Stonk Bench: Unified Benchmark for Synthetic Data Generation for Financial Time Series

Uyen Lam Ho*
Eddison Pham*
uyenlam.ho@mail.utoronto.ca
eddison.pham@mail.utoronto.ca
University of Toronto
Toronto, Ontario, Canda

Figure 1: Toronto Stock Exchange, Toronto, 1981

Abstract

The ever emerging field of Synthetic Data Generation (SDG) has gain traction within many financial domains, including Finacial Time Series (FTS) data. However, there has been discorse and lack in universal agreement in what determines a better SDGFTS (Synthetic Data Generation for Financial Time Series), which maybe hindering progress in the field. In this paper, we propose and contribute a comprehensive benchmark for SDGFTS, covering various aspects such as data quality, privacy, and utility. We evaluate several state-of-the-art SDG methods using our benchmark and provide insights into their strengths and weaknesses. Our first-of-its-kind benchmark aims to facilitate the development of more effective SDG methods for FTS data, incorperating both classical statistical measures and utility benefits, and test over (insert number) to decicively determine what is the best SDGFTS model right now and for the future.

*Both authors contributed equally to this research.

Conference'17, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YYYY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Keywords

Synthetic Data Generation, Financial Time Series, Benchmarking

ACM Reference Format:
Uyen Lam Ho and Eddison Pham. 2025. Stonk Bench: Unified Benchmark for Synthetic Data Generation for Financial Time Series. In . ACM, New York, NY, USA, 30 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

Synthetic Data Generation (SDG) has emerged as a crucial tool in financial technology, particularly for Financial Time Series (FTS) data [**?**]. The ability to generate high-quality synthetic financial data addresses several critical challenges in the field, including data privacy concerns, limited data availability, and the need for diverse training datasets in machine learning applications [9].

Despite the growing importance of SDGFTS (Synthetic Data Generation for Financial Time Series), there is a notable lack of standardization in evaluating the quality and effectiveness of these generative models [9]. Current evaluation methods vary widely across studies, making it difficult to compare different approaches objectively and determine their relative strengths and weaknesses.

Our research addresses this gap by introducing a comprehensive benchmark framework that encompasses:

- Statistical fidelity measures for comparing synthetic and real FTS
- Privacy preservation metrics to ensure sensitive financial information remains protected
- Utility metrics that assess the practical value of synthetic data in downstream tasks

- Performance evaluation across multiple SDGFTS models and datasets

By analyzing the results from our MLflow experiments and applying our unified benchmark, we aim to provide clear guidelines for evaluating SDGFTS models and establish a standard for future research in this domain.

1.1 Limitations in the SDGFTS Literature

Among other things, we observed that there is currently no universal, generally accepted approach to evaluating synthetic time series[**?**]; this issue extends beyond FTS to generative frameworks like GANs as a whole [12]. Many evaluation measures are insufficiently defined and lack public implementations, making reuse and reproduction troublesome and error-prone [**?**]. This presents a challenge unique to the generation task compared to areas like time series forecasting or classification [**?**]. Hence, future research would immensely benefit from a widely accepted, reasonably sized set of qualified measures for the central evaluation criteria.

1.2 Our Contributions

To put it in simple terms, we are adopting the time series evaluation taxonomy outlined from Stenger et al. [**?**], develop a comprehensive and unified SDG benchmark expanded from the works of Ang et al. [1] in specifics to FTS by incorporating a utility evaluation framework inspired by Boursin et al. [2] through portfolio evaluations generated by a deep hedger. Our key contributions include:

[C1] **Consolidated Evaluation Framework:** We systematically review and consolidate evaluation methods from leading papers (models and surveys) in SDG and financial time series [1**?** , 2], creating a comprehensive assessment toolkit based on established practices.

[C2] **Unified Statistical and Utility Measures:** For the first time, we integrate both statistical fidelity metrics and practical utility measures in a single SDGFTS benchmark, providing a more complete evaluation of synthetic data quality.

[C3] **Open Benchmark Platform:** We deliver an open-source benchmark framework for the research community, aiming to establish a gold standard for SDGFTS model evaluation and comparison.

2 Preliminaries

2.1 Problem Definition

Let us formally define the Synthetic Data Generation problem for Financial Time Series. Given a financial time series $\mathbf{X} = (\mathbf{x}_1, ..., \mathbf{x}_R)^T$ with R individual series represented as an L-dimensional vector $\mathbf{x}_i = (x_{i1}, ..., x_{iL})$, where x_{ij} denotes the j-th point of time series $\mathbf{x}_i$. Let $p(\mathbf{x}_1, ..., \mathbf{x}_R)$ denote the real distribution of the given time series $\mathbf{X}$. The objective of SDG for FTS is to generate a synthetic time series $\hat{\mathbf{X}} = (\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_n)^T$ such that its distribution $p(\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_n)$ approximates $p(\mathbf{x}_1, ..., \mathbf{x}_R)$, while preserving the key properties exhibited in the real financial time series, such as autocorrelation structure, volatility clustering, and distributional moments (mean, variance, skewness, kurtosis).

2.2 Scope of Project

2.2.1 Scope of Methods. Our benchmark encompasses a diverse range of SDGFTS methods, from traditional parametric approaches to modern deep learning architectures. We selected models based on three main criteria: (1) proven industry adoption, (2) research community acceptance, and (3) recent methodological innovations.

Our evaluation includes classical parametric models, which rely on predefined statistical distributions and assumptions about the underlying data generation process. These models have been extensively used in financial institutions for their interpretability and theoretical foundations.

We also evaluate modern non-parametric approaches, particularly deep learning-based models that learn the data distribution directly from observations without assuming a specific form. These include generative adversarial networks, variational autoencoders, and diffusion models, representing the cutting edge in synthetic data generation.

2.2.2 Scope of Datasets. We evaluate models on diverse financial datasets spanning:

- Stock market indices (S&P 500, NASDAQ, TSX)
- Individual stock prices from major exchanges
- Forex trading pairs

2.2.3 Scope of Evaluation Taxonomical Criteria. Our evaluation framework follows the taxonomy structure proposed by Stenger et al. [9], incorporating various evaluation measures from different papers in the field.

We systematically integrate evaluation metrics from multiple sources while maintaining this structured taxonomical approach, ensuring comprehensive coverage of all critical aspects of SDGFTS evaluation and setting a standard for future research in time series SDG as a whole.

3 Overview of Synthetic FTS Methods

We categorize synthetic financial time series generation methods into two main families: classical parametric (stochastic) models, which rely on explicit assumptions about the data-generating process, and modern non-parametric (deep learning) models, which learn complex dependencies directly from data without predefined functional forms. The following subsections provide a detailed overview of the methods included in our benchmark.

3.1 Classical Stochastic (Parametric) Methods

Parametric models assume a specific functional form for the data-generating process, characterised by a finite set of parameters. These models are interpretable and analytically tractable, allowing for closed-form solutions in many cases. However, they may struggle to capture complex stylised facts or high-dimensional dependencies beyond their structural assumptions. Below we outline several widely-used parametric models in FTS.

[A1] **Geometric Brownian Motion (GBM).** The price process follows a continuous-time stochastic process with constant drift and volatility. Model-specific parameters are μ and σ.

$$dS_t = \mu S_t\, dt + \sigma S_t\, dW_t. \tag{1}$$

Table 1: Common notation and variables.

Symbol	Description
S_t	Asset price
$X_t = \ln S_t$	Log-price
$r_t = \ln(S_t/S_{t-1})$	Log-return
μ	Drift
σ	Volatility
W_t	Standard Brownian motion
Δt	Discrete time step

[A2] Ornstein–Uhlenbeck (OU). A mean-reverting Gaussian process for log-prices or spreads. Model-specific parameters are $\theta > 0$ (reversion rate), μ (long-term mean), and σ (volatility).

$$dX_t = \theta(\mu - X_t)\,dt + \sigma\,dW_t. \tag{2}$$

[A3] Merton Jump Diffusion (MJD). Extends GBM by incorporating normally-distributed jumps. Model-specific parameters are λ (jump intensity), μ_j and σ_j (mean and standard deviation of jump sizes).

$$\frac{dS_t}{S_t} = \mu\,dt + \sigma\,dW_t + J\,dN_t, \tag{3}$$

where $J \sim \mathcal{N}(\mu_j, \sigma_j^2)$, $\quad N_t \sim \text{Poisson}(\lambda t)$.

[A4] Double-Exponential Jump Diffusion (DEJD). A jump-diffusion model with asymmetric double-exponential jump sizes. Model-specific parameters are p (probability of upward jump) and η_1, η_2 (exponential parameters for upward/downward jumps).

$$\frac{dS_t}{S_t} = \mu\,dt + \sigma\,dW_t + Y\,dN_t, \tag{4}$$

$$Y = \begin{cases} \text{Exp}(\eta_1), & \text{with probability } p, \\ -\text{Exp}(\eta_2), & \text{with probability } 1-p. \end{cases}$$

[A5] GARCH(1,1). A discrete-time conditional heteroskedastic model for returns. Model-specific parameters are $\omega > 0$, $\alpha \geq 0$, $\beta \geq 0$ (GARCH coefficients) and $\mathcal{D}$ (innovation distribution, usually standard Normal or Student-t). Captures volatility clustering via time-varying conditional variance.

$$r_t = \sigma_t z_t, \quad z_t \sim \mathcal{D}, \tag{5}$$
$$\sigma_t^2 = \omega + \alpha r_{t-1}^2 + \beta \sigma_{t-1}^2.$$

[A6] Block Bootstrap. A resampling technique that preserves short-range dependence by sampling contiguous blocks of returns. Although not parametric, it is included here as it is a statistical model.

3.2 Deep Learning (Non-parametric) Methods

Non-parametric models, in this context, refer to generative (often implicit) models that do not assume a fixed parametric form for the underlying data-generating process. Instead, they learn latent structures directly from data through flexible architectures such as neural networks, enabling the modeling of highly nonlinear, high-dimensional, and multi-modal dependencies (e.g., multivariate financial time series with interactions across assets, time horizons, and features). While these models trade off interpretability and analytical tractability, they offer substantially greater expressive power and flexibility. This expressiveness, however, comes at the cost of increased data requirements, more intensive hyperparameter tuning, and higher computational demands.

We categorize these approaches into three major families of deep generative models: Generative Adversarial Networks (GANs), Variational Autoencoders (VAEs), and Diffusion Models.

3.2.1 Diffusion Models. Diffusion models (or score-based generative models) have emerged as powerful methods for modeling complex data distributions via a forward (noise-adding) process and a learned reverse (denoising) process [3**?**]. In the forward direction, one gradually corrupts a data sample x_0 into noise via a Markov chain:

$$q(x_{1:T} \mid x_0) = \prod_{t=1}^{T} q(x_t \mid x_{t-1}), \tag{6}$$
$$q(x_t \mid x_{t-1}) = \mathcal{N}\left(x_t; \sqrt{1-\beta_t}\,x_{t-1}, \beta_t I\right).$$

The reverse (generative) model is parameterized as

$$p_\theta(x_{t-1} \mid x_t) = \mathcal{N}\big(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)\big),$$

where μ_θ and Σ_θ are often expressed via a neural network that estimates the score $\nabla_\theta \log p_\theta(x_t)$. A common training objective is the simplified denoising score-matching loss:

$$\mathbb{E}_{t,x_0,\epsilon}\left[\left\|\epsilon - \epsilon_\theta(x_t, t)\right\|^2\right],$$

where $x_t = \sqrt{\bar{\alpha}_t}\,x_0 + \sqrt{1-\bar{\alpha}_t}\,\epsilon$.

In finance/time-series, diffusion models have recently been adapted to generate synthetic asset paths by converting multivariate time series into suitable representations (e.g. wavelet-transformed images) and then sampling via reverse diffusion [11]. For instance, Takahashi and Mizuno shows that such an approach can reproduce key statistical properties of financial data (e.g. volatility clustering, tail behavior) [10]. Another relevant work is "A Financial Time Series Denoiser Based on Diffusion Model", which shows how diffusion can help improve downstream predictability and reduce noise in financial signals [13].

3.2.2 Variational Autoencoders (VAEs). Variational Autoencoders [8**?**] are latent-variable generative models that posit a probabilistic encoder $q_\phi(z \mid x)$ and decoder $p_\theta(x \mid z)$. One maximizes the evidence lower bound (ELBO):

$$\mathcal{L}_{\text{VAE}} = \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x \mid z)\right] - \text{KL}\big(q_\phi(z \mid x) \,\|\, p(z)\big).$$

Often one uses a standard Gaussian prior $p(z) = \mathcal{N}(0, I)$. The encoder–decoder structure allows sampling by first drawing $z \sim p(z)$, then generating $x \sim p_\theta(x \mid z)$.

In financial time-series generation, specialized versions such as the Time-Causal VAE (TC-VAE) have been proposed to enforce causality in the encoding/decoding of temporal data. For instance, Acciaio et al. (2024) propose TC-VAE, which ensures that generated paths respect temporal causality and show that the model reproduces stylized facts (e.g., heavy tails, volatility clustering) on real

markets [**?**]. Because VAEs provide a principled latent representation, they are useful in finance to model underlying latent drivers of asset dynamics and to sample coherent trajectories.

3.2.3 Generative Adversarial Networks (GANs). Generative Adversarial Networks [**?**] approach generation as a two-player game between a generator G and a discriminator D. The discriminator is trained to maximize the probability of correctly classifying real data and generated data, with loss

$$L_D = -\mathbb{E}_{x\sim p_{\text{data}}}\big[\log D(x)\big] - \mathbb{E}_{z\sim p(z)}\big[\log(1 - D(G(z)))\big].$$

The generator aims to fool the discriminator and is trained to minimize

$$L_G = -\mathbb{E}_{z\sim p(z)}[\log D(G(z))].$$

Many GAN variants, such as Wasserstein GAN or Least Squares GAN, modify these loss functions to promote more stable training and address issues like mode collapse.

In financial data synthesis, GANs have been widely used to generate realistic synthetic time series. For example, GAN-based financial data generation models (often using WGAN or improvements) show that one can improve the authenticity and predictive ability of generated financial statements or time series [7]. Another example is VRNNGAN, which uses a recurrent VAE as the generator and a recurrent discriminator to capture temporal dependencies in synthetic sequence generation [4]. GANs are relevant in finance because they can capture complex joint distributions and dependencies in multivariate time-series without requiring explicit likelihood models.

