to copulas*. en. second. Springer series in statistics.
3261 New York Berlin Heidelberg: Springer. ISBN: 978-0-387-28659-4.
- 3262 Nichol, Alex and Prafulla Dhariwal (Feb. 2021). *Improved Denoising Diffusion Probabilistic
3263 Models*. DOI: [10.48550/arXiv.2102.09672](https://doi.org/10.48550/arXiv.2102.09672). arXiv: [2102.09672 \[cs\]](https://arxiv.org/abs/2102.09672). (Visited on
3264 10/27/2025).
- 3265 Nikoloulopoulos, Aristidis K., Harry Joe, and Haijun Li (2009). “Extreme value properties
3266 of multivariate t copulas”. In: *Extremes* 12.2, pp. 129–148. DOI: [10.1007/s10687-008-0072-4](https://doi.org/10.1007/s10687-008-0072-4).
- 3268 Ouyang, Long, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin,
3269 Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. (2022). *Train-
3270 ing language models to follow instructions with human feedback*. NeurIPS 2022. arXiv:
3271 [2203.02155](https://arxiv.org/abs/2203.02155).
- 3272 Papamakarios, G., E. Nalisnick, D. J. Rezende, S. Mohamed, and B. Lakshminarayanan
3273 (2021). “Normalizing flows for probabilistic modeling and inference”. In: *The Journal
3274 of Machine Learning Research* 22.1, pp. 2617–2680.
- 3275 Prandini, M. and O. J. Watkins (2005). “Probabilistic Aircraft Conflict Detection”. In:
3276 *HYBRIDGE WP3: Reachability analysis for probabilistic hybrid systems*.
- 3277 Resnick, S. I. (1987). *Extreme Values, Regular Variation and Point Processes*. en. Springer
3278 Series in Operations Research and Financial Engineering. New York, NY: Springer.
3279 ISBN: 978-0-387-75952-4. DOI: [10.1007/978-0-387-75953-1](https://doi.org/10.1007/978-0-387-75953-1). URL: <http://link.springer.com/10.1007/978-0-387-75953-1> (visited on 05/16/2023).
- 3281 Resnick, Sidney I. (1987). *Extreme Values, Regular Variation and Point Processes*. Springer
3282 Series in Operations Research and Financial Engineering. New York: Springer. DOI:
3283 [10.1007/978-0-387-75953-1](https://doi.org/10.1007/978-0-387-75953-1).
- 3284 Rezende, Danilo Jimenez and Shakir Mohamed (2015). “Variational Inference with Nor-
3285 malizing Flows”. In: *Proceedings of the 32nd International Conference on Machine
3286 Learning*. PMLR, pp. 1530–1538.
- 3287 Robert, Christian P. and George Casella (2004). *Monte Carlo Statistical Methods*. 2nd.
3288 New York: Springer. DOI: [10.1007/978-1-4757-4145-2](https://doi.org/10.1007/978-1-4757-4145-2).
- 3289 Robert, P. (2003). *Stochastic networks and queues*. Vol. 52. Applications of Mathematics.
3290 Springer-Verlag, Berlin.

