ISSN 2281-4299

DIPARTIMENTO DI INGEGNERIA INFORMATICA AUTOMATICA E GESTIONALE ANTONIO RUBERTI

SAPIENZA UNIVERSITÀ DI ROMA

Value at Risk and Expected Shortfall Estimation with Generative Adversarial Networks

Alberto De Santis
Ioannis Seghios Dori

Technical Report n. 04, 2023

Value at Risk and Expected Shortfall estimation with Generative Adversarial Networks

Ioannis Serghios Dori[*]
Rome, Italy
ioannisdori@gmail.com

Alberto De Santis [†]
Sapienza University of Rome
Rome, Italy
desantis@diag.uniroma1.it

Abstract

In recent years, the field of Quantitative Finance has witnessed significant advancements in the application of Machine Learning (ML) modelling techniques, as they offer greater expressive power than traditional methods, reduce the need for domain experts and generally rely on less rigid assumptions. Generative models such as Generative Adversarial Networks (GANs) learn the underlying distribution of financial time series in an unsupervised fashion, generating meaningful synthetic data from noise and could greatly enrich the tools of financial engineers. We give an overview of two widely adopted methods to compute Value at Risk (VaR) and Expected Shortfall (ES), namely Historical Simulation (HS) and Peaks Over Threshold (POT) and propose a simple algorithm to compute such estimates using GANs, qualitatively comparing the methods on data simulated from two well known stochastic processes with promising results, especially for yearly time horizons.

Keywords: Generative Adversarial Networks, Value at Risk, Quantitative Finance.

[*]MSc graduate in Management Engineering - Department of Computer, Control and Management Engineering "Antonio Ruberti", Sapienza University of Rome.

[†]Sapienza University of Rome - Department of Computer, Control and Management Engineering "Antonio Ruberti".

1 Introduction

Value at Risk (VaR) and Expected Shortfall (ES) provide single measures of the potential loss of a portfolio of financial assets. Given a confidence level α the former estimates the α-quantile of the loss distribution over a specified time horizon while the latter takes into account the expected loss conditioned on exceeding the VaR.

The VaR framework initially developed by J.P. Morgan in the late 1980s and made public in the 1990s with the software RiskMetrics [1], has been widely adopted both by risk managers and regulators, especially for capital requirements in the Banking and Insurance sectors. Accurately estimating the capital requirements of such institutions is of paramount importance for a healthy economy: on one hand, it offers protection against financial crises and on the other, it enables growth opportunities by promoting more efficient resource allocation, i.e. lowering the budget for risk mitigation.

Although VaR is harshly criticized (e.g. ES could be regarded as more informative), it is the risk measure upon which many regulations are based, such as the Basel Accords (I – III) issued by the Basel Committee on Banking Supervision (BCBS) which laid the foundations for analogous policies worldwide and the more recent Solvency II directive issued by the European Insurance and Occupational Pensions Authority (EIOPA).

Initially, standard VaR estimation techniques were used, however, since more refined modelling tools have been developed, both the international Basel III Accord and the European Solvency II directive currently allow regulated institutions to use internal models tailored to their own risk profile [7, 30].

To be employed, an internal model for VaR estimation must be approved by the regulator, the validation process mainly entails computing daily VaR estimates and then performing a backtesting procedure on historical data, e.g. Kupiec's Proportion of Failures (POF) [29] or Christoffersen's test [12]. The daily VaR can then be extrapolated to longer temporal horizons; the easiest method to do so, explicitly mentioned in Basel II regulation [2], is to multiply the daily estimate by the square root of the time frame's duration[1]. The extrapolation method, as well as any other a priori assumption, must also be tested.

Given the importance of accurately assessing risk measures, the problem of estimating and forecasting VaR has extensively been studied and several methodologies have been developed for its computation which can be broadly categorized as follows (see [3] for a comprehensive review).

- Non-parametric: methods like Historical Simulation (HS) and Kernel Density Estimation (KDE), empirically estimating the return distribution's quantiles from the available data, i.e. without specifying a distribution for the risk factor returns nor the dynamics of the risk factor.
- Parametric: methods such as RiskMetrics or the ones based on the more advanced Generalized Autoregressive Conditional Heteroscedastic (GARCH) model [11] which specify the dynamics of the risk factor and make distributional assumptions on the errors in order to estimate the quantiles of the innovation's distribution.
- Semi-parametric: methods making assumptions about the risk factor dynamics and then using Monte Carlo (MC) simulation to estimate the quantiles of the return distribution or fitting a GARCH model to the returns and then using HS on bootstrapped standardized residuals (Filtered HS [8]); methods based on Extreme Value Theory (EVT), i.e. making assumptions on the limiting distribution of extreme returns observed over a long period of

[1]Assuming that the risk factors' returns are independent and identically distributed (i.i.d.) over time, that there is a linear relationship between the risk factors' returns and the portfolio's returns and that the volatility remains constant [25].

time, e.g. fitting a Generalized Pareto Distribution (GPD), see [3, 38] for more details; methods directly modelling the quantile instead of the whole distribution via an Autoregressive (AR) process as in Conditional Autoregressive Value at Risk (CAViaR) [16].

From their review of the existing literature [3], Abad et al. conclude that the methodologies that seem to perform best overall are the ones that make the least distributional assumptions on the returns, i.e. the ones based on EVT and Filtered HS and CAViaR, while the GARCH family performs well when asymmetric and fat-tail distributions are involved.

Recently, with the increasing availability of computational means, considerable effort has been devoted to applying Machine Learning (ML) models to time series modelling with the ultimate goal of replacing more traditional ones, e.g. the GARCH family. Recurrent Neural Networks (NNs) such as Long Short-Term Memory (LSTM) and Convolutional Neural Networks (CNNs) are deemed to be state-of-the-art for price prediction and trend forecasting; although these architectures have successfully been employed in both supervised and unsupervised settings, accurate predictions are currently limited to short-term time horizons of days or weeks [39].

Instead of using forecasting ML models, we propose the use of generative models, Generative Adversarial Networks (GANs) [23] in particular, to learn the underlying data distribution in an unsupervised fashion. The most straightforward application is to learn to generate realistic paths for the risk factors without having to specify their stochastic dynamics and then use MC simulation to approximate the VaR. GANs could, in principle, be employed in other parametric or semi-parametric methods to learn the underlying distributions without having to specify their family, e.g. learning the distribution of the errors after fitting a GARCH model or the limiting distribution of extreme events in the EVT framework, assuming there is enough data to do so.

Using GANs to model the risk factors' paths, provided the model correctly captures the data's distribution, has many advantages:

- the VaR could be estimated for any time horizon by simulating the losses over such period directly, without the need for an extrapolation method;
- the model could be trained on portfolio returns, eliminating the need to explicitly capture the correlation between its components;
- since GANs are not AR models, once the NN is trained, simulation can easily be parallelized.

Sampling from GANs is straightforward, however, training them is not; as we will see, the main disadvantages are due to training instability. While this is still an open problem, we introduce a simple pre-training algorithm and a stopping criterion, which help mitigate this issue.

In this work, we will focus on three different methods to compute VaR and ES, namely HS, the EVT and the GAN approach: HS, being the simplest method, is still widely adopted, whereas approaches based on EVT seem to outperform other methodologies for very high quantiles ($\alpha \geq 0.99$) [38].

Instead of using backtesting procedures, we will compare the results in a controlled environment by applying the techniques to simulated daily prices from two known stochastic processes, the Geometric Brownian Motion (GBM) and Merton's Jump-Diffusion (JD) process [31]. This choice is motivated by the following considerations: the results of statistical tests highly depend on the number of observations, therefore the reliability of Kupiec's POF test could suffer considerably for longer time horizon VaR estimates[2], moreover GANs require a nonnegligible amount of time to train, therefore it would be very expensive to perform such procedures.

[2]This is the reason why regulations such as Basel Accords require backtesting on daily estimates.

2 Backround

2.1 Value at Risk and Expected Shortfall: Historical Simulation and Peaks Over Threshold

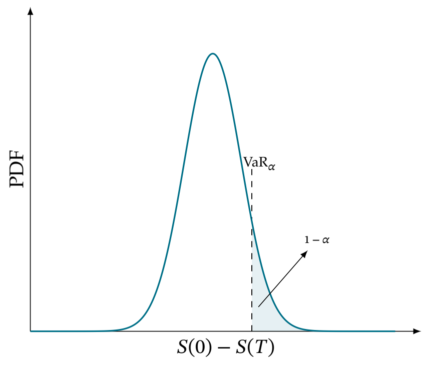
In order to discuss the two methodologies used for our comparison, we shall first formally define both VaR and ES, following [5, 38].