3.3 Miscellaneous

This section consists of models that are classified as neither parametric nor non-parametric.

[A1] Block Bootstrap (non-parametric). A resampling technique that preserves short-range dependence by sampling contiguous blocks of returns. Although not parametric, it is included here for completeness and baseline comparison.

4 Stonk Bench Architecture

4.1 Datasets and Preprocessing

The preprocessing procedure builds upon the TSGBench pipeline, which standardizes segmentation, normalization, and train/test splitting of multivariate time series. Several modifications were introduced to better accommodate financial time series and stochastic models.

4.1.1 Data type. Stock price data (e.g., AAPL) at daily frequency with OHLC columns (Open, High, Low, Close) are used. Let $X \in \mathbb{R}^{L\times N}$ denote a multivariate time series with $N = 4$ channels and length L.

4.1.2 Transformations. Raw prices are converted to log-returns per channel via

$$r_t = \log P_t - \log P_{t-1} = \log(P_t/P_{t-1}), \quad r_t \in \mathbb{R}^N,$$

yielding a transformed series of length $L - 1$.

4.1.3 Model-specific preprocessing. The preprocessing follows the general TSGBench framework, with the following adaptations:

- Log returns are used directly instead of TSGBench's min-max normalization, preserving financial properties such as heavy tails and volatility clustering.
- For parametric models, the transformed series is divided into contiguous training, validation, and test segments with ratios $(1 - \alpha - \beta) : \alpha : \beta$, where $\alpha = 0.1$ and $\beta = 0.1$. This ensures that parametric models fit parameters directly to contiguous historical data.
- For non-parametric models, overlapping sliding windows $\mathcal{W} \in \mathbb{R}^{R\times L\times N}$ with stride 1 are extracted. Unlike TSGBench, which employs an autocorrelation-based hard-coded window of 125 (often yielding a size of 1 for stock data due to fast autocorrelation decay), the window length L is determined via the partial autocorrelation function (PACF). Specifically, PACF is computed for each channel up to $\lfloor 10 \log_{10} n \rfloor$ lags, and the lag corresponding to the maximum significant PACF peak across channels is selected, resulting in $L = 13$. This approach captures relevant temporal dependencies within the data.
- The dataset is split into train, validation, and test sets while preserving temporal order to prevent future information from leaking into the past. Only the training set is shuffled during model training to improve convergence. In contrast, TSG-Bench shuffles time series samples before splitting, which introduces data leakage. Our approach ensures strictly forward-looking evaluation for realistic forecasting.
  proper evaluation, consistent with TSGBench.

4.1.4 Data loaders. For non-parametric models, mini-batches are generated from the windowed data $\mathcal{W}$ using independent fixed seeds for training, validation, and test sets, enabling reproducible epoch-wise evaluation. Training batches are shuffled, while validation and test sets maintain sequential ordering. To ensure fair comparison, parametric models generate sequences of identical length L and matching sample count to the windowed data used by non-parametric models.

4.2 Statistical Evaluation Measures

All metrics below are applied **per channel**, i.e., each univariate time series (OHLC column) is evaluated independently. For multivariate data, results are reported as channel-wise statistics (e.g., mean or distribution across channels).

4.2.1 Feature-based Distance Measures. These metrics quantify how well synthetic data captures key marginal and second-order properties of real data.

[M1] Marginal Distribution Difference (MDD): Distance between real and synthetic marginals (e.g., 1-Wasserstein distance) per channel:

$$\text{MDD}_j = W_1\big(\widehat{F}^j_{real}, \widehat{F}^j_{synth}\big),$$

$$W_1(F, G) = \int_0^1 \big|F^{-1}(u) - G^{-1}(u)\big|\, du,$$

where $\widehat{F}^j_{real}$ and $\widehat{F}^j_{synth}$ are empirical CDFs of channel j.

[M2] Mean Difference (MD): Absolute difference of per-channel means:

$$\mathrm{MD}_j = \left|\mu^j_{real} - \mu^j_{synth}\right|.$$

[M3] Standard Deviation Difference (SDD): Absolute difference of per-channel standard deviations:

$$\mathrm{SDD}_j = \left|\sigma^j_{real} - \sigma^j_{synth}\right|.$$

[M4] Kurtosis Difference (KD): Absolute difference of per-channel excess kurtosis:

$$\mathrm{KD}_j = \left|\kappa^{\mathrm{ex}}(X^j_{\mathrm{real}}) - \kappa^{\mathrm{ex}}(X^j_{\mathrm{synth}})\right|,$$
$$\kappa^{\mathrm{ex}}(X^j) = \frac{\mathbb{E}[(X^j - \mu^j)^4]}{(\sigma^j)^4} - 3.$$

[M5] AutoCorrelation Difference (ACD): Absolute difference in lag-1 autocorrelation per channel:

$$\rho^j(1) = \frac{\sum_{t=2}^{L}(x^j_t - \bar{x}^j)(x^j_{t-1} - \bar{x}^j)}{\sum_{t=1}^{L}(x^j_t - \bar{x}^j)^2},$$
$$\mathrm{ACD}_j = \left|\rho^j_{real}(1) - \rho^j_{synth}(1)\right|.$$

4.2.2 Visualization Methods. Visual tools complement numeric metrics for face-validity and communication [1].

[M6] t-SNE: 2D embeddings of window-level features per channel for real vs. synthetic overlap and cluster structure.

[M7] Distribution: Channel-wise histograms of returns to visualize empirical distribution.

4.2.3 Diversity Metrics. Measures that prevent mode collapse, computed per channel. Pairwise distances only consider the same channel across samples.

[M8] Intra-Class Euclidean Distance (ICD-ED):

$$\text{ICD-ED}^j = \frac{2}{R(R-1)} \sum_{1 \le a < b \le R} \left\| \mathbf{x}^j_a - \mathbf{x}^j_b \right\|_2,$$

where $\mathbf{x}^j_a \in \mathbb{R}^L$ is the j-th channel of the a-th sample.

[M9] Intra-Class Dynamic Time Warping (ICD-DTW):

$$\text{ICD-DTW}^j = \frac{2}{R(R-1)} \sum_{1 \le a < b \le R} d_{\mathrm{DTW}}\left(\mathbf{x}^j_a, \mathbf{x}^j_b\right).$$

4.2.4 Efficiency Assessment.

[M10] Generation Time: Wall-clock time to generate S samples:

$$T_{\mathrm{gen}} = t_1 - t_0 \quad \text{(seconds for } S \text{ samples)}.$$

4.2.5 Stylized Facts Verification. Canonical stylized facts, measured per channel:

[M11] Heavy Tails: Per-channel excess kurtosis:

$$\kappa^{\mathrm{ex}}(X^j) = \frac{\mathbb{E}[(X^j - \mathbb{E}[X^j])^4]}{(\mathrm{Var}[X^j])^2} - 3.$$

[M12] Return Autocorrelation: Lag-1 autocorrelation per channel, should be close to zero for liquid assets.

[M13] Volatility Clustering: Lag-1 autocorrelation of squared or absolute returns per channel:

$$\rho^j_{x^2}(1) = \mathrm{Corr}\left((x^j_t)^2, (x^j_{t-1})^2\right),$$
$$\rho^j_{|x|}(1) = \mathrm{Corr}\left(|x^j_t|, |x^j_{t-1}|\right).$$

4.3 Utility Evaluation Measures: Deep Hedging

Utility measures are critical for evaluating synthetic data beyond statistical metrics, as they assess the practical value of generated data in real-world financial applications [2]. Our benchmark incorporates deep hedging as a utility measure for several key reasons:

[U1] Real-world Application Testing: Deep hedging provides a concrete way to evaluate how synthetic data performs in actual financial tasks, particularly in derivatives pricing and risk management.

[U2] Industry-relevant Metrics: By comparing hedging strategies trained on synthetic versus real data, we can assess the practical utility of synthetic data through metrics that matter to financial practitioners, such as replication errors and hedging performance.

[U3] Model Robustness Validation: Deep hedging helps verify if synthetic data maintains the complex relationships and market dynamics necessary for developing reliable trading strategies.

4.3.1 Deep Hedger Problem Defintion. We consider a continuous-time financial market defined on a probability space $(\Omega, \mathcal{F}, \mathbb{P})$, over a finite time horizon $0 < T < \infty$, equipped with a filtration $\mathcal{F} = (\mathcal{F}_t)_{0 \le t \le T}$ that represents the evolution of information through time. The market consists of $d+1$ tradable assets $S = (S^0, \ldots, S^d)$, where S^j_t denotes the price of asset j at time t. For simplicity, we assume a zero interest rate environment.

We study the hedging problem associated with a contingent claim delivering a payoff $g(S_T)$ at maturity T, where S_T represents the terminal value of the underlying asset vector. The hedging strategy is implemented at a discrete set of times $0 = t_0 < t_1 < \cdots < t_{N-1} < t_N = T$. A *self-financing portfolio* is described by a d-dimensional $\mathcal{F}_t$ -adapted process Δ_t, and its terminal wealth, denoted $X^{\Delta,p}_T$, is given by

$$X^{\Delta,p}_T = p + \sum_{i=1}^{d} \sum_{j=0}^{N-1} \Delta^i_{t_j} \left(F^i_{t_{j+1}} - F^i_{t_j} \right), \tag{7}$$

where $p \in \mathbb{R}$ represents the initial premium.

The objective is to determine the optimal premium and trading strategy $(p^{\mathrm{opt}}, \Delta^{\mathrm{opt}})$ that minimize the expected squared hedging error:

$$(p^{\mathrm{opt}}, \Delta^{\mathrm{opt}}) = \underset{p,\Delta}{\mathrm{Argmin}}\, \mathbb{E}\left[\left(X^{\Delta}_T - g(S_T) \right)^2 \right]. \tag{8}$$

To address this optimization problem, we employ the global approach proposed by Fécamp *et al.* (2020), chosen for its computational efficiency and scalability. In this framework, the control policy Δ is approximated by a feed-forward neural network—referred to as a *deep hedger*—which jointly learns the optimal trading strategy and corresponding premium through end-to-end training.

4.3.2 Deep Hedger Architecture. Our deep hedger architecture centeres around the Train-Synthetic-Test-Real (TSTR) evaluation framework [5], adapted for financial time series generation. We begin with our real FTS dataset D—partitioned to a training, validation, and test set respectively as $D_r = \{D^{train}_r, D^{validate}_r, D^{test}_r\}$—along with a specific hedging task T and a performance score S to evaluate hedging effectiveness, to which we will go indepth in the next

Table 2: Notations for Deep Hedger Architecture.

Symbol	Description
D	Time-series dataset
T	Task
S	Score
A	Deep hedger algorithm
$\mathcal{A}_n$	Set of n deep hedger algorithms
M	Deep hedger model
$\mathcal{M}_{TS}$	(SDGFTS) model

subsubsection. We consider a deep hedger algorithm A (a 5-layer perceptron) that takes as input the dataset D_r and task T,

In the context of SDGFTS evaluation, we consider a SDGFTS model $\mathcal{M}_{TS}$ trained on D_r^{train} and validated with $D_r^{validate}$ to generate synthetic FTS data D_g.

Now we can train a deep hedger algorithm A on the datasets D_r^{train} and D_g^{train} respectively, resulting in two deep hedger models M_r and M_g.

Finally, we evaluate both models on the real test set D_r^{test} to obtain the corresponding performance scores s_r and s_g.

The goal of the deep hedger portfolio evaluation is to evaluate the effectiveness of the model $\mathcal{M}_{TS}$ with downstream tasks (deep hedging) by comparing the scores s_r and s_g. The model $\mathcal{M}_{TS}$ is considered effective if the the results yielded similar scores, i.e. $s_r \approx s_g$ —preferably having higher performance with results of $s_g > s_r$ —indicating that the synthetic data generated by $\mathcal{M}_{TS}$ is useful for training deep hedger models [9].

4.3.3 Deep Hedger Extentions. We proposed two extensions to the standard deep hedger architecture to better suit our benchmark need that drew inspiration from recent literature [9].

[E1] Augmented Testing [9] To better evaluate the performance of deep hedgers trained on synthetic data, we train our deep hedger A on a new dataset D_{aug} that combines both real training data D_r^{train} and synthetic data D_g. In other words, we traid model M_g on D_{aug} where $D_{aug} := D_r^{train} \cup D_g$. Then we proceed to evaluate scores between modelsM_g and M_r on the real test set, where M_r i s still trained on only real data D_r^{train}.

[E2] Algorithm Comparison [6] Another extension is to compare multiple deep hedger algorithms indicating to what degree each algorithm performs equally on the generated data relative to the other algorithms, compared to their performance on real data.

Given a set of n deep hedger algorithms $\mathcal{A}_n = \{A_1, \ldots, A_n\}$, we train each algorithm $A_i \in mathcalA_n$ on both real training data D_r^{train} and synthetic data D_g^{train} respectively, resulting in two sets of deep hedger models $\{M_r^1, M_r^2, \ldots, M_r^n\}$ and $\{M_g^1, M_g^2, \ldots, M_g^n\}$ and two ordered setes of performance scores $s_r := \{s_r^1, s_r^2, \ldots, s_r^n\}$ and $s_g := \{s_g^1, s_g^2, \ldots, s_g^n\}$ respectively.

Finally we compare the relative performance of each algorithm on real data versus synthetic data by computing the Spearman's rank correlation coefficient r_S [6] between the two score sets s_r and s_g:

$$r_S = 1 - \frac{6 \sum_{i=1}^{n} (\text{rank}(s_r^i) - \text{rank}(s_g^i))^2}{n(n^2 - 1)}, \tag{9}$$

where $\text{rank}(s_r^i)$ and $\text{rank}(s_g^i)$ denote the ranks of scores s_r^i and s_g^i within their respective sets.

We interpret a high positive correlation (i.e., r_S close to 1) as an indication that the synthetic data generated by $mathcalM_{TS}$ preserves the relative performance of different deep hedger algorithms.

5 Results and Analysis

5.1 Statistical Evaluation Results

We report per-model, per-channel metrics. Copy values from the generated JSON (e.g., `complete_evaluation.json`) into the following templates.

