SYNTHETIC TO SPX: MODEL-BASED PRE-TRAINING FOR DIFFUSION MODELS OF FINANCIAL TIME SERIES

Anonymous authors
Paper under double-blind review

ABSTRACT

Diffusion models have achieved remarkable success in image, video, and text generation. We apply them to financial time series. The main obstacle is the scarcity of data: only one realized price trajectory is available for each asset. Finance has, however, a major advantage: fifty years of mathematical models that capture the main mechanisms of markets and can be simulated at will. These models provide unlimited training data, while their known statistical properties enable controlled evaluation of generated trajectories. We therefore propose *model-based pre-training*: first, train the score network on parametric models of increasing complexity; second, fine-tune it on the single real trajectory. In the first phase, we compare two approaches: one learns the joint path distribution in $\mathbb{R}^{N\times d}$, the other factorizes it into conditional distributions over $\mathbb{R}^d$. The autoregressive formulation is the most data-efficient, reaching a given fidelity with up to four times fewer trajectories; with abundant data, all architectures perform alike. We then fine-tune on the overlapping windows of a single trajectory, first on a path-dependent volatility model calibrated on SPX and held out from pre-training, where the ground truth is known, and then on SPX itself. Pre-training improves adaptation in this low-data regime, and what matters is the proximity of the source models to the target rather than their diversity. On SPX, the pre-trained model recovers volatility clustering, heavy tails and the leverage effect, which training from scratch misses.

1 INTRODUCTION

Diffusion models have emerged as a powerful paradigm for generative modeling by learning to reverse a gradual noise corruption process (Sohl-Dickstein et al., 2015; Song & Ermon, 2019; Ho et al., 2020; Song et al., 2021). They have recently been adapted to time series, for forecasting and imputation (Rasul et al., 2021; Tashiro et al., 2021; Lopez Alcaraz & Strodthoff, 2023; Shen & Kwok, 2023; Kollovieh et al., 2023) and for unconditional generation (Yuan & Qiao, 2024), and to finance (Huang et al., 2024; Takahashi & Mizuno, 2025), where generative models had mostly relied on GANs (Yoon et al., 2019; Wiese et al., 2020; Ni et al., 2021; Cont et al., 2026). In these works, the training set is abundant, and how many samples the generator actually needs is rarely studied.

Financial time-series generation. Generating time series raises specific difficulties: they have a strict temporal order and strong temporal dependence that limits effective sample sizes. Also, evaluating the quality of generated samples is difficult. Financial time series make the task even harder due to non-stationarity, macroeconomic regime shifts and fat-tailed return distributions. Furthermore, their signal-to-noise ratio is exceptionally low. Above all, only one historical trajectory exists per asset.

How can diffusion models be trained on financial data with only one realized trajectory?

Our approach. Finance, however, has a resource that most domains lack: fifty years of mathematical models of price dynamics, from geometric Brownian motion to stochastic and path-dependent volatility (Black & Scholes, 1973; Heston, 1993; Guyon & Lekeufack, 2023). These models are misspecified but informative; the realized trajectory comes from the true law, but is a single sample of it. We combine the two in one pipeline: pre-train on the models, fine-tune on the trajectory (Figure 1). The premise is that the pre-trained network already encodes the stylized facts shared by these models, so that fine-tuning only has to correct their misspecification. The models can be simulated at will to

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

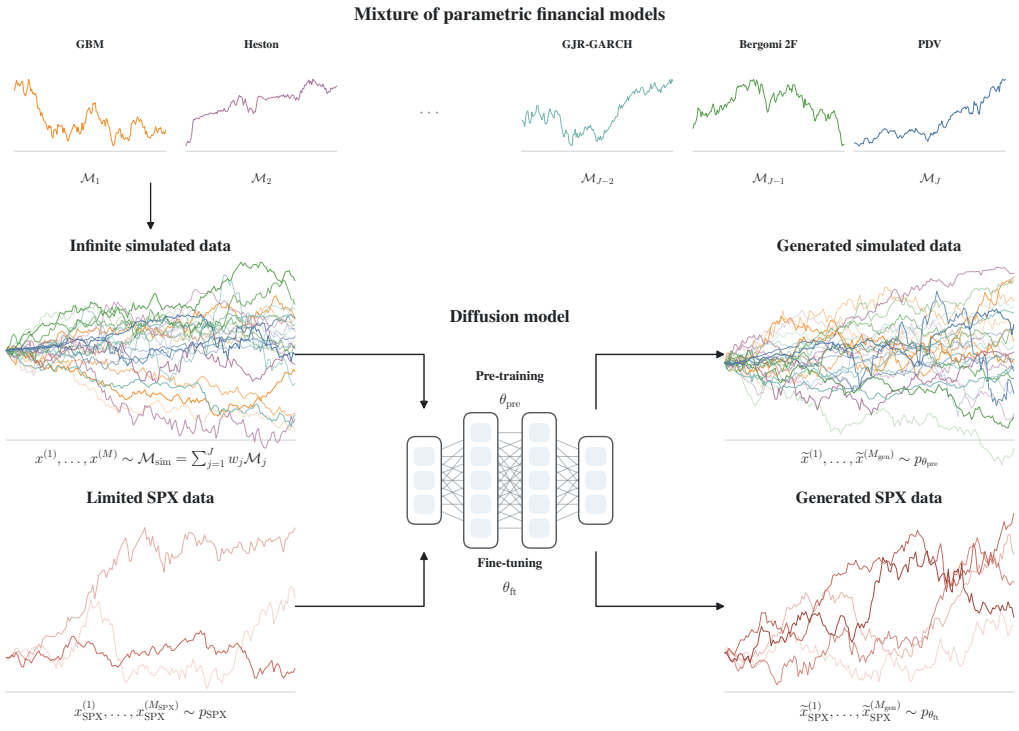

Figure 1: Pipeline of *model-based pre-training*: a mixture of parametric models provides unlimited trajectories to pre-train a diffusion model, which is then fine-tuned on the few available SPX windows.

provide unlimited training sets, while their known statistical properties enable precise evaluation of generated trajectories. More precisely, we split the problem into two stages.

(1) *Pre-training on parametric models*: the score network is trained on synthetic trajectories from models of increasing complexity; data are unlimited and, since the ground truth is known, one can measure what the network learns, whether it memorizes, and how much data it needs.

(2) *Fine-tuning on the real trajectory*: the pre-trained network is adapted to the single available trajectory (e.g. SPX), for which none of these diagnostics exist. We first do this on synthetic data simulated from a path-dependent volatility (PDV) model calibrated on SPX and held out from pre-training, where the diagnostics of stage (1) still apply, and then on SPX itself, comparing several pre-training sources.

In stage (1), we compare two formulations: path-space diffusion (U-Net, DiT), which learns the joint law of a window in $\mathbb{R}^{N \times d}$, and an autoregressive model, which factorizes it into conditional laws.

The evaluation problem. Images, video and text are judged at a glance: a coherent sample convinces a human that the generator works. Financial returns admit no such test. By eye, a good generated path, a poor one and a copy of a training window look alike; the true law is unknown, and stylized facts are only partial checks. Without this feedback, one cannot compare architectures, detect memorization or tell whether pre-training helps. We therefore first run the whole pipeline on a PDV model calibrated on SPX and held out from pre-training: like SPX, it is observed through a single trajectory, but its law is known. Only then do we move to SPX.

Related work. Time-series foundation models are trained once on a large collection of time series coming from many domains, and are then applied to new time series with little or no further training (Rasul et al., 2023; Ansari et al., 2024; Woo et al., 2024; Das et al., 2024). Some of them complement or replace the real corpus with synthetic time series drawn from generic priors, such as trend and seasonality models (Ansari et al., 2024; Dooley et al., 2023). However, these priors carry no financial structure, and which part of the prior actually transfers is rarely examined. To our knowledge,

parametric financial models have never been used to enlarge the dataset needed to train efficient diffusion models.

Main results. The two formulations differ mainly in how much data they need. To reach a given parameter discrepancy, the autoregressive model uses four times fewer training trajectories than the U-Net and twice fewer than the diffusion transformer. Beyond $20\,000$ trajectories, the three architectures perform the same. With a hundred training trajectories, training long enough drives the memorization fraction to one, and the generated paths are then copies of the training set. In the controlled PDV experiment, we fine-tune on up to 256 windows of a single trajectory; for most budgets, pre-training lowers the discrepancy compared with training from scratch. Pre-training then reduces the discrepancy by an order of magnitude compared with a non-financial prior, and what the source models contain matters more than how many of them are used: a mixture of four stochastic and path-dependent volatility models beat the full six-family mixture. On historical SPX windows, the pre-trained models keep volatility clustering, negative skewness, excess kurtosis and a negative leverage effect. Trained from scratch, they produce returns close to uncorrelated Gaussian noise.

2 Diffusion Models for Time Series

2.1 Diffusion model in path space $\mathbb{R}^{N\times d}$

Forward noising process. Let $t_n = n\Delta t$ with $n = 1, \dots, N$ denote the observation grid. Let ρ_0 be the law of a time series $\mathbf{X}_0 = (X_{t_1}, \dots, X_{t_N}) \in \mathbb{R}^{N\times d}$, known only through M independent realizations $\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(M)}$. Let $\tau \mapsto \mu_\tau$ and $\tau \mapsto w_\tau$ be two regular functions. Consider the time-inhomogeneous Ornstein–Uhlenbeck process:

$$d\mathbf{X}_\tau = -\mu_\tau \mathbf{X}_\tau \, d\tau + \sqrt{2w_\tau^2}\, d\mathbf{W}_\tau. \tag{1}$$

Time reversal. Let ρ_τ denote the probability density of $\mathbf{X}_\tau$. This process can be reversed in time (Anderson, 1982): initialize $\mathbf{Y}_0 \sim \rho_{\tau_{\max}}$ and define:

$$d\mathbf{Y}_\tau = \left[\mu_{\tau_{\max}-\tau}\mathbf{Y}_\tau + 2w^2_{\tau_{\max}-\tau}\nabla \log \rho_{\tau_{\max}-\tau}(\mathbf{Y}_\tau)\right] d\tau + \sqrt{2w^2_{\tau_{\max}-\tau}}\, d\mathbf{B}_\tau. \tag{2}$$

Then $(\mathbf{Y}_\tau)_{\tau\in[0,\tau_{\max}]}$ and $(\mathbf{X}_{\tau_{\max}-\tau})_{\tau\in[0,\tau_{\max}]}$ have the same law; in particular $\mathbf{Y}_{\tau_{\max}} \sim \rho_0$.

