Data-driven portfolio drawdown optimization with generative modeling

Emiel Lemahieu

Ghent University, InvestSuite

Abstract

This paper investigates the use of generative machine learning for portfolio construction, by learning drawdown distributions under $\mathbb{P}$ and then minimizing *expected* portfolio drawdown risk. We introduce a method for matching the drawdown moments in the path space, based on recently introduced signature-based maximum mean discrepancy measures. This allows us to translate *geometric* and *economic priors* on the underlying drawdown distribution into new sample paths, without requiring an a priori specification of the underlying drawdown generating process. This process stays implicit in the parameters of our machine learning model, more specifically a conditional variational autoencoder model. On a numerical example for US ETFs, we illustrate the use of data-driven scenario-based portfolio construction for better apprehending optimal portfolios and their sensitivities to market conditions.

Keywords: Portfolio Construction, Machine Learning, Geometric Learning, Signatures, Maximum Mean Discrepancy, Economic Priors

1. Introduction

The application of data-driven or machine learning methods to financial problems such as asset pricing [1][2][3], optimal trade execution [4][5][6] and risk and portfolio management [7][8][9][10][11] has seen a proliferation in the literature over the last years. One way to broadly separate these techniques into two distinct classes of models is to distinguish between so-called *generative* and *discriminative* modeling. In this introduction, we will first explain this difference and make the link between generative models in machine learning and traditional generative models in finance. Next, we will delve into the difference between generating return versus drawdown sequences. To end this introduction, we introduce the portfolio drawdown optimization problem. The remainder of this paper is then structured as follows. In Section 2 we briefly list the main contributions of this work. In Section 3, we give a brief overview of the existing literature on market generators. Section 4 covers the methodological building blocks of our portfolio optimizer, namely the signature kernel-based MMD measure, the CVAE and the explainable machine learning (XML) component of our framework. Section 4 discusses the numerical results on a universe of US ETFs. Section 5 concludes.

1.1. Generative modeling in machine learning

Machine learning (ML) models can broadly be separated into two classes of models:

- **Discriminative modeling**: These models revolve around the conditional distribution $\mathbb{P}$(Y|X), and learn a set of parameters Θ from the data to predict labels Y given a distribution of features X. Formally, given some $Y : \mathbb{R}^M$, $X : \mathbb{R}^N$, with N typically large, we learn a set of parameters Θ using a flexible mapping f (e.g. a neural network): $f_\Theta(X) : \mathbb{R}^N \to \mathbb{R}^M$ such that some loss measure $\mathcal{L}(Y, f_\Theta(X))$ is minimized. Examples include simple regularized regressions (LASSO, Ridge, Elastic nets [12]), support vector machines and neural network regressors [13].

- **Generative modeling**: These models revolve around the unconditional distribution $\mathbb{P}$(X), and learn a set of parameters Θ that capture the structure or symmetries in the high-dimensional distribution $\mathbb{P}$(X). The aim is generally to compress the data in much fewer dimensions, while preserving the most important features of the original data. Formally, given some $X : \mathbb{R}^N$, with N typically large, we learn a set of parameters Θ, again using a flexible mapping f, but now on some space $Z : \mathbb{R}^K$, with $K << N$, called a *representation*, $f_\Theta(X) : \mathbb{R}^N \to \mathbb{R}^K : X \to Z$. The goal of such a representation is that the mapping $f_\Theta^{-1}(Z) : \mathbb{R}^K \to \mathbb{R}^N : Z \to X'$ can be used for sampling new samples X' such that X and X' are not distinguishable statistically, i.e. according to some loss metric $\mathcal{L}(X, X')$. Z, the *latent* space, is sampled from some simple source distribution, e.g. $Z \sim N(0, 1)$ or $Z \sim U(0, 1)$. $f_\Theta^{-1}(Z)$ can thus be seen as a mapping that translates a sample of noise into a sample that is indistinguishable from the original data distribution, and thus generates an unlimited amount of new samples $X' \sim \mathbb{P}_{\Theta}(X)$. Popular ML pipelines here include variational autoencoders [14] [15], generative adversarial networks [16], restricted Boltzmann machines [17] [18], and normalizing flows [19].

While many of the above-cited papers on financial machine learning have focused on better *prediction* with discriminative (deep) learning models, this paper focuses on better *simulation* with the latter class of techniques. Here it comes down to learning an optimal mapping between some high-dimensional distribution of paths and some low-dimensional representation that allows us to reconstruct paths that are indistinguishable from the original in terms of some important features. One of the main arguments in this paper is that these features are determined by the application at hand (*geometric* prior).

1.2. Generative modeling versus traditional Monte Carlo techniques

Finding an optimal mapping between a source distribution and the original data $\mathbb{P}(X)$ is not new to finance. This has been a key area of research in Monte Carlo simulation [20] and the development of bottom-up stochastic processes. This has been instrumental in calibrating risk measures and optimizing portfolios under the physical measure $\mathbb{P}$, but crucial in constructing derivative pricing tools under the risk-neutral measure $\mathbb{Q}$. The core difference with the machine learning approach, is that in a traditional Monte Carlo the map $f_{\Theta}^{-1}(X)$ has to be specified as some closed-form (system of) equation(s) called the data generating process (DGP). Arguably the most well-known process is the Black-Scholes equation [21] that describes the diffusion paths of asset prices as geometric Brownian motions. In this example, $Z \sim N(0, 1)$ and Θ is a tuple of the drift and volatilities (μ, σ) such that the corresponding market generator becomes:

$$f_{\Theta}^{-1}(Z) : X_t = \mu + \sigma \epsilon_t \tag{1}$$

where X_t is the logreturn at t, $\Theta = (\mu, \sigma)$, and ϵ_t is an instance of Z at t. Remark that $\mu = r$, the risk-free rate under $\mathbb{Q}$. Another difference is that such an a priori specified $f_{\Theta}^{-1}(Z)$ does not require the estimation of $f_{\Theta}(X)$ and the evaluation of $\mathcal{L}(X, \hat{X})$, but rather relies on estimating Θ directly using some form of loglikelihood maximization on historical data (called *calibration*). There are thousands of papers that focused on improving the specification of the data generating process[1], incorporating autocorrelated means or mean-reversion properties, local and autoregressive stochastic volatilities, long-memory or rough volatility, fat-tailed and infinite-variance processes, copula correlation structures, and so and so forth. This has led to an increasingly rich reproduction of statistical features or *stylized* facts (see Section 1.3 below) of financial paths, at the cost of increasingly complex formulations of (1).

1.3. Drawdown versus return paths

A core concept in this paper is a *path*. It may appear as a trivial concept, but it is highly non-trivial to define it in this context. In general, a path γ in $\mathbb{R}^D$ is a continuous map from some interval $[a, b]$ to $\mathbb{R}^D$, written as $\gamma : [a, b] \to \mathbb{R}^D$. We use subscript $\gamma_t = \gamma(t)$ to denote the time as parameter $t \in [a, b]$, and usually for convenience we take $a = 0, b = T, t \in [0, T]$. In our examples we will assume that paths are piecewise linear, smooth and differentiable, i.e. the path has derivates of all orders over [0,T][2].

One path in D dimensions can be written as

$$\gamma : [0, T] \to \mathbb{R}^D, \gamma = \{\gamma^1, \gamma^2, ..., \gamma^D\} \tag{2}$$

Traditionally, financial paths are tantamount to return sequences. Say $S_i(t)$ is the price of a financial asset i, such as a stock, bond, an index or exchange rate. Then as per above, we denote $X_i(t, \Delta t) = ln(S_i(t + \Delta t)) - ln(S_i(t))$ as the logreturn at t over the previous period Δt, say daily. $corr(X_i(t + \tau, \Delta t), X_i(t, \Delta t))$ is the linear autocorrelation function of X_i for lag τ. Say we have a D-dimensional universe of financial instruments, then for every instrument i we can define a return path r as:

$$r_i : [0, T] \to \mathbb{R}^2, r_i = \{t, (X_i(0, \Delta t), X_i(1, \Delta t), X_i(T, \Delta t))\} \tag{3}$$

Where the path at the first dimension is the time evolution, and the path at $D = 2$ is the evolution of returns over t. This is also called the *time-augmented* path. Now let us call the (D+1)-dimensional path R_j

$$R_j : [0, T] \to \mathbb{R}^{D+1}, R_j = \{t, r_1, r_2, ..., r_D\} \tag{4}$$

[1]Some famous ones such as Heston [22], SABR [23] and *rough* vol [24].

[2]However, the same properties hold for general (rough) paths of bounded variation, see [25].

a *return space*. This is a focal object in quantitative finance. The drift of this vector space in each of its components r_i, μ_i, is the average return and often modelled using factor decompositions of the space R. The similarity between vectors r_i is often assessed with correlation ρ_i, while its dispersion or risk is mostly analyzed using the standard deviation across the coordinates σ_i.

From a portfolio construction perspective, the traditional aim (in mean-variance frameworks [26]) is to find r^* as a dotproduct of a D-dimensional weight vector w^* and R, such that w^*R as a single *portfolio return path* r^* has maximal drift μ^* over standard deviation σ^* (or a maximal *Sharpe* ratio).

However, setting T to the historical sample (e.g. $N_{obs} = 2500$ by looking back 10 years of daily data with approx. 250 trading days per year) only yields one estimate of μ_i, ρ_i and σ_i, which gives us no statistical confidence of that estimate or probabilities, nor is every observation in the total sample as indicative for the current market regime. Therefore, T should be chosen smaller than the number of observations N_{obs}, e.g. monthly sample paths $T = 20$, such that the historical sample gives rise to $N_{sim} = \lfloor N_{obs}/T \rfloor$ non-overlapping or $N_{sim} = N_{observations} - T$ overlapping return spaces called *return sequences* R:

$$R = (R_1, R_2, ..., R_{N_{sim}}) \tag{5}$$

R can thus be seen as one return tensor with N_{sim} return spaces. Optionally, one can attach weights w_j to each R_j to denote its importance in the estimation of the statistics on R. Some common examples: (1) exponential smoothing (EWMA) would attach exponentially decreasing w_j to smaller j, (2) conditional sampling would attach $w_j = 0$ to sequences not satisfying the historical conditions, and $w_j = 1$ if they do, and (3) volatility-filtered sampling would set $w_j = \frac{\sigma}{\sigma_j}$.

Based on this set of return sequences R, stats for μ_i and σ_i (or ρ_i, possibly r^*) can be determined *probabilistically*, and *expectations* of stats can be developed (e.g. based on conditional distributions). The estimation of these parameters is exactly what we discussed in 1.2 in terms of calibrating the parametric specification of the DGP. R could also be used outright[3], or sequences j could be resampled with replacement[4], but this results in limited unique samples, can give rise to (excess) duplicates and thus multicollinearity in the optimization problem. With a generative model, in principle we can

[3]i.e. *historical simulation*

[4]i.e. a non-parametric (w_j-weighted) block bootstrap with block size T

create an unlimited amount of samples from noise.

When modeling return sequences, there is widespread knowledge and to a certain extent consensus about the statistical features we want to reproduce in R, known as the *stylized facts* of financial returns. This was surveyed by Cont [27] and are most importantly: (1) the existence of *fat tails* in the return distribution, (2) the absence of linear autocorrelation (cf. above), (3) volatility clusters (large *absolute* returns are highly autocorrelated), and (4) leverage effects (*absolute* returns and returns are negatively correlated). Much of the work regarding stochastic DGPs discussed in 1.2 come down to addressing these stylized facts.

Return paths are, partly due to the lack of linear autocorrelation, often not considered as paths as formally described above, but viewed from their return distribution right away. The return distribution, or in practice often called the profit-and-loss (P&L), is essentially a static measure. Consider it in our notation the density function of R when we flatten the tensor, i.e. $\mathbb{P}(X)$ for all X in r for all scenarios R_j in R. Once Δt is decided (e.g. daily), we can shuffle the returns and maintain the same P&L. Estimates for μ_i, σ_i and $r*$ will remain the same, as well as popular risk conditionals on the P&L distribution such as value-at-risk (VaR) and expected shortfall (ES). Valuable information about the sequential structure, i.e. the path structure, is lost.

Investors are also interested in the path characteristics of their risk, e.g. the autocorrelation of their risk. They could correct for the *scaling* between their VaR and Δt using the monofractal scaling of volatility σ_i and essentially generate a multi-scale P&L. However, a more intuitive approach is to, rather than returns, model *drawdown* paths.

Again $S_i(t)$ denotes the price of our asset i. Then let us consider, in accordance to [28], the absolute drawdown conditional $x_i(t, \Delta t) = \max_{t_k<t}(S_i(t_k)) - S_i(t)$. This measure is a dynamic generalization of a deviation measure on the path space [28]. Every instrument in our D-dimensional universe has a drawdown series ξ_i defined as:

$$\xi_i : [0, T] \to \mathbb{R}^2, \xi_i = \{t, (x_i(0, \Delta t), x_i(1, \Delta t), x_i(T, \Delta t))\} \quad (6)$$

And the *drawdown space* can analogously be defined as:

$$\Xi_j : [0, T] \to \mathbb{R}^{D+1}, \Xi_j = \{t, \xi_1, \xi_2, ..., \xi_D\} \quad (7)$$

Giving rise to *drawdown sequences* on a $T < N_{obs}$ path space as per above:

$$\Xi = (\Xi_1, \Xi_2, ..., \Xi_{N_{sim}}) \quad (8)$$

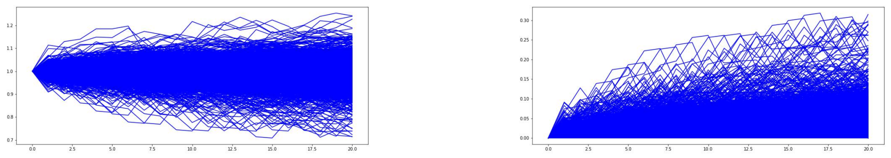

Figure 1: Return paths (left) versus drawdown paths (right)

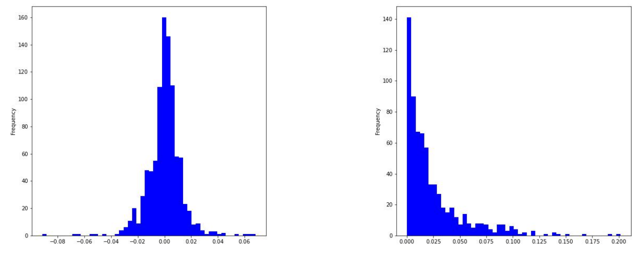

Figure 2: $\mathbb{P}(X)$ of returns (left) versus $\mathbb{P}(x)$ of drawdowns (right)

Figure 1 and 2 illustrates the difference between return and drawdown paths, as well as $\mathbb{P}(X)$ and ı(x).

As per above for returns, the treatment of Ξ in terms of w_j and the choice of Θ allows us to move from a single historical drawdown to a distribution of (conditional) drawdowns and *expected* drawdown (optimization). However, there are some major differences and related challenges:

- The drawdown path ξ itself has important path structure, i.e. the maximum and average drawdown are determined by consecutive and highly autocorrelated observations of $x_i(t)$. It is therefore important to match the distribution of the Ξ, $\mathbb{P}(\xi)$ for all scenarios Ξ_j in Ξ and not just the flattened $\mathbb{P}(x)$ for all x in ξ for scenarios Ξ_j in Ξ. In other words, effective drawdown simulation hinges on the distribution in the path space, not the flat x distribution, while for returns this is often synonymous.

- Stochastic processes have not been developed for ξ-processes yet. There are no off-the-shelf DGPs that seem to reflect the statistical reality of ξ-processes. A fortiori, no stylized facts for ξ-processes have been proposed or agreed on.

This has important implications for portfolio drawdown optimization in general, and the choice of Ξ and the generator Θ in particular: (1) since we have to compare two distributions in the path space (i.e. *sequential* random variables), we cannot use traditional distributional methods such as loglikelihood or discrepancy measures from information theory (such as KL or JS divergences), (2) since no DGPs have been developed for ξ-processes, DGP-free modeling of them leapfrogs that gap (given that the output paths are sufficiently realistic).

1.4. Portfolio drawdown optimization

Now that we have defined drawdown sequences and the particular challenges posed in simulating paths that are loyal to the historical drawdown structure, we can define our portfolio optimization problem.

Drawdown optimization is of particular interest to fund managers, as well as pertinent to the recent boom in robo-advisors. It does not only better reflect risk as perceived by (retail) investors: both the size and frequency of losses, as well as the time required to recoup them is taken into account. It's also conducive for maximizing customer retention, or to put it differently, avoiding churn. Retail investors are particularly sensitive to losses and the time to recover from them, and will unlikely tolerate large drawdowns or even small drawdowns that persist for a long time. By a mouse click, or removing an app, they can close down an account and go elsewhere. Therefore, drawdown optimization (versus volatility or value-at-risk) also makes sense from a retention point of view in this day of robo-advisory.

Recall that portfolio optimization revolves around finding an optimal vector of weights w^* such that the portfolio timeseries w^*R has some desirable properties, such as minimum volatility (standard deviation), maximum return (drift) or the optimal ratio between them. Remark now, that the path space on which these weights/dotproducts apply does not necessarily have to be returns. This could also be drawdown paths or the original price paths.

Given the above, the simplest drawdown optimization algorithm would be to minimize:

$$\begin{array}{ll} \min_w & \mathbb{E}_j(\xi(w)) \\ \text{s.t.} & \xi_j = w\Xi_j \\ & w\mathbf{I}^D = 1 \end{array}$$

(9)

This optimization problem minimizes the portfolio holding-weighted average drawdown of the instrument drawdown in the universe. However, this is not the same as minimizing portfolio drawdown and leaves room for the optimizer to game this difference such that we might end up with corresponding portfolio drawdowns that are not optimal. Rather, we propose a refactoring to portfolio values that reflects the portfolio drawdown as objective:

$$
\begin{array}{ll}
\min_w & \mathbb{E}_j(\xi(w)) \\
\text{s.t.} & \xi_{j,t} = m_{j,t} - w\Pi_{j,t} \\
 & m_{j,t} \geq m_{j,t-1} \\
 & w\mathbf{I}^D = 1
\end{array}
\tag{10}
$$

where Π is a space of asset *price paths* that has a *correspondence* to Ξ_j. We will come back to this crucial point in Section 4. For now it becomes apparent that the path structure of Π is critical because the explicit *portfolio* drawdown optimization necessitates an optimization variable m_t which reflects the local maxima of the assets at t for scenario j. Generally, this is not preserved when modeling R, but it is our crucial *path feature* in Ξ.

2. Contributions

The contributions of this paper are threefold:

- It explores the use of **portfolio optimization** with generative modeling, which was not the focus in earlier studies (cf. Section 3) and, because of the relevant path features for this application, necessitates a new **input data representation** (e.g. Ξ versus R) compared to generators introduced previously in literature.
- It introduces a **loss metric** in the generator's architecture that focuses on learning structure (or non-linear common factors) in the downside/drawdown risk of the universe.
- Finally, it elaborates on the use of **conditions** or conditional distributions, to better match the non-stationary properties of financial timeseries as well as to better apprehend the **sensitivities** of the optimal portfolio to market conditions.

3. Brief overview of market generator literature

Market generators are generative models with the specificity of modeling financial markets, such as spot asset prices S, option prices and (implied) volatilities, or order streams in limit order books. The topic has seen a recent surge in interest as generative machine learning architectures (see Section 4.2.1) have found their way in simulation engines for optimal hedging, optimal order execution, backtesting trading strategies, and many more. Table 1 gives a brief overview and covers a non-exhaustive list of some of these market generators, and their application. They are sorted first on architecture and then on date. Remark that the literature on neural (and even generative) architectures in finance is vast, and this table only focuses on models that satisfy our definition of a market generator, and are thus closely related to this research.

Paper	**Year**	**Architecture**	**Application**
Henry-Labordere [29]	2019	GAN	Option prices
Wiese et al. [30]	2019	GAN	Hedging strategies
Cuchiero et al. [31]	2020	GAN	Volatility models
Ni et al. [32]	2020	GAN	Spot prices
Wiese et al. [33]	2020	GAN	Spot prices
Li et al. [34]	2020	GAN	Order book simulation
Storchan et al. [35]	2020	GAN	Spot prices
Benedetti [36]	2020	GAN	Yield models
Xu et al. [37]	2020	GAN	Spot prices
Pardo and López [38]	2020	GAN	Spot prices
Buehler et al. [39]	2021	GAN	Hedging strategies
Ni et al. [40]	2021	GAN	Spot prices
Pfenninger et al. [41]	2021	GAN	Spot prices
Rosolia and Osterrieder [42]	2021	GAN	Spot prices
Koshiyama et al. [43]	2021	GAN	Spot prices
van Rhijn et al. [44]	2021	GAN	Spot prices
Marti et al. [45]	2021	GAN	Correlation matrices
Coyle et al. [46]	2021	GAN	Spot prices
Wiese et al. [47]	2021	NF	Spot and Option prices
Kondratyev and Schwarz [48]	2019	RBM	Spot prices
Lezmi et al. [49]	2020	RBM / GAN	Spot prices
Wang [50]	2021	RBM / VAE	Spot prices
Buehler et al. [51]	2020	VAE	Spot prices
Fung [52]	2021	VAE	Option prices
Frandsen [53]	2021	VAE	Hedging strategies
Bergeron et al. [54]	2021	VAE	Volatility models
Ning et al. [55]	2021	VAE	Volatility models

Table 1: Overview of the market generator literature

4. Methodologies

This section discusses the main methodologies used to produce the results in Section 5. These include the signature-based MMD measure to match drawdown moments in Section 4.1, the choice of generative modeling architecture and details behind the CVAE in Section 4.2 and the explainable machine learning methods in Section 4.3.

4.1. Signature kernel

The *signature kernel* is a recent development in mathematics, and more specifically stochastic analysis, with fruitful applications in machine learning. It uses a transformation of the path space to the so-called *signature* space, which resembles a graded summary of path[5]-structured data, while preserving important *geometrical* features of the path. They offer a mathematically principled feature representation of paths, and have proven efficient tools in machine learning applications such as recognition of handwritten Chinese characters [56], classification and validation of bipolar and borderline disorders [57], malware detection [58], detection of Alzheimer disease [59], human action recognition [60], and many more. Applications in finance include market simulation [51] and optimal trade execution [61]. Moreover, recent work argues that they are *provably optimal* feature maps, being both *universal* and *unique* [62]. We will introduce these concepts below, as much from the ground up as possible while still focusing on the essence.

We start with a recap of kernels and their advantage in general. Then we delve into the use of kernels for distance measures in machine learning. Next we introduce signatures and the signature kernel, as well as its implications for distances in the path space.

4.1.1. Kernel techniques

In general, kernels k are a class of functions of two random variables that measure the similarity between the two random variables. For instance:

$$k(X,Y) : [a,b] \times [a,b] \to \mathbb{R} \tag{11}$$

is a kernel since it maps two random variables X and Y with support on $[a,b]$ on a metric that is (commonly) small when X and Y are close to each other, and vice versa. Common examples are radial basis functions (RBF) such as the exponential kernel and Gaussian, Euclidean, Polynomial kernels, and so and so forth. Being introduced very general here, they also span a vast range of applications in machine learning (and beyond, see [63]): (1) *feature maps* where kernels are essentially inner products between feature vectors X (which allows for using linear methods in non-linear problems, e.g. support vector machines), (2) *basis functions for approximation spaces* (i.e. changing the basis of data to approximate functions by allowing more

[5]Possibly very high-dimensional paths, called streams.

variation in regions with more data), and many more. Clearly the use case will determine the choice of the appropriate kernel, and there is no panacea kernel that outperforms on all of these use cases.

Although this explanation introduces kernels in the most general way possible, we can already discuss their main advantage by introducing property (13) and MMD in the next section. Positive definite kernels, such as the Gaussian kernel, that satisfy

$$\sum_{i=1}^{n}\sum_{j=1}^{n} c_i c_j k(x_i, x_j) \geq 0 \tag{12}$$

for any x_i in X and any pair $c_i, c_j \in \mathbb{R}$, also called Mercer kernels[6] have the property that there exists a mapping ϕ between X and Y and a space $\mathcal{H}$ equipped with an inner product, such that the kernel value $k(x, y)$ can be rewritten as an inner product in $\mathcal{H}$:

$$k(x, y) = \langle \phi(x), \phi(y) \rangle \tag{13}$$

Since $\mathcal{H}$ should only be equipped with an inner product it is a so-called Hilbert space, and it *reproduces* the kernel by means of that inner product of two mapped features $\phi(.)$. This is known as a reproducing kernel Hilbert space (RKHS) in machine learning.

4.1.2. Max mean discrepancy

The maximum mean discrepancy (MMD) is a popular measure of distance between two distributions in machine learning. Suppose we have two sets of samples X and Y and we want to measure the distance between them. The following MMD computes the mean squared difference of the statistics ϕ between the two sets:

$$MMD = ||\frac{1}{N}\sum_{i=1}^{N}\phi(x_i) - \frac{1}{M}\sum_{j=1}^{M}\phi(y_j)||^2 \tag{14}$$

[6]In reference to the British mathematician James Mercer and the Mercer theorem, a central result for kernel approximation methods and kernel machines such as support vector machines.

This can consequently be rewritten as:

$$MMD = \frac{1}{N^2}\sum_{i=1}^{N}\sum_{i'=1}^{N}\phi(x_i)\phi(x_i') - \frac{2}{NM}\sum_{i=1}^{N}\sum_{j=1}^{M}\phi(x_i)\phi(y_i) + \frac{1}{M^2}\sum_{j=1}^{M}\sum_{j'=1}^{M}\phi(y_i)\phi(y_i') \tag{15}$$

Now for instance taking ϕ equal to be identity $\phi(x) = x$, this gives rise to the squared difference in means between X and Y, and other choices give rise to higher order moments of X and Y.

Remark that in (15) the distance between X and Y are only written in terms of the inner products between the mappings $\phi(.)$ of X and Y, which means that we can propose a (positive definite) kernel such that:

$$MMD = \frac{1}{N^2}\sum_{i=1}^{N}\sum_{i'=1}^{N}k(x_i, x_i') - \frac{2}{NM}\sum_{i=1}^{N}\sum_{j=1}^{M}k(x_i, y_i) + \frac{1}{M^2}\sum_{j=1}^{M}\sum_{j'=1}^{M}k(y_i, y_i') \tag{16}$$

This summarizes the main purpose of kernels in this application, namely that the distance between two samples in terms of a map ϕ can be evaluated without having to actually compute all the mappings $\phi(.)$ of X and Y, which can lead to dramatic improvements computationally. This famous result is often referred to as the *kernel trick*.

The MMD will be zero if and only if $\mathbb{P}(X) = \mathbb{P}(Y)$, and distances increase when the moments diverge more. For instance, using a Gaussian kernel $k(x, y) = \exp(-\frac{1}{2\sigma}|x - y|^2)$ with *bandwith* parameter σ [64] argues that by means of a Taylor expansion it can be shown that minimizing MMD corresponds to minimizing a distance between *all* moments of X and Y. They coined a generative model with this distance metric a *generative moment matching network*.

For now let us conclude that kernel methods allow us to embed two random variables X and Y in a space $\mathcal{H}$ that allow us to evaluate the distance between the two variables, often computationally orders of magnitude less expensive than evaluating a feature map $\phi(.)$ over the values of X and Y and then evaluating the inner products of all ϕ. One key assumption for now is that X and Y were simple random variables in $\mathbb{R}$ and not sequantial variables, i.e. random variables in the *path space* $\mathbb{R}^D$

4.1.3. Signatures and the signature kernel

Let us now move to distances in the path space by means of the signature transform and signature kernel. We already formally introduced a path γ in equation (2). Let us now recall path integrals. For a path $\gamma : [0,T] \to \mathbb{R}$ and a function $f : \mathbb{R} \to \mathbb{R}$, the path integral of γ against f is defined by

$$\int_0^T f(\gamma_t) d\gamma_t = \int_0^T f(\gamma_t) \frac{d\gamma_t}{dt} dt = \int_0^T f(\gamma_t) \dot{\gamma_t} dt \tag{17}$$

in which context f is called a *1-form* [65]. The last integral is the 'usual' Riemann integral. Note that f itself is a real-valued path on $[0,T]$. This is a special case of the Riemann-Stieltjes integral of one path against another [25]. In general, one can integrate any two paths on $[0,T]$, $\kappa : [0,T] \to \mathbb{R}, \gamma : [0,T] \to \mathbb{R}$, against one another:

$$\int_0^T \kappa_t d\gamma_t = \int_0^T \kappa_t \dot{\gamma_t} dt \tag{18}$$

Signature definition. Let us consider a particular path integral defined for any *single* index $i \in \{1, 2, ..., D\}$:

$$S(\gamma)^i_{0,T} = \int_0^T d\gamma^i = \gamma^i_T - \gamma^i_0 \tag{19}$$

which is the increment of the path along the dimension i in $\{1, 2, ..., D\}$. Now for any *pair* of indexes $i, j \in \{1, 2, ..., D\}$, let us define:

$$S(\gamma)^{i,j}_{0,T} = \int_0^T \int_0^{t_j} d\gamma^i d\gamma^j \tag{20}$$

and likewise for *triple* indices in $i, j, k \in \{1, 2, ..., D\}$:

$$S(\gamma)^{i,j,k}_{0,T} = \int_0^T \int_{t_k}^{t_j} \int_0^{t_k} d\gamma^i d\gamma^j d\gamma^k \tag{21}$$

and we can continue for the collection of indices $i_1, i_2, ..., i_k \in \{1, 2, ..., D\}$:

$$S(\gamma)^{i_1,i_2,...,i_j,...,i_k}_{0,T} = \int_0^T ... \int_{t_j}^{t_{j+1}} ... \int_{t_1}^{t_2} \int_0^{t_1} d\gamma^{i_1} d\gamma^{i_2} ... d\gamma^{i_k} \tag{22}$$

which we call the k-fold iterated integral of γ along $\{i_1, i_2, ..., i_k\}$.

Definition 4.1 (Signature). *From [25]: The signature of a path $\gamma : [0, T] \to \mathbb{R}^D$ denoted $S(\gamma)_{0,T}$ is the collection (an infinite series) of all the iterated integrals of γ. Formally, $S(\gamma)_{0,T}$ is the sequence of real numbers*

$$S(\gamma)_{0,T} = (1, S(X)^1_{0,T}, S(X)^2_{0,T}, ..., S(X)^D_{0,T}, S(X)^{1,1}_{0,T}, S(X)^{1,2}_{0,T}, ...) \tag{23}$$

where the zeroth term is 1 by convention and the superscript runs along the set of multi-indices:

$$W = \{(i_1, i_2, ..., i_k) | k \geq 1; i_1, i_2, ..., i_k \in \{1, 2, ..., D\}\} \tag{24}$$

In other words, the signature is the collection of all the iterated integrals consisting of all possible combinations of the indices in D (for any length of combination, hence it is an infinite series). However, it is important to note that these signatures are ordered along this length, which is called the *order* or *level* of the signature.

We often consider the M-th level truncated signature, defined as the finite collection of all terms where the superscript is of max length M:

$$S_M(\gamma) = (1, S^1(\gamma), S^2(\gamma), ..., S^M(\gamma)) \tag{25}$$

where $S^k(\gamma)$ denotes all the signature terms of order k, e.g.

$$S^1(\gamma) = (S(\gamma)^1, S(\gamma)^2, ..., S(\gamma)^D) \tag{26}$$

$$S^2(\gamma) = (S(\gamma)^{1,1}, S(\gamma)^{1,2}, ..., S(\gamma)^{D,D}) \tag{27}$$

Geometric and financial interpretation. As shown in (19), the geometric interpretation of the first order is the increment of the path along each dimension. In financial terms, this corresponds to the *drift*. It can be shown that the second order terms correspond to the *Levy area* [25], or the surface covered between the chord connecting the first and last coordinate in each dimension and the actual path, corresponding to a measure of *volatility* of the path. These two *global* features are captured by the first two orders, while more fine-grained, *local* features are captured by higher-order terms, as becomes apparent when looking at the factorial decay of S:

Factorial decay. One key property of signatures is factorial decay, which makes it a *graded summary* of paths.

As an analogue to the distributional setting (cf. Section 1.3 considering from R $\mathbb{P}(X)$ and using factor decomposition) consider the well-known principal component analysis (PCA). In PCA we use linear combinations of the data X to decompose it into its components that maximise their variance while being mutually orthogonal (uncorrelated). It is equivalent to the eigendecomposition of the covariance matrix of X. A key feature that we commonly see is exponential decay or rate decay, namely that the sorted absolute values of the eigenvalues of the covariance matrix of $X : \mathbb{R}^D$ decay fast enough, i.e. the j^{th} largest coefficient $|\beta|_j \leq Aj^{-a}, a \geq 1/2, \forall j$ and constants a and A do not depend on the dimension D. The latter simply implies that the first N components ($N << D$) already explain a vast part of the shared variance in the data set.

Informally, this intuition can be applied to paths using signatures as well. Lyons [66] shows that for paths of bounded variation[7] the following similar norm can be imposed on the signature terms (with $1 \leq i_1, ..., i_n \leq D$):

$$||\int ... \int d\gamma^{i_1} d\gamma^{i_2} ... d\gamma^{i_n}|| \leq \frac{||\gamma^n||^1}{n!} \tag{28}$$

with

$$||\gamma||^1 = \sup_{t_i \subset [0,T]} \sum_i |\gamma_{t_{i+1}} - \gamma_{t_i}| \tag{29}$$

where we take the supremum over all partitions of [0,T].

This theorem proven in [66] guarantees that higher-order terms of the signature have factorial decay, i.e. that the order of signatures imply a graded summary of the path, first describing global and increasingly more local characteristics of the path. This implies that the truncated signature for increasing orders throws away less and less information, similar to the low-rank approximation in PCA.

Signature as path moment generating function. Another key result that was recently developed by Chevyrev and Oberhauser [62] is that the signature can be seen as the moment generating function in the path space.

As discussed above, in the distributional setting there are well-established metrics to compare two distributions. In machine learning, we often encounter distributional distance metrics from information theory, such as the Kullback-Leibler (KL) and Jensen-Shannon (JS) divergences between two

[7] $\gamma : [0,T] \rightarrow \mathbb{R}$ is of bounded variation if all changes $\sum_i |\gamma_{t_{i+1}} - \gamma_{t_i}|$ are bounded (finite) for all partitions $0 \leq t_0 \leq t_1 \leq ... \leq T$

distributions. For stochastic processes that generate vector-valued data, there are well-known statistical tests for determining whether two samples are generated by the same stochastic process, such as the sequence of (normalised) moments and the Fourier transform (complex moments). As discussed, the MMD allows to compare these moments by embedding two random variables in Hilbert space using kernel approximation.

For path-valued data, Chevyrev and Oberhauser [62] introduce an analogue to normalised moments using the signature. They prove that for suitable normalizations λ, the sequence

$$(\mathbb{E}[\lambda(X)^m \int dX^{\otimes m}])_{m \geq 0} \tag{30}$$

determines the law of X *uniquely*[8]. They argue that the moments in the path space (or *sequential* moments) up to order m are preserved (i.e. a bijective property) for the truncated signature up to order m. [25] proposes the use of this result with efficient algorithms and tools from machine learning such as MMD and kernel approximation (e.g. [67]) for machine learning paths. [25] also argue in favor of signatures as a *provably* optimal feature map $\phi(.)$ for embedding paths generated by a stochastic process in into a linear space. The reasons are twofold: (1) *universality*, which implies that non-linear functions of the data are approximated by linear functionals in feature space and (2) *characteristicness*, which is exactly their merit, i.e. that the expected value of the feature map *characterizes the law of the random variable.*

Signature kernel. Let us now define the signature kernel as introduced in [67] and [68]. In essence, it is just the inner product between the two signature vectors of two random variables in the path space.

Definition 4.2 (Signature Kernel). *Let x and y be two paths supported on $[0, T]$, $x : [0, T] \to \mathbb{R}^D$ and $y : [0, T] \to \mathbb{R}^D$. The signature kernel k: $[0, T] \times [0, T] \to \mathbb{R}$ is defined as $k_S(x, y) = \langle S(x), S(y) \rangle$.*

Intuitively, say for S truncated at order 1, k_S measures the similarity between the drifts of the two paths. Truncated at order 2, k_S looks at drift and volatility similarity, and so and so forth.

[8]Up to tree-like equivalance, see [62].

Signature MMD. The MMD can be used as a deterministic loss function in a generative model as it is a distance measure between two random variables, for instance the *fake* generated data and the *true* input data. When the path structure of the random variable is crucial, the choice of traditional kernels is inappropriate and we should use a sequential kernel. As described above, this is exactly what signatures allow us to do. Let us first generalize the MMD expression in (14) to:

$$MMD(\mu,\nu) = sup_{f\in\mathcal{H}} E_{X\sim\mu}[f(X)] - E_{X\sim\nu}[f(X)] \tag{31}$$

Hence, MMD is literally the maximum expected distance between two functions in the embedded space $\mathcal{H}$. Closer resemblance to (16) is rewriting (31) into:

$$MMD_S(\mu,\nu) = E_{XX'\sim\mu}[k_S(X,X')] - 2E_{X\sim\mu,Y\sim\nu}[k_S(X,Y)] + E_{YY'\sim\nu}[k_S(Y,Y')] \tag{32}$$

The signature MMD. Expression (32) in itself is easy to compute, but its computational performance hinges on how efficiently we can evaluate k_S.

PDE kernel trick. An interesting recent result concerns a kernel trick for sequential kernels, the signature partial differential equation (PDE) kernel trick [68]. A first kernel trick for the signature kernel was introduced by Kiraly and Oberhauser [69]. [68] then proved that the signature kernel can be written as the solution of a hyperbolic PDE belonging to the family of so-called Goursat problems. This substantially speeds up the evaluation of k_S, and allows for GPU-optimized parallellization of the PDE solver. Formally, we can write:

$$\frac{\delta^2 k_S}{\delta s \delta j} = \langle \dot{X}(s), \dot{Y}(j)\rangle k_S \tag{33}$$

where $k_S(X(0),.) = k_S(.,Y(0)) = 1$ and $\dot{X}(s) = \frac{dX}{dt}|_{t=s}$ and $\dot{Y}(j) = \frac{dY}{dt}|_{t=j}$, which is a so-called Goursat PDE. The proof can be found in [68]. Supporting on (13) they further show that this PDE can be written as a function of a static kernel κ, e.g. the RBF or Matern kernel:

$$\frac{\delta^2 k_S}{\delta s \delta j} = (\kappa(X(s),Y(j)) - \kappa(X(s-1),X(j)) - \kappa(X(s),Y(j-1)) + \kappa(X(s),Y(j-1)))k_S \tag{34}$$

After an appropriate choice of κ, equation (34) can then be solved using state-of-the-art PDE solvers and efficiently parallellized over GPU. This allows for an efficient evalution of k_S in MMD_S (32).

4.2. Generative modeling architectures

This section briefly covers the main families of generative modeling architectures in the machine learning literature, linking them with their applications in the market generator literature so far in Section 4.2.1. They all have their advantages and disadvantages in terms of efficiency and convergence in training, innate complexity and tractability, flexibility to specific use cases, et cetera. This will concisely be covered for all of them. Next, Section 4.2.2 describes our conditional variational autoencoder model (CVAE) in more detail. Finally, 4.3 describes how conditional sampling, together with explainable machine learning (XML) tools such as Shapley (SHAP) values can help us better understand the optimal portfolios and sensitivities to changing market conditions in a data-driven portfolio construction context.

4.2.1. Main architectures in literature

Generative Adversarial Networks (GAN). Arguably the most popular architecture in generative ML are GANs. GANs [16] are a family of models trained by a game between two networks, a decoder network (previously f_{Θ}^{-1}(Z)) and a discriminator network. It samples a latent variable z from a simple prior distribution $\mathbb{P}(Z)$, e.g. Gaussian or Uniform, followed by a decoder network, the transform $G(z)$, called the Generator. The discriminator $D(.)$ outputs a probability of a given sample coming from the real data distribution. Its task is to distinguish samples from the real distribution $\mathbb{P}(X)$ from $G(z)$. The decoder tries to produce samples as close to the original distribution possible, as to fool the discriminator. This gives rise to the following well-known minimax problem:

$$\min_G \max_D \mathbb{E}_{x\sim\mathbb{P}(X)}[log(D(x)] + \mathbb{E}_{z\sim\mathbb{P}(Z)}[log(1 - D(G(z))] \tag{35}$$

GANs are by far the most popular choice in generative modeling in general [70], and for market generators in particular [30][33][71][37][49][72][45][43][32][73][74]. However, they are notoriously data hungry and difficult to train. This means they often require loads of data before convergence is achieved, which is a major issue in sparse data environments such as finance where thousands of samples are available rather than millions. Moreover, they suffer from *mode collapse*, they do not have tractable loglikelihoods in the loss function or interpretable posteriors or even latent densities after training, and suffer from general non-convergence and instability related to varying issues specific to the chosen architecture [70].

Generative moment matching networks (GMMN) a.k.a. MMD networks. GMMNs [64] are a family of relatively simple generative models that use a simple single forward pass through a neural network (rather than expensive adversarial training) to generate samples, and share in architecture that they use the MMD measure between the true and generated data in the loss function. Hence, they are often called MMD networks. They are called *moment matching* networks exactly for the reasons discussed in Section 4.1.2 on MMD being (under the Gaussian kernel) a discrepancy measure between all the moments of the generated '*fake*' and the true distribution. Although simple, the performance of GMMN has not always been consistent (e.g. blurry generated images), partly because of a high sensitivity to the choice of hyperparameters (such as widely diverging results as a function of the training batch size). Therefore, GMMNs are often used in combination with an autoencoder architecture. There are currently no market generators that focus on GMMNs, although it was included in a comparison by [32] and yielded comparable results as GANs for most evaluation metrics.

Variational autoencoders (VAE). VAEs were introduced by Kingma and Welling in 2014 [14] and are the second (after GAN) most popular architecture in generative modeling, with applications to market generators in [51][52][54]. VAEs are *autoencoders* in the sense that $f_\Theta(X)$ and $f_\Theta^{-1}(Z)$ are both neural networks, here respectively called the encoder and decoder network. VAEs are characterized by their joint distribution over the latent variables Z and the observed variables X: $\mathbb{P}(x, z) = \mathbb{P}(x|z)\mathbb{P}(z)$. Inference on $\mathbb{P}(X)$ is in this context called *variational inference using Bayes*, by applying Bayes rule on the joint distribution, evaluating the posterior distribution of X given the stochastic latent space Z. Again, the prior $\mathbb{P}(z)$ is chosen over a simple tractable distribution (mostly Gaussian), while $\mathbb{P}(x|z)$ is the distribution parametrized by by the decoder $f_\Theta^{-1}(Z)$. [14] approximates the posterior function $\mathbb{P}(z|x)$ using an encoder model $f_\Theta(X)$, which is unknown. Two contributions are key in appraising [14]. Firstly, they derive a lower bound for $\mathbb{P}(X)$ by comparing this posterior with samples from an actual Gaussian using the Kullback-Leibler divergence [75]:

$$\log(\mathbb{P}(x)) \geq \mathbb{E}_{f_\Theta(x)}[log(\mathbb{P}(x|z))] - KL(f_\Theta(x)||\mathbb{P}(z)) \tag{36}$$

Maximizing the right-hand side[9] of (36) thus corresponds to maximizing the loglikelihood of the data distribution as a function of Θ. Secondly, they use a mathematical trick called the *reparametrization trick* that allows for backpropagation (cf. Section 4.2.2) over the latent space Z [14]. The model converges fast, i.e. requires less data than GAN, and is generally more stable than the above-mentioned techniques. It offers interpretable conditional distributions and latent space. However, it is somewhat less flexible than GAN for alternative, bespoke loss functions that better reflect the downstream application of the generative model.

