# On the Simulation of Extreme Events with Neural Networks

## Generative AI Modelling for Extreme Events

Vianey Palacios

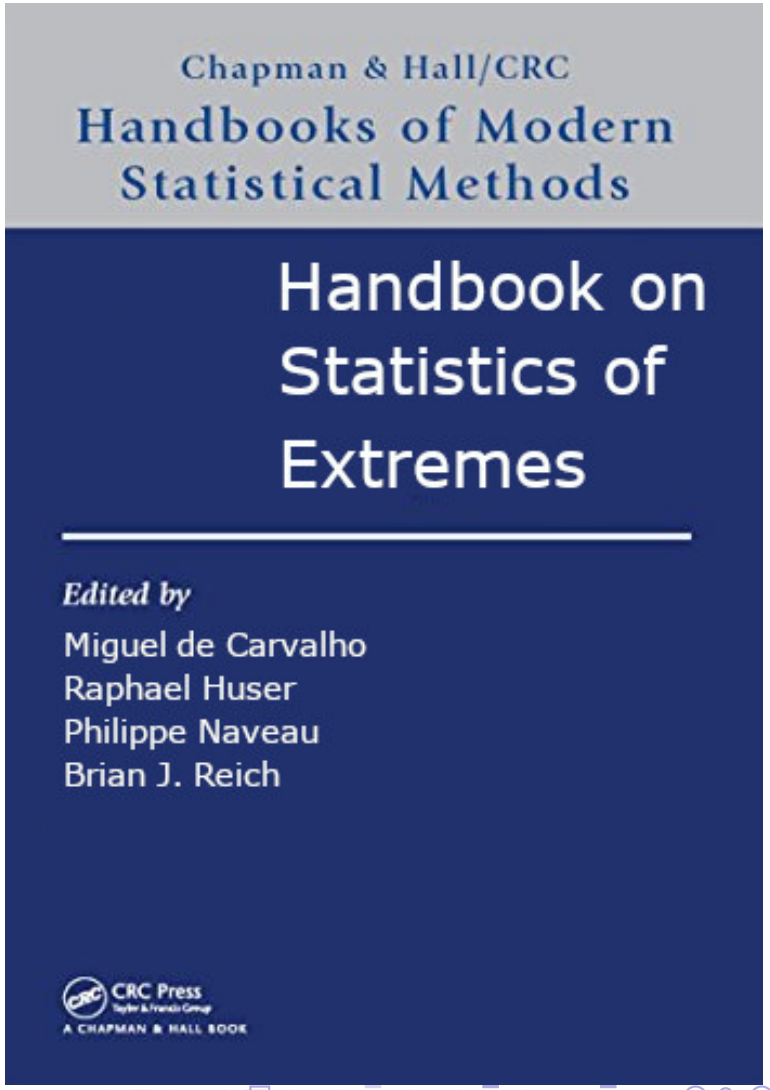
# Based on Chapter from Handbook on Statistics of Extremes

**This talk is based on:**

Chapter 20 – *"On the Simulation of Extreme Events with Neural Networks"*

by Michaël Allouche, Stéphane Girard & Emmanuel Gobet

In *Handbook on Statistics of Extremes*



# Introduction, Motivation & Background

## Generative Modeling

- This chapter explores AI in the context of extreme events, with a focus on **generative models**.
- It highlights the **limitations** of mainstream AI approaches in simulating **heavy-tailed** phenomena and presents fresh **solutions.**
- This talk focus on **Generative Adversarial Networks (GANs)**.
- The chapter also discussed **Variational Auto-Encoders** and **Difussion Models**.

# What is Generative Modeling?

- Given $\{\mathbf{x}_i\}_{i=1}^n$ from an **unknown** distribution $p_{\mathbf{X}}$.
- The **goal** of generative modeling is to find:
  - **latent distribution**: $p_{\mathbf{Z}}$ defined on some set $\mathcal{Z}$.
  - **generator**: $G : \mathcal{Z} \to \mathcal{X}$.

  such that

$$G(\mathbf{Z}) \stackrel{\mathrm{d}}{=} \mathbf{X}, \quad \mathbf{Z} \sim p_{\mathbf{Z}}$$

- We **do not** want to **infer** the unknown distribution $p_{\mathbf{X}}$ from the data.

**Theorem**

*(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 practice, distributions **easy to simulate** are considered for $p_{\mathbf{Z}}$ (Gaussian, uniform, etc). The **modelling effort** is usually put on the generator.

