TAIL-GAN:

Nonparametric Scenario Generation for Tail Risk Estimation

Rama Cont\textsuperscript{1}, Mihai Cucuringu\textsuperscript{1, 2, 3}, Renyuan Xu\textsuperscript{4}, and Chao Zhang\textsuperscript{2}

\textsuperscript{1}Mathematical Institute, University of Oxford, Oxford, UK
\textsuperscript{2}Department of Statistics, University of Oxford, Oxford, UK
\textsuperscript{3}The Alan Turing Institute, London, UK
\textsuperscript{4}Department of Industrial and Systems Engineering, University of Southern California, Los Angeles, US

March, 2022

Abstract

The estimation of loss distributions for dynamic portfolios requires the simulation of scenarios representing realistic joint dynamics of their components, with particular importance devoted to the simulation of tail risk scenarios. Commonly used parametric models have been successful in applications involving a small number of assets, but may not be scalable to large or heterogeneous portfolios involving multiple asset classes.

We propose a novel data-driven approach for the simulation of realistic multi-asset scenarios with a particular focus on the accurate estimation of tail risk for a given class of static and dynamic portfolios selected by the user. By exploiting the joint elicitability property of Value-at-Risk (VaR) and Expected Shortfall (ES), we design a Generative Adversarial Network (GAN) architecture capable of learning to simulate price scenarios that preserve tail risk features for these benchmark trading strategies, leading to consistent estimators for their Value-at-Risk and Expected Shortfall. We demonstrate the accuracy and scalability of our method via extensive simulation experiments using synthetic and market data. Our results show that, in contrast to other data-driven scenario generators, our proposed scenario simulation method correctly captures tail risk for both static and dynamic portfolios.

Keywords: Scenario Simulation, Generative Adversarial Network (GAN), Time Series, Expected Shortfall, Value at Risk, Dynamic Strategies.
1 Introduction

Scenario simulation is extensively used in finance for estimating the loss distribution of portfolios and trading strategies, often with a focus on the estimation of risk measures such as Value-at-Risk and Expected Shortfall [20]. The estimation of such risk measures for static and dynamic portfolios involves the simulation of scenarios representing realistic joint dynamics of their components. This requires both a realistic representation of the temporal dynamics of individual assets (temporal dependence), as well as an adequate representation of their co-movements (cross-asset dependence).
The risk estimation has become more important for financial applications in recent years, in light of the Basel Committee’s Fundamental Review of the Trading Book (FRTB), an international standard that regulates the amount of capital banks ought to hold against market risk exposures [34]. FRTB particularly revisits and emphasizes the use of value at risk vs expected shortfall [15] as a measure of risk under stress, thus ensuring that banks capture tail risk events. In addition, FRTB requires banks to develop clear methodologies to specify how various extreme scenarios are simulated and how the stress scenario risk measures are constructed using these scenarios. This suite of capital rules takes effect January 2022 to strengthen the financial system, with an eye towards capturing tail risk events that came to light during the 2007-2008 financial crisis.

A common approach in scenario simulation is to use parametric models, either time series models such as GARCH models, diffusion and jump processes, stochastic volatility models, and copulas. While these approaches have been successful in low-dimensional applications involving a small number of assets, very few of them are scalable to large or heterogeneous portfolios involving multiple asset classes. Even the joint modeling of a relatively small number of risk factors exhibiting heterogeneous dynamics, as observed for instance in “hybrid” diversified portfolios involving interest rates, exchange rates and equity risk factors, turns out to be a challenging problem from a modeling perspective, and usually leads to the risk of model misspecification.

As a result of this difficulty and the lack of scalability of bespoke parametric models, Gaussian models with constant coefficients (dubbed as the “multivariate Black-Scholes model”) have been often used as the default approach in most high-dimensional applications, even when there is ample econometric evidence for their inadequacy, given the stylized features – heavy tails, volatility clustering, tail dependence, etc. – observed in many financial time series [12].

Generative Adversarial Networks (GANs) [22] have emerged in the recent years as an efficient alternative to parametric models for the simulation of patterns whose features are extracted from complex and high-dimensional data sets. GANs are generative models between two competing neural networks: a generator network \( G \) and a discriminator network \( D \). The generator network \( G \) attempts to fool the discriminator network by converting random noise into sample data, while the discriminator network \( D \) tries to identify whether the input sample is fake or real. The goal for the generator \( G \) is to output (generated) samples that resemble as best as possible the true samples under a certain criterion. GANs have been successfully applied in generation of images [22, 38], audio [14, 35], and text [16, 45], which can be further combined with downstream tasks such as image reconstruction [46], face recognition [24] and abnormal detection [11].
1.1 Scenario simulation using Generative Adversarial Networks (GAN)

Simulation methods based on GANs have also been recently used in several instances for the simulation of financial market scenarios [8, 19, 25, 29, 30, 32, 35, 39, 42, 43]. Takahashi et al. [39] used GAN to generate one-dimensional financial time-series and observed that GAN is able to capture certain stylized facts of univariate price returns, such as heavy-tailed return distribution and volatility clustering. Wiese et al. [42] introduced the Quant-GAN architecture, where the generator utilizes the temporal convolutional network (TCN), first proposed in [35], to capture long-range dependencies in financial data. Marti et al. [30] applied a convolutional-network-based GAN framework (denoted as DCGAN) to simulate empirical correlation matrices of asset returns. However, no dynamic patterns such as autocorrelation could be captured in the framework. The Conditional-GAN (CGAN) architecture, first introduced in [31], and its variants were proposed to simulate financial data or time series in a line of works [19, 25, 29, 32]. Compared to the classic GAN architecture, CGAN has an additional input variable for both the generator and discriminator, in order to incorporate certain structural information into the training stage, for example, the lag information inherent in the time series. Furthermore, Yoon et al. [43] proposed a framework named TimeGAN, for producing realistic multivariate time-series, that combines the flexibility of the unsupervised GAN approach with the control afforded by supervised learning. Namely, in addition to the traditional unsupervised adversarial loss on both market data and simulated data, TimeGAN introduced a stepwise supervised loss, which helps to learn from the transition dynamics of the market data.

Unlike image generation, which can be validated by visual inspection, model validation for data-driven financial scenarios has remained an open and challenging question in practice. Hence, the design of a loss function and evaluation criterion are critical for simulating financial scenarios via a data-driven approach. In addition, GANs often suffer from inferior performance due to the unstable training [3]. One possible reason may stem from the fact that the criterion of the classic GAN architecture [22] and its variants, including Wasserstein GAN (WGAN) [4], focuses on some global properties of the full distribution, instead of some problem-dependent statistics which are more essential for particular applications.

1.2 Main contributions

To simulate high-dimensional financial scenarios across assets with accurate estimation of tail risk measures for a given class of benchmark strategies, we introduce in this work a novel approach for scenario simulation which specifically addresses the above mentioned issue of lacking problem-dependent statistics as evaluation criterion. In particular, we develop a new framework denoted as Tail Generative Adversarial Network (TAIL-GAN) to simulate multivariate financial time series data that preserve the same specified statistics for
benchmark strategies as the training data. Compared to the GAN architectures studied in the literature, the proposed Tail-GAN framework has the following novel ingredients which has not been investigated before:

1. **Tail risks as evaluation criterion:** Tail risk refers to the risk of large portfolio losses. Value at Risk (VaR) and Expected Shortfall (ES) are commonly used statistics for measuring the tail risk of a variety of trading strategies. To correctly measure the tail risks of trading strategies applied to the input price scenarios and the generated price scenarios, we incorporate VaR and ES of a given set of trading strategies (introduced below) in the loss function by utilizing the joint elicitability property of VaR and ES. Namely, (VaR, ES) can be written as a minimizer of some score function therefore it can be learned by solving some optimization problem.

2. **Discrimination via dynamic trading strategies:** In the loss function of Tail-GAN architecture, we introduce the risk measures of multiple static portfolios and dynamic trading strategies applied to the price series, with static portfolios capturing cross-asset dependence information and dynamic strategies capturing temporal dependence information in the financial scenarios. Dynamic trading strategies are nonlinear functions and possibly path-dependent functions of the underlying price scenarios. On one hand, the computational complexity of the training process increases when including these strategies into the discriminator's loss function. On the other hand, this exploration in the space of nonlinear functions of the underlying price scenarios significantly improves the simulation accuracy of GAN models. Previous works on using GAN for simulating financial scenarios only focus on matching return distributions between the simulated data and the input data [42]. This corresponds to considering a naive buy-and-hold strategy as the evaluation criterion and resulting in no guarantee for consistent behaviors of other (nonlinear) strategies applied to the simulated data and the input data.

Compared to the above mentioned Gaussian models and GAN models, Tail-GAN is able to capture the stylized features observed in the financial markets, including tail risk behaviors, heavy tail distributions, and complex dependence structures among assets. In particular, the advantages of Tail-GAN can be summarized as follows.

- **(A) Desired tail risk measure for simulated data:** We propose the first GAN architecture which is able to yield consistent tail risks (of multiple trading strategies) between input financial scenarios and simulated financial scenarios. In contrast, all prior works on data-driven approach for generating financial data focus on the average performance over the full distribution of simulated financial scenarios [8, 19, 25, 29, 31, 32, 42, 43]. Since the closeness of two distributions under the Wasserstein distance does not imply the closeness of quantiles, these average performance criteria may lead to inaccurate characterizations of the risks from the real financial data. Trading strategies tested with financial scenarios sampled from such simulators may lead
to huge financial losses in the real markets.

- **(B) Generalization power:** A key feature of our Tail-GAN simulator lies in its generalization ability, which we thoroughly study and compare with a supervised-learning model. Generalization ability of machine learning models still remains an open question in the deep learning community [5, 44]. We empirically confirm that Tail-GAN outperforms this supervised-learning model in terms of both simulation accuracy and generalization power. This important perspective, in particular to demonstrate whether discriminator helps to improve from supervised-learning methods, has been missing in the literature of GAN for financial applications.

- **(C) Scalability:** In practice, most of the portfolios held by asset managers are constructed with more than 20 financial assets, and most often significantly more. However, much of the research up to now has only concentrated on simulating scenarios for a single asset or a small number of assets [42, 43]. In this work, we explore and assess the scalability of Tail-GAN. We show that by leveraging Principal Component Analysis (PCA), Tail-GAN scales to generating price scenarios with a large number of heterogeneous assets.

- **(D) Customized GAN simulator:** The Tail-GAN framework is flexible and able to provide simulated scenarios with accurate risk measures for the trading strategies included in the loss function. From a practical perspective, it is computational infeasible to include all possible strategies in the Tail-GAN framework and most of the financial institutions are only interested in a sub-class of strategies. Therefore, it suffices to include a group strategies of interest in the Tail-GAN framework. This leads to a personalized simulator which is tailored to customized strategies.

**Related literature.** We would like to mention that the idea of incorporating quantile properties into the simulation model has been explored in [37], which introduced an autoregressive implicit quantile network (AIQN). The goal is to train a simulator via supervised learning so that the quantile divergence between the empirical distributions of the training data and the generated data is minimized. However, the quantile divergence adopted in AIQN is an average performance across all quantiles, which provides no guarantees for the tail risks. In addition, the simulator trained with supervised learning may suffer from accuracy issues and the lack of generalization power (see Section 5.1 for a detailed discussion).

Tail risk is also closely related to extreme events in the financial markets. To this end, Bhatia et al. [7] proposed a GAN-based approach, denoted ExGAN, to model the extremes of the training distribution via Extreme Value Theory (EVT). The idea is to use CGANs conditioned on the statistics for extreme events according to EVT, and to generate new samples according to the given probability measure for extreme events.

The idea of exploiting input price scenarios (or simulated price scenarios) via non-linear functionals has
also been proposed in recent studies [8, 9] via the notion of signature. In particular, Buehler et al. [8, 9] developed a generative model based on Variational Autoencoder (VAE) and signatures of price series, which is capable of operating with small amount of training data. The concept of signatures was coined in the context of rough paths, and provides an efficient and parsimonious toolkit to encode the most essential information contained in price scenarios.

**Paper outline.** The rest of the paper is organized as follows. Section 2 introduces the concepts of tail risk measures and elicitation, and discusses the choice of score functions and their optimization landscapes. Section 3 explains our methodology and introduces the architecture of Tail-GAN. Section 4 provides numerical experiments with synthetic scenarios for model validation. Section 5 presents the generalization and scalability power of Tail-GAN. Section 6 discusses the performance of Tail-GAN on real-world intraday scenarios.

## 2 Tail risk measures and score function

### 2.1 Tail risk measures

Tail risk refers to the risk of large portfolio losses. *Value at Risk (VaR)* and *Expected Shortfall (ES)* are commonly used statistics for measuring the tail risk of portfolios.

The gain of a portfolio at a certain horizon may be represented as a random variable, i.e., a map $X : \Omega \to \mathbb{R}$ on the set $\Omega$ of market scenarios. Given a probabilistic model, represented by a probability measure $\mathbb{P}$ on $\Omega$, the Value-at-Risk (VaR) at confidence level $0 < \alpha < 1$ is defined as the $\alpha$-quantile of $X$ under $\mathbb{P}$

$$\text{VaR}_\alpha(X; \mathbb{P}) := \inf\{x \in \mathbb{R} : \mathbb{P}(X \leq x) \geq \alpha\}.$$  
(1)

We will consider such tail risk measures under different probabilistic models, each represented by a probability measure $\mathbb{P}$ on the space $\Omega$ of market scenarios, and the notation above emphasizes the dependence on $\mathbb{P}$.

ES is an alternative to VaR that is sensitive to the tail of the loss distribution beyond the level $\alpha$

$$\text{ES}_\alpha(X; \mathbb{P}) := \frac{1}{\alpha} \int_0^\alpha \text{VaR}_\beta(X; \mathbb{P}) d\beta.$$  
(2)

**Elicitability and score functions.** In statistics, M-estimators are a broad class of extremum estimators for which the objective function is a sample average. More generally, an M-estimator may be defined to be a root of an estimating function. This estimating function is often the derivative of another statistical
function. The property of a statistical functional to possess a (consistent) $M$-estimator has become known as elicitability [21, 27, 36]. More specifically, we say that a statistical functional $T$ is elicitable for a set of random variables $\mathcal{X}$ under measure $\mathbb{P}$ if there is a score function $S(x,y)$ such that

$$T(X;\mathbb{P}) = \arg\min_x \int S(x,y) \mathbb{P}(X \in dy),$$

(3)

for any random variables $X \in \mathcal{X}$ under measure $\mathbb{P}$. Examples of elicitable statistical functionals and their strictly consistent score functions include the mean $T(X;\mathbb{P}) = \int x \mathbb{P}(X \in dx)$ with $S(x,y) = (x-y)^2$, and the median $T(X;\mathbb{P}) = \inf \{ x \in \mathbb{R} : \mathbb{P}(X \leq x) \geq 0.5 \}$ with $S(x,y) = |x-y|$.

It has been shown in [21, 41] that ES is not elicitable, whereas VaR at level $\alpha \in (0,1)$ is elicitable for random variables with a unique $\alpha$-quantile. However, it turns out that ES is elicitable of higher order, in the sense that the pair $(\text{VaR}_\alpha(X;\mathbb{P}), \text{ES}_\alpha(X;\mathbb{P}))$ is jointly elicitable. In particular, the following result [17, Theorem 5.2] gives a family of score functions which are strictly consistent for $(\text{VaR}_\alpha(X;\mathbb{P}), \text{ES}_\alpha(X;\mathbb{P}))$.

**Proposition 2.1.** [17, Theorem 5.2] Assume $\int |x| \mathbb{P}(X \in dx) < \infty$. If $H_2 : \mathbb{R} \rightarrow \mathbb{R}$ is strictly convex and $H_1 : \mathbb{R} \rightarrow \mathbb{R}$ is such that

$$v \mapsto R_\alpha(v,e) := \frac{1}{\alpha} vH_2'(e) + H_1(v),$$

(4)

is strictly increasing for each $e \in \mathbb{R}$, then the score function

$$S_\alpha(v,e,x) = (\mathbb{I}_{\{x \leq v\}} - \alpha)(H_1(v) - H_1(x)) + \frac{1}{\alpha} H_2'(e) \mathbb{I}_{\{x \leq v\}}(x-v) + H_2'(e)(e-v) - H_2(e),$$

(5)

is strictly consistent for $(\text{VaR}_\alpha(X;\mathbb{P}), \text{ES}_\alpha(X;\mathbb{P}))$, i.e.

$$(\text{VaR}_\alpha(X;\mathbb{P}), \text{ES}_\alpha(X;\mathbb{P})) = \arg\min_{(v,e) \in \mathbb{R}^2} \int S_\alpha(v,e,x) \mathbb{P}(X \in dx).$$

(6)

### 2.2 Score function

The computation of the estimator (6) involves the optimization of

$$s_\alpha(v,e) := \int S_\alpha(v,e,x) \mathbb{P}(X \in dx),$$

(7)
for a given one-dimensional distribution \( \nu \). While any choice of \( H_1, H_2 \) satisfying the conditions of Theorem 2.1 theoretically leads to consistent estimators in (6), different choices of \( H_1 \) and \( H_2 \) lead to optimization problems with different landscapes \( s_\alpha \), and some are easier to optimize than others. We consider here a parametric specification for the score function which leads to a tractable optimization problem, and study some of its properties.

The following score function was proposed by Acerbi & Szekely [1], and has been adopted by practitioners for back-testing purposes

\[
S_\alpha(v, e, x) = \frac{W_\alpha}{2} (1 \{ x \leq v \} - \alpha)(x^2 - v^2) + 1 \{ x \leq v \} e(v - x) + \alpha e \left( \frac{e}{2} - v \right), \quad \text{with} \quad \frac{\text{ES}_\alpha(X; \mathbb{P})}{\text{VaR}_\alpha(X; \mathbb{P})} \geq W_\alpha \geq 1. \tag{8}
\]

This choice is special case of (5), where \( H_1 \) and \( H_2 \) are given by

\[
H_1(v) = -\frac{W_\alpha}{2} v^2, \quad H_2(e) = \frac{\alpha}{2} e^2, \quad \text{with} \quad \frac{\text{ES}_\alpha(X; \mathbb{P})}{\text{VaR}_\alpha(X; \mathbb{P})} \geq W_\alpha \geq 1. \tag{9}
\]

It is easy to check that (8) satisfies the conditions in Proposition 2.1 on the subspace \( \{ (v, e) \in \mathbb{R}^2 \mid W_\alpha v \leq e \leq v \leq 0 \} \); to this end we refer the reader to Proposition 2.2.

![Figure 1: Landscape of \( s_\alpha(v, e) \) with \( \alpha = 0.05 \) for the uniform distribution on \([-1,1]\).](image)

Next, we provide the following theoretical guarantee for the well behaved optimization landscape of the score function; also, see Figure 1 for a visualization.

**Proposition 2.2.** (1) Assume \( \text{VaR}_\alpha(X; \mathbb{P}) < 0 \), for \( \alpha < 1/2 \). Then the score function (8) is strictly consistent for \( (\text{VaR}_\alpha(X; \mathbb{P}), \text{ES}_\alpha(X; \mathbb{P})) \) and the Hessian of \( s_\alpha(v, e) \) is positive semi-definite on the region

\[
\mathcal{B} = \{ (v, e) \mid v \leq \text{VaR}_\alpha(X; \mathbb{P}), \text{ and } W_\alpha v \leq e \leq v \leq 0 \}.
\]
\( (2) \) In addition, we assume there exist \( \delta_\alpha \in (0, 1) \), \( \xi_\alpha \in (0, \frac{1}{2} - \alpha) \), \( z_\alpha \in (0, \frac{1}{2} - \alpha) \), and \( W_\alpha > \frac{1}{\sqrt{\alpha}} \) such that

\[
\mathbb{P}(X \in dx) \geq \delta_\alpha \text{ for } x \in [\text{VaR}_\alpha(X; \mathbb{P}), \text{VaR}_{\alpha + \xi_\alpha}(X; \mathbb{P})] \quad \text{and} \quad \text{ES}_\alpha(X; \mathbb{P}) \geq W_\alpha \text{Var}_\alpha(X; \mathbb{P}) + z_\alpha. \tag{10}
\]

Then the Hessian of \( s_\alpha(v, e) \) is positive semi-definite on the region \( \widetilde{B} = \{ (v, e) \mid v \leq \text{VaR}_{\alpha + \beta_\alpha}(X; \mathbb{P}), \text{ and } W_\alpha v + z_\alpha \leq e \leq v \leq 0 \} \) where \( \beta_\alpha = \min \left\{ \varepsilon_\alpha, \frac{z_\alpha \delta_\alpha}{2W_\alpha} \right\} \).

**Example 2.3** (Example for condition (10)). Condition (10) holds when \( X \) has a strictly positive density under measure \( \mathbb{P} \). Take an example where \( X \) follows the standard normal distribution under measure \( \mathbb{P} \). Denote \( \phi(x) = \frac{1}{\sqrt{2\pi}} \exp \left( -\frac{x^2}{2} \right) \) as the density function, and \( \Phi(y) = \int_{-\infty}^{y} \phi(x)dx \) as the cumulative density function for \( X \). Then we have \( \text{VaR}_\alpha(X; \mathbb{P}) = \Phi^{-1}(\alpha) \) and \( \text{ES}_\alpha(X; \mathbb{P}) = -\frac{\alpha(\Phi^{-1}(\alpha))}{\alpha} \). Setting \( \alpha = 0.05 \) and \( \varepsilon_\alpha = 0.05 \), we have \( \text{VaR}_{0.05}(X; \mathbb{P}) \approx -1.64 \) and \( \text{ES}_{0.05}(X; \mathbb{P}) \approx -2.06 \) by direct calculation. Then we can set \( \delta_\alpha = \phi(\Phi^{-1}(0.05)) \approx 0.103, \) \( W_\alpha = 5 \) and \( z_\alpha = \frac{1}{4} \). Hence (10) holds for \( \beta_\alpha = \min \{ \varepsilon_\alpha, \frac{z_\alpha \delta_\alpha}{2W_\alpha} \} \approx 0.0025 \).

Proposition 2.2 implies that \( s_\alpha(v, e) \) has a good optimization landscape on region \( B \) or \( \widetilde{B} \) if the corresponding conditions are satisfied. In particular, the minimizer of \( s_\alpha(v, e) \), i.e., \( (\text{VaR}_\alpha(X; \mathbb{P}), \text{ES}_\alpha(X; \mathbb{P})) \), is on the boundary of region \( B \). \( \widetilde{B} \) contains an open ball with center \( (\text{VaR}_\alpha(X; \mathbb{P}), \text{ES}_\alpha(X; \mathbb{P})) \).

**Proof.** Proof of Proposition 2.2. First we check the elicitation condition for \( H_1(v) \) and \( H_2(e) \) on region \( B \).

When \( H_2(e) = \frac{\alpha}{2} e^2 \), we have \( H'_2(e) = \alpha e \) and \( H''_2(e) = \alpha \). For any \( (v, e) \in B \), this amounts to

\[
\frac{\partial R_\alpha(v, e)}{\partial v} = e - W_\alpha v \geq 0, \tag{11}
\]

where \( R_\alpha(v, e) \) is defined in Eqn (4).

Recall the score function \( S_\alpha(v, e, x) \) defined in Eqn (8), and \( s_\alpha(v, e) \) defined in Eqn (7). Then

\[
s_\alpha(v, e) = \left( \mathbb{P}(X \leq v) - \alpha \right) \frac{W_\alpha}{2} v^2 + \frac{W_\alpha}{2} \int_{-\infty}^{v} x^2 \mathbb{P}(X \in dx) + \mathbb{P}(X \leq v) e
\]

\[
- e \int_{-\infty}^{v} x \mathbb{P}(X \in dx) + \alpha e \left( \frac{e}{2} - v \right) + \text{const}. \tag{12}
\]

Therefore,

\[
\frac{\partial s_\alpha}{\partial v} (v, e) = \left( \mathbb{P}(X \leq v) - \alpha \right) (-W_\alpha v + e),
\]

\[
\frac{\partial s_\alpha}{\partial e} (v, e) = \mathbb{P}(X \leq v) e - \int_{-\infty}^{v} x \mathbb{P}(X \in dx) + \alpha(e - v).
\]
And hence
\[
\frac{\partial^2 s_\alpha}{\partial v^2}(v, e) = \frac{\mathbb{P}(X \in dv)}{dv} (-W_\alpha v + e) - W_\alpha(\mathbb{P}(X \leq v) - \alpha),
\]  
(13)
\[
\frac{\partial^2 s_\alpha}{\partial e^2}(v, e) = \alpha,
\]  
(14)
\[
\frac{\partial^2 s_\alpha}{\partial e \partial v}(v, e) = \mathbb{P}(X \leq v) - \alpha.
\]  
(15)

Since \( \frac{\mathbb{P}(X \in dv)}{dv} \geq 0 \) and \(-W_\alpha v + e > 0\) hold on region \( \mathcal{B} \), we have
\[
\frac{\partial^2 s_\alpha}{\partial v^2}(v, e) \geq -W_\alpha(\mathbb{P}(X \leq v) - \alpha), \text{ on } \mathcal{B}.
\]

Therefore \( \frac{\partial^2 s_\alpha}{\partial v^2}(v, e) \geq 0 \) holds since \( v \leq \text{VaR}_\alpha(X; \mathbb{P}) \) on region \( \mathcal{B} \). Next when \((v, e) \in \mathcal{B} \),
\[
\frac{\partial^2 s_\alpha}{\partial v^2}(v, e) - \left( \frac{\partial^2 s_\alpha}{\partial v \partial e} \right)^2 = \alpha \frac{\mathbb{P}(X \in dv)}{dv} (-W_\alpha v + e) - \alpha W_\alpha(\mathbb{P}(X \leq v) - \alpha) - (\mathbb{P}(X \leq v) - \alpha)^2 \]
\[
\geq (\alpha - \mathbb{P}(X \leq v))(\alpha W_\alpha - \alpha + \mathbb{P}(X \leq v))
\]
\[
\geq (\alpha - \mathbb{P}(X \leq v))\mathbb{P}(X \leq v) \geq 0.
\]
(16)

(17)

Note that (16) holds since \(-W_\alpha v + e \geq 0\), and (17) holds since \( W_\alpha \geq 1 \) and \( \mathbb{P}(X \leq v) \leq \alpha \) on \( \mathcal{B} \). Therefore \( \nabla^2 s_\alpha \) is positive semi-definite on the region \( \mathcal{B} \).

In addition, when condition (10) holds, we show that \( s_\alpha(v, e) \) is positive semi-definite on \( \bar{\mathcal{B}} \).

Denote \( \bar{\mathcal{B}}^1 = \bar{\mathcal{B}} \cap \{ (v, e) \in \mathbb{R}^2 | v \leq \text{VaR}_\alpha(X; \mathbb{P}) \} \) and \( \bar{\mathcal{B}}^2 = \bar{\mathcal{B}} \cap \{ (v, e) \in \mathbb{R}^2 | v > \text{VaR}_\alpha(X; \mathbb{P}) \} \). Then \( \bar{\mathcal{B}}^1 \cup \bar{\mathcal{B}}^2 = \bar{\mathcal{B}} \) and \( \bar{\mathcal{B}}^1 \cap \bar{\mathcal{B}}^2 = \emptyset \). The positive semi-definiteness property of \( s_\alpha \) on \( \bar{\mathcal{B}}^1 \) follows a similar proof as above.

We only need to show that \( s_\alpha \) is positive semi-definite on \( \bar{\mathcal{B}}^2 \). In this case, we have
\[
\frac{\partial^2 s_\alpha}{\partial v^2}(v, e) = \frac{\mathbb{P}(X \in dv)}{dv} (-W_\alpha v + e) - W_\alpha(\mathbb{P}(X \leq v) - \alpha)
\]
\[
\geq \delta_\alpha z_\alpha - W_\alpha(\mathbb{P}(X \leq v) - \alpha) \geq 0,
\]  
(18)

which holds since \( \frac{\partial^2 s_\alpha}{\partial v^2} + \alpha \geq \beta_\alpha + \alpha \geq \mathbb{P}(X \leq v) \) on \( \bar{\mathcal{B}} \). In addition,
\[
\frac{\partial^2 s_\alpha}{\partial v^2} - \left( \frac{\partial^2 s_\alpha}{\partial v \partial e} \right)^2 = \alpha \frac{\mathbb{P}(X \in dv)}{dv} (-W_\alpha v + e) - \alpha W_\alpha(\mathbb{P}(X \leq v) - \alpha) - (\mathbb{P}(X \leq v) - \alpha)^2 \]
\[
\geq \alpha \delta_\alpha z_\alpha + (\mathbb{P}(X \leq v) - \alpha)(-\alpha W_\alpha + \alpha - \mathbb{P}(X \leq v))
\]
\[
\geq \alpha \delta_\alpha z_\alpha - \beta_\alpha(\alpha W_\alpha + \beta_\alpha) \geq 0.
\]  
(19)
Here Eqn (19) holds since $\frac{P(X \in dv)}{dv} \geq \delta_\alpha$ and $z_\alpha \geq (-W_\alpha v + e)$. Eqn (20) holds since $P(X \leq v) \in (\alpha, \alpha + \beta_\alpha]$ on $\mathcal{B}^2$. To show (20), it suffices to show

$$\alpha \delta_\alpha z_\alpha - \frac{\delta_\alpha z_\alpha}{2W_\alpha} (\alpha W_\alpha + \frac{\delta_\alpha z_\alpha}{2W_\alpha}) \geq 0,$$

since $\beta_\alpha \leq \frac{\delta_\alpha z_\alpha}{2W_\alpha}$. Finally, (21) holds since $W_\alpha > \frac{1}{\sqrt{\alpha}}$, $\delta_\alpha \in (0, 1)$, and $z_\alpha \in (0, \frac{1}{2} - \alpha)$. This completes the proof.

In summary, $s_\alpha(v, e)$ has a positive semi-definite Hessian in a neighborhood of the minimum (see Proposition 2.2), which leads to desirable properties for convergence. There are other alternative score functions with different choices of $H_1$ and $H_2$, but some of them may have undesirable properties [17, 18] as the following example shows.

**Example 2.4.** When $H_2(x) = \exp(x)$, there does not exist a constant $c < 0$ such that

$$\frac{\partial^2 s_\alpha}{\partial e^2} \geq 0 \text{ on } (-\infty, c],$$

when $X$ is uniformly distributed on $[-1, 1]$ and $\alpha = 0.05$. Eqn (22) implies that the objective function $s_\alpha$ is not convex which renders the optimization problem difficult.

**Proof.** Proof of (22). If $X$ is uniformly distributed on $[-1, 1]$ and $\alpha = 0.05$, we have

$$P(X \leq v) = \int_{-\infty}^v xP(X \in dx) + (e - v) = \frac{1}{4} v^2 + \left(\frac{1}{2} - 0.05\right) v + \frac{1}{4} + 0.05e,$$

which yields

$$\frac{\partial^2 s_\alpha}{\partial e^2} = \frac{\exp(e)}{\alpha} \left[ \frac{1}{4} (v + 0.9)^2 + 0.0475 + \alpha + 0.05e \right].$$

Letting $v = -0.9$, we arrive at $\frac{\partial^2 s_\alpha}{\partial e^2}|_{v=-0.9} < 0$ for all $e < -1.95$. 

## 3 Learning to generate tail scenarios

We now introduce the Tail Generative Adversarial Network (Tail-GAN) for simulating multivariate price scenarios. Given a set of input price scenarios as training data, Tail-GAN learns to simulate new price scenarios with the same tail risk statistics as the training data, by solving a mini-max game between a generator and a discriminator. The generator creates samples that are intended to approximate the distribution of the training data. The discriminator evaluates the quality of the simulated samples using tail risk measures,
namely Value-at-Risk (VaR) and Expected Shortfall (ES), across a set of benchmark trading strategies, including static portfolios and dynamic trading strategies. Static portfolios and dynamic strategies capture properties of the price scenarios from different perspectives: static portfolios explore the correlation structure among the assets, while dynamic trading strategies, such as mean-reversion and trend-following strategies, discover time series properties such as trends, volatility and mean-reversion properties. To train the network we use an objective function which leverages the elicitability of VaR and ES and guarantees consistency of the estimator.

The theoretical setup of Tail-GAN is discussed in Section 3.1. Scenario simulation is discussed in Section 3.2, and the design of the discriminator is discussed in Section 3.3.

3.1 Framework of Tail-GAN

We start with an introduction on the class of strategies considered to evaluate the generator, building on the previously introduced concept of joint elicitability property of VaR and ES. Finally, we explain how VaR, ES and the joint elicitability property can altogether be leveraged by the discriminator to evaluate the simulated price scenarios.

3.1.1 Benchmark trading strategies

Our goal is to generate market scenarios which correctly capture the tail risk of a range of trading strategies, including static portfolios and dynamic strategies such as mean-reversion and trend-following.

Let us consider $M$ assets and $K$ different trading strategies of interest. Denote \( \mathbf{p} = \{(p_{m,t})_{t=1}^{T}\}_{m=1}^{M} \) as the matrix in \( \mathbb{R}^{M \times T} \) that records the price of each asset \( m \), at the beginning of consecutive time intervals with duration \( \Delta \). That is, each price scenario \( \mathbf{p} \) contains the price information of \( M \) assets over a total period of length \( \Delta \times T \). Depending on the purpose of the simulator, \( \Delta \) could range from milliseconds to minutes, and even to days. Each strategy is allocated an initial capital \( c \), and trading decisions are made at discrete timestamps \( t = 1, 2, \ldots, T \). In order for the strategies to be self-financed, there is no exogenous injection or withdrawal of capital. For each strategy \( k = 1, 2, \ldots, K \), a mapping (i.e., a random variable) \( \Pi^k : \mathbb{R}^{M \times T} \rightarrow \mathbb{R} \) is defined to map the price scenarios \( \mathbf{p} \) to the final PnL \( x^k \) at terminal time \( T \), that is, \( \Pi^k(\mathbf{p}) = x^k \).

Finally, we use \( \Pi := (\Pi^1, \ldots, \Pi^K) \) to define the mapping of all strategies. We denote these \( K \) strategies the benchmark strategies. Next, we provide several examples of the self-financed strategies.

**Example 3.1** (Static portfolios). For static portfolios, the strategy can be fully parameterized by an allocation \( \mathbf{w} = (w_1, w_2, \ldots, w_M) \) for which the trader will invest \( w_i \) shares in asset \( i \) at time \( t = 1 \), with \( w_i > 0 \)
representing a long position, \( w_i < 0 \) representing a short position and \( w_i = 0 \) representing an empty position. The associated profit and loss (PnL) of the strategy will be calculated at the end of the trading period \( T \) for the performance of the strategy.

Example 3.2 (Mean-reversion strategy). Many statistical arbitrage strategies [6] are based on identifying combinations \( \mathbf{w} = (w_1, w_2, \ldots, w_M) \) of \( M \) assets whose market price follows a stationary and mean-reversion process [2, 6], using methods such as index tracking or co-integration [2]. The market price \( p_t(\mathbf{w}) = \sum_{i=1}^{M} w_i p_{i,t} \) of such a stationary combination is then expected to revert to its mean, which may be estimated using, for instance, a moving average estimator \( A_t \), leading to the trading signal \( s_t(\mathbf{w}) = p_t(\mathbf{w}) - A_t \) which is expected to revert to zero. Therefore the commonly used mean-reversion strategy, parameterized by \( (\mathbf{w}, c, s_U, s_L) \), can be described as follows: the trader starts with a cash budget \( c > 0 \); if \( s_t(\mathbf{w}) \) exceeds the level \( s_U > 0 \), a short position is taken with all currently available cash until the signal \( s_t(\mathbf{w}) \) falls back to 0. Similarly, a long position is initiated with all available cash if \( s_t(\mathbf{w}) \) falls below the level \( -s_L < 0 \), and is cleared when the signal \( s_t(\mathbf{w}) \) exceeds 0 from below.

Example 3.3 (Trend-following strategy). A trend-following strategy captures potentially profitable opportunities whenever there is a trend or momentum in a signal. The signal could be designed with two moving averages of different durations. Denote \( \overline{p}_t(\mathbf{w}, d) = \frac{1}{d} \sum_{s=t-d+1}^{t} (\sum_{i=1}^{M} w_i p_{i,s}) \) as the average price of the portfolio with duration \( d \in \mathbb{Z}^+ \). The trading signal can then be constructed as the difference between the long-term average and the short-term average \( s_t(\mathbf{w}, d_1, d_2) = p_t(\mathbf{w}, d_1) - \overline{p}_t(\mathbf{w}, d_2) \), with \( d_1 > d_2 > 0 \). Similar to the mean-reversion strategy, the trend-following strategy, fully parameterized by \( (\mathbf{w}, d_1, d_2, c, s_U, s_L) \), can be then defined as follows: the trader starts with a cash budget \( c > 0 \); when \( s_t(\mathbf{w}, d_1, d_2) \) exceeds the level \( s_U > 0 \), a short position is taken with all currently available cash until the signal \( s_t(\mathbf{w}, d_1, d_2) \) falls back to 0. Similarly, a long position is initiated with all the currently available cash if \( s_t(\mathbf{w}, d_1, d_2) \) falls below \( -s_L < 0 \), and is cleared when the signal \( S_t \) exceeds 0 from below.

3.1.2 Loss function of Tail-GAN

Next, we elaborate on how the benchmark strategies introduced above can be employed in the Tail-GAN loss function.

Two-step optimization problem. Given two classes of functions \( \mathcal{G} := \{ G : \mathbb{R}^{N_{z}} \to \mathbb{R}^{M \times T} \} \) and \( \mathcal{D} := \{ D : \mathbb{R}^n \to \mathbb{R}^2 \} \), our goal is to find a generator \( G^* \in \mathcal{G} \) and a discriminator \( D^* \in \mathcal{D} \) via the following
two-step (or constrained) optimization problem
\[
G^* \in \min_{G \in \mathcal{G}} \frac{1}{Kn} \sum_{k=1}^{K} \sum_{j=1}^{n} S_\alpha \left( D^* \left( \Pi^k(q_i); i \in \{1, \ldots, n\} \right), \Pi^k(p_j) \right),
\]
where
\[
D^* \in \arg \min_{D \in \mathcal{D}} \frac{1}{Kn} \sum_{k=1}^{K} \sum_{j=1}^{n} S_\alpha \left( D \left( \Pi^k(p_i); i \in \{1, \ldots, n\} \right), \Pi^k(p_j) \right).
\]

Here \(\{p_i\}_{i=1}^{n}\) are the training data, which we assume to be identically distributed. \(\{q_i := G(z_i)\}_{i=1}^{n}\) are the samples from generator \(G\) with \(z_i \in \mathbb{R}^{N_z}\) the IID random variables (for example, as drawn from the uniform distribution or Student’s t-distribution) as noise inputs.

Denote \(P_r \in \mathcal{P}(\mathbb{R}^{M \times T})\) as the distribution of the training data and \(P_G \in \mathcal{P}(\mathbb{R}^{M \times T})\) as the distribution of the samples from \(G\). In the two-step optimization problem (23)-(24), the discriminator \(D^*\) aims to map \(n\) samples of the \(P_{nL}\) to the associated \(\alpha\)-VaR and \(\alpha\)-ES values of the \(P_{nL}\) distribution. In particular, (24) can be viewed as a finite-sample version of the following optimization problem
\[
\tilde{D}^* \in \arg \min_{D} \mathbb{E}_{p \sim P_r} \left[ \frac{1}{K} \sum_{k=1}^{K} S_\alpha \left( \tilde{D}(P_r(\Pi^k)); p \right) \right],
\]
with \(P_r(\Pi^k) \in \mathcal{P}(\mathbb{R})\) denotes the distribution of \(\Pi^k(p)\) under \(p \sim P_r\) and \(\tilde{D} : \mathcal{P}(\mathbb{R}) \to \mathbb{R}^2\) maps a one-dimensional distribution to two real numbers. Given the definition of the score function and the joint elicitation property of VaR and ES, we have \(\tilde{D}^*(\cdot) := (\text{VaR}_\alpha, \text{ES}_\alpha)(\cdot)\) according to Eqn. (6). Assume \(D^*\) solves Eqn. (24). The simulator \(G^* \in \mathcal{G}\) in (23) aims to map the noise input to a price scenario that has the same VaR and ES values of the strategy \(P_{nL}\)s as the price scenario from \(P_r\).

**Loss function via Lagrangian relaxation.** In practice, constrained optimization problems are difficult to solve, and one can instead relax the constraint by applying the Lagrangian relaxation method with a dual parameter \(\lambda > 0\), leading to a min-max game between two neural networks \(D\) and \(G\),
\[
\max \min_{D \in \mathcal{D}, G \in \mathcal{G}} \frac{1}{Kn} \sum_{k=1}^{K} \sum_{j=1}^{n} \left[ S_\alpha \left( D(\Pi^k(q_i); i \in \{1, \ldots, n\}), \Pi^k(p_j) \right) - \lambda S_\alpha \left( D(\Pi^k(p_i); i \in \{1, \ldots, n\}), \Pi^k(p_j) \right) \right]
\]
where \(p_i, p_j \sim P_r\) and \(q_i \sim P_G\) \((i, j = 1, 2, \ldots, n)\). The discriminator \(D\) takes \(n\) \(P_{nL}\) samples as the input and aims to provide the VaR and ES values of the sample distribution as the output. (See Section 3.3 for
the design of the discriminator.) The score function $S_\alpha$ is defined in Eqn (8). Eqn (25) recovers the original problem (23)-(24) when $\lambda \to +\infty$. The min-max game (25) can also be viewed as a robust version of the optimization problem (23)-(24) with $D$ serving as an adversary playing against $G$. In addition, the min-max structure of (25) encourages the exploration of the generator to simulate scenarios that are not exactly the same as what is observed in the input price scenarios, but are equivalent under the criterion of the score function and loss function, hence improving generalization. We refer the reader to Section 4.2 for a numerical example to demonstrate the generalization power of Tail-GAN and to Section 5 for a comparison between Tail-GAN and supervised learning methods.

![Architecture of Tail-GAN](image.png)

Figure 2: Architecture of Tail-GAN. The (blue) thick arrows represent calculations with learnable parameters, and the (black) thin arrows represent calculations with fixed parameters.

Compared to the Jensen-Shannon divergence used in the loss function of standard GAN architectures [22], the objective function in (25) with the score function $S_\alpha$ is more sensitive to tail risk and leads to an output which better approximates the $\alpha$-ES and $\alpha$-VaR values. We parameterize $G$ and $D$ as multilayer neural networks. The architecture of Tail-GAN is depicted in Figure 2, and the detailed design of the generator and the discriminator are introduced in Sections 3.2 and 3.3, respectively.
3.2 Generator

For the generator we use a neural network with \( L \in \mathbb{Z}^+ \) layers. Denoting by \( n_l \) the width of the \( l \)-th layer, the functional form of the generator is given by

\[
G(z; \gamma) = W_L \cdot \sigma(W_{L-1} \ldots \sigma(W_1 z + b_1) \ldots + b_{L-1}) + b_L,
\]

in which \( \gamma := (W, b) \) represents the parameters in the neural network, with \( W = (W_1, W_2, \ldots, W_L) \) and \( b = (b_1, b_2, \ldots, b_L) \). Here \( W_l \in \mathbb{R}^{n_l \times n_{l-1}} \) and \( b_l \in \mathbb{R}^{n_l \times 1} \) for \( l = 1, 2, \ldots, L \), where \( n_0 = N_z \) is set as the dimension of the input variable. The operator \( \sigma(\cdot) \) takes a vector of any dimension as input, and applies a function component-wise. Specifically, for any \( q \in \mathbb{Z}^+ \) and any vector \( u = (u_1, u_2, \ldots, u_q) \top \in \mathbb{R}^q \), we have that

\[
\sigma(u) = (\sigma(u_1), \sigma(u_2), \ldots, \sigma(u_q)) \top.
\]

In the neural networks literature, the \( W_l \)'s are often called the weight matrices, the \( b_l \)'s are called bias vectors, and \( \sigma(\cdot) \) is referred to as the activation function. Several popular choices for the activation function include ReLU with \( \sigma(u) = \max(u, 0) \), Leaky ReLU with \( \sigma(u) = a_1 \max(u, 0) - a_2 \max(-u, 0) \) and \( a_1, a_2 > 0 \), and smooth functions such as \( \sigma(\cdot) = \tanh(\cdot) \).

We define \( G \) as a class of generators that satisfy certain given regularity conditions on the neural network parameters.

\[
G(L, n_1, n_2, \ldots, n_L) = \left\{ G : \mathbb{R}^{N_z} \to \mathbb{R}^{M \times T} \mid G \text{ takes the form in (26) with } L \text{ layers and } n_l \text{ as the width of each layer, } \|W_l\|_\infty, \|b_l\|_\infty < \infty \text{ for } l = 1, 2, \ldots, L \right\},
\]

where \( \| \cdot \|_\infty \) denotes the max-norm that takes the max absolute value of all elements in the input matrix or vector. To ease the notation, we may use the abbreviation \( G_\gamma(\cdot) \) or drop the dependency of \( G(\cdot; \gamma) \) on the neural network parameters \( \gamma \) and conveniently write \( G(\cdot) \). We further denote \( P_G \) as the distribution of price series generated by \( G \) under the initial distribution \( z \sim P_z \).

3.3 Discriminator

Given \( n \) price scenarios \( \{\mathbf{q}_i\}_{i=1}^n \) from \( P_G \), and \( n \) price scenarios \( \{\mathbf{p}_i\}_{i=1}^n \) from \( P_r \), our goal is to learn a mapping \( D \) from \( \{\Pi^k(\mathbf{q}_i)\}_{i=1}^n \) and \( \{\Pi^k(\mathbf{p}_i)\}_{i=1}^n \) to the corresponding \( \alpha \)-VaR and \( \alpha \)-ES values, i.e., \( (\text{VaR}_\alpha(\Pi^k; P_G), \text{ES}_\alpha(\Pi^k; P_G)) \) and \( (\text{VaR}_\alpha(\Pi^k; P_r), \text{ES}_\alpha(\Pi^k; P_r)) \), respectively. We search \( D \) over all Lipschitz functions parameterized by
Discriminator of Tail-GAN

The architecture of the Tail-GAN Discriminator has two key ingredients, as depicted in Figure 3. For the first ingredient, a differentiable sorting algorithm [23] is employed to rank the PnLs. The second part adopts a similar form of (27), taking the ranked PnLs as the input, and providing the estimated \( \alpha \)-VaR and \( \alpha \)-ES values as the output.

- **First ingredient of the Tail-GAN discriminator:** Differentiable neural sorting. For a fixed sample size \( n \), the empirical distribution \( \sum_{i=1}^{n} \frac{1(x_i \leq \cdot)}{n} \) constructed with \( n \) samples \( \{x_1, \ldots, x_n\} \) is equivalent to the set of samples themselves \( \{x_1, \ldots, x_n\} \). Therefore, we consider PnL samples with a fixed size \( n \) as the input of the discriminator.

The \( \alpha \)-VaR of a distribution can be approximated by the \( \lfloor \alpha n \rfloor \)th smallest value in a sample of size \( n \) from this distribution, which is permutation-invariant to the ordering of the samples. Therefore, including the sorting function in our architecture design could potentially improve the stability of the discriminator, the goal of which is to predict the \( \alpha \)-VaR and \( \alpha \)-ES for the PnL distribution of a given strategy. We follow the design in [23] to include the differentiable sorting architecture, so that the input of the discriminator will be the ranked PnL’s (sorted in decreasing order). This design, based on the idea of using the **SOFT-MAX** operator to approximate the \( \text{ARG-MAX} \) operator, enables back-propagation of the gradient of the sorting function during the network training process.

Denote \( x^k = (x_1^k, x_2^k, \ldots, x_n^k)^\top \) as a real-valued vector of length \( n \), representing the PnL samples of strategy \( k \). Let \( B(x^k) \) denote the matrix of absolute pairwise differences of the elements of \( x^k \), such that \( B_{i,j}(x^k) = |x_i^k - x_j^k| \). We then define the following permutation matrix \( \Gamma(x^k) \) following [23, 33]

\[
\Gamma_{i,j}(x^k) = \begin{cases} 
1, & \text{if } j = \text{arg max}((n + 1 - 2i) - B(x^k)1), \\
0, & \text{otherwise},
\end{cases}
\]

\( (28) \)
In particular, the almost everywhere with respect to the elements of in which operator with permutation matrix for gradient computation. Instead, Grover et al. [23] propose to replace the ARG-MAX operator with SOFT-MAX, in order to obtain a continuous relaxation \( \hat{\Gamma}^\tau \) with a temperature parameter \( \tau > 0 \). In particular, the \((i,j)\)-th element of \( \hat{\Gamma}^\tau(x^k) \) is given by

\[
\hat{\Gamma}^\tau_{i,j}(x^k) = \frac{\exp \left( (n + 1 - 2i) - B(x^k)_i \right)}{\sum_{i=1}^n \exp \left( (n + 1 - 2i) - B(x^k)_i \right)},
\]

in which \( B(x^k)_i \) is the \( i \)-th row of matrix \( B(x^k) \). This relaxation is continuous everywhere and differentiable almost everywhere with respect to the elements of \( x^k \). In addition, [23, Theorem 4] shows that \( \hat{\Gamma}^\tau_{i,j}(x^k) \) converges to \( \Gamma_{i,j}(x^k) \) almost surely when \( x^k_1, \ldots, x^k_n \) are sampled IID from a distribution which is absolutely continuous with respect to the Lebesgue measure in \( \mathbb{R} \).

- **Second ingredient of the Tail-GAN discriminator:** Multi-layer neural network. The second part adopts a similar neural network architecture as (26) with \( \tilde{L} \) layers, with the exceptions that the input dimension is \( \tilde{n}_1 := n \) and the output dimension is \( \tilde{n}_{\tilde{L}} := 2 \). In summary, the discriminator is the composition of the above mentioned two ingredients, and is given by

\[
D(x^k; \delta) = \tilde{W}_{\tilde{L}} \cdot \sigma \left( \tilde{W}_{\tilde{L}-1} \cdots \sigma (\tilde{W}_1 \hat{\Gamma}^\tau(x^k) + b_1) \cdots + b_{\tilde{L}-1} \right) + b_{\tilde{L}},
\]

where \( \delta = (\tilde{W}, \tilde{b}) \) represent all the parameters in the neural network. Here \( \tilde{W} = (\tilde{W}_1, \tilde{W}_2, \ldots, \tilde{W}_{\tilde{L}}) \) and \( \tilde{b} = (b_1, b_2, \ldots, b_{\tilde{L}}) \) with \( \tilde{W}_l \in \mathbb{R}^{n_l \times n_{l-1}}, \ b_l \in \mathbb{R}^{n_l \times 1} \) for \( l = 1, 2, \ldots, \tilde{L} \). The outputs of the discriminator are two values, representing the \( \alpha \)-VaR and \( \alpha \)-ES. Accordingly, we define \( D \) as a class of discriminators

\[
D(\tilde{L}, \tilde{n}_1, \ldots, \tilde{n}_{\tilde{L}}) = \left\{ D: \mathbb{R}^n \rightarrow \mathbb{R}^2 \mid D \text{ takes the form in (30) with } \tilde{L} \text{ layers and } n_l \text{ as the width of each layer, } \|\tilde{W}_l\|_\infty, \|\tilde{b}_l\|_\infty < \infty \text{ for } l = 1, 2, \ldots, \tilde{L} \right\}.
\]

We sometimes use the abbreviation \( D_\delta \) or \( D \) instead of \( D(\cdot; \delta) \) for notation simplicity. The table below summarizes the Tail-GAN algorithm.

## 4 Model validation

In this section, we test the performance of Tail-GAN on two synthetic data sets, for which we can validate the performance of Tail-GAN by comparing to the true input price scenario distribution. We divide each synthetic data set into two subsets, i.e. the training data and the testing data, with no overlap in time.
Algorithm 1 Tail-GAN.

Input:
- Price scenarios $\mathbf{p}_1, \ldots, \mathbf{p}_N \in \mathbb{R}^{M \times T}$.
- Description of trading strategies: $\Pi = (\Pi^1, \ldots, \Pi^K) : \mathbb{R}^{M \times T} \to \mathbb{R}^K$.
- Hyperparameters: learning rate $l_D$ for discriminator and $l_G$ for generator; number of training epochs; batch size $N_B$; dual parameter $\lambda$.

1: for number of epochs do
2:   for $j = 1 \rightarrow \lfloor N/N_B \rfloor$ do
3:     Generate $N_B$ IID noise samples $\{\mathbf{z}_i, i \in [N_B]\} \sim P_z$.
4:     Sample a batch $\mathcal{B}_j \subset \{1, \ldots, N\}$ of size $N_B$ from the input data $\mathcal{P}_i, i = 1, \ldots, N$.
5:     Compute the loss of the discriminator on the batch $\mathcal{B}_j$
\begin{align*}
L_D(\delta) &= \frac{1}{KN_B} \sum_{k=1}^K \sum_{n \in \mathcal{B}_j} \left[ S_\alpha(D_\delta(\Pi^k(G_\gamma(\mathbf{z}_i)), i \in [N_B]), \Pi^k(p_n)) \right. \\
&\quad \left. - \lambda S_\alpha(D_\delta(\Pi^k(p_i), i \in \mathcal{B}_j), \Pi^k(p_n)) \right].
\end{align*}
6:     Update the discriminator
\[\delta \leftarrow \delta + \frac{l_D}{N_B} \nabla L_D(\delta).\]
7:     Generate $N_B$ IID noise samples $\{\tilde{\mathbf{z}}_i, i \in [N_B]\} \sim P_z$.
8:     Compute the loss of the generator
\begin{align*}
L_G(\gamma) &= \frac{1}{KN_B} \sum_{k=1}^K \sum_{n \in \mathcal{B}_j} S_\alpha(D_\delta(\Pi^k(G_\gamma(\tilde{\mathbf{z}}_i)), i \in [N_B]), \Pi^k(p_n)).
\end{align*}
9:     Update the generator
\[\gamma \leftarrow \gamma - \frac{l_G}{N_B} \nabla L_G(\gamma).\]
10: end for
11: end for
12: $\delta^* = \gamma, \quad \delta^* = \delta$.

Outputs:
- $\delta^*$: trained discriminator weights; $\gamma^*$: trained generator weights.
- Simulated scenarios $G_\gamma^*(\mathbf{z}_i)$ where $\mathbf{z}_i \sim P_z$ IID.
The training data is used to estimate the model parameters, and the testing data is used to evaluate the out-of-sample performance of different models. In this examination, 50,000 samples are used for training and 10,000 samples are used for performance evaluation.

For ease of exposition, unless otherwise stated, we denote Tail-GAN as the model (introduced in Section 3) trained with both static and dynamic trading strategies across multiple assets. We compare Tail-GAN with three benchmark models:

- (1) Tail-GAN-Raw: Tail-GAN trained (only) with raw returns where buy-and-hold strategies with individual assets are used in the evaluation criterion,

- (2) Tail-GAN-Static: Tail-GAN trained (only) with static multi-asset portfolios.

- (3) Historical Simulation Method (HSM): Using VaR and ES computed from historical data as the approximation for VaR and ES of future data.

As Tail-GAN-Raw is trained with the PnLs of single-asset portfolios, it shares the same spirit as Quant-GAN [42] as well as many GAN-based generators for financial time series [25, 29, 30, 32, 39]. Tail-GAN-Static is trained with the PnLs of multi-asset portfolios which is more flexible than Tail-GAN-Raw by allowing different capital allocations among different assets.

We consider, in Section 4.1, the following criteria to compare the performance of scenarios simulated using Tail-GAN with other simulation approaches: (1) tail behavior comparison; (2) structural characterizations such as correlation and auto-correlation; and (3) model verification via (statistical) hypothesis testings such as Score-based test and Coverage test.

The main take-away from our comparison against benchmark architectures is that the consistent tail-risk evaluation of dynamic trading strategies is difficult to achieve by only training on price sequence, without incorporating these dynamic trading strategies in the evaluation criterion, as we propose to do in our pipeline.

4.1 Methodology

We start by introducing general methodologies which are later applied throughout the numerical analysis.

**Tail behavior.** To evaluate how closely the estimates of VaR (ES) of strategy PnLs, computed under the generated scenarios, match the ground-truth VaR (ES) of the strategy PnLs computed under input scenarios, we employ both quantitative and qualitative assessment methods to gain a thorough understanding.

The quantitative performance measure is relative error of VaR and ES. For any strategy $k$ ($1 \leq k \leq K$), the relative error of VaR is defined as $\left| \frac{\text{VaR}_\alpha (\Pi^k; \mathbb{P}_r^n) - \text{VaR}_\alpha (\Pi^k; \mathbb{P}_r)}{\text{VaR}_\alpha (\Pi^k; \mathbb{P}_r^n)} \right|$, where $\text{VaR}_\alpha (\Pi^k; \mathbb{P}_r)$ is the $\alpha$-VaR of the
PnL for strategy $k$ evaluated under $\mathbb{P}_r$ and $\text{VaR}_\alpha \left( \Pi^k; \mathbb{P}^{(n)}_G \right)$ is the empirical estimate of VaR for strategy $k$ evaluated with $n$ samples under $\mathbb{P}_G$. Similarly, for the estimates $\text{ES}_\alpha \left( \Pi^k; \mathbb{P}^{(n)}_G \right)$, we define the relative error as $\frac{\left| \text{ES}_\alpha \left( \Pi^k; \mathbb{P}^{(n)}_G \right) - \text{ES}_\alpha \left( \Pi^k; \mathbb{P}_r \right) \right|}{\left| \text{ES}_\alpha \left( \Pi^k; \mathbb{P}_r \right) \right|}$ for the ES of strategy $k$. We then use the following average relative errors of VaR and ES as the overall measure of model performance

$$\text{RE}(n) = \frac{1}{2K} \sum_{k=1}^{K} \left( \frac{\left| \text{VaR}_\alpha \left( \Pi^k; \mathbb{P}^{(n)}_G \right) - \text{VaR}_\alpha \left( \Pi^k; \mathbb{P}_r \right) \right|}{\left| \text{VaR}_\alpha \left( \Pi^k; \mathbb{P}_r \right) \right|} + \frac{\left| \text{ES}_\alpha \left( \Pi^k; \mathbb{P}^{(n)}_G \right) - \text{ES}_\alpha \left( \Pi^k; \mathbb{P}_r \right) \right|}{\left| \text{ES}_\alpha \left( \Pi^k; \mathbb{P}_r \right) \right|} \right).$$

(32)

One useful benchmark is to compare the above relative error with the sampling error bellow, when only using a finite number of real samples to calculate VaR and ES

$$\text{SE}(n) = \frac{1}{2K} \sum_{k=1}^{K} \left( \frac{\left| \text{VaR}_\alpha \left( \Pi^k; \mathbb{P}^{(n)}_r \right) - \text{VaR}_\alpha \left( \Pi^k; \mathbb{P}_r \right) \right|}{\left| \text{VaR}_\alpha \left( \Pi^k; \mathbb{P}_r \right) \right|} + \frac{\left| \text{ES}_\alpha \left( \Pi^k; \mathbb{P}^{(n)}_r \right) - \text{ES}_\alpha \left( \Pi^k; \mathbb{P}_r \right) \right|}{\left| \text{ES}_\alpha \left( \Pi^k; \mathbb{P}_r \right) \right|} \right),$$

(33)

where $\text{VaR}_\alpha \left( \Pi^k; \mathbb{P}^{(n)}_r \right)$ and $\text{ES}_\alpha \left( \Pi^k; \mathbb{P}^{(n)}_r \right)$ are the estimates for VaR and ES of strategy $k$ using market data with sample size $n$.

GANs are usually trained with a finite number of samples, and thus it is difficult for the trained GAN to achieve a better sampling error of the training data with the same sample size. To this end, we may conclude that the trained GAN reaches its best finite-sample performance if $\text{RE}(n)$ is comparable to $\text{SE}(n)$. This benchmark comparison and model validation steps are important to evaluate the performance of GAN models. However, these steps are missing in the GAN literature for simulating financial time series [42].

The qualitative assessment method we employ is a rank-frequency distribution, which is a discrete form of the quantile function, i.e., the inverse cumulative distribution, giving the size of the element at a given rank. By comparing the rank-frequency distribution of the market data and simulated data of different strategies, we gain an understanding of how good the simulated data is in terms of the risk measures of different strategies. It is worth pointing out that a rank-frequency distribution is neither a probability distribution nor a cumulative distribution function.

**Structural characterization.** We are interested in testing whether Tail-GAN is capable of capturing structural properties, such as temporal and spatial correlations, of the input price scenarios. To do so, we calculate and compare the following statistics of the price scenarios generated by each simulator: (1) the sum of the absolute difference between the correlation coefficients of the input price scenario and those of generated price scenario, and (2) the sum of the absolute difference between the autocorrelation coefficients (up to 10 lags) of the input price scenario and those of the generated price scenario.
Simulation experiments. Given the benchmark strategies introduced in Section 3.1.1 and a simulation method (generically referred to as M), we are interested in testing (or rejecting) whether risk measure for benchmark strategies estimated from simulated scenarios M are as accurate as ‘oracle’ estimates given knowledge of the true data generating process. Here, M may represents variants of our method: Tail-GAN, Tail-GAN-Raw, Tail-GAN-Static, or empirical estimators based on the same sample size.

We explore two methods, Score-based Test and Coverage Test, to verify the relationship between the simulator M and the true model. We first introduce the Score-based Test to verify the hypothesis

\[ H_0 : \mathbb{E}_{p \sim \mathbb{P}_r} \left[ S_\alpha \left( \text{VaR}_\alpha \left( \Pi^k; \mathbb{P}_M \right), \text{ES}_\alpha \left( \Pi^k; \mathbb{P}_M \right), \Pi^k(p) \right) \right] = \mathbb{E}_{p \sim \mathbb{P}_r} \left[ S_\alpha \left( \text{VaR}_\alpha \left( \Pi^k; \mathbb{P}_r \right), \text{ES}_\alpha \left( \Pi^k; \mathbb{P}_r \right), \Pi^k(p) \right) \right]. \]

By making use of the joint elicitability property of VaR and ES, Fissler et al. [18] proposed the following test statistic to verify \( H_0 \)

\[ T^k = \frac{\bar{S}^k_M - \bar{S}^k_{\text{Ground-Truth}}}{\hat{\sigma}^k}, \]

where

\[ \bar{S}^k_M = \frac{1}{n} \sum_{i=1}^{n} S_\alpha \left( \text{VaR}_\alpha \left( \Pi^k; \mathbb{P}_M \right), \text{ES}_\alpha \left( \Pi^k; \mathbb{P}_M \right), \Pi^k(p_i) \right), \]

\[ \bar{S}^k_{\text{Ground-Truth}} = \frac{1}{n} \sum_{i=1}^{n} S_\alpha \left( \text{VaR}_\alpha \left( \Pi^k; \mathbb{P}_r \right), \text{ES}_\alpha \left( \Pi^k; \mathbb{P}_r \right), \Pi^k(p_i) \right), \]

\[ \hat{\sigma}^k = \sqrt{\frac{s^2_M + s^2_{\text{Ground-Truth}}}{n}}. \]

Here \( \{p_i\}_{i=1}^{n} \) represents the observations from \( \mathbb{P}_r \) and \( \{\Pi^k(p_i)\}_{i=1}^{n} \) represents the PnL observations of strategy \( k \) under \( \mathbb{P}_r \). \( \mathbb{P}_M \) denotes the distribution of generated data from simulator M. \( \text{VaR}_\alpha \left( \Pi^k; \mathbb{P}_M \right) \) and \( \text{ES}_\alpha \left( \Pi^k; \mathbb{P}_M \right) \) represent the estimates of VaR and ES for PnLs of strategy \( k \) evaluated under \( \mathbb{P}_M \). \( \text{VaR}_\alpha \left( \Pi^k; \mathbb{P}_r \right) \) and \( \text{ES}_\alpha \left( \Pi^k; \mathbb{P}_r \right) \) represent the groundtruth estimates of VaR and ES for PnLs of strategy \( k \) evaluated under \( \mathbb{P}_r \). \( s^2_M \) and \( s^2_{\text{Ground-Truth}} \) are the empirical variances of \( S_\alpha \left( \text{VaR}_\alpha \left( \Pi^k; \mathbb{P}_M \right), \text{ES}_\alpha \left( \Pi^k; \mathbb{P}_M \right), \Pi^k(p) \right) \) and \( S_\alpha \left( \text{VaR}_\alpha \left( \Pi^k; \mathbb{P}_r \right), \text{ES}_\alpha \left( \Pi^k; \mathbb{P}_r \right), \Pi^k(p) \right) \), respectively.

Under \( H_0 \), the test statistic \( T^k \) has expected value equal to zero, and the asymptotic normality of the test statistics \( T^k \) can be similarly proved as in [13].

The second test we explore is the Coverage Test, also known as Kupiec Test [10, 26]. It measures the simulator performance by comparing the observed violation rate of estimates from a simulator with the
expected violation rate. The null hypothesis of Coverage Test is

\[ H_0 : \mathbb{P}(\Pi^k(p) < \text{VaR}_{\alpha}(\Pi^k; P_M)) = \alpha. \]

Here \( \Pi^k(p) \) is a random variable which represents the PnL of strategy \( k \) under \( P_r \).

Kupiec [26] proposed the following statistics, which is a likelihood ratio between two Binomial likelihoods, to verify the null hypothesis. Furthermore, [26, 28] proved that that \( LR \sim \chi^2_1 \) under \( H_0 \)

\[ LR = -2 \ln \left( \frac{(1 - \alpha)^{n-C^k(n)} \alpha^{C^k(n)}}{(1 - \frac{C^k(n)}{n})^{n-C^k(n)} \left( \frac{C^k(n)}{n} \right)^{C^k(n)}} \right), \tag{37} \]

where \( C^k(n) = \sum_{i=1}^{n} 1_{\{\Pi^k(p_i) < \text{VaR}_{\alpha}(\Pi^k; P_M)\}} \) represents the number of violations observed in the estimates from simulator \( M \) and \( \{\Pi^k(p_i)\}_{i=1}^{n} \) represents the PnL observations of strategy \( k \) under \( P_r \).

**In-sample test vs out-of-sample test.** Throughout the experiments, both an in-sample test and an out-of-sample test are used to evaluate the trained simulators. In particular, the in-sample test is performed on the training data set, whereas the out-of-sample test is performed with the testing data set. For each trained simulator, the in-sample test uses the same set of strategies as in its loss function. For example, in-sample test for TAIL-GAN-Raw is performed with buy-and-hold strategies on individual assets. On the other hand, all benchmark strategies specified in Section 3.1.1 are used in the out-of-sample test for every simulator.

### 4.2 Demonstration with a toy example - the single-asset scenario

We first illustrate the performance of TAIL-GAN on a one-dimensional toy example, where the price series of the training data have a clear V-shape pattern, as illustrated in Figure 4(a).

**Design of \( P_r \).** To generate a one-dimensional V-shape pattern, we assume the input price scenario follows \( p_t = p_0 + \sum_{s=1}^{t} \Delta p_s \) with \( p_0 = 1, \Delta p_s = u_s + \varepsilon_s \) and \( \{\varepsilon_s\}_{s=1}^{100} \sim \mathcal{N}(0, 1 \times 10^{-3}) \). Here \( u_s \sim \mathcal{U}(-1.5 \times 10^{-3}, -8 \times 10^{-4}) \) for \( 1 \leq s \leq 50 \) and \( u_s \sim \mathcal{U}(4 \times 10^{-4}, 8 \times 10^{-4}) \) for \( 51 \leq s \leq 100 \). This model generates price scenarios with a decrease in the first half, followed by an increase in the second half. See Figure 4(a) for some simulated paths.

**Experiment Set-up.** TAIL-GAN is trained with three strategies: a static portfolio, a mean-reversion strategy and a trend-following strategy. We set the moving average window to be 10 in the mean-reversion
strategy, and $d_1 = 10$ and $d_2 = 20$ in the trend-following strategy. For both strategies, $s_U$ and $s_L$ are half of the standard deviations of the trading signals. (See Section 3.1.1 for the detailed introduction of the strategies.) Note that Tail-GAN-Static is equivalent to Tail-GAN-Raw with one single asset. Hence it suffices to compare the performance of Tail-GAN with that of Tail-GAN-Raw.

![Input price scenarios](image1.png) ![Paths generated by Tail-GAN](image2.png) ![Generated log paths from Tail-GAN](image3.png)

Figure 4: Comparison of 50 random paths ($\log(p_t/p_0)$) of different models.

![Static buy-and-hold strategy](image4.png) ![Mean-reversion strategy](image5.png) ![Trend-following strategy](image6.png)

Figure 5: Distribution of strategy PnLs computed from input price scenarios (blue) and simulated price scenarios from Tail-GAN (orange).

**Comparison of Tail-GAN simulators.** We train Tail-GAN-Raw and Tail-GAN on the single asset with V-shape pattern as described above. A set of 50 price scenarios generated from the trained Tail-GAN-Raw and Tail-GAN are shown in Figures 4(b) and 4(c), respectively. We observe that the price scenarios generated by Tail-GAN-Raw have no clear structure, whereas the price scenarios generated by Tail-GAN have a clear pattern with a decreasing trend in the first half of the time steps, and an increasing trend in the second half. Based on these observations, we thus draw the following conclusions.
• The loss function of Tail-GAN-Raw only involves the PnL of the buy-and-hold strategy. This implies that Tail-GAN-Raw can only capture the relative difference between the final price at time $T$ and the initial price at time $0$, with no incentive to mimic the dynamics in between. To conclude, Tail-GAN-Raw is not able to capture the temporal information along the price trajectory.

• Tail-GAN is able to capture the temporal information along the price trajectory via the PnLs of dynamic strategies. The price scenarios plotted in Figure 4(c) are similar to the samples from $\mathbb{P}_r$, as plotted in Figure 4(a). To further demonstrate the advantage of Tail-GAN quantitatively, we compare in Figure 5 the distributions of the strategy PnLs of three different types of strategies, namely static buy-and-hold, mean-reversion and trend-following strategies. We observe that the PnL distributions of all three strategies evaluated with the simulated scenarios from Tail-GAN are similar to the ones of the input price scenario.

4.3 Multi-asset scenario

Synthetic financial scenarios. In this section, we simulate five financial instruments under a given correlation structure, with different temporal patterns and tail behaviors in return distributions. The marginal distributions of these assets are: Gaussian distribution, AR(1) with autocorrelation $\phi_1 > 0$, AR(1) with autocorrelation $\phi_2 < 0$, GARCH(1, 1) with $t(\nu_1)$ noise and GARCH(1, 1) with $t(\nu_2)$ noise. Here, $t(\nu)$ denotes the Student’s t-distribution with degree of freedom $\nu$. The AR(1) models with positive and negative autocorrelations represent the trending scenario and mean-reversion scenario, respectively. The GARCH(1,1) models with noise from Student’s t-distribution with different degrees of freedom provide us with heavy-tailed return distributions.

Mathematically, for any given time $t \in [0, T]$, we first sample $u_t = (u_{1,t}, \ldots, u_{5,t})^T \sim \mathcal{N}(0, \Sigma)$ with covariance matrix $\Sigma \in \mathbb{R}^{5 \times 5}$, $v_{1,t} \sim \chi^2(\nu_1)$ and $v_{2,t} \sim \chi^2(\nu_2)$. Here $v_{1,t}$, $v_{2,t}$ are independent of $u_t$. We then calculate the price increments according to the following equations

$$
\Delta p_{1,t} = u_{1,t}; \quad \Delta p_{2,t} = \phi_1 \Delta p_{2,t-1} + u_{2,t}; \quad \Delta p_{3,t} = \phi_2 \Delta p_{3,t-1} + u_{3,t};
$$

$$
\Delta p_{4,t} = \varepsilon_{4,t} = \sigma_{4,t} \eta_{1,t}; \quad \Delta p_{5,t} = \varepsilon_{5,t} = \sigma_{5,t} \eta_{2,t}, \quad (38)
$$

where $\sigma_{4,t}^2 = \gamma_4 + \kappa_4 \varepsilon_{4,t-1}^2 + \beta_4 \sigma_{4,t-1}^2$, $\eta_{1,t} = \frac{u_{4,t}}{\sqrt{\nu_{1,t}/\nu_1}}$, and $\sigma_{5,t}^2 = \gamma_5 + \kappa_5 \varepsilon_{5,t-1}^2 + \beta_5 \sigma_{5,t-1}^2$, $\eta_{2,t} = \frac{u_{5,t}}{\sqrt{\nu_{2,t}/\nu_2}}$. 26
|                        | Static buy-and-hold |                  | Mean-reversion |                  | Trend-following |                  |
|------------------------|----------------------|------------------|----------------|------------------|-----------------|------------------|
|                        | VaR                  | ES               | VaR            | ES               | VaR             | ES               |
| Gaussian               | -0.489               | -0.615           | -0.432         | -0.553           | -0.409          | -0.515           |
| AR(1) with $\phi_1 = 0.5$ | -0.876               | -1.100           | -0.850         | -1.066           | -0.671          | -0.829           |
| AR(1) with $\phi_2 = -0.12$ | -0.461               | -0.581           | -0.399         | -0.513           | -0.387          | -0.488           |
| GARCH(1,1) with $t(5)$ | -0.480               | -0.603           | -0.420         | -0.535           | -0.400          | -0.501           |
| GARCH(1,1) with $t(10)$ | -0.403               | -0.507           | -0.354         | -0.453           | -0.328          | -0.410           |

Table 1: Empirical VaR and ES values for trading strategies evaluated on the training data.

Table 1 reports the 5%-VaR and 5%-ES values of several strategies calculated with the synthetic financial scenarios designed above.

**Setup.** We set $T = 100$ as the number of observations over one trading day. We first generate a correlation matrix $\rho$ with elements uniformly sampled from $[0, 1]$. We then sample the annualized standard deviations $s$ with values between 0.3 and 0.5, and set $\Sigma_{ij} = \frac{s_i s_j}{255 \times T} \rho_{ij}$ ($i, j = 1, 2, \ldots, 5$); $\phi_1 = 0.5$ and $\phi_2 = -0.15$; $\nu_1 = 5$ and $\nu_2 = 10$; $\kappa_4$ and $\kappa_5$ are sampled uniformly from $[0.08, 0.12]$; $\beta_4$ and $\beta_5$ are sampled uniformly from $[0.825, 0.875]$; and finally $\gamma_4$ and $\gamma_5$ are sampled uniformly from $[0.03, 0.07]$. We choose one quantile $\alpha = 0.05$ for this experiment. The Adam stochastic optimization is used in training. The architecture of the network configuration is summarized in Table 2.

![Table 2: Network architecture configuration.](image-url)

Figure 6 reports the convergence of in-sample errors, and Table 3 summarizes the out-of-sample errors of Tail-GAN-Raw, Tail-GAN-Static, and Tail-GAN.
Performance accuracy. We draw the following observations from Figure 6 and Table 3.

- For the evaluation criterion RE(1000) (see Figure 6), all three simulators, Tail-GAN-Raw, Tail-GAN-Static and Tail-GAN, converge within 2000 epochs with errors smaller than 10%. This implies that all three generators are able to capture the static information contained in the market data.

- For the evaluation criterion RE(1000), with both static portfolios and dynamic strategies on out-of-sample tests (see Table 3), only Tail-GAN converges to an error 4.6%, whereas the other two generators fail to capture the dynamic information in the market data.

- Compared to Tail-GAN-Raw and Tail-GAN-Static, Tail-GAN has the lowest training variance across multiple experiments (see standard deviations in Table 3). This implies that Tail-GAN has the most stable performance among all three simulators.

| Out-Of-Sample Error (%) | SE(1000) | HSM | Tail-GAN-Raw | Tail-GAN-Static | Tail-GAN |
|-------------------------|----------|-----|--------------|----------------|---------|
|                         | 3.0      | 3.4 | 83.3         | 86.7           | 4.6     |
|                         | (2.2)    | (2.6)| (3.0)        | (2.5)          | (1.6)   |

Table 3: Mean and standard deviation (in parentheses) of relative errors for out-of-sample tests. We recall that HSM denotes the Historical Simulation Method, using VaR and ES computed from historical data as the approximation for VaR and ES of future data. (Each experiment is repeated five times with different random seeds.)
Figure 7: Tail behavior via the empirical rank-frequency distribution of the strategy PnL. The rows index the various models used for generating the ground truth synthetic data, while the column index the strategy types.

Figure 7 shows the empirical quantile function of the strategy PnLs evaluated with price scenarios sampled from Tail-GAN-Raw, Tail-GAN-Static and Tail-GAN. The testing strategies are, from left to right (in Figure 7), static single-asset portfolio, single-asset mean-reversion strategy and single-asset trend-following strategy. The i-th asset, from top to bottom in Figure 7, is the i-th row of the price scenario. The (marginal)
distribution of each asset in the input price scenario is marked on the left of each row, following the dynamics in Eqn (38). In each subfigure, we compare the rank-frequency distribution of strategy PnLs evaluated with input price scenario (in blue), and three Tail-GAN simulators (in orange, red and green). Based on the results depicted in Figure 7, we conclude that

- All three simulators are able to capture the tail properties of the static single-asset portfolio at quantile levels above 1%, as shown in the first column of Figure 7.
- For the PnL distribution of the dynamic strategies, only Tail-GAN is able to generate scenarios with compatible tail statistics of the PnL distribution, as shown in the second and third columns of Figure 7. That is, only data sampled from Tail-GAN can correctly describe the tail distribution of trend-following and mean-reversion strategies.
- Tail-GAN-Raw and Tail-GAN-Static underestimate the risk of the mean-reversion strategy at $\alpha = 5\%$ quantile level, and overestimate the risk of the trend-following strategy at $\alpha = 5\%$ quantile level, as illustrated in the second and third columns of Figure 7.

Learning the temporal and correlation patterns. Figures 8 and 9 show the correlation and auto-correlation patterns of market data (Figures 8(a) and 9(a)) and simulated data from Tail-GAN-Raw (Figures 8(b) and 9(b)), Tail-GAN-Static (Figures 8(c) and 9(c)) and Tail-GAN (Figures 8(d) and 9(d)).

Figure 8: Correlations of the price increments from different trained GAN models: (1) Tail-GAN-Raw, (2) Tail-GAN-Static, and (3) Tail-GAN. Numbers at the top of each plot denote the mean and standard deviation (in parentheses) of the sum of the absolute element-wise difference between the correlation matrices, computed with 10,000 training samples and 10,000 generated samples.
Figure 9: Auto-correlations of the price increments from different trained GAN models: (1) Tail-GAN-Raw, (2) Tail-GAN-Static, and (3) Tail-GAN. Numbers at the top of each plot denote the mean and standard deviation (in parentheses) of the sum of the absolute difference between the auto-correlation coefficients computed with 10,000 training samples and 10,000 generated samples.

Figures 8 and 9, show that auto-correlation and cross-correlations returns are best reproduced by Tail-GAN, trained on multi-asset dynamic portfolios, while Tail-GAN-Raw, trained on raw returns, has the lowest accuracy in this respect. This illustrates the importance of training the algorithm on returns of benchmark strategies.

Model accuracy. Table 4 summarizes the statistical test results for Tail-GAN, Tail-GAN-Raw, Tail-GAN-Static, and Historical Simulation Method. Table 4 suggests that Tail-GAN achieves the lowest average rejection rate of the null hypothesis described in Section 4.1. In other words, Tail-GAN produces more accurate estimates of VaR and ES for benchmark strategies compared to other simulators.

\[
\begin{array}{|c|c|c|c|c|}
\hline
& HSM & Tail-GAN-Raw & Tail-GAN-Static & Tail-GAN \\
\hline
\text{Score-based Test (%) } & 0.00 & 21.3 & 15.4 & 0.00 \\
\hline
\text{Coverage Test (%) } & 17.0 & 53.6 & 22.9 & 16.2 \\
\hline
\end{array}
\]

Table 4: Average rejection rate of the null hypothesis in two tests across strategies. We use sample size \( n = 1,000 \) and repeat the above experiment 100 times on testing data.

5 Generalization error and Scalability

This section focuses on two desirable aspects of machine learning algorithms, namely generalization error and scalability, in the context of synthetic price scenarios.
5.1 Generalization error

A learning algorithm is said to generalize well if the training and test errors closely track each other [5]. Generalization error quantifies the ability of machine learning models to capture certain inherent properties from the data or the ground-truth model. In general, machine learning models with good generalization property are meant to learn “underlying rules” associated with the data generation process, rather than only memorize the training data, so that they are able to extrapolate learned rules from the training data to new unseen data. Thereby, the generalization error of a generator $G$ can be measured as the difference between the empirical divergence of the training data $d(\mathbb{P}_r^{(N)}, \mathbb{P}_G^{(N)})$ and the ground-truth divergence $d(\mathbb{P}_r, \mathbb{P}_G)$.