Restricted Boltzmann machines (RBM). RBMs are energy-based models that date back to the Harmonium model introduced by Smolensky in 1986 [17]. They are essentially undirected bipartite graphs (two-layer neural networks), with one visible layer v that represents $\mathbb{P}(X)$ and one hidden layer h representing $\mathbb{P}(Z)$. Restricted refers to the fact that there are no connections or model weights Θ between nodes within each layer, only across the two layers. Each node in the graph represents a binary stochastic variable. Boltzmann refers to the Boltzmann energy function that measures the likelihood of the states of the graph (which in statistical physics is called a Markov Random Field) by its joint distribution:

$$\mathbb{P}(v, h) = \frac{1}{Z} \exp\left(-E(v, h)\right) \tag{37}$$

$$E(v, h) = -\sum_{i=1}^{m} a_i v_i - \sum_{j=1}^{n} b_j h_j - \sum_{i=1}^{m} \sum_{j=1}^{n} w_{i,j} v_i h_j \tag{38}$$

where v_i and h_j denote the individual nodes or state variables in resp. v and h. In this case v_i and h_j are stochastic binary, hence Bernouilli, variables, but this can be approximated with Gaussian-Bernouilli variables for continuous distributions such as financial returns. The goal of training this network is maximizing its joint likelihood, which corresponds to minimizing the energy of the graph's state. Through Markov Chain Monte Carlo (MCMC) techniques such as Gibbs sampling and improved alterations of it such as contrastive divergence, it can be shown that the energy decreases as $\mathbb{P}_\Theta(X)$, the distribution of the visible layer with parameters Θ, approaches the true $\mathbb{P}(X)$, or the distribution of the data. Once training has converged,

[9]In this context referred to as the Evidence Lower Bound (ELBO).

one can iteratively sample noise in v and back and forth with h until we have new samples of $\mathbb{P}(X')$. This was the approach in the original Market Generator paper by Kondratyev and Schwarz [48]. The results were confirmed by [49]. RBMs are surprisingly fast to train and the binary states open unique segues for training RBM on quantum computer infrastructures. Moreover, the results in [48] offer very realistic simulations of multi-variate time series, reproducing stylized facts such as fat tails, autocorrelation and volatility clustering. However, as an energy-based model it has a very inflexible loss function in terms of customizability for downstream applications of the generative model.

Normalizing flows (NFs). NFs is a class of neural networks which use differentiable mappings to approximate bijective functions called *diffeomorphisms.* In our notation $f_\Theta^{-1}(Z)$ would be a neural network that stacks these diffeomorphisms (such as linear neural splines [76]) as to approximate some divergence measure between the target distribution $\mathbb{P}(X)$ and the sampled distribution $\mathbb{P}_\Theta(X')$. For instance, NFs approximate using neural nets these transformations by applying gradient descent to the Monte Carlo-approximated (MC) KL-divergence [47]:

$$\nabla_\Theta KL(\mathbb{P}(X)||\mathbb{P}_\Theta(X')) = -\mathbb{E}_{x\sim\mathbb{P}(X)}(\nabla_\Theta ln(\mathbb{P}_\Theta(X'))) \tag{39}$$

$$\approx \frac{1}{n}\sum_{i=1}^{n} \nabla_\Theta(ln(|detJ_{f_\Theta^{-1}}(f_\Theta(x_i))|) - ln\mathbb{P}(f_\Theta(x_i))) \tag{40}$$

where $J_{f_\Theta^{-1}}$ represents the Jacobian of the neural network f_Θ^{-1}, the matrix of first order derivatives of the network to the latent space values. The determinant of the Jacobian thus plays a crucial role in approximating the KL using MC. For the computation of the determinant to be efficient, the computation of the determinant of the individual diffeomorphisms is typically chosen simple (e.g. linear splines). Making them sufficiently simple but expressive enough is a key element of research in NFs [76]. As far as our knowledge stretches only [47] have introduced a flow-based market generator for multi-asset spot and option market simulation, in combination with an autoencoder architecture. They mainly use NFs to leverage the invertibility property and scale their methods to multi-asset simulation by fitting a Gaussian copula structure on the Gaussian latent space Z. Apart from the clear increase in complexity, there are no specific limitations of NFs mentioned in the literature, and they can easily be integrated with above-mentioned architectures such as autoencoders.

4.2.2. Conditional variational autoencoder

Given the above considerations, we decided to proceed with a variational autoencoder architecture. It converges fast[10] and is much more stable than competing architectures, and it will allow us to interpret the (conditional) distributions after training. In this section, we provide the inner details of our architecture in terms of the notation that we have established throughout the paper. We discuss the mappings $f_\Theta(X)$ and $f_\Theta^{-1}(Z)$, the loss function $\mathcal{L}(X, X')$, the training algorithm and hyperparameters and finally the conditions C.

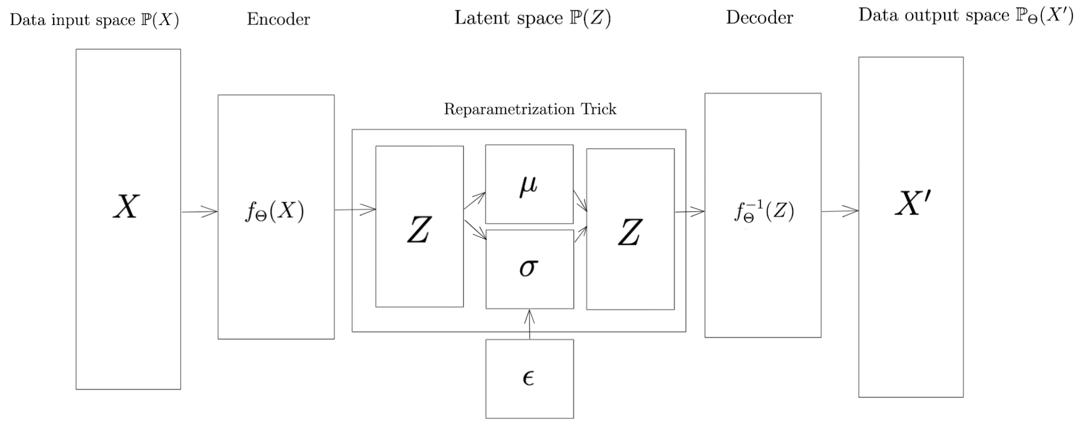

Figure 3: Variational autoencoder architecture

A detailed look at the architecture. The architecture of a VAE is summarized in Figure 3. As input we have the D-dimensional *ambient* space X or the physical data domain that we can measure (e.g. R or Ξ). Using a flexible neural network mapping $f_\Theta : \mathbb{R}^D \to \mathbb{R}^K, K << N$, called the encoder, we compress the dimension of the data into a K-dimensional *latent* space Z, e.g. 10-dimensional. Using the reparametrization trick [15] we map Z onto a mean μ and standard deviation σ vector, i.e. onto a K-dimensional Gaus-

[10]First experiments with VAE resulted in similar performance metrics with GAN, where VAE was trained c.30 seconds and GAN c.30 minutes.

sian, e.g. a 10-dimensional multi-variate normal distribution. Starting from multi-variate normal data, we can recombine μ and σ into a K-dimensional Z. The decoder neural network $f_\Theta^{-1} : \mathbb{R}^K \to \mathbb{R}^D$ maps the latent space back to the output space $\mathbb{P}_\Theta(X')$ where X' can be considered *reconstructed* samples in the training step, or genuinely new or fake samples in a generator step. The quality of the VAE clearly depends on the similarity between $\mathbb{P}(X)$ and $\mathbb{P}_\Theta(X')$.

Let us now zoom in on $f_\Theta(X)$ and $f_\Theta^{-1}(Z)$. Each neural network consist of one layer of J mathematical units called neurons:

$$f_{\Theta j} := A(\sum_i^D \theta_{i,j} x_i) \tag{41}$$

Every neuron takes *linear* combinations θ_i of the input data point x_i and is then *activated* using a *non-linear* activation function A, such as rectified linear units (ReLU), hyperbolic tangent (tanh) or sigmoid. In this paper we use a variant of ReLU called a *leaky ReLU*:

$$LReLU(x) = 1_{x<0}\alpha x + 1_{x\geq 0} x \tag{42}$$

where α is a small constant called the slope of the ReLU. All neurons J are linearly combined into the next layer (in this case Z):

$$Z_k := \sum_j^J \theta_{j,k} f_{\Theta j} \tag{43}$$

for every k in K. The decoder map can formally be written exactly like the encoder, but in reverse order.

The loss function of a VAE generally consists of two components, the latent loss ($\mathcal{L}_L$) and the reconstruction loss ($\mathcal{L}_R$):

$$\mathcal{L}(X, X') = \beta \mathcal{L}_L + (1 - \beta)\mathcal{L}_R \tag{44}$$

The latent loss is the Kullback-Leibler discrepancy between the latent distribution under its encoded parametrization, the posterior $f_\Theta(X) = \mathbb{P}_\Theta(Z|X)$, and its theoretical distribution, e.g. multi-variate Gaussian $\mathbb{P}(Z)$. Appendix B in [14] offers a simple expression for $\mathcal{L}_L$. The reconstruction loss is the cost of reproducing $\mathbb{P}_\Theta(X')$ after the dimension reduction step, and originally computed by the root of the mean squared error (RMSE or $L2$-loss) between X and X'.

$$\mathcal{L}(X, X') = \beta\frac{1}{2}\sum_{k}^{K}(1 + \sigma - \mu^2 - \exp(\sigma)) + (1 - \beta)\mathbb{E}(||X - X'||^2) \tag{45}$$

This loss function will be revisited below. The parameter β can be tuned to get so-called *disentangled* latent representations in the β-VAE architecture [77].

In terms of training, the learning algorithm is quasi identical to most deep learning methods. Optimal loss values $\mathcal{L}^*$ are determined by stochastically sampling batches of data and alternating forward and backward passes through the VAE. For each batch the data is first passed through the encoder network and decoder network (*forward pass*), after which $\mathcal{L}$ is evaluated in terms of Θ. At each layer, the derivative of $\mathcal{L}$ vis-a-vis Θ can easily be evaluated. Next (*the backward pass*), we say the calculated loss *backpropagates* through the network, and Θ are adjusted in the direction of the gradient $\nabla_\Theta \mathcal{L}$ with the *learning rate* as step size. The exact optimizer algorithm we used for this is Adam (Adaptive moments estimation, [78]). Finally, we can also use a concept called regularization, which penalizes neural models that become too complex or overparametrized. We used a tool called dropout, that during training randomly sets a proportion of parameters in Θ equal to zero, and leaves those connections at zero that contribute the least to the prediction.

In summary, the hyperparameters of this architecture are: (1) the number of neurons in the encoder, (2) the number of neurons in the decoder, (3) the number of latent dimensions K, (4) the learning rate and (5) the optimizer algorithm and (6) the dropout rate. We opted for the following set-up (which was optimized using Grid Search): 100, 100, 10, 0.001, Adam, 0.0.