# Generative Modeling Framework

- Focus on a **parametric** family of generators $\{G_{\boldsymbol{\theta}}\}_{\boldsymbol{\theta}\in\boldsymbol{\Theta}}$ where $\boldsymbol{\Theta} \subset \mathbb{R}^p$ such that $G_{\boldsymbol{\theta}}(\mathbf{Z}) \sim P_{\boldsymbol{\theta}}$.
- The problem comes in finding the **'best' parameter** $\boldsymbol{\theta}^*$ such that $p_{\boldsymbol{\theta}}$, and $p_{\mathbf{X}}$ are as close as possible:

$$G_{\boldsymbol{\theta}^*}(\mathbf{Z}) \overset{\mathrm{d}}{\approx} X, \quad \text{for a given } \mathbf{Z} \sim p_{\mathbf{Z}}$$

- Ingredients:
  - The observations $\mathbf{x}_1, \ldots, \mathbf{x}_n$ with their underlying properties.
  - Inputs: $G_{\boldsymbol{\theta}}$ and the latent distribution $p_{\mathbf{Z}}$.
  - The distance or the similarity criterion between $p_\theta$ and $p_{\mathbf{X}}$, and the optimization process that defines the optimal $\boldsymbol{\theta}$.

# Generative Modeling Framework

- Focus on a **parametric** family of generators $\{G_{\boldsymbol{\theta}}\}_{\boldsymbol{\theta}\in\boldsymbol{\Theta}}$ where $\boldsymbol{\Theta} \subset \mathbb{R}^p$ such that $G_{\boldsymbol{\theta}}(\mathbf{Z}) \sim P_{\boldsymbol{\theta}}$.
- The problem comes in finding the **'best' parameter** $\boldsymbol{\theta}^*$ such that $p_{\boldsymbol{\theta}}$, and $p_{\mathbf{X}}$ are as close as possible:

$$G_{\boldsymbol{\theta}^*}(\mathbf{Z}) \overset{\mathrm{d}}{\approx} X, \quad \text{for a given } \mathbf{Z} \sim p_{\mathbf{Z}}$$

- Ingredients:
  - The observations $\mathbf{x}_1, \ldots, \mathbf{x}_n$ with their underlying properties.
  - Inputs: $G_{\boldsymbol{\theta}}$ and the latent distribution $p_{\mathbf{Z}}$.
  - The distance or the similarity criterion between $p_\theta$ and $p_{\mathbf{X}}$, and the optimization process that defines the optimal $\boldsymbol{\theta}$.

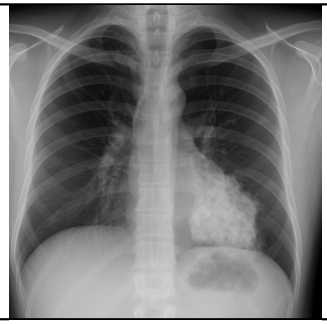
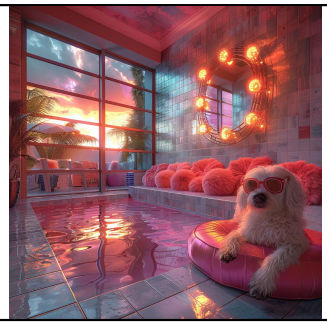
# Generative Modeling Framework

- Focus on a **parametric** family of generators $\{G_{\boldsymbol{\theta}}\}_{\boldsymbol{\theta}\in\boldsymbol{\Theta}}$ where $\boldsymbol{\Theta} \subset \mathbb{R}^p$ such that $G_{\boldsymbol{\theta}}(\mathbf{Z}) \sim P_{\boldsymbol{\theta}}$.
- The problem comes in finding the **'best' parameter** $\boldsymbol{\theta}^*$ such that $p_{\boldsymbol{\theta}}$, and $p_{\mathbf{X}}$ are as close as possible:

$$G_{\boldsymbol{\theta}^*}(\mathbf{Z}) \overset{\mathrm{d}}{\approx} X, \quad \text{for a given } \mathbf{Z} \sim p_{\mathbf{Z}}$$

- Ingredients:
  - The observations $\mathbf{x}_1, \ldots, \mathbf{x}_n$ with their underlying properties.
  - Inputs: $G_{\boldsymbol{\theta}}$ and the latent distribution $p_{\mathbf{Z}}$.
  - The distance or the similarity criterion between $p_\theta$ and $p_{\mathbf{X}}$, and the optimization process that defines the optimal $\boldsymbol{\theta}$.

