SAPIENZA
UNIVERSITÀ DI ROMA

Sapienza University of Rome

Computer Science Department
PhD in Computer Science

Thesis For The Degree Of Doctor Of Philosophy

Analysis and Synthetic Generation of Financial Time-Series

Thesis Advisor
Prof. Novella Bartolini

Co-Advisor
Dr. Viviana Arrigoni

Candidate
Giuseppe Masi
1962771

Academic Year 2025-2026 (XXXVIII cycle)

Abstract

Financial markets generate large volumes of high-frequency data characterized by volatility, heavy-tailed distributions, non-stationarity, and complex asset dependencies. Accurately modeling and predicting financial time-series is, therefore, challenging yet essential for risk management, algorithmic trading, and market stability. This thesis investigates data-driven approaches for understanding, forecasting, and simulating financial dynamics, with particular emphasis on abrupt market shocks, short-term trend prediction, and multivariate temporal dependencies derived from limit order book (LOB) data.

First, a formal framework for stock shock detection and forecasting is introduced, leveraging Lévy-stable modeling and machine learning to anticipate abrupt market movements. Next, a comprehensive benchmark of deep learning models for LOB-based trend prediction reveals significant gaps between experimental performance and real-world generalizability, motivating more robust evaluation practices. To address data scarcity and enable realistic simulations, the thesis proposes generative models that synthesize multivariate time series while preserving correlation dynamics and causal relationships. Finally, it advances robust causal discovery in real-world temporal data exhibiting heavy-tailed behavior.

Together, these contributions provide new methodologies and insights for reliable financial time-series analysis, thereby supporting improved risk mitigation, market understanding, and data-driven decision-making.

Keywords: Financial time-series forecasting, limit order books, machine learning, causality

Contents

Chapter 1

Introduction

Financial markets are complex dynamical systems characterized by high volatility, nonlinear dependencies, heavy-tailed return distributions, and intricate interactions among multiple assets [1, 2]. Understanding, predicting, and simulating market behavior remains a fundamental challenge for researchers, regulators, and practitioners. Accurate modeling of financial data is essential not only for trading and risk management but also for stress testing, market surveillance, portfolio allocation, and the design of robust algorithmic trading and execution strategies that operate under uncertainty and rapidly changing liquidity conditions [3].

Financial datasets can generally be viewed as multivariate time-series describing the joint evolution of prices, returns, volumes, volatility, and liquidity across assets and time. An important example is Limit Order Book (LOB) data, which records the discrete-time (or event-time) evolution of buy and sell orders at multiple price levels. LOB data provide a high-resolution representation of supply-demand dynamics and market microstructure and can be represented as high-dimensional multivariate time-series [4, 5].

Modeling and forecasting financial time-series remain notoriously difficult. Classical time-series assumptions such as stationarity, linearity, and Gaussian noise are routinely violated. Financial series exhibit heavy tails, volatility clustering, long memory effects, structural breaks, and regime shifts [2, 6]. Moreover, markets are influenced by exogenous factors (including macroeconomic news, geopolitical events, regulatory changes, and behavioral responses) that are only partially observable in price data, further complicating inference and prediction [7, 8]. These characteristics contribute to non-stationarity, distribution shifts, hidden dependencies, and extreme events, making financial time-series analysis particularly challenging [9, 10].

The availability of large-scale, high-frequency financial data has created new opportunities for data-driven modeling while simultaneously amplifying these challenges. Modern electronic markets generate massive streams of order-flow data that reflect market participants' behavior at fine temporal scales. Extracting predictive structure from such data requires methods capable of modeling nonlinear relationships, high-dimensional dependencies, and temporal dynamics.

In recent years, the quantitative finance and machine learning communities have increasingly turned to machine learning and deep learning methods to address these challenges. Advances in representation learning, sequence modeling, and generative modeling have enabled improved forecasting, market microstructure modeling, and synthetic data generation [11–14]. These approaches aim to capture complex nonlinear patterns and temporal dependencies that traditional econometric

models often fail to represent.

This thesis investigates the modeling, prediction, and generation of financial time-series through machine learning and deep learning methodologies. The research addresses three major challenges:

- predicting abrupt market shocks and short-term market trends from high-frequency order-book data;
- generating realistic synthetic time-series that preserve correlations and causal relationships;
- improving the robustness of causal discovery methods in real-world time-series exhibiting power-law behavior and noise.

Together, these contributions advance the understanding of financial time-series dynamics and provide tools for risk mitigation, market analysis, and simulation. The thesis is structured around three core research questions, each corresponding to an original contribution.

The **first research question** addresses the challenge of forecasting market shocks, defined as abrupt and significant price movements that can generate substantial losses or create short-lived trading opportunities. While large crashes attract widespread attention, smaller but frequent shocks contribute materially to market risk and execution costs. Despite their practical importance, these events remain difficult to detect and forecast. One key difficulty is the absence of a universally accepted mathematical formulation that formally characterizes what constitutes a "shock" in financial markets. Financial returns exhibit intermittent bursts of volatility, volatility clustering, and regime-dependent dynamics, which obscure clear thresholds separating ordinary fluctuations from extreme events [2]. As a result, shock identification is often heuristic, dataset-dependent, or based on arbitrary thresholds, limiting reproducibility and comparability across studies.

This thesis fills this gap by introducing a formal, statistically grounded definition of stock market shocks. The proposed definition builds on the empirical observation that asset returns are better described by heavy-tailed distributions than by Gaussian models. Specifically, shocks are identified using Lévy-stable distributions, which provide a principled framework for modeling extreme deviations and tail risk. Building upon this definition, the thesis develops a machine learning framework that leverages high-frequency limit order book features to anticipate shock events before they occur.

The **second research question** examines the extent to which deep learning methods can be reliably used for forecasting short-term price movements from high-frequency limit order book (LOB) data. While numerous studies report strong predictive performance [14, 15] in controlled experimental settings, their real-world applicability remains uncertain due to overfitting, dataset bias, regime changes, and sensitivity to market microstructure variations.

To address this gap, this thesis formulates the problem as a systematic benchmarking study and asks the following question: "To what extent can deep learning models effectively forecast financial time-series derived from LOB data, and which architectural mechanisms provide the greatest robustness and predictive efficiency?" Rather than proposing a single predictive model, the thesis evaluates representative state-of-the-art architectures, including convolutional, recurrent, and attention-based models, under realistic market conditions. A modular benchmarking framework is introduced to assess predictive accuracy and generalizability across assets and time periods. The results highlight the discrepancy between in-sample predictive success and real-world deployment performance, identify architectural components that contribute most to robustness, and provide practical guidance for developing more reliable LOB-based forecasting systems.

The **third research question** addresses the generation of synthetic financial time-series that

preserve realistic dependency structures across assets. Access to high-quality financial data is often restricted due to proprietary, regulatory, or privacy constraints, limiting reproducibility and hindering the development and validation of data-driven methods. Synthetic data generation offers a practical alternative by enabling stress testing, scenario analysis, and model training without exposing sensitive information.

Recent advances in generative modeling (including variational autoencoders (VAEs), generative adversarial networks (GANs), and diffusion-based models) have demonstrated the ability to learn complex high-dimensional distributions and reproduce realistic data characteristics [16–18]. In finance, generative models have been explored to reproduce stylized facts such as heavy tails, volatility clustering, as well as temporal dependence essential for portfolio risk analysis [2, 19]. However, generating multi-asset time-series that faithfully preserve dynamic correlations and co-movements remains a significant challenge.

This thesis proposes a generative adversarial framework for synthesizing correlated financial time-series. Unlike approaches that model assets independently, the proposed method explicitly captures correlation dynamics, enabling the generation of multi-asset sequences that reproduce realistic joint behavior. Such synthetic datasets support portfolio simulation, risk management, stress testing, and benchmarking of forecasting methods.

At this stage, the thesis moves beyond correlation-based modeling toward a deeper investigation of causal structure in time-series. While correlations describe statistical co-movements, they do not reveal the direction of influence or the mechanisms through which information propagates across variables. Causality, in contrast, seeks to identify directed relationships and dynamic mechanisms that govern how changes in one variable influence another over time [20, 21]. Understanding causal structure is essential for explainability, decision-making, policy analysis, and counterfactual reasoning, where the goal is not merely to predict outcomes but to understand how interventions or shocks propagate through a system.

Most existing approaches to time-series modeling in finance and other domains focus on predictive accuracy or correlation structure, while causal discovery methods for temporal data remain challenging due to noise, hidden confounders, non-stationarity, and high dimensionality [22, 23]. Recent advances in causal discovery and generative modeling have explored the integration of structural causal models with deep learning, yet current methods often assume simplified dynamics, limited noise structures, or static causal graphs, restricting their applicability to complex real-world systems.

This thesis addresses these limitations by proposing a diffusion-based generative framework for producing multivariate time-series with explicit causal dependencies. The approach reconstructs underlying causal graphs and embeds causal mechanisms directly into the data-generation process, enabling the synthesis of time-series in which information flow and dynamic dependencies are structurally encoded rather than inferred post hoc. Such synthetic data provide a controlled environment for benchmarking causal discovery algorithms, enabling systematic evaluation under known ground-truth causal structures.

Importantly, the objective is not to perform causal discovery on financial markets themselves — where ground truth is inherently unobservable and confounding factors are pervasive — but to develop a domain-agnostic framework for generating time-series with known causal structure.

The thesis proceeds by investigating a fundamental complementary question: how reliable are causal discovery methods when confronted with the statistical complexities of real-world data?

Building on this foundation, the thesis investigates the robustness of causal discovery methods when applied to time-series exhibiting power-law behavior, strong noise components, and non-stationarity. In particular, it examines how causal relationships are preserved or distorted under spectral transformations and proposes methodologies to improve causal structure recovery under such conditions.

The remainder of this thesis is organized as follows:

- Chapter 2 introduces a formal model for stock shocks and presents a machine-learning framework for shock forecasting.
- Chapter 3 benchmarks deep learning models for limit order book-based price trend prediction and analyzes robustness and generalizability.
- Chapter 4 presents a generative adversarial framework for synthesizing correlated financial time-series.
- Chapter 5 introduces a diffusion-based approach for generating time-series with causal dependencies.
- Chapter 6 investigates robust causal discovery in real-world time-series exhibiting power-law behavior.
- Chapter 7 concludes the thesis, summarizing key findings, discussing limitations, and outlining future research directions.

The contribution of this thesis has been published in five research papers (one workshop [24], one conference [25] and two journals [26, 27], while one paper is currently under review).

In addition to the research presented in the main content of this thesis, my doctoral work also included contributions to topics beyond financial time-series analysis. In collaboration with the Networking and Artificial Intelligence research team, I investigated optimization problems arising in modern communication and computing infrastructures, which are included in the Appendix for completeness.

One line of work focused on aerial networks, where unmanned aerial vehicles and flying ad-hoc networks are used for persistent monitoring and surveillance [28]. This research addressed the problem of efficiently patrolling heterogeneous targets with different visit-frequency requirements, proposing scalable heuristics and performance metrics to improve monitoring effectiveness in security-critical scenarios.

A second line of work addressed optimization in geographically distributed computing infrastructures supporting large-scale AI services. In this context, I studied the real-time dispatching of AI inference workloads across heterogeneous data centers, proposing algorithms that jointly optimize throughput, latency, accuracy, and carbon emissions. This work contributes to the emerging field of sustainable and carbon-aware AI systems.

Chapter 2

Stock Shocks modeling and Forecasting

Abstract

Stock shocks are abrupt changes in a stock price time series. Major shocks, such as the 1929 Big Crash and, more recently, the subprime mortgage crisis, have greatly influenced the world economy in the last century. Nevertheless, shocks of minor intensity are frequent events that often go unnoticed but whose prediction can significantly help an investor to protect their investments. In this chapter, we provide a formal definition of stock shocks and use limit order-book data to implement an algorithm to forecast stock shocks. The proposed algorithm is built upon a formal mathematical model approximating stock prices and return distributions with fat-tailed Lévy-stable models. A preliminary study of different machine-learning approaches allowed us to design an algorithm based on Random Forest, hierarchical clustering, and Bayesian optimization that outperforms other machine-learning methods that exhibit tunable and higher Precision and Recall values.

This work has been published in the 2023 IEEE 43rd International Conference on Distributed Computing Systems Workshops (ICDCSW) [24].

2.1 Introduction

Financial crashes are recurring events that have influenced the economic history of the past decades. Different factors can trigger them, be they purely economic, such as the subprime mortgage crisis of 2008, which led to the bankruptcy of the Lehman Brothers bank, or sociopolitical, such as the COVID-19-induced recession of the past three years. However, minor crashes (e.g., inconsistencies in penny stocks or cryptocurrencies with a small market cap) often go unnoticed by the media as they are irrelevant in a global economic landscape. Forecasting short-term stock shocks (i.e., abrupt alterations in a stock's price) can help investors take action to protect their portfolios or trade for capital gain. By identifying potential shocks before they happen, traders can adjust their positions or even exit the market altogether to minimize their exposure to the risk. On the other hand, the capability to forecast market shocks can serve as a profitable trading opportunity for investors. Anticipating future price trends resulting from such shocks can enable traders, especially those working at high frequency [29], to make informed trading decisions. In this work, we develop a new tool for technical analysis, i.e., for examining past price patterns to identify market trends and trading opportunities, under the assumption that prices move in trends and that analyzing these

trends can provide insights about future market activity. Some authors have studied how the fat-tailed Lévy-stable distributions can model returns in the stock market [30, 31]. Yet, none provided a programmatic and robust way of defining and forecasting shocks, exploiting this model. For the first time, we use the theory of Lévy-stable distributions for market trend modeling to provide a formal definition of shocks and to create a baseline for shock forecasting for future research. Our mathematical model is validated by an experimental campaign where we test different machine learning algorithms for shock prediction. From a preliminary test phase, we observe that the best-performing model among the tested approaches is the Random Forest algorithm. Based on this observation, we formalize an optimized model incorporating hierarchical clustering for relevant feature selection and a hyperparameter tuning module based on Bayesian optimization. We run our tests on the Lobster dataset [32], which provides order-book and order message data with up to nanoseconds time granularity.

The major contributions of this chapter are the following:

- We give a formal definition of shocks in a stock market, which exploits Lévy-stable distributions to approximate the trend of observed returns. Lévy-stable distribution's parameters are included in the set of extracted features for stock shocks forecasting.
- We use our shock characterization for labeling a real NASDAQ dataset extracted from the LOBSTER archive [32].
- We evaluate different machine-learning-based predictors, using Random Forests (RF), Multi-Layer Perceptron (MLP), and Hierarchical Density-based Spatial Clustering of Applications with Noise (HDBSCAN). By means of experiments, we highlight the superiority of the Random Forest approach in terms of precision and recall.
- We suggest an algorithmic approach to forecast market shocks, which involves combining Random Forests, Hierarchical Clustering, and Bayesian Optimization. We emphasize the potential of this approach to safeguard investors from risks and facilitate access to profit opportunities.

2.2 Related work

Shock stock prediction and modeling have gained significant interest in recent years, particularly with the proliferation of algorithmic and high-frequency trading [29], which began in the early 2000s. For example, the work in [33] proposes a framework for investigating trend anomalies explainability among different stocks. In [34], prices and shocks are estimated by exploiting the well-known ARMA-GARCH model. Then, the optimal subset of features involving time-series elements and technical indicators is extracted and classified to predict the shock's sign and value. Unlike our approach, shocks are defined as a function of the trend estimation error, and returns are generated synthetically. The authors of [35] empirically find positive correlations between social attention and returns trend. In [36], several machine learning techniques (Random Forest, MLP, CNN, LSTM) are tested to predict daily returns by observing a window of past daily returns. A return is said to be "significant" if it exceeds a threshold that considers the standard deviation of the returns observed over the training set. Different from our solution, this work does not provide a model for shocks and only classifies whether the predicted return sensibly differs from previous ones on a daily basis.

Another research line studies how Lévy-stable distributions can model returns in the stock

market. Some authors are only interested in the goodness of fitting Lévy-stable distributions to the stock market [30, 31]. At the same time, recent researchers focused on studying the correlation between the Lévy-stable distribution of the returns and financial crashes [37]. However, this line of research only shows correlations between crashes and changes in the parameters defining these distributions and only considers major events happening in an extensive time frame. To the best of our knowledge, this method is the first to exploit the Lévy-stable distribution model to forecast stock shocks with a machine-learning approach.

2.3 Model for shocks

The *random walk hypothesis* (theorized for the first time in 1900 by Bachelier [38]) states that stock market prices evolve according to a random walk. Under this hypothesis, the trend of stock market prices is modeled as a stochastic process $\{S_t : t \in \mathbb{N}\}$ following the normal distribution. This assumption has been used for describing market trends both for its simplicity and for the idea that the central limit theorem provides a good model for price fluctuation [39, 40]. The assumption that stock prices are normally distributed was first questioned in [41]. As a matter of fact, the inherent fluctuations of the market and the presence of shocks may cause abnormal stock returns, which cannot be captured by a normal distribution whose tails decay exponentially. Therefore, high variance fluctuations and sharp stock shocks would appear as outliers of the normal distribution, despite them being increasingly frequent in the short term and peculiar to market trends. Fat-tailed distributions, such as the Lévy-stable distributions with power law-trended tails, are therefore more suitable for modeling systems that exhibit significant volatility, including stock price trends (e.g., [30, 37]). We aim to exploit the modeling of stock returns based on Lévy-stable distributions to provide stock shock predictors in Section 2.4.

Definition 2.3.1 ([31])**.** *A probability distribution is Lévy-stable, say* $\sim \mathcal{S}(\alpha, \beta, \gamma, \delta)$*, if, being* X *a random variable following such distribution, then for any* X_1 *and* X_2 *independent copies of* X*, it holds that:* $aX_1 + bX_2 \overset{d}{=} cX + d$*, where* $\overset{d}{=}$ *means equality in distribution, and* $\forall a, b > 0$*, for some* $c > 0$ *and* $d \in \mathbb{R}$*.*

Four parameters characterize Lévy-stable distributions: α, representing the stability parameter; β, representing the skewness; γ, the scale parameter; and δ, the shift parameter. The general expression for the probability density function of a random variable $X \sim S(\alpha, \beta, \gamma, \delta)$ is not analytical. For this reason, it is expressed by a characteristic function $\phi(X)$, as we highlight in the following definition:

Definition 2.3.2. *Let* X *be a random variable following a Lévy-stable distribution. Its probability density function is defined as* $f_X(x) = \frac{1}{2\pi}\int_{-\infty}^{\infty} \phi(k)e^{-ixk}dk$*, where* $\phi(k)$ *is its characteristic function, which is defined as* $\phi(k) = exp\{i\delta k - |\gamma k|^{\alpha}[1 + \beta sign(k)\omega(k, \alpha)]\}$*, being* $\omega(k, \alpha) = \tan(\pi\alpha/2)$ *for* $\alpha \neq 1$ *and* $\omega(k, \alpha) = 2/\pi \log|k|$ *for* $\alpha = 1$*, with* $\alpha \in (0, 2]$*,* $\beta \in [-1, 1]$*,* $\gamma \in [0, \inf)$ *and* $\delta \in (-\infty, \infty)$*.*

We highlight that there exist three Lévy-stable distributions with probability density functions that exhibit closed-form expressions: the Normal, the Cauchy, and the Lévy distributions. For such distributions, the stability and skewness parameters α and β are fixed, and the normal distribution can be seen as a degenerate thin-tail case.

The work in [31] shows that the Central Limit Theorem can be extended to sequences of random variables $\{X_n\}$ with diverging variances, which is the case for fat-tailed distributions, as the ones describing stock returns. This result allows to model stock returns using fat-tailed Lévy-stable distributions.

We provide the following formal definition of stock shocks:

Definition 2.3.3. *Given a threshold $k \in (0, 1)$ and a time window w of size n, we say that a* shock *has happened at time $t \in w$ if:*

$$\begin{cases} F_{X_n}(r_t) \geq 1 - k & r_t \geq 0 \\ F_{X_n}(r_t) \leq k & r_t < 0 \end{cases} \tag{2.1}$$

where F_{X_n} is the cumulative distribution of the returns in the time window, and r_t is the return at time t.

Figure 2.1 depicts the returns distribution over one day for the GME stock. In particular, the histogram shows the shape of the returns distribution, normalized by their density. The blue line is the cumulative distribution function of the Lévy-stable distribution fitted on the returns. The threshold k is set to 0.05. The intervals on which the cumulative distribution of the returns is below k or above $1 - k$ correspond to the tails of the histogram. Such returns are recognized as shocks.

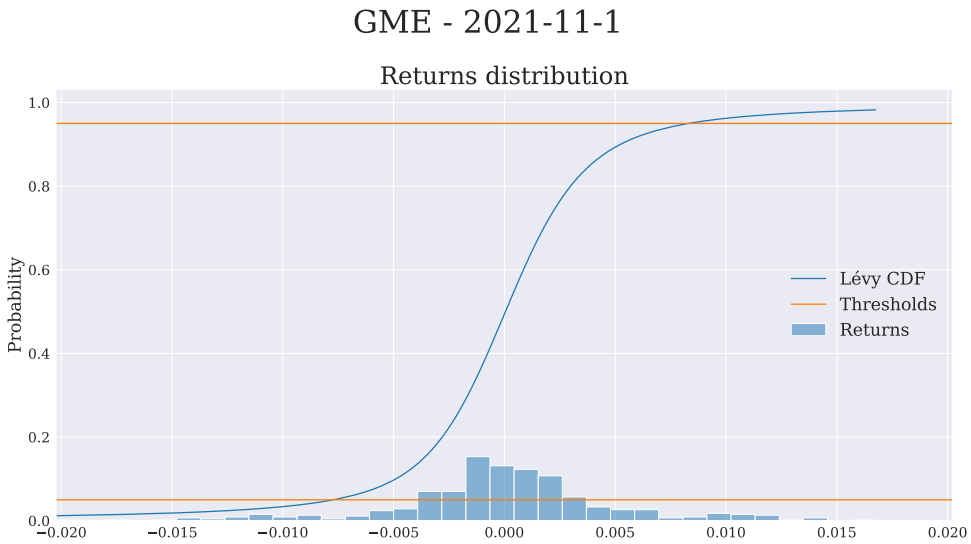


Figure 2.1: Return distribution over one day.

2.4 Shock prediction

2.4.1 Dataset

We use the LOBSTER [32] dataset, which provides Limit Order Book (LOB), and order message data, for Nasdaq securities (stocks), with tick time of 1 ns. A LOB is specific for a given security and records all the outstanding buy and sell orders currently available on an exchange or a trading platform. LOBs are complex data structures that collect peculiar and granular information on historical stocks' trading and have been widely used in the literature for stock trend prediction (see e.g. [14]). The LOB is structured as a list of buy and sell orders, each with a price and quantity,

ranked in order of priority based on the time of submission and price level. The LOB is provided by the exchange where the security is traded. The exchange or platform maintains the LOB and updates it in real-time as new orders are submitted, executed, or cancelled. The LOB is typically made available to traders, investors, and other market participants, often through a data feed. Each order is characterized by several variables, including the order timestamp (seconds after midnight), the event type (order submission, cancellation, deletion, execution), the order ID (a unique order reference number), the volume (number of shares), and the price. LOBs are organized in levels, aggregating orders with the same direction and specific price points. Each level l is characterized by the following values: the bid/buy volume orders vbuy_l, the ask/sell volume orders vsell_l, the bid/buy price orders pbuy_l, and the ask/sell price orders psell_l, where the level index l ranges from 1 to 10. The structure of the LOB and the related nomenclature are shown in Figure 2.2.

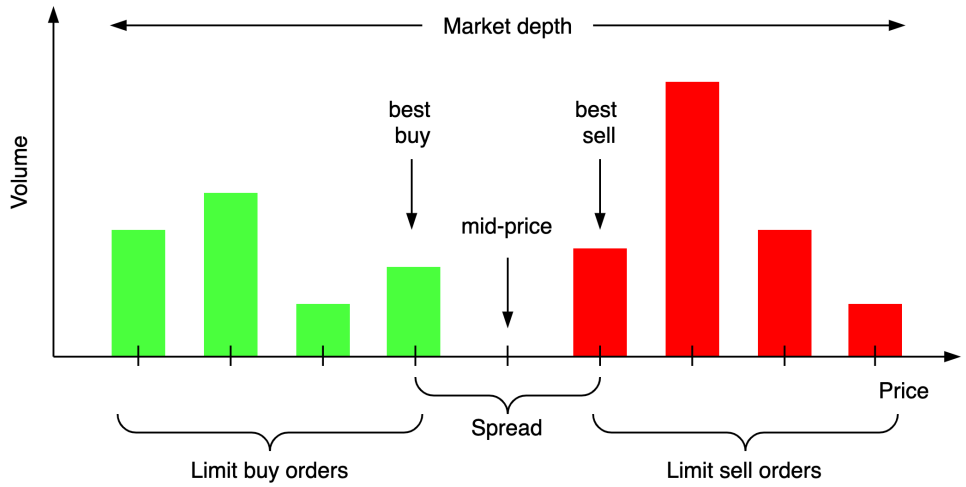


Figure 2.2: The structure of the Limit Order Book

In our experiments, we select the GME stock data, ranging from November 1^{st}, 2021, to April 29^{th}, 2022. The first 70% of this period constitutes the training set, and the remaining 30% the test set. The raw data are aggregated with a granularity $\Delta t = 30$ seconds. The last price in the aggregation window is maintained in the aggregation, whereas volumes are summed together. We compute the returns, the log returns, and the moving average of the price on a sliding window of 25 observations, $\mu_{25}(t)$.

Labeling

Given a window of length $w = 300 \times \Delta t = 9 \times 10^3$s and a threshold k, we slide the window over the log-returns with a step size $s = w$ and label the events the algorithm detects as shocks according to Definition 2.3.3. The frequency and intensity of the shocks depend on the analysed asset and the (k, w) parameters: the smaller the k, the less likely we classify an event as a shock. The time window w serves as a granularity parameter. For instance, if we look for shocks in a time window of 1 day (1440 minutes), shocks happening at lower frequencies (e.g., within 1 hour) will likely go undetected. Therefore changing the size of w allows us to control the duration of the shocks we include in the training set. Whenever a shock is found at time t, we also mark a *pre-shock offset* o_{ps}, set by default at $5 \times \Delta t = 150$s. The pre-shock offset allows us to predict when the shock will happen (150s in advance in our setting) and hence the investor to have time to take action. For

this reason, we define $t_s = t - o_{ps}$ as the *shock time*.

Feature engineering

After labeling, we enrich our data by extracting the following additional features. If not otherwise specified, we compute each feature at different look-behind windows of size $b \times \Delta t$, where $b \in \{5, 10, 50, 100\}$.

- **Lévy-parameters:** We set time windows w of $300 \times \Delta t = 9 \times 10^3$s and the same threshold k used for data labeling. We slide the windows over log returns with a stride $s = \Delta t = 30$s and fit the log returns within the interval w to a Lévy-stable distribution. In this way, we obtain five parameters for each time step t, namely $\alpha_w(t)$, $\beta_w(t)$, $\gamma_w(t)$ and $\delta_w(t)$ (see Definition 2.3.2), and the price moving average $\mu_w(t)$. These features are essential to detect a shock according to Definition 2.3.3; furthermore, irregular changes and unexpected behaviour in α and β can indicate imminent shock events [37].
- **Market depth:** As we show in Figure 2.2, it corresponds to $psell_{10} - pbuy_{10}$ at time $t - b$, and is an interest indicator of an asset price.
- **Percentage change:** For each feature extracted from the dataset (namely, vbuy_l, vsell_l, pbuy_l, psell_l, size, price, returns, log-returns and moving avg), we compute the percentage change. Then we compute the mean and standard deviation of the percentage changes over the window $[t-b, t]$. This feature provides information about the distribution of price differences over the window.
- **Mean and standard deviation:** For each feature extracted from the dataset (namely, vbuy_l, vsell_l, pbuy_l, psell_l, size, price, returns, log-returns and moving avg), we compute the mean and standard deviation over the window $[t - b, t]$. This operation allows us to observe how the distribution of a feature changes over time.
- **Order-book moments:** For all the order-book data (pbuy, psell, vbuy, vsell) we compute the first four moments of the distributions over the different levels at time $t - b$. This represents a snapshot of the whole order book before the shock. Note that this is different from the previous feature since the latter provides information about the distribution of a specific level over a time window. In contrast, the former provides information about all levels' distribution at a given time. The moments we consider are the mean, the standard deviation, the skewness, and the kurtosis.
- **Order-book resiliency:** This feature measures the robustness of the order book against one-sided, high-volume orders. The higher the resilience, the more robust the order book. Order book resilience is defined as the trade volume (i.e., number of shares by price) necessary to move the best sell price by five or ten levels at time $t - b$ [42].
- **Order-book volume cumulative sum (*cumsum*):** This feature is defined as the sum of all order book levels at time t over the past b time steps. We compute different *cumsum* for buy and sell events. If the sell volume is much greater than the buy volume, the price is likely to decrease, as the supply would be much greater than the demand.

  Table 2.1 summarizes the extracted features.

To enable shock prediction, non-shock events must also be included in the data. This way, the classifier learns to tell shock events apart from non-shock events. Therefore, after having computed the selected features for all the shock events, we added non-shock events to the dataset.

Machine learning models work best when the data distribution at inference time fit as much as

Table 2.1: Summary of computed features.

Feature	Derived from
Lévy	log-returns
Mean, std, percentage change (time-wise)	size, price, returns, α, β, γ, σ, μ, vbuy_l, vsell_l, pbuy_l, psell_l for $l \in \{1, \dots, 10\}$
Market depth	pbuy_{10}, psell_{10}
Order-book, mean, std skewness, kurtosis	vbuy_l, vsell_l, pbuy_l, psell_l for $l \in \{1, \dots, 10\}$
Order-book resiliency	vbuy_l, vsell_l, pbuy_l, psell_l for $l \in \{1, \dots, 10\}$
Cumulative sum	vbuy_l, vsell_l for $l \in \{1, \dots, 10\}$

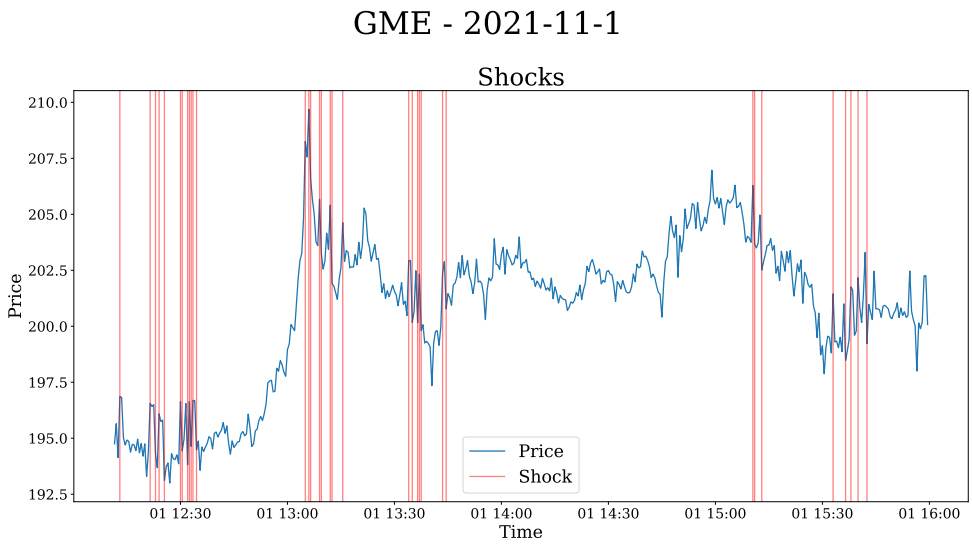
possible the data distribution used for training. In our context, this means having a shock frequency similar to the one in the market. Since Definition 2.3.3 defines a shock as an event happening with a probability smaller than k, $k \in (0, 1)$, a reasonable proportion of shocks in the training data would be $100 \cdot k\%$. Accordingly, if $n = \lfloor k \cdot N \rceil$ is the rounded number of shock events in the training set (whose size is N), we extract a random sample of $N - n$ non-shock events from the data, for which we compute the aforementioned features. Each row in the training set is labelled as 1 if it is a shock and 0 otherwise. We use class weights to cope with class imbalance.

A visual example of data labeling is given in Figures 2.3a and 2.3b. Both figures represent the return trend for the GME stock on November 1^{st}, 2021. Figure 2.3a depicts the curve between 12:00 p.m. and 4:00 p.m. Each vertical red line is a shock according to Definition 2.3.3. As we can see, shocks occur in correspondence with abrupt changes in the price trend. This phenomenon is highlighted in Figure 2.3b, where the trend curve is normalized and zoomed in a 20-minute time interval. In addition to the price trend and the shocks, the figure also shows the derivative of the price, in orange, which highlights how shocks lay in correspondence to the derivative's local minima and maxima of high magnitude.

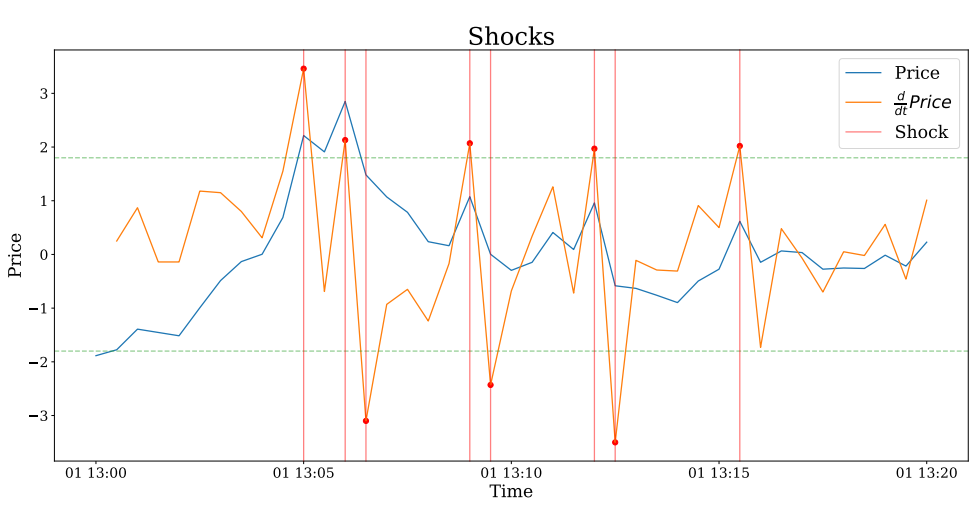
Finally, we normalize the features using the z-score. In order not to cause an information leak, we tune the scaler only on the training set and later apply it to the test set.

2.4.2 Preliminary results

In a preliminary study, we evaluate three classical machine learning algorithms for prediction: Random Forest, HDBSCAN, and a Multi-layer perceptron (MLP) network. Random Forests are powerful classifiers that reduce overfitting, are robust to noise, and have built-in criteria to determine feature importance. Density-Based Clustering Based on Hierarchical Density Estimates (HDBSCAN) is a density-based clustering algorithm that splits the space spanned by the data according to density regions and uses hierarchical density estimates to build optimal clusters. The HDBSCAN clustering algorithm is a convenient choice for our scenario, thanks to its stability with respect to changes in the hyperparameters and to its density-based approach, which well fits the fact that shocks lay on the tails of the data distribution. MLPs are feed-forward neural networks able to approximate complex functions by regression analysis. We use a network with five hidden layers of sizes 100, 70, 50, 30, 10 with `ReLU` activation function, weight decay (`l2`-normalization) set to 10^{-3}, and Adam optimizer. The advantage of using `ReLU` over other functions such as the `tanh` or the `sigmoid` is to limit the problem of gradient vanishing in hidden layers. As a baseline, we use



(a) One-day period.



(b) 20 minutes, normalized.

Figure 2.3: GME mid-price (blue) with shocks (red lines).

a simple probabilistic model, which, given an event, predicts it as a shock with probability $\frac{n}{N}$. Such class priors are 0.95 for non-shocks and 0.05 for shocks. Our preliminary results are evaluated in terms of precision, recall, and F1 and reported in Table 2.2.

Despite the high recall, HDBSCAN has a very low precision: the model classifies many events as shocks. This fact highlights that distance-based approaches might not be suited to this problem: shocks form well-defined clusters (hence the high recall) that contain lots of non-shocks (thus the low precision). MLP achieves results similar to the random forest classifier. Deep learning is not the best-performing approach on tabular data [43]. In fact, tabular data is heterogeneous and sparse, and unlike images, audio, and language, there can be multiple variations in a table's column. This leads to features living in a high-dimensional space that is generally not dense and continuous, making it difficult to exploit for a deep neural network. Regarding random forests, fitting a model out of the box (e.g., with default parameters and without fine-tuning) dramatically improves the precision with respect to the baseline.

Table 2.2: Preliminary results.

Model	Class	Precision	Recall	F1
Baseline	0	0.95	0.95	0.95
	1	0.03	0.03	0.03
HDBSCAN	0	0.94	0.98	0.96
	1	0.08	0.67	0.14
MLP	0	0.96	0.98	0.97
	1	0.37	0.18	0.24
Random forest default	0	0.96	0.98	0.97
	1	0.75	0.15	**0.25**

We exploit the evidence collected in this preliminary study to design an optimized model that we detail in the next section.

2.4.3 Model optimization

Since the random forest is the model with the highest F1 score in Table 2.2, we use it as a baseline for our optimized model.

Feature Selection

The first step of our optimization is reducing the feature space by studying the features' multicollinearity. A feature is multicollinear when it can be predicted from other features with very high accuracy. Perfect multicollinearity is when a feature is linearly dependent on the data vector space. Multicollinearity does not reduce the model's predictive power but jeopardizes its interpretability and generalization [44]. In fact, if two features have similar effects on the final prediction, it might be difficult to rank their relative importance. Their contribution to the performance of the classification might be equally split among them. This can lead to a non-realistic assessment of the impact of such features. Since they are collinear (hence one can be obtained from the other with high accuracy), choosing to keep only one of them does not impact the classifier's performance; at the same time, the selected feature will be correctly acknowledged for its contribution to the classification. For this reason, removing multicollinear features eases the model's interpretability and decreases training time.

To perform feature selection while accounting for feature multi-collinearity, we perform hierarchical clustering on the features' Spearman rank-order correlations $\rho(\cdot, \cdot)$, as suggested in [45]. The Spearman rank-order correlation index lies in the interval $[-1, 1]$ and measures the dependence between the rankings of two variables: it is high when observations have a similar rank (e.g., the relative position label of the observations within the variable). For each couple of features f_1, f_2, we formally define their distance as a function of the Spearman index as $d(f_1, f_2) = 1 - |\rho(f_1, f_2)|$.

The HDBSCAN clustering algorithm first builds a minimum spanning tree over a graph where nodes are features, and the weight of an edge $\{f_1, f_2\}$ is the value of $d(f_1, f_2)$. The tree can then be translated into a dendrogram, as in Figure 2.4. Lastly, we manually set a threshold $\tau = 1.5$ and keep a single feature from each cluster, chosen uniformly at random. This feature selection is repeated for each considered time granularity. In Table 2.4, we report the results obtained by training a new random forest classifier with only the selected features. As we can see, this feature selection phase significantly improves the initial performance of the Random Forest (Table 2.2), especially in terms

Table 2.3: Hyperparameters.

Hyperparameter	Tested	Optimal
n_estimators	range(20, 550)	120
max_depth	range(5, 50)	25
criterion	Gini impurity Shannon Information gain	Gini impurity
ccp_alpha	range(0, 1)	0

Table 2.4: Final results.

Model	Class	Precision	Recall	F1
Random forest	0	0.96	0.98	0.97
(important features only)	1	0.81	0.31	**0.45**
Random forest	0	0.97	0.98	0.97
(important features only + optimal hyperparameters)	1	0.80	0.37	**0.50**

of recall. As a matter of fact, reducing the number of features by keeping only the most relevant signals also reduces noise.

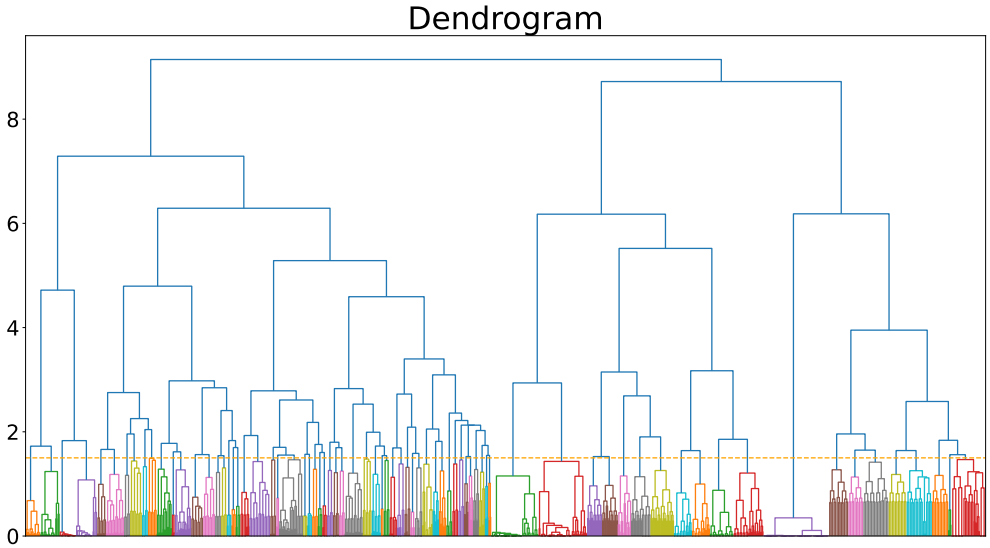


Figure 2.4: Dendrogram of Spearman rank-order correlations.

Hyperparameter tuning

The second phase of our optimization consists of tuning the random forest classifier's hyperparameters. To do so, we perform a Bayesian search over a grid of hyperparameters [46]. Bayesian optimization is an iterative procedure that builds a probability model of the objective function (that is, the function optimized by the random forest classifier) and uses it to tune the most effective hyperparameter values. At each iteration, the hyperparameter choice is assessed by using the binary cross entropy loss as an acquisition function. This process allows us to refine the progressively computed solutions and quickly reduce the search space. The hyperparameters selected by the Bayesian Optimization are listed in Table 2.3. As shown in Table 2.4, using optimal hyperparameters is a further performance boost: it increases the recall to 0.37 at the cost of only one percentage point in precision.

Table Table 2.5 shows the results for different stocks and granularities. Only the metrics for the shock events are reported.

Table 2.5: Results on other stocks.

Stock	Precision	Recall	F1
GME @ 1m	0.80	0.29	0.42
GME @ 30s	0.80	0.37	0.50
KO @ 30s	0.84	0.23	0.36
AAPL @ 30s	0.87	0.26	0.40

2.4.4 Comments

For each instance, the random forest returns a probability score p that represents the confidence of the binary classification, where "1" is a shock, and "0" is a non-shock event. In particular, a prediction is considered to be a shock if $p \geq 0.5$. Depending on the use case, optimizing the model for precision or recall is possible by moving this classification threshold. Optimizing for precision is crucial if an investor is interested in exploiting trading opportunities derived from the shock prediction. As a matter of fact, false positives are extremely dangerous in such a scenario: forecasting a shock event (in any direction) that will not occur causes potential capital loss to an investor that applies such a trading strategy. On the other hand, false negatives are less risky phenomena, as they prevent the investor from taking any action, without leading to any capital loss. So, a high recall is essential if our task is to mitigate risks by re-balancing portfolios before a shock happens.

2.5 Future work

The work presented in this chapter proposes for the first time a formal definition of stock shocks based on Lévy-stable distributions for stock price modeling. This approach paves the way for several future research directions:

- Directed shock forecasting, namely the prediction of the shock direction. We highlight that this kind of classification requires a qualitative analysis of financial indicators, including order-book volumes and prices signal extraction, and of the market volatility, which is mutually dependent on extreme market shocks [47].
- Testing the behavior of our shock predictor and our shock labeling model for stock price trends in periods of massive market volatility with different time scales (e.g., flash crash of May 2010 [48], Covid crash of March 2020).
- Considering additional exogenous factors influencing stock shocks. Price's abrupt alterations are often influenced by phenomena that do not directly come from financial decisions (e.g., geographic disasters and political events). LOB data can be enriched with sentiment data coming from other sources, such as Twitter and other breaking news media.

2.6 Conclusions

This chapter provides a formal model for stock shocks based on fat-tailed Lévy-stable distributions. The proposed model is used for stock shocks forecasting, which is performed by an optimized

Machine Learning algorithm relying on random forests, HDBSCAN clustering, and Bayesian optimization for hyperparameter selection. The algorithm is able to forecast shock events with high precision using Limit Order Book (LOB) data, a data structure used by stock exchanges to describe all outstanding orders for a given stock. The promising results reported in this chapter encourage the use of the proposed approach for more sophisticated scenarios.

Chapter 3

LOB-Based Deep Learning Models for Stock Price Trend Prediction: A Benchmark Study

Abstract

The recent advancements in DL research have notably influenced the finance sector. We examine the robustness and generalizability of fifteen state-of-the-art DL models, focusing on SPTP based on LOB data. To carry out this study, we developed LOBCAST, an open-source framework that incorporates data preprocessing, DL model training, evaluation, and profit analysis. Our extensive experiments reveal that all models exhibit a significant performance drop when exposed to new data, thereby raising questions about their real-world market applicability. Our work serves as a benchmark, illuminating the potential and the limitations of current approaches and providing insight for innovative solutions.

This work has been published in Artificial Intelligence Review [26].

3.1 Introduction

Predicting stock market prices is a complex endeavor due to the intrinsic and tangled nature of a multitude of factors that influence the market, including macroeconomic conditions, natural events, and investor sentiment [49]. Professional traders and researchers usually forecast price movements by understanding key market properties, such as volatility or liquidity, and recognizing patterns to anticipate future market trends [50]. Effective mathematical models are essential for capturing complex market dependencies. The recent surge in artificial intelligence has led to significant work in using machine learning algorithms to predict future market trends [51–53]. Recent Deep Learning (DL) models have achieved over 88% in F1-Score in predicting market trends in simulated settings using historical data [15]. However, replicating these performances in real markets is challenging, suggesting a possible *simulation-to-reality* gap [54, 55].

Due to the broad interest in this field, in recent years, many researchers have proposed survey papers that study the vast literature on stock market predictions. Most of these surveys propose meticulous classification and analysis of the existing literature and review of the implemented models

based on the results shown in the original papers.

Nevertheless, most of these works consist of desk-based research, and only a few of them propose new experiments to validate the collected results. In this chapter, we benchmark the most recent and promising DL approaches to Stock Price Trend Prediction (SPTP) based on Limit Order Book (LOB) data, one of the most valuable information sources available to traders on the stock markets.

The LOB aggregates orders for shares of a given stock over time, characterizing each order by its associated price and volume of shares. SPTP is the problem of forecasting the stock price trend based on LOB data, classifying it as either *upward*, *downward*, or *stable*.

We compare novel data-driven approaches from Machine Learning (ML) and Deep Learning (DL) that analyze the market at its finest resolution, using high-frequency Limit Order Book (LOB) data. Our benchmark evaluates their robustness and generalizability [56–58]. In particular, we assess the models' robustness by comparing the stated performance with our reproduced results on the same dataset FI-2010 [59]. We also assess their generalizability by testing their performance on unseen market scenarios using LOBSTER data [32].

Furthermore, we enrich our experiments by including classical ML tree-based models and ensemble methods. In addition, we also provide a profit analysis by conducting a trading simulation. Our code is organized in a modular framework, called LOBCAST, which is openly accessible for users. Our findings reveal that while the best models exhibit robustness, achieving solid F1-Scores on FI-2010, they show poor generalizability, as their performance significantly drops when applied to unseen LOB market data. Our experiments show that most of the attention-based DL models outperform the other approaches. Our results provide insightful evidence of possible weaknesses of the current state-of-the-art in SPTP, which allow us to add a critical discussion about how to improve models' generalizability, data labeling, and representation.

The main contributions of our work are the following:

- We release a highly modular open-source framework called **LOBCAST**[1], to pre-process data, train, and test stock market models. Our framework employs the latest DL libraries to provide all researchers with an easy, performant, and maintainable solution. Furthermore, to support future studies, we release two meta-learning models and a backtesting environment for profit analysis.
- We evaluate existing LOB-based stock market trend predictors, showing that most of them overfit the FI-2010 dataset with remarkably lower performance on unseen stock data.
- In order to guide model selection in real-world applications, we evaluate the sensitivity of the models to the data labeling parameters, compare the performance of both DL and non-DL models, and evaluate and discuss the financial performance of existing models under different market scenarios.
- We discuss the strengths and limitations of existing methodology and identify areas for future research toward more reliable, robust, and reproducible approaches to stock market prediction.

The remainder of this chapter is organized as follows: in Section 3.2, we review existing work; in Section 3.3, we introduce the stock trend prediction problem; in Section 3.4 we present all the machine learning models scrutinized in this work and in Section 3.5 we describe the datasets used to train the model for the task; in Section 3.6 we benchmark the analyzed approaches. In Section 3.7, we discuss future directions and conclusions reached by our research. Finally, we devote Section 3.8.1

[1]The code is publicly available at https://github.com/giuseppemasi99/LOBCAST

to a detailed description of the selected models, whose robustness and generalizability study is further enriched by the additional experimental results reported in Section 3.8.2.

3.2 Related Work

The increasing interest in DL for price trend prediction motivated several researchers to collect and analyze state-of-the-art (SOTA) solutions in benchmark surveys. The study by Jiang et al. [52] analyzes papers published between 2017 and 2019 that focused on stock price and market index prediction. In their literature review, the authors studied Neural Network (NN) structures and evaluation metrics used in selected papers, as well as implementation and reproducibility. This work was extended by Kumbure et al. [60], including an in-depth analysis of the data (i.e., market indices and input variables used for stock market predictions). Ozboyoglu et al. [61] provide a comprehensive overview of SOTA DL and ML algorithms that are commonly used for finance applications. The authors then survey numerous papers tackling some of such applications with DL and ML models, e.g., portfolio management, fraud detection, and risk assessment. With a similar approach, Sezer et al. [53] summarize the most used DL models for several finance applications, including stock price forecasting. The work by Hu et al. [62] surveys 86 papers on stock and foreign exchange price prediction. The authors review the datasets, variables, models, and performance metrics used in each surveyed article. In Nti et al. [63], the authors conduct a systematic and critical review of 122 papers for stock prediction. To evaluate the results of the surveyed papers, the authors also implement three baselines: DL and ML algorithms, which are commonly exploited in the reviewed literature: Decision Trees (DTs), Support Vector Machines (SVMs), and Artificial Neural Networks (ANNs). The authors show that ANN achieves the best performance in terms of different error metrics, followed by DTs and SVM.

In contrast to the aforementioned works that have primarily surveyed and reviewed the literature on the SPTP task in general, our focus is specifically on papers addressing this task using LOB data, as will be discussed in Section 3.3. Moreover, our work is not limited to evaluating the results reported in the surveyed papers and to proposing a validation through classical baselines. Instead, we also evaluate the generalizability of the models by running tests on different datasets.

Several studies include sentiment analysis data for price trend prediction. The works by Shah et al. [64] and Al-Alawi et al. [65] analyze solutions based on sentiment analysis through Natural Language Processing (NLP) to investigate the impact of social media on the stock market, showing that this combination improves the accuracy of stock prediction models. Similar conclusions are reported in [66, 67].

A different research topic than the one proposed here is the design of recommendation systems to select the most profitable stocks. This field of research relies on the observation that some investors might be interested in predicting the top profitable k stocks instead of price trends. Saha et al. [68] propose a new measure for stock ranking prediction to maximize investors' profit. The work in [69] explores rank-based ML-based approaches and identifies a feature set that contains various statistics indicating the performance of stock market companies that can be used to train several ranking models. Song et al. [70] use two DL models to design learning-to-rank algorithms to construct equity portfolios based on sentiment news.

Rundo et al. [71] presented a comprehensive overview of traditional and ML-based approaches for

stock market prediction and highlighted some limitations of traditional approaches, showing that DL models outperform them in terms of accuracy. Similar findings are reported by Mintarya et al. [72]. Lim et al. [73] discussed recent developments in hybrid DL models, which combine statistical and learning components for both one-step-ahead and multi-horizon time-series forecasting. Similarly, Shah et al. [74] discussed hybrid approaches in their work on the SOTA algorithms commonly applied to stock market prediction. Additionally, they provided a taxonomy of computational approaches for stock market analysis and prediction. Olorunnimbe et al. [75] explore applications of DL in finance and stock markets, with a particular emphasis on works proposing backtesting meeting the requirements for real-world use. They reviewed various scenarios of DL models in finance, with a focus on trade strategy, price prediction, portfolio management, and others. The authors also underline whether the surveyed papers are reproducible. Nevertheless, the reproducibility is not studied by means of experiments but rather just by checking whether the authors of the collected papers provided an open-source code. Finally, Lucchese et al. [76] study a similar problem focusing on order book-driven predictability by using deep learning techniques. They focus on the order book representation, including a novel representation of volumes, and they evaluate the relation between the price predictability and how far ahead we can actually predict. They meticulously address questions related to data representation, yet they focus their study on only three models. Different from this work, our study considers a broader range of existing works, providing related implementation and comparative analysis.

Several works exploit technical indicators such as moving average convergence/divergence, momentum analysis, volume on balance, and relative strength index to predict market trends with ML (see e.g., [77–79]). In this chapter, we benchmark papers that propose models for stock price trend prediction relying on LOB data. While data using technical indicators offer explainability and can be effective for market predictions, our work studies models that only include market data at its finest granularity and aims to investigate if DL models can extract useful information from raw data, going beyond technical indicators.

Our work is the first to provide a benchmark of recent DL approaches applied to the SPTP task, utilizing LOB data. In contrast to previous work, we re-implement the surveyed papers to evaluate their robustness and generalizability. Furthermore, we have released an open-source framework designed for data preprocessing, model training, and testing. Our framework also incorporates profit analysis capabilities that users can exploit to test their own price trend prediction model.

3.3 The Stock Price Trend Prediction Problem

The common ground that unifies the models studied here is the goal of solving the SPTP problem via Deep Neural Networks (DNNs) trained on LOB data. LOB data are particularly enlightening as they provide raw and granular information on stocks' trades. By observing the LOB in a fixed period of time, SPTP models return a distribution over the possible future market movements.

Limit Order Book (LOB) A stock exchange employs a matching engine for storing and matching the orders issued by the trading agents. This is achieved by updating the so-called Limit Order Book (LOB) data structure. Each security (tradable asset) has an LOB, recording all the outstanding bid and ask orders currently available on an exchange or a trading platform. The shape of the

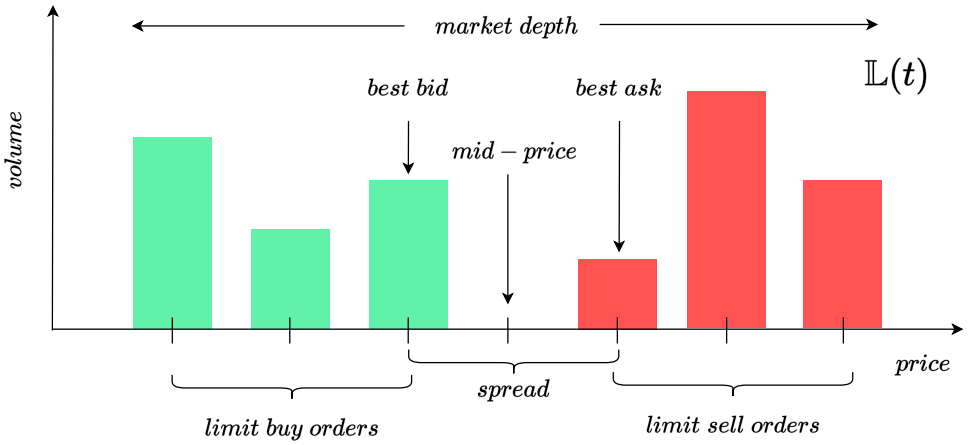


Figure 3.1: An example of LOB.

order book gives traders a simultaneous view of the market demand and supply.

There are three major types of orders. *Market orders* are executed immediately at the best available price. *Limit orders*, instead, include the specification of a desired target price: a limit sell [buy] order will be executed only when it is matched to a buy [sell] order whose price is greater [lower] than or equal to the target price. Finally, a *cancel order* removes a previously submitted limit order.

Figure 3.1 depicts an example of an LOB snapshot, characterized by *buy* orders (*bid*) and *sell* orders (*ask*) of different prices. A *level*, shown on the horizontal axis, represents the number of shares with the same price either on the bid or ask side. In the example of Figure 3.1, there are three bid and three ask levels. The *best bid* is the price of the shares with the highest price on the buy side; analogously, the *best ask* is the price of the shares with the lowest price on the sell side. When the former exceeds or equals the latter, the corresponding limit ask and bid orders are executed. The LOB is updated with each event (order insertion/modification/cancellation) and can be sampled at regular time intervals.

In [80–85] it has been empirically demonstrated, using both linear and non-linear models, that the orders behind the best bid and ask prices have a significant impact to price discovery and contain information about short-term future price movements, supporting the hypothesis that leveraging deeper levels of the limit order book is essential for improving the performance of SPTP tasks. This is the main reason for not restricting the input to the best levels. Additionally, deep learning models are well-suited to handle high-dimensional input.

We represent the evolution of an LOB as a time series $\mathbb{L}$, where each $\mathbb{L}(t) \in \mathbb{R}^{4L}$ is called an LOB record, for $t = 1, \ldots, N$, being N the number of LOB observations and L the number of levels. In particular,

$$\mathbb{L}(t) = \{P^s(t), V^s(t)\}_{s \in \{\texttt{ask},\texttt{bid}\}}$$

where $P^{\texttt{ask}}(t), P^{\texttt{bid}}(t) \in \mathbb{R}^L$ represent the prices of levels 1 to L of the LOB, on the *ask* ($s = \texttt{ask}$) side and *bid* ($s = \texttt{bid}$) side, respectively, at time t. Analogously, $V^{\texttt{ask}}(t), V^{\texttt{bid}}(t) \in \mathbb{R}^L$ represent the volumes. This means that for each t and every $j \in \{1, \ldots, L\}$ on the *ask* side, $V_j^{\texttt{ask}}(t)$ shares can be sold at price $P_j^{\texttt{ask}}(t)$. The *mid-price* $m(t)$ of the stock at time t is defined as the average

value between the best bid and the best ask,

$$m(t) = \frac{P^{\texttt{ask}}(t) + P^{\texttt{bid}}(t)}{2}.$$

On average, if most of the executed orders are on the ask [bid] side, the mid-price increases [decreases] accordingly.

Trend Definition We use a ternary classification for trends: `U` ("upward") if the price trend is increasing; `D` ("downward") for decreasing prices; and `S` ("stable") for prices with negligible variations. Among all the possible single values, mid-prices provide the most reliable indication of the actual stock price for equity markets. Nevertheless, because of the market's inherent fluctuations and shocks, they can exhibit highly volatile trends. For this reason, using a direct comparison of consecutive mid-prices, i.e., $m(t)$ and $m(t+1)$, for stock price labeling would result in a noisy labeled dataset. As a result, labeling strategies typically employ smoother mid-price functions instead of raw mid-prices. Such functions consider mid-prices over arbitrarily long time intervals, called *horizons*. Our experiments adopt the labeling proposed in [59] and repurposed in several other SOTA solutions we selected for benchmarking. The adopted labeling strategy compares the current mid-price to the average mid-prices $a^+(k,t)$ in a future *horizon* of k time units, formally:

$$a^+(k,t) = \frac{1}{k}\sum_{i=1}^{k} m(t+i). \tag{3.1}$$

The average mid-prices are used to define a static threshold $\theta \in (0,1)$ that is used to identify an interval around the current mid-price and define the class of the trend at time t as follows:

$$\begin{gathered}\texttt{U}\colon a^+(k,t) > m(t)(1+\theta),\ \texttt{D}\colon a^+(k,t) < m(t)(1-\theta),\\ \texttt{S}\colon a^+(k,t) \in [m(t)(1-\theta), m(t)(1+\theta)].\end{gathered} \tag{3.2}$$

With this labeling, we beat the effect of mid-price fluctuations by considering their average over a desired horizon k and considering a trend to be stable when the average mid-price variations do not change significantly, thus avoiding over-fitting. We highlight that timestamp t can come either from a homogeneous or an event-based process. In our experiments, we consider the latter approach. Hence, the horizon k is expressed in the number of future events.

Models I/O Given the time series of an LOB $\mathbb{L}$ and a temporal window $T = [t-h, t]$, $h \in \mathbb{N}$, we can extract *market observations* on T, $\mathbb{M}(T)$, by considering the sub-sequence of LOB observations starting from time $t-h$ up to t. Figure 3.2 gives a representation of a market observation $\mathbb{M}(T) \in \mathbb{R}^{h \times 4L}$. The market observation over the window $[t-h, t]$ is associated with the label computed through Equations (3.1) and (3.2) at time t. An SPTP predictor takes as input a market observation and outputs a probability distribution over the trend classes `U`, `D`, and `S`.

$\mathbb{M}(T)$

		side →	*ask*			*bid*		
		volume →	V_1	$\cdots$	V_L	V_1	$\cdots$	V_L
		price →	P_1	$\cdots$	P_L	P_1	$\cdots$	P_L
time	$t-h$		$v_1(t-h)$	$\cdots$	$v_L(t-h)$	$v_1(t-h)$	$\ldots$	$v_L(t-h)$
			$p_1(t-h)$	$\cdots$	$p_1(t-h)$	$p_1(t-h)$	$\cdots$	$p_1(t-h)$
	$t-h+1$		$v_1(t-h+1)$	$\cdots$	$v_L(t-h+1)$	$v_1(t-h+1)$	$\cdots$	$v_L(t-h+1)$
			$p_1(t-h+1)$	$\cdots$	$p_L(t-h+1)$	$p_1(t-h+1)$	$\cdots$	$p_L(t-h+1)$
	$\vdots$		$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
	t		$v_1(t)$	$\cdots$	$v_L(t)$	$v_1(t)$	$\cdots$	$v_L(t)$
			$p_1(t)$	$\cdots$	$p_L(t)$	$p_1(t)$	$\cdots$	$p_L(t)$

Figure 3.2: An example of market observation.

3.4 Models

We selected and surveyed 13 SOTA models based on DL for the SPTP task using LOB data. Model selection was made based on their prominence in the literature and widespread usage. These models represent a diverse set, covering various DL architectures and methods. These models were published in papers between 2017 and 2022 and are described in detail in Section 3.8.1. We also include in our analysis two additional baselines, namely Multi Layer Perceptron (MLP) and Long Short-Term Memory (LSTM), which were used as a benchmark in [86] and in [87], respectively. All proposed models are based on DNN and were originally trained and tested on the FI-2010 dataset. We also study two ensemble methods, described in Section 3.4.2. Table 3.1 summarizes the most peculiar characteristics of the studied models, which we comment on in Section 3.4.1.

3.4.1 Summary of Models

Table 3.1 summarizes the most peculiar characteristics of the selected models. The *temporal shape* represents the length of the input market observation for the model. In the table, the *features shape* refers to the number of features used by the models to infer the trend in the original papers. In the Table, we also indicate whether the authors released the code, and if so, whether they have used PyTorch [88] or TensorFlow [89]. This is relevant because, to ensure consistency and compatibility within our proposed framework, based on PyTorch Lightning, we found it necessary to re-implement models for which the code was not available or was only available in Tensorflow. We made every effort to validate and verify the correctness of our re-implementations, including making our code publicly available, which facilitates collaboration and scrutiny from the research community. We remark that to improve the reproducibility of the results, it would be advisable for the research community to publish the code developed to carry out the experiments.

In High-Frequency Trading (HFT) and algorithmic trading in general, minimizing latency be-

Table 3.1: Relevant characteristics of the selected models.

	temporal shape (h)	**features shape**	**code available**	**nr trainable parameters**	**inference time (ms)**
Tsantekidis et al. [86] MLP (2017)	100	40	✗	10^6	0.08
Tsantekidis et al. [86] LSTM (2017)	100	40	✗	$1.6 \cdot 10^4$	0.21
Tsantekidis et al. [94] CNN1 (2017)	100	40	✗	$3.5 \cdot 10^4$	0.36
Tran et al. [95] CTABL (2018)	10	40	TensorFlow	$1.1 \cdot 10^4$	0.48
Zhang et al. [14] DEEPLOB (2019)	100	40	PyTorch	$1.4 \cdot 10^5$	1.31
Passalis et al. [96] DAIN (2019)	15	144	PyTorch	$2.1 \cdot 10^6$	0.15
Tsantekidis et al. [87] CNNLSTM (2020)	300	42	✗	$5.3 \cdot 10^4$	0.50
Tsantekidis et al .[87] CNN2 (2020)	300	40	✗	$2.8 \cdot 10^5$	0.49
Wallbridge et al. [97] TRANSLOB (2020)	100	40	TensorFlow	$1.1 \cdot 10^5$	2.40
Passalis et al. [98] TLONBoF (2020)	15	144	PyTorch	$6.5 \cdot 10^5$	0.43
Tran et al. [15] BINCTABL (2021)	10	40	✗	$1.1 \cdot 10^4$	0.71
Zhang et al. [99] DEEPLOBATT (2021)	50	40	TensorFlow	$1.8 \cdot 10^5$	1.73
Guo et al. [100] DLA (2022)	5	144	✗	$1.2 \cdot 10^5$	0.23
Tran et al. [101] ATNBoF (2022)	100	40	PyTorch	$1.3 \cdot 10^7$	3.90
Kisel et al. [102] AXIALLOB (2021)	40	40	✗	$2 \cdot 10^4$	1.91

tween model querying and order placement is of utmost importance [90]. To explore this aspect, we analyzed the inference time, in milliseconds, of all models based on the experiments reported in Section 3.6.3. As shown in Table 3.1, DEEPLOB, DEEPLOBAT, AXIALLOB, TRANSLOB, and ATNBoF had inference times in the order of milliseconds, potentially unsuitable for HFT applications compared to other models with shorter times. Finally, we have reported the number of trainable parameters for each model. A noteworthy observation is that the average number of parameters is very low compared to other classical deep learning fields, such as computer vision [91] and natural language processing [92, 93]. This leads us to conjecture that current systems are inadequate in effectively handling the complexity of LOB data, as we will verify in the rest of this chapter.

3.4.2 Ensemble Methods

To explore the possibility of achieving new SOTA performance by combining the predictions of all 15 models, we have implemented two ensemble methods: MAJORITY and METALOB.

The MAJORITY ensemble assigns the class label that appears most frequently among the predictions of the classifiers. To account for variations in the performance of individual classifiers, we incorporate a weighting scheme based on their F1-Scores. This ensures that predictions from higher-performing models carry more influence in the final decision.

The METALOB meta-classifier is implemented as a multilayer perceptron (MLP) with two fully

connected layers. It is designed to learn how to effectively combine the outputs of the 15 DL models, which serve as the base classifiers to produce the final output. The input to the meta-classifier is a 1D tensor with a probability distribution over the trends (*up, stationary, down*) for each of the models, resulting in a tensor of $3 \cdot 15$ elements.

3.5 Datasets

LOB data are not often publicly available and very expensive: stock exchanges (e.g., NASDAQ) provide fine-grained data only for high fees. The high cost and low availability restrict the application and development of DL algorithms in the research community. In the sections that follow, we will introduce the reader to two datasets that will be used to analyze the performances of the models under the robustness and generalizability point of view, that are FI-2010 and LOB-2021/2022 respectively.

3.5.1 FI-2010 to test Robustness

The most widely spread public LOB dataset is **FI-2010**, which is licensed under *Creative Commons Attribution 4.0 International (CC BY 4.0)* and was proposed in 2017 by Ntakaris et al. [59] with the objective of evaluating the performance of machine learning models on the SPTP task. The dataset consists of LOB data from five Finnish companies: Kesko Oyj, Outokumpu Oyj, Sampo, Rautaruukki, and Wärtsilä Oyj of the NASDAQ Nordic stock market. Data spans the time period between June 1st to June 14th, 2010, corresponding to 10 trading days (trading happens only on business days). About 4 million limit order messages are stored for the ten levels of the LOB. The dataset has an event-based granularity, meaning that the time series records are not uniformly spaced in time. LOB observations are sampled at intervals of 10 *events*, resulting in a total of 394,337 events. This dataset has the intrinsic limitation of being already pre-processed (filtered, normalized, and labeled), so that the original LOB cannot be backtracked, thus hampering thorough experimentation. Additionally, the labeling method employed is found to be prone to instability, as demonstrated by Zhang et al. in [14].

The dataset provides the time series and the classes relative to five horizons $k \in \mathcal{K} = \{1, 2, 3, 5, 10\}$ by leveraging the trend definitions described in Equation (3.2). Such a labeling scheme is very sensitive to the threshold θ regarding the resulting balancing between "upward", "downward", and "stable" trends. Table 3.2 shows the class balancing for different horizons $k \in \mathcal{K}$. The authors of the dataset employed a single threshold $\theta = 0.002$ for all horizons, but it balances only the case of $k = 5$. As it can be observed, the stationary class `S` is progressively less predominant in favor of the upward and downward classes. In our experimental campaign, the class imbalance is not addressed to guarantee a fair robustness evaluation since the considered works do not claim to have done so.

3.5.2 LOB-2021/2022 to test Generalizability

To test the generalizability of the models in a more realistic scenario, we used data extracted from *LOBSTER* [32], an online LOB *data provider* for order book data, which is publicly available for the research community with an annual fee. LOBSTER limit order books are reconstructed directly from NASDAQ-traded stocks. To compare the performance of the algorithms in a wide range of

Table 3.2: Class balancing on FI-2010.

Horizon k	**Train Set** {U,S,D} (%)	**Val Set** {U,S,D} (%)	**Test Set** {U,S,D} (%)
1	$20 - 60 - 20$	$19 - 63 - 18$	$15 - 71 - 14$
2	$26 - 49 - 25$	$24 - 52 - 24$	$20 - 62 - 18$
3	$30 - 41 - 29$	$27 - 46 - 27$	$23 - 56 - 21$
5	$35 - 30 - 35$	$32 - 37 - 31$	$28 - 47 - 25$
10	$41 - 18 - 41$	$37 - 26 - 37$	$34 - 34 - 32$

scenarios, we have created a large LOB dataset, including several stocks and time periods. The chosen pool of stocks includes those from the top 50% most liquid stocks of NASDAQ. To construct a diversified evaluation scenario, we selected six stocks, namely: SoFi Technologies (SOFI), Netflix (NFLX), Cisco Systems (CSCO), Wing Stop (WING), Shoals Technologies Group (SHLS), and Landstar System (LSTR). The periods in consideration are *July 2021* (2021-07-01 to 2021-07-15, 10 trading days) making up **LOB-2021**, and *February 2022* (2022-02-01 to 2022-02-15, 10 trading days) making up **LOB-2022**. The selection of these two periods aimed to capture data from periods with different levels of market volatility. February 2022 exhibited higher volatility compared to July 2021, largely influenced by the Ukrainian crisis. This allows for an assessment of models across varying market conditions. Further details on stock selection and processing are provided in the following two paragraphs.

Stocks Selection

To make up *LOB-2021* and *LOB-2022* and consider variegated evaluation scenarios, we curated a pool of 630 stocks from the NASDAQ exchange with a market capitalization that ranged from ~ 2 billion to ~ 3 trillion dollars. Data was gathered from NASDAQ Stock Screener [103]. From the pool of stocks, we generated 6 clusters with *t-distributed Stochastic Neighbor Embedding* (*t*-SNE) to capture stock differences in the years 2021-2023. We used the following features: daily return, hourly return, volatility, outstanding shares, P/E ratio, and market capitalization. The P/E ratio indicates the ratio between the price of a stock (P) and the company's annual earnings per share (E). The analysis led to the identification of 6 stocks that are nearest to the cluster centroids of the generated 3-dimensional latent space. The stocks make up the set $\mathcal{S} = \{$SOFI, NFLX, CSCO, WING, SHLS, LSTR$\}$. Table 3.3 captures the main features of these stocks for the period of July 2021. The selected stocks have very variable average daily returns, the minimum being SHLS and the maximum being NFLX. Daily and Hourly returns highlight that some stocks are more volatile than others. The market capitalization represents the total value of the outstanding common shares that stockholders own. Stocks show different class balancing in the training set. CSCO is the stock with a major imbalance toward the stable class, whereas NFLX and LSTR are more unbalanced towards the up and down classes, respectively. In Section 3.6, we analyze the reasons behind the occurrence of class imbalance specific to individual stocks and discuss its impact. The mid-price movements for these two periods and the selected stocks are depicted in Figure 3.3.

Table 3.3: LOB-2021 stocks main features

Stock	Daily Return (%)	Hourly Return (%)	Market Cap.	P/E Ratio	Train Set { U, S, D } (%), $k = 5$	Train Set (%), $k = 5$
SOFI	-2.3 ± 3.1	-0.3 ± 1.2	$4.26 \cdot 10^9$	-27.84	$41 - 19 - 40$	14.8
NFLX	0.6 ± 1.7	0.05 ± 0.6	$1.58 \cdot 10^{11}$	38.28	$45 - 5 - 50$	21.7
CSCO	0.2 ± 0.7	0.02 ± 0.4	$2 \cdot 10^{11}$	17.59	$18 - 65 - 17$	46.2
WING	-0.3 ± 3.2	-0.04 ± 0.9	$6.06 \cdot 10^9$	96.87	$44 - 7 - 49$	6.1
SHLS	-2.4 ± 4.9	-0.3 ± 1.9	$4.05 \cdot 10^9$	26.24	$42 - 14 - 44$	7.4
LSTR	0.1 ± 2.8	-0.03 ± 0.73	$6.16 \cdot 10^9$	16.55	$48 - 5 - 47$	3.8

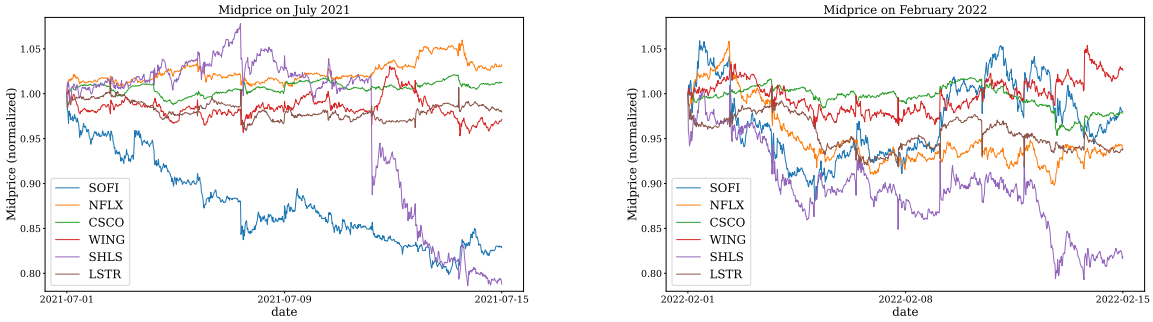


Figure 3.3: Stocks returns from day 0 for LOB-2021/2022.

Stock Processing

We build *LOB-2021* and *LOB-2022* resembling the structure of the FI-2010 dataset, described in the previous section and proposed in [59]. In particular, to generate the LOB-2021/2022 datasets, we utilize data from the LOBSTER data provider, which consists of LOB records (i.e., $\mathbb{L}(t)$ vectors) resulting from events caused by traders at the exchange. LOBSTER associates these records with the specific events that caused changes in the LOB. We isolated the following types of events: order *submissions*, *deletions*, and *executions*, which account for almost all the events in the markets.

For each stock in the set $\mathcal{S}$ we construct a *stock time series* of LOB records $\mathbb{L}_s(t) \in \mathbb{R}^{4L}$, with $L = 10, s \in \mathcal{S}$, N_s being the amount of records of the stock s in the considered temporal interval (e.g., (2021-07-01, 2021-07-15) for LOB-2021), $t \in [1, N_s]$. We recall that the $4 \cdot 10$ features represent the prices and volumes on the buy and sell sides for the ten levels of the LOB. We highlight that the time series $\mathbb{L}_s$ are non-uniform in time since LOB events can occur at irregular intervals driven by traders' actions. We do not impose temporal uniformization. Instead, we sample the market observation every ten events, similarly to the stock processing performed for FI-2010 dataset. Furthermore, we do not account for liquidity beyond the 10th order level in the LOB. This approximation is necessary to ensure computational tractability while retaining the most influential levels. It is a commonly employed technique in stock market prediction models, also employed in FI-2010.

Each stock time series $\mathbb{L}_s$ is split into *training*, *validation*, and *testing* sets using a 6-2-2 days split. Normalization is performed on stock time series using a z-score approach, separately normalizing the prices and volumes. The mean and standard deviation are calculated from the union of the training and validation splits for all stock time series. These statistics are then used to normalize the entire dataset, including the test splits. The final dataset is constructed by vertically stacking (i.e., concatenating along the rows) the six training splits (i.e., one for each stock), six validation

Table 3.4: Class balancing on LOB-2021 and LOB-2022.

	LOB-2021			**LOB-2022**		
Horizon k	**Train Set** {U,S,D} (%)	**Val Set** {U,S,D} (%)	**Test Set** {U,S,D} (%)	**Train Set** {U,S,D} (%)	**Val Set** {U,S,D} (%)	**Test Set** {U,S,D} (%)
1	18 − 63 − 19	19 − 62 − 19	21 − 59 − 20	20 − 60 − 20	18 − 64 − 18	18 − 63 − 19
2	25 − 50 − 25	25 − 50 − 25	27 − 46 − 27	26 − 47 − 27	24 − 51 − 25	25 − 50 − 25
3	28 − 43 − 29	28 − 43 − 29	30 − 40 − 30	30 − 40 − 30	28 − 43 − 29	29 − 42 − 29
5	32 − 35 − 33	32 − 35 − 33	34 − 33 − 33	34 − 31 − 35	33 − 34 − 33	34 − 32 − 34
10	37 − 25 − 38	37 − 25 − 38	38 − 24 − 38	41 − 28 − 41	40 − 20 − 40	41 − 18 − 41

splits, and six test splits in this order.

The dataset is used to extract market observations with a sliding window approach, as explained in Section 3.3. The training set is randomly permuted according to the standard procedure adopted by the SOTA papers in this field.

Labeling market observations is accomplished by leveraging the trend definitions described in Equation (3.2), mapping market observations to the corresponding trend based on a predefined prediction horizon $k \in \mathcal{K}$. It is important to note that a new dataset is generated for each prediction horizon $\mathcal{K}$. Consequently, LOB-2021 and LOB-2022 consist of five (i.e., $|\mathcal{K}|$) distinct datasets, each corresponding to one of the five prediction horizons. As for FI-2010 dataset, we chose the labeling threshold θ of Equation (3.2) such that the resulting dataset is balanced for $k = 5$. In Table 3.4, the class balancing of LOB-2021/2022 datasets for each of the horizons and each split is reported. For a fair comparison, the reported balancing is similar with respect to FI-2010 dataset shown in Table 3.2.

3.5.3 Data Distribution Shift

LOB time series are susceptible to distribution shift due to the dynamic nature of the stock markets. This implies that when used in production, the input data seen by the model may deviate from the data it was trained on [104]. Such shifts can lead to serious consequences that directly translate into misclassification and significant economic losses if the model is not effectively monitored. It is important to note that the primary focus of this work, and the considered studies, is on the trend classification task. Deploying these models into production demands additional effort, particularly in addressing challenges like distribution shift. Being able to observe the market in time allows for new data collection, facilitating continual model retraining, data weighting for past forgetting, and fine-tuning. These methods should be adopted in conjunction with the usual techniques to prevent data overfitting, such as early stopping, data augmentation, and regularization.

3.6 Experiments

We conducted an extensive evaluation to assess the *robustness* and *generalizability* of 15 DL models to solve the SPTP task, as presented in Section 3.3. Among these, 13 were SOTA models, and 2 DL baseline models were commonly used in the literature. More details on the models are given

in Section 3.4 and in Section 3.8.1.

In line with many other studies, we adopt the definition of robustness and generalizability introduced by J. Pineau et al. in their work [56]. Robustness is the ability of a model to replicate its performance when tested on the same data but under different analyses, such as reimplementation of the code, testing of the code on a different computer architecture, and other contextual changes. Generalizability is the ability of a model to replicate its performance when tested on different data and different analytical tools. Robustness is evaluated by testing the proposed models on **FI-2010**, the benchmark dataset employed in all surveyed papers. To evaluate the generalizability, we use **LOB-2021** and **LOB-2022**, retrieved from LOBSTER data provider [32]. In some cases, the authors of the considered works have not provided crucial information, such as the code or the hyperparameters of their models, making reimplementation and hyperparameter search necessary. Our experiments were carried out using **LOBCAST** [105], the open-source framework we developed and made available online. The framework allows the definition of new price trend predictors based on LOB data.

In Section 3.6.1 we describe our framework and its potential applications. In Section 3.6.2 we introduce a complete description of the hyperparameters search. In Section 3.6.3 we discuss the results deriving from the robustness (Section 3.6.3) and generalizability (Section 3.6.3) studies, and a focus to the performance of ensemble methods (Section 3.6.3). In Section 3.6.4 we expand on the performance achieved for the SPTP task when adopting varying labeling strategies (Section 3.6.4) and using non-deep models (Section 3.6.4). We conclude with a profitability study in Section 3.6.4.

3.6.1 LOBCAST Framework for SPTP

We present **LOBCAST** [105], a Python-based framework developed for stock market trend forecasting using LOB data. LOBCAST is an open-source framework that enables users to test DL models for the SPTP task. LOBCAST contains the implementation of the 15 DL models that were used in the experiments. We believe that LOBCAST, along with the advancements in DL models and the utilization of LOB data, has the potential to improve the state of the art on price trend forecasting in the financial domain.

Applications and Features

The core application of LOBCAST is *to provide a standardized benchmarking* of DL-based models for the SPTP task. There are several choices to make while implementing a DL model for SPTP. We collected these choices in LOBCAST to ease the development and evaluation of new models. This not only offers a methodological and standardized approach to addressing the problem but also simplifies the comparison between these choices.

LOBCAST features include: (1) LOB data pre-processing utilities dealing with normalization, splitting, and labeling. (2) A training environment for DL models implemented in PyTorch Lightning [88]. (3) Integrated interfaces with the popular hyperparameter tuning framework WANDB [106], which allows users to tune and optimize model performance efficiently. (4) Generation of detailed reports for the trained models, including performance metrics regarding the learning task (F1-Score, Accuracy, Recall, etc.). (5) Generation of reports measuring the complexity of the models in terms of the number of parameters, inference, and training time. (6) Support for back-

Table 3.5: Hyperparameters adopted in our experiments.

	FI-2010 (Robustness)					LOB-2021/2022 (Generalizability)				
Model	Learning Rate	Optimizer	Batch Size	Epochs	Dropout	Learning Rate	Optimizer	Batch Size	Epochs	Dropout
LSTM	0.001	Adam	32	100	-	0.0001	Adam	64	100	-
MLP	0.001	Adam	64	100	-	0.00001	Adam	64	100	-
CNN1	0.0001	Adam	64	100	-	0.0001	Adam	32	100	-
CTABL	0.01	Adam	256	200	-	0.001	Adam	64	200	-
DAIN	0.0001	RMSprop	32	100	0.5	0.0001	RMSprop	64	100	0.5
DEEPLOB	0.01	Adam	32	100	-	0.01	Adam	32	100	-
CNNLSTM	0.001	RMSprop	32	20	0.1	0.001	RMSprop	128	100	0.1
CNN2	0.001	RMSprop	32	100	-	0.001	RMSprop	128	100	-
TRANSLOB	0.0001	Adam	32	150	-	0.001	Adam	128	100	-
TLONBoF	0.0001	Adam	128	100	-	0.00001	Adam	32	100	-
BINCTABL	0.001	Adam	128	200	-	0.001	Adam	32	200	-
DEEPLOBATT	0.001	Adam	32	100	-	0.0001	Adam	128	100	-
AXIALLOB	0.01	SGD	64	50	-	0.01	SGD	64	50	-
ATNBoF	0.001	Adam	128	80	0.2	0.00001	Adam	32	80	0.2
DLA	0.01	Adam	256	100	-	0.001	Adam	64	100	-
METALOB	0.0001	SGD	64	100	-	0.0001	SGD	64	100	-

testing for profit analysis, utilizing the Backtesting.py[2] library. (7) The PyTorch implementation of the SOTA models used in the experiments.