Representation of the marginals. Let $\alpha_\tau = e^{-\int_0^\tau \mu_s\, ds}$ and $\sigma_\tau^2 = 2\int_0^\tau e^{-2\int_s^\tau \mu_u\, du} w_s^2\, ds$. Since (1) is linear, $\mathbf{X}_\tau \mid \mathbf{X}_0 \sim \mathcal{N}\big(\alpha_\tau \mathbf{X}_0, \sigma_\tau^2 I_{Nd}\big)$, that is, for each fixed τ, if

$$\widetilde{\mathbf{X}}_\tau = \alpha_\tau \mathbf{X}_0 + \sigma_\tau \boldsymbol{\varepsilon}, \qquad \boldsymbol{\varepsilon} \sim \mathcal{N}(0, I_{Nd}), \qquad \boldsymbol{\varepsilon} \perp \mathbf{X}_0, \tag{3}$$

then $\widetilde{\mathbf{X}}_\tau$ has the same law as $\mathbf{X}_\tau$. The fact that the forward SDE marginals have an exact representation is essential for *score learning* which relies on the ability to sample from ρ_τ, which can be done without bias (and very rapidly) here using (3).

Score learning. The process (2) can be simulated using various SDE solvers, the simplest being the Euler–Maruyama scheme. However, this requires access to the score functions $(\nabla \log \rho_\tau)_{\tau\in[\tau_{\min},\tau_{\max}]}$, which are unknown. The heart of the procedure is that these score functions can be estimated efficiently thanks to two key facts.

First, (3) provides samples from ρ_τ at no cost: $\mathbf{x}_\tau^{(m)} = \alpha_\tau \mathbf{x}^{(m)} + \sigma_\tau \boldsymbol{\varepsilon}^{(m)}$ with $\boldsymbol{\varepsilon}^{(m)} \sim \mathcal{N}(0, I_{Nd})$. Second, the score of a density from which one can sample can be estimated by *score matching*.

Tweedie's formula for denoising. Denoising score matching (Hyvärinen, 2005; Vincent, 2011) rests on *Tweedie's formula*, which expresses the score as a conditional expectation. With the notation of (3), for all $\mathbf{y} \in \mathbb{R}^{N\times d}$,

$$\nabla \log \rho_\tau(\mathbf{y}) = -\,\mathbb{E}_{\substack{\mathbf{X}_0\sim\rho_0 \\ \boldsymbol{\varepsilon}\sim\mathcal{N}(0,I_{Nd})}}\left[\frac{\boldsymbol{\varepsilon}}{\sigma_\tau}\,\middle|\,\mathbf{X}_\tau = \mathbf{y}\right]. \tag{4}$$

Equivalently, "noise prediction" estimates $\mathbb{E}[\boldsymbol{\varepsilon} \mid \mathbf{X}_\tau] = -\sigma_\tau \nabla \log \rho_\tau(\mathbf{X}_\tau)$; and "data prediction" estimates $\mathbb{E}[\alpha_\tau \mathbf{X}_0 \mid \mathbf{X}_\tau] = \mathbf{X}_\tau + \sigma_\tau^2 \nabla \log \rho_\tau(\mathbf{X}_\tau)$.

Population loss. Since a conditional expectation is an L^2 projection, (4) characterizes $\nabla \log \rho_\tau$ as the minimizer of $f \mapsto \mathbb{E}\big[|f(\mathbf{X}_\tau) + \boldsymbol{\varepsilon}/\sigma_\tau|^2\big]$. Applying Tweedie's formula to all $\tau \in [\tau_{\min}, \tau_{\max}]$, we can learn all score functions $(\nabla \log \rho_\tau)_{\tau \in [\tau_{\min}, \tau_{\max}]}$ by integrating over τ. In the data prediction model, one therefore chooses a sampling law ν on $[\tau_{\min}, \tau_{\max}]$ and minimizes

$$\mathcal{L}(\theta) = \mathbb{E}_{\substack{\tau \sim \nu \\ \mathbf{X}_0 \sim \rho_0 \\ \boldsymbol{\varepsilon} \sim \mathcal{N}(0, I_{Nd})}} \left[\omega(\tau) \, |s_\theta(\tau, \alpha_\tau \mathbf{X}_0 + \sigma_\tau \boldsymbol{\varepsilon}) - \alpha_\tau \mathbf{X}_0|^2 \right]. \tag{5}$$

The law ν of the noise levels and the weight ω are design choices of the stochastic optimization, not part of the diffusion; we take $\nu = \mathcal{U}[\tau_{\min}, \tau_{\max}]$ and $\omega(\tau) = 1$.

Empirical loss. $\mathcal{L}(\theta)$ is intractable: ρ_0 is only accessible through the dataset $\mathcal{D} = \{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(M)}\}$, and the expectation over $(\tau, \boldsymbol{\varepsilon})$ has no closed form. It is replaced by a Monte-Carlo estimator built on minibatches of size $B \ll M$. This amounts to replacing ρ_0 by the empirical measure $\hat{\rho}_0 = \frac{1}{M} \sum_{m=1}^{M} \delta_{\mathbf{x}^{(m)}}$; the score is then estimated by $\nabla \log \rho_\tau(\mathbf{x}) \approx (s_\theta(\tau, \mathbf{x}) - \mathbf{x})/\sigma_\tau^2$. The exact minimizer of this empirical loss is the score of a Gaussian mixture centered on the training trajectories, i.e. a model that memorizes the dataset; useful generalization comes from the inductive bias of s_θ and from early stopping (Kadkhodaie et al., 2024). The parameters are updated using AdamW.

Sampling. Initialize $\mathbf{Y}_0 \sim \rho_{\tau_{\max}} \approx \mathcal{N}(0, \sigma_{\tau_{\max}}^2 I_{Nd})$ and solve (2) numerically, replacing the unknown score by its estimate. The terminal value $\widehat{\mathbf{X}} = \mathbf{Y}_{\tau_{\max}} \in \mathbb{R}^{N \times d}$ is a new sample.

Score network architectures. The network is conditioned on τ through a sinusoidal encoding of τ followed by a small MLP (Vaswani et al., 2017). Since the empirical loss alone leads to memorization, the structure of s_θ, its inductive bias, is what determines what is learned from limited data. The U-Net, a multi-scale encoder-decoder architecture initially introduced for biomedical image segmentation (Ronneberger et al., 2015), later became a standard backbone for diffusion models (Ho et al., 2020). More recently, Diffusion Transformers replaced U-Nets with a transformer operating on latent patches and demonstrated strong scaling properties on image-generation benchmarks (Peebles & Xie, 2023). However, there is no clear consensus regarding the "best" architecture: their relative performance depends on the data distribution, sample size, dimensionality, and computational budget. The U-Net and DiT architectures are illustrated in Figures 6 and 7.

2.2 Autoregressive Diffusion Model in $\mathbb{R}^d$

Autoregressive factorization. We factorize ρ_0 into conditional distributions on $\mathbb{R}^d$. For $n = 1, \dots, N$, let $p_{t_n}(\cdot \mid x_{t_1}, \dots, x_{t_{n-1}})$ denote the conditional density of X_{t_n} given the past observations, with p_{t_1} denoting the density of X_{t_1}. The chain rule gives

$$\rho_0(x_{t_1}, \dots, x_{t_N}) = p_{t_1}(x_{t_1}) \prod_{n=2}^{N} p_{t_n}(x_{t_n} \mid x_{t_1}, \dots, x_{t_{n-1}}). \tag{6}$$

Thus, the joint path law can be learned through N conditional laws in $\mathbb{R}^d$. If all conditional distributions are learned exactly, sequential sampling from them recovers the joint law ρ_0.

Conditional diffusion and score. For a fixed physical time t_n, we corrupt only the target observation using (3): $X_{\tau,t_n} = \alpha_\tau X_{t_n} + \sigma_\tau \varepsilon_{t_n}$, $\varepsilon_{t_n} \sim \mathcal{N}(0, I_d)$ and $\varepsilon_{t_n} \perp \mathbf{X}_0$. Let $p_{\tau,t_n}(\cdot \mid x_{t_1}, \dots, x_{t_{n-1}})$ denote the conditional density of X_{τ,t_n}. Its score $\nabla_x \log p_{\tau,t_n}(x \mid x_{t_1}, \dots, x_{t_{n-1}})$ is given by the conditional form of Tweedie's formula (4), the expectation being taken conditionally on $(X_{t_1}, \dots, X_{t_{n-1}}) = (x_{t_1}, \dots, x_{t_{n-1}})$. Again, we will estimate this quantity with a neural network with shared parameters across $n \in \{1, \dots, N\}$: $s_\theta(\tau, x, x_{t_1}, \dots, x_{t_{n-1}}) \in \mathbb{R}^d$.

Representing the past. The conditional score depends on the history $(x_{t_1}, \dots, x_{t_{n-1}}) \in \mathbb{R}^{(n-1) \times d}$, whose dimension grows with n. To address this, there are two possibilities:

(i) *Consider that the underlying process is k-Markov (Cao et al., 2026).* Set $h_0 = \varnothing$ and, for $n \geq 2$, define $h_{n-1} = (x_{t_{\max(1,n-k)}}, \dots, x_{t_{n-1}})$.

(ii) *Train an RNN to summarize the past (Rasul et al., 2021).* Without any assumption, learn a summary of the past with an RNN: initialize $h_0^\phi = 0$ and define $h_n^\phi = R_\phi(h_{n-1}^\phi, X_{t_n})$. In our experiments we use the second option, denoted AR(RNN).

Table 1: Score-network architectures.

Architecture	Backbone	Parameters
U-Net (Ronneberger et al., 2015)	Multi-scale 1D encoder–decoder.	3.8M
DiT (Peebles & Xie, 2023)	Transformer over temporal patches	9.8M
AR(RNN) (Rasul et al., 2021)	GRU history encoder, residual-MLP denoiser	2.5M

Population loss. Let H_{n-1}^{ϕ} denote either the k-lag history (therefore with no additional parameters ϕ) or the RNN representation of $(X_{t_1}, \dots, X_{t_{n-1}})$. For data prediction, we minimize

$$\mathcal{L}_{\mathrm{AR}}(\theta, \phi) = \frac{1}{N} \sum_{n=1}^{N} \mathbb{E}_{\substack{\mathbf{x}_0 \sim \rho_0 \\ \tau \sim \nu \\ \varepsilon_{t_n} \sim \mathcal{N}(0, I_d)}} \left[\omega(\tau) \left| s_\theta\Big(\tau, \alpha_\tau X_{t_n} + \sigma_\tau \varepsilon_{t_n}, H_{n-1}^{\phi}\Big) - \alpha_\tau X_{t_n} \right|^2 \right]. \quad (7)$$

By the conditional Tweedie formula, its population minimizer equals $\mathbb{E}[\alpha_\tau X_{t_n} \mid X_{\tau,t_n}, H_{n-1}^{\phi}]$. Its empirical version is built as in Section 2.1, all N conditionals being trained at once from each path, with h_{n-1}^{ϕ} computed from the *observed* history, whereas sampling relies on generated histories.

