Feature Programming for Multivariate Time Series Prediction

Alex Reneau [*1] **Jerry Yao-Chieh Hu** [*1] **Ammar Gilani** [1] **Han Liu** [1 2]

Abstract

We introduce the concept of programmable feature engineering for time series modeling and propose a feature programming framework. This framework generates large amounts of predictive features for noisy multivariate time series while allowing users to incorporate their inductive bias with minimal effort. The key motivation of our framework is to view any multivariate time series as a cumulative sum of fine-grained trajectory increments, with each increment governed by a novel spin-gas dynamical Ising model. This fine-grained perspective motivates the development of a parsimonious set of operators that summarize multivariate time series in an abstract fashion, serving as the foundation for large-scale automated feature engineering. Numerically, we validate the efficacy of our method on several synthetic and real-world noisy time series datasets.

1. Introduction

We investigate the problem of automated time series feature engineering for prediction tasks in the regression setting. A programmable feature engineering framework is proposed, named feature programming, for multivariate time series modeling. Our framework facilitates the automatic generation of large amounts of meaningful features from raw data. Simultaneously, it enables the incorporation of domain knowledge through the use of feature templates, which are customizable lists of both raw and hand-crafted features provided by users.

Our key motivation comes from a novel dynamical Ising-like model, the spin-gas Glauber dynamics, originated from a newly debuted gas-like interaction that includes momentum and acceleration information. By using spin-gas Glauber dynamics as the fundamental model for time series generating processes at the smallest time scale, we explore the potential of treating time series as the path-sum of infinitesimal increments generated by a series of Markovian coin tosses following the spin-gas Glauber dynamics. From such a fine-grained perspective, a set of operators is motivated for extracting informative features in an abstract fashion. We introduce the idea of the feature programming framework as a three-step pipeline for feature generation (Figure 1):

1. Design a customized three-level feature template, which, at each level (list), includes both raw features from the data and user-specified (discretionary) features.
2. Implement a programmable operation module consisting of pre-specified operations by the user based on the proposed operator combinations.
3. Generate a large number of predictive features automatically by feeding the feature template into the programmable module, following a user-specified hierarchical feature generation rule encoded in the operation module.

For more comprehensive details, see Section 4, and for concrete case studies, please see Appendix E.2.

Contributions. Our contribution is three-fold.

- Methodologically, we present a novel dynamical Ising-style model for time series data generation. This model motivates a parsimonious set of operators for mining valuable features from time series data. Building on this, we introduce a feature programming framework, which enables the systematic generation of a large number of features by leveraging the motivated operators and a model-based hierarchical feature generation rule. Additionally, this framework provides the flexibility for users to incorporate discretionary inductive biases through feature templates.
- Theoretically, to the best of our knowledge, our framework is the first automated feature generation method for

[*]Equal contribution [1]Department of Computer Science, University of Northwestern, Evanston, USA [2]Department of Statistics and Data Science, University of Northwestern, Evanston, USA. Correspondence to: Alex Reneau <alexreneau@u.northwestern.edu>, Jerry Yao-Chieh Hu <jhu@u.northwestern.edu>, Ammar Gilani <ammargilani2024@u.northwestern.edu>, Han Liu <hanliu@northwestern.edu>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

Code is available at GitHub; the most updated version is available on arXiv with a full list of authors, including Chenwei Xu and Weijian Li. We kindly request that citations refer to the arXiv version: https://arxiv.org/abs/2306.06252.

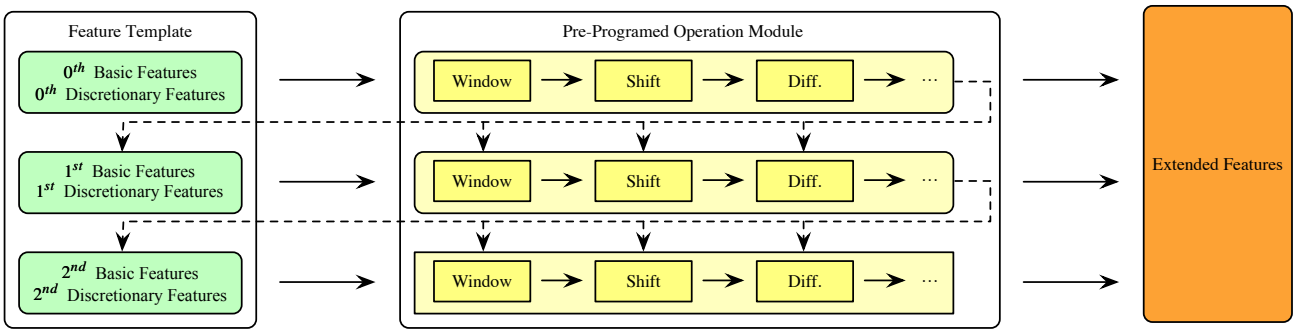

Figure 1. Feature Programming Pipeline. **Left:** Customizable Feature Template. **Mid:** Pre-Programmed Operation Module. **Right:** Generated Extended Features. Customization involves user-specific feature templates, operation modules, and user-designed feature flows (arrows). Solid arrows represent programmed flow within each order level, while dashed arrows indicate order-by-order feature generation flow between levels. See Appendix E.2 for practical examples illustrating the customization procedures.

time series equipped with a solid theoretical foundation. We show that appropriate selection of the interaction term in the spin-gas Glauber dynamics facilitates the derivation of a multivariate dynamical model from univariate exponential family distributions, thus enabling efficient learning and inference. The conditions for such a construction and its unique determination are also identified.

- Empirically, the superiority of the generated features is demonstrated through extensive experiments on multivariate time series prediction tasks. These experiments involve various advanced models, practical scenarios, customization examples, and ablation studies, showcasing our method's adaptability, flexibility, and robust feature generation. Specifically, in our most challenging multi-horizon tasks (predicting 20 future values using a length-20 sequence, see Table 5), our features significantly improve prediction accuracy (on average in 88+% R^2 and 27+% in Pearson correlation) across all models.

Related Works. Time series modeling has been regarded as one of the most difficult problems in machine learning for the past few decades given its temporal ordering nature and noise-sensitivity from sequential structure (Masini et al., 2023; Fawaz et al., 2019; Ozaki, 2012). It is well-known that features play a major role in time series ML prediction problems (Meisenbacher et al., 2022; Cerqueira et al., 2021; Christ et al., 2016). While there is rich literature on data augmentation (Wen et al., 2020; Iwana & Uchida, 2021) and architectures (Tealab, 2018; Balkin & Ord, 2000; Sezer et al., 2020), little has been done toward handling features for time series modeling and most methods prefer an end-to-end feature selection (Barandas et al., 2020; Christ et al., 2016; Sun et al., 2015; Längkvist et al., 2014; Chandrashekar & Sahin, 2014) or hand-crafted (pre-defined) feature design (Zhou et al., 2021; Gu et al., 2020; Christ et al., 2018; Muralidhar et al., 2018; Kakushadze, 2016; Christ et al., 2016) without fundamental principles.

For each application domain, abovementioned approaches are popular and intuitive, however, they suffer from two potential issues: (i) being too purpose-specific and not transferable from one domain to another (Zhuang et al., 2020; Weiss et al., 2016; Glorot et al., 2011; Pan et al., 2010; Pan & Yang, 2009); (ii) quite some representation effort is spent on recovering the "noise" part of time series data (Wen et al., 2020; Iwana & Uchida, 2021; Chepurko et al.; Raissi et al., 2019). To this end, our framework represents the first theoretically grounded automated feature engineering method for time series, characterized by its completeness and model-based fundamental principles.

Organization. In Section 2, we lay out the problem setting and preliminaries. In Section 3, we introduce the motivating physics model. In Section 4 we present the proposed feature programming framework. In Section 5, the construction of the motivating model from univariate exponential family distributions is discussed. In Section 6, experimental studies are conducted. In Section 7, concluding discussions are provided. More related works are discussed in Appendix A.

2. Problem Setting and Background

In this section, we first layout the problem setup of multivariate time series prediction and then introduce the idea we build upon: Glauber dynamics.

2.1. Multivariate Time Series Prediction Problem

Consider a multivariate time series dataset $\mathcal{D}$ (of size $|\mathcal{D}|$) with each multivariate time series being made up of N correlated univariate time series,

$$\mathcal{D} : \left\{ \left((\mathbf{X}_{t-T+1}, \ldots, \mathbf{X}_t), \mathbf{Y}_{t+1} \right)_\mu \right\}_{\mu=1}^{|\mathcal{D}|}.$$

$\mathbf{Y}_{T+1}$ is the one-step-ahead target, and $\mathbf{X}_t = (\mathbf{x}_{1,t}, \ldots, \mathbf{x}_{N,t})^\top \in \mathbb{R}^{N \times d}$ and $\mathbf{Y}_t = \left(\mathbf{y}_{1,t}, \ldots, \mathbf{y}_{M,t}\right)^\top \in \mathbb{R}^{M \times d'}$ are the input features and targets for each variate at each time step, respectively.

We investigate the problem of multivariate time series prediction in discrete-time in regression setting. Our goal is to predict the one-step-ahead target value, $\mathbf{Y}_{t+1}$, using a prediction model, f, that is trained on the dataset $\mathcal{D}$. The input to the model is a sequence of past T time steps, $(\mathbf{X}_{t-T+1}, \ldots, \mathbf{X}_t)$, and the output is the predicted next step $\widehat{\mathbf{Y}}_{t+1} := f(\mathbf{X}_{t-T+1}, \ldots, \mathbf{X}_t)$ for general sequence-to-one forecasting tasks. Throughout this work, for a given time step t, we use $\mathbf{x}_{i,t} \in \mathbb{R}^d$ to represent a set of covariates (features for ML models) at time step t, indexed by $i \in [N]$; we consider only scalar output for each variate and match the number of variates by setting $d' = 1$ and $M = N$.

In this work, instead of investigating the effectiveness of autoregressive and cross-sectional architectures, we focus on capturing these important time series characteristics by resorting to the quality of features. With any given architecture, we aim to improve the feature quality by engineering them to be more informative. Namely, for each variate, in addition to using the basic (raw) features $\mathbf{x}_{i,t}$ as the input, we can include engineered (extended) features $\overline{\mathbf{x}}_{i,t} \in \mathbb{R}^{\bar{d}}$ that summarize all information from the basic features, across both time (autoregressive) and series (correlations), for the model training.

2.2. Dynamical Ising Model: Glauber Dynamics

Glauber dynamics (Nguyen et al., 2017; Glauber, 1963), also known as the dynamical Ising model, a well-studied reversible Markov chain defined for any Markov random field originated from statistical mechanics. It has been applied to a variety of problems, including lattice models in condensed matter physics (Goldenfeld, 1992), spin glasses (Janssen, 1976), neural networks (Mezard & Montanari, 2009; Hinton et al., 1986), and traffic (Pan et al., 2023). It has also been extended to non-equilibrium systems (Swendsen & Wang, 1987) and used with mean-field theory to study complex systems (Montanari & Sen, 2022; Kadanoff, 2000).

To describe such a dynamical Ising model, we shall start with Ising model. We consider an Ising model of N spins as an exponential family model for binary N-spin data up to quadratic sufficient statistic taking the Boltzmann form

$$P(\boldsymbol{\sigma}) = \frac{1}{\mathcal{Z}} \exp\left\{ -\beta \left(\sum_{e_{ij} \in \mathcal{E}} J_{ij}\sigma_i\sigma_j + \sum_{v_i \in \mathcal{V}} h_i\sigma_i \right) \right\}, \tag{2.1}$$

where $\boldsymbol{\sigma} := \{\sigma_1, \cdots, \sigma_N\} \in \{\pm 1\}^N$ is the configuration of N binary random variables (spins) $\sigma_i \in \{\pm 1\}$ assigned to the Ising graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, $\beta \geq 0$ is the inverse temperature, and $\mathcal{Z} := \sum_{\boldsymbol{\sigma}} e^{-\beta E(\boldsymbol{\sigma})}$ is the partition function ensuring the normalization of $P(\boldsymbol{\sigma})$. The symmetric $N \times N$ strength matrix $J \in \mathbb{R}^{N \times N}$, with zeros on the diagonal, and the external field vector $h \in \mathbb{R}^N$ encode the graphical structure of the Ising model. For simplicity, we set the inverse temperature $\beta = 1$ throughout this paper.

The dynamics appears as we start treating (2.1) as a dynamical process and considering the time ordering of the configurations, namely with time index $\boldsymbol{\sigma}_t$. For the purpose of this work, we introduce only the discrete-time version of Glauber dynamics. This implies that for each time step t, the process begins at a potentially random initial configuration, σ_t. Subsequently, we assign new values to all spins independently (for each spin) for the next step in accordance with the stochastic updating rule:

$$P\left(\sigma_{i,t+1} | \boldsymbol{\sigma}_t\right) = \frac{\exp\{\sigma_{i,t+1}\gamma_{i,t}\}}{2\cosh \gamma_{i,t}}, \tag{2.2}$$

where $\gamma_{i,t} := \sum_j J_{ij}\sigma_j + h_i$ is the ith effective local field at time step t. Notably, $\gamma_{i,t}$ is computed only for the ith spin, and the parameters of the Ising model, (J, h), are time-independent. In this work, we consider a Glauber dynamics with temporal update satisfying the following evolutionary independent assumption.

Assumption 2.1 (Evolutionary Independence)**.** We assume the spin updates are independent events on same time slice, or equivalently, $P(\boldsymbol{\sigma}_{t+\delta t}|\boldsymbol{\sigma}_t) = \prod_{i=1}^N P(\sigma_{i,t+\delta t}|\boldsymbol{\sigma}_t)$.

3. Spin-Gas Glauber Dynamics

In this section, we present the spin-gas Glauber dynamics as the motivating model for feature programming by incorporating momentum and acceleration effects into the standard Glauber dynamics to make the spin values gas-like in the effective local field. We shall see that this modified Glauber dynamics can naturally serve as a foundational model for multivariate time series from a fine-grained perspective. For the ease of presentation, we reduce the feature dimension to $d = 1$ by representing each covariate $\mathbf{x}_t$ as a scalar $x_t \in \mathbb{R}$, and therefore $\mathbf{X}_t = (x_{1,t}, \ldots, x_{N,t})^\mathsf{T} \in \mathbb{R}^N$. Yet, the setting of $d = 1$ is primarily for illustrative purposes. In fact, as we shall see in Section 4, our methodology is general enough to handle $d \geq 1$.

Fine-Graining in Time. Consider a multivariate time series consisting of two endpoints $\mathbf{X}_t$ and $\mathbf{X}_{t+\Delta t}$, separated by a time interval of Δt. We divide this time interval into L segments by introducing a sufficiently large L, i.e., $\Delta t := L\delta t$. The multivariate time series path difference $\Delta\mathbf{X}_t := \mathbf{X}_{t+\Delta t} - \mathbf{X}_t$ is therefore the sum of L consecutive increments for each univariate series. We then treat each increment as a binary outcome coin toss with $\sigma = \{\pm 1\}$ for each univariate series (or $\boldsymbol{\sigma} = \{\pm 1\}^N$ for multivariate.) Consequently, a random process of L consecutive coin tosses generates the path difference $\Delta\mathbf{X}$, and it gives the endpoint value as:

$$\mathbf{X}_{t+\Delta t} := \mathbf{X}_t + \Delta\mathbf{X}_t = \mathbf{X}_t + c\sum_{l=1}^{L} \boldsymbol{\sigma}_{t+l\delta t}, \tag{3.1}$$

where c is a rescaling factor ensuring $\Delta \mathbf{X}_t$ is bounded.

Furthermore, we assume that each coin toss with binary random variables is sampled from the conditional $\boldsymbol{\sigma}_{t+l\delta t} \sim P(\boldsymbol{\sigma}_{t+l\delta t}|\boldsymbol{\sigma}_{t+(l-1)\delta t})$, governed by the following *irreversible*[1] N-spin modified Glauber dynamics. This results in a fundamental model for time series generation from the fine-grained perspective.

Spin-Gas Glauber Dynamics. We express the coin tossing process as a dynamical Ising-like system, whose time evolution is a Markov process, and hence each coin toss $\boldsymbol{\sigma}_{t+\delta t}$ (the infinitesimal increment after a δt time interval conditioned on current time t) is governed by

$$P\left(\sigma_{i,t+\delta t}|\boldsymbol{\sigma}_t\right) = \frac{\exp\{\sigma_{i,t+\delta t}\Gamma_{i,t}\}}{2\cosh\Gamma_{i,t}}, \tag{3.2}$$

where[2] $\sigma_{i,t+\delta t} = \{\pm 1\}$, and $\Gamma_{i,t}$ is a novel effective local field associated to the i'th spin at time t defined as follows. Let $\boldsymbol{p}_t := c\frac{\delta \boldsymbol{\sigma}_t}{\delta t} = \frac{c(\boldsymbol{\sigma}_t - \boldsymbol{\sigma}_{t-\delta t})}{\delta t}$ and $\boldsymbol{a}_t := c\frac{\delta}{\delta t}\left(\frac{\delta \boldsymbol{\sigma}_t}{\delta t}\right) = \frac{\boldsymbol{p}_t - \boldsymbol{p}_{t-\delta t}}{\delta t}$ be the discrete counterparts of momentum and accelaration on node i. We have

$$\Gamma_{i,t} := \sum_j J_{ij}(t)\sigma_{j,t} + h_i(t) + g_i\left(\boldsymbol{p}_t, \boldsymbol{a}_t\right), \tag{3.3}$$

with a newly debuted *gas-like* interaction term g including the effects of momentum $\boldsymbol{p}_t$ and acceleration $\boldsymbol{a}_t$. Intuitively, $\Gamma_{i,t}$ summarizes the structural contributions from all other series with the first two terms in (3.3) similar to the traditional Glauber dynamics. In addition, with the newly introduced g, it includes contributions from momentum and acceleration of the series itself and all other series at t, hence gas-like. We emphasis that, the newly introduced dynamical Ising model (3.2) is the fine-grained model for the δt-infinitesimal change of multivariate time series (with stepwise change $\boldsymbol{\sigma}_t$), and serves as the fundamental dynamical model of the time series generation.

Reverting to Continuous Time through Coarse-Graining. To tailor our model to handle generic time series, we must revert (3.2) back to a continuous timeframe. This involves a process known as coarse-graining, where we perform path-sum of $\boldsymbol{\sigma}_t$ from δt to $L\delta t$ according to (3.1). Following this, we take the continuous time limit and identify the endpoint value of the trajectory, which is the cumulative sum of the random process over time starting from $\mathbf{X}_t$. This approach bears similarities to the path-integral representation utilized for diffusion processes in physics, as elaborated in (Graham, 1977; Wissel, 1979).

Interestingly, when each coin toss follows the Markov spin-gas Glauber dynamics (3.2), it is possible to calculate the conditional path probability, given an initial condition $\mathbf{X}_t$. This is achieved by first decomposing the joint distribution $P(\boldsymbol{\sigma}_t, \ldots \boldsymbol{\sigma}_{t+L\delta t})$ as

$$P(\boldsymbol{\sigma}_t, \ldots \boldsymbol{\sigma}_{t+L\delta t}) = \prod_{l=1}^{L} P(\boldsymbol{\sigma}_{t+l\delta t}|\boldsymbol{\sigma}_{t+(l-1)\delta t}), \tag{3.4}$$

and then summing out all intermediate variables ranging from $\boldsymbol{\sigma}_{t+\delta t}$ to $\boldsymbol{\sigma}_{t+(L-1)\delta t}$ such that[3]

$$\begin{aligned} P\left(\mathbf{X}_{t+\Delta t}|\mathbf{X}_t\right) &= P\left(\boldsymbol{\sigma}_{t+L\delta t}|\boldsymbol{\sigma}_t\right) \qquad (3.5)\\ &= \prod_{l=2}^{L} \sum_{\{\boldsymbol{\sigma}_{t+(l-1)\delta t}\}} P(\boldsymbol{\sigma}_{t+l\delta t}|\boldsymbol{\sigma}_{t+(l-1)\delta t}) \cdot P(\boldsymbol{\sigma}_{t+\delta t}|\boldsymbol{\sigma}_t)\\ &= \int \prod_{l=2}^{L} \mathrm{d}\boldsymbol{\sigma}_{t+(l-1)\delta t}\, P(\boldsymbol{\sigma}_{t+l\delta t}|\boldsymbol{\sigma}_{t+(l-1)\delta t}) \cdot P(\boldsymbol{\sigma}_{t+\delta t}|\boldsymbol{\sigma}_t)\\ &\propto \int \prod_{l'=2}^{L} \mathrm{d}\boldsymbol{\sigma}_{t+(l'-1)\delta t} \exp\left\{\sum_{l=1}^{L}\left(\sum_{i=1}^{N}\Gamma_{i,t+(l-1)\delta t}\sigma_{i,t+l\delta t}\right)\right\}. \end{aligned}$$

To match the common notation used in physics, in the third line we abuse the notation by writing the summations over configurations as integrals, for more notational details see (Graham, 1977). Denoting $\mathcal{D}\left[\boldsymbol{\sigma}_\tau\right] := \prod_{l'=2}^{L} \mathrm{d}\boldsymbol{\sigma}_{t+(l'-1)\delta t}$, the last line of (3.5) can be understood as a well-defined discrete path integral

$$P\left(\mathbf{X}_{t+\Delta t}|\mathbf{X}_t\right) \propto \int \mathcal{D}\left[\boldsymbol{\sigma}_\tau\right] \exp\left\{\sum_{l=1}^{L} \mathcal{L}\left(\boldsymbol{\sigma}_{t+(l-1)\delta t}, \boldsymbol{\sigma}_{t+l\delta t}\right)\right\},$$

with Lagrangian

$$\mathcal{L}\left(\boldsymbol{\sigma}_{t+(l-1)\delta t}, \boldsymbol{\sigma}_{t+l\delta t}\right) := \sum_{i=1}^{N} \Gamma_{i,t+(l-1)\delta t} \cdot \sigma_{i,t+l\delta t} - \ln\left(e^{\Gamma_{i,t}} + e^{-\Gamma_{i,t}} + 1\right),$$

up to a constant coefficient (Graham, 1977)[4]. Here, $\exp\left\{\sum_{l=1}^{L} \mathcal{L}\left(\boldsymbol{\sigma}_{t+(l-1)\delta t}, \boldsymbol{\sigma}_{t+l\delta t}\right)\right\}$ is a functional (the exponentiated action functional) on the space of all (discrete) paths (denoted as $[\boldsymbol{\sigma}_\tau] := [\boldsymbol{\sigma}_t, \boldsymbol{\sigma}_{t+\delta t}, \ldots, \boldsymbol{\sigma}_{t+L\delta t}]$), i.e., discrete functions that represent these paths. Moreover, $\mathcal{D}\left[\boldsymbol{\sigma}_\tau\right]$ is a measure of integration/summation over all possible

[1] By irreversible, we consider an out-of-equilibrium Glauber dynamics by generalizing (J, h) to be time-dependent, see (Nguyen et al., 2017; Vázquez et al., 2017) and reference therein.

[2] Although the binary random variables of standard Ising model leads to discrete increments, this model of coin toss can also be used to model continuous time series by rescaling $\sigma_{i,t+\delta t}$. That is, with a sufficiently large L, by taking $c \ll 1$, the time series $\mathbf{X}_t$ becomes approximately continuous as $\Delta \mathbf{X}$ becomes continuous.

[3] Recall that, since the each coin toss is Markovian, the joint distribution $P(\boldsymbol{\sigma}_t, ..., \boldsymbol{\sigma}_{t+L\delta t})$ can be recursively expanded as $\prod_{l=1}^{L} P(\boldsymbol{\sigma}_{t+l\delta t}|\boldsymbol{\sigma}_{t+(l-1)\delta t})$.

[4] Note that, it is a common practice to term $\mathcal{L}$ as "Lagarngian" here since it ties the path integral formulation with the principle of least action in classical mechanics (Feynman et al., 2010). More precisely, the Lagrangian $\mathcal{L}$, when integrated or summed over a time period, provides us with a quantity called "action". This action makes an appearance in the exponent of (3.5). As we shall see next, this action takes a central role in the path integral formula, as the path taken by a system between two configurations is the one for which the action is minimized.

paths $[\boldsymbol{\sigma}_\tau]$ connecting $\mathbf{X}_t$ and $\mathbf{X}_{t+\Delta t}$ in that space. Importantly, the space of functions under consideration is defined by the details of the Lagrangian $\mathcal{L}$.

When extending to continuous time limit, we reduce the time interval $\delta t \to 0$ and all infinitesimal increment size $c\boldsymbol{\sigma}_t \to 0$ (by setting $c \to 0$), and assume the multivariate $\mathbf{X}_\tau$ for $\tau \in [t, t+\Delta]$ is a Markovian stochastic process with continuous sample paths. The multivariate time series endpoint value (position) $\mathbf{X}_{t+\Delta t}$ and time $t + \Delta t$ of the system can be therefore expressed as (with the starting position $\mathbf{X}_t$ and time t): $\mathbf{X}_t + c\sum_{l=1}^{L} \boldsymbol{\sigma}_{t+l\delta t} \to \mathbf{X}_t + \Delta\mathbf{X}_t$, and $t + L\delta t \to t + \Delta t$, where $\Delta\mathbf{X}_t$ and Δt are now continuous. The difference momentum $\frac{\delta \mathbf{X}_t}{\delta t}$ and acceleration $\frac{\delta}{\delta t}\left(\frac{\delta \mathbf{X}_t}{\delta t}\right)$ hence become their derivative counterparts by taking large L limit (with proper scaled $\sigma_{i,t+\delta t}$): $\boldsymbol{p}_t := \frac{\partial \mathbf{X}_t}{\partial t}$ and $\boldsymbol{a}_t := \frac{\partial}{\partial t}\left(\frac{\partial \mathbf{X}_t}{\partial t}\right)$. Consequently, in continuous time limit, we obtain the formal expression of the continuous analog of conditional (3.5):

$$P\left(\mathbf{X}_{t+\Delta t}|\mathbf{X}_t\right) \propto \int_{\mathbf{X}_t}^{\mathbf{X}_{t+\Delta t}} \mathcal{D}[\mathbf{X}_\tau] \exp\{-S\left([\mathbf{X}_\tau]\right)\}, \quad (3.6)$$

up to a normalization constant; where $\mathcal{D}[\mathbf{X}_\tau]$ is the integration measure in functional space of all possible paths $[\mathbf{X}_\tau]$ between $\mathbf{X}_t$ and $\mathbf{X}_{t+\Delta t}$, $S\left([\mathbf{X}_\tau]\right) := \int_t^{t+\Delta t} \mathrm{d}\tau\ \mathcal{L}(\tau)$ is the action functional. Both $\mathcal{D}[\mathbf{X}_\tau]$ and $S\left([\mathbf{X}_\tau]\right)$ are explicitly determined by details of the spin-gas Glauber dynamics via Lagrangian $\mathcal{L}(\tau)$. It is worthy to note that, only (3.5) is suitable for direct calculations, whereas (3.6) is purely formal with more subtle use cases. This is because $S\left([\mathbf{X}_\tau]\right)$ cannot be trivially derived from the Riemann sum $\sum_l \mathcal{L}\left(\boldsymbol{\sigma}_{t+l\delta t}, \boldsymbol{\sigma}_{t+(l-1)\delta t}\right)$, see (Graham, 1977; Lau & Lubensky, 2007; Weber & Frey, 2017) for detailed constructions and examples.

As a motivating model, (3.5) and the formal expression (3.6) provide strong intuition for time series feature extraction and modeling from the data-driven perspective: the coarse-grained model $P(\mathbf{X}_{t+\Delta t}|\mathbf{X}_t)$ is characterized by building blocks $\{\boldsymbol{\sigma}_{t+l\delta t}\}_{l=1}^{L}$, $\{\mathbb{E}\left[\boldsymbol{\sigma}_{t+l\delta t}|\boldsymbol{\sigma}_{t+(l-1)\delta t}\right]\}_{l=1}^{L}$, and $\{\boldsymbol{p}_{t+l\delta t}, \boldsymbol{a}_{t+l\delta t}\}_{l=1}^{L}$ for the following reasons. Firstly, an expression like (3.6) naturally suggests the most probable path between two endpoints is given by the the variation $\delta\left[\int_t^{t+\Delta t} \mathrm{d}\tau\ \mathcal{L}(\tau)\right]$, while, in practice, $\mathcal{L}(\tau)$ is estimated from data and conditionally dependent step by step on the most recent realization according to (3.5). Therefore, data from *finer* timeframe(s), $\{\boldsymbol{\sigma}_{t+l\delta t}\}_{l=1}^{L}$ and, their *smoothed* conditional expectations (conditional mean), $\{\mathbb{E}\left[\boldsymbol{\sigma}_{t+l\delta t}|\boldsymbol{\sigma}_{t+(l-1)\delta t}\right]\}_{l=1}^{L}$, are required to take into account both the fluctuations and the denoised most probable path in the modeling process. Moreover, the derivative data $\{\boldsymbol{p}_{t+l\delta t}, \boldsymbol{a}_{t+l\delta t}\}_{l=1}^{L}$ from the finer timeframe(s) is also required by the definition of $\mathcal{L}$. We emphasize the building block $\{\boldsymbol{p}_{t+l\delta t}, \boldsymbol{a}_{t+l\delta t}\}_{l=1}^{L}$ is not merely a straight readout from $\mathcal{L}$. Instead, as we shall see later in Section 5, it is complemented by our theoretical insights (Theorem 5.1).

We conclude this section by summarizing several advantages of modeling multivariate time series in the fine-grained perspective with the proposed dynamical Ising-like model: (i) it accommodates both autoregressive and cross-sectional interactions; (ii) its strong physics intuition leads to the important characteristics for feature extraction; (iii) as we shall see in Section 5, by carefully selecting the form of g, it has the ability to model a wide range of distributions and has closed-form multivariate densities, which enables efficient inference and learning with statistical guarantees.

4. Methodology

The feature programming framework comprises three crucial components (see Figure 1):

- A set of programmable operators (Difference, Window, Shift) that form the basis for generating features;
- A feature template that enables users to select the fundamental features they want to employ;
- A semi-automated order-by-order feature generation rule encoded in the operation module, that automatically creates extended features within each level and between levels, adhering to an upgrade rule.

Consequently, a feature program (which produces a set of extended features) consists of a user-specific feature template and a pre-programmed operation module. We discuss each part in the following.

4.1. The Difference, Window and Shift Operators

Inspired by the spin-gas Glauber dynamics, we propose three abstract operators for extracting features from time series data and discuss their operational specifics here.

Difference Operator. Motivated by $\{\boldsymbol{p}_{t+l\delta t}, \boldsymbol{a}_{t+l\delta t}\}_{l=1}^{L}$, we propose the difference operator that incorporates both continuous and finite differences and performs series-wise subtraction between any two series. Operationally, we define the difference operator, `Difference[series1, series2]`, as the generalized derivative operation that performs first smoothing then subtracting two input series (basic features), and generates curvature-like features resembling the momentum and acceleration in physics. With the difference operator, we characterize features (both basic and extended) into three hierarchical classes based on their *order* of derivative: 0th-, 1st-, and 2nd- order features, which correspond to the generalized notion of position, momentum, and acceleration of the input raw features $\mathbf{x}_{i,t}$, respectively.

Window Operator. Motivated by both $\{\boldsymbol{\sigma}_{t+l\delta t}\}_{l=1}^{L}$ and $\{\mathbb{E}\left[\boldsymbol{\sigma}_{t+l\delta t}|\boldsymbol{\sigma}_{t+(l-1)\delta t}\right]\}_{l=1}^{L}$, we propose the window operator which summarizes the information in a fixed look-back size, denoted as Δt, from multiple resolutions using denoised summary statistics such as maximum, minimum, and mean. Operationally, the window operator,

`Window[series, lookback_size]`, is defined as a function that takes an input series and a `lookback_size`, for each resolution, and subsequently outputs a series of summary statistics for the given lookback window. By applying the window operator with different lookback sizes, one can derive informative features from the time-rescaling property of the input series (Tallec & Ollivier, 2018).

Shift Operator. Lastly, we propose the shift operator that can create new series with arbitrary time differences from any existing series to complement the other operators. The shift operator, defined as `Shift[series, Δτ]`, allows for the input series to be shifted by any desired time difference, $\Delta\tau$, in order to incorporate more auto-correlated information.

4.2. Order-Upgrade Rule

Equipped with the aforementioned difference operator, we can generate higher-order features by applying it to lower-order features. Then, the window and shift operators can be used to create summary-features from these higher-order features. For example, we can create 1-st order series by applying the difference operator to two 0-th order series, and 2-nd order series by applying the difference operator to two 1-st order series. Additionally, applying the window and shift operators on these higher-order series can create summary-features without changing the order of the series.

4.3. Feature Template: Injecting Hand-Crafted Features

The feature template is a crucial component of the feature programming framework and serves as the starting point for the feature generation process. It consists of three *basic series lists*, one for each order of features (0th, 1st, 2nd) as demonstrated by the green boxes in Figure 1. These lists are initialized as a combination of a set of basic features derived from the raw data and a discretionary design list of hand-crafted features. As the feature generation process progresses, these lists are updated with features from the previous order, as indicated by the dashed arrows in Figure 1. This design allows for a high degree of flexibility and customization, as the basic series lists can be fully discretionary, completely default, or a combination of the two. Moreover, the basic series lists are fed into pre-programmed operators to generate extended features (the solid arrows in Figure 1), making the setup of the basic series lists at each order a critical step in the feature generation process.

4.4. Automate Order-by-Order Feature Generation

With the feature template and programmable operators, we generate extended features through a semi-automated process by controlling the flow into the operation module. Starting with the basic series lists specified in the feature template, we feed each list into an operation module that contains the pre-programmed operators. This process is performed order-by-order, where each basic series list of an order is operated on by the corresponding operations of that order in the module. This results in a hierarchical feature generation that generates extended features automatically within each level and between levels, following the upgrade rule. Through the combinatorial manipulation of operators in the module and computation flow (represented by the arrows), our feature programming framework enables the automated and programmable generation of extended features from basic ones. For specific examples of custom feature programs that illustrate the customization steps needed in real-world applications, please refer to Appendix E.2.

5. Theoretical Analysis

In this section, we show that, the node-conditional (3.2) uniquely specifies a joint distribution $P(\boldsymbol{\sigma}_t, \boldsymbol{\sigma}_{t+\delta t})$ under identified sufficient conditions. This theoretical analysis further complements our physics intuitions for feature extraction in Section 3.

5.1. Dynamical Ising via a Temporal Joint Graph

Here, we formulate the joint distribution of the coin toss process, i.e. the *path* probability $P(\boldsymbol{\sigma}_{t+\delta t}, \boldsymbol{\sigma}_t)$ between consecutive time steps t and $t+\delta t$, given by (3.2) as a graphical model factored according to a temporal joint graph $\mathcal{G}_{t,t+\delta t}$ described below.

Let $\mathcal{G}_{t,t+\delta t} = (\mathcal{V}_t, \mathcal{V}_{t+\delta t}, \mathcal{E})$ be the undirected temporal joint graph constituted by two subgraphs of equal cardinality, i.e. $|\mathcal{V}_t| = |\mathcal{V}_{t+\delta t}|$. The configuration of this model, $\mathcal{G}_{t,t+\delta t}$, is given as

$$\begin{aligned}\mathbf{X} &= (\boldsymbol{\sigma}_t, \boldsymbol{\sigma}_{t+\delta t}) \\ &= \left(\{\sigma_{i,t}\}_{i=1}^N, \{\sigma_{i,t+\delta t}\}_{i=1}^N\right) = \left(\{\sigma_q\}_{q=1}^{2N}\right), \end{aligned} \tag{5.1}$$

where $\boldsymbol{\sigma}_t$ and $\boldsymbol{\sigma}_{t+\delta t}$ are binary random vectors of length N. It it important to clarify that, we use two types of node indices interchangeably in this and subsequent sections: $\sigma_{i,\tau}$ (where $\tau = t$ or $t+\delta t$) and σ_q. We use the appropriate index depending on whether we are examining the sub-graph $\mathcal{G}_\tau$ or the joint graph $\mathcal{G}_{t,t+\delta t}$.

We write the temporal joint model $\mathcal{G}_{t,t+\delta t}$ to be a graphical model up to pairwise sufficient statistics and extend the sufficient statistics to include extra 1st- and 2nd- order derivatives with respect to time (at t). By Assumption 2.1, we have $\mathcal{E}_{t+\delta t} = \emptyset$ from pairwise Markov property. Moreover, we have $\mathcal{E}_t \neq \emptyset$ and $\mathcal{E}_{t,t+\delta t} \neq \emptyset$. For simplicity, we adopt shorthand notation $\widetilde{\Phi}(\sigma_q)$ (later $\widetilde{\phi}(\sigma_q)$) for derivative-extended sufficient statistics throughout this paper. Furthermore, we require that, for all $\mathcal{G}_{t,t+\delta t}$, the time derivatives are always evaluated at t, not $t+\delta t$. Explicitly, the univariate and quadratic sufficient statistics are

$$\widetilde{\Phi}(\sigma_q) = \Phi(\sigma_q), \quad q \in \mathcal{V}_{t+\delta t}, \tag{5.2}$$

and

$$\widetilde{\Phi}\left(\sigma_q, \sigma_{q'}\right) \tag{5.3}$$
$$= \begin{cases} \Phi\left(\sigma_q, \sigma_{q'}, \frac{\delta\sigma_q}{\delta t}, \frac{\delta\sigma_{q'}}{\delta t}, \frac{\delta}{\delta t}\left(\frac{\delta\sigma_q}{\delta t}\right), \frac{\delta}{\delta t}\left(\frac{\delta\sigma_{q'}}{\delta t}\right)\right), & q, q' \in \mathcal{V}_t \\ \Phi\left(\sigma_q, \sigma_{q'}, \frac{\delta\sigma_q}{\delta t}, \frac{\delta}{\delta t}\left(\frac{\delta\sigma_q}{\delta t}\right)\right), & q \in \mathcal{V}_t, q' \in \mathcal{V}_{t+\delta t}, \\ \Phi\left(\sigma_q, \sigma_{q'}\right), & q, q' \in \mathcal{V}_{t+\delta t}. \end{cases}$$

Higher-order sufficient statistics can be easily generalized. Consequently, the pairwise graphical model based on the graph $\mathcal{G}_{t,t+\delta t}$ over $\mathbf{X}$ takes the form

$$P(\boldsymbol{\sigma}_t, \boldsymbol{\sigma}_{t+\delta t}) \propto \exp\left\{w\widetilde{\Phi}(\boldsymbol{\sigma}_t, \boldsymbol{\sigma}_{t+\delta t})\right\} \tag{5.4}$$
$$\propto \exp\left\{\sum_{q\in\mathcal{V}} w_q \widetilde{\Phi}(\sigma_q) + \sum_{(q,q')\in\mathcal{E}} w_{qq'}\widetilde{\Phi}(\sigma_q, x_{q'})\right\}.$$

In the following, we aim to construct this joint distribution from its node-conditionals where the node-conditionals are specified by a univariate exponential family.

Following (Yang et al., 2015), we assume the node-conditional distributions of this graphical model $\mathcal{G}_{t,t+\delta t}$

$$P\left(\sigma_q|\mathbf{X}\right) \tag{5.5}$$
$$= \exp\left\{\Psi_q\left(\mathbf{X}_{\mathcal{V}\setminus q}\right)\widetilde{\phi}(\sigma_q) + B(\sigma_q) - \overline{D}\left(\mathbf{X}_{\mathcal{V}\setminus q}\right)\right\},$$

follow a univariate exponential family for all $q \in \mathcal{V}$, where $\widetilde{\phi}(\sigma_q)$ is the derivative-extended *univariate* sufficient statistics function for random variable σ_q. Note that, (5.5) includes both $\sigma_q \in \mathcal{V}_t$ and $\sigma_q \in \mathcal{V}_{t+\delta t}$ cases, where (3.2) only corresponds to the latter. For completeness, we further assume that, for nodes $q \in \mathcal{V}_t$ (or equivalently $\{i,t\} \in \mathcal{V}_t$), the node-conditional follows the standard Glauber dynamics (2.2) with the effective local field $\overline{\Gamma}_{q,t}$ (or equivalently $\overline{\Gamma}_{i,t}$):

$$P\left(\sigma_q|\mathbf{X}\right) = P\left(\sigma_{i,t}|\mathbf{X}_{\setminus\{i,t\}}\right) = \frac{\sigma_q \cdot \overline{\Gamma}_{i,t}}{2\cosh\overline{\Gamma}_{i,t}}, \tag{5.6}$$

$\forall q \notin \mathcal{V}_{t+\delta t}$. Here, $\overline{\Gamma}_{i,t} := \sum_{p\in\mathcal{V}_{\setminus\{i,t\}}} \overline{J}_{\{i,t\},p}(t)\sigma_{p,t} + h_i(t)$ with $\overline{J}(t) \in \mathbb{R}^{2N\times 2N}$ being zero-padded $J \in \mathbb{R}^{N\times N}$ and $h \in \mathbb{R}^N$, where (J, h) are the parameters of an Ising model factored according to subgraph $\mathcal{G}_t$.

Next, we provide the theorem for the existence of a unique joint distribution whose node-conditionals are specified by the Spin-Gas Glauber dynamics (3.2) and (5.6).

Theorem 5.1 (Joint Distribution of Spin-Gas Glauber Dynamics (3.2))**.** Suppose the joint graph $\mathcal{G}_{t,t+\delta t}$ consists of cliques at most size k. The augmented Glauber dynamics (3.2), with gas-like interaction g, uniquely determines a joint distribution that factors according to $\mathcal{G}_{t,t+\delta t}$, if and only if,

I: the node-conditionals are given by

$$P\left(\sigma_q|\mathbf{X}\right) := \begin{cases} \frac{\sigma_q\cdot\Gamma_{q,t}}{2\cosh\Gamma_{q,t}}, & \forall q \in \mathcal{V}_{t+\delta t}, \\ \frac{\sigma_q\cdot\overline{\Gamma}_{q,t}}{2\cosh\overline{\Gamma}_{q,t}}, & \forall q \in \mathcal{V}_t, \end{cases} \tag{5.7}$$

where $\Gamma_{q,t}$ is given by (3.2) and $\overline{\Gamma}_{q,t}$ is given by (5.6).

II: for $i \in \mathcal{V}_t$, g_i takes the form

$$g_i\left(\frac{\delta\boldsymbol{\sigma}_t}{\delta t}, \frac{\delta}{\delta t}\left(\frac{\delta\boldsymbol{\sigma}_t}{\delta t}\right)\right)$$
$$:= \sum_{j\in\mathcal{V}_t} G^{(1)}_{ij}(t)\frac{\delta\sigma_{j,t}}{\delta t} + \sum_{j\in\mathcal{V}_t} G^{(2)}_{ij}(t)\frac{\delta}{\delta t}\left(\frac{\delta\sigma_{j,t}}{\delta t}\right)$$
$$+ \sum_{m,n\in\mathcal{V}_t} G^{(1)}_{imn}(t)\frac{\delta\sigma_{m,t}}{\delta t}\frac{\delta\sigma_{n,t}}{\delta t} \tag{5.8}$$
$$+ \sum_{m,n\in\mathcal{V}_t} G^{(2)}_{imn}(t)\frac{\delta}{\delta t}\left(\frac{\delta\sigma_{m,t}}{\delta t}\right)\frac{\delta}{\delta t}\left(\frac{\delta\sigma_{n,t}}{\delta t}\right) + \cdots,$$

which is up to $(k-1)$-th order product with coupling constants $\{G\}$ for 1st- and 2nd-order time derivatives and their cross terms. For $i \in \mathcal{V}_{t+\delta t}$, $g_i = 0$.

Proof. A detailed proof is shown in the Appendix C.1. □

Here, we state the theorem in the general form where $\mathcal{G}_{t,t+\delta t}$ consists up to k-cliques, and then, by setting $k = 2$, it reduces to pairwise joint distribution (5.4) of our interest.

Corollary 5.1. *The corresponding joint distribution in Theorem 5.1 is in the form*

$$P(\boldsymbol{X}) =$$
$$\exp\left\{\sum_{q\in\mathcal{V}} \widetilde{w}_q\widetilde{\phi}(\sigma_q) + \sum_{q\in\mathcal{V}}\sum_{p\in\mathcal{N}(q)} \widetilde{w}_{qp}\widetilde{\phi}(\sigma_q)\widetilde{\phi}(\sigma_p) + \cdots\right.$$
$$\left. + \sum_{q\in\mathcal{V}}\sum_{p_1,\cdots,p_k\in\mathcal{N}(q)} \widetilde{w}_{q,p_1,\cdots,p_k}\widetilde{\phi}(\sigma_q)\prod_{j=2}^{k}\widetilde{\phi}(\sigma_{p_k}) - A(\{\widetilde{w}\})\right\},$$

with derivative-extended sufficient statistics $\widetilde{\phi}(\sigma_q) := \left(\sigma_{\sigma_q}, \frac{\delta\sigma_{\sigma_q}}{\delta t}, \frac{\delta}{\delta t}\left(\frac{\delta\sigma_{\sigma_q}}{\delta t}\right)\right)$ *being a 3-vector for* $q \in \mathcal{V}_t$, $\widetilde{\phi}(\sigma_q) = \sigma_q$ *being a zero-padded 3-vector, for* $q \in \mathcal{V}_{t+\delta t}$ *and* $\{\widetilde{w}\}$ *being contants 1-1 corresponding to* $\{G\}$.

Corollary 5.2. *For* $k = 2$*, we have pairwise model*

$$P(\boldsymbol{X}) =$$
$$\exp\left\{\sum_{q\in\mathcal{V}} h_q(t)\sigma_q + \sum_{q\in\mathcal{V}}\sum_{p\in\mathcal{N}(q)} \widetilde{w}_{qp}\widetilde{\phi}(\sigma_q)\widetilde{\phi}(\sigma_p) - A(\{\widetilde{w}\})\right\},$$

where, for $q \in \mathcal{V}_t$, $\frac{1}{2}\sum_{p\in\mathcal{N}(q)} \widetilde{w}_{qp}\widetilde{\phi}(\sigma_p) = \sum_{p\in\mathcal{N}(q)} J_{qp}(t)\sigma_{p,t} + \sum_{p\in\mathcal{N}(q)} G^{(1)}_{qp}(t)\frac{\delta\sigma_{p,t}}{\delta t} + \sum_{p\in\mathcal{N}(q)} G^{(2)}_{qp}(t)\frac{\delta}{\delta t}\left(\frac{\delta\sigma_{p,t}}{\delta t}\right)$ *and, for* $q \in \mathcal{V}_{t+\delta t}$, $\sum_{p\in\mathcal{N}(q)} \widetilde{w}_p\widetilde{\phi}(\sigma_p) = \sum_{p\in\mathcal{N}(q)} \overline{J}_{qp}(t)\sigma_{p,t}$.

Proof. Corollary 5.1 and Corollary 5.2 are direct consequences of Theorem 5.1. □

Following are two conclusions of our theoretical study that provide practical guidance for our methodology.

Table 1. Comparison of Basic and Extended Feature Accuracy.

Metric	Dataset	MLP		CNN		LSTM	
		Basic	Extended	Basic	Extended	Basic	Extended
R^2 Score %	Synthetic	97.71 $\pm$ 0.00	**99.18** $\pm$ **0.00**	97.72 $\pm$ 0.00	**99.16** $\pm$ **0.00**	97.71 $\pm$ 0.03	**99.01** $\pm$ **0.04**
	Taxi	73.21 $\pm$ 0.00	**77.50** $\pm$ **0.01**	73.17 $\pm$ 0.00	**79.02** $\pm$ **0.06**	73.19 $\pm$ 0.01	**76.62** $\pm$ **0.04**
	Electricity	97.47 $\pm$ 0.00	**98.97** $\pm$ **0.00**	97.47 $\pm$ 0.00	**99.09** $\pm$ **0.01**	94.83 $\pm$ 0.00	**95.43** $\pm$ **0.00**
	Traffic	75.04 $\pm$ 0.00	**87.41** $\pm$ **0.00**	75.04 $\pm$ 0.00	**86.45** $\pm$ **0.01**	74.66 $\pm$ 0.00	**83.12** $\pm$ **0.00**
Pearson Correlation %	Synthetic	98.86 $\pm$ 0.00	**99.60** $\pm$ **0.00**	98.86 $\pm$ 0.00	**99.62** $\pm$ **0.00**	98.86 $\pm$ 0.01	**99.56** $\pm$ **0.01**
	Taxi	85.59 $\pm$ 0.00	**88.76** $\pm$ **0.00**	85.57 $\pm$ 0.00	**88.95** $\pm$ **0.03**	85.58 $\pm$ 0.01	**88.46** $\pm$ **0.00**
	Electricity	98.73 $\pm$ 0.00	**99.49** $\pm$ **0.00**	98.73 $\pm$ 0.00	**99.54** $\pm$ **0.00**	**98.04** $\pm$ **0.00**	97.70 $\pm$ 0.00
	Traffic	86.64 $\pm$ 0.00	**93.51** $\pm$ **0.00**	86.64 $\pm$ 0.00	**93.02** $\pm$ **0.00**	86.42 $\pm$ 0.00	**91.40** $\pm$ **0.00**

(i) Although Theorem 5.1 does not delve into the analysis of statistical estimation for such a model, it does offer methodological guidance that mirrors the neighborhood-estimation[5] approach used for pairwise graphical models in (Yang et al., 2015). This suggests that the ground-truth random process can be uniquely represented by derivative-extended sufficient statistics. This insight is crucial for feature engineering, especially in cases where the target model is unknown.

(ii) In the same vein, Corollary 5.2, in conjunction with (Yang et al., 2015), proposes an amenable form of g to enhance learning and inference. This aligns with our physics intuitions from Section 3, which regard $\{\boldsymbol{p}, \boldsymbol{a}\}$ as fundamental characteristics for feature extraction.

6. Experimental Studies

In this section, we demonstrate the validity of our feature generation method by testing it on a synthetic dataset and three real-world datasets, with three distinct neural network architectures (MLP, CNN, LSTM) representing three aspects of deep learning architecture design. To evaluate the generated features, we use a predictive model, F, that only takes in univariate $\mathbf{x}_{i,t}$ as input and predicts the one-step-ahead target $\mathbf{y}_{i,t+1}$, i.e. $F(\mathbf{x}_{i,t}) = \widehat{\mathbf{y}}_{i,t+1}, \forall i, t$. By doing this, the model does not consider the correlations between univariate series or the autocorrelation beyond a single time step. This means that the advantages of using autoregressive and graphical (cross-sectional or convolutional) architectures are limited. To effectively capture important time series characteristics, we need input features that can summarize all information from the basic/raw features across both time (autoregressive) and univariates (correlations) for model training. Additional experiments including more deep learning models for both one-step-ahead and multi-horizon prediction tasks can be found in Appendix E.

[5]Estimate the neighborhood of each node separately, and then stitch then together to form the global graph estimation.

6.1. Experimental Setup

Data. The data utilized in our experiments consists of a synthetic dataset constructed to adhere to the assumptions of our method, as well as an electricity dataset, a traffic dataset, and a taxi dataset. Each of these datasets is partitioned in an 80/20 ratio to derive our training data (known as in-sample data) and testing data (referred to as out-of-sample data). Further details regarding the datasets can be found in Appendix D.1.

Feature Programming. We set the feature template to *default*[6] (without additional hand-crafted features). We set the order-to-order generation rule to pass *all* extended features from the preceding order along with the basic series list of the current order to the programmed module. For each dataset, we generate the 0th order, 1st order, and 2nd order features. These features are then combined and treated as extended features. We trained three deep learning models, MLP, CNN, and LSTM, using input features consisting of (i) only the basic features; and (ii) a concatenation of basic features and extended features.

Benchmark and Evaluation Metrics. We train each of the models described above using (i) as the benchmark. Namely, we compare the performance of models trained on the basic features vs. models trained on both the basic and extended features. For our evaluation metrics, we use out-of-sample R^2-score and Pearson correlation between the predicted and true value (one-step-ahead). For each dataset and all models, we repeat the multivariate time series prediction task 10 times with basic and extended features. We defer implementation details to Appendix D.3.

6.2. Evaluation of Accuracy

In Table 1, our results demonstrate that the proposed methods can improve model accuracy significantly with high

[6]It is important to notice that one major advantage of our feature programming framework is that it is hyperparameter-free. The generated features are solely controlled by user-designed feature program (feature template and operation module).

quality features. Comparing across all datasets (except for electricity-LSTM), models that were trained incorporating both basic and extended features consistently outperformed those trained solely on basic features. The introduction of extended features led to an augmentation in both the R^2 Score and the Pearson correlation. The magnitude of this improvement ranged between 1.3% and 5.85%, depending on the specific dataset and model utilized. Notably, even with the most challenging dataset (the **taxi** dataset), the inclusion of extended features facilitated an improvement of approximately 4.3/5.9/3.4% in the R^2 score for MLP/CNN/LSTM, and around 2.3/3.4/3.0% in the Pearson correlations for MLP/CNN/LSTM. The results are summarized in Table 1.

6.3. Computational Time Comparison

As seen in Table 2, the feature programming framework demonstrates high efficiency. The computational time of feature generation is negligible compared to the time spent on training the downstream models, even when considering the MLP model, which is the simplest and fastest-to-train prediction model in our experiments.

Table 2. Computational Time Comparison for Feature Generation and Model Training. This table presents the computational time required for generating extended features based on the default setting described in Section 6.1, alongside the time required for downstream model training. We report the computational time of simplest (and fastest) model in our experiments, i.e. MLP. The feature programming framework demonstrates high efficiency, with the feature generation time being negligible in comparison to the time spent on training the downstream models.

	Taxi	Electric	Traffic	MLP-Synthetic	MLP-Taxi	MLP-Electric	MLP-Traffic
Time (s)	24.34	82.71	218.23	4272.36	5461.75	16833.59	23883.94

6.4. Additional Experiments

Further experiments on the quality of extended features using a variety of common time series prediction models and ablation studies can be found in Appendix E, including

- **Appendix E.1:** Six commonly-used time series forecasting models (Transformer (Vaswani et al., 2017), XGBM (Chen et al., 2015), LightGBM (Ke et al., 2017), TFT (Lim et al., 2021), TCN (Chen et al., 2020), NBEATS (Oreshkin et al., 2019)), with an emphasis on standard one-step-ahead predictions. In addition, we also incorporate multi-horizon prediction task experiments to showcase practical applicability.
- **Appendix E.2:** Ablation studies that utilize three illustrative examples to demonstrate the customization process that may be necessary in real-world scenarios. These examples illustrate how to adjust the feature program to resemble known hand-crafted features, corresponding to practical situations where we have some understanding of the types of features that might be particularly useful for a specific task, domain, or application. This approach allows for a clear understanding of how to customize the feature program and how the operators influence the generated features. We evaluate feature quality using the same metrics across different feature programming settings.

Through our experiments, the performance enhancements provided by our extended features are illustrated. In particular, in the multi-horizon prediction tasks (Table 5), our results show that when predicting the next 20 values using a sequence of length 20 (arguably the most challenging task when relying solely on basic features), the features generated by our approach have exhibited substantial improvements (on average 88+% in R^2 and 27+% in Pearson correlation metrics) in prediction accuracy across all models. These experiments showcase the adaptability, flexibility, and feature generation capabilities of our method.

7. Conclusion

We propose a programmable automated feature engineering approach for multivariate time series modeling, called feature programming, inspired by a novel Ising-like dynamical model. Theoretically, our model-based approach draws practical guidance from constructing multivariate graphical models using univariate exponential family, aligning with insights from the physics model. Empirically, the generated features effectively improve noisy multivariate time series prediction in various settings. Yet, there is a noticeable limitation: the flexibility of our method comes with the trade-off of not including any feature selection or pruning mechanism beyond user-specific programs. For future investigations, we plan to integrate feature selection mechanisms into feature programming and explore the joint temporal graph in the context of graphical diffusion (Kondor & Lafferty, 2002).

Acknowledgments

We would like to acknowledge the contributions of Chenwei Xu and Weijian Li for the additional experiments section (Appendix E) during the review and rebuttal periods. Due to conference restrictions, their authorship is not recognized in this version. Hence, for accurate authorship and citation, we kindly direct the readers to the `arXiv` version.

JH would like to thank to Donglin Yang and Andrew Chen for enlightening discussions, and Jiayi Wang for invaluable support in facilitating experimental deployments. The authors would like to thank the anonymous reviewers and program chairs for constructive comments. JH is partially supported by the Walter P. Murphy Fellowship. HL is partially supported by NIH R01LM1372201, NSF CAREER1841569 and a NSF TRIPODS1740735. The content is solely the responsibility of the authors and does not necessarily represent the official views of the funding agencies.

References

Balkin, S. D. and Ord, J. K. Automatic neural network modeling for univariate time series. *International Journal of Forecasting*, 16(4):509–515, 2000.

Barandas, M., Folgado, D., Fernandes, L., Santos, S., Abreu, M., Bota, P., Liu, H., Schultz, T., and Gamboa, H. Tsfel: Time series feature extraction library. *SoftwareX*, 11: 100456, 2020.

Bengio, Y., Courville, A., and Vincent, P. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8): 1798–1828, 2013.

Besag, J. Spatial interaction and the statistical analysis of lattice systems. *Journal of the Royal Statistical Society: Series B (Methodological)*, 36(2):192–225, 1974.

Bousmalis, K., Trigeorgis, G., Silberman, N., Krishnan, D., and Erhan, D. Domain separation networks. *Advances in neural information processing systems*, 29, 2016.

Cerqueira, V., Moniz, N., and Soares, C. Vest: Automatic feature engineering for forecasting. *Machine Learning*, pp. 1–23, 2021.

Chandrashekar, G. and Sahin, F. A survey on feature selection methods. *Computers & Electrical Engineering*, 40 (1):16–28, 2014.

Chen, T., He, T., Benesty, M., Khotilovich, V., Tang, Y., Cho, H., Chen, K., Mitchell, R., Cano, I., Zhou, T., et al. Xgboost: extreme gradient boosting. *R package version 0.4-2*, 1(4):1–4, 2015.

Chen, Y., Kang, Y., Chen, Y., and Wang, Z. Probabilistic forecasting with temporal convolutional neural network. *Neurocomputing*, 399:491–501, 2020.

Chepurko, N., Marcus, R., Zgraggen, E., Fernandez, R. C., Kraska, T., and Karger, D. Arda: Automatic relational data augmentation for machine learning. *Proceedings of the VLDB Endowment*, 13(9).

Christ, M., Kempa-Liehr, A. W., and Feindt, M. Distributed and parallel time series feature extraction for industrial big data applications. *arXiv preprint arXiv:1610.07717*, 2016.

Christ, M., Braun, N., Neuffer, J., and Kempa-Liehr, A. W. Time series feature extraction on basis of scalable hypothesis tests (tsfresh–a python package). *Neurocomputing*, 307:72–77, 2018.

Clifford, P. Markov random fields in statistics. *Disorder in physical systems: A volume in honour of John M. Hammersley*, pp. 19–32, 1990.

De Brabandere, A., Op De Beéck, T., Hendrickx, K., Meert, W., and Davis, J. Tsfuse: Automated feature construction for multiple time series data. *Machine Learning*, pp. 1–56, 2022.

Doersch, C., Gupta, A., and Efros, A. A. Unsupervised visual representation learning by context prediction. In *Proceedings of the International Conference on Computer Vision*, pp. 1422–1430, 2015.

Fawaz, H. I., Forestier, G., Weber, J., Idoumghar, L., and Muller, P.-A. Deep learning for time series classification: a review. *Data mining and knowledge discovery*, 33(4): 917–963, 2019.

Feynman, R. P., Hibbs, A. R., and Styer, D. F. *Quantum mechanics and path integrals*. Courier Corporation, 2010.

Glauber, R. J. Time-dependent statistics of the ising model. *Journal of mathematical physics*, 4(2):294–307, 1963.

Glorot, X., Bordes, A., and Bengio, Y. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pp. 513–520, 2011.

Goldenfeld, N. *Lectures on phase transitions and the renormalization group*. Reading, MA: Addison-Wesley, 1992.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. *Advances in neural information processing systems*, pp. 2672–2680, 2014.

Graham, R. Path integral formulation of general diffusion processes. *Zeitschrift für Physik B Condensed Matter*, 26 (3):281–290, 1977.

Gu, S., Kelly, B., and Xiu, D. Empirical asset pricing via machine learning. *The Review of Financial Studies*, 33 (5):2223–2273, 2020.

Guyon, I. and Elisseeff, A. An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar):1157–1182, 2003.

Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A.-r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T. N., et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, 29(6):82–97, 2012.

Hinton, G. E., Sejnowski, T. J., et al. Learning and relearning in boltzmann machines. *Parallel distributed processing: Explorations in the microstructure of cognition*, 1 (282-317):2, 1986.

Iwana, B. K. and Uchida, S. An empirical survey of data augmentation for time series classification with neural networks. *Plos one*, 16(7):e0254841, 2021.

Janssen, H. On a lagrangean for classical field dynamics and renormalization group calculations of dynamical critical properties. *Zeitschrift für Physik B Condensed Matter*, 23 (2):377–380, 1976.

Jiang, Y., Luo, Q., Wei, Y., Abualigah, L., and Zhou, Y. An efficient binary gradient-based optimizer for feature selection. *Math. Biosci. Eng*, 18(4):3813–3854, 2021.

Kadanoff, L. P. *Statistical physics: Statics, dynamics and renormalization*. World Scientific, 2000.

Kakushadze, Z. 101 formulaic alphas. *Wilmott*, 2016(84): 72–81, 2016.

Kaul, A., Maheshwary, S., and Pudi, V. Autolearn—automated feature generation and selection. In *2017 IEEE International Conference on data mining (ICDM)*, pp. 217–226. IEEE, 2017.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., and Liu, T.-Y. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30, 2017.

Khurana, U., Turaga, D., Samulowitz, H., and Parthasrathy, S. Cognito: Automated feature engineering for supervised learning. In *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*, pp. 1304–1307. IEEE, 2016.

Kondor, R. I. and Lafferty, J. Diffusion kernels on graphs and other discrete structures. In *Proceedings of the 19th international conference on machine learning*, volume 2002, pp. 315–322, 2002.

Längkvist, M., Karlsson, L., and Loutfi, A. A review of unsupervised feature learning and deep learning for time-series modeling. *Pattern Recognition Letters*, 42:11–24, 2014.

Lau, A. W. and Lubensky, T. C. State-dependent diffusion: Thermodynamic consistency and its path integral formulation. *Physical Review E*, 76(1):011123, 2007.

Le, Q. V. and Mikolov, T. Distributed representations of sentences and documents. In *International Conference on Machine Learning*, pp. 1188–1196, 2014.

Lim, B., Arık, S. Ö., Loeff, N., and Pfister, T. Temporal fusion transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting*, 37(4):1748–1764, 2021.

Masini, R. P., Medeiros, M. C., and Mendes, E. F. Machine learning advances for time series forecasting. *Journal of economic surveys*, 37(1):76–111, 2023.

Meisenbacher, S., Turowski, M., Phipps, K., Rätz, M., Müller, D., Hagenmeyer, V., and Mikut, R. Review of automated time series forecasting pipelines. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 12(6):e1475, 2022.

Mezard, M. and Montanari, A. *Information, physics, and computation*. Oxford University Press, 2009.

Miikkulainen, R., Liang, J., Meyerson, E., Rawal, A., Fink, D., Francon, O., Raju, B., Shahrzad, H., Navruzyan, A., Duffy, N., et al. Evolving deep neural networks. In *Artificial intelligence in the age of neural networks and brain computing*, pp. 293–312. Elsevier, 2019.

Montanari, A. and Sen, S. A short tutorial on mean-field spin glass techniques for non-physicists. *arXiv preprint arXiv:2204.02909*, 2022.

Muralidhar, N., Islam, M. R., Marwah, M., Karpatne, A., and Ramakrishnan, N. Incorporating prior domain knowledge into deep neural networks. In *2018 IEEE international conference on big data (big data)*, pp. 36–45. IEEE, 2018.

Ng, N., Cho, K., and Ghassemi, M. Ssmba: Self-supervised manifold based data augmentation for improving out-of-domain robustness. *arXiv preprint arXiv:2009.10195*, 2020.

Nguyen, H. C., Zecchina, R., and Berg, J. Inverse statistical problems: from the inverse ising problem to data science. *Advances in Physics*, 66(3):197–261, 2017.

Oreshkin, B. N., Carpov, D., Chapados, N., and Bengio, Y. N-beats: Neural basis expansion analysis for interpretable time series forecasting. *arXiv preprint arXiv:1905.10437*, 2019.

Ozaki, T. *Time series modeling of neuroscience data*. CRC press, 2012.

Pan, S. J. and Yang, Q. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10): 1345–1359, 2009.

Pan, S. J., Ni, X., Sun, J.-T., Yang, Q., and Chen, Z. Cross-domain sentiment classification via spectral feature alignment. In *Proceedings of the 19th international conference on World wide web*, pp. 751–760, 2010.

Pan, Z., Sharma, A., Hu, J. Y.-C., Liu, Z., Li, A., