On the simulation of extreme events with neural networks

Michaël Allouche, Stéphane Girard, Emmanuel Gobet

▶ **To cite this version:**

Michaël Allouche, Stéphane Girard, Emmanuel Gobet. On the simulation of extreme events with neural networks. M. de Carvalho, R. Huser, P. Naveau, and B. J. Reich. Handbook on Statistics of Extremes, Chapman & Hall/CRC, inPress. ⟨hal-04416809v2⟩

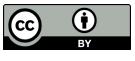
HAL Id: hal-04416809

https://inria.hal.science/hal-04416809v2

Submitted on 30 Aug 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Distributed under a Creative Commons CC BY 4.0 - Attribution - International License

On the simulation of extreme events with neural networks

Michaël Allouche[(1)], Stéphane Girard[(2,⋆)] and Emmanuel Gobet[(3)]

[(1)] Kaiko - Quantitative Data. 2 rue de Choiseul 75002 Paris, France.
[(2)] Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP, LJK, 38000 Grenoble, France.
[(⋆)] Corresponding author: Stephane.Girard@inria.fr
[(3)] Centre de Mathématiques Appliquées (CMAP), CNRS, Ecole Polytechnique, Institut Polytechnique de Paris, 91128 Palaiseau Cedex, France.

1 Introduction

This work aims at investigating the use of generative methods based on neural networks to simulate extreme events. Although very popular, these methods are mainly invoked in empirical works. Therefore, providing theoretical guidelines for using such models in an extreme-value context is of primary importance. To this end, we overview the most recent generative methods dedicated to extremes, giving some theoretical results and practical advice on their performance for sampling tails, thanks to both extreme-value and copula tools. We begin by recalling the basic principles of generative modelling in Section 2 focusing on the three most popular methods: Generative Adversarial Networks (GANs), Variational Auto-Encoders (VAEs) and Diffusion models. The limitations of classical GANs for dealing with extreme events are highlighted in Section 3 on both simulated and real financial data. Section 4 describes the extreme-value framework used in Section 5 to interpret the new generative methods proposed in the literature dedicated to extremes. Some of these improvements are illustrated on simulated multivariate data. Extra references, comments and future directions are exposed in Section 6. Finally, Appendix 7 provides a short description of copula tools to assess dependence between sampled points.

2 Generative Modeling

Generative models aim at mimicking the distribution of a random object, taking values in a possibly large dimensional space. These models are referred to as digital twins in industrial engineering, see [42, 46] for reviews. Generative models are particularly useful for instance in the field of data-augmentation; enriching a dataset may reduce overfitting and improve the performance of statistical models. Another perspective of high interest is data privacy, sharing generated data

with the same statistical properties as confidential data. In the context of generative modeling, two different point of views may be distinguished. On the one hand, sampling complex motions of physical objects was originally done by solving dynamical equations describing the objects behavior under constraints and by generating trajectories given different initial conditions. Such an approach requires prior knowledge of the exact evolution formulae or building the physical model by hand using mathematical equations. The latter are referred to as physics-based models. On the other hand, a new class of data-based generative models has recently emerged in the paradigm of artificial intelligence. Instead of constructing a physics-based model, one can learn it directly from the data using random noise as input (data-driven model). Such algorithms have the advantage of being fast in the simulation phase compared with their physical model counterparts, even though they may be slow in the inference phase. Thanks to the numerical and theoretical advances in the 21st century, neural networks have proven to be excellent candidates as they are universal approximation functions. Among the neural network generative models developed so far [23], the most popular have been the Variational Auto-Encoder (VAE) [38], based on variational inference, and the Generative Adversarial Network (GAN) [29], on which we shall focus, based on a min-max game. More recent models such as Normalizing flows [50] and Diffusion models [34, 52] have gained some popularity. Nowadays, both the construction and the optimization of complex neural network models are made easy by open-source libraries (e.g., `TensorFlow` [1] or `PyTorch` [48]), resulting in extraordinary interest from people from different communities and various mathematical backgrounds.

2.1 Theoretical Framework

Given observations $\{\mathbf{x}_i\}_{i=1}^n$ supposedly drawn independently from an unknown distribution $p_{\mathbf{X}}$ on a measurable space $(\mathcal{X}, \mathcal{B}_{\mathcal{X}})$, the goal of generative modeling is to find a function $G : \mathcal{Z} \to \mathcal{X}$, called *generator*, and a probability distribution $p_{\mathbf{Z}}$, called *latent distribution* and defined on some set $\mathcal{Z}$, such that

$$G(\mathbf{Z}) \overset{\mathrm{d}}{=} \mathbf{X}, \quad \mathbf{Z} \sim p_{\mathbf{Z}}. \tag{1}$$

Note that in contrast with usual statistical methods, we do not seek to infer the unknown distribution $p_{\mathbf{X}}$ from the data. The existence of G and $p_{\mathbf{Z}}$ is provided by Kuratowski's Theorem.

Theorem 2.1 (Kuratowski)**.** *Let $(\mathcal{Z}, \mu_{\mathbf{Z}})$ and $(\mathcal{X}, \mu_{\mathbf{X}})$ be two Polish probability spaces. Then, there exists a (non-unique) measurable bijection $G : \mathcal{Z} \to \mathcal{X}$ such that $\mu_{\mathbf{Z}}\{G^{-1}(E)\} = \mu_{\mathbf{X}}(E)$ and $\mu_{\mathbf{X}}\{G(F)\} = \mu_{\mathbf{Z}}(F)$, for all Borel sets $E \subset \mathcal{X}$ and $F \subset \mathcal{Z}$.*

In the following, $\mathbf{X}$ is D-dimensional, but in principle, it could be infinite-dimensional as well (in [4], $\mathbf{X}$ is the path of a fractional Brownian motion). Here, we focus on a parametric family of generators $\{G_{\boldsymbol{\theta}}\}_{\boldsymbol{\theta}\in\boldsymbol{\Theta}}$ where $\boldsymbol{\Theta} \subset \mathbb{R}^p$ and $\{p_{\boldsymbol{\theta}}\}_{\boldsymbol{\theta}\in\boldsymbol{\Theta}}$ denotes the set of associated parametric distributions such that

$G_{\boldsymbol{\theta}}(Z) \sim p_{\boldsymbol{\theta}}$. The problem comes in finding the "best" parameter $\boldsymbol{\theta}^\star$ such that $p_{\boldsymbol{\theta}^\star}$ and $p_{\mathbf{X}}$ are as close as possible, or equivalently

$$G_{\boldsymbol{\theta}^\star}(\mathbf{Z}) \overset{\mathrm{d}}{\approx} \mathbf{X}, \tag{2}$$

for a given $\mathbf{Z} \sim p_{\mathbf{Z}}$. A generative modeling framework mainly consists of combining three ingredients:

1. The observations $\mathbf{x}_1, \ldots, \mathbf{x}_n$ with their underlying properties, one usually does not have choice of this data set, the goal is just to enrich it;
2. The parametrization $G_{\boldsymbol{\theta}}$ and the latent distribution $p_{\mathbf{Z}}$ to use as inputs;
3. The distance or the similarity criterion between the probability distributions $p_{\boldsymbol{\theta}}$ and $p_{\mathbf{X}}$, as well as the optimization process that defines the optimal $\boldsymbol{\theta}$.

Observe that, in light of Kuratowski's theorem, one has theoretically a large flexibility in the choice of the latent distribution $p_{\mathbf{Z}}$. Thus, in practice easy-to-simulate distributions are considered (Gaussian, uniform, etc). As a consequence, the modelling effort is usually put on the generator. In the paradigm of artificial intelligence, it is natural to consider a neural network parametrization of $G_{\boldsymbol{\theta}}$.

2.2 Neural Networks

A neural network is a non-linear function built with a fixed number of neurons, each one representing a function, and distributed across several hidden layers.Neurons are scaled and translated in the network by parameters called, respectively, weights and biases. Among many different existing neural network architectures, we consider the classical one-hidden layer feedforward neural network $G_{\boldsymbol{\theta}} : \mathbb{R}^q \to \mathbb{R}$ composed of K neurons

$$G_{\boldsymbol{\theta}}(\mathbf{z}) = b^{(2)} + \sum_{k=1}^{K} w_k^{(2)} \sigma(\langle \mathbf{w}_k^{(1)}, \mathbf{z} \rangle + b_k^{(1)}), \tag{3}$$

with parameters

$$\boldsymbol{\theta} := \{b_2\} \cup \{\mathbf{w}_k^{(1)}, w_k^{(2)}, b_k^{(1)}\}_{k=1}^K \in \Theta := \mathbb{R} \times (\mathbb{R}^q \times \mathbb{R} \times \mathbb{R})^K,$$

and where $\langle \cdot, \cdot \rangle$ is a scalar product on $\mathbb{R}^q$, and $\sigma : \mathbb{R} \to \mathbb{R}$ is a non-linear "activation" function. In the context of generative modelling, the input dimension q is often the dimension of the latent space in which $\mathbf{Z}$ is sampled; it is one of the hyperparameters. Note that the total number of parameters is $p = K(q+2)+1$. The generator can be easily extended to D dimensional outputs by replication of (3), that is by considering the Dth dimensional vector $(G_{\boldsymbol{\theta}}^{(1)}, \ldots, G_{\boldsymbol{\theta}}^{(D)})$. See Figure 1 for a schematic representation of a neural network with $K = 4$ neurons and $q = 3$ as the input dimension.

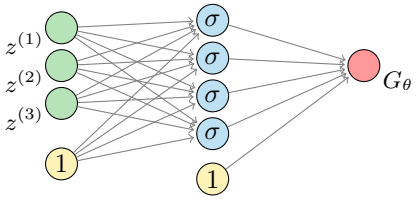


Figure 1: Example of a one-hidden layer neural network with $K = 4$ neurons and input dimension $q = 3$. The symbol σ represents the transformation with the activation function in (3), while the arrows stand for different parameters $\boldsymbol{\theta}$.

Examples of activation functions include:

- The cosine squasher

$$\sigma(x) = \frac{\cos(x + 3\pi/2) + 1}{2} \mathbb{I}\left(x \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right]\right) + \mathbb{I}\left(x \in \left(\frac{\pi}{2}, \infty\right)\right); \tag{4}$$

- The logistic squasher

$$\sigma(x) = \frac{1}{1 + \mathrm{e}^{-x}}; \tag{5}$$

- The exponential Linear Unit (eLU) defined for all $\alpha > 0$ by

$$\sigma_\alpha(x) = \alpha\{\exp(x) - 1\}\mathbb{I}(x < 0) + x\mathbb{I}(x \geq 0); \tag{6}$$

- The Rectified Linear Unit (ReLU)

$$\sigma(x) = \max(x, 0). \tag{7}$$

More generally, any increasing function σ such that $\sigma(x) \to 0$ as $x \to -\infty$ and $\sigma(x) \to 1$ as $x \to +\infty$ is called a "sigmoidal squashing" function. This definition includes (4) and (5) but excludes (6) and (7). In the following discussion, the activation function is assumed to be continuous.