5.2 Utility Evaluation Results

Results of the deep hedging evaluation (replication error, P&L distribution, risk-adjusted metrics) can be summarized in tables analogous to the above once computed.

5.3 Comprehensive Model Comparison

Provide radar plots or scorecards that combine normalized metrics (fidelity, diversity, stylized facts, efficiency) into composite ranks per task.

5.4 Ranking Analysis

Discuss trade-offs: fidelity vs. diversity, training time vs. quality, and sensitivity to window length and sample count.

6 Conclusion and Future Work

In our research, we recognize the existing limitations in the evaluation of synthetic time series, particularly the absence of a universally accepted framework. To address this, we adopt the evaluation taxonomy proposed by Stenger et al. and expand upon it to create a comprehensive benchmark specifically tailored for Synthetic Data Generation for Financial Time Series (SDGFTS). Our contributions include the development of a consolidated evaluation framework that systematically reviews and integrates various assessment methods from leading studies in the field. This approach not only enhances the rigor of evaluations but also facilitates the comparison of different SDGFTS models, ultimately providing a more standardized and reliable means of assessing synthetic data quality.

6.1 Summary of Findings

6.2 Implications for SDGFTS Research

6.3 Current Limitations and Shortcomings

6.4 Future Research Directions

Acknowledgments

To professor Irene Huang, University of Toronto, for her supervision and guidance throughout the project.

Table 3: Fidelity metrics by model and channel (lower ↓ is better for differences; best value in each column bolded). Channels O, H, L, C correspond to dataset channels (Open, High, Low, Close).

	MDD ↓				MD ↓				SDD ↓			
Model	O	H	L	C	O	H	L	C	O	H	L	C
GBM	4.35	4.48	4.48	3.41	0.0010	0.0004	0.0005	0.0014	0.0115	0.0108	0.0119	0.0110
OU_Process	4.31	4.48	4.43	3.37	0.0010	0.0003	0.0005	0.0014	0.0115	0.0107	0.0119	0.0110
MJD	3.74	3.89	3.86	2.87	0.0014	0.0004	0.0004	0.0004	0.0104	0.0115	0.0133	0.0107
GARCH11	**2.60**	**2.47**	2.88	**1.92**	**0.0000**	**0.0001**	**0.0001**	**0.0001**	**0.0004**	0.0005	**0.0020**	**0.0012**
DEJD	2.79	**2.38**	**2.55**	2.17	0.0015	0.0009	0.0018	0.0011	0.0138	0.0116	0.0160	0.0141
BlockBootstrap	2.82	2.75	2.78	2.20	0.0014	0.0005	0.0003	0.0007	0.0102	**0.0102**	0.0108	0.0118
TimeGAN	4.83	5.04	5.31	3.85	0.0029	0.0025	0.0038	0.0022	0.0072	0.0074	0.0084	0.0063
QuantGAN	3.87	3.64	4.28	2.50	0.0053	0.0006	0.0077	0.0030	0.0081	0.0076	0.0112	0.0059
TimeVAE	9.98	9.77	10.42	8.35	0.0014	0.0019	0.0021	0.0022	0.0191	0.0163	0.0178	0.0193
Takahashi	5.63	5.42	5.66	4.37	0.0508	0.0827	0.1241	0.0713	1.1989	1.2980	1.3178	1.3779

	KD ↓				ACD ↓			
Model	O	H	L	C	O	H	L	C
GBM	3.64	6.09	4.03	6.18	0.159	0.175	0.235	0.269
OU_Process	3.64	6.07	4.02	6.17	0.158	0.148	0.204	0.273
MJD	3.28	13.01	23.14	12.47	0.188	0.198	0.244	0.271
GARCH11	3.50	**5.89**	**3.36**	**5.98**	0.224	0.199	0.230	0.257
DEJD	92.26	61.12	133.55	61.41	0.167	0.158	0.257	0.265
BlockBootstrap	**2.20**	5.27	8.33	3.69	0.177	0.181	0.172	**0.305**
TimeGAN	4.99	6.73	4.92	7.32	0.345	0.390	0.445	0.327
QuantGAN	3.62	6.11	4.08	7.37	**0.154**	0.225	0.272	0.357
TimeVAE	5.14	3.16	3.95	7.23	2.762	3.255	3.462	1.888
Takahashi	3.12	5.48	3.42	6.84	0.216	**0.207**	**0.229**	0.327

Table 4: Temporal and diversity metrics by model and channel (lower ↓ is better for differences; higher ↑ is better for diversity distances). Best value in each column bolded.

	ICD-ED ↑				ICD-DTW ↑			
Model	O	H	L	C	O	H	L	C
GBM	0.150	0.133	0.145	0.150	0.100	0.088	0.097	0.099
OU_Process	0.150	0.133	0.145	0.149	0.100	0.088	0.096	0.100
MJD	0.140	0.130	0.143	0.142	0.096	0.092	0.101	0.098
GARCH11	0.090	0.077	0.095	0.088	0.061	0.051	0.064	0.060
DEJD	0.135	0.118	0.132	0.143	0.102	0.091	0.104	0.108
BlockBootstrap	0.138	0.125	0.132	0.146	0.100	0.091	0.097	0.107
TimeGAN	0.133	0.121	0.132	0.131	0.081	0.073	0.080	0.079
QuantGAN	0.137	0.122	0.146	0.130	0.092	0.081	0.097	0.086
TimeVAE	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Takahashi	**5.974**	**6.454**	**6.529**	**6.859**	**4.078**	**4.502**	**4.503**	**4.744**

References

[1] Yihao Ang, Qiang Huang, Yifan Bao, Anthony K. H. Tung, and Zhiyong Huang. 2023. TSGBench: Time Series Generation Benchmark. *Proc. VLDB Endow.* 17, 3 (2023), 305–318.

[2] Nicolas Boursin, Carl Remlinger, Joseph Mikael, and Carol Anne Hargreaves. 2022. Deep Generators on Commodity Markets; application to Deep Hedging. arXiv:2205.13942 [q-fin.RM] https://arxiv.org/abs/2205.13942

[3] Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. In *Proceedings of the 34th International Conference on Neural Information Processing Systems* (Vancouver, BC, Canada) *(NIPS '20)*. Curran Associates Inc., Red Hook, NY, USA, Article 574, 12 pages.

[4] J. Lee et al. 2022. VRNNGAN: A Recurrent VAE-GAN Framework for Synthetic Time-Series Generation. In *Conference*.

[5] Mark Leznik, Patrick Michalsky, Peter Willis, Benjamin Schanzel, Per-Olov Östberg, and Jörg Domaschka. 2021. Multivariate Time Series Synthesis Using Generative Adversarial Networks. In *Proceedings of the ACM/SPEC International Conference on Performance Engineering* (Virtual Event, France) *(ICPE '21)*. Association for Computing Machinery, New York, NY, USA, 43–50. doi:10.1145/3427921.3450257

[6] Zinan Lin, Alankar Jain, Chen Wang, Giulia Fanti, and Vyas Sekar. 2021. Using GANs for Sharing Networked Time Series Data: Challenges, Initial Promise, and Open Questions. arXiv:1909.13403 [cs.LG] doi:10.1145/3419394.3423643

[7] Feng Qi. 2025. GAN-Based Financial Data Generation and Prediction: Improving The Authenticity and Prediction Ability of Financial Statements. *Informatica*

Table 5: Stylized facts by model and channel. Differences are reported as real minus synthetic; for all metrics, lower ↓ is better. Best value in each column bolded.

	Heavy Tails (diff) ↓				Autocorr Raw (diff) ↓				Volatility Clustering (diff) ↓			
Model	O	H	L	C	O	H	L	C	O	H	L	C
GBM	0.573	0.833	0.753	0.627	0.047	0.093	0.059	0.035	0.019	0.048	0.040	0.033
OU_Process	0.622	0.858	0.794	0.608	0.025	0.007	0.007	0.061	0.032	0.061	0.018	0.018
MJD	**0.108**	0.229	0.183	**0.126**	0.065	0.073	0.057	0.041	0.038	0.023	0.030	0.013
GARCH11	0.687	0.928	0.784	0.685	0.050	0.098	0.057	0.017	0.055	0.062	0.059	0.024
DEJD	0.217	0.360	0.380	0.210	0.049	0.107	0.067	0.034	0.022	0.051	0.048	0.029
BlockBootstrap	0.207	**0.081**	0.257	0.152	**0.022**	0.014	0.034	0.046	0.017	0.014	0.028	**0.012**
TimeGAN	1.074	1.363	1.206	1.097	0.163	0.188	0.149	0.180	0.014	0.015	**0.005**	0.061
QuantGAN	0.387	0.615	0.630	0.387	0.046	0.095	0.069	**0.007**	**0.010**	**0.009**	0.007	0.014
TimeVAE	1.225	2.307	0.277	0.962	0.491	0.267	0.415	0.512	0.523	0.292	0.350	0.553
Takahashi	0.482	0.621	0.650	0.480	0.055	**0.007**	**0.030**	0.046	0.012	0.011	0.017	0.030

(2025).

[8] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32* (Beijing, China) *(ICML'14)*. JMLR.org, II–1278–II–1286.

[9] Michael Stenger, Robert Leppich, Ian Foster, Samuel Kounev, and André Bauer. 2024. Evaluation is key: a survey on evaluation measures for synthetic time series. *J Big Data* 11, 66 (May 2024). doi:10.1186/s40537-024-00924-7

[10] Tomonori Takahashi and Takayuki Mizuno. 2024. Generation of synthetic financial time series by diffusion models. arXiv:2410.18897 [q-fin.CP] https://arxiv.org/abs/2410.18897

[11] Yuki Tanaka, Ryuji Hashimoto, Takehiro Takayanagi, Zhe Piao, Yuri Murayama, and Kiyoshi Izumi. 2025. CoFinDiff: Controllable Financial Diffusion Model for Time Series Generation. arXiv:2503.04164 [q-fin.CP] https://arxiv.org/abs/2503.04164

[12] Zhengwei Wang, Qi She, and Tomas E. Ward. 2020. Generative Adversarial Networks in Computer Vision: A Survey and Taxonomy. arXiv:1906.01529 [cs.LG] https://arxiv.org/abs/1906.01529

[13] Zhuohan Wang and Carmine Ventre. 2024. A Financial Time Series Denoiser Based on Diffusion Models. In *Proceedings of the 5th ACM International Conference on AI in Finance* (Brooklyn, NY, USA) *(ICAIF '24)*. Association for Computing Machinery, New York, NY, USA, 72–80. doi:10.1145/3677052.3698649

A Preprocessing Diagnostics
A.1 Channel-wise Distributions
A.2 Autocorrelation Analyses
B Embedding Visualizations
C Model Visualizations
C.1 GBM
C.2 OU Process
C.3 MJD
C.4 GARCH11
C.5 DEJD
C.6 BlockBootstrap
C.7 TimeGAN
C.8 QuantGAN
C.9 TimeVAE
C.10 Takahashi
D Evaluation Metrics Visualizations

Figure 2: Histograms and Q–Q plots of real vs. synthetic returns per channel. Replace with generated figures from `VisualAssessmentEvaluator`.

Figure 3: ACF/PACF of returns and absolute returns for real vs. synthetic data.

Figure 4: t-SNE/UMAP embeddings of window-level features; color-coded real vs. synthetic for overlap assessment.

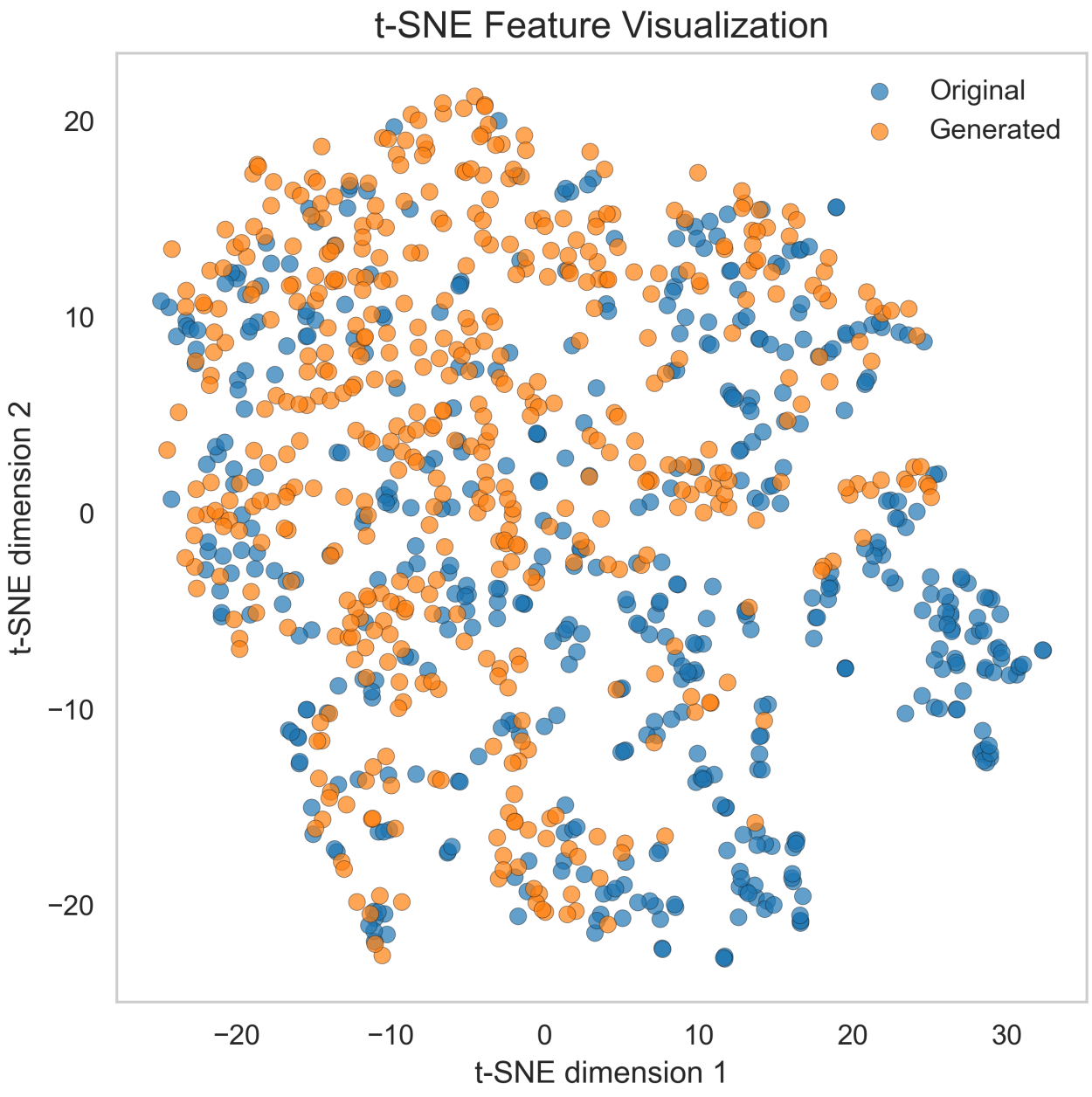

Figure 5: GBM: t-SNE embeddings of window-level features.