After training, in the sampling or generation step, we start from a random K-dimensional noise $\epsilon \sim \mathbb{P}(Z)$ which is K-variate Gaussian. Now, we only need a decode step to generate new samples of $\mathbb{P}_\Theta(X')$

Conditional VAEs. It is worth mentioning here that autoencoders are the non-linear generalization of the principal component analysis. It can be shown that with linear activation, e.g. $A : \mathbb{R} \to \mathbb{R} : A(x) = x$, and squared loss ($L2$-loss) the K factors in Z become the PCA factors. For PCA in financial applications, it is common knowledge that these factors can change over time when certain conditions change. To account for this, a popular method is to include *instruments*, or Instrumented PCA (IPCA), that are in se exogenous to the timeseries in X but endogenous to the factor model. These exogenous variables are e.g. macro-economic conditions, while the timeseries are asset returns, and the factors are traditional factors from asset pricing (*market*, *momentum*, *size*,...).

For VAEs it could be argued that the factors are also conditional, i.e. there are exogenous variables or conditions C such that factors conditional on these variables lead to useful $\mathbb{P}_\Theta(X'|C)$. This is a small change to the VAE's architecture as we depict below in Figure 4. Note that the reparametrization in Z is still happening but we left that out for simplicity. During training, both the encoder and decoder now take the values for the condition variables that were prevalent when the particular batch X_i was sampled as additional input variables. In the generator step, we need both a source of noise, as well as the *currently prevalent* conditions. Alternatively, we could alter the conditions today, to see what happens with X'. This will be discussed in the Section 4.3.

In [48] the prevalent level of volatility of the modeled assets were used as a condition in a conditional RBM. This was crucial to get heteroskedasticity in the output paths and the related stylized fact of volatility clusters. In [51] the authors use both the lagged path, the index level and the contemporanous level of volatility as conditions in a CVAE architecture. Their common motivation was that, compared to $P_\Theta(X')$, $P_\Theta(X'|C)$ much better reflects the non-stationary aspects of the paths. The conditions that are relevant for our problem will be introduced in Section 4.3, and elaborated (numerically) in Section 5.

***MMD-VAEs*. Let us revisit the loss function in (45).**

[ENTER DISCUSSION ON SIG-MDD]

Main argument in this paper: vastly dependent on X (R, Ξ ?) and the downstream application.

4.3. Explainable machine learning (XML): attributing conditions to portfolios

Conditions form *economic priors* to the model. They let us generate scenarios under the prevalent market circumstances (such as contemporaneous volatility), but they also alow us to *vary* these external indicators and evaluate what that implies for our paths (and eventually our optimal portfolios) today.

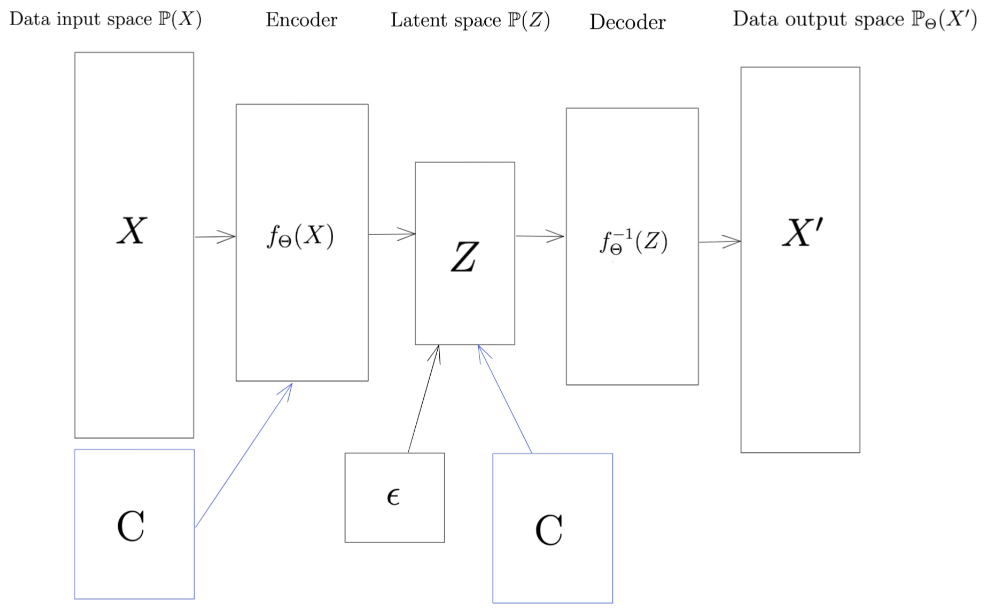
Data input space $\mathbb{P}(X)$
Encoder
Latent space $\mathbb{P}(Z)$
Decoder
Data output space $\mathbb{P}_\Theta(X')$
X
$f_\Theta(X)$
Z
$f_\Theta^{-1}(Z)$
X'
C
ϵ
C

Figure 4: Conditional variational autoencoder architecture

The *path features* that we want to preserve (e.g. drift and vol in R or drawdown in Ξ) can now be considered the *geometric priors* of the model. Whereas traditional simulation techniques have focused on the DGP, machine learning allows us to approximate the DGP using flexible mappings and shifts the focus of the modeling exercise to these two *priors*. It allows us to *train on historical conditions*, while *simulating on current conditions*. This also gives leeway for introducing nowcasting timeseries (such as real-time macro data) into the model.

As a simpler toy example, we focused on the US market and collected 100 conditions C_i from the Federal Reserve Economic Data (FRED) database, including credit and monetary data, interest rates, employment, commodity prices, stress indicators, volatility indices, and consumer sentiment. Figure 6 gives an overview of the high-level categories. Table 2 in Appendix gives a list of all the indicators we considered. To get a first idea of the most apposite conditions, we look at the total drawdown path[11] of the total US stock market

[11]I.e. one $\xi, T \approx 5000, D = 2$

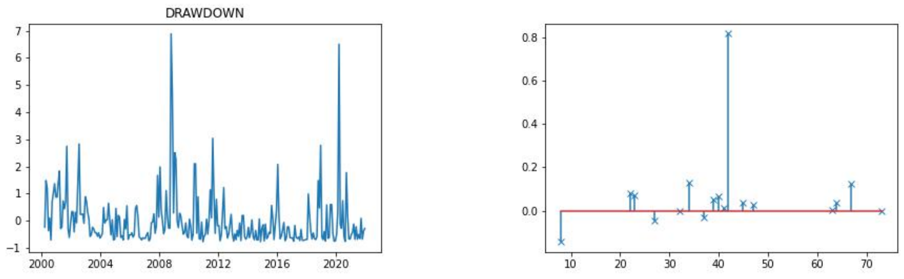

Figure 5: The evolution of ξ of US Stock market index (Wilshire, left), the LASSO coefficients of the conditions (right)

(Wilshire) and do a LASSO[12] regression to select the conditions with the most shared variance with ξ. This gives the weights displayed in Table 2. As expected, the CBOE volatility index dominates the other coefficients. Other stress indicators, such as the FED St. Louis Financial Stress Index [79] and declining Consumer Sentiment as measured by the University of Michigan [80] contribute significantly to general market drawdown. Moreover, timeseries of manufacturing, export, currency and long-term mortgage rate date were found significant. Only 4 factors had a significant negative impact on the historical market drawdown.

The aim of this analysis is to introduce appropriate C_i to our generative model, such that we can evaluate $\mathbb{P}(X'|C)$ at the current level of C as well as for our own scenarios of C. For instance, given the current level of volatility, what do drawdown paths and the optimal portfolio look like, and which positions are most affected if one gradually increases the volatility to levels seen during the GFC or the Covid-19-induced March 2020 meltdown? What does one's portfolio look like with current market sentiment, and which positions are likely to be first and mostly affected when sentiment turns sour gradually? This is what we conceptually want to introduce here, and will numerically evaluate in Section 5.

As a tool to evaluate changing paths and portfolios to changing condi-

[12]Least Absolute Shrinkage and Selection Operator, a simple linear regression with a L1-norm penalty on the coefficients, https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Lasso. We used 10-fold cross-validation to find the optimal penalty hyperparameter.

Largest positive contributors to ξ		**Largest negative contributors to ξ**	
CBOE Volatility Index	0,815680	US Gov't Securities at All Com. Banks	-0,142223
Avg Weekly OT Hours: Manufacturing	0,129751	Long Term Unemployment: 27 WKS	-0,043401
Exports to Mexico	0,126585	JPN/USD Currency Exchange Rate	-0,029723
Univ. of Michigan: Consumer Sentiment	0,079161	Avg Hourly Earnings: Manufacturing	-0,001523
St. Louis Financial Stress Index	0,072814		
CNY/USD Currency Exchange Rate	0,068154		
CAD/USD Currency Exchange Rate	0,053743		
Imports from UK	0,038683		
30-yr Conventional Mortgage Rate	0,037272		
Effective Federal Funds Rate	0,029571		

Table 2: Lasso coefficients of C_i to ξ

tions, we use Shapley (SHAP) values [81]. Given one set of n_{cond} conditions $C = (C_i)_{i=\{1,\dots,n_{cond}\}}$, an optimal portfolio can be seen as a linear combination w^*_d, for $d \in D$, where the weights reflect some contribution (of risk, return, drawdown) to the optimal portfolio timeseries w^*R or $w^*\Pi$. However, given a set of N_s condition sets $\mathcal{C} = (C^k)_{k=\{1,\dots,N_s\}}$, each set corresponding to a C that generates sequences R or Π, each C will also correspond to a unique optimal portfolio, i.e. for each k. Now we can see the w^*_k as the output, and evaluate the contribution of each condition C_i in C^k to the optimal portfolio. The SHAP values to each w^*_d can then formally be defined as:

$$\Phi_i(w^*_d) = \sum_{S \subset [N_s \setminus \{i\}]} \frac{|S|!(N_s - |S| - 1)!}{N_s!} (w^*_d(S \cup \{i\}) - w^*_d(S))) \tag{46}$$

This is the SHAP Φ_i for condition i in C in terms of optimal weight w^*_d. Intuitively, for the N_s optimal portfolios we evaluate all the subsets S where condition i did not contribute to the optimal portfolio $w^*_d(S)$ and compare with the optimal portfolios where it was $w^*_d(S \cup \{i\})$. The average contribution of this condition to the optimal weight thus constitutes the SHAP value. This allows for visualizations of the *conditional* optimal portfolios, such as waterfall and beeswarm plots [81], that are popular explainable machine learning tools for applications in deep learning and computer vision.