Several authors studied the ability of $G_{\boldsymbol{\theta}}$, defined in (3), to approximate a given function G as $K \to \infty$, under various assumptions on G and with respect to different norms. Let us summarize some of them:

- If G is a square integrable function on $[0, 1]^q$, then [24] showed that there exists a "Fourier neural network" $G_{\boldsymbol{\theta}}$ designed with cosine squasher activation functions (4) which converges (in the L_2-sense) to G as $K \to \infty$.

- If G is a continuous function on the q-dimensional unit cube $[0, 1]^q$, then [16] proved that there exists a neural network $G_{\boldsymbol{\theta}}$, designed with sigmoidal squashing functions that converge uniformly to G as $K \to \infty$.

- The latter result related to the unit cube is extended in [35] to functions G continuous on any compact set of $\mathbb{R}^q$, and to bounded and nonconstant activation functions. Moreover, [44] proved that the uniform convergence holds on compact sets if and only if the activation functions are not polynomial.

Nowadays, the state-of-the-art result is the following.

Theorem 2.2 (Universal approximation theorem)**.** *Suppose G is a continuous function on a compact space $\mathcal{Z} \subset \mathbb{R}^q$ and σ is not a polynomial. Then, for any $\varepsilon > 0$, there exists a one-hidden layer neural network $G_{\boldsymbol{\theta}}$ (for some K depending on ε) defined in* (3) *such that*

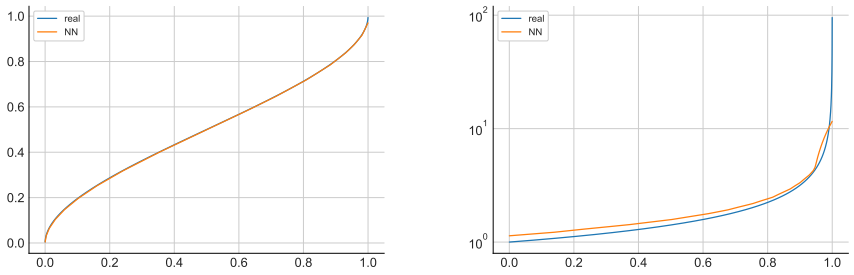
$$\sup_{\mathbf{z} \in \mathcal{Z}} |G(\mathbf{z}) - G_{\boldsymbol{\theta}}(\mathbf{z})| < \varepsilon.$$

The key assumption of the universal approximation theorem is that the function G to approximate is continuous on a compact set and thus bounded. This theoretical result is illustrated in Figure 2, where a one-hidden-layer ReLU neural network is trying to approximate a one-dimensional quantile on synthetic data. To this end, the problem is considered in a simplified regression framework where $(Z_i)_{i=1}^n$ are random samples from a one-dimensional uniform distribution and $X_i = G(Z_i)$ are the associated samples from the target (usually unknown) distribution. This method is related to the well-known inversion method [21] since $G = F_X^{-1}$ is the quantile function of the target distribution. Optimizing with respect to $\boldsymbol{\theta}$ the Mean Square Error $1/n \sum_{i=1}^n (G_{\boldsymbol{\theta}}(Z_i) - X_i)^2$, we get a simplified one-dimensional neural network generator with a one-dimensional uniform latent variable. On the first hand, considering a well bounded Beta quantile function as the theoretical result requires, the neural network does not have difficulty approximating any part of such a quantile function (see Figure 2(a)). On the other hand, the neural network fails to fit extreme quantiles in the right tail of a heavy-tailed Pareto variable (see Figure 2(b)). We shall see in Section 3 that the practical consequences remain the same for simulating heavy-tailed random variables in a generative modeling framework - which consists in comparing the distributions as a whole rather than doing it sample by sample (indeed, one can not observe the uniform noises Z_i in concrete situations). Before that, we briefly describe Generative Adversarial Networks (GANs), Variational Auto-Encoders (VAEs) and Diffusion models that are different parameterizations of $G_{\boldsymbol{\theta}}$ with adhoc optimization procedures; they are among the most popular methods in generative modelling and they can give accurate results when the conditions of Theorem 2.2 are satisfied.

2.3 Generative Adversarial Network

A GAN scheme aims at approximating the unknown generator $G : \mathbb{R}^q \to \mathbb{R}^D$ through a parametric family of neural networks

$$\{G_{\boldsymbol{\theta}} : \mathbb{R}^q \to \mathbb{R}^D\}_{\boldsymbol{\theta} \in \boldsymbol{\Theta}}, \tag{8}$$



(a) (b)

Figure 2: One-hidden layer neural network approximation of quantile functions $z \in [0,1) \mapsto F_X^{-1}(z)$ from (a) Beta ($\alpha = \beta = 2$) and (b) Pareto ($\gamma = 0.5$) distributions. The y axis of (b) is in log-scale.

and to estimate the optimal parameter $\boldsymbol{\theta}^\star$ from a data set $\{\mathbf{x}_i\}_{i=1}^n$ of independent samples from the unknown distribution p_X. The estimation is performed by optimizing an objective function which can be interpreted as an adversarial game between a generator in (8) and a *discriminator* chosen in a parametric family of functions

$$\{D_{\boldsymbol{\phi}} : \mathbb{R}^D \to [0,1]\}_{\boldsymbol{\phi} \in \boldsymbol{\Phi}},$$

where $\boldsymbol{\Phi}$ stands for the set in which the discriminator parameters take values. In other words, $D_{\boldsymbol{\phi}}(\mathbf{x})$ represents the probability that an observation $\mathbf{x} \in \mathbb{R}^D$ is drawn from $\mathbf{X} \sim p_{\mathbf{X}}$. Both the generator and the discriminator are neural networks with opposing objectives: The former tries to generate realistic data, while the latter tries to discriminate between synthetic generated data and real observations. See Figure 3 for an illustration. In [29], this optimization problem is defined as:

$$\arg\min_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \max_{\boldsymbol{\phi} \in \boldsymbol{\Phi}} \left(\mathrm{E}_{p_{\mathbf{X}}} \{\log D_{\boldsymbol{\phi}}(\mathbf{X})\} + \mathrm{E}_{p_{\mathbf{Z}}} \left\{\log \left[1 - D_{\boldsymbol{\phi}} \left\{G_{\boldsymbol{\theta}}(\mathbf{Z})\right\}\right]\right\} \right), \tag{9}$$

where $\mathbf{Z}$ denotes the latent variable, see (1). The goal of the discriminator is to maximize the criterion with respect to $\boldsymbol{\phi}$ which is achieved when both $D_{\boldsymbol{\phi}}(\mathbf{X}) \simeq 1$ (realistic outputs) and $D_{\boldsymbol{\phi}}\left\{G_{\boldsymbol{\theta}}(\mathbf{Z})\right\} \simeq 0$ (fake outputs). In contrast, the generator aims at minimizing the above criterion with respect to $\boldsymbol{\theta}$ in such a way that $G_{\boldsymbol{\theta}}(\mathbf{Z}) \overset{\mathrm{d}}{\approx} \mathbf{X}$ leading to $D_{\boldsymbol{\phi}}\left\{G_{\boldsymbol{\theta}}(\mathbf{Z})\right\} \simeq 1$ (increasing the realism of fake outputs). Statistical results on the estimators obtained by considering the empirical counterpart of the above optimization problem can be found in [11].

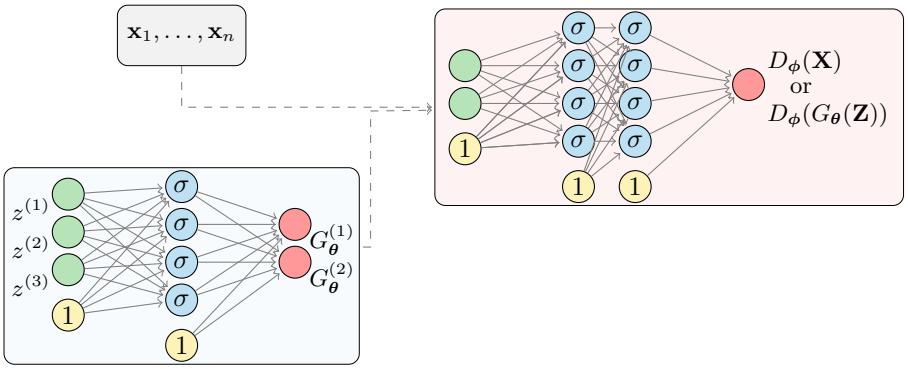


Figure 3: A GAN model with $q = 3$ (dimension of the latent noise) and $D = 2$ (dimension of data points). On the left: the observed data (top) and the synthetic data generated by a neural network $G_{\boldsymbol{\theta}}$ (bottom). On the right: the neural network G_{ϕ} discriminating true and fake data.

2.4 Variational Auto-Encoders (VAE)

The VAE approach differs by considering two parametric families of mapping functions on $\mathcal{X}$ and $\mathcal{Z}$. As we will see in the final generative equation (14), this still fits the theoretical framework of Section 2.1, but the learning is mixing two phases explained below. The *encoding set* is defined as

$$\{\mathcal{C}_{\boldsymbol{\phi}} : \mathbb{R}^D \to \mathbb{R}^q, \boldsymbol{\phi} \in \boldsymbol{\Phi}\}$$

and the *decoding set* is

$$\{\mathcal{D}_{\boldsymbol{\theta}} : \mathbb{R}^q \to \mathbb{R}^D, \boldsymbol{\theta} \in \boldsymbol{\Theta}\},$$

with their associated families of parametric densities $\{p^{\mathcal{C}_{\boldsymbol{\phi}}}\}_{\boldsymbol{\phi}\in\boldsymbol{\Phi}}$ and $\{p^{\mathcal{D}_{\boldsymbol{\theta}}}\}_{\boldsymbol{\theta}\in\boldsymbol{\Theta}}$. To achieve (1), the VAE setting aims at, for all $\mathbf{x} \in \mathcal{X}$:

1. maximizing the likelihood $p^{\mathcal{D}_{\boldsymbol{\theta}}}(\mathbf{x}) = \int p^{\mathcal{D}_{\boldsymbol{\theta}}}_{\mathbf{X}\,|\,\mathbf{Z}}(\mathbf{x}\,|\,\mathbf{z}) p_Z(\mathbf{z})\,\mathrm{d}\mathbf{z}$ induced by the distribution of $\mathbf{Z}$ (which is usually known) in the decompression phase, $\mathbf{Z}$ plays the role of latent random variable;

2. minimizing a distance or a divergence between $p^{\mathcal{C}_{\boldsymbol{\phi}}}_{\mathbf{Z}\,|\,\mathbf{X}=\mathbf{x}}$ and $p^{\mathcal{D}_{\boldsymbol{\theta}}}_{\mathbf{Z}\,|\,\mathbf{X}=\mathbf{x}}$.