All these features are implemented through a modular and scalable framework that can be easily expanded with new models and components.

3.6.2 Hyperparameters Search

For evaluating the *robustness* of the surveyed models, we used the hyperparameters reported in the original papers whenever they were available. However, we encountered cases where hyperparameters were not declared at all, such as in LSTM [86] and CNN1 [94], while in other cases, including CNNLSTM [87], AXIALLOB [102], ATNBOF [101] and DAIN [96] only partial information was provided. To address these gaps, we performed a grid search exploring different values for the **batch size**, including $\{16, 32, 64, 128, 256\}$ and the **learning rate**, including $\{0.01, 0.001, 0.0001, 0.00001\}$.

Regarding the *generalizability* experiment, we found that the majority of models using the hyperparameters from the robustness experiment performed poorly on the LOB-2021/2022 datasets. So, we conducted a comprehensive hyperparameter search on horizon $k = 5$ (which is the most balanced) using a grid search approach for all 16 models. For this search, we maintained the same number of epochs and optimizer used in the robustness analysis while searching for batch size and learning rate using the same domains mentioned above. F1-Score maximization over the validation set was the chosen criterion for optimizing the hyperparameters. For a complete overview of the hyperparameters utilized in our experiments, refer to Table 3.5.

3.6.3 Performance, Robustness and Generalizability

To test robustness and generalizability, we conducted our experiments for each model using five different seeds to mitigate the impact of random initialization of network weights and training dataset shuffling. The necessity to produce reliable results within reasonable time constraints led us to opt for a set of five seeds. The training process involved training the 15 models for each seed on each of the considered prediction horizons ($\mathcal{K} = \{1, 2, 3, 5, 10\}$). On average, the training process for all the models took approximately 155 hours for FI-2010 and 258 hours for each LOB dataset, utilizing a cluster comprised of 8 GPUs (1 NVIDIA GeForce RTX 2060, 2 NVIDIA GeForce RTX 3070, and 5 NVIDIA Quadro RTX 6000).

[2]https://kernc.github.io/backtesting.py/

Table 3.6: Robustness, generalizability, and performance scores of the models. Arrows indicate whether the measured F1-Score of a system is higher or lower than stated in the original paper. Color saturation highlights systems with the best (green) and worst (red) robustness and generalizability scores.

	FI-2010				LOB-2021			LOB-2022		
Model	**F1 Claim**	**F1 LOBCAST**	**F1 Rank**	**Rob. Score (%)**	**F1 LOBCAST**	**F1 Rank**	**General. Score (%)**	**F1 LOBCAST**	**F1 Rank**	**General. Score (%)**
MLP	51.8 ± 3.2	↓ 48.0 ± 2.6	14	91.8	↑ 55.5 ± 3.9	14	95.0	↑ 53.1 ± 2.5	13	96.6
LSTM	63.4 ± 2.1	↓ 63.4 ± 3.6	7	95.5	↓ 56.9 ± 4.1	11	85.9	↓ 56.1 ± 2.8	9	88.8
CNN1	57.9 ± 1.9	↑ 58.1 ± 13.1	10	80.9	↓ 57.5 ± 3.0	8	97.0	↓ 57.1 ± 2.7	6	99.3
CTABL	74.3 ± 5.2	↓ 69.6 ± 4.3	5	91.3	↓ 59.7 ± 2.7	3	78.4	↓ 58.1 ± 3.3	5	78.4
DEEPLOB	78.9 ± 4.4	↓ 71.4 ± 5.3	4	87.6	↓ 59.5 ± 3.0	4	73.7	↓ $\mathbf{59.5 \pm 2.9}$	**1**	74.7
DAIN	66.8 ± 1.5	↓ 55.6 ± 5.9	11	81.4	↓ 55.9 ± 4.4	12	79.5	↓ 54.1 ± 2.1	12	83.9
CNNLSTM	47.0 ± 0.0	↑ 63.2 ± 8.4	8	75.7	↑ 57.0 ± 3.3	10	87.8	↑ 56.8 ± 2.5	7	90.3
CNN2	45.0 ± 0.8	↑ 50.5 ± 17.3	12	70.5	↑ 55.5 ± 3.5	13	86.6	↑ 55.8 ± 3.2	10	88.6
TRANSLOB	$\mathbf{87.3 \pm 4.0}$	↓ 59.4 ± 2.6	9	69.9	↓ 57.7 ± 2.9	7	64.2	↓ 50.4 ± 6.1	14	56.4
TLONBoF	53.0 ± 0.0	↓ 49.7 ± 10.5	13	81.5	↑ 57.3 ± 2.9	9	**99.1**	↑ 54.2 ± 3.1	11	**99.9**
BINCTABL	80.1 ± 6.9	↑ $\mathbf{82.6 \pm 7.0}$	**1**	**99.7**	↓ $\mathbf{61.2 \pm 2.7}$	**1**	73.5	↓ 59.2 ± 3.3	2	72.3
DEEPLOBATT	78.8 ± 3.1	↓ 67.3 ± 9.0	6	81.2	↓ 60.1 ± 3.0	2	75.7	↓ 58.9 ± 2.8	3	74.5
DLA	78.7 ± 0.7	↓ 73.4 ± 12.1	2	93.2	↓ 57.7 ± 3.7	6	74.9	↓ 56.6 ± 2.4	8	76.9
ATNBoF	67.1 ± 5.5	↓ 40.9 ± 7.7	15	66.1	↓ 53.1 ± 3.7	15	80.9	↓ 48.0 ± 6.9	15	81.2
AXIALLOB	82.0 ± 3.7	↓ 73.4 ± 5.7	3	88.2	↓ 59.5 ± 3.3	5	71.3	↓ 58.6 ± 2.6	4	70.7
METALOB	–	82.2 ± 7.3	–	–	55.9 ± 2.6	–	–	53.2 ± 1.5	–	–
MAJORITY	–	60.0 ± 12.7	–	–	55.5 ± 2.3	–	–	47.9 ± 2.0	–	–

In Table 3.6, we summarize the results of our experiments. Our choice of F1-Score as the evaluation metric was motivated by the following reasons: (1) it captures both precision and recall in a single value, (2) the datasets are not well balanced, and F1-Score is robust to the class imbalance problem that affects the accuracy measure, (3) it is the only metric that is reported in every SOTA paper. The Table compares the claimed performance of each system (column F1 Claim) with that measured in the robustness (FI-2010) and generalizability (LOB-2021 and 2022) experiments. For each dataset, we show the average performance and the standard deviation achieved by each model in all the horizons, along with its rank.

To evaluate the robustness and the generalizability of the models, we compute the **robustness** and the **generalizability scores**, a value ≤ 100 that is computed as $100 - (|A| + S)$, where A and S are defined as follows. A is the average difference between the F1-Score reported in the original paper and the one that we observed in our experiments on FI-2010 for robustness, and on LOB-2021 and LOB-2022 for generalizability. S is the standard deviation of these differences. The score penalizes models that demonstrate higher variability in their performance by subtracting the standard deviation. The average and standard deviation were computed over the declared horizons for each model and considering all five seeds.

Table 3.6 clearly highlights the following:

1. Except for a few systems, there is a considerable difference between the claimed performances and those measured in both robustness and generalizability experiments. Note that while the performance gap is negative on average and considerably negative in the scenario of LOB-2021 and 2022, a few systems outperform the claimed results, as highlighted by the arrows in Table 3.6.
2. All models are very sensitive to hyperparameters; in fact, for about half of the runs, they diverged (F1-Score $\leqslant 33\%$) during the hyperparameters search.
3. The ranking of the systems changes considerably if we compare the declared performances

with those measured in our experiments. On the other hand, the best six systems in FI-2010 remain the same in LOB-2021 and 2022.

4. The best-ranked systems do not consistently hold the lead in terms of robustness and generalizability - except for BINCTABL. On the contrary, some of them obtained poor generalizability scores, suggesting that they overfitted the FI-2010 dataset.
5. Five of the best six models incorporate attention mechanisms. In particular, the best-performing model is BINCTABL, which enhances the original CTABL model by adding an Adaptive Bilinear Normalization layer, enabling joint normalization of the input time series along both temporal and feature dimensions. On average, BINCTABL improves the F1-Score by up to 9.2% compared to DLA, i.e., the second-best model, and up to 13% compared to CTABL.
6. Regrettably, ensemble models (the last two rows in Table 3.6) do not exceed the performance of the top-performing models, which is probably due to the relatively high agreement rate among systems, as shown in Figure 3.13 and Figure 3.14 in Section 3.8.2 of the supplementary material.

Robustness on FI-2010

As far as the robustness experiments are concerned, it is important to note that some models discussed in the literature incorporate additional market observation features for predictions. This is the case for the models DAIN, CNNLSTM, TLOBOF, and DLA. To ensure a fair comparison among the models, we included them in our study but reduced their feature set to only the 40 raw LOB features. Due to the presence of these additional features, a strict robustness study could not be conducted for these models. However, the reduction of features did not necessarily cause a deterioration in performance: of particular interest is the case of CNNLSTM, for which the authors used stationary features derived from the LOB, stating that they were better than the raw ones. Impressively, CNNLSTM achieves the greatest average improvement of 20.9% among all the models, proving that, for this model, the raw LOB features are better suited to forecast the mid-price movement than the features proposed by the original authors.

Based on these experiments (summarized in Table 3.6), the BINCTABL model demonstrates the **highest F1-Score** when averaged over the seeds and prediction horizons, achieving an average of $82.6\% \pm 7.0$. Notably, the BINCTABL model also exhibits the strongest robustness score of 99.7. For a more comprehensive analysis, Figure 3.4 provides the confusion matrices of the BINCTABL model's predictions for three horizons ($k = 1$, $k = 5$, and $k = 10$). The confusion matrices demonstrate that the model is slightly biased toward the stationary class. This pattern is consistent across all the models, especially for the first three horizons, reflecting the imbalance of the dataset towards the stationary class, as specified in Section 3.5.

Remarkably, a significant number of models in our study failed to achieve the claimed performance levels. Two possible reasons are the lack of the original code and the missing hyperparameters declaration. Among the models, TRANSLOB and ATNBOF exhibit the largest discrepancies, ranking as the second and first worst performers, respectively. Notably, ATNBoF performs the poorest among all models, both in terms of robustness score and F1-Score.

We observed that CNN1, CNN2, CNNLSTM, TLONBOF, and DLA are the most sensitive models in terms of network weight initialization and dataset shuffling; in fact, these models exhibit

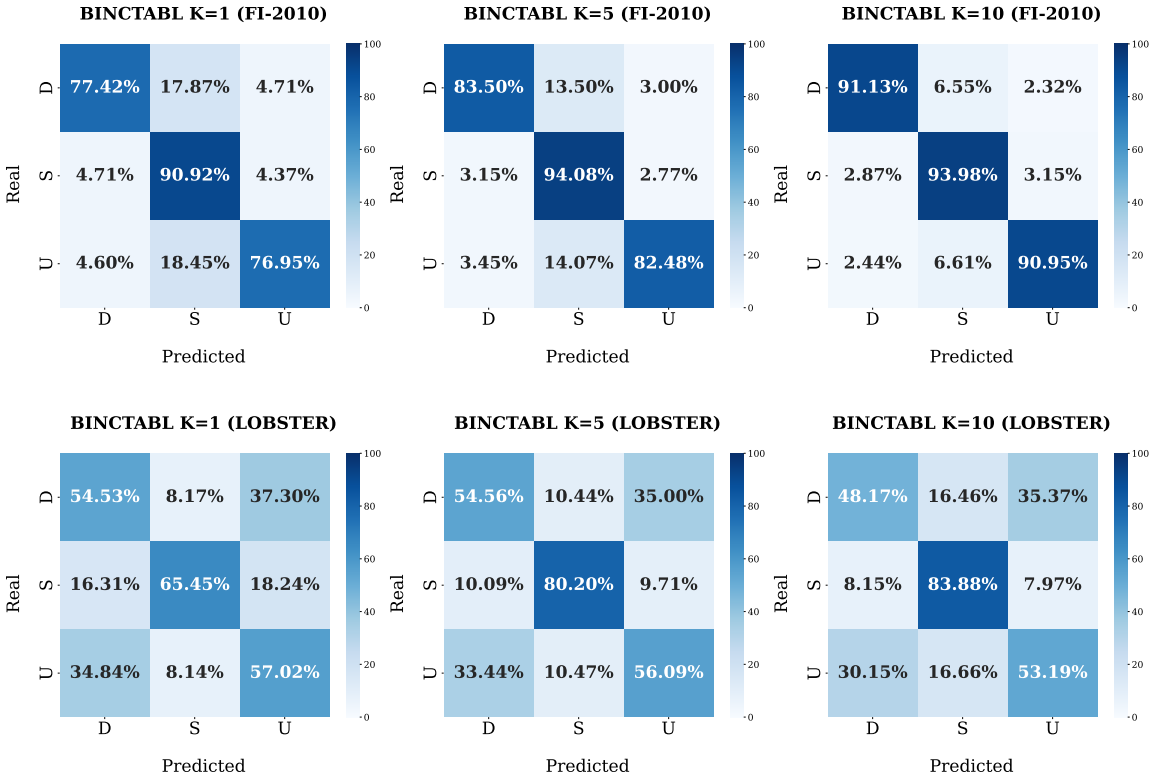
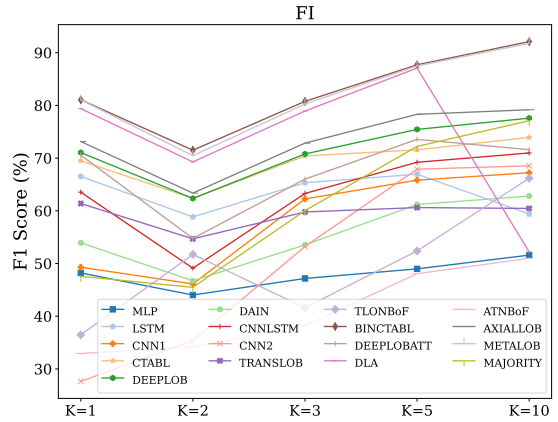


Figure 3.4: Confusion matrices for BINCTABL ($k = 1, 5, 10$) on FI-2010 and LOB-2021 datasets.

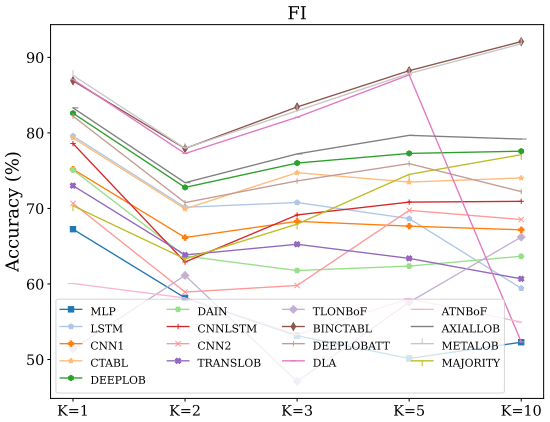
a standard deviation over the runs that exceeds 5 basis points, indicating a high degree of variability in their performance. Finally, we highlight that none of the top three models in our study utilize a temporal shape $h = 100$ for market observations as input, despite it being a common practice in the literature [14, 86, 94, 97, 101], meaning that they are able to achieve good results without relying on a large historical context. This suggests that the most influential and relevant dynamics impacting their predictions tend to occur within a short time frame.

Figure 3.5 depicts the F1-Score, Accuracy, Precision, Recall, and MCC of the surveyed models obtained through LOBCAST, for the time horizons $\mathcal{K} = \{1, 2, 3, 5, 10\}$. Most of the models show similar behaviour with respect to the prediction horizons. Specifically, concerning the F1-Score, the lowest performance is noted when $k = 2$. However, an improvement is evident as the value of k increases, indicating a longer prediction horizon. This may appear counterintuitive, as higher values of k imply forecasting the price trend further into the future. However, for very short horizons, the labeling system adopted may be susceptible to noise, affecting the model's capability to extract relevant patterns. This hypothesis is supported by an experiment in [14], in which the authors tried a smoother labeling method and reported a significant decline in the performance of their deep learning model as the prediction horizons increased.

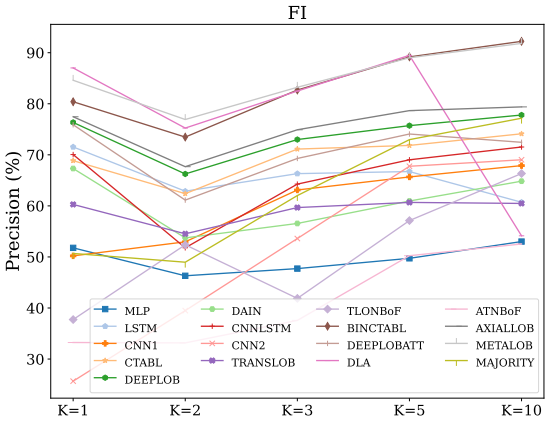
Figure 3.6 shows the PR-curves of all models for the time horizon $k = 10$, where classes $\{\texttt{U}, \texttt{S}, \texttt{D}\}$ are distributed as $\{37\%, 25\%, 38\%\}$. Since the models perform a ternary classification, each curve represents the micro-average precision-recall value and is generated by setting different thresholds for the classification. Thresholds play a role in defining the number of false negatives and false positives, affecting the resulting values of the Precision and the Recall. The best models are the ones with the largest area under the curve, as they are able to make the most accurate predictions



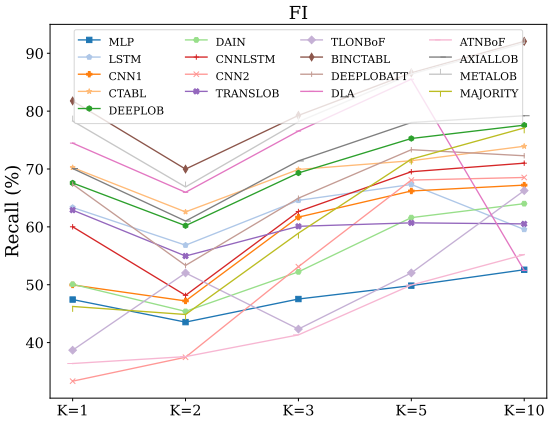
(a) F1-Score



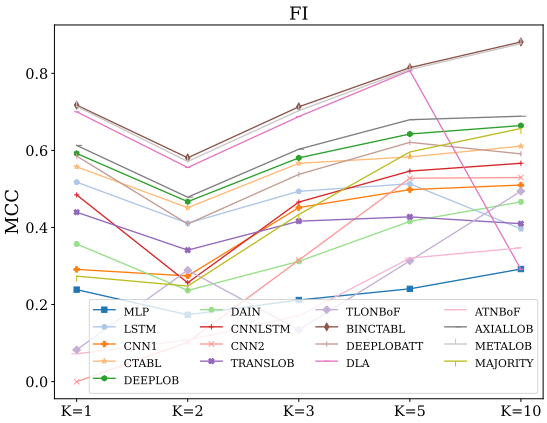
(b) Accuracy



(c) Precision



(d) Recall



(e) MCC

Figure 3.5: Evaluation metrics on different horizons $\mathcal{K}$ on FI-2010 dataset.

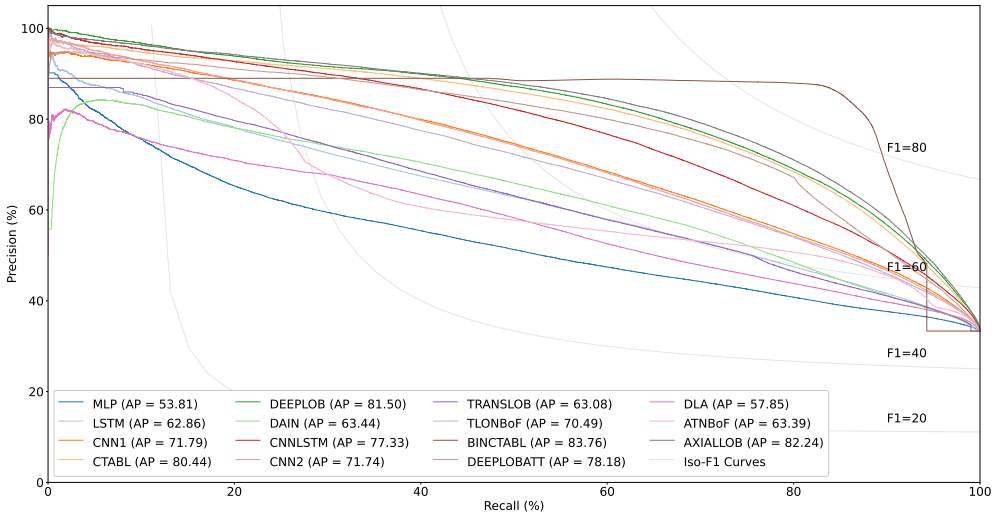


Figure 3.6: PR-Curve on FI-2010 for time horizon $k = 10$.

(high Precision) while minimizing the false negative rate (high Recall). The figure also shows the iso-F1 Curves on the PR plane. The best-performing model is BINCTABL, with an area under the curve of 8680.

To further compare the performance of the models, we also conducted a T-test for each couple of models reporting the p-values in Table 3.10 (Section 3.8.2). The sample of scores for each model is made up of the F1-Score varying the random seed. We state the null hypothesis h_0 as: *there is no statistical difference between the average performance of the two models*. We highlight in bold the values exceeding a threshold $\alpha = 0.05$, i.e. the couple of models for which h_0 is accepted, that is, the difference in performance has statistical significance.

In Section 3.8.2, we show a different representation of the models' performances at varying prediction horizons and the agreement matrix of their predictions.

Generalizability on LOB-2021/2022

When comparing the performance of models on the FI-2010 and LOB-2021/2022 datasets, we observe that models showing high performance on the FI-2010 dataset demonstrate a deterioration in performance. Conversely, some of the models that performed poorly on the FI-2010 dataset show an improvement in performance on the LOB-2021/2022 datasets. However, the overall performance of all models on the LOB-2021/2022 dataset is still significantly lower than on the FI-2010 dataset, ranging 48-61% in F1-Score. Furthermore, we conjecture that the overall performance is worse in LOB-2022 than in LOB-2021 due to the higher stocks' volatility. We mention two potential factors contributing to this observed phenomenon. Firstly, the LOB-2021/2022 datasets present a higher level of complexity than the FI-2010 dataset despite having been generated with a similar approach. Indeed, NASDAQ is a more efficient and liquid market than the Finnish one, as evidenced by the fact that LOB-2021/2022 datasets have approximately three times the size of FI-2010 in terms of events for the same period length. Secondly, the best-performing models may overfit the FI-2010 dataset, leading to a decrease in their performance when applied to LOB-2021/2022 datasets. In particular, BINCTABL experiences an average decrease of approximately 19.6% in F1-Score across all horizons, resulting in a generalizability score of 73.5%. In Figure 3.7, we present the results of our tests for the time horizon $k = 5$ on each individual stock from LOB-2021 dataset. Among the

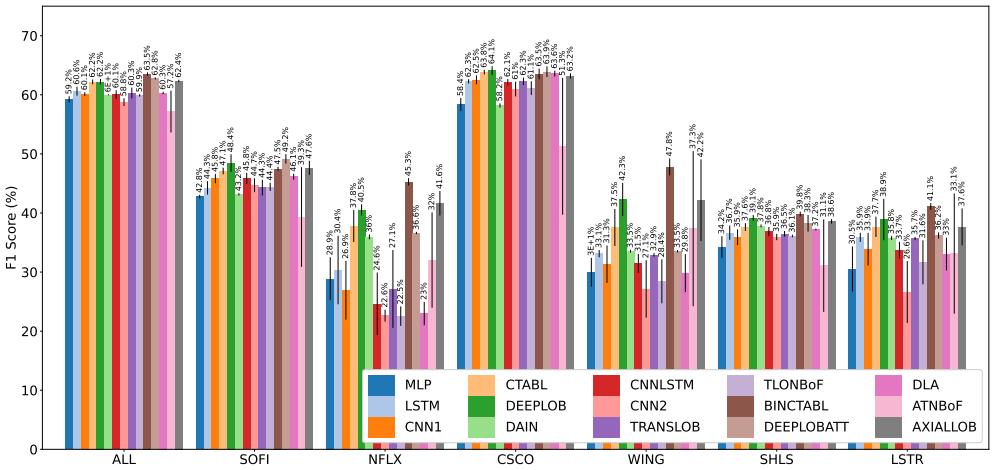


Figure 3.7: F1-Score per stock, time horizon $k = 5$, on LOB-2021.

tested models, CSCO stands out as yielding the highest performance. This may be attributed to the high stationarity of CSCO (balance 18-65-17% in the train set, see Table 3.3), indicating more stable and predictable behavior.

This hypothesis is supported by the confusion matrices in Figure 3.4, which consistently show the best performance in the stationary class across all models; we reported only those of BINCTABL since all other models show similar patterns.

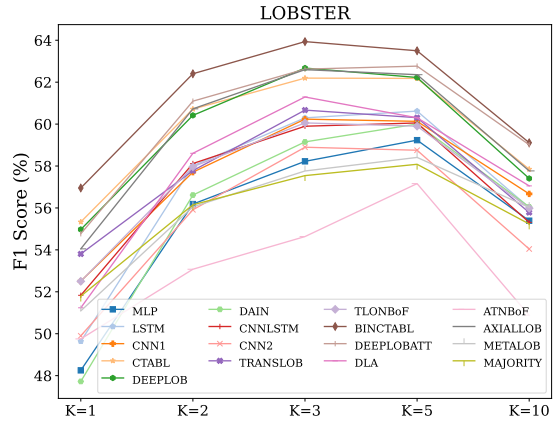
We highlight that extracting the per-stock information on the FI-2010 dataset was impossible because it was already assembled, and the authors did not provide information on that procedure. We also show the performance of the models on LOB-2021, including F1-Score, Accuracy, Precision, Recall, and MCC, which are displayed in Figure 3.8.

Figure 3.9 shows the PR-Curves for the time horizon $k = 10$ on LOB-2021. With respect to the same plot on FI-2010 in Figure 3.6, the performance of all methods is more similar to one another, which is in line with the findings reported in Figure 3.8. The best-performing method is CTABL, with an area under the curve of 5379.5, which only slightly differs from the other top-performing models. On average, the integral of the curves is 5279.4. This result highlights that the models are less reliable on the LOB-2021 dataset and fall afoul of misclassifying price trends.

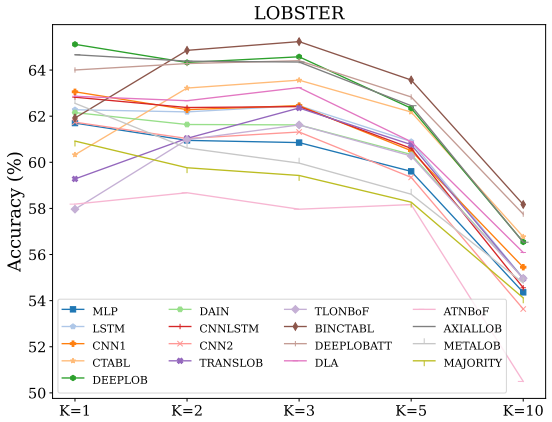
Table 3.11 shows the p-values of the T-test on LOB-2021 for the horizon $k = 5$. We performed this test following the same approach used for the results shown in Table 3.10. Bold p-values correspond to high accordance among the models. Notice that, while on the FI-2010 dataset, there are only nine couples of models with a p-value exceeding the threshold $\alpha = 0.05$, there are as many as 41 pairs of models that are not considerably statistically different on LOB-2021. These results confirm the results depicted in Figures 3.8 and 3.9.

The plots for LOB-2022 are omitted since they show similar properties; indeed, we observe that most models exhibit a similar trend in both LOB-2021 and LOB-2022 datasets. However, the performance curves in these generalizability tests differ from the results obtained on the FI-2010 dataset, shown in Figure 3.5. Specifically, for the LOB-2021/2022 datasets, the F1-Score of most models shows an increasing trend as the prediction horizon increases up to $k = 3$, after which it starts to decrease.

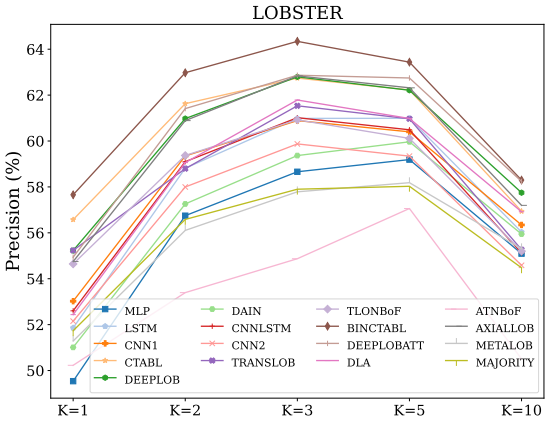
To ease readability, in Table 3.7 we report the F1-Score of all the models, horizons, and periods. The performance of the models, as reported by the authors of the selected paper, exhibits changes



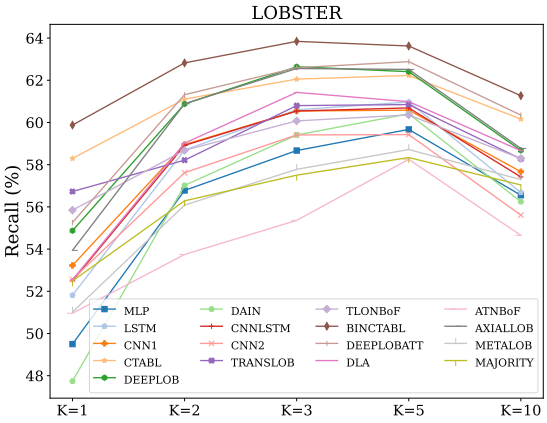
(a) F1-Score



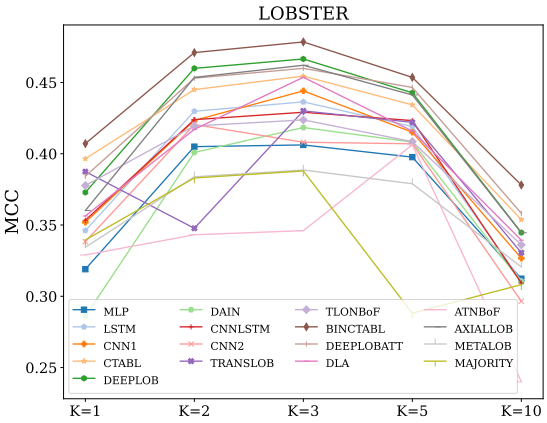
(b) Accuracy



(c) Precision



(d) Recall



(e) MCC

Figure 3.8: Evaluation metrics on different horizons K on LOB-2021.

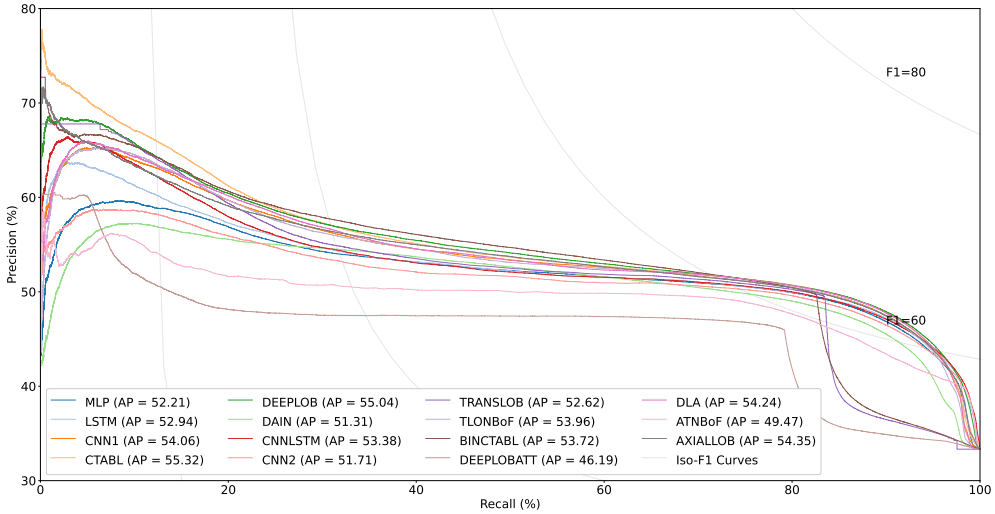


Figure 3.9: PR-Curve on LOB-2021 for time horizon $k = 10$.

Table 3.7: F1-Score on LOB-2021 and LOB-2022. Columns FI 2010, FI^r 2010, LOB 2021, and LOB 2022, respectively, represent the claimed performance of the models in the respective papers, the performance reproduced with LOBCAST on FI, LOB-2021, LOB-2022.

	$k=1$				$k=2$				$k=3$				$k=5$				$k=10$			
Model	**FI 2010**	**FI^r 2010**	**LOB 2021**	**LOB 2022**	**FI 2010**	**FI^r 2010**	**LOB 2021**	**LOB 2022**	**FI 2010**	**FI^r 2010**	**LOB 2021**	**LOB 2022**	**FI 2010**	**FI^r 2010**	**LOB 2021**	**LOB 2022**	**FI 2010**	**FI^r 2010**	**LOB 2021**	**LOB 2022**
MLP	48.3	48.2	48.3	51.1	51.1	44.0	56.2	54.1	–	47.2	58.2	55.9	56.0	49.0	59.2	55.0	–	51.6	55.4	49.3
LSTM	66.3	66.5	49.6	53.7	62.4	58.8	58.0	57.4	–	65.3	60.3	60.6	61.4	66.9	60.6	56.2	–	59.4	56.0	52.6
CNN1	55.2	49.3	52.5	55.3	59.2	46.1	57.7	59.8	–	62.3	60.2	59.3	59.4	65.8	60.1	58.5	–	67.2	56.7	52.6
CTABL	77.6	69.5	55.3	57.8	66.9	62.4	60.7	60.9	–	70.4	62.2	60.8	78.4	71.6	62.2	58.8	–	73.9	57.8	52.0
DEEPLOB	83.4	71.1	55.0	57.0	72.8	62.4	60.4	**62.0**	–	70.8	62.7	**62.4**	80.4	75.4	62.2	60.8	–	77.6	57.4	55.2
DAIN	68.3	53.9	47.7	52.2	65.3	46.7	56.6	54.9	–	53.5	59.1	55.8	–	61.2	60.0	56.5	–	62.8	56.1	51.2
CNNLSTM	47.0	63.5	51.8	55.0	–	49.1	58.1	59.8	–	63.3	59.9	59.2	47.0	69.2	60.1	57.1	47.0	71.0	55.3	53.1
CNN2	46.0	27.6	49.9	51.9	–	35.4	55.9	59.0	–	53.2	58.9	58.7	45.0	67.9	58.8	57.3	44.0	68.5	54.0	52.0
TRANSLOB	**88.7**	61.4	53.8	43.7	**80.6**	54.7	57.8	43.0	–	59.8	60.7	57.5	**88.2**	60.6	60.3	56.6	**91.6**	60.5	55.8	51.0
TLONBoF	53.0	36.5	52.5	53.1	–	51.7	58.0	56.5	–	41.6	60.1	57.1	–	52.4	59.9	55.7	–	66.2	56.0	48.5
BINCTABL	81.0	**81.1**	**57.0**	**58.4**	71.2	**71.5**	**62.4**	**62.0**	–	**80.8**	**63.9**	62.2	88.1	**87.7**	**63.5**	60.4	–	**92.1**	**59.1**	53.2
DEEPLOBATT	82.4	70.6	54.8	55.8	73.7	54.8	61.1	60.5	76.9	66.0	62.6	62.1	79.4	73.6	62.8	**60.9**	81.5	71.6	59.0	**55.3**
DLA	77.8	79.4	51.2	54.4	–	69.3	58.6	58.0	79.4	78.9	61.3	60.0	79.0	87.1	60.3	57.3	–	52.2	57.1	53.4
ATNBoF	67.9	32.9	49.8	47.8	60.0	34.2	53.1	50.3	–	38.2	54.6	41.3	73.4	48.1	57.2	59.8	–	51.0	50.9	40.9
AXIALLOB	85.1	73.2	54.0	56.9	75.8	63.4	60.7	60.1	**80.1**	72.8	62.6	62.0	83.3	78.3	62.4	59.6	85.9	79.2	57.8	54.6
METALOB	–	81.1	51.1	52.3	–	70.5	56.1	53.3	–	80.3	57.8	55.3	–	87.5	58.4	54.5	–	91.8	56.0	50.9
MAJORITY	–	47.1	51.8	50.6	–	44.9	56.2	49.2	–	59.7	57.5	48.1	–	71.8	56.9	46.7	–	76.3	55.2	44.7

when evaluated on the LOB-2021 and LOB-2022 datasets. These changes show varying degrees of generalizability among the models. Notably, the ATNBoF model demonstrates the most substantial improvement with respect to the declared performances, showing an average increase of 12.2% across all prediction horizons. A similar improvement is exhibited by MLP and TLONBoF. Despite this improvement, ATNBoF still exhibits the lowest overall performance, with an average score of 53.1%. It is worth mentioning that ATNBoF is the most sensitive to random initialization. In contrast, the other models experience a significant decline in performance when evaluated on LOB-2021 and LOB-2022 datasets. For example, the previously best-performing model on the FI-2010 dataset, BINCTABL, shows an average decrease in F1-Score of approximately 19.6% across all prediction horizons. This decline results in a generalizability score of 73.5% (as mentioned in Table 3.6). However, despite this decline, BINCTABL remains the top-performing model when evaluated on the LOB-2021 dataset on almost all the prediction horizons. On these datasets, it exhibits similar performance to DEEPLOB and DEEPLOBATT models.

In Section 3.8.2, we show the agreement matrix of the models' predictions.

Ensemble Method Discussion

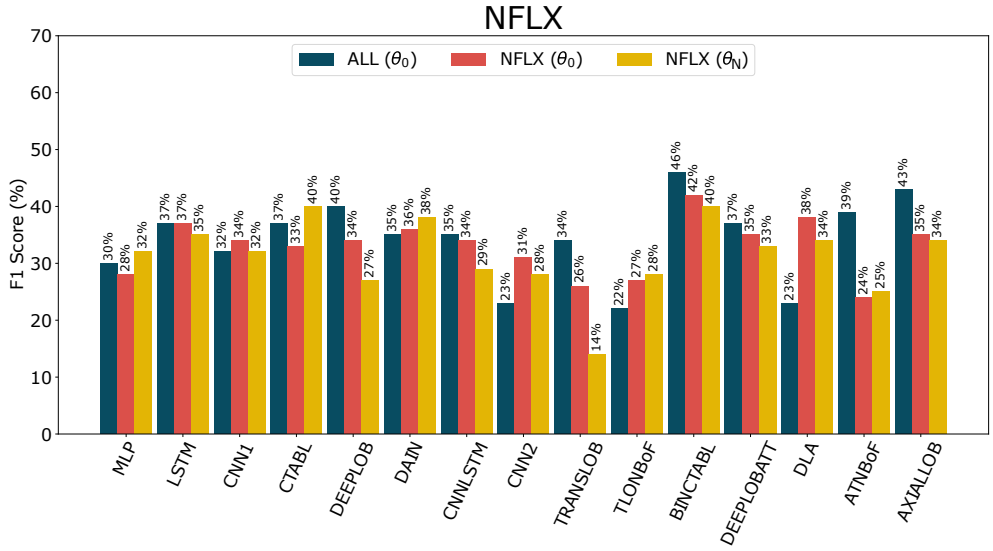
To train *METALOB* without falling into overfitting, we divided the test set of LOB-2021/2022 into three distinct subsets. We allocated 70% of the data for training, 15% for validating, and the remaining 15% for testing the meta-classifier. By implementing these ensemble methods, our objective was to leverage the collective intelligence of ensemble models and potentially achieve performance that surpasses that of individual models. Unfortunately, the ensemble models did not achieve the expected level of performance, as they failed to surpass the performance of the best individual models. A plausible explanation for this phenomenon is the relatively high degree of consensus among the systems, as evidenced by Figure 3.13 and Figure 3.14 in Section 3.8.2 of the supplementary material. Moreover, it is likely that the methods converge on cases that are easy to classify and diverge on cases that are difficult to classify.

3.6.4 Additional Experiments: Labeling, non-DL Models & Profit

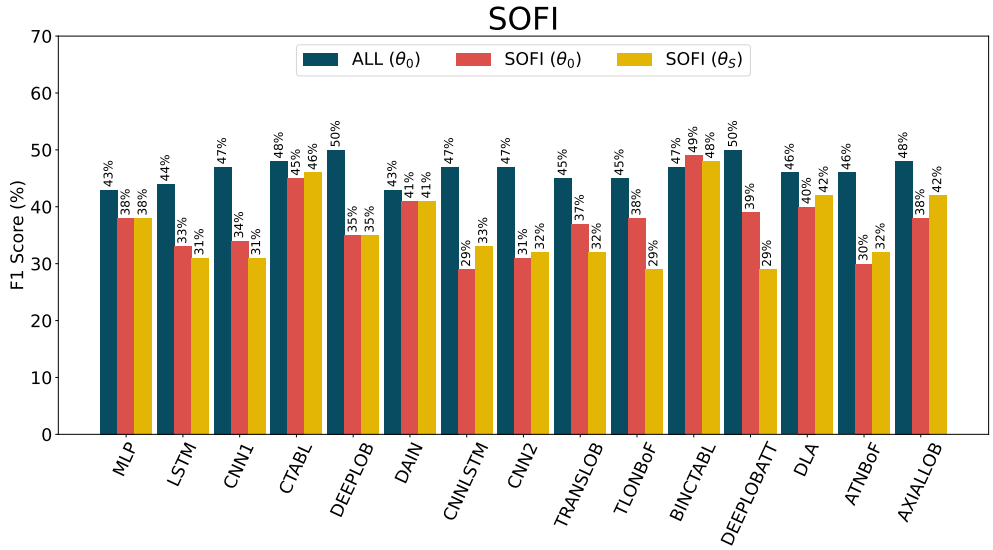
In this section, we delve into additional experiments. In Section 3.6.4, we measure the impact of labeling parameters on the quality of the SPTP task. In Section 3.6.4, we go beyond deep learning models and explore how tree-based methods perform on the SPTP task using the same experimental setting presented in the previous section. In Section 3.6.4, we incorporate profit considerations through backtesting.

Labeling

The experiments analyzed in Section 3.6 highlight that the models' performance does not exhibit a clear trend with respect to the prediction horizon. The labeling method is probably the cause of this phenomenon; in fact, classifying trends based on the mid-price tends to embody noise on the nearest horizons. This hypothesis is supported by the work of Zhang et al. [14]; specifically, they generated a dataset using an alternative labeling method that relies on the mean of the previous and next k mid-prices to identify trends. Interestingly, they show an inverse trend in performance with respect to the horizons: the best performance is achieved with the shortest horizon and deteriorates when



(a) NFLX



(b) SOFI

Figure 3.10: Different labeling strategies on NFLX and SOFI stocks for $k = 5$.

it increases. While exploring various labeling techniques is beyond the scope of this benchmark, we provide an initial investigation in this direction. Specifically, focusing on $k = 5$ in LOB-21, we select two stocks, NFLX and SOFI.

Based on Equations (3.1) and (3.2), we can define θ_N and θ_S as the thresholds that balance the occurrences of the classes for the stocks NFLX and SOFI, respectively. Similarly, we can define θ_0 as the threshold that balances the occurrences of the classes for the ensemble of six stocks within the dataset.

Figure 3.10 shows the results of three different training settings: (i.) **ALL (θ_0)** represents the training of the models over the ensemble of all the six stocks using the threshold θ_0; (ii.) **NFLX (θ_0)** (**SOFI (θ_0)**) represents the training of the models over NFLX (SOFI) stock using the threshold θ_0. (iii.) **NFLX (θ_N)** (**SOFI (θ_S)**) represents the training of the models over NFLX (SOFI) stock using the threshold θ_N.

In the case of SOFI, all methods, except for BINCTABL, achieve the highest performance in the **ALL (θ_0)** setting. This indicates that these models are able to extract useful signals from other stocks, reducing overfitting and improving overall performance. On the other hand, comparing the **SOFI (θ_0)** and **SOFI (θ_S)** settings does not provide significant insights. This suggests that the balancing of the three classes is not crucial for achieving higher performance. This is even more the case for NFLX in Figure 3.10a, considering that the imbalance due to θ_0 is much higher (see Table 3.3).

These results indicate that the labeling mechanism should be revised from its current definition and be agnostic with respect to the balancing involved. Therefore, trend definition should not solely depend on the magnitude of the future price shift relative to the current price. Other factors, such as persistence over time and volume considerations, should also be taken into account. A more comprehensive discussion of the limitations and challenges associated with the labeling mechanisms can be found in the final discussion and conclusions section.

Random Forest & XGBoost for SPTP

To test the quality of the predictions of DL over non-DL models, we conducted an empirical investigation focusing on the predictive capabilities of two of the most popular non-DL models: Random Forests and XGBoost. These experiments are motivated by the results presented in some previous works (e.g., [43, 107]), which show that some tree-based models outperform recently proposed DL models on tabular data. Employing the standard experimental setup detailed in Section 3.6 and carefully tuning hyperparameters (refer to Table 3.8 and Table 3.9), using also class weights parameters, our analysis revealed an F1-Score of **51%** for the Random Forest model and **65%** for XGBoost on the FI-2010 dataset with horizon $k = 5$. We acknowledge that for Tree-based algorithms, standardization might lead to worse performance, but as said before, the FI-2010 dataset is already released in a normalized form, and to guarantee a fair comparison, we decided to apply standardization to the LOB 2021/22 dataset.

As illustrated in Figure 3.5a and Figure 3.12, our results indicate a competitive performance of these non-DL models when compared to several DL models, including ATNBoF, MLP, TLONBoF, TRANSLOB, and DAIN. However, the hypothesis that non-DL models outperform DL counterparts in this specific task does not seem to hold. While non-DL models exhibit notable performance, DL methods show a substantial advantage in predicting price trends. We recall that the F1-Score of

Table 3.8: Random Forest parameters.

Hyper Parameter	Values
`n_estimators`	[200, 400, 600, **800**, 1000, 1200, 1400, 1600, 1800, 2000]
`max_depth`	[**10**, 25, 50, 75, 100]
`min_samples_leaf`	[1, **2**, 4]
`min_samples_split`	[**2**, 5, 10]
temporal shape (h)	[**5**, 10, 15, 50, 100, 300]

Table 3.9: XGBoost parameters.

Hyper Parameter	Values
`n_estimators`	[100, 250, 500, 750, 1000, **1250**]
`max_depth`	[3, 4, 5, **6**, 7, 8, 9, 10]
`booster`	**gbtree**
`eta`	[0.01, **0.1**, 0.2, 0.3, 0.4]
`min_child_weight`	[0, 2, **4**, 6, 8]
`colsample_bytree`	[0.5, **0.75**]
`colsample_bylevel`	[**0.5**, 0.75]
temporal shape (h)	[**5**, 10, 15, 50, 100, 300]

the best-performing model on the FI-2010 with $k = 5$ was 87.7%, obtained by BINCTABL. This is in line with the results of the experiments in Nti et al. [63]. The tabular nature of LOB data, with its geometric properties, such as local dependence [108], and visual indicators embedded in column positions of the LOB, seems to align better with the strengths of DL models, especially those based on convolution. Furthermore, we remark that LOB data are multivariate time series, and SOTA forecasting papers in this domain are primarily dominated by deep learning models [73, 109, 110].

Profit Analysis

As a final benchmark test, we conducted a trading simulation using our framework, relying on Backtesting.py Python library [3]. As highlighted by [75], most of the existing literature in the SPTP field neglects backtesting, even though it is essential for evaluating the performance of algorithmic trading strategies and for potential real-world use.

We performed backtesting using the same period as the test set of the LOB-2021 dataset, i.e., from 2021-07-13 to 2021-07-15. To perform backtesting, we generated an Open High Low Close (OHLC) time series with a 10-event period. The OHLC is an aggregation technique to summarize periods of a time series, e.g., minutes, hours, days, or a number of events (10 in this case). Each data point of the series represents four aggregates of the considered period. The *Open* represents the first price of the period; *High* is the highest price of the period; *Low* is the lowest price of the period; *Close* is the last price of the period.

We underline that the use of LOB data is most often associated with High-Frequency Trading (HFT), i.e., strategies that analyze this data in real time to make split-second decisions about trade executions. We remark that a trading action (buy/sell/hold) is taken every ten events, so at the end of the backtesting simulation, for each stock, hundreds of thousands of orders are placed and filled.

We base our trading simulation on the methodology of the seminal paper [14] in this field, in which the authors conducted a similar experiment. We established certain parameters for our simulation. Firstly, we set the number of shares per trade to a fixed value of 1, simplifying our analysis and assuming a negligible market impact. Furthermore, our simulated trader begins with an initial capital of $10.000, and we make the assumption of no transaction fees.

[3] https://kernc.github.io/backtesting.py/

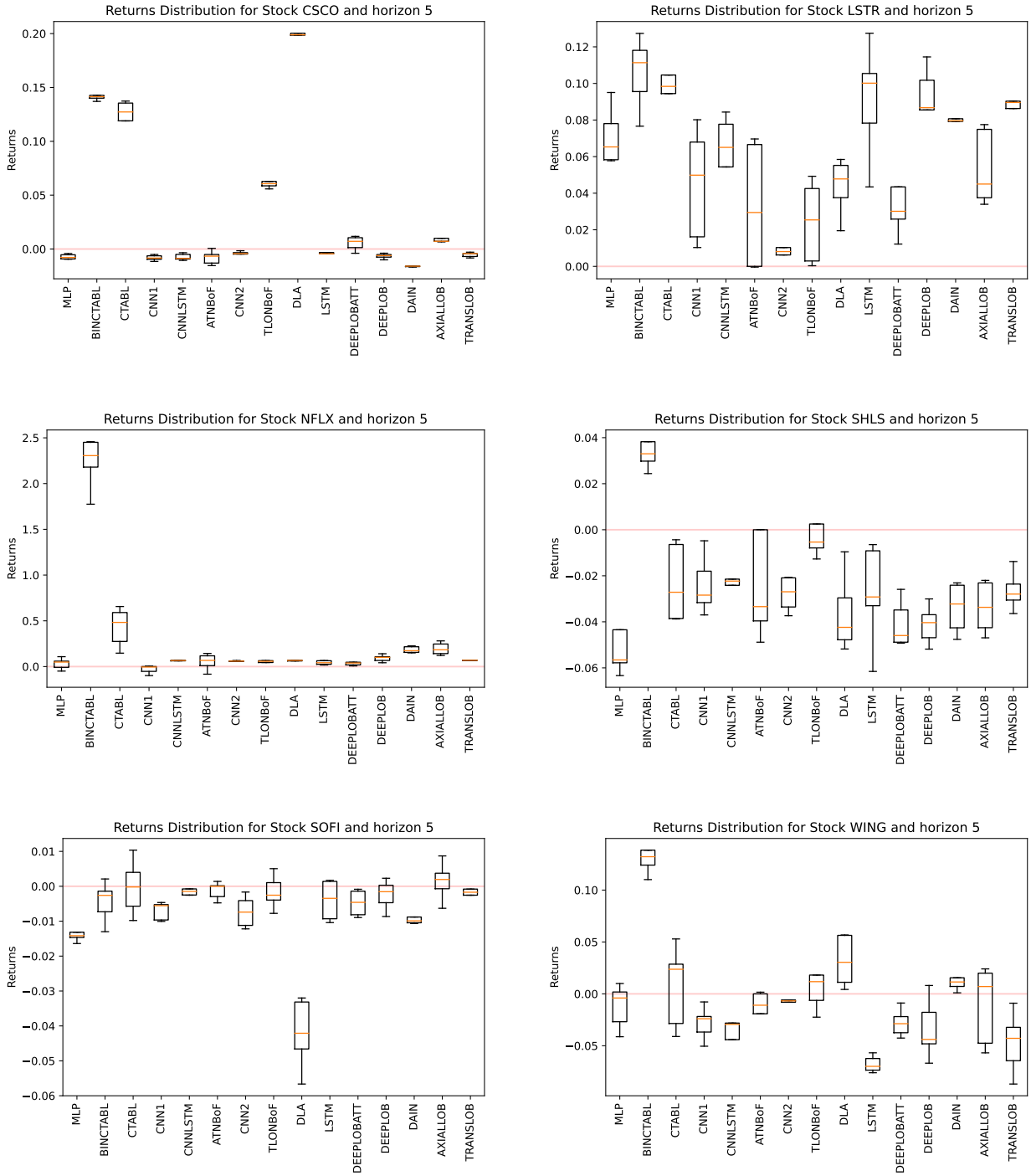


Figure 3.11: Distribution of returns on five seeds.

The *trading strategy* relies on the models and operates by generating signals every 10 events to predict subsequent price movements. These signals, categorized as *up*, *stationary*, or *down*, determine the trading action. When the signal is *up*, the simulated trader places a buy order. Conversely, if the signal is *down* and the trader currently holds a long position, he places a sell order. In cases where the signal is *stationary*, the trader takes no action. The orders are filled at the next open price.

The results of the trading simulation for each stock are presented in Figure 3.11. The strongest correlation observed is between the daily returns of the stocks, as shown in Table 3.3, and the returns of the strategy described above. In fact, the two stocks with the highest positive daily returns (namely LSTR and NFLX) are the only ones for which the strategy is profitable. On the other hand, the two stocks with the highest negative daily returns (SOFI and SHLS) are the ones for which most models show a negative return. Another correlation, albeit less strong, is between the volatility of the stocks and the return of the models. Specifically, lower volatility is associated with higher model returns.

We recognize the limitations of this simulation. For instance, we do not perform portfolio optimization or position sizing, we assume the trades execution at the mid-price, and we ignore transaction costs, but a realistic and sophisticated algorithmic trading simulation is beyond the scope of this study and remains an interesting aspect for future research.

3.7 Discussion and Conclusions

Our findings highlight that price trend predictors based on DNN using LOB data are not consistently reliable, as they often exhibit non-robust and non-generalizable performance. Our experiments demonstrate that the existing models are very susceptible to hyperparameter selection, randomization, and experimental context (stocks, volatility, market, historical period). In addition, the experimental setup fails to capture the intricacies of the real-world scenario. This lack of robustness and generalizability makes them inadequate for practical applications in real-world settings.

Models: Our results lead to a crucial observation: on the LOBSTER dataset, SOTA DL models for LOB data exhibit low generalizability. We suggest that this phenomenon is due to two factors: the higher complexity of the LOBSTER dataset compared to the FI-2010 dataset and the overfitting of the best-performing models to the FI-2010 dataset, which lowers their performance on the LOBSTER dataset. In fact, in the original papers, all the considered models (except for DEEPLOB, DEEPLOBAT and DLA) were trained, validated, and tested only on FI-2010, which is a smaller dataset with less frequent and voluminous orders than those contained in LOBSTER-derived datasets. Our conjecture is supported by insightful findings reported in the existing literature: several works (e.g., [111–113]) have shown that while some datasets exhibit simple patterns that can be effectively captured by a shallow model, others may require deeper architectures to model complex relationships.

Another key finding of this study is that the top models with the highest performance on both datasets employ attention mechanisms. This suggests that the attention technique enhances the extraction of informative features and the discovery of patterns in LOB data. However, in general, it appears that current models cannot cope with the complexity of financial forecasting with LOB

data.

Dataset: Financial trends can be influenced by both local and international political events; in fact, political actions and decisions can significantly impact economic conditions, market sentiment, and investor confidence [49]. These factors are not captured by LOB data alone. For this reason, we believe that price predictors may benefit from integrating LOB data with additional information, for example, sentiment analysis relying on social media and press data, representing an easily accessible source of exogenous factors impacting the market [114]. This is particularly true for mid- and long-term price trend prediction, whereas it might not hold for HFT strategies [50]. We remark that micro and macroscopic market trends are fundamentally different, and the microscopic behavior of the market is very much driven by HFT algorithms, making it almost exclusively dependent on financial movements rather than external factors. In this scenario, granular and raw LOB data may suffice to provide data for price trend prediction. Several works have used sentiment analysis for price trend prediction. The work in [115] combines comments made by investors on the online platform StockTwits with Apple stock price time series and uses LSTM for stock closing price prediction. By creating a dataset composed of tweets and historical stock prices, the work in [116] proposes a new architecture for stock price prediction. Similar approaches are used in [66, 67], where the authors show that the performance of their predictors improves when their dataset is enriched with sentiment data. Despite the wide research in this field, to the best of our knowledge, no one has ever used LOB data together with sentiment analysis for price trend prediction. The existing literature suggests that this could be a valuable research direction.

Another weakness in dataset generation is the potential for training, validation, and test splits to have dissimilar distributions. This occurs due to the distinct characteristics of the historical periods covered by the stock time series. This can negatively affect the model's ability to generalize effectively and make reliable predictions on unseen data. A last limitation regarding the dataset is the representation of the limit order book, which has been shown to be sensitive to permutations by Wu et al. [117]. In [118], the same authors proposed robust alternative representations.

Labeling: As we discussed in Equation (3.2) and sections 3.5 and 3.6.3, the choice of the threshold for class definition in Equation (3.2) plays a crucial role in determining the trend associated with a market observation. We believe that current solutions present room for improvement. As discussed in Section 3.5, in FI-2010, the parameter θ was chosen to obtain a balanced dataset in the number of classes for the horizon $k = 5$ (which is the mean value of the considered interval in the set $\mathcal{K}$). Thus, θ is not chosen in accordance with its financial implications but rather serves the purpose of improving model performance. We recall that the dataset is made up of different stocks. With such a labeling system, with a fixed θ, stocks with low volatility become associated with stable trends, as their behavior is overshadowed by stocks exhibiting higher volatility. Good practices that could be investigated are to use a weighted look-behind moving average to absorb data noise instead of mid-prices as in Equation (3.2) or to define a dynamically adapting θ which accounts for changing trends of a stock's mid-price. Moreover, the labeling approach of Equation (3.2), used by all surveyed models, fails to leverage important aspects available in LOB data, so another possible improvement is the definition and use of other insightful features that can be extrapolated from the LOB in addition to the mid-price. Such values could encapsulate other peculiar and informative features, such as stocks' spreads and volumes, which directly influence stock volatility and so the returns.

Profit: In the context of stock prediction tasks, it is of utmost importance to go beyond standard statistical performance metrics such as accuracy and F1-Score and incorporate trading simulations to assess the practical value of algorithms. SPTP predictors' ultimate measure of success lies in their ability to generate profits under real market conditions. It is essential to conduct trading simulations using real simulators that go beyond testing on historical data. Recent progress has been made in the context of reactive simulators [119–122].

Limitations and risks: We acknowledge that our study is subject to some limitations, which should be considered when interpreting our findings. First, we conducted a grid hyperparameter search for the models that did not specify them. Since hyperparameter search is not exhaustive, our chosen best hyperparameters could potentially undermine the quality of the original systems. Secondly, due to computational resource limitations, we could not train the benchmarked models on LOB datasets spanning longer periods, e.g., years rather than weeks. We recognize that doing so could have led to different results.

We also highlight that using DL or, more generally, AI models for solving SPTP and exploiting them for trading can have a number of risks. Some of them are inherently technical. This is the case for data biases, that is, incomplete or unrepresentative data, which can cause predictive algorithms to favor groups that are better represented in the training data [123]. Lack of explainability is another risk that could expose organizations to vulnerabilities (such as biased data, unsuitable modeling techniques, or incorrect decision-making) and potentially undermine the trust in their robustness [124]. AI models for trading are also vulnerable to cyber-attacks. Malicious users can exploit AI model vulnerabilities to evade detection and prompt the models to make the wrong decisions or to extract information by manipulating data at some stage of the model lifecycle [125]. Other risks have an ethical nature and can impact financial stability. One of them is inequality among investors. As training and predicting are expensive in terms of hardware equipment and energy, and because of the challenges in model interpretation and prediction, AI trading can lead to a concentration of information among those who can afford the required technology, exacerbating income inequality and asymmetry in the market, with an uncertain impact on financial stability [123, 126]. Because of limited regulation of the AI trading systems, there is a lack of transparency, making it difficult to detect possible unfair strategies [126]. Governing legal and regulatory framework regulators should welcome the advancements of AI in finance and undertake the necessary preparations to harness its potential advantages and address its associated risks.

3.8 Appendix

3.8.1 Models Description

Tsantekis et al. [86] (2017) use an LSTM to predict price directions considering moving averages of the mid-price over the past and the future k steps. In the same year, the same authors proposed in [94] a model based on a CNN (CNN1) for future mid-price movement predictions from large-scale high-frequency limit order data. The proposed architecture is composed of a series of convolutional and pooling layers followed by a set of fully connected layers that are used to classify the input. The parameters of the model are learned by minimizing the categorical cross-entropy. In [87] (2020), the same research group proposed two new architectures. The first one (CNN2) uses a series of convolutional layers for capturing the temporal dynamics of time series extracted from an LOB

and for correlating temporally distant features. In the last convolutional layer, CNN2 retains the temporal ordering by flattening only the dimensions of the convolution. The authors then propose an architecture that merges the described CNN with an LSTM, which we call CNNLSTM. Initially, the CNN is used for feature extraction for the LOB time series. It produces a new time series of features with the same length as the original one, which is then passed to the LSTM module for classification.

Tran et al. [95] in 2018 introduced a new NN architecture for multivariate time series that incorporates an attention mechanism in the temporal mode. The authors call this architecture Temporal Attention-Augmented Bilinear (TABL), as it applies a bilinear transformation to the input, which consists of a set of samples at different time stamps. The Bilinear Layer (BL) is able to detect feature and time dependencies within the same sample and is augmented with a temporal attention mechanism to capture interactions between different time instances. The authors define three different network configurations, called A(TABL), B(TABL), and C(TABL), with 0, 1, and 2 hidden layers, respectively. In our experiments, we consider C(TABL), which outperforms the others. In [15] (2021), the same authors extended the solutions implemented in [95] by integrating a data-driven normalization strategy that takes into account statistics from both temporal and feature dimensions to tackle potential problems posed by non-stationarity and multimodalities of the input series. The new model is called BINCTABL.

Passalis (2019) et al. [96] introduce the DAIN (Deep Adaptive Input Normalization) three-step layer that adaptively normalizes data depending on the task at hand, instead of using some fixed statistics calculated beforehand as in traditional normalization approaches. DAIN works as follows: in the first layer, called the adaptive shifting layer, the mean of the current time series is scaled by the weight matrix of the first neural layer. The resulting vector is passed to the adaptive scaling layer, which first computes the standard deviation of the original feature vector with respect to the shifted one and then scales this result using the weight matrix of the scaling layer. The last layer, called *adaptive gating layer*, is meant to suppress features that are not relevant by applying a sigmoid function in order to neglect features with excessive variance, which could hinder network generalization. The authors integrate DAIN in three different architectures, an MLP proposed in [127], a CNN as in [94], and a Recurrent Neural Network (RNN) [128]. In our experiments, we consider the architecture with the highest performance, namely the MLP.

Zhang et al. [14] (2019) propose DEEPLOB. The authors propose a smooth data labeling approach based on mid-prices to limit noise and discard small oscillations. They propose a 3-block architecture composed of standard convolutional layers, an Inception Module, and an LSTM layer. The first two elements are used for feature extraction, whereas the LSTM layer captures time dependencies among the extracted features.

Wallbridge et al. [97] (2020) introduce TransLOB, a new DL architecture for mid-price movement prediction, composed of two main components: a convolutional module made up of five dilated causal convolutional layers and a transformer module, composed by two transformer encoder layers, each made up of a combination of multi-head self-attention, residual connections, normalization, and feedforward layers. Between the convolutions and the transformer module, the tensor is passed to a normalization layer and concatenated to a positional encoding.

Passalis et al. [98] (2020) propose a model for high-frequency limit order book data based on Temporal Logistic Neural Bag-of-Features formulation (TLoNBoF). Given a collection of time series,

TLoNBoF extracts features with a 1-D convolutional layer to capture the temporal relationships between succeeding feature vectors. Then the features are transformed into vectors of constant length, i.e., their length must be invariant to the length of the input time series. To cope with this, the authors define a Temporal Logistic Neural Bag-of-Features formulation to aggregate the extracted feature vectors. A fine-grained temporal segmentation scheme is also proposed to capture the temporal dynamics of the time series. To this end, the transformed feature vectors are segmented into three temporal regions to capture the short-term, mid-term, and long-term behaviour of the time series.

In 2021, Zhang et al.[129] adopt Sequence-to-Sequence (Seq2Seq) [128, 130] and Attention [131] to recursively generate multi-horizon forecasts and build a predictor called DEEPLOBATT. A typical Seq2Seq model consists of an encoder that analyses the input time steps to extract meaningful features. Then, only the last hidden state from the encoder is used to make estimations, which penalizes the processing of long sequence input. To overcome this limitation, the Attention module accesses hidden states of the encoder and assigns a proper weight to each hidden state. Each input contains the most recent 50 updates, and each update includes information for both the ask and bid of an LOB. Therefore, a single input has the dimension (50, 40), and each output consists of a multi-horizon prediction of all 5 points of the FI-2010 dataset. As an encoder, they adapt a previous model, namely DeepLob [14], to extract representative features from raw LOB data while they experiment with both Seq2Seq and Attention models for the decoder.

Guo et al. [100] (2022) propose a novel architecture for price trend prediction named Deep Learning Architecture (DLA). Firstly, the dataset is preprocessed and aggregated at different time windows. Once extracted, the features are given as input to the three-phase proposed architecture. The first phase uses Temporal Attention to adaptively assign attention weights to each moment of the sliding window. The processed data is passed to a stacked Gated Recurrent Unit (GRU) architecture to obtain an accurate representation of the analysed trends, which is complex and nonlinear. The GRU architecture consists of two hidden GRU layers to generate as output the hidden state at each time period. This is given to the second temporal attention stage, which is used to generate more accurate attention weights. The proposed solution is compared to several other models in the literature, including C(TABL) [95], DeepLOB [14], and TLo-NBoF [98]. The proposed solution achieves very high performance on the FI-2010 dataset and outperforms the other models. The authors analyse the performance of their model by varying several parameters, including label thresholds and the choice of the time step.

Tran et al. [101] extend the solution proposed in [132], which introduces a neural bag-of-features (N-BoF)-based method for building a *codeword* that is eventually fed to a classifier. In [101], the neural bag-of-feature model was enhanced by incorporating a 2D-Attention (2DA) module that highlights important elements in the matrix data while discarding irrelevant ones by zeroing them out. The 2D-Attention function performs a linear interpolation between the input data matrix and the input data matrix filtered by an attention mask matrix that encodes the importance of the columns of the original input. The proposed 2DA block can be applied to the features to highlight or discard the outputs of certain quantization neurons, whose results are considered equally important in the NBoF model for every input sequence (Codeword Attention). The resulting model is called ATNBoF. The 2DA function can also be applied to lend weight to salient temporal information, which is otherwise aggregated and equally contributes to the quantized features in the NBoF model

(Temporal Attention).

Kiesel et al. [102] (2022) propose Axial-LOB, a model based on axial attention for price trend prediction. Unlike the naive attention mechanism, axial attention factorizes 2D attention into two 1D attention modules, one along the width (feature) axis and a second one along the height (time) dimension. Raw values of the LOB are preprocessed and passed to the axial attention block: Each layer of the attention block is preceded and followed by a module composed of 1×1 convolutions, batch normalization, and ReLu activation to adjust the number of channels in the intermediate layers of the network. For training the axial attention module, the authors use mini-batch Stochastic Gradient Descent (SGD) by minimizing the cross-entropy loss between the predicted class probabilities and the ground truth label. The authors compare the performance of the proposed model against the solutions adopted in [14, 94, 95] in terms of precision, recall, and F1-Score on the FI-2010 dataset. Axial-LOB proves to have improved performance with respect to these works while being simpler in terms of the number of parameters.

3.8.2 Additional Experimental Results

Robustness

Figure 3.12 shows a bar chart representing the F1-Score of the 17 models reproduced using the LOBCAST framework for the five prediction horizons $\mathcal{K}$. The plot shows black empty bars representing the declared performance in the corresponding paper, when applicable. In the figure, the number on the bar represents the obtained performance in LOBCAST on the FI-2010 dataset, and the value in brackets indicates the difference between the obtained performance and the originally declared performance in the respective paper. We highlight that not all papers declare their performance for all the horizons. The figure clearly highlights how robust the considered models are. Surprisingly, for CNN2 and CNNLSTM, our experiments achieved noticeably higher performance than the one declared in the original paper.

The largest discrepancy is observed for TRANSLOB and ATNBoF (or TNBoF-TA), whose average performances differ by 28% from the original results. On average, ATNBoF achieves an F1-Score of only 40.9%. This substantial deviation from the claimed performance highlights the challenges and limitations associated with this particular model.

Figure 3.13 shows the agreement matrix of the models for the horizon $k = 5$. As expected, the highest agreement ($\approx$80%) is among the best-performing models, namely BINCTABL, AXIALLOB, DEEPLOB, CTABL, DLA, and DEEPLOBATT. The model that exhibits the least correlation with the other models is MLP.

The best-performing model in our benchmark is BINCTABL, reaching 92.1% of F1-Score on time horizon $k = 10$. Moreover, it notably closely aligns with the performance reported in the paper presenting it. Specifically, BINCTABL introduces an Adaptive Bilinear Normalization layer to CTABL, enabling joint normalization of the input time series along both temporal and feature dimensions. This enhancement yields a remarkable improvement, with an average increase of 9.2% in the F1-Score compared to the second-best model (DLA). Interestingly, BINCTABL is composed only of 11.446 parameters, which makes it very fast at inference time (0.0005s).

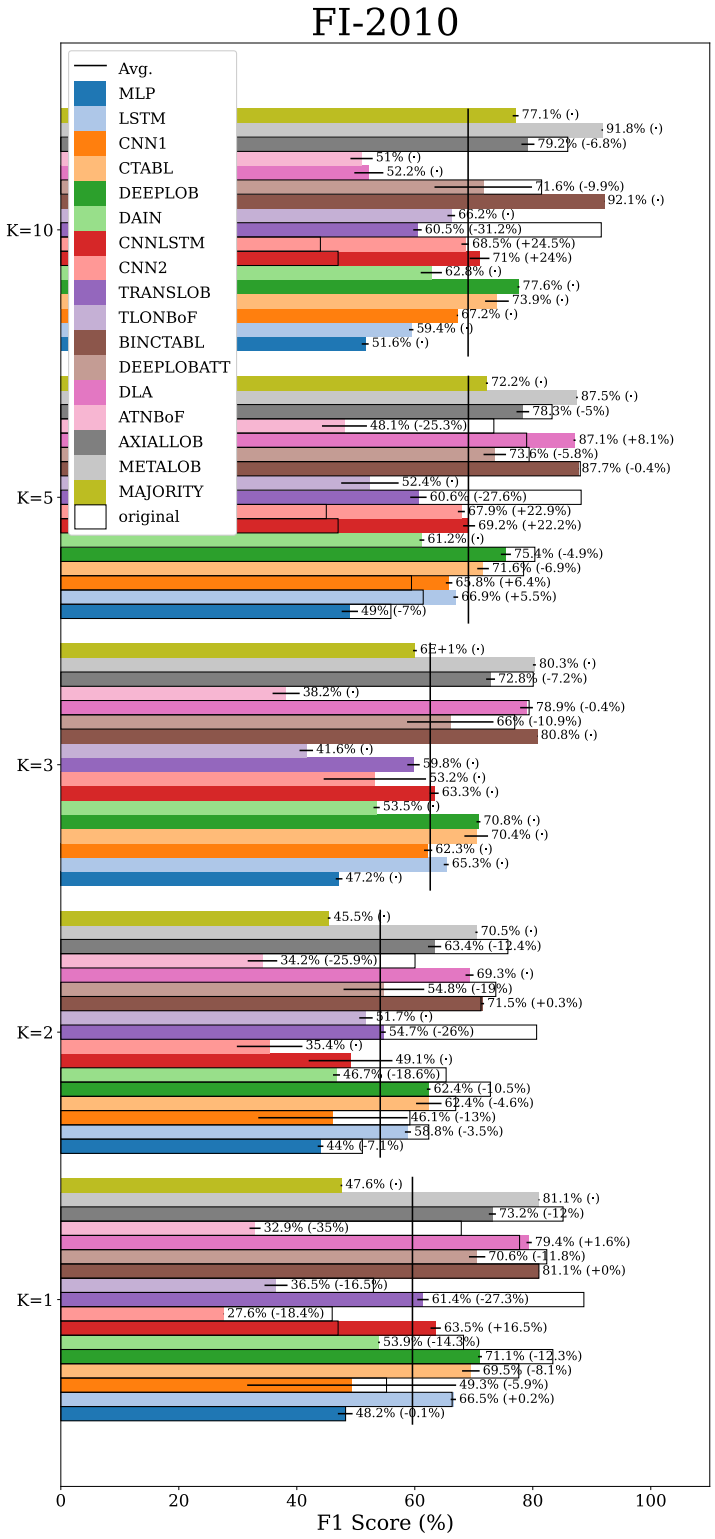


Figure 3.12: F1-Score on FI-2010.

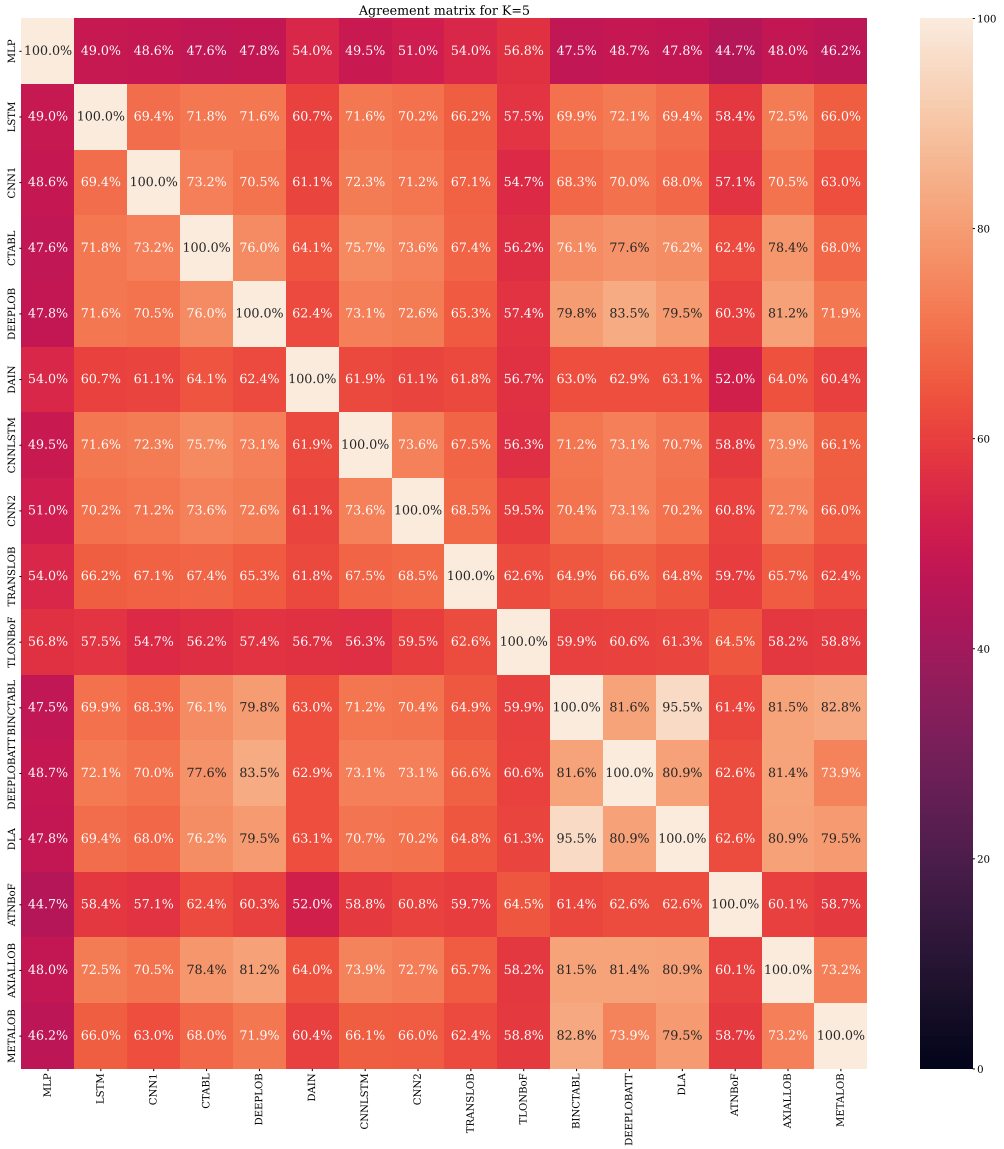


Figure 3.13: Agreement matrix FI-2010 in the horizon $k = 5$.

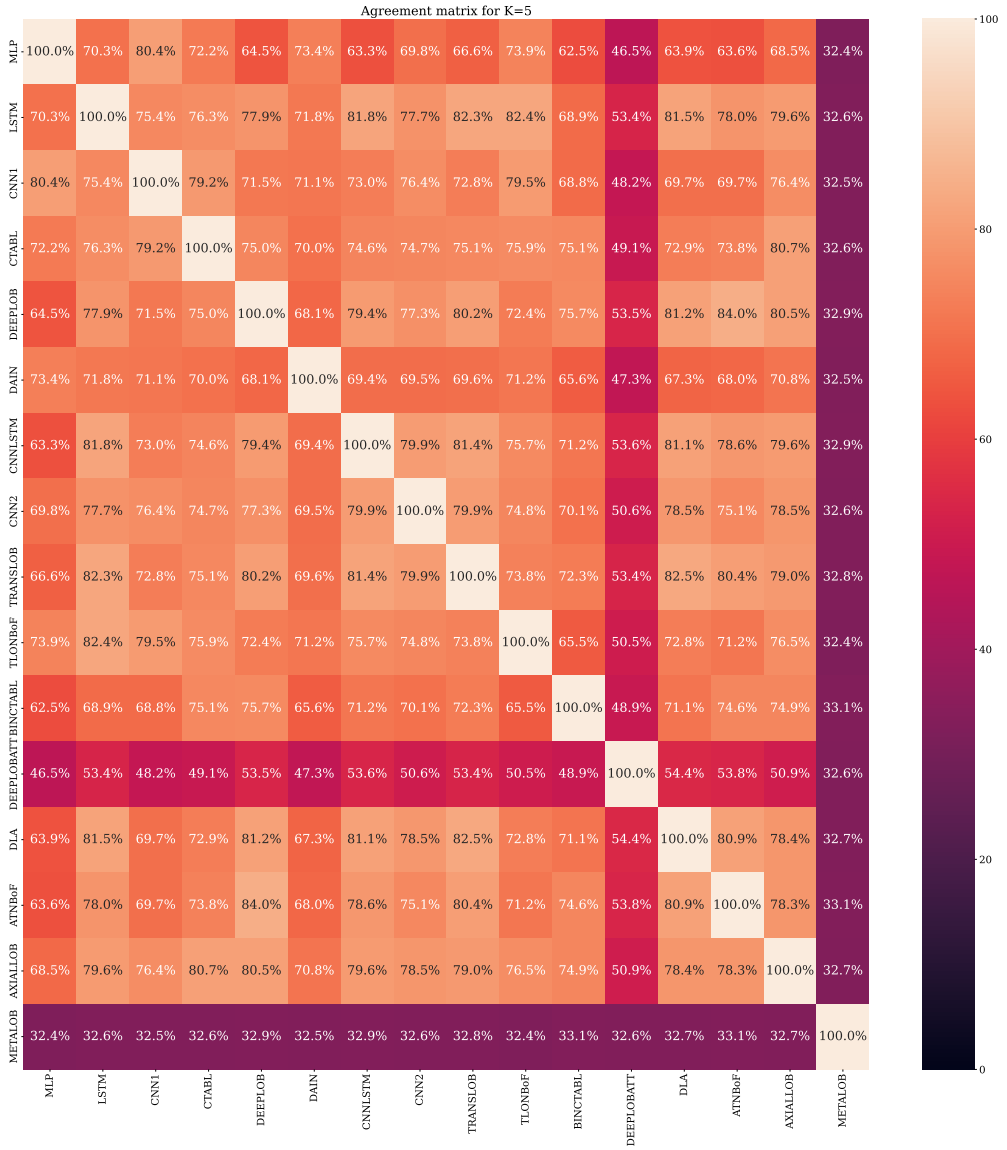


Figure 3.14: Agreement matrix on LOB-2022.

Generalizability

Figure 3.14 shows the agreement matrix on LOB-2022. Considering the more flattened performances of the models on the LOB-2021/2022 dataset compared to the FI-2010 dataset, the agreement percentages among the models are consistently high, and no distinct patterns are observed. Unlike FI-2010, where METALOB predicted as BINCTABL 82.8% of the time, on LOB-2022 (and also LOB-2021) METALOB showed no preference for any model, resulting in a balanced agreement rate ($\approx 33\%$) among all models. We decided not to include the agreement matrix of LOB-2021 because it was similar to LOB-2022.

Table 3.10: p-values of the T-test on FI-2010 for horizon $k = 5$.

	LSTM	CNN1	CTABL	DEEPLOB	DAIN	CNNLSTM	CNN2	TRANSLOB	TLONBoF	BINCTABL	DEEPLOBATT	DLA	ATNBoF	AXIALLOB	METALOB	MAJORITY
MLP	4e-06	2e-06	2e-08	1.3e-08	2.1e-05	3e-08	1e-06	2e-06	**.240675**	6e-07	9e-08	5e-07	**.671884**	2e-09	3.5e-06	4e-06
LSTM	-	.01194	.000253	3e-06	3.9e-08	.008082	.032604	.000457	.003819	4.6e-08	.001799	1e-09	.000536	5e-06	2e-09	1e-06
CNN1	-	-	3.8e-05	1.9e-06	2e-06	.000953	.000898	.000777	.005021	1.4e-07	.000751	1.8e-08	.000634	8.4e-07	3.2e-08	5e-06
CTABL	-	-	-	.000373	4e-06	.010175	.000479	3e-06	.001094	5e-06	**.113016**	4e-06	.000135	1.6e-05	4e-06	**.256306**
DEEPLOB	-	-	-	-	1.6e-07	1.6e-05	2e-06	1e-06	.000551	9e-06	**.125099**	6e-06	8.1e-05	.003406	6e-06	.001316
DAIN	-	-	-	-	-	1.8e-05	2.7e-07	**.452821**	.022074	2e-08	.00013	0.0	.002186	5.8e-07	1e-09	2.1e-08
CNNLSTM	-	-	-	-	-	-	**.062585**	1.6e-05	.001841	3e-06	.006636	2e-06	.000209	2e-06	2e-06	.003497
CNN2	-	-	-	-	-	-	-	.000131	.002906	2.8e-07	.002747	5.2e-08	.000397	2e-06	8.3e-08	5.1e-05
TRANSLOB	-	-	-	-	-	-	-	-	.025307	3e-06	8e-06	2e-06	.001566	8e-08	2e-06	6.2e-05
TLONBoF	-	-	-	-	-	-	-	-	-	.000131	.000388	.000139	**.202507**	.000296	.000133	.001225
BINCTABL	-	-	-	-	-	-	-	-	-	-	.000119	.00176	3.2e-05	6.2e-05	.037839	4e-09
DEEPLOBATT	-	-	-	-	-	-	-	-	-	-	-	.000131	2.4e-05	.004296	.000119	**.234532**
DLA	-	-	-	-	-	-	-	-	-	-	-	-	3.3e-05	6e-05	.007015	0.0
ATNBoF	-	-	-	-	-	-	-	-	-	-	-	-	-	3.9e-05	3.2e-05	.000219
AXIALLOB	-	-	-	-	-	-	-	-	-	-	-	-	-	-	5.5e-05	.000271
METALOB	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0.0

Table 3.11: p-values of the T-test on LOB-2021 for horizon $k = 5$.

	LSTM	CNN1	CTABL	DEEPLOB	DAIN	CNNLSTM	CNN2	TRANSLOB	TLONBoF	BINCTABL	DEEPLOBATT	DLA	ATNBoF	AXIALLOB	METALOB	MAJORITY
MLP	.018607	.021589	2.6e-05	2.6e-05	.038899	**.10573**	**.283349**	**.08208**	**.05829**	5e-06	6.4e-05	.010697	**.308099**	7.7e-05	.029677	.009123
LSTM	-	**.279997**	.010331	.00918	**.176818**	**.310756**	.00593	**.609023**	**.129668**	.00074	.003929	**.453778**	**.122387**	.008292	.002487	.002129
CNN1	-	-	5.7e-05	.000129	**.460271**	**.85318**	.010043	**.728193**	**.250598**	2.4e-07	5e-06	**.394171**	**.168575**	7e-06	3.9e-05	3.2e-05
CTABL	-	-	-	**.887803**	.000265	.001914	6.5e-05	.010641	5.7e-05	.000942	.039844	.000154	.046493	**.455996**	1e-06	8e-06
DEEPLOB	-	-	-	-	.00047	.001623	4.9e-05	.009341	.000143	.00241	**.076299**	.000354	.045168	**.612903**	3e-06	1.9e-05
DAIN	-	-	-	-	-	**.893312**	.01842	**.539226**	**.398116**	4e-06	4e-08	.046864	**.183409**	1e-06	.000187	7.9e-08
CNNLSTM	-	-	-	-	-	-	.028972	**.675565**	**.700262**	.000243	.001296	**.545926**	**.178718**	.002261	.007487	.004825
CNN2	-	-	-	-	-	-	-	.026103	.022311	2.3e-05	.000153	.007106	**.423282**	.000192	**.381685**	**.111617**
TRANSLOB	-	-	-	-	-	-	-	-	**.42584**	.00128	.004928	**.989729**	**.15165**	.009266	.011145	.007343
TLONBoF	-	-	-	-	-	-	-	-	-	1.2e-07	3.8e-08	.026764	**.196436**	1.2e-07	6.1e-05	1e-06
BINCTABL	-	-	-	-	-	-	-	-	-	-	.00391	2.5e-07	.022831	.000308	8e-09	7.8e-08
DEEPLOBATT	-	-	-	-	-	-	-	-	-	-	-	1.8e-07	.033816	.012705	1.5e-07	0.0
DLA	-	-	-	-	-	-	-	-	-	-	-	-	**.150606**	1e-06	1.2e-05	1e-06
ATNBoF	-	-	-	-	-	-	-	-	-	-	-	-	-	.042348	**.519783**	**.629823**
AXIALLOB	-	-	-	-	-	-	-	-	-	-	-	-	-	-	8.9e-08	2e-09
METALOB	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	**.091866**

Chapter 4

On Correlated Stock Market Time Series Generation

Abstract

In this chapter, we present `CoMeTS-GAN` (**Co**rrelated **M**ultivariat**e** **T**ime **S**eries **GAN**), a framework based on Conditional Generative Adversarial Networks (C-GANs), designed to generate mid-prices and volumes time series of correlated stocks. This tool provides a light and responsive solution for a realistic stock market simulation. It is able to accurately learn and reproduce inter-asset correlations, a crucial aspect for achieving realness in multi-stock simulation environments. Our experimental campaign assesses the model using acknowledged stylized facts of stock markets as well as additional metrics capturing inter-asset correlations. We compare our model to leading architectures, highlighting our approach's strengths. These findings suggest the potential of `CoMeTS-GAN` in realistically simulating correlated price movements, offering a responsive market environment and valuable input for trading strategy formulation.

This work has been published in the Proceedings of the Fourth ACM International Conference on AI in Finance [25].

4.1 Introduction

AI advancements have significantly impacted the finance industry, enabling forecasting and advanced techniques for market simulations. Financial markets create enormous volumes of data on a daily basis, which opens up new avenues for machine learning models to be applied and used to gain a better understanding of the market.

Though daily stock price data is accessible, more detailed data like fine-grained market features and Limit Order Book data are often restricted due to regulations and cost. Hence, synthetic data, which can be customized to specific needs, has become a valuable tool. Over recent years, a number of deep learning models have been designed to generate synthetic data for fundamental financial applications. Some notable examples include market simulations [133], model calibration [134], and portfolio construction [135, 136].

For these reasons, developing data production techniques capable of capturing the intricacies of global financial markets is crucial. We emphasize the relevance of well-acknowledged empirically

observed stylized facts of financial time series [2], characterizing stock returns, volatility, and other features. We note, however, that producing synthetic data that correctly captures the emerging inter-asset correlation dynamics is crucial for testing approaches involving portfolio management aiming at diversification [137].

Generative Adversarial Networks (GANs) [17] have gained momentum in recent years in many fields, including complex networks [138], medical time series [139], and DNA sequences [140]. As for the context of financial time series, several works have been published in the last few years, even though developments are evidently still at an early stage. For instance, current approaches consider only the intrinsic properties of the generated series, giving little or no attention to cross-series characteristics.

In the financial domain, correlation dynamics play an essential role in managing the risks within a portfolio by ensuring that the individual assets are not overly correlated with one another and are not impacted by similar market conditions. There are several reasons why stocks may exhibit correlation. Correlation may be sector-specific: for instance, the technology sector is affected by industry-wide regulations and changes in customer demands for that product. Macroeconomic factors like interest rates, inflation, etc., also impact entire sectors, causing the underlying stocks to react similarly. Furthermore, exogenous news and events can impact stocks belonging to different economic fields. We underline that even stocks that do not have fundamental similarities may show relevant correlation, possibly due to investor sentiment [141, 142]. During periods of market uncertainty or volatility, investors tend to adopt a risk-aware approach, affecting multiple stocks simultaneously.

Finally, another source of correlation comes from composite financial instruments such as ETFs (Exchange-Traded Funds). Since these involve a basket of stocks, correlation among otherwise unrelated assets could arise as they align with the funds' overall performance [143]. However, it is important to note that correlation does not imply causation. While two stocks may exhibit a high correlation, it doesn't necessarily mean one stock directly affects the other. Nonetheless, the existence of the described correlation dynamics cannot be neglected when designing AI tools for the prediction of realistically generated synthetic datasets for simulating the market.

This work proposes a novel framework called `CoMeTS-GAN` (**Co**rrelated **M**ultivariat**e** **T**ime **S**eries **GAN**) for the synthetic generation of realistic market data that explicitly aims at providing a responsive market environment whilst capturing the inherent correlation among assets. In detail, the major contributions of this work are the following:

- A C-GAN-based framework to allow the realistic and responsive generation of time series of correlated assets. The framework enables swift training thanks to a very light convolution-based architecture. The conditional nature of the model makes it suitable for autoregressive applications, which allows the generation of arbitrarily long time series regardless of the model's input and output size, and without the need to be retrained.
- A new metric called *cross-correlation distance* which quantifies the distance in the correlation between the generated and real time series.
- A comparison with existing approaches, which reveals that our method outperforms existing work in its capability to capture inter-asset correlation, producing longer time series, with moderate training effort.
- A thorough experimental campaign that points out the realism of the model, capturing all the

known stylized facts and the correlation dynamics. The campaign also evaluates the model's responsiveness to the behavior of experimental agents.

4.2 Related Work

In this work, we present a generative framework to output realistic time series. It is founded on a Conditional Generative Adversarial Network (C-GAN). Our main focus lies in generating realistic time series data for multiple financial assets simultaneously, such as prices and volume time series. It is crucial for our approach to maintain the correlation relationships among the assets and enable responsive market simulation.

Generative models aim at generating financial datasets and are divided into two categories: the generation of stock price datasets and the generation of order-based datasets. Concerning the first, Vuletić et al. [144] exploit a GAN architecture trained with an economic-driven loss function for the generator. They aim at generating the ETF-excess returns with respect to the underlying stocks to place trades and make profits. Cont et al. [145] proposed an approach based on GANs to generate price traces that retain tail risk features observed in the input dataset, to improve the estimation of loss distributions in dynamic portfolios. However, these approaches do not consider the simultaneous generation of multiple stocks and their correlation dynamics.

Order time series generators aim at producing synthetic Limit Order Book (LOB) datasets. A first attempt by Li et al. [146] proposes an approach to generate realistic and high-fidelity stock market order streams based on GANs. The proposed Stock-GAN model employs a Conditional Wasserstein GAN to capture the history dependence of orders. This study aims to generate synthetic data that does not include features that may link back to the originator of the order in the training dataset for anonymity reasons. Recently, Coletta et al. [133] extended this work by considering responsiveness and multi-agent simulations. However, these works do not consider multiple intercorrelated stocks, lacking realism when considering cross-correlated assets.

Among the first works in generating time series, there is C-RNN-GAN [147], where a GAN using LSTM networks for generator and discriminator is applied to generate classical music. Data is generated recurrently, taking as inputs a noise vector and the data generated from the previous time step. The RCGAN model proposed in [139] follows a similar approach to generate medical data. It introduces a novel architecture that drops the dependence on the previous output while conditioning on additional input. However, unlike our proposed technique, these approaches rely only on binary adversarial feedback for learning, which by itself may not be sufficient to guarantee that the network efficiently captures the temporal dynamics in the training data.

A significant improvement was made in [148], where the authors propose the time series GAN (TimeGAN), a novel framework composed of four network components (i.e., an embedding function, recovery function, sequence generator, and sequence discriminator), which combines the unsupervised adversarial loss on real and synthetic data with a step-wise supervised loss on the original data. TimeGAN significantly outperforms previous architectures in terms of several evaluation metrics, such as discriminative and predictive scores on benchmark datasets. However, while TimeGAN outputs mini-sequences of fixed size, our model is able to generate arbitrarily long time series thanks to its conditioning capabilities. Moreover, when applied to the financial sector, TimeGAN shows prohibitive training time in contrast to the time-efficient training required by our approach.

S. Takahashi et al. [149] propose FIN-GAN, an application of the classic GAN approach [17] to the finance domain with the goal of generating synthetic time series that replicate the stylized facts observed in real data. The authors propose three different structures for the generator and the discriminator: multi-layer perceptron, convolutional neural networks, and a combination of the two. M. Wiese et al. [19] propose QuantGAN, consisting of a GAN variation that utilizes temporal convolutional networks (TCNs) to capture long-range dependencies such as volatility clustering. Also, novel transformer-based architectures are employed in a GAN framework in [150].

Unlike our work, none of the proposals discussed above provides a multi-stock approach to generate arbitrarily long time series modeling emerging interactions among different stocks.

4.3 Background

In this section, we briefly review some important pillars that are useful for the reader to approach this chapter.

4.3.1 Conditional GANs

Generative Adversarial Networks (GANs) [17] are a subclass of generative models based on game theory. GANs have two neural networks: a generator denoted as G, and a discriminator denoted as D. The generator creates synthetic data samples from random noise, while the discriminator distinguishes between real and generated samples. During training, the generator competes against the discriminator to improve its ability to generate synthetic data samples $\boldsymbol{x} \sim p_{\text{gen}}$ that are realistic, i.e., distributed as the original data distribution p_{real}.

The generator's output is $\boldsymbol{x} = G(\boldsymbol{z}; \boldsymbol{\theta}^{(G)})$ where $\boldsymbol{z} \sim \mathcal{N}(0, 1)$ is an input noise to ensure variance at generation time, and $\boldsymbol{\theta}^{(G)}$ the parameters of the generator. The discriminator outputs $y = D(\boldsymbol{x}; \boldsymbol{\theta}^{(D)}) \in [0, 1]$ representing the probability of $\boldsymbol{x}$ being drawn from p_{real} and $\boldsymbol{\theta}^{(D)}$ the parameters of the discriminator. Both G and D are trained to optimize their payoff until neither player can unilaterally improve its cost. The discriminator is trained to maximize the probability of assigning the correct label to the data, either *real* or *generated*; conversely, the generator is trained to minimize it.

GANs can also be extended to consider conditional input, resulting in what is known as conditional-GANs [151]. In conditional-GANs, or CGANs, both the generator and discriminator networks may take additional conditioning information $\boldsymbol{c}$ as input, such as class labels or other contextual features. This allows for generating specific types of data samples based on the provided conditioning information. The generator's and discriminator's input vectors are combined with $\boldsymbol{c}$.

In this work, we use a Wasserstein Generative Adversarial Network (WGAN) [152], where the discriminator is even called critic and outputs a realness score of the input samples, $y = D(\boldsymbol{x}; \boldsymbol{\theta}^{(D)}) \in \mathbb{R}$, even called *critic score*. Formally, the optimal generator is G^*:

$$\arg\min_{G}\max_{D} \quad \mathop{\mathbb{E}}_{\boldsymbol{x}\sim p_{\text{data}}}[D(\boldsymbol{x}|\boldsymbol{c})] - \mathop{\mathbb{E}}_{\boldsymbol{z}\sim \boldsymbol{\mathcal{N}}(0,1)}[D(G(\boldsymbol{z}|\boldsymbol{c}))]. \tag{4.1}$$

4.3.2 Stylized Facts

The variation of asset prices exhibits several statistical properties that are common across a wide range of markets and time frames. These properties capture the market behavior over different time periods and are referred to as *stylized facts* [2, 153]. In this work, we use them to evaluate the quality of samples generated by the model. In particular, we consider the following stylized facts:

- **Fat-tailed distribution** or **heavy tails**. The distribution of the asset returns displays a power-law or Pareto-like tail, meaning that the asset returns follow a normal distribution with fat tails.
- **Aggregational normality**. As the time period for computing returns (Δt) increases, the distribution of returns resembles a normal distribution.
- **Linear unpredictability** or **absence of autocorrelation**. The linear autocorrelation of the return $\text{corr}(r_{t,T}, r_{t+\Delta t,T})$ rapidly decays to zero within minutes and becomes insignificant for periods longer than $\Delta t = 20$ minutes, with T the length of the series.
- **Volatility clustering**. Large changes in prices tend to cluster together; that is, volatility shows a positive autocorrelation over several trading days.
- **Volume/Volatility Correlation**. Trading volume and volatility are positively correlated.

Specifically, these properties allow us to assess the preservation of the realism of generated traces. Interdependency and correlation dynamics need new analytical tools to analyze the set of stocks in relation to each other.

4.3.3 Correlation Dynamics

In portfolio management, the use of correlation as a measure to evaluate portfolio diversification is a common practice. The principal aim is to mitigate the total risk by strategically including assets that exhibit a low correlation with each other. To achieve this, it is imperative to maintain a realistic understanding of the correlation dynamics among stocks.

In order to evaluate the effectiveness of the model in capturing these dynamics, we suggest a quantitative metric, the *cross-correlation distance* described in Section 4.5.2. We will utilize this metric to scrutinize the model's performance with respect to stocks that show varying degrees of correlation. For instance, let us consider two prominent companies, Coca-Cola and PepsiCo, which generally demonstrate a strong positive correlation. This correlation is attributable to their simultaneous presence in the same market, competing for similar consumer demographics, and facing exposure to similar exogenous factors. On the other hand, during times of market turmoil, investors often seek out defensive stocks as opposed to riskier, growth-oriented stocks. This pattern of trading behavior may result in a shift to Coca-Cola, a beverage firm, from a technology company, for instance, Nvidia, leading to a negative correlation between these two stocks.

Understanding these correlation patterns is not just academically interesting; it is critical for managing risk and unlocking profit opportunities.

4.4 CoMeTS-GAN System Model

The purpose of our framework is to generate multivariate time series capturing crucial aspects of the stock markets for multiple stocks simultaneously, which include mid-prices and volumes. A key

focus of our approach is to ensure the preservation of the correlation between the considered stocks. The framework's underlying model is trained to maximize both the similarity and the match in the correlation of the generated time series to real ones. This architecture can be used to generate time series also in other domains. Nonetheless, we will mostly stress its strengths in the financial field; more specifically, we are interested in mid-prices and volumes of a chosen set of stocks. In the following, we analyze in detail the main components of the framework.

Output. The output of our framework is a multivariate time series of length F units of time, one series for each of the stocks in the set of the considered set of stocks $\mathcal{S}$. This output is structured as a matrix $\hat{\boldsymbol{x}}_{\text{future}} \in \mathbb{R}^{F \times n}$, with $n = |\mathcal{S}|$ (i.e., the number of stocks). To generate longer time series, an auto-regressive approach can be used, that is, iteratively feeding the conditional generator with its own output. More details on the auto-regressive generation are provided in the following paragraphs.

Training Data. We will elaborate on the dataset generation process specifically for the mid-price time series. However, the methodology is analogous for volumes as well. The dataset used to train the model is generated from historical stock traces in the temporal period $[0, T]$. Let $x_s \in \mathbb{R}^T$ be the time series representing the mid-price of stock $s \in \mathcal{S}$ in the considered temporal period. For the given set of stocks, we stack the individual time series x_s along the columns to create the multivariate time series $\mathbf{X}_{\mathcal{S}} \in \mathbb{R}^{T \times n}$. In order to capture the dependencies between past and future stock movements, we segment the multivariate time series $\mathbf{X}_{\mathcal{S}}$ over time to construct the dataset as follows. For each time step t, we create a pair consisting of the past P mid-prices, denoted as $\boldsymbol{x}_{\text{past}}$ and the future F mid-prices denoted as $\boldsymbol{x}_{\text{future}}$. By applying this segmentation for $t \in [P, T - F]$, we generate the dataset which we denote as:

$$\mathcal{D} = \{(\mathbf{X}_{\mathcal{S}}[t - P : t],\ \mathbf{X}_{\mathcal{S}}[t + 1 : t + 1 + F])\}_{t=P}^{T-F}$$

The objective of the model is to generate a continuation $\hat{\boldsymbol{x}}_{\text{future}}$ of an observed time series $\boldsymbol{x}_{\text{past}}$, by training the model on tuples of the form $(\boldsymbol{x}_{\text{past}}, \boldsymbol{x}_{\text{future}}) \in \mathcal{D}$. The simultaneous generation of n time series allows for preserving the correlation dynamics observed in the training set.

CoMeTS-GAN Architecture. The underlying model of our framework is a Conditional Wasserstein Generative Adversarial Network (C-WGAN) [151, 152]. This model is trained to learn the joint probability distribution of the collection of future time series $\boldsymbol{x}_{\text{future}}$ in the training dataset $\mathcal{D}$ by optimizing the critic score. We adopt spectral normalization to enforce the Lipschitz continuity constraint, due to its superiority to other techniques like weight clipping or gradient penalty, and for its ease of applicability. As for the generator and the critic of the C-WGAN, we used a mixture of vanilla and temporal convolutional neural networks. Figure 4.1 shows the architecture of our model.

The ***generator*** G is the main component of `CoMeTS-GAN`. It is responsible for the actual generation of the time series $\hat{\boldsymbol{x}}_{\text{future}}$, which retains the statistical properties of the real-time series observed by the generator. The architecture used for the generator network is composed of a series of 7 temporal convolutional blocks [154] with increasing dilation size and a final linear layer to adjust the size of the output vector. Each temporal block is followed by a `leaky ReLU` activation function and a `dropout` layer.

The ***critic*** D is responsible for assessing the realness of the generated samples. The architecture used for the critic network is made of a series of convolutional layers having increasing filter sizes,

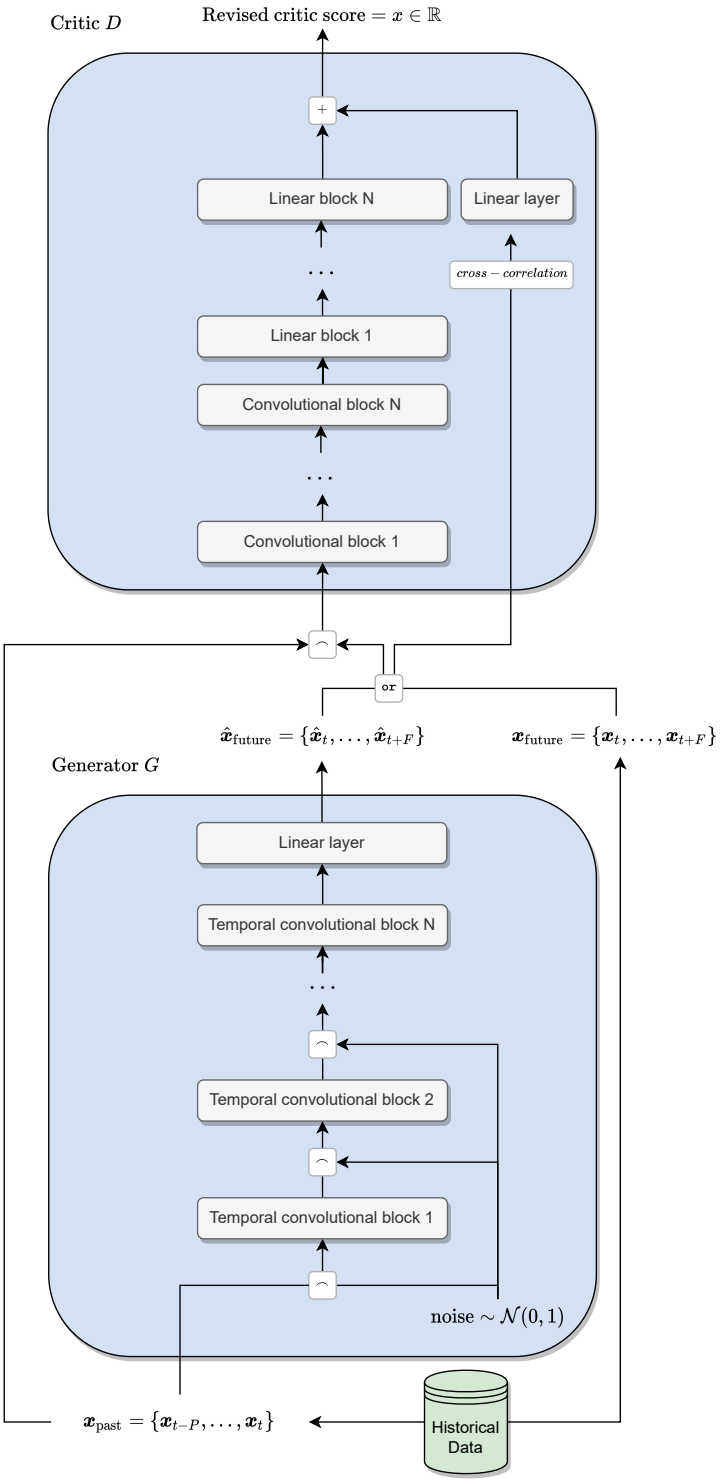


Figure 4.1: C-WGAN architecture. The $\frown$ operator represents the concatenation of two vectors. The `or` operator iteratively alternates between the real series $\boldsymbol{x}_{\text{future}}$ and the generated series $\hat{\boldsymbol{x}}_{\text{future}}$.

followed by a series of linear layers. Each convolutional and linear layer is followed by a `leaky ReLU` activation and a `dropout` layer, and spectral normalization is applied on top of each network layer. No activation function is used for the last layer of the network. We denote the output of this sequential network as o_1, which is devoted to measuring the realness of the generated time series. A linear layer is present within the critic (top right of the figure) and is meant to address the correlations of the multivariate time series. The linear layer takes as input the correlation coefficients (even referred to as *cross-correlation* values) computed between all pairs of the n stocks (i.e., $\binom{n}{2}$ pairs) and outputs o_2.

The overall output of the critic (i.e., the critic score) is the sum $o = o_1 + \alpha \cdot o_2$, with $\alpha \in [0, 1]$ being a hyperparameter to weigh the relevance of the correlation. By summing these two scores, we aim to capture the realism and correlation of the n stocks.

CoMeTS-GAN Training. Upon input pair $(\boldsymbol{x}_{\text{past}}, \boldsymbol{x}_{\text{future}}) \in \mathcal{D}$, the framework is trained as follows. The input of the generator is the concatenation of the past sequence $\boldsymbol{x}_{\text{past}}$ and a noise vector $z \sim \mathcal{N}(0, 1)$ of the same length. Each temporal block is fed with the output of the previous, concatenated with the noise vector z along the features dimension. The noise injection technique allows diversity in the model's output. Finally, the output of the last temporal block is directly fed to the last linear layer. The output of the generator network is the future sequence $\hat{\boldsymbol{x}}_{\text{future}}$.

The input of the critic is the concatenation of the past sequence $\boldsymbol{x}_{\text{past}}$ with either the real continuation $\boldsymbol{x}_{\text{future}}$ drawn from the dataset, or a generated one $\hat{\boldsymbol{x}}_{\text{future}}$ from the generator. The result of the concatenation is fed to the convolution and linear blocks to produce a single value in output, representing the standard critic score o_1. Additionally, the correlations between the features of the future sequence in the input are computed. All the $\binom{n}{2}$ correlation coefficients are input to the linear layer to produce the score o_2. The output of the critic is the sum of the two contributions, as mentioned above.

4.5 Experiments

In this section, we will conduct an evaluation of the performance of `CoMeTS-GAN` by comparing it against SOTA approaches. We will employ benchmark and newly curated datasets to ensure a rigorous assessment. Our primary focus will be on assessing the fidelity of the generated time series in comparison to real ones. We ran our Python-based framework on an NVIDIA V100 GPU. The source code is publicly available on GitHub[1], allowing interested parties to use it for their own experiments.

4.5.1 Datasets

We test the performance of our framework across time-series data showing varying periodicity, discreteness, level of noise, regularity of time steps, and correlation across time and features. We also include real stock market time series. It follows the list of time series data that we used to build the dataset $\mathcal{D}$, two of the three dataset families are benchmark datasets from the literature:

- *Sines* [148]: Multivariate sinusoidal sequences with different frequencies η and phases θ. Specifically, the dataset is obtained with $s_i(t) = \sin(2\pi\eta_i t + \theta_i)$, where $i \in \{1, \ldots, 5\}$, $\eta_i \sim$

[1] `CoMeTS-GAN` GitHub repository `https://github.com/giuseppemasi99/COMETS-GAN`.

Table 4.1: Pearson correlation between the stock prices in the validation set of the multi-stock dataset.

	KO	PEP	NVDA	KSU
KO	1.0	0.94	−0.66	−0.52
PEP	0.94	1.0	−0.81	−0.56
NVDA	−0.66	−0.81	1.0	0.65
KSU	−0.52	−0.56	0.65	1.0

$\mathcal{U}[0,1]$ and $\theta_i \sim \mathcal{U}[-\pi, \pi]$.

- *Multivariate Gaussian Model* [148]: Sequences from autoregressive multivariate Gaussian models, defined as: $g_i(t) = \phi_i g_i(t-1) + q$ where $q \sim \mathcal{N}(0, \sigma_i + (1-\sigma_i))$, $\phi_i \in [0,1]$, and $\sigma_i \in [-1,1]$. The coefficients ϕ_i and σ_i allow us to control the correlation across time and features, respectively.
- *Stock Mid-Prices & Volumes* [32]: The source of stock market data is LOBSTER[2], an online limit order book data tool to provide limit order book data for the NASDAQ-traded stocks. From limit order book data, it is easy to compute the mid-price time series with the temporal resolution of choice, which, in our case, is a minute. We focus on four stocks, namely, Coca-Cola (KO), PepsiCo (PEP), Nvidia (NVDA), and Kansas City Southern (KSU). The time period of choice is from 2018-02-02 to 2018-10-11 for the training set, and from 2018-10-12 to 2018-11-14 for the validation set. Table 4.1 shows the Pearson correlation coefficient between them. We observe how some are highly positively correlated (KO and PEP, NVDA and KSU), and some are highly negatively correlated (PEP and NVDA, PEP and KSU, KO and KSU, KO and NVDA). To test the scalability of our generative framework, we selected the 30 stocks composing the DJIA index from 2017-02-02 to 2017-10-11 for the training set and from 2017-10-12 to 2017-11-14 for the validation set. For these time series, we follow the same preprocessing pipeline. For the mid-prices, we compute the log-returns normalized using a z-score approach. Instead, volumes are scaled in the $[-1,1]$ with a `min-max` approach, then followed by `tanh` activation function, which scales the volumes to the positive domain.

4.5.2 Performance Evaluation

We measure the performance of our framework in terms of the following aspects: *Discriminative Score*, *Diversity*, *Stylized Facts*, *Reactivity*, *Scalability*.

Discriminative Score. This score offers a quantitative measure of the similarity between *real* and *synthetic* (or *generated*) time series. It was proposed in [148] and is computed as follows. An LSTM is trained to differentiate between sequences from the original and generated datasets. A standard supervised learning task is performed, and the resulting classification error on the test set (the lower the better) provides a quantitative measure of similarity.

We compare our framework with TimeGAN [148], RCGAN [139] and C-RNN-GAN [147]. For purely autoregressive approaches, we compare against RNNs trained with teacher-forcing (T-Forcing) [155] as well as professor-forcing (P-Forcing) [156]. For additional comparison, we consider the performance of WaveNet [157] as well as its GAN counterpart WaveGAN [158].

The results obtained on the Gaussian model dataset and on the Sines dataset are reported in Table 4.2. Numerical results show that our model is comparable with TimeGAN and beats all

[2]LOBSTER `https://lobsterdata.com/`

Table 4.2: Discriminative score for the Gaussian and Sines datasets. First and second best are highlighted in black and blue respectively.

Model	Gaussian Dataset						Sines Dataset
	Temporal Correlations (fixing $\sigma = 0.8$)			Feature Correlations (fixing $\phi = 0.8$)			
	$\phi = 0.2$	$\phi = 0.5$	$\phi = 0.8$	$\sigma = 0.2$	$\sigma = 0.5$	$\sigma = 0.8$	
	Discriminative score (lower the better)						
`CoMeTS-GAN (w/o cross-corr.)`	0.211 ± 0.007	0.195 ± 0.005	0.137 ± 0.012	0.187 ± 0.006	0.206 ± 0.018	0.143 ± 0.009	$--$
`CoMeTS-GAN (w/ cross-corr.)`	0.187 ± 0.003	**0.183 ± 0.005**	**0.115 ± 0.005**	**0.180 ± 0.009**	**0.164 ± 0.011**	**0.102 ± 0.006**	**0.013 ± 0.007**
TimeGAN	**0.175 ± 0.006**	**0.174 ± 0.012**	**0.105 ± 0.005**	**0.181 ± 0.006**	**0.152 ± 0.011**	**0.105 ± 0.005**	**0.011 ± 0.008**
RCGAN	**0.177 ± 0.012**	0.190 ± 0.011	0.133 ± 0.019	0.186 ± 0.012	0.190 ± 0.012	0.133 ± 0.019	0.022 ± 0.008
C-RNN-GAN	0.391 ± 0.006	0.227 ± 0.017	0.220 ± 0.016	0.198 ± 0.011	0.202 ± 0.010	0.220 ± 0.016	0.229 ± 0.040
T-Forcing	0.500 ± 0.000	0.500 ± 0.000	0.499 ± 0.001	0.499 ± 0.001	0.499 ± 0.001	0.499 ± 0.001	0.495 ± 0.001
P-Forcing	0.498 ± 0.002	0.472 ± 0.008	0.396 ± 0.018	0.460 ± 0.003	0.408 ± 0.016	0.396 ± 0.018	0.430 ± 0.027
WaveNet	0.337 ± 0.005	0.235 ± 0.009	0.229 ± 0.013	0.217 ± 0.010	0.226 ± 0.011	0.229 ± 0.013	0.158 ± 0.011
WaveGAN	0.336 ± 0.011	0.213 ± 0.013	0.230 ± 0.023	0.192 ± 0.012	0.205 ± 0.015	0.230 ± 0.023	0.277 ± 0.013

the other state-of-the-art approaches. In particular, our framework ranks as the first or second best in terms of discriminative score in all of the considered scenarios. We pursued an ablation study and measured the impact of the cross-correlation term in the critic score. In particular, in rows one and two of the table, we report the discriminative score when $\alpha = 0$ and $\alpha = 1$, respectively. Evidence shows that the additional cross-correlation factor contributes positively to the model's performance, especially in cases of high correlation between features (i.e., for high values of ϕ), allowing us to obtain results comparable to the performance of TimeGAN.

Our model differs structurally from TimeGAN because of the conditional nature in which our synthetic sequences are generated. In that work, the original time series is factorized into short sub-sequences, and the model outputs a synthetic set of such sub-sequences at inference time, but there is no natural and intuitive way to recombine them to obtain a longer sequence, retaining temporal continuation. Furthermore, training time for TimeGAN took 39 hours versus 4 hours and 20 minutes for our model. This makes our model particularly adequate for the stock market time series generation, given that the model can be efficiently trained on new data during market closure.

Diversity. We can generate arbitrarily long stock traces thanks to the *autoregressive application* of the model to the input time series. The generation process can be defined as follows: (1) the model is fed with a multivariate time series of length P, that is, a tensor in $\mathbb{R}^{P \times n}$ of original stock data; (2) the output of the model is a multivariate time series of length F that is a tensor in $\mathbb{R}^{F \times n}$ of synthetic stock data; (3) the last P time steps of the synthetic sequence are fed to the model; (4) steps 2 and 3 are repeated until a total number of generative steps is done.

By varying the random seed, we are able to generate several multivariate time series. In Figure 4.2 we see the execution of three runs, in addition to the real trace (orange line). As we can observe, the model does not incur self-induction (i.e., it does not generate a trend that remains indefinitely influenced by it). Furthermore, the model respects the diversity property, generating different sequences from the same input if the random noise changes, preserving the correlation dynamics, as will be further studied later in this chapter.

Stylized Facts. We investigate the properties of the synthetic traces by analyzing the most common stylized facts in the financial domain as discussed in Section 4.3.2. We measure the stylized facts on the autoregressive generation of stock mid-prices and volumes for 24 trading days. Given that a trading day lasts for 6.5 hours, we generate 9360 minutes in total. Results are averaged over 10 runs where we vary the seed and the randomization of the generation.

Figure 4.3 shows the histogram of the intraday log-returns with period $\Delta t = 1$ minute. Results

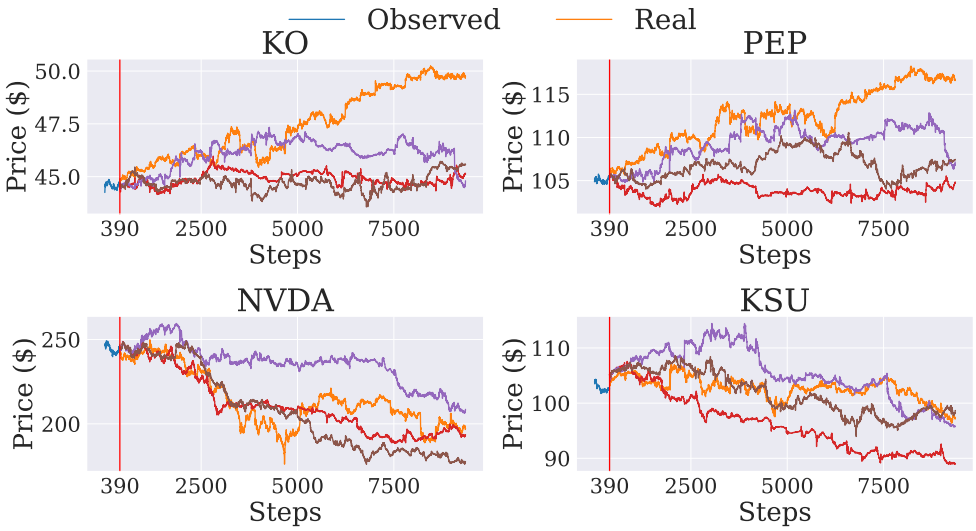


Figure 4.2: Diversity in price generation.

show a fat-tailed distribution as expected.

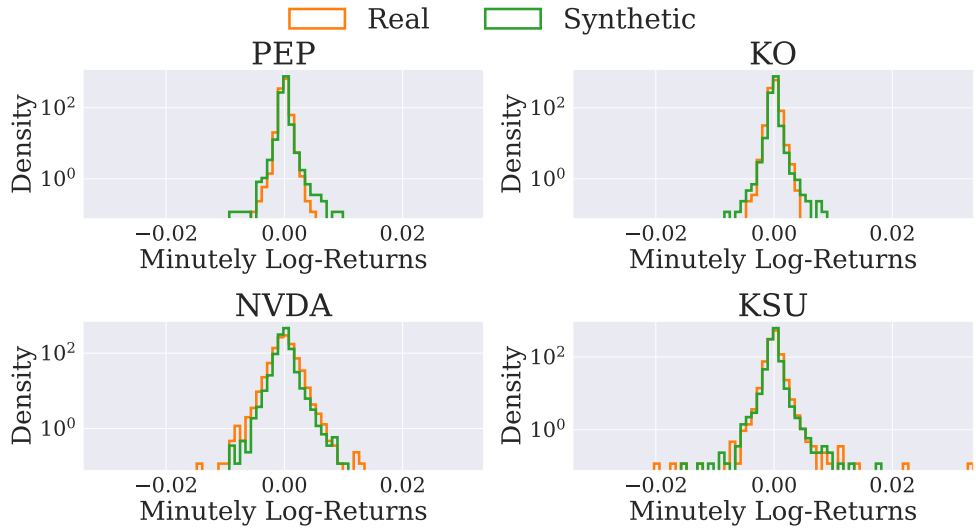


Figure 4.3: Intraday log-return distribution ($\Delta t = 1$ minute).

In Figure 4.4, as we increase the period Δt to 15 minutes, the distribution of returns better fits a normal distribution. The described property is often referred to as *aggregational normality*. As seen in the figures, both real and generated data closely follow these known properties for all the stocks.

In Figure 4.5, we plot the autocorrelation of the minutely intraday log-returns of the PEP stock with different time lags of 1, 10, 20, and 30 minutes. As we can observe, at increasing lag, the returns tend to cluster around a null correlation value. The stylized fact related to the *absence of autocorrelation* is thus present in both real and generated traces.

Figure 4.6 shows the autocorrelation coefficient of the volatilities at varying day lag. The volatility is computed as the standard deviation of the returns. We observe that the correlation is much higher when the lag is short and decreases as the lag increases, confirming that the volatility tends to cluster in time. The plot also confirms that all the considered stocks show this behavior

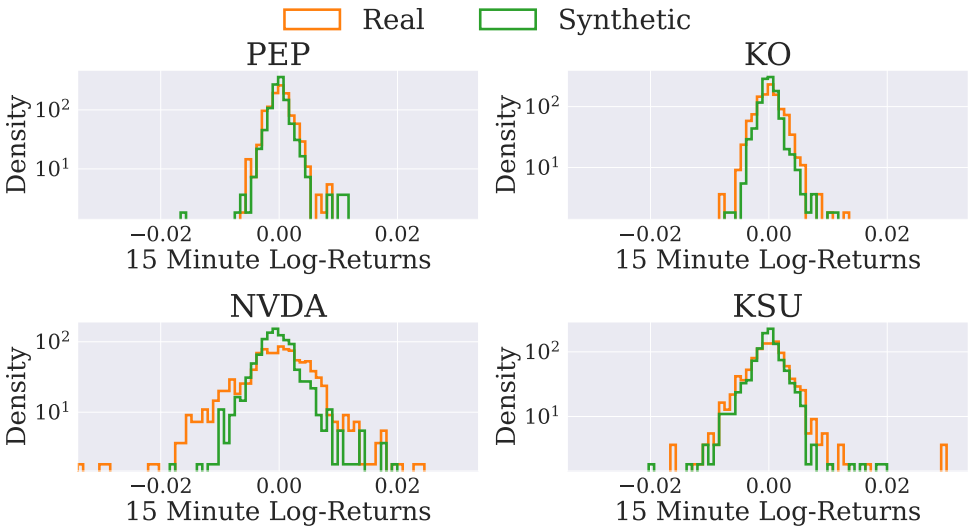


Figure 4.4: Intraday log-return distribution ($\Delta t = 15$ minutes).

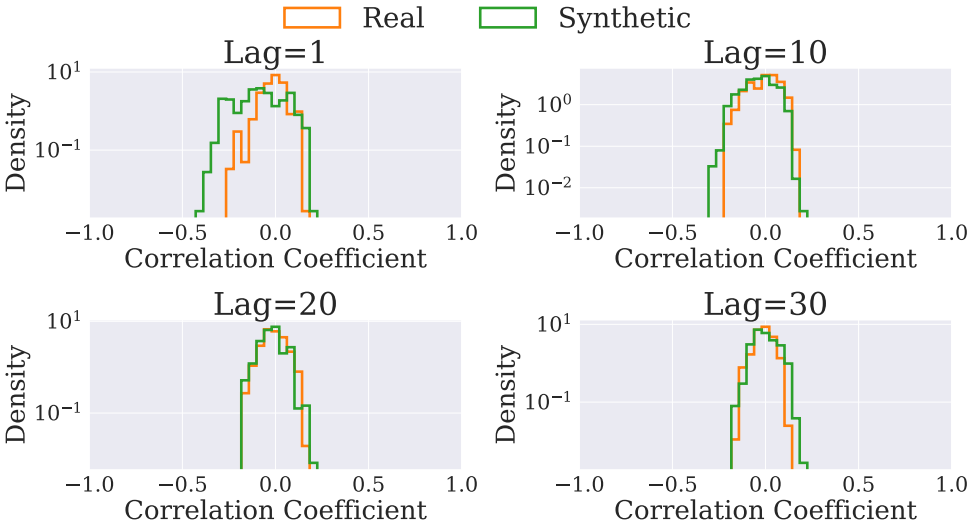


Figure 4.5: Autocorrelation of returns at increasing lags Δt of 1, 10, 20, 30 minutes for the PEP stock.

both in the real and generated traces.

Figure 4.7 shows the distribution of the correlation between the average traded volume and the volatility over two trading days. According to the stylized fact, trading volume and volatility are positively correlated. Nevertheless, this is not the case in the time period that we chose for our test, as the real distribution shows, the generator captures the real behavior well.

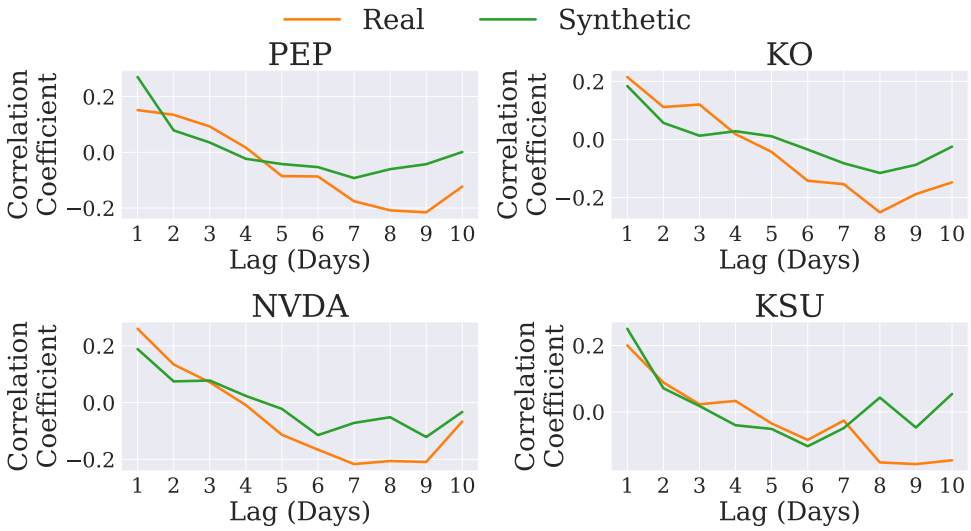


Figure 4.6: Correlation of the volatility at increasing day lag.

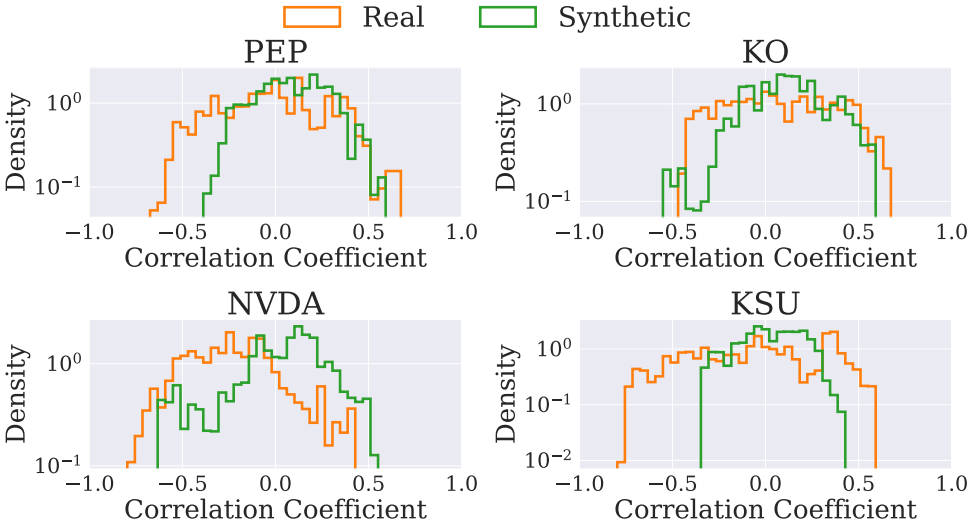


Figure 4.7: Volume-volatility correlation.

Cross-correlation distance. In this section, we examine the model's ability to capture the correlation dynamics that exist between stocks. We measure the distance between the cross-correlations of real and generated stocks. Formally, let $\rho(\cdot,\cdot)$ be the *Pearson correlation coefficient* (PCC). At the end of each epoch, for each of the $\binom{n}{2}$ pairs of stocks (S_i, S_j), we measure:

$$d_\rho(S_i, S_j) = \text{MSE}(\rho(S_i, S_j), (\rho(\widehat{S_i}, \widehat{S_j})),$$

where $\widehat{S_i}$ and $\widehat{S_j}$ are the synthetic version of the stocks S_i and S_j and MSE being the mean squared error.

As we can see in Figure 4.8, as the training of the framework proceeds, the average cross-correlation distance among all the pairs of stocks becomes negligible. This shows that the model is able to capture the correlation dynamics.

This is also evident in Figure 4.9 where we plot the prices of the stocks, and the average cross-correlation distance is 0.05.

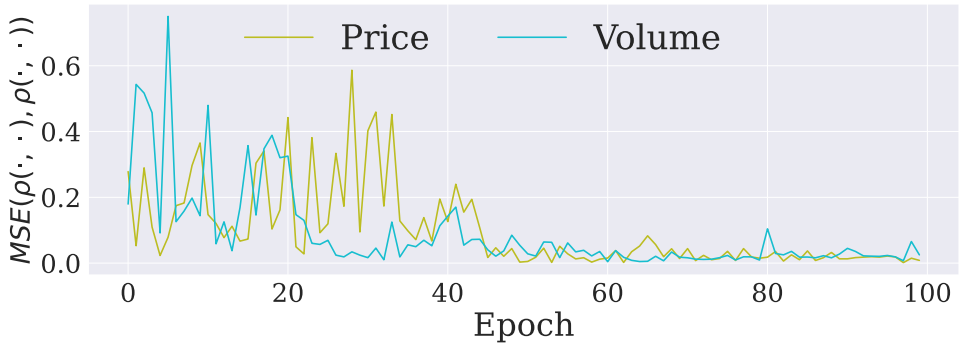


Figure 4.8: Average cross-correlation distance during training.

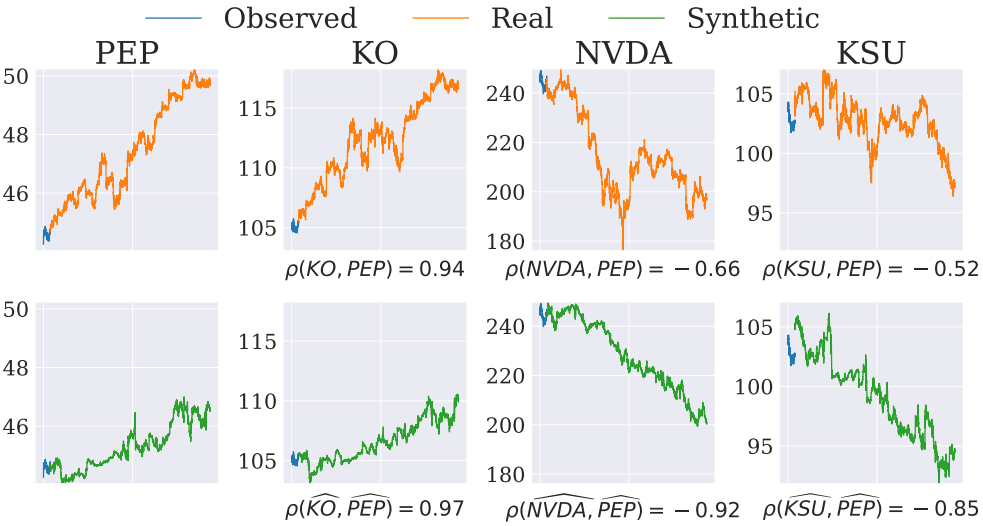


Figure 4.9: Price PCC between PEP and the other stocks.

Reactivity. We conducted an experiment in which a perturbation was manually introduced to the time series of KO to observe how the model reacts in generating the other stocks. In particular, we focus on the reaction of PEP, which is highly positively correlated with KO.

Specifically, let $\hat{\boldsymbol{x}}_{t:t+F}$ be the sub-sequence generated by the model starting at time t. A perturbation is introduced by substituting what the model actually generates with $\hat{\boldsymbol{x}}_{t:t+F} + \alpha \cdot \sigma(\hat{\boldsymbol{x}}_{t:t+F})$, where $\sigma(\cdot)$ is the standard deviation function.

Figure 4.10 shows the perturbation introduced on KO in the red window. It is interesting to observe how the model adjusts the generation of the non-perturbed stock in order to preserve the correlation following the perturbation event. Figure 4.11 shows the correlation of KO and PEP, varying the intensity of the perturbation. The orange line is the correlation between real KO and real PEP, which is highly positive, close to 1. The green line is the correlation between their synthetic versions $\widehat{\text{KO}}$ and $\widehat{\text{PEP}}$, which is still positive and high. Let $\widehat{\text{KO}_p}$ be the synthetic version of KO subject to the perturbation. The red line is the correlation between $\widehat{\text{KO}_p}$ and $\widehat{\text{PEP}}$. Let $\widehat{\text{PEP}_r}$ be the synthetic version of PEP generated by the model in reaction to the perturbation observed in KO. The purple line is the correlation between $\widehat{\text{KO}_p}$ and $\widehat{\text{PEP}_r}$. The experiment was repeated for several seeds, generating the standard error evidenced by the bands in the figure. As we can see, the model punctually intervenes in adjusting the generation of PEP to preserve the correlation with KO after the perturbation.

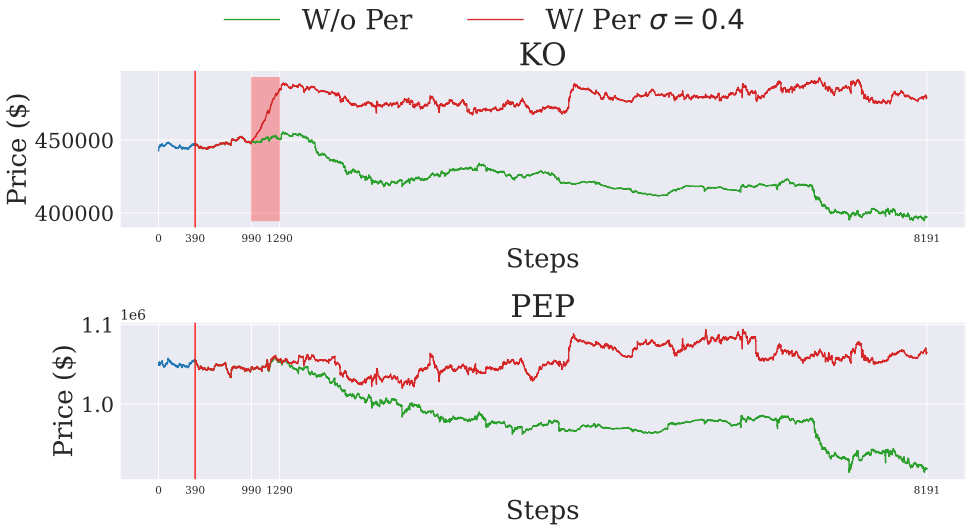


Figure 4.10: Perturbation of KO and the reaction of PEP.

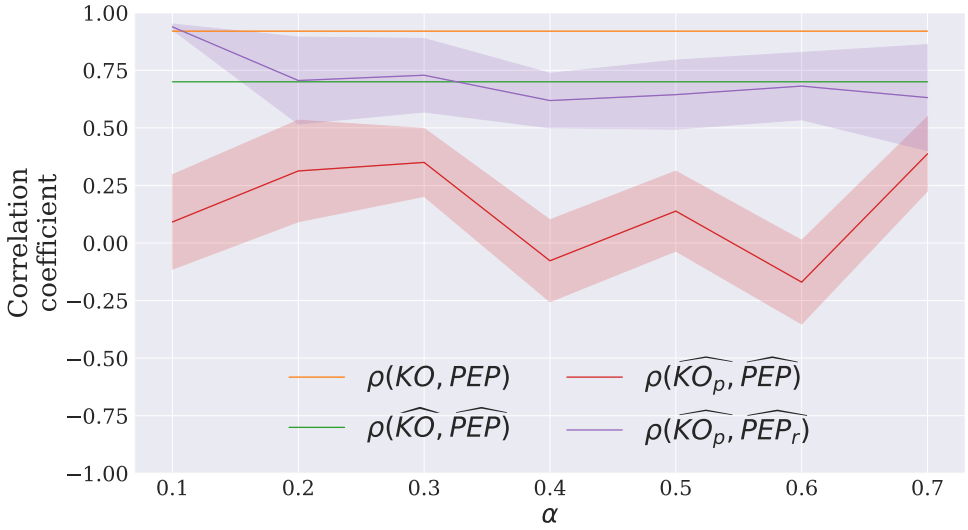


Figure 4.11: Correlation of the KO and PEP stocks varying the intensity of the perturbation.

Scalability. Considering the model's architecture in Figure 4.1, the size of the input and output of the generator and the input of the discriminator linearly depend on n. On the other hand, the intermediate hidden layers of the networks remain constant with respect to n. The linear layer in the critic takes as input a vector of size $\binom{n}{2}$. To test the model's scalability, we run the autoregressive application on all 30 stocks included in the DJIA index, obtaining realistic correlation dynamics. Indeed, the synthetic traces still show the expected correlation properties, even when considered in the simultaneous generation of numerous time series. Furthermore, all the stylized facts that characterize the returns are preserved. We do not show the visual results of this experiment for space reasons.

4.6 Discussion and Conclusion

In this work, we took a first step towards the analysis of interdependent markets through the use of deep learning models. We contribute on several levels. First, we create a new critic score for a C-WGAN, specifically suitable for time series generation, based on the degree of realism of the cross-correlation between the features. Then, we introduce a new architecture based on a C-WGAN to capture aspects of interdependence among financial markets and to generate time series with similar characteristics to real ones. Finally, we introduce a new metric, the cross-correlation distance, to keep track of the model's ability to capture the above interdependence dynamics during training. We thoroughly analyze the model's performance both on benchmark datasets and new stock market traces, measuring the performance in terms of *Discriminative Score*, *Diversity*, *Stylized Facts*, *Reactivity*, and *Scalability*. The results are promising: in all respects, the proposed architecture yields good results with respect to state-of-the-art approaches such as TimeGAN, solving its inherent issues related to training time and temporal continuation of long time series.

It is of interest to explore several promising directions to advance our research. The focus lies on investigating alternative generative models, such as diffusion models, to overcome the known instability of GANs. Another interesting avenue is the expansion of the work of Coletta et al. [133] by incorporating order book level data, enabling applications in multi-stock simulation contexts. Lastly, we will modify the model's architecture and explore new features to better understand stock interdependence. In particular, future studies will analyze cascading behaviors across different stocks. This deeper understanding will improve simulation realism and offer valuable insights into market behavior dynamics.

Chapter 5

DiffCATS: Causally Associated Time-Series Generation through Diffusion Models

Abstract

Modeling and recovering causal relationships in time-series data can be crucial for supporting real-world interventions and decision-making, but progress in Time-Series Causal Discovery (TSCD) is often limited by the lack of high-quality datasets with diverse and realistic temporal causal relationships. This highlights the need to provide synthetic time-series generation tools, with realism as a primary objective, an aspect that requires incorporating causal relationships beyond mere correlation. To address this challenge, we propose a diffusion model called DiffCATS. It simultaneously generates multiple causally associated time-series as well as a ground truth causal graph that reflects their mutual temporal dependencies, requiring only observational time-series data for training. Experiments demonstrate that it outperforms state-of-the-art methods in producing realistic time-series with causal graphs that closely resemble those of real-world phenomena. We highlight the practical utility of our data on three downstream tasks, including benchmarking widely used TSCD algorithms.

This work has been published in Transactions on Machine Learning Research [27].

5.1 Introduction

Many real-world time-series can be usefully modeled as arising from directed (causal) relationships among variables, which motivates methods for representing and learning such structure from data [159, 160]. Understanding such causal relationships is a well-recognized and important challenge for decision-making and policy formulation, as it facilitates predicting the consequences of interventions on underlying systems and variables [161].

Over the years, several works have studied these underlying causal structures, starting from Granger Causality (GC) [162]. Unable to capture how time affects causal relationships between interdependent time-series, GC has been complemented by Causal Graphs that incorporate the temporal lag in which causality unfolds [20]. Many approaches tackling the Time-Series Causal

Discovery (TSCD) problem [161] achieve satisfactory performance using statistical and machine learning techniques [23, 163–165], with discovered causal graphs closely resembling the ground-truth counterparts. However, the limited data available may hinder the development of new methodologies and studies, raising concerns about how existing algorithms would perform in unseen real-world scenarios [166].

Novel methodologies to generate realistic datasets with rigorously defined causal graphs are needed to support research and development of algorithms on time-series causal graphs. This challenge has been recently tackled by the works of [167] and [166], which mark an initial step in this direction, proposing two deep learning models to generate synthetic time-series data, while extracting the corresponding causal graphs. The first model focuses on the concept of Granger causality and proposes a recurrent Variational Autoencoder (CR-VAE) framework that naturally encodes causality into the weight matrix connecting input and hidden states. The second model introduces a comprehensive framework that supports prior causal graphs to generate realistic time-series data. However, when an input causal graph is not provided, the method extracts a hypothesized causal graph using explainability tools for feature importance (e.g., DeepSHAP [168]), which are inherently slow and only provide a posterior approximation of the ground-truth graph.

In this chapter, we introduce a novel generative framework called `DiffCATS` that combines the advantages of previous approaches by jointly generating time-series along with its causal graph, directly within a diffusion model architecture. Specifically, our model incorporates a τ-lag vector autoregressive structure (VAR(τ)) for multivariate time-series [169], where the coefficients are learned and generated through the diffusion process. This approach enables the simultaneous generation of realistic time-series data and the derivation of the corresponding ground-truth causal graphs from the VAR coefficients [170]. `DiffCATS` can be trained directly on time-series data without requiring prior causal graphs, eliminating the need for additional explainability tools.

Our work facilitates research and development of efficient algorithms for uncovering cause-and-effect relationships in multivariate time-series across diverse fields. In particular, the generated synthetic data can complement real-world data by providing known ground-truth with a diverse set of causal structures, which is especially valuable when real data are scarce or have limited ground-truth causal structures. Different from existing work, our approach specifically addresses the coherence between the synthetic sample and its corresponding causal graph (see Figure 5.1), resulting in more realistic and useful synthetic data. As the experiments highlight, the generated causal graphs are key to understanding the underlying dynamics of time-series, improving predicting capabilities (see Figure 5.2).

The main contributions of this work are the following:

- We present `DiffCATS`, a novel pipeline that employs a diffusion model to generate realistic time-series along with their related causal graphs.
- With extensive experiments, we demonstrate that our method outperforms existing approaches, in terms of synthetic time-series quality and fidelity of causal graphs to real-world phenomena.
- We conduct an evaluation of existing causal discovery algorithms using our synthetically generated datasets, highlighting the practical benefits of our data.
- We demonstrate the utility of the causal graph and its coherence with time-series data through two additional downstream tasks.

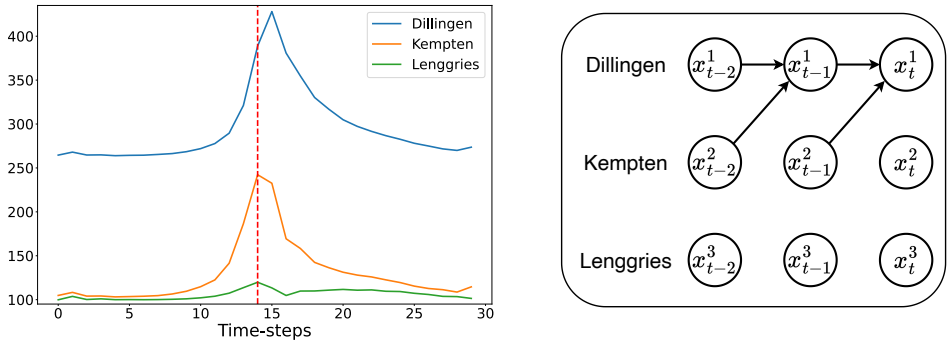


Figure 5.1: A synthetic sample and causal graph of three river discharges in which Kempten has an effect on Dillingen with a lag of 1.

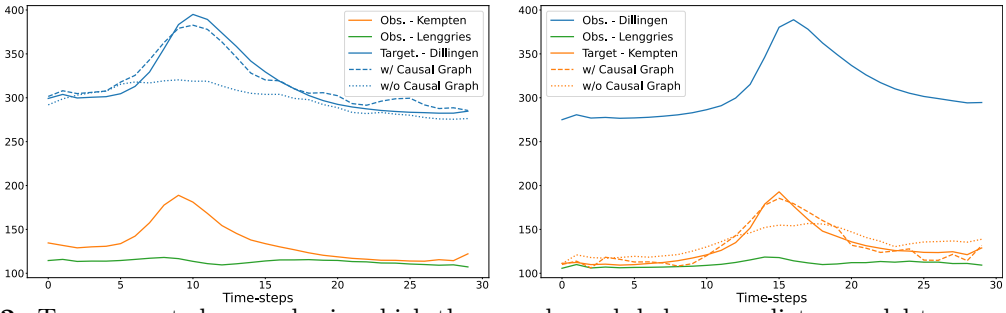


Figure 5.2: Two generated examples in which the causal graph helps a predictor model to reconstruct the target feature.

5.2 Related work

Several works have addressed the generation of synthetic time-series starting from real datasets [171–173]. Some approaches have focused on specific aspects, such as the correlation dynamics among variables [25, 174], user-specified constraints [175], or interpretable generation methods [176, 177]. However, only a few works delve into the generation of time-series along with their causal structure [166, 167].

The work of [167] proposed a VAE-based framework capable of learning Granger causal relationships from real multivariate time-series. This approach derives causal relationships from the weight matrix of the model connecting the input and hidden states, learning a unique Granger causality matrix from the data to which all generated samples adhere. A recent work of [166] proposed a pipeline to generate realistic time-series along with the causal graph. However, their framework does not output an interpretable-by-design time-series, but it performs the hypothetical causal graph inference through DeepSHAP [178] on the trained generative model, introducing a considerable time overhead.

Our goal is to further explore this area and address the gaps in the current literature. Specifically, we aim to provide a novel generative approach to simultaneously generate multiple causally associated time-series and their causal graphs, incorporating temporal lags. We strive to generate a unique causal graph for each synthetic sample, introducing greater variety in the data and providing a naturally interpretable architecture.

5.3 Problem Formulation

5.3.1 Background Knowledge

Causal Discovery The Causal Discovery task aims to identify cause-effect relationships among the variables of a d-variate time-series $\boldsymbol{x} = (\boldsymbol{x}^1, \ldots, \boldsymbol{x}^d)$. We say that $\boldsymbol{x}^i$ has an effect on (or causes) $\boldsymbol{x}^j$ if the two variables reflect a real phenomenon in which events reflected in the values of $\boldsymbol{x}^i$ affect $\boldsymbol{x}^j$. Trivially, the cause must precede the effect, so it is important to consider also the lag τ that elapses between observing the cause event on $\boldsymbol{x}^i$ and the effect event on $\boldsymbol{x}^j$. Causal Discovery algorithms are employed to observe real data and point out the existence of causal relationships according to which $\boldsymbol{x}^i$ causes $\boldsymbol{x}^j$, after τ time-steps, returning $(\boldsymbol{x}^i, \boldsymbol{x}^j, \tau)$.

Causal Graphs Causal relationships are often represented in the form of *Causal Graphs*. Let $\tau_{max} \in \mathbb{N}^+$ be the maximum number of discrete time-steps (δt) we are interested in to model the cause-effect phenomena of $\boldsymbol{x}$. We define a Causal Graph $G = (V, E)$ where the vertices V represent the time-series variables for the various time-steps between $-\tau_{max}$ and 0, and the edges E represent their causal relationships. In particular, an edge $(x^i_{t_1}, x^j_{t_2}) \in E$ indicates that the variable x^i has a causal implication on the value of the variable x^j with a lag of $t_2 - t_1$ time-steps (i.e., $x^i_{t_1} \Rightarrow x^j_{t_2}$). Formally, $V = \{x^i_{t-l} \,|\, 0 < i \leq d,\, 0 \leq l \leq \tau_{max}\}$ and $E = \{(x^i_{t_1}, x^j_{t_2}) \,|\, x^i \Rightarrow x^j \text{ with a lag of } t_2 - t_1 \geq 0\}$. Figure 5.1 illustrates a causal graph describing the interdependencies among rivers according to the observations made by [179]. It shows that variations in the water level of one river affect the level of the other one.

5.3.2 Task Definition

Let $\mathcal{D} = \{\boldsymbol{x} \mid \boldsymbol{x} \in \mathbb{R}^{L \times d}\}$ be a set of d-dimensional input time-series of length L. Our goal is to use the data in $\mathcal{D}$ to train a generative model that best approximates the distribution of the time-series, while simultaneously learning the corresponding causal structures. In particular, we aim at generating couples $\langle \hat{\boldsymbol{x}}, \hat{g} \rangle$ where $\hat{\boldsymbol{x}} \in \mathbb{R}^{L \times d}$ is a synthetic time-series and $\hat{g}$ is a causal graph. We want $\hat{\boldsymbol{x}}$ to be similar to the time-series observed in $\mathcal{D}$, i.e., to show a realistic behavior that reflects similar statistical properties. We also require $\hat{\boldsymbol{x}}$ to reproduce realistic causal relationships, as observed in $\mathcal{D}$, and described by the causal graph $\hat{g}$ that will explain $\hat{\boldsymbol{x}}$ in terms of causal relationships. We formally define a causal relationship between two time-series variables in the following section (Definition 5.4.1).

5.4 Methodology

Our framework, illustrated in Figure 5.3, builds upon a powerful class of generative models, the diffusion models [18]. [1] Originally designed to generate realistic images, diffusion models are used in this context to generate synthetic samples $\langle \hat{\boldsymbol{x}}, \hat{g} \rangle$. Unless otherwise noted, we adopt two common assumptions of the Causal Discovery literature [23, 163, 164, 166]: *Markovian conditions*

[1] Although the proposed methodology is general and can, in principle, be applied to alternative generative paradigms such as VAEs [16] or GANs [180], these models often underperform diffusion models due to more restrictive assumptions and less stable training dynamics. Consequently, we focus on a diffusion-based architecture, which we carefully design for our setting.

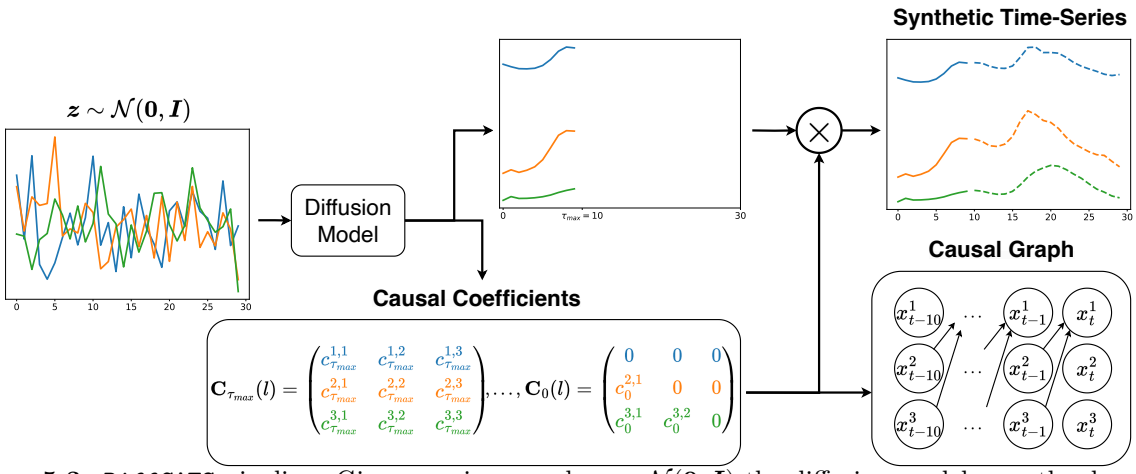


Figure 5.3: `DiffCATS` pipeline. Given a noisy sample $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ the diffusion model runs the denoising steps and outputs (i) an initial prefix of τ_{max} steps of the multivariate time-series and (ii) causal/VAR coefficients $\{\boldsymbol{C}_\tau(l)\}_{\tau=0}^{\tau_{\max}}$, whose entries $c_\tau^{i_1,i_2}(l)$ parameterize directed influences from variable i_1 at lag τ to variable i_2 at time l. For each $l > \tau_{\max}$, the remaining samples are produced via causal reconstruction (circle "×"), i.e., a VAR-style update using the previously reconstructed window and the generated matrices (including $\boldsymbol{C}_0(l)$). This yields the full synthetic time-series (solid prefix with dashed continuation). A sample-specific causal graph is then derived from the generated coefficients by retaining the most significant relationships, with nodes representing variables across time lags (e.g., $x_{t-10}^i, \ldots, x_t^i$).

and *faithfulness*, as discussed in detail in Section 5.8.1. While these assumptions are crucial for causal discovery, the key consideration for generation from observational data is ensuring structural consistency between the synthetic time-series and the causal graph. Indeed, unlike previous works, we do not need to assume *stationarity* of the underlying causal phenomena, as our causal graphs will be strictly related to individual samples.

5.4.1 Diffusion framework

A diffusion model is a type of latent variable model that operates through two key processes: the *forward process* and the *reverse process*. Given a sample $\boldsymbol{x}_0 \in \mathcal{D}$[2], the forward process gradually adds Gaussian noise to obtain a noisy sample $\boldsymbol{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. Specifically, given the parameters $\beta_t \in (0, 1)$ to schedule the amount of noise added at diffusion step $t \in [1, T]$, the noisy sample $\boldsymbol{x}_t$ is given by $\boldsymbol{x}_t = \sqrt{\hat{\alpha}_t} \cdot \boldsymbol{x}_0 + \sqrt{1 - \hat{\alpha}_t} \cdot \boldsymbol{\epsilon}$ where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, $\alpha_t = 1 - \beta_t$, and $\hat{\alpha}_t = \prod_{i=1}^{t} \alpha_i$.

The reverse process performs the actual generation of a new sample starting from Gaussian noise. Following the formulation of [18], we perform the denoising procedure from $\boldsymbol{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ according to the following equation:

$$\boldsymbol{x}_{t-1} = \beta_t \cdot \frac{\sqrt{\hat{\alpha}_{t-1}}}{1 - \hat{\alpha}_t} \cdot \hat{\boldsymbol{x}}_0 + \frac{(1 - \hat{\alpha}_{t-1}) \cdot \sqrt{\alpha_t}}{1 - \hat{\alpha}_t} \cdot \boldsymbol{x}_t + \mathbb{1}_{\{t>0\}} \cdot \beta_t \cdot \frac{1 - \hat{\alpha}_{t-1}}{1 - \hat{\alpha}_t} \cdot \boldsymbol{\epsilon}. \tag{5.1}$$

In the above equation $\mathbb{1}_{\{.\}}$ is the indicator function, $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, and $\hat{\boldsymbol{x}}_0 = \textsc{Den}_\theta(\boldsymbol{x}_t, t)$ is the output of a neural network $\textsc{Den}_\theta$ parametrized by θ, trained with respect the following loss function: $\mathcal{L}_{Rec}(\boldsymbol{x}_0, \hat{\boldsymbol{x}}_0; \theta) = \|\boldsymbol{x}_0 - \hat{\boldsymbol{x}}_0\|_2^2$, where $\|\cdot\|_p$ indicates the ℓ_p-norm. In practice, $\textsc{Den}_\theta$ reconstructs the original sample taken from the dataset by filtering out the noise added during the forward process.

In addition to the ℓ_2-norm, we also consider other loss functions to improve the performance of the reconstruction, such as the Dynamic Time Warping-based term $\mathcal{L}_{DTW}(\boldsymbol{x}_0, \hat{\boldsymbol{x}}_0; \theta)$ introduced

[2] In this section, the subscript refers to the diffusion step, not the lag index used elsewhere.

by [181]. The training objective is:

$$\mathcal{L}(\boldsymbol{x}_0, \hat{\boldsymbol{x}}_0; \theta) = \mathbb{E}_{\substack{t\sim\mathcal{U}(1,T) \\ \boldsymbol{x}_0\sim\mathcal{D}}}[\mathcal{L}_{Rec}(\boldsymbol{x}_0, \hat{\boldsymbol{x}}_0; \theta) + \lambda_1 \cdot \mathcal{L}_{DTW}(\boldsymbol{x}_0, \hat{\boldsymbol{x}}_0; \theta)], \tag{5.2}$$

where λ_1 is a parameter weighting the additional term.

The architecture of DEN_θ consists of an initial convolutional layer followed by a series of RESNET and ATTENTION blocks (see Section 5.8.2 for more details).

5.4.2 Causal reconstruction of the time-series

This section details how the output process inherently embeds a causal structure, allowing for the generation of a coherent sample $\langle \hat{\boldsymbol{x}}_0, \hat{g} \rangle$. Given $\boldsymbol{x}_0 \in \mathcal{D}$, we denote with $\boldsymbol{x}_0(l)$ the value of the time-series at time l, for $l \in [1, L]$, and considering its components as distinct features, we denote with $x_0^i(l)$ the value of the i-th feature at time l, for $i \in [1, d]$.

Let $\tau_{max} \in \mathbb{N}^+$ be the maximum lag for the causal relationships in the synthetic time-series[3]. Simultaneously for each feature i, DEN_θ outputs the first τ_{max} steps of the time-series, i.e., $\hat{x}_0^i(l)$, $\forall\, 1 \leq l \leq \tau_{max}$, and a set of causal matrices $\mathbf{C}(l) = \left\{ \mathbf{C}_\tau(l) = \left(c_\tau^{i_1,i_2}(l)\right)_{1\leq i_1,i_2\leq d} \in \mathbb{R}^{d\times d} |\ 0 \leq \tau \leq \tau_{max} \right\}$ for all the remaining steps $\tau_{max} < l \leq L$, where $c_\tau^{i_1,i_2}(l)$ represents the impact of feature i_1 on i_2 with a lag of τ at time l. The reconstruction of the whole time-series according to causal relationships evolves through a Vector Autoregressive (VAR) model [170]: proceeding one step l at a time with $\tau_{max} < l \leq L$, $\hat{\boldsymbol{x}}(l) = \sum_{\tau=0}^{\tau_{max}} \mathbf{C}_\tau(l) \cdot \hat{\boldsymbol{x}}(l-\tau)$[4].

We underline that, even though the reconstruction can be described by a VAR model, the generation framework is not autoregressive. This is because the model does not consider previously generated outputs as inputs. It instead generates the initial time-steps and the coefficients simultaneously.

Finally, to encourage the model to focus on the most important causal relationships, we add a regularization term for the coefficients. While the ideal choice for such a function would be the ℓ_0-norm, this is difficult to optimize; therefore, we consider the ℓ_2-norm, as in [164, 167]. Specifically, the regularization is defined as:

$$\mathcal{L}_{Reg}(\hat{\boldsymbol{x}}_0; \theta) = \lambda_2 \cdot \sum\nolimits_{l=\tau_{max}+1}^{L} \sum\nolimits_{\tau=0}^{\tau_{max}} \|\mathbf{C}_\tau(l)\|_2, \tag{5.3}$$

where λ_2 is the weight associated with such a regularization term.

5.4.3 Causal Graph Extraction

While the broader field does not offer a single universally accepted characterization of causality in observational time-series, we will provide a flexible definition based on a predictive formulation that is both widely adopted in practice and grounded in robust theoretical principles. Indeed, given a synthetic sample $\hat{\boldsymbol{x}}_0$ reconstructed through the series of coefficients matrices $\boldsymbol{c}(l)$ it means that for each time-step $\tau_{max} \leq l \leq L$, for each feature $1 \leq i \leq d$, we have importance weights $\mathbf{C}_1(l), \ldots, \mathbf{C}_{\tau_{max}}$ assigned to the previously generated time-steps, i.e. the window $[\hat{\boldsymbol{x}}_0(l-\tau_{max}), \ldots, \hat{\boldsymbol{x}}_0(l-1)]$ and

[3]The maximum lag and time-step granularity can be decided according to the domain and expert knowledge.

[4]Note that the matrix modeling instantaneous relationships $\mathbf{C}_0(l), \forall l$ is generated as a lower triangular matrix with the elements on the main diagonal also set to 0, in agreement with [182].

instantaneous relationships in $\hat{\boldsymbol{x}}_0(l)$ according to $\mathbf{C}_0(l)$. We also call these coefficients the *explanation* of the synthetic sample. To infer the causal graph $\hat{g}$, we summarize the causal relationships from the VAR coefficients according to the following formal definition.

Definition 5.4.1. *Let ρ be the percentage of causal relationships we want to represent in the synthetic dataset. For a synthetic sample $\hat{\boldsymbol{x}}$, we say that $\hat{\boldsymbol{x}}^i$ $\langle\rho, q\rangle$-causes $\hat{\boldsymbol{x}}^j$ with a lag of τ if the q-quantile of the corresponding coefficients of the VAR model at lag τ is among the ρ% highest absolute values. Notice that ρ and q refer to the whole dataset and the single sample, respectively.*

Intuitively, the q-quantile value allows the identification of significant causal events within a sample (e.g., Figure 5.1) by aggregating the coefficients over the time-series. Additionally, the threshold ρ can be employed as a reference point to identify and constrain causality to focus on the most relevant causal relationships. The whole mechanism allows samples to have different causal graphs, i.e., some may exhibit dense connections while others may have none. In fact, unlike previous work, we do not assume stationarity of the causal relationships in the dataset: the causal graphs are strictly related to individual samples, enabling more realistic and diverse samples.

This approach enables a fair evaluation of TSCD algorithms (see Section 5.6.1). Specifically, a TSCD algorithm may fail to recover the ground-truth causal graph — derived from our prior knowledge of the system — if the given sample does not exhibit any causal effect but is still linked to that prior causal graph. Our dataset serves as an accurate and representative benchmark: each generated sample is explicitly linked to its corresponding causal graph, which may denote the absence of causal relationships when none emerge from the time-series.

Finally, we notice that our definition ensures that each identified causal link captures not only statistical predictability but also a meaningful, information-theoretic causal effect. In fact, our formulation is consistent with prior work that extends Granger causality with lag-specific formulations (e.g., [182]), as a significant predictive link also implies high transfer entropy from the cause to the effect [183], indicating a substantial information flow between two variables. This information flow is the type of relationship captured by our definition of causal links as discussed in more detail in Section 5.8.1.

5.5 Experiments

In the experiments section, we show that the proposed pipeline is able to generate high-quality synthetic samples along with coherent and realistic causal graphs. In this regard, we conducted an experimental campaign involving three different datasets. We compared our model against several state-of-the-art approaches to highlight its advantages. We evaluate the generated samples both quantitatively and qualitatively, using well-established metrics for synthetic time-series as well as metrics specifically designed to assess the realism of the causal graphs. The code to reproduce the experiments is publicly released[5].

5.5.1 Datasets

To evaluate the models' capability to generate time-series alongside their causal relationships, we utilize two real-world datasets (Rivers and AQI) and a synthetic dataset (Hénon) constructed using

[5] `https://github.com/giuseppemasi99/DiffCATS`

closed-form equations, for which we have ground-truth knowledge. The datasets are described in the following paragraphs, while we refer the reader to Section 5.8.2 for additional details, including the visualization of the ground-truth causal graphs.

• **Rivers**: introduced by [179], it consists of the average daily ($\delta t = 1$ day) discharges of the Iller River at Kempten, the Danube River at Dillingen, and the Isar River at Lenggries between the years 2017 and 2019. The Iller is a tributary of the Danube, and we expect that an increase in the water level of the former will flow into the latter within a day, i.e., with a lag of 1 time-step. In this case, $d = 3$ and the only causal relationship is $x_{t-1}^{\text{Kempten}} \Rightarrow x_t^{\text{Dillingen}}$.

• **Air Quality Index** (AQI): introduced by [166], it consists of the PM2.5 pollution index monitored hourly ($\delta t = 1$ hour) over the course of one year by 36 stations spread across Chinese cities. In this case, $d = 36$ and the available causal relationships are modeled through a Granger Causality matrix, which is based on the pairwise distances between sensors.

• **Hénon**: introduced by [167], this synthetic dataset consists of $d = 6$ coupled Hénon chaotic maps [184] in which there is one positive ($x_{t-2}^i \Rightarrow x_t^i$) and two negative causal relationships ($-x_{t-1}^i \Rightarrow x_t^i$ and $-x_{t-1}^i \Rightarrow x_t^{i+1}$), $\forall 1 \leq i \leq d$. The equations generating this dataset are described in Section 5.8.2.

In the experiments, the maximum lag τ_{max}, introduced in Section 5.4.2, is fixed to 2 for all the datasets, consistent with the maximum lag observed in the ground truth. A sensitivity analysis and a discussion of parameter τ_{max} are provided in Section 5.8.3. The sequence length is 32 for the Hénon and the Rivers datasets, and 24 for the AQI dataset.

5.5.2 Models

Benchmarks. We compare our model against the two most recent state-of-the-art works described in Section 5.2, namely CausalTime [166] and CR-VAE [167]. We also include two other baseline solutions using diffusion models: (i) Base-Diffusion, a vanilla diffusion framework applied to time-series generation; (ii) CSDI [185], appropriately adapted to a generation task as described by [175]. A detailed description of these models can be found in Section 5.8.2.

`DiffCATS`: We trained our model setting the λ parameters in Equation (5.2) and Equation (5.3) to $\lambda_1 = 0.01$ and $\lambda_2 = 1$, respectively. An extensive ablation study about the impact of each loss component along with additional loss functions, namely the ℓ_1-norm and a Fourier-based loss, is presented in Section 5.8.3. The full list of hyper-parameters is shown in the Appendix in Table 5.7. To extract the causal graph from the VAR coefficients, we set $\rho = 1\%$ and $q = 0.95$ (see Definition 5.4.1).

5.5.3 Evaluation Metrics

Evaluation of time-series. We evaluated the quality of the synthetic time-series using the following metrics for fidelity, usefulness, and diversity: Discriminative Score (Discr.) [171], Predictive Score (Pred.) [171], Authenticity (Auth.) [186], Maximum Mean Discrepancy (MMD) [187], and Cross-Correlation (xCorr.). All of them are well described in Section 5.8.2.

Evaluation of Causal Graphs. We note that, despite the existence of a causal phenomenon relating the variables of the datasets, not all the samples extracted from the long time-series may

exhibit clear evidence of this. For instance, concerning the Rivers dataset, even if the Iller is a tributary of the Danube, if there is no significant variation in the water level of the former, the phenomenon of causality cannot be observed. Indeed, the water level of the three rivers simply remains stable over substantial periods of time. As a result, many time-step windows extracted from the dataset will not provide evidence of the causal relationship. Considering this issue, we employ metrics that do not penalize missing causal relationships from a sample. In contrast, we want to penalize those causal relationships identified by the model when domain knowledge clearly indicates that such relationships do not exist in real data (for instance, a tributary cannot be caused by the recipient river). The recent works of [179] and [161] addressed this problem by introducing metrics based on the false positive rate of causal relationships.

Accordingly, we consider the Granger Causality False Positive Rate (GC-FPR) and the Causal Graph False Positive Rate (Graph-FPR), which account for the fraction of edges in the graph known to be incorrect. Notice that, since CR-VAE does not output a causal graph for each sample, we use the F1-Score to evaluate its causal relationships with respect to the ground-truth Granger Causality matrix.

We emphasize that the generative method proposed here is not designed to be used as a causal discovery algorithm, so we do not focus on the exact causal relationships expected to be extrapolated from the datasets. Rather than that, we focus on the realism of the causal graphs and their consistency with the corresponding generated synthetic time-series, ensured by the design of the generative pipeline.

Finally, we evaluated the Inference Time (Inf. time) of the models, i.e., the time to generate a synthetic sample and the corresponding graph.

5.5.4 Results

In this section, we discuss the results of our experiments. All the quantitative scores are shown in Table 5.1. We report the results of `DiffCATS` with respect to the state-of-the-art approaches discussed in Section 5.5.2, namely, Base-Diffusion and CSDI for comparing the generated time-series, and CausalTime and CR-VAE for comparing both time-series and causal graphs. We point out that Base Diffusion and CSDI are included as reference points, to show that `DiffCATS`' time-series quality is not far from high-performing time-series-only generators, while additionally providing causal graphs. All the results report the mean and standard deviation across 10 different seeds.

Considering only the models that are able to generate both the time-series and the corresponding causal graph, we can see that regarding the fidelity and the quality of the synthetic time-series, `DiffCATS` outperforms the other approaches in terms of MMD on all three datasets, maintaining a satisfactory degree of Authenticity. It is also the best model concerning the Discriminative Score on two out of three datasets, and in all the other cases, it obtains scores very close to the benchmark. This validates our generated samples with respect to their originality, usefulness, and indistinguishability from real data.

The results comparing `DiffCATS`' time-series with those generated by the two state-of-the-art models capable of producing only time-series demonstrate a comparable level of quality between them. This ensures that the generation of the time-series through the VAR coefficients does not sacrifice the ability of the model to produce high-quality samples.

Table 5.1: Results of the models on the three datasets, where ↓ indicates *lower is better* and ↑ indicates *higher is better*. For each metric, the best result is highlighted in bold, and the second-best result is underlined (restricted to models able to generate both time-series and causal graphs).

Dataset	Metric	Models Time-Series Only		Models Time-Series & Causal-Graphs		
		Base Diffusion	CSDI	DiffCATS	CausalTime	CR-VAE
Hénon	Discr. ↓	0.032 ± 0.008	0.026 ± 0.011	$\mathbf{0.032 \pm 0.017}$	0.311 ± 0.142	$\underline{0.243 \pm 0.113}$
	Pred. ↓	0.170 ± 0.005	0.221 ± 0.006	$\mathbf{0.156 \pm 0.009}$	$\underline{0.200 \pm 0.011}$	0.244 ± 0.012
	Auth. ↑	0.679 ± 0.005	0.905 ± 0.015	$\underline{0.693 \pm 0.008}$	$\mathbf{0.720 \pm 0.031}$	0.651 ± 0.111
	MMD ↓	0.001 ± 0.000	0.001 ± 0.000	$\mathbf{0.001 \pm 0.000}$	$\underline{0.002 \pm 0.000}$	0.012 ± 0.009
	xCorr ↓	0.034 ± 0.005	0.094 ± 0.005	$\mathbf{0.029 \pm 0.005}$	$\underline{0.060 \pm 0.022}$	0.131 ± 0.031
	GC-FPR ↓	—	—	$\mathbf{0.311 \pm 0.001}$	$\underline{0.482 \pm 0.041}$	$0.520 \pm 0.070^*$
	Graph-FPR ↓	—	—	$\mathbf{0.017 \pm 0.000}$	$\underline{0.231 \pm 0.010}$	—
	Inf. time ↓	1231ms	241ms	$\underline{1548\text{ms}}$	8790ms	$\mathbf{194\text{ms}}^*$
Rivers	Discr. ↓	0.074 ± 0.007	0.019 ± 0.022	$\mathbf{0.067 \pm 0.010}$	$\underline{0.090 \pm 0.050}$	0.110 ± 0.090
	Pred. ↓	0.030 ± 0.001	0.043 ± 0.002	$\underline{0.033 \pm 0.001}$	$\mathbf{0.026 \pm 0.001}$	0.036 ± 0.002
	Auth. ↑	0.586 ± 0.008	1.000 ± 0.000	$\underline{0.630 \pm 0.010}$	0.560 ± 0.030	$\mathbf{0.720 \pm 0.020}$
	MMD ↓	0.001 ± 0.000	0.002 ± 0.000	$\mathbf{0.001 \pm 0.000}$	$\underline{0.009 \pm 0.011}$	0.059 ± 0.029
	xCorr ↓	0.014 ± 0.004	0.018 ± 0.006	$\underline{0.020 \pm 0.010}$	$\mathbf{0.010 \pm 0.000}$	0.120 ± 0.020
	GC-FPR ↓	—	—	$\mathbf{0.220 \pm 0.000}$	$\underline{0.570 \pm 0.010}$	$0.370 \pm 0.140^*$
	Graph-FPR ↓	—	—	$\mathbf{0.070 \pm 0.000}$	$\underline{0.220 \pm 0.010}$	—
	Inf. time ↓	1252ms	239ms	$\underline{1492\text{ms}}$	4248ms	$\mathbf{148\text{ms}}^*$
AQI	Discr. ↓	0.287 ± 0.010	0.238 ± 0.068	$\underline{0.293 \pm 0.050}$	0.460 ± 0.020	$\mathbf{0.250 \pm 0.040}$
	Pred. ↓	0.043 ± 0.001	0.048 ± 0.001	$\underline{0.047 \pm 0.001}$	0.054 ± 0.001	$\mathbf{0.043 \pm 0.001}$
	Auth. ↑	0.860 ± 0.008	0.980 ± 0.013	$\mathbf{0.820 \pm 0.010}$	0.770 ± 0.010	$\underline{0.800 \pm 0.100}$
	MMD ↓	0.001 ± 0.000	0.018 ± 0.001	$\mathbf{0.001 \pm 0.000}$	$\underline{0.008 \pm 0.001}$	0.017 ± 0.001
	xCorr ↓	0.078 ± 0.005	0.080 ± 0.003	$\underline{0.070 \pm 0.010}$	$\mathbf{0.030 \pm 0.010}$	0.120 ± 0.010
	GC-FPR ↓	—	—	$\mathbf{0.390 \pm 0.000}$	$\underline{0.490 \pm 0.000}$	$0.270 \pm 0.000^*$
	Graph-FPR ↓	—	—	—	—	—
	Inf. time ↓	1079ms	242ms	$\underline{1395\text{ms}}$	205s	$\mathbf{442\text{ms}}^*$

Regarding the causal graphs, DiffCATS achieves both the best GC-FPR and the best Graph-FPR scores in all three datasets, demonstrating its effectiveness in extracting causal relationships from the VAR coefficients. This result is of critical importance given that it ensures the reliability of the graphs as a representation of the causal relationships exhibited by the time-series. For the AQI dataset, we report only the GC-FPR metric that evaluates the Granger Causality matrix, as no lag information is provided in the ground-truth causal phenomena. We recall that CR-VAE does not output a causal matrix for each sample, but it is learned and fixed in the trained model. Therefore, since the Granger causality matrix is the same for all generated samples, the GC-FPR metric does not fully capture the model's ability in this context. For this reason, we reported the F1-score of the learned matrix with respect to the ground-truth Granger causality matrix, highlighting room for improving performance.

Summarizing, we highlight that our model achieves the lowest Discriminative Score and MMD along with the best Graph-FPR, ensuring that the synthetic samples exhibit a high level of realness and the causal graphs are reliable. Furthermore, as evidenced in Figure 5.1, the model is able to generate synthetic traces with strictly associated causal graphs, eliminating the need for stationarity assumptions in causal relationships employed by previous works.

We also evaluated the inference time of the models to obtain a sample made up of the synthetic time-series and the corresponding causal graph. CR-VAE turned out to be the fastest, thanks to its VAE-based architecture. However, great time saving occurs because the causal graph is fixed for each sample since it is extracted from the parameters of the model. Our model achieves an inference time significantly lower than CausalTime, mainly because the post-processing of the feature importance through DeepSHAP is very time-consuming. Instead, in our architecture, the

causal graph is generated simultaneously with the time-series, with only a moderate overhead. Moreover, the sampling of diffusion models can be accelerated, using, for example, implicit diffusion models [188].

Additional experiments and results can be found in the Appendix, including the evaluation of the time-series through dimensionality reduction techniques, namely t-SNE and PCA [5.8.3], kernel density estimation [5.8.3], the evolution of the evaluation metrics during the training [5.8.3], the robustness of the graph extraction to noise in the time-series [5.8.3] and a different approach to extract the causal graph from the VAR coefficients using the Dixon's Q Test [5.8.3].

5.6 Downstream Tasks

5.6.1 Benchmark of Causal Discovery Algorithms

Our primary downstream task is to benchmark causal discovery algorithms. Given a generated couple $\langle \hat{\boldsymbol{x}}, \hat{g} \rangle$, we feed the algorithm with the generated time-series $\hat{\boldsymbol{x}}$ and we compare the predicted causal graph against the generated graph $\hat{g}$. Related work presenting this type of benchmark is described in Section 5.8.5. In our benchmark, we included the following approaches:

- **Granger-Causality**-based: Granger Causality (GC, [162]); Neural Granger Causality (NGC, [189]); economy-SRU (eSRU, [190]); Temporal Causal Discovery Framework (TCDF, [191]); CUTS [192]; CUTS+ [165];
- **LiNGAM**-based: ICA-LiNGAM [193]; VARLiNGAM [182]; DirectLiNGAM [194];
- **Constraint**-based: PCMCI+ [195];
- **Gradient**-based: NTS-NOTEARS [164]; DYNOTEARS [163]; Rhino [196];
- **CCM**-based: Latent Convergent Cross Mapping (LCCM, [197]);
- **Other**: Neural Graphical Model (NGM, [198]) employing neural ordinary differential equations.

The results are shown in Table 5.2, evaluated in terms of AUROC and AUPRC (Area Under Precision-Recall Curve). To always have a well-defined ground-truth, for this benchmark, we selected the strongest 15% causal connections for each sample. As additional experiments, we also executed the benchmark using the top 1% approach described in Section 5.4.3. These results are reported in Section 5.8.5.

The comparison between real and synthetic data performance demonstrates encouraging coherence across multiple causal discovery algorithms, suggesting that the synthetic data effectively captures key characteristics of real-world causal relationships. Several high-performing methods, including PCMCI+, CUTS, and CUTS+ show remarkably consistent performance between real and synthetic datasets, with AUROC differences typically within 0.05, indicating that the synthetic generation process successfully preserves the essential causal structure and complexity. The synthetic data proves particularly effective for methods like NGM and eSRU, which achieve consistently strong AUPRC scores (0.73-0.81) across both real and synthetic versions, demonstrating the synthetic data's ability to maintain predictive relationships. Notably, the AQI dataset shows the strongest real-synthetic correspondence across methods, while the synthetic Rivers dataset successfully replicates the challenging characteristics of its real counterpart. While some methods, such as NTS-NOTEARS and TCDF, exhibit performance variations, these discrepancies appear more attributable to algorithmic sensitivity rather than fundamental limitations in synthetic data quality. Overall, these results indicate that the synthetic data provides a high-quality approximation of

real-world causal dynamics, offering a valuable benchmark that captures the essential complexity needed for robust causal discovery evaluation.

Table 5.2: Results of the Causal Discovery algorithms on real and synthetic data. For real data, the **highest value** is shown in bold and the second-highest value is underlined. For synthetic data, green background indicates comparable performance to real data, while red background indicates a high difference from real data performance.

Metric	Method	Real Data			Synthetic Data		
		Hénon	Rivers	AQI	Hénon	Rivers	AQI
AUROC	GC	0.54 ± 0.09	0.69 ± 0.24	0.53 ± 0.04	0.52 ± 0.03	0.57 ± 0.07	0.50 ± 0.00
	DYNOTEARS	0.62 ± 0.11	0.54 ± 0.23	0.49 ± 0.03	0.60 ± 0.04	0.51 ± 0.03	0.50 ± 0.00
	NTS-NOTEARS	0.48 ± 0.00	0.50 ± 0.00	0.50 ± 0.01	0.57 ± 0.04	0.69 ± 0.10	0.50 ± 0.00
	PCMCI+	0.70 ± 0.14	0.71 ± 0.28	0.72 ± 0.04	0.74 ± 0.02	0.77 ± 0.06	0.74 ± 0.00
	Rhino	0.53 ± 0.01	0.45 ± 0.03	0.62 ± 0.02	0.51 ± 0.01	0.53 ± 0.06	0.50 ± 0.00
	CUTS	$\mathbf{0.80 \pm 0.10}$	0.62 ± 0.04	0.78 ± 0.02	0.75 ± 0.02	0.76 ± 0.06	0.74 ± 0.00
	CUTS+	0.76 ± 0.01	0.72 ± 0.01	$\mathbf{0.79 \pm 0.00}$	0.75 ± 0.02	0.75 ± 0.08	0.74 ± 0.00
	Neural-GC	0.68 ± 0.01	0.56 ± 0.02	0.50 ± 0.00	0.72 ± 0.01	0.52 ± 0.05	0.50 ± 0.01
	NGM	0.60 ± 0.03	0.68 ± 0.10	0.53 ± 0.03	0.61 ± 0.06	0.69 ± 0.12	0.50 ± 0.00
	LCCM	0.54 ± 0.02	0.50 ± 0.00	0.51 ± 0.00	0.55 ± 0.00	0.50 ± 0.00	0.52 ± 0.00
	eSRU	0.50 ± 0.00	$\mathbf{0.73 \pm 0.07}$	0.50 ± 0.00	0.50 ± 0.00	0.75 ± 0.11	0.50 ± 0.00
	TCDF	0.49 ± 0.02	0.50 ± 0.02	0.55 ± 0.02	0.52 ± 0.03	0.50 ± 0.01	0.50 ± 0.00
	ICA-LiNGAM	0.45 ± 0.06	0.72 ± 0.27	0.52 ± 0.01	0.44 ± 0.07	0.76 ± 0.29	0.49 ± 0.01
	VARLiNGAM	0.50 ± 0.06	0.56 ± 0.27	0.59 ± 0.03	0.50 ± 0.06	0.62 ± 0.31	0.48 ± 0.03
	DirectLiNGAM	0.51 ± 0.06	0.72 ± 0.29	0.52 ± 0.01	0.50 ± 0.06	0.68 ± 0.31	0.48 ± 0.01
AUPRC	GC	0.42 ± 0.07	0.40 ± 0.29	0.57 ± 0.02	0.47 ± 0.13	0.46 ± 0.10	0.57 ± 0.09
	DYNOTEARS	0.56 ± 0.02	0.62 ± 0.05	0.63 ± 0.01	0.52 ± 0.08	0.58 ± 0.03	0.65 ± 0.00
	NTS-NOTEARS	0.17 ± 0.00	0.17 ± 0.00	0.36 ± 0.03	0.45 ± 0.07	0.54 ± 0.13	0.42 ± 0.10
	PCMCI+	0.63 ± 0.09	0.65 ± 0.15	0.74 ± 0.04	0.68 ± 0.02	0.64 ± 0.05	0.73 ± 0.02
	Rhino	0.59 ± 0.04	0.64 ± 0.02	0.58 ± 0.03	0.70 ± 0.07	0.66 ± 0.07	0.69 ± 0.04
	CUTS	0.43 ± 0.07	0.65 ± 0.05	0.64 ± 0.03	0.68 ± 0.02	0.64 ± 0.05	0.73 ± 0.00
	CUTS+	0.76 ± 0.03	0.55 ± 0.05	0.78 ± 0.03	0.68 ± 0.02	0.62 ± 0.08	0.73 ± 0.00
	Neural-GC	0.66 ± 0.01	0.56 ± 0.06	0.62 ± 0.04	0.68 ± 0.01	0.55 ± 0.08	0.64 ± 0.05
	NGM	0.75 ± 0.03	0.73 ± 0.09	$\mathbf{0.81 \pm 0.06}$	0.77 ± 0.05	0.73 ± 0.11	0.80 ± 0.05
	LCCM	0.68 ± 0.01	$\mathbf{0.77 \pm 0.00}$	0.58 ± 0.00	0.67 ± 0.00	0.78 ± 0.00	0.57 ± 0.00
	eSRU	$\mathbf{0.76 \pm 0.01}$	0.77 ± 0.09	0.77 ± 0.00	0.78 ± 0.02	0.76 ± 0.10	0.81 ± 0.00
	TCDF	0.37 ± 0.09	0.51 ± 0.02	0.35 ± 0.02	0.65 ± 0.14	0.57 ± 0.03	0.64 ± 0.07
	ICA-LiNGAM	0.34 ± 0.05	0.51 ± 0.37	0.30 ± 0.02	0.33 ± 0.05	0.61 ± 0.40	0.28 ± 0.01
	VARLiNGAM	0.36 ± 0.06	0.33 ± 0.35	0.39 ± 0.04	0.36 ± 0.06	0.44 ± 0.41	0.27 ± 0.02
	DirectLiNGAM	0.37 ± 0.07	0.53 ± 0.39	0.30 ± 0.01	0.37 ± 0.06	0.52 ± 0.42	0.27 ± 0.01

Regarding the performance of the algorithms, three of them — namely, PCMCI+, CUTS, and CUTS+ — achieve the best tradeoff between AUROC and AUPRC on all datasets. Additionally, NGM achieves satisfactory results on the Hénon and Rivers datasets, yielding AUPRC values among the highest. Instead, Neural-GC performed well only on the synthetic dataset of our benchmark, while eSRU performed well only on the Rivers dataset. While LiNGAM-based approaches struggled

with the non-linear Hénon dataset, ICA-LiNGAM and DirectLiNGAM demonstrated competitive performance on the real-world datasets (Rivers and AQI), particularly in terms of AUROC. Among the constraint-based approaches, only PCMCI+ achieved satisfying performances, while, in general, the Granger-Causality-based approaches proved to be the best ones. None of the methods got an AUROC lower than 0.5, meaning that there were no inverted classifications. The overall performance of tested algorithms is lower than what has been reported on simpler synthetic datasets, such as Lorenz-96 [165, 189], where some methods achieved near-perfect scores. This performance gap may suggest that current algorithms are still imprecise for certain samples and datasets, and they could be further refined to improve accuracy. In general, more challenging synthetic datasets should be used to rigorously test and potentially improve existing TSCD methods. The performance degradation observed in some algorithms when exposed to new data further underscores the need for this approach.

Dealing with Latent Confounders. The existence of latent factors, called *confounders*, that influence both the independent variable (the cause) and the dependent variable (the effect) may lead to spurious associations, making it harder to determine the true causal relationship. Our generative model is designed to faithfully replicate the underlying characteristics of the input real dataset, including any statistical and causal properties that are present. As such, if the real dataset exhibits associations induced by latent confounders, synthetic data generated by our framework will reflect these characteristics as well.

[199] conducted an experiment in which the observed variable X is modeled as an Ornstein-Uhlenbeck (OU) process, the hidden confounder U as an independent spectrally-sparse OU process, and the effect time-series Y according to the equation $Y_t = \beta X_t + U_t + \eta_t$, where η_t is i.i.d. Gaussian noise. The objective is to infer the true causal effect of X on Y, i.e. the estimation of β. The authors show that their DecoR method achieves a lower Mean Absolute Error (MAE) in estimating β with respect to a standard Ordinary Least Squares (OLS) approach.

Building on this framework, we assessed the ability of `DiffCATS` to preserve the relevant statistical and causal properties of data generated from the true OU process. Specifically, we trained the generative model on data simulated according to the above OU processes, with the parameters described by [199]. We evaluate how well causal effect estimation methods could recover β on both real (directly simulated) and synthetic (generative model-produced) data. The results in Table 5.3 indicate that the generative model is able to retain the essential statistical and causal properties of the original OU-based data, as evidenced by the consistent estimation performance of both OLS and DecoR.

Table 5.3: MAE in estimating β.

	Real	Synthetic
OLS	0.76 ± 0.23	0.78 ± 0.19
DecoR	0.16 ± 0.14	0.19 ± 0.15

These findings show that: *(i)* existing deconfounding algorithms can be effectively applied to our synthetic data; *(ii)* synthetic datasets produced by our approach can serve as valuable testbeds for developing and validating new algorithms targeting latent confounding in observational data.

5.6.2 Causal Prediction and Classification

To demonstrate the utility of our dataset beyond benchmarking and improving causal discovery algorithms, we introduce the following two downstream tasks:

- **Causal Prediction**. We show that conditioning a prediction model on the causal graph (see example in Figure 5.2) significantly improves time-series reconstruction performance, underscoring both the utility and coherence of our graphs. In detail, we trained a 2-layer LSTM to predict the i-th feature of a time-series given the remaining features and the corresponding causal graph. Results are shown in Table 5.4. Additional details are reported in Section 5.8.3.
- **Causal Classification**. Classifying observational data (i.e., time-series) based on the underlying system dynamics (i.e., causal graphs) is a critical task across numerous domains. Indeed, causal classification has broad applicability, ranging from emergency detection (i.e., identify transitions to unsafe system dynamics from time-series data) to customer segmentation (i.e., assign customers to meaningful segments based on their responses to incentives — such as purchase history, engagement, or response to promotions — modeled as causal graphs). We demonstrate that our synthetic dataset can effectively support (augment) the training of classification models, particularly in scenarios where certain classes are underrepresented in real-world data. For demonstration, we organized the synthetic and real Rivers data into 11 distinct classes according to their causal graphs. We then trained a 2-layer LSTM to classify unseen real time-series, achieving an F1-score of 0.69, which highlights the utility of synthetic data. All the details are described in Section 5.8.3.

Table 5.4: Downstream task - Causal Prediction (Mean Absolute Error).

Dataset	**Predictor**	
	w/ Causal Graph	w/o Causal Graph
Hénon	$\mathbf{0.13 \pm 0.01}$	0.19 ± 0.01
Rivers	$\mathbf{0.01 \pm 0.00}$	0.02 ± 0.00

5.7 Conclusions & Limitations

We introduced `DiffCATS`, a novel pipeline to generate faithful time-series along with realistic and coherent causal graphs specifically suited for the TSCD task. To the best of our knowledge, this is the first work to incorporate the simultaneous generation of causal graphs for causally related time-series generation without the need for the stationarity assumption. We demonstrated that our model can effectively generate synthetic datasets to support the causal discovery community in enhancing their algorithms in various domains, learning directly from real-world observational data.

We acknowledge that one limitation of this work is that only linear causal relationships are present in the synthetic samples. In Section 5.8.3, we show a possible solution to accommodate non-linear causal links by increasing the polynomial degree employed by the VAR. We also note that `DiffCATS`' expressivity depends on the chosen maximum lag $\tau_{\max}$, which bounds the longest temporal dependencies the model can represent. Finally, the requirement of generating both the time-series and the causal graph could make its time-series-only metrics trail specialized generators (namely CSDI) and its inference slower due to iterative diffusion sampling with respect to competitors like CR-VAE. So practitioners may prefer CR-VAE for fastest sampling (but with a fixed

causal pattern for all synthetic samples), CSDI/time-series-only diffusion models when graphs are unnecessary, and `DiffCATS` when coherent per-sample graphs are needed without expensive post-hoc explanation.

In future works, this approach can be extended to add a loss-based guidance of the coefficients so that the generation can be conditioned on a prior-known causal graph. In fact, a key advantage of our approach is the realism and flexibility that diffusion models provide, which allows the implementation of sophisticated conditioning strategies on trained models.

5.8 Appendix

5.8.1 Theory

Assumptions

Our work makes the following assumptions, aligned with several TSCD algorithms:

- **Markovian Condition:** The joint distribution of the multi-variate time-series can be factorized into $P(\boldsymbol{x}) = \prod_i P(x_i|\mathcal{P}(x_i))$, i.e., every variable is dependent only on its parents.
- **Causal Faithfulness:** It assumes that the relationships between variables in the data faithfully reflect the true causal connections between them.

On the other hand, thanks to the fact that our approach generates the time-series and its strictly associated causal graph, we do not need to assume *causal stationarity*:

- **Causal Stationarity:** It states that all the causal links do not change over time.

Relation of Definition 1 to Granger Causality and Transfer Entropy

In this section, we provide additional details and intuitions to clarify how our Definition 5.4.1 relates to established notions such as Granger causality (GC) and Transfer Entropy (TE).

The intuition behind this is as follows:

1. **Connection to Granger causality.** Our method parameterizes inter-variable temporal dependencies using a VAR-type structure, which is the same modeling family typically used to define linear GC. When trained with an L_2 (MSE) objective, the resulting estimation is closely related to least-squares fitting of a linear autoregressive model. Under standard preprocessing/regularity conditions (e.g., standardized variables and limited predictor collinearity), larger VAR coefficients tend to correspond to predictors with larger marginal contributions in reducing prediction error.
2. **GC–TE equivalence under Gaussianity.** We rely on the result of [200], which shows that for jointly Gaussian processes (equivalently, linear models with Gaussian residuals), GC and TE quantify the same directed dependence up to a constant factor.

Connection to Granger causality. Our formulation identifies candidate links by thresholding the magnitude of learned VAR coefficients. Let $c_\tau^{i,j}(l)$ denote the coefficient multiplying $x_i(l-\tau)$ in the predictor for $x_j(l)$ at lag τ and time index l (with $\tau \in \{1, \dots, \tau_{\max}\}$). In our framework, these coefficients arise from the causal reconstruction module, while the overall model is trained to minimize an L_2 reconstruction loss: $\mathcal{L}_{\text{Rec}} = \|x_0 - \hat{x}_0\|_2^2$.

Minimizing MSE encourages the model to fit the conditional mean of each target variable given its predictors, which is consistent with least-squares estimation of a linear VAR. Moreover, under a linear-Gaussian noise interpretation (approximately Gaussian, homoscedastic residuals), minimizing squared error is equivalent to maximizing a conditional Gaussian likelihood.

In linear VAR formulations, GC is defined via the improvement in the prediction of a target when including the past of a candidate driver, relative to a restricted model that excludes it. Concretely, consider predicting the target coordinate $x_j(l)$ from the past window $\{x(l-1), \ldots, x(l-\tau_{\max})\}$. Let the *full* linear predictor include all variables' histories up to $\tau_{\max}$, and let the *restricted* predictor remove the regressor corresponding to $x_i(l-\tau)$ (or, in a stronger form, remove all lags of x_i). The "no directed influence at lag τ" null can be expressed as $H_0 : \ c_\tau^{i,j} = 0$, (and the "no Granger causality from i to j up to $\tau_{\max}$" null as $H_0 : \ c_\tau^{i,j} = 0 \ \forall \tau \in \{1, \ldots, \tau_{\max}\}$).

In our *sample-specific, time-varying* setting, the natural analogue is local in time: $H_0(l) : c_\tau^{i,j}(l) = 0$. Moreover, since Definition 1 aggregates coefficient magnitudes over time using a quantile operator, an "absence of link" consistent with our extraction rule can be stated as: $H_0 : Q_q\Big(\Big\{|c_\tau^{i,j}(l)| : \ \tau_{\max} \le l \le L\Big\}\Big) \le \varepsilon$, for a small ε and $Q_q(\cdot)$ denoting the q-quantile.

A common definition of linear GC strength is: $\mathcal{F}_{i\to j} = \log \frac{\mathrm{Var}(\varepsilon_j^{\mathrm{res}})}{\mathrm{Var}(\varepsilon_j^{\mathrm{full}})}$, where $\varepsilon_j^{\mathrm{full}}$ and $\varepsilon_j^{\mathrm{res}}$ denote the residuals of the optimal full vs. restricted linear predictors (with analogous multivariate generalizations using covariance determinants). Our model is trained to minimize the reconstruction loss (Section 4.1), and the causal reconstruction produces $\hat{x}$ through the VAR coefficients. Minimizing squared reconstruction error is precisely the criterion that (in a linear regression interpretation) minimizes residual variance. Thus, in the linear predictive regime, the learned coefficients $\{c_\tau^{i,j}(l)\}$ parameterize the same type of conditional-mean predictor that underlies Granger-style predictability.

We recognize that, our Definition 1 is *not* an explicit computation of $\mathcal{F}_{i\to j}$, nor a formal hypothesis test. Instead, it is a practical *edge-selection* rule based on the magnitude and persistence of the learned coefficients. Summarizing, this is consistent with GC in the following sense:

- If $c_\tau^{i,j}(l) \equiv 0$ for $\tau_{\max} \le l \le L$, then $Q_q(|c_\tau^{i,j}(l)|) = 0$ and the corresponding link will not be selected, matching the Granger-style "no predictive contribution" null.
- Under standardized variables and weak predictor collinearity, $|c_\tau^{i,j}(l)|$ serves as a practical proxy for the predictive importance of regressor $x_i(l-\tau)$ in the conditional-mean model for $x_j(l)$: larger $|c_\tau^{i,j}(l)|$ is typically associated with a larger increase in residual variance when that regressor is removed, and hence with larger Granger-style predictability gains. We explicitly state this as an *interpretability proxy*.

Equivalence of Granger causality and Transfer Entropy (TE). Standard GC measures how much the prediction-error variance decreases when a variable (or its past) is added to the predictor set. Transfer entropy (TE) can be written as a conditional mutual information:

$$\mathcal{T}_{i\to j} = I\Big(X_i^{\mathrm{past}};\, X_j(l) \mid X_{-i}^{\mathrm{past}}\Big),$$

where X_{-i}^{past} denotes the past of all variables except i (up to $\tau_{\max}$).

[200] shows that for jointly Gaussian processes (equivalently, linear models with Gaussian residuals), linear GC $\mathcal{F}_{i\to j}$ and TE $\mathcal{T}_{i\to j}$ quantify the same directed dependence up to a constant factor.

Using natural logarithms, this relationship can be written as $\mathcal{F}_{i\to j} = 2\,\mathcal{T}_{i\to j}$, (with the constant changing if a different logarithm base is used). Therefore, within the linear-Gaussian approximation, stronger directed predictability gains (GC) correspond to stronger information flow (TE). Consistent with the discussion above, our coefficient-thresholding rule should be interpreted as a practical proxy for such directed dependence in the linear-Gaussian regime, rather than an explicit computation of GC/TE in full generality.

5.8.2 Implementation Details

Datasets

Table 5.5 reports the most important statistics of our datasets and Figure 5.4 shows the real causal graphs of the datasets. We also include additional details not discussed in Section 5.5.1.

Table 5.5: Statistics of the datasets.

Dataset	Number of Training Samples	Sequence Length	Number of Variables	Number of Causal Relations
Hénon	11295	32	6	11
Rivers	9969	32	3	1
AQI	7246	24	36	354

(a) Hénon (b) Rivers

Figure 5.4: Real Causal Graphs (green and red arrows represent positive and negative causal relationships, respectively).

Hénon: As described by [167], it is generated by the following equations:

$$\mathbf{x}^1_{t+1} = 1.4 - (\mathbf{x}^1_t)^2 + 0.3 \cdot \mathbf{x}^1_{t-1}$$

$$\mathbf{x}^i_{t+1} = 1.4 - (e \cdot \mathbf{x}^{i-1}_t + (1-e) \cdot \mathbf{x}^i_t)^2 + 0.3 \cdot \mathbf{x}^i_{t-1}$$

with $i = 2, \ldots, d$, where the number of dimensions $d = 6$ and $e = 0.3$. The initial values are sampled from a standard Gaussian distribution. In this dataset, the causal graph consists of one positive relationship with a lag of 2 between a variable and itself ($x^i_{t-2} \Rightarrow x^i_t$) and two negative relationships. The first one is between the variable and itself with a lag of 1 ($-x^i_{t-1} \Rightarrow x^i_t$); the second one is between two consecutive variables again with a lag of 1 ($-x^i_{t-1} \Rightarrow x^{i+1}_t$).

Rivers: The data are provided by the Bavarian Environmental Agency: `https://www.gkd.bayern.de`.

Air Quality Index (AQI): It can be found at: `https://www.microsoft.com/en-us/research/project/urban-computing`. We recall that, as in [166] state, the causal relations in the AQI dataset are highly dependent on geometric distances. The graph contained in the dataset they released has

been extracted considering the Gaussian kernel and a threshold with respect to the geographic distances of the sensors. In particular,

$$w_{ij} = \begin{cases} 1, & \texttt{dist}(i,j) \leq \sigma \\ 0, & \text{otherwise} \end{cases}$$

where `dist` measures the distance between two sensors and σ is set to ≈ 40 km. See the work of [166] for more details.

Benchmark

We compare our approach against two state-of-the-art approaches, whose code is available from the respective repositories:

- CausalTime [166]: `https://github.com/jarrycyx/UNN`.
  It is an autoregressive model based on normalizing flows, able to observe some time-steps of the time-series and generate the subsequent step. Thanks to this architecture, the authors can extract the importance of each feature in the input time-series using an explainability technique,i.e., DeepSHAP, provided by [178], and eventually extract a causal graph.
- CR-VAE [167]: `https://github.com/hongmingli1995/CR-VAE`.
  It is based on a recurrent VAE made up of a multi-head decoder, in which the p-th head is responsible for generating the p-th feature of the time-series. Encouraged by a sparsity penalty on the weights of the decoder, it learns a sparse causal matrix able to encode causal relationships among the variables. Since the causal matrix is part of the model's parameter, it will be the same for each synthetic sample generated by the model, in contrast to our approach and CausalTime. Moreover, a notable limitation of CR-VAE is that it is restricted to the notion of Granger Causality, implying that it does not consider the concept of time lag in observing the causal relationships.
- CSDI [185]: `https://github.com/ermongroup/CSDI`. We modified the original CSDI model to make it suitable for the unconditional time-series generation task, following [175].

We tuned the hyper-parameters of these models on all the datasets and they are reported in Table 5.6.

Architecture of Denoising Network

Here we discuss the denoising network architecture. We slightly modified the architecture of the work of [188] released in the authors' repository[6] by adding a convolution layer to output the coefficients. The overall architecture of Den_θ is depicted in Figure 5.5. It consists of an initial convolution layer and a series of Resnet and Attention blocks shown in Figures 5.6 and 5.7, respectively.

To represent the denoising time-step t the model employs cosine embedding and an MLP block made up of 2 linear layers with the activation function $f(x) = x \cdot \sigma(x)$ in the middle, where σ represents the Sigmoid function $\sigma(x) = \frac{1}{1+e^{-x}}$. The time-step information is injected in all the Resnet blocks.

[6] `https://github.com/mirthAI/Fast-DDPM`

Table 5.6: Hyper-parameters of CausalTime, CR-VAE and CSDI.

Model	Hyper-parameter	Dataset		
		Hénon	Rivers	AQI
CausalTime	Share type	Decoder	Decoder	Decoder
	N. Epochs Train Phase 1	40	20	10
	N. Epochs Train Phase 2	10	10	5
	Learning rate	0.001	0.0001	0.0001
	Batch size	32	32	32
	Hidden_size	128	128	128
	N. Layers	2	2	2
	N. Heads	4	4	4
	Dropout p	0.1	0.1	0.1
	Flow length	4	4	4
CR-VAE	Hidden	64	64	64
	N. Iterations Train Phase 1	1000	1000	1000
	N. Iterations Train Phase 2	90000	9000	90000
	Learning rate	0.05	0.05	0.05
	Batch size	1024	1024	1024
CSDI	Epochs	39	79	59
	Batch size	16	16	16
	Learning rate	0.0001	0.0001	0.0001
	β Schedule	Quadratic	Quadratic	Quadratic
	β Start	$1e-06$	$1e-06$	$1e-06$
	β End	0.5	0.5	0.5
	Diffusion Timesteps (T)	50	50	50

In the pictures, Non-linearity and GroupNorm refer to the function $f(x)$ and Group Normalization, respectively. The Downsample block is just a $1d$-convolution with a stride equal to 2. The Upsample block is made up of a Nearest Interpolation and a $1d$-convolution.

The end part of the architecture is made up of $d+1$ convolutional layers. The first one is responsible for outputting the first τ_{max} steps of the time-series. Each of the remaining d convolutional layers is responsible for the coefficients related to the causality impact on one feature. Then, the coefficients are multiplied with the initial steps of the time-series and the final output is reconstructed following the formalization in Section 5.4.2.

The most important hyper-parameters are reported in Table 5.7.

Table 5.7: Hyper-parameters of the generative model.

Training epochs	Batch size	Learning rate	Diffusion Timesteps (T)	β schedule	Time-step embedding
50	32	1e-4	100	Linear `start`=0.0001, `end`=0.02	Cosine `dim`=128

Evaluation Metrics

In this section, we describe each evaluation metric in detail.

• Discriminative Score [171]: It measures the fidelity of synthetic time-series, evaluating to what extent they are indistinguishable from real ones. It consists of training an off-the-shelf 2-layer LSTM to distinguish real samples from synthetic ones. The model is trained for 30 epochs with a learning rate of $1e-4$, hidden size equals 8, and batch size set to 32. The loss function to be optimized is the Binary Cross Entropy where real samples are labeled as 1 and synthetic samples as 0. The score is formally defined as $|0.5 - \text{AUROC}|$, where AUROC is the area under the ROC (Receiver-Operating Characteristic) curve of the trained discriminator.

• Predictive Score [171]: It measures the usefulness of synthetic time-series for a downstream

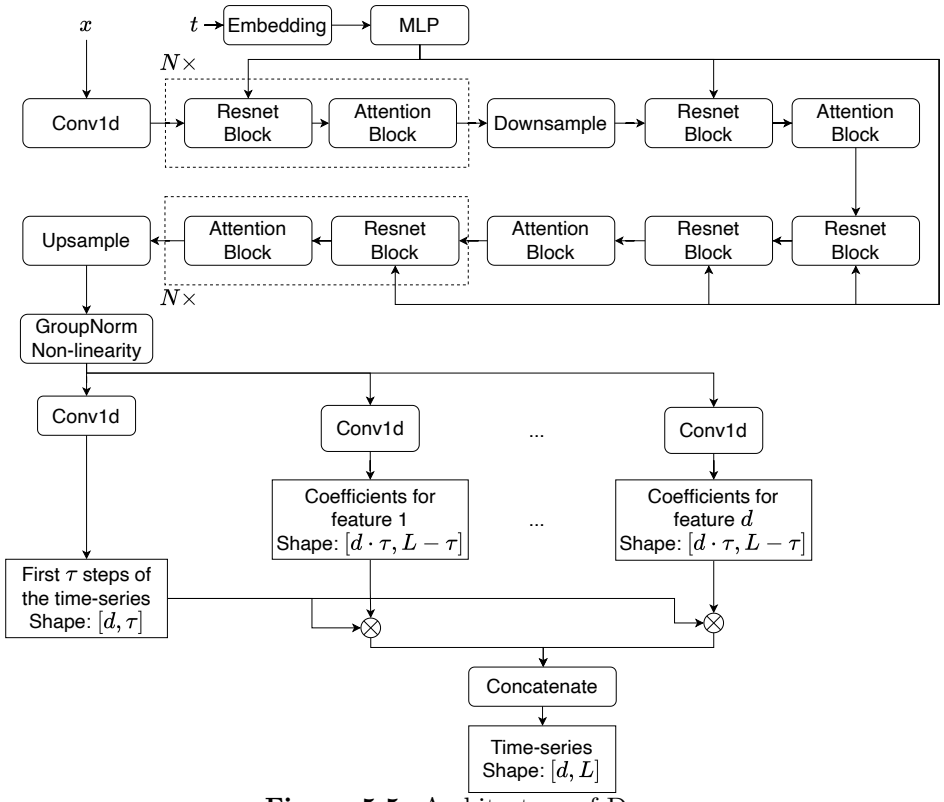


Figure 5.5: Architecture of DEN_θ.

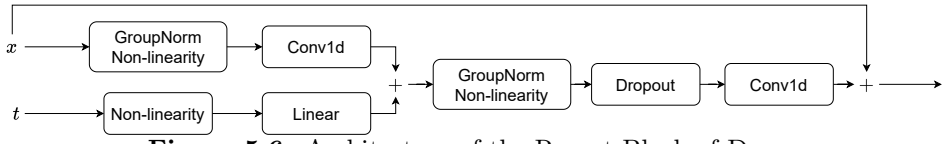


Figure 5.6: Architecture of the Resnet Block of DEN_θ.

prediction task. Following the *train-on-synthetic* and *test-on-real* criterion, a post-hoc sequence-prediction model is trained to predict the subsequent steps of a time-series on synthetic data and evaluated on real data in terms of the Mean Absolute Error (MAE) of the reconstructions. We trained a 2-layer LSTM-based predictor to forecast the last $\frac{1}{10} \cdot \texttt{seq\_len}$ time-steps over each synthetic sample for 10 epochs, with a learning rate of $1e-3$, hidden size equal to 32, and batch size set to 32. The loss function to be optimized is the ℓ_1-loss. Then, the predictor is evaluated on real data, and the error is quantified using the Mean Absolute Error (MAE). Formally, given a real sequence $\boldsymbol{x}$ of length $\texttt{seq\_len}$ let $\boldsymbol{x}_{first}$ and $\boldsymbol{x}_{last}$ be the first $\frac{9}{10} \cdot \texttt{seq\_len}$ and the last $\frac{1}{10} \cdot \texttt{seq\_len}$ time-steps, respectively. The predictor observe $\boldsymbol{x}_{first}$ and predicts the subsequent $\frac{1}{10} \cdot \texttt{seq\_len}$ time-steps, denoted as $\tilde{\boldsymbol{x}}_{pred}$. The MAE-based performance consists of $\frac{1}{\frac{1}{10} \cdot \texttt{seq\_len}} \sum_{t=1}^{\frac{1}{10} \cdot \texttt{seq\_len}} |\boldsymbol{x}_{last}(t) - \tilde{\boldsymbol{x}}_{pred}(t)|$.

• AUTHENTICITY [186]: It measures the proportion of synthetic data $A \in [0, 1]$ that is *authentic*, i.e., the models should not simply memorize the training dataset by generating copies of real samples it has observed, but should instead *create* new, original samples. We considered the original implementation provided by the work of [186]. The metric is evaluated through a hypothesis test

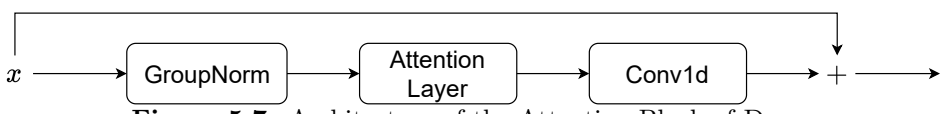


Figure 5.7: Architecture of the Attention Block of DEN_θ.

for data copying, which employs a nearest-neighbor classifier. A synthetic sample is considered unauthentic if it is closest to a real training sample. A score close to 1 indicates that the model is generating novel, unseen data.

• Maximum Mean Discrepancy [187]: It measures the similarity of synthetic and real time-series distributions. Formally, it is defined as $\text{MMD}^2(P,Q) = \mathbb{E}_P[k(X,X)] - 2 \cdot \mathbb{E}_{P,Q}[k(X,Y)] + \mathbb{E}_Q[k(Y,Y)]$ where $k(\cdot,\cdot)$ is the Radial Basis Function (RBF) kernel. We used the scikit-learn[7] implementation of the RBF kernel.

• Cross-Correlation: It measures the extent to which multiple synthetic time-series preserve the cross-correlation present in real data. Specifically, we evaluate the MAE between the correlation values of the real and synthetic features. We computed the Cross-Correlation distance for each lag up to 4. Formally, let $\boldsymbol{x}$ and $\hat{\boldsymbol{x}}$ be a real and a synthetic sample, respectively. Moreover, let $\boldsymbol{x}_i$ and $\hat{\boldsymbol{x}}_i$ be the i-th feature of the real and the synthetic sample ($\forall 1 \leq i \leq d$), respectively. The score is formally defined as $\sum_{\tau=0}^{4} \frac{1}{\binom{d}{2}} \cdot \sum_{\{i,j\} \in \binom{\{1,\dots,d\}}{2}} |(\boldsymbol{x}_i \star \boldsymbol{x}_j)(\tau) - (\hat{\boldsymbol{x}}_i \star \hat{\boldsymbol{x}}_j)(\tau)|$, where $(\boldsymbol{x}_i \star \boldsymbol{x}_j)(\tau)$ denotes the cross-correlation between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ with respect to lag τ.

5.8.3 Additional Results

Samples

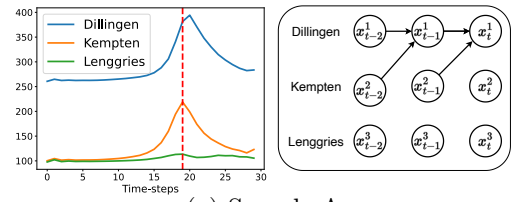
Figure 5.8 visually shows the algorithm for extracting the causal graph from the coefficients samples of the Rivers dataset, according to Definition 5.4.1. In the figure, (a) and (b) are two synthetic samples with different causal dynamics. Figure (c) shows the coefficients generating the Dillingen feature. In particular, for both samples, each histogram considers the impact of each of the other features on Dillingen. For instance, the pink histograms represent the impact of Kempten with a lag of 1 time-step on Dillingen since they are the coefficients of the 1 time-step lagged Kempten that are used to generate Dillingen. Similar to Figure (c), Figures (d) and (e) show the coefficients weighting the impact on the Kempten and the Lenggries features. The dashed black line in the figures is the q-quantile ($q = 0.95$)used in Definition 5.4.1, employed to quantitatively evaluate the generated coefficients over the time-series time-steps. All the q-quantiles are then reported in (f) and (g), for Sample A and Sample B, respectively, where the highest (in terms of magnitude) coefficients are selected to get the ρ% sparsity parameter used in Definition 5.4.1. Indeed, the dashed black line in Figures (f) and (g) is the threshold such that only the ρ% highest values are selected over the entire set of generated samples.

Figures 5.9 and 5.10 show additional examples of real and generated samples for the Rivers and Hénon datasets, respectively.

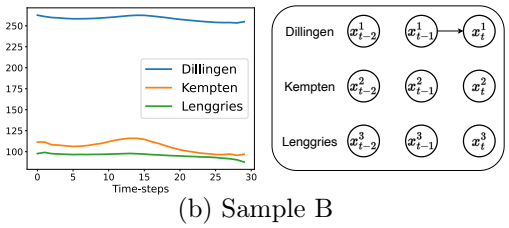
Additional ablation studies

Table 5.8 shows the quantitative results for the other variants of our model, highlighting the impact of each loss component. Additionally, `DiffCATS` w/L1 w/DTW considers a DTW-based loss and a ℓ_1-norm to regularize the coefficients; `DiffCATS` w/L2 w/Fourier considers a Fourier-based loss (weighted by a term $\lambda_3 = 100$), and ℓ_2-norm to regularize the coefficients. We considered the

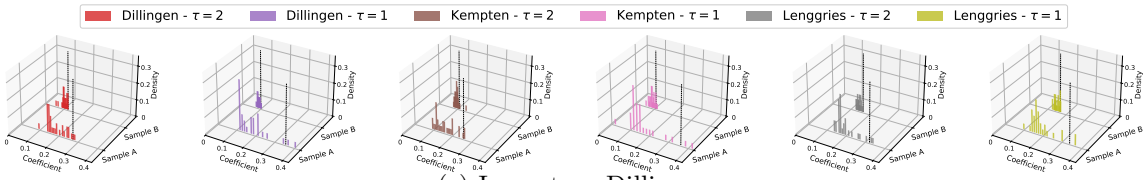
[7] https://scikit-learn.org/



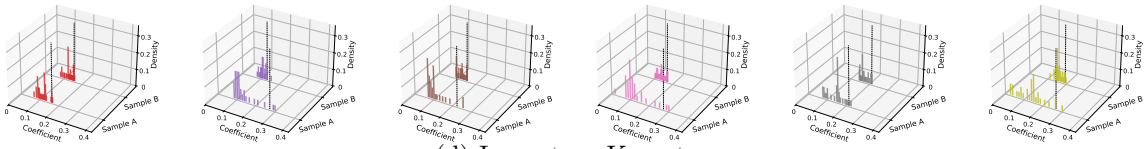
(a) Sample A



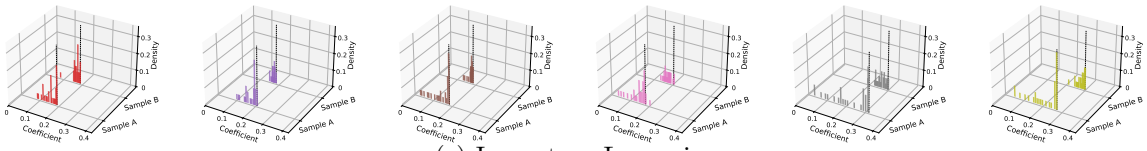
(b) Sample B



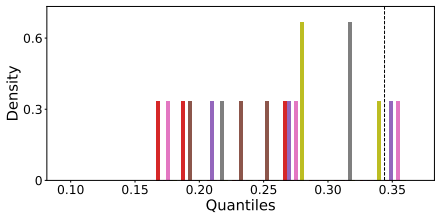
(c) Impact on Dillingen



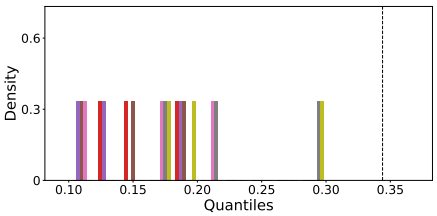
(d) Impact on Kempten



(e) Impact on Lenggries



(f) Sample A - Quantiles



(g) Sample B - Quantiles

Figure 5.8: Procedure to extract the causal graphs from the coefficients.

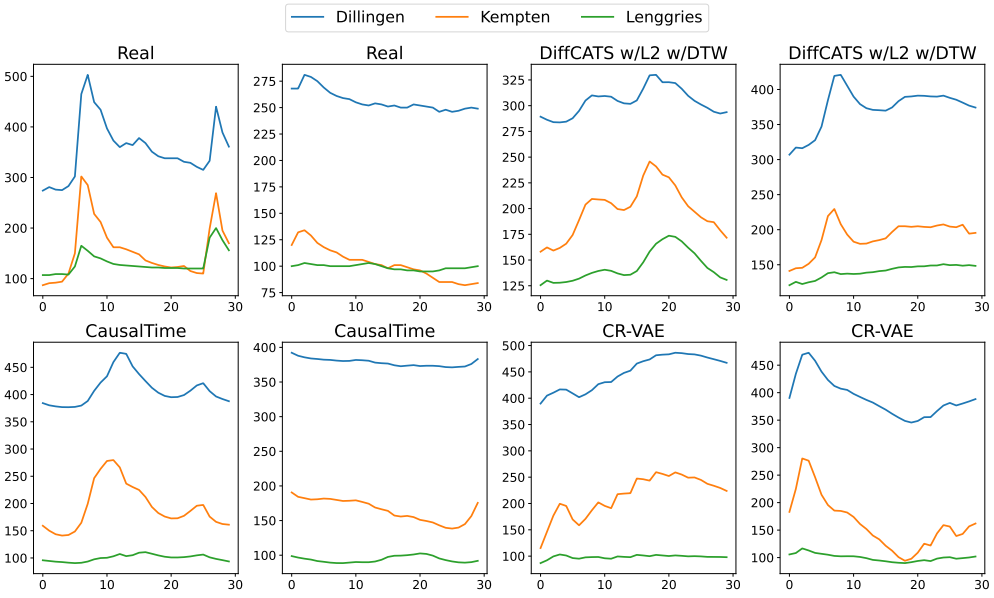


Figure 5.9: Examples from the Rivers dataset.

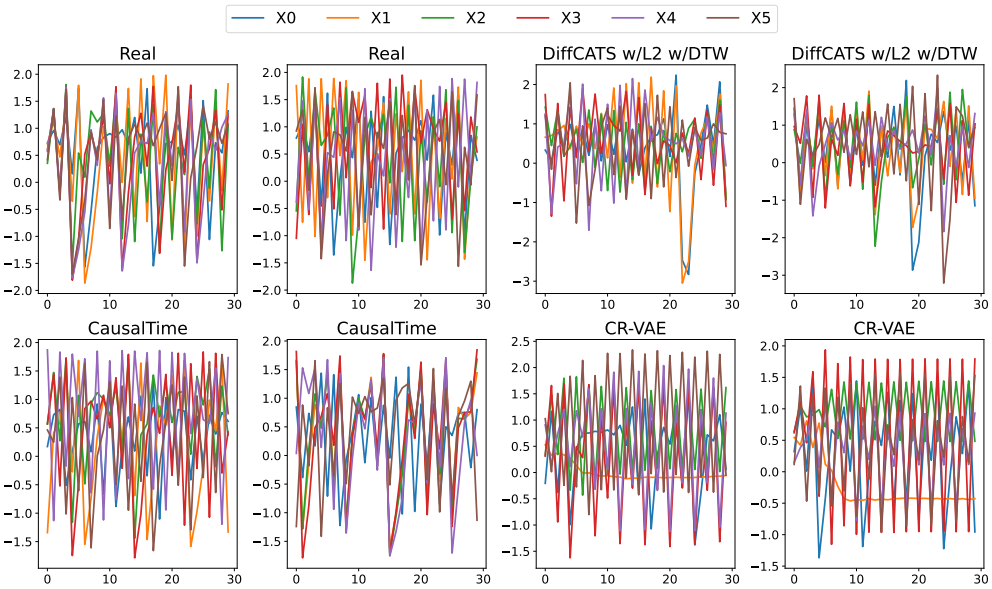


Figure 5.10: Examples from the Hénon dataset.

Fourier-based term employed by [176]:

$$\mathcal{L}_{Fourier}(\boldsymbol{x}_0, \hat{\boldsymbol{x}}_0; \theta) = \|\mathcal{FFT}(\boldsymbol{x}_0) - \mathcal{FFT}(\hat{\boldsymbol{x}}_0)\|_2^2, \tag{5.4}$$

where $\mathcal{FFT}(\cdot)$ indicates the Fast Fourier Transformation [201].

As the results show, incorporating the ℓ_2-norm of the coefficients into the objective loss as an attempt to sparsify the causal graph reduces the number of wrong connections. Moreover, the DTW-based loss considerably aids in extracting synchronization signals among the temporal sequences, significantly improving overall performance.

Sensitivity Analysis of Lag Parameters τ_{max}

In this subsection, we perform a sensitivity analysis by varying τ_{max} on the *rivers* dataset. In general, the parameter τ_{max} should be selected according to the time resolution and the expected causal dynamics of the underlying phenomena. However, when the true value is uncertain, a slight overestimation can be safely adopted. As shown in Table 5.9, the model maintains strong performance for both the generated time series and the causal graph, even when τ_{max} is overestimated or deviates from the true lag of the causal relationships (the maximum ground-truth lag is 2). This demonstrates the model's robustness in generating reliable synthetic data, even with imperfect domain knowledge.

Dimensionality Reduction

It is used to evaluate the diversity of synthetic samples, i.e., they cover the full variability of real samples. We employed t-SNE [202] and PCA [203] on both real and synthetic data to easily visualize how similar the two distributions are in a 2-dimensional space. We used the scikit-learn[7] implementation for both PCA and t-SNE. For each sample, we flattened the dimension of the features by computing the mean.

Figures 5.11 and 5.12 show the t-SNE and PCA plots of our best model against the state-

Table 5.8: Results of other models on the three datasets. ↓ indicates *lower is better* and ↑ indicates *higher is better*.

Dataset	Metric	Models Time-Series & Causal-Graphs				
		DiffCATS	DiffCATS w/L2	DiffCATS w/DTW	DiffCATS w/L1 w/DTW	DiffCATS w/L2 w/Fourier
Hénon	Discr. ↓	0.041 ± 0.026	0.045 ± 0.025	0.038 ± 0.022	0.035 ± 0.015	0.040 ± 0.024
	Pred. ↓	0.216 ± 0.010	0.217 ± 0.008	0.216 ± 0.010	0.217 ± 0.008	0.217 ± 0.0010
	Auth. ↑	0.663 ± 0.006	0.713 ± 0.007	0.662 ± 0.007	0.677 ± 0.008	0.700 ± 0.006
	MMD ↓	0.001 ± 0.000	0.001 ± 0.000	0.001 ± 0.000	0.001 ± 0.000	0.001 ± 0.000
	xCorr. ↓	0.031 ± 0.003	0.035 ± 0.004	0.029 ± 0.004	0.037 ± 0.005	0.035 ± 0.05
	GC-FPR. ↓	0.403 ± 0.001	0.421 ± 0.001	0.423 ± 0.001	0.397 ± 0.000	0.403 ± 0.001
	Graph-FPR. ↓	0.090 ± 0.000	0.080 ± 0.000	0.023 ± 0.000	0.040 ± 0.000	0.090 ± 0.000
Rivers	Discr. ↓	0.080 ± 0.010	0.130 ± 0.010	0.070 ± 0.023	0.480 ± 0.010	0.100 ± 0.010
	Pred. ↓	0.035 ± 0.001	0.037 ± 0.001	0.041 ± 0.002	0.043 ± 0.001	0.036 ± 0.001
	Auth. ↑	0.580 ± 0.010	0.620 ± 0.010	0.585 ± 0.012	0.870 ± 0.020	0.610 ± 0.010
	MMD ↓	0.001 ± 0.000	0.001 ± 0.000	0.001 ± 0.000	0.054 ± 0.003	0.001 ± 0.000
	xCorr. ↓	0.060 ± 0.000	0.060 ± 0.010	0.022 ± 0.004	0.080 ± 0.010	0.070 ± 0.010
	GC-FPR. ↓	0.230 ± 0.000	0.220 ± 0.000	0.158 ± 0.001	0.270 ± 0.000	0.230 ± 0.000
	Graph-FPR. ↓	0.100 ± 0.000	0.070 ± 0.000	0.074 ± 0.002	0.160 ± 0.000	0.070 ± 0.000
AQI	Discr. ↓	0.410 ± 0.020	0.430 ± 0.010	0.360 ± 0.015	0.440 ± 0.010	0.380 ± 0.030
	Pred. ↓	0.048 ± 0.001	0.048 ± 0.001	0.060 ± 0.002	0.049 ± 0.001	0.050 ± 0.001
	Auth. ↑	0.810 ± 0.020	0.810 ± 0.020	0.883 ± 0.008	0.820 ± 0.010	0.810 ± 0.010
	MMD ↓	0.001 ± 0.000	0.001 ± 0.000	0.001 ± 0.000	0.001 ± 0.000	0.001 ± 0.000
	xCorr. ↓	0.090 ± 0.010	0.110 ± 0.010	0.100 ± 0.003	0.130 ± 0.010	0.100 ± 0.010
	GC-FPR. ↓	0.480 ± 0.000	0.400 ± 0.000	0.390 ± 0.000	0.390 ± 0.000	0.400 ± 0.000
	Graph-FPR. ↓	—	—	—	—	—

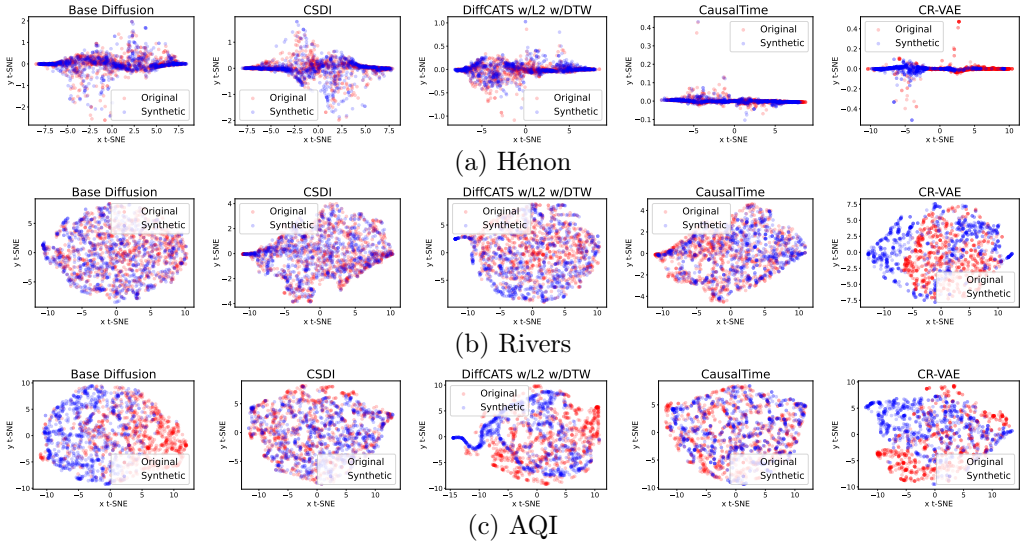
Table 5.9: Model performance under different lag values.

τ_{max}	Discr. score	Pred. score	Graph-FPR
1	0.099	0.032	0.022
2	0.067	0.033	0.070
3	0.060	0.031	0.078
4	0.050	0.033	0.086

of-the-art approaches. It can be observed that the distribution of the synthetic samples closely resembles the real one in two of the three datasets (Hénon and Rivers). This visually ensures that the model is generating realistic time-series in a diverse set of fields. Figures 5.13 and 5.14 show the dimensionality reduction results for the other variants of the model on all considered datasets.

Kernel Density Estimation

Figure 5.15 shows the data distributions drawn from kernel density estimation (KDE) of DiffCATS against the state-of-the-art approaches (ablations in Figure 5.16). KDE provides a visual assessment of the model's ability to capture the marginal distributions of the real data. A key insight from these plots is the high degree of overlap between the Original real data (red) and DiffCATS' data (blue) curves. This indicates that our model faithfully reproduces the statistical properties of the underlying data, including multi-modal distributions and peak densities. In contrast, baseline methods such as CR-VAE often exhibit smoother distributions that fail to capture the sharper density peaks observed in the Rivers and Hénon datasets, suggesting a tendency to over-regularize or average out specific data characteristics. The tight alignment achieved by DiffCATS confirms that the generated time-series preserves the fundamental distributional nature of the real-world phenomena.

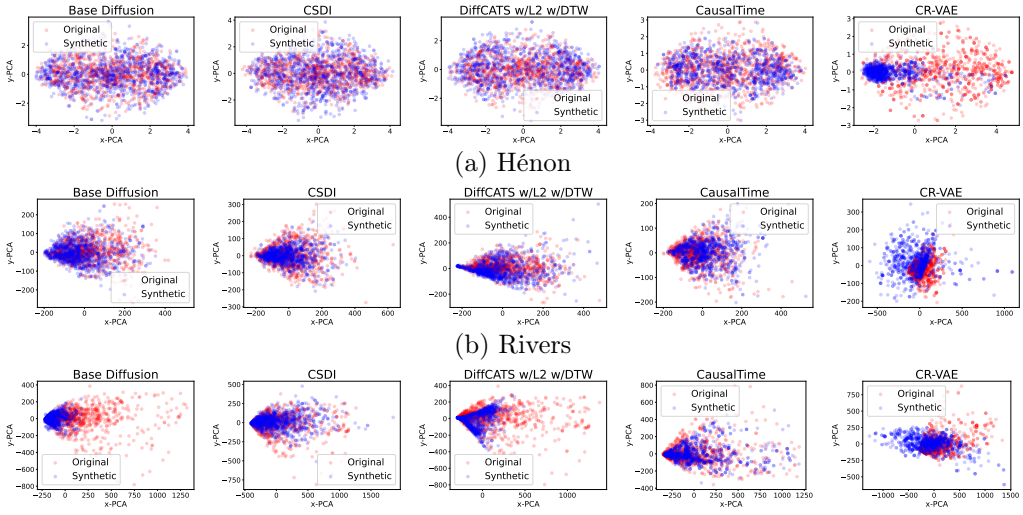


(a) Hénon

(b) Rivers

(c) AQI

Figure 5.11: Dimensionality reduction: t-SNE.

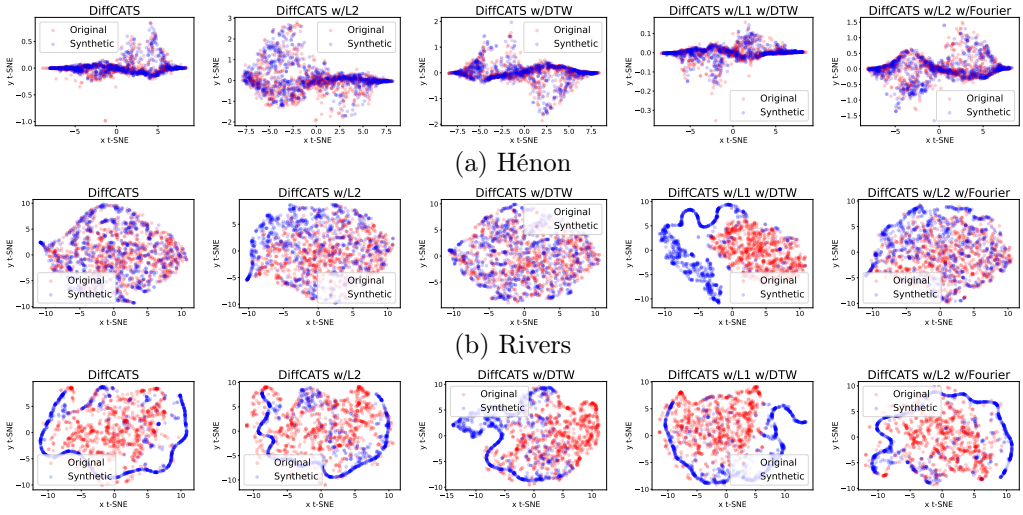


(a) Hénon

(b) Rivers

(c) AQI

Figure 5.12: Dimensionality reduction: PCA.

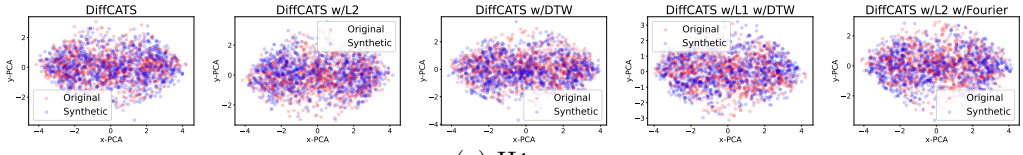


(a) Hénon

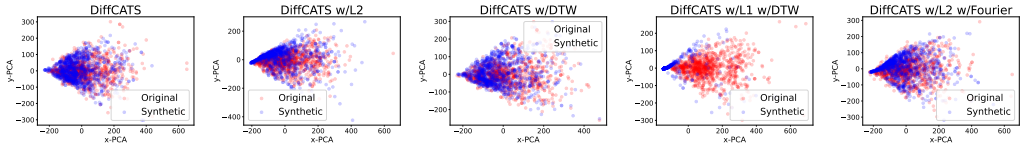
(b) Rivers

(c) AQI

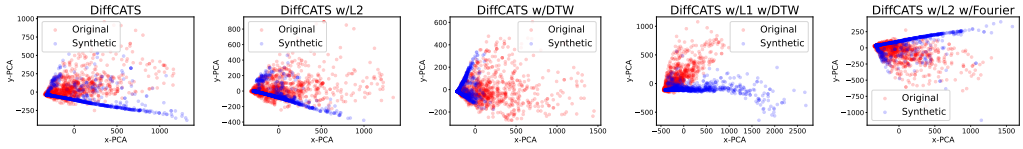
Figure 5.13: Ablation - Dimensionality reduction: t-SNE.



(a) Hénon

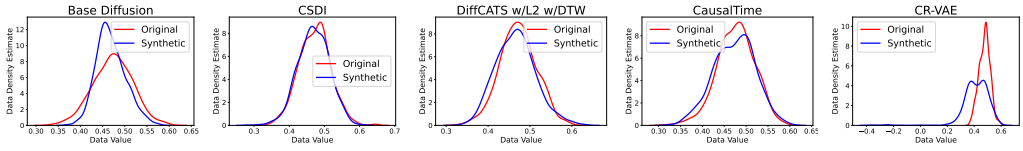


(b) Rivers

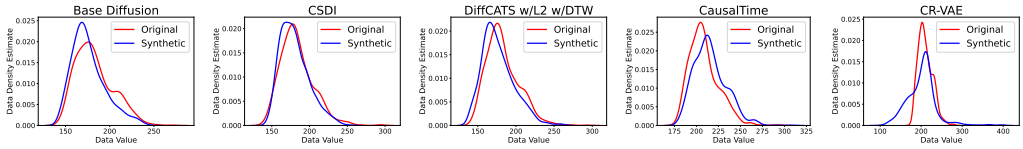


(c) AQI

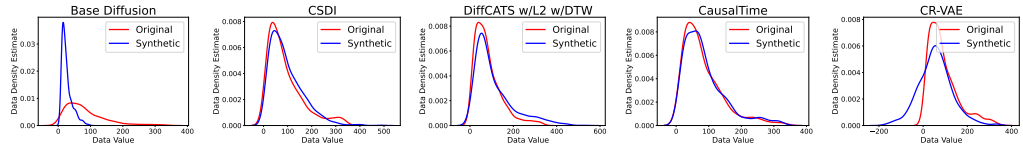
Figure 5.14: Ablation - Dimensionality reduction: PCA.



(a) Hénon

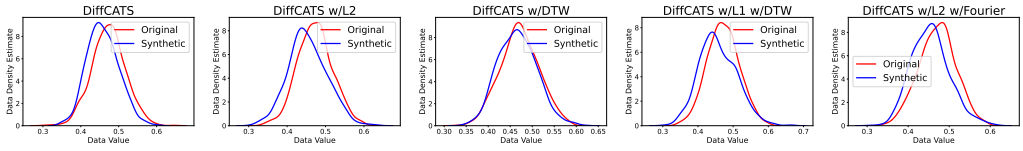


(b) Rivers

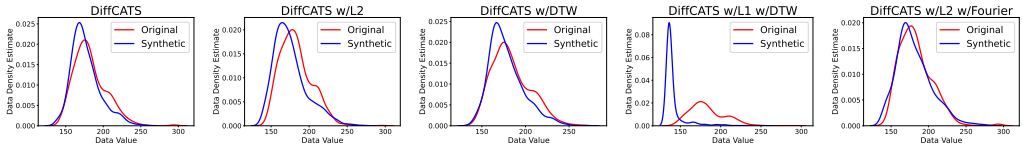


(c) AQI

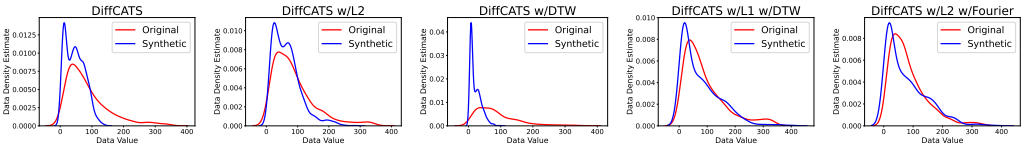
Figure 5.15: Kernel Density Estimation.



(a) Hénon



(b) Rivers



(c) AQI

Figure 5.16: Ablation - Kernel Density Estimation.

Evaluation Metrics during Training

Figures 5.17 to 5.19 show the evolution of the evaluation metrics during training on the Hénon, Rivers, and AQI datasets, respectively. This offers several insights into the learning stability and convergence of `DiffCATS`.

First, we observe a rapid convergence of the MMD and Discriminative Score, which drops significantly within the first 20 epochs. This indicates that the diffusion model quickly learns to generate samples that are statistically similar to, and difficult to distinguish from, real data.

Second, the Authenticity metric, while decreasing from an initial value of 1.0 (where noise is purely novel), stabilizes at a robust level (typically between 0.6 and 0.7). This confirms that while the model learns to approximate the real distribution, it does not simply memorize the training set; it continues to generate novel samples rather than copies.

Finally, the causal metrics (GC-FPR and Graph-FPR) show a consistent downward trend or stabilization concurrent with the time-series quality metrics. This demonstrates the effectiveness of the joint optimization strategy: the model successfully refines its understanding of the causal structure (the causal graph) simultaneously with the temporal dynamics, without one objective destabilizing the other.

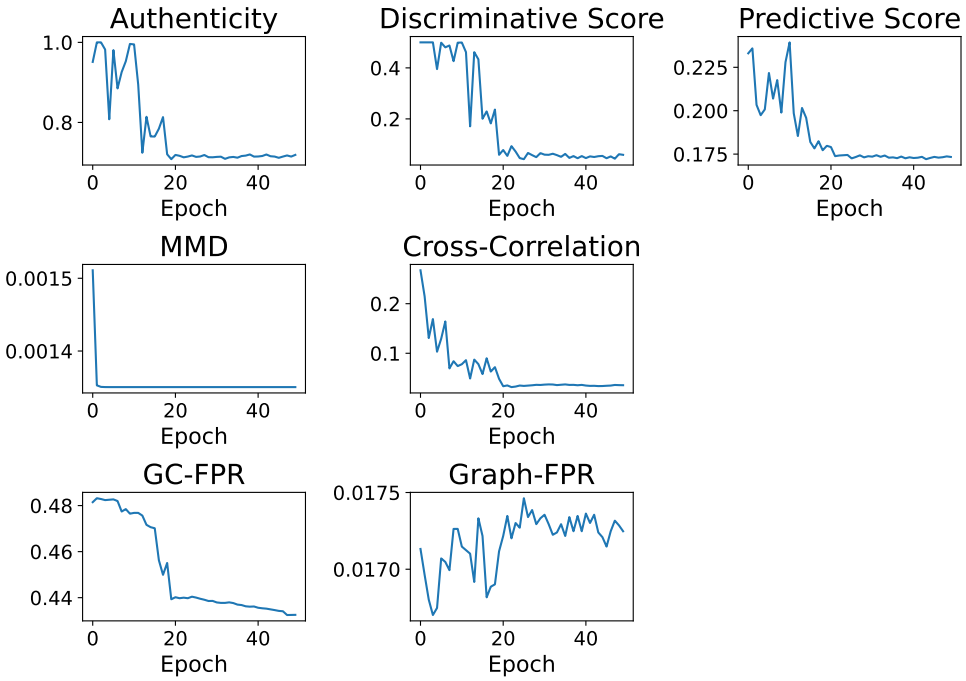


Figure 5.17: Evaluation metrics during training - Hénon dataset.

Graph Extraction Module

Robustness of Graph Extraction In this section, we report an additional experiment to highlight the robustness of the causal graphs to noise in the time-series. In particular, we trained `DiffCATS` in a setting where Gaussian noise $z \sim \mathcal{N}(0, \sigma^2)$ has been added to the training time-series. Table 5.10 reports the results of the Graph-FPR for $\sigma^2 = 0.005$ and $\sigma^2 = 0.01$ showing the robustness of the extraction to noise in the time-series.

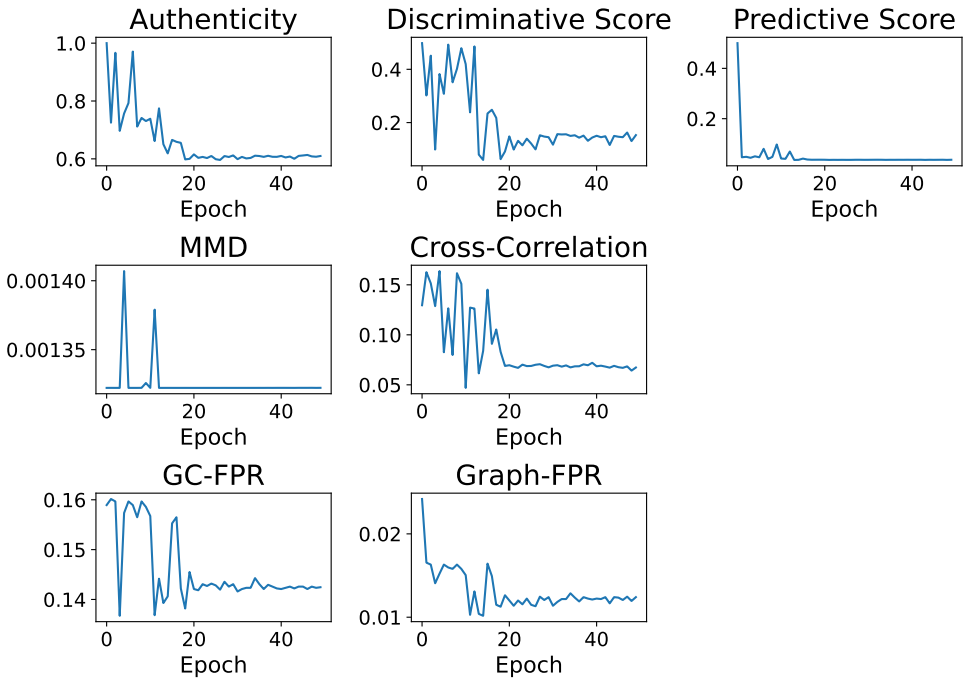


Figure 5.18: Evaluation metrics during training - Rivers dataset.

Statistical test Alternative approaches can be integrated to extract the causal graph based on the VAR coefficients. For example [182] suggests conducting a statistical test of the significance of the coefficient. We tested Dixon's Q Test to assess if there is at least one of the generated coefficients significantly higher than the others, resulting in a notable causal link. We evaluated the extracted causal graphs using this method with a p-value of 0.05. The Graph-FPR metric is 0.016 and 0.066 for the Rivers and Hénon datasets, respectively, which is very close to the score we obtained with the approach of Definition 5.4.1.

Impact of ρ and q The results in Figure 5.20 show that the error in introducing edges remains contained when varying $q \in \{0.80, 0.85, 0.90, 0.95\}$ and $\rho \in \{0.001, 0.005, 0.01, 0.02\}$. As the parameter ρ controlling the sparsity of the dataset increases, the Graph-FPR increases as well, meaning that the more relationships are kept in the dataset, the more it is the probability that they are

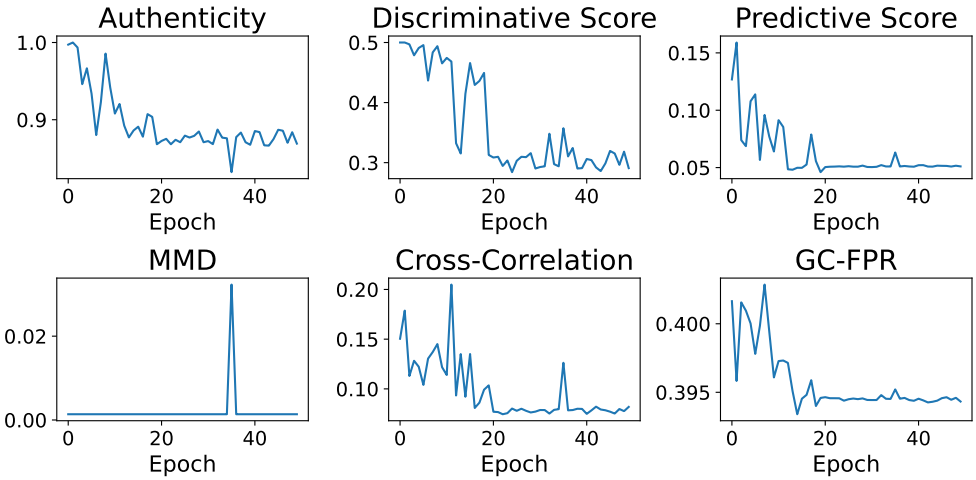
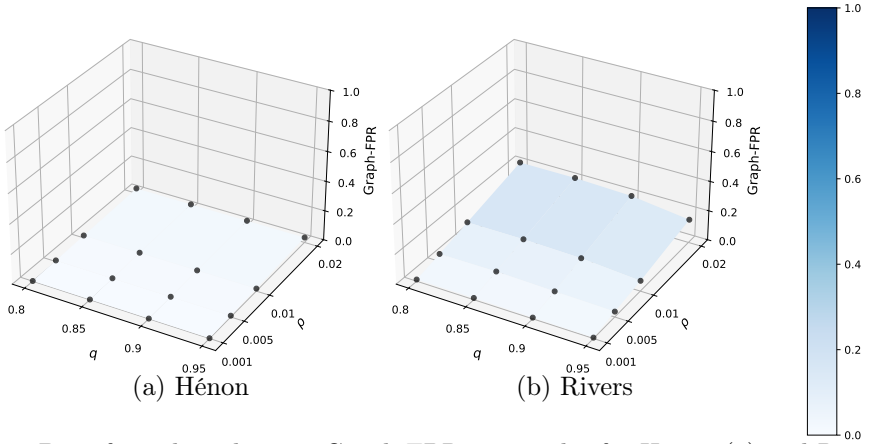


Figure 5.19: Evaluation metrics during training - AQI dataset.

Table 5.10: Graph-FPR with respect to noisy time-series.

Dataset	Noise variance (σ^2)		
	0	0.005	0.01
Hénon	0.04 ± 0.00	0.02 ± 0.00	0.02 ± 0.00
Rivers	0.07 ± 0.00	0.01 ± 0.00	0.01 ± 0.00

wrong. However, it can be observed that the parameter q is able to alleviate this by forcing each individual sample to keep only the coefficient of high magnitude.



(a) Hénon (b) Rivers

Figure 5.20: 3D surface plots showing Graph-FPR vs q and ρ for Hénon (a) and Rivers (b) datasets.

We would like to highlight that the parameters ρ and q are used to extract causal graphs for the dataset from the generated coefficients. Critically, this means that a potential user of the model does not need to retrain the model to find the best ρ and q values.

Execution Time

We ran the experiments on a machine equipped with Intel Core i9-10920X CPU @ 3.50GHz, NVIDIA GeForce RTX 2060 GPU, and 8×32 GB DDR4 RAM.

A training phase of our model required ∼1 hour for 60 epochs. We have trained 6 variants of our model on 3 different datasets, for a total of ∼18 training hours.

In more detail, Table 5.11 shows the inference time of the models isolating the generation of the time-series and the extraction of the graph. It turns out that even if CausalTime is faster than `DiffCATS` in generating the time-series, the graph extraction through DeepSHAP introduces an important overload, making it the slowest model.

Table 5.11: Inference time.

Dataset	Model				
	`DiffCATS`	`DiffCATS` with Graph	CausalTime	CausalTime with Graph	CR-VAE
Hénon	1481ms	1548ms	465ms	8790ms	194ms
Rivers	1425ms	1492ms	235ms	4248ms	148ms
AQI	1395ms	1535ms	1349ms	205s	442ms

Increasing the Polynomial Degree

To accommodate non-linear causal links, the polynomial degree employed in the VAR reconstruction can be increased to approximate non-linear relationships. With a certain degree of approximation, any (non-linear) continuous function can be represented by a polynomial (Stone-Weierstrass Theorem), trading off approximation error of the real function with computational time, since there will be more coefficients generated by the model. As an example, we performed an experiment on the Hénon dataset with a polynomial of degree set to 2 since this dataset comprises a quadratic relationship. Results shown in Table 5.12 show that the metrics to evaluate the time-series and the causal graph remain satisfactory even by raising the degree of the polynomial.

Table 5.12: Results on the Hénon dataset increasing the polynomial of the VAR.

Metric	**Polynomial Degree**	
	1	2
Discr.	0.032	0.021
Pred.	0.156	0.226
Graph-FPR	0.017	0.019

Causal Prediction Downstream Task involving Causal Graphs

In this section, we show the results of a downstream task involving the causal graphs. In particular, we considered the task of predicting the i-th feature of the multivariate time-series given its other dimensions and the corresponding causal graph.

Let $\boldsymbol{x}_{/i}$ be the $(d-1)$-variate time-series consisting of all the features but the i-th. The predictor model consists of a 2-layer LSTM to compute the embedding of the observed time-series and a linear layer L_{graph} to learn the embedding of the associated graph g. The two embeddings are summed and passed through a linear layer (L_{out}) to output $\hat{\boldsymbol{x}}_i$. Formally, $\hat{\boldsymbol{x}}_i = L_{out}(e_{ts} + e_g)$ where $e_{ts} = \text{LSTM}(\boldsymbol{x}_{/i})$ and $e_g = L_{graph}(g)$. The model is trained to minimize the ℓ_1-loss with respect to $\boldsymbol{x}_i$, i.e. the real i-th feature of $\boldsymbol{x}$.

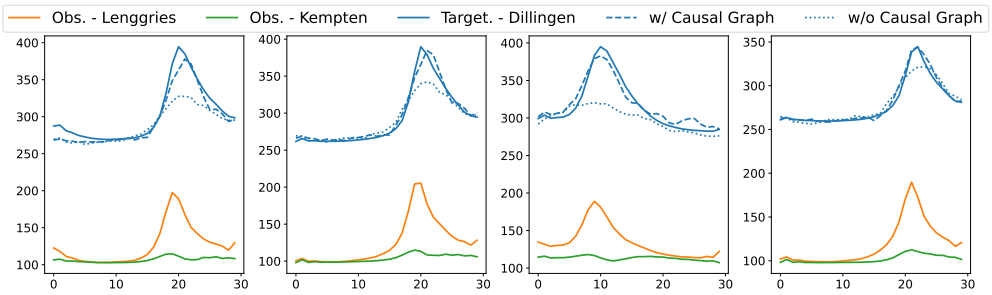
We compared the above model with the case in which only the $(d-1)$-variate time-series $\boldsymbol{x}_{/i}$ is exploited to forecast $\boldsymbol{x}_i$, i.e., the model does not see the causal graph. The models are evaluated in terms of Mean Absolute Error (MAE) on an independent validation set. Table 5.13 shows the quantitative results highlighting that the causal graphs are useful to improve the reconstruction ability of the predictor, and Figure 5.21 shows some examples.

Table 5.13: Downstream task - Causal Prediction (Mean Absolute Error).

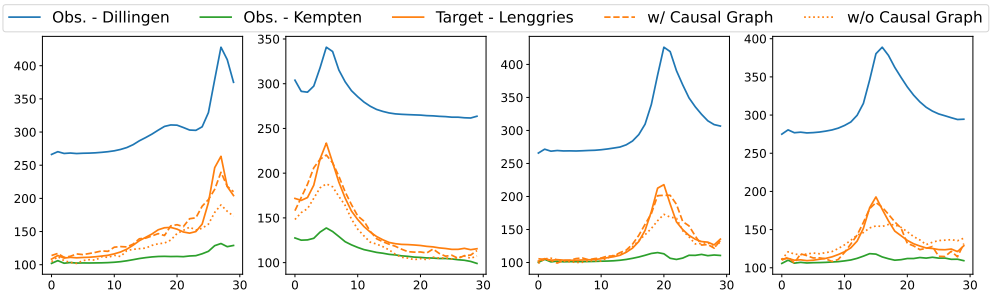
Dataset	**Predictor**	
	w/ Causal Graph	w/o Causal Graph
Hénon	$\mathbf{0.13 \pm 0.01}$	0.19 ± 0.01
Rivers	$\mathbf{0.01 \pm 0.00}$	0.02 ± 0.00

Causal Classification Downstream Task

Classifying time-series data based on their underlying causal dynamics represents a critical challenge in machine learning, especially because the causal structure of the system influences the observable behaviors.



(a) Target: Dillingen



(b) Target: Lenggries

Figure 5.21: Causal Reconstruction.

The broader significance of the causal classification task lies in its ability to enable meaningful grouping of observations by their causal nature. This can be applied to problems like emergency detection, where identifying unsafe system dynamics relies on understanding the causal mechanisms driving those dynamics, or customer segmentation, where groups are formed based on inferred causal responses to interventions.

The `DiffCATS` framework supports this challenge by generating synthetic time-series explicitly paired with their associated causal graphs, enabling models to capture and leverage causal dependencies for classification tasks. Therefore, in this downstream task, the causal graph serves as a label or defining feature for the grouping of samples based on their underlying system dynamics, rather than being directly integrated into the prediction process as in the **Causal Predictive** task.

As a demonstration, we used the Rivers dataset. We identified the 10 most frequent causal graphs within the dataset and treated them as class labels; we also added an additional class to capture residual or less common causal dynamics. Each class represents a distinct causal graph that reflects different patterns in river discharge behavior. We then trained a 2-layer LSTM network on synthetic time series generated by `DiffCATS`, with the goal of predicting the underlying causal class (i.e., label) from the observed time-series. When evaluated on real time-series samples, the classifier achieved an F1-score of 0.69. This result highlights the potential of using synthetic datasets to enhance classification tasks by providing explicit structural labels.

To conclude, we believe `DiffCATS` provides a robust support for classification tasks by generating datasets with coupled time-series and causal graphs, ensuring high fidelity in both the signals and their causality. This capability is particularly valuable in scenarios where certain classes are underrepresented or entirely absent. In fact, our experiments demonstrate that the synthetic data produced by `DiffCATS` serve as effective surrogates for training accurate classifiers.

5.8.4 Algorithms

We show the algorithm to reconstruct the whole time-series from the output of DEN_θ (i.e. the initial time-steps $\boldsymbol{x}_{start}$ and the set of coefficients $\boldsymbol{c}$) in Algorithm 1.

The sampling procedure of a synthetic couple $\langle \hat{\boldsymbol{x}}, \hat{g} \rangle$ is described in Algorithm 2.

Algorithm 1 Reconstruction of $\hat{\boldsymbol{x}}$ from $\boldsymbol{x}_{start}$ and $\boldsymbol{c}$.

Input: $\boldsymbol{x}_{start}$, $\boldsymbol{c}$
Output: $\hat{\boldsymbol{x}}$
▷ $\boldsymbol{x}_{start}$`.shape = [`d`,` τ_{max}`]`
▷ $\boldsymbol{c}$`.shape = [`d`,` $d \cdot \tau_{max}$`,` $L - \tau_{max}$`]`
$\hat{\boldsymbol{x}}_0 = \boldsymbol{x}_{start}$
for all i from 0 to $L - \tau_{max}$ **do**
  `sup` $\leftarrow \hat{\boldsymbol{x}}_0$`[:,` $-\tau_{max}$ `:].flatten()`
  $c \leftarrow \boldsymbol{c}$`[:,:,`$i$`]`
  $x \leftarrow$ `torch.einsum('a,ba->b', sup, c)`
  $\hat{\boldsymbol{x}}_0 \leftarrow$ `torch.cat([`$\hat{\boldsymbol{x}}_0$`,`x`.unsqueeze(`-1`)], dim=`-1`)`
end for
Return $\hat{\boldsymbol{x}}_0$

Algorithm 2 Sampling of $\langle \hat{\boldsymbol{x}}, \hat{g} \rangle$.

Input: Trained denoising network DEN_θ
Output: $\hat{\boldsymbol{x}}, \hat{g}$
$\boldsymbol{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for all t from T to 0 **do**
  $(\boldsymbol{x}_{\text{start}}, \boldsymbol{c}) \leftarrow \text{DEN}_\theta(\boldsymbol{x}_t, t)$
  $\hat{\boldsymbol{x}}_0 \leftarrow \text{RECONSTRUCT}(\boldsymbol{x}_{\text{start}}, \boldsymbol{c})$
  $\boldsymbol{x}_{t-1} \leftarrow \beta_t \cdot \frac{\sqrt{\hat{\alpha}_{t-1}}}{1-\hat{\alpha}_t} \cdot \hat{\boldsymbol{x}}_0 + \frac{(1-\hat{\alpha}_{t-1}) \cdot \sqrt{\alpha_t}}{1-\hat{\alpha}_t} \cdot \boldsymbol{x}_t$
  if $t > 0$ **then**
    $\boldsymbol{x}_{t-1} \leftarrow \boldsymbol{x}_{t-1} + \beta_t \cdot \frac{1-\hat{\alpha}_{t-1}}{1-\hat{\alpha}_t} \cdot \boldsymbol{\epsilon}$
  end if
end for
$\hat{g} \leftarrow \text{EXTRACTGRAPH}(\boldsymbol{c})$
Return $\hat{\boldsymbol{x}}_0, \hat{g}$

5.8.5 TSCD Algorithms Benchmark

Related work

Recent works have studied and tested causal discovery algorithms in several scenarios and domains. [161] provide a benchmark of 5 algorithms on both a synthetic and a real dataset, evaluating them using several binary classification metrics. [204] use their framework to generate numerical datasets and evaluate 5 causal discovery algorithms, with an in-depth performance analysis concerning their diverse assumptions and hyper-parameters selection. [166] employs the synthetic version of three real datasets to benchmark 13 representative state-of-the-art causal discovery algorithms. Finally, the very recent work of [205] incorporates LLMs to discover causal relationships from observational and interventional data. Their method is compared with 4 state-of-the-art baselines.

Details on the algorithms

To evaluate the different TSCD algorithms, we adapt/test them to our task using their source code available, whose repositories are listed below.

- GC: Granger Causality test implemented in the statsmodels[8] library.
- DYNOTEARS: `https://github.com/mckinsey/causalnex`
- NTS-NOTEARS: `https://github.com/xiangyu-sun-789/NTS-NOTEARS`
- PCMCI+: `https://github.com/jakobrunge/tigramite`
- Rhino: `https://github.com/microsoft/causica`
- CUTS / CUTS+: `https://github.com/jarrycyx/UNN`
- Neural-GC: `https://github.com/iancovert/Neural-GC`
- NGM: `https://github.com/alexisbellot/Graphical-modelling-continuous-time`
- LCCM: `https://github.com/edebrouwer/latentCCM`
- eSRU: `https://github.com/sakhanna/SRUforGCI`
- TCDF: `https://github.com/M-Nauta/TCDF`

The used hyper-parameters of the algorithms are reported in Table 6.11 (they are the same for all datasets).

Table 5.14: Hyper-parameters of the causal discovery algorithms.

Algorithm	Hyper-parameter	Value
GC	`maxlag`	2
DYNOTEARS	`p`	2
	`max_iter`	100
NTS-NOTEARS	`lags`	2
	`w_threshold`	0.3
	`h_tol`	$1e-60$
PCMCI+	τ_{max}	2
	$\texttt{PC}_\alpha$	0.01
Rhino	Noise Distribution	Gaussian
	`init_rho`	30
	`init_alpha`	0.2
CUTS	Input step	2
	λ	0.1
	τ	$0.1 \rightarrow 1$
CUTS+	Input step	2
	λ	0.01
	τ	$0.1 \rightarrow 1$
Neural-GC	Learning rate	0.05
	λ_{ridge}	0.01
	λ	$0.002 \rightarrow 0.02$
NGM	Steps	500
	Horizon	5
	GL_reg	0.1
LCCM	`hidden_size`	20
	Learning rate	0.01
eSRU	μ_1	1
	Learning rate	0.005
	Batch size	30
	Epochs	500
TCDF	τ	10
	Epochs	1000
	Learning rate	0.01

[8] https://www.statsmodels.org/stable/index.html

Benchmarking Causal Discovery Algorithms

Table 5.15 shows the results of our benchmark on a synthetic dataset where the causal graphs are extracted globally, following the procedure in Section 5.4.3.

Table 5.15: Other results of the benchmark of Causal Discovery Algorithms. Bold and underline are used to highlight the best and the second best result, respectively.

Method	AUROC			AUPRC		
	Hénon	Rivers	AQI	Hénon	Rivers	AQI
GC	0.55 ± 0.10	0.73 ± 0.16	0.50 ± 0.00	0.45 ± 0.11	0.54 ± 0.09	0.48 ± 0.08
DYNOTEARS	0.45 ± 0.11	0.52 ± 0.08	0.50 ± 0.00	0.52 ± 0.15	0.56 ± 0.08	$\underline{0.51 \pm 0.02}$
NTS-NOTEARS	0.64 ± 0.14	0.73 ± 0.15	0.50 ± 0.00	0.40 ± 0.13	0.55 ± 0.14	0.30 ± 0.23
PCMCI+	$\mathbf{0.84 \pm 0.08}$	$\underline{0.82 \pm 0.08}$	$\mathbf{0.68 \pm 0.00}$	0.54 ± 0.09	0.64 ± 0.08	0.50 ± 0.03
Rhino	0.50 ± 0.02	0.57 ± 0.12	0.50 ± 0.00	0.52 ± 0.01	0.65 ± 0.10	0.51 ± 0.03
CUTS	0.81 ± 0.10	$\mathbf{0.86 \pm 0.09}$	$\underline{0.68 \pm 0.01}$	$\underline{0.54 \pm 0.07}$	0.55 ± 0.08	$\underline{0.51 \pm 0.02}$
CUTS+	0.81 ± 0.09	0.75 ± 0.09	0.67 ± 0.01	0.53 ± 0.07	0.58 ± 0.08	$\underline{0.51 \pm 0.02}$
Neural-GC	0.67 ± 0.00	0.52 ± 0.07	0.50 ± 0.01	0.52 ± 0.01	0.53 ± 0.05	0.48 ± 0.10
NGM	$\underline{0.84 \pm 0.13}$	0.80 ± 0.13	0.50 ± 0.01	$\mathbf{0.63 \pm 0.16}$	$\mathbf{0.81 \pm 0.12}$	0.47 ± 0.13
LCCM	0.50 ± 0.00	0.50 ± 0.00	0.50 ± 0.00	0.51 ± 0.00	$\underline{0.78 \pm 0.00}$	0.21 ± 0.00
eSRU	0.50 ± 0.0	0.71 ± 0.10	0.50 ± 0.00	0.53 ± 0.01	0.76 ± 0.08	$\mathbf{0.53 \pm 0.01}$
TCDF	0.50 ± 0.0	0.50 ± 0.01	0.50 ± 0.00	0.50 ± 0.03	0.53 ± 0.09	0.45 ± 0.15

5.8.6 Limitations and Trade-off

Even if Table 5.1 reports the performance of strong generators (e.g., CSDI), our intended like-for-like comparison is against methods that can generate both (i) the multivariate time series and (ii) an associated causal graph, ideally with an explicit mechanism that promotes graph-sample consistency. `DiffCATS` is designed precisely for this paired generation setting: each synthetic sample is produced together with a corresponding causal graph, and the two are structurally coherent by construction. Instead, Base Diffusion and CSDI are optimized only for the fidelity of the synthetic time-series and are included as reference points, to show that `DiffCATS`' time-series quality is not far from high-performing time-series-only generators, while additionally providing causal graphs. This is a trade-off induced by requiring causal-graph generation and consistency, rather than as an across-the-board weakness.

Furthermore, regarding efficiency, Table 5.1 shows `DiffCATS` inference around 1.4-1.5 s/sample, which is slower than CR-VAE (hundreds of ms), consistent with diffusion sampling requiring multiple denoising steps. At the same time, `DiffCATS` is substantially faster than CausalTime, whose inference is dominated by post-hoc feature-importance (DeepSHAP), reaching seconds to minutes in (e.g., up to 205s on AQI).

We also evaluated the number of GLOPs required by the models to generate a synthetic sample. The results for `DiffCATS` include all the denoising steps (100 in our case) while the results for CausalTime include all the steps involved in their autoregressive generation.

Table 5.16: GFLOPs per sample.

	`DiffCATS`	CR-VAE	CausalTime
Hénon	14.885	0.004	6.613
Rivers	14.819	0.002	3.301
AirQuality	15.725	0.022	22.695

CR-VAE is consistently the cheapest (0.002-0.022 GFLOPs/sample) because it produces a sam-

ple in essentially a single lightweight forward pass of a variational autoencoder. `DiffCATS` requires 14.8-15.7 GFLOPs/sample across all three datasets due to its iterative diffusion sampling procedure. CausalTime's generation cost is competitive on small/moderate settings (3.301 GFLOPs on Rivers; 6.613 on Hénon), but increases markedly on the high-dimensional AirQuality setting (22.695 GFLOPs), consistent with the fact that its autoregressive generation must be repeated across the sequence and its per-step computation grows with dimensionality.

Importantly, the CausalTime GFLOPs reported here only count the forward computation needed to output a synthetic sample and do not include its post-processing for extracting the causal graph by interpreting the model with DeepSHAP. In practice, most of CausalTime's wall-clock inference overhead comes from this DeepSHAP-based attribution step, which requires many additional evaluations and can dominate runtime even when the raw GFLOPs/sample for generation looks moderate. By contrast, in our pipeline, even if `DiffCATS` requires more GFLOPs to generate a sample, the subsequent causal-graph extraction step is lightweight, so the end-to-end overhead is not driven by an expensive interpretability pass.

Finally, the GFLOPs trends across datasets also highlight scaling with the number of features (Rivers: 3, Hénon: 6, AirQuality: 36). `DiffCATS` stays roughly constant ($\sim$ 15 GFLOPs) across these experiments, while CR-VAE increases modestly (from 0.002 to 0.022 GFLOPs) and CausalTime grows substantially, especially at 36 features (22.695 GFLOPs). This suggests `DiffCATS`' generation compute is comparatively less sensitive to feature dimensionality in our implementation, whereas CausalTime's autoregressive generator becomes significantly more expensive as dimensionality increases.

Chapter 6

Robust Causal Discovery in Real-World Time Series with Power-Laws

Abstract

Exploring causal relationships in stochastic time series is a challenging yet crucial task with a vast range of applications, including finance, economics, neuroscience, and climate science. Many algorithms for Causal Discovery (CD) have been proposed; however, they often exhibit a high sensitivity to noise, resulting in spurious causal inferences in real data. In this chapter, we observe that the frequency spectra of many real-world time series follow a power-law distribution, notably due to an inherent self-organizing behavior. Leveraging this insight, we build a robust CD method based on the extraction of power-law spectral features that amplify genuine causal signals. Our method consistently outperforms state-of-the-art alternatives on both synthetic benchmarks and real-world datasets with known causal structures, demonstrating its robustness and practical relevance.

6.1 Introduction

Causal Discovery (CD) from stochastic time series aims to identify causal relationships among time-evolving variables purely from observational data. CD algorithms represent a domain-agnostic alternative to analytical modeling, which can be impractical in many scientific domains characterized by complex dynamics. The resulting causal model is typically represented with a *causal graph*, where nodes are variables, and directed edges reflect asymmetric causal dependencies between them. This methodology has been successfully employed on a vast range of fields, including climate science [206, 207], neuroscience [208, 209], finance [210, 211], and, more recently, generative AI [212–214]. Nevertheless, inferring causal relationships in time series is particularly challenging due to factors such as noise and non-stationarity (i.e., time-varying dynamics), which can obscure the underlying causal structure and reduce the robustness of causal discovery algorithms. Classical CD methods, most notably Granger Causality and its extensions, rely on restrictive assumptions such as noise stationarity and the existence of a single characteristic scale to define vector autoregressive (VAR) models appropriately. Unfortunately, these assumptions are frequently violated, as real-world systems are typically non-equilibrium, history-dependent, and often display scale-free temporal correlations and power-law frequency spectra [215]. In such contexts, conventional CD al-

gorithms can easily incur errors, detect spurious relationships, or fail to detect true interactions. To address these shortcomings, we introduce **PLaCy** (**P**ower-**La**w **C**ausal discover**y**), which is specifically designed to leverage the scale-free properties commonly observed in real-world time series. Instead of comparing variables at individual time points, it fits a power-law model to the frequency spectrum of each process and tracks the evolution of the fitted spectral exponents and amplitudes. In this way, **PLaCy** isolates structural causal changes that propagate from one variable to another by filtering out non-stationary and nonlinear external influences, bearing the absence of a characteristic scale. Classical Granger-type hypothesis tests are then applied to the trajectories of power-law spectral exponents and amplitudes, rather than to the raw signals, preserving the statistical power of established testing theory. By running extensive experiments on synthetic benchmarks with controlled nonlinear and non-stationary noise, or scale-free characteristics, as well as on two real-world data sets, we demonstrate that **PLaCy** outperforms state-of-the-art CD methods, particularly in regimes where the non-equilibrium, nonlinear, or scale-free properties of the time series are more pronounced.

The main contributions of this work are the following:

- We propose **PLaCy**, a novel framework that leverages spectral trends for robust causal discovery in time-series with power-law frequency distributions.
- We theoretically demonstrate that the frequency-domain transformation used in **PLaCy** preserves the underlying causal graph structure, guaranteeing results consistent with the time-domain graph.
- We empirically show that **PLaCy** provides more robust and accurate estimations, validated through extensive experiments on both synthetic and real-world datasets.

6.2 Preliminaries

6.2.1 Causal Discovery

Causal Discovery is the task of identifying the underlying structure of cause-and-effect relationships among the components of a multivariate system. Given a collection of observed multivariate time series, the goal is to infer a directed graph that encodes which variables influence others in a causal sense. Formally, given a time-series $\mathbf{x} \in \mathbb{R}^{L \times d}$, the task is to determine a directed graph $G = (V, E)$, where $V = \{1, 2, \ldots, d\}$ represents the variables of the system, and $E \subseteq V \times V$ is the set of directed edges. A directed edge (i, j) exists if and only if $\mathbf{x}_i$ is inferred to be a cause of $\mathbf{x}_j$. The goal of our approach is to derive the causal graph representing the causal relationships among the variables.

6.2.2 Granger Causality

Granger causality holds when past values of one time series provide statistically significant predictive information for another. In particular, we say that $\mathbf{x}_i$ *Granger-causes* $\mathbf{x}_j$ if the past values of $\mathbf{x}_i$ are useful to predict $\mathbf{x}_j$, given the past of all other time series. In time series data, Granger causality is typically studied using a multivariate vector autoregressive model (VAR) [216]:

$$\mathbf{x}(t) = \sum\nolimits_{\tau=1}^{T} \mathbf{A}_\tau \mathbf{x}(t-\tau) + \boldsymbol{\varepsilon}_t,$$

where $\mathbf{x}(t)$ is the multivariate time-series at time t, with each component defined as a linear combination of the past T values of all variables. Granger causal analysis involves fitting a VAR model and testing the statistical significance of the autoregressive coefficient matrices $\mathbf{A}_\tau$, typically using a Wald test [217]. This requires comparing two models: the unrestricted model, which includes lagged terms of both $\mathbf{x}_i$ and $\mathbf{x}_j$, and the restricted model, which excludes the lagged terms of $\mathbf{x}_i$ from the prediction of $\mathbf{x}_j$. The null hypothesis states that all coefficients related to the lagged $\mathbf{x}_i$ terms are zero. Failing to reject this null hypothesis implies that $\mathbf{x}_i$ does not Granger-cause $\mathbf{x}_j$. Notice that the Granger causality definition does not explicitly account for the time elapsed between cause and effect, since it jointly tests all specified lags together. Similarly, in our work, we focus on identifying the existence of causal relationships, regardless of the specific time lag between cause and effect.

6.2.3 Power-laws in the real-world

Over the past six decades, extensive empirical evidence has shown that power-law spectra of the form $S(f) \propto f^{-2\lambda}$, with $\lambda > 0$, are ubiquitous in real-world time series. Classic examples can be found in finance [218, 219], climate science [220, 221] or neuroscience [222–225]. Power-law spectra frequently arise in systems composed of many interacting units, such as traders in a market or nodes in communication networks, that *self-organize* into structured behavior without any external regulator/coordinator [226, 227]. Specifically, self-organizing systems often exhibit scale invariance [228], precisely due to the absence of any external coordinator enforcing a characteristic scale. A stochastic process $\{x(t)\}$, is *scale invariant* if $\forall a \in \mathbb{R}^+$, the rescaled process $\{x(at)\}$ is statistically equivalent to $\{a^H x(t)\}$, for some $H \in \mathbb{R}^+$. This property implies that any magnified fragment of a scale-invariant stochastic process looks identical to the original series and, for this reason, scale invariance is sometimes referred to as self-similarity and is very related to the geometric concept of a fractal [215]. It is also known that, under very loose assumptions, scale-invariant stochastic processes are also *scale-free*, meaning that they exhibit power-law correlations, and power-law distributed frequency spectra with exponent $\lambda = H - 1/2$ [229]. Given the ubiquity of power-law distributed frequency spectra in the real-world, this structural regularity can be leveraged to improve the extraction of causal signals from time series, reducing spurious temporal dependencies.

6.3 Proposed Methodology

A well-established approach in signal processing involves analyzing the frequency content of a signal via its spectral representation. To this end, we employ the Discrete Fourier Transform (DFT). Given a real-valued time series $x(t)$ of length L, the DFT is defined as:

$$\phi(k) = \sum_{t=0}^{L-1} x(t)\, e^{-i2\pi \frac{k}{L} t}, \quad k \in \{0, \ldots, L-1\}, \tag{6.1}$$

where $\phi(k) \in \mathbb{C}$ denotes the complex-valued coefficient corresponding to the k-th discrete frequency. The associated normalized frequency is given by $f_k = \frac{k}{L}$. The magnitude of each Fourier coefficient quantifies the contribution of the corresponding frequency component to the overall signal. We therefore denote the *spectral amplitude* as $A(f_k) = |\phi(k)|$.

As discussed previously, many natural and social systems exhibit long-range dependencies and scale-free behavior in their frequency content, often associated with self-organized phenomena. A defining

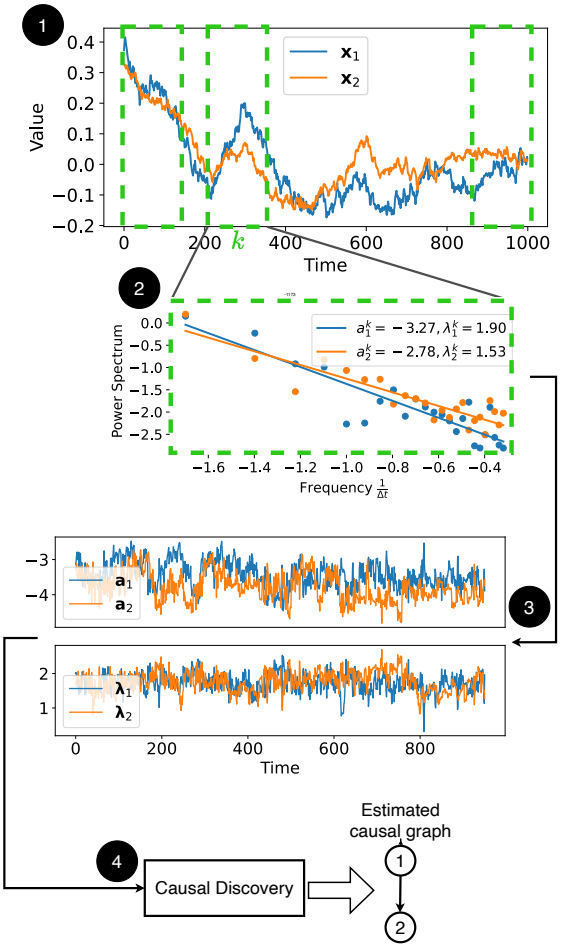


Figure 6.1: Schematic illustration of the proposed methodology. The original time series, here $\mathbf{x}_1$ and $\mathbf{x}_2$, are segmented into overlapping windows (**step ❶**). Then, for each window k, the amplitudes (a_1^k, a_2^k) and the exponents (λ_1^k, λ_2^k) of the power-law distributed spectra are computed (**step ❷**). These give rise to new, multi-dimensional, time series: ($\mathbf{a}_1$,$\boldsymbol{\lambda}_1$) for $\mathbf{x}_1$ and ($\mathbf{a}_2$,$\boldsymbol{\lambda}_2$) for $\mathbf{x}_2$ respectively (**step ❸**). Finally, multivariate Granger causality tests are performed on these new series, and the causal graph is constructed (**step ❹**).

characteristic of these systems is the power-law decay of their power spectral amplitude, typically modeled as $A(f) = \mathrm{e}^a \cdot f^{-\lambda}$, where e^a is a scaling constant and $\lambda > 0$ is the *spectral exponent*. The exponent λ is tightly linked to structural features of the process, such as its autocorrelation. Importantly, the spectral parameters a and λ may vary over time due to exogenous perturbations or endogenous interactions. These variations provide an opportunity to study causal structures through their temporal dynamics. Instead of analyzing the raw time series directly, we propose to monitor the evolution of $(\mathbf{a}, \boldsymbol{\lambda})$ as informative summaries of the underlying processes. To achieve this, we segment each time series into overlapping windows (**step ❶** in Figure 6.1) and compute the local spectral parameters within each window. This is done by estimating the slope and intercept of the spectrum in log-log space:

$$\log A(f) = a - \lambda \log f. \tag{6.2}$$

The linear form permits efficient estimation via ordinary least squares, yielding one value of a and λ per window (**step ❷** in Figure 6.1). Repeating this procedure across the entire series results in two new time series per original signal: $\mathbf{a}$ and $\boldsymbol{\lambda}$ (**step ❸** in Figure 6.1). To capture the spectral behavior

Algorithm 3 PLaCy

Require: Time series $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_d)$ of length L; stride s; window size l.
Ensure: Causal Graph $\mathcal{G}$.

```
Divide each x_i into ⌊(L−l)/s⌋ + 1 sliding windows, namely w_i^k, of size l with stride s.
for each i ∈ {1, ..., d} do
    for each k ∈ {0, ..., ⌊(L−l)/s⌋} do
        Apply the DFT (Equation (6.1)) to w_i^k to get φ_i^k.
        Obtain (a_i^k, λ_i^k) by using the fit in Equation (6.2) on φ_i^k.
    end for
    Concatenate (a_i^k, λ_i^k) over k to obtain time series (a_i, λ_i).
end for
for each i, j ∈ {1, ..., d} such that i ≠ j do
    G_{i,j} ← Granger Causality test Section 6.2.2 with (a_i, λ_i) as causing series and λ_j as caused series.
end for
return G.
```

exhibited within each analysis window, we apply overlapping windows. This design is critical to preserve the detection of short-lived or temporally localized causal effects. To maximize sensitivity, the stride between consecutive windows can be fixed at 1, so that each new window shifts by a single time step. This dense sampling guarantees that even subtle or rapid changes in the spectral parameters (a, λ) are preserved in the constructed feature time series. The window length is selected adaptively to balance two competing requirements: it must be short enough to capture temporal variations in the spectral parameters, yet long enough to ensure a reliable estimation of the power-law behavior. To meet this trade-off, we evaluate the p-value of a Wald test on the linear fit in log-log space for each candidate window size, and select the shortest window for which the fit achieves a statistical significance threshold of $p = 0.05$. Further details of this procedure are provided in the Appendix. Once the feature series $(\mathbf{a}, \boldsymbol{\lambda})$ are built for a couple of original signals, we perform multivariate Granger causality tests, as described in Section 6.2.2 (**step ❹** in Figure 6.1). Since the causal information is primarily encoded in the λ parameter, the Granger test is applied to assess whether $(\boldsymbol{\lambda}_i, \mathbf{a}_i)$ of the candidate causing series $\mathbf{x}_i$ provide statistically significant information about the dynamics of $\boldsymbol{\lambda}_j$ in the target series $\mathbf{x}_j$ (see Appendix for further details). In the end, a causal edge is retained in the resulting graph if the corresponding p-value falls below the fixed threshold of 0.05. This procedure is repeated across all variable pairs to reconstruct the full causal graph, as detailed in Algorithm 3.

6.3.1 Invariance of the Causal Graph under Spectral Feature Mapping

Unlike conventional Granger methods, which analyze lagged relationships in the original signal space, our approach infers causality from the coordinated evolution of spectral properties.

Moreover, the spectral fitting acts as a natural denoising step, improving robustness to non-Gaussian fluctuations and high-frequency noise. In the following Theorem 6.3.1, we discuss the correctness of this approach by showing that the causal graph of a stochastic process is invariant under the spectral transformation applied in Algorithm 3, which preserves the causal semantics of the original process.

Theorem 6.3.1 (Preservation of Linear Causal Graphs under Spectral Transformations)**.** *Let* $\mathbf{x}$ *be a multivariate time series generated by a linear structural causal process with ground-truth causal graph* $\mathcal{G}^*$*. Let* $\mathcal{T}$ *be the spectral transformation in Algorithm 3, which, for each component* $\mathbf{x}_i$*, extracts a sequence of time-evolving features* $(\mathbf{a}_i, \boldsymbol{\lambda}_i)$*. Assume that* $\mathbf{x}$ *has power-law spectra with a common frequency dependence across all frequencies* f*. Then, under standard identifiability conditions for VAR-type causal discovery, causal discovery performed on the feature sequence* $(\mathbf{a}, \boldsymbol{\lambda})$ *recovers the same causal graph* $\mathcal{G}^*$ *as time-domain analysis.*

Proof sketch. The proof proceeds in two steps. First, we show that, under the stated assumptions, the feature series $(\mathbf{a}, \boldsymbol{\lambda})$ satisfies the classical conditions required for valid inference in vector autoregressive (VAR) models, as established in [216]. Second, we show that, for amplitude-expressive causal mechanisms, the mapping $\mathcal{T}$ preserves the causal dependencies encoded in the ground-truth graph $\mathcal{G}^*$. Combining these two results, we conclude that applying Granger causality analysis to the transformed sequence $(\mathbf{a}, \boldsymbol{\lambda})$ recovers the same causal graph $\mathcal{G}^*$ as time-domain analysis. The complete formal proof of Theorem 6.3.1 is provided in the Appendix. □

6.4 Related Work

Causal discovery from observational time series has been extensively studied [230, 231], leading to a broad spectrum of methods, from classical statistical tests to more advanced machine learning and spectral approaches. We review many of them here, highlighting in **bold** the ones we compare against in this work. Moving beyond **Granger Causality** described in Section 6.2.2, constraint-based methods have been developed and adapted for the temporal domain. Notably, the PC (Peter-Clark) algorithm [232] (and its extension, FCI [233]) serves as the foundation for several approaches. These utilize conditional independence tests to infer graphical causal structures while accounting for temporal ordering. Building on these, the **PCMCI** algorithm [23] enhances causal discovery in time series by combining the PC methodology with the Momentary Conditional Independence (MCI) test, which rigorously controls for autocorrelation and indirect associations. This algorithm was recently extended with **PCMCI$_\Omega$** [234] to the case of semi-stationary structural causal models. Optimization-based and deep-learning approaches have further broadened the field. **DYNOTEARS** [163] casts causal discovery as a continuous optimization problem subject to acyclicity constraints, preserving efficiency in handling high-dimensional data. **Rhino** [235] represents an innovative deep learning-based approach where the CD task is addressed in scenarios where the noise distributions may depend on historical information. On the one hand, the use of neural networks allows Rhino to handle history-dependent and non-stationary noise; on the other hand, this advantage comes at the cost of significant computational overhead during training. Other techniques, such as Convergent Cross Mapping (CCM) [236], although robust in theory, exhibit significant performance degradation in noisy settings. Recent efforts have been made to enhance the noise-resilience of existing algorithms. **CCM-Filtering** (**CCM**) [237] improves the performance of CCM by simply pre-processing the time series with an averaging filter. **RCV-VarLiNGAM** (**RCV**) [238] integrates the K-fold cross-validation technique with the VarLiNGAM method [182], addressing the challenges of a lack of noise robustness encountered in the standard method.

Previous works have also studied causal discovery in the frequency domain. **Geweke**'s seminal work [239] extended Granger causality by decomposing directional influence across frequencies,

revealing dynamic interdependencies often hidden in time-domain analyses. Subsequent studies applied this methodology to diverse domains, including economic cycles and oscillatory phenomena [240], network and finance [241], commodity markets [242], and market volatility [243]. Among frequency-domain approaches, the **Directed Transfer Function (DTF)** [244, 245] quantifies directional interactions as a function of frequency through the multivariate VAR transfer function. A **nonparametric spectral Granger (GewekeNP)** variant computes Geweke's frequency-domain causality measure directly from the empirically estimated power spectra, instead of the canonical VAR-based derivation [246, 247]. We also implement the **BCGeweke** system used by Wang et al. [241], which combines Geweke's frequency decomposition with the Breitung-Candelon [240] band-constrained statistical testing framework to detect directional dependencies within specific frequency bands.

Despite these advancements, many current methods remain vulnerable to noise, spurious dependencies, and deviations from Gaussianity. Motivated by these limitations, this work proposes a novel frequency-domain strategy designed explicitly for robust causal discovery within stochastic power-law processes. Our approach inherently leverages the frequency-dependent structure of power-law processes, enhancing resilience to nonlinear, complex, noisy signals.

6.5 Experiments

Unless otherwise stated, our method employs a sliding window of size $l = 50$, selected through the p-value procedure outlined in Section 6.3, a stride $s = 1$ (refer to Algorithm 3), and 10 lagged values. Quantitative results are averaged over 100 runs for all methods, except for Rhino, which is averaged over only 10 random seeds due to the high computational cost of neural network training.

We consider both synthetic and real-world datasets to rigorously evaluate our approach in terms of its robustness to noise and spurious associations. In particular, we create four synthetic datasets with increasing complexity, and we consider two real-world benchmark datasets with known causal graphs. Some prior datasets were excluded due to the absence of ground-truth causal graph or insufficient time series length for spectral estimation (see Section 6.6).

Metrics We evaluate the performance of the algorithms based on their ability to accurately identify causal relationships among variables. By comparing the edges of the predicted causal graph with those of the ground-truth graph, we compute the following metrics: *F1-score* (F1) that measures the performance of algorithms in correctly identifying causal relationships; and *True Negative Rate* (TNR) to evaluate the robustness of the algorithms to noise and spurious associations, by measuring its ability to correctly identify the absence of causal links. The latter metric is particularly insightful, as it evaluates a method's ability to exclude erroneous relationships in the generated causal graphs, an aspect not directly captured by the F1-score.

6.5.1 Synthetic Scenarios

Data Generation To generate complex benchmark datasets, we use the well-known Ornstein-Uhlenbeck (OU) processes, originally introduced in statistical physics to describe the velocity of a Brownian particle under friction [248]. These stochastic processes are widely used to model systems exhibiting mean-reverting behavior and power-law spectral characteristics. For example, they have been applied to capture the complexity of financial data [249]. In the frequency domain,

OU processes exhibit a characteristic power-law decay, reflecting the behavior of various natural systems that our approach seeks to address. We simulate the baseline dynamics of each time series using a generalized OU process, defined as follows:

$$x(t+\Delta t) = x(t) + \frac{\Delta t}{\tau_c}\big(\mu - x(t)\big) + \Big(\sigma_b \epsilon_b(t) + \sigma_g^a \epsilon_g^a(t) + \sigma_g^m \epsilon_g^m(t) \cdot x(t)\Big)\sqrt{\Delta t} \tag{6.3}$$

where $\Delta t = 0.01$ is the time step, $\tau_c = 0.5$ denotes the timescale of mean reversion, and $\mu = 1$ is the long-term mean. $\forall t$, $\epsilon_b(t)$, $\epsilon_g^a(t)$, and $\epsilon_g^m(t)$ represent the noise terms modeled as independent stochastic variables: the first is Brownian noise, while the latter two are standard Gaussian white noise processes. Specifically, $\epsilon_g^a(t)$ is an additive noise component, whereas $\epsilon_g^m(t)$ acts as a multiplicative noise term that induces non-stationarity as its impact scales with the process value. The parameters σ_b, σ_g^a, and σ_g^m represent different sources of noise volatility. This formulation enables the system to capture both additive and multiplicative stochastic effects, which are common in real-world dynamic systems. To represent causal relationships between different time series, we construct a Directed Acyclic Graph (DAG) $\mathcal{G}$, represented by an upper triangular adjacency matrix ($M \in \{0,1\}^{N\times N}$), which enforces a unidirectional flow of causality and prevents cycles. Each entry in the matrix is randomly set to 1 with a probability of 0.3, indicating a causal influence from one series to another.

Finally, causal dependencies are introduced by applying the generated ground truth causal matrix to the time series. In particular, if $M_{i,j} = 1$, indicating that series i influences series j, then

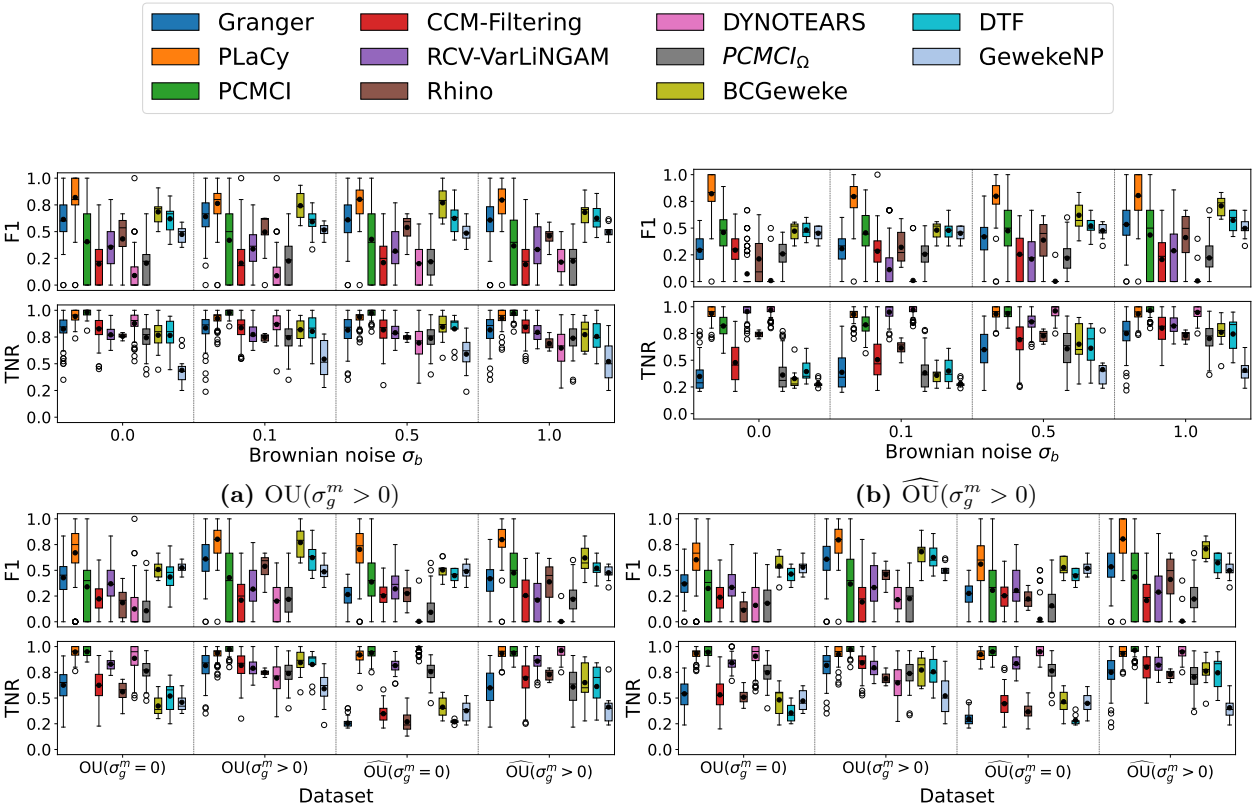
$$\forall t,\ x_j(t) \leftarrow x_j(t) + C \cdot x_i(t-\tau),$$

where C represents the *causal strength*, and $\tau = 5$ is the number of time-steps between the cause and the effect. This ensures that the current value of the influenced series incorporates a lagged contribution from the influencing series. Finally, we scale all time series to their original range to remove any unintended amplifications or distortions during the causal injection.

Datasets Following the generation process described in the previous paragraph, we define four representative scenarios to evaluate the robustness of our method under different dynamic conditions according to Equation (6.3): **(1)** OU($\sigma_g^m = 0$) represents an OU process with no noise component proportional to the process itself; **(2)** OU($\sigma_g^m > 0$) includes a Gaussian noise term that is proportional to the current value of the process. Both processes are initialized in equilibrium conditions, with the first time step $t = 0$ set to 1. Non-equilibrium and phase-transitioning systems display complex behaviors, as observed in several domains (e.g., in financial markets [250]). Therefore, we introduce a transition phase by initializing the process at 100. By extending the previous two scenarios, we obtain **(3)** $\widehat{\text{OU}}(\sigma_g^m = 0)$ and **(4)** $\widehat{\text{OU}}(\sigma_g^m > 0)$. For each dataset, we generate different scenarios with $N = 5$ and $N = 10$ variables (i.e., time series), each of length $L = 5000$ time-steps.

Results Figure 6.2 reports the F1-score and the TNR obtained on synthetic datasets with $N = 5$ variables, causal strength $C = 0.5$ and $\sigma_g^a = 0$. A complete overview of the results can be found in Table 6.1 for the case of $N = 5$ and $\sigma_g^a = 1$, averaging over $\sigma_b \in \{0, 0.1, 0.5, 1\}$. Additional results, including scenarios with $N = 10$ and $\sigma_g^a = 0.5$, can be found in the Appendix. Results obtained with the basic **Geweke** method have been excluded from Figure 6.2, Tables 6.2 and 6.5 due to its consistently poor performance. In particular, the *True Negative Rate* (TNR) across the

same experimental settings is uniformly zero, as the method fails to reject any spurious connections.



(a) OU($\sigma_g^m > 0$) **(b)** $\widehat{\text{OU}}(\sigma_g^m > 0)$

(c) $\sigma_b = 0.5$. **(d)** $\sigma_b = 1.0$.

Figure 6.2: Results on synthetic datasets, with $N = 5$, $C = 0.5$, $\sigma_g^a = 1.0$.

Key takeaway: Our method consistently outperforms existing approaches across all scenarios, demonstrating strong robustness to both structural variations and noise. In particular, Figures 6.2a and 6.2b show that our approach, **PLaCy**, achieves overall the best performance in terms of F1-score for all the noise settings σ_b, for OU($\sigma_g^m > 0$) and $\widehat{\text{OU}}(\sigma_g^m > 0)$. In fact, the presence of multiplicative Gaussian noise introduces non-stationarity, to which other methods are highly sensitive. In contrast, analyzing causal dynamics via spectral parameters enables **PLaCy** to filter variability from multiplicative noise, improving robustness. Our approach outperforms the other methods even in the absence of multiplicative noise, for $\sigma_b = 0.5$ and $\sigma_b = 1$ (Figures 6.2c and 6.2d): while other methods heavily rely on stationary assumptions, **PLaCy** can capture structural shifts in spectral parameters, identifying genuine causal relationships without confusing transient behaviors for structural causal patterns.

In all the experiments in Figure 6.2, the TNR of **PLaCy** is always very high, sometimes moderately outperformed by **PCMCI**. This is due to the fact that **PCMCI** is designed to explicitly control false positives, with a conservative approach to edge selection (MCI phase). **PLaCy**, however, shows a more permissive behavior in edge inclusion. This causes occasional acceptance of marginal causal association, but it is also the key to capturing genuine causal relations, as confirmed by the higher F1-score. Figure 6.2b considers the impact of both multiplicative noise and non-equilibrium initialization, two aspects that constitute a significant challenge for traditional CD approaches. In this setting, **PLaCy** significantly outperforms the other methods, with the highest F1-score and TNR close to 1, thanks to its capability to distinguish meaningful causal perturbations in spectral trends from spurious correlations possibly due to transient non-stationary dynamics.

Table 6.1: F1 Score - $N = 5$, $\sigma_g^a = 1.0$

	C	Granger	PLACy	PCMCI	RCV-VarLiNGAM	DYNOTEARS	PCMCI$_\Omega$	BCGeweke	DTF	GewekeNP
OU($\sigma_g^m = 0$)	0.2	0.58±0.26	0.14±0.22	0.15±0.24	0.40±0.19	0.17±0.18	0.07±0.13	**0.61**±0.18	0.35±0.18	0.59±0.06
	0.5	0.56±0.25	**0.73**±0.24	0.20±0.27	0.39±0.19	0.08±0.18	0.11±0.14	0.58±0.15	0.52±0.18	0.52±0.07
	1.0	0.53±0.25	**0.77**±0.17	0.28±0.29	0.32±0.21	0.08±0.17	0.13±0.15	0.56±0.15	0.54±0.13	0.52±0.08
$\widetilde{\text{OU}}$($\sigma_g^m = 0$)	0.2	0.47±0.28	0.24±0.25	0.16±0.25	0.41±0.19	0.02±0.10	0.08±0.13	**0.58**±0.18	0.44±0.07	0.49±0.10
	0.5	0.45±0.27	**0.69**±0.22	0.21±0.26	0.37±0.20	0.01±0.06	0.10±0.14	0.55±0.16	0.44±0.07	0.46±0.07
	1.0	0.45±0.26	**0.70**±0.17	0.29±0.30	0.32±0.19	0.01±0.06	0.13±0.15	0.54±0.16	0.43±0.09	0.46±0.09
OU($\sigma_g^m > 0$)	0.2	0.63±0.21	0.72±0.24	0.23±0.29	0.39±0.18	0.22±0.14	0.16±0.16	**0.79**±0.11	0.50±0.16	0.51±0.08
	0.5	0.62±0.20	**0.80**±0.17	0.40±0.33	0.34±0.20	0.15±0.17	0.22±0.16	0.74±0.11	0.64±0.14	0.50±0.07
	1.0	0.57±0.18	**0.77**±0.18	0.60±0.30	0.28±0.21	0.17±0.18	0.23±0.18	0.69±0.14	0.59±0.12	0.51±0.08
$\widetilde{\text{OU}}$($\sigma_g^m > 0$)	0.2	0.39±0.20	**0.75**±0.21	0.32±0.27	0.22±0.23	0.02±0.10	0.22±0.14	0.61±0.17	0.46±0.13	0.46±0.08
	0.5	0.39±0.18	**0.80**±0.17	0.46±0.26	0.17±0.21	0.01±0.05	0.24±0.14	0.57±0.13	0.52±0.11	0.46±0.08
	1.0	0.39±0.18	**0.78**±0.17	0.55±0.25	0.15±0.20	0.01±0.07	0.26±0.14	0.54±0.13	0.52±0.10	0.46±0.08

PCMCI's lack of robustness in the non-Gaussian and non-stationary noise scenario is evident from its low F1-score. The same limitations apply to its generalized version, **PCMCI$_\Omega$**, designed to handle semi-stationary causal relationships.

Regarding the other methods, although **Granger causality** is not designed for nonstationary data or non-Gaussian noise, it often retains moderate effectiveness in inferring causal relationships in such settings. Even if **CCM-Filtering** is designed to improve the robustness of the original CCM by reducing high-frequency noise and preserving lower-frequency signals, this solution does not generalize to stochastic processes, where delay embeddings fail to capture a deterministic manifold. **RCV-VarLiNGAM** does not show good results in a non-stationary noise environment, as expected by its assumption of stationarity. Although designed to uncover causal relationships via exploitation of non-Gaussian noise, the improvement with respect to other models remains limited. **DYNOTEARS** is designed to handle additive noise, but in our experiments, it struggles under multiplicative perturbations and strong non-stationarity. As a result, the method tends to miss true causal links, despite maintaining a high TNR. **Rhino** achieves moderate performance as proved by the experiments in Figures 6.2a and 6.2b, validating the authors' claim that the method is robust to increasing non-Gaussian noise. However, its computational complexity does not justify the lower performance compared to our approach, which also preserves sample efficiency. Finally, the three frequency-domain algorithms **BCGeweke**, **DTF**, and **GewekeNP**, achieve good F1 scores throughout the experimental campaign, suggesting that exploiting spectral properties provides a valuable avenue for detecting causal dependencies. Nevertheless, their high F1 score comes at the expense of a low TNR. Remarkably, PLaCy not only surpasses these methods in terms of F1 accuracy but also mitigates their TNR weakness, achieving a more balanced performance.

6.5.2 Real Data Scenarios

As real-world scenarios, we consider two datasets with known causal graphs:

• **Rivers** dataset[1] [179] contains $N = 6$ time series from three hydrological stations located in southern Germany, namely Dillingen, Kempten, and Lenggries. At each location, river level and

[1]Bavarian Environmental Agency data provider: `https://www.gkd.bayern.de`.

Table 6.2: Performance on real-world datasets.

	Rivers		AirQuality	
Algorithm	**F1**	**TNR**	**F1**	**TNR**
Granger	0.47±0.07	0.64±0.09	0.41±0.02	0.22±0.05
PLaCy	**0.51**±0.10	**0.75**±0.13	**0.45**±0.04	0.66±0.07
PCMCI	0.47±0.07	0.74±0.05	0.25±0.03	**0.95**±0.02
CCM	0.28±0.01	0.19±0.00	0.40±0.00	0.04±0.00
RCV	0.16±0.12	0.51±0.12	—	—
Rhino	0.29±0.03	0.35±0.05	0.44±0.01	0.23±0.04
DYNO.	0.12±0.07	0.53±0.06	0.37±0.08	0.92±0.03
PCMCI$_\Omega$	0.10±0.09	0.57±0.05	0.36±0.04	0.69±0.07
BCGeweke	0.41±0.06	0.61±0.08	0.39±0.02	0.16±0.07
DTF	0.36±0.05	0.42±0.12	0.43±0.02	0.36±0.09
GewekeNP	0.26±0.05	0.31±0.06	0.40±0.02	0.13±0.07

local precipitation are recorded over several thousand time steps. Since the Iller feeds into the Danube, increases in its discharge are expected to impact the Danube with a one-day lag.

• **AirQuality** (**AQI**) dataset[2] [166] contains hourly PM2.5 pollution measurements collected over one year from $N = 36$ monitoring stations across various Chinese cities. The ground-truth causal graph is derived from a causal matrix based on pairwise sensor distances. We extract subsamples of length 500 from the original time series.

Results Table 6.2 reports the results for both datasets. Our method achieves overall competitive or the best performance in terms of F1-score and TNR for both datasets.

Key takeaway: The Rivers dataset poses an additional challenge due to its heterogeneous dynamics and the presence of exogenous factors such as seasonal precipitation. Because precipitation series lack clear power-law behavior, this dataset highlights PLaCy's ability to generalize beyond its core assumptions. Indeed, our method achieves robust performance and surpasses the other methods in detecting the causal effects of precipitation and rivers' flow in both F1 and TNR. On the other hand, the AirQuality dataset includes missing values, which were filled using linear interpolation. Missing data, common in environmental datasets, challenges most CD methods. **PLaCy** maintains competitive performance despite these imperfections, highlighting its robustness to real-world data issues thanks to the advantages of performing causal discovery in the frequency domain, which is inherently more resilient to missing data and noise. **RCV-VarLiNGAM**, failed to converge. In our tests, we observe that this behavior is due to the impossibility of running the Cholesky decomposition on the residual covariance matrix, which results in being non-positive definite as a consequence of missing data.

Despite the original CCM promises to be robust in correctly excluding causal links between non-coupled variables in the presence of external forcing, the performance of **CCM-Filtering**, and in particular the achieved TNR, degrades on more complex data, influenced by unobserved exogenous factors. The same considerations mentioned about the TNR attained by **PCMCI** on the synthetic datasets (section 6.5.1) hold on AirQuality, where the high TNR is achieved at the expense of the lowest F1.

Interestingly, on the AirQuality dataset, **BCGeweke** exhibits a comparatively low TNR. This limitation likely stems from the algorithm's design: it analyzes distinct sub-bands of the spectral domain and infers causality whenever at least one sub-band yields a positive result in the Breitung-

[2]Microsoft data provider: `https://www.microsoft.com/en-us/research/project/urban-computing`.

Candelon (BC) causal test. In this case study, the method likely produced false positives within one of these sub-bands, leading to an overall degradation in TNR.

6.6 Limitations

Despite the strong performance demonstrated in the experimental campaign described in Section A.2.5, **PLaCy** presents some limitations that should be acknowledged. First, it is not able to assess causal relationships in the presence of slowly varying spectra. Some strategies may improve the algorithm's performance in such scenarios, such as increasing the number of lags considered in the causal analysis or extending the length of the analysis window (see Appendix). Nevertheless, in these cases, time-domain analyses like Granger causality may be more appropriate. While this represents a limitation, a reasonable choice of the causal analysis method can be guided by a preliminary spectral inspection. Second, the method is not well-suited for very short time series, as it relies on local spectral estimation, which requires a minimum sequence length to produce stable features.

6.7 Conclusions

This study introduces **PLaCy**, a novel algorithm for causal discovery in stochastic time series, leveraging power spectrum analysis to identify underlying causal structures. An extensive experimental campaign on both synthetic and real-world datasets reveals the effectiveness of this methodology in comparison to state-of-the-art methods. Our findings underscore the advantages of frequency-domain analysis for causal discovery, highlighting its potential to avoid detecting spurious associations as causal relationships while maintaining high F1 scores. Future work will focus on extending the idea of exploiting the power spectrum to enhance non-VAR causal discovery methods. A deeper investigation should be conducted on the summary statistical parameters of the spectral density. Furthermore, an additional study must be conducted to address the possible presence of latent confounders.

6.8 Appendix

6.8.1 Theoretical Analysis

The proof of Theorem 6.3.1 is articulated in two main steps:

(A) We first show that the feature sequence $(\mathbf{a}, \boldsymbol{\lambda})$ satisfies the classical assumptions required for valid inference using vector autoregressive (VAR) models, as established in [216].

(B) We then demonstrate that the transformation $\mathcal{T}$, described in Algorithm 3, preserves the structure of the ground-truth causal graph $\mathcal{G}^*$, meaning that no spurious edges are introduced and no genuine dependencies are lost.

By combining these two results, we conclude that applying Granger causality analysis to the transformed sequence $(\mathbf{a}, \boldsymbol{\lambda})$ enables consistent recovery of the original causal graph $\mathcal{G}^*$.

In order to validate step **(A)**, we decompose it into the following supporting claims:

(A1) Weak Stationarity and Noise Assumptions: The sequences $(\mathbf{a}, \boldsymbol{\lambda})$ are approximately weakly stationary and with an asymptotically Gaussian noise.

(A2) Preservation of Linear Dependence under Spectral Transformation: The transformation $\mathcal{T}$, described in Algorithm 3, preserves linear relationships between corresponding time series. That is, if two time series are linearly dependent (e.g., one is a scalar multiple of the other), then their corresponding spectral features remain linearly dependent. Conversely, if the time series are not collinear in the time domain, their spectral representations will also be linearly independent.

Step **(B)** relies on just one structural condition:

(B1) Spectral Causality Preservation: Causal relationships in the time domain that manifest through changes in spectral amplitude induce dependencies among the corresponding spectral features.

Step **(A1)** establishes the key requirements needed to apply a VAR analysis for studying Granger causality.

Theorem 6.8.1 discusses the assumptions of weak stationarity and Gaussianity of the noise. It shows that the procedure $\mathcal{T}$, described in Algorithm 3, transforms any colored noise affecting $\mathbf{x}$ into an asymptotically Gaussian noise on $(\mathbf{a}, \boldsymbol{\lambda})$. This proves step **(A1)**.

Theorem 6.8.2 addresses the component **(A2)**, proving the preservation of linear dependencies under the transformation $\mathcal{T}$.

Finally, Theorem 6.8.3 concludes the proof showing that causal relationships are preserved in the transformed time series $(\mathbf{a}, \boldsymbol{\lambda})$ **(B1)**.

(A1) Weak Stationarity and Noise Assumptions

Theorem 6.8.1 (Asymptotic Gaussianity and Stationarity under Colored Noise)**.** *Let $\mathbf{x} = \{x(t)\}$ be a weakly stationary process with power spectral density $S(f_k) \propto 1/f_k^\alpha$ for $\alpha \geq 0$, where f_k denotes the k-th frequency component. Let $\phi(k)$ be the DFT over a window of size l, and define $(\mathbf{a}, \boldsymbol{\lambda})$ as in Algorithm 3. Then, for sufficiently large l, the sequences $(\mathbf{a}, \boldsymbol{\lambda})$ are approximately Gaussian and weakly stationary.*

Proof. As shown in [251], if $\mathbf{x}$ is a weakly stationary process with decaying autocorrelation, then the Discrete Fourier Transform (DFT) coefficients computed over a window of size l converge in distribution to complex Gaussian variables with variance given by the power spectral density $S(f_k)$. That is,

$$\phi_k \xrightarrow{d} \mathcal{CN}(0, S(f_k)).$$

The spectral amplitudes $A_k = |\phi_k|$ follow a Rayleigh distribution with scale parameter determined by $S(f_k)$. Let $(\mathbf{a}, \boldsymbol{\lambda})$ be computed as in Algorithm 3, by linear regression on the pairs $(\log f_k, \log A_k)$ within each window. Although the $\log A_k$ are not Gaussian, the regression combines information from many such values. Since the estimators are linear combinations of independent (or weakly dependent) inputs with finite variance, they are approximately Gaussian by the Central Limit Theorem.

Therefore, for sufficiently large window length l, the estimated features $(\mathbf{a}, \boldsymbol{\lambda})$ can be treated as approximately Gaussian variables.

Regarding the i.i.d. assumption of the noise, this trivially holds in the case of non-overlapping windows. However, when the windows overlap, the input data for consecutive Fourier transforms

share common segments, which induces statistical dependence between the resulting spectral estimates. As a consequence, the noise affecting the estimated parameters $(\mathbf{a}, \boldsymbol{\lambda})$ is not independent across windows, but exhibits weak temporal correlation. □

(A2) Preservation of Linear Dependence under Spectral Transformation

Theorem 6.8.2 (Preservation of Linear Dependence under Spectral Transformation)**.** *Let $\mathbf{x}_i$ and $\mathbf{x}_j$ be two power-law time series and let $\mathcal{T}$ denote the transformation described in Algorithm 3.*

Then, if $\mathbf{x}_j = \alpha \cdot \mathbf{x}_i$ for some constant $\alpha \in \mathbb{R}$ and all t, then $\boldsymbol{\lambda}_j = \boldsymbol{\lambda}_i$ and $\boldsymbol{a}_j = \boldsymbol{a}_i + \log|\alpha|$.

Therefore, the spectral transformation $\mathcal{T}$ preserves both linear dependence and linear independence between time series.

Proof. Suppose $\mathbf{x}_j = \alpha \cdot \mathbf{x}_i$. Then, the power spectra satisfy:

$$A_j(f) = |\mathcal{F}[\mathbf{x}_j]| = |\alpha \cdot \mathcal{F}[\mathbf{x}_i]| = |\alpha| \cdot A_i(f),$$

which implies:

$$\log A_j(f) = \log A_i(f) + \log|\alpha|,$$

and so $\forall f$,

$$-\boldsymbol{\lambda}_j \log f + \mathbf{a}_j = -\boldsymbol{\lambda}_i \log f + \mathbf{a}_i + \log|\alpha|.$$

Therefore, both series have the same spectral slope

$$\boldsymbol{\lambda}_j = \boldsymbol{\lambda}_i,$$

and the intercepts differ by a constant:

$$\boldsymbol{a}_j(t) = \boldsymbol{a}_i(t) + \log|\alpha|.$$

Hence, $\mathcal{T}$ preserves both linear dependence between $\boldsymbol{\lambda}_i$ and $\boldsymbol{\lambda}_j$.

□

(B1) Spectral Causality Preservation

Theorem 6.8.3 (Preservation of Linear Causal Structure under Spectral Transformation)**.** *Let $\mathbf{x}_i$ and $\mathbf{x}_j$ be two power-law time series such that*

$$\mathbf{x}_i = g(\mathbf{x}_j) + \boldsymbol{\eta}, \tag{6.4}$$

where g is a linear function and $\boldsymbol{\eta} = \{\eta(t)\}$ is additive noise independent of $\mathbf{x}_j$. Then, the spectral parameters $(\mathbf{a}_i, \boldsymbol{\lambda}_i)$ defined in Algorithm 3 retain information about the causal influence from $\mathbf{x}_j$ to $\mathbf{x}_i$.

Proof. Applying the Fourier transform to both sides of Equation (6.4) and exploiting the linearity of the Fourier transform, we get:

$$\mathcal{F}[(\mathbf{x}_i)] = \mathcal{F}[g(\mathbf{x}_j)] + \mathcal{F}[\boldsymbol{\eta}].$$

We prove that in linear settings the spectral amplitude $A_i(f)$ retains features shaped by the causal dependency on $\mathbf{x}_j$. Consequently, for linear causal relationships, the pair $(\mathbf{a}_i, \boldsymbol{\lambda}_i)$ derived from a log-log fit of $A_i(f)$ preserves causal information.

If g is a linear operator, then by the linearity of $\mathcal{F}$,

$$\mathcal{F}[g(\mathbf{x}_j)] = g(\mathcal{F}[\mathbf{x}_j]).$$

Taking the modulus yields:

$$A_i(f) = |g(A_j(f))| + \mathcal{F}[\boldsymbol{\eta}].$$

If $A_j(f) \propto f^{-\boldsymbol{\lambda}_j}$ (i.e., $\mathbf{x}_j$ exhibits a power-law spectrum), then under mild regularity assumptions on g, the transformed amplitude $A_i(f)$ also exhibits a power-law decay:

$$A_i(f) \propto f^{-\boldsymbol{\lambda}_i}, \quad \text{with } \boldsymbol{\lambda}_i = \boldsymbol{\lambda}_j + \Delta\boldsymbol{\lambda},$$

where $\Delta\boldsymbol{\lambda}$ captures the spectral effect of g. Thus, the spectral slope $\boldsymbol{\lambda}_i$ contains information induced by the causal transformation g.

□

Nonlinear causal relationship g. For nonlinear g, $\mathcal{F}[g(\mathbf{x}_j)] \neq g(\mathcal{F}[\mathbf{x}_j])$ in general. However, due to the orthogonality and completeness of the Fourier basis, the operation g might induce structured interactions among frequencies. This might distort without destroying the underlying spectral shape. In practice, we found experimentally that even under nonlinear transformations, the global decay behavior captured by the spectral parameters $\boldsymbol{\lambda}$ typically remains a meaningful summary of the causal influence, as suggested by the experiments of Section 6.8.4.

Mathematical computation of λ for our Synthetic Datasets

In this section, we provide a mathematical derivation of the spectral parameter λ for the linear additive system used to generate the synthetic datasets in Section 6.5.1.

Let $\mathbf{x}_1$, $\mathbf{x}_2$, and $\mathbf{x}_3$ be three time series whose spectral amplitudes follow a power-law profile. In the case of an additive interaction in the time domain, the resulting spectral amplitude of $\mathbf{x}_1$ can be expressed—under idealized linearity and independence assumptions—as:

$$\forall f \quad A_1(f) = A_2(f) + c \cdot A_3(f)$$

where c is a scalar coefficient regulating the strength of the contribution from $\mathbf{x}_3$ to $\mathbf{x}_1$, and f is the value of a frequency of the spectrum.

$$A_1(f) = e^{\mathbf{a}_1} \cdot f^{-\boldsymbol{\lambda}_1}, \quad A_2(f) = e^{\mathbf{a}_2} \cdot f^{-\boldsymbol{\lambda}_2}, \quad A_3(f) = e^{\mathbf{a}_3} \cdot f^{-\boldsymbol{\lambda}_3}$$

Substituting into the first equation gives:

$$e^{\mathbf{a}_1} \cdot f^{-\boldsymbol{\lambda}_1} = e^{\mathbf{a}_2} \cdot f^{-\boldsymbol{\lambda}_2} + c \cdot e^{\mathbf{a}_3} \cdot f^{-\boldsymbol{\lambda}_3} \tag{6.5}$$

Case 1: Assume the following:

$$\frac{c \cdot e^{\mathbf{a}_3}}{e^{\mathbf{a}_2}} \cdot f^{\boldsymbol{\lambda}_2 - \boldsymbol{\lambda}_3} \gg 1. \tag{6.6}$$

Notice that we can write the right-hand side of Equation (6.5) as $e^{\mathbf{a}_2} \cdot f^{-\boldsymbol{\lambda}_2} \left[1 + \frac{c \cdot e^{\mathbf{a}_3}}{e^{\mathbf{a}_2}} \cdot f^{\boldsymbol{\lambda}_2 - \boldsymbol{\lambda}_3}\right]$. By applying this substitution, taking the natural logarithm in Equation (6.5), and using logarithmic properties, we get:

$$\log\left(e^{\mathbf{a}_1} \cdot f^{-\boldsymbol{\lambda}_1}\right) = \log\left(e^{\mathbf{a}_2} \cdot f^{-\boldsymbol{\lambda}_2}\left[1 + \frac{c \cdot e^{\mathbf{a}_3}}{e^{\mathbf{a}_2}} \cdot f^{\boldsymbol{\lambda}_2 - \boldsymbol{\lambda}_3}\right]\right) \tag{6.7}$$

$$= \mathbf{a}_2 - \boldsymbol{\lambda}_2 \log f + \log\left(1 + \frac{c \cdot e^{\mathbf{a}_3}}{e^{\mathbf{a}_2}} \cdot f^{\boldsymbol{\lambda}_2 - \boldsymbol{\lambda}_3}\right) \tag{6.8}$$

Given the assumption in Equation (6.6), the logarithmic term in the previous equation can be approximated as:

$$\log\left(1 + \frac{c \cdot e^{\mathbf{a}_3}}{e^{\mathbf{a}_2}} \cdot f^{\boldsymbol{\lambda}_2 - \boldsymbol{\lambda}_3}\right) \approx \log\left(\frac{c \cdot e^{\mathbf{a}_3}}{e^{\mathbf{a}_2}} \cdot f^{\boldsymbol{\lambda}_2 - \boldsymbol{\lambda}_3}\right)$$

By substituting this expression in Equation (6.7), we obtain the following equation:

$$\mathbf{a}_1 - \boldsymbol{\lambda}_1 \log f \approx \mathbf{a}_2 - \boldsymbol{\lambda}_2 \log f + \log\left(\frac{c \cdot e^{\mathbf{a}_3}}{e^{\mathbf{a}_2}}\right) + (\boldsymbol{\lambda}_2 - \boldsymbol{\lambda}_3) \log f$$

Which can be simplified as follows:

$$\mathbf{a}_1 - \boldsymbol{\lambda}_1 \log f \approx \log c + \mathbf{a}_3 - \boldsymbol{\lambda}_3 \log f$$

Finally, by solving for $\boldsymbol{\lambda}_1$, we get the following expression for $\boldsymbol{\lambda}_1$:

$$\boldsymbol{\lambda}_1 \approx \boldsymbol{\lambda}_3 + \frac{\log c + \mathbf{a}_3 - \mathbf{a}_1}{\log f}$$

This shows that the spectral decay parameter $\boldsymbol{\lambda}_1$ approximates $\boldsymbol{\lambda}_3$ with a correction term depending on the scaling factor c, the spectral amplitude, and frequency.

Case 2: Now consider the case where $\frac{c \cdot e^{\mathbf{a}_3}}{e^{\mathbf{a}_2}} \cdot f^{\boldsymbol{\lambda}_2 - \boldsymbol{\lambda}_3} \ll 1$, such as when $c \to 0$. The logarithmic term is approximated using the first-order Taylor expansion:

$$\log\left(1 + \frac{c \cdot e^{\mathbf{a}_3}}{e^{\mathbf{a}_2}} \cdot f^{\boldsymbol{\lambda}_2 - \boldsymbol{\lambda}_3}\right) \approx \frac{c \cdot e^{\mathbf{a}_3}}{e^{\mathbf{a}_2}} \cdot f^{\boldsymbol{\lambda}_2 - \boldsymbol{\lambda}_3}$$

Substituting into the main expression:

$$\mathbf{a}_1 - \boldsymbol{\lambda}_1 \log f \approx \mathbf{a}_2 - \boldsymbol{\lambda}_2 \log f + \frac{c \cdot e^{\mathbf{a}_3}}{e^{\mathbf{a}_2}} \cdot f^{\boldsymbol{\lambda}_2 - \boldsymbol{\lambda}_3}$$

Solving for $\boldsymbol{\lambda}_1$:

$$\boldsymbol{\lambda}_1 \approx \boldsymbol{\lambda}_2 + \frac{\mathbf{a}_2 - \mathbf{a}_1}{\log f} + c \cdot e^{\mathbf{a}_3 - \mathbf{a}_2} \cdot \frac{f^{\boldsymbol{\lambda}_2 - \boldsymbol{\lambda}_3}}{\log f}$$

As $c \to 0$, the correction vanishes, and $\boldsymbol{\lambda}_1 \to \boldsymbol{\lambda}_2$, showing that the spectrum of the target converges to that of the source.

Discussion on the VAR Inputs

From the expressions derived in Section 6.8.1, it becomes evident that in the linear case, one can identify certain quantities that are valuable inputs to a VAR model. As an example, given the two time series $\mathbf{x}_1$ and $\mathbf{x}_3$ of Section 6.8.1, the relevant quantities are:

- $\mathbf{a}_1$: spectral intercept of the caused time series
- $\mathbf{a}_3$: spectral intercept of the causing time series
- $\boldsymbol{\lambda}_1$: spectral slope of the caused time series
- $\boldsymbol{\lambda}_3$: spectral slope of the causing time series

In this analysis, we neglect higher-order terms (e.g., exponentials in $\mathbf{a}$), as they empirically fail to improve system performance and are not present in the asymptotic regime discussed in Section 6.8.1.

Each time series thus provides two spectral parameters: $\boldsymbol{\lambda}$ and $\mathbf{a}$. Among these, the $\boldsymbol{\lambda}$ parameters are identified as the main carriers of causal information. The simplest model we can consider involves assessing Granger causality between the $\boldsymbol{\lambda}$ sequences, i.e., testing for $\boldsymbol{\lambda}_3 \rightarrow \boldsymbol{\lambda}_1$ relationships.

Enhancing the system's robustness in the presence of stationary processes, including the $\mathbf{a}$ parameter as a covariate in the VAR model, has shown some performance improvements. However, in non-stationary settings, adding the $\mathbf{a}$ provides no additional causal information beyond $\boldsymbol{\lambda}$.

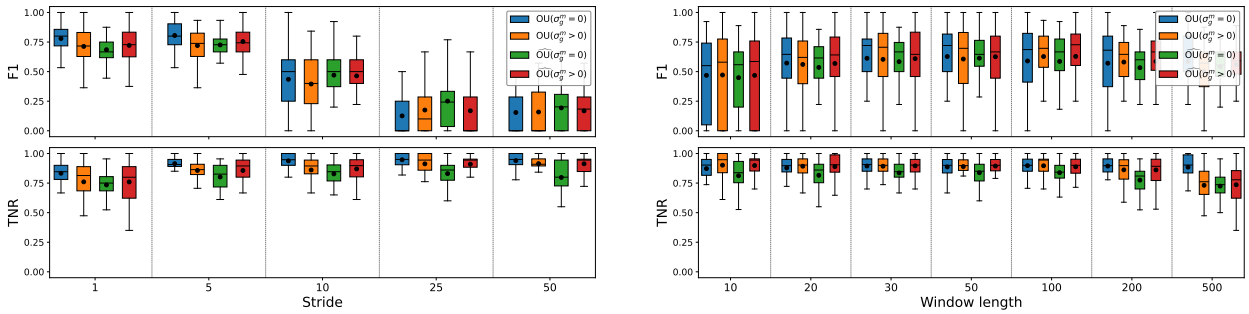
Finally, using the $\mathbf{a}$ parameter of the caused series either as an input or as a target in the VAR model has proven ineffective: while it does not meaningfully improve performance in the stationary case, it introduces false positives in non-stationary regimes. Therefore, its inclusion is not theoretically nor empirically justified.

Study of p-values for Setting Window Length and Stride

In the proposed framework, both the stride and the sliding window length are configurable parameters that can be tuned by the user. The stride primarily serves to increase the number of available data points for analysis and plays a crucial role in the system's ability to detect short-range causal dependencies. If the true causal lag is shorter than the selected stride, the system may fail to capture such dynamics. Therefore, it is generally recommended to set the stride to 1. The sliding window length is a more delicate parameter, as it is influenced by multiple factors. A window that is too short may yield unreliable estimates due to limited data, whereas an excessively long window may blur or attenuate the causal signal, thus reducing detection sensitivity. To mitigate this trade-off, we introduce a data-driven procedure to estimate a suitable window length. This involves a preliminary evaluation of the fitting procedure across various window sizes, based on the distribution of p-values over the entire spectrum. The smallest window size that satisfies a p-value of 0.05 is then selected for subsequent causal inference.

A reasonable requirement is also to fix a lower bound of this experimental criterion to a window length of a minimum of 50 datapoints. A smaller window may still satisfy the p-value requirement, but it will also make the fitted $\boldsymbol{\lambda}$ parameter too sensitive to endogenous variation of the autocorrelation caused by the phenomenon.

A study in function of stride and window length is shown in Figure 6.3.



(a) Plot $N = 5$, $s_b = 0$, window length $= 50$ **(b)** Plot $N = 5$, $s_b = 0$, stride $= 1$

Figure 6.3: Stride and window length analysis.

Slow-Varying Spectrum Systems

There are cases in which the system is sampled at such a high rate that it exhibits minimal local spectral variation. This typically occurs when the white noise component is small compared to other system dynamics. In these scenarios, spectral analysis becomes challenging, as we rely on observing how the spectrum evolves over time to infer causal relationships.

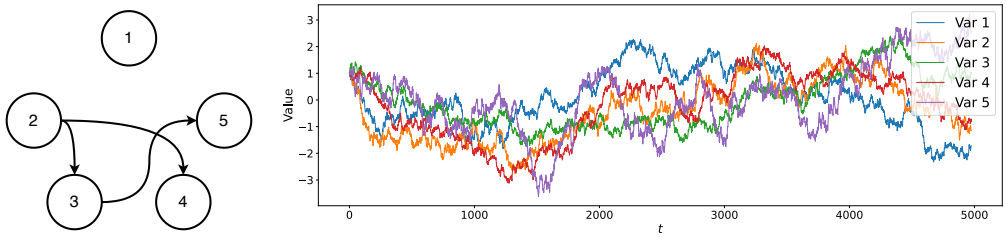
To address this issue, one should increase the window length and, if possible, reduce the stride of the sliding window. This allows the VAR model in the frequency domain to capture meaningful variations in the time series and better estimate potential causal links.

In these situations, time-domain methods like Granger causality may be more appropriate. While this introduces a limitation, the selection of a suitable causal inference technique can be effectively informed by an initial spectral analysis.

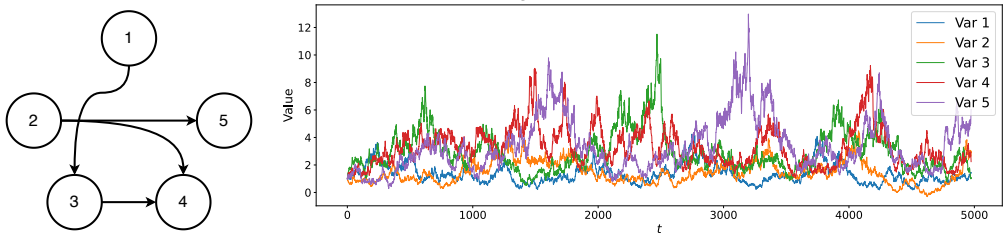
6.8.2 Further Experimental Details and Results

6.8.3 OU Processes

Figure 6.4 shows two examples of the simulated OU processes, and the related causal graphs. In particular, an example of OU($\sigma_g^m = 0$) is shown in Figure 6.4a and an example of OU($\sigma_g^m > 0$) is shown in Figure 6.4b.



(a) OU($\sigma_g^m = 0$)



(b) OU($\sigma_g^m > 0$)

Figure 6.4: Generated synthetic processes.

6.8.4 Additional Results

Tables 6.3 to 6.10 present all the experimental results for different configurations. We analyze the impact of varying the number of variables ($N \in \{5, 10\}$) and the scaling factor $\sigma_g^a \in \{0.5, 1\}$ on the methods' performance. In bold, we highlight the results of the best-performing algorithm in each scenario for a given estimator. Since some algorithms mainly detect causal relationships, their TNR values tend to be high. In the TNR tables, we also highlighted in green the best TNR experiments that achieved an F1 score of at least 0.5. These experiments clearly demonstrate that PLaCy is the most reliable algorithm, achieving the highest F1 score in most scenarios and a competitive or even superior TNR among all methods analyzed. Please note that we omit the results of RHINO and CCM in the tables: as explained in Section 6.8.5, their prohibitive execution times made it impossible to run a sufficient number of experiments. In fact, Rhino lacks cross-configuration generalization along sequence length, noise level, and causal strength, necessitating 192[3] distinct trainings to cover our experimental campaign. Combined with markedly greater runtime (see Section Section 6.8.5) and lower F1 and TNR in the results of Section A.2.5, this yields an unfavorable accuracy-efficiency trade-off. For these reasons, we omit it from the additional experiments reported in this section. A similar reasoning applies to CCM, for which, using the mean execution time reported in Section 6.8.5, the total computation would amount to approximately one month in total.

Partial results on these two methods did not considerably differ from those reported in Section A.2.5.

Hyper-Parameters

Table 6.11 shows the list of the hyper-parameters that we set for each causal discovery algorithm. Wherever possible, we adopted the hyper-parameters specified in the original papers, adjusting them only in cases of extremely slow computation or lack of algorithmic convergence.

PCMCI in Frequency

We observed that performing causal discovery on the $(\mathbf{a}, \boldsymbol{\lambda})$ parameters leads to significant performance improvements for other CD algorithms. For instance, regarding the **PCMCI** algorithm, Table 6.12 shows that instead of applying the algorithm directly to the time-domain data, running it on the spectral parameters results in higher F1 scores with a small reduction in TNR in some cases. These results suggest that our preprocessing approach can be beneficial also when applied to other causal discovery paradigms, paving the way for future works in this direction.

Non linear system analysis

Some additional experiments have been conducted in cases where the underlying process wasn't linear in order to test the stability of this algorithm in more challenging scenarios. The equation that was used to generate the process used in these experiments is Equation (6.9):

$$x(t + \Delta t) = x(t) + \frac{\Delta t}{\tau_c}\big(\mu - x(t)\big)^2 + \big(\sigma_b \epsilon_b(t) + \sigma_g^a \epsilon_g^a(t) + \sigma_g^m \epsilon_g^m(t) \cdot x(t)\big)\sqrt{\Delta t}, \tag{6.9}$$

Results of these experiments are reported in Figure 6.5

[3]Our tested configurations include 4 OU processes, 4 values for σ_b, 2 values for σ_g, 3 values for C, and 2 for N.

Table 6.3: F1 Score - $N = 5,\ \sigma_g^a = 0.5$

Dataset	C	σ_b	Granger	PLACy	PCMCI	RCV-VarLiNGAM	DYNOTEARS	PCMCI$_\Omega$	BCGeweke	DTF	GewekeNP
OU($\sigma_g^m = 0$)	0.2	0.0	**0.85**±0.19	0.12±0.19	0.06±0.15	0.44±0.17	0.02±0.11	0.06±0.12	0.84±0.08	0.44±0.09	0.61±0.05
		0.1	0.44±0.17	0.15±0.22	0.09±0.19	0.44±0.17	0.12±0.21	0.08±0.12	0.57±0.17	0.27±0.21	**0.60**±0.05
		0.5	0.40±0.16	0.14±0.19	0.24±0.27	0.36±0.23	0.23±0.13	0.10±0.14	0.47±0.08	0.46±0.09	**0.60**±0.07
		1.0	0.49±0.18	0.10±0.18	0.11±0.20	0.34±0.24	0.21±0.13	0.14±0.15	0.55±0.10	0.44±0.08	**0.60**±0.06
	0.5	0.0	0.82±0.18	**0.84**±0.17	0.11±0.21	0.44±0.18	0.01±0.10	0.06±0.12	0.77±0.06	0.73±0.08	0.52±0.06
		0.1	0.43±0.17	**0.77**±0.23	0.13±0.21	0.41±0.18	0.02±0.10	0.07±0.11	0.51±0.09	0.42±0.23	0.52±0.07
		0.5	0.38±0.14	**0.60**±0.26	0.32±0.27	0.32±0.24	0.10±0.18	0.15±0.16	0.47±0.07	0.48±0.09	0.52±0.05
		1.0	0.44±0.14	0.41±0.27	0.21±0.27	0.32±0.23	0.11±0.18	0.24±0.17	0.48±0.09	0.46±0.08	**0.53**±0.06
	1.0	0.0	0.78±0.19	**0.78**±0.16	0.20±0.28	0.40±0.17	0.01±0.10	0.07±0.13	0.76±0.10	0.64±0.10	0.53±0.08
		0.1	0.44±0.18	**0.77**±0.16	0.20±0.26	0.37±0.18	0.04±0.11	0.07±0.11	0.49±0.10	0.45±0.19	0.52±0.07
		0.5	0.35±0.12	**0.74**±0.17	0.42±0.28	0.22±0.21	0.17±0.22	0.24±0.17	0.45±0.09	0.44±0.10	0.53±0.07
		1.0	0.41±0.13	**0.65**±0.21	0.31±0.28	0.18±0.22	0.18±0.22	0.31±0.16	0.48±0.09	0.43±0.09	0.54±0.07
$\widehat{\text{OU}}(\sigma_g^m = 0)$	0.2	0.0	**0.85**±0.19	0.20±0.24	0.06±0.18	0.44±0.17	0.00±0.00	0.06±0.12	0.83±0.11	0.44±0.07	0.44±0.07
		0.1	0.33±0.13	0.29±0.23	0.16±0.24	0.42±0.17	0.00±0.00	0.05±0.11	**0.46**±0.07	0.43±0.07	0.44±0.07
		0.5	0.27±0.11	0.17±0.22	0.29±0.30	0.38±0.18	0.03±0.12	0.12±0.14	0.47±0.08	0.44±0.07	**0.49**±0.05
		1.0	0.36±0.14	0.20±0.23	0.15±0.23	0.33±0.19	0.06±0.15	0.14±0.15	0.52±0.09	0.44±0.07	**0.57**±0.07
	0.5	0.0	**0.81**±0.19	0.74±0.17	0.09±0.18	0.44±0.17	0.00±0.00	0.08±0.13	0.78±0.10	0.44±0.07	0.44±0.07
		0.1	0.32±0.12	**0.61**±0.20	0.17±0.23	0.42±0.18	0.00±0.00	0.09±0.13	0.47±0.08	0.44±0.07	0.44±0.07
		0.5	0.27±0.11	**0.57**±0.25	0.36±0.28	0.34±0.22	0.01±0.05	0.16±0.15	0.47±0.07	0.44±0.07	0.47±0.10
		1.0	0.35±0.12	0.46±0.28	0.15±0.24	0.30±0.21	0.03±0.10	0.27±0.18	0.50±0.07	0.44±0.07	**0.51**±0.06
	1.0	0.0	**0.78**±0.19	0.71±0.17	0.18±0.25	0.38±0.17	0.00±0.00	0.09±0.13	0.75±0.09	0.43±0.09	0.43±0.09
		0.1	0.32±0.12	**0.58**±0.16	0.19±0.26	0.35±0.17	0.00±0.00	0.07±0.12	0.47±0.09	0.43±0.09	0.43±0.09
		0.5	0.27±0.11	**0.65**±0.19	0.37±0.28	0.34±0.22	0.01±0.04	0.22±0.17	0.44±0.09	0.43±0.09	0.46±0.10
		1.0	0.33±0.12	**0.60**±0.18	0.28±0.29	0.24±0.23	0.03±0.13	0.34±0.17	0.49±0.07	0.43±0.09	0.51±0.08
OU($\sigma_g^m > 0$)	0.2	0.0	0.74±0.20	0.45±0.31	0.14±0.24	0.45±0.17	0.19±0.16	0.14±0.15	**0.88**±0.07	0.49±0.10	0.58±0.05
		0.1	0.68±0.20	0.57±0.29	0.11±0.22	0.41±0.17	0.20±0.19	0.11±0.14	**0.78**±0.08	0.33±0.07	0.54±0.06
		0.5	0.66±0.21	0.60±0.29	0.11±0.21	0.40±0.17	0.22±0.13	0.14±0.16	**0.76**±0.10	0.23±0.21	0.54±0.07
		1.0	0.69±0.22	0.70±0.20	0.08±0.19	0.43±0.17	0.24±0.12	0.13±0.14	**0.76**±0.10	0.42±0.16	0.50±0.06
	0.5	0.0	0.70±0.19	0.81±0.14	0.24±0.31	0.42±0.17	0.06±0.16	0.16±0.16	**0.82**±0.06	0.66±0.15	0.52±0.07
		0.1	0.66±0.18	0.81±0.19	0.26±0.29	0.40±0.17	0.04±0.13	0.15±0.16	**0.83**±0.07	0.65±0.09	0.51±0.06
		0.5	0.64±0.19	0.77±0.20	0.20±0.26	0.37±0.19	0.13±0.16	0.17±0.17	**0.80**±0.12	0.53±0.24	0.56±0.08
		1.0	0.65±0.19	0.80±0.19	0.22±0.29	0.39±0.19	0.14±0.15	0.18±0.16	**0.86**±0.08	0.61±0.15	0.52±0.08
	1.0	0.0	0.66±0.19	**0.78**±0.15	0.43±0.34	0.39±0.18	0.07±0.15	0.20±0.17	0.75±0.12	0.63±0.09	0.53±0.07
		0.1	0.63±0.19	**0.77**±0.14	0.44±0.34	0.35±0.17	0.03±0.11	0.21±0.19	0.74±0.12	0.62±0.10	0.52±0.06
		0.5	0.64±0.18	**0.78**±0.17	0.38±0.34	0.35±0.20	0.13±0.18	0.21±0.18	0.74±0.13	0.61±0.11	0.52±0.09
		1.0	0.62±0.20	**0.74**±0.19	0.33±0.34	0.33±0.21	0.15±0.16	0.22±0.16	0.69±0.12	0.53±0.15	0.54±0.07
$\widehat{\text{OU}}(\sigma_g^m > 0)$	0.2	0.0	0.27±0.11	**0.55**±0.31	0.35±0.21	0.09±0.18	0.04±0.14	0.26±0.11	0.48±0.08	0.44±0.07	0.44±0.07
		0.1	0.27±0.11	**0.64**±0.28	0.36±0.24	0.11±0.21	0.04±0.14	0.26±0.11	0.47±0.07	0.46±0.07	0.44±0.07
		0.5	0.38±0.15	**0.67**±0.25	0.23±0.26	0.35±0.22	0.04±0.12	0.22±0.12	0.64±0.18	0.42±0.06	0.46±0.07
		1.0	0.56±0.20	0.66±0.27	0.14±0.25	0.42±0.19	0.05±0.16	0.14±0.15	**0.69**±0.11	0.34±0.23	0.49±0.11
	0.5	0.0	0.27±0.11	**0.84**±0.14	0.45±0.22	0.06±0.14	0.01±0.08	0.26±0.11	0.48±0.07	0.44±0.06	0.44±0.07
		0.1	0.27±0.11	**0.79**±0.18	0.44±0.23	0.08±0.17	0.01±0.07	0.26±0.11	0.48±0.07	0.46±0.06	0.44±0.07
		0.5	0.39±0.15	**0.79**±0.17	0.37±0.29	0.32±0.21	0.01±0.06	0.24±0.14	0.57±0.11	0.53±0.11	0.46±0.08
		1.0	0.57±0.19	**0.82**±0.18	0.26±0.30	0.36±0.20	0.01±0.07	0.15±0.15	0.71±0.13	0.45±0.13	0.48±0.08
	1.0	0.0	0.27±0.11	**0.80**±0.15	0.53±0.21	0.07±0.16	0.02±0.09	0.25±0.11	0.46±0.10	0.44±0.08	0.43±0.09
		0.1	0.28±0.11	**0.81**±0.17	0.53±0.21	0.09±0.20	0.02±0.12	0.25±0.12	0.46±0.10	0.44±0.08	0.43±0.09
		0.5	0.38±0.16	**0.76**±0.18	0.48±0.30	0.24±0.21	0.02±0.11	0.25±0.14	0.54±0.12	0.48±0.08	0.46±0.08
		1.0	0.53±0.18	**0.78**±0.17	0.39±0.32	0.28±0.21	0.03±0.14	0.21±0.15	0.68±0.13	0.53±0.10	0.48±0.08

Table 6.4: TNR Score - $N = 5,\ \sigma_g^a = 0.5$

Dataset	C	σ_b	Granger	PLACy	PCMCI	RCV-VarLiNGAM	DYNOTEARS	PCMCI$_\Omega$	BCGeweke	DTF	GewekeNP
OU($\sigma_g^m = 0$)	0.2	0.0	**0.96**±0.04	0.96±0.04	**0.99**±0.02	0.72±0.06	0.98±0.04	0.78±0.09	0.89±0.05	0.93±0.05	0.63±0.11
		0.1	0.65±0.14	0.96±0.04	**0.98**±0.03	0.73±0.07	0.88±0.09	0.78±0.09	0.55±0.20	0.79±0.08	**0.60**±0.14
		0.5	0.59±0.12	0.95±0.05	**0.96**±0.04	0.84±0.07	0.53±0.16	0.78±0.08	0.35±0.09	0.45±0.09	**0.60**±0.11
		1.0	0.72±0.11	0.94±0.05	**0.99**±0.02	0.87±0.06	0.46±0.14	0.77±0.09	0.52±0.09	0.31±0.05	**0.62**±0.08
	0.5	0.0	0.94±0.05	**0.95**±0.04	**0.99**±0.02	0.74±0.07	0.95±0.05	0.77±0.11	0.82±0.05	0.85±0.09	0.46±0.13
		0.1	0.63±0.14	**0.95**±0.05	**0.98**±0.03	0.78±0.07	0.98±0.05	0.75±0.11	0.46±0.09	0.77±0.12	0.48±0.10
		0.5	0.55±0.13	**0.95**±0.05	**0.95**±0.05	0.85±0.07	0.94±0.10	0.74±0.11	0.36±0.09	0.42±0.07	0.47±0.10
		1.0	0.66±0.12	0.93±0.06	**0.98**±0.03	0.89±0.07	0.93±0.09	0.75±0.11	0.38±0.09	0.34±0.08	**0.47**±0.12
	1.0	0.0	**0.92**±0.08	0.91±0.07	**0.99**±0.02	0.75±0.08	0.90±0.08	0.76±0.12	0.81±0.09	0.68±0.13	0.49±0.14
		0.1	0.63±0.17	**0.91**±0.07	**0.98**±0.04	0.79±0.07	0.94±0.05	0.76±0.11	0.41±0.12	0.61±0.20	0.48±0.14
		0.5	0.51±0.14	**0.91**±0.07	**0.94**±0.05	0.87±0.07	0.92±0.11	0.73±0.13	0.32±0.07	0.36±0.11	0.48±0.15
		1.0	0.61±0.15	**0.90**±0.08	**0.97**±0.04	0.92±0.06	0.91±0.10	0.73±0.13	0.40±0.10	0.29±0.06	0.51±0.14
$\widetilde{\text{OU}}(\sigma_g^m = 0)$	0.2	0.0	**0.96**±0.04	0.93±0.05	0.99±0.02	0.72±0.06	**1.00**±0.00	0.78±0.09	0.89±0.07	0.26±0.02	0.26±0.02
		0.1	0.45±0.10	0.86±0.08	0.97±0.03	0.73±0.06	**1.00**±0.00	0.78±0.09	0.33±0.05	0.26±0.02	0.26±0.02
		0.5	0.26±0.04	0.92±0.06	0.95±0.05	0.82±0.06	**0.99**±0.03	0.77±0.08	0.36±0.05	0.26±0.02	0.38±0.14
		1.0	0.52±0.12	0.91±0.06	**0.98**±0.03	0.85±0.05	0.96±0.05	0.78±0.09	0.48±0.10	0.26±0.02	**0.55**±0.13
	0.5	0.0	**0.94**±0.06	0.92±0.05	1.00±0.01	0.75±0.08	**1.00**±0.00	0.76±0.11	0.85±0.06	0.26±0.02	0.26±0.02
		0.1	0.44±0.10	**0.85**±0.08	0.95±0.05	0.78±0.07	**1.00**±0.00	0.75±0.11	0.36±0.05	0.26±0.02	0.26±0.02
		0.5	0.26±0.05	**0.91**±0.06	0.94±0.05	0.84±0.07	**0.99**±0.03	0.76±0.10	0.35±0.05	0.26±0.02	0.34±0.11
		1.0	0.50±0.12	0.90±0.07	**0.97**±0.03	0.87±0.07	0.95±0.05	0.76±0.11	0.41±0.13	0.26±0.02	**0.45**±0.10
	1.0	0.0	**0.92**±0.07	0.88±0.09	1.00±0.02	0.77±0.09	**1.00**±0.00	0.76±0.11	0.81±0.09	0.26±0.02	0.26±0.02
		0.1	0.43±0.12	**0.80**±0.10	0.96±0.04	0.78±0.08	**1.00**±0.00	0.75±0.12	0.37±0.09	0.26±0.02	0.26±0.02
		0.5	0.26±0.04	**0.88**±0.09	0.93±0.06	0.87±0.08	**0.98**±0.04	0.73±0.11	0.30±0.05	0.26±0.02	0.33±0.11
		1.0	0.47±0.12	**0.85**±0.10	**0.97**±0.03	0.91±0.07	0.93±0.06	0.73±0.13	0.43±0.10	0.26±0.02	0.45±0.13
OU($\sigma_g^m > 0$)	0.2	0.0	**0.91**±0.06	0.96±0.04	**0.99**±0.02	0.73±0.06	0.72±0.09	0.77±0.09	0.91±0.06	0.91±0.06	0.63±0.12
		0.1	0.89±0.07	**0.95**±0.05	**0.99**±0.03	0.73±0.07	0.77±0.13	0.76±0.10	0.86±0.08	0.86±0.08	0.57±0.08
		0.5	0.88±0.07	**0.96**±0.06	**0.99**±0.02	0.74±0.07	0.49±0.19	0.76±0.11	0.81±0.08	0.87±0.09	0.63±0.10
		1.0	0.88±0.10	**0.96**±0.04	**0.98**±0.03	0.74±0.07	0.40±0.15	0.75±0.10	0.84±0.07	0.90±0.07	0.57±0.11
	0.5	0.0	0.88±0.09	**0.94**±0.05	**0.99**±0.02	0.75±0.07	0.95±0.05	0.75±0.10	0.86±0.07	0.75±0.18	0.46±0.12
		0.1	0.86±0.08	**0.95**±0.06	**0.98**±0.03	0.76±0.08	0.96±0.05	0.73±0.12	0.87±0.09	0.82±0.12	0.47±0.12
		0.5	0.84±0.11	**0.94**±0.06	**0.98**±0.03	0.76±0.09	0.81±0.15	0.74±0.12	0.84±0.15	0.80±0.08	0.51±0.18
		1.0	0.84±0.10	**0.94**±0.06	**0.98**±0.03	0.78±0.08	0.75±0.16	0.72±0.12	0.90±0.06	0.81±0.14	0.50±0.18
	1.0	0.0	0.85±0.13	**0.91**±0.09	**0.98**±0.03	0.76±0.07	0.91±0.08	0.75±0.11	0.80±0.14	0.69±0.15	0.49±0.17
		0.1	0.82±0.13	**0.91**±0.08	**0.98**±0.03	0.77±0.09	0.93±0.07	0.73±0.12	0.78±0.16	0.68±0.15	0.48±0.15
		0.5	0.83±0.13	**0.91**±0.08	**0.98**±0.03	0.78±0.08	0.81±0.14	0.72±0.14	0.77±0.14	0.71±0.16	0.46±0.17
		1.0	0.81±0.15	**0.90**±0.08	**0.98**±0.03	0.79±0.08	0.75±0.14	0.73±0.12	0.75±0.11	0.66±0.14	0.51±0.16
$\widetilde{\text{OU}}(\sigma_g^m > 0)$	0.2	0.0	0.28±0.08	**0.96**±0.04	0.84±0.07	0.97±0.06	**0.97**±0.05	0.28±0.08	0.37±0.08	0.36±0.07	0.26±0.02
		0.1	0.30±0.10	**0.96**±0.05	0.86±0.07	0.96±0.06	**0.97**±0.05	0.30±0.11	0.36±0.07	0.40±0.08	0.26±0.02
		0.5	0.56±0.15	**0.96**±0.04	0.94±0.04	0.81±0.09	0.96±0.05	0.53±0.13	0.62±0.19	0.63±0.17	0.34±0.08
		1.0	0.79±0.11	**0.96**±0.04	**0.98**±0.03	0.76±0.08	0.95±0.05	0.70±0.12	0.78±0.10	0.81±0.09	0.54±0.16
	0.5	0.0	0.28±0.08	**0.95**±0.06	0.84±0.07	**0.97**±0.05	0.96±0.05	0.28±0.08	0.37±0.07	0.32±0.06	0.26±0.02
		0.1	0.28±0.09	**0.94**±0.05	0.86±0.07	0.96±0.06	**0.96**±0.05	0.29±0.09	0.37±0.10	0.35±0.06	0.27±0.03
		0.5	0.56±0.18	**0.94**±0.06	0.95±0.05	0.83±0.09	**0.96**±0.05	0.53±0.15	0.55±0.15	0.51±0.18	0.34±0.09
		1.0	0.80±0.11	**0.95**±0.05	**0.98**±0.03	0.79±0.08	0.93±0.06	0.70±0.13	0.76±0.14	0.68±0.17	0.41±0.10
	1.0	0.0	0.27±0.08	**0.92**±0.08	0.84±0.08	**0.97**±0.05	0.96±0.06	0.28±0.08	0.34±0.08	0.29±0.05	0.26±0.02
		0.1	0.29±0.11	**0.93**±0.07	0.86±0.07	**0.96**±0.06	**0.96**±0.06	0.29±0.08	0.36±0.10	0.29±0.06	0.26±0.02
		0.5	0.52±0.18	**0.92**±0.07	**0.95**±0.05	0.83±0.08	0.95±0.06	0.51±0.15	0.52±0.15	0.42±0.10	0.34±0.11
		1.0	0.74±0.15	**0.91**±0.09	**0.97**±0.03	0.80±0.08	0.92±0.07	0.69±0.12	0.73±0.12	0.54±0.15	0.40±0.13

Table 6.5: F1 Score - $N = 5,\ \sigma_g^a = 1.0$

Dataset	C	σ_b	Granger	PLACy	PCMCI	RCV-VarLiNGAM	DYNOTEARS	PCMCI$_\Omega$	BCGeweke	DTF	GewekeNP
OU($\sigma_g^m = 0$)	0.2	0.0	**0.85**±0.20	0.13±0.21	0.08±0.19	0.44±0.17	0.05±0.16	0.06±0.12	0.82±0.10	0.45±0.13	0.60±0.06
		0.1	0.63±0.21	0.14±0.21	0.06±0.14	0.43±0.17	0.15±0.21	0.04±0.11	**0.68**±0.16	0.18±0.20	0.58±0.06
		0.5	0.43±0.17	0.13±0.21	0.22±0.28	0.36±0.19	0.23±0.15	0.10±0.13	0.47±0.07	0.40±0.13	**0.60**±0.05
		1.0	0.39±0.14	0.15±0.24	0.24±0.26	0.36±0.21	0.23±0.13	0.10±0.14	0.48±0.08	0.37±0.12	**0.59**±0.06
	0.5	0.0	0.82±0.19	**0.83**±0.16	0.08±0.19	0.44±0.18	0.02±0.12	0.08±0.12	0.76±0.08	0.75±0.09	0.52±0.06
		0.1	0.62±0.22	**0.81**±0.19	0.07±0.17	0.43±0.16	0.02±0.09	0.08±0.12	0.64±0.08	0.42±0.13	0.51±0.07
		0.5	0.43±0.16	**0.67**±0.28	0.34±0.27	0.37±0.19	0.12±0.20	0.11±0.15	0.47±0.08	0.44±0.16	0.53±0.07
		1.0	0.37±0.13	**0.60**±0.25	0.32±0.27	0.33±0.19	0.16±0.22	0.18±0.16	0.46±0.07	0.47±0.09	0.51±0.08
	1.0	0.0	0.78±0.20	**0.78**±0.15	0.19±0.27	0.40±0.17	0.02±0.09	0.08±0.13	0.73±0.09	0.67±0.14	0.52±0.08
		0.1	0.61±0.21	**0.77**±0.18	0.15±0.25	0.39±0.17	0.04±0.13	0.09±0.13	0.61±0.10	0.56±0.08	0.51±0.07
		0.5	0.39±0.16	**0.78**±0.17	0.38±0.29	0.27±0.22	0.12±0.18	0.15±0.16	0.44±0.09	0.46±0.09	0.50±0.07
		1.0	0.35±0.12	**0.73**±0.19	0.37±0.27	0.22±0.20	0.14±0.22	0.20±0.16	0.44±0.09	0.46±0.08	0.52±0.08
$\widehat{\text{OU}}(\sigma_g^m = 0)$	0.2	0.0	**0.85**±0.19	0.21±0.24	0.06±0.18	0.44±0.17	0.00±0.00	0.06±0.12	0.83±0.11	0.44±0.07	0.44±0.07
		0.1	0.47±0.19	0.34±0.24	0.04±0.13	0.44±0.17	0.00±0.00	0.06±0.12	**0.57**±0.11	0.43±0.07	0.44±0.07
		0.5	0.27±0.11	0.22±0.24	0.31±0.27	0.38±0.19	0.02±0.12	0.09±0.13	0.45±0.08	0.44±0.08	**0.50**±0.09
		1.0	0.28±0.11	0.19±0.24	0.24±0.26	0.38±0.20	0.05±0.16	0.09±0.13	0.46±0.09	0.44±0.07	**0.58**±0.08
	0.5	0.0	**0.81**±0.19	0.79±0.18	0.10±0.19	0.43±0.17	0.00±0.00	0.08±0.12	0.78±0.10	0.44±0.07	0.44±0.07
		0.1	0.45±0.18	**0.71**±0.18	0.06±0.15	0.43±0.17	0.00±0.00	0.07±0.12	0.52±0.06	0.44±0.08	0.44±0.07
		0.5	0.26±0.11	**0.70**±0.22	0.39±0.25	0.32±0.20	0.00±0.04	0.09±0.15	0.45±0.06	0.44±0.07	0.46±0.07
		1.0	0.28±0.11	**0.56**±0.25	0.31±0.26	0.30±0.22	0.02±0.10	0.16±0.16	0.46±0.08	0.44±0.07	0.50±0.05
	1.0	0.0	**0.78**±0.19	0.74±0.18	0.18±0.25	0.37±0.17	0.00±0.00	0.08±0.13	0.75±0.09	0.43±0.09	0.43±0.09
		0.1	0.46±0.19	**0.68**±0.17	0.12±0.21	0.37±0.17	0.00±0.00	0.08±0.13	0.53±0.10	0.43±0.09	0.43±0.09
		0.5	0.27±0.11	**0.71**±0.16	0.45±0.27	0.29±0.20	0.01±0.05	0.11±0.13	0.43±0.09	0.42±0.09	0.46±0.08
		1.0	0.28±0.11	**0.66**±0.17	0.42±0.30	0.25±0.20	0.02±0.10	0.23±0.17	0.45±0.08	0.43±0.09	0.51±0.07
OU($\sigma_g^m > 0$)	0.2	0.0	0.65±0.22	0.68±0.25	0.25±0.31	0.41±0.19	0.20±0.16	0.16±0.16	**0.84**±0.09	0.56±0.11	0.51±0.06
		0.1	0.64±0.22	0.70±0.25	0.25±0.30	0.39±0.18	0.18±0.16	0.15±0.16	**0.83**±0.07	0.48±0.11	0.51±0.07
		0.5	0.64±0.19	0.71±0.24	0.23±0.28	0.37±0.18	0.24±0.12	0.17±0.17	**0.76**±0.12	0.54±0.17	0.50±0.07
		1.0	0.59±0.21	**0.79**±0.22	0.21±0.27	0.38±0.18	0.26±0.11	0.15±0.16	0.74±0.12	0.42±0.20	0.50±0.11
	0.5	0.0	0.61±0.19	**0.82**±0.17	0.41±0.34	0.35±0.19	0.09±0.17	0.20±0.16	0.76±0.08	0.68±0.10	0.49±0.07
		0.1	0.64±0.19	**0.76**±0.18	0.42±0.32	0.34±0.19	0.09±0.17	0.23±0.17	0.75±0.12	0.69±0.10	0.53±0.04
		0.5	0.61±0.20	**0.80**±0.16	0.43±0.32	0.32±0.21	0.20±0.16	0.22±0.16	0.71±0.10	0.67±0.13	0.48±0.09
		1.0	0.61±0.21	**0.80**±0.16	0.37±0.32	0.33±0.22	0.21±0.14	0.22±0.16	0.73±0.11	0.50±0.14	0.50±0.05
	1.0	0.0	0.57±0.18	**0.78**±0.18	0.62±0.28	0.32±0.20	0.11±0.17	0.22±0.16	0.70±0.13	0.60±0.12	0.51±0.08
		0.1	0.57±0.18	**0.76**±0.18	0.61±0.29	0.28±0.21	0.13±0.19	0.21±0.17	0.67±0.14	0.57±0.12	0.52±0.07
		0.5	0.56±0.19	**0.76**±0.17	0.55±0.32	0.25±0.21	0.23±0.17	0.24±0.18	0.71±0.12	0.61±0.13	0.50±0.08
		1.0	0.58±0.19	**0.77**±0.19	0.60±0.29	0.28±0.21	0.23±0.16	0.25±0.19	0.67±0.16	0.59±0.13	0.50±0.08
$\widehat{\text{OU}}(\sigma_g^m > 0)$	0.2	0.0	0.29±0.12	**0.74**±0.21	0.37±0.24	0.12±0.21	0.02±0.08	0.25±0.12	0.49±0.08	0.43±0.05	0.44±0.08
		0.1	0.30±0.12	**0.74**±0.21	0.36±0.24	0.12±0.20	0.03±0.13	0.26±0.12	0.50±0.09	0.46±0.04	0.44±0.07
		0.5	0.43±0.19	**0.77**±0.20	0.31±0.26	0.28±0.22	0.02±0.07	0.19±0.14	0.64±0.13	0.44±0.17	0.49±0.09
		1.0	0.54±0.22	0.77±0.23	0.25±0.30	0.35±0.18	0.02±0.10	0.17±0.15	**0.79**±0.14	0.52±0.15	0.48±0.07
	0.5	0.0	0.29±0.12	**0.82**±0.16	0.46±0.20	0.07±0.16	0.01±0.06	0.26±0.11	0.49±0.07	0.48±0.08	0.44±0.07
		0.1	0.31±0.12	**0.80**±0.17	0.46±0.21	0.11±0.20	0.01±0.07	0.26±0.12	0.50±0.08	0.47±0.09	0.45±0.07
		0.5	0.42±0.16	**0.80**±0.18	0.47±0.28	0.21±0.20	0.00±0.03	0.22±0.14	0.61±0.08	0.57±0.10	0.48±0.08
		1.0	0.53±0.19	**0.80**±0.18	0.44±0.33	0.29±0.22	0.01±0.05	0.22±0.15	0.69±0.15	0.56±0.13	0.49±0.10
	1.0	0.0	0.29±0.11	**0.78**±0.17	0.51±0.21	0.07±0.15	0.01±0.05	0.27±0.12	0.48±0.09	0.46±0.08	0.44±0.08
		0.1	0.30±0.13	**0.77**±0.17	0.52±0.22	0.09±0.17	0.02±0.11	0.26±0.13	0.48±0.10	0.47±0.08	0.44±0.08
		0.5	0.43±0.17	**0.77**±0.17	0.58±0.26	0.22±0.22	0.00±0.03	0.25±0.15	0.55±0.10	0.56±0.11	0.46±0.09
		1.0	0.52±0.18	**0.78**±0.17	0.60±0.29	0.22±0.20	0.01±0.05	0.25±0.18	0.66±0.12	0.57±0.10	0.50±0.08

Table 6.6: TNR Score - $N = 5,\ \sigma_g^a = 1.0$

Dataset	C	σ_b	Granger	PLACy	PCMCI	RCV-VarLiNGAM	DYNOTEARS	PCMCI$_\Omega$	BCGeweke	DTF	GewekeNP
OU($\sigma_g^m = 0$)	0.2	0.0	**0.96**±0.04	0.96±0.04	**0.99**±0.02	0.72±0.06	0.94±0.08	0.78±0.09	0.86±0.08	0.93±0.05	0.60±0.11
		0.1	**0.84**±0.09	0.96±0.04	**0.99**±0.03	0.72±0.06	0.82±0.16	0.79±0.08	0.71±0.17	0.89±0.05	0.58±0.09
		0.5	0.63±0.13	**0.96**±0.04	0.95±0.05	0.77±0.08	0.52±0.18	0.77±0.09	0.35±0.07	0.61±0.11	**0.60**±0.13
		1.0	0.58±0.14	**0.96**±0.04	**0.96**±0.04	0.82±0.07	0.49±0.15	0.78±0.09	0.38±0.07	0.35±0.09	**0.61**±0.08
	0.5	0.0	0.94±0.05	**0.95**±0.04	**0.99**±0.02	0.74±0.07	0.96±0.05	0.76±0.11	0.82±0.06	0.87±0.09	0.46±0.13
		0.1	0.83±0.12	**0.96**±0.04	**0.99**±0.02	0.76±0.07	0.97±0.06	0.75±0.12	0.68±0.08	0.80±0.08	0.45±0.12
		0.5	0.63±0.15	**0.95**±0.05	**0.95**±0.04	0.83±0.06	0.89±0.13	0.76±0.10	0.35±0.11	0.61±0.12	0.49±0.11
		1.0	0.54±0.12	**0.93**±0.05	**0.94**±0.05	0.84±0.06	0.91±0.10	0.75±0.10	0.32±0.05	0.40±0.11	0.44±0.15
	1.0	0.0	**0.92**±0.08	0.91±0.07	**0.99**±0.02	0.75±0.08	0.91±0.08	0.76±0.11	0.80±0.09	0.70±0.14	0.48±0.14
		0.1	0.81±0.13	**0.92**±0.07	**0.99**±0.02	0.76±0.08	0.94±0.06	0.76±0.12	0.63±0.12	0.70±0.13	0.47±0.13
		0.5	0.57±0.16	**0.92**±0.07	**0.95**±0.04	0.85±0.08	0.90±0.12	0.75±0.11	0.31±0.07	0.45±0.13	0.45±0.11
		1.0	0.50±0.15	**0.90**±0.08	**0.93**±0.05	0.87±0.08	0.89±0.15	0.73±0.12	0.29±0.06	0.35±0.09	0.48±0.15
$\widetilde{\text{OU}}(\sigma_g^m = 0)$	0.2	0.0	**0.96**±0.04	0.95±0.05	1.00±0.02	0.72±0.06	**1.00**±0.00	0.78±0.09	0.89±0.07	0.26±0.02	0.26±0.02
		0.1	0.68±0.13	0.89±0.07	0.98±0.03	0.73±0.06	**1.00**±0.00	0.78±0.09	**0.57**±0.09	0.27±0.02	0.26±0.02
		0.5	0.26±0.04	0.94±0.06	0.93±0.05	0.77±0.07	**0.98**±0.03	0.77±0.09	0.30±0.05	0.28±0.04	**0.43**±0.12
		1.0	0.31±0.07	0.94±0.05	0.95±0.05	0.81±0.07	**0.96**±0.04	0.77±0.08	0.33±0.07	0.26±0.02	**0.58**±0.14
	0.5	0.0	**0.94**±0.06	0.93±0.05	1.00±0.01	0.75±0.07	**1.00**±0.00	0.77±0.10	0.85±0.06	0.26±0.02	0.26±0.02
		0.1	0.67±0.13	**0.90**±0.06	0.98±0.02	0.76±0.08	**1.00**±0.00	0.76±0.10	0.47±0.11	0.27±0.02	0.26±0.02
		0.5	0.26±0.04	**0.92**±0.07	0.94±0.05	0.81±0.06	**0.99**±0.03	0.76±0.09	0.30±0.03	0.28±0.03	0.33±0.08
		1.0	0.30±0.07	**0.92**±0.06	**0.95**±0.04	0.83±0.07	0.95±0.05	0.76±0.10	0.32±0.07	0.27±0.03	0.41±0.12
	1.0	0.0	**0.92**±0.07	0.89±0.09	1.00±0.01	0.77±0.08	**1.00**±0.00	0.76±0.11	0.81±0.09	0.26±0.02	0.26±0.02
		0.1	0.67±0.14	**0.86**±0.09	0.98±0.03	0.77±0.08	**1.00**±0.00	0.75±0.13	0.51±0.11	0.26±0.02	0.26±0.02
		0.5	0.26±0.05	**0.88**±0.09	0.93±0.05	0.83±0.08	**0.98**±0.04	0.75±0.11	0.28±0.04	0.26±0.02	0.34±0.09
		1.0	0.30±0.06	**0.88**±0.09	**0.94**±0.05	0.87±0.08	0.93±0.06	0.74±0.12	0.32±0.05	0.26±0.02	0.46±0.15
OU($\sigma_g^m > 0$)	0.2	0.0	0.87±0.09	**0.96**±0.04	**0.98**±0.03	0.75±0.07	0.64±0.15	0.76±0.10	0.92±0.06	0.92±0.09	0.61±0.14
		0.1	0.87±0.09	**0.96**±0.04	**0.98**±0.03	0.76±0.08	0.66±0.17	0.76±0.10	0.90±0.06	0.91±0.04	0.64±0.12
		0.5	0.87±0.09	**0.95**±0.05	**0.98**±0.03	0.76±0.07	0.44±0.15	0.76±0.11	0.84±0.13	0.89±0.08	0.67±0.10
		1.0	0.86±0.10	**0.96**±0.05	**0.98**±0.04	0.77±0.08	0.35±0.13	0.76±0.11	0.86±0.08	0.93±0.06	0.64±0.11
	0.5	0.0	0.82±0.12	**0.94**±0.05	**0.98**±0.03	0.77±0.08	0.88±0.10	0.74±0.12	0.80±0.13	0.81±0.12	0.43±0.15
		0.1	0.83±0.13	**0.92**±0.07	**0.98**±0.03	0.77±0.08	0.87±0.10	0.74±0.12	0.77±0.18	0.79±0.14	0.57±0.17
		0.5	0.82±0.14	**0.94**±0.06	**0.97**±0.04	0.79±0.07	0.69±0.15	0.74±0.11	0.78±0.10	0.79±0.09	0.40±0.11
		1.0	0.82±0.14	**0.93**±0.07	**0.97**±0.04	0.79±0.08	0.65±0.16	0.74±0.12	0.81±0.08	0.77±0.12	0.52±0.20
	1.0	0.0	0.78±0.15	0.91±0.08	**0.97**±0.04	0.79±0.09	0.84±0.11	0.73±0.12	0.75±0.17	0.66±0.18	0.45±0.15
		0.1	0.78±0.16	0.91±0.08	**0.97**±0.04	0.80±0.08	0.83±0.12	0.73±0.12	0.72±0.16	0.68±0.18	0.51±0.17
		0.5	0.77±0.16	0.90±0.09	**0.96**±0.06	0.81±0.08	0.69±0.14	0.74±0.12	0.77±0.13	0.73±0.15	0.53±0.13
		1.0	0.79±0.15	0.91±0.09	**0.97**±0.04	0.82±0.09	0.63±0.15	0.74±0.13	0.70±0.20	0.67±0.20	0.48±0.15
$\widetilde{\text{OU}}(\sigma_g^m > 0)$	0.2	0.0	0.35±0.14	**0.96**±0.05	0.82±0.10	0.95±0.07	**0.97**±0.05	0.37±0.14	0.39±0.11	0.41±0.13	0.30±0.05
		0.1	0.38±0.15	**0.95**±0.04	0.84±0.08	0.95±0.06	**0.97**±0.05	0.41±0.16	0.43±0.12	0.47±0.12	0.31±0.06
		0.5	0.65±0.17	**0.95**±0.05	0.94±0.06	0.85±0.09	**0.96**±0.05	0.62±0.16	0.67±0.15	0.75±0.14	0.46±0.16
		1.0	0.80±0.12	**0.96**±0.04	**0.97**±0.04	0.79±0.08	0.94±0.06	0.71±0.12	0.85±0.10	0.83±0.12	0.54±0.19
	0.5	0.0	0.35±0.14	**0.94**±0.06	0.82±0.10	0.96±0.06	**0.97**±0.05	0.36±0.14	0.39±0.10	0.40±0.10	0.28±0.04
		0.1	0.39±0.16	**0.93**±0.06	0.83±0.09	0.95±0.07	**0.97**±0.05	0.38±0.14	0.42±0.11	0.40±0.10	0.30±0.06
		0.5	0.60±0.19	**0.93**±0.06	0.94±0.05	0.86±0.08	**0.96**±0.05	0.61±0.17	0.62±0.11	0.64±0.16	0.39±0.12
		1.0	0.75±0.16	**0.93**±0.07	**0.96**±0.04	0.82±0.08	0.95±0.06	0.70±0.13	0.73±0.13	0.73±0.18	0.42±0.16
	1.0	0.0	0.35±0.15	**0.91**±0.09	0.81±0.10	0.97±0.06	**0.97**±0.05	0.36±0.14	0.39±0.12	0.36±0.10	0.31±0.09
		0.1	0.37±0.17	**0.91**±0.08	0.83±0.10	0.95±0.06	**0.97**±0.05	0.39±0.15	0.40±0.13	0.38±0.12	0.30±0.09
		0.5	0.60±0.20	0.91±0.08	**0.93**±0.06	0.87±0.08	**0.96**±0.06	0.60±0.16	0.53±0.15	0.58±0.18	0.36±0.11
		1.0	0.73±0.17	0.91±0.08	**0.97**±0.04	0.84±0.08	0.93±0.06	0.69±0.14	0.70±0.14	0.61±0.16	0.44±0.15

Table 6.7: F1 Score - $N = 10$, $\sigma_g^a = 0.5$

Dataset	C	σ_b	Granger	PLACy	PCMCI	RCV-VarLiNGAM	DYNOTEARS	PCMCI$_\Omega$	BCGeweke	DTF	GewekeNP
OU($\sigma_g^m = 0$)	0.2	0.0	**0.87**±0.07	0.17±0.12	0.05±0.09	0.46±0.08	0.02±0.05	0.07±0.07	0.78±0.02	0.42±0.14	0.54±0.02
		0.1	0.46±0.09	0.22±0.13	0.09±0.10	0.45±0.08	0.12±0.09	0.08±0.06	0.50±0.06	0.33±0.05	**0.55**±0.03
		0.5	0.41±0.07	0.16±0.11	0.24±0.10	0.40±0.09	0.23±0.08	0.10±0.07	0.43±0.04	0.36±0.05	**0.54**±0.01
		1.0	0.51±0.09	0.13±0.10	0.11±0.10	0.41±0.09	0.21±0.09	0.12±0.06	0.50±0.05	0.39±0.06	**0.55**±0.02
	0.5	0.0	0.67±0.08	**0.80**±0.07	0.16±0.11	0.43±0.08	0.01±0.02	0.10±0.06	0.62±0.05	0.53±0.02	0.43±0.03
		0.1	0.41±0.07	**0.76**±0.07	0.12±0.11	0.41±0.09	0.04±0.07	0.10±0.06	0.47±0.05	0.39±0.08	0.43±0.02
		0.5	0.36±0.06	**0.63**±0.10	0.29±0.13	0.36±0.09	0.07±0.10	0.14±0.06	0.42±0.03	0.39±0.04	0.43±0.02
		1.0	0.41±0.06	0.44±0.12	0.19±0.12	0.36±0.11	0.08±0.10	0.20±0.06	**0.46**±0.02	0.41±0.04	0.44±0.02
	1.0	0.0	**0.59**±0.09	**0.59**±0.09	0.38±0.18	0.38±0.08	0.00±0.02	0.12±0.06	0.58±0.06	0.46±0.04	0.41±0.03
		0.1	0.40±0.08	**0.59**±0.09	0.21±0.14	0.38±0.08	0.04±0.07	0.12±0.06	0.44±0.03	0.41±0.04	0.42±0.03
		0.5	0.33±0.06	**0.58**±0.09	0.32±0.14	0.31±0.10	0.12±0.11	0.19±0.07	0.41±0.03	0.40±0.03	0.41±0.04
		1.0	0.36±0.06	**0.53**±0.09	0.23±0.14	0.28±0.10	0.13±0.12	0.24±0.07	0.42±0.03	0.39±0.04	0.41±0.03
$\widehat{\text{OU}}(\sigma_g^m = 0)$	0.2	0.0	**0.87**±0.07	0.26±0.12	0.06±0.09	0.46±0.09	0.00±0.00	0.07±0.06	0.77±0.06	0.40±0.03	0.40±0.03
		0.1	0.34±0.06	0.35±0.10	0.12±0.11	**0.46**±0.08	0.00±0.00	0.07±0.06	0.45±0.03	0.40±0.03	0.40±0.03
		0.5	0.28±0.05	0.20±0.11	0.25±0.14	0.41±0.10	0.01±0.04	0.11±0.07	0.42±0.04	0.40±0.03	**0.43**±0.04
		1.0	0.37±0.05	0.19±0.11	0.15±0.11	0.40±0.09	0.04±0.06	0.12±0.06	**0.50**±0.02	0.40±0.03	0.47±0.01
	0.5	0.0	0.67±0.07	**0.73**±0.07	0.17±0.13	0.44±0.08	0.00±0.00	0.10±0.06	0.61±0.07	0.40±0.03	0.40±0.03
		0.1	0.33±0.05	**0.57**±0.07	0.15±0.12	0.42±0.09	0.00±0.00	0.11±0.06	0.44±0.04	0.40±0.03	0.40±0.03
		0.5	0.28±0.05	**0.57**±0.09	0.28±0.12	0.35±0.10	0.00±0.01	0.15±0.07	0.42±0.03	0.40±0.03	0.41±0.03
		1.0	0.34±0.04	0.45±0.10	0.18±0.13	0.33±0.09	0.02±0.05	0.21±0.07	**0.46**±0.03	0.40±0.03	0.43±0.02
	1.0	0.0	**0.59**±0.09	0.54±0.08	0.38±0.19	0.37±0.08	0.00±0.00	0.13±0.06	0.58±0.06	0.39±0.03	0.39±0.03
		0.1	0.32±0.06	**0.47**±0.07	0.22±0.13	0.36±0.08	0.00±0.00	0.13±0.07	0.42±0.03	0.39±0.03	0.39±0.03
		0.5	0.28±0.05	**0.53**±0.08	0.29±0.14	0.32±0.11	0.00±0.02	0.18±0.06	0.41±0.03	0.39±0.03	0.40±0.03
		1.0	0.32±0.05	**0.48**±0.07	0.21±0.12	0.30±0.11	0.01±0.03	0.24±0.06	0.42±0.03	0.39±0.03	0.41±0.03
OU($\sigma_g^m > 0$)	0.2	0.0	0.74±0.09	0.55±0.14	0.13±0.13	0.44±0.07	0.15±0.09	0.11±0.06	**0.75**±0.07	0.40±0.11	0.51±0.05
		0.1	0.68±0.10	0.63±0.13	0.15±0.11	0.43±0.07	0.14±0.08	0.13±0.07	**0.74**±0.08	0.38±0.09	0.47±0.05
		0.5	0.66±0.09	0.68±0.11	0.14±0.12	0.44±0.08	0.20±0.07	0.14±0.06	**0.68**±0.05	0.38±0.09	0.49±0.04
		1.0	0.66±0.09	0.65±0.12	0.11±0.13	0.43±0.07	0.22±0.07	0.12±0.07	**0.72**±0.03	0.40±0.15	0.43±0.03
	0.5	0.0	0.60±0.09	**0.72**±0.09	0.31±0.19	0.42±0.07	0.03±0.05	0.16±0.07	0.62±0.07	0.52±0.05	0.44±0.03
		0.1	0.57±0.08	**0.69**±0.09	0.30±0.15	0.40±0.08	0.02±0.05	0.17±0.07	0.66±0.06	0.48±0.02	0.44±0.04
		0.5	0.56±0.08	**0.69**±0.08	0.28±0.16	0.39±0.09	0.08±0.09	0.17±0.07	0.64±0.08	0.54±0.03	0.46±0.02
		1.0	0.57±0.10	**0.68**±0.08	0.28±0.16	0.39±0.10	0.08±0.08	0.18±0.07	0.61±0.04	0.48±0.05	0.45±0.01
	1.0	0.0	0.53±0.10	**0.58**±0.09	0.54±0.17	0.34±0.09	0.05±0.06	0.18±0.08	0.57±0.05	0.47±0.05	0.42±0.03
		0.1	0.51±0.10	**0.58**±0.08	0.51±0.16	0.35±0.10	0.03±0.05	0.18±0.07	0.57±0.07	0.48±0.05	0.42±0.04
		0.5	0.48±0.10	**0.58**±0.09	0.51±0.16	0.32±0.10	0.07±0.08	0.21±0.07	0.53±0.06	0.45±0.04	0.42±0.03
		1.0	0.49±0.08	**0.58**±0.08	0.47±0.16	0.31±0.11	0.09±0.09	0.21±0.07	0.55±0.05	0.46±0.06	0.41±0.04
$\widehat{\text{OU}}(\sigma_g^m > 0)$	0.2	0.0	0.28±0.05	**0.59**±0.14	0.34±0.12	0.12±0.13	0.03±0.06	0.26±0.05	0.43±0.03	0.42±0.03	0.40±0.04
		0.1	0.28±0.05	**0.67**±0.12	0.33±0.13	0.14±0.13	0.02±0.06	0.26±0.05	0.45±0.06	0.44±0.03	0.40±0.03
		0.5	0.42±0.09	**0.68**±0.11	0.19±0.12	0.39±0.09	0.03±0.06	0.20±0.07	0.58±0.08	0.46±0.03	0.44±0.03
		1.0	0.60±0.11	0.66±0.10	0.12±0.12	0.44±0.08	0.05±0.07	0.16±0.07	**0.66**±0.07	0.45±0.04	0.48±0.03
	0.5	0.0	0.28±0.05	**0.70**±0.09	0.48±0.13	0.10±0.12	0.01±0.04	0.27±0.05	0.42±0.04	0.42±0.03	0.40±0.03
		0.1	0.28±0.05	**0.69**±0.09	0.47±0.13	0.14±0.13	0.01±0.03	0.27±0.05	0.44±0.03	0.42±0.03	0.40±0.03
		0.5	0.39±0.08	**0.70**±0.09	0.40±0.16	0.34±0.10	0.01±0.03	0.22±0.06	0.56±0.05	0.43±0.03	0.42±0.02
		1.0	0.53±0.08	**0.69**±0.10	0.27±0.17	0.38±0.10	0.01±0.03	0.18±0.07	0.60±0.02	0.48±0.05	0.44±0.01
	1.0	0.0	0.28±0.05	**0.59**±0.09	0.56±0.10	0.08±0.10	0.01±0.03	0.26±0.05	0.41±0.03	0.40±0.03	0.39±0.03
		0.1	0.28±0.05	**0.59**±0.09	0.56±0.09	0.11±0.10	0.01±0.03	0.27±0.05	0.41±0.03	0.40±0.03	0.39±0.03
		0.5	0.36±0.07	**0.60**±0.10	0.54±0.13	0.27±0.10	0.00±0.02	0.24±0.06	0.48±0.06	0.42±0.03	0.40±0.03
		1.0	0.47±0.08	**0.58**±0.09	0.47±0.15	0.29±0.09	0.01±0.02	0.21±0.06	0.54±0.06	0.43±0.04	0.41±0.04

Table 6.8: TNR Score - $N = 10$, $\sigma_g^a = 0.5$

Dataset	C	σ_b	Granger	PLACy	PCMCI	RCV-VarLiNGAM	DYNOTEARS	PCMCI$_\Omega$	BCGeweke	DTF	GewekeNP
OU($\sigma_g^m = 0$)	0.2	0.0	**0.95**±0.02	0.95±0.02	**1.00**±0.01	0.69±0.04	0.97±0.02	0.72±0.06	0.83±0.04	0.83±0.06	0.50±0.07
		0.1	0.61±0.09	0.95±0.03	**0.98**±0.01	0.69±0.04	0.85±0.09	0.73±0.06	0.41±0.12	0.74±0.09	**0.53**±0.10
		0.5	0.54±0.08	0.95±0.02	**0.97**±0.02	0.77±0.04	0.65±0.12	0.72±0.06	0.22±0.05	0.41±0.08	**0.52**±0.08
		1.0	**0.68**±0.08	0.93±0.03	**0.99**±0.01	0.80±0.04	0.64±0.12	0.72±0.06	0.41±0.06	0.21±0.04	0.52±0.10
	0.5	0.0	0.83±0.08	**0.93**±0.04	**0.99**±0.01	0.75±0.05	0.93±0.04	0.61±0.10	0.63±0.11	0.54±0.08	0.23±0.07
		0.1	0.53±0.12	**0.93**±0.04	**0.98**±0.02	0.78±0.05	0.95±0.05	0.62±0.10	0.32±0.17	0.56±0.12	0.23±0.07
		0.5	0.42±0.11	**0.93**±0.04	0.95±0.02	0.82±0.04	**0.96**±0.07	0.63±0.09	0.18±0.06	0.23±0.07	0.22±0.06
		1.0	0.52±0.10	0.90±0.04	**0.98**±0.02	0.84±0.04	0.97±0.05	0.61±0.09	0.30±0.08	0.19±0.04	0.24±0.09
	1.0	0.0	0.76±0.11	**0.77**±0.11	**0.98**±0.02	0.77±0.05	0.88±0.05	0.61±0.10	0.59±0.09	0.34±0.12	0.20±0.07
		0.1	0.50±0.15	**0.78**±0.11	**0.97**±0.02	0.80±0.05	0.89±0.05	0.58±0.11	0.28±0.09	0.29±0.09	0.21±0.08
		0.5	0.34±0.12	**0.79**±0.12	**0.94**±0.03	0.85±0.05	0.93±0.10	0.58±0.10	0.18±0.04	0.19±0.06	0.19±0.07
		1.0	0.42±0.13	**0.78**±0.12	**0.97**±0.02	0.89±0.05	0.94±0.07	0.55±0.11	0.21±0.06	0.15±0.03	0.20±0.07
$\widetilde{\text{OU}}(\sigma_g^m = 0)$	0.2	0.0	**0.95**±0.03	0.92±0.03	**1.00**±0.00	0.69±0.04	**1.00**±0.00	0.73±0.06	0.83±0.06	0.13±0.00	0.13±0.00
		0.1	0.37±0.07	0.83±0.04	0.98±0.02	0.69±0.04	**1.00**±0.00	0.73±0.06	0.29±0.09	0.13±0.00	0.13±0.00
		0.5	0.15±0.03	0.90±0.04	0.97±0.02	0.76±0.04	**0.99**±0.01	0.72±0.05	0.19±0.02	0.13±0.00	0.23±0.07
		1.0	0.45±0.08	0.90±0.03	**0.99**±0.01	0.79±0.04	0.95±0.02	0.72±0.05	0.40±0.08	0.13±0.00	0.35±0.11
	0.5	0.0	0.83±0.08	**0.88**±0.05	1.00±0.01	0.76±0.05	**1.00**±0.00	0.61±0.10	0.62±0.11	0.13±0.00	0.13±0.00
		0.1	0.32±0.08	**0.78**±0.06	0.96±0.02	0.77±0.05	**1.00**±0.00	0.62±0.10	0.25±0.07	0.13±0.00	0.13±0.00
		0.5	0.15±0.03	**0.87**±0.05	0.95±0.02	0.82±0.04	**0.99**±0.02	0.64±0.10	0.20±0.04	0.13±0.00	0.16±0.05
		1.0	0.35±0.09	0.86±0.05	**0.98**±0.02	0.85±0.04	0.94±0.03	0.61±0.10	0.31±0.08	0.13±0.00	0.23±0.08
	1.0	0.0	**0.76**±0.11	0.73±0.11	0.98±0.02	0.79±0.06	**1.00**±0.00	0.61±0.10	0.58±0.09	0.13±0.00	0.13±0.00
		0.1	0.31±0.09	0.64±0.11	0.96±0.03	0.80±0.05	**1.00**±0.00	0.58±0.11	0.23±0.07	0.13±0.00	0.13±0.00
		0.5	0.16±0.03	**0.74**±0.11	0.94±0.03	0.84±0.05	**0.99**±0.02	0.57±0.11	0.18±0.03	0.13±0.00	0.14±0.02
		1.0	0.31±0.09	0.71±0.13	**0.97**±0.02	0.89±0.05	0.93±0.04	0.56±0.11	0.22±0.06	0.13±0.00	0.19±0.07
OU($\sigma_g^m > 0$)	0.2	0.0	0.88±0.05	**0.95**±0.02	**0.99**±0.01	0.71±0.04	0.77±0.06	0.70±0.07	0.83±0.05	0.82±0.09	0.49±0.12
		0.1	0.85±0.06	**0.95**±0.02	**0.99**±0.01	0.72±0.05	0.79±0.09	0.68±0.08	0.83±0.05	0.80±0.10	0.48±0.11
		0.5	0.83±0.06	**0.95**±0.03	**0.98**±0.01	0.72±0.04	0.61±0.17	0.68±0.07	0.76±0.03	0.84±0.06	0.42±0.15
		1.0	0.84±0.06	**0.95**±0.03	**0.99**±0.01	0.72±0.04	0.58±0.16	0.68±0.07	0.82±0.05	0.87±0.05	0.38±0.14
	0.5	0.0	0.77±0.11	**0.88**±0.06	**0.99**±0.01	0.75±0.05	0.92±0.04	0.61±0.10	0.65±0.09	0.60±0.13	0.30±0.11
		0.1	0.75±0.09	**0.87**±0.07	**0.98**±0.02	0.76±0.06	0.94±0.05	0.60±0.11	0.69±0.09	0.62±0.16	0.29±0.11
		0.5	0.72±0.11	**0.87**±0.06	**0.98**±0.01	0.77±0.05	0.85±0.10	0.60±0.10	0.66±0.12	0.63±0.14	0.30±0.12
		1.0	0.73±0.13	**0.86**±0.07	**0.98**±0.02	0.77±0.05	0.84±0.12	0.59±0.10	0.61±0.12	0.63±0.16	0.28±0.11
	1.0	0.0	0.68±0.15	0.77±0.11	**0.96**±0.04	0.79±0.05	0.87±0.06	0.60±0.10	0.57±0.10	0.39±0.12	0.22±0.08
		0.1	0.65±0.15	0.77±0.10	**0.96**±0.04	0.80±0.05	0.88±0.06	0.59±0.11	0.57±0.11	0.43±0.13	0.22±0.09
		0.5	0.62±0.17	0.77±0.11	**0.96**±0.03	0.81±0.05	0.83±0.11	0.57±0.10	0.50±0.12	0.36±0.13	0.21±0.08
		1.0	0.64±0.14	**0.77**±0.10	**0.96**±0.04	0.82±0.05	0.80±0.14	0.57±0.11	0.54±0.10	0.39±0.15	0.22±0.06
$\widetilde{\text{OU}}(\sigma_g^m > 0)$	0.2	0.0	0.17±0.07	**0.94**±0.03	0.88±0.04	0.96±0.04	**0.97**±0.03	0.18±0.06	0.24±0.06	0.25±0.07	0.13±0.01
		0.1	0.18±0.07	**0.95**±0.03	0.90±0.04	0.95±0.05	**0.97**±0.03	0.20±0.07	0.30±0.12	0.29±0.10	0.14±0.02
		0.5	0.55±0.12	**0.95**±0.03	**0.96**±0.02	0.78±0.06	0.96±0.03	0.49±0.10	0.63±0.08	0.56±0.09	0.34±0.11
		1.0	0.79±0.08	**0.95**±0.02	**0.98**±0.02	0.73±0.05	0.94±0.04	0.65±0.08	0.73±0.03	0.69±0.09	0.43±0.11
	0.5	0.0	0.16±0.07	**0.88**±0.06	0.88±0.04	0.96±0.04	**0.98**±0.03	0.16±0.05	0.21±0.05	0.18±0.03	0.13±0.00
		0.1	0.17±0.07	**0.87**±0.07	0.90±0.04	0.95±0.05	**0.98**±0.03	0.18±0.06	0.26±0.06	0.18±0.03	0.14±0.02
		0.5	0.46±0.14	**0.87**±0.07	**0.97**±0.02	0.81±0.05	0.96±0.04	0.44±0.11	0.54±0.07	0.35±0.06	0.21±0.05
		1.0	0.70±0.10	**0.87**±0.07	**0.98**±0.01	0.78±0.05	0.93±0.04	0.57±0.11	0.61±0.08	0.48±0.13	0.26±0.08
	1.0	0.0	0.15±0.07	0.78±0.10	**0.86**±0.05	0.97±0.04	**0.98**±0.03	0.16±0.05	0.19±0.04	0.15±0.02	0.13±0.00
		0.1	0.16±0.08	0.78±0.11	**0.87**±0.05	0.96±0.04	**0.98**±0.03	0.19±0.07	0.20±0.05	0.14±0.01	0.13±0.00
		0.5	0.39±0.14	0.78±0.11	**0.94**±0.04	0.85±0.05	**0.96**±0.04	0.41±0.12	0.39±0.11	0.22±0.05	0.16±0.04
		1.0	0.61±0.14	**0.77**±0.10	**0.96**±0.03	0.82±0.05	0.92±0.05	0.54±0.11	0.51±0.11	0.29±0.06	0.18±0.07

Table 6.9: F1 Score - $N = 10$, $\sigma_g^a = 1.0$

Dataset	C	σ_b	Granger	PLACy	PCMCI	RCV-VarLiNGAM	DYNOTEARS	PCMCI$_\Omega$	BCGeweke	DTF	GewekeNP
OU($\sigma_g^m = 0$)	0.2	0.0	**0.88**±0.07	0.16±0.11	0.06±0.09	0.46±0.08	0.08±0.09	0.07±0.06	0.78±0.02	0.43±0.14	0.55±0.01
		0.1	**0.65**±0.10	0.17±0.11	0.06±0.08	0.46±0.08	0.13±0.09	0.07±0.06	0.62±0.05	0.33±0.09	0.55±0.01
		0.5	0.46±0.09	0.17±0.11	0.20±0.13	0.44±0.08	0.21±0.09	0.09±0.06	0.46±0.04	0.36±0.05	**0.56**±0.02
		1.0	0.41±0.07	0.16±0.12	0.24±0.12	0.41±0.09	0.23±0.09	0.11±0.06	0.44±0.03	0.39±0.06	**0.55**±0.01
	0.5	0.0	0.67±0.08	**0.80**±0.07	0.16±0.14	0.43±0.08	0.03±0.06	0.10±0.06	0.62±0.06	0.53±0.03	0.43±0.03
		0.1	0.55±0.08	**0.79**±0.07	0.10±0.11	0.43±0.09	0.04±0.07	0.10±0.06	0.57±0.06	0.49±0.05	0.44±0.02
		0.5	0.39±0.07	**0.73**±0.07	0.28±0.13	0.38±0.09	0.07±0.09	0.12±0.06	0.44±0.02	0.37±0.06	0.44±0.02
		1.0	0.36±0.05	**0.60**±0.11	0.27±0.13	0.36±0.11	0.07±0.10	0.14±0.06	0.44±0.03	0.41±0.04	0.43±0.03
	1.0	0.0	0.59±0.09	0.59±0.09	0.38±0.19	0.38±0.09	0.03±0.07	0.12±0.06	**0.59**±0.06	0.47±0.04	0.41±0.03
		0.1	0.53±0.08	**0.60**±0.09	0.24±0.16	0.38±0.08	0.05±0.08	0.13±0.06	0.52±0.04	0.44±0.05	0.41±0.03
		0.5	0.37±0.06	**0.59**±0.09	0.31±0.13	0.31±0.09	0.13±0.13	0.14±0.07	0.40±0.03	0.39±0.07	0.41±0.03
		1.0	0.34±0.05	**0.58**±0.09	0.30±0.12	0.32±0.10	0.14±0.11	0.19±0.07	0.40±0.03	0.40±0.03	0.41±0.03
$\widetilde{\text{OU}}(\sigma_g^m = 0)$	0.2	0.0	**0.87**±0.07	0.24±0.13	0.05±0.08	0.45±0.09	0.00±0.00	0.07±0.06	0.75±0.05	0.41±0.04	0.40±0.04
		0.1	0.49±0.09	0.35±0.11	0.07±0.09	0.45±0.08	0.00±0.00	0.08±0.06	**0.53**±0.02	0.41±0.03	0.41±0.04
		0.5	0.28±0.05	0.25±0.12	0.28±0.12	**0.44**±0.08	0.01±0.03	0.09±0.06	0.43±0.03	0.41±0.04	0.43±0.02
		1.0	0.29±0.05	0.20±0.10	0.25±0.13	0.39±0.09	0.04±0.07	0.10±0.06	0.44±0.03	0.41±0.04	**0.50**±0.01
	0.5	0.0	0.67±0.07	**0.75**±0.07	0.17±0.13	0.43±0.08	0.00±0.00	0.10±0.06	0.58±0.02	0.41±0.04	0.40±0.04
		0.1	0.43±0.09	**0.65**±0.06	0.10±0.12	0.43±0.09	0.00±0.00	0.10±0.06	0.52±0.02	0.41±0.04	0.41±0.04
		0.5	0.28±0.05	**0.69**±0.08	0.29±0.12	0.38±0.10	0.00±0.00	0.11±0.07	0.42±0.04	0.40±0.04	0.42±0.03
		1.0	0.29±0.05	**0.59**±0.09	0.29±0.14	0.34±0.11	0.02±0.04	0.14±0.06	0.44±0.03	0.41±0.04	0.44±0.02
	1.0	0.0	**0.59**±0.09	0.56±0.08	0.38±0.19	0.37±0.08	0.00±0.00	0.13±0.06	0.58±0.06	0.39±0.03	0.39±0.03
		0.1	0.42±0.08	**0.51**±0.07	0.23±0.17	0.36±0.08	0.00±0.00	0.14±0.06	0.47±0.05	0.39±0.03	0.39±0.03
		0.5	0.28±0.05	**0.56**±0.08	0.30±0.14	0.32±0.09	0.01±0.03	0.13±0.07	0.40±0.03	0.39±0.03	0.40±0.03
		1.0	0.29±0.05	**0.54**±0.08	0.31±0.12	0.29±0.11	0.01±0.03	0.19±0.06	0.41±0.03	0.39±0.03	0.41±0.04
OU($\sigma_g^m > 0$)	0.2	0.0	0.64±0.11	**0.69**±0.11	0.26±0.16	0.43±0.08	0.17±0.09	0.15±0.08	0.66±0.08	0.48±0.09	0.46±0.03
		0.1	0.61±0.11	**0.70**±0.10	0.32±0.15	0.42±0.08	0.18±0.09	0.16±0.07	0.60±0.08	0.47±0.08	0.49±0.01
		0.5	0.61±0.10	**0.73**±0.10	0.26±0.15	0.41±0.08	0.23±0.07	0.17±0.06	0.65±0.07	0.43±0.02	0.44±0.03
		1.0	0.60±0.10	**0.74**±0.09	0.28±0.17	0.41±0.08	0.23±0.07	0.18±0.08	0.67±0.04	0.53±0.05	0.45±0.06
	0.5	0.0	0.54±0.10	**0.69**±0.09	0.49±0.17	0.39±0.08	0.07±0.09	0.19±0.08	0.57±0.04	0.51±0.08	0.43±0.03
		0.1	0.54±0.09	**0.68**±0.08	0.47±0.16	0.38±0.09	0.09±0.09	0.20±0.07	0.57±0.06	0.44±0.06	0.43±0.03
		0.5	0.51±0.10	**0.66**±0.08	0.47±0.15	0.34±0.09	0.15±0.10	0.20±0.08	0.61±0.05	0.52±0.09	0.43±0.03
		1.0	0.51±0.11	**0.67**±0.09	0.47±0.14	0.35±0.09	0.19±0.09	0.20±0.08	0.57±0.05	0.50±0.07	0.44±0.03
	1.0	0.0	0.47±0.09	0.58±0.08	**0.63**±0.14	0.31±0.09	0.07±0.07	0.20±0.07	0.53±0.05	0.46±0.06	0.41±0.03
		0.1	0.46±0.09	0.58±0.09	**0.63**±0.12	0.28±0.10	0.07±0.08	0.22±0.07	0.54±0.06	0.46±0.06	0.41±0.04
		0.5	0.46±0.08	0.58±0.09	**0.61**±0.12	0.27±0.11	0.15±0.09	0.23±0.06	0.52±0.05	0.47±0.07	0.41±0.03
		1.0	0.44±0.08	0.58±0.10	**0.61**±0.13	0.28±0.10	0.15±0.09	0.21±0.07	0.49±0.08	0.44±0.07	0.42±0.03
$\widetilde{\text{OU}}(\sigma_g^m > 0)$	0.2	0.0	0.30±0.06	**0.71**±0.10	0.40±0.12	0.18±0.13	0.02±0.05	0.26±0.06	0.45±0.08	0.46±0.05	0.41±0.05
		0.1	0.32±0.06	**0.71**±0.09	0.38±0.11	0.21±0.12	0.02±0.05	0.25±0.06	0.45±0.06	0.47±0.05	0.41±0.04
		0.5	0.46±0.10	**0.74**±0.10	0.34±0.14	0.36±0.10	0.03±0.05	0.21±0.07	0.60±0.09	0.50±0.06	0.44±0.05
		1.0	0.56±0.12	**0.75**±0.10	0.27±0.15	0.39±0.09	0.04±0.06	0.18±0.08	0.56±0.04	0.46±0.03	0.45±0.03
	0.5	0.0	0.30±0.06	**0.68**±0.09	0.51±0.10	0.14±0.12	0.01±0.03	0.26±0.05	0.45±0.07	0.45±0.04	0.41±0.04
		0.1	0.31±0.06	**0.68**±0.09	0.51±0.12	0.15±0.12	0.01±0.03	0.25±0.05	0.48±0.07	0.45±0.03	0.41±0.05
		0.5	0.42±0.09	**0.67**±0.09	0.50±0.13	0.30±0.11	0.01±0.02	0.24±0.07	0.54±0.06	0.47±0.01	0.43±0.03
		1.0	0.49±0.10	**0.66**±0.09	0.48±0.16	0.32±0.10	0.02±0.04	0.22±0.06	0.57±0.04	0.44±0.07	0.43±0.03
	1.0	0.0	0.30±0.05	**0.59**±0.09	0.56±0.09	0.11±0.10	0.00±0.02	0.26±0.05	0.42±0.04	0.41±0.04	0.39±0.04
		0.1	0.31±0.06	**0.58**±0.08	0.58±0.10	0.12±0.11	0.00±0.01	0.25±0.05	0.43±0.05	0.41±0.04	0.39±0.04
		0.5	0.38±0.08	0.59±0.08	**0.61**±0.12	0.21±0.12	0.00±0.02	0.23±0.06	0.49±0.06	0.43±0.04	0.40±0.03
		1.0	0.43±0.08	0.58±0.09	**0.62**±0.11	0.24±0.10	0.01±0.03	0.23±0.07	0.52±0.04	0.46±0.04	0.41±0.04

Table 6.10: TNR Score - $N = 10,\ \sigma_g^a = 1.0$

Dataset	C	σ_b	Granger	PLACy	PCMCI	RCV-VarLiNGAM	DYNOTEARS	PCMCI$_\Omega$	BCGeweke	DTF	GewekeNP
OU($\sigma_g^m = 0$)	0.2	0.0	**0.95**±0.02	0.95±0.02	**0.99**±0.01	0.69±0.04	0.92±0.07	0.72±0.06	0.83±0.05	0.86±0.04	0.53±0.09
		0.1	**0.82**±0.06	0.96±0.02	**0.99**±0.01	0.69±0.04	0.82±0.12	0.73±0.06	0.62±0.09	0.84±0.06	0.50±0.08
		0.5	0.61±0.09	0.95±0.02	**0.96**±0.02	0.70±0.05	0.63±0.15	0.72±0.05	0.29±0.03	0.56±0.03	**0.53**±0.09
		1.0	0.53±0.09	0.94±0.02	**0.97**±0.02	0.78±0.04	0.63±0.15	0.71±0.06	0.24±0.03	0.36±0.12	**0.51**±0.09
	0.5	0.0	0.83±0.08	**0.93**±0.04	**0.99**±0.01	0.75±0.05	0.94±0.04	0.61±0.10	0.62±0.12	0.55±0.10	0.23±0.08
		0.1	0.72±0.09	**0.93**±0.04	**0.99**±0.01	0.76±0.05	0.95±0.04	0.61±0.11	0.52±0.15	0.65±0.10	0.22±0.08
		0.5	0.49±0.12	**0.93**±0.03	0.95±0.02	0.81±0.04	**0.96**±0.06	0.63±0.09	0.23±0.05	0.43±0.08	0.24±0.11
		1.0	0.42±0.10	**0.92**±0.04	0.96±0.02	0.81±0.04	**0.97**±0.06	0.62±0.09	0.23±0.06	0.24±0.05	0.20±0.08
	1.0	0.0	0.76±0.11	**0.77**±0.11	**0.98**±0.02	0.77±0.05	0.87±0.06	0.61±0.11	0.60±0.10	0.37±0.11	0.20±0.07
		0.1	0.70±0.10	**0.78**±0.11	**0.98**±0.02	0.78±0.05	0.89±0.06	0.58±0.11	0.46±0.11	0.38±0.11	0.20±0.08
		0.5	0.42±0.14	**0.79**±0.11	**0.95**±0.03	0.83±0.05	0.92±0.09	0.59±0.12	0.15±0.04	0.23±0.06	0.19±0.05
		1.0	0.36±0.11	**0.79**±0.11	**0.94**±0.03	0.84±0.05	0.92±0.09	0.57±0.11	0.16±0.03	0.21±0.07	0.19±0.07
$\widetilde{\text{OU}}(\sigma_g^m = 0)$	0.2	0.0	**0.95**±0.03	0.93±0.03	1.00±0.01	0.69±0.04	**1.00**±0.00	0.73±0.06	0.81±0.04	0.13±0.01	0.13±0.00
		0.1	0.65±0.08	0.89±0.03	0.99±0.01	0.69±0.04	**1.00**±0.00	0.73±0.06	**0.46**±0.09	0.13±0.00	0.13±0.00
		0.5	0.15±0.02	0.92±0.03	0.96±0.02	0.71±0.04	**0.99**±0.01	0.72±0.06	0.20±0.02	0.14±0.01	0.19±0.08
		1.0	0.21±0.04	0.92±0.03	**0.97**±0.02	0.77±0.04	0.95±0.02	0.72±0.05	0.22±0.03	0.14±0.00	0.40±0.11
	0.5	0.0	0.83±0.08	**0.89**±0.05	1.00±0.01	0.76±0.05	**1.00**±0.00	0.60±0.10	0.58±0.06	0.13±0.00	0.13±0.00
		0.1	0.55±0.12	**0.83**±0.06	0.98±0.02	0.76±0.05	**1.00**±0.00	0.61±0.10	0.42±0.14	0.13±0.00	0.13±0.00
		0.5	0.15±0.03	**0.90**±0.05	0.94±0.02	0.81±0.05	**0.99**±0.01	0.63±0.10	0.16±0.03	0.13±0.01	0.16±0.03
		1.0	0.19±0.05	**0.89**±0.04	**0.95**±0.02	0.82±0.04	0.94±0.03	0.63±0.10	0.22±0.07	0.14±0.01	0.22±0.08
	1.0	0.0	**0.76**±0.11	0.74±0.11	0.98±0.02	0.79±0.06	**1.00**±0.00	0.61±0.10	0.58±0.09	0.13±0.00	0.13±0.00
		0.1	0.53±0.13	**0.69**±0.11	0.98±0.02	0.79±0.05	**1.00**±0.00	0.57±0.12	0.36±0.09	0.13±0.00	0.13±0.00
		0.5	0.15±0.03	**0.75**±0.11	0.94±0.03	0.83±0.05	**0.99**±0.02	0.60±0.11	0.16±0.02	0.13±0.00	0.15±0.02
		1.0	0.19±0.05	**0.75**±0.12	**0.94**±0.02	0.85±0.05	0.93±0.04	0.57±0.12	0.18±0.05	0.13±0.00	0.19±0.08
OU($\sigma_g^m > 0$)	0.2	0.0	0.83±0.08	**0.94**±0.03	**0.98**±0.02	0.74±0.05	0.71±0.12	0.70±0.08	0.76±0.09	0.75±0.18	0.45±0.06
		0.1	0.82±0.09	**0.94**±0.04	**0.98**±0.02	0.73±0.05	0.71±0.12	0.69±0.07	0.67±0.13	0.70±0.15	0.42±0.07
		0.5	0.81±0.09	**0.93**±0.04	**0.98**±0.02	0.74±0.04	0.53±0.15	0.69±0.07	0.72±0.10	0.70±0.15	0.50±0.12
		1.0	0.81±0.09	**0.93**±0.04	**0.97**±0.02	0.74±0.05	0.50±0.16	0.68±0.08	0.76±0.09	0.79±0.17	0.47±0.10
	0.5	0.0	0.71±0.13	**0.86**±0.07	**0.97**±0.03	0.78±0.05	0.88±0.08	0.63±0.10	0.60±0.05	0.52±0.17	0.28±0.10
		0.1	0.71±0.13	**0.86**±0.07	**0.97**±0.03	0.78±0.05	0.87±0.08	0.63±0.09	0.58±0.09	0.53±0.12	0.26±0.08
		0.5	0.67±0.13	**0.84**±0.07	**0.96**±0.04	0.80±0.05	0.75±0.12	0.61±0.10	0.63±0.08	0.54±0.22	0.30±0.13
		1.0	0.67±0.16	**0.84**±0.08	**0.96**±0.04	0.81±0.06	0.68±0.16	0.61±0.10	0.60±0.08	0.52±0.15	0.28±0.11
	1.0	0.0	0.61±0.15	0.77±0.11	**0.94**±0.06	0.82±0.05	0.83±0.08	0.62±0.10	0.51±0.09	0.39±0.12	0.23±0.08
		0.1	0.60±0.16	0.77±0.10	**0.94**±0.04	0.82±0.05	0.82±0.08	0.60±0.10	0.54±0.09	0.38±0.12	0.24±0.10
		0.5	0.60±0.15	0.77±0.10	**0.94**±0.05	0.84±0.05	0.73±0.13	0.60±0.09	0.50±0.14	0.45±0.16	0.25±0.11
		1.0	0.58±0.16	0.77±0.10	**0.94**±0.05	0.84±0.05	0.71±0.12	0.60±0.10	0.43±0.16	0.38±0.17	0.25±0.08
$\widetilde{\text{OU}}(\sigma_g^m > 0)$	0.2	0.0	0.28±0.13	**0.93**±0.04	0.88±0.04	0.93±0.05	**0.97**±0.04	0.31±0.11	0.35±0.16	0.34±0.18	0.20±0.09
		0.1	0.33±0.14	**0.93**±0.04	0.89±0.04	0.92±0.05	**0.97**±0.04	0.34±0.12	0.36±0.15	0.41±0.16	0.25±0.12
		0.5	0.64±0.13	**0.93**±0.04	**0.96**±0.02	0.80±0.05	0.95±0.04	0.57±0.12	0.67±0.13	0.60±0.13	0.38±0.04
		1.0	0.76±0.11	**0.94**±0.04	**0.97**±0.02	0.77±0.05	0.94±0.04	0.66±0.09	0.67±0.06	0.72±0.15	0.38±0.13
	0.5	0.0	0.25±0.13	0.85±0.07	**0.87**±0.05	0.95±0.04	**0.97**±0.03	0.29±0.11	0.28±0.15	0.28±0.11	0.17±0.05
		0.1	0.27±0.14	0.85±0.08	**0.88**±0.05	0.93±0.04	**0.97**±0.03	0.31±0.11	0.33±0.16	0.30±0.13	0.16±0.04
		0.5	0.53±0.16	0.84±0.09	**0.94**±0.04	0.84±0.05	**0.95**±0.04	0.52±0.11	0.55±0.10	0.42±0.07	0.29±0.10
		1.0	0.65±0.16	**0.84**±0.08	**0.96**±0.03	0.82±0.06	0.92±0.05	0.58±0.10	0.57±0.10	0.54±0.19	0.28±0.08
	1.0	0.0	0.23±0.13	0.77±0.10	**0.84**±0.05	0.96±0.03	**0.97**±0.04	0.28±0.10	0.24±0.08	0.19±0.07	0.15±0.04
		0.1	0.26±0.15	0.77±0.10	**0.86**±0.05	0.95±0.04	**0.97**±0.04	0.31±0.10	0.28±0.13	0.20±0.09	0.15±0.04
		0.5	0.46±0.17	0.78±0.09	**0.92**±0.05	0.87±0.05	**0.94**±0.05	0.51±0.12	0.42±0.11	0.29±0.09	0.18±0.05
		1.0	0.55±0.16	0.77±0.10	**0.93**±0.05	0.85±0.05	0.91±0.05	0.58±0.11	0.47±0.11	0.35±0.10	0.19±0.07

Table 6.11: Hyper-parameters of the causal discovery algorithms.

Algorithm	Hyper-Parameter	Value
Granger	`maxlag` p-value	10 0.05
PLaCy	window length (l) stride (s) p-value	50 1 0.05
PCMCI	τ_{max} Conditional Independence Test $\mathtt{PC}_\alpha$	10 Partial Correlation 0.05
CCM-Filtering	Filter Size Stride τ L	5 1 10 `range(25,` L`, 250)`
RCV-VarLiNGAM	k Sequence Length τ_c τ_v	7 300 0.7 0.4

Rhino	Noise Distribution
	`init_rho`
	`init_alpha`
		Gaussian
		30
		0.2

DYNOTEARS	`p` `max_iter`	10 100
PCMCI$_\Omega$	τ_{max}	10
BCGeweke	n_{freqs} `Frequency band`	128 `Three equal band in the spectrum ‘‘low’’, ‘‘mid’’, ‘‘high’’`
DTF	n_{freqs} n_{perm}	128 100
GewekeNP	n_{freqs} n_{perm}	128 100

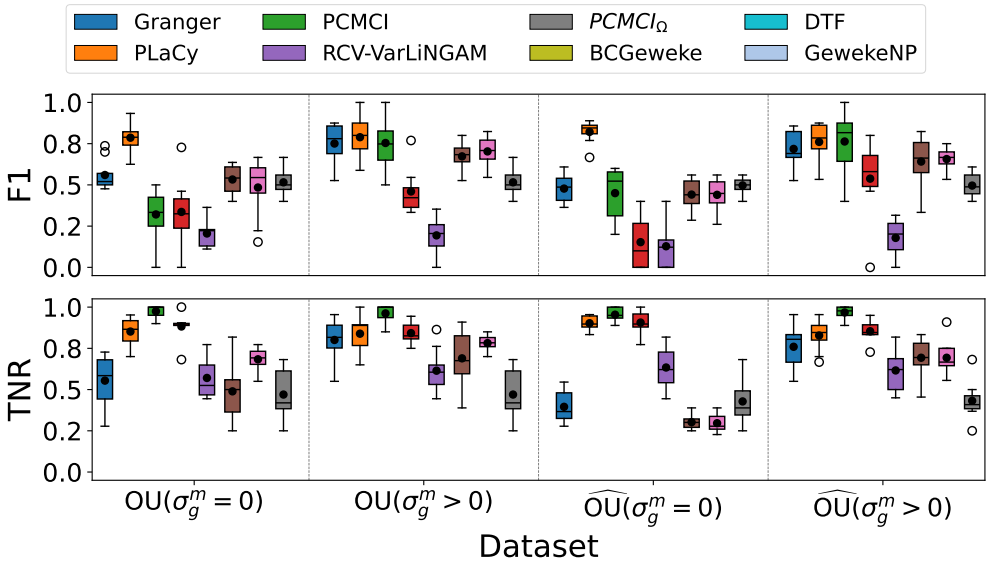


Figure 6.5: Non linear process. $N = 5$, $C = 1.0$, $s_g = 1.0$, $s_b = 1.0$.

Negative Control Experiments

We performed additional numerical experiments and evaluated the Petersen test [252] on the datasets of Tables 6.1 and 6.2, as well as on the datasets of Table 6.1 with N increased from 5 to 10. Across all settings, the negative-control baseline yields consistently lower F1 and TNR

Table 6.12: Performance Improvement - $N = 5,\ \sigma_g^a = 1.0$

		Granger		PCMCI	
	C	**F1**	**TNR**	**F1**	**TNR**
$\mathrm{OU}(\sigma_g^m = 0)$	0.2	-0.44$_{\pm 0.32}$	+0.21$_{\pm 0.19}$	+0.02$_{\pm 0.32}$	-0.03$_{\pm 0.07}$
	0.5	+0.16$_{\pm 0.26}$	+0.21$_{\pm 0.20}$	+0.38$_{\pm 0.44}$	-0.03$_{\pm 0.07}$
	1.0	+0.23$_{\pm 0.26}$	+0.21$_{\pm 0.21}$	+0.46$_{\pm 0.34}$	-0.03$_{\pm 0.08}$
$\widetilde{\mathrm{OU}}(\sigma_g^m = 0)$	0.2	-0.22$_{\pm 0.35}$	+0.38$_{\pm 0.30}$	+0.13$_{\pm 0.39}$	-0.05$_{\pm 0.10}$
	0.5	+0.24$_{\pm 0.27}$	+0.38$_{\pm 0.29}$	+0.37$_{\pm 0.38}$	-0.06$_{\pm 0.09}$
	1.0	+0.25$_{\pm 0.26}$	+0.34$_{\pm 0.29}$	+0.42$_{\pm 0.34}$	-0.05$_{\pm 0.09}$
$\mathrm{OU}(\sigma_g^m > 0)$	0.2	+0.09$_{\pm 0.28}$	+0.09$_{\pm 0.10}$	+0.38$_{\pm 0.37}$	-0.01$_{\pm 0.05}$
	0.5	+0.18$_{\pm 0.20}$	+0.11$_{\pm 0.12}$	+0.43$_{\pm 0.35}$	-0.01$_{\pm 0.05}$
	1.0	+0.20$_{\pm 0.20}$	+0.13$_{\pm 0.14}$	+0.27$_{\pm 0.31}$	-0.00$_{\pm 0.05}$
$\widetilde{\mathrm{OU}}(\sigma_g^m > 0)$	0.2	+0.36$_{\pm 0.25}$	+0.41$_{\pm 0.24}$	+0.35$_{\pm 0.35}$	+0.08$_{\pm 0.10}$
	0.5	+0.41$_{\pm 0.21}$	+0.41$_{\pm 0.23}$	+0.39$_{\pm 0.27}$	+0.08$_{\pm 0.10}$
	1.0	+0.39$_{\pm 0.22}$	+0.40$_{\pm 0.24}$	+0.31$_{\pm 0.25}$	+0.08$_{\pm 0.11}$

scores, confirming the effectiveness of our approach. We also remark that the p-values returned by the tests were always statistically significant. Results can be found in Table 6.13.

Additionally, we note that the Petersen test can vary simply due to the number of edges reconstructed by each algorithm. To account for this effect, we performed an alternative evaluation in which the Petersen test is computed using the true number of edges in the underlying causal graph, rather than the number predicted by each method. This alternative evaluation does not depend on the specific algorithm under consideration and therefore yields a single value for each graph size. The corresponding results are reported below in Table 6.14 and again demonstrate that our approach is substantially more robust than a random guess.

6.8.5 Acyclicity Assumption

Finally, while the synthetic graphs used for our controlled experiments were generated as DAGs, we do not necessarily assume acyclicity of the underlying graphs, nor do we impose this constraint on the recovered graphs. This design choice is intentional, as it allows the method to remain as general and broadly applicable as possible. However, we performed additional experiments to evaluate the behavior of PLaCy when acyclicity is explicitly enforced. Specifically, we iteratively detect cycles

Table 6.13: Comparison of PLaCy with the Petersen test across synthetic and real-world datasets. Results are reported as mean ± standard deviation over multiple runs with $\sigma_g = 0.5$: $C = 0.5$ $\sigma_b = 0.5$.

	Dataset / Condition	F1 (PLaCy)	TNR (PLaCy)	F1 (Petersen test)	TNR (Petersen test)
$N = 5$	$\mathrm{OU}(\sigma_g^m = 0)$	0.58 ± 0.27	0.93 ± 0.06	0.11 ± 0.17	0.83 ± 0.11
	$\widehat{\mathrm{OU}}(\sigma_g^m = 0)$	0.57 ± 0.25	0.89 ± 0.08	0.15 ± 0.19	0.79 ± 0.10
	$\mathrm{OU}(\sigma_g^m = 0.5)$	0.77 ± 0.20	0.92 ± 0.08	0.18 ± 0.22	0.79 ± 0.11
	$\widehat{\mathrm{OU}}(\sigma_g^m = 0.5)$	0.78 ± 0.20	0.92 ± 0.08	0.16 ± 0.19	0.79 ± 0.11
$N = 10$	$\mathrm{OU}(\sigma_g^m = 0.0)$	0.63 ± 0.10	0.92 ± 0.04	0.15 ± 0.10	0.82 ± 0.06
	$\widehat{\mathrm{OU}}(\sigma_g^m = 0.0)$	0.57 ± 0.09	0.86 ± 0.06	0.17 ± 0.09	0.76 ± 0.07
	$\mathrm{OU}(\sigma_g^m = 0.5)$	0.69 ± 0.08	0.86 ± 0.07	0.19 ± 0.09	0.74 ± 0.08
	$\widehat{\mathrm{OU}}(\sigma_g^m = 0.5)$	0.70 ± 0.09	0.86 ± 0.08	0.19 ± 0.09	0.73 ± 0.09
	Rivers	0.51 ± 0.10	0.68 ± 0.16	0.26 ± 0.08	0.61 ± 0.15
	AirQuality	0.45 ± 0.04	0.65 ± 0.07	0.32 ± 0.03	0.59 ± 0.08

Table 6.14: Modified Petersen test using the true number of edges.

Dataset	Mean F1	Mean TNR
$N = 5$	0.15 ± 0.20	0.85 ± 0.08
$N = 10$	0.16 ± 0.10	0.84 ± 0.04
Rivers	0.16 ± 0.15	0.83 ± 0.03
AirQuality	0.24 ± 0.02	0.74 ± 0.01

in the recovered graph and remove, at each step, the single edge with the highest p-value (i.e., the lowest statistical significance), until no cycles remain. The results of this procedure are reported in Table 6.15.

Table 6.15: Comparison between acyclic PLaCy and standard PLaCy for $N = 5$; $\sigma_g = 0.5$; $C = 0.5$; $\sigma_b = 0.5$.

Dataset / Condition	Acyclic PLaCy (F1)	Acyclic PLaCy (TNR)	PLaCy (F1)	PLaCy (TNR)
$\mathrm{OU}(\sigma_g^m = 0)$	0.65 ± 0.19	0.96 ± 0.06	0.58 ± 0.27	0.93 ± 0.06
$\widehat{\mathrm{OU}}(\sigma_g^m = 0)$	0.60 ± 0.20	0.90 ± 0.06	0.57 ± 0.25	0.89 ± 0.08
$\mathrm{OU}(\sigma_g^m = 0.5)$	0.90 ± 0.10	0.96 ± 0.04	0.77 ± 0.20	0.92 ± 0.08
$\widehat{\mathrm{OU}}(\sigma_g^m = 0.5)$	0.91 ± 0.09	0.94 ± 0.06	0.78 ± 0.20	0.92 ± 0.08

These results show that imposing acyclicity does not deteriorate the reconstructive performance of our method and, in several cases, even leads to a slight improvement

Complexity analysis

Let N be the number of time series (nodes), L the length of each time series, l the window length used for the spectral analysis, and s the stride of the sliding window.

Spectral preprocessing. Each time series is divided into overlapping windows of length l and stride s. The number of windows generated from a single time series is therefore

$$W = \left\lfloor \frac{L - l}{s} \right\rfloor + 1 = O\left(\frac{L}{s}\right).$$

In each window, the algorithm performs: (i) an FFT of size l with cost $O(l \log l)$, (ii) a log–log transformation of amplitudes and frequencies ($O(l)$), and (iii) a linear regression to estimate the spectral coefficients (a, λ) ($O(l)$). The FFT dominates the per-window computation, so the cost per window is $O(l \log l)$. Hence, the cost for a single time series is

$$O\left(\frac{L}{s}\right) \cdot O(l \log l) = O\left(\frac{L}{s}\, l \log l\right),$$

and for all N time series:

$$O\left(N \cdot \frac{L}{s}\, l \log l\right).$$

Since the stride s is typically very small (often $s = 1$), we may simplify this as $O(NL\ l \log l)$.

After the spectral preprocessing, the method performs pairwise Granger causality tests across all ordered pairs of the N variables, resulting in $N(N-1) = O(N^2)$ tests. A single Granger test on two time series of length L (with fixed lag order) requires solving a small linear regression and therefore costs $O(L)$. The overall cost of this step is thus $O(N^2 L)$.

Combining both contributions, the total computational complexity of the method is $O(NL\, l \log l) + O(N^2 L)$.

Execution Times

The experiments were conducted on a high-performance computing cluster comprising 50 DELL EMC PowerEdge R7425 servers, each equipped with dual AMD EPYC 7301 processors (32 cores total per node at 2.2GHz). Among the nodes, 19 are enhanced with NVIDIA Quadro RTX 6000 GPUs (24GB VRAM). These GPUs were used only for running **Rhino**, as this method requires the training of deep neural networks. As such, for $N = 5$, the execution of **Rhino** took ~ 4 hours.

Table 6.16: Elapsed time for a single experiment (seconds).

Method	**N = 5**	**N = 10**
Granger	$0.16_{\pm 0.01}$	$0.71_{\pm 0.00}$
PLaCy	$5.80_{\pm 0.01}$	$11.94_{\pm 0.03}$
PCMCI	$1.05_{\pm 0.00}$	$4.86_{\pm 0.03}$
CCM-Filtering	$426.35_{\pm 1.57}$	$1924.20_{\pm 19.32}$
RCV-VarLiNGAM	$1.98_{\pm 0.10}$	$7.03_{\pm 0.27}$
DYNOTEARS	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$
PCMCIω	$7.06_{\pm 0.04}$	$44.42_{\pm 0.13}$
BCGeweke	$2.45_{\pm 0.10}$	$4.63_{\pm 0.26}$
DTF	$14.19_{\pm 1.33}$	$93.10_{\pm 2.01}$
GewekeNP	$3.34_{\pm 0.03}$	$9.69_{\pm 0.30}$

Chapter 7

Conclusion

This thesis investigated the modeling, prediction, and generation of financial time-series through data-driven methodologies, addressing fundamental challenges posed by volatility, non-stationarity, heavy-tailed distributions, and complex interdependencies among assets. By combining machine learning, deep learning, generative modeling, and causal inference techniques, the work advances the understanding of high-frequency financial dynamics.

The first contribution introduced a formal framework for detecting and forecasting stock market shocks. By modeling returns with Lévy-stable distributions and leveraging limit order book microstructure features, the proposed approach enables early identification of abrupt market movements. The integration of feature selection and Bayesian-optimized Random Forest models demonstrated strong predictive performance, offering a practical tool for anticipating disruptive market events and supporting proactive decision-making.

The thesis then examined the problem of short-term price trend prediction using limit order book data. Through a systematic benchmark of state-of-the-art deep learning models, the study revealed a significant gap between experimental results and real-world performance due to overfitting, dataset bias, and distribution shifts. The proposed evaluation framework and experimental analysis provide guidance toward more robust, reproducible, and practically applicable forecasting systems.

To address the scarcity and sensitivity of financial data, the thesis proposed generative approaches for synthesizing realistic multivariate time-series. The developed framework enables the generation of correlated financial sequences that preserve stylized facts and cross-asset dependencies, facilitating stress testing, scenario analysis, and training of data-driven models in privacy-sensitive contexts.

Building upon this, the thesis introduced a diffusion-based methodology for generating time-series with explicit causal dependencies. By reconstructing causal structures and embedding them into the generation process, the proposed approach enables the creation of synthetic datasets suitable for benchmarking causal discovery algorithms and enhancing downstream predictive tasks.

Finally, the thesis addressed the reliability of causal discovery in real-world time-series exhibiting noise and power-law behavior. By studying causal invariance under spectral transformations, the proposed methodology improves the robustness of causal inference in complex temporal systems, contributing to more reliable understanding of dynamic dependencies.

Overall, this research provides a unified perspective on financial time-series analysis, spanning shock forecasting, robust trend prediction, correlated and causal data generation, and causal dis-

covery. These contributions support safer trading strategies, improved risk assessment, realistic simulation environments, and more interpretable data-driven models.

7.1 Limitations and Future Directions

Despite these advances, several challenges remain open. Shock prediction could be extended to directional forecasting and to the integration of exogenous signals such as news sentiment and macroeconomic indicators. Trend prediction models would benefit from improved robustness to regime shifts and adaptive learning strategies capable of handling evolving market conditions. Generative models could be extended to capture multi-scale temporal dependencies and extreme tail events more faithfully. Finally, causal discovery in financial systems remains challenging due to latent variables and feedback loops, motivating future work on hybrid causal-representation learning approaches. Advancing these directions will further strengthen the reliability, interpretability, and real-world applicability of data-driven financial modeling, contributing to more resilient financial systems and informed decision-making in increasingly complex markets.

Bibliography

[1] Benoit Mandelbrot et al. The variation of certain speculative prices. *Journal of business*, 36(4):394, 1963.

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

[3] Álvaro Cartea, Sebastian Jaimungal, and José Penalva. *Algorithmic and high-frequency trading*. Cambridge University Press, 2015.

[4] Martin D Gould, Mason A Porter, Stacy Williams, Mark McDonald, Daniel J Fenn, and Sam D Howison. Limit order books. *Quantitative Finance*, 13(11):1709–1742, 2013.

[5] Frédéric Abergel, Marouane Anane, Anirban Chakraborti, Aymen Jedidi, and Ioane Muni Toke. *Limit order books*. Cambridge University Press, 2016.

[6] Ruey S Tsay. *Analysis of financial time series*. John wiley & sons, 2005.

[7] Robert Engle. Risk and volatility: Econometric models and financial practice. *American economic review*, 94(3):405–420, 2004.

[8] John Y Campbell, Andrew W Lo, A Craig MacKinlay, and Robert F Whitelaw. The econometrics of financial markets. *Macroeconomic Dynamics*, 2(4):559–562, 1998.

[9] Delmar Permann and Ian Hamilton. Wavelet analysis of time series for the weakly forced and weakly damped morse oscillator. *The Journal of chemical physics*, 100(1):379–386, 1994.

[10] Jianqing Fan, Fang Han, and Han Liu. Challenges of big data analysis. *National science review*, 1(2):293–314, 2014.

[11] James B Heaton, Nicholas G Polson, and Jan Hendrik Witte. Deep learning in finance. *arXiv preprint arXiv:1602.06561*, 2016.

[12] Justin Sirignano and Rama Cont. Universal features of price formation in financial markets: perspectives from deep learning. In *Machine learning and AI in finance*, pages 5–15. Routledge, 2021.

[13] Thomas Fischer and Christopher Krauss. Deep learning with long short-term memory networks for financial market predictions. *European journal of operational research*, 270(2):654–669, 2018.

[14] Zihao Zhang, Stefan Zohren, and Stephen Roberts. Deeplob: Deep convolutional neural networks for limit order books. *IEEE Transactions on Signal Processing*, 67(11):3001–3012, 2019.

[15] Dat Thanh Tran, Juho Kanniainen, Moncef Gabbouj, and Alexandros Iosifidis. Data normalization for bilinear structures in high-frequency financial time-series. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 7287–7292. IEEE, 2021.

[16] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[17] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[18] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[19] Magnus Wiese, Robert Knobloch, Ralf Korn, and Peter Kretschmer. Quant GANs: Deep generation of financial time series. *Quantitative Finance*, 20(9):1419–1440, September 2020.

[20] J Pearl. *Causality*. Cambridge University Press, 2009.

[21] Jonas Peters, Dominik Janzing, and Bernhard Scholkopf. *Elements of causal inference: foundations and learning algorithms*. MIT press, 2017.

[22] Michael Eichler. Causal inference with multiple time series: principles and problems. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 371(1997), 2013.

[23] Jakob Runge, Peer Nowack, Marlene Kretschmer, Seth Flaxman, and Dino Sejdinovic. Detecting and quantifying causal associations in large nonlinear time series datasets. *Science advances*, 5(11):eaau4996, 2019.

[24] Viviana Arrigoni, Giuseppe Masi, Emanuele Mercanti, Novella Bartolini, and Svitlana Vyetrenko. Stock shocks modelling and forecasting. In *2023 IEEE 43rd International Conference on Distributed Computing Systems Workshops (ICDCSW)*, pages 67–72. IEEE, 2023.

[25] Giuseppe Masi, Matteo Prata, Michele Conti, Novella Bartolini, and Svitlana Vyetrenko. On correlated stock market time series generation. In *Proceedings of the Fourth ACM International Conference on AI in Finance*, pages 524–532, 2023.

[26] Matteo Prata, Giuseppe Masi, Leonardo Berti, Viviana Arrigoni, Andrea Coletta, Irene Cannistraci, Svitlana Vyetrenko, Paola Velardi, and Novella Bartolini. Lob-based deep learning models for stock price trend prediction: a benchmark study. *Artificial Intelligence Review*, 57(5):116, 2024.

[27] Giuseppe Masi, Andrea Coletta, Elizabeth Fons, Svitlana Vyetrenko, and Novella Bartolini. Diffcats: Causally associated time-series generation through diffusion models. *Transactions on Machine Learning Research*.

[28] Novella Bartolini, Giuseppe Masi, Matteo Prata, and Federico Trombetti. Patrolling heterogeneous targets with fanets. In *IEEE INFOCOM 2024-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 1–6. IEEE, 2024.

[29] Albert J. Menkveld. High frequency trading and the new market makers. *Journal of Financial Markets*, 16(4):712–740, 2013.

[30] Xavier Gabaix, Parameswaran Gopikrishnan, Vasiliki Plerou, and H Eugene Stanley. A theory of power-law distributions in financial market fluctuations. *Nature*, 423(6937):267–270, 2003.

[31] John P Nolan. Univariate stable distributions. *Springer Series in Operations Research and Financial Engineering*, 10:978–3, 2020.

[32] Germany. The Efficient Reconstructor at Humboldt Universität zu Berlin. Lobster: Limit order book system.

[33] Mei-See Cheong, Mei-Chen Wu, and Szu-Hao Huang. Interpretable stock anomaly detection based on spatio-temporal relation networks with genetic algorithm. *IEEE Access*, 9:68302–68319, 2021.

[34] Jinwen Sun, Keli Xiao, Chuanren Liu, Wenjun Zhou, and Hui Xiong. Exploiting intra-day patterns for market shock prediction: A machine learning approach. *Expert Systems with Applications*, 127:272–281, 2019.

[35] Keli Xiao, Qi Liu, Chuanren Liu, and Hui Xiong. Price shock detection with an influence-based model of social attention. *ACM Transactions on Management Information Systems (TMIS)*, 9(1):1–21, 2017.

[36] Firuz Kamalov. Forecasting significant stock price changes using neural networks. *Neural Computing and Applications*, 32:17655–17667, 2020.

[37] Andrii Bielinskyi, Serhii Semerikov, Viktoria Solovieva, and Vladimir Soloviev. Levy's stable distribution for stock crash detecting. 2019.

[38] Louis Bachelier. *Jeu de speculation (The Theory of Speculation)*. PhD thesis, PhD thesis, University of Paris, 1900.

[39] Eugene F Fama. Random walks in stock market prices. *Financial analysts journal*, 51(1):75–80, 1995.

[40] Maurice George Kendall and A Bradford Hill. The analysis of economic time-series-part i: Prices. *Journal of the Royal Statistical Society. Series A (General)*, 116(1):11–34, 1953.

[41] Benoit B Mandelbrot and Benoit B Mandelbrot. *The variation of certain speculative prices*. Springer, 1997.

[42] Stefan Teis and Georg Gross. Reading the signs of order book and price movements. Mondo Visione, 2015.

[43] Ravid Shwartz-Ziv and Amitai Armon. Tabular data: Deep learning is not all you need. *Information Fusion*, 81:84–90, 2022.

[44] Yann-Yann Shieh and Rachel T Fouladi. The effect of multicollinearity on multilevel modeling parameter estimates and standard errors. *Educational and psychological measurement*, 63(6):951–985, 2003.

[45] Izar Azpiroz, Noelia Oses, Marco Quartulli, Igor G Olaizola, Diego Guidotti, and Susanna Marchi. Comparison of climate reanalysis and remote-sensing data for predicting olive phenology through machine-learning methods. *Remote Sensing*, 13(6):1224, 2021.

[46] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, 25, 2012.

[47] Lu Wang, Feng Ma, and Guoshan Liu. Forecasting stock volatility in the presence of extreme shocks: Short-term and long-term effects. *Journal of Forecasting*, 39(5):797–810, 2020.

[48] Matt Phillips. Nasdaq: Here's our timeline of the flash crash. May 2010.

[49] Robert F Engle, Eric Ghysels, and Bumjean Sohn. Stock market volatility and macroeconomic fundamentals. *Review of Economics and Statistics*, 95(3):776–797, 2013.

[50] Jean-Philippe Bouchaud, Julius Bonart, Jonathan Donier, and Martin Gould. Trades, quotes and prices: Financial markets under the microscope. *Cambridge University Press*, 2018.

[51] Longbing Cao. Ai in finance: challenges, techniques, and opportunities. *ACM Computing Surveys (CSUR)*, 55(3):1–38, 2022.

[52] Weiwei Jiang. Applications of deep learning in stock market prediction: recent progress. *Expert Systems with Applications*, 184:115537, 2021.

[53] Omer Berat Sezer, Mehmet Ugur Gudelek, and Ahmet Murat Ozbayoglu. Financial time series forecasting with deep learning: A systematic literature review: 2005–2019. *Applied soft computing*, 90:106181, 2020.

[54] Xiao-Yang Liu, Ziyi Xia, Jingyang Rui, Jiechao Gao, Hongyang Yang, Ming Zhu, Christina Wang, Zhaoran Wang, and Jian Guo. Finrl-meta: Market environments and benchmarks for data-driven financial reinforcement learning. *Advances in Neural Information Processing Systems*, 35:1835–1849, 2022.

[55] Ilia Zaznov, Julian Kunkel, Alfonso Dufour, and Atta Badii. Predicting stock price changes based on the limit order book: a survey. *Mathematics*, 10(8):1234, 2022.

[56] Joelle Pineau, Philippe Vincent-Lamarre, Koustuv Sinha, Vincent Larivière, Alina Beygelzimer, Florence d'Alché Buc, Emily Fox, and Hugo Larochelle. Improving reproducibility in machine learning research (a report from the neurips 2019 reproducibility program). *The Journal of Machine Learning Research*, 22(1):7459–7478, 2021.

[57] Odd Erik Gundersen and Sigbjørn Kjensmo. State of the art: Reproducibility in artificial intelligence. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[58] Monya Baker. Reproducibility crisis. *Nature*, 533(26):353–66, 2016.

[59] Adamantios Ntakaris, Martin Magris, Juho Kanniainen, Moncef Gabbouj, and Alexandros Iosifidis. Benchmark dataset for mid-price forecasting of limit order book data with machine learning methods. *Journal of Forecasting*, 37(8):852–866, 2018.

[60] Mahinda Mailagaha Kumbure, Christoph Lohrmann, Pasi Luukka, and Jari Porras. Machine learning techniques and data for stock market forecasting: a literature review. *Expert Systems with Applications*, page 116659, 2022.

[61] Ahmet Murat Ozbayoglu, Mehmet Ugur Gudelek, and Omer Berat Sezer. Deep learning for financial applications: A survey. *Applied Soft Computing*, 93:106384, 2020.

[62] Zexin Hu, Yiqi Zhao, and Matloob Khushi. A survey of forex and stock price prediction using deep learning. *Applied System Innovation*, 4(1):9, 2021.

[63] Isaac Kofi Nti, Adebayo Felix Adekoya, and Benjamin Asubam Weyori. A systematic review of fundamental and technical analysis of stock market predictions. *Artificial Intelligence Review*, 53(4):3007–3057, 2020.

[64] Jaimin Shah, Darsh Vaidya, and Manan Shah. A comprehensive review on multiple hybrid deep learning approaches for stock prediction. *Intelligent Systems with Applications*, page 200111, 2022.

[65] Adel Ismail Al-Alawi and Yusuf Ahmed Alaali. Stock market prediction using machine learning techniques: Literature review analysis. In *2023 International Conference On Cyber Management And Engineering (CyMaEn)*, pages 153–157, 2023.

[66] Thien Hai Nguyen, Kiyoaki Shirai, and Julien Velcin. Sentiment analysis on social media for stock movement prediction. *Expert Systems with Applications*, 42(24):9603–9611, 2015.

[67] Xiaodong Li, Haoran Xie, Li Chen, Jianping Wang, and Xiaotie Deng. News impact on stock price return via sentiment analysis. *Knowledge-Based Systems*, 69:14–23, 2014.

[68] Suman Saha, Junbin Gao, and Richard Gerlach. Stock ranking prediction using list-wise approach and node embedding technique. *IEEE Access*, 9:88981–88996, 2021.

[69] Mohammad Alsulmi. From ranking search results to managing investment portfolios: Exploring rank-based approaches for portfolio stock selection. *Electronics*, 11(23):4019, 2022.

[70] Qiang Song, Anqi Liu, and Steve Y Yang. Stock portfolio selection using learning-to-rank algorithms with news sentiment. *Neurocomputing*, 264:20–28, 2017.

[71] Francesco Rundo, Francesca Trenta, Agatino Luigi di Stallo, and Sebastiano Battiato. Machine learning for quantitative finance applications: A survey. *Applied Sciences*, 9(24):5574, 2019.

[72] Latrisha N Mintarya, Jeta NM Halim, Callista Angie, Said Achmad, and Aditya Kurniawan. Machine learning approaches in stock market prediction: A systematic literature review. *Procedia Computer Science*, 216:96–102, 2023.

[73] Bryan Lim and Stefan Zohren. Time-series forecasting with deep learning: a survey. *Philosophical Transactions of the Royal Society A*, 379(2194):20200209, 2021.

[74] Dev Shah, Haruna Isah, and Farhana Zulkernine. Stock market analysis: A review and taxonomy of prediction techniques. *International Journal of Financial Studies*, 7(2):26, 2019.

[75] Kenniy Olorunnimbe and Herna Viktor. Deep learning in the stock market—a systematic survey of practice, backtesting, and applications. *Artificial Intelligence Review*, 56(3):2057–2109, 2023.

[76] Lorenzo Lucchese, Mikko Pakkanen, and Almut Veraart. The short-term predictability of returns in order book markets: a deep learning perspective. *arXiv preprint arXiv:2211.13777*, 2022.

[77] Yirou Fei and Yan Zhou. Intelligent prediction model of shanghai composite index based on technical indicators and big data analysis. *Highlights in Business, Economics and Management*, 17:370–389, 2023.

[78] CHUN YUAN LAI, RUNG-CHING CHEN, and RE Caraka. Prediction average stock price market using lstm. *ir. lib. cyut. edu. tw*, 2019.

[79] Andrea Picasso Ratto, Simone Merello, Luca Oneto, Yukun Ma, Lorenzo Malandri, and Erik Cambria. Ensemble of technical analysis and machine learning for market trend prediction. In *2018 IEEE symposium series on computational intelligence (ssci)*, pages 2090–2096. IEEE, 2018.

[80] Roger D Huang and Hans R Stoll. Market microstructure and stock return predictions. *The Review of Financial Studies*, 7(1):179–213, 1994.

[81] Dat Thanh Tran, Juho Kanniainen, and Alexandros Iosifidis. How informative is the order book beyond the best levels? machine learning perspective. *arXiv preprint arXiv:2203.07922*, 2022.

[82] Roberto Pascual and David Veredas. What pieces of limit order book information do are informative? an empirical analysis of a pure order-driven market. 2003.

[83] Charles Cao, Oliver Hansch, and Xiaoxin Wang. Order placement strategies in a pure limit order book market. *Journal of Financial Research*, 31(2):113–140, 2008.

[84] Charles Cao, Oliver Hansch, and Xiaoxin Wang. The information content of an open limit-order book. *Journal of Futures Markets: Futures, Options, and Other Derivative Products*, 29(1):16–41, 2009.

[85] Huu Nhan Duong and Petko S Kalev. Anonymity and the information content of the limit order book. *Journal of International Financial Markets, Institutions and Money*, 30:205–219, 2014.

[86] Avraam Tsantekidis, Nikolaos Passalis, Anastasios Tefas, Juho Kanniainen, Moncef Gabbouj, and Alexandros Iosifidis. Using deep learning to detect price change indications in financial

markets. In *2017 25th European Signal Processing Conference (EUSIPCO)*, pages 2511–2515, 2017.

[87] Avraam Tsantekidis, Nikolaos Passalis, Anastasios Tefas, Juho Kanniainen, Moncef Gabbouj, and Alexandros Iosifidis. Using deep learning for price prediction by exploiting stationary limit order book features. *Applied Soft Computing*, 93:106401, 2020.

[88] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32, 2019.

[89] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: a system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283. Savannah, GA, USA, 2016.

[90] Peter Gomber and Martin Haferkorn. High frequency trading. *Encyclopedia of Information Science and Technology, Third Edition*, pages 1–9, 2015.

[91] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[92] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[93] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *arXiv:2005.14165*, 2020.

[94] Avraam Tsantekidis, Nikolaos Passalis, Anastasios Tefas, Juho Kanniainen, Moncef Gabbouj, and Alexandros Iosifidis. Forecasting stock prices from the limit order book using convolutional neural networks. In *2017 IEEE 19th Conference on Business Informatics (CBI)*, volume 1, pages 7–12. IEEE, 2017.

[95] Dat Thanh Tran, Alexandros Iosifidis, Juho Kanniainen, and Moncef Gabbouj. Temporal attention-augmented bilinear network for financial time-series data analysis. *IEEE Transactions on Neural Networks and Learning Systems*, 30(5):1407–1418, 2018.

[96] Nikolaos Passalis, Anastasios Tefas, Juho Kanniainen, Moncef Gabbouj, and Alexandros Iosifidis. Deep adaptive input normalization for time series forecasting. *IEEE Transactions on Neural Networks and Learning Systems*, 31(9):3760–3765, 2019.

[97] James Wallbridge. Transformers for limit order books. *arXiv preprint arXiv:2003.00130*, 2020.

[98] Nikolaos Passalis, Anastasios Tefas, Juho Kanniainen, Moncef Gabbouj, and Alexandros Iosifidis. Temporal logistic neural bag-of-features for financial time series forecasting leveraging limit order book data. *Pattern Recognition Letters*, 136:183–189, 2020.

[99] Zihao Zhang, Bryan Lim, and Stefan Zohren. Deep learning for market by order data. *Applied Mathematical Finance*, 28(1):79–95, 2021.

[100] Yanhong Guo and Xinxin Chen. Forecasting the mid-price movements with high-frequency lob: A dual-stage temporal attention-based deep learning architecture. *Arabian Journal for Science and Engineering*, pages 1–22, 2022.

[101] Dat Thanh Tran, Nikolaos Passalis, Anastasios Tefas, Moncef Gabbouj, and Alexandros Iosifidis. Attention-based neural bag-of-features learning for sequence data. *IEEE Access*, 10:45542–45552, 2022.

[102] Damian Kisiel and Denise Gorse. Axial-lob: High-frequency trading with axial attention. *arXiv preprint arXiv:2212.01807*, 2022.

[103] Nasdaq. Stock screener. `https://www.nasdaq.com/market-activity/stocks/screener`.

[104] Stefanos Bennett and Jase Clarkson. Time series prediction under distribution shift using differentiable forgetting. *arXiv preprint arXiv:2207.11486*, 2022.

[105] Lobcast. https://github.com/matteoprata/LOBCAST.

[106] Lukas Biewald. Experiment tracking with weights and biases, 2020. Software available from wandb.com.

[107] Leo Grinsztajn, Edouard Oyallon, and Gael Varoquaux. Why do tree-based models still outperform deep learning on typical tabular data? In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022.

[108] Justin A Sirignano. Deep learning for limit order books. *Quantitative Finance*, 19(4):549–570, 2019.

[109] Amal Mahmoud and Ammar Mohammed. A survey on deep learning for time-series forecasting. *Machine learning and big data analytics paradigms: analysis, applications and challenges*, pages 365–392, 2021.

[110] José F Torres, Dalil Hadjout, Abderrazak Sebaa, Francisco Martínez-Álvarez, and Alicia Troncoso. Deep learning for time series forecasting: a survey. *Big Data*, 9(1):3–21, 2021.

[111] Larry Olanrewaju Orimoloye, Ming-Chien Sung, Tiejun Ma, and Johnnie EV Johnson. Comparing the effectiveness of deep feedforward neural networks and shallow architectures for predicting stock price indices. *Expert Systems with Applications*, 139:112828, 2020.

[112] Lukas Ruff, Jacob R Kauffmann, Robert A Vandermeulen, Grégoire Montavon, Wojciech Samek, Marius Kloft, Thomas G Dietterich, and Klaus-Robert Müller. A unifying review of deep and shallow anomaly detection. *Proceedings of the IEEE*, 109(5):756–795, 2021.

[113] Maryam M Najafabadi, Flavio Villanustre, Taghi M Khoshgoftaar, Naeem Seliya, Randall Wald, and Edin Muharemagic. Deep learning applications and challenges in big data analytics. *Journal of big data*, 2(1):1–21, 2015.

[114] Rui Ren, Desheng Dash Wu, and Tianxiang Liu. Forecasting stock market movement direction using sentiment analysis and support vector machine. *IEEE Systems Journal*, 13(1):760–770, 2018.

[115] Zhigang Jin, Yang Yang, and Yuhong Liu. Stock closing price prediction based on sentiment analysis and lstm. *Neural Computing and Applications*, 32:9713–9729, 2020.

[116] Yumo Xu and Shay B Cohen. Stock movement prediction from tweets and historical prices. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1970–1979, 2018.

[117] Yufei Wu, Mahmoud Mahfouz, Daniele Magazzeni, and Manuela Veloso. How robust are limit order book representations under data perturbation? *arXiv preprint arXiv:2110.04752*, 2021.

[118] Yufei Wu, Mahmoud Mahfouz, Daniele Magazzeni, and Manuela Veloso. Towards robust representation of limit orders books for deep learning models. *arXiv preprint arXiv:2110.05479*, 2022.

[119] Andrea Coletta, Matteo Prata, Michele Conti, Emanuele Mercanti, Novella Bartolini, Aymeric Moulin, Svitlana Vyetrenko, and Tucker Balch. Towards realistic market simulations: A generative adversarial networks approach. In *Proceedings of the Second ACM International Conference on AI in Finance (ICAIF)*, New York, NY, USA, 2022.

[120] Andrea Coletta, Aymeric Moulin, Svitlana Vyetrenko, and Tucker Balch. Learning to simulate realistic limit order book markets from data as a world agent. In *Proceedings of the Third ACM International Conference on AI in Finance*, pages 428–436, 2022.

[121] Takanobu Mizuta. A brief review of recent artificial market simulation (agent-based model) studies for financial market regulations and/or rules. *Available at SSRN 2710495*, 2016.

[122] Zijian Shi and John Cartlidge. Neural stochastic agent-based limit order book simulation: A hybrid methodology. *arXiv preprint arXiv:2303.00080*, 2023.

[123] El Bachir Boukherouaa, Mr Ghiath Shabsigh, Khaled AlAjmi, Jose Deodoro, Aquiles Farias, Ebru S Iskender, Mr Alin T Mirestean, and Rangachary Ravikumar. Powering the digital economy: Opportunities and risks of artificial intelligence in finance, 2021.

[124] Jake Silberg and James Manyika. Notes from the ai frontier: Tackling bias in ai (and in humans). *McKinsey Global Institute*, 1(6), 2019.

[125] Marcus Comiter. Attacking artificial intelligence. *Belfer Center Paper*, 8:2019–08, 2019.

[126] Ines Robledo Costales. Benefits and risks of using ai in trading. `https://www.cityindex.com/en-uk/news-and-analysis/benefits-and-risks-of-ai/`, 2023.

[127] Paraskevi Nousi, Avraam Tsantekidis, Nikolaos Passalis, Adamantios Ntakaris, Juho Kanniainen, Anastasios Tefas, Moncef Gabbouj, and Alexandros Iosifidis. Machine learning for forecasting mid-price movements using limit order book data. *IEEE Access*, 7:64722–64736, 2019.

[128] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[129] Zihao Zhang and Stefan Zohren. Multi-horizon forecasting for limit order books: Novel deep learning approaches and hardware acceleration using intelligent processing units. *arXiv preprint arXiv:2105.10430*, 2021.

[130] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27, 2014.

[131] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

[132] Nikolaos Passalis, Avraam Tsantekidis, Anastasios Tefas, Juho Kanniainen, Moncef Gabbouj, and Alexandros Iosifidis. Time-series classification using neural bag-of-features. In *2017 25th European Signal Processing Conference (EUSIPCO)*, pages 301–305. IEEE, 2017.

[133] Andrea Coletta, Matteo Prata, Michele Conti, Emanuele Mercanti, Novella Bartolini, Aymeric Moulin, Svitlana Vyetrenko, and Tucker Balch. Towards realistic market simulations: A generative adversarial networks approach. In *Proceedings of the Second ACM International Conference on AI in Finance*, ICAIF '21, pages 1–9, November 2021.

[134] Christa Cuchiero, Wahid Khosrawi, and Josef Teichmann. A Generative Adversarial Network Approach to Calibration of Local Stochastic Volatility Models. *Risks*, 8(4):101, December 2020.

[135] Jochen Papenbrock, Peter Schwendner, Markus Jaeger, and Stephan Krügel. Matrix Evolutions: Synthetic Correlations and Explainable Machine Learning for Constructing Robust Investment Portfolios. *The Journal of Financial Data Science*, 3(2):51–69, April 2021.

[136] G. Marti. CORRGAN: Sampling Realistic Financial Correlation Matrices Using Generative Adversarial Networks. *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020.

[137] Richard Roll. Volatility, correlation, and diversification in a multi-factor world. *Journal of Portfolio Management*, 39(2):11–18, 2013.

[138] Aleksandar Bojchevski, Oleksandr Shchur, Daniel Zügner, and Stephan Günnemann. NetGAN: Generating Graphs via Random Walks. In *Proceedings of the 35th International Conference on Machine Learning*, pages 610–619. PMLR, July 2018.

[139] Cristóbal Esteban, Stephanie L. Hyland, and Gunnar Rätsch. Real-valued (Medical) Time Series Generation with Recurrent Conditional GANs. *arXiv:1706.02633 [cs, stat]*, December 2017.

[140] Anvita Gupta and James Zou. Feedback GAN (FBGAN) for DNA: A Novel Feedback-Loop Architecture for Optimizing Protein Functions. *arXiv:1804.01694 [cs, q-bio]*, April 2018.

[141] Malcolm Baker and Jeffrey Wurgler. Investor sentiment in the stock market. *Journal of economic perspectives*, 21(2):129–151, 2007.

[142] David Hirshleifer and Siew Hong Teoh. Herd behaviour and cascading in capital markets: A review and synthesis. *European Financial Management*, 9(1):25–66, 2003.

[143] Markus Leippold, Lujing Su, and Alexandre Ziegler. How index futures and etfs affect stock return correlations. *Available at SSRN 2620955*, 2016.

[144] Milena Vuletić, Felix Prenzel, and Mihai Cucuringu. Fin-gan: Forecasting and classifying financial time series via generative adversarial networks. *Available at SSRN 4328302*, 2023.

[145] Rama Cont, Mihai Cucuringu, Renyuan Xu, and Chao Zhang. Tail-gan: Learning to simulate tail risk scenarios. *Available at SSRN 3812973*, 2022.

[146] Junyi Li, Xintong Wang, Yaoyang Lin, Arunesh Sinha, and Michael Wellman. Generating Realistic Stock Market Order Streams. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(01):727–734, April 2020.

[147] Olof Mogren. C-RNN-GAN: Continuous recurrent neural networks with adversarial training. *arXiv:1611.09904 [cs]*, November 2016.

[148] Jinsung Yoon, Daniel Jarrett, and Mihaela van der Schaar. Time-series Generative Adversarial Networks. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[149] Shuntaro Takahashi, Yu Chen, and Kumiko Tanaka-Ishii. Modeling financial time-series with generative adversarial networks. *Physica A: Statistical Mechanics and its Applications*, 527:121261, 2019.

[150] Xiaomin Li, Vangelis Metsis, Huangyingrui Wang, and Anne Hee Hiong Ngu. Tts-gan: A transformer-based time-series generative adversarial network. In *Artificial Intelligence in Medicine: 20th International Conference on Artificial Intelligence in Medicine, AIME 2022, Halifax, NS, Canada, June 14–17, 2022, Proceedings*, pages 133–143. Springer, 2022.

[151] Mehdi Mirza and Simon Osindero. Conditional Generative Adversarial Nets. *arXiv:1411.1784 [cs, stat]*, November 2014.

[152] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein Generative Adversarial Networks. In *Proceedings of the 34th International Conference on Machine Learning*, pages 214–223. PMLR, July 2017.

[153] Svitlana Vyetrenko, David Byrd, Nick Petosa, Mahmoud Mahfouz, Danial Dervovic, Manuela Veloso, and Tucker Balch. Get real: Realism metrics for robust limit order book market simulations. In *Proceedings of the First ACM International Conference on AI in Finance*, pages 1–8, 2020.

[154] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

[155] Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

[156] Alex M Lamb, Anirudh Goyal ALIAS PARTH GOYAL, Ying Zhang, Saizheng Zhang, Aaron C Courville, and Yoshua Bengio. Professor forcing: A new algorithm for training recurrent networks. *Advances in neural information processing systems*, 29, 2016.

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

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

[159] Jakob Runge. Causal network reconstruction from time series: From theoretical assumptions to practical estimation. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 28(7), 2018.

[160] Jakob Runge, Andreas Gerhardus, Gherardo Varando, Veronika Eyring, and Gustau Camps-Valls. Causal inference for time series. *Nature Reviews Earth & Environment*, 4(7):487–505, 2023.

[161] Uzma Hasan, Emam Hossain, and Md Osman Gani. A survey on causal discovery methods for iid and time series data. *Transactions on Machine Learning Research*, 2023.

[162] Clive WJ Granger. Investigating causal relations by econometric models and cross-spectral methods. *Econometrica: journal of the Econometric Society*, pages 424–438, 1969.

[163] Roxana Pamfil, Nisara Sriwattanaworachai, Shaan Desai, Philip Pilgerstorfer, Konstantinos Georgatzis, Paul Beaumont, and Bryon Aragam. Dynotears: Structure learning from time-series data. In *International Conference on Artificial Intelligence and Statistics*, pages 1595–1605. Pmlr, 2020.

[164] Xiangyu Sun, Oliver Schulte, Guiliang Liu, and Pascal Poupart. Nts-notears: Learning non-parametric dbns with prior knowledge. In *International Conference on Artificial Intelligence and Statistics*, pages 1942–1964. PMLR, 2023.

[165] Yuxiao Cheng, Runzhao Yang, Tingxiong Xiao, Zongren Li, Jinli Suo, Kunlun He, and Qionghai Dai. Cuts: Neural causal discovery from irregular time-series data. In *The Eleventh International Conference on Learning Representations*, 2023.

[166] Yuxiao Cheng, Ziqian Wang, Tingxiong Xiao, Qin Zhong, Jinli Suo, and Kunlun He. Causaltime: Realistically generated time-series for benchmarking of causal discovery. In *The Twelfth International Conference on Learning Representations*, 2024.

[167] Hongming Li, Shujian Yu, and Jose Principe. Causal recurrent variational autoencoder for medical time series generation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pages 8562–8570, 2023.

[168] Scott Lundberg. A unified approach to interpreting model predictions. *arXiv preprint arXiv:1705.07874*, 2017.

[169] James D Hamilton. *Time series analysis*. Princeton university press, 2020.

[170] Eric Zivot and Jiahui Wang. Vector autoregressive models for multivariate time series. *Modeling financial time series with S-PLUS®*, pages 385–429, 2006.

[171] Jinsung Yoon, Daniel Jarrett, and Mihaela Van der Schaar. Time-series generative adversarial networks. *Advances in neural information processing systems*, 32, 2019.

[172] Daniel Jarrett, Ioana Bica, and Mihaela van der Schaar. Time-series generation by contrastive imitation. *Advances in Neural Information Processing Systems*, 34:28968–28982, 2021.

[173] Kashif Rasul, Calvin Seward, Ingmar Schuster, and Roland Vollgraf. Autoregressive denoising diffusion models for multivariate probabilistic time series forecasting. In *International Conference on Machine Learning*, pages 8857–8868. PMLR, 2021.

[174] Ali Seyfi, Jean-Francois Rajotte, and Raymond Ng. Generating multivariate time series with common source coordinated gan (cosci-gan). *Advances in Neural Information Processing Systems*, 35:32777–32788, 2022.

[175] Andrea Coletta, Sriram Gopalakrishnan, Daniel Borrajo, and Svitlana Vyetrenko. On the constrained time-series generation problem. *Advances in Neural Information Processing Systems*, 36, 2023.

[176] Xinyu Yuan and Yan Qiao. Diffusion-ts: Interpretable diffusion for general time series generation. In *The Twelfth International Conference on Learning Representations*, 2024.

[177] Elizabeth Fons, Alejandro Sztrajman, Yousef El-Laham, Andrea Coletta, Alexandros Iosifidis, and Svitlana Vyetrenko. ihypertime: Interpretable time series generation with implicit neural representations. *Transactions on Machine Learning Research*, 2024.

[178] Mukund Sundararajan and Amir Najmi. The many shapley values for model explanation. In *International conference on machine learning*, pages 9269–9278. PMLR, 2020.

[179] Wasim Ahmad, Maha Shadaydeh, and Joachim Denzler. Causal discovery using model invariance through knockoff interventions. In *ICML 2022: Workshop on Spurious Correlations, Invariance and Stability*, 2022.

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

[181] Marco Cuturi and Mathieu Blondel. Soft-dtw: a differentiable loss function for time-series. In *International conference on machine learning*, pages 894–903. PMLR, 2017.

[182] Aapo Hyvärinen, Kun Zhang, Shohei Shimizu, and Patrik O Hoyer. Estimation of a structural vector autoregression model using non-gaussianity. *Journal of Machine Learning Research*, 11(5), 2010.

[183] Jakob Runge, Jobst Heitzig, Norbert Marwan, and Jürgen Kurths. Quantifying causal coupling strength: A lag-specific measure for multivariate time series related to transfer entropy. *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics*, 86(6):061121, 2012.

[184] Dimitris Kugiumtzis. Direct-coupling information measure from nonuniform embedding. *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics*, 87(6):062918, 2013.

[185] Yusuke Tashiro, Jiaming Song, Yang Song, and Stefano Ermon. Csdi: Conditional score-based diffusion models for probabilistic time series imputation. *Advances in Neural Information Processing Systems*, 34:24804–24816, 2021.

[186] Ahmed Alaa, Boris Van Breugel, Evgeny S Saveliev, and Mihaela van der Schaar. How faithful is your synthetic data? Sample-level metrics for evaluating and auditing generative models. In *International Conference on Machine Learning*, pages 290–306. PMLR, 2022.

[187] Arthur Gretton, Karsten Borgwardt, Malte Rasch, Bernhard Schölkopf, and Alex Smola. A kernel method for the two-sample-problem. *Advances in neural information processing systems*, 19, 2006.

[188] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021.

[189] Alex Tank, Ian Covert, Nicholas Foti, Ali Shojaie, and Emily B Fox. Neural granger causality. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(8):4267–4279, 2021.

[190] Saurabh Khanna and Vincent YF Tan. Economy statistical recurrent units for inferring nonlinear granger causality. In *International Conference on Learning Representations*, 2019.

[191] Meike Nauta, Doina Bucur, and Christin Seifert. Causal discovery with attention-based convolutional neural networks. *Machine Learning and Knowledge Extraction*, 1(1):19, 2019.

[192] Yuxiao Cheng, Runzhao Yang, Tingxiong Xiao, Zongren Li, Jinli Suo, Kunlun He, and Qionghai Dai. Cuts: Neural causal discovery from irregular time-series data. In *The Eleventh International Conference on Learning Representations*, 2022.

[193] Shohei Shimizu, Patrik O Hoyer, Aapo Hyvärinen, Antti Kerminen, and Michael Jordan. A linear non-gaussian acyclic model for causal discovery. *Journal of Machine Learning Research*, 7(10), 2006.

[194] Shohei Shimizu, Takanori Inazumi, Yasuhiro Sogawa, Aapo Hyvarinen, Yoshinobu Kawahara, Takashi Washio, Patrik O Hoyer, Kenneth Bollen, and Patrik Hoyer. Directlingam: A direct method for learning a linear non-gaussian structural equation model. *Journal of Machine Learning Research-JMLR*, 12(Apr):1225–1248, 2011.

[195] Jakob Runge, Xavier-Andoni Tibau, Matthias Bruhns, Jordi Muñoz-Marí, and Gustau Camps-Valls. The causality for climate competition. In *NeurIPS 2019 Competition and Demonstration Track*, pages 110–120. Pmlr, 2020.

[196] Wenbo Gong, Joel Jennings, Cheng Zhang, and Nick Pawlowski. Rhino: Deep causal temporal relationship learning with history-dependent noise. In *The Eleventh International Conference on Learning Representations*, 2023.

[197] Edward De Brouwer, Adam Arany, Jaak Simm, and Yves Moreau. Latent convergent cross mapping. In *International Conference on Learning Representations*, 2020.

[198] Alexis Bellot, Kim Branson, and Mihaela van der Schaar. Neural graphical modelling in continuous-time: consistency guarantees and algorithms. In *International Conference on Learning Representations*, 2021.

[199] Felix Schur and Jonas Peters. Decor: Deconfounding time series with robust regression. *arXiv preprint arXiv:2406.07005*, 2024.

[200] Lionel Barnett, Adam B Barrett, and Anil K Seth. Granger causality and transfer entropy are equivalent for gaussian variables. *Physical review letters*, 103(23):238701, 2009.

[201] DF Elliott and KR Rao. Fast fourier transform and convolution algorithms, 1982.

[202] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[203] Fred B Bryant and Paul R Yarnold. Principal-components analysis and exploratory and confirmatory factor analysis. *L. G. Grimm & P. R. Yarnold (Eds.), Reading and understanding multivariate statistics (pp. 99–136). American Psychological Association.*, 1995.

[204] Andrew R Lawrence, Marcus Kaiser, Rui Sampaio, and Maksim Sipos. Data generating process to evaluate causal discovery techniques for time series data. *stat*, 1050:16, 2021.

[205] Peiwen Li, Xin Wang, Zeyang Zhang, Yuan Meng, Fang Shen, Yue Li, Jialong Wang, Yang Li, and Wenwu Zhu. Realtcd: Temporal causal discovery from interventional data with large language model. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pages 4669–4677, 2024.

[206] Timothy J Mosedale, David B Stephenson, Matthew Collins, and Terence C Mills. Granger causality of coupled climate processes: Ocean feedback on the north atlantic oscillation. *Journal of climate*, 19(7):1182–1194, 2006.

[207] Peer Nowack, Jakob Runge, Veronika Eyring, and Joanna D Haigh. Causal networks for climate model evaluation and constrained projections. *Nature communications*, 11(1):1415, 2020.

[208] Steven L Bressler and Anil K Seth. Wiener–granger causality: a well established methodology. *Neuroimage*, 58(2):323–329, 2011.

[209] Anil K Seth, Adam B Barrett, and Lionel Barnett. Granger causality analysis in neuroscience and neuroimaging. *Journal of Neuroscience*, 35(8):3293–3297, 2015.

[210] Monica Billio, Mila Getmansky, Andrew W Lo, and Loriana Pelizzon. Econometric measures of connectedness and systemic risk in the finance and insurance sectors. *Journal of financial economics*, 104(3):535–559, 2012.

[211] Francis X Diebold and Kamil Yılmaz. On the network topology of variance decompositions: Measuring the connectedness of financial firms. *Journal of econometrics*, 182(1):119–134, 2014.

[212] Murat Kocaoglu, Christopher Snyder, Alexandros G Dimakis, and Sriram Vishwanath. Causalgan: Learning causal implicit generative models with adversarial training. *arXiv preprint arXiv:1709.02023*, 2017.

[213] Mengyue Yang, Furui Liu, Zhitang Chen, Xinwei Shen, Jianye Hao, and Jun Wang. Causalvae: Disentangled representation learning via neural structural causal models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9593–9602, 2021.

[214] Licheng Jiao, Yuhan Wang, Xu Liu, Lingling Li, Fang Liu, Wenping Ma, Yuwei Guo, Puhua Chen, Shuyuan Yang, and Biao Hou. Causal inference meets deep learning: A comprehensive survey. *Research*, 7:0467, 2024.

[215] Per Bak. *How nature works: the science of self-organized criticality*. Springer Science & Business Media, 2013.

[216] Helmut Lütkepohl. *New Introduction to Multiple Time Series Analysis*. Springer-Verlag, Berlin, Heidelberg, 2005.

[217] Ludwig Fahrmeir, Thomas Kneib, Stefan Lang, and Brian D Marx. *Regression: Models, Methods and Applications*. Springer, 2013.

[218] Giovanni Bonanno, Fabrizio Lillo, and Rosario N Mantegna. Dynamics of the number of trades of financial securities. *Physica A: Statistical Mechanics and its Applications*, 280(1-2):136–141, 2000.

[219] Tiziana Di Matteo, Tomaso Aste, and Michel M Dacorogna. Long-term memories of developed and emerging markets: Using the scaling analysis to characterize their stage of development. *Journal of banking & finance*, 29(4):827–851, 2005.

[220] Peter Huybers and William Curry. Links between annual, milankovitch and continuum temperature variability. *Nature*, 441(7091):329–332, 2006.

[221] Hege-Beate Fredriksen and Kristoffer Rypdal. Spectral characteristics of instrumental and climate model surface temperatures. *Journal of Climate*, 29(4):1253–1268, 2016.

[222] J. Matias Palva, Alexander Zhigalov, Jonni Hirvonen, Onerva Korhonen, Klaus Linkenkaer-Hansen, and Satu Palva. Neuronal long-range temporal correlations and avalanche dynamics are correlated with behavioral scaling laws. *Proceedings of the National Academy of Sciences*, 110(9):3585–3590, 2013.

[223] Biyu J He. Scale-free brain activity: past, present, and future. *Trends in cognitive sciences*, 18(9):480–487, 2014.

[224] Máté Gyurkovics, Grace M Clements, Kathy A Low, Monica Fabiani, and Gabriele Gratton. Stimulus-induced changes in 1/f-like background activity in eeg. *Journal of Neuroscience*, 42(37):7144–7151, 2022.

[225] Vicente Medel, Martín Irani, Nicolás Crossley, Tomás Ossandón, and Gonzalo Boncompte. Complexity and 1/f slope jointly reflect brain states. *Scientific reports*, 13(1):21700, 2023.

[226] Per Bak, Chao Tang, and Kurt Wiesenfeld. Self-organized criticality: An explanation of the 1/f noise. *Physical review letters*, 59(4):381, 1987.

[227] Per Bak and Kan Chen. Self-organized criticality. *Scientific American*, 264(1):46–53, 1991.

[228] Alex Proekt, Jayanth R Banavar, Amos Maritan, and Donald W Pfaff. Scale invariance in the dynamics of spontaneous behavior. *Proceedings of the National Academy of Sciences*, 109(26):10564–10569, 2012.

[229] Patrick Flandrin. On the spectrum of fractional brownian motions. *IEEE Transactions on information theory*, 35(1):197–199, 1989.

[230] C Gong, D Yao, C Zhang, W Li, and J Bi. Causal discovery from temporal data: An overview and new perspectives. *ACM Computing Surveys*, 2024.

[231] Charles K Assaad, Emilie Devijver, and Eric Gaussier. Survey and evaluation of causal discovery methods for time series. *Journal of Artificial Intelligence Research*, 73:767–819, 2022.

[232] Peter Spirtes, Clark N Glymour, and Richard Scheines. *Causation, prediction, and search*. MIT press, 2000.

[233] Eric V Strobl, Shyam Visweswaran, and Peter L Spirtes. Fast causal inference with non-random missingness by test-wise deletion. *International journal of data science and analytics*, 6:47–62, 2018.

[234] Shanyun Gao, Raghavendra Addanki, Tong Yu, Ryan Rossi, and Murat Kocaoglu. Causal discovery in semi-stationary time series. *Advances in Neural Information Processing Systems*, 36, 2023.

[235] Wenbo Gong, Joel Jennings, Cheng Zhang, and Nick Pawlowski. Rhino: Deep causal temporal relationship learning with history-dependent noise. In *The Eleventh International Conference on Learning Representations*, 2023.

[236] George Sugihara, Robert May, Hao Ye, Chih-hao Hsieh, Ethan Deyle, Michael Fogarty, and Stephan Munch. Detecting causality in complex ecosystems. *science*, 338(6106):496–500, 2012.

[237] Elise Zhang, François Mirallès, Raphaël Rousseau-Rizzi, Di Wu, Arnaud Zinflou, and Benoit Boulet. Enhancing convergent cross mapping: Simple preprocessing for noise-resilient causal discovery. *Authorea Preprints*, 2024.

[238] Gene Yu, Ce Guo, and Wayne Luk. Robust time series causal discovery for agent-based model validation. *arXiv preprint arXiv:2410.19412*, 2024.

[239] John Geweke. Measurement of linear dependence and feedback between multiple time series. *Journal of the American Statistical Association*, 77(378):304–313, 1982.

[240] Jörg Breitung and Bertrand Candelon. Testing for short- and long-run causality: A frequency-domain approach. *Journal of Econometrics*, 132(2):363–378, 2006.

[241] Gang-Jin Wang, Hui-Bin Si, Yang-Yang Chen, Chi Xie, and Julien Chevallier. Time domain and frequency domain granger causality networks: Application to china's financial institutions. *Finance Research Letters*, 39:101662, 2021.

[242] Richard Ashley and Randal J Verbrugge. Frequency dependence in regression model coefficients: an alternative approach for modeling nonlinear dynamic relationships in time series. *Econometric Reviews*, 28(1-3):4–20, 2008.

[243] Debasish Maitra and Saumya Ranjan Dash. Sentiment and stock market volatility revisited: A time–frequency domain approach. *Journal of Behavioral and Experimental Finance*, 15:74–91, 2017.

[244] Maciej J. Kamiński and Katarzyna J. Blinowska. A new method of the description of the information flow in the brain structures. *Biological Cybernetics*, 65(3):203–210, 1991.

[245] Anna Korzeniewska, Mirosław Mańczak, Maciej Kamiński, Katarzyna J. Blinowska, and Szymon Kasicki. Determination of information flow direction among brain structures by a modified directed transfer function method. *Journal of Neuroscience Methods*, 125(1-2):195–207, 2003.

[246] Peter D. Welch. The use of fast fourier transform for the estimation of power spectra: a method based on time averaging over short, modified periodograms. *IEEE Transactions on Audio and Electroacoustics*, 15(2):70–73, 1967.

[247] James Theiler, Stephen Eubank, André Longtin, Bryan Galdrikian, and J. Doyne Farmer. Testing for nonlinearity in time series: the method of surrogate data. *Physica D: Nonlinear Phenomena*, 58(1-4):77–94, 1992.

[248] George E. Uhlenbeck and Leonard S. Ornstein. On the theory of the brownian motion. *Physical Review*, 36(5):823–841, 1930.

[249] Ross A Maller, Gernot Müller, and Alex Szimayer. Ornstein–uhlenbeck processes and extensions. *Handbook of financial time series*, pages 421–437, 2009.

[250] Andrew Ang and Allan Timmermann. Regime changes and financial markets. *Annu. Rev. Financ. Econ.*, 4(1):313–337, 2012.

[251] David R. Brillinger. *Time Series: Data Analysis and Theory*. Holt, Rinehart and Winston, 2nd edition, 1981. Ch. 4.3.

[252] Anne Helby Petersen. Are you doing better than random guessing? a call for using negative controls when evaluating causal discovery algorithms. In Silvia Chiappa and Sara Magliacane, editors, *Proceedings of the Forty-first Conference on Uncertainty in Artificial Intelligence*, volume 286 of *Proceedings of Machine Learning Research*, pages 3465–3479. PMLR, 21–25 Jul 2025.

[253] Milan Erdelj, Enrico Natalizio, Kaushik R Chowdhury, and Ian F Akyildiz. Help from the sky: Leveraging uavs for disaster management. *IEEE Pervasive Computing*, 16(1):24–32, 2017.

[254] Hazim Shakhatreh, Ahmad H Sawalmeh, Ala Al-Fuqaha, Zuochao Dou, Eyad Almaita, Issa Khalil, Noor Shamsiah Othman, Abdallah Khreishah, and Mohsen Guizani. Unmanned aerial vehicles (UAVs): A survey on civil applications and key research challenges. *IEEE Access*, 7:48572–48634, 2019.

[255] Steve Alpern, Alec Morton, and Katerina Papadaki. Patrolling games. *Operations research*, 59(5):1246–1257, 2011.

[256] Hugo Santana, Geber Ramalho, Vincent Corruble, and Bohdana Ratitch. Multi-agent patrolling with reinforcement learning. In *AAMAS*, volume 4, pages 1122–1129. IEEE Computer Society, 2004.

[257] Novella Bartolini, Andrea Coletta, Gaia Maselli, and Ala' Khalifeh. A multi-trip task assignment for early target inspection in squads of aerial drones. *IEEE Transactions on Mobile Computing*, 20(11), 2020.

[258] Novella Bartolini, Andrea Coletta, Gaia Maselli, and Matteo Prata. Tamara: A task management and routing algorithm for fanets. *IEEE Transactions on Mobile Computing*, pages 1–13, 2023.

[259] Jing Li, Yonghua Xiong, and Jinhua She. Uav path planning for target coverage task in dynamic environment. *IEEE Internet of Things Journal*, 10(20):17734–17745, 2023.

[260] Yann Chevaleyre. Theoretical analysis of the multi-agent patrolling problem. In *Proceedings of IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, 2004.

[261] Soroush Alamdari, Elaheh Fata, and Stephen L Smith. Persistent monitoring in discrete environments: Minimizing the maximum weighted latency between observations. *The International Journal of Robotics Research*, 33(1):138–154, 2014.

[262] Krishna Kalyanam, Satyanarayana Manyam, Alexander Von Moll, David Casbeer, and Meir Pachter. Scalable and exact MILP methods for UAV persistent visitation problem. In *IEEE CCTA*, 2018.

[263] Jürgen Scherer and Bernhard Rinner. Multi-robot patrolling with sensing idleness and data delay objectives. *Journal of Intelligent & Robotic Systems*, 99:949–967, 2020.

[264] Ahmad Bilal Asghar, Stephen L Smith, and Shreyas Sundaram. Multi-robot routing for persistent monitoring with latency constraints. In *IEEE ACC*, pages 2620–2625, 2019.

[265] Yehuda Elmaliach, Noa Agmon, and Gal A Kaminka. Multi-robot area patrol under frequency constraints. *Annals of Mathematics and Artificial Intelligence*, 57:293–320, 2009.

[266] Chuanbo Yan and Tao Zhang. Multi-robot patrol: A distributed algorithm based on expected idleness. *International Journal of Advanced Robotic Systems*, 13(6):1729881416663666, 2016.

[267] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[268] Hao Wang, Chi Harold Liu, Haoming Yang, Guoren Wang, and Kin K Leung. Ensuring threshold aoi for uav-assisted mobile crowdsensing by multi-agent deep reinforcement learning with transformer. *IEEE/ACM Transactions on Networking*, 2023.

[269] Jingzhi Hu, Hongliang Zhang, Lingyang Song, Robert Schober, and H Vincent Poor. Cooperative Internet of UAVs: Distributed trajectory design by multi-agent deep reinforcement learning. *IEEE Transactions on Communications*, 68(11):6807–6821, 2020.

[270] Xin Jin and Jiawei Han. *K-Means Clustering*, pages 563–564. Springer US, Boston, MA, 2010.

[271] Nicos Christofides. Worst-case analysis of a new heuristic for the travelling salesman problem. *Operations Research Forum*, 3, 1976.

[272] IEA. Electricity 2024. `https://www.iea.org/reports/electricity-2024,Licence:CCBY4.0`, 2024.

[273] D. Patterson, J. Gonzalez, Q. Le, C. Liang, L. M. Munguia, D. Rothchild, D. So, M. Texier, and J. Dean. Carbon emissions and large neural network training. *arXiv preprint arXiv:2104.10350*, 2021.

[274] A. de Vries. The growing energy footprint of artificial intelligence. *Joule*, 7(10):2191–2194, 2023.

[275] Z. Liu, M. Lin, A. Wierman, S. H. Low, and L. Andrew. Greening geographical load balancing. *ACM SIGMETRICS Performance Evaluation Review*, 39(1):193–204, 2011.

[276] Z. Liu, Y. Chen, C. Bash, A. Wierman, D. Gmach, Z. Wang, M. Marwah, and C. Hyser. Renewable and cooling aware workload management for sustainable data centers. In *SIGMETRICS/PERFORMANCE*, 2012.

[277] P. X. Gao, A. R. Curtis, B. Wong, and S. Keshav. It's not easy being green. *ACM SIGCOMM Computer Communication Review*, 42(4):211–222, 2012.

[278] S. Su, Z. Zhou, T. Ouyang, R. Zhou, and X. Chen. Learning to be green: Carbon-aware online control for edge intelligence with colocated learning and inference. In *IEEE ICDCS*, 2023.

[279] E. Strubell, A. Ganesh, and A. McCallum. Energy and policy considerations for modern deep learning research. In *AAAI*, volume 34, pages 13693–13696, 2020.

[280] S. Nguyen, B. Zhou, and Y.D. Liu. S. towards sustainable large language model serving. In *ACM HotCarbon*, 2024.

[281] Z. Zhou, F. Liu, R. Zou, J. Liu, H. Xu, and H. Jin. Carbon-aware online control of geo-distributed cloud services. *IEEE Transactions on Parallel and Distributed Systems*, 27(9):2506–2519, 2015.

[282] Google. Environmental report 2023. `https://www.gstatic.com/gumdrop/sustainability/google-2023-environmental-report.pdf`.

[283] M. Neely. *Stochastic network optimization with application to communication and queueing systems*. Springer Nature, 2022.

[284] P. Li, J. Yang, A. Wierman, and S. Ren. Towards environmentally equitable AI via geographical load balancing. In *ACM e-Energy*, 2024.

[285] A. Lacoste, A. Luccioni, V. Schmidt, and T. Dandres. Quantifying the carbon emissions of machine learning. *arXiv preprint arXiv:1910.09700*, 2019.

[286] Í. Goiri, K. Le, M. Haque, R. Beauchea, T. D. Nguyen, J. Guitart, J. Torres, and R. Bianchini. Greenslot: scheduling energy consumption in green datacenters. In *ACM/IEEE SC*, 2011.

[287] H. B. Beytur, A. G. Aydin, G. de Veciana, and H. Vikalo. Optimization of offloading policies for accuracy-delay tradeoffs in hierarchical inference. In *IEEE INFOCOM 2024*, pages 1989–1998, 2024.

[288] B. Güler and A. Yener. A framework for sustainable federated learning. In *IEEE WiOpt*, 2021.

[289] P. Wiesner, R. Khalili, D. Grinwald, P. Agrawal, L. Thamsen, and O. Kao. Fedzero: Leveraging renewable excess energy in federated learning. In *e-Energy*, 2024.

[290] Y. Li, T. Ouyang, X. Chen, and X. Cao. Fedcarbon: Carbon-efficient federated learning with double flexible controls for green edge AI. In *IEEE/ACM IWQoS*, 2024.

[291] M.S. Kumar, Y. Bao, X. Wang, S. Drew, et al. Energy-efficient federated learning with dynamic model size allocation. *arXiv preprint arXiv:2411.15481*, 2024.

[292] W. A. Hanafy, Q. Liang, N. Bashir, D. Irwin, and P. Shenoy. Carbonscaler: Leveraging cloud workload elasticity for optimizing carbon-efficiency. *ACM POMACS*, 2023.

[293] A. Radovanović, R. Koningstein, I. Schneider, B. Chen, A. Duarte, B. Roy, D. Xiao, M. Haridasan, P. Hung, N. Care, et al. Carbon-aware computing for datacenters. *IEEE Transactions on Power Systems*, 38(2):1270–1280, 2022.

[294] D. Yi, X. Zhou, Y. Wen, and R. Tan. Toward efficient compute-intensive job allocation for green data centers: A deep reinforcement learning approach. In *IEEE ICDCS*, 2019.

[295] G. Wilkins, S. Keshav, and R. Mortier. Hybrid heterogeneous clusters can lower the energy consumption of llm inference workloads. In *ACM e-Energy*, 2024.

[296] A. A. Chien, L. Lin, H. Nguyen, V. Rao, T. Sharma, and R. Wijayawardana. Reducing the carbon impact of generative AI inference (today and in 2035). In *Proceedings of the 2nd workshop on sustainable computer systems*, pages 1–7, 2023.

[297] J. Lim, N. B. Lakshminarayana, H. Kim, W. Song, S. Yalamanchili, and W. Sung. Power modeling for GPU architectures using mcpat. *ACM Transactions on Design Automation of Electronic Systems*, 19(3):1–24, 2014.

[298] S. Savazzi, V. Rampa, S. Kianoush, and M. Bennis. An energy and carbon footprint analysis of distributed and federated learning. *IEEE Transactions on Green Communications and Networking*, 7(1):248–264, 2022.

[299] L. Deng, G. Li, S. Han, L. Shi, and Y. Xie. Model compression and hardware acceleration for neural networks: A comprehensive survey. *Proceedings of the IEEE*, 108(4):485–532, 2020.

[300] H. Jiang and C. Dovrolis. Passive estimation of tcp round-trip times. *ACM SIGCOMM Computer Communication Review*, 32(3):75–88, 2002.

[301] S. Sengupta, H. Kim, and J. Rexford. Continuous in-network round-trip time monitoring. In *Proceedings of the ACM SIGCOMM 2022 Conference*, pages 473–485, 2022.

[302] N. Lazic, B. Frey, and P. Aarabi. Solving the uncapacitated facility location problem using message passing algorithms. JMLR, 2010.

[303] S. Guha and S. Khuller. Greedy strikes back: Improved facility location algorithms. *Journal of Algorithms*, 31(1):228–248, 1999.

[304] N. Karmarkar. A new polynomial-time algorithm for linear programming. In *ACM STOC*, 1984.

[305] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models. In *IEEE/CVF CVPR*, pages 10684–10695, 2022.

[306] Our World in Data. Carbon intensity of electricity generation. `https://ourworldindata.org/grapher/carbon-intensity-electricity`, 2024.

[307] S. Yan, X. Wang, X. Zheng, Y. Xia, D. Liu, and W. Deng. ACC: Automatic ECN tuning for high-speed datacenter networks. In *ACM SIGCOMM*, 2021.

Appendix A

Additional Research Contributions

A.1 Patrolling Heterogeneous Targets with FANETs

Abstract

Periodic patrolling by Unmanned Aerial Vehicles (UAVs) is crucial for security and monitoring applications, including border surveillance, port security, or environmental monitoring. However, existing approaches often fall short when confronted with the intricate demands of these applications. They tend to overlook the heterogeneity in visit frequency requirements across different areas, compromising overall security effectiveness. We contribute a comprehensive formulation of the patrolling problem alongside new evaluation metrics that accurately assess patrolling strategies' performance. We propose a heuristic called Balanced Clusters Patrolling (BCP), which partitions the points of interest into threshold-based clusters and assigns UAVs to match each cluster's demands while meeting the visit frequency constraints. Our extensive simulations under a variety of scenarios reveal that BCP consistently outperforms existing solutions across all significant performance metrics, enhancing performance by up to 92% in reducing total delay.

This work has been published in the IEEE INFOCOM 2024 Conference Workshops [28].

A.1.1 Introduction

In recent years, Flying Ad-Hoc Networks (FANETs) have become increasingly popular for their use in critical applications such as search and rescue, monitoring natural disasters (e.g., earthquakes, floods, or wildfires) [253], and patrolling of borders, and coasts [254]. For example, UAVs are employed by the European Border and Coast Guard Agency to patrol the Mediterranean, aiding in migrant rescue and fighting migrant smuggling. UAVs are also incrementally utilized to monitor the growing incidence of destructive wildfires. In these situations, early detection and regular patrolling are essential for effective management and response.

Patrolling is the act of traveling around an area, at regular intervals, to supervise it [255] and collect sensing data or detect anomalies. Employing UAVs as dynamic robotic patrollers, rather than relying on human personnel, mitigates risks for humans in critical environments and reduces the impact of the human factor, its limitations, and the mistakes it may generate.

Previous solutions fail to address all the intricacies related to accurately monitoring the environment and the constraints of the application. In particular, existing works neglect the heterogeneity

of the points of interest and of their intervisit time requirements, whose violation directly compromises security. We propose a heuristic solution that preserves the strength of past works [256, 257], whilst addressing the challenges of the patrolling problem in real-world security-critical scenarios. We provide a novel formulation of the problem and propose a polynomial-time heuristic called Balanced Clusters Patrolling (BCP). The heuristic partitions the points of interest, taking into account the heterogeneity of the visit frequency requirements. It then assigns UAVs to the obtained clusters, considering their patrolling workload. We formulate novel evaluation metrics that accurately reflect the performance of patrolling solutions. Through extensive simulations conducted in a wide range of scenarios, we carry out performance comparisons of BCP with several previous solutions and baselines. The evaluation shows that BCP outperforms existing solutions in all the devised performance metrics.

The original contributions of this work are the following:

- We give a formal definition of the patrolling problem, which extends prior formulation considering heterogeneous constraints on the inter-visit time of the target to be monitored.
- We present a two-phase patrolling heuristic: first, clustering inspection points, and then assigning a proper number of UAVs to the clusters to exploit efficient routes.
- We introduce novel performance metrics for patrolling tasks, emphasizing crucial aspects that have frequently been overlooked in evaluating patrolling solutions.
- We evaluate our solution and compare it to other state-of-the-art approaches through extensive simulations. The experiments show that our solution outperforms previous work and baseline approaches in terms of all the key metrics, in all the considered scenarios.

A.1.2 Related Work

The patrolling problem has been tackled in various ways in the literature, addressing either simple target inspections or persistent surveillance. While there is ample research on the first problem, we will only mention some works falling into this category due to space limitations. In particular, Bartolini et al. in [258] and [257] developed polynomial-time algorithms to minimize target inspection delays with or without persistent connectivity. Li et al. in [259] introduced a path-planning solution for target coverage using an ant colony optimization algorithm in dynamic environments. These works, although primarily focused on single inspections, can be adapted to persistent patrolling solutions through the repetitive implementation of the one-shot target inspection scheduling they provide. In the second category of works, which focuses on the persistent patrolling problem, a common performance optimization criterion is *idleness*, also known as *Age of Information, AOI*. This metric evaluates the time elapsed between two consecutive visits to a target requiring inspection. Specifically, much of the research in this area, including the work of Chevaleyre [260] concentrates on minimizing either the maximum idleness experienced at any target or the average idleness across all targets. More generally, other studies minimize the worst weighted idleness, or the average weighted idleness, where the weight is a measure of the priority of the target, penalizing the idleness proportionally [261]. An optimal formulation for the minimum worst-weighted idleness was proposed by Kalyanamin et al. in [262]. However, computing optimal solutions takes too much time, even for small instances of the problem. Moreover, none of these works considers an inter-visit constraint on the target. Scherer et al. [263], along with the idleness criterion, also consider the delay, i.e., the time passing between sensing the data and its arrival at the base station. Another shade of

the problem is addressed by Asghar et al. in [264] with an approximate algorithm to compute the minimum number of robots required to satisfy the idleness constraints.

Our focus is to solve a much more generic patrolling variant where every target node in the area is assumed to have a required visit frequency. This problem is known in the literature as frequency-based patrolling, as defined by Elmaliach et al. in [265]. Three performance criteria are adopted: maximize uniformity (all targets visited with the same frequency); maximize average frequency; and maximize minimal frequency. Elmaliach et al [265] restrict their work to the case in which the targets are placed in a grid.

Yan et al. [266] address the frequency-based patrolling assuming communication between UAVs, coming up with a distributed algorithm that is fault-tolerant and scalable. Another class of approaches is based on Deep Reinforcement Learning [267]. In the literature, there are records of attempts to use RL in patrolling, either frequency-based [256, 268], or just weighted [269]. The mentioned RL approaches, however, do not guarantee meeting heterogeneous constraints on idleness and provide poor scalability in large-scale scenarios such as the ones addressed here.

A.1.3 Preliminaries

The Patrolling Problem

Let $\mathcal{U}$ be the set of UAVs involved in the patrolling mission and let $\mathcal{I}$ be the set of *inspection points* (IPs). Let $a_p(t)$ be the function representing the *age of information* (AOI) at time t, i.e., the time (seconds) elapsing between two consecutive visits of any UAV to an IP $p \in \mathcal{I}$.

An instance of the patrolling problem is a 3-tuple $\langle \mathcal{U}, \mathcal{I}, \theta \rangle$ where $\theta : \mathcal{I} \to \mathbb{R}$ defines the *inter-visit threshold* of each $p \in \mathcal{I}$ as the AOI it can tolerate. At any point in time t, if $a_p(t) > \theta(p)$ a *violation* of the IP p happened.

All UAVs have a predefined starting position. The task is to plan their trajectories so that their sensors inspect the set $\mathcal{I}$, optimizing the age of information and keeping it below the desired inter-visit threshold.

A *visit* or *inspection* to an IP p of a UAV u is *valid* if $d(\dot{p}, \dot{u}) \leq r_u$ where $d(\cdot, \cdot)$ is the Euclidean distance between the coordinates in the euclidean space of the IP and the UAV, respectively, and r_u is the sensing range of the UAV. In other words, the UAV should hover on the IP at a distance not higher than its sensing range.

The solution to the instance is a patrolling policy $\pi : \mathbb{U} \to \mathcal{I}^T, \forall u \in \mathcal{U}$, mapping a UAV to a list of inspection points to visit, where T is the maximum tour length allowed. During the mission, the UAV will repeat its policy cyclically.

Evaluation metrics

Let M be the mission duration expressed in seconds. There are several optimization criteria for each inspection point that a patrolling policy may want to *minimize*. We refer to a generic criteria for the IP p as $c_i(p)$:

- the ***cumulative AOI*** of the IP to keep the sensed information as fresh as possible. This corre-

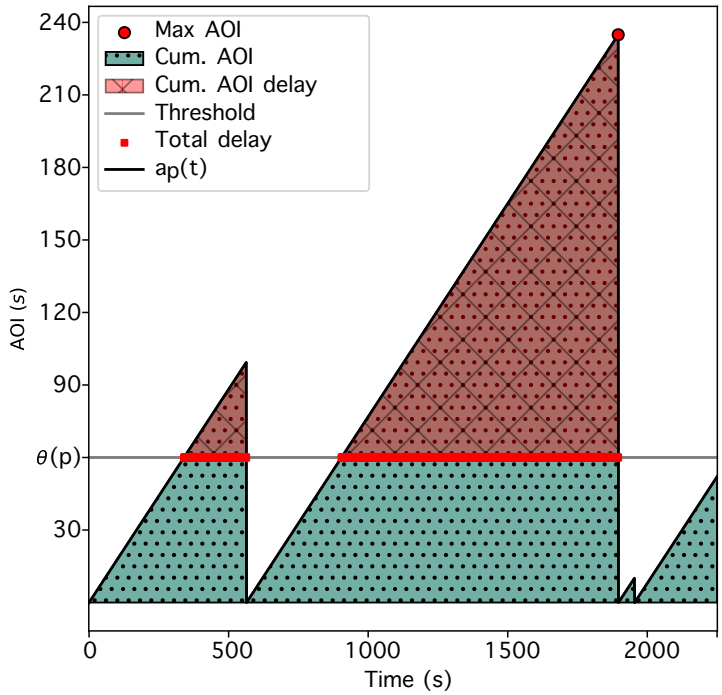


Figure A.1: The AOI of the IP p and its evaluation metrics.

sponds to the turquoise area under the black curve in Figure A.1.

$$c_1(p) : \int_{t=0}^{M} a_p(t)\, dt \tag{A.1}$$

- the ***cumulative AOI delay*** reflecting the freshness of the information when a violation occurred. This corresponds to the red area between $a_p(t)$ and the threshold $\theta(p)$ in Figure A.1.

$$c_2(p) : \int_{t=0}^{M} a_p(t) \cdot \mathbb{1}(a_p(t) > \theta(p))\, dt \tag{A.2}$$

  where $\mathbb{1}(\cdot)$ is an indicator function.
- the ***total delay*** reflecting the amount of time that the target remained in violation. This corresponds to the sum of the lengths of the red segments, which are the bases of the red triangles in Figure A.1.

$$c_3(p) : \int_{t=0}^{M} \mathbb{1}(a_p(t) > \theta(p))\, dt \tag{A.3}$$

- the ***max AOI*** corresponding to the worst delay in collecting information. This corresponds to the red dot in Figure A.1.

$$c_4(p) : \max_{0 \le t \le M} a_p(t) \tag{A.4}$$

An aggregation function can be defined to summarize the performance over all the inspection points and measure the performance of the patrolling policy. In evaluating the effectiveness of patrolling strategies, considering an average performance metric may provide a broad overview of the IPs' condition. However, in scenarios where preventing intrusions at IPs is paramount, relying

solely on average performance might not capture potential breaches. In such cases, a more insightful approach involves aggregating the performance metrics based on the poorly guarded IP.

The ***maximum*** (worst) criterion over the IPs:

$$\min \max_{p \in \mathcal{I}} c_i(p) \tag{A.5}$$

approach prioritizes identifying and addressing weaknesses in the patrolling policy. By focusing on the performance of the most neglected IP, the evaluation of a policy becomes more targeted to potential security risks.

In scenarios where the threshold is not uniform over all the IPs, it is meaningful to assess the performance of a patrolling strategy using what is referred to as the *AOI ratio*, denoted as $\widehat{\text{AOI}}$ hereafter. The $\widehat{\text{AOI}}$ function for an IP p is defined as the ratio between the AOI function and its threshold, i.e., $\hat{a}_p(t) = a_p(t)/\theta(p)$. It's important to note that this function lies in the interval $[0, M/\theta(p)]$, and $\hat{a}_p(t) = 1$ when the IP's deadline has expired. For all other values, it indicates how many $\theta(p)$ intervals the IP was violated.

As an example, take two IPs p_1, p_2 having $\theta(p_1) = 50, \theta(p_2) = 100$. Let $t = 200$, and observe $a_{p_1}(200) = a_{p_1}(200) = 100$. In absolute terms, both p_1 and p_2 will have accumulated the same age, but relatively to their threshold, p_1 is much more urgent than p_2. Specifically, they will have AOI ratio of 2 and 1, respectively. An AOI ratio of 2 for IP p_1 implies that it experienced a violation twice its threshold, indicating a potential performance concern. On the other hand, an AOI ratio of 1 for IP p_2 suggests a relatively better adherence to its threshold. Therefore, by evaluating the AOI ratios of various IPs, we can gain insights into the degree of compliance with their individual thresholds and identify areas that may require attention in the patrolling strategy. In the experimental section below, we will use the $\widehat{\text{AOI}}$ function.

A.1.4 BCP: Balanced Clusters Patrolling

The addressed problem generalizes the Travelling Salesman Problem (TSP), a classic NP-hard problem in combinatorial optimization, which involves determining the shortest possible route that visits a set of cities and returns to the starting city. Therefore, our problem is at least as hard as the TSP.

For this reason, in this section, we introduce the BCP (Balanced Clusters Patrolling) Algorithm, a polynomial-time algorithm to efficiently look for a solution to a patrolling instance with inter-visit thresholds. The algorithm initially determines the optimal clustering of IPs within the area of interest, subsequently identifying the optimal number of UAVs to assign to these clusters to minimize delays.

We detail the proposed heuristic in Algorithm 4. The algorithm takes as input the set of UAVs $\mathcal{U}$, the set of inspection points IPs, and their thresholds θ.

The initial phase of the algorithm (**line 1**) involves the computation of the residuals matrix R. This matrix plays a crucial role in the subsequent clustering of the IPs within the targeted area. In constructing R, the algorithm assigns to $R[i, j]$ the distance between all pairs of IPs (i, j), subtracting the minimum threshold of the pair. This approach increases the proximity of IPs that are close in space and have similar inter-visit thresholds.

In the subsequent phase (**lines 2 - 3**), the optimal number of clusters for IPs is determined

Algorithm 4 Balanced Clusters Patrolling (BCP)

Require: a set of UAVs $\mathcal{U}$, a set of IPs $\mathcal{I}$, a set of thresholds θ.
Ensure: a patrolling policy π for all the UAVs.
// distance matrix for clustering
1: $R = \{d(i,j) - \min(\theta(i), \theta(j))\}_{i,j \in \mathcal{I}}$
// discover best number of clusters
2: $k \leftarrow$ find $k \in [1, |\mathcal{U}|]$ with *elbow method* executing *k-means* on R
3: $\mathcal{C} \leftarrow$ clusters of the IPs with *k-means* on R
// discover best number of drones per cluster
4: $N \leftarrow \{|c|\}_{c \in \mathcal{C}}$
5: $M \leftarrow \{\min_{p \in c} tsp\_cost(c)/\theta(p)\}_{c \in \mathcal{C}}$
6: $M \leftarrow M / \sum M$
7: $A \leftarrow \{1\}_{c \in |\mathcal{C}|}$
8: $A \leftarrow A + \lfloor M \cdot (|\mathcal{U}| - |\mathcal{C}|) \rfloor$
9: $A[A > N] = \text{minimum}(A, N)[A > N]$
10: Assign remaining $|\mathcal{U}| - \sum A$ UAVs with *round-robin* on clusters sorted by decreasing M, if $A[i] + 1 < N[i]$ $\forall i \in [1, |\mathcal{C}|]$
// assign drones to clusters
11: $\pi \leftarrow \emptyset$
12: **for** $i \in [1, |\mathcal{C}|]$ **do**
13: $\quad k \leftarrow A[i]$
14: $\quad \widehat{\mathcal{C}} \leftarrow$ clusters of the IPs in $\mathcal{C}[i]$ with *k-means* on R
15: $\quad \pi(u) \leftarrow$ *tsp* tour of $\widehat{\mathcal{C}}[u]$ $\forall u \in [1, k]$
16: **end for**
17: **return** π

using the *k-means* algorithm [270] and the elbow method. Specifically, by varying the number of potential clusters k within the range $[1, |\mathcal{U}|]$, we transition from grouping all UAVs into a single cluster to assigning each UAV its own cluster. We identify the elbow point — the k value at which the *k-means* inertia (i.e., the sum of squared distances of samples to their closest centroid) begins to exhibit diminishing returns. The elbow point is the optimal number of clusters that we consider for the IPs clustering process. In (**line 3**) we use k to compute the set of clusters $\mathcal{C}$.

In the next phase (**lines 4 - 10**), the objective is to determine the optimal number of UAVs to allocate to each cluster, taking into account the specific characteristics of the clusters. Specifically, M vector is introduced to reflect the demand for UAVs in each cluster (**line 5**). This demand is proportional to the ratio of the cost of the TSP solution approximated using the Christofides heuristic [271], measured in terms of edge length, to the inter-visit thresholds of each IP. This prioritization mechanism places higher importance on clusters with longer TSP paths and more immediate thresholds for the IPs contained within the cluster.

Let A be the assignment vector; initially, it is a vector containing 1 in all positions, indicating one UAV per cluster (**line 7**). Subsequently, A is incremented with a number of UAVs proportional to M (**line 8**). Adjustments are made for cases where more UAVs are assigned to a cluster than the number of IPs in the cluster. For instance, A is capped to the minimum between the number of IPs in the cluster and the initially assigned UAVs (**line 9**). We assign the remaining UAVs to clusters in a round-robin fashion, sorted by their priority in M (**line 10**). This approach prevents additions in clusters where the inclusion may exceed the number of IPs in the cluster.

Finally (**lines 11 - 15**), UAVs are allocated to their respective clusters. Specifically, for each cluster i, a k-means on R is executed, with $k = A[i]$, indicating that the number of sub-clusters is the same as the number of UAVs in the cluster. UAVs within a sub-cluster follow a TSP tour within their assigned cluster. The output of the algorithm is π, a function mapping every UAV to their tour.

Theorem A.1.1 (Time Complexity of BCP)**.** *The BCP algorithm has polynomial time complexity in the number of UAVs $u = |\mathcal{U}|$ and IPs $n = |\mathcal{I}|$.*

Proof sketch. The heaviest two operations in the algorithm are the computation of the clusters through *k-means* , approximated with the Lloyd method [270], and the execution of the Christofides [271] heuristics for the TSP problem.

In our algorithm, the complexity of a single execution of the *k-means* on the set of IPs is $\mathcal{O}(nu)$, since it runs for a constant number of iterations over n 2-dimensional data points with a number of centroids $k \leq u$. In line 2 of the algorithm, *k-means* is executed for an increasing number of clusters up to the number of UAVs u. Overall, the cost of the line is $\mathcal{O}(nu^2)$.

The computational complexity of the TSP with the Christofides heuristic is cubic in the number of input coordinates. In line 5, the heuristic is used for at most u clusters, each containing at most n coordinates. Thus the cost is $\mathcal{O}(un^3)$.

The complexity of the loop in line 12 is dominated by the execution of the TSP heuristic for at most u clusters, and for each UAV u, thus a computational cost of $\mathcal{O}(u^2n^3)$. So the total complexity of the algorithm, assuming that the number of UAVs is upper bounded by the number of IPs in the area $u \leq n$, is given by $\mathcal{O}(nu^2 + un^3 + u^2n^3) = \mathcal{O}(n^5)$. □

A.1.5 Performance Evaluation

In this section, we evaluate the performance of BCP by means of simulations and compare it with previous solutions and baseline approaches.

Simulator. The experiments described in this section were carried out on DRONET, a simulation framework including libraries to visualize and process ad-hoc networks, simulate patrolling scenarios, and implement patrolling policies, communication, and task management algorithms. DRONET is publicly available for download[1] to ensure the reproducibility of our analysis.

Simulation Setup. In our simulated environment, we explore patrolling scenarios within a square area of interest measuring $1500\,m \times 1500\,m$. A set of 40 IPs with heterogeneous inter-visit thresholds is distributed in the area based on diverse deployment schemes as described below. The number of UAVs varies between 1 and 25 and is fixed at 10 when a particular number is required. The mission duration M is set to 2250 seconds (37.5 minutes) with UAVs flying at a speed of $5\,m/s$. Throughout the simulation, unless explicitly stated, UAV batteries last for the entire mission duration M.

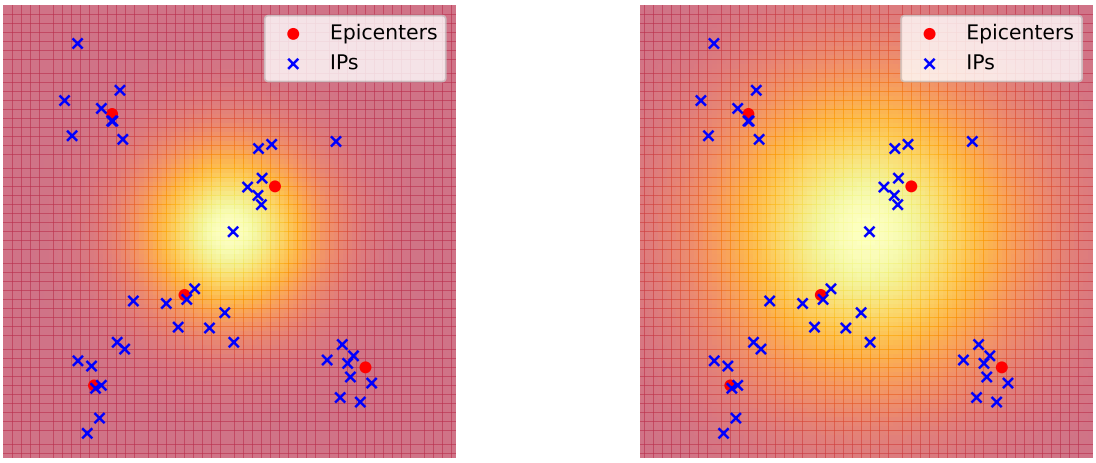
The spatial distribution of IPs follows two models:

- *Uniform deployment:* IPs are randomly distributed with uniform probabilities across the square area, reflecting a dispersed distribution throughout the entire region.
- *Clustered deployment:* Five epicenters are chosen randomly, around which IPs are evenly distributed, representing localized concentrations of interest. This scenario simulates situations where specific regions have a higher density of critical points, necessitating focused patrolling efforts.

The thresholds for the IP-related inter-visit delay follow two distributions:

- *Gaussian thresholds:* The IPs located near the center of the area necessitate more frequent inspection. To model the extent of the zone with this increased requirement, we use an *urgency*

[1] `https://github.com/giuseppemasi99/DRONET-for-Patrolling`



(a) Urgency factor 1. **(b)** Urgency factor 2.

Figure A.2: Clustered targets and Gaussian thresholds. The brighter color in the area signifies a more urgent inspection area.

factor that proportionally expands the area according to a Gaussian distribution. Figure A.2a illustrates a scenario with a lower urgency factor, resulting in more urgent thresholds concentrated in the middle of the area. In contrast, Figure A.2b shows the case of a higher urgency factor, leading to an expansion of the area requiring increased inspection frequency.

- *Clustered thresholds:* This distribution is linked to the clustered spatial deployment, where thresholds in the five clusters remain constant at $[20\,s, 20\,s, 200\,s, 600\,s, 600\,s]$.

Key Performance Indicators (KPIs). Our experimental campaign is evaluated based on the metrics described in Section A.1.3, as articulated in Equation (A.5). To measure the delay accumulated by the IPs, we utilize the estimated $\widehat{\text{AOI}}$ ratio which is formally introduced in the same section.

State of the Art Comparison

We compare the performance of BCP with respect to previous literature solutions and baseline approaches. In particular, we include two intuitive baseline methods:

- **Go-Max-AOI.** Free UAVs dynamically select different IPs based on the highest AOI values, with no collisions due to mutual awareness and ubiquitous communication with all UAVs.
- **Go-Max-AOI-Ratio.** Each UAV selects as its next IP the one with the highest ratio between its AOI and threshold. Ubiquitous communication is also assumed.

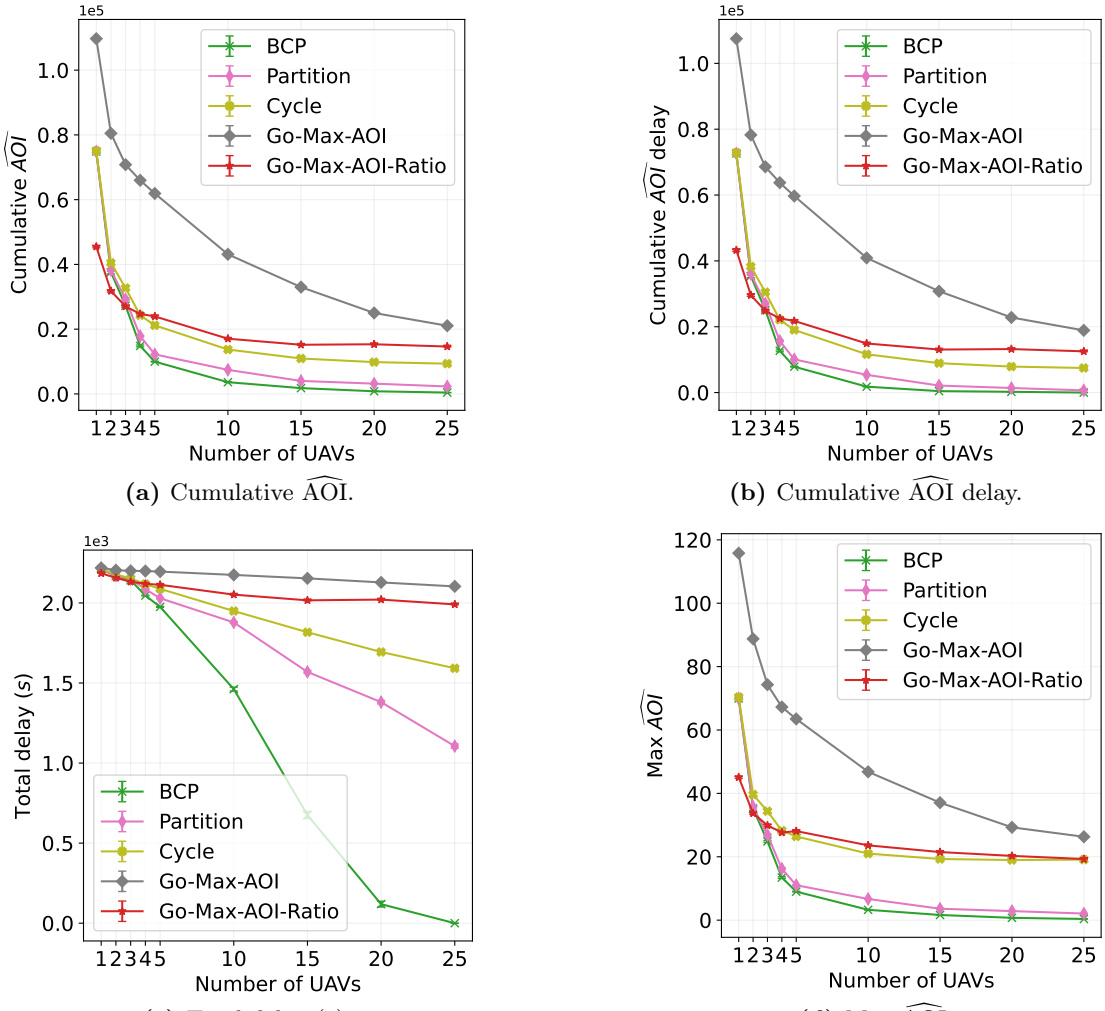
We also run comparisons with the following state-of-the-art solutions proposed in [257, 260]:

- **Cycle.** Introduced in [260], it runs the Christofides algorithm [271] to obtain an approximate solution of the TSP problem on the entire set of IPs. The UAVs sequentially follow the same TSP tour, maintaining a constant and uniform gap between each other.
- **Partition.** Introduced in [260], it computes a partition of the set of IPs. One UAV is assigned to each part and internally follows the approximate solution of the TSP.
- **Greedy-and-Prune.** Originally designed to ensure early inspection of IPs using multiple subsequent trips under battery constraints, the Greedy-and-Prune (GaP) [257] algorithm optimizes the selection of the UAV trajectories to minimize the average inspection delay. In our study, to produce a patrolling solution, we let the UAV team execute the GaP schedule iteratively.

Simulation Results

The experimental campaign presented here involves several scenarios, each differing in the positioning of inspection points within the area and the distribution of thresholds assigned to these points. Results are randomized over 30 runs, in which the position of the IPs and visit requirement follow the selected model.

Clustered IPs, Clustered thresholds. Figure A.3 shows the results of our approach over 40 IPs deployed in a clustered manner, with the number of UAVs employed varying in the range $[1, 25]$. BCP outperforms alternative approaches across various metrics, including Cumulative AOI (Figure A.3a), Cumulative AOI delay (Figure A.3b), total delay (Figure A.3c), and Max AOI (Figure A.3d). Figure A.3a highlights that to maintain AOI freshness below 2000, only 4 UAVs are needed with BCP, while Go-Max-AOI and Go-Max-AOI-Ratio require 10 and 25 UAVs, respectively. As observed in Figure A.3c, our approach is highly effective also in reducing the total delay as the number of UAVs increases. For 15 and 20 UAVs, we achieve a total delay that is 57% and 92% lower than the best state-of-the-art algorithm (Partition). With 25 UAVs, the total delay reaches 0s. This superiority is particularly evident in scenarios with high variance in threshold deployments, such as clustered deployments with thresholds ranging from very low to very high values.



(a) Cumulative $\widehat{\text{AOI}}$.

(b) Cumulative $\widehat{\text{AOI}}$ delay.

(c) Total delay (s).

(d) Max $\widehat{\text{AOI}}$.

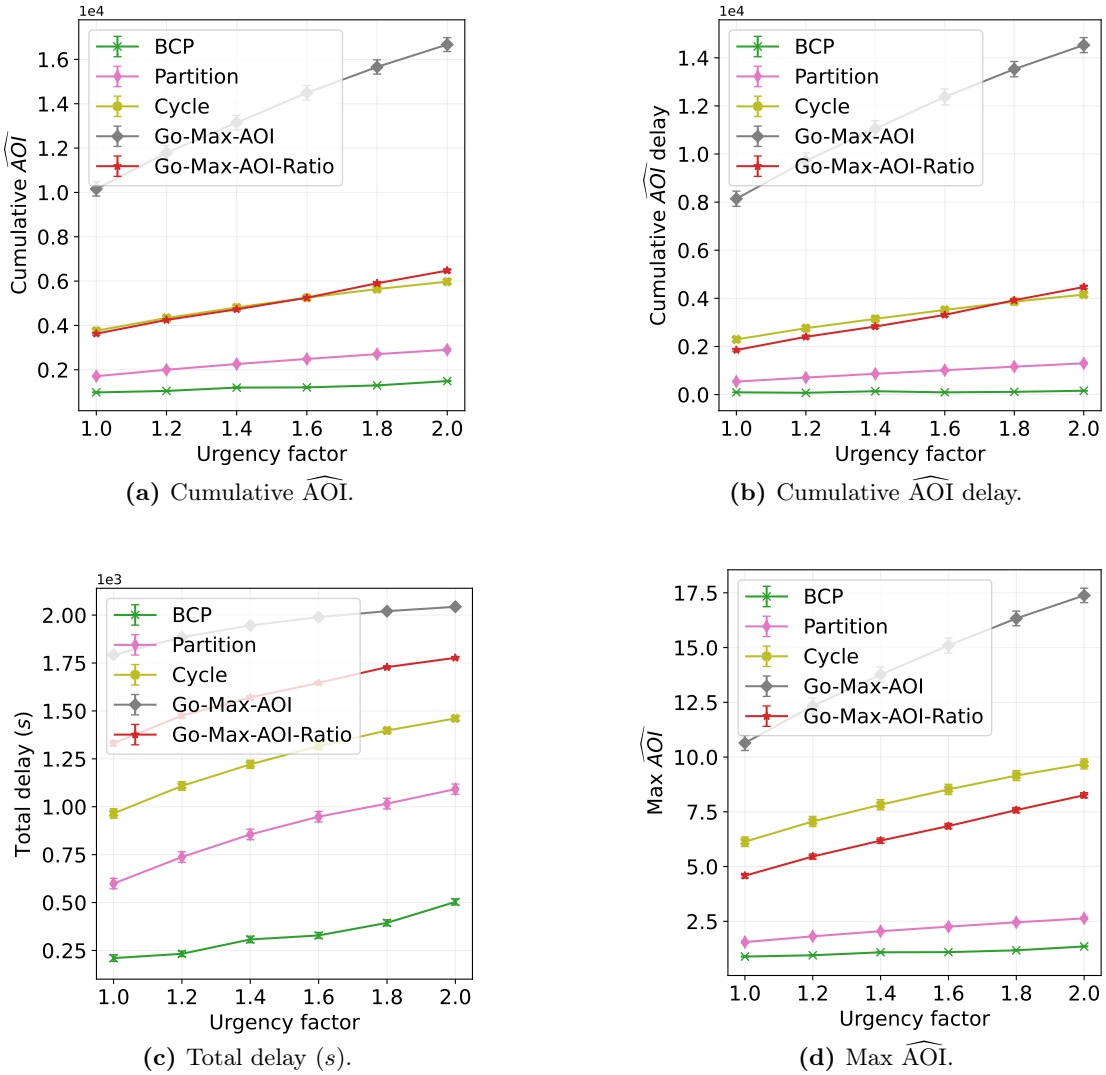
Figure A.3: 40 Clustered IPs. Clustered thresholds. Varying number of UAVs.

In these cases, BCP excels by strategically allocating resources to clusters with significant variations in both traveled distance and threshold values. The effectiveness of BCP in handling the challenges posed by diverse threshold distributions is a key factor contributing to its exceptional performance in scenarios like clustered deployments, which are frequently encountered in real-world patrolling missions.

Clustered IPs, Gaussian thresholds. Figure A.4 shows the results of our approach with 10 UAVs, over 40 IPs deployed in a clustered manner. Thresholds are assigned with a Gaussian distribution of varying urgency factors within the area. We recall that the urgency factor is proportional to the spread of the urgent area. In this scenari,o we can derive similar conclusions as the ones discussed in Section A.1.5.

It is worth noting that, examining Figure A.4c, BCP achieves a Total delay that is 63% lower than that of Partition, the top-performing state-of-the-art algorithm in our benchmark.

BCP strategically concentrates UAVs in the middle of the area for lower urgency factors and dynamically adapts to distribute UAVs more evenly as urgency increases across the entire area. The substantial reduction in total delay and consistently better performance in the other metrics underscore the superiority of BCP, especially in scenarios where the urgency of the IPs varies.



(a) Cumulative $\widehat{\text{AOI}}$. **(b)** Cumulative $\widehat{\text{AOI}}$ delay.

(c) Total delay (s). **(d)** Max $\widehat{\text{AOI}}$.

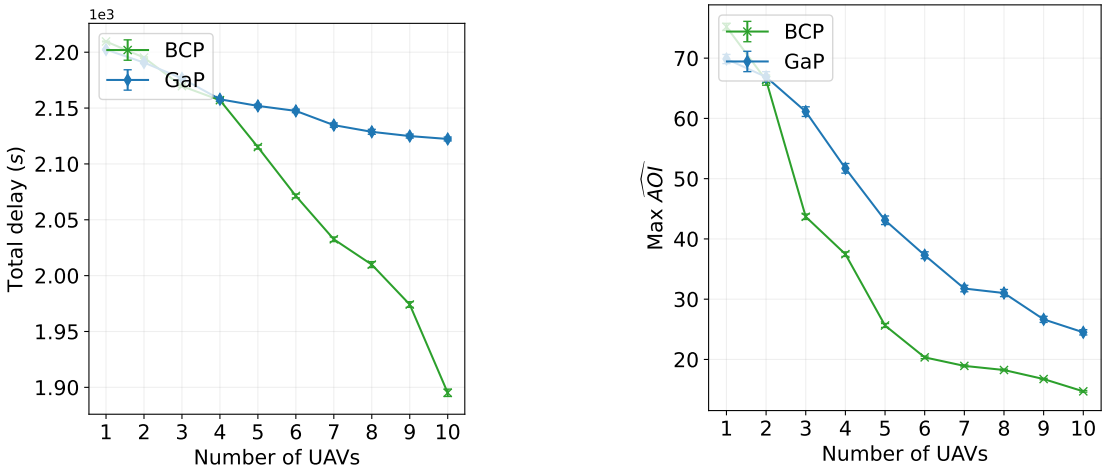
Figure A.4: 40 Clustered IPs. Gaussian thresholds. Varying urgency factor.

Uniform IPs, Gaussian thresholds. Figure A.4 shows the results of our approach over 40 IPs deployed in a clustered manner, thresholds assigned with a Gaussian distribution, and a number of employed UAVs varying in the range $[1, 10]$.

GaP is specifically designed to operate with battery constraints. To ensure a fair comparison, both GaP and BCP were provided with an equivalent level of energy. In the case of BCP, we enforced UAVs to return to the base since the residual energy would only support the UAVs to reach the base station.

Figure A.5 compares BCP and GaP in terms of total delay (Figure A.5a) and Max AOI (Figure A.5b). Our evaluation focuses on the number of UAVs employed in the mission, showcasing a significant enhancement in performance across all metrics. Notably, when considering the total delay, in Figure A.5a, our approach achieves an improvement of up to 10% compared to the state-of-the-art algorithm. Figure A.5b illustrates how BCP consistently maintains significantly fresher information across all IPs, with a worst-case age of information as low as half of that achieved by GaP.

The results highlight the effectiveness of thorough cluster formation and UAV assignment carried out by BCP as a favorable strategy for IPs exploration.



(a) Total delay (s). **(b)** Max $\widehat{\text{AOI}}$.

Figure A.5: 40 Uniform IPs. Gaussian thresholds. Varying number of UAVs.

A.1.6 Conclusions

In this chapter, we tackle the problem of patrolling a set of heterogeneous inspection points within an area using a fleet of UAVs. We formalize the problem by introducing novel performance metrics that capture points of heterogeneity, constraints (thresholds) on the age of information, and coverage completeness to ensure persistent monitoring in time and space. Having observed the NP-hardness of the problem, we propose Balanced Clusters Patrolling (BCP), a polynomial-time patrolling algorithm that initially determines a suitable threshold-aware clustering of the IPs and then identifies the optimal number of UAVs to assign to these clusters to minimize inter-visit delays. We evaluate the performance of BCP using the newly introduced metrics, comparing our solution with other heuristics. Through simulations across a wide range of experimental scenarios, we demonstrate the strength of our approach. By comparing BCP with other state-of-the-art approaches, we show that it outperforms previous methods in all performance metrics, achieving up to a 92% improvement in

total delay.

A.2 Sustainable Geo-Distributed AI Inference

Abstract

AI web services are becoming increasingly widespread, counting millions of users worldwide. These applications run large AI models on edge servers and cloud clusters to satisfy the users' inference demands. However, this growth raises concerns regarding the environmental impact of high-performance computing nodes and, particularly, their carbon emissions. In this chapter, we consider an online AI-service provider leasing computational resources spread across multiple heterogeneous geographically distributed data centers. The dispatcher receives user inference demands online and determines how to assign them to the computing nodes while maximizing the system's throughput and accuracy, minimizing carbon emissions, and minimizing the response time of each machine. We model this system by formalizing MTDO (Maximum Throughput Dynamic Optimization). To solve it, we propose STOP (STep OPtimizer), an algorithm based on the Lyapunov theory for stochastic network optimization that guarantees an optimality approximation. We conducted extensive experiments and compared the results of STOP with other state-of-the-art solutions. Our tests show that STOP consistently meets performance requirements and effectively balances the trade-off among carbon emissions, response time, and expected accuracy, guaranteeing system stability and maximizing the number of served demands.

A.2.1 Introduction

In recent years, services based on Machine Learning (ML) and Artificial Intelligence (AI) have grown rapidly. Prominent examples include Amazon Alexa, Google Assistant, Grammarly, YouTube, Netflix, Spotify recommendation systems, and speech-to-text transcription services like OpenAI's Whisper. These applications have significantly increased reliance on distributed data centers worldwide to serve millions of users, a trend that carries a growing environmental cost. According to the 2024 IEA report [272], the electricity consumption from data centers, AI, and cryptocurrency is expected to more than double by 2026 compared to 2022, reaching approximately 1000TWh. While training large AI models is energy-intensive, inference workloads, far more frequent than training, account for up to 90% of the total data center power consumption [273]. Each ChatGPT query alone is estimated to consume 2.9Wh [274]. Despite growing awareness, existing approaches fall short of addressing the joint optimization of latency, accuracy, emissions, and throughput in real-time inference scheduling. Relevant prior work includes decentralized data center management [275, 276], which typically relies on static or periodic strategies limited by coarse-grained updates and simplified latency assumptions. Other proposals explore dynamic routing [277], energy-aware training and collocation [278], and carbon profiling of GPUs [279, 280]. Although valuable, these methods do not jointly tackle performance and sustainability under real-time constraints. Closer to our work, Zhou et al. [281] apply Lyapunov optimization with virtual queues to control emissions, latency, and cost. However, their solution neglects queueing and processing delays and does not support per-query model selection or accuracy control. These limitations motivate the need for an adaptive, fine-grained inference scheduler that jointly manages sustainability and performance metrics

in a unified optimization framework. In this paper, we address this challenge and consider an online AI-service whose inference models are replicated across multiple computing nodes. The nodes leased by the AI service provider are assumed to be heterogeneous in computational capabilities and energy source supply, resulting in different carbon emissions. A dispatcher receives the users' inference requests dynamically and assigns them to nodes in real time, maximizing throughput while meeting long-term constraints on carbon emissions, latency, and accuracy. We formalize this objective as the *Maximum Throughput Dynamic Optimization* (MTDO) problem, which captures the trade-offs between performance and sustainability. Inspired by recent trends toward carbon-aware computing [282], MTDO explicitly models machine heterogeneity, carbon intensity, network delay, queueing dynamics, and computational overhead. To solve MTDO, we rely on an approach based on the *Min-Drift-Plus-Penalty* (MDPP) algorithm [283], which relies on Lyapunov theory for stochastic optimization. We propose a novel algorithm called STOP (STep OPtimizer), which efficiently solves an optimization problem, called STEP (STEP-by-step), at each iteration of MDPP. This technique enables real-time, per-request inference decisions that guarantee the satisfaction of long-term constraints and provide optimality approximation. STOP detects low-complexity instances and uses structural insights to greatly reduce the solution space for improved tractability. To our knowledge, this is the first application of Lyapunov-based MDPP in a real-time AI inference dispatching setting with accuracy and emission constraints. We validate STOP through extensive simulations considering two different real-world ML workloads, comparing it against two variants of a previous solution introduced in [284] and two baselines. Across all experimental scenarios, STOP ensures queue stability and outperforms all competitors.

The main contributions of our paper are:

- We introduce MTDO, a dynamic stochastic optimization model for environmentally sustainable and accurate AI inference dispatching.
- We propose STOP, a real-time algorithm that solves MTDO with optimal performance guarantees. STOP is structured into distinct modules, some of which are perfectly parallelizable, significantly enhancing computational efficiency.
- We run a comparative experimental campaign that shows how STOP outperforms state-of-the-art solutions in terms of throughput, accuracy, latency, and carbon impact under a wide range of experimental settings and workload profiles. We also extend STOP to implement an efficient deferral mechanism under the condition of system stability.

A.2.2 Related Work

As power consumed during inference represents the largest slice of the total energy consumed by data centers [273], conscious dispatching of inference tasks can strongly reduce emissions [285]. The first efforts to reduce energy consumption and carbon emissions in distributed architectures by load balancing date back more than a decade. Goiri et al. [286] introduce a green-energy-aware job scheduler. Liu et al. [275, 276] proposed centralized and decentralized algorithms for reducing energy consumption and costs and favoring green energy sources in a distributed data center architecture. The proposed solutions decide how to deploy jobs that are already accepted in the systems, with polling periods of one hour. The work in [277] introduces FORTE, a framework for traffic engineering that reduces carbon emissions, energy costs, and latency of incoming requests. While the authors state that their framework can be integrated into a dynamic scheduler to decide

job assignments, our solution already considers the dynamics of the system and tackles the related challenges. Similarly to our solution, Zhou et al. [281] use Lyapunov functions and virtual queues to optimize carbon emissions, electricity costs, and latency in a distributed architecture. Our approach differs from this solution in several ways. Firstly, we provide a more realistic model for latency, which not only depends on the network delay but also on the processing and queueing delay of the servers. Secondly, we consider a more dynamic setting, with inference requests arriving at high rates. Finally, our solution is tailored for AI inference and optimizes model accuracy. Recently, the work in [287] uses Lyapunov theory to design a distributed online offloading algorithm for AI inference. AI models are run on clients' machines and offloaded to a single powerful server only when necessary. This is different from our system, which collects clients' demands continuously, and optimizes their allocation in a distributed server architecture. Recently, several solutions for reducing carbon emissions have been developed for Federated Learning [288–291], a paradigm for distributing training over users' devices without sharing local data. In contrast, our method is designed for distributed inference over power-hungry accelerator-equipped hardware. Some authors have designed carbon and energy-aware job schedulers [292–294]. Some researchers have studied carbon emissions for the inference of AI tasks. Su et al. [278] propose CARE, a carbon-aware control framework that collocates training and inference tasks within a single node. The works in [279, 280] propose experimental studies showing the energy consumption and consequent carbon emissions of different GPUs and AI models. Several works focus on inference tasks for Large Language Models (LLMs). The work in [284] designs an efficient framework for LLM inference by introducing the concept of "generation directives", i.e., instructions associated with a prompt input that dictate the manner in which a generative language model produces tokens for the prompt. Other works [295, 296] design scheduling and redirect strategies for LLMs. The optimization problem that we formulate in this paper accounts for long-term constraint satisfaction and highly dynamic user queries while considering fundamental aspects such as response time and query dynamics. Li et al. in [284] propose a dynamic problem for deploying incoming demands for AI tasks on distributed compute nodes while minimizing carbon footprint, energy consumption, and water usage. To provide comparative experiments, we adapt their approach to our formulation and use it as a benchmark.

A.2.3 Problem Formulation

We consider a geographically distributed architecture composed of K heterogeneous machines, $M_1, \ldots, M_K$, and a dispatcher M_c. The architecture runs an ML service that users can access online to query inference tasks. The same set of J machine learning models with different sizes (i.e., number of parameters) and expected accuracy is deployed on each node. We consider a discrete-time stochastic process indexed by $t = t_0, t_1, \ldots$. We assume that $\Delta t = t_{i+1} - t_i = 1$ second. Users submit their demands through a client-side web application located on M_c. Every t, the dispatcher collects the requests and decides how to assign them to the compute nodes. A fixed amount of resources for each M_k is reserved for the ML service. These resources include a GPU for each M_k, characterized by the number of giga floating-point operations it can execute per second, GFLOPS_k, its energy consumption, e_k, and its carbon intensity, which depends on its geographic location, as it determines the energy source powering the machine. These parameters are known by the dispatcher. Every machine M_k can buffer incoming requests on a queue containing the jobs assigned to it. For each query, we consider the number of required operations to complete it and denote with $Q_k(t)$

the number of operations in M_k's queue at time t, $\forall k$. We assume that the dispatcher can obtain estimates of communication delay to and from all machines, without the need for complex telemetry tools.

The dispatcher assigns demands to the machines so that the number of satisfied demands is maximized under constraints of long-term queue stability, carbon emissions, response time, and accuracy. Hereafter, we formalize the carbon emission, the accuracy, and the response time models.

Carbon Emission Model

The carbon emission of a node is the number of released grams of CO_2, which is characterized by the number of operations that the machine has to execute, n_{op}, its frequency GFLOPS_k, its energy consumption e_k, its carbon intensity $CI_k(t)$, and its Power Usage Effectiveness (PUE_k). The carbon intensity measures the grams of CO_2 emitted per kWh. The PUE is the ratio between the total power supplied to the data center hosting the machine and the power used for computation. Notice that PUE_k is always greater than 1. A data center is more effective as its PUE is closer to 1, meaning it can convert almost all the power it is supplied with for computation. We employ the formulations introduced in [297] and [298] to model the energy consumption and the carbon emissions of a GPU running an ML model on a node M_k, based on its GFLOPS_k, PUE_k, and CI_k. We call $C_k(t, n_{op})$ the grams of CO_2 emitted at time t to execute n_{op} operations:

$$C_k(t, n_{op}) = CI_k(t) \cdot PUE_k \cdot E_k(n_{op}) = CO2_k(t) \cdot n_{op}, \tag{A.6}$$

where $E_k(n_{op})$ is the energy consumption of the GPU to perform n_{op} operations, and $CO2_k(t)$ is the number of grams of CO_2 emitted per operation at time t.

Accuracy Model

We assume that each compute node has J pre-trained ML models reduced with some model reduction technique [299]. The models differ in the number of operations they must perform, either because they have a different number of parameters (model compacting, tensor decomposition, network sparsification), or they must perform simpler operations (quantization), or a combination of these. Larger models usually guarantee better classification performance at the expense of higher computational cost and, therefore, more carbon emissions and higher response time. In our formulation, we assume that the dispatcher knows the average accuracy of each model j (denoted as π_j), obtained at test time, and uses this information to compute the probability of responding to the users' queries correctly. We assume $\pi_j < \pi_{j'}$, $\forall j < j'$. We also assume that the dispatcher knows the number of operations each model needs to perform an inference task on a demand of size σ_i and the output size. This information depends on the ML task and is known for many models, including convolutional neural networks, multi-layer perceptrons, and transformer-based architectures. Given an input i of size σ_i and a model j, we denote with $\omega_{i,j}$ the number of operations required to execute model j on the input. We assume $\omega_{i,j} < \omega_{i,j'}$, $\forall j < j'$.

Response Time Model

In our scenario, the dispatcher can assign multiple queries to each M_k at a given time, and we assume that they are served by M_k following the FIFO paradigm. We refer to the *response time of a machine M_k to a batch of n queries* as the response time of the last query in the batch. We also assume that each query $i \in \{1, \ldots, n\}$ has size σ_i (expressed in Bytes), and denote with $\boldsymbol{\sigma}$ the vector $[\sigma_1, \ldots, \sigma_n]$. We call $R_k(t, \boldsymbol{\sigma})$ the response time of M_k to the batch of size $\boldsymbol{\sigma}$ at time t, representing the end-to-end response time to users' queries, which is given by the following three components:

$$R_k(t, \boldsymbol{\sigma}) = d_k^{comm}(t, \boldsymbol{\sigma}) + d_k^{proc}(\boldsymbol{\sigma}) + d_k^{Q}(t, \boldsymbol{\sigma}), \tag{A.7}$$

where $d_k^{comm}(t, \boldsymbol{\sigma})$ is the *communication delay*, $d_k^{proc}(\boldsymbol{\sigma})$ is the *processing delay* and $d_k^{Q}(t)$ is the *queuing delay*. As previously mentioned, we assume that the dispatcher has access to measurements to and from all the machines to estimate communication delays $d_k^{comm}(t, \boldsymbol{\sigma})$, i.e., the communication component of the time required to send data of size $\boldsymbol{\sigma}$ to machine M_k and receive a response, whose size depends on the AI service. For instance, if the model performs an image classification task, the size of the input is the size of the image, whereas the size of the output is the classification (i.e., a string with a few characters). This assumption is both lightweight and practically feasible, by means of a mix of active and passive measurements, and does not require complex telemetry tools[2] The processing delay is the time required by M_k to process all demands in the batch and is computed as follows:

$$d_k^{proc}(\boldsymbol{\sigma}) = \frac{\omega_{1,j_1} + \ldots + \omega_{n,j_n}}{\text{GFLOPS}_k}, \tag{A.8}$$

where ω_{i,j_i} is the number of operations required to process the demand of size σ_i with model j_i (in general, $\omega_{i,j} \leq \omega_{i',j}$, $\forall \sigma_i < \sigma_{i'}$). Finally, $d_k^{Q}(t)$ is the time required by M_k to process the operations buffered in its queue, i.e. $d_k^{Q}(t) = \frac{Q_k(t)}{\text{GFLOPS}_k}$. All delays are expressed in seconds.

MTDO formulation

At every time step t, the dispatcher receives $a(t)$ queries from the users and decides how to assign them to machines $M_1, \ldots, M_K$ to maximize the throughput, i.e., the number of satisfied demands per time step. Each query i has size σ_i. We call $\boldsymbol{\sigma}(t)$ the demands' size vector, i.e., $\boldsymbol{\sigma}(t) = [\sigma_1(t), \ldots, \sigma_{a(t)}(t)]$, and $\mathbf{X}(t) \in \{0,1\}^{K \times a(t) \times J}$ the tensor of the decision variables $x_{kij}(t)$, defined as follows:

$$x_{kij}(t) = \begin{cases} 1, & \text{demand } i \text{ is assigned to machine } M_k \text{ and model } j \\ 0, & \text{otherwise} \end{cases} \tag{A.9}$$

With $\mathbf{X}_k(t)$ we represent the $k-$th matrix of $\mathbf{X}(t)$, that is, $\mathbf{X}_k(t) \triangleq \big(x_{kij}(t)\big)_{1 \leq i \leq a(t), 1 \leq j \leq J}$, indicating the demand-to-model allocation at machine M_k. Recall that $Q_k(t)$ is the number of operations in M_k's queue at time t and, assuming that $Q_k(0) = 0$, its updating rule is defined as follows:

$$Q_k(t+1) = Q_k(t) - \min\{Q_k(t), \text{GFLOPS}_k\} + \langle \mathbf{X}_k(t), \boldsymbol{\Omega}(t) \rangle_F \tag{A.10}$$

[2] A large body of literature (see e.g., [300, 301]) provides ways to obtain application- or TCP-level measurements. The specific details are beyond the scope of this work. Minor inaccuracies in delay estimation do not compromise the effectiveness of the proposed solution since communication delays typically have much less impact than queueing and processing delays, except in the rare case of severe network congestion.

where $\mathbf{\Omega}(t) \in \mathbb{N}^{a(t)\times J}$ is the matrix whose generic element $\omega_{i,j}(t)$ represents the number of operations required to execute the demand of size σ_i with the model j. The symbol $\langle \cdot, \cdot \rangle_F$ represents the Frobenius inner product. The dispatcher assigns the demands to the machines and models by observing the machines' queues, carbon emissions, response time, and accuracy. We constrain all these metrics to be below a given value on average (above for accuracy). Specifically, we assume that machines do not have to emit more than C^{avg} grams of CO_2 per second on average. The response time for each machine does not have to be higher than R^{avg} seconds on average. The average accuracy of the system over time has to be at least P^{avg}. Furthermore, the machine queues need to be stable. Under these constraints, we formulate MTDO as follows:

$$\min \ \limsup_{t\to\infty} \bar{y}(t) \tag{A.11a}$$

$$\text{s.t.} \ \limsup_{t\to\infty} \bar{c}_k(t) \leq 0 \quad \forall k \in \{1,\dots,K\} \tag{A.11b}$$

$$\limsup_{t\to\infty} \bar{r}_k(t) \leq 0 \quad \forall k \in \{1,\dots,K\} \tag{A.11c}$$

$$\limsup_{t\to\infty} \bar{p}(t) \leq 0 \tag{A.11d}$$

$$\lim_{t\to\infty} \frac{\mathbb{E}_t[Q_k(t)]}{t} = 0 \quad \forall k \in \{1,\dots,K\} \tag{A.11e}$$

$$\sum_{k=1}^{K}\sum_{j=1}^{J} x_{kij}(t) \leq 1 \quad \forall i \in \{1,\dots,a(t)\}\, \forall t \tag{A.11f}$$

$$x_{kij}(t) \in \{0,1\}\, \forall k \in \{1,\dots,K\}\, i \in \{1,\dots,a(t)\}\, j \in \{1,\dots,J\}\, \forall t \tag{A.11g}$$

where $\bar{y} = \frac{1}{t}\sum_{\tau=0}^{t} \mathbb{E}[y(\mathbf{X}(\tau))]$ (Equation A.11a) is equal to minus the time average of the accepted demands, i.e., $y(\mathbf{X}(t)) = -\sum_{k\in\{1,\dots,K\},i\in\{1,\dots,a(t)\},j\in\{1,\dots,J\}} x_{kij}(t)$. Where needed, in the remainder of the paper, we omit the definition space of the variables k, i and j for readability, which are always respectively $\{1,\dots,K\}$, $\{1,\dots,a(t)\}$ and $\{1,\dots,J\}$. The terms appearing in Equations A.11b-A.11d are defined accordingly, i.e., $\bar{c}_k = \frac{1}{t}\sum_{\tau=0}^{t} \mathbb{E}[c_k(\mathbf{X}(\tau))]$, $\bar{r}_k = \frac{1}{t}\sum_{\tau=0}^{t} \mathbb{E}[r_k(\mathbf{X}(\tau))]$ $\forall k = 1,\dots,K$ and $\bar{p} = \frac{1}{t}\sum_{\tau=0}^{t} \mathbb{E}[p(\mathbf{X}(\tau))]$. In particular, for each machine M_k, $c_k(\mathbf{X}(t))$ is the difference between the carbon emitted by M_k under the decision $\mathbf{X}(t)$ and C^{avg}:

$$c_k(\mathbf{X}(t)) = C_k(t, \langle \mathbf{X}_k(t), \mathbf{\Omega}(t) \rangle_F) - C^{avg}, \tag{A.12}$$

whereas $r_k(\mathbf{X}(t))$ is the difference between the response time of M_k under the decision $\mathbf{X}(t)$ and R^{avg}:

$$r_k(\mathbf{X}(t)) = \sum_{i,j} x_{kij} d_k^{comm}(t,\sigma_i) + \frac{\langle \mathbf{X}_k(t), \mathbf{\Omega}(t) \rangle_F}{\text{GFLOPS}_k} + \max_{i,j}\{x_{kij}\} d_k^Q - R^{avg}. \tag{A.13}$$

Notice that, for each k, the term $\max_{i,j}\{x_{kij}\} d_k^Q$ counts the queueing time of M_k only once if some demand is assigned to M_k. Finally, $p(\mathbf{X}(t))$ is the total deviation from the average accuracy the system should obtain over time, i.e.:

$$p(\mathbf{X}(t)) = \sum_{k=1}^{K}\sum_{i=1}^{a(t)}\sum_{j=1}^{J} x_{kij} \cdot (P^{avg} - \pi_j) \tag{A.14}$$

Equations A.11b-A.11d enforce the objective of obtaining the averages over time of the carbon emissions, the response time, and the accuracy as close as possible to the corresponding targets. Equation A.11e enforces that all the queues are *mean rate stable* as defined in [283]. Finally, Equation A.11f constrains each demand to be satisfied by at most one machine and one model.

We notice that the search space for finding the optimal policy of a decision process to solve MTDO has exponentially many possible outcomes. For this reason, we opt for an algorithmic solution based on Lyapunov's theory. The main strength of Lyapunov optimization is its ability to quickly converge to an adaptive solution for job-to-machine and model assignment. Such a solution provides stability and near-optimal performance under variable demands, without assuming precise statistical knowledge of the input distribution.

A.2.4 Algorithmic Solution

To solve MTDO, we use the *Min-Drift-Plus-Penalty* (MDPP) algorithm, an approach introduced in [283] and described in Section A.2.4. This algorithm makes use of mathematical artifacts to formalize a new optimization problem to be solved at each time step. We note that our formulation results in an NP-hard Integer Linear Programming Problem (ILP). We approach it with STOP, detailed in Section A.2.4.

Min-Drift-Plus-Penalty

The MDPP algorithm is designed for stochastic optimization problems and uses Lyapunov functions to formalize the concept of *virtual queues* to enforce long-term constraints. The virtual queues for the constraints A.11b-A.11d are defined as follows:

$$\begin{aligned} &Z_k(t+1) = \max\{Z_k(t) + y_k(\mathbf{X}(t)), 0\},\ \forall k \in \{1,\dots,K\} \\ &W_k(t+1) = \max\{W_k(t) + r_k(\mathbf{X}(t)), 0\},\ \forall k \in \{1,\dots,K\} \\ &Y(t+1) = \max\{Y(t) + p(\mathbf{X}(t)), 0\} \end{aligned} \tag{A.15}$$

where $Z_k(0) = W_k(0) = 0$ ($\forall k = 1,\dots,K$) and $Y(0) = 0$. They measure the historical carbon emissions, response times, and achieved accuracy of the system, respectively. The MDPP algorithm guarantees *mean rate stability* of the queues, thus achieving the satisfaction of the respective constraints in MTDO. The system queues Q_k update by the rule in Equation A.10. Following the approach in [283], we can formalize the following optimization problem for each time step t, whose objective function includes the virtual queues:

$$\begin{aligned} \min_{\mathbf{X}(t)} \tilde{F}(\mathbf{X}(t)) = &-V \sum_{k,i,j} x_{kij} + Y(t)p(\mathbf{X}(t)) \\ &+ \sum_k \Big[Q_k(t)(\langle \mathbf{X}_k(t), \mathbf{\Omega}(t) \rangle_F - \text{GFLOPS}_k) \qquad \text{(A.16a)} \\ &+ Z_k(t)c_k(\mathbf{X}_k(t)) + W_k(t)r_k(\mathbf{X}_k(t)) \Big] \\ s.t &\sum_{k,i,j} x_{kij} \leq 1 \quad \forall i \in \{1,...,a(t)\} \qquad \text{(A.16b)} \\ &x_{kij} \in \{0,1\}\ \forall k \in \{1,...,K\}, i \in \{1,...,a(t)\}, j \in \{1,...,J\} \qquad \text{(A.16c)} \end{aligned}$$

where $\tilde{F}(\mathbf{X}(t))$ in Equation A.16a is the objective function, which depends on the parameter V, called *penalty*, weighing the system's throughput. Equations A.16b and A.16c are the same constraints as in Equations A.11f and A.11g. Note that the problem of Equation A.16 can be rewritten equivalently by only considering the non-constant elements of $\tilde{F}(\mathbf{X}(t))$ and by linearizing the max function in the objective. This can be done by introducing new decision variables γ_k. We get the following problem:

$$\begin{aligned}
\min_{\mathbf{X}(t)} \quad & F(\mathbf{X}(t)) = -V \sum_{k,i,j} x_{kij} + Y(t) \sum_{k,i,j} x_{kij}(P^{avg} - \pi_j) \\
& + \sum_k \Big(Q_k(t) + Z_k(t) \cdot CO2_k + \frac{W_k(t)}{\text{GFLOPS}_k} \Big) \langle \mathbf{X}_k(t), \mathbf{\Omega}(t) \rangle_F && \text{(A.17a)} \\
& + \sum_{k,i,j} W_k(t) d_k^{comm}(t, \sigma_i) x_{kij} + \sum_k W_k(t) d_k^Q(t) \gamma_k \\
s.t & \sum_{k,j} x_{kij} \leq 1 \quad \forall i && \text{(A.17b)} \\
& \gamma_k \geq x_{kij} \quad \forall k, i, j && \text{(A.17c)} \\
& x_{kij}, \gamma_k \in \{0, 1\} \, \forall k, i, j && \text{(A.17d)}
\end{aligned}$$

The new constraints in Equation A.17b enforce all auxiliary variables $\gamma_k \in \{0, 1\}$ to be equal to the maximum x_{kij}. The MDPP algorithm consists of observing the number of demands arriving at each time t, $a(t)$, and their sizes, and minimizing the objective function in Equation A.17a under the constraints A.17b-A.17d. Note that different queues and coefficients appearing in the objective function have different meanings and values. For instance, as we will see in Section A.2.5, the grams of emitted carbon at each time step for each machine is of the order of 10^{-2} or 10^{-3}, whereas the response time is a bunch of seconds. Hence, the queues evolve at very different scales. To cope with this, we change MTDO by re-defining the functions in the constraints as follows[3]:

$$c_k(\mathbf{X}(t)) \leftarrow \frac{100}{C^{avg}} \cdot c_k(\mathbf{X}(t)). \tag{A.18}$$

$$r_k(\mathbf{X}(t)) \leftarrow \frac{100}{R^{avg}} \cdot r_k(\mathbf{X}(t)). \tag{A.19}$$

$$p(\mathbf{X}(t)) \leftarrow \frac{100}{a(t) \cdot P^{avg}} \cdot p(\mathbf{X}(t)). \tag{A.20}$$

In this way, we are not enforcing the absolute difference between each $c_k(\mathbf{X}(t))$, $r_k(\mathbf{X}(t))$ and $p(\mathbf{X}(t))$ and C^{avg}, R^{avg} and P^{avg}, respectively, to be less than 0, but rather their relative deviation. Finally, we can formulate StEP (Step-by-step Problem) as follows:

$$\begin{aligned}
\min_{\mathbf{X}(t)} \quad & -V \sum_{k,i,j} x_{kij} + \frac{100 \cdot Y(t)}{a(t) P^{avg}} \sum_{k,i,j} x_{kij}(P^{avg} - \pi_j) + \\
& 100 \cdot \sum_k \Big(Q_k(t) + Z_k(t) \frac{CO2_k}{C^{avg}} + \frac{W_k(t)}{\text{GFLOPS}_k R^{avg}} \Big) \langle \mathbf{X}_k(t), \mathbf{\Omega}(t) \rangle_F + && \text{(A.21a)} \\
& \sum_k \Big[\frac{100 \cdot W_k(t)}{R^{avg}} \Big(d_k^Q(t) \gamma_k + \sum_{i,j} d_k^{comm}(t, \sigma_i) x_{kij} \Big) \Big]
\end{aligned}$$

[3]Optimal approximation guarantees of MDPP for $t \to \infty$ still hold and do not depend on scaling factors in the objective.

$$s.t \sum_{k,i,j} x_{kij} \leq 1 \quad \forall i \tag{A.21b}$$

$$\gamma_k \geq x_{kij} \quad \forall k,i,j \tag{A.21c}$$

$$x_{kij}, \gamma_k \in \{0,1\} \, \forall k,i,j \tag{A.21d}$$

In the remainder of this section, we will omit the dependence on time (t) unless necessary.

Theorem A.2.1. *The problem in Equation A.21 is NP-Hard.*

Proof (sketch). We show a reduction of the Uncapacitated Facility Location Problem (UFLP [302]) to StEP. UFLP considers N facilities to serve M demands. Each facility has an opening cost $f(i)$, and the cost of serving a demand j by a facility i is $c(i,j) \geq 0$, $\forall i \in \{1,\dots,N\}$, $\forall j \in \{1,\dots,M\}$. The goal is to minimize the facilities' cost while serving all the demands. We can build a special instance of StEP by considering a single model ($J = 1$) and choosing a large value of V. We can then map UFLP to StEP by setting $K = N$, $a(t) = M$, the terms of $F(\mathbf{X}(t))$ related to the carbon emissions, system's queues, and processing delay equal to $c(i,j)$, and the queuing delay equal to $f(i)$. We can then prove that the optimal solution to any instance of UFLP is optimal also for the special instance of StEP, and vice-versa. Being UFLP *Max SNP-Hard* [303], so is StEP. ∎

STOP

We propose an algorithmic approach to solve StEP that exploits auxiliary LP problems, defined hereafter.

StEP$^{\mathbf{R}}$ We call StEP$^{\mathrm{R}}$ the problem obtained by relaxing the domain of x_{kij} and γ_k in Equation A.21d to the interval $[0,1]$. The resulting problem is an LP, which can be efficiently solved by an LP solver in polynomial time [304].

StEP$^{\mathbf{NQ}}$ We introduce StEP$^{\mathrm{NQ}}$ as the problem of finding the $\mathbf{X}$ that minimizes the following function:

$$F^{NQ}(\mathbf{X}) = -V\sum_{k,i,j} x_{kij} + \frac{100Y}{a(t)P^{avg}} \sum_{k,i,j} x_{kij}(P^{avg} - \pi_j) + \sum_{k,i,j} \frac{100W_k}{R^{avg}} d_k^{comm}(\sigma_i) x_{kij} + 100 \sum_k \Big(\frac{Q_k}{100} + Z_k \frac{CO2_k}{C^{avg}} + \frac{W_k}{\mathrm{GFLOPS}_k R^{avg}} \Big) \langle \mathbf{X}_k, \mathbf{\Omega} \rangle_F$$

under constraints A.16b and with $x_{kij} \in [0,1]$.

StEP$^{\mathbf{\Sigma Q}}$ Finally, we define StEP$^{\mathbf{\Sigma Q}}$ as the problem of minimizing the following function:

$$\begin{aligned} F^{\Sigma Q}(\mathbf{X}) = & -V\sum_{k,i,j} x_{kij} + \frac{100Y}{a(t)P^{avg}} \sum_{k,i,j} x_{kij}(P^{avg} - \pi_j) + \\ & \sum_{k,i,j} \frac{100W_k}{R^{avg}} \Big[d_k^{comm}(\sigma_i) + d_k^Q \Big] x_{kij} + \\ & 100 \sum_k \Big(\frac{Q_k}{100} + Z_k \frac{CO2_k}{C^{avg}} + \frac{W_k}{\mathrm{GFLOPS}_k R^{avg}} \Big) \langle \mathbf{X}_k, \mathbf{\Omega} \rangle_F \end{aligned} \tag{A.22}$$

under constraints A.16b and with $x_{kij} \in [0, 1]$.

Notice that StEP$^{\text{NQ}}$ does not consider the term d_k^Q whereas StEP$^{\mathbf{\Sigma Q}}$ counts it once for each demand assigned to each M_k. These problems reduce the complexity of MTDO in two ways: *(i)* They relax the domain of the decision variables x_{kij}, making the problems solvable in polynomial time with an algorithm for LP optimization. In Observation A.2.1, we prove that optimal solutions to StEP$^{\text{NQ}}$ are always integer-valued. The same argumentation holds for StEP$^{\mathbf{\Sigma Q}}$. *(ii)* They remove the dependence on the γ_k variables in Equation A.21. In this way, these problems decouple decisions on single demands, making the decisions of serving demands or dropping them independent of one another. From this fact follows the result of Observation A.2.2, which is used in Observations A.2.3 and A.2.4 to limit the solution search space of StEP. In the remainder of this section, we denote with $\mathbf{X}^*$, $\mathbf{X}^{*R}$, $\mathbf{X}^{*NQ}$ and $\mathbf{X}^{*\Sigma Q}$ the optimal solutions to StEP, StEP$^{\text{R}}$, StEP$^{\text{NQ}}$ and StEP$^{\mathbf{\Sigma Q}}$, respectively. Furthermore, we call $\mathbf{X}_i \in \{0, 1\}^{K \times J}$ and $F_i(\mathbf{X}_i)$ the decision variables related to a demand i and the terms of the function F that are multiplied by some $x \in \mathbf{X}_i$, $\forall i \in \{1, \dots, a(t)\}$.

Observation A.2.1. *StEP^{NQ} has $\{0, 1\}$-valued solutions, i.e., $x_{kij}^{*NQ} \in \{0, 1\}$ $\forall k, i, j$.*

Proof. The constraints of StEP$^{\text{NQ}}$ only enforce each demand to be assigned to at most one (k, j), $\forall k, j$, therefore we can decide the assignment of each demand independently from the decisions made on the others. Hence, we can obtain $\mathbf{X}^{*NQ}$ by finding $\mathbf{X}_i^{*NQ} \in \{0, 1\}^{K \times J}$ for each demand i as the optimal solution to the problem $\min_{\mathbf{X}_i} \sum_{k,j} F_i^{NQ}(\mathbf{X}_i)$, subject to $\sum_{k,j} x_{kij} \leq 1$, $x_{kij} \in [0, 1]$. By linearity, the objective is minimized at the bounds of x_{kij}, i.e., for either $x_{kij} = 0$ $\forall k, j$, or for $x_{kij} = 1$ for some (k, j). ∎

The analogous result holds for StEP$^{\mathbf{\Sigma Q}}$.

Observation A.2.2. *It is trivial to see that $\mathbf{X}^{*NQ}$ assigns a demand i to some (k, j) if and only if $F_i^{NQ}(\mathbf{X}_i^{*NQ}) \leq 0$. Analogously, $\mathbf{X}^{*\Sigma Q}$ assigns a demand i to some (k, j) if and only if $F_i^{\Sigma Q}(\mathbf{X}_i^{*\Sigma Q}) \leq 0$. If $\mathbf{X}_i^{*NQ} = 0$, then $F_i^{NQ}(\mathbf{X}_i^{*NQ}) = 0$, and analogously for $\mathbf{X}_i^{*\Sigma Q}$.*

In the next Observations, we notice that the optimal solutions to StEP$^{\text{NQ}}$ and StEP$^{\mathbf{\Sigma Q}}$ are helpful in reducing the search space of $\mathbf{X}^*$.

Observation A.2.3. *$\forall i$, if $\exists k, j$ s.t. $x_{kij}^{*\Sigma Q} = 1$, then $\exists \bar{k}, \bar{j}$ s.t. $x_{\bar{k}i\bar{j}}^* = 1$.*

Proof. The thesis is a consequence of Observation A.2.2. If $\mathbf{X}^{*\Sigma Q}$ assigns a demand i despite the resulting $F^{NQ}(\mathbf{X}^{*\Sigma Q})$ counts the positive term $W_k \cdot d_k^Q$ potentially more than once, even more so shall $\mathbf{X}^*$ assign i since it counts it only once per machine, if some demand is assigned to it. ∎

Observation A.2.4. *$\forall i$ s.t. $x_{kij}^{*NQ} = 0$ $\forall k, j$, $x_{kij}^* = 0$ $\forall k, j$.*

Proof. By Observation A.2.2, the optimal solution $\mathbf{X}^{*NQ}$ does not assign a demand i if and only if $\theta_{kj} = -V + \frac{100Y}{a(t)P^{avg}}(P^{avg} - \pi_j) + 100(\frac{Q_k}{100} + Z_k \frac{CO2_k}{C^{avg}} + \frac{W_k}{\text{GFLOPS}_k R^{avg}}) \cdot \omega_{i,j} + 100 W_k \cdot \frac{d_k^{comm}(\sigma_i)}{R^{avg}} \geq 0$ $\forall k, j$. It follows that the same demand is not assigned by $\mathbf{X}^*$ either, since assigning it would mean incrementing the objective function $F(\mathbf{X}^*)$ either by the positive term θ_{kj} if M_k is already used for other demands, or by some strictly larger terms otherwise. ∎

Lemma A.2.2. *If $\mathbf{X}^{*NQ} = \mathbf{X}^{*\Sigma Q}$, then $\mathbf{X}^* = \mathbf{X}^{*NQ} = \mathbf{X}^{*\Sigma Q}$.*

Proof [sketch]. The thesis can be easily derived from Observations A.2.3 and A.2.4. ∎

Finally, we notice that some instances of StEP are easy and can be solved in $O(K)$:

Observation A.2.5. *Assume $\exists k$ such that $Q_k = W_k = Z_k = 0$. Then the optimal solution to StEP assigns all demands to M_k and the most accurate model, J.*

Proof. By hypothesis, if all queues of some M_k are zero, then all positive terms of F are zero. The objective is minimized by maximizing the throughput and the model accuracy, i.e., by assigning all demands to machines with zero-valued queues and on the most accurate model. □

STOP is detailed in Algorithm 1 and runs as follows: upon receiving new demands and their sizes, the dispatcher checks whether there is a machine M_k with zero-valued queues. In this case, it gets the optimal solution by assigning all $a(t)$ demands to such a machine on the most accurate model J (as a tiebreaker, we assign all the demands to the fastest machine among those with 0-queues) according to Observation A.2.5. Otherwise, the dispatcher computes the number of operations required to run each model on the received demands, $\mathbf{\Omega}$, and solves StEP^{R} with an LP solver. If the solution is an integer, then it is an optimal solution to StEP. Otherwise, we solve StEP^{NQ} and $\text{StEP}^{\mathbf{\Sigma}\text{Q}}$. If their optimal solutions are equal, then by Lemma A.2.2, we have the optimal solution to StEP. Otherwise, we identify all the demands that are not assigned in the optimal solution of StEP^{NQ}, as well as those that are assigned in the optimal solution of $\text{StEP}^{\mathbf{\Sigma}\text{Q}}$, and we accordingly restrict the search space of the StEP solution (Observations A.2.3-A.2.4). Finally, we solve StEP with a solver for Integer Linear Problems on the reduced space.

The complexity of STOP is strongly dependent on the input instances and the state of the architecture. Nevertheless, the algorithm runs in exponential time in the worst case. In Algorithm 1, we highlight five different blocks, $B_1, \ldots, B_5$. Blocks $B_1, \ldots, B_4$ can be executed in perfect parallelism by four processes $P_1, \ldots, P_4$ running on the dispatcher. The first process that finishes successfully (i.e., finds the optimal solution) sends an alert, interrupting the others. Notice that process P_4 in B_4 needs to read the solution $\mathbf{X}^{*NQ}(t)$ computed by P_3 in B_3. The same process runs B_5 if the condition in line 11 is not satisfied and if it does not receive any interrupting alert from other processes. In the algorithm, we also provide the computational complexity for each block. $C(\text{LP})$ and $C(\text{ILP})$ denote the complexity for an LP and an ILP solver, respectively. In Section A.2.5, we experimentally assess the effectiveness of STOP in detecting and efficiently solving instances that admit a polynomial-time solution, based on its execution time.

Optimality In the following Theorem A.2.3, we show the approximation to the optimal solution of MTDO of the MDPP algorithm solved with STOP.

Theorem A.2.3. *Let y^{opt} be the value of the optimal solution to MTDO (Equation A.11) and let $\{y\}_{t=0,1,\ldots}$ be the values of the solutions achieved by STOP. If all $\{a(t)\}$ are i.i.d., then:*

1. $\limsup_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}[y(\tau)] \leq y^{opt} + \frac{B}{V}$.
2. *All queues Q_k, Z_k, W_k and P are mean rate stable.*

V is the penalty, B is a positive finite value.

Proof. The thesis follows from Theorem IV.8 in [283], from the fact that STOP finds the optimal

Algorithm 1: STOP

Input: $a(t), \boldsymbol{\sigma}$: set of demands and their size
Output: Optimal solution to StEP, $\mathbf{X}^*(t)$

Procedure `STOP` $(a(t), \boldsymbol{\sigma})$
  if $\exists k$ *such that* $Q_k = Z_k = W_k = Y = 0$ **then** $B_1, P_1, O(K)$
    Set $x^*_{kij} = 1 \ \forall i = 1, \ldots, a(t)$;
    return $\mathbf{X}^*$;
  Compute $\boldsymbol{\Omega}$;
  Solve StEP$^{\mathrm{R}}$, obtain $\mathbf{X}^{*R}$; $B_2, P_2, C(\mathrm{LP})$
  if $x^{*R}_{kij} = 0 \vee x^{*R}_{kij} = 1 \ \forall k, i, j$ **then**
    return $\mathbf{X}^{*R}$;
  Solve StEP$^{\mathrm{NQ}}$, obtain $\mathbf{X}^{*NQ}$; $B_3, P_3, O(K \cdot J \cdot a(t))$
  Solve StEP$^{\Sigma\mathrm{Q}}$, obtain $\mathbf{X}^{*\Sigma Q}$; $B_4, P_4, O(K \cdot J \cdot a(t))$
  if $\mathbf{X}^{*NQ} = \mathbf{X}^{*\Sigma Q}$ **then** $B_5, P_4, C(\mathrm{ILP})$
    return $\mathbf{X}^{*NQ}$;
  $I^0 \leftarrow \emptyset$, $I^1 \leftarrow \emptyset$;
  for $i = 1, \ldots, a(t)$ *such that* $x^{*NQ}_{kij} = 0 \, \forall k, j$ **do**
    $I^0 \leftarrow I^0 \cup \{i\}$;
  for $i = 1, \ldots, a(t)$ *such that* $\exists k, j$ *such that* $x^{*\Sigma Q}_{kij}(t) = 1$ **do**
    $I^1 \leftarrow I^1 \cup \{i\}$;
  Solve StEP to obtain $\mathbf{X}^*$ in the restricted search space where $x^*_{kij} = 0 \ \forall i \in I^0, \forall j, k$ and where $\exists k, j$ such that $x^*_{kij} = 1 \ \forall i \in I^1$;
  return $\mathbf{X}^*$;

solution to StEP, and from the finiteness of the following expression:

$$\frac{1}{2}\sum_k \mathbb{E}\Big[\Big(\sum_{i,j} x_{kij}(t)\Big)^2 + \mathrm{GFLOPS}_k^2\Big] + \frac{1}{2}\mathbb{E}\big[p(\mathbf{X}(t))^2\big] + \frac{1}{2}\sum_k \mathbb{E}\big[c_k(\mathbf{X}_k(t))^2\big] + \frac{1}{2}\sum_k \mathbb{E}\big[r_k(\mathbf{X}_k(t))^2\big]. \tag{A.23}$$

□

For space limits, we omit the proof of finiteness in Equation A.23, of which B is an upper bound. Nevertheless, we highlight that to achieve this result, we must assume that the variance and the expected value of the arrivals $a(t)$ are finite, which is true for distributions commonly used to describe arrival rates, as discrete uniform between two finite values and Poisson distributions.

A.2.5 Experiments

We show the performance of our approach by running comparative experiments with other state-of-the-art solutions in various experimental settings.

Experimental Setup

We run experiments simulating two ML services.

Image Recognition. We implement three variants of `YOLO11`[4], a state-of-the-art model for image recognition and classification, specifically, the *Nano*, *Medium*, and *Large* versions. Each is characterized by a distinct number of parameters and corresponding accuracy levels (namely 85, 90 and 95%). The model can process input images of different sizes.

Speech-to-text. We use `Whisper`[5], the ultimate tool for speech-to-text translation introduced by OpenAI in 2022, in three different sizes (i.e. *Tiny*, *Base* and *Small*). Each version has a different complexity and achieved accuracy (namely 93, 94 and 95%). For both models, the number of operations is fixed with respect to the input size and can be known a priori: Yolo uses convolutional layers, where the total number of operations depends deterministically on the image's resolution; Whisper processes input audio by dividing it into segments of 30 seconds, performing a fixed and predictable sequence of operations for each segment. Other popular examples include text-to-image generation models like Stable Diffusion [305] and several other ML tasks for which the number of operations and output size can be estimated (e.g., classification, regression, image segmentation, graph neural network for predictions on graphs, etc.). Our approach also abstracts from the model reduction technique employed to obtain the various sizes of the models. We consider variable numbers of machines equipped with a heterogeneous set of Nvidia GPUs (5060, 5070, 5080, 5090, A100, H100). Carbon intensities for 4 regions were selected from data available at [306]. We generated four different PUEs and selected realistic values for the parameters in the communication delay [307].

Benchmarks

We compare our solution to the eGLB (Online Geographical Low Balancing for Environmentally Equitable AI) algorithm proposed in [284]. The authors consider a distributed data center architecture for serving online user AI queries, minimizing energy consumption, water footprint, and carbon emissions in the long term. At each time step, after observing the number of demands, eGLB minimizes the primal problem, i.e., a linear combination of the objective functions weighted by Lagrangian multipliers that are iteratively updated using dual mirror descent. We adapt eGLB to our scenario by replacing its objectives with a combination of Equations A.12, A.13, and A.14. The original formulation of the primal decision problem in eGLB poses the following constraints: *(i)* all demands must be satisfied (i.e., $\sum_{k,i,j} x_{kij} = a(t)$); *(ii)* no more than a_k demands can be assigned to each machine M_k $\forall t$, where a_k is the maximum number of demands a machine M_k can serve in a single time step. We highlight that the satisfaction of these constraints cannot be guaranteed for high arrival rates, i.e., if $\exists t$ such that $a(t) > \sum_k a_k$. Additionally, we consider heterogeneous demands that potentially require a different number of operations, even if executed by the same model. Furthermore, the model used to process each demand is not known a priori, but depends on the decision made by the optimizer to serve the demand (ω_{ij}). As we consider the capacity of a machine in terms of GFLOPS instead of jobs per second, constraint *(ii)* becomes $\sum_{i,j} x_{kij}\omega_{ij} \leq \text{GFLOPS}_k$, $\forall k$. To cope with these issues and to widen the space of feasible solutions for eGLB, we also implement an adapted version of eGLB, which we call eGLB_U (eGLB Uncon-

[4] `https://ultralytics.com/`

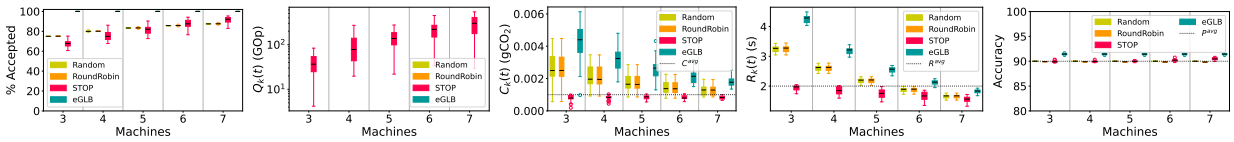
[5] `https://openai.com/index/whisper/`

strained), that has the same feasible solution space as STOP. In eGLB_U, we drop constraints *(i)* and *(ii)* and introduce a new term in the objective of the primal problem, which accounts for the number of accepted queries, similarly to our formulation in Equation A.17a. In this way, eGLB_U has the same flexibility to drop some demands as STOP, while maximizing the acceptance rate. We also implement two baseline solutions: Random, and Round Robin (RR). The first assigns each demand uniformly at random to a machine M_k with $k \in \{0, 1, \dots, K\}$ and a model $j \in \{1, \dots, J\}$, where assigning to M_0 means dropping the demand. In the RR strategy, upon receiving $a(t)$ demands, the algorithm assigns $\lfloor \frac{a(t)}{K+1} \rfloor$ demands to $K + 1 - a(t) \mod (K + 1)$ machines, $\lceil \frac{a(t)}{K+1} \rceil$ demands to other $a(t) \mod (K + 1) - 1$ machines, whereas $\lceil \frac{a(t)}{K+1} \rceil$ demands are discarded. The i-th demand assigned to a machine M_k is solved with model $i \mod (J)$. We evaluate all methods based on their acceptance rate as a measure of the achieved throughput.

Implementation

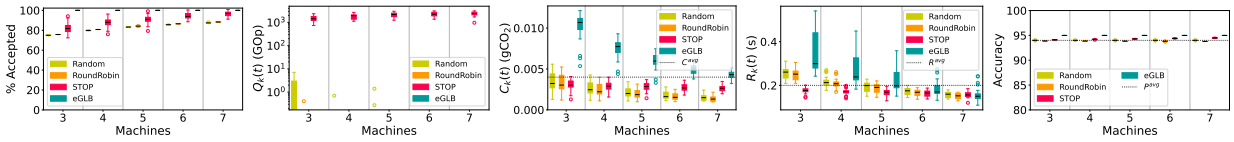
We implement our system and all benchmark solutions in C++ on a cluster of 50 DELL EMC nodes equipped with processors 2x AMD EPYC 7301 2.2GHz, 16 Core. LP and ILP problems in STOP and eGLB are solved with the `Highs`[6] solver. The source code is available in the GitHub repository[7].

Results



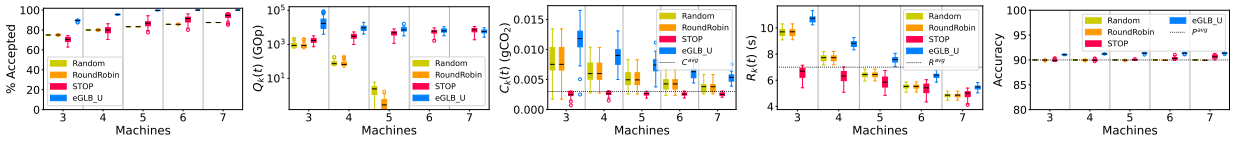
(a) Accepted queries. **(b)** System queues. **(c)** Emissions. **(d)** Response time. **(e)** Accuracy.

Figure A.6: Yolo, $A^{\max} = 1000$, $C^{avg} = 0.001$, $R^{avg} = 2$s, $P^{avg} = 90\%$.



(a) Accepted queries. **(b)** System queues. **(c)** Emissions. **(d)** Response time. **(e)** Accuracy.

Figure A.7: Whisper, $A^{\max} = 100$, $C^{avg} = 0.004$, $R^{avg} = 0.2$s, $P^{avg} = 94\%$.



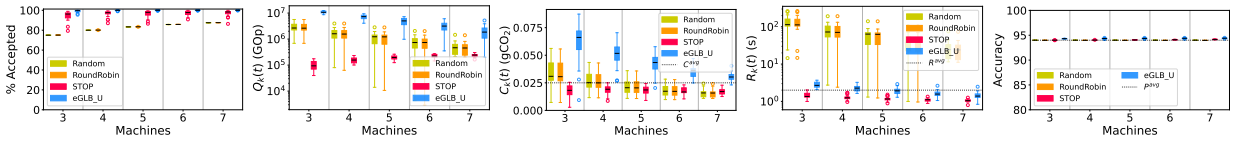
(a) Accepted queries. **(b)** System queues. **(c)** Emissions. **(d)** Response time. **(e)** Accuracy.

Figure A.8: Yolo, $A^{\max} = 3000$, $C^{avg} = 0.003$, $R^{avg} = 7$, $P^{avg} = 90\%$.

We run comparative tests by varying the number of heterogeneous machines $K \in \{3, \dots, 7\}$ and running all methods for $T = 300$ time steps. The scope of the experiments is to show how the implemented methods **maximize the acceptance rate while satisfying long-term average**

[6] `https://highs.dev/`

[7] `https://github.com/giuseppemasi99/STOP`



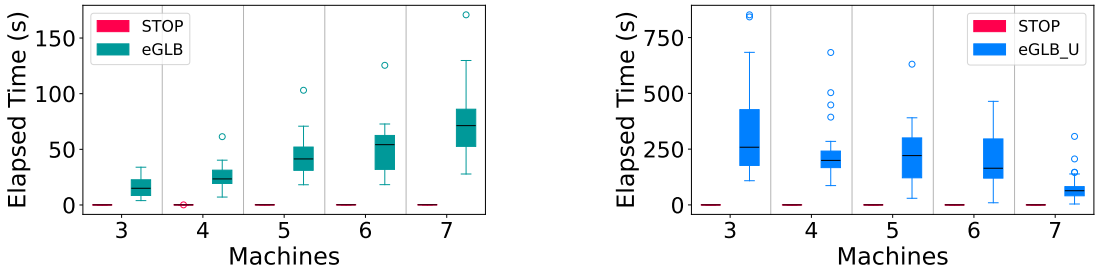
(a) Accepted queries. **(b)** System queues. **(c)** Emissions. **(d)** Response time. **(e)** Accuracy.

Figure A.9: Whisper, $A^{\max} = 1000$, $C^{avg} = 0.025$, $R^{avg} = 2$, $P^{avg} = 94\%$.

constraints. We study different experimental scenarios, varying the ML model and the maximum number of demand arrivals, $A^{\max} \triangleq \max_t a(t)$. In the first two scenarios (Figures A.6 and A.7), we run Yolo and Whisper with $A^{\max} = 1000$ and $A^{\max} = 100$, respectively. In these cases, the arrival rates are such that all instances are feasible for eGLB. Notice that we set a lower arrival rate for Whisper, since this model is more computationally intensive than Yolo. The other two scenarios in Figures A.8 and A.9 show the results on Yolo with $A^{\max} = 3000$ and Whisper with $A^{\max} = 1000$, respectively. Here, eGLB might fail, as the demand load might be higher than the system capacity. For this reason, we omit eGLB from these experiments, and we show the results of eGLB_U instead. Notice that the system capacity increases with the number of machines. As a consequence, since the arrivals are fixed $\forall K \in \{3, \dots, 7\}$, the average percentage of accepted demands (Figures A.6a-A.9a) grows for all methods, while each machine is less loaded, resulting in decreasing emissions (Figures A.6c-A.9c) and response times (Figures A.6d-A.9d). In all experimented scenarios, Random and RR have similar behaviors and assign accepted demands in a balanced way across machines and models (notice that their percentage of accepted demands is the same across all experiments, and that, by assigning uniformly the demands across the models, they always achieve P^{avg}, Figures A.6e-A.9e). Nevertheless, they do so without considering the imposed constraints. In scenarios with low arrival rates, as expected, eGLB consistently achieves maximum throughput (Figures A.6a, A.7a), but violates the emissions (Figures A.6c, A.7c) and response time Figures A.6d, A.7d) constraints, especially when the system has lower capacity (i.e., for the smallest values of K). In contrast, the same figures show how STOP can adjust the number of accepted demands to meet the constraints in all configurations. Notice that STOP can lower the accuracy when the system has limited capacity (smaller Ks) while still being $\geq P^{avg}$, in order to adjust the CO_2 emissions and the response times. This flexibility is not guaranteed with eGLB. Although RR and Random do not account for the long-term constraints, they almost always meet the emissions and response time bounds for the experiments on Whisper in Figure A.7. We highlight that they do so while achieving less throughput than STOP, which is the metric that we aim at maximizing in MTDO. Although sometimes their emissions are lower than STOP's ones (Figure A.7c), our goal is not to minimize emissions, but to keep them below the C^{avg} threshold. Unsurprisingly, in the first two scenarios, STOP has non-zero system queues (Figures A.6b, A.7b), which are nevertheless well below the machines' average capacity (46973 GLOPS). This phenomenon is expected, as eGLB satisfies the constraint of not exceeding the machines' capacities, without benefiting from their buffers. In the high arrival rates experiments (Figures A.8 and A.9), STOP's performance is consistent with the results of the first scenarios, as it is capable of maximizing the percentage of accepted demands while adjusting it to meet the long-term constraints and keeping the queues stable. In contrast, eGLB_U, RR and Random violate the constraints (Figures A.8c, A.8d, A.9c, A.9d), except for the cases with more machines. We highlight that, for RR and Random, this is just a consequence of the higher capacity of the system when more machines are available. While eGLB_U accounts for the long-term constraints, its formulation is less effective than STOP in satisfying them and in

guaranteeing the backlog's stability, proving the superior ability of our Lyapunov-based algorithm to maximize the throughput while satisfying metrics bounds.

Execution time To evaluate the execution times of STOP, eGLB, and eGLB_U, we select the scenarios of Figures A.6 and A.9 (the other two are omitted for space reasons). Figure A.10 shows the effectiveness of STOP to recognize easy instances and solve them in polynomial time and to reduce the solution search space when an ILP solver is required (we recall that algorithms for ILP problems, such as Branch-and-Bound or Branch-and-Cut, have exponential worst-case time complexity). eGLB and eGLB_U instead only call ILP solvers. The high execution times of the eGLB methods prevented us from running comparative experiments with 4 Whisper model sizes available: after three days of computations, we could not complete the tests. To overcome this issue, we set a 600-second time limit for the solver. In many cases, this amount of time was not enough for eGLB to find feasible solutions, whereas eGLB_U's achieved throughput was well below that of STOP, RR, and Random. This study shows that STOP can be effectively implemented on a dispatcher to find the optimal solution to StEP within the 1-second interval between two consecutive time steps.



(a) Yolo, $A^{\max} = 1000$. **(b)** Whisper, $A^{\max} = 1000$.

Figure A.10: Execution times.

Deferral Policy Finally, we propose a *deferral mechanism*: under the assumption that the arrival rate λ is lower than the service rate μ, demands that are not immediately instantiated upon arrival are placed in a queue at the dispatcher, which will assign them to a machine in subsequent time steps. This can happen if the machines' queues are currently too loaded, or if the most efficient machines (i.e., the fastest ones and those with the lowest PUE and CI) are busy. To implement the deferral, we introduce a new virtual queue $D(t)$ in the objective of StEP (Equation A.21), which updates as in Equation A.15, depending on the following quantity:

$$dis(\mathbf{X}(t)) = \sum_{i} (t - t_i^{arr}) \Big(1 - \sum_{k,j} x_{k,i,j}\Big) - 1, \tag{A.24}$$

where t_i^{arr} is the arrival time step of demand i, and the constraint $\limsup_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t} \mathbb{E}[dis(\mathbf{X}(\tau))] \leq 0$ is also considered. The "-1" in Equation A.24 allows the dispatcher to hold the demands while limiting the waiting time to be on average 1 second at maximum. The term $dis(\mathbf{X}(t))$ penalizes the delay in demand service, with the penalty increasing as the waiting time for a demand in the queue, $(t - t_i^{arr})$, grows. Hence, the dispatcher finds a tradeoff between serving them on the fly, possibly on some non-efficient or slower machine, and keeping them on hold. As we assume a 1-second time unit, each iteration where a demand is not allocated increases the response time of the machine

that eventually serves it by 1 second. We highlight that, by adding the demands standing by in the dispatcher's queue to the arrivals, the resulting process might not be i.i.d., possibly violating the hypothesis of Theorem A.2.3. Nevertheless, a similar optimal $[O(1/V), O(V)]$-approximation, as well as the mean rate stability of the queues, including $D(t)$, is still guaranteed by STOP [283] (details are omitted for space reasons). With this mechanism, if $\lambda < \mu$ the system will eventually serve all the incoming demands (acceptance rate =100%), even though at some time steps the incoming demands and their sizes exceed the capacity of the system (i.e., $\sum_k \text{GFLOPS}_k$). This is in contrast with the hard constraint of serving all the demands $\forall t$ posed in eGLB: every single instance of StEP is still feasible, and STOP still makes decisions that satisfy the long-term average carbon/response time/performance constraints. In Figure A.11, we show the evolution of the dispatcher's queue for the same instances of Figure A.8, when $K = 6$. Shades represent the standard deviation over 30 seeds. By terminating the simulation and interrupting the arrivals, the acceptance rate is 100%. The maximum waiting time for a demand in the dispatcher's queue was 4 seconds. By applying Little's Law, we get that the average amount of time the incoming demands spend traversing the dispatcher is 0.14s. Although the response time has been increased to 6.5 seconds compared to Figure A.8d, it still respects the constraint of 7 seconds. We leave a more extensive analysis of the deferral mechanism and non-i.i.d. arrivals for future work.

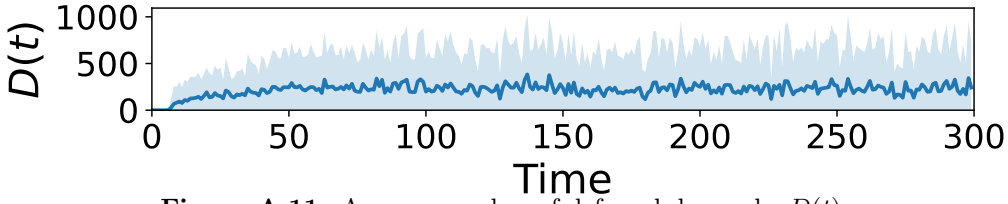


Figure A.11: Average number of deferred demands, $D(t)$.

A.2.6 Conclusions

We consider the problem of maximizing the throughput of users' inference demands under carbon footprint, response time, and accuracy constraints. We propose STOP, an online framework grounded in the application of Lyapunov's theory for stochastic network optimization that guarantees long-term constraint satisfaction and system stability. STOP shows high flexibility and self-adaptation capability, allowing it to be deployed in dynamic scenarios. The theoretical analysis of the proposed algorithm is further supported by an extensive experimental campaign highlighting its advantages against the current state-of-the-art. A deferral mechanism is included under the assumption of system stability. As future work, we will provide a more detailed theoretical and experimental analysis of the deferral mechanism under skewly distributed arrivals. We will extend our study to AI models with non-deterministic output sizes.