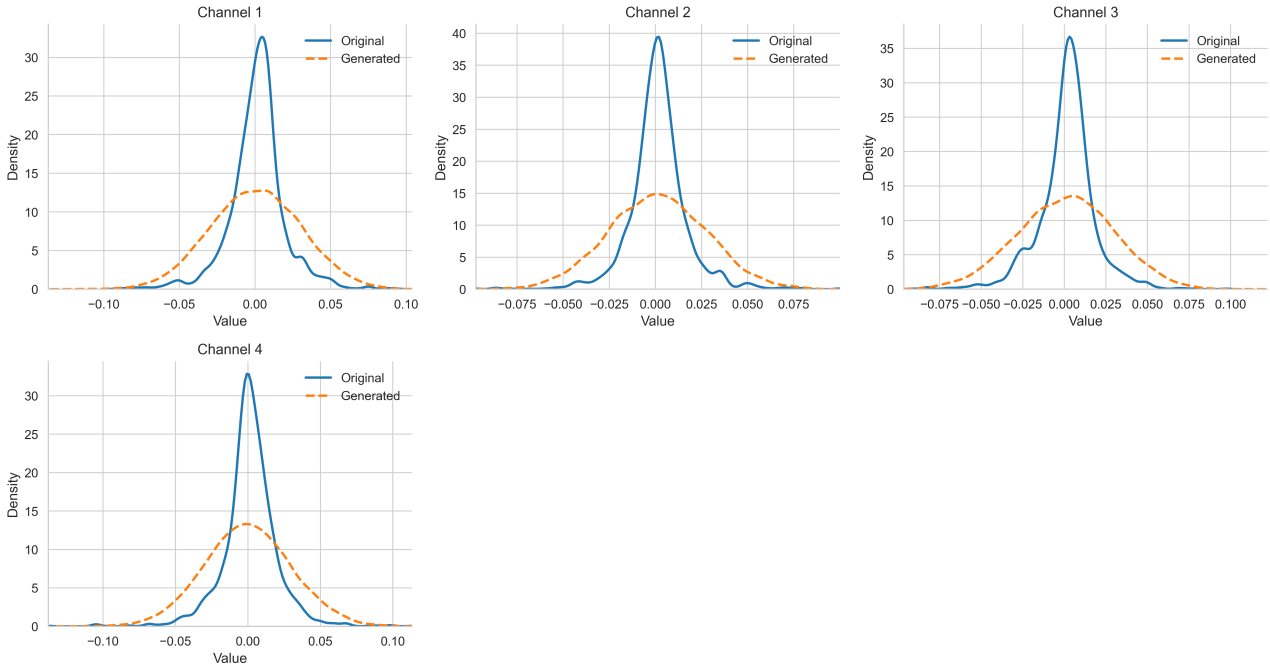

Figure 6: GBM: Channel-wise distributions for real vs. synthetic data.

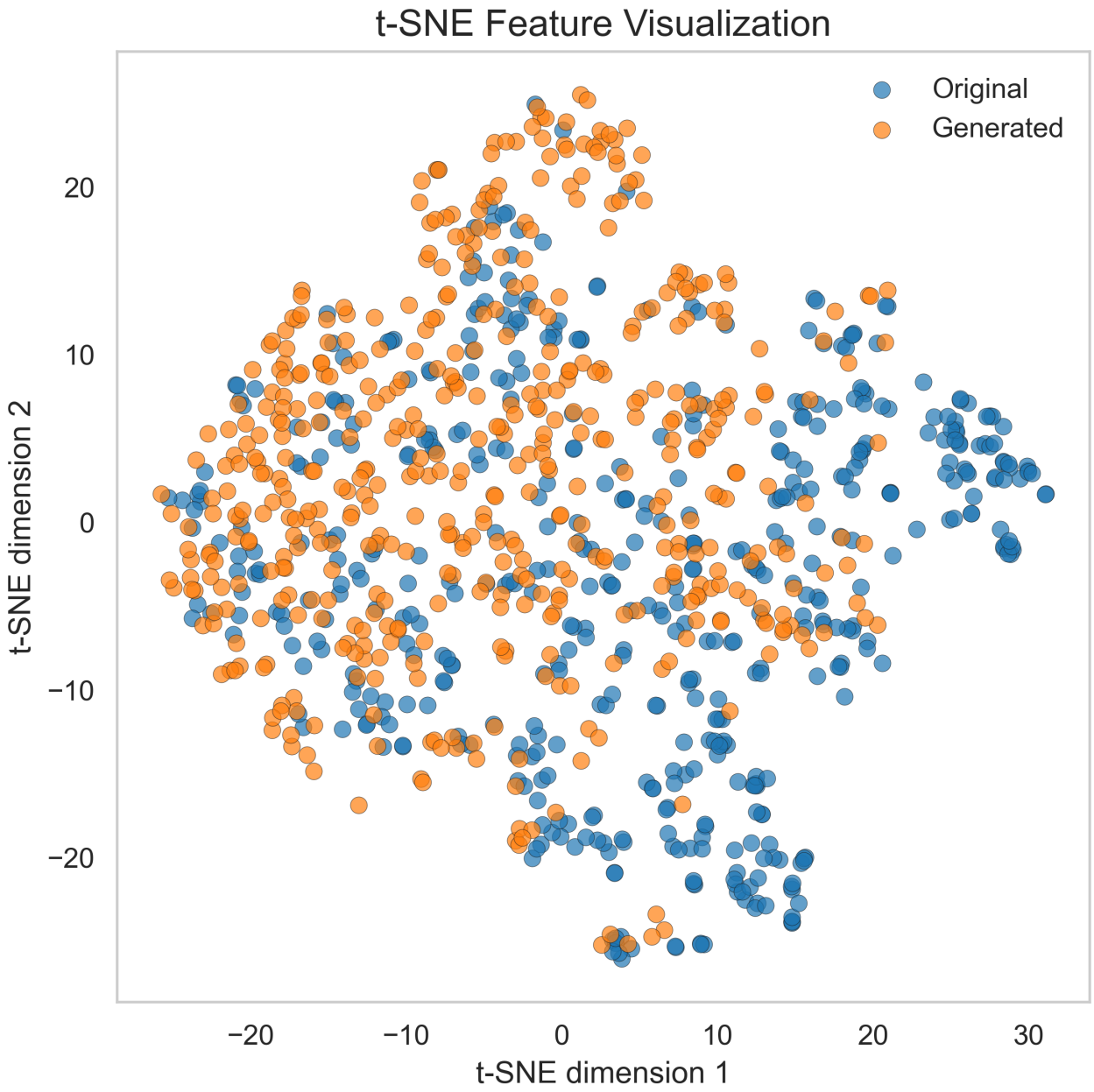

Figure 7: OU Process: t-SNE embeddings of window-level features.

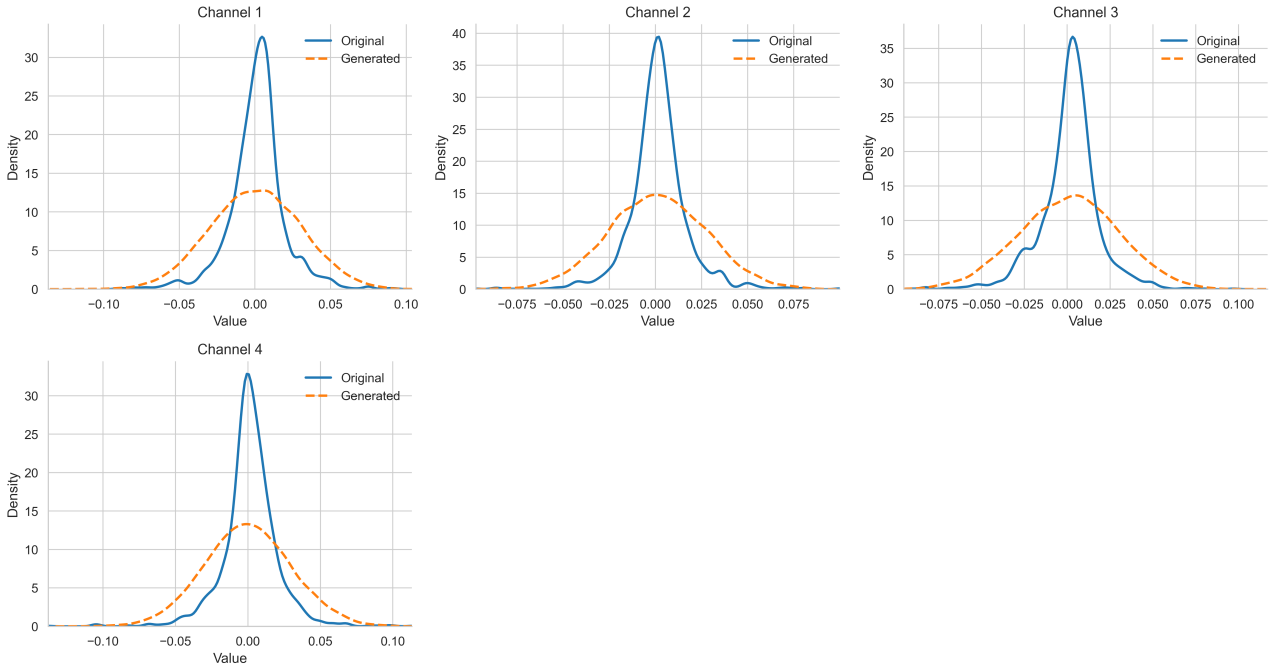

Figure 8: OU Process: Channel-wise distributions for real vs. synthetic data.

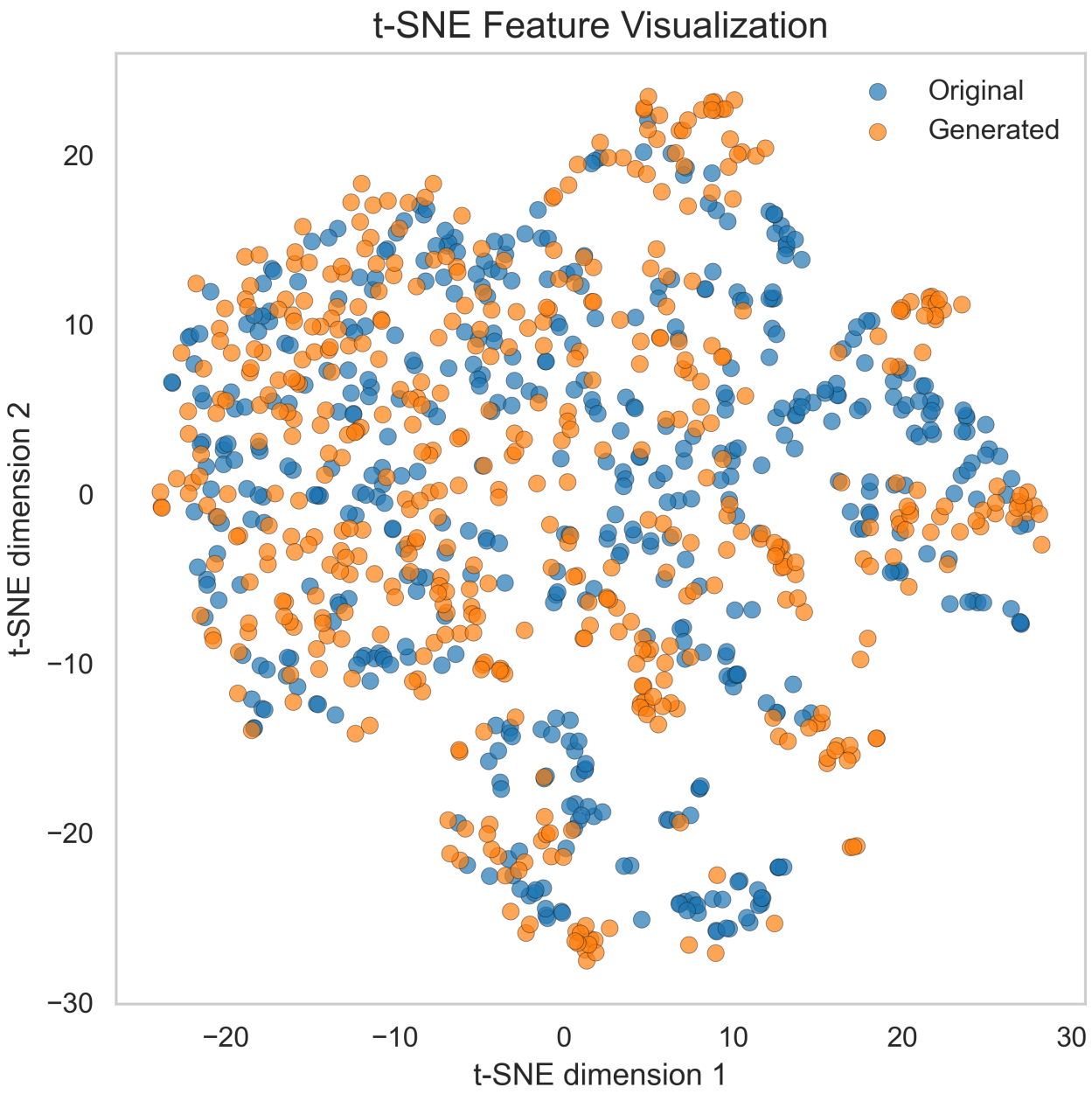

Figure 9: MJD: t-SNE embeddings of window-level features.