Considering the KL-divergence

$$\mathrm{D}_{\mathrm{KL}}(p^{\mathcal{C}_{\boldsymbol{\phi}}}_{\mathbf{Z}\,|\,\mathbf{X}=\mathbf{x}}, p^{\mathcal{D}_{\boldsymbol{\theta}}}_{\mathbf{Z}\,|\,\mathbf{X}=\mathbf{x}}) = -\int_{\mathcal{Z}} p^{\mathcal{C}_{\boldsymbol{\phi}}}_{\mathbf{Z}\,|\,\mathbf{X}}(\mathbf{z}\,|\,\mathbf{x}) \log\left\{\frac{p^{\mathcal{D}_{\boldsymbol{\theta}}}_{\mathbf{Z}\,|\,\mathbf{X}}(\mathbf{z}\,|\,\mathbf{x})}{p^{\mathcal{C}_{\boldsymbol{\phi}}}_{\mathbf{Z}\,|\,\mathbf{X}}(\mathbf{z}\,|\,\mathbf{x})}\right\}\mathrm{d}\mathbf{z}$$

in the above second objective and using Bayes' rule, we get

$$\mathcal{G}(\mathbf{x}) := \log p^{\mathcal{D}_{\boldsymbol{\theta}}}(\mathbf{x}) - \mathrm{D}_{\mathrm{KL}}(p^{\mathcal{C}_{\boldsymbol{\phi}}}_{\mathbf{Z}\,|\,\mathbf{X}=\mathbf{x}}, p^{\mathcal{D}_{\boldsymbol{\theta}}}_{\mathbf{Z}\,|\,\mathbf{X}=\mathbf{x}}) \tag{10}$$

$$= \mathrm{E}_{p^{\mathcal{C}_{\boldsymbol{\phi}}}_{\mathbf{Z}\,|\,\mathbf{X}}}(\log p^{\mathcal{D}_{\boldsymbol{\theta}}}_{\mathbf{X}\,|\,\mathbf{Z}}(\mathbf{x}\mid\mathbf{Z})) - \mathrm{D}_{\mathrm{KL}}(p^{\mathcal{C}_{\boldsymbol{\phi}}}_{\mathbf{Z}\,|\,\mathbf{X}=\mathbf{x}}, p^{\mathcal{C}_{\boldsymbol{\phi}}}_{\mathbf{Z}}), \tag{11}$$

where $\mathcal{G}$ is called the Evidence Lower Bound.

It appears in (10) that $\mathcal{G}$ is indeed a lower bound for the log-likelihood $\log p^{\mathcal{D}_{\boldsymbol{\theta}}}(\mathbf{x})$, the error term $\mathrm{D}_{\mathrm{KL}}(p^{\mathcal{C}_{\boldsymbol{\phi}}}_{\mathbf{Z}\,|\,\mathbf{X}=\mathbf{x}}, p^{\mathcal{D}_{\boldsymbol{\theta}}}_{\mathbf{Z}\,|\,\mathbf{X}=\mathbf{x}})$ being small if $p^{\mathcal{C}_{\boldsymbol{\phi}}}_{\mathbf{Z}\,|\,\mathbf{X}=\mathbf{x}}$ is able to produce $\mathbf{z}$'s that can yield realistic $\mathbf{x}$ in the decoding phase. The second representation of $\mathcal{G}$ in (11) is used for the numerical algorithm. The optimization program is:

$$\max_{\boldsymbol{\theta},\boldsymbol{\phi}} \frac{1}{n}\sum_{i=1}^{n} \mathcal{G}(\mathbf{x}_i). \tag{12}$$

The optimization of (12) proposed in [38] mainly relies on Gaussian multivariate densities with diagonal covariance matrices:

$$\begin{aligned} \mathbf{Z} &\sim \mathrm{N}(\mathbf{0}, \mathbf{I}_q), \\ \mathbf{Z}\,|\,\mathbf{X}=\mathbf{x} &\sim \mathrm{N}(\boldsymbol{\mu}^{\mathcal{C}_{\boldsymbol{\phi}}}(\mathbf{x}), \mathrm{diag}\{\Sigma^{\mathcal{C}_{\boldsymbol{\phi}}}(\mathbf{x})\}) \\ &\overset{\mathrm{d}}{=} \boldsymbol{\mu}^{\mathcal{C}_{\boldsymbol{\phi}}}(\mathbf{x}) + [\mathrm{diag}\{\Sigma^{\mathcal{C}_{\boldsymbol{\phi}}}(\mathbf{x})\}]^{1/2}\mathrm{N}(\mathbf{0}, \mathbf{I}_q), \end{aligned} \tag{13}$$

where $\{\boldsymbol{\mu}^{\mathcal{C}_{\boldsymbol{\phi}}}(\mathbf{x}), \Sigma^{\mathcal{C}_{\boldsymbol{\phi}}}(\mathbf{x})\}_{\boldsymbol{\phi}\in\boldsymbol{\Phi}}$ are parametrized by neural networks in order to have a closed form of the KL-divergence and a tractable gradient in (11). The parameterization of $p^{\mathcal{D}_{\boldsymbol{\theta}}}_{\mathbf{X}\,|\,\mathbf{Z}}$ is either Gaussian or Bernoulli, depending on whether the data are, respectively, continuous or discrete. See Figure 4 for an illustration.

The output of this VAE methodology (once optimized) is simply the generative model

$$\mathcal{D}_{\boldsymbol{\theta}^*}(\mathbf{Z}) \overset{\mathrm{d}}{\approx} \mathbf{X}, \qquad \text{with} \qquad \mathbf{Z} \sim \mathrm{N}(\mathbf{0}, \mathbf{I}_q). \tag{14}$$

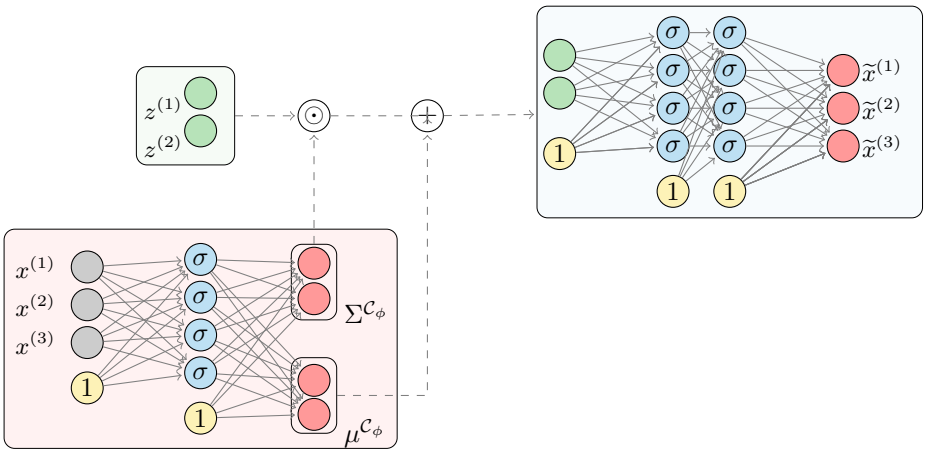


Figure 4: A VAE model with $q = 2$ (dimension of the latent space) and $D = 3$ (dimension of data points), using Gaussian encoding (with parameters $\boldsymbol{\mu}^{\mathcal{C}_\phi}$ and $\Sigma^{\mathcal{C}_\phi}$ as explained in (13)). The operator $\odot$ denotes the component wise product and $\widetilde{x}^{(j)}$ corresponds to the jth component of the generated data vector.

2.5 Diffusion Models

Generative diffusion models can be viewed as a decomposition of the Encoder and Decoder stages of VAE using infinitely many infinitesimal transformations, that are performed by running a continuous-time *forward* and *backward* stochastic differential equation (a.k.a., diffusion equation).

Let us start with the *forward stage* by considering a D-dimensional diffusion process $\mathbf{U}$ driven by a Brownian motion $\mathbf{W}$, with a drift coefficient $\mathbf{b}(\cdot)$:

$$\mathbf{U}_t = \mathbf{U}_0 + \int_0^t \mathbf{b}(\mathbf{U}_s)\,\mathrm{d}s + \mathbf{W}_t, \quad \mathbf{U}_0 \overset{\mathrm{d}}{=} \mathbf{X}. \tag{15}$$

Running the diffusion $\mathbf{U}$ from $t = 0$ to $t = T$ (forward path) corresponds to the *Encoder*, yielding a latent distribution $p_{\mathbf{U}_T} =: p_T$ as for the VAE. Usually, for an arbitrary drift b, the distribution p_T is not tractable and therefore directly sampling from it is impossible. But taking b of the form $b(\mathbf{x}) = -\mathbf{x}$ (Langevin scheme), $p_T =: \mathbf{Z}$ is close to a Gaussian distribution (at least when T is large enough), from which one can easily sample and thus can be really used as a latent distribution.

Now, we shall restart from this latent distribution p_T and by going backward in time, retrieve the data distribution $p_{\mathbf{U}_0} \overset{\mathrm{d}}{=} \mathbf{X}$: this *backward stage* plays the role of *Decoder* in the VAE terminology. Actually $\{\mathbf{V}_t = \mathbf{U}_{T-t} : 0 \le t \le T\}$ has the same law [7] as the solution to the diffusion equation

$$\mathbf{V}_t = \mathbf{V}_0 + \int_0^t \{-\mathbf{b}(\mathbf{V}_s) + \partial_{\mathbf{v}} \log p_{T-s}(\mathbf{V}_s)\}\,\mathrm{d}s + \mathbf{Z}'_t, \quad \mathbf{V}_0 \sim p_T,$$

where $\mathbf{Z}'$ is another standard Brownian motion (independent from $\mathbf{V}_0$) and where p_t denotes the density of $\mathbf{U}_t$. The so-called score $\partial_{\mathbf{v}} \log p_{T-s}(\mathbf{v})$ can be learned with a neural network $\mathrm{NN}_{\boldsymbol{\theta}}(s, \mathbf{v})$. At the end, the generation phase consists simply in sampling the joint latent distribution $(\mathbf{Z}, \mathbf{Z}')$ for $\mathbf{V}_0$ and the Brownian motion, and then solving

$$\mathbf{V}_T = \mathbf{V}_0 + \int_0^T \{-\mathbf{b}(\mathbf{V}_s) + \mathrm{NN}_{\boldsymbol{\theta}^*}(s, \mathbf{V}_s)\} \, \mathrm{d}s + \mathbf{Z}'_t, \quad \mathbf{V}_0 \sim \mathbf{Z} \tag{16}$$

using a Euler-Maruyama scheme. Observe that the representation (16) fits the theoretical generative modeling framework of Subsection 2.1 with $\mathbf{X} \overset{\mathrm{d}}{\approx} \mathbf{V}_T = G_{\boldsymbol{\theta}^*}(\mathbf{Z}, \mathbf{Z}')$.

This "diffusion model" approach has gained a lot of popularity in recent years, with successful achievements (it is used in DALL-E 2 by OpenAI, Imagen by Google). However, applications to sample extreme distributions have not been considered so far, to our knowledge; it will be certainly investigated in the near future.