Sequential sampling. Unlike the path-space approach, the autoregressive approach solves one conditional reverse-time SDE per physical time step. Starting from an empty context, at each $n = 1, \dots, N$ we encode the generated past $(\widehat{X}_{t_1}, \dots, \widehat{X}_{t_{n-1}})$ into $\widehat{h}_{n-1}$. We initialize $Y_{0,t_n} \sim \mathcal{N}(0, \sigma^2_{\tau_{\max}} I_d)$, and solve the reverse-time SDE (2) in $\mathbb{R}^d$ with the score estimate $\big(s_\theta(\bar{\tau}, Y_{\tau,t_n}, \widehat{h}_{n-1}) - Y_{\tau,t_n}\big)/\sigma^2_{\bar{\tau}}$, $\bar{\tau} = \tau_{\max} - \tau$. We then set $\widehat{X}_{t_n} = Y_{\tau_{\max},t_n}$, update the context, and repeat. Thus, generating a trajectory requires solving N reverse-time SDEs, each in dimension d.

Autoregressive score architecture. Since the network has to learn a conditional score on $\mathbb{R}^d$, a simpler architecture suffices. We concatenate the three inputs and process them by residual MLP blocks shared across n. The full architecture is illustrated in Figure 8.

3 Pre-training Framework for Financial Time Series

We transpose the *pre-training* and *fine-tuning* paradigm of large language models to diffusion models for financial time series, introducing a benchmark of six parametric models of increasing complexity, summarized in Table 2, illustrated in Figure 2 and defined in Appendix C.

Notations. We write $\mathbf{S} = (S_{t_0}, \dots, S_{t_N})$, with $S_{t_n} > 0$, for a price trajectory, $\mathbf{r} = (r_1, \dots, r_N)$ for its log-returns $r_n = \log(S_{t_n}/S_{t_{n-1}})$, and $\mathbf{z} = (z_1, \dots, z_N)$ for their z-score $z_n = (r_n - \widehat{\mu}_{\mathrm{train}})/\widehat{\sigma}_{\mathrm{train}}$, input of the diffusion model, where $\widehat{\mu}_{\mathrm{train}}$ and $\widehat{\sigma}_{\mathrm{train}}$ are estimated once on the training returns and held fixed for validation, generation and evaluation. A generated $\widehat{\mathbf{z}}$ is mapped back to $\widehat{\mathbf{r}}$ by $\widehat{r}_n = \widehat{\mu}_{\mathrm{train}} + \widehat{\sigma}_{\mathrm{train}} \widehat{z}_n$, and to a price path $\widehat{\mathbf{S}}$ rebased at $\widehat{S}_{t_0} = 1$ by $\widehat{S}_{t_n} = \exp(\sum_{j \le n} \widehat{r}_j)$.

3.1 Parametric models and their calibration

Four-factor Guyon–Lekeufack PDV model. Introduced by Guyon & Lekeufack (2023), the *four-factor path-dependent volatility* model defines instantaneous volatility as a deterministic function of past returns and past realized variance, with parameters $\eta = (\beta_0, \beta_1, \beta_2, \lambda_{10}, \lambda_{11}, \lambda_{20}, \lambda_{21}, \vartheta_1, \vartheta_2)$:

$$\begin{gathered} dS_t = S_t \sigma_t \, dW_t, \quad \sigma_t = \beta_0 + \beta_1 R_t^{(1)} + \beta_2 \sqrt{R_t^{(2)}}, \quad R_t^{(i)} = (1 - \vartheta_i) R_t^{(i,0)} + \vartheta_i R_t^{(i,1)}, \\ dR_t^{(1,j)} = \lambda_{1j} \left(\sigma_t \, dW_t - R_t^{(1,j)} \, dt \right), \qquad dR_t^{(2,j)} = \lambda_{2j} \left(\sigma_t^2 - R_t^{(2,j)} \right) dt. \end{gathered} \quad (8)$$

With $i \in \{1, 2\}$ indexes the return and variance factors and $j \in \{0, 1\}$ their two time scales. Here β_0, β_1 and β_2 control baseline volatility, return feedback and variance feedback, while λ_{ij} and ϑ_i set the memory scales and their weights. With equity-calibrated parameters and heavy-tailed innovations, PDV reproduces the main stylized facts of asset returns (Cont, 2001) introduced in Section 3.2.

Table 2: Parametric benchmark, ordered by increasing volatility complexity.

Model	**Markov state**	**Volatility**
$\mathcal{M}_1$: GBM (Black & Scholes, 1973)	S_t	Constant
$\mathcal{M}_2$: GJR-GARCH (Glosten et al., 1993)	$(r_{t_n}, \sigma^2_{t_n})$	Path-dependent, asymmetric, heavy-tailed
$\mathcal{M}_3$: PDV-4F (Guyon & Lekeufack, 2023)	$(S_t, R_t^{(i,j)})$	Path-dependent, multi-scale feedback
$\mathcal{M}_4$: Heston (Heston, 1993)	(S_t, V_t)	Stochastic, one mean-reverting factor
$\mathcal{M}_5$: Bergomi-2F (Bergomi, 2005)	(S_t, U_t^1, U_t^2)	Stochastic, two factors, multi-scale
$\mathcal{M}_6$: Bates (Bates, 1996)	(S_t, V_t)	Stochastic, with jumps in the price

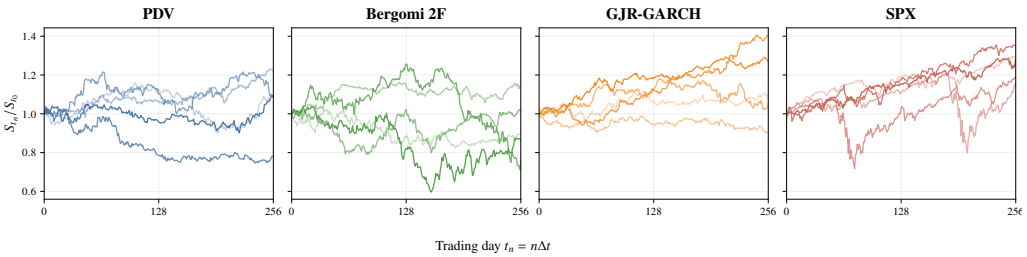

Figure 2: Five price trajectories of $N = 256$ daily log-returns, rebased to one, for three of the six families calibrated on SPX over 1970–2018 and for SPX windows from 2019–2021.

Calibration with negative log-likelihood (NLL). The families are ranked by their negative log-likelihood per return. For a model $(\mathcal{M}, \eta)$ and a trajectory $\mathbf{r} = (r_1, \ldots, r_N)$, we report

$$\mathrm{NLL}(\eta) = -\frac{1}{MN} \sum_{m=1}^{M} \sum_{n=1}^{N} \log p_\eta\big(r_n^{(m)} \mid \mathcal{F}_{n-1}^{(m)}\big), \qquad \mathcal{F}_n^{(m)} = \sigma\big(r_1^{(m)}, \ldots, r_n^{(m)}\big). \tag{9}$$

Lower is better: at the population level it is a cross-entropy, so (9) ranks families by Kullback–Leibler divergence to the data-generating law. The conditional density is explicit for the observation-driven families, whose state is a deterministic function of past returns, and requires integrating out a latent variance for Heston, Bergomi-2F and Bates, which we handle with a bootstrap particle filter (Gordon et al., 1993; Doucet & Johansen, 2011). See Table 3 for results and Appendix D for more details.

3.2 Evaluation metrics

Fidelity (D_{MLE}). We measure fidelity by estimating parameters η trajectory by trajectory, then comparing their distribution on test-set and gen-set. The estimate $\widehat{\eta}(\mathbf{r}) \in \mathbb{R}^p$ minimizes (9). We obtain two distributions per coordinate and compute their Wasserstein-1 distance:

$$D_{\mathrm{MLE}} = \frac{1}{|\mathcal{J}|} \sum_{j \in \mathcal{J}} \frac{W_1\big(\widehat{\eta}_j^{\mathrm{test}}, \widehat{\eta}_j^{\mathrm{gen}}\big)}{\widehat{\sigma}_j^{\mathrm{test}}}, \qquad \mathcal{J} = \big\{j : \widehat{\sigma}_j^{\mathrm{test}} > 0\big\}.$$

The normalization makes coordinates comparable. $D_{\mathrm{MLE}} = 0$ for a generator that reproduces the target, but also for one that memorizes the train-set, hence the need for a diversity metric.

Diversity ($\widehat{f}_{\mathrm{mem}}$). Let $d_1(\mathbf{z}) \le d_2(\mathbf{z})$ be the Euclidean distances from a standardized trajectory $\mathbf{z}$ to its two nearest train-set neighbours. Following Bonnaire et al. (2025), we call $\mathbf{z}$ memorized when $d_1/d_2 < 1/3$, and report the fraction of generated trajectories that are:

$$\widehat{f}_{\mathrm{mem}} = \frac{1}{M_{\mathrm{gen}}} \sum_{m=1}^{M_{\mathrm{gen}}} \mathbf{1}\left\{\frac{d_1\big(\widehat{\mathbf{z}}^{(m)}\big)}{d_2\big(\widehat{\mathbf{z}}^{(m)}\big)} < \frac{1}{3}\right\}.$$

Using the ratio rather than d_1 alone separates a copy of one training path from a draw in a region where the train-set happens to be dense.

Table 3: Negative log-likelihood per return on the 2019–2021 SPX validation period. Lower is better, best in bold. For latent-state models, ± is the spread across five particle seeds.

	PDV-4F	GJR-GARCH	Bergomi-2F	Bates	Heston	GBM
NLL ↓	**−3.165**	−3.144	−3.134±0.001	−3.121±0.002	−3.082±0.002	−2.423

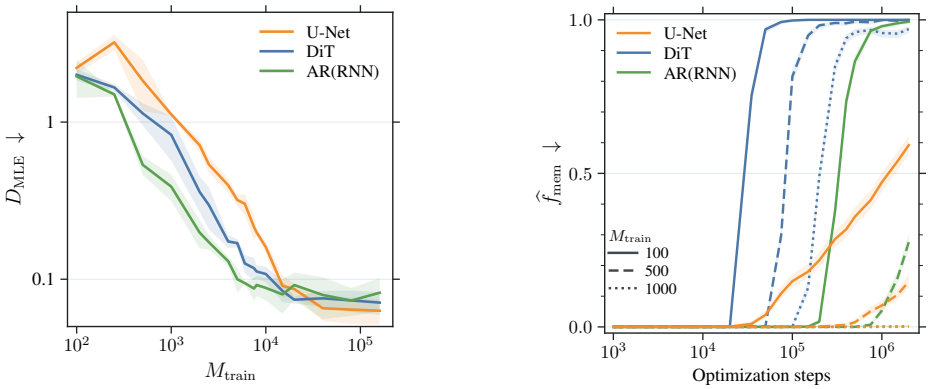

Figure 3: Fidelity and diversity on PDV-4F. Left: D_{MLE} against M_{train}. Right: $\widehat{f}_{\mathrm{mem}}$ against optimization steps. Bands are standard deviations across seeds.