- 3291 Rosenblatt, Murray (Sept. 1952). “Remarks on a Multivariate Transformation”. In: *The*
3292 *Annals of Mathematical Statistics* 23.3, pp. 470–472. ISSN: 0003-4851. DOI: [10.1214/aoms/1177729394](https://doi.org/10.1214/aoms/1177729394). (Visited on 07/21/2025).
- 3294 Sabourin, Anne, Philippe Naveau, and Anne-Laure Fougères (2013). “Bayesian Dirichlet
3295 mixture model for multivariate extremes: A re-parametrization”. In: *Computational*
3296 *Statistics & Data Analysis* 71, pp. 542–567. DOI: [10.1016/j.csda.2013.04.021](https://doi.org/10.1016/j.csda.2013.04.021).
- 3297 Schucany, William R. (1972). “Order statistics in simulation”. In: *Journal of Statistical*
3298 *Computation and Simulation* 1.3, pp. 281–286. DOI: [10.1080/00949657208810024](https://doi.org/10.1080/00949657208810024).
- 3299 Scott, David W. (2015). *Multivariate Density Estimation: Theory, Practice, and Visualiza-*
3300 *tion*. 2nd. Hoboken, NJ: Wiley. DOI: [10.1002/9781118575574](https://doi.org/10.1002/9781118575574).
- 3301 Silverman, Bernard W. (1986). *Density Estimation for Statistics and Data Analysis*. Lon-
3302 don: Chapman and Hall. DOI: [10.1007/978-1-4899-3324-9](https://doi.org/10.1007/978-1-4899-3324-9).
- 3303 Sklar, M. (1959). “Fonctions de répartition à N dimensions et leurs marges”. fr. In: *Pub-*
3304 *lications de lInstitut Statistique de lUniversité de Paris* 8, pp. 229–231.
- 3305 Sohl-Dickstein, J., E. Weiss, N. Maheswaranathan, and S. Ganguli (2015). “Deep unsuper-
3306 *vised learning using nonequilibrium thermodynamics*”. In: *International conference on*
3307 *machine learning*. PMLR, pp. 2256–2265.
- 3308 Sohl-Dickstein, Jascha, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli (Nov.
3309 2015). *Deep Unsupervised Learning using Nonequilibrium Thermodynamics*. DOI: [10.48550/arXiv.1503.03585](https://doi.org/10.48550/arXiv.1503.03585) [cs]. (Visited on 09/22/2025).
- 3311 Song, Jiaming, Chenlin Meng, and Stefano Ermon (2021). “Denoising Diffusion Implicit
3312 *Models*”. In: *International Conference on Learning Representations*.
- 3313 Song, Yang and Stefano Ermon (2019). “Generative Modeling by Estimating Gradients of
3314 the Data Distribution”. In: *Advances in Neural Information Processing Systems*. Ed. by
3315 H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett.
3316 Vol. 32. Curran Associates, Inc.
- 3317 Song, Yang, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon,
3318 and Ben Poole (2021). “Score-Based Generative Modeling through Stochastic Differen-
3319 *tial Equations*”. In: *International Conference on Learning Representations*.
- 3320 Sukhatme, Pandurang V. (1937). “Tests of significance for samples of the χ^2 -population
3321 with two degrees of freedom”. In: *Annals of Eugenics* 8.1, pp. 52–56.
- 3322 Tancik, Matthew, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Ragha-
3323 van, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng (June
3324 2020). *Fourier Features Let Networks Learn High Frequency Functions in Low Dimen-*
3325 *sional Domains*. DOI: [10.48550/arXiv.2006.10739](https://doi.org/10.48550/arXiv.2006.10739) [cs]. (Visited
3326 on 10/27/2025).
- 3327 Tang, Wenpin (Mar. 2024). *Fine-tuning of diffusion models via stochastic control: entropy*
3328 *regularization and beyond*. DOI: [10.48550/arXiv.2403.06279](https://doi.org/10.48550/arXiv.2403.06279). arXiv: [2403.06279](https://arxiv.org/abs/2403.06279)
3329 [math]. (Visited on 09/22/2025).
- 3330 Uehara, Masatoshi, Yulai Zhao, Kevin Black, Ehsan Hajiramezanali, Gabriele Scalia, Nathaniel
3331 Lee Diamant, Alex M. Tseng, Tommaso Biancalani, and Sergey Levine (Feb. 2024).