3 Simulating Extremes with GANs

The GAN's ability to simulate observations in heavy tails is assessed in two situations: Real financial data and simulated bivariate data from a Gumbel copula with Burr margins. We begin by providing some implementation details in the next two paragraphs.

3.1 Performance Assessment

Let $\{\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_n\}$ be the outputs generated by the GAN model, where $\tilde{\mathbf{x}}_i = G_{\theta^\star}(\mathbf{z}_i)$, with $\mathbf{z}_i$ drawn from a uniform distribution with several latent dimensions q tested in the cross-validation phase, and with $\tilde{x}_{1,n}^{(j)} \leq \cdots \leq \tilde{x}_{n,n}^{(j)}$ denoting the order statistics associated with the jth coordinate, $j \in \{1, \ldots, D\}$. The fit on the marginal distribution tails is assessed using the Mean Squared Logarithmic Error (MSLE) defined as

$$\mathrm{MSLE}(\xi) = \frac{1}{D\lceil (1-\xi)n \rceil} \sum_{j=1}^{D} \sum_{i=1}^{\lceil (1-\xi)n \rceil} \left\{ \frac{\log(x_{n-i+1,n}^{(j)}) - \log(\tilde{x}_{n-i+1,n}^{(j)})}{\log(2)} \right\}^2 .$$

The MSLE is the usual Mean Squared Error (MSE) criterion with some adaptations to the extreme framework: It is computed on the largest log-observations. The logarithm scale is natural when dealing with heavy-tailed random variables, see Equation (19) below and its graphical consequences on Figure 5. We use $\xi \in \{0.90, 0.95, 0.99\}$ to focus on the tails. The $\log(2)$ factor is introduced so that a 100% relative error on the log-marginals (i.e. $\tilde{x}_{n-i+1,n}^{(j)} = 2x_{n-i+1,n}^{(j)}$ or $\tilde{x}_{n-i+1,n}^{(j)} = x_{n-i+1,n}^{(j)}/2$ for all $i \in \{1, \ldots, \lceil (1-\xi)n \rceil\}$ and $j \in \{1, \ldots, D\}$) yields

$\mathrm{MSLE}(\xi) = 1$. Considering the dependence structure, one may graphically compare the estimated Kendall's dependence functions on the n observations associated with the original and generated samples. We also compare the estimated Kendall's tau $\hat{\tau}_n$ and $\tilde{\tau}_n$ on the original and GAN samples, respectively.

3.2 Computational Aspects

The Adam optimizer [19] is implemented for model training and parameters optimization, this is nowadays a standard method for stochastic gradient descent. It is used with default parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$ for the exponential decay rates. For all tests, we have performed 1 000 iterations and every 5 iterations, the MSLE metric has been computed (thus giving 200 values): the parameters of the ReLU neural network associated with the best results among the 200 checkpoints are selected. The experiments have been conducted on the Cholesky computing cluster from Ecole Polytechnique. The code was implemented in Python 3.8.2 using the library `PyTorch` 1.7.1 for the GANs' training,

3.3 Real Financial Data

The GAN approach is tested on closing prices of daily financial stock market indices downloaded from `https://stooq.com/db/h/` on the October 1st, 2020. Six indices are selected NKX (Nikkei, Japan), KOSPI (Korea), HSI (Hong-Kong), CAC (France), AMX (Amsterdam Exchange, Netherlands), Nasdaq (USA) from three market zones: Asia, Europe, USA. As a pre-processing step, the daily log-returns are computed for each index and positive values are discarded to focus on simulation of losses. In case of missing data for a given business day, the next available day is removed from the dataset. Besides, we kept only the data available at the same date for all selected indices. The performance of the GAN approach is assessed on four datasets of increasing dimensions: NKX ($D = 1$), Europe (AEX, CAC, $D = 2$), Asia (NKX, KOSPI, HSI, $D = 3$) and world (AEX, CAC, NKX, KOSPI, HSI, NDQ, $D = 6$), see Table 1.

Table 1: $\mathrm{MSLE}(\xi)$ associated with the GAN approach on real financial data.

	NKX	Europe	Asia	World
Dimension D	1	2	3	6
Sample size n	3173	2504	1378	548
$\xi = 0.90$	0.984	8.034	4.897	6.881
$\xi = 0.95$	1.544	10.251	3.082	9.297
$\xi = 0.99$	2.874	5.811	2.129	10.407

It is clear that the GAN method is unable to reproduce the marginal distribution tails; for all quantile levels $\xi \in \{0.90, 0.95, 0.99\}$ and all considered datasets, the numerical results point towards relative errors close to or larger than 100%. The quality of the results even deteriorates when the dimension D

increases, a similar deterioration globally happens when ξ increases. In any case, GAN results are quite unsatisfactory and this is intuitively coherent with the toy example depicted on Figure 2.

These disappointing results can be graphically interpreted from Figure 5 where the performance of the generator is visualised by comparing log quantile-quantile plots, namely the pairs $(\log\{(n+1)/i\}, \log x^{(j)}_{n-i+1,n})$ and $(\log\{(n+1)/i\}, \log \tilde{x}^{(j)}_{n-i+1,n})$ for $i \in \{1, \ldots, \lceil (1-\xi)n \rceil\}$ and $j \in \{1, \ldots, 6\}$. The quantile-quantile plots computed on all indices at level $\xi = 0.95$ are approximately linear which provides graphical evidence of the tail heaviness of all six marginal distributions; see Section 4 for theoretical details. It is apparent that GAN samples do not reproduce well the heavy tail property of the original samples. This under-estimation of the tail heaviness can be quantified by computing the slopes associated with all quantile-quantile plots, thus providing an estimation of the tail-index (the parameter γ defined in (18)) on all six datasets, see Table 2. Clearly, all GAN samples have much lighter tails than the original ones.

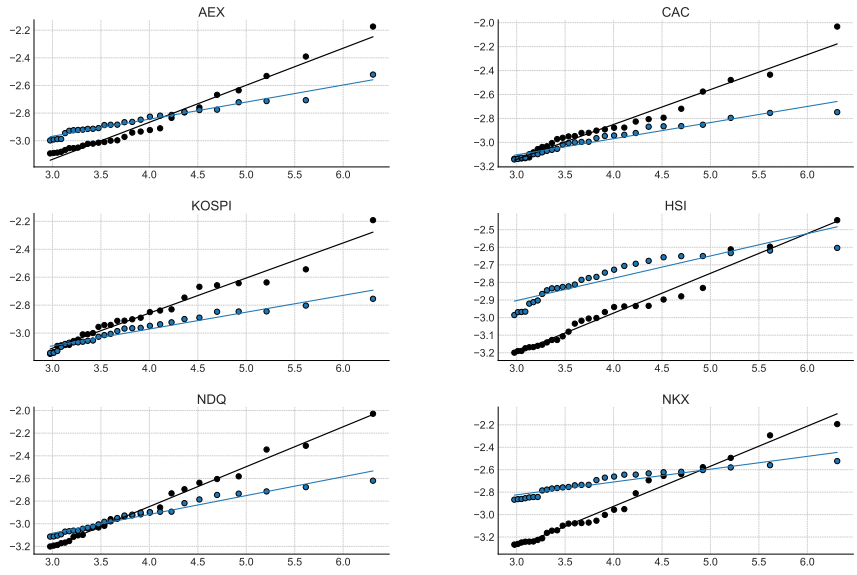


Figure 5: Log quantile-quantile plots from GAN approach applied to six financial indices at probability level $\xi = 0.95$ (black: real data, blue: GAN data). The regression line is superimposed onto each plot.

Table 2: Estimated tail indices from GAN approach applied to real financial data.

Ticker	Original data	GAN data
AEX	0.268	0.124
CAC	0.292	0.135
NKX	0.357	0.114
KOPSI	0.251	0.120
HSI	0.226	0.127
NDQ	0.352	0.166

3.4 Simulated Bivariate Data

In this experiment, we consider simulated data from a Gumbel copula C_μ^{G} model, see Section 7, with Burr(γ, ρ) margins. Its cumulative distribution function is given, for all $x \geq 0$, by

$$F_{\mathrm{Burr}}(x) = 1 - (1 + x^{-\rho/\gamma})^{1/\rho}, \tag{17}$$

with $\gamma > 0$ and $\rho < 0$. The Burr distribution is heavy-tailed in the sense that it belongs to the Fréchet maximum domain of attraction [18, Theorem 1.2.1], with tail-index γ and second-order parameter ρ, see [18, Section 2.3] for theoretical details. A sample of size $n = 10\,000$ is simulated from the random vector $\mathbf{X} = (\mathbf{X}^{(1)}, \mathbf{X}^{(2)})$ with margins $\mathbf{X}^{(1)} \sim \mathrm{Burr}(\gamma, \rho_1)$ and $\mathbf{X}^{(2)} \sim \mathrm{Burr}(\gamma, \rho_2)$ linked by C_μ^{G} for different combinations of $(\mu, \gamma, \rho_1, \rho_2)$ in $\{1.1, 2, 10\} \times \{0.1, 0.5, 0.9\} \times \{-1, -2, -3\}^2$. The obtained values of MSLE$(\xi = 0.99)$ are reported in Table 3. It appears that, when the tail-index is large ($\gamma = 0.9$), all MSLE values are close to, or even larger than, 1, which corresponds to a relative error of 100%. This simulation experiment confirms that GANs cannot reproduce heavy-tailed phenomena. This vexing property does not seem to depend on the second-order parameters nor on the dependence parameter. In contrast, Figure 6 and Table 4 show that the dependence structure is correctly reproduced in this low-dimensional situation (recall that $d = 2$): Estimated Kendall's tau and dependence functions for the generated and original data are close to each other, and very similar to the theoretical ones. We refer to Section 5.3 for experiments in higher dimensional settings.

4 Extreme-Value Framework

Let us first consider a real-valued random variable X with cumulative distribution function F_X. In this one-dimensional setting, problem (1) benefits from an explicit solution based on the quantile function $F_X^{-1}(z) := \inf\{x : F_X(x) \geq z\}$. The inversion method [21] shows that one can set $G := F_X^{-1}$ and $Z \sim \mathrm{Unif}([0, 1])$.

Table 3: MSLE($\xi = 0.99$) associated with the GAN approach applied to simulated bivariate data.

Tail-index	2nd-order parameters	Theoretical Kendall's tau		
γ	(ρ_1, ρ_2)	$\tau_{C^G_{1,1}} = 0.1$	$\tau_{C^G_2} = 0.5$	$\tau_{C^G_{10}} = 0.9$
0.1	$(-1, -2)$	0.019	0.011	0.005
	$(-1, -3)$	0.019	0.015	0.012
	$(-2, -3)$	0.014	0.017	0.015
0.5	$(-1, -2)$	0.074	0.220	0.053
	$(-1, -3)$	0.040	0.144	0.079
	$(-2, -3)$	0.225	0.209	0.027
0.9	$(-1, -2)$	0.994	1.152	1.190
	$(-1, -3)$	0.936	1.068	0.955
	$(-2, -3)$	1.424	0.756	0.933

Table 4: Estimated Kendall's tau computed with the GAN approach applied to simulated bivariate data.