5. Numerical Results

6. Conclusions

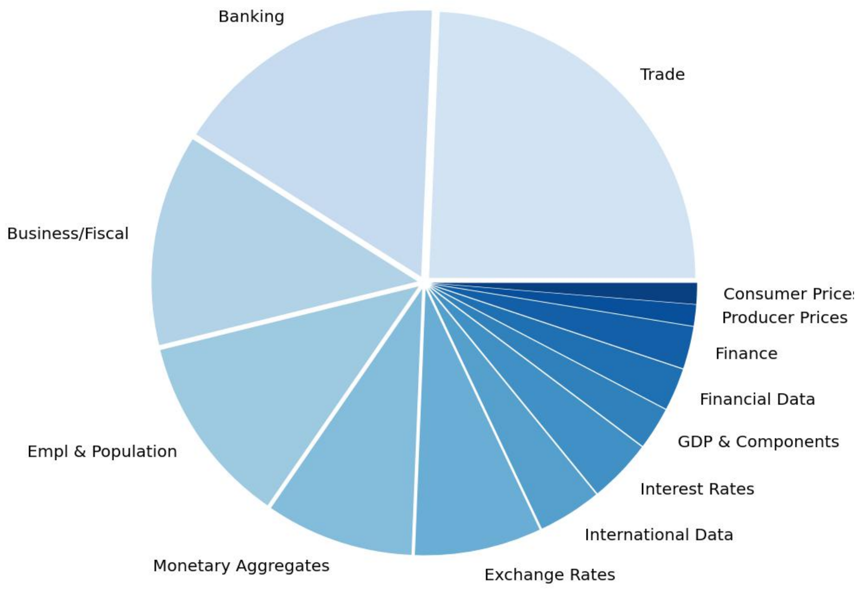

Figure 6: Macro condition high-level categories

7. Appendix

Table 3: Macro-economic conditions

ID	FRED ID	FRED Cat.	Detailed Cat.	Indicator
0	TREAST	Finance	Monetary Data	US Treasuries Held by the Fed
1	MBST	Finance	Monetary Data	Mortgage Backed Sec Held by the Fed
2	WALCL	Banking	Monetary Factors	All Fed Reserve Banks - Total Assets
3	TLAACBW027SBOG	Banking	Monetary Factors	All Commercial Banks - Total Assets
4	BOPBCA	Banking	Conditions	Number of US Banks
5	USNUM	Banking	Conditions	Number of US Commercial Banks
6	EQTA	Banking	Conditions	Equity/Asset Ratio
7	TOTBKCR	Banking	Commercial Credit	Bank Credit of All Commercial Banks
8	TOTALSEC	Banking	Commercial Credit	Securitized Total Consumer Loans
9	TOTALSL	Banking	Commercial Credit	Total Consumer Credit Outstanding
10	INVEST	Banking	Investment	Total Investments All Commercial Banks
11	USGSEC	Banking	Investment	US Gov't Securities at All Com. Banks
12	CONSUMER	Banking	Loans	Total Consumer Loans
13	BUSLOANS	Banking	Loans	Total Commercial/Industrial Loans
14	DALLCACBEP	Banking	Delinquencies	Delinquencies On All Loans And Leases
15	T10Y2Y	Banking	Interest Rates	US 10-YR / 2-YR Spread
16	TB3MS	Banking	Interest Rates	3-Month T-Bill: Secondary Market Rate
17	DGS10	Banking	Interest Rates	10-Yr Treasury Const. Maturity Rate
18	GFDEBTN	Business/Fiscal	Federal Government	Federal Government Debt (Public)
19	FYOINT	Business/Fiscal	Federal Government	Interest on National Debt
20	FYONET	Business/Fiscal	Federal Government	Federal Spending
21	FYFR	Business/Fiscal	Federal Government	Federal Receipts
22	FYFSD	Business/Fiscal	Federal Government	Budget Deficit/Surplus
23	CDSP	Business/Fiscal	Household Sector	Consumer Debt/Income Ratio
24	PERMIT	Business/Fiscal	Household Sector	New Home Permits
25	HSN1F	Business/Fiscal	Household Sector	New Home Sales
26	CMDEBT	Business/Fiscal	Household Sector	Outstanding Mortgage Debt
27	DGORDER	Business/Fiscal	Ind. Production	Manufacturers' New Orders
28	TCU	Business/Fiscal	Ind. Production	Capacity Utilization: Total Industry
29	TTLCONS	Business/Fiscal	Construction	Total Construction Spending
30	BUSINV	Business/Fiscal	Other	Total Business Inventories
31	ALTSALES	Business/Fiscal	Other	Light Weight Vehicle Sales
32	UMCSENT	Business/Fiscal	Other	Univ of Michigan: Consumer Sentiment
33	STLFSI	Business/Fiscal	Other	St. Louis Financial Stress Index
34	OILPRICE	Business/Fiscal	Other	Spot Oil Price - West Texas Intermediate
35	CPIAUCSL	Consumer Prices	CPI	Consumer Price Index: Seasonally Adj.
36	UNRATE	Empl & Population	Household Survey	Civilian Total Unemployment Rate
37	UEMP27OV	Empl & Population	Household Survey	Long Term Unemployment: 27 WKS
38	UEMPMED	Empl & Population	Household Survey	Length of Unemployment
39	CE16OV	Empl & Population	Household Survey	Total US Workforce
40	EMRATIO	Empl & Population	Household Survey	US Employment/Population Ratio
41	POP	Empl & Population	Population	US Population
42	AHEMAN	Empl & Population	Est. Survey	Avg Hourly Earnings: Manufacturing
43	AWHMAN	Empl & Population	Est. Survey	Avg Weekly Hours: Manufacturing
44	AWOTMAN	Empl & Population	Est. Survey	Avg Weekly OT Hours: Manufacturing
45	DEXUSUK	Exchange Rates	Daily Rates	USD/GBP Currency Exchange Rate
46	DEXUSEU	Exchange Rates	Daily Rates	USD/EUR Currency Exchange Rate
47	DEXJPUS	Exchange Rates	Daily Rates	JPN/USD Currency Exchange Rate
48	DEXMXUS	Exchange Rates	Daily Rates	MXP/USD Currency Exchange Rate
49	DEXCAUS	Exchange Rates	Daily Rates	CAD/USD Currency Exchange Rate
50	DEXCHUS	Exchange Rates	Daily Rates	CNY/USD Currency Exchange Rate
51	COMPOUT	Financial Data	Monetary	Commercial Paper Outstanding

Continued on next page

Table 3 – continued from previous page

ID	FRED ID	FRED Cat.	Detailed Cat.	Indicator
52	VIXCLS	Financial Data	Volatility Indexes	CBOE Volatility Index
53	GDP	GDP & Components	GDP/GNP	US Gross Domestic Product
54	GNP	GDP & Components	GDP/GNP	US Gross National Product
55	NETFI	GDP & Components	Imports & Exports	US Current Account Balance
56	EXPGS	GDP & Components	Imports & Exports	US Exports Goods & Services
57	IMPGS	GDP & Components	Imports & Exports	US Imports Goods & Services
58	DGI	GDP & Components	Govt Accounting	Fed Govt: Defense Budget
59	FGRECPT	GDP & Components	Govt Accounting	Fed Govt: Tax Receipts
60	TGDEF	GDP & Components	Govt Accounting	Fed Govt: Budget Deficit
61	CP	GDP & Components	Industry	Corporate Profits After Tax
62	DIVIDEND	GDP & Components	Industry	Corporate Dividends
63	PI	GDP & Components	Personal	Personal Income
64	PSAVE	GDP & Components	Savings & Inv.	Personal Savings
65	PSAVERT	GDP & Components	Savings & Inv.	Personal Savings Rate
66	MORTGAGE30US	Interest Rates	30yr Mortgage	30-yr Conventional Mortgage Rate
67	DPCREDIT	Interest Rates	FRB Rates	Discount Rate
68	FEDFUNDS	Interest Rates	FRB Rates	Effective Federal Funds Rate
69	GRCPROINDMISMEI	International Data	Indicators	Production of Total Industry in Greece
70	GRCSARTMISMEI	International Data	Indicators	Total Retail Trade in Greece
71	GRCURHARMMDSMEI	International Data	Indicators	Unemployment Rate - Greece
72	M1	Monetary Aggregates	M1	M1 Money Supply
73	M2	Monetary Aggregates	M2	M2 Money Supply
74	MZM	Monetary Aggregates	MZM	MZM Money Supply
75	M1V	Monetary Aggregates	M1	Velocity of M1 Money Stock
76	M2V	Monetary Aggregates	M2	Velocity of M2 Money Stock
77	MZMV	Monetary Aggregates	MZM	Velocity of MZM Money Stock
78	MULT	Monetary Aggregates	M1	M1 Money Multiplier
79	PPIACO	Producer Prices	PPI	Producer Price Index: All Commodities
80	IMPCH	Trade	Imports	Imports from China
81	IMPJP	Trade	Imports	Imports from Japan
82	IMPMX	Trade	Imports	Imports from Mexico
83	IMPCA	Trade	Imports	Imports from Canada
84	IMPGE	Trade	Imports	Imports from Germany
85	IMPUK	Trade	Imports	Imports from UK
86	EXPCH	Trade	Exports	Exports to China
87	EXPJP	Trade	Exports	Exports to Japan
88	EXPMX	Trade	Exports	Exports to Mexico
89	EXPCA	Trade	Exports	Exports to Canada
90	EXPGE	Trade	Exports	Exports to Germany
91	EXPUK	Trade	Exports	Exports to UK
92	BOPGEXP	Trade	Exports	Exports: Goods
93	BOPGIMP	Trade	Imports	Imports: Goods
94	BOPGTB	Trade	Balance	Balance: Goods
95	EXPGS	Trade	Exports	Exports: Services
96	BOPSIMP	Trade	Imports	Imports: Services
97	BOPSTB	Trade	Balance	Balance: Services
98	BOPGSTB	Trade	Balance	Balance: Goods & Services

Z

X

X'

μ

σ

ϵ

Encoder

Decoder

Reparametrization Trick

C

Data input space $\mathbb{P}(X)$

Latent space $\mathbb{P}(Z)$

Data output space $\mathbb{P}_\Theta(X')$
$\propto$

References

[1] S. Gu, B. Kelly, D. Xiu, Empirical asset pricing via machine learning, Technical Report, National bureau of economic research, 2018.

[2] S. Gu, B. Kelly, D. Xiu, Autoencoder asset pricing models, Journal of Econometrics 222 (2021) 429–450.

[3] G. Feng, S. Giglio, D. Xiu, Taming the factor zoo: A test of new factors, The Journal of Finance 75 (2020) 1327–1370.

[4] Y. Nevmyvaka, Y. Feng, M. Kearns, Reinforcement learning for optimized trade execution, in: Proceedings of the 23rd international conference on Machine learning, pp. 673–680.

[5] L. Leal, M. Laurière, C.-A. Lehalle, Learning a functional control for high-frequency finance, arXiv preprint arXiv:2006.09611 (2020).

[6] O. Mounjid, C.-A. Lehalle, Improving reinforcement learning algorithms: towards optimal learning rate policies, arXiv preprint arXiv:1911.02319 (2019).

[7] M. L. De Prado, Advances in financial machine learning, John Wiley & Sons, 2018.

[8] M. M. L. de Prado, Machine learning for asset managers, Cambridge University Press, 2020.

[9] M. F. Dixon, I. Halperin, P. Bilokon, Machine Learning in Finance, Springer, 2020.

[10] M. Jaeger, S. Krügel, D. Marinelli, J. Papenbrock, P. Schwendner, Interpretable machine learning for diversified portfolio construction, The Journal of Financial Data Science (2021).

[11] J. Papenbrock, P. Schwendner, M. Jaeger, S. Krügel, Matrix evolutions: synthetic correlations and explainable machine learning for constructing robust investment portfolios, The Journal of Financial Data Science 3 (2021) 51–69.

[12] H. Zou, T. Hastie, Regularization and variable selection via the elastic net, Journal of the royal statistical society: series B (statistical methodology) 67 (2005) 301–320.