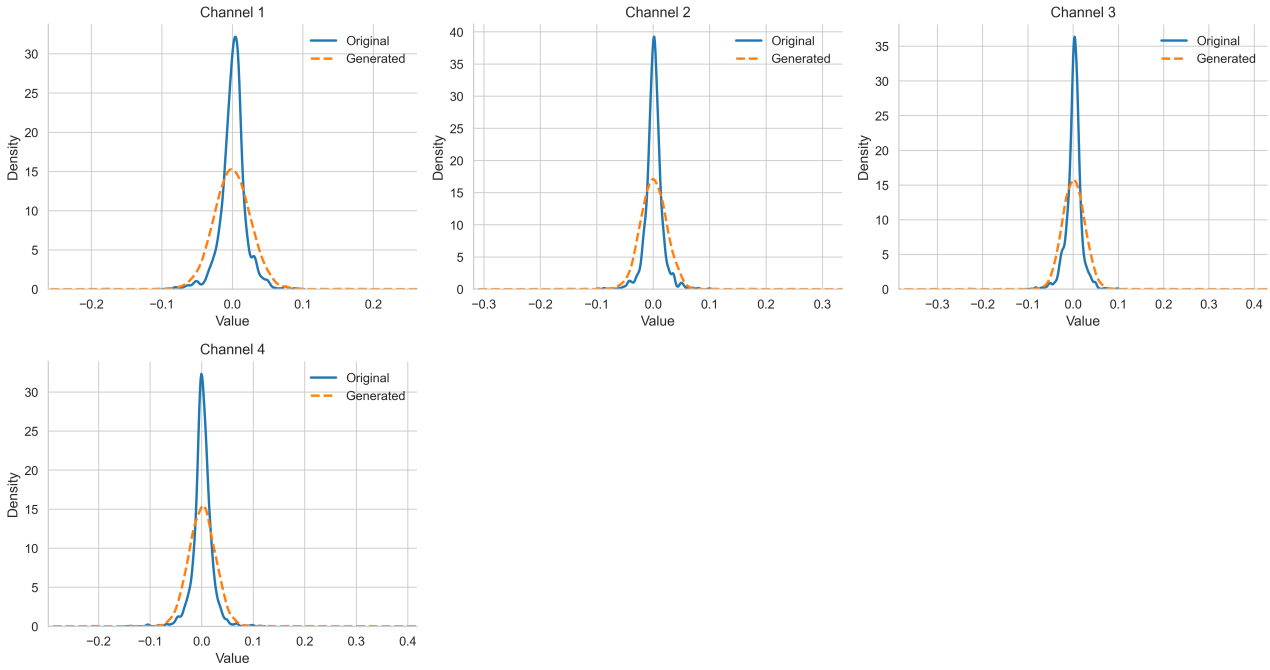

Figure 10: MJD: Channel-wise distributions for real vs. synthetic data.

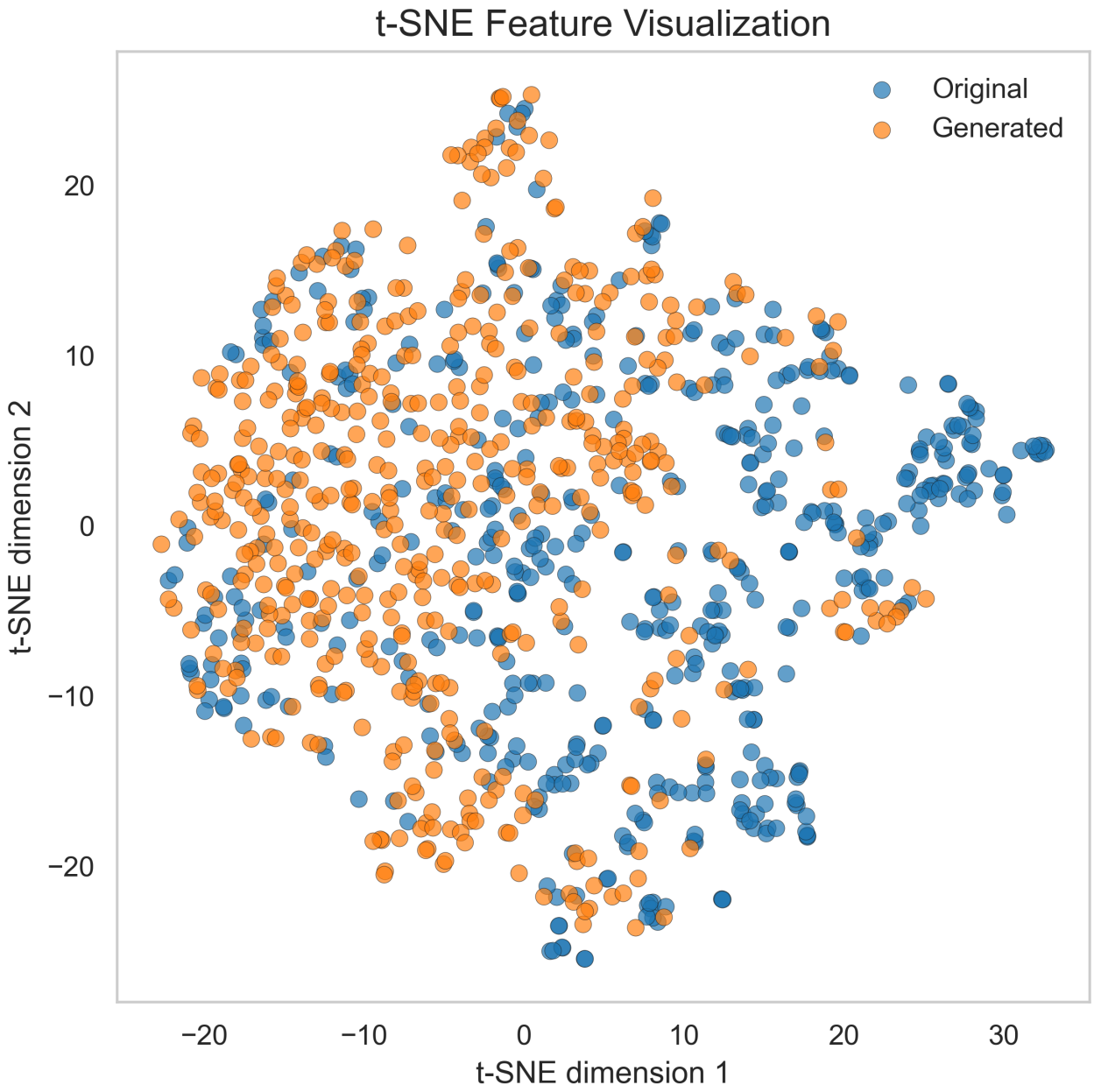

Figure 11: GARCH11: t-SNE embeddings of window-level features.

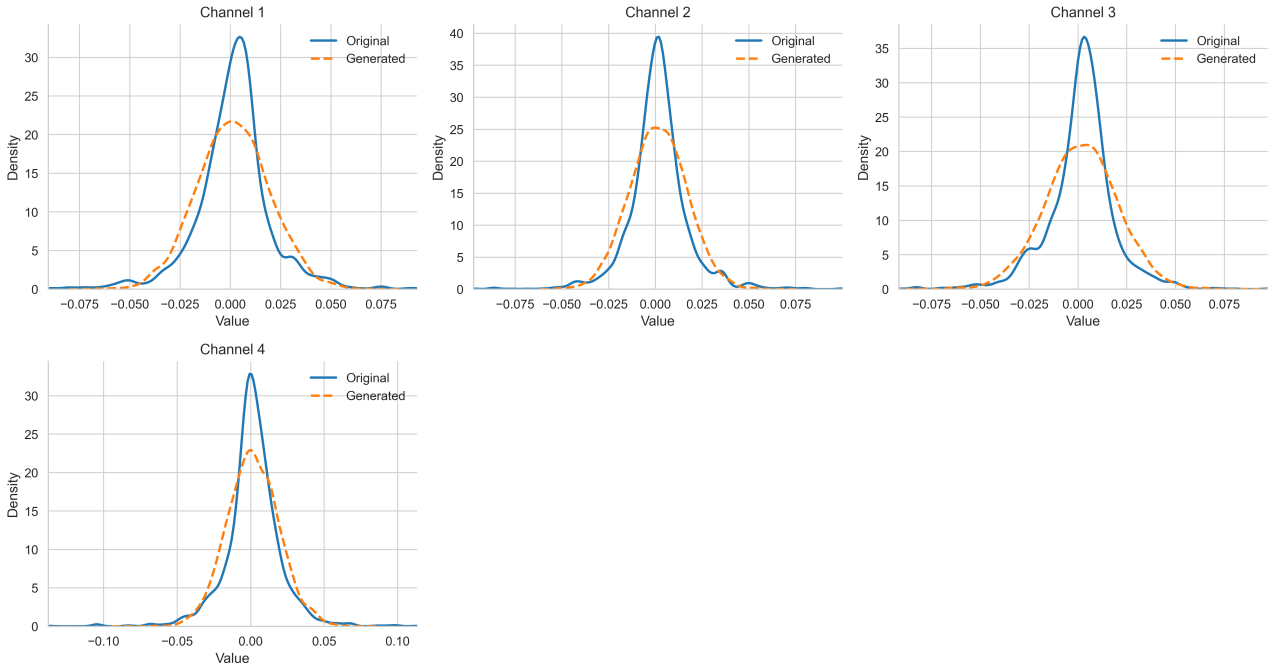

Figure 12: GARCH11: Channel-wise distributions for real vs. synthetic data.

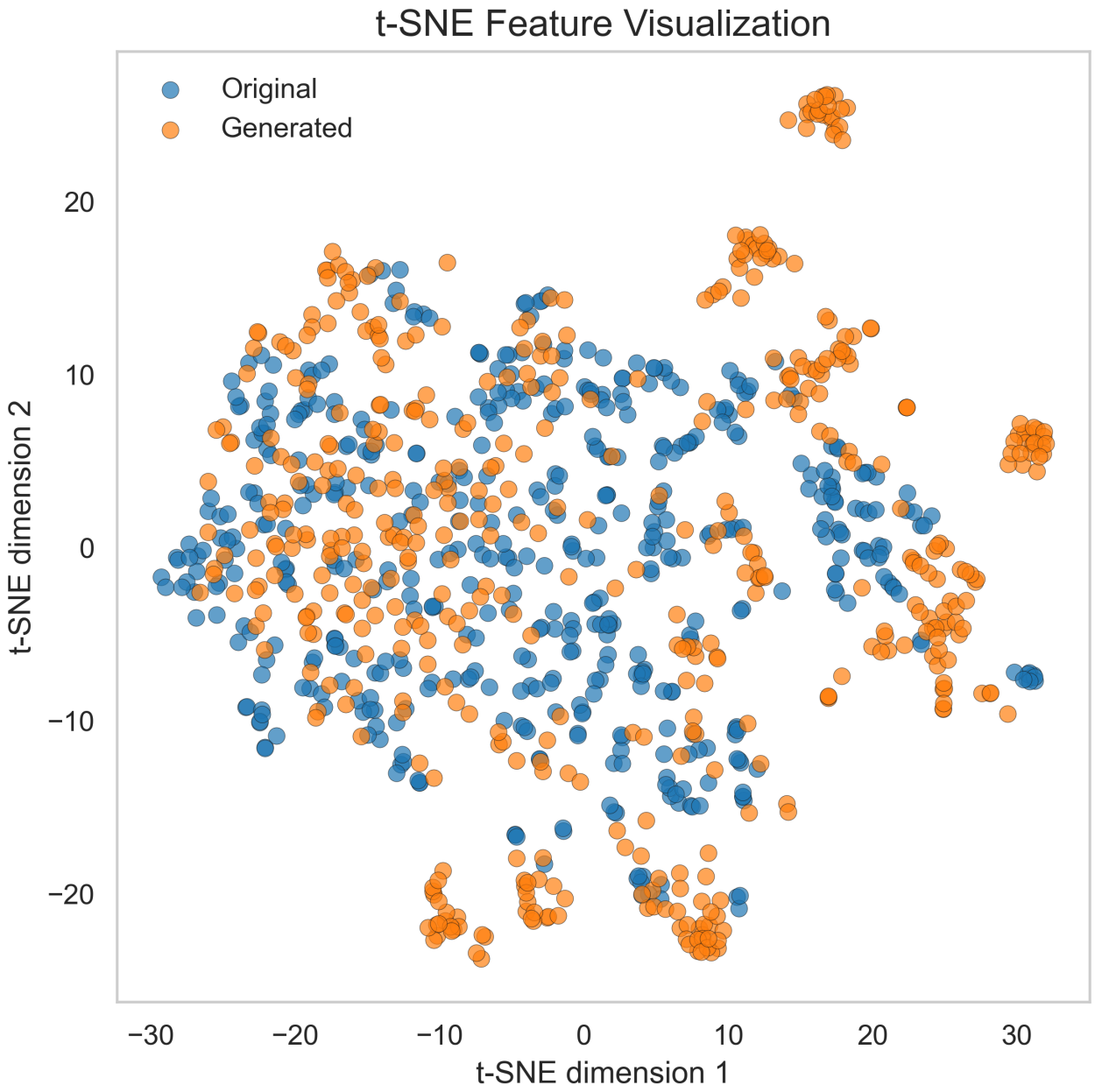

Figure 13: DEJD: t-SNE embeddings of window-level features.