Tail-index	2nd-order parameters	Theoretical Kendall's tau		
γ	(ρ_1, ρ_2)	$\tau_{C^G_{1,1}} = 0.1$	$\tau_{C^G_2} = 0.5$	$\tau_{C^G_{10}} = 0.9$
0.1	$(-1, -2)$	0.092	0.514	0.905
	$(-1, -3)$	0.093	0.477	0.900
	$(-2, -3)$	0.086	0.511	0.899
0.5	$(-1, -2)$	0.090	0.493	0.903
	$(-1, -3)$	0.106	0.506	0.901
	$(-2, -3)$	0.093	0.473	0.885
0.9	$(-1, -2)$	0.088	0.500	0.903
	$(-1, -3)$	0.091	0.484	0.899
	$(-2, -3)$	0.073	0.487	0.900

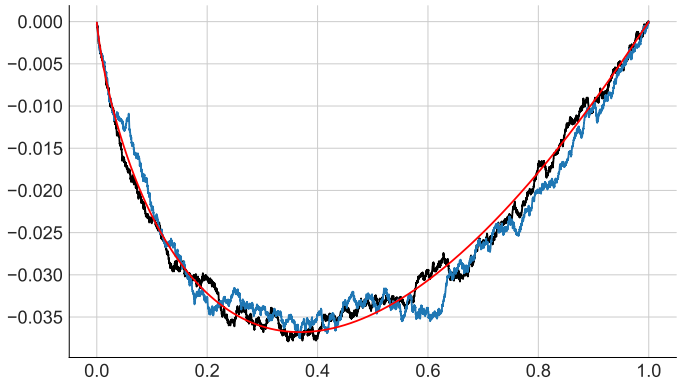


Figure 6: Estimated $\lambda(\cdot)$ functions. Black: original simulated data ($\gamma = 0.9$, $\rho_1 = -1$, $\rho_2 = -3$ and $\mu = 10$), blue: data generated with the GAN model. The theoretical $\lambda_{C_{10}^{\mathrm{G}}}$ function is superimposed in red.

4.1 Heavy-Tailed (Univariate) Distributions

We focus on heavy-tailed distributions, i.e. when F belongs to the Fréchet maximum domain of attraction [18, Theorem 1.2.1]. From [13], the survival function $\bar{F}_X := 1 - F_X$ can be written, for x large enough, as

$$\bar{F}_X(x) = x^{-1/\gamma} \mathcal{L}_X(x), \tag{18}$$

where $\mathcal{L}_X$ is a slowly-varying function at infinity: $\mathcal{L}_X(\lambda x)/\mathcal{L}_X(x) \to 1$ as $x \to \infty$ for all $\lambda > 0$. In such a case, $\bar{F}_X$ is said to be regularly-varying with index $-1/\gamma$ at infinity, which is denoted, in short, by $\bar{F}_X \in \mathrm{RV}_{-1/\gamma}$. Similarly, we shall note $\mathcal{L}_X \in \mathrm{RV}_0$. The tail-index γ tunes the tail-heaviness of the cumulative distribution function F_X; the larger γ is, the heavier the tail of X.

As a consequence of (18), the tail quantile function $F_X^{-1}(1-1/x)$ is regularly-varying with index γ at infinity, see [18, Proposition B.1.9.9], or, equivalently,

$$\log F_X^{-1}(z) = \gamma \log\left(\frac{1}{1-z}\right) + \log L\left(\frac{1}{1-z}\right), \tag{19}$$

for all $z \in (0,1)$ with $L \in \mathrm{RV}_0$. Now, since L is slowly-varying, $\log L(v)/\log v \to 0$ as $v \to \infty$ from [13, Proposition 1.3.6] and then,

$$\log F_X^{-1}(z) = \gamma \log\left(\frac{1}{1-z}\right)\{1 + o(1)\}, \text{ as } z \to 1.$$

The linearity of the log-quantile-quantile plots (Figure 5) is a consequence of the above property. In addition, $F_X^{-1}(z) \to \infty$ as $z \to 1$ so that F_X^{-1} does not fulfill the assumptions of Theorem 2.2: There is no theoretical guarantee that a neural

network (3) could uniformly approximate G. Moreover, since Z is a bounded random variable, when the activation function σ is continuous, $G_{\theta_K}(Z)$ is also a bounded random variable and thus cannot be heavy-tailed. The disappointing behavior of the GAN observed in the previous two paragraphs can thus be explained in light of these remarks.

In the following, an additional assumption is introduced on F_X, or equivalently on L, to refine the heavy-tail model (18). To this end, consider the Karamata representation:

$$L(x) = c(x) \exp\left\{ \int_1^x \frac{\varepsilon(t)}{t} \, \mathrm{d}t \right\}, \tag{20}$$

where $c(x) \to c_\infty$ as $x \to \infty$ and ε is a measurable function such that $\varepsilon(x) \to 0$ as $x \to \infty$. Our second main assumption then writes:

$$c(x) = c_\infty > 0 \text{ for all } x \geq 1 \text{ and } \varepsilon \in \mathrm{RV}_\rho \text{ with } \rho < 0. \tag{21}$$

The assumption that c is a constant function is equivalent to assuming that L is normalized [40] and ensures that L and thus F_X^{-1} are differentiable. The condition $\varepsilon \in \mathrm{RV}_\rho$ with $\rho < 0$ entails that $L(x) \to c_\infty$ as $x \to \infty$. The index of regular variation ρ is referred to as the second-order parameter. It is the main driver of the bias in the estimation of extreme quantiles from heavy-tailed distributions. Besides, (21) implies that F_X satisfies the so-called second-order condition [18, Equation (2.3.22)] which is the cornerstone of all proofs of asymptotic normality in extreme-value statistics.

4.2 Examples of Heavy-Tailed Distributions

The following examples of heavy-tailed marginal distributions are repeatedly used in this work. First, comparing (17) and (18), it is clear that the $\mathrm{Burr}(\gamma, \rho)$ distribution is heavy-tailed with slowly-varying function $\mathcal{L}(x) = (1 + x^{\rho/\gamma})^{1/\rho}$, $x \geq 0$. The log-quantile is given for all $z \in (0, 1)$ by

$$\log F_X^{-1}(z) = -\frac{\gamma}{\rho} \log\left\{ (1-z)^\rho - 1 \right\},$$

so that $L(x) = (1 - x^\rho)^{-\gamma/\rho}$, $x > 0$ and (21) holds with $c_\infty = 1$ and $\varepsilon(t) = \gamma/(t^{-\rho} - 1)$. The rate of divergence of $z \mapsto \log F_X^{-1}(z)$ in the neighbourhood of $z = 1$ is mainly driven by the tail-index γ, see the top panel of Figure 7.

We also consider the Generalized Pareto (GP) and the Generalized Extreme Value (GEV) distributions.

5 Adapting Generative Methods to Extremes

It was from 2020 onwards that the scientific community became aware of the need to adapt generative methods to extremes, and most of the ensuing works were dedicated to GANs (see Paragraph 5.1 for an overview); for other architectures including VAEs, see Paragraph 5.2.

5.1 Improvements of GANs

Three main directions have been investigated to adapt GANs to heavy tails: A preprocessing of the data to get rid of the tail heaviness, the use of heavy-tailed latent variables, and the introduction of new parametrizations to adapt the optimization problem (9) to the heavy-tail situation.

Preprocessing: Quant-GAN and evtGAN

If the latent variable Z is chosen to be Gaussian, then piecewise linear transforms of Z obtained with (3) combined with ReLU functions (7) remain Weibull-tailed [54] and therefore cannot be heavy-tailed. Any activation function with sub-polynomial growth (i.e. any currently used activation functions) leads to the same conclusion. In [55], it is remarked that log-returns of some financial indices are well represented by Lambert W transforms of Gaussian random variables. Following the ideas of [28], they thus propose to use an inverse Lambert W transform to make the data Gaussian. This preprocessing step is part of the Quant-GAN methodology introduced in [55]. The generator outputs are finally transformed back using the direct Lambert function $W(x) = x \exp(\gamma x^2/2)$ for recovering the heavy-tailed data property. Here $\gamma > 0$ is the tail-index as defined in (18) which has to be estimated outside the GAN methodology. Similarly, in evtGAN [14], GEV distributions are fitted to the margins, which are then transformed to uniform random variables. A classical GAN is then applied before transforming back the margins of the simulated samples using the fitted extreme-value distributions.

New Latent Variables: Pareto-GAN

Another similar approach is to directly use a heavy-tailed latent variable in the GAN setting [22, 36]. It is proposed in [36] to use a GP distribution. It is then shown that the generator outputs follow the desired heavy-tailed distribution: a piecewise linear transform of the above GP distribution retrieves a tail-index equal to γ. Alternative metric spaces are also introduced to ensure that the loss function is finite. To be effective, the so-called Pareto-GAN method, similarly to Quant-GAN, requires accurate estimation of the tail-index associated with each heavy-tailed marginal distribution. This is a challenging task in extreme-value theory, see [18, Chapter 3].

New Parametrizations: EV-GAN and Tail-GAN

In [3], it is proposed to take advantage of the quantile representation (19), established in the heavy-tail framework (18), to introduce a new parametrization of GANs. Specifically, it is remarked that the tail-index function (TIF)

$$f^{\mathrm{TIF}}(z) = -\frac{\log\{F_X^{-1}(z)\}}{\log\{\varphi(z)\}} \quad \text{with} \quad \varphi(z) = \frac{1-z^2}{2}, \quad z \in [0,1), \tag{22}$$

is continuous, bounded on $[0,1]$ and tends to the tail-index as $z \to 1$, see the middle panel of Figure 7 for an illustration in the Burr case. As such, f^{TIF} fulfils the assumptions of Theorem 2.2 and can thus be uniformly approximated by a neural network. However, it appears that f^{TIF} is usually not differentiable at $z = 1$ and thus, a neural network approximation is hardly accurate; this leads to a refinement of the tail-index function. Under the additional assumption (21), a corrected version of the TIF is introduced:

$$f_{\boldsymbol{\beta}}^{\mathrm{CTIF}}(z) = f^{\mathrm{TIF}}(z) - \sum_{j=1}^{6} \beta_j e_j(z),$$

where $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_6)$ is to be estimated, and $\{e_1, \ldots, e_6\}$ are universal functions that are given in [3, Eq. 13]. It is then shown [3, Proposition 1] that, if the second-order parameter ρ is strictly smaller than -1, then $f_{\boldsymbol{\beta}}^{\mathrm{CTIF}}$ is continuously differentiable on $[0,1]$, and the approximation error in Theorem 2.2 is derived as a function of K, the number of neurons in (3). It is proved that, when $\rho \in [-2,-1)$, there exist $\boldsymbol{\theta} \in \mathbb{R}^{3K+1}$, $\boldsymbol{\beta} \in \mathbb{R}^6$ and $C > 0$ such that

$$\sup_{z \in [0,1]} |f_{\boldsymbol{\beta}}^{\mathrm{CTIF}}(z) - G_{\boldsymbol{\theta}}(z)| \leq C \cdot K^{\tau},$$

with $\tau \in (\rho, -1)$. We refer to [3, Theorem 4] for technical details and other ranges of ρ values. The above approximation result can be interpreted in terms of Wasserstein-1 distance between the true data distribution and the simulated one. Assume $\gamma < 1$. Then, the Wasserstein-1 distance is given by

$$W_1(\boldsymbol{\theta}, \boldsymbol{\beta}) = \int_0^1 \left| F_X^{-1}(z) - \varphi(z)^{-\tilde{G}_{\boldsymbol{\theta},\boldsymbol{\beta}}(z)} \right| \mathrm{d}z,$$

where φ is defined in (22) and $\tilde{G}_{\boldsymbol{\theta},\boldsymbol{\beta}}$ is the enriched version of the generator $G_{\boldsymbol{\theta}}$ defined as

$$\tilde{G}_{\boldsymbol{\theta},\boldsymbol{\beta}}(z) = G_{\boldsymbol{\theta}}(z) + \sum_{j=1}^{6} \beta_j e_j(z).$$