[13] G. James, D. Witten, T. Hastie, R. Tibshirani, Statistical learning, in: An introduction to statistical learning, Springer, 2021, pp. 15–57.

[14] D. P. Kingma, M. Welling, Stochastic gradient vb and the variational auto-encoder, in: Second International Conference on Learning Representations, ICLR, volume 19, p. 121.

[15] D. P. Kingma, S. Mohamed, D. J. Rezende, M. Welling, Semi-supervised learning with deep generative models, in: Advances in neural information processing systems, pp. 3581–3589.

[16] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, Advances in neural information processing systems 27 (2014).

[17] P. Smolensky, Information processing in dynamical systems: Foundations of harmony theory, Technical Report, Colorado Univ at Boulder Dept of Computer Science, 1986.

[18] Y. Bengio, E. Laufer, G. Alain, J. Yosinski, Deep generative stochastic networks trainable by backprop, in: International Conference on Machine Learning, PMLR, pp. 226–234.

[19] D. Rezende, S. Mohamed, Variational inference with normalizing flows, in: International conference on machine learning, PMLR, pp. 1530–1538.

[20] P. Glasserman, Monte Carlo methods in financial engineering, volume 53, Springer, 2004.

[21] F. Black, M. Scholes, The pricing of options and corporate liabilities, Journal of political economy 81 (1973) 637–654.

[22] S. L. Heston, A closed-form solution for options with stochastic volatility with applications to bond and currency options, The review of financial studies 6 (1993) 327–343.

[23] P. S. Hagan, D. Kumar, A. S. Lesniewski, D. E. Woodward, Managing smile risk, The Best of Wilmott 1 (2002) 249–296.

[24] J. Gatheral, T. Jaisson, M. Rosenbaum, Volatility is rough, Quantitative finance 18 (2018) 933–949.

[25] I. Chevyrev, A. Kormilitzin, A primer on the signature method in machine learning, arXiv preprint arXiv:1603.03788 (2016).

[26] H. M. Markowitz, G. P. Todd, Mean-variance analysis in portfolio choice and capital markets, volume 66, John Wiley & Sons, 2000.

[27] R. Cont, Empirical properties of asset returns: stylized facts and statistical issues, Quantitative finance 1 (2001) 223.

[28] A. Chekhlov, S. Uryasev, M. Zabarankin, Drawdown measure in portfolio optimization, International Journal of Theoretical and Applied Finance 8 (2005) 13–58.

[29] P. Henry-Labordere, Generative models for financial data, Available at SSRN 3408007 (2019).

[30] M. Wiese, L. Bai, B. Wood, H. Buehler, Deep hedging: learning to simulate equity option markets, arXiv preprint arXiv:1911.01700 (2019).

[31] C. Cuchiero, W. Khosrawi, J. Teichmann, A generative adversarial network approach to calibration of local stochastic volatility models, Risks 8 (2020) 101.

[32] H. Ni, L. Szpruch, M. Wiese, S. Liao, B. Xiao, Conditional sig-wasserstein gans for time series generation, arXiv preprint arXiv:2006.05421 (2020).

[33] M. Wiese, R. Knobloch, R. Korn, P. Kretschmer, Quant gans: Deep generation of financial time series, Quantitative Finance 20 (2020) 1419–1440.

[34] J. Li, X. Wang, Y. Lin, A. Sinha, M. Wellman, Generating realistic stock market order streams, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 34, pp. 727–734.

[35] V. Storchan, S. Vyetrenko, T. Balch, Mas-gan: Adversarial calibration of multi-agent market simulators. (2020).

[36] G. Benedetti, Yield curve quantization and simulation with neural networks, Available at SSRN 3577555 (2020).

[37] T. Xu, L. K. Wenliang, M. Munn, B. Acciaio, Cot-gan: Generating sequential data via causal optimal transport, arXiv preprint arXiv:2006.08571 (2020).

[38] F. D. M. Pardo, R. C. López, Mitigating overfitting on financial datasets with generative adversarial networks, The Journal of Financial Data Science 2 (2020) 76–85.

[39] H. Buehler, P. Murray, M. S. Pakkanen, B. Wood, Deep hedging: Learning to remove the drift under trading frictions with minimal equivalent near-martingale measures, arXiv preprint arXiv:2111.07844 (2021).

[40] H. Ni, L. Szpruch, M. Sabate-Vidales, B. Xiao, M. Wiese, S. Liao, Sig-wasserstein gans for time series generation, arXiv preprint arXiv:2111.01207 (2021).

[41] M. Pfenninger, S. Rikli, D. N. Bigler, Wasserstein gan: Deep generation applied on financial time series, Available at SSRN 3877960 (2021).

[42] A. Rosolia, J. Osterrieder, Analyzing deep generated financial time series for various asset classes, Available at SSRN 3898792 (2021).

[43] A. Koshiyama, N. Firoozye, P. Treleaven, Generative adversarial networks for financial trading strategies fine-tuning and combination, Quantitative Finance 21 (2021) 797–813.

[44] J. van Rhijn, C. W. Oosterlee, L. A. Grzelak, S. Liu, Monte carlo simulation of sdes using gans, arXiv preprint arXiv:2104.01437 (2021).

[45] G. Marti, V. Goubet, F. Nielsen, ccorrgan: Conditional correlation gan for learning empirical conditional distributions in the elliptope, in: International Conference on Geometric Science of Information, Springer, pp. 613–620.

[46] B. Coyle, M. Henderson, J. C. J. Le, N. Kumar, M. Paini, E. Kashefi, Quantum versus classical generative modelling in finance, Quantum Science and Technology 6 (2021) 024013.

[47] M. Wiese, B. Wood, A. Pachoud, R. Korn, H. Buehler, P. Murray, L. Bai, Multi-asset spot and option market simulation, arXiv preprint arXiv:2112.06823 (2021).

[48] A. Kondratyev, C. Schwarz, The market generator, Available at SSRN 3384948 (2019).

[49] E. Lezmi, J. Roche, T. Roncalli, J. Xu, Improving the robustness of trading strategy backtesting with boltzmann machines and generative adversarial networks, Available at SSRN 3645473 (2020).

[50] R. Wang, Discriminating modelling approaches for point in time economic scenario generation, arXiv preprint arXiv:2108.08818 (2021).

[51] H. Buehler, B. Horvath, T. Lyons, I. Perez Arribas, B. Wood, A data-driven market simulator for small data environments, Available at SSRN 3632431 (2020).

[52] N. J. C.-k. Fung, Variational Autoencoders for Volatility Surfaces, Ph.D. thesis, 2021.

[53] M. G. Frandsen, Greeks need not apply (2021).

[54] M. Bergeron, N. Fung, Z. Poulos, J. C. Hull, A. Veneris, Variational autoencoders: A hands-off approach to volatility, Available at SSRN 3827447 (2021).

[55] B. Ning, S. Jaimungal, X. Zhang, M. Bergeron, Arbitrage-free implied volatility surface generation with variational autoencoders, arXiv preprint arXiv:2108.04941 (2021).

[56] B. Graham, Sparse arrays of signatures for online character recognition, arXiv preprint arXiv:1308.0371 (2013).

[57] I. P. Arribas, G. M. Goodwin, J. R. Geddes, T. Lyons, K. E. Saunders, A signature-based machine learning model for distinguishing bipolar disorder and borderline personality disorder, Translational psychiatry 8 (2018) 1–7.

[58] T. Cochrane, P. Foster, V. Chhabra, M. Lemercier, T. Lyons, C. Salvi, Sk-tree: a systematic malware detection algorithm on streaming trees via the signature kernel, in: 2021 IEEE International Conference on Cyber Security and Resilience (CSR), IEEE, pp. 35–40.

[59] P. Moore, T. Lyons, J. Gallacher, A. D. N. Initiative, Using path signatures to predict a diagnosis of alzheimer's disease, PloS one 14 (2019) e0222212.

[60] W. Yang, T. Lyons, H. Ni, C. Schmid, L. Jin, Developing the path signature methodology and its application to landmark-based human action recognition, arXiv preprint arXiv:1707.03993 (2017).

[61] J. Kalsi, T. Lyons, I. P. Arribas, Optimal execution with rough path signatures, SIAM Journal on Financial Mathematics 11 (2020) 470–493.

[62] I. Chevyrev, H. Oberhauser, Signature moments to characterize laws of stochastic processes, arXiv preprint arXiv:1810.10971 (2018).

[63] R. Schaback, H. Wendland, Kernel techniques: from machine learning to meshless methods, Acta numerica 15 (2006) 543–639.

[64] Y. Li, K. Swersky, R. Zemel, Generative moment matching networks, in: International Conference on Machine Learning, PMLR, pp. 1718–1727.

[65] K.-T. Chen, Iterated path integrals, Bulletin of the American Mathematical Society 83 (1977) 831–879.

[66] T. Lyons, Rough paths, signatures and the modelling of functions on streams, arXiv preprint arXiv:1405.4537 (2014).

[67] C. Salvi, M. Lemercier, C. Liu, B. Hovarth, T. Damoulas, T. Lyons, Higher order kernel mean embeddings to capture filtrations of stochastic processes, arXiv preprint arXiv:2109.03582 (2021).

[68] C. Salvi, T. Cass, J. Foster, T. Lyons, W. Yang, The signature kernel is the solution of a goursat pde, SIAM Journal on Mathematics of Data Science 3 (2021) 873–899.

[69] F. J. Király, H. Oberhauser, Kernels for sequentially ordered data, Journal of Machine Learning Research 20 (2019) 1–45.

[70] D. Saxena, J. Cao, Generative adversarial networks (gans) challenges, solutions, and future directions, ACM Computing Surveys (CSUR) 54 (2021) 1–42.

[71] R. Luo, W. Zhang, X. Xu, J. Wang, A neural stochastic volatility model, in: Thirty-second AAAI conference on artificial intelligence.

[72] G. 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), IEEE, pp. 8459–8463.

[73] F. De Meer Pardo, Enriching financial datasets with generative adversarial networks (2019).

[74] S. Takahashi, Y. Chen, K. Tanaka-Ishii, Modeling financial time-series with generative adversarial networks, Physica A: Statistical Mechanics and its Applications 527 (2019) 121261.

[75] P. Hall, On kullback-leibler loss and density estimation, The Annals of Statistics (1987) 1491–1519.

[76] I. Kobyzev, S. Prince, M. Brubaker, Normalizing flows: An introduction and review of current methods, IEEE Transactions on Pattern Analysis and Machine Intelligence (2020).

[77] C. P. Burgess, I. Higgins, A. Pal, L. Matthey, N. Watters, G. Desjardins, A. Lerchner, Understanding disentangling in β-vae, arXiv preprint arXiv:1804.03599 (2018).

[78] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980 (2014).

[79] K. L. Kliesen, D. C. Smith, et al., Measuring financial market stress, economic synopses (2010).

[80] R. T. Curtin, Indicators of consumer behavior: The university of michigan surveys of consumers, Public Opinion Quarterly 46 (1982) 340–352.

[81] S. M. Lundberg, S.-I. Lee, A unified approach to interpreting model predictions, in: Proceedings of the 31st international conference on neural information processing systems, pp. 4768–4777.