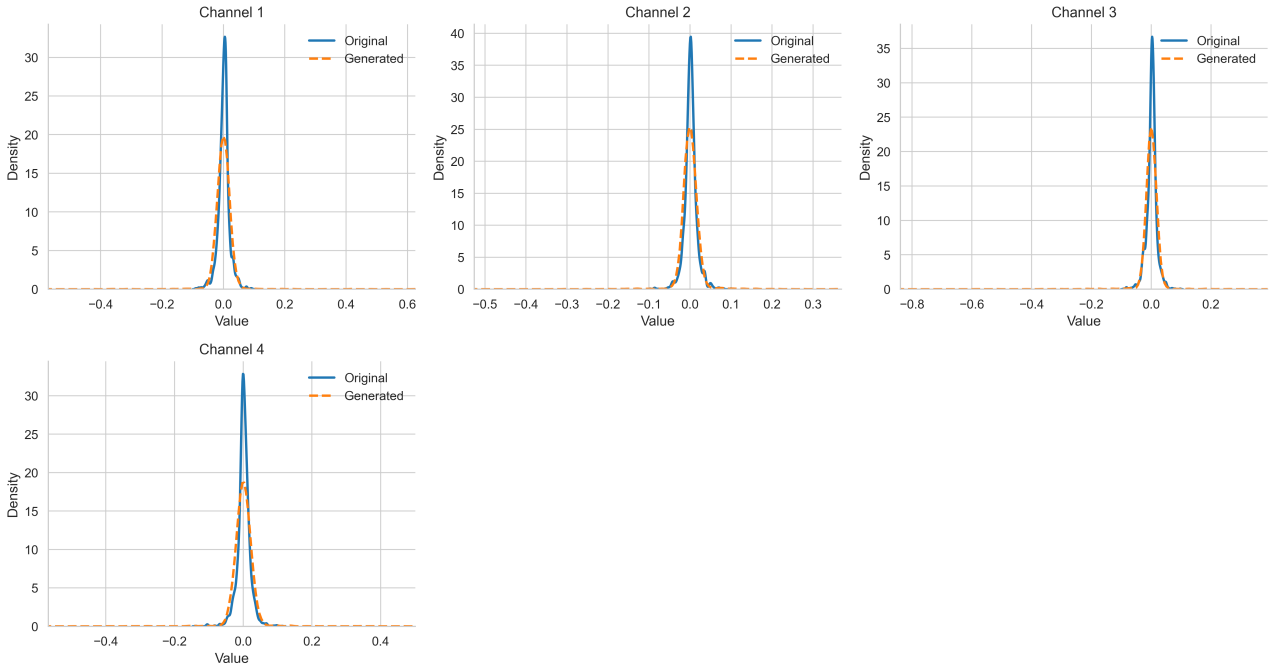

Figure 14: DEJD: Channel-wise distributions for real vs. synthetic data.

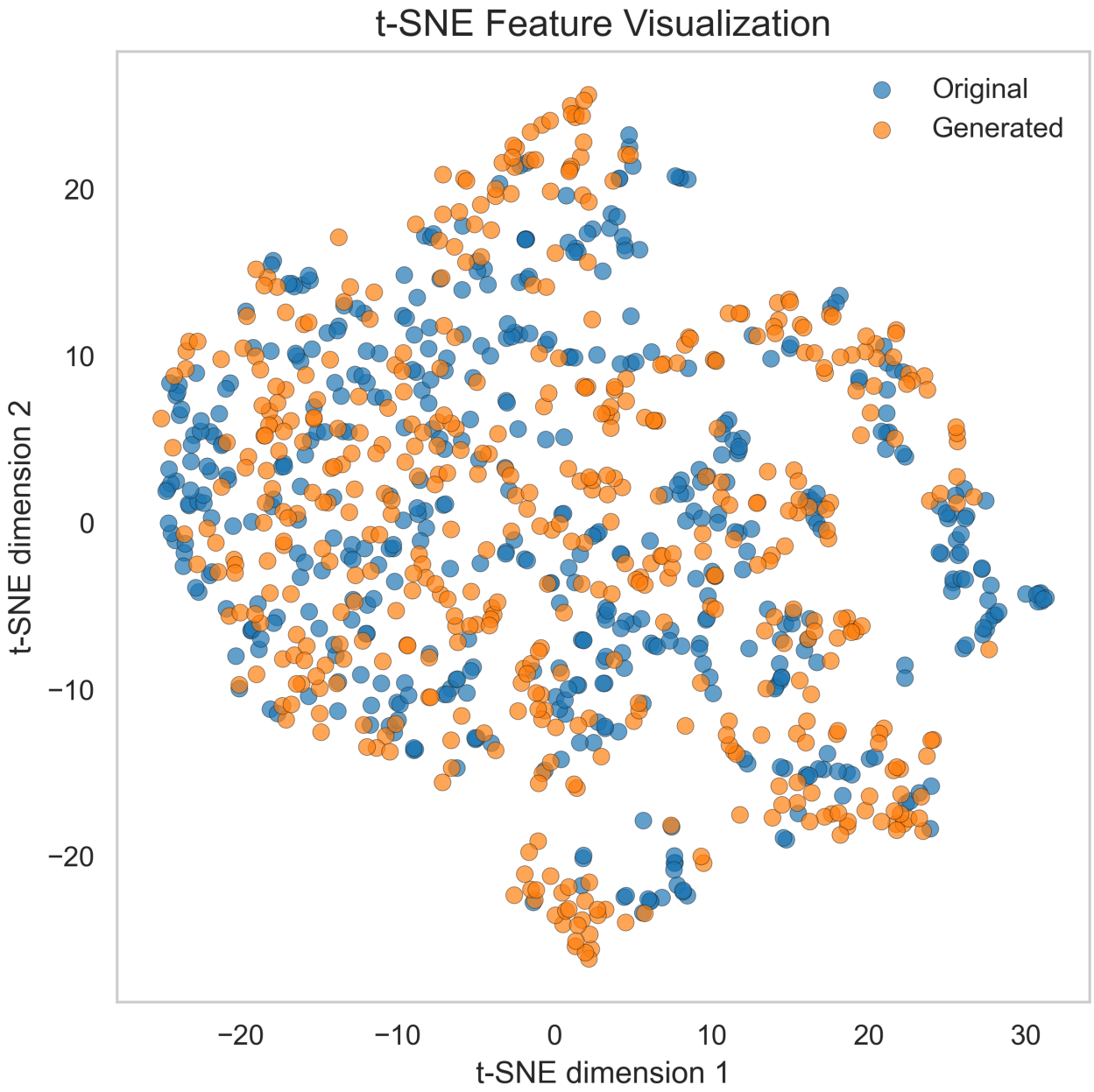

Figure 15: BlockBootstrap: t-SNE embeddings of window-level features.

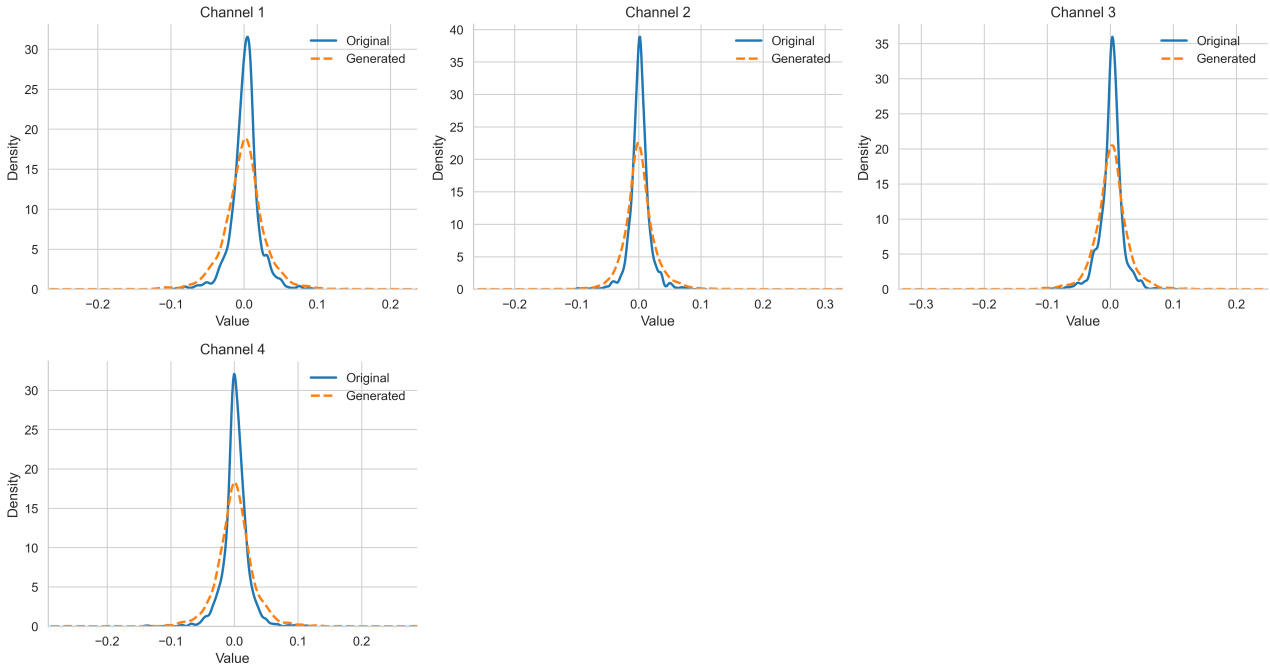

Figure 16: BlockBootstrap: Channel-wise distributions for real vs. synthetic data.

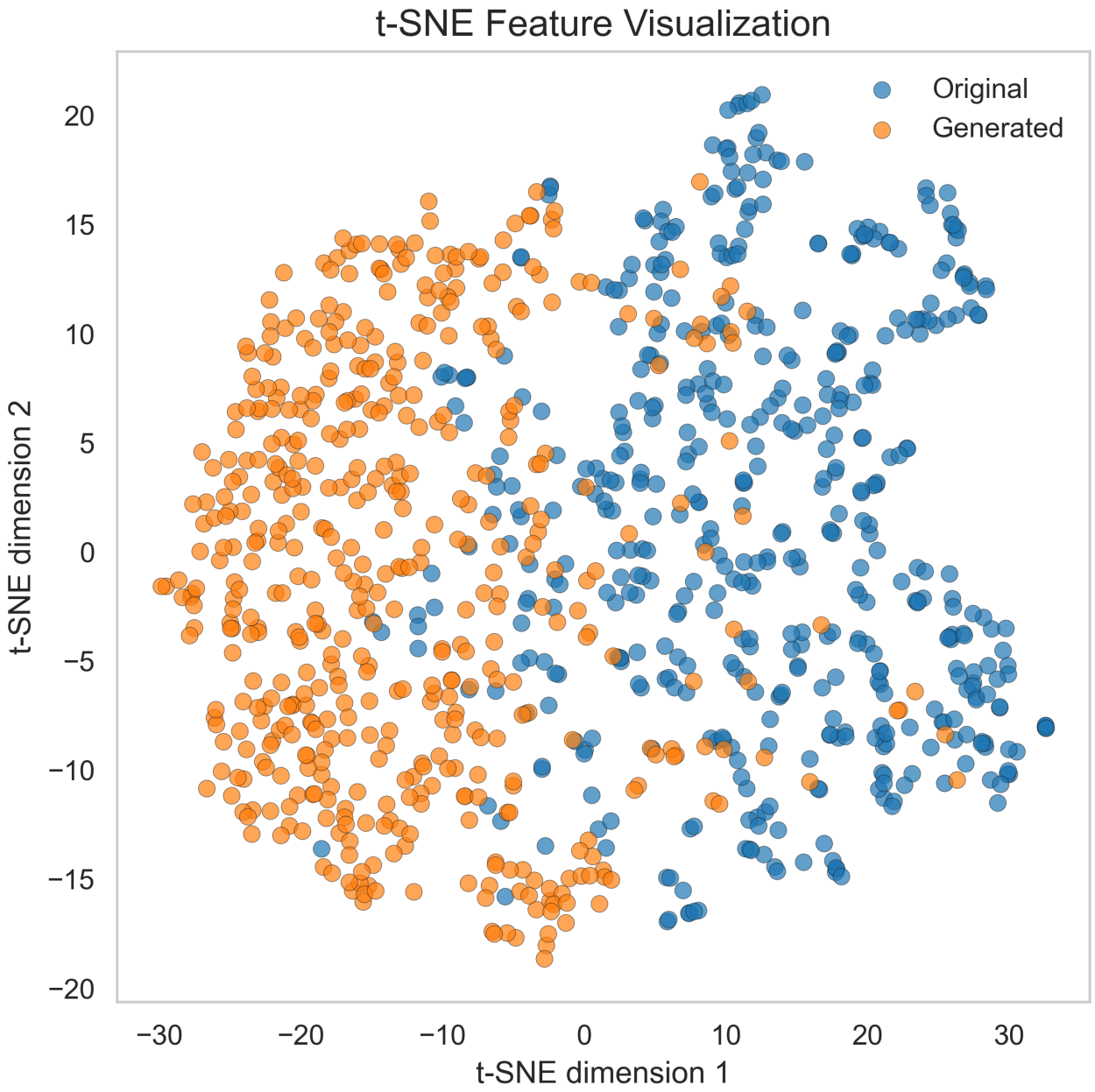

Figure 17: TimeGAN: t-SNE embeddings of window-level features.

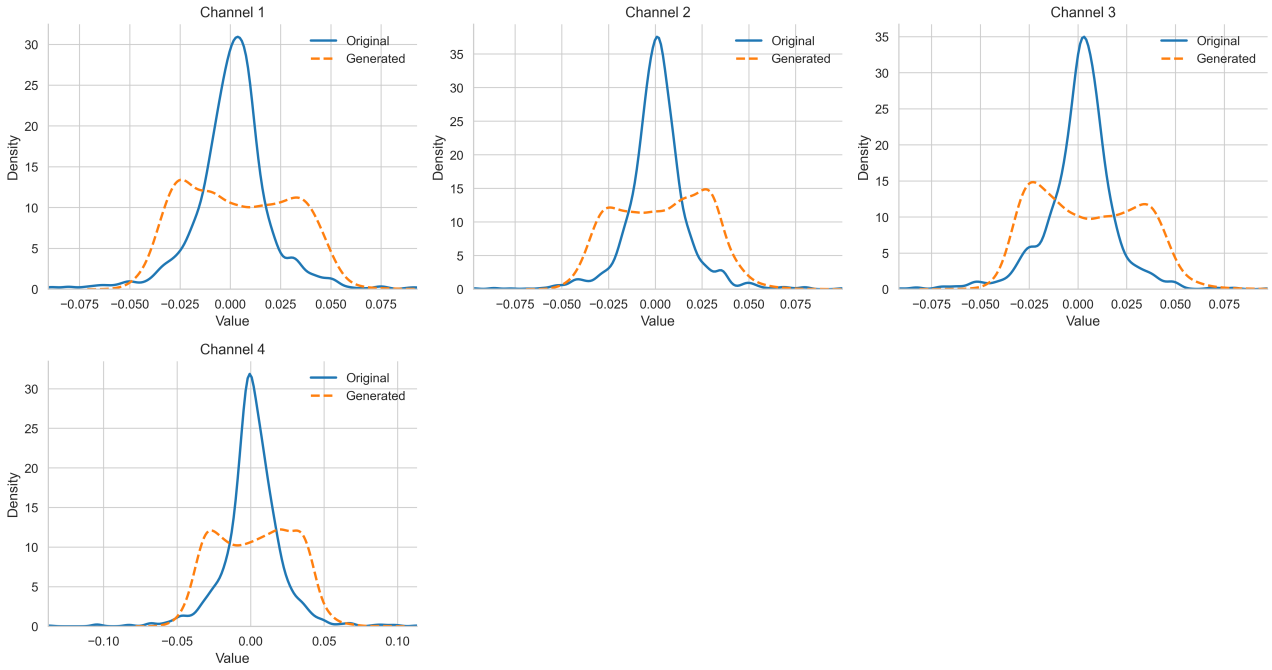

Figure 18: TimeGAN: Channel-wise distributions for real vs. synthetic data.

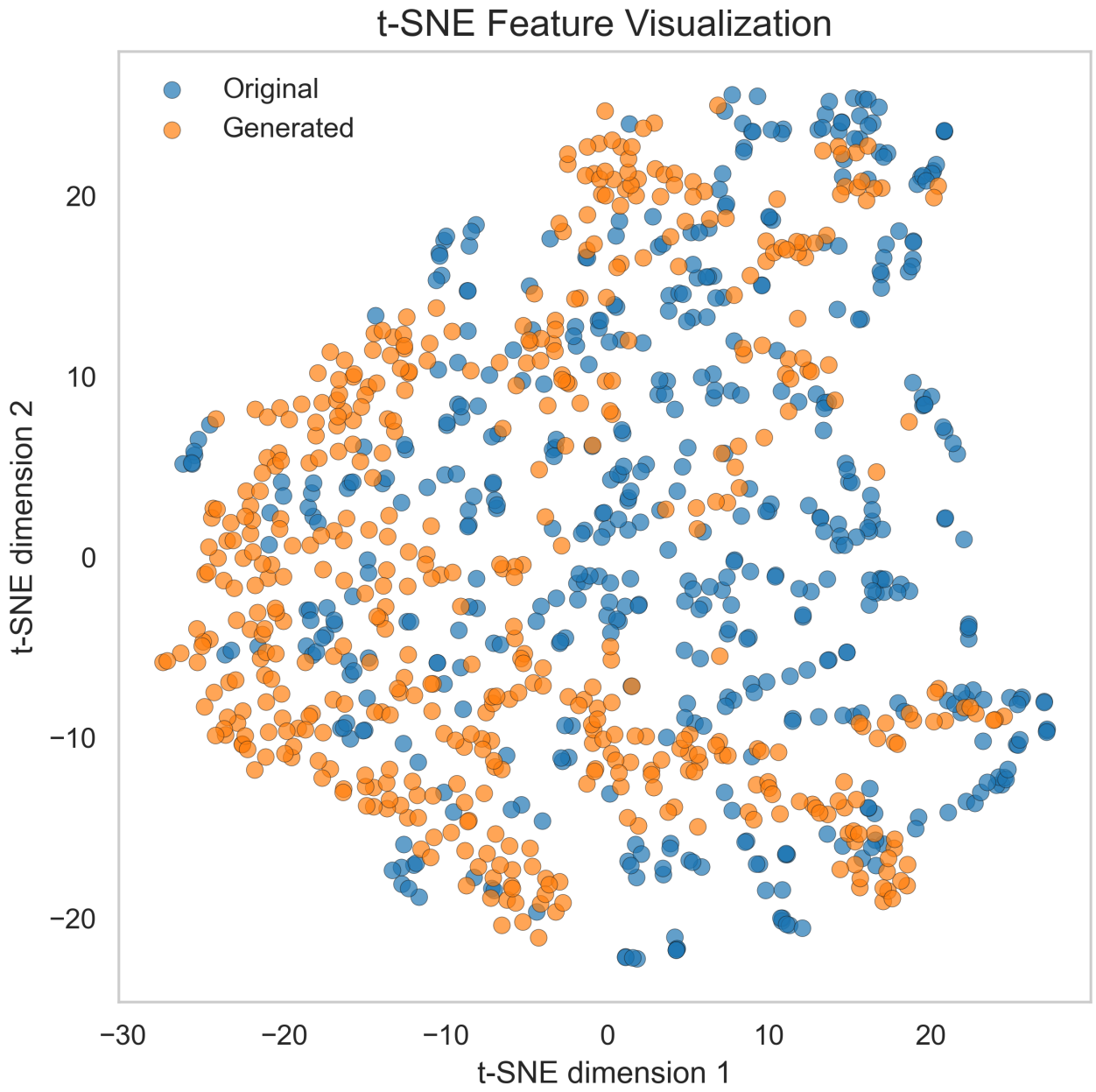

Figure 19: QuantGAN: t-SNE embeddings of window-level features.

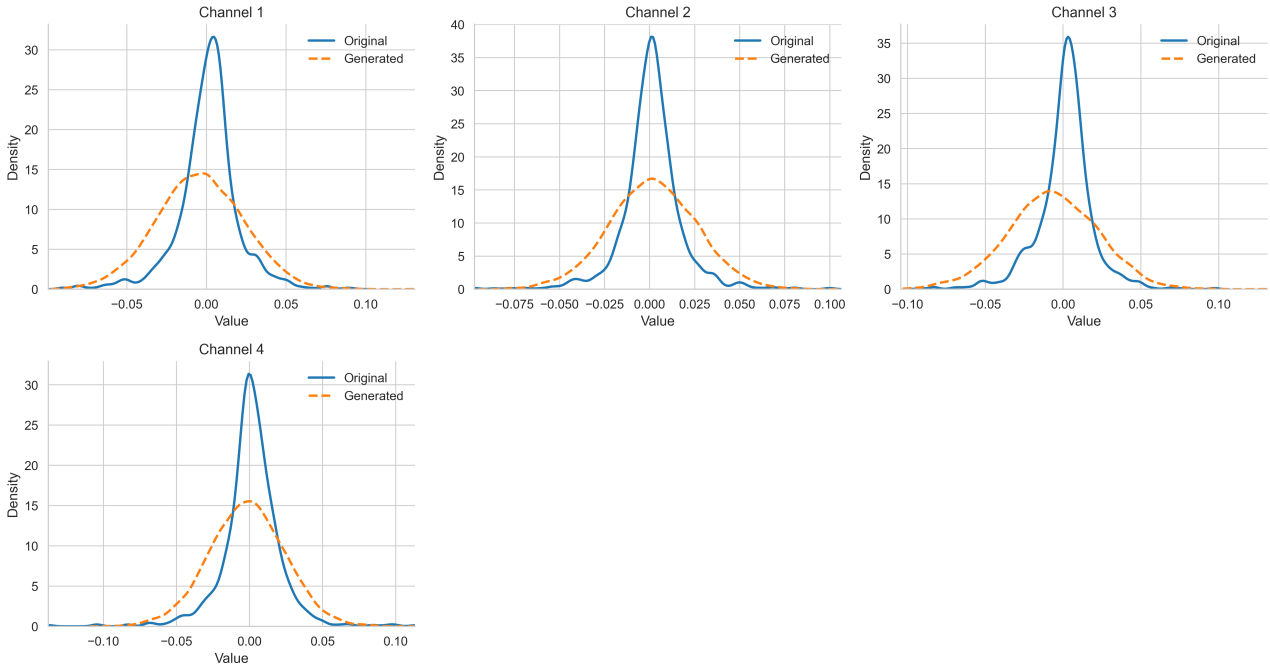

Figure 20: QuantGAN: Channel-wise distributions for real vs. synthetic data.

Figure 21: TimeVAE: t-SNE embeddings of window-level features.

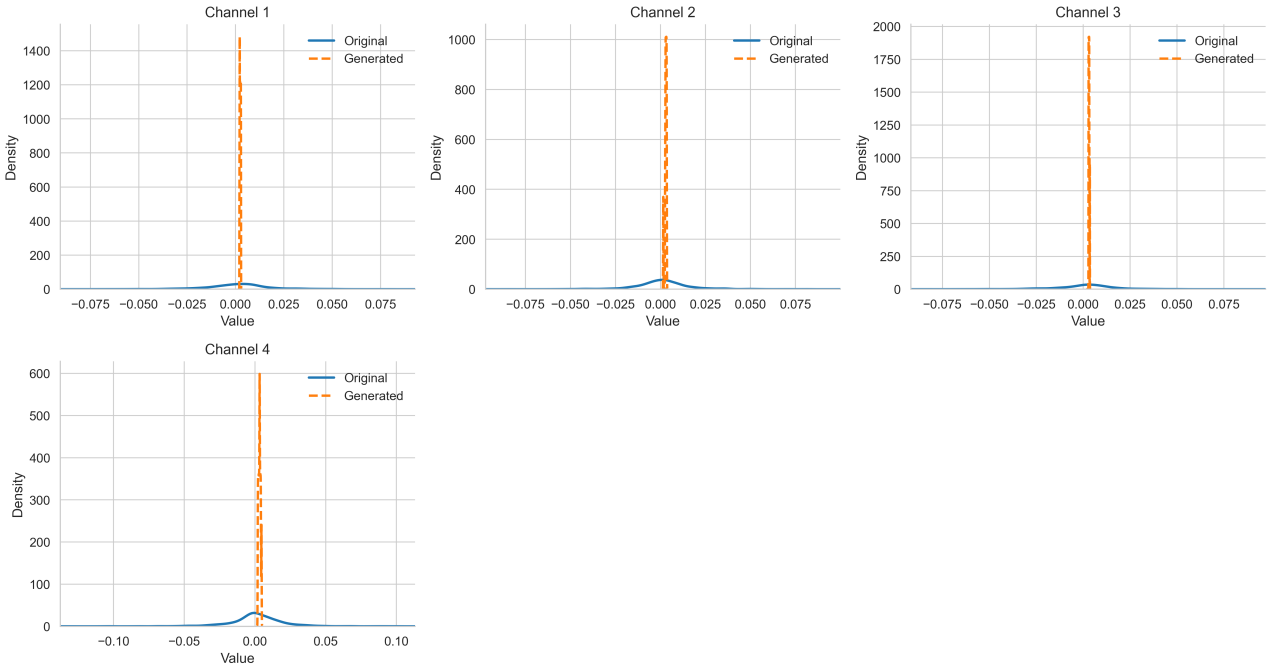

Figure 22: TimeVAE: Channel-wise distributions for real vs. synthetic data.

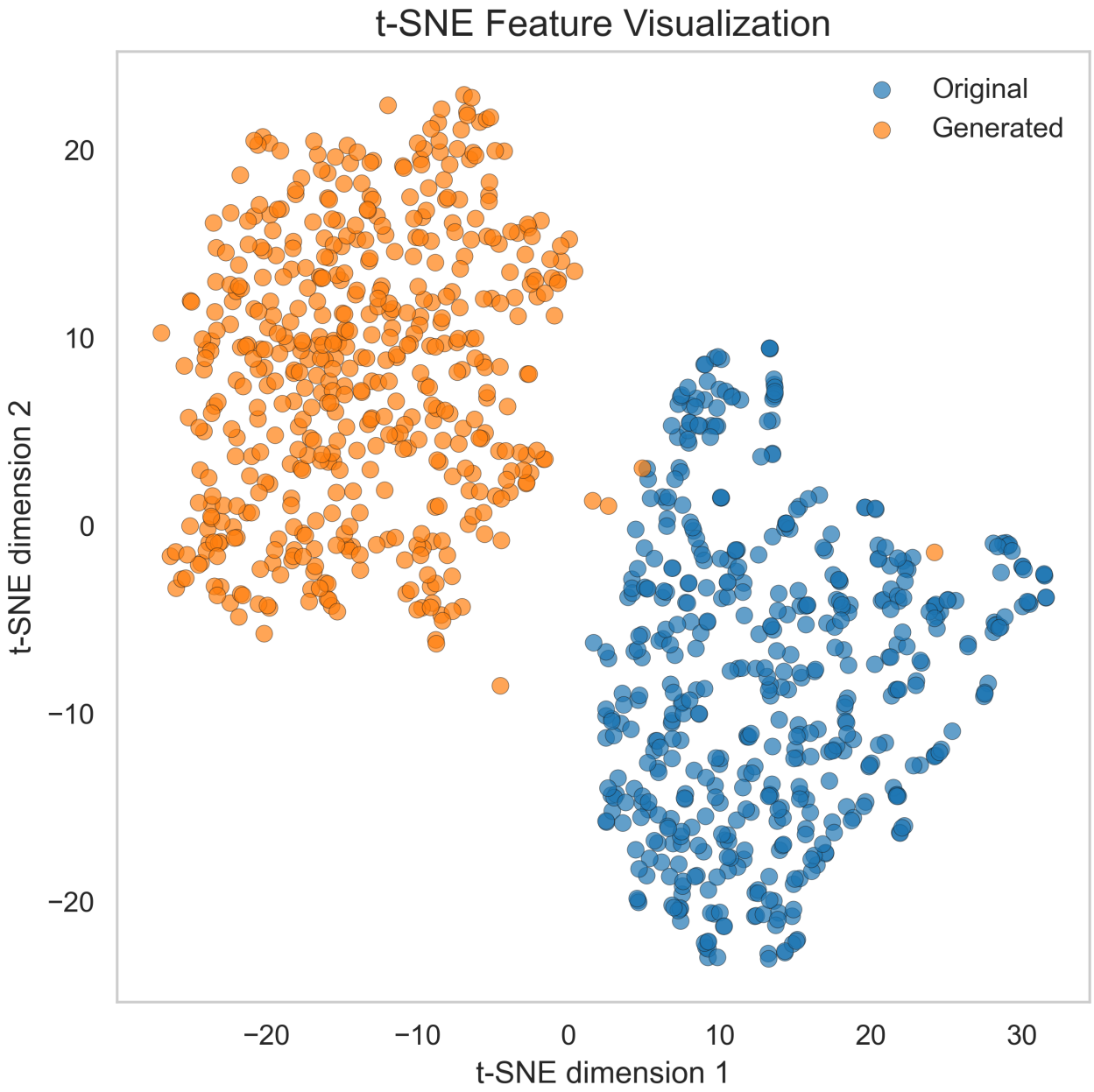

Figure 23: Takahashi: t-SNE embeddings of window-level features.

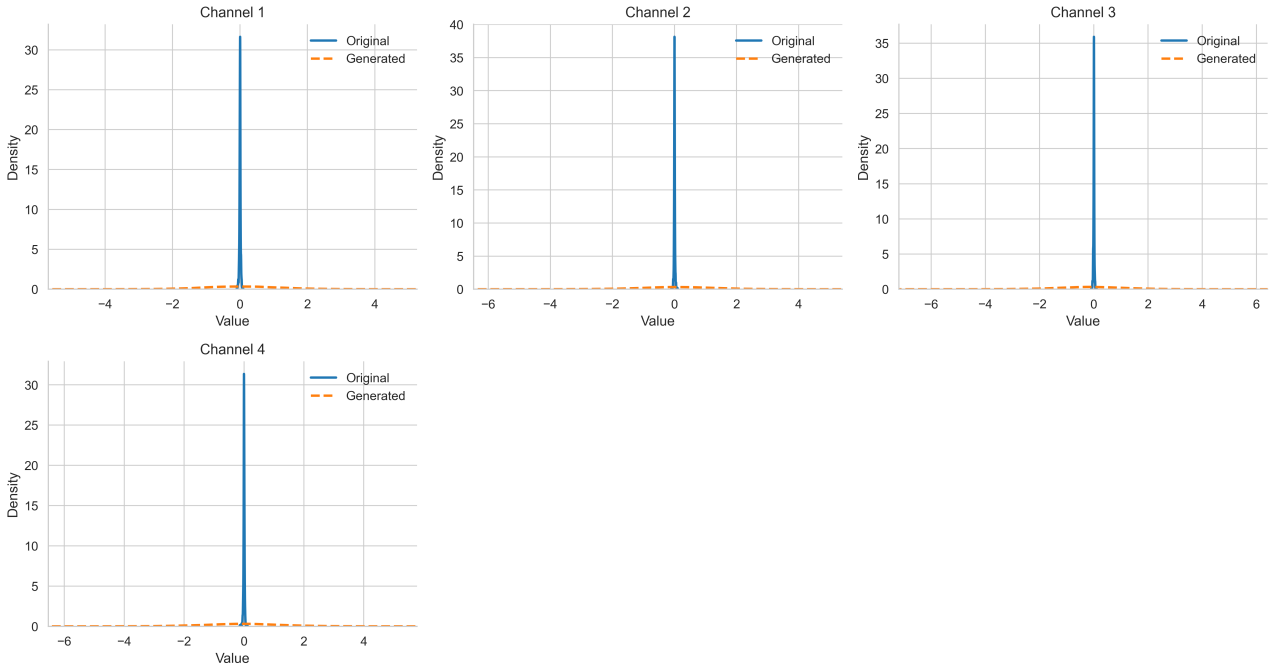

Figure 24: Takahashi: Channel-wise distributions for real vs. synthetic data.

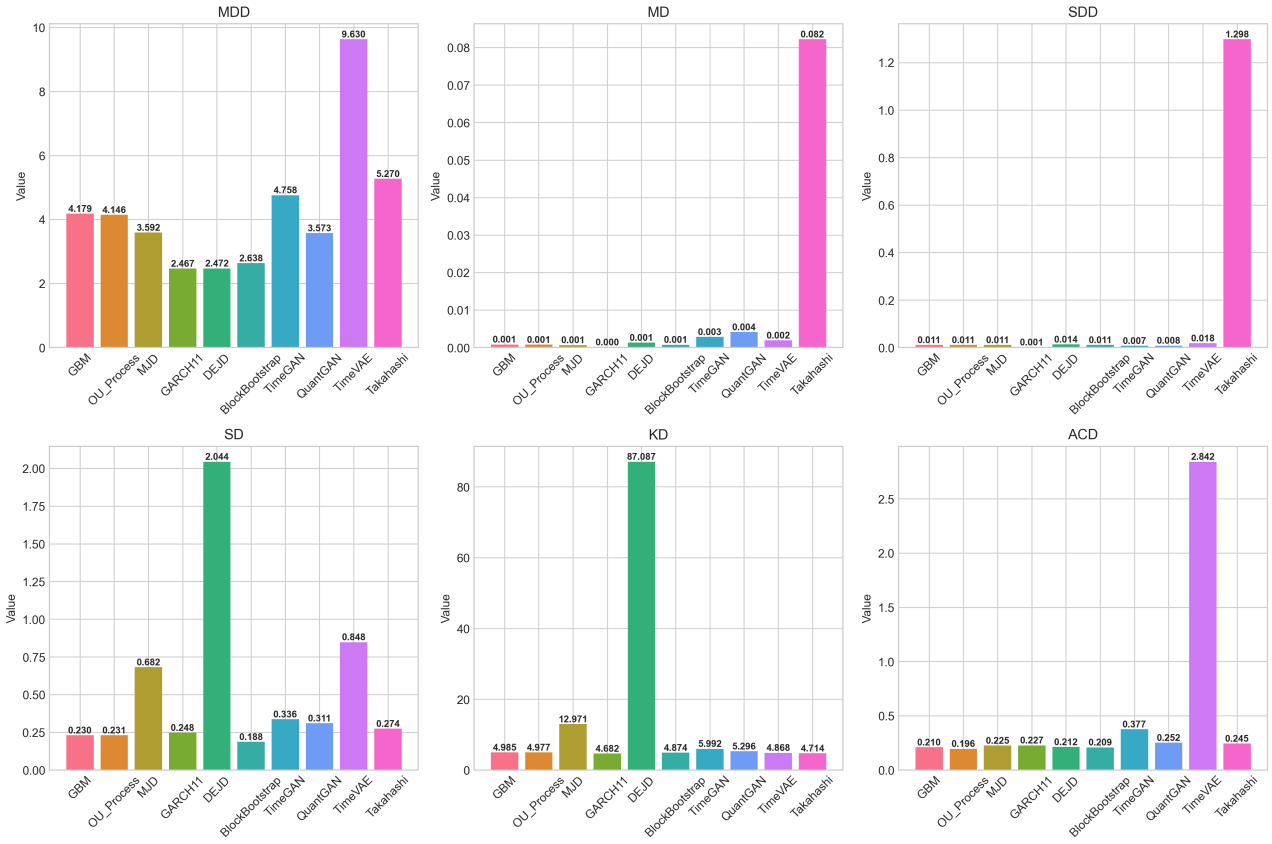

Figure 25: Feature-based evaluation metrics across all models.

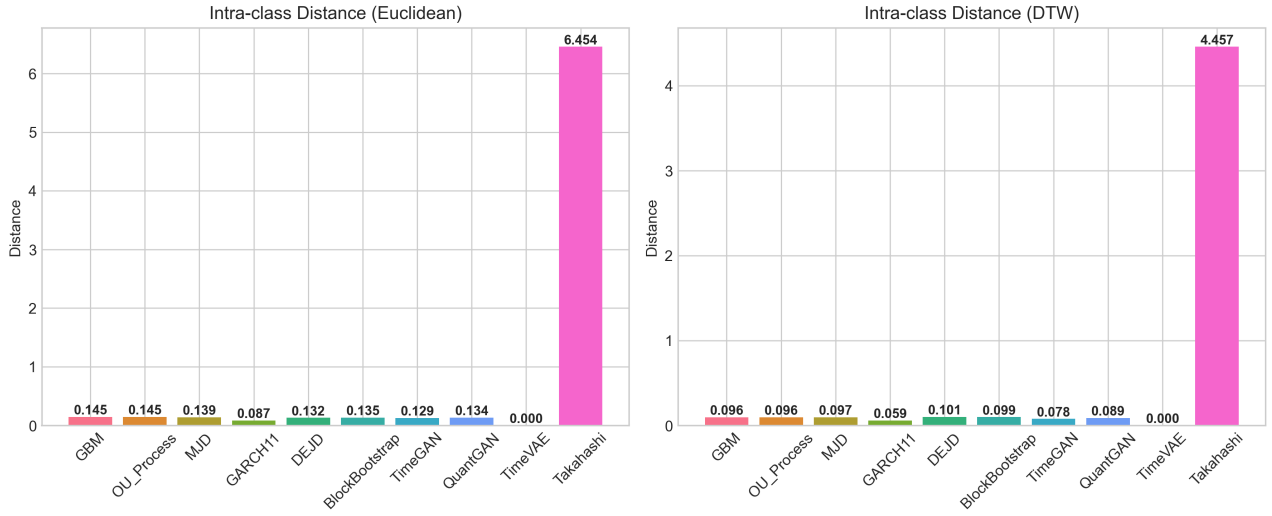

Figure 26: Similarity metrics across all models.

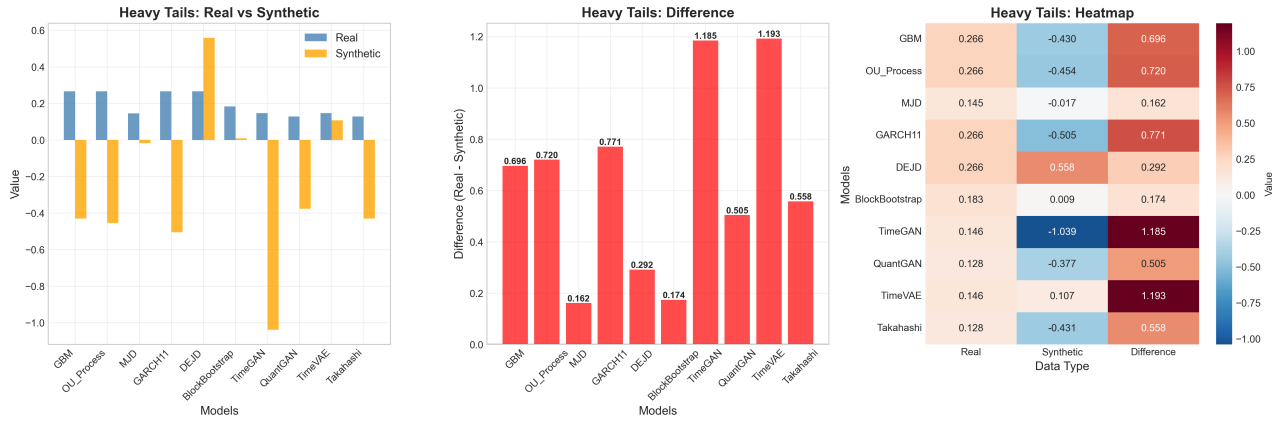

Figure 27: Heavy tails stylized fact comparison across all models.

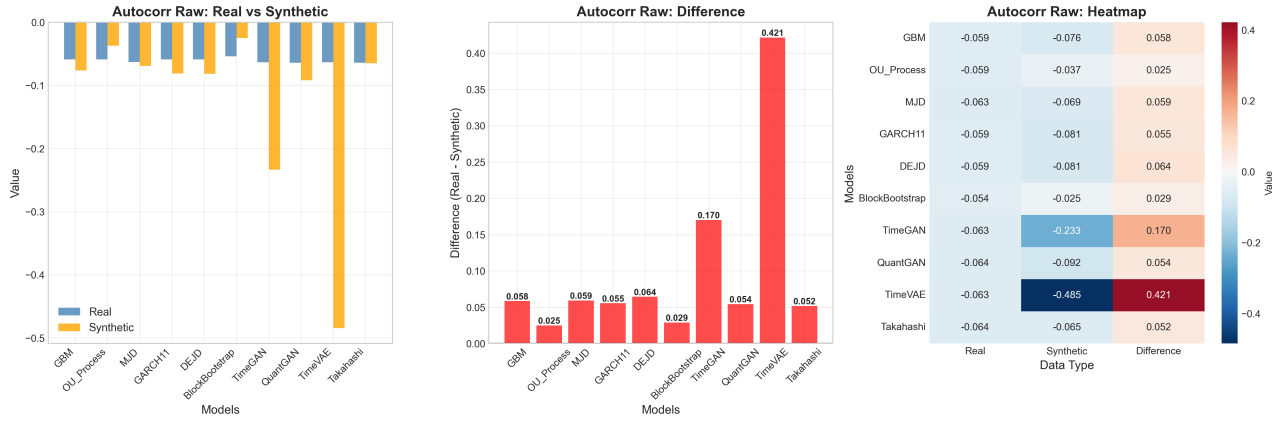

Figure 28: Autocorrelation stylized fact comparison across all models.

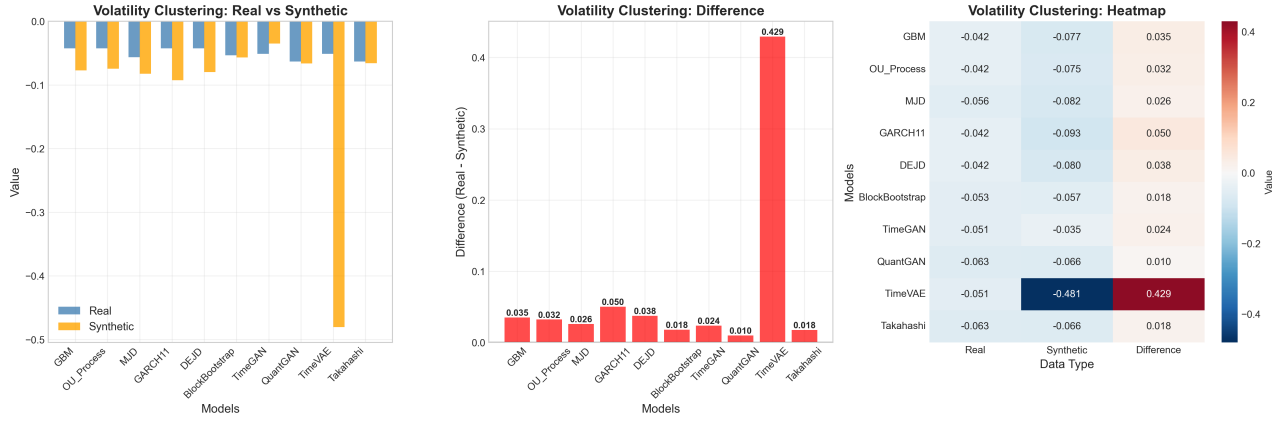

Figure 29: Volatility clustering stylized fact comparison across all models.