One can then prove that there exist $\boldsymbol{\theta} \in \mathbb{R}^{3K+1}$, $\boldsymbol{\beta} \in \mathbb{R}^6$ and $C > 0$ such that $W_1(\boldsymbol{\theta}, \boldsymbol{\beta}) \leq C \cdot K^{\tau}$, see [3, Corollary 5]. The optimization process of the EV-GAN is the same as the one in [29, Algorithm 1 and Section 3]. The only difference comes from the generator's design described in Algorithm 1. Last, observe that the new parametrization still fits the theoretical generative framework of Subsection 2.1 since

$$X \overset{\mathrm{d}}{\approx} \varphi(Z)^{-\tilde{G}_{\boldsymbol{\theta},\boldsymbol{\beta}}(Z)} \quad \text{with} \quad Z \overset{\mathrm{d}}{=} \mathrm{Unif}(0,1).$$

An extension to multidimensional data is described in Algorithm 1: Each marginal is approximated using the above principle and the dependence between margins follows from the term $G_{\boldsymbol{\theta}}(\mathbf{Z})$. More details can be found in [3].

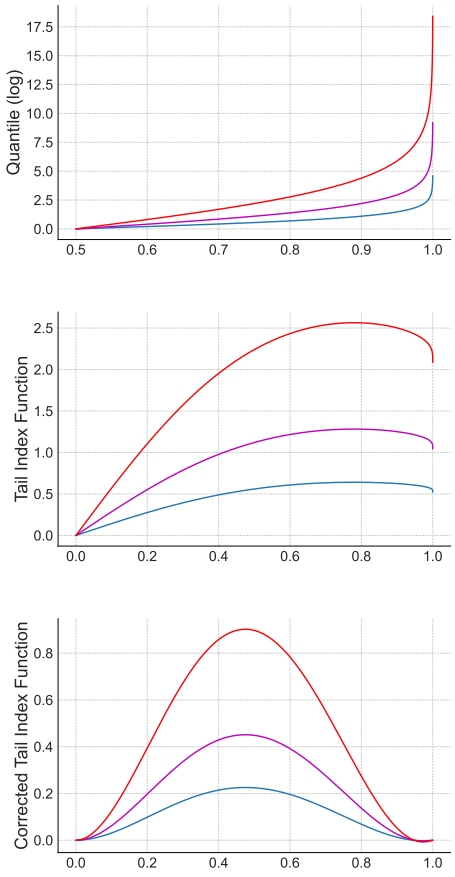


Figure 7: Log-quantile function (top panel), tail-index function (middle panel), and corrected tail-index function (bottom panel) associated with the Burr($\gamma, \rho = -1$) distribution with tail-index $\gamma = 1/2$ (blue), $\gamma = 1$ (purple) and $\gamma = 2$ (red).

Algorithm 1: EV-GAN Generator

Input: *trained parameters* $(\boldsymbol{\theta}, \boldsymbol{\beta})$

Generate *latent variable* $\mathbf{Z}$ *in dimension* $q \geq D$, with independent and Unif$(0,1)$-margins

for *each marginal* $d = 1 : D$ **do**

$G^{\mathrm{EVGAN},(d)}_{\boldsymbol{\theta},\boldsymbol{\beta}}(\mathbf{Z}) = \varphi(Z^{(d)})^{-\left(G_{\boldsymbol{\theta}}(\mathbf{Z}) + \sum_{j=1}^{6} \beta_j e_j(Z^{(d)})\right)}$

Heuristics: Ex-GAN

Alternatively, in [10], a distribution shifting is first introduced to reduce the lack of training data in the tails. Second, a GAN parametrization conditioned by samples drawn from a GP distribution is fitted to the shifted data. Finally, an additional term representing some distance to a desired extremeness is added to the loss function. Although numerical results on images are promising, the theoretical underpinnings are questionable; getting error or complexity bounds in the NN architecture of the generator is seemingly out of reach.

5.2 Other Architectures

We finally list two works outside the GAN framework and dedicated to the simulation of multivariate extremes.

Improvements of VAEs

It is shown in [43, Corollary 7] that a VAE built with piecewise linear activation functions and Gaussian distributions for both $p^{\mathcal{C}_{\boldsymbol{\phi}}}_{\mathbf{Z}\,|\,\mathbf{X}}$ and $p^{\mathcal{D}_{\boldsymbol{\theta}}}_{\mathbf{X}\,|\,\mathbf{Z}}$ cannot reproduce heavy-tailed margins. It is also proved, under some assumptions, that the angular measure (that appears in multivariate extreme value theory) associated with a classical VAE output is necessarily discrete, see [43, Proposition 8]. To overcome these problems, the authors consider a univariate heavy-tailed distribution to sample the radius R, and, conditionally on the latter, an angle $\boldsymbol{\theta}$ is sampled from a multivariate normal distribution. The product of the two gives 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 [57], the authors develop a VAE incorporating the max-id spatial model in order to sample spatial extremes with flexible and non-stationary dependence properties.

D-Max-Decreasing Neural Networks

A new architecture called D-max-decreasing neural network is proposed in [32], inspired by Maxout networks [30]. It naturally encodes the constraints associated with an angular measure and therefore the outputs of the neural network are simulated from a valid multivariate extreme-value distribution. The proof of an approximation rate is part of the authors' future work.

5.3 Simulating Extremes with GANs, Numerical Illustrations in Higher Dimension

While the theoretical analysis focused on the marginals, the ability of a GAN and of its refinement EV-GAN to properly scale in high dimension is now investigated. Using the `R` package `copulas` [41], $n = 10\,000$ samples are simulated from the D-variate Gumbel copula $C^{\mathrm{G}}_{\mu=2}$ for increasing dimensions

$D \in \{4, 8, 16, 32, 64, 128, 256, 512\}$ and with Burr($\gamma = 1/2$, $\rho = -1$) margins. MSLE(ξ) at level $\xi \in \{0.90, 0.95, 0.99\}$ are reported in Table 5 for both GAN and EV-GAN methods. EV-GAN yields realistic margins for all considered dimensions and for high levels of quantiles $\xi \in \{0.90, 0.95\}$. In the case of higher levels ($\xi = 0.99$) the dimension is limited to 128. In contrast, the classic GAN model is limited to a dimension of about 8 for all considered levels ξ. Figure 9 illustrates the dependence associated with samples in dimension $D \in \{4, 8, 16, 32, 64, 128\}$. First, remark that the $\lambda(\cdot)$ function associated with the original data tends toward the asymptotic independence function $\lambda_{\Pi,\infty}(\cdot)$ as D increases (see (25)), accordingly to [25, Section 3.3]. Second, it appears that EV-GAN manages to reproduce well the dependence structure of the original data up to $D = 16$, but tends to generate more and more independent margins in higher dimensions.

Finally, the performance of GAN to simulate six financial indices ($D = 6$) previously illustrated in Figure 5 is compared with the one of the EV-GAN in Figure 8. It appears that EV-GAN is able to generate financial indices with more realistic marginal tail behaviors than GAN can do.

Table 5: Performance comparison between GAN and EV-GAN on simulated D-variate data with respect to the MSLE(ξ) criteria computed at levels $\xi \in \{0.90, 0.95, 0.99\}$, MSLE($\xi$) ≥ 1 are not reported.

dimension	MSLE(0.90)		MSLE(0.95)		MSLE(0.99)	
D	GAN	EV-GAN	GAN	EV-GAN	GAN	EV-GAN
4	0.065	0.020	0.117	0.036	0.353	0.125
8	0.237	0.071	0.366	0.109	-	0.264
16	0.991	0.235	-	0.264	-	0.198
32	0.988	0.261	0.908	0.209	-	0.666
64	-	0.280	-	0.265	-	0.318
128	-	0.403	-	0.307	-	0.603
256	-	0.376	-	0.642	-	-
512	-	0.404	-	0.393	-	-

6 Notes and Comments

A reference for Kuratowski's Theorem mentioned in Subsection 2.1 is [9, Proposition 7.15]; this theorem is sometimes also called the measurable isomorphism Theorem [53, Page 7], in the case where the input and output spaces are Polish spaces (complete and separable metric spaces). This result supports the principle of generative modeling. About the universal approximation theorem (Theorem 2.2) for neural networks, we refer the reader to the broad overview by Pinkus [49], including historical references. Theorem 2.2 can be extended to discontinuous activation functions σ, see [49, Proposition 3.8] for the detailed assumptions. Approximation properties of deep neural networks are investigated

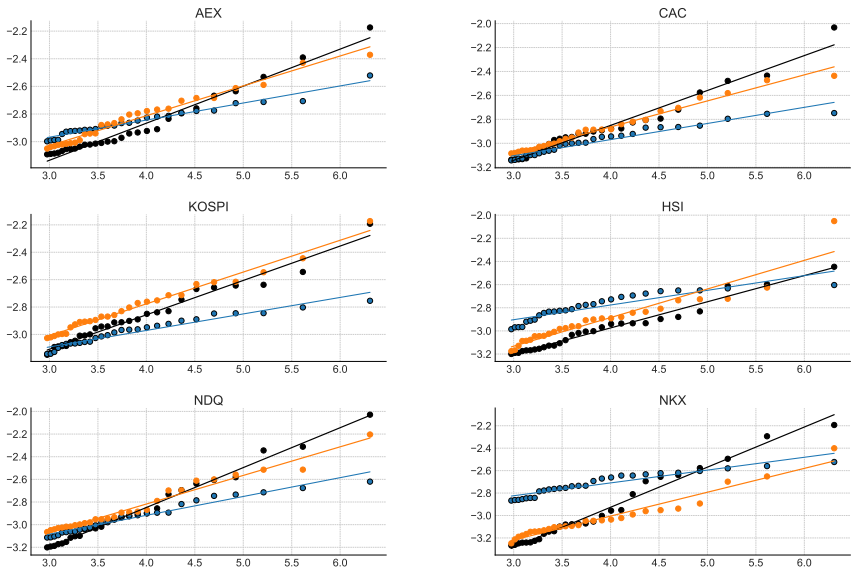


Figure 8: Log quantile-quantile plots from GAN and EV-GAN approaches applied to six financial indices at probability level $\xi = 0.95$ (black: real data, blue: GAN data, orange: EV-GAN data). The regression line is superimposed onto each plot.