# How AI Uses Generative Models

| **Model Type** | **Example Models** | **Applications** | **Image** |
|---|---|---|---|
| **Variational Autoencoders (VAEs)** | VAE, $\beta$-VAE | Image reconstruction<br>Anomaly detection<br>Medical imaging |  |
| **Generative Adversarial Networks (GANs)** | StyleGAN<br>CycleGAN | Realistic face generation<br>Image-to-image translation<br>Deepfakes |  |
| **Diffusion Models** | DALL·E 2<br>Imagen<br>Midjourney | Text-to-image generation<br>High-resolution artwork |  |

# Neural networks

- **Neural networks**: Building block for generative models.
- Consider a neural network parametrization of $G_\theta$.

**Neural networks:**

- Non-linear function built with a fixed number of neurons, each one representing a function, and distributed across several hidden layers.
- **One-hidden layer** neural network $G_\theta : \mathbb{R}^q \to \mathbb{R}$ composed of $K$ **neurons**

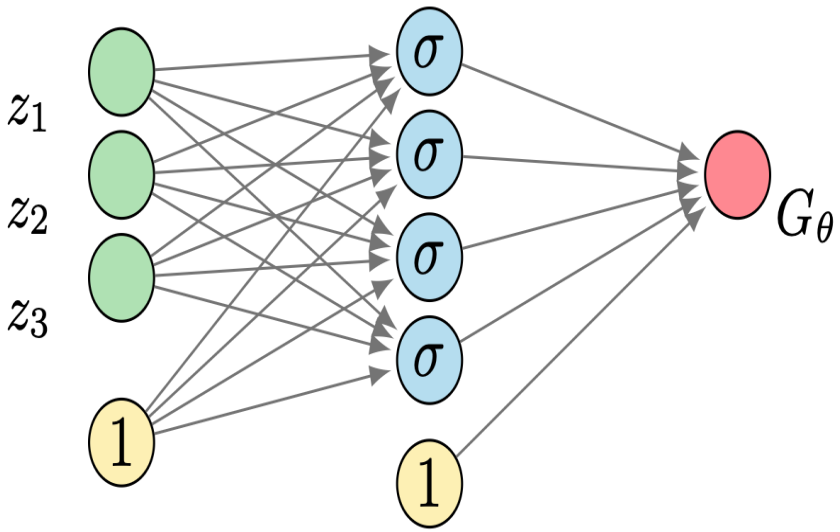
$$G_{\boldsymbol{\theta}}(\mathbf{z}) = b_2 + \sum_{k=1}^{K} w_{2,k}\sigma\left(\langle \mathbf{w}_{1,k}, \mathbf{z}\rangle + b_{1,k}\right),$$

$$\boldsymbol{\theta} := \{b_2\} \cup \{\mathbf{w}_{1,k}, w_{2,k}, b_{1,k}\}_{k=1}^{K} \in \Theta := \mathbb{R} \times \left(\mathbb{R}^q \times \mathbb{R} \times \mathbb{R}\right)^K$$

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**, $q$ is the dimension of $\mathbf{Z}$.

# Neural networks

One-hidden layer neural network




$K = 4$ neurons and input dimension $q = 3$, the arrows are different parameters $\boldsymbol{\theta}$.
*Figure from Handbook on Statistics of Extremes*

Examples of activation functions:

- The cosine squasher

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

- The logistic squasher

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

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

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

- The Rectified Linear Unit (ReLU)

$$\sigma(x) = \max(x, 0)$$

# Neural networks

- Ideally $G_{\boldsymbol{\theta}} \to G$ as $K \to \infty$ for different assumptions of $G$.

**Theorem**

*(Universal approximation theorem). Suppose $G$ is a continuous function on a compact space $\mathcal{Z} \subset \mathbb{R}^q$ and $\sigma$ 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 $\varepsilon$ ) such that*

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

- **Assumption**: function $G$ to approximate is continuous on a compact set and thus **bounded**.