Definition 1 (Value at Risk) *Given a confidence level α and a time horizon T, the VaR is the α-quintile of the loss distribution, calling the loss $L := L(T) = S(0) - S(T)$ the VaR is such that:*

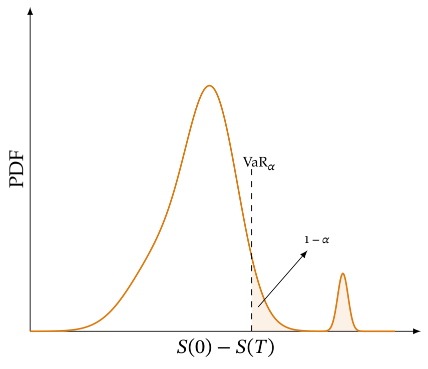
$$\mathrm{VaR}_\alpha^T(L) := \inf\{l \in \mathbb{R} : \mathbb{P}(L > l) \le 1 - \alpha\} = \inf\{l \in \mathbb{R} : F_L(l) \ge \alpha\} = F_L^{-1}(\alpha) \qquad (1)$$

where S is a risk factor, e.g. the price of a risky asset or portfolio and $F_L(l)$ is the Cumulative Distribution Function (CDF) of L.

The VaR is criticized for many reasons, one of them being the fact that two risky assets may have the same VaR but very different expected losses, as qualitatively described in figure 1.



(a)



(b)

Figure 1: Two different assets (a) and (b) share the same VaR, but (b) has higher expected losses.

To solve this issue, instead of the VaR, one could consider the ES which is the expected value of the right tail of the loss distribution.

Definition 2 (Expected Shortfall) *Given a confidence level α and a time horizon T the ES is the expected value of the loss L conditioned on exceeding the VaR:*

$$\mathrm{ES}_\alpha(L) = \mathbb{E}[L | L \ge \mathrm{VaR}_\alpha(L)] = \frac{1}{1-\alpha} \int_\alpha^1 \mathrm{VaR}_u(L)\,\mathrm{d}u. \qquad (2)$$

2.1.1 Historical Simulation

The most straightforward method to estimate these quantities is Historical Simulation, which consists in estimating the quantiles of the empirical loss (or profit) distribution from historical data. Suppose we have N loss observations over a time horizon T: $L_i^T,\ i = 1, \ldots, N$ (e.g. negative daily log-returns) and the corresponding order statistic such that $L_{(1)} \ge L_{(2)}^T \ge \cdots \ge L_{(N)}^T$.

We can estimate the α-quantile by taking $\widehat{\mathrm{VaR}}_\alpha^T = \left\{ L_{(i)}^T : \lfloor \frac{i}{N} \rfloor = 1 - \alpha \right\}$, the HS procedure to estimate VaR and ES is summarized and visualized in algortithm 1 and figure 2 respectively.

Algorithm 1. Historical Simulation

```
Input: A dataset of historical losses $\mathcal{D} = \{L_i\}_{i=1}^{N}$;
       the confidence level $\alpha$
1 Sort the losses such that $L_{(1)} \geq L_{(2)} \geq \cdots \geq L_{(N)}$
2 $\widehat{\mathrm{VaR}}_\alpha(L) \coloneqq \left\{L_{(i)} : \left\lfloor \frac{i}{N} \right\rfloor = 1 - \alpha\right\}$
3 $\widehat{\mathrm{ES}}_\alpha(L) \coloneqq \frac{1}{J}\sum_{i=1}^{J} L_{(i)}, \quad J : L_{(J)} = \widehat{\mathrm{VaR}}_\alpha$
return $\widehat{\mathrm{VaR}}_\alpha(L), \widehat{\mathrm{ES}}_\alpha(L)$
```

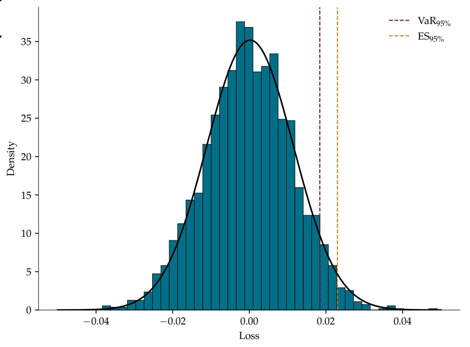


Figure 2: HS applied to normally distributed daily log-returns.

Due to its simplicity and low computational cost, HS is the most widely adopted non-parametric approach [3]. The main advantages of this method are: that it does not depend on assumptions about the loss distribution, therefore it can accommodate fat tails, skewness and any other non-normal features observed in financial time series; it can be easily paired with parametric approaches (Filtered HS) or modified to increase its accuracy and provide confidence intervals (e.g. applying it on bootstrapped losses). However, it also has disadvantages: it is highly dependent on the dataset, e.g. if the period in consideration is exceptionally volatile or includes extreme losses which are unlikely to recur, this could lead to overestimation; it cannot take into account plausible events that have not yet occurred in our sample period and it is slow to reflect changes (see [3, 14]).

2.1.2 Peaks Over Threshold

The second approach we consider in this work, closely following[14, 38], is based on EVT which studies the asymptotic distribution of extreme losses observed over a long time. The theoretical foundations of EVT were laid in [19, 21] in which it was proven that the distribution of extreme (maximum) values of an i.i.d. sample from most known CDFs, appropriately rescaled, converge to one of three possible families: Fréchet, Gumbel and Weibull which were unified under the Generalized Etreme Value (GEV) distribution family [26]. The GEV depends on a *shape* parameter $\xi \in \mathbb{R}$ so that it corresponds to either the Fréchet ($\xi > 0$), the Gumbel ($\xi = 0$) or the Weibull ($\xi < 0$) family.

In this context, three different methods may be identified:

- the Block Maxima Method (BMM), in which the data is divided into blocks of n observations each and the GEV distribution is fitted on the sequence of block maxima, e.g. using Maximum Likelihood Estimation (MLE). The main disadvantage is that the data use is not efficient since extreme values falling in the same block would be discarded while only the block maximum would be kept.

- The Peaks Over Threshold (POT) method, where the extreme values in a sample are chosen to be the ones that exceed a specified threshold u. The excess distribution over the threshold for most CDFs has, as a limiting distribution, the GPD, which can be proven to share the same shape parameter as the GEV for high values of u; i.e. those CDFs whose excess distribution over the threshold converges to a GPD with shape parameter ξ, are the same ones which lie in the maximum domain of attraction of a GEV distribution with the same shape parameter ξ. Similarly to the BMM the GPD can be fitted to the right tail using MLE.

- Non-parametric approaches where the shape parameter ξ of the GPD (or GEV) is estimated directly from the available data, e.g. using the Hill estimator.

Among these methods, the POT model is considered to be the most useful for practical applications due to the more efficient use of the data for extreme values compared to the BMM [3], therefore it is the approach we will consider for our comparison.

By definition, the excess distribution over the threshold u of a random variable L, i.e. $X = L - u$, with CDF F, is

$$F_u(x) := \mathbb{P}(L-u \leq x | L > u) = \frac{\mathbb{P}(0 < L-u \leq x)}{\mathbb{P}(L > u)} = \frac{F(x+u) - F(u)}{1 - F(u)},\ 0 \leq x \leq x_F - u \quad (3)$$

where $x_F = \sup\{x \in \mathbb{R} : F(x) < 1\}$ and its asymptotic distribution distribution, for most known CDFs, is the GPD with shape parameter ξ and scale parameter $\beta > 0$:

$$\mathrm{GPD}_{\xi,\beta}(x) := \begin{cases} 1 - \left(1 + \xi \frac{x}{\beta}\right)^{-1/\xi}, & \xi \neq 0 \\ 1 - \mathrm{e}^{\frac{x}{\beta}}, & \xi = 0 \end{cases}, \text{ where } \begin{cases} x \geq 0, & \text{if } \xi \geq 0 \\ 0 \leq x \leq {-\beta}/{\xi}, & \text{if } \xi < 0 \end{cases}. \quad (4)$$

In extreme value analysis, the cases in which we are usually interested are the ones where the shape parameter ξ, also referred to as tail index, is nonzero and particularly when $\xi > 0$ as it corresponds to heavy-tail behaviour.