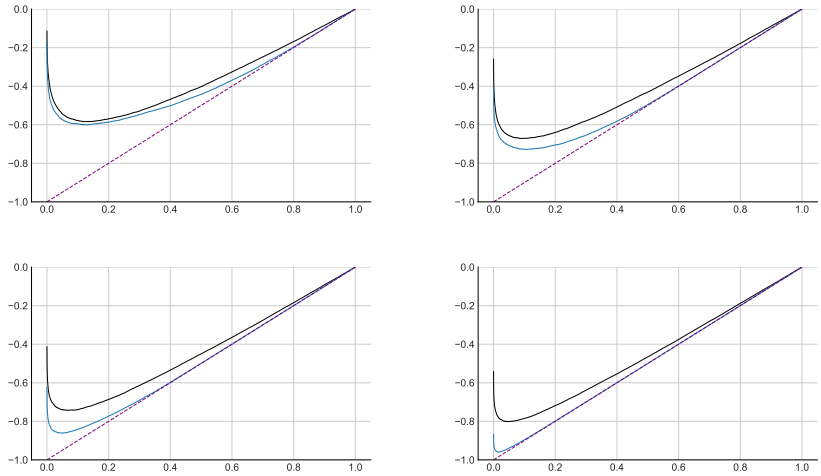


Figure 9: Estimated $\lambda(\cdot)$ functions on D-variate simulated data with $D \in \{16, 32, 64, 128\}$ (from left to right and top to bottom). Black: initial sample, blue: sample from the EV-GAN model, dashed purple: asymptotic independence case $t \mapsto \lambda_{\Pi,\infty}(t) = t - 1$.

by Yarotsky in [56]. In the context of generative modeling of extremes, to our knowledge, there is no clear (numerical or theoretical) evidence as-of-today that using deep neural networks yields better performance compared to shallow neural networks. Indeed, on the theoretical side, it is known that deep neural networks can approximate efficiently smooth functions, but in our context of extremes and heavy-tails, the issues come from the explosion in the tails not from the smoothness properties.

Neural network generative modeling is increasingly attracting attention due to impressive empirical results. Focusing on (2), this problem first requires information on the quantity of interest (regularity, structure) to give theoretical guidelines to build an appropriate generator with desired properties (convergence and stability of the optimization process, richness of the generated samples). When the target random quantity is supposed to be heavy-tailed, we have shown (theoretically and on several data experiments) that usual generative models cannot reproduce this property without dedicated architecture improvements.

An overview of a number of proposals along these lines has been proposed in this work for GAN and VAE frameworks. VAE has been first introduced by Kingma and Welling in 2013, see [38]. We refer to [20, 39] for tutorials and overviews on VAEs. We also refer to [8] for the alternative Wasserstein GAN method that has not be tested here for simulating extremes and to [12, 31] for the associated theoretical properties. Diffusion models [34, 52] are the most recent class of generative neural networks. The dynamics of the diffusion (forward and backward) are parameterized by neural networks in the drift of a Gaussian noise, see [17] for detailed discussions. So far, to our knowledge, these diffusion models have not been used to generate extreme values. It would be interesting to investigate whether these new generative methods are able to simulate realistic tail events. This is of primary importance in risk assessment where simulating too light-tailed events may yield a severe underestimation of extreme risks. We are grateful to an anonymous referee who pointed to us another quite recent approach [33] based on normalizing flows to generate heavy-tailed data, this would be certainly interesting to compare the performance of all these different methods.

We also believe that neural networks can be used to estimate tail quantities such as extreme risk measures. In [15], the joint elicitability property [2] of the VaR and ES risk measures is exploited to propose a new GAN parametrization. An universal approximation theorem is provided for a broad class of tail risk measures: any Hölder continuous spectral risk measure can be approximated with an arbitrary precision by the proposed GAN architecture. From a practical point of view, one can for instance increase the sample size using generative methods so that estimation no longer requires extrapolation. Another solution explored in [5, 6] is to exploit the powerful optimization techniques associated with neural networks to fit higher-order extreme-value models, to reduce the estimation bias; the authors prove that using an eLU function as the activation function of neural networks plays a crucial role for improving accuracy.

7 Appendix: Copulas

Let us consider a cumulative distribution function $F_{\mathbf{X}}$ defined on $\mathbb{R}^D$ with continuous margins denoted by $F_{\mathbf{X}}^{(j)}$, $j \in \{1, \ldots, D\}$. From Sklar's Theorem [51], there exists a unique function $C : [0,1]^D \to [0,1]$ such that

$$F_X(\mathbf{x}) = C(F_{\mathbf{X}}^{(1)}(\mathbf{x}^{(1)}), \ldots, F_{\mathbf{X}}^{(D)}(\mathbf{x}^{(D)})),$$

for all $\mathbf{x} = \left(\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(D)}\right) \in \mathbb{R}^D$. The function C is called the copula of $F_{\mathbf{X}}$. Introducing the uniform random variables $\mathbf{U}^{(j)} = F^{(j)}(\mathbf{X}^{(j)})$ for all $j \in \{1, \ldots, D\}$, the copula C is the D dimensional distribution function of the random vector $\left(\mathbf{U}^{(1)}, \ldots, \mathbf{U}^{(D)}\right)$ defined on the unit cube $[0,1]^D$ with uniform margins on $[0,1]$. Copulas are a flexible tool to impose a given dependence structure on the marginal distributions of interest, see [47] for a detailed account on copulas. The independence between margins corresponds to the product copula $\Pi(\mathbf{u}) = \mathbf{u}^{(1)} \times \cdots \times \mathbf{u}^{(D)}$, while comonotonicity corresponds to the Fréchet copula $M(\mathbf{u}) = \min(\mathbf{u}^{(1)}, \ldots, \mathbf{u}^{(D)})$.

7.1 Archimedean Copulas

An Archimedean copula is defined, for all $\mathbf{u} = (\mathbf{u}^{(1)}, \ldots, \mathbf{u}^{(D)}) \in [0,1]^D$, by

$$C_\mu(\mathbf{u}) = \psi_\mu(\psi_\mu^{-1}(\mathbf{u}^{(1)}) + \cdots + \psi_\mu^{-1}(\mathbf{u}^{(D)})),$$

where $\psi_\mu : [0, \infty) \to [0,1]$ is a parametric function (with parameter μ) called "generator" which has to satisfy some properties listed, for instance, in [45]. It is easily seen that the independence copula Π is Archimedean, generated by $\psi(t) = \exp(-t)$, $t \geq 0$. In the following, we shall focus on the Gumbel copula,

$$C_\mu^{\mathrm{G}}(\mathbf{u}) = \exp\left[-\left\{\sum_{j=1}^{D}(-\log \mathbf{u}^{(j)})^\mu\right\}^{1/\mu}\right],$$

where $\mu \geq 1$ tunes the dependence between the margins, see the next paragraph. The associated generator is $\psi_{C_\mu^{\mathrm{G}}}(t) = \exp(-t^{1/\mu})$ defined for all $\mu \geq 1$ and $t \geq 0$. Observe that both the product copula and Fréchet copula belong to this Gumbel copula parametric family: $\Pi(\mathbf{u}) = C_1^{\mathrm{G}}(\mathbf{u})$ and $M(\mathbf{u}) = C_\infty^{\mathrm{G}}(\mathbf{u})$.

7.2 Quantifying Dependence

First, Kendall's dependence function [27] characterizes the dependence structure associated with a copula C and is the univariate cumulative distribution function defined for all $t \in [0,1]$ by

$$K_C(t) = \mathrm{P}(C(\mathbf{U}^{(1)}, \ldots, \mathbf{U}^{(D)}) \leq t). \tag{23}$$

In the case of an Archimedean copula C_μ, it can be derived as [25, Eq. (6)]:

$$K_{C_\mu}(t) = t + \sum_{j=1}^{D-1} \frac{\{-\psi_\mu^{-1}(t)\}^j}{j!} \psi_\mu^{(j)}\{\psi_\mu^{-1}(t)\},$$

where $\psi_\mu^{(j)}$ stands for the jth derivative of ψ_μ, and we shall thus consider $\lambda_{C_\mu}(t) := t - K_{C_\mu}(t)$. It is then easily seen that $\lambda_M(t) = 0$ and

$$\lambda_\Pi(t) = -t \sum_{j=1}^{D-1} \frac{\{-\log t\}^j}{j!} \tag{24}$$

for all $t \in (0, 1]$. Let us also highlight that

$$\lambda_\Pi(t) \underset{D\to\infty}{\longrightarrow} t - 1 =: \lambda_{\Pi,\infty}(t). \tag{25}$$

Moreover, Kendall's dependence function associated with the Gumbel copula is given in the bivariate case ($D = 2$) by $K_{C_\mu^{\mathrm{G}}}(t) = t - t\log(t)/\mu$ leading to $\lambda_{C_\mu^{\mathrm{G}}}(t) = t\log(t)/\mu$ for all $t \in (0, 1]$. Clearly, $\lambda_{C_1^{\mathrm{G}}}(t) = \lambda_\Pi(t) = t\log(t)$ and $\lambda_{C_\mu^{\mathrm{G}}}(t) \to \lambda_M(t) = 0$ as $\mu \to \infty$.

Second, Kendall's tau [37] is a measure of dependence between the margins of a bivariate random vector ($D = 2$). Let $\mathbf{X}$ and $\tilde{\mathbf{X}}$ be two independent bivariate random vectors from $F_{\mathbf{X}}$. Kendall's tau is defined as the probability of concordance minus the probability of discordance of $\mathbf{X} = (\mathbf{X}^{(1)}, \mathbf{X}^{(2)})$ and $\tilde{\mathbf{X}} = (\tilde{\mathbf{X}}^{(1)}, \tilde{\mathbf{X}}^{(2)})$. From [47, Theorem 5.1.3], this quantity only depends on the copula C of $F_{\mathbf{X}}$ and is given by

$$\tau_C = 4\mathrm{E}\{C(\mathbf{U}^{(1)}, \mathbf{U}^{(2)})\} - 1 = 4\int_0^1 \int_0^1 C(u, v)\,\mathrm{d}C(u, v) - 1, \tag{26}$$

with $\tau_M = 1$ and $\tau_\Pi = 0$ as special cases. In the case of an Archimedean copula C_μ, Kendall's tau and Kendall's dependence functions are linked [26]:

$$\tau_{C_\mu} = 1 + 4\int_0^1 \lambda_{C_\mu}(v)\,\mathrm{d}v,$$

meaning that τ_{C_μ} can be interpreted as a summary of the dependence information encoded in $\lambda_{C_\mu}(\cdot)$. As an example, the Kendall's tau associated with the Gumbel copula is given, for all $\mu \geq 1$, by $\tau_{C_\mu^{\mathrm{G}}} = 1 - 1/\mu$. The dependence between the margins is thus an increasing function of μ. Note in particular that $\tau_{C_1^{\mathrm{G}}} = \tau_\Pi = 0$ while $\tau_{C_\mu^{\mathrm{G}}} \to \tau_M = 1$ as $\mu \to \infty$.