Stylized facts. Market returns satisfy empirical *stylized facts* (Cont, 2001), which are ultimately what we want a generator of SPX to reproduce. We compare the main stylized facts between the test-set and the gen-set: volatility clustering, the leverage effect, and the skewness and excess kurtosis of returns aggregated over h days. Each stylized fact is computed trajectory by trajectory and averaged within a dataset. Let $r_n^{(h)} = \sum_{q=0}^{h-1} r_{n+q}$ and $m_q^{(h)}$ the q-th centered moment of $\{r_n^{(h)}\}_n$,

$$C_{|r|}(h) = \mathrm{Corr}_n\left(|r_n|, |r_{n+h}|\right), \qquad L(h) = \mathrm{Corr}_n\left(r_n, r_{n+h}^2\right),$$

$$S(h) = \frac{m_3^{(h)}}{\left(m_2^{(h)}\right)^{3/2}}, \qquad K(h) = \frac{m_4^{(h)}}{\left(m_2^{(h)}\right)^2} - 3.$$

4 Results

All experiments use the three score networks of Table 1, trained and evaluated under the same protocol. Section 4.1 is the *pre-training* part, on parametric models and unlimited data; Section 4.2 the *fine-tuning* part, on a PDV-4F target calibrated on SPX and held out from pre-training; Section 4.3 the transfer to historical SPX and an application to forecasting.

4.1 Pre-training: Learning financial dynamics from synthetic data

Sample efficiency. Figure 3 shows that architectures separate in the low-data regime: reaching $D_{\mathrm{MLE}} \leq 0.15$ requires 4 000 trajectories for AR(RNN), 6 000 for DiT and 15 000 for U-Net. Beyond $2 \cdot 10^4$ the curves meet, and past $4 \cdot 10^4$ they stop improving. More training is no substitute. At $M_{\mathrm{train}} = 100$ all three architectures memorize after $2 \cdot 10^6$ updates, $\widehat{f}_{\mathrm{mem}}$ reaching 1.00 for DiT, 0.99 for AR(RNN) and 0.59 for U-Net. Our checkpoints never reach that regime: $U^\star$, the update count at the lowest validation loss, is at least three times smaller than the update at which $\widehat{f}_{\mathrm{mem}}$ leaves zero, in every configuration where both are measured. Fidelity and diversity must be read jointly.

4.2 Fine-tuning: Adapting pre-trained models to scarce target data

We pre-train on $M_{\mathrm{train}} = 60\,000$ trajectories drawn in equal proportion from the five non-PDV families, then fine-tune on $M_{\mathrm{ft}} \in \{16, 32, 64, 128, 256\}$ windows of a single PDV trajectory and

Table 4: Pre-training source ablation for AR(RNN), fine-tuned on $M_{\text{ft}} = 256$ PDV windows. Lower is better, best in bold.

Metric	i.i.d. Gaussian	GBM	GJR-GARCH	Full Mixture	Bates	Heston	Bergomi-2F	Heston, Bergomi-2F, Bates, GJR-GARCH Mixture	PDV (oracle)
D_{MLE}	2.380	1.026	0.385	0.366	0.341	0.318	0.264	0.220	**0.152**

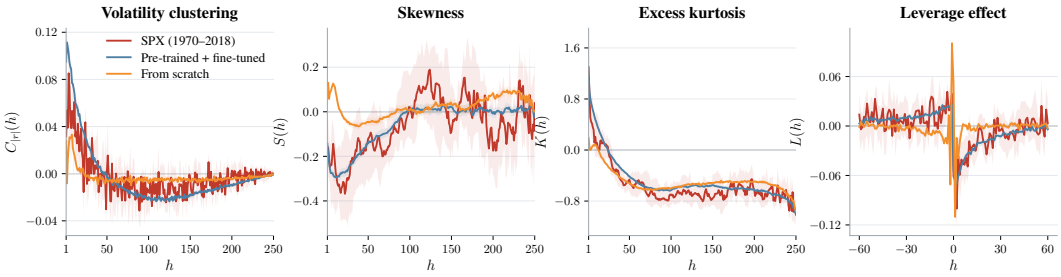

Figure 4: Stylized facts on 256-day windows: SPX 1970–2018 against AR(RNN) fine-tuned on $M_{\text{ft}} = 256$ of its windows, pre-trained or from scratch (3 seeds, 3 000 paths; medians across paths).

compare with training from scratch. We use AR(RNN), being the more data-efficient of the three. We then vary the pre-training source, and finally move to historical SPX. Fine-tuning hyperparameters are selected on validation data; details are in Appendix E.

Controlled fine-tuning on PDV. The target is a single long PDV trajectory, simulated independently of the pre-training data, then split into training, validation and test sets. The fine-tuning sets are nested families of overlapping windows of $N = 256$ daily log-returns extracted with a stride of 21 days, so that increasing M_{ft} only adds earlier windows. Metrics are computed on 5 000 independent trajectories and all results are averaged over three training seeds.

Pre-training improves low-data adaptation. Synthetic pre-training reduces D_{MLE} relative to training from scratch across most target-data budgets. In Table 4, i.i.d. Gaussian and GBM leaves D_{MLE} above 1, an eleven-fold gap to the 0.220 of the four-volatility mixture, which also beats the full six-family one at 0.366: what the sources contain matters more than how many there are. We therefore carry the five non-PDV mixture and the SPX-calibrated PDV source to SPX.

4.3 Application to SPX forecasting

Adaptation to historical SPX. We pre-train AR(RNN) on the five non-PDV families and fine-tune it on $M_{\text{ft}} = 256$ windows of SPX 1970–2018, against the same architecture trained from scratch on the same windows. Figure 4 compares their stylized facts with the observed ones. Pre-training therefore transfers to historical data, and not only between parametric families.

Adaptation to forecasting. We can apply directly the autoregressive model to forecasting. Fix a forecast origin $n_0 \in \{1, \dots, N-1\}$. Dividing (6) by the marginal of the first n_0 observations leaves

$$\rho_0\big(x_{t_{n_0+1}}, \dots, x_{t_N} \mid x_{t_1}, \dots, x_{t_{n_0}}\big) = \prod_{n=n_0+1}^{N} p_{t_n}\big(x_{t_n} \mid x_{t_1}, \dots, x_{t_{n-1}}\big)\,, \tag{10}$$

so the target law is a product of conditionals the model already learned: forecasting requires no additional training. We only need to start the recurrent state from the given context.

Sampling the continuation. We first encode the observed prefix by iterating $h_n^\phi = R_\phi(h_{n-1}^\phi, x_{t_n})$ for $n = 1, \dots, n_0$ from $h_0^\phi = 0$, which is the recursion used at training time, so the context fed to the score network is the one it was trained on. We then proceed as in sequential sampling for $n = n_0+1, \dots, N$: we solve the reverse-time SDE (2) in $\mathbb{R}^d$ with the score estimate $\big(s_\theta(\bar\tau, Y_{\tau,t_n}, \widehat{h}_{n-1}) - Y_{\tau,t_n}\big)/\sigma_{\bar\tau}^2$, set $\widehat{X}_{t_n} = Y_{\tau_{\max},t_n}$, and update the context through $\widehat{h}_n = R_\phi(\widehat{h}_{n-1}, \widehat{X}_{t_n})$, the recursion being initialized at $\widehat{h}_{n_0} = h_{n_0}^\phi$. Only $N - n_0$ reverse-time SDEs are solved instead of N. Figure 5 applies this to SPX, with context from September 2019 until 10 January 2020.

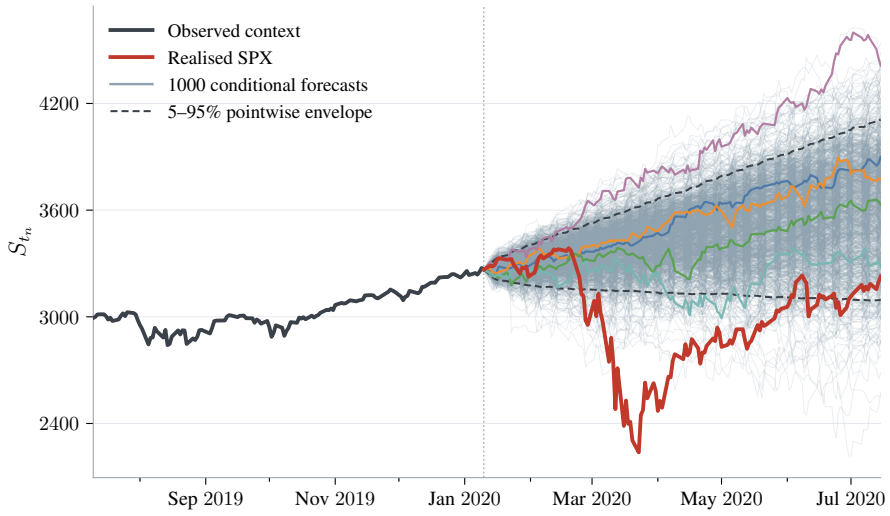

Figure 5: Forecast of the Covid year using AR(RNN). Given the observed context, the realized continuation ranks 0.9997 in volatility and 0.0010 in drawdown among 10 000 scenarios, five of which are highlighted at even quantiles of their realized volatility.

5 Perspectives

Summary. Parametric financial models provide both a controlled benchmark for diffusion-based generation and a useful source of pre-training data. Our experiments distinguish two parts: learning a synthetic distribution and adapting to a small target dataset. AR(RNN) is particularly effective in the low-data regime, while the source ablations indicate that financially relevant dynamics matter more than source diversity alone. Finally, transfer to SPX improves several stylized facts under the evaluated protocol, and the autoregressive model allows forecasting from a given context with no additional training.

Limitations. Our protocol buys its diagnostics from a deliberately narrow setting. Everything here concerns univariate daily log-returns on windows of $N = 256$ days for a single index, so nothing speaks to cross-sectional dependence, to other asset classes, or to longer horizons. The fine-tuning windows moreover overlap: extracted with a stride of 21 days from a single trajectory, two consecutive windows share 235 of their 256 points. Evaluation is narrow too. D_{MLE} sees a trajectory only through its fitted model parameters, and stylized facts are insufficient for evaluating the quality of synthetic SPX data. These limitations will be studied in future work.

Future directions. We outline several directions for future work.

(1) **More advanced pre-training.** Rather than include or exclude each family, we could weight them: let $\boldsymbol{\pi}$ lie in the simplex and pre-train on $\sum_k \pi_k \rho_0^{\mathcal{M}_k}$. Table 4 already suggests this is worth doing, since a mixture of four families beats one of six, and $\boldsymbol{\pi}$ would let us search for the right blend instead of enumerating a few. Varying it during training would give a curriculum, from the simplest dynamics to the hardest. The sources themselves raise a second question: all our parameters are fitted to SPX, so proximity to the target is built in. Fitting them instead to dispersed regimes, or to other markets, would test whether proximity really beats diversity.

(2) **Other fidelity metrics.** One idea is to train a classifier to tell the six families apart, and to compare how often it recognizes generated paths with how often it recognizes real ones. D_{MLE} only ever sees a trajectory through the parameters fitted to it, whereas a classifier would look at it through learned features, and would either confirm the metric or reveal what it misses.