Rearranging the right-hand side of equation (3) we can move from the excess distribution over the threshold to the parent distribution $F(l)$ defined over the losses [14], assuming the limiting distribution of $F_u(x) \approx \mathrm{GPD}_{\xi,\beta}(x)$ where $x = l - u$ we have:

$$F_u(l-u) = \frac{F(l) - F(u)}{1 - F(u)} = \mathrm{GPD}_{\xi,\beta}(l-u) = 1 - \left(1 + \frac{\xi}{\beta}(l-u)\right)^{-1/\xi}$$

therefore

$$F(l) = 1 - \Big(1 - F(u)\Big)\left(1 + \frac{\xi}{\beta}(l-u)\right)^{-1/\xi}. \quad (5)$$

We can now estimate the VaR by inverting equation (5), i.e. finding $l = F^{-1}(\alpha)$:

$$\alpha = 1 - \Big(1 - F(u)\Big)\left(1 + \frac{\xi}{\beta}(l-u)\right)^{-1/\xi}$$

recalling definition 1, we obtain

$$\mathrm{VaR}_\alpha(L) = F^{-1}(\alpha) = u + \frac{\beta}{\xi}\left[\left(\frac{1-\alpha}{1-F(u)}\right)^{-\xi} - 1\right]. \quad (6)$$

Using definition 2, the ES estimate can be easily obtained by integrating equation (6):

$$\mathrm{ES}_\alpha(L) = \frac{1}{1-\alpha}\int_\alpha^1 \mathrm{VaR}_u(L)\,\mathrm{d}u = \frac{\mathrm{VaR}_\alpha(L)}{1-\xi} + \frac{\beta - \xi u}{1-\xi}. \quad (7)$$

Before we outline the POT method there are still some aspects we have to clarify regarding the unknowns: how to choose the threshold u, how to estimate the parameters $\hat{\xi}$ and $\hat{\beta}$ of the GPD and how to estimate the tail distribution of the losses $1 - F(u)$.

The threshold u must be chosen so that it is high enough for the EVT theorems to hold, i.e. for the GPD to be the limiting distribution of the excess losses regardless of the distribution of the losses; however, it must be low enough to include a reasonable amount of observations for reliable estimates.

A widely used criterion is to choose u so that the tail is populated by no more than 10-15% of the data and, as a rule of thumb, the value of 5-10% is often used [38].

Given a dataset $\mathcal{D} = \{L_i\}_{i=1}^N$ of i.i.d. samples, once u is chosen, the parameters ξ and β can be estimated by fitting, via MLE, the GPD on the sequence $\left\{X_i\right\}_{i=1}^{N_u}$, where $X_i = \widetilde{L}_i - u$ are the data exceeding the threshold and N_u is the total number of excesses.

The log-likelihood for the case where $\xi \neq 0$ can be written as [14]

$$\ell(\xi, \beta) = -N_u \ln \beta - (1 + {}^1\!/\!_\xi) \sum_{i=1}^{N_u} \ln\left(1 + \xi \frac{X_i}{\beta}\right) \tag{8}$$

and the parameters can be estimated by maximizing equation (8) subject to the constraints under which the GPD is defined:

$$\begin{aligned} (\hat{\xi}, \hat{\beta}) = \operatorname{argmax} \quad & \ell(\xi, \beta) \\ \text{s.t.} \qquad & \beta > 0 \\ & 1 + \frac{\xi}{\beta} X_i > 0 \end{aligned} \quad . \tag{9}$$

To compute the VaR and the ES we also need to estimate $1 - F(u)$ and the most natural estimator is the empirical proportion of observations over the threshold, i.e. $\frac{N_u}{N}$. Now that all the quantities involved have been specified, we can finally outline the POT method for VaR and ES estimation.

Algorithm 2. Peaks Over Threshold

```
Input: A dataset of historical losses $\mathcal{D} = \{L_i\}_{i=1}^N$; the confidence level $\alpha$
1 Choose a threshold $u$ such that $\frac{N_u}{N} \leq 15\%$ and compute the excesses $\left\{X_i\right\}_{i=1}^{N_u}$, $X_i = \widetilde{L}_i - u$
2 Assuming that the data are in the maximum domain of attraction of a GEV distribution, estimate
  the parameters of the corresponding GPD, $(\hat{\xi}, \hat{\beta})$ solving problem (9)
3 Estimate $1 - F(u) \coloneqq \frac{N_u}{N}$
4 Compute the VaR estimate according to equation (6):
      $\widehat{\mathrm{VaR}}_\alpha(L) \coloneqq u + \frac{\hat{\beta}}{\hat{\xi}}\left[\left(\frac{N}{N_u}(1-\alpha)\right)^{-\hat{\xi}} - 1\right]$
5 Compute the ES estimate according to equation (7):
      $\widehat{\mathrm{ES}}_\alpha(L) \coloneqq \frac{1}{1-\alpha}\int_\alpha^1 \mathrm{VaR}_u(L)\,\mathrm{d}u = \frac{\mathrm{VaR}_\alpha(L)}{1-\hat{\xi}} + \frac{\hat{\beta} - \hat{\xi}u}{1-\hat{\xi}}.$
return $\widehat{\mathrm{VaR}}_\alpha(L), \widehat{\mathrm{ES}}_\alpha(L)$
```

While this method still greatly depends on key assumptions such as having a dataset representing extreme events sufficiently well and that the asymptotic distribution of the excesses tends to a GPD, it no longer constrains us to a particular family for the distribution of the losses by exclusively modelling the tail behaviour and leads to satisfying results, especially when considering very high quantiles ($\alpha \geq 0.99$).

2.2 Generative Adversarial Networks

Generative unsupervised learning refers to any learning process explicitly or implicitly modelling the true unknown data distribution $\mathbb{P}(\boldsymbol{x})$. There are many different ML models, Deep Learning (DL) ones [24] in particular, that attempt to do so, some of which are based on Latent Variable Models (LVMs) and require some form of inference during training, sample generation or both, e.g. Variational Autoencoders (VAEs) [28].

GANs were introduced by Goodfellow et al. [23] with the main goal of avoiding explicit inference steps during training and sample generation. Relying on an adversarial process in which a generative model G competes with a discriminative one D which estimates the probability that a sample came from the training data rather than G, this framework eliminates the need to train an approximate inference model such as in VAEs and does not require complicated and computationally expensive sampling methods such as Markov Chain Monte Carlo (MCMC) in denoising Autoencoders (AEs) or ancestral sampling in Fully Visible Belief Networks (FVBNs) [24].

GANs share the same LVM as VAEs (in figure 3) where the generative model is given by $p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z}) = p(\boldsymbol{z})p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$, but instead of choosing the likelihood $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$ among a family of distributions and parametrizing it with a NN, the likelihood is implicitly defined by a Generator network $G(\boldsymbol{z}; \boldsymbol{\theta}_G)$ representing a deterministic, differentiable mapping from latent to data space $G : \boldsymbol{z} \mapsto \boldsymbol{x}$.

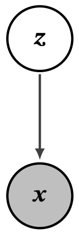


Figure 3: GAN and VAE LVM.

Therefore we have the empirical distribution $p_G(\boldsymbol{x}|\boldsymbol{z}) := \delta(\boldsymbol{x} - G(\boldsymbol{z}))$ where δ is Dirac's delta and the marginal likelihood is given by [13, 42]:

$$p_G(\boldsymbol{x}) = \int p(\boldsymbol{z})p_G(\boldsymbol{x}|\boldsymbol{z})\, \mathrm{d}\boldsymbol{z}. \tag{10}$$

The objective is to train the generative model so that it approximates the true data distribution $p_G(\boldsymbol{x}) \approx p_{\text{data}}(\boldsymbol{x})$. In order to do so, the Discriminator $D(\boldsymbol{x}; \boldsymbol{\theta}_D)$ is introduced, a classifier which outputs a scalar $D : \boldsymbol{x} \mapsto y \in [0, 1]$ representing the probability that the input $\boldsymbol{x}$ comes from the data-generating process $p_{\text{data}}(\boldsymbol{x})$ $(y = 1)$ rather than $p_G(\boldsymbol{x})$ $(y = 0)$. Both models are then trained to compete against each other in a minimax game

$$\min_{\boldsymbol{\theta}_G} \max_{\boldsymbol{\theta}_D} V(D, G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}\Big[\ln D(\boldsymbol{x})\Big] + \underbrace{\mathbb{E}_{\boldsymbol{x}\sim p_G(\boldsymbol{x})}\Big[\ln\big[1 - D(\boldsymbol{x})\big]\Big]}_{\mathbb{E}_{\boldsymbol{z}\sim p(\boldsymbol{z})}\left[\ln\left[1 - D\left(G(\boldsymbol{z})\right)\right]\right]} \tag{11}$$

where the Discriminator maximizes the probability of assigning the correct label to its input, i.e. the log-likelihood for estimating the conditional probability $\mathbb{P}(Y = y|\boldsymbol{x})$, whereas the Generator is trained to minimize it [23]. In short, the structure of a GAN, shown in figure 4, is composed by:

- a prior $p(\boldsymbol{z})$ on the latent variable $\boldsymbol{z}$ from which to sample noise vectors;
- the Generator $G(\boldsymbol{z}; \boldsymbol{\theta}_G)$: a NN producing synthetic data from noise input, representing a differentiable mapping from latent to data space $G : \boldsymbol{z} \mapsto \boldsymbol{x}$ which implicitly defines the probability distribution $p_G(\boldsymbol{x})$ of the samples $G(\boldsymbol{z})$;
- the Discriminator $D(\boldsymbol{x}; \boldsymbol{\theta}_D)$: a differentiable classifier that outputs a scalar in $[0, 1]$ representing the probability that a sample $\boldsymbol{x}$ comes from the data-generating process $p_{\text{data}}(\boldsymbol{x})$ rather than $p_G(\boldsymbol{x})$.

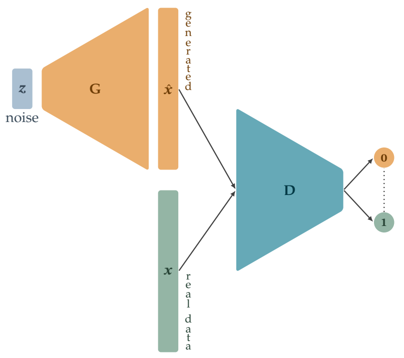


Figure 4: General structure of a GAN

Goodfellow et al. [23] prove that given enough capacity to both the Generator and the Discriminator, i.e. in the non-parametric limit, and given that we have access to the true data-generating process, the global optimum of problem (11) is achieved when $p_G(\boldsymbol{x}) = p_{\text{data}}(\boldsymbol{x})$ [23]. The training criterion for the Generator can be rewritten as:

$$C(G) = -\ln 4 + 2\,\mathbb{JS}(p_{\text{data}}(\boldsymbol{x})\|p_G(\boldsymbol{x}))$$

where $\mathbb{JS}$ is the Jensen-Shannon divergence (JS divergence), a symmetric version of the Kullback–Leibler divergence (KL divergence):

$$\mathbb{JS}(\mathbb{P}\|\mathbb{Q}) \coloneqq \frac{1}{2}\,\mathbb{KL}\left(\mathbb{P}\middle\|\frac{\mathbb{P}+\mathbb{Q}}{2}\right) + \frac{1}{2}\,\mathbb{KL}\left(\mathbb{Q}\middle\|\frac{\mathbb{P}+\mathbb{Q}}{2}\right)$$

with $0 \leq \mathbb{JS}(\mathbb{P}\|\mathbb{Q}) \leq 1$ and equal to zero if and only if $\mathbb{P} = \mathbb{Q}$, therefore the training criterion is theoretically sound as solving problem (11) drives $p_G \to p_{\text{data}}$.

In practice, however, even when the training dataset is sufficiently representative of the true data-generating process, GANs can parametrize limited[3] families of distributions. Moreover, when using deep NNs, solving problem (11) requires finding a Nash equilibrium of a non-convex high-dimensional game, which is not trivial.

Goodfellow et al. [23] proposed to use minibatch Stochastic Gradient Descent (SGD) to optimize Discriminator and Generator alternately in order to find an approximate solution[4] to such problem, arguing that if G changes slowly enough, D is maintained near its optimum; the procedure is outlined in algortithm 3.

Arjovsky and Bottou [4] prove that the closer the Discriminator is to optimality, the more the Generator's loss saturates, resulting in a weak or even vanishing gradient when the optimal Discriminator is perfectly able to classify the data. While no formal proof was given, the problem was known since the original paper [23], being especially evident early in training when G is poor and the Discriminator can reject generated samples with high confidence.

To address the issue, Goodfellow et al. [23] modified the loss so that the Generator, instead of minimizing the log-probability of the Discriminator being correct, maximizes the log-probability of it being mistaken $L_G^{\text{NS}}(\boldsymbol{z})$ [22]:

$$L_G(\boldsymbol{z}) = \ln[1 - D(G(\boldsymbol{z}))] \quad \Rightarrow \quad L_G^{\text{NS}}(\boldsymbol{z}) = -\ln D(G(\boldsymbol{z})). \tag{12}$$

This heuristically motivated non-saturating version of the game between D and G is no longer zero-sum and, while lacking a theoretical basis, it allows GANs trained with algortithm 3 to produce very high-quality samples compared to other generative models.

[3]Even when the universal approximation theorem applies the architecture needed for the model to be identifiable could be infeasible.

[4]The authors prove that the algorithm converges in the non-parametric limit since the logarithm is strictly concave and expectations preserve convexity w.r.t. the distributions, but this is no longer true when D and G are NNs, known to produce highly non-convex losses [34].

Algorithm 3. GAN training with minibatch SGD

```
Input: A dataset $\mathcal{D} = \{\boldsymbol{x}^{(i)}\}_{i=1}^{N}$; the Discriminator $D(\boldsymbol{x}; \boldsymbol{\theta}_D)$ with its loss
       $L_D(\boldsymbol{x}, \boldsymbol{z}) = -\ln D(\boldsymbol{x}) - \ln[1 - D(G(\boldsymbol{z}))]$; the Generator $G(\boldsymbol{x}; \boldsymbol{\theta}_G)$ with its loss
       $L_G^{\text{NS}}(\boldsymbol{z}) = -\ln D(G(\boldsymbol{z}))$; a routine performing one minimization step of a
       gradient-based optimization algorithm optimizer_step.
for number of training epochs do
    for number of minibatches do
        for $k$ steps do
1           Select a minibatch of $m$ examples $\boldsymbol{x} = \{\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(m)}\}$ from $\mathcal{D}$ and $m$ noise
            samples $\boldsymbol{z} = \{\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(m)}\}$ from the prior $p(\boldsymbol{z})$.
2           Compute $\nabla_{\boldsymbol{\theta}_D} J(\boldsymbol{\theta}_D) = \frac{1}{m}\sum_{i=1}^{m} \nabla_{\boldsymbol{\theta}_D} L_D\left(\boldsymbol{x}^{(i)}, \boldsymbol{z}^{(i)}\right)$
3           Update the Discriminator: $\boldsymbol{\theta}_D = \text{optimizer\_step}(\boldsymbol{\theta}_D, \nabla_{\boldsymbol{\theta}_D} J(\boldsymbol{\theta}_D))$
        end
4       Sample $m$ noise vectors $\boldsymbol{z} = \{\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(m)}\}$ from the prior $p(\boldsymbol{z})$.
5       Compute $\nabla_{\boldsymbol{\theta}_G} J(\boldsymbol{\theta}_G) = \frac{1}{m}\sum_{i=1}^{m} \nabla_{\boldsymbol{\theta}_G} L_G\left(\boldsymbol{x}^{(i)}, \boldsymbol{z}^{(i)}\right)$
6       Update the Generator: $\boldsymbol{\theta}_G = \text{optimizer\_step}(\boldsymbol{\theta}_G, \nabla_{\boldsymbol{\theta}_G} J(\boldsymbol{\theta}_G))$
    end
end
```

The GAN framework offers many advantages over other generative models, mainly that training requires no inference and that sampling is straightforward (no MCMC needed) and easily parallelized.

However, GANs are notoriously hard to train and suffer from what is known as mode collapse, i.e. when the Generator learns to map several different input values to the same output, producing a subset of very similar samples [22], as shown in figure 5. The problems of training instability and mode

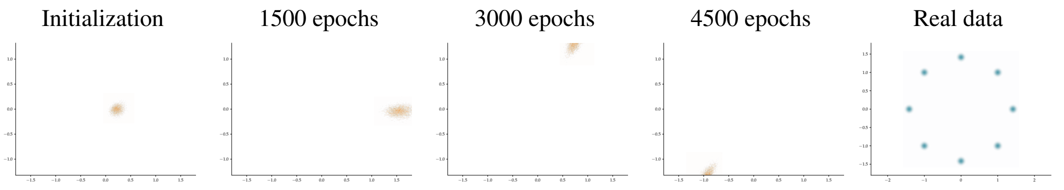


Figure 5: Example of mode collapse in a GAN applied to a 2D multimodal toy dataset.

collapse arise both when using the original loss function and the non-saturating one (equation (12)). In fact, in the first case, we are trying to find a solution (assuming it exists) to a non-convex Nash Equilibrium Problem (NEP) with an iterative optimization method that has no theoretical guarantees of convergence [22]. In the second case, Arjovsky and Bottou [4] prove that updating the Generator by minimizing the new cost function is equivalent to approximately finding

$$\operatorname*{argmin}_{p_G} \mathbb{KL}(p_G \| p_{\text{data}}) - 2\, \mathbb{JS}(p_G \| p_{\text{data}}),$$

thus directly contributing to mode-dropping with the reverse $\mathbb{KL}$ term and to training instability with the JS divergence actually driving the two distributions apart.

Despite these theoretical issues, alas leading to very practical consequences, GANs are able to produce very realistic synthetic samples when correctly fine-tuned and are, to this day, considered cutting-edge technology (see [15] for a recent review on GANs in Finance).

2.2.1 GANs in Value at Risk and Expected Shortfall estimation

The applications of GANs in Finance are relatively recent, with the first ones dealing with realistic synthetic time series generation [41]. To our knowledge, specific applications to VaR estimation are generally focused on 1-day VaR estimates by training a GAN to reproduce the log-return distribution of a financial time series.

In [18] a thoroughly fine-tuned Recurrent Conditional GAN [17] was compared both to unconditional (Variance-Covariance, HS and KDE methods) and conditional[5] VaR models (GARCH family) using several backtesting procedures including Kupiec's and Christoffersen's. The author concludes that, when precisely fine-tuned, GANs can outperform traditional models in unconditional VaR estimation; the same cannot be said for conditional VaR, where the performance is worse compared to state-of-the-art GARCH models.

Although 1-day VaR estimation for backtesting purposes is important, we believe the true strength of using GANs in VaR and ES estimation lies in longer time horizon projections. If the GAN model is able to reproduce the log-return distribution to a certain degree of accuracy, we can generate unlimited scenarios sampling from it as we would in MC simulation, with the important difference that the model from which we simulate is not specified by an expert, but rather it is learnt in an unsupervised fashion.

While the accuracy might not be sufficient to outperform prediction models on shorter time scales, it could potentially compensate for their limitations on longer ones; moreover, using such an approach, there is no need for extrapolation methods when there is not enough data to learn the N-day return distribution directly, the only limitation is that the daily distribution must be sufficiently stationary. We can now outline such a method for VaR and ES estimation in algortithm 4.

Algorithm 4. MC-GAN: VaR and ES estimation

```
Input: A GAN model; a dataset of historical daily log-returns $\mathcal{D} = \{r^{(i)}\}_{i=1}^{N}$; the confidence
       level $\alpha$; the time horizon $T$ in days
1 Train a GAN on $\mathcal{D}$ using algortithm 3
2 Sample $N_{\text{runs}} \times T$ returns from the GAN, sum over the time horizon and compute the losses
  vector:
      $\boldsymbol{L} = -\text{GAN.sample}(N_{\text{runs}}, T)\text{.sum(dim = 1)}$
3 Sort the losses such that $L_{(1)} \geq L_{(2)} \geq \cdots \geq L_{(N)}$
4 $\widehat{\text{VaR}}_\alpha(L) \coloneqq \left\{L_{(i)} : \left\lfloor \frac{i}{N} \right\rfloor = 1 - \alpha\right\}$
5 $\widehat{\text{ES}}_\alpha(L) \coloneqq \frac{1}{J} \sum_{i=1}^{J} L_{(i)}, \quad J : L_{(J)} = \widehat{\text{VaR}}_\alpha$
  return $\widehat{\text{VaR}}_\alpha(L), \widehat{\text{ES}}_\alpha(L)$
```

This method incorporates aspects of both MC simulation and HS, however, it does not share the limitations of the latter regarding the sample size.

[5]In this context, conditional refers to the conditioning of the VaR w.r.t. a specific filtration, i.e. a specific set of past information.

3 Experiments

3.1 Testing environment: synthetic datasets

We tested algorithms 1, 2 and 4 in a controlled environment by simulating daily prices of an asset S from two stochastic processes with known properties:

1. the GBM, a diffusion process with constant drift (μ) and diffusion (σ) underlying the Black-Scholes-Merton (BSM) option pricing model [10, 32]

$$\mathrm{d}S_t = \mu S_t \,\mathrm{d}t + \sigma S_t \,\mathrm{d}W_t.$$

2. Merton's JD process [31], a GBM with jumps, following Glasserman [20]

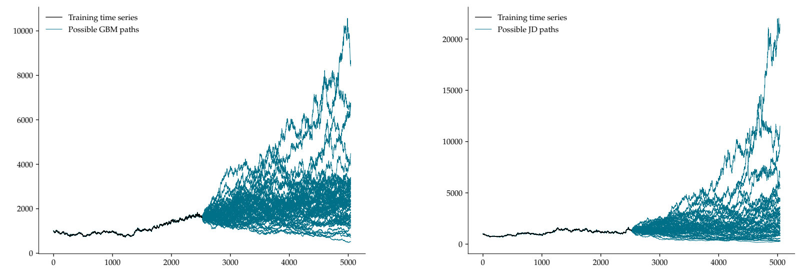
$$\mathrm{d}S_t = \mu S_{t^-} \,\mathrm{d}t + \sigma S_{t^-} \,\mathrm{d}W_t + S_{t^-} \,\mathrm{d}J_t, \quad J(t) = \sum_{i=1}^{N(t)} (Y_i - 1)$$

   where $N(t)$ is a Poisson process with rate λ (J_t is a compound Poisson process) and $Y_i \sim \mathcal{LN}(\mu_Y, \sigma_Y^2)$; we adopt the convention that the process is right-continuous, i.e. $S(t) = \lim_{\tau \downarrow t} S(\tau)$, therefore indicate with S_{t^-} the value of the price right before a jump.

In both cases[6] we used plain vanilla MC simulation (without any variance reduction techniques, e.g. antithetic variables) sampling from the analytic distributions of the log returns. It is possible to show that prices following the Stochastic Differential Equations (SDEs) above are distributed according to a lognormal distribution (GBM) and a Poisson mixture of lognormal distributions (JD) [20]:

$$\begin{aligned}
&\text{GBM: } \mathbb{P}(S_t \le x) = \Phi\left(\frac{\ln(x) - m}{s^2}\right), \quad \begin{cases} m := \ln S_0 + (\mu - \frac{1}{2}\sigma^2)t \\ s^2 := \sigma^2 t \end{cases} \\
&\text{JD: } \mathbb{P}(S_t \le x) = \sum_{n=0}^{\infty} \mathrm{e}^{-\lambda t} \frac{(\lambda t)^n}{n!} \Phi_{n,t}\left(\frac{\ln(x) - m_J}{s_J^2}\right), \quad \begin{cases} m_J := \ln S_0 + (\mu - \frac{1}{2}\sigma^2)t + n\mu_Y \\ s_J^2 := \sigma^2 t + n\sigma_Y^2 \end{cases}
\end{aligned} \tag{13}$$

where Φ indicates the Gaussian CDF. We simulated a 10-year time series of daily prices with 252 trading days per year for training and 100000 possible paths in the following 10 years for testing.



(a) GBM process　　(b) JD process.

Figure 6: Training time series and 50 possible paths 10 years in the future for an asset's price starting at $S_0 = 1000$ and following a GBM (a) and a JD (b) process respectively.

[6]For the JD process such a distribution we numerically approximated the infinite series [6] using Julia programming language [9] for the implementation.

3.2 GAN architecture and training

The architecture chosen for the following study is a Deep Convolutional GAN (DCGAN) variant implemented using PyTorch [35], given its out-of-the-box success in reproducing most of the stylized facts of the Standard & Poor's 500 (S&P 500) index time series [15].

In our implementation of DCGAN we follow the original paper's guidelines for a stable architecture Radford et al. [37] replacing the 2-dimensional convolutions with 1-dimensional ones:

1. Replace pooling layers with strided convolutions for the Discriminator and fractionally-strided (transposed) convolutions for the Generator.
2. Use Batch Normalization (BN) in both G and D.
3. Remove fully connected hidden layers for deeper architectures.
4. Use Rectified Linear Unit (ReLU) activation in the Generator and Leaky ReLU (LeakyReLU) in the Discriminator.

The complete description of the architecture used is given in tables 1 and 2 in appendix A.1.

The dataset was pre-processed using a rolling window of 32 days, up-scaled by a factor of 100, shuffled and divided into batches of size 32 for the training procedure.

The Generator was pre-trained using a denoising autoencoder [24] as it seems to significantly speed up the training and to reduce mode collapse to some degree, as described in appendix A.2. The GAN was trained using algortithm 3 on the log returns $\ln \frac{S_{t+1}}{S_t}$ of the first 10 years of simulation optimizing the weights with Adam algorithm [27], as in the original DCGAN paper. To deal with training instability, a simple stopping criterion was adopted: 100 Kolmogorov-Smirnov (KS) tests were performed each epoch and the training was interrupted if a good number of the tests were passed (usually at least 90%).

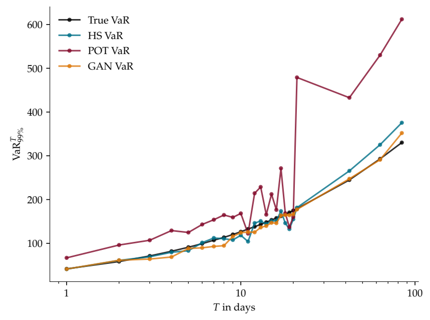
3.3 Results

Before describing the qualitative results obtained from the comparisons, there are some considerations to keep in mind regarding the POT method. We chose a setting in which the datasets are fairly i.i.d. but we have a limited amount of data (only 10 years), this penalizes the POT method which requires more observations of rare events in order to correctly capture the tail behaviour. Moreover, the analysis will be carried out on the returns rather than the log returns, which would be normally distributed in the case of the GBM. We used the library evd [40] in the R programming language [36] to fit the GPD and the threshold was chosen such that $\frac{N_u}{N} \in [5\%, 15\%]$.

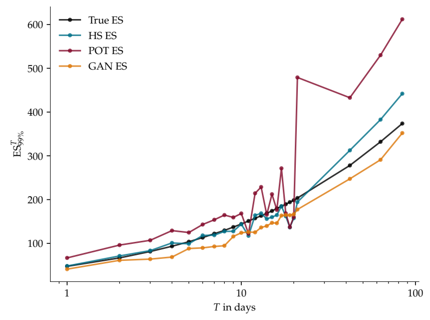
We compared algorithms 1, 2 and 4 for the estimation of both VaR and ES on the GBM and on the JD processes for different time horizons. For $T \leq 100$ days we used no extrapolation, while for longer horizons we scaled the VaR using $\sqrt{T}$ rule of thumb.

3.3.1 Geometric Brownian Motion

For the GBM process HS has slightly better performance than the GAN model and performs reasonably well for horizons up to 100 days. The POT tends to overestimate both VaR and ES especially for $T > 10$ days as shown in figure 7.



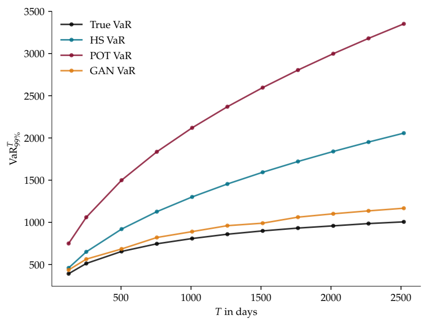
(a) VaR



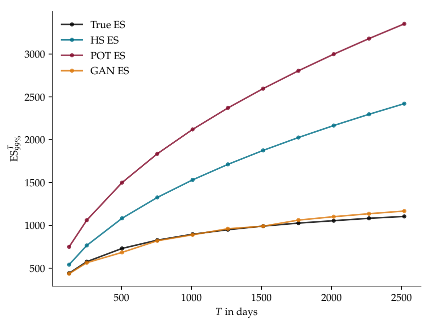
(b) ES

Figure 7: VaR and ES estimation using HS, POT and GAN methods in the daily–monthly range.

As there isn't enough data to perform HS and POT past 100 days, we necessarily have to extrapolate the results on longer time horizons, negatively impacting the estimates. The GAN method on the other hand, yields considerably better results, especially for ES estimation as shown in figure 10.



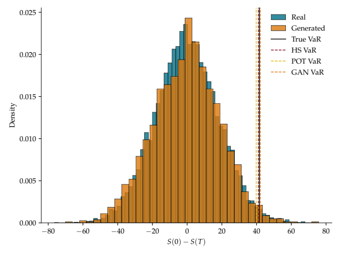
(a) VaR



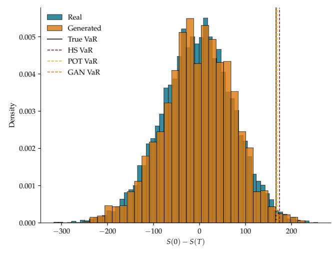
(b) ES

Figure 8: VaR and ES estimation using HS, POT and GAN methods in the yearly range.

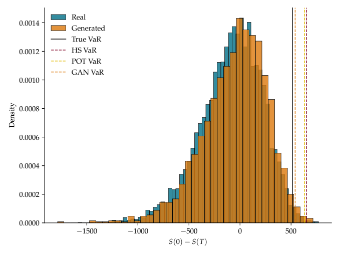
The POT method is very sensitive to the threshold choice, when tweaking it, the method seems to give better results, particularly in the daily range, as shown in figure 9. The GAN model, however, still outperforms both on longer time horizons.



(a) $\mathrm{VaR}^{\text{1-day}}_{99\%}$



(b) $\mathrm{VaR}^{\text{18-day}}_{99\%}$

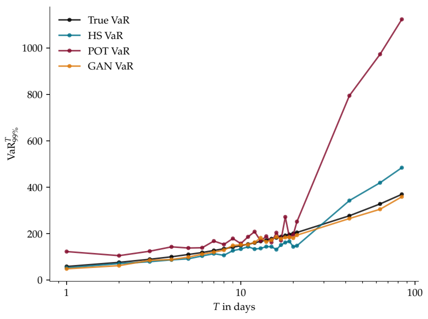


(c) $\mathrm{VaR}^{\text{252-day}}_{99\%}$

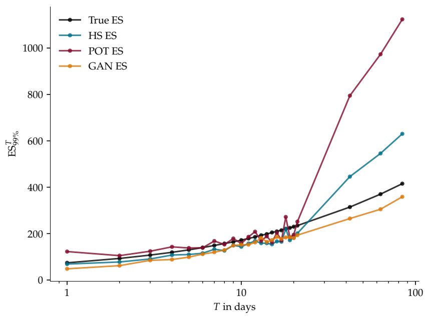
Figure 9: VaR estimation for the GBM process using HS, POT and GAN methods for $T = [1, 18, 252]$ together with the histograms of Real (simulated) and Generated losses.

3.3.2 Jump-Diffusion

For the JS divergence process HS has comparable performance to the GAN model and, similarly to the previous case, performs reasonably well for horizons up to 100 days. The POT tends to overestimate both VaR and ES, however, since the process has wider tails, it seems to perform better than the previous case, as shown in figure 10.



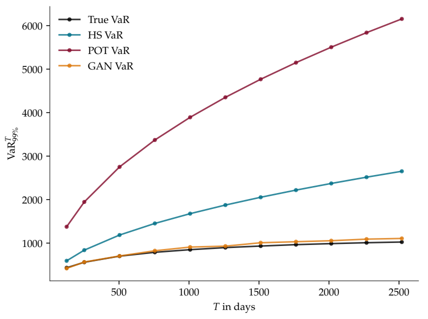
(a) VaR



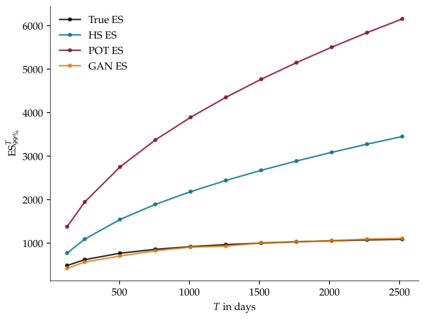
(b) ES

Figure 10: VaR and ES estimation using HS, POT and GAN methods in the daily–monthly range.

Similarly to the GBM, the GAN method, yields considerably better results on longer time horizons, especially for ES estimation as shown in figure 11.



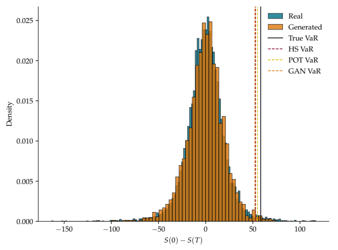
(a) VaR



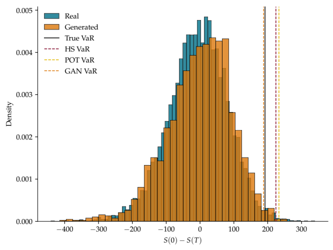
(b) ES

Figure 11: VaR and ES estimation using HS, POT and GAN methods in the yearly range.

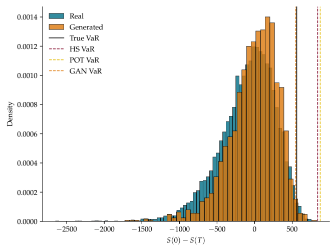
Tweaking the threshold has a positive effect also in this case, however results seem comparable to HS, as shown in figure 12.



(a) $\mathrm{VaR}^{\text{1-day}}_{99\%}$



(b) $\mathrm{VaR}^{\text{18-day}}_{99\%}$



(c) $\mathrm{VaR}^{\text{252-day}}_{99\%}$

Figure 12: VaR estimation for the JD process using HS, POT and GAN methods for $T = [1, 18, 252]$ together with the histograms of Real (simulated) and Generated losses.

4 Conclusions and future work

The results reported above are quite promising considering the fact that we learned to simulate data from a single 10-year time series in an unsupervised fashion, moreover, we used a relatively standard GAN model which could be further fine-tuned. While these results remain qualitative and more comprehensive tests should be run, they demonstrate the great potential generative models have to revolutionize the field of Quantitative Finance and show how GANs could immediately be employed as valuable additions to the financial engineer toolbox, albeit with some caveats.

The flexibility of models such as GANs comes at the high cost of some training instability and lack of robustness to changes in hyperparameters. This, coupled with the lack of quantitative evaluation metrics to assess the training progress makes them challenging to adopt in contexts where conformity to standards is not only important but many times required as part of strict regulations. We have seen how GANs applied to VaR and ES estimation seem to perform best on longer time frames, however, it is increasingly difficult to carry out statistically significant backtesting procedures on such horizons when considering real data.

We believe the true power of models such as GANs lies in their ability to generate realistic medium to long-term scenarios that could potentially bridge the gap between conditional and unconditional VaR models; employing GANs together with more traditional techniques such as GARCH models, in the EVT framework or together with predictive ML models might be worth exploring further.

Nevertheless, in order to employ such models with confidence, better training techniques must be developed and great effort must be devoted to devising better quantitative evaluation metrics.

Risk management is only one of the possible areas of application of generative models, many of which remain to this day largely unexplored, e.g. option pricing, such models could yield tangible competitive advantages not only for big financial institutions but also for single investors.

References

[1] RiskMetrics: Technical document. Technical report, J.P. Morgan and Reuters Limited Ltd and Morgan Guaranty Trust Company, New York, 1996. URL `https://books.google.it/books?id=Jm66tgAACAAJ`.

[2] Revisions to the Basel II market risk framework. Technical report, Basel Committee on Banking Supervision, Basel, Switzerland, 2009. URL `https://www.bis.org/publ/bcbs158.pdf`. 92-9197-774-8.

[3] Pilar Abad, Sonia Muela, and Carmen Lopez. A comprehensive review of value at risk methodologies. *The Spanish Review of Financial Economics*, 12, 01 2013. doi:10.1016/j.srfe.2013.06.001.

[4] Martin Arjovsky and Léon Bottou. Towards principled methods for training generative adversarial networks. 2017. doi:10.48550/ARXIV.1701.04862. URL `https://arxiv.org/abs/1701.04862`.

[5] Philippe Artzner, Freddy Delbaen, Eber Jean-Marc, and David Heath. Coherent measures of risk. *Mathematical Finance*, 9:203 – 228, 07 1999. doi:10.1111/1467-9965.00068.

[6] Laura Ballotta and Gianluca Fusai. Tools from stochastic analysis for mathematical finance: A gentle introduction. https://ssrn.com/abstract=3183712. *SSRN Electronic Journal*, 06 2018. doi:10.2139/ssrn.3183712.

[7] L. Balthazar. *From Basel 1 to Basel 3: The Integration of State of the Art Risk Modelling in Banking Regulation*. Finance and Capital Markets Series. Palgrave Macmillan UK, 2006. ISBN 9780230501171. URL `https://link.springer.com/book/10.1057/9780230501171`.

[8] Giovanni Barone-Adesi, Kostas Giannopoulos, and Les Vosper. VaR without correlations for portfolios of derivative securities. *Journal of Futures Markets*, 19(5):583–602, August 1999. URL `https://ideas.repec.org/a/wly/jfutmk/v19y1999i5p583-602.html`.

[9] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B Shah. Julia: A fresh approach to numerical computing. *SIAM Review*, 59(1):65–98, 2017. doi:10.1137/141000671. URL `https://epubs.siam.org/doi/10.1137/141000671`.

[10] Fischer Black and Myron Scholes. The pricing of options and corporate liabilities. *Journal of Political Economy*, 81(3):637–654, 1973. ISSN 00223808, 1537534X. URL `http://www.jstor.org/stable/1831029`.

[11] Tim Bollerslev. Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics*, 31(3):307–327, 1986. ISSN 0304-4076. doi:https://doi.org/10.1016/0304-4076(86)90063-1. URL `https://www.sciencedirect.com/science/article/pii/0304407686900631`.

[12] Peter F. Christoffersen. Evaluating interval forecasts. *International Economic Review*, 39(4):841–862, 1998. ISSN 00206598, 14682354. URL `http://www.jstor.org/stable/2527341`.

[13] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. *CoRR*, abs/1605.09782, 2016. URL `http://arxiv.org/abs/1605.09782`.

[14] K. Dowd. *Measuring Market Risk*. The Wiley Finance Series. Wiley, 2005. ISBN 9780470013038.

[15] Florian Eckerli and Joerg Osterrieder. Generative adversarial networks in finance: an overview. 2021. doi:10.48550/ARXIV.2106.06364. URL `https://arxiv.org/abs/2106.06364`.

[16] Robert F. Engle and Simone Manganelli. Caviar: Conditional autoregressive value at risk by regression quantiles. *Journal of Business & Economic Statistics*, 22(4):367–381, 2004. ISSN 07350015. URL `http://www.jstor.org/stable/1392044`.

[17] Cristóbal Esteban, Stephanie L. Hyland, and Gunnar Rätsch. Real-valued (medical) time series generation with recurrent conditional gans. 2017.

[18] Lucas-Benedikt Fiechner. Risk management with generative adversarial networks. Msc in mathematical and computational finance, University of Oxford, 2019.

[19] R. A. Fisher and L. H. C. Tippett. Limiting forms of the frequency distribution of the largest or smallest member of a sample. *Mathematical Proceedings of the Cambridge Philosophical Society*, 24(2):180–190, 1928. doi:10.1017/S0305004100015681.

[20] P. Glasserman. *Monte Carlo Methods in Financial Engineering*. Applications of mathematics : stochastic modelling and applied probability. Springer, 2004. ISBN 9780387004518. URL `https://books.google.it/books?id=e9GWUsQkPNMC`.

[21] B. Gnedenko. Sur la distribution limite du terme maximum d'une serie aleatoire. *Annals of Mathematics*, 44(3):423–453, 1943. ISSN 0003486X. URL `http://www.jstor.org/stable/1968974`.

[22] Ian J. Goodfellow. NIPS 2016 tutorial: Generative adversarial networks. *CoRR*, abs/1701.00160, 2017. URL `http://arxiv.org/abs/1701.00160`.

[23] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. 2014. doi:10.48550/ARXIV.1406.2661. URL `https://arxiv.org/abs/1406.2661`.

[24] Ian J. Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[25] J. Hull. *Options, Futures and Other Derivatives*. Eastern economy edition. Pearson/Prentice Hall, 2009. ISBN 9780136015864. URL `https://books.google.it/books?id=sEmQZoHoJCcC`.

[26] A. F. Jenkinson. The frequency distribution of the annual maximum (or minimum) values of meteorological elements. *Quarterly Journal of the Royal Meteorological Society*, 81:158–171, 1955.

[27] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. 2017.

[28] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2013. URL `https://arxiv.org/abs/1312.6114`.

[29] Paul Kupiec. Techniques for verifying the accuracy of risk measurement models. Finance and Economics Discussion Series 95-24, Board of Governors of the Federal Reserve System (U.S.), 1995. URL `https://EconPapers.repec.org/RePEc:fip:fedgfe:95-24`.

[30] P. Marano and M. Siri. *Insurance Regulation in the European Union: Solvency II and Beyond*. Springer International Publishing, 2018. ISBN 9783319870274. URL `https://books.google.it/books?id=Jiy9wQEACAAJ`.

[31] Robert Merton. Option prices when underlying stock returns are discontinuous. *Journal of Financial Economics*, 3:125–144, 01 1976. doi:10.1016/0304-405X(76)90022-2.

[32] Robert C. Merton. Theory of rational option pricing. *The Bell Journal of Economics and Management Science*, 4(1):141–183, 1973. ISSN 00058556. URL `http://www.jstor.org/stable/3003143`.

[33] Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. Unrolled generative adversarial networks. *CoRR*, abs/1611.02163, 2016. URL `http://arxiv.org/abs/1611.02163`.

[34] Laura Palagi. Global optimization issues in deep network regression: an overview. *Journal of Global Optimization*, 73(2):239–277, 2019. doi:10.1007/s10898-018-0701-7. URL `https://doi.org/10.1007/s10898-018-0701-7`.

[35] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.

[36] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2021. URL `https://www.R-project.org/`.

[37] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. 2015. doi:10.48550/ARXIV.1511.06434. URL `https://arxiv.org/abs/1511.06434`.

[38] Marco Rocco. Extreme value theory for finance: a survey. Questioni di Economia e Finanza (Occasional Papers) 99, Bank of Italy, Economic Research and International Relations Area, July 2011. URL `https://www.bancaditalia.it/pubblicazioni/qef/2011-0099`.

[39] Omer Sezer, Ugur Gudelek, and Murat Ozbayoglu. Financial time series forecasting with deep learning : A systematic literature review: 2005–2019. *Applied Soft Computing*, 90:106181, 02 2020. doi:10.1016/j.asoc.2020.106181.

[40] A. G. Stephenson. evd: Extreme value distributions. *R News*, 2(2):0, June 2002. URL `https://CRAN.R-project.org/doc/Rnews/`.

[41] Shuntaro Takahashi, Yu Chen, and Kumiko Tanaka-Ishii. Modeling financial time-series with generative adversarial networks. *Physica A: Statistical Mechanics and its Applications*, 527:121261, 2019. ISSN 0378-4371. doi:https://doi.org/10.1016/j.physa.2019.121261. URL `https://www.sciencedirect.com/science/article/pii/S0378437119307277`.

[42] J.M. Tomczak. *Deep Generative Modeling*. Springer International Publishing, 2022. ISBN 9783030931575. URL `https://books.google.it/books?id=GxO9zgEACAAJ`.

A GAN

A.1 Architecture

Table 1: Generator Network

Layer	Parameters
(0): Linear	(in features=100, out features=512, bias=True)
(1): Unsqueeze	Add dimension
(2): BatchNorm1d	(512, eps=1e-05, momentum=0.1, affine=True, track running stats = True)
(3): ReLU	(inplace=True)
(4): ConvTranspose1d	(512, 256, kernel size=(4,), stride=(2,), padding=(1,), bias=False)
(5): BatchNorm1d	(256, eps=1e-05, momentum=0.1, affine=True, track running stats = True)
(6): ReLU	(inplace=True)
(7): BatchNorm1d	(256, eps=1e-05, momentum=0.1, affine=True, track running stats = True)
(8): ConvTranspose1d	(256, 128, kernel size=(4,), stride=(2,), padding=(1,), bias=False)
(9): ReLU	(inplace=True)
(10): BatchNorm1d	(128, eps=1e-05, momentum=0.1, affine=True, track running stats = True)
(11): ConvTranspose1d	(128, 64, kernel size=(4,), stride=(2,), padding=(1,), bias=False)
(12): ReLU	(inplace=True)
(13): BatchNorm1d	(64, eps=1e-05, momentum=0.1, affine=True, track running stats = True)
(14): ConvTranspose1d	(64, 32, kernel size=(4,), stride=(2,), padding=(1,), bias=False)
(15): ReLU	(inplace=True)
(16): ConvTranspose1d	(32, 1, kernel size=(4,), stride=(2,), padding=(1,), bias=False)
(17): Squeeze	Remove dimension

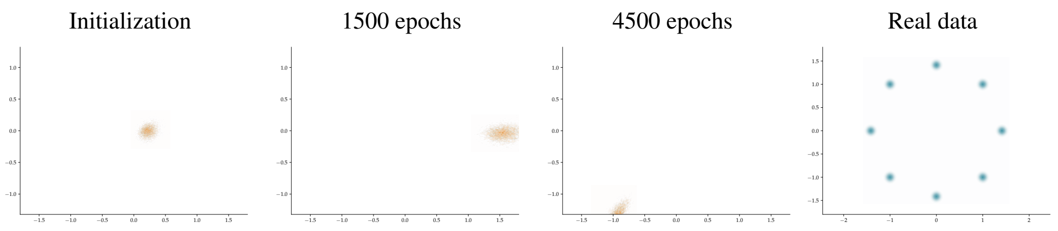
Table 2: Discriminator Network

Layer	Parameters
(0): Unsqueeze	Add dimension
(1): Conv1d	(1, 32, kernel size=(4,), stride=(2,), padding=(1,), bias=False)
(2): BatchNorm1d	(32, eps=1e-05, momentum=0.1, affine=True, track running stats = True)
(3): LeakyReLU	(negative slope=0.2, inplace=True)
(4): Conv1d	(32, 64, kernel size=(4,), stride=(2,), padding=(1,), bias=False)
(5): BatchNorm1d	(64, eps=1e-05, momentum=0.1, affine=True, track running stats = True)
(6): LeakyReLU	(negative slope=0.2, inplace=True)
(7): Conv1d	(64, 128, kernel size=(4,), stride=(2,), padding=(1,), bias=False)
(8): BatchNorm1d	(128, eps=1e-05, momentum=0.1, affine=True, track running stats = True)
(9): LeakyReLU	(negative slope=0.2, inplace=True)
(10): Conv1d	(128, 256, kernel size=(4,), stride=(2,), padding=(1,), bias=False)
(11): BatchNorm1d	(256, eps=1e-05, momentum=0.1, affine=True, track running stats = True)
(12): LeakyReLU	(negative slope=0.2, inplace=True)
(13): Conv1d	(256, 1, kernel size=(1,), stride=(2,), bias=False)
(14): Squeeze	Remove dimension
(15): Sigmoid	

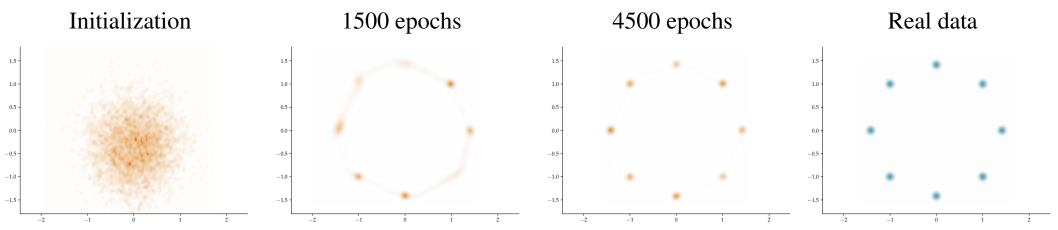
A.2 Autoencoder pre-training

From tests on toy datasets, we found that pretraining the Generator using it as the decoder of a denoising AE [24] can speed up the training process. Additionally, such an initialization has in some cases produced qualitatively better results in terms of mode collapse.

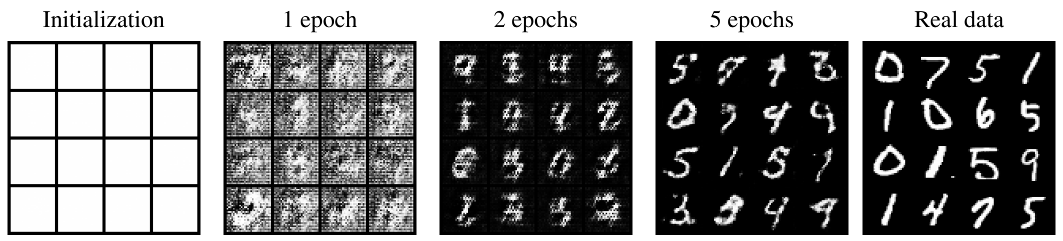
Though a more thorough investigation is required, this technique seems to generalize to different architectures and datasets, in figure 13 we show the results of applying AE pretraining to a GAN using Multi Layer Perceptron (MLP) networks for both G and D trained on a toy dataset similar to the one used in [33] and to DCGAN trained on the Modified National Institute of Standards and Technology (MNIST) dataset.



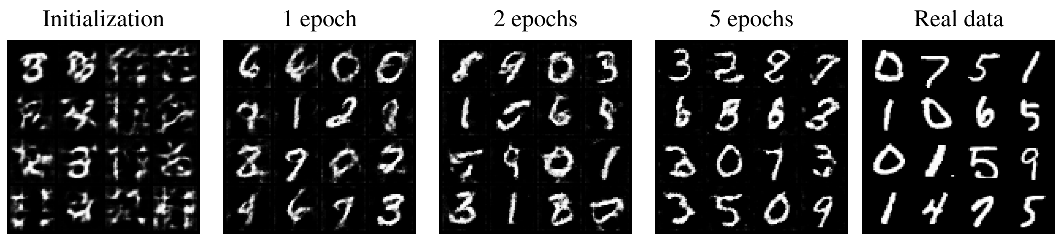
(a) MLP GAN with No pretraining



(b) MLP GAN with AE pretraining



(c) DCGAN with no pretraining



(d) DCGAN with AE pretraining

Figure 13: No pretraining (a,c) vs. AE pretraining (b,d) technique applied to two different GAN architectures and two different datasets. In the first comparison, KDE is used on the Generator's samples; in this specific case, a better initialization seems to greatly reduce mode collapse.