7.3 Inference for Kendall's Dependence Metrics

Estimation of Kendall's dependence function uses the pseudo-observations $\{W_1, \ldots, W_n\}$ from the cumulative distribution function K_C defined in (23)

and computed as

$$W_i = \frac{1}{n-1} \sum_{j \neq i}^{n} \mathbb{I}(\mathbf{X}_j^{(1)} < \mathbf{X}_i^{(1)}, \ldots, \mathbf{X}_j^{(D)} < \mathbf{X}_i^{(D)}), \tag{27}$$

for all $i \in \{1, \ldots, n\}$, see [27]. The estimator of K_C is computed using the associated empirical cumulative distribution function:

$$\hat{K}_{Cn}(t) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}(W_i \leq t),$$

and we set $\hat{\lambda}_n(t) = t - \hat{K}_{Cn}(t)$, for all $t \in [0, 1]$. Similarly, see [27, Eq. (7)], Kendall's tau is estimated by

$$\hat{\tau}_n = \frac{4}{n} \sum_{i=1}^{n} W_i - 1.$$

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. `https://www.tensorflow.org/`.

[2] C. Acerbi and B. Szekely. Back-testing expected shortfall. *Risk*, 27(11):76–81, 2014.

[3] M. Allouche, S. Girard, and E. Gobet. EV-GAN: Simulation of extreme events with ReLU neural networks. *Journal of Machine Learning Research*, 23(150):1–39, 2022.

[4] M. Allouche, S. Girard, and E. Gobet. A generative model for fBm with deep ReLU neural networks. *Journal of Complexity*, 73:101667, 2022.

[5] M. Allouche, S. Girard, and E. Gobet. Learning extreme expected shortfall and conditional tail moments with neural networks. Application to cryptocurrency data, 2023. `https://hal.science/hal-04347859v3`.

[6] M. Allouche, S. Girard, and E. Gobet. Estimation of extreme quantiles from heavy-tailed distributions with neural networks. *Statistics and Computing*, 34:12, 2024.

[7] B.D.O. Anderson. Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326, 1982.

[8] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein generative adversarial networks. In D. Precup and Y. W. Teh, editors, *Proceedings of the 34th Int. Conf. on Mach. Learn.*, volume 70 of *Proceedings of Machine Learning Research*, pages 214–223. PMLR, 2017.

[9] D. P. Bertsekas and S. E. Shreve. *Stochastic optimal control*, volume 139 of *Mathematics in Science and Engineering*. Academic Press, Inc., New York-London, 1978.

[10] S. Bhatia, A. Jain, and B. Hooi. ExGAN: Adversarial generation of extreme samples. arXiv preprint `arXiv:2009.08454`, 2020.

[11] G. Biau, B. Cadre, M. Sangnier, and U. Tanielian. Some theoretical properties of GANs. *The Annals of Statistics*, 48(3):1539–1566, 2020.

[12] G. Biau, M. Sangnier, and U. Tanielian. Some theoretical insights into Wasserstein GANs. arXiv preprint `arXiv:2006.02682`, 2020.

[13] N. H. Bingham, C. M. Goldie, and J. L. Teugels. *Regular variation*, volume 27 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, Cambridge, 1987.

[14] Y. Boulaguiem, J. Zscheischler, E. Vignotto, K. van der Wiel, and S. Engelke. Modeling and simulating spatial extremes by combining extreme value theory with generative adversarial networks. *Environmental Data Science*, 1:e5, 2022.

[15] R. Cont, M. Cucuringu, R. Xu, and C. Zhang. Tail-GAN: Nonparametric scenario generation for tail risk estimation. arXiv preprint `arXiv:2203.01664`, 2022.

[16] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control Signals Systems*, 2(4):303–314, 1989.

[17] V. De Bortoli, J. Thornton, J. Heng, and A. Doucet. Diffusion Schrödinger bridge with applications to score-based generative modeling. *Advances in Neural Information Processing Systems*, 34:17695–17709, 2021.

[18] L. de Haan and A. Ferreira. *Extreme value theory*. Springer Series in Operations Research and Financial Engineering. Springer, New York, 2006.

[19] P. K. Diederik and J. Ba. Adam: A method for stochastic optimization. arXiv preprint `arXiv:1412.6980`, 2017.

[20] C. Doersch. Tutorial on variational autoencoders. arXiv preprint `arXiv:1606.05908`, 2016.

[21] R. Eckhardt. Stam Ulam, John Von Neumann and the Monte-Carlo method. *Los Alamos Science*, Special Issue:131–143, 1987.

[22] R. M. Feder, P. Berger, and G. Stein. Nonlinear 3D cosmic web simulation with heavy-tailed generative adversarial networks. *Physical Review D.*, 102(10):103504, 18, 2020.

[23] D. Foster. *Generative deep learning: teaching machines to paint, write, compose, and play.* O'Reilly Media, 2019.

[24] A R. Gallant and H. White. There exists a neural network that does not make avoidable mistakes. In *IEEE 1988 International Conference on Neural Networks*, pages 657–664, 1988.

[25] M. Garcin, D. Guegan, and B. Hassani. A novel multivariate risk measure: the Kendall VaR. `https://halshs.archives-ouvertes.fr/halshs-01467857`, 2018.

[26] C. Genest and J. MacKay. The joy of copulas: bivariate distributions with uniform marginals. *The American Statistician*, 40(4):280–283, 1986.

[27] C. Genest and L.-P. Rivest. Statistical inference procedures for bivariate Archimedean copulas. *Journal of American Statistical Association*, 88(423):1034–1043, 1993.

[28] G. M. Goerg. The Lambert way to gaussianize heavy-tailed data with the inverse of Tukey's h transformation as a special case. *The Scientific World Journal*, ID 909231, 2015.

[29] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, volume 27, pages 2672–2680, 2014.

[30] I. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio. Maxout networks. In *Proceedings of the 30th International Conference on Machine Learning*, pages 1319–1327. PMLR, 2013.

[31] M. Haas and S. Richter. Statistical analysis of Wasserstein GANs with applications to time series forecasting. arXiv preprint `arXiv:2011.03074`, 2020.

[32] A. Hasan, K. Elkhalil, Y. Ng, J. M. Pereira, S. Farsiu, J. H. Blanchet, and V. Tarokh. Modeling extremes with d-max-decreasing neural networks. arXiv preprint `arXiv:2102.09042`, 2022.

[33] Tennessee Hickling and Dennis Prangle. Flexible tails for normalising flows, with application to the modelling of financial return data, 2023.

[34] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851, 2020.

[35] K. Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257, 1991.

[36] T. Huster, J. E. J. Cohen, Z. Lin, K. Chan, C. Kamhoua, N. Leslie, C. Y. J. Chiang, and V. Sekar. Pareto GAN: Extending the representational power of GANs to heavy-tailed distributions. In *International Conference on Machine Learning*, pages 4523–4532. PMLR, 2021.

[37] M. Kendall. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93, 1938.

[38] D. P. Kingma and M. Welling. Auto-encoding variational Bayes. arXiv preprint `arXiv:1312.6114`, 1st version, 2013.

[39] D. P. Kingma and M. Welling. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392, 2019.

[40] E. Kohlbecker. Weak asymptotic properties of partitions. *Transactions of The American Mathematical Society*, 88(2):346–365, 1958.

[41] I. Kojadinovic and J. Yan. Modeling multivariate distributions with continuous margins using the copula R package. *Journal of Statistical Software*, 34(9):1–20, 2010.

[42] W. Kritzinger, M. Karner, J. Traar, G.and Henjes, and W. Sihn. Digital twin in manufacturing: A categorical literature review and classification. *IFAC-PapersOnLine*, 51(11):1016–1022, 2018.

[43] N. Lafon, P. Naveau, and R. Fablet. A VAE approach to sample multivariate extremes. arXiv preprint `arXiv:2306.10987`, 2023.

[44] M. Leshno, W.Y. Lin, A. Pinkus, and S. Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural Networks*, 6(6):861–867, 1993.

[45] A. McNeil and J. Nešlehová. Multivariate Archimedean copulas, d-monotone functions and l_1-norm symmetric distributions. *The Annals of Statistics*, 37(5B):3059–3097, 2009.

[46] E. Negri, L. Fumagalli, and M. Macchi. A review of the roles of digital twin in CPS-based production systems. *Procedia Manufacturing*, 11:939–948, 2017.

[47] R. Nelsen. *An introduction to copulas*. Springer Series in Statistics. Springer, New York, second edition, 2006.

[48] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and Soumith C. Pytorch: An imperative style,

high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *33rd Conference on Neural Information Processing Systems*, pages 8024–8035. Curran Associates, Inc., 2019.

[49] A. Pinkus. Approximation theory of the MLP model in neural networks. In *Acta numerica*, volume 8, pages 143–195. Cambridge University Press, Cambridge, 1999.

[50] D. Rezende and S. Mohamed. Variational inference with normalizing flows. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37, pages 1530–1538. PMLR, 2015.

[51] A. Sklar. Fonctions de répartition à n dimensions et leurs marges. *Publications de l'Institut de Statistique de l'Université de Paris*, 8:229–231, 1959.

[52] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37, pages 2256–2265. PMLR, 2015.

[53] C. Villani. *Optimal transport*, volume 338 of *Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences]*. Springer-Verlag, Berlin, 2009.

[54] M. Vladimirova, S. Girard, N. Hien, and J. Arbel. Sub-Weibull distributions: generalizing sub-Gaussian and sub-Exponential properties to heavier-tailed distributions. *Stat*, 9:e318, 2020.

[55] M. Wiese, R. Knobloch, R. Korn, and P. Kretschmer. Quant GANs: deep generation of financial time series. *Quantitative Finance*, 20(9):1419–1440, 2020.

[56] D. Yarotsky. Optimal approximation of continuous functions by very deep ReLU networks. In *Proceedings Mach. Learn. Res.*, pages 639–649, 2018.

[57] L. Zhang, X. Ma, C. K. Wikle, and R. Huser. Flexible and efficient spatial extremes emulation via variational autoencoders, 2024. arXiv preprint `arXiv:2307.08079`.