# Examples $G = F_X^{-1}$ approximate the quantile function.

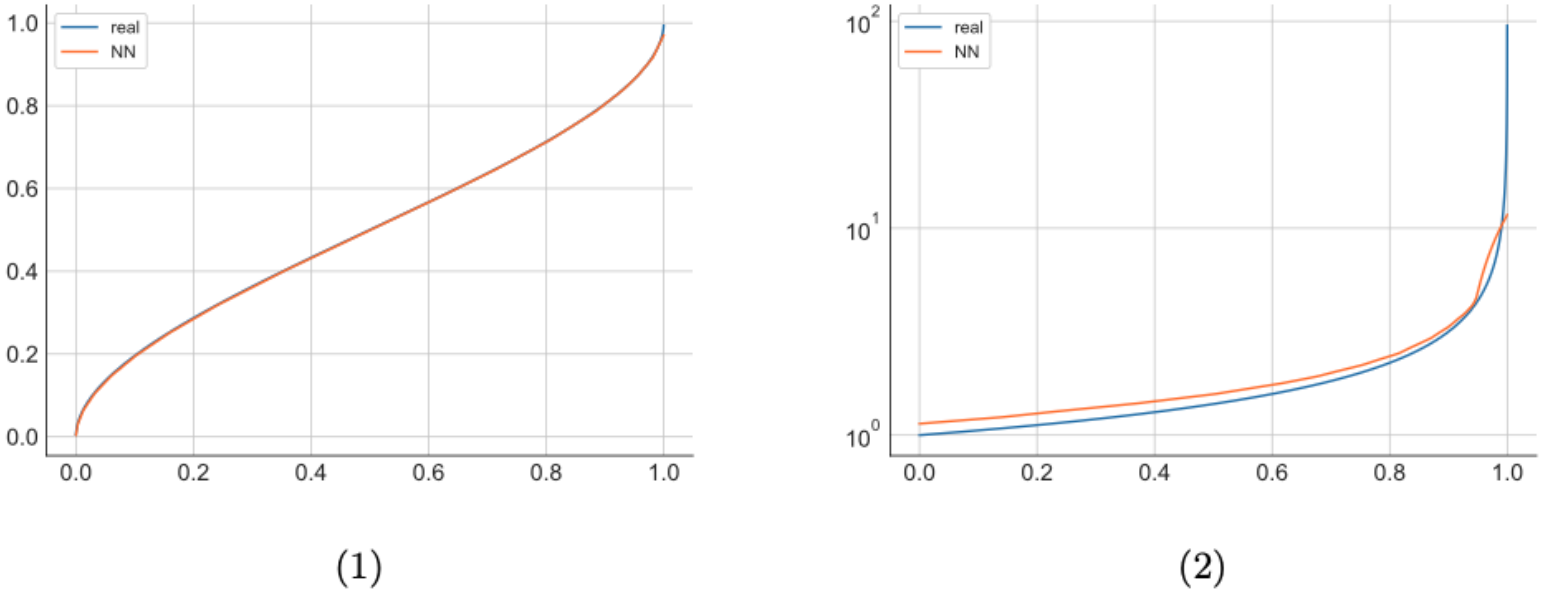



Figure: One-hidden layer neural network approximation of quantile functions $z \in [0, 1) \mapsto F_X^{-1}(z)$ from (a) Beta (2,2) and (b) Pareto ($\xi = 0.5$) distributions. The y-axis of (b) is in log-scale. *Figure from Handbook on Statistics of Extremes.*

# Generative Adversarial Network

- **Goal:** Have a generator $G$ such that $G(\mathbf{Z}) \overset{\mathrm{d}}{\approx} X$, for $\mathbf{Z} \sim p_{\mathbf{Z}}$
- A GAN aims at approximating the unknown generator $G$ through a parametric **family of neural networks**

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

- Estimate the optimal $\boldsymbol{\theta}^*$ by optimizing an objective function: **adversarial game** between a **generator** and a **discriminator**.

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

- Optimization problem:

$$\text{argmin}_{\theta \in \Theta} \max_{\phi \in \Phi} \left( \mathrm{E}_{p_X} \left\{ \log D_{\phi}(\mathbf{X}) \right\} + \mathrm{E}_{p_Z} \left\{ \log \left[ 1 - D_{\phi} \left\{ G_{\theta(\mathbf{Z})} \right\} \right] \right\} \right)$$

# Generative Adversarial Network

## Learning about EDI from Data

- **Generator**: generate realistic data.
- **Discriminator**: tries to discriminate between synthetic generated data and real observations. $D_\phi(\mathbf{x})$ represents the probability that an observation $\mathbf{x} \in \mathbb{R}^D$ is drawn from $X \sim p_X$.