(3) **Richer autoregressive model.** We never tested the k-lag history of Section 2.2, so our recurrent encoder is a default rather than a choice we measured; sweeping k would settle the question. The score could also be conditioned on information it does not currently see, such as realized volatility, interest rates, or a label for the macroeconomic regime.

AI USE STATEMENT

In this work, we used generative AI tools to provide feedback on research methodology and experimental design, to assist with the interpretation of preliminary results, to audit and edit software code, to support literature searches, and to edit the structure, readability, and formatting of the manuscript. Generative AI tools did not directly generate the reported experimental data and were not used to formulate or prove mathematical claims. All AI-assisted material included in the paper was reviewed by the authors; code-related suggestions were inspected and tested, and literature-related suggestions were checked against the cited sources. The authors take full responsibility for the final content of this work, including text, claims, code, and other artifacts produced with the aid of generative AI.

ETHICS STATEMENT

The only real data in this work are daily levels of SPX, a public index that carries no personal information. Everything else is simulated. Synthetic return generators are used to augment backtests and to calibrate risk models. A generator that has memorized its training set can make tail-risk estimates look far better supported than they are. Figure 5 is not a trading signal, and nothing in this paper is investment advice. We comply with the ICLR Code of Ethics.

REPRODUCIBILITY STATEMENT

Every synthetic dataset can be regenerated from the paper alone. The formulations are Equations (1)–(7), the six families and their discretization schemes are in Appendix C, and their SPX-calibrated η in Table 6. Table 7 reports every choice that affects training or evaluation. Results from trained models average three training seeds. The complete code will be released in a later version.

REFERENCES

B. D. O. Anderson. Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326, 1982. URL https://core.ac.uk/download/pdf/82826666.pdf.

Abdul Fatir Ansari, Lorenzo Stella, Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, Jasper Zschiegner, Danielle C. Maddix, Hao Wang, Michael W. Mahoney, Kari Torkkola, Andrew Gordon Wilson, Michael Bohlke-Schneider, and Yuyang Wang. Chronos: Learning the language of time series. *Transactions on Machine Learning Research*, 2024. URL https://arxiv.org/abs/2403.07815.

David S. Bates. Jumps and stochastic volatility: Exchange rate processes implicit in deutsche mark options. *The Review of Financial Studies*, 9(1):69–107, 1996. doi: 10.1093/rfs/9.1.69.

Lorenzo Bergomi. Smile dynamics II. *Risk*, 18(10):67–73, 2005.

Fischer Black and Myron Scholes. The pricing of options and corporate liabilities. *Journal of Political Economy*, 81(3):637–654, 1973. doi: 10.1086/260062.

Tony Bonnaire, Raphaël Urfin, Giulio Biroli, and Marc Mézard. Why Diffusion Models Don't Memorize: The Role of Implicit Dynamical Regularization in Training, 2025. URL https://arxiv.org/abs/2505.17638.

Haoyang Cao, Minshuo Chen, Yinbin Han, and Renyuan Xu. Diffusion models for adaptive sequential data generation, 2026. URL https://arxiv.org/abs/2606.06007.

Rama Cont. Empirical properties of asset returns: Stylized facts and statistical issues. *Quantitative Finance*, 1(2):223–236, 2001. doi: 10.1080/713665670. URL https://doi.org/10.1080/713665670.

Rama Cont, Mihai Cucuringu, Renyuan Xu, and Chao Zhang. Tail-GAN: Learning to simulate tail risk scenarios. *Management Science*, 72(4):2917–2936, 2026. doi: 10.1287/mnsc.2023.00936. URL https://doi.org/10.1287/mnsc.2023.00936.

Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A decoder-only foundation model for time-series forecasting. In *International Conference on Machine Learning*, 2024. URL https://arxiv.org/abs/2310.10688.

Samuel Dooley, Gurnoor Singh Khurana, Chirag Mohapatra, Siddartha Naidu, and Colin White. Forecastpfn: Synthetically-trained zero-shot forecasting. In *Advances in Neural Information Processing Systems*, 2023. URL https://arxiv.org/abs/2311.01933.

Arnaud Doucet and Adam M. Johansen. A tutorial on particle filtering and smoothing: fifteen years later. In *The Oxford Handbook of Nonlinear Filtering*, pp. 656–704. Oxford University Press, 2011.

Lawrence R. Glosten, Ravi Jagannathan, and David E. Runkle. On the relation between the expected value and the volatility of the nominal excess return on stocks. *The Journal of Finance*, 48(5): 1779–1801, 1993. doi: 10.1111/j.1540-6261.1993.tb05128.x.

Neil J. Gordon, David J. Salmond, and Adrian F. M. Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEE Proceedings F (Radar and Signal Processing)*, 140(2): 107–113, 1993.

Julien Guyon and Jordan Lekeufack. Volatility is (mostly) path-dependent. *Quantitative Finance*, 23 (9):1221–1258, 2023. doi: 10.1080/14697688.2023.2221281. URL https://doi.org/10.1080/14697688.2023.2221281.

Steven L. Heston. A closed-form solution for options with stochastic volatility with applications to bond and currency options. *The Review of Financial Studies*, 6(2):327–343, 1993. doi: 10.1093/rfs/6.2.327.

J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, 2020. URL https://papers.nips.cc/paper/2020/hash/4c5bcfec8584af0d967f1ab10179ca4b-Abstract.html.

Hongbin Huang, Minghua Chen, and Xiao Qiao. Generative learning for financial time series with irregular and scale-invariant patterns. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/f90fc76b199fe6b0ec2a51aaf72c3277-Abstract-Conference.html.

A. Hyvärinen. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6:695–709, 2005. URL https://jmlr.org/papers/volume6/hyvarinen05a/hyvarinen05a.pdf.

Z. Kadkhodaie, F. Guth, E. P. Simoncelli, and S. Mallat. Generalization in diffusion models arises from geometry-adaptive harmonic representations. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=ANvmVS2Yr0.

Marcel Kollovieh, Abdul Fatir Ansari, Michael Bohlke-Schneider, Jasper Zschiegner, Hao Wang, and Yuyang Wang. Predict, refine, synthesize: Self-guiding diffusion models for probabilistic time series forecasting. In *Advances in Neural Information Processing Systems*, volume 36, 2023. doi: 10.52202/075280-1232. URL https://papers.neurips.cc/paper_files/paper/2023/hash/5a1a10c2c2c9b9af1514687bc24b8f3d-Abstract-Conference.html.

Juan Miguel Lopez Alcaraz and Nils Strodthoff. Diffusion-based time series imputation and forecasting with structured state space models. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=hHiIbk7ApW.

Hao Ni, Lukasz Szpruch, Marc Sabate-Vidales, Baoren Xiao, Magnus Wiese, and Shujian Liao. Sig-wasserstein GANs for time series generation. In *Proceedings of the Second ACM International Conference on AI in Finance*, pp. 1–8. Association for Computing Machinery, 2021. doi: 10.1145/3490354.3494393. URL https://doi.org/10.1145/3490354.3494393.

W. Peebles and S. Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023. URL https://openaccess.thecvf.com/content/ICCV2023/html/Peebles_Scalable_Diffusion_Models_with_Transformers_ICCV_2023_paper.html.

Kashif Rasul, Calvin Seward, Ingmar Schuster, and Roland Vollgraf. Autoregressive denoising diffusion models for multivariate probabilistic time series forecasting. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 8857–8868. PMLR, 2021. URL https://proceedings.mlr.press/v139/rasul21a.html.

Kashif Rasul, Arjun Ashok, Andrew Robert Williams, Hena Ghonia, Rishika Bhagwatkar, Arian Khorasani, Mohammad Javad Darvishi Bayazi, George Adamopoulos, Roland Riachi, Nadhir Hassen, Marin Biloš, Sahil Garg, Anderson Schneider, Nicolas Chapados, Alexandre Drouin, Valentina Zantedeschi, Yuriy Nevmyvaka, and Irina Rish. Lag-llama: Towards foundation models for probabilistic time series forecasting, 2023. URL https://arxiv.org/abs/2310.08278.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, volume 9351 of *Lecture Notes in Computer Science*, pp. 234–241. Springer, 2015. doi: 10.1007/978-3-319-24574-4_28. URL https://doi.org/10.1007/978-3-319-24574-4_28.

Lifeng Shen and James Kwok. Non-autoregressive conditional diffusion models for time series prediction. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 31016–31029. PMLR, 2023. URL https://proceedings.mlr.press/v202/shen23d.html.

J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *Proceedings of the 32nd International Conference on Machine Learning*, 2015. URL https://arxiv.org/abs/1503.03585.

Y. Song and S. Ermon. Generative modeling by estimating gradients of the data distribution. In *Advances in Neural Information Processing Systems*, 2019. URL https://papers.nips.cc/paper/9361-generative-modeling-by-estimating-gradients-of-the-data-distribution.

Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=PxTIG12RRHS.

Tomonori Takahashi and Takayuki Mizuno. Generation of synthetic financial time series by diffusion models. *Quantitative Finance*, 25(10):1507–1516, 2025. doi: 10.1080/14697688.2025.2528697. URL https://doi.org/10.1080/14697688.2025.2528697.

Yusuke Tashiro, Jiaming Song, Yang Song, and Stefano Ermon. CSDI: Conditional score-based diffusion models for probabilistic time series imputation. In *Advances in Neural Information Processing Systems*, volume 34, pp. 24804–24816. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/hash/cfe8504bda37b575c70ee1a8276f3486-Abstract.html.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017. URL https://arxiv.org/pdf/1706.03762.

P. Vincent. A connection between score matching and denoising autoencoders. *Neural Computation*, 23(7):1661–1674, 2011. URL https://www.iro.umontreal.ca/~vincentp/Publications/smdae_techreport.pdf.

Magnus Wiese, Robert Knobloch, Ralf Korn, and Peter Kretschmer. Quant GANs: Deep generation of financial time series. *Quantitative Finance*, 20(9):1419–1440, 2020. doi: 10.1080/14697688.2020.1730426. URL https://doi.org/10.1080/14697688.2020.1730426.

Gerald Woo, Chenghao Liu, Akshat Kumar, Caiming Xiong, Silvio Savarese, and Doyen Sahoo. Unified training of universal time series forecasting transformers. In *International Conference on Machine Learning*, 2024. URL https://arxiv.org/abs/2402.02592.

Jinsung Yoon, Daniel Jarrett, and Mihaela van der Schaar. Time-series generative adversarial networks. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://papers.nips.cc/paper/8789-time-series-generative-adversarial-networks.

Xinyu Yuan and Yan Qiao. Diffusion-TS: Interpretable diffusion for general time series generation. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/b5b66077d016c037576cc56a82f97f66-Abstract-Conference.html.

A NOTATION

Table 5: Summary of the main notation.