- 3332 *Fine-Tuning of Continuous-Time Diffusion Models as Entropy-Regularized Control.*
3333 DOI: [10.48550/arXiv.2402.15194](https://doi.org/10.48550/arXiv.2402.15194). arXiv: [2402.15194 \[cs\]](https://arxiv.org/abs/2402.15194). (Visited on 09/22/2025).
- 3334 Villani, C. (2009). *Optimal transport: old and new*. Vol. 338. Springer.
- 3335 Vincent, Pascal (2011). “A Connection Between Score Matching and Denoising Autoen-
3336 coders”. In: *Neural Computation* 23.7, pp. 1661–1674. DOI: [10.1162/NECO_a_00142](https://doi.org/10.1162/NECO_a_00142).
- 3337 Wei, Xiang, Zixia Liu, Liqiang Wang, and Boqing Gong (2018). “Improving the improved
3338 training of wasserstein GANs”. In: *International conference on learning representa-*
3339 *tions*. Comment: NIPS camera-ready. URL: <https://openreview.net/forum?id=SJx9GQb0->
- 3340
- 3341 Wiese, Magnus, Robert Knobloch, Ralf Korn, and Peter Kretschmer (2019). *Deep Hedging:*
3342 *Learning to Simulate Equity Option Markets*. arXiv: [1911.01700](https://arxiv.org/abs/1911.01700).
- 3343 — (2020). “Quant GANs: Deep generation of financial time series”. In: *Quantitative Fi-*
3344 *nance* 20.9, pp. 1419–1440.
- 3345 Wu, J., X.-Y. Chen, H. Zhang, L.-D. Xiong, H. Lei, and S.-H. Deng (2019). “Hyperparam-
3346 eter Optimization for Machine Learning Models Based on Bayesian Optimization”. en.
3347 In: 17.1, pp. 26–40.
- 3348 Xu, Jiazheng, Xiao Liu, Yuchen Wu, Yuxuan Tong, Qinkai Li, Ming Ding, Jie Tang, and
3349 Yuxiao Dong (Dec. 2023). *ImageReward: Learning and Evaluating Human Preferences*
3350 *for Text-to-Image Generation*. DOI: [10.48550/arXiv.2304.05977](https://doi.org/10.48550/arXiv.2304.05977). arXiv: [2304.05977](https://arxiv.org/abs/2304.05977)
3351 [cs]. (Visited on 04/28/2025).
- 3352 Xu, Jing, Andrew Liu, Andrea Wu, Parisa Saeedi, Jennifer Neville, Mona Diab, and Asli
3353 Celikyilmaz (2024). *ReFL: Reflective Feedback Learning for Language Model Alignment*.
3354 arXiv: [2403.01673](https://arxiv.org/abs/2403.01673).
- 3355 Yarotsky, Dmitry (2017). “Error bounds for approximations with deep ReLU networks”.
3356 In: *Neural Networks* 94, pp. 103–114. DOI: [10.1016/j.neunet.2017.07.002](https://doi.org/10.1016/j.neunet.2017.07.002).
- 3357 Yoon, Sungbin, Juho Park, Jaehyeon Lee, and Gunhee Kim (2023). *Score-based Genera-*
3358 *tive Modeling through Stochastic Evolution Equations in Hilbert Spaces*. Heavy-tailed
3359 diffusion models using Lévy processes. arXiv: [2305.19241](https://arxiv.org/abs/2305.19241).
- 3360 Zhang, Likun, Andrew Bray, and Joseph Guinness (2024). “Flexible and efficient spatial
3361 extremes emulation via variational autoencoders”. In: *arXiv preprint arXiv:2407.02868*.
- 3362 Zscheischler, J., O. Martius, S. Westra, E. Bevacqua, C. Raymond, R. M. Horton, B. van
3363 den Hurk, A. AghaKouchak, A. Jézéquel, M. D. Mahecha, D. Maraun, A. M. Ramos,
3364 N. N. Ridder, W. Thiery, and E. Vignotto (2020). “A typology of compound weather
3365 and climate events”. In: *Nature Reviews Earth & Environment* 1.7, pp. 333–347.
- 3366 Zscheischler, Jakob, Seth Westra, Bart J. J. M. van den Hurk, Sonia I. Seneviratne, Philip J.
3367 Ward, Andy Pitman, Amir AghaKouchak, David N. Bresch, Michael Leonard, Thomas
3368 Wahl, and Xuebin Zhang (2018). “Future climate risk from compound events”. In:
3369 *Nature Climate Change* 8, pp. 469–477. DOI: [10.1038/s41558-018-0156-3](https://doi.org/10.1038/s41558-018-0156-3).

³³⁷⁰ **Supplementary Material**

³³⁷¹ **.1 Additional Figures**

³³⁷² **.2 Additional Tables**

³³⁷³ **.3 Research Instruments**