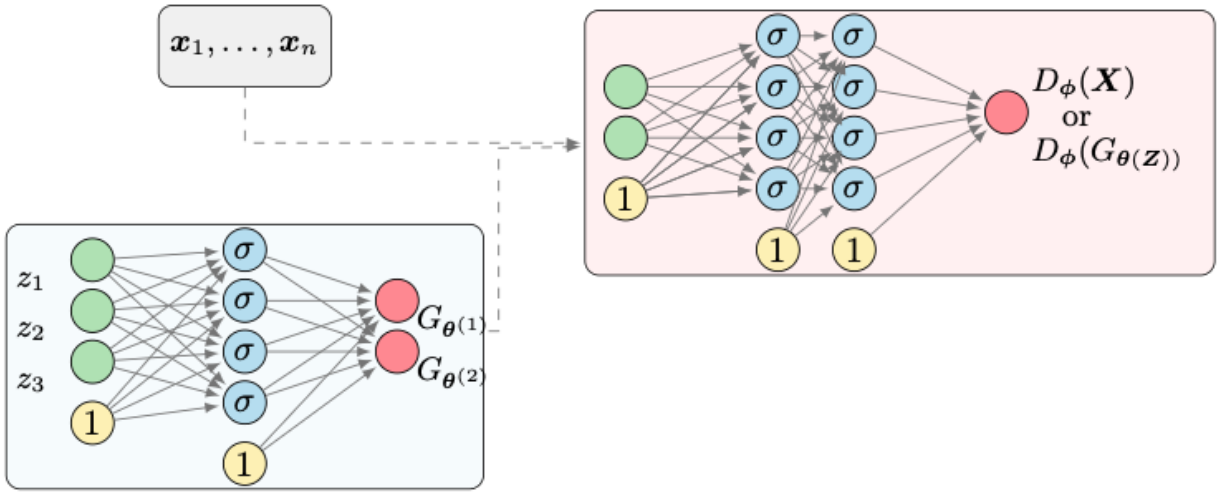



Figure: A GAN model with $q = 3$ (dimension of the latent noise) and $D = 2$ (dimension of data points). *Figure from Handbook on Statistics of Extremes.*

# Main problem?

## Simulating extremes

**Theorem**

*(Universal approximation theorem). Suppose $G$ is a* **continuous function on a compact space** $\mathcal{Z} \subset \mathbb{R}^q$ *and $\sigma$ 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 $\varepsilon$ ) such that*

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

- If $G$ is **not bounded** there is no theoretical guarantee that a neural network could uniformly approximate $G$.
- **Generator**: $G(Z) \overset{\mathrm{d}}{=} X$, for $Z \sim P_Z$
- Set $G := F_X^{-1}$ and $Z \sim U(0, 1)$.
- Since $Z$ is a bounded random variable, when the activation function $\sigma$ is continuous, $G_{\theta_K}(Z)$ is also a bounded random variable.

# Why is this a problem?

## Simulating extremes

- **Heavy tailed-distributions** (regularly varying $\mathcal{RV}_{-1/\gamma}$):

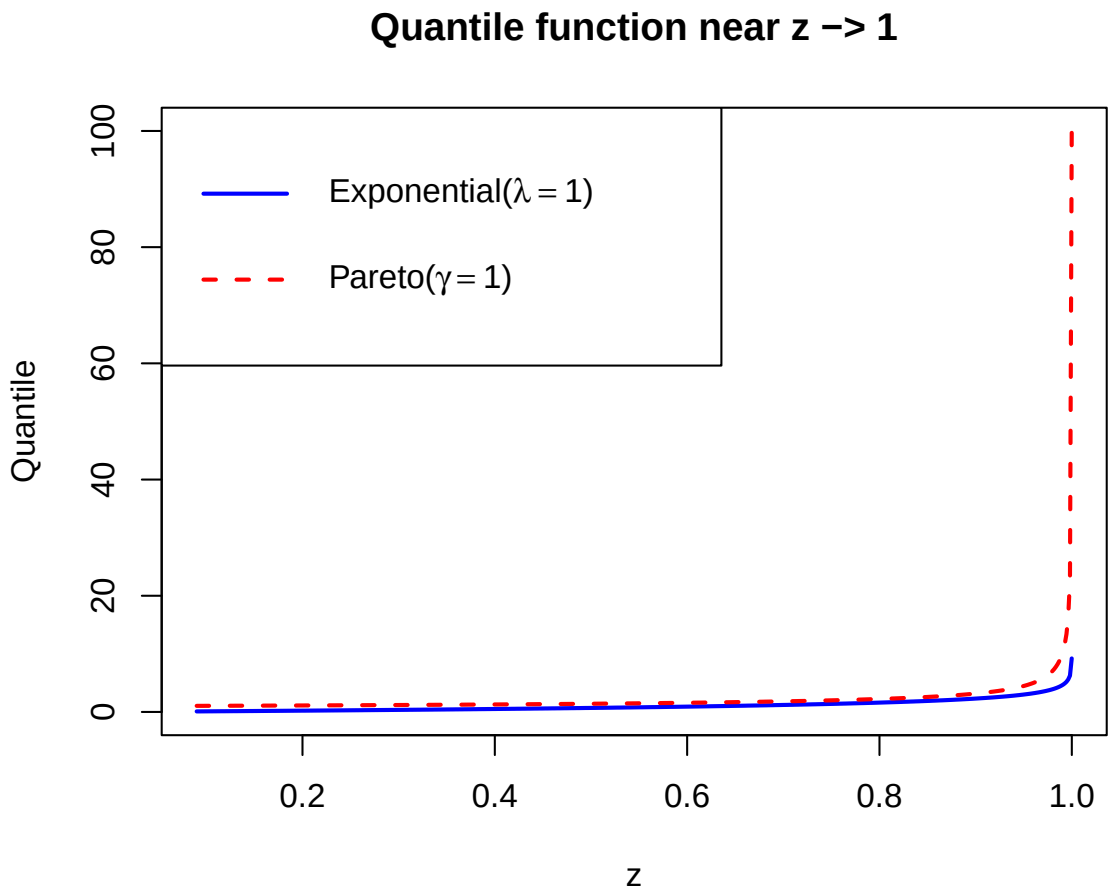
$$1 - F_X(x) = x^{-1/\gamma}\mathcal{L}_X(x), \quad x \to \infty$$

  with $\mathcal{L}_X$ a slow varying function $\mathcal{L}(tx)/\mathcal{L}(x) \to 1$ for all $t > 0$.
- **Tail index** $\gamma$ controls the tail: the larger the heavier.
- **Property of class** $\mathcal{RV}$: $F_X^{-1}(z) \sim (1-z)^{-\gamma}\ell(1/(1-z))$.
- And $F_X^{-1}(z) \to \infty$ as $z \to 1$ (fast).
- **Light-tailed**: Exponential distribution $1 - F(x) = e^{-\lambda x}$ and $F^{-1}(z) = -log(1-z)/\lambda$. $F_X^{-1}(z) \to \infty$ as $z \to 1$ (slow).

# Why is this a problem?

## Simulating extremes

- For light-tailed the quantile function diverges to infinity **slowly** compared to heavy tails.



# Simulating extremes with GANs

## Financial Illustration from Chapter

- Data: daily **log-returns** of financial stock market indices.
- Six **indices**: NKX (Nikkei, Japan), KOSPI (Korea), HSI (Hong-Kong), CAC (France), AMX (Amsterdam Exchange, Netherlands), Nasdaq (USA).
- GAN is assessed on:

Table: Tail index estimates from GAN approach applied to real financial data.

| **Index** | **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 |

# Simulating extremes with GANs

## Financial Illustration from Chapter

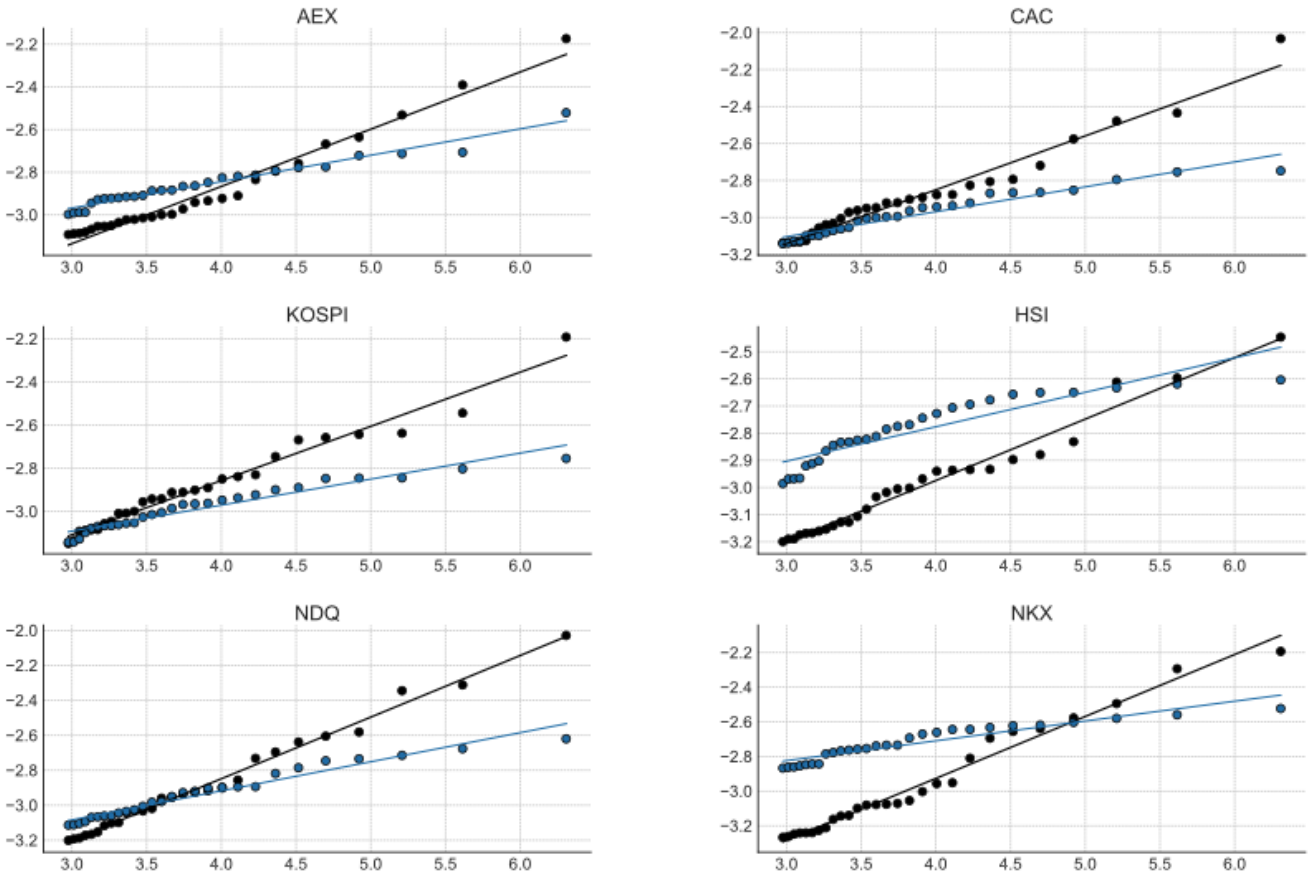



Figure: Log quantile-quantile plots from GAN approach $\xi = 0.95$ (black: real data, blue: GAN data). *Figure from Handbook on Statistics of Extremes.*

# Improvements of GANs

- Some ideas:
  - Pre-processing of the data to get rid of the tail heaviness,
  - Use of heavy-tailed latent variables.
  - **Adapt the optimization problem to the heavy-tail situation.**
- EV-GAN and Tail-GAN

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

  is continuous, bounded on $[0,1]$ and tends to the tail index as $z \to 1$.
- Theoretical **guarantee** that a neural network could uniformly approximate $G$.
- But usually not differentiable at $z = 1$.

# EV-GAN

- 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 $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_6)^{\mathrm{T}}$ is to be estimated, and $\{e_1, \ldots, e_6\}$ are universal functions. 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

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

- Leading to

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

where $\tilde{G}_{\boldsymbol{\theta},\boldsymbol{\beta}}$ is the enriched version of the generator $G_{\boldsymbol{\theta}}$ defined as

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

# EV-GAN

- The **optimization** process is the **same** as before.
- The only difference comes from the **generator's design**.
- EV-GAN Generator
  1. Require: Trained parameters ( $\boldsymbol{\theta}, \boldsymbol{\beta}$ )
  2. Generate latent variable $\mathbf{Z}$ in dimension $q \geq D$, from $U(0,1)$
  3. for each marginal $d = 1 : D$ do
  4. $G_{\boldsymbol{\theta},\boldsymbol{\beta}}^{\text{EVGAN},(d)}(\mathbf{Z}) = \varphi\left(Z_d\right)^{-\left\{G_{\boldsymbol{\theta}}(\mathbf{Z})+\sum_{j=1}^{6}\beta_j e_j Z_d\right\}}$
  5. end for
- The generator is:

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

# Simulating extremes with GANs

## Financial Illustration from Chapter

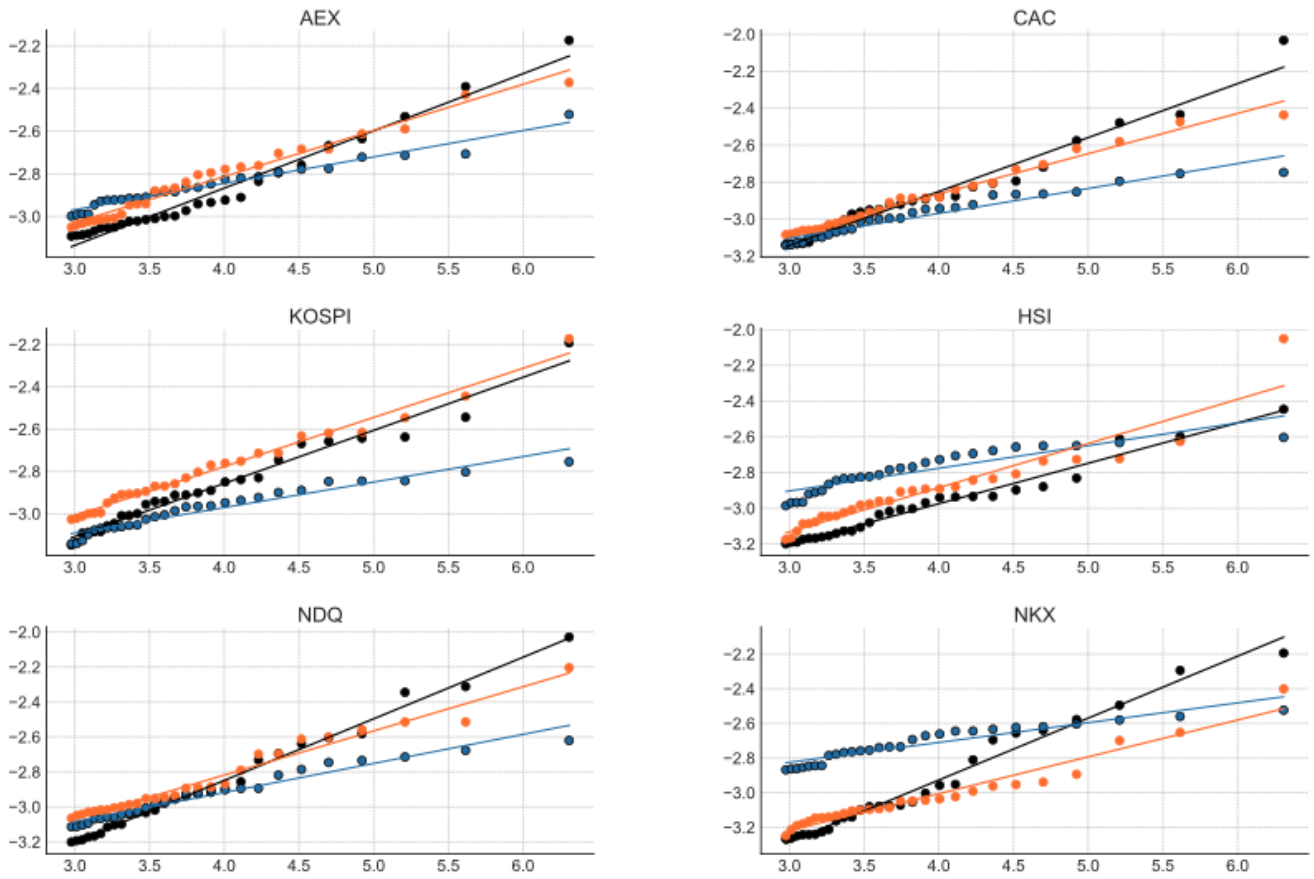



Figure: Log quantile-quantile plots from GAN approach $\xi = 0.95$ (black: real data, blue: GAN data, orange: EV-GAN). *Figure from Handbook on Statistics of Extremes.*

# Closing Remarks

## Final Comments

- When the **target random quantity** is supposed to be **heavy-tailed**, the usual generative models **cannot reproduce** this property without dedicated architecture improvements.
- For example EV-GAN.
- The chapter also discussed improvements for **Variational Auto-Encoders** and **Difussion Models**.

# Based on Chapter from Handbook on Statistics of Extremes

**This talk is based on:**

Chapter 20 – *"On the Simulation of Extreme Events with Neural Networks"*

by Michaël Allouche, Stéphane Girard & Emmanuel Gobet

In *Handbook on Statistics of Extremes*