Notation	Meaning
General hyperparameters	
N	Number of log-returns learned by the generative model.
M	Number of trajectories in a dataset.
M_{train}	Number of trajectories in the training set.
M_{val}	Number of trajectories in the validation set.
M_{test}	Number of trajectories in the test set.
d	Dimension of each element of the time series.
t	Continuous physical time.
Δt	Time interval between two consecutive price observations.
n	Discrete index, with $n \in \{0, \dots, N\}$ for price observations and $n \in \{1, \dots, N\}$ for log-returns.
$t_n = n\Delta t$	Observation time associated with index n.
Financial representation	
$S_{t_n} \in \mathbb{R}_+$	Random asset price at observation time t_n.
$s_{t_n}^{(m)} \in \mathbb{R}_+$	Realized price at observation time t_n in trajectory m.
$\mathbf{s}^{(m)} \in \mathbb{R}_+^{N+1}$	m-th observed price trajectory.
$\ell_{t_n} = \log S_{t_n}$	Random log-price at observation time t_n.
$\ell_{t_n}^{(m)} = \log s_{t_n}^{(m)}$	Realized log-price at observation time t_n in trajectory m.
$r_n^{(m)} = \ell_{t_n}^{(m)} - \ell_{t_{n-1}}^{(m)}$	Log-return over the interval $[t_{n-1}, t_n]$.
$\mathbf{r}^{(m)} \in \mathbb{R}^N$	Log-return trajectory obtained from the m-th price trajectory.
$\bar{r}_{\text{train}}$	Global mean computed from the log-returns in the train set.
s_{train}	Global std computed from the log-returns in the train set.
ε_{std}	Positive numerical stabilizer used in the z-score normalization.
$a_{\text{train}} = s_{\text{train}} + \varepsilon_{\text{std}}$	Stabilized normalization scale.
$z_n^{(m)} = \frac{(r_n^{(m)} - \bar{r}_{\text{train}})}{a_{\text{train}}}$	Standardized log-return at index n in trajectory m.
$\mathbf{z}^{(m)} \in \mathbb{R}^N$	m-th standardized log-return trajectory used as input to the generative model.
$\widehat{z}_n$	Generated standardized log-return.
$\widehat{r}_n = \bar{r}_{\text{train}} + a_{\text{train}}\widehat{z}_n$	Generated log-return after reversing the z-score normalization.
$\widehat{s}_{t_n} = s_{t_0} \exp\left(\sum_{j=1}^n \widehat{r}_j\right)$	Generated price reconstructed from the generated log-returns.
Diffusion process	

Notation	Meaning
$X_{t_n} \in \mathbb{R}^d$	Clean random model input at observation index n; in the financial experiments, it represents a standardized log-return.
$\mathbf{X}_0 = (X_{t_1}, \ldots, X_{t_N}) \in \mathbb{R}^{N\times d}$	Clean random trajectory, equivalently the trajectory at diffusion time $\tau = 0$.
$\mathbf{x}^{(m)} = \mathbf{z}^{(m)}$	m-th observed realization of $\mathbf{X}_0$ in the financial experiments.
$\tau_{\min}, \tau_{\max}$	Minimum and maximum diffusion times.
$\tau \in [\tau_{\min}, \tau_{\max}]$	Diffusion time, distinct from the observation times $t_0, \ldots, t_N$.
μ_τ, w_τ	Drift and diffusion-scale coefficients of the forward noising process.
α_τ	Mean-scaling coefficient of the forward marginal distribution.
σ_τ	Noise standard deviation of the forward marginal distribution.
I_{Nd}	Identity covariance matrix on the vectorized path space $\mathbb{R}^{Nd}$.
$\boldsymbol{\varepsilon} \in \mathbb{R}^{N\times d}$	Standard Gaussian noise applied to a complete trajectory.
$\varepsilon_{t_n} \in \mathbb{R}^d$	Standard Gaussian noise applied to one component of a trajectory.
$\mathbf{W}_\tau \in \mathbb{R}^{N\times d}$	Brownian motion driving the forward diffusion process.
$\mathbf{X}_\tau \in \mathbb{R}^{N\times d}$	Forward-diffusion trajectory at diffusion time τ.
$X_{\tau,t_n} \in \mathbb{R}^d$	Component of $\mathbf{X}_\tau$ associated with observation time t_n.
$\widetilde{\mathbf{X}}_\tau$	Exact Gaussian marginal representation of $\mathbf{X}_\tau$.
ρ_0	Probability density of the clean random trajectory $\mathbf{X}_0$.
ρ_τ	Probability density of the noised trajectory $\mathbf{X}_\tau$.
$\mathbf{B}_\tau \in \mathbb{R}^{N\times d}$	Brownian motion driving the reverse-time diffusion process.
$\mathbf{Y}_\tau \in \mathbb{R}^{N\times d}$	Reverse-time diffusion process.
$\widehat{\mathbf{X}}$	Generated clean trajectory obtained at the end of the reverse-time process.
Learning and autoregressive factorization	
θ	Parameters of the denoising neural network.
s_θ	Neural network used to parameterize the denoising prediction.
ν	Sampling distribution of diffusion times used during training.
$\omega(\tau)$	Loss weight associated with diffusion time τ.
p_{t_n}	Conditional density of X_{t_n} given the preceding clean observations.
p_{τ,t_n}	Conditional density of X_{τ,t_n} given the preceding clean observations.
ϕ	Parameters of the recurrent history encoder.
h_n^ϕ	RNN state after processing observations up to observation time t_n.
H_{n-1}^ϕ	Context used to predict the observation associated with time t_n.
k	Context length under the k-Markov assumption.
n_0	Forecast origin: index of the last observed point of the context.

B SCORE-NETWORK ARCHITECTURES

The three score-network architectures considered in our experiments are presented below.

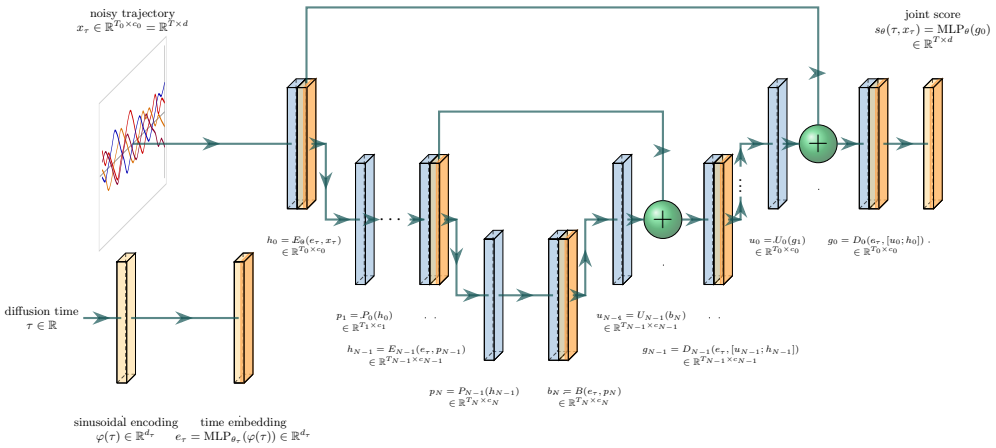

Figure 6: U-Net score network. The diffusion-time embedding e_τ conditions the encoder, bottleneck, and decoder blocks. Skip connections transmit information between corresponding resolutions.

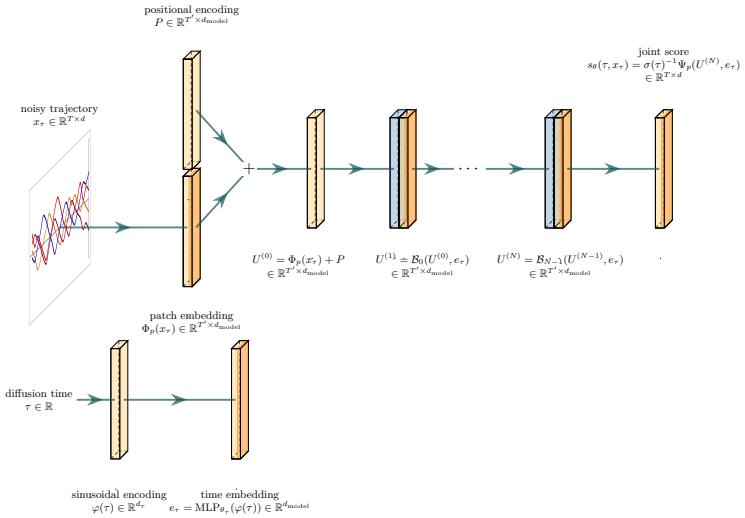

Figure 7: Diffusion Transformer score network. The noisy trajectory is divided into temporal patches and projected into tokens. Positional encodings are added before processing by diffusion-time-conditioned Transformer blocks.

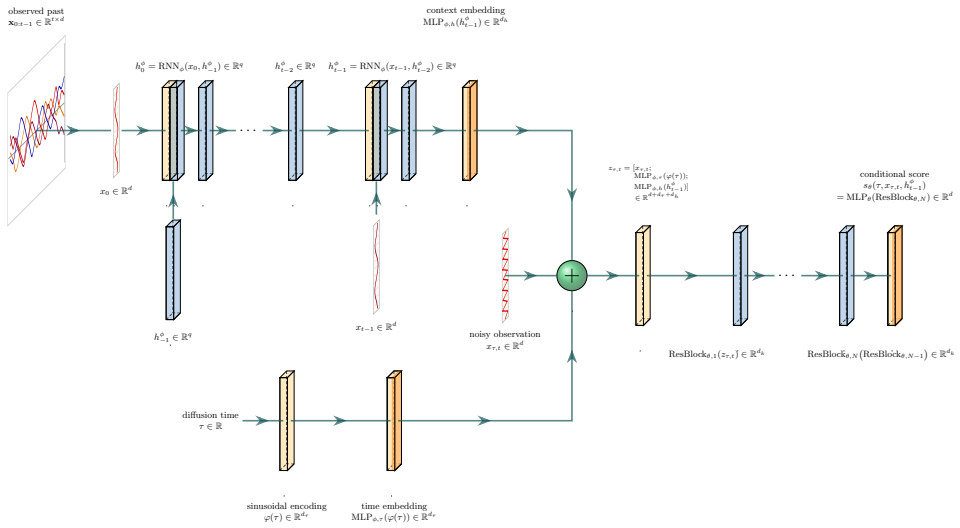

Figure 8: Autoregressive conditional score network. The noisy target X_{τ,t_n} is combined with the diffusion-time embedding and the representation h_{n-1} of the observed past to predict a conditional score in $\mathbb{R}^d$.

C More details on Parametric models

Every simulator starts at $S_{t_0} = 1$. In the figures below, the standardized log-returns are $z_n = (r_n - \bar{r}_{\text{train}})/a_{\text{train}}$, with $a_{\text{train}} = s_{\text{train}} + \varepsilon_{\text{std}}$ and the two moments estimated once on the whole train-set, as in Appendix A.

Geometric Brownian motion. The geometric Brownian motion underlying the Black–Scholes model (Black & Scholes, 1973) models the asset price under the physical measure by

$$\begin{cases} dS_t = S_t\,(\mu\,dt + \sigma\,dW_t)\,, \\ S_0 > 0, \qquad \mu \in \mathbb{R}, \qquad \sigma > 0. \end{cases}$$

Here, μ is the expected instantaneous rate of return, while σ is the constant instantaneous volatility. Applying Itô's formula to $\ell_t = \log S_t$ gives $d\ell_t = (\mu - \sigma^2/2)\,dt + \sigma\,dW_t$. We simulate the log-price exactly on the grid using Gaussian increments and reconstruct prices through $S_{t_n} = \exp(\ell_{t_n})$.

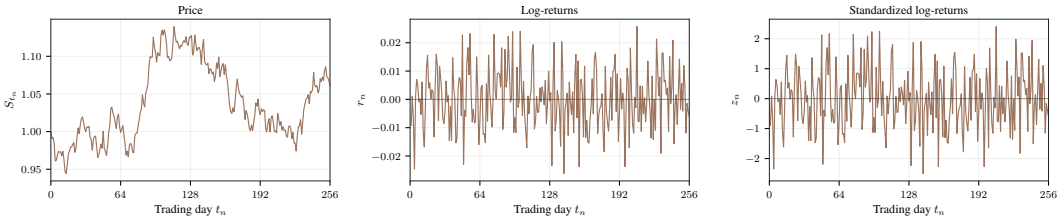

Figure 9: One trajectory of $\mathcal{M}_1$ (GBM) simulated at η_{01}, shown as a price, as log-returns, and as standardized log-returns.

GJR-GARCH model. The GJR-GARCH model (Glosten et al., 1993) is a discrete-time heteroskedastic model in which negative innovations may have a larger effect on future volatility than positive

innovations of the same magnitude. For $t_n = n\Delta t$, its dynamics are

$$\begin{cases} r_{t_n} = \mu\Delta t + \varepsilon_{t_n}, \\ \varepsilon_{t_n} = \sqrt{h_{t_n}}\, z_n, \\ h_{t_{n+1}} = \omega + \alpha\varepsilon_{t_n}^2 + \gamma\varepsilon_{t_n}^2 \mathbf{1}_{\{\varepsilon_{t_n}<0\}} + \beta h_{t_n}, \\ X_{t_n} = X_{t_{n-1}} + r_{t_n}, \\ S_{t_n} = \exp(X_{t_n}), \\ X_{t_0} = 0, \qquad h_{t_1} = \dfrac{\omega}{1-\alpha-\beta-\gamma}, \\ z_n \overset{\text{i.i.d.}}{\sim} \text{SkewStudent}(\nu, \lambda_s), \qquad \mathbb{E}[z_n] = 0, \quad \text{Var}(z_n) = 1, \end{cases} \tag{11}$$

where $\mu \in \mathbb{R}$ is the annualized mean return, $\omega > 0$ is the conditional-variance intercept, $\alpha, \beta, \gamma \geq 0$ respectively control the response to past innovations, volatility persistence, and the additional response to negative innovations. The parameters $\nu > 2$ and $\lambda_s \in (-1, 1)$ determine the tail thickness and skewness of the standardized innovations. We impose the sufficient stability condition $\alpha + \beta + \gamma < 1$. The simulator discards the first 2 000 observations as burn-in and retains the following $N = 256$ log-returns.



Figure 10: One trajectory of $\mathcal{M}_3$ (GJR-GARCH) simulated at η_{03}, shown as a price, as log-returns, and as standardized log-returns.

Guyon–Lekeufack PDV model. The PDV model of Guyon & Lekeufack (2023) specifies an asset price whose volatility is an affine function of exponentially weighted return and variance factors:

$$\begin{cases} dS_t = S_t \sigma_t\, dW_t, \\ dR_t^{(1,j)} = -\lambda_{1,j} R_t^{(1,j)}\, dt + \lambda_{1,j}\sigma_t\, dW_t, \qquad j \in \{0,1\}, \\ dR_t^{(2,j)} = -\lambda_{2,j} R_t^{(2,j)}\, dt + \lambda_{2,j}\sigma_t^2\, dt, \qquad j \in \{0,1\}, \\ R_t^{(1)} = (1-\vartheta_1) R_t^{(1,0)} + \vartheta_1 R_t^{(1,1)}, \\ R_t^{(2)} = (1-\vartheta_2) R_t^{(2,0)} + \vartheta_2 R_t^{(2,1)}, \\ \sigma_t = \beta_0 + \beta_1 R_t^{(1)} + \beta_2 \sqrt{R_t^{(2)}}, \\ S_0 > 0, \qquad \lambda_{i,j} > 0, \qquad \vartheta_i \in [0,1]. \end{cases}$$

The first factors encode signed past returns, while the second factors encode past squared returns at two time scales. The parameters $\lambda_{i,j}$ determine their decay rates, ϑ_i their relative weights, β_0 the baseline volatility, β_1 the leverage contribution, and β_2 the volatility-feedback contribution. Applying Itô's formula gives $d\ell_t = -\sigma_t^2 dt/2 + \sigma_t\, dW_t$. We use a discrete-time adaptation whose innovation is a centered three-component Gaussian mixture calibrated to approximate a Student distribution with parameter ν_{PDV}. The drift is corrected using the exact moment-generating function of the mixture so that the discretized price remains a martingale in the zero-drift specification.

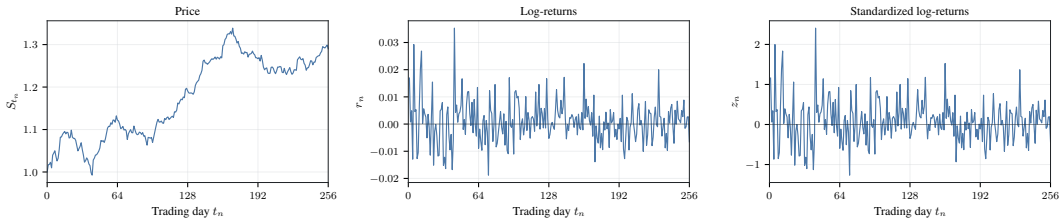

Figure 11: One trajectory of $\mathcal{M}_4$ (Guyon–Lekeufack PDV) simulated at η_{04}, shown as a price, as log-returns, and as standardized log-returns.

Heston stochastic volatility model. The Heston model (Heston, 1993) couples the asset price with a mean-reverting stochastic variance:

$$\begin{cases} dS_t = S_t\left(\mu\,dt + \sqrt{V_t}\,dW_t^S\right), \\ dV_t = \kappa(\theta - V_t)\,dt + \xi\sqrt{V_t}\,dW_t^V, \\ d\langle W^S, W^V\rangle_t = \rho\,dt, \\ S_0 > 0, \qquad V_0 \geq 0, \\ \mu \in \mathbb{R}, \qquad \kappa, \theta, \xi > 0, \qquad \rho \in [-1, 1]. \end{cases}$$

The parameters μ and θ determine the price drift and long-run variance, κ controls the variance mean-reversion speed, ξ is the volatility of variance, and ρ controls the leverage effect. Applying Itô's formula gives $d\ell_t = (\mu - V_t/2)\,dt + \sqrt{V_t}\,dW_t^S$. We simulate the variance with a full-truncation Euler scheme and the log-price with a log-Euler step using correlated Gaussian innovations.

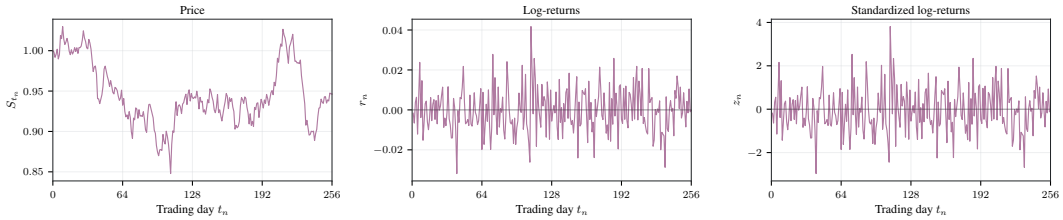

Figure 12: One trajectory of $\mathcal{M}_5$ (Heston) simulated at η_{05}, shown as a price, as log-returns, and as standardized log-returns.

Two-factor Bergomi model. The two-factor Bergomi model (Bergomi, 2005) specifies the asset price through a stochastic variance driven by two latent Ornstein–Uhlenbeck factors:

$$\begin{cases} dS_t = S_t\left(\mu\,dt + \sqrt{V_t}\,dW_t^S\right), \\ dU_t^i = -\kappa_i U_t^i\,dt + dW_t^i, \qquad i \in \{1, 2\}, \\ V_t = \xi_0 \exp\left(\eta\big(\omega U_t^1 + (1-\omega)U_t^2\big) - \dfrac{\eta^2 q_t}{2}\right), \\ q_t = \omega^2 \dfrac{1 - e^{-2\kappa_1 t}}{2\kappa_1} + (1-\omega)^2 \dfrac{1 - e^{-2\kappa_2 t}}{2\kappa_2}, \\ dW_t^F = \dfrac{\omega\,dW_t^1 + (1-\omega)\,dW_t^2}{\sqrt{\omega^2 + (1-\omega)^2}}, \\ dW_t^S = \rho\,dW_t^F + \sqrt{1-\rho^2}\,dW_t^{\perp}, \\ S_0 > 0, \qquad U_0^1 = U_0^2 = 0, \\ \mu \in \mathbb{R}, \qquad \xi_0, \eta, \kappa_1, \kappa_2 > 0, \qquad \omega \in [0, 1], \qquad \rho \in [-1, 1], \end{cases}$$

where W^1, W^2, and $W^{\perp}$ are independent Brownian motions. The parameter μ determines the price drift, ξ_0 the initial variance level, η the magnitude of volatility fluctuations, (κ_1, κ_2) the

two time scales, ω their relative weight, and ρ the leverage effect. Applying Itô's formula gives $d\ell_t = (\mu - V_t/2)\,dt + \sqrt{V_t}\,dW_t^S$. The latent factors are simulated with their exact Gaussian transitions jointly with the Brownian increments entering the log-Euler price step. The implementation caps the variance at $V_{\max} = 4$ for numerical stability. The augmented state (ℓ_t, U_t^1, U_t^2) is Markov.



Figure 13: One trajectory of $\mathcal{M}_6$ (two-factor Bergomi) simulated at η_{06}, shown as a price, as log-returns, and as standardized log-returns.

Bates jump-diffusion model. The Bates model (Bates, 1996) augments the Heston price process with compound-Poisson jumps:

$$\begin{cases} \dfrac{dS_t}{S_{t-}} = (\mu - \lambda\kappa_J)\,dt + \sqrt{V_t}\,dW_t^S + (J-1)\,d\mathcal{N}_t, \\ dV_t = \kappa(\theta - V_t)\,dt + \xi\sqrt{V_t}\,dW_t^V, \\ d\langle W^S, W^V\rangle_t = \rho\,dt, \\ \mathcal{N}_t \sim \mathrm{Poisson}(\lambda t), \\ \log J \sim \mathcal{N}(\mu_J, \sigma_J^2), \\ \kappa_J = \mathbb{E}[J-1] = \exp\left(\mu_J + \dfrac{\sigma_J^2}{2}\right) - 1, \\ S_0 > 0, \qquad V_0 \geq 0, \\ \mu, \mu_J \in \mathbb{R}, \qquad \kappa, \theta, \xi, \lambda, \sigma_J > 0, \qquad \rho \in [-1, 1]. \end{cases}$$

The parameters $(\kappa, \theta, \xi, \rho)$ govern the latent variance and leverage effect, μ determines the price drift, λ is the jump intensity, and (μ_J, σ_J) determine the log-jump distribution. The compensator $\lambda\kappa_J$ preserves the specified drift. Applying Itô's formula with jumps gives $d\ell_t = (\mu - \lambda\kappa_J - V_t/2)\,dt + \sqrt{V_t}\,dW_t^S + \log(J)\,d\mathcal{N}_t$. The implementation combines full-truncation Euler for the variance, log-Euler for the continuous price component, and exact conditional sampling of the aggregate compound-Poisson log-jump.

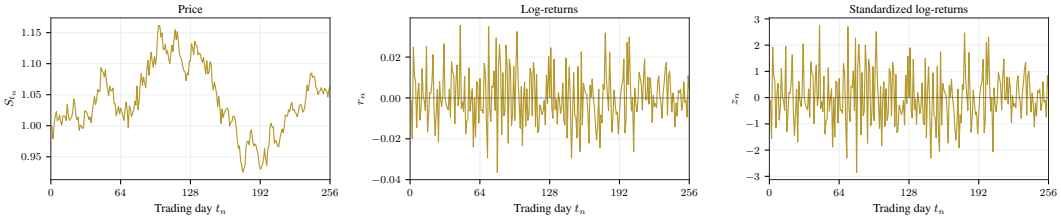

Figure 14: One trajectory of $\mathcal{M}_7$ (Bates jump-diffusion) simulated at η_{07}, shown as a price, as log-returns, and as standardized log-returns.

D More details on evaluation metrics

Notations. Each model is fitted using a *train-set* of M_{train} trajectories. During training, we use a *val-set* of M_{val} trajectories for early stopping. After training, we evaluate the model on a *test-set* of M_{test} trajectories, seen neither during training nor during selection, and on a *gen-set* of M_{gen} trajectories sampled from the trained model. *Fidelity* metrics compare the test-set with the gen-set;

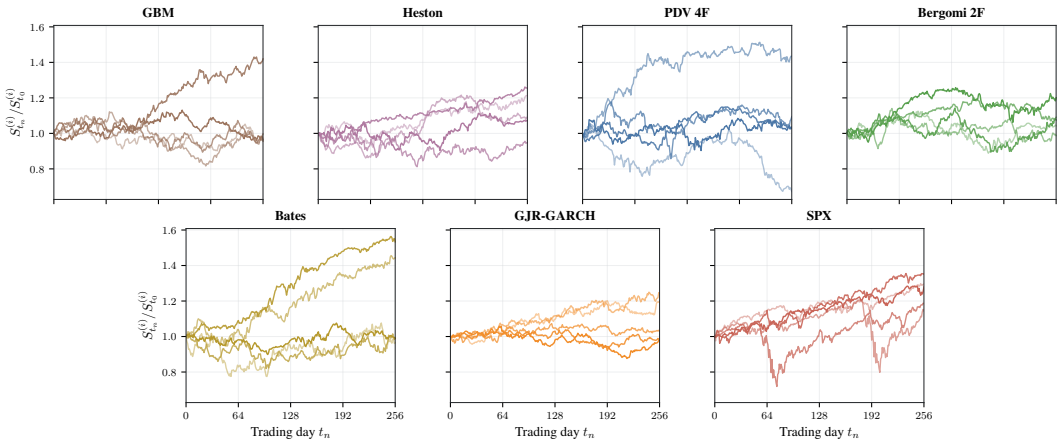

Figure 15: Sample paths of the six parametric families at their SPX-calibrated parameter vector, together with held-out SPX windows (bottom right). All panels share the same vertical scale. Each panel shows five independent paths of length $N = 256$ trading days, normalized by their initial value.

diversity (e.g. memorization) metrics compare the train-set with the gen-set. We write $\mathbf{r}$ for a raw log-return trajectory and $\mathbf{z}$ for its standardized version, all trajectories being standardized with the mean and standard deviation of the train-set so that distances remain comparable across sets.

Table 6: SPX-calibrated parameter vector of each of the six models.

Model family	Calibrated parameter vector
$\mathcal{M}_1$: GBM	$\eta_1 = (\mu,\ \sigma) = (0.083,\ 0.169)$
$\mathcal{M}_2$: GJR-GARCH	$\eta_3 = (\mu,\ \omega,\ \alpha,\ \gamma,\ \beta,\ \nu,\ \zeta) = (0.110,\ 1.492{\times}10^{-6},\ 0.044,\ 0.040,\ 0.913,\ 8.127,\ -0.051)$
$\mathcal{M}_3$: PDV-4F	$\eta_4 = (\beta_0,\ \beta_1,\ \beta_2,\ \lambda_{1,0},\ \lambda_{1,1},\ \theta_1,\ \lambda_{2,0},\ \lambda_{2,1},\ \theta_2) = (0.022,\ -0.037,\ 0.809,\ 71.538,\ 8.085,\ 0.526,\ 20.193,\ 2.447,\ 0.200)$
$\mathcal{M}_4$: Heston	$\eta_5 = (\kappa,\ \theta,\ \xi,\ \rho,\ \mu,\ v_0) = (4.114,\ 0.028,\ 0.350,\ -0.502,\ 0.100,\ 0.040)$
$\mathcal{M}_5$: Bergomi-2F	$\eta_6 = (\mu,\ \xi_0,\ \eta,\ \kappa_1,\ \kappa_2,\ \omega,\ \rho) = (0.139,\ 0.034,\ 2.892,\ 18.239,\ 2.194,\ 0.256,\ -0.494)$
$\mathcal{M}_6$: Bates	$\eta_7 = (\kappa,\ \theta,\ \xi,\ \rho,\ \mu,\ \lambda,\ m_J,\ s_J,\ v_0) = (2.732,\ 0.042,\ 0.373,\ -0.552,\ 0.059,\ 0.282,\ -0.005,\ 0.078,\ 0.040)$

E MORE EXPERIMENTAL DETAILS AND HYPERPARAMETERS

Table 7: Hyperparameters of the main experiments.

Hyperparameter	**Value**
Representation and discretization	
N	256
d	1
Δt	$1/256$
Maximum internal simulation step	$1/256$
$\varepsilon_{\mathrm{std}}$	10^{-8}
Forward diffusion	
Diffusion family	variance preserving
Noise schedule	unnormalized linear
Diffusion-time interval	$[\tau_{\min}, \tau_{\max}] = [10^{-3}, 5]$
Schedule parameters	$\beta_{\min} = 10^{-3}$, $\beta_{\mathrm{ref}} = \beta(1) = 0.5$
Training-time distribution	$\mathcal{U}[\tau_{\min}, \tau_{\max}]$
Prediction target and weighting	score, uniform weighting
Optimization and checkpoint selection	
Optimizer	AdamW
Learning rate	10^{-4}

Adam coefficients and numerical constant	$(0.9, 0.999), 10^{-8}$
Weight decay and maximum gradient norm	$0, 1$
Requested/effective batch size	$64/\min(64, M_{\text{train}})$
Mixed precision and EMA decay	enabled, 0.9999
Validation perturbations	fixed across checkpoints
Perturbations per validation path (pre-training/fine-tuning)	$1/64$
Checkpoint criterion	minimum EMA validation loss
Training budgets: pre-training/fine-tuning	
Maximum updates	$200\,000/50\,000$
Minimum updates before stopping	$50\,000/10\,000$
Validation interval	$1000/500$ updates
Early-stopping patience	$100/30$ validations
Minimum improvement	$10^{-4}/10^{-4}$
Learning-rate scheduler	ReduceLROnPlateau
Reduction factor and patience	$0.5/10$ validations
Relative threshold and minimum rate	$10^{-3}/10^{-6}$
U-Net	
Channel widths	$(32, 64, 128, 256)$
Convolution kernel and encoder strides	$21, (1, 2, 2, 2)$
Diffusion-time embedding dimension	1024
Normalization and activation	GroupNorm (8 groups), SiLU
DiT	
Width, blocks, and attention heads	$256, 8, 8$
Patch size and MLP ratio	$4, 4$
Position and time conditioning	learned positions, adaLN-Zero
Attention dropout	0
AR(RNN)	
History encoder	two-layer GRU, hidden size 128
Conditional denoiser	four residual MLP blocks, width 512
Diffusion-time embedding dimension	128
Training context	teacher forcing
Reverse-time sampling	
Solver	Euler–Maruyama reverse SDE
Integration interval	5 to 10^{-3}
U-Net/DiT steps (pre-training/fine-tuning)	$2000/1000$
Steps per AR conditional diffusion	500
Generated trajectories and batch size	$1000, 250$

F Optimization budget and compute

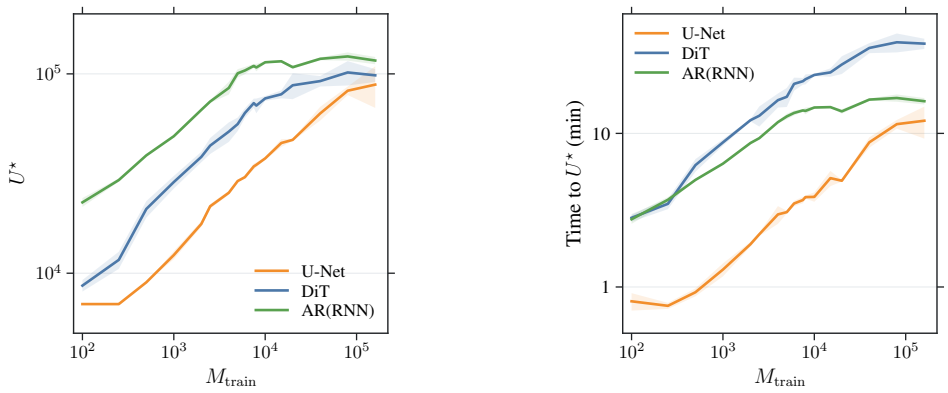

Figure 16: Cost of reaching the early-stopping checkpoint on PDV, shared colors with Figure 3; bands are one standard deviation across three seeds. Left: selected checkpoint $U^\star$ against M_{train}. AR(RNN) needs the most updates, 22 700 against 7 000 for U-Net at $M_{\text{train}} = 100$. Right: GPU time to reach $U^\star$, on an H100 for every run. The ordering changes: AR(RNN) saturates near 16 minutes whereas DiT keeps climbing to 38, its steps being far more expensive.