To provide a systematic quantification of the generalization capability, we adopt the notion of generalization from [5]. A generated distribution $\mathbb{P}_G$ is said to generalize, under the divergence $d(\cdot, \cdot)$ and with generalization error $\varepsilon$, if the following property holds with high probability

$$\left| d(\mathbb{P}_r^{(n)}, \mathbb{P}_G^{(n)}) - d(\mathbb{P}_r, \mathbb{P}_G) \right| \leq \varepsilon, \quad (39)$$

where $\mathbb{P}_r^{(n)}$ is the empirical distribution of $\mathbb{P}_r$ with $n$ samples, i.e., the distribution of the training data, and $\mathbb{P}_G^{(n)}$ is the empirical distribution of $\mathbb{P}_G$ with $n$ samples drawn after the generator $G$ is trained. Condition (39) implies that GANs with good generalization property should have consistent performances with the empirical distributions (i.e., $\mathbb{P}_r^{(n)}$ and $\mathbb{P}_G^{(n)}$) and with the true distributions (i.e., $\mathbb{P}_r$ and $\mathbb{P}_G$).

Here, we provide two choices for the divergences. First, the quantile divergence between two distributions $P$ and $Q$ is defined as [37]

$$q(P, Q) := \int_0^1 \left[ \int_{F_P^{-1}(\tau)}^{F_Q^{-1}(\tau)} (F_P(x) - \tau) \, dx \right] \, d\tau,$$

where $F_P$ (resp. $F_Q$) is the CDF of $P$ (resp. $Q$). Motivated by this definition of quantile divergence, we propose the following local-“divergence”, which focuses on the tail distribution of the strategy $PnLs$

$$d_q(\mathbb{P}_r, \mathbb{P}_G) := \frac{1}{K} \sum_{k=1}^{K} \int_0^\alpha \left[ \int_{F_{PnLts}^{-1}^{\leq k}(\tau)}^{F_{PnLts}^{-1}^{\geq k}(\tau)} (F_{PnLts}(x) - \tau) \, dx \right] \, d\tau, \quad (40)$$

where $F_{PnLts}$ (resp. $F_{PnLts}$) is the CDF of $\Pi^k(p)$ with $p \sim \mathbb{P}_r$ (resp. $\Pi^k(q)$ with $q \sim \mathbb{P}_G$).

Recall that the score function used in the loss function (25) can also be constructed as a “divergence” to measure the difference between two distributions in terms of their respective VaR and ES values

$$d_s(\mathbb{P}_r, \mathbb{P}_G) := \frac{1}{K} \sum_{k=1}^{K} \mathbb{E}_{p \sim \mathbb{P}_r} \left[ S_\alpha(\text{VaR}_\alpha(\Pi^k; \mathbb{P}_G), \text{ES}_\alpha(\Pi^k; \mathbb{P}_G), \Pi^k(p)) - S_\alpha(\text{VaR}_\alpha(\Pi^k; \mathbb{P}_r), \text{ES}_\alpha(\Pi^k; \mathbb{P}_r), \Pi^k(p)) \right]. \quad (41)$$
To illustrate the generalization capabilities of Tail-GAN, we compare it with a supervised learning benchmark using the same loss function. Given the optimization problem (23)-(24), one natural idea is to construct a simulator (or a generator) using empirical VaR and ES values in the evaluation. To this end, we consider the following optimization

$$\min_{G \in \mathcal{G}} \frac{1}{Kn} \sum_{k=1}^{K} \sum_{j=1}^{n} S_{\alpha} \left( \left( \text{VaR}_{\alpha} \left( \Pi^{k}_{G} ; P_{G}^{(n)} \right), \text{ES}_{\alpha} \left( \Pi^{k}_{G} ; P_{G}^{(n)} \right) \right), \Pi^{k} \left( p_{j} \right) \right),$$

(42)

where $P_{G}^{(n)}$ is the empirical measure of $n$ samples drawn from $P_{G}$, and $p_{j}$ are samples under the measure $P_{r}$ ($j = 1, 2, \ldots, n$). The optimization problem (42) falls into the category of training simulators with supervised learning [37].

Compared with Tail-GAN, the presented supervised learning framework has several disadvantages, which we illustrate in a set of empirical studies. The first issue is the bottleneck in statistical accuracy. When using $P_{G}^{(n)}$ as the guidance for supervised learning, as indicated in Eqn (42), it is not possible for the $\alpha$-VaR and $\alpha$-ES values of the simulated price scenarios $P_{G}$ to improve on the sampling error of the empirical $\alpha$-VaR and $\alpha$-ES values estimated with the $n$ samples. In particular, ES is very sensitive to tail events, and the empirical estimate of ES may not be stable even with 10,000 samples. The second issue concerns the limited ability in generalization. A generator constructed via supervised learning tends to mimic the exact patterns in the input financial scenarios $P_{r}$, instead of generating new scenarios that are equally realistic compared to the input financial scenarios under the evaluation of the score function. We refer the reader to Section 5.1 for detailed experiments demonstrating these two drawbacks of a simulator constructed via supervised learning.

To compare Tail-GAN with simulators constructed via such supervised learning, we train a Generator-Only Model (GOM) according to (23). We then compare the performance of Tail-GAN with GOM, from the point of view of both statistical accuracy and generalization ability.

**Setup for GOM.** We follow the procedure in Section 3.3 and use $\left( x_{(\left\lfloor \alpha n \right\rfloor)}^{k} , \frac{1}{\left\lfloor \alpha n \right\rfloor} \sum_{i=1}^{\left\lfloor \alpha n \right\rfloor} x_{(i)}^{k} \right)$ to estimate the $\alpha$-VaR and $\alpha$-ES values, where $x_{(n)}^{k} \geq \cdots \geq x_{(2)}^{k} \geq x_{(1)}^{k}$ are the sorted PnLs of $x^{k}$ via the differentiable neural sorting architecture. We train the GOM on synthetic price scenarios, as described in Eqn (38), with both multi-asset portfolio and dynamic strategies. The setting of GOM is the same as that of Tail-GAN (described in Table 2), except that there is no discriminator.

\[1\] We have also trained GOM on real price scenarios, and observed that the performance and conclusion are similar.
**Performance accuracy.** Figure 10 reports the convergence of in-sample errors, and Table 5 summarizes the out-of-sample errors of GOM and Tail-GAN. From Table 5, we observe that the relative error of Tail-GAN is 4.6%, which is a 30% reduction compared to the relative error of GOM of around 7.2%. Compared to (42), the advantage of using neural networks to learn the VaR and ES values, as designed in Tail-GAN, is that it memorizes information in previous iterations during the training procedure, and therefore the statistical bottleneck with $n$ samples can be overcome when the number of iterations increases. Therefore, we conclude that Tail-GAN outperforms GOM in terms of simulation accuracy, demonstrating the importance of the discriminator.

Figure 10: Training performance of GOM and Tail-GAN, as a function of the number of iterations in training. Grey horizontal line: average simulation error $SE(1000)$. Dotted line: average simulation error plus one standard deviation. Each experiment is repeated five times with different random seeds. The performance is visualized with mean (solid lines) and standard deviation (shaded areas).

| Out-Of-Sample Error (%) | Tail-GAN | GOM |
|--------------------------|----------|-----|
|                          | 4.6      | 7.2 |
|                          | (1.6)    | (0.2) |

Table 5: Mean and standard deviation (in parentheses) of relative errors for out-of-sample tests. Each experiment is repeated five times with different random seeds.

Table 6 provides the generalization errors, under both $d_q$ and $d_s$, for Tail-GAN and GOM. We observe that under both criteria, the generalization error of GOM is twice the generalization error of Tail-GAN, implying that Tail-GAN has better generalization power, in addition to higher performance accuracy.
| Error metric | TAIL-GAN | GOM |
|--------------|----------|-----|
| $d_q$        | 0.2139 (0.1785) | 0.5810 (0.4199) |
| $d_s$        | 0.0169 (0.0139) | 0.0317 (0.0262) |

Table 6: Mean (in percentage) and standard deviation (in parentheses) of generalization errors under both divergences (40) and (41). Results are averaged over 10 repeated experiments (synthetic data sets).

5.2 Scalability

In practice, most portfolios held by asset managers are constructed with 20-30 or more financial assets. In order to scale TAIL-GAN with a comparable number of assets, we use PCA-based eigenvectors. The resulting eigenportfolios are uncorrelated and able to explain the most variation in the cross-section of returns with the smallest number of portfolios. This idea enables to train TAIL-GAN with the minimum number of portfolios, hence rendering TAIL-GAN scalable to generate price scenarios with a large number of assets.

In this section, we train TAIL-GAN with eigenportfolios of 20 assets, and compare its performance with TAIL-GAN trained on 50 randomly generated portfolios. TAIL-GAN with the eigenportfolios shows dominating performance, which is also comparable to simulation error (with the same number of samples).

Construction of eigenportfolios. Portfolios constructed from the principal eigenvectors of the return correlation matrix are denoted as eigenportfolios [6]. Eigenportfolios have several advantages when used in the portfolio analysis process. They have reduced dimensions compared to the original basket of portfolios, which often alleviates the issue of transactions costs. In addition, they are uncorrelated and able to explain the most variation of the returns with the smallest number of portfolios, as the eigenvectors of the correlation matrix are mutually orthogonal.

To construct eigenportfolios, the first step is to extract the principal components of the empirical correlation matrix $\hat{\rho}$ of returns based on the eigendecomposition [40]

$$\hat{\rho} = Q\Lambda Q^{-1},$$

and rank the eigenvalues in decreasing order. In the above, $Q$ is the orthogonal matrix with the $i$-th column being the eigenvector $q_i \in \mathbb{R}^M$ of $\hat{\rho}$, and $\Lambda$ is the diagonal matrix whose diagonal elements are the corresponding eigenvalues, such that $\Lambda_{1,1} \geq \Lambda_{2,2} \geq \cdots \geq \Lambda_{M,M} \geq 0$.

The second step is to construct eigenportfolios associated with the extracted principal components. Denote $h = \text{diag}(\sigma_1, \ldots, \sigma_M)$, where $\sigma_i$ is the empirical standard deviation of asset $i$. For the $i$-th eigenvector
we consider its corresponding eigenportfolio

\[
\frac{(h^{-1}q_i)^T p}{\|h^{-1}q_i\|_1},
\]

(44)

where \( p \in \mathbb{R}^{M \times T} \) is the price scenario, and \( \|h^{-1}q_i\|_1 \) is used to normalize the portfolio weights so that the absolute weights sum to unity.

**Data.** To showcase the scalability of Tail-GAN, we simulate the price scenarios of 20 financial assets for a given correlation matrix \( \rho \), with different temporal patterns and tail behaviors in return distributions. Among these 20 financial assets, five of them follow Gaussian distributions, another five follow AR(1) models, another five of them follow GARCH(1, 1) with light-tailed noise, and the rest follow GARCH(1, 1) with heavy-tailed noise. Other settings are the same as in Section 4.3.

**Results.** Figure 11 shows the percentage of explained variance of the principal components, i.e. \( \sum_{i=1}^{M} \frac{\Lambda_{i,i}}{\sum_{m=1}^{M} \Lambda_{m,m}} \).

We observe that the first principal component accounts for more than 23% of the total variation across the 20 asset returns.

To identify and demonstrate the advantages of the eigenportfolios, we compare the following two Tail-GAN architectures

(1) **TAIL-GAN(Rand):** GAN trained with 50 multi-asset portfolios and dynamic strategies,

(2) **TAIL-GAN(Eig):** GAN trained with 20 multi-asset eigenportfolios and dynamic strategies.
The weights of static portfolios in Tail-GAN(Rand) are randomly generated such that the absolute values of the weights sum up to one. The out-of-sample test consists of $K = 90$ strategies, including 50 convex combinations of eigenportfolios (with weights randomly generated), 20 mean-reversion strategies, and 20 trend-following strategies.

**Performance accuracy.** Figure 12 reports the convergence of in-sample errors, and Table 7 summarizes the out-of-sample errors of Tail-GAN(Rand) and Tail-GAN(Eig), showing that Tail-GAN(Eig) achieves better performance than Tail-GAN(Rand) with fewer number of portfolios.

![Figure 12: Training performance on 50 random portfolios vs 20 eigenportfolios as described in Section 5.2: mean of relative error RE(1000) and standard deviation (shaded areas). Grey horizontal line: average simulation error. Dotted line: average simulation error plus one standard deviation. Each experiment is repeated five times with different random seeds. The performance is reported with mean (solid lines) and standard deviation (shaded areas).](image)

| Out-Of-Sample Error (%) | HSM | Tail-GAN(Rand) | Tail-GAN(Eig) |
|-------------------------|-----|---------------|---------------|
|                         | 3.5 | 10.4          | 6.9           |
|                         | (2.3)| (1.8)         | (1.5)         |

Table 7: Mean and standard deviation (in parentheses) for relative errors in out-of-sample tests. Each experiment is repeated five times with different random seeds.

6 Application to simulation of intraday market scenarios

**Setup for intraday price scenarios.** We use the Nasdaq ITCH data from LOBSTER\(^2\) during the intraday time interval 10:00AM-3:30PM, for the period 2019-11-01 until 2019-12-06. The reason for excluding

\(^2\)https://lobsterdata.com/
the first and last 30 minutes of the trading day stems from the increased volatility and volume inherent in the market following the open session, and preceding the closing session. The Tail-GAN simulator is trained on the following five stocks \{AAPL, AMZN, GOOG, JPM, QQQ\}.

The mid-prices (average of the best bid and ask) of these assets are sampled at a $\Delta = 9$-second frequency, with $T = 100$ for each price series representing a financial scenario during a 15-minute interval. We sample the 15-minute paths every one minute, leading to an overlap of 14 minutes between two adjacent paths\(^3\). The architecture and configurations are the same as those reported in Table 2, except that the training period here is 2019-11-01 to 2019-11-30, and the testing period is the first week of 2019-12. Thus, the size of the training data is $N = 6300$. Table 8 reports the 5%-VaR and 5%-ES values of several strategies calculated with the market data of AAPL, AMZN, GOOG, JPM, and QQQ.

|                | Static buy-and-hold | Mean-reversion | Trend-following |
|----------------|---------------------|----------------|-----------------|
|                | VaR     | ES      | VaR     | ES      | VaR     | ES      |
| AAPL           | -0.351  | -0.548  | -0.295  | -0.479  | -0.316  | -0.485  |
| AMZN           | -0.460  | -0.720  | -0.398  | -0.639  | -0.399  | -0.628  |
| GOOG           | -0.316  | -0.481  | -0.272  | -0.426  | -0.273  | -0.419  |
| JPM            | -0.331  | -0.480  | -0.275  | -0.419  | -0.290  | -0.427  |
| QQQ            | -0.254  | -0.384  | -0.202  | -0.321  | -0.210  | -0.328  |

Table 8: Empirical VaR and ES values for trading strategies evaluated on the training data.

**Performance accuracy.** Figure 13 reports the convergence of in-sample errors, and Table 9 summarizes the out-of-sample errors of Tail-GAN-Raw, Tail-GAN-Static, and Tail-GAN.

\(^3\)Tail-GAN are also trained on market data with no time overlap and the conclusions are similar.
Figure 13: Training performance: relative error RE(1000) with 1000 samples. Grey horizontal line: average simulation error SE(1000). Dotted line: average simulation error plus one standard deviation. Each experiment is repeated 5 times with different random seeds. The performance is visualized with mean (solid lines) and standard deviation (shaded areas).

| Out-Of-Sample Error (%) | “Oracle” | HSM  | Tail-GAN-Raw | Tail-GAN-Static | Tail-GAN |
|-------------------------|----------|------|---------------|-----------------|---------|
|                         | 2.4      | 10.4 | 112.8         | 119.5           | 10.1    |
|                         | (1.6)    | (3.6)| (7.8)         | (8.0)           | (0.6)   |

Table 9: Mean and standard deviation (in parentheses) for relative errors in out-of-sample tests. “Oracle” represents the sampling error of the testing data; Each experiment is repeated five times with different random seeds.

We draw the following conclusions from the results of Figure 13 and Table 9.

- For the evaluation criterion RE(1000) based on in-sample data (see Figure 13), all three GAN simulators, Tail-GAN-Raw, Tail-GAN-Static and Tail-GAN, converge within 20,000 epochs and reach in-sample errors smaller than 5%.

- For the evaluation criterion RE(1000), with both static portfolios and dynamic strategies based on out-of-sample data (Table 9), only Tail-GAN converges to an error of 10.1%, whereas the other two Tail-GAN-based simulators fail to capture the temporal information in the input price scenarios. Only the HSM method comes close with an error of 10.4%. As expected, all methods attain higher errors than the sampling error of the testing data (denoted by “oracle” in Table 9).
Figure 14: Tail behavior via the empirical rank-frequency distribution of the strategy PnL, for each of the five stocks considered, across three different types of strategies, indexed the columns.

To study the tail behavior of the intraday scenarios, we implement the same rank-frequency analysis as in Section 4. We observe from Figure 14 that

- All three Tail-GAN simulators are able to capture the tail properties of static single-asset portfolio for quantile levels above 1%.
- For the PnL distribution of the dynamic strategies, only Tail-GAN is able to generate scenarios with comparable (tail) PnL distribution. That is, only scenarios sampled from Tail-GAN can correctly
describe the risks of the trend-following and the mean-reversion strategies.

- Tail-GAN-Raw and Tail-GAN-Static underestimate the risk of loss from the mean-reversion strategy at the $\alpha = 5\%$ quantile level, and overestimate the risk of loss from the trend-following strategy at the $\alpha = 5\%$ quantile level.

Note that some of the blue curves corresponding to the market data (almost) coincide with the red curves corresponding to Tail-GAN, indicating promising performance of Tail-GAN to capture the tail risk of various trading strategies.

**Learning temporal and cross-correlation patterns.** Figures 15 and 16 present the in-sample correlation and auto-correlation patterns of the market data (Figures 15(a) and 16(a)), and simulated data from Tail-GAN-Raw (Figures 15(b) and 16(b)), Tail-GAN-Static (Figures 15(c) and 16(c)) and Tail-GAN (Figures 15(d) and 16(d)).

As shown in Figures 15 and 16 Tail-GAN trained on dynamic strategies learns the information on cross-asset correlations much more accurately than Tail-GAN-Raw, trained on raw returns.

**Scalability.** To test the scalability property of Tail-GAN on realistic scenarios, we conduct a similar experiment as in Section 5.2. The stocks considered here include the top 20 stocks in the S&P500 index. The training period is between 2019-11-01 and 2019-11-30.

Figure 17 reports the convergence of in-sample errors, and Table 10 summarizes the out-of-sample errors of Tail-GAN(Rand) and Tail-GAN(Eig). Table 10 shows that Tail-GAN(Eig) achieves better performance than Tail-GAN(Rand) with fewer training portfolios.

---

**Figure 15:** Cross-asset correlations of the price increments in the market data (a) and from different trained GAN models Tail-GAN-Raw (b), Tail-GAN-Static (c), and Tail-GAN (d). Numbers on the top: mean and standard deviation (in parentheses) of the sum of the absolute difference between the correlation coefficients computed with all training samples and 1,000 generated samples.
Figure 16: Auto-correlations of the price increments from different trained GAN models: (1) Tail-GAN-Raw, (2) Tail-GAN-Static, and (3) Tail-GAN. Numbers on the top: mean and standard deviation (in parentheses) of the sum of the absolute element-wise difference between auto-correlation coefficients computed with all training samples and 1,000 generated samples.

Figure 17: Training performance on 50 random portfolios vs 20 eigenportfolios, as in Section 5.2: mean of relative error $\text{RE}(1000)$ and standard deviation (shaded areas). Grey horizontal line: average simulation error. Dotted line: average simulation error plus one standard deviation. Each experiment is repeated 5 times with different random seeds.

| Out-Of-Sample Error (%) | "Oracle" | HSM | Tail-GAN(Rand) | Tail-GAN(Eig) |
|--------------------------|----------|-----|----------------|---------------|
|                          | 2.2      | 25.9| 31.0           | 25.6          |
|                          | (1.7)    | (5.1)| (1.0)          | (1.0)         |

Table 10: Mean and standard deviation (in parentheses) for relative errors in out-of-sample tests. “Oracle” represents the sampling error of the testing data. Each experiment is repeated five times with different random seeds.
References

[1] Carlo Acerbi and Balazs Szekely. Back-testing expected shortfall. *Risk*, 27(11):76–81, 2014.

[2] Carol Alexander and Anca Dimitriu. Indexing and statistical arbitrage. *The Journal of Portfolio Management*, 31(2):50–63, 2005.

[3] Martin Arjovsky and Léon Bottou. Towards principled methods for training generative adversarial networks. *arXiv preprint arXiv:1701.04862*, 2017.

[4] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR, 2017.

[5] Sanjeev Arora, Rong Ge, Yingyu Liang, Tengyu Ma, and Yi Zhang. Generalization and equilibrium in generative adversarial nets (gans). In *International Conference on Machine Learning*, pages 224–232. PMLR, 2017.

[6] Marco Avellaneda and Jeong-Hyun Lee. Statistical arbitrage in the US equities market. *Quantitative Finance*, 10(7):761–782, 2010.

[7] Siddharth Bhatia, Arjit Jain, and Bryan Hooi. ExGAN: Adversarial Generation of Extreme Samples. *arXiv preprint arXiv:2009.08454*, 2020.

[8] Hans Buehler, Blanka Horvath, Terry Lyons, Imanol Perez Arribas, and Ben Wood. A data-driven market simulator for small data environments. *Available at SSRN 3632431*, 2020.

[9] Hans Buehler, Blanka Horvath, Terry Lyons, Imanol Perez Arribas, and Ben Wood. Generating financial markets with signatures. *Available at SSRN 3657366*, 2020.

[10] Sean D Campbell. A review of backtesting and backtesting procedures. *The Journal of Risk*, 9(2):1, 2006.

[11] Haoyang Cao, Xin Guo, and Guan Wang. Meta-learning with gans for anomaly detection, with deployment in high-speed rail inspection system. *arXiv preprint arXiv:2202.05795*, 2022.

[12] Rama Cont. Empirical properties of asset returns: stylized facts and statistical issues. *Quantitative finance*, 1(2):223, 2001.

[13] Francis X Diebold and Robert S Mariano. Comparing predictive accuracy. *Journal of Business & economic statistics*, 20(1):134–144, 2002.

[14] Chris Donahue, Julian McAuley, and Miller Puckette. Adversarial audio synthesis. *arXiv preprint arXiv:1802.04208*, 2018.

[15] Zaichao Du and Juan Carlos Escanciano. Backtesting expected shortfall: accounting for tail risk. *Management Science*, 63(4):940–958, 2017.

[16] William Fedus, Ian Goodfellow, and Andrew M Dai. MaskGAN: Better text generation via filling in the_. *arXiv preprint arXiv:1801.07736*, 2018.

[17] Tobias Fissler, Johanna F Ziegel, et al. Higher order elicitation and osband’s principle. *The Annals of Statistics*, 44(4):1680–1707, 2016.
[18] Tobias Fissler, Johanna F Ziegel, and Tilmann Gneiting. Expected shortfall is jointly elicitable with value at risk-implications for backtesting. *arXiv preprint arXiv:1507.00244*, 2015.

[19] Rao Fu, Jie Chen, Shutian Zeng, Yipeng Zhuang, and Agus Sudjianto. Time series simulation by conditional generative adversarial net. *arXiv preprint arXiv:1904.11419*, 2019.

[20] Paul Glasserman. *Monte Carlo methods in financial engineering*. Springer, 2003.

[21] Tilmann Gneiting. Making and evaluating point forecasts. *Journal of the American Statistical Association*, 106(494):746–762, 2011.

[22] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27:2672–2680, 2014.

[23] Aditya Grover, Eric Wang, Aaron Zweig, and Stefano Ermon. Stochastic optimization of sorting networks via continuous relaxations. In *International Conference on Learning Representations*, 2019.

[24] Rui Huang, Shu Zhang, Tianyu Li, and Ran He. Beyond face rotation: Global and local perception gan for photorealistic and identity preserving frontal view synthesis. In *Proceedings of the IEEE international conference on computer vision*, pages 2439–2448, 2017.

[25] Adriano Koshiyama, Nick Firoozye, and Philip Treleaven. Generative adversarial networks for financial trading strategies fine-tuning and combination. *Quantitative Finance*, pages 1–17, 2020.

[26] Paul Kupiec. Techniques for verifying the accuracy of risk measurement models. *The J. of Derivatives*, 3(2), 1995.

[27] Nicolas S Lambert, David M Pennock, and Yoav Shoham. Eliciting properties of probability distributions. In *Proceedings of the 9th ACM Conference on Electronic Commerce*, pages 129–138, 2008.

[28] Erich L Lehmann and Joseph P Romano. *Testing statistical hypotheses*. Springer Science & Business Media, 2006.

[29] Junyi Li, Xintong Wang, Yaoyang Lin, Arunesh Sinha, and Michael Wellman. Generating realistic stock market order streams. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(01):727–734, 2020.

[30] Gautier Marti. CorrGAN: Sampling realistic financial correlation matrices using generative adversarial networks. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8459–8463. IEEE, 2020.

[31] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[32] Hao Ni, Lukasz Szpruch, Magnus Wiese, Shujian Liao, and Baoren Xiao. Conditional Sig-Wasserstein GANs for time series generation. *arXiv preprint arXiv:2006.05421*, 2020.

[33] Wlodzimierz Ogryczak and Arie Tamir. Minimizing the sum of the k largest functions in linear time. *Information Processing Letters*, 85(3):117–122, 2003.
[34] Basel Committee on Banking Supervision. Minimum capital requirements for market risk. https://www.bis.org/bcbs/publ/d457.pdf, 2019.

[35] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. arXiv preprint arXiv:1609.03499, 2016.

[36] Kent Osband. Providing incentives for better cost forecasting. PhD thesis, University of California, Berkeley, 1985.

[37] Georg Ostrovski, Will Dabney, and Rémi Munos. Autoregressive quantile networks for generative modeling. In International Conference on Machine Learning, pages 3936–3945. PMLR, 2018.

[38] Alec Radford,Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434, 2015.

[39] 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.

[40] Lloyd N Trefethen and David Bau III. Numerical linear algebra, volume 50. Siam, 1997.

[41] Stefan Weber. Distribution-invariant risk measures, information, and dynamic consistency. Mathematical Finance: An International Journal of Mathematics, Statistics and Financial Economics, 16(2):419–441, 2006.

[42] Magnus Wiese, Robert Knobloch, Ralf Korn, and Peter Kretschmer. Quant GANs: Deep generation of financial time series. Quantitative Finance, pages 1–22, 2020.

[43] Jinsung Yoon, Daniel Jarrett, and Mihaela van der Schaar. Time-series generative adversarial networks. Advances in Neural Information Processing Systems, 32:5508–5518, 2019.

[44] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning (still) requires rethinking generalization. Communications of the ACM, 64(3):107–115, 2021.

[45] Yizhe Zhang, Zhe Gan, Kai Fan, Zhi Chen, Ricardo Henao, Dinghan Shen, and Lawrence Carin. Adversarial feature matching for text generation. arXiv preprint arXiv:1706.03850, 2017.

[46] Peng Zhou, Lingxi Xie, Bingbing Ni, Cong Geng, and Qi Tian. Omni-gan: On the secrets of cigans and beyond. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 14061–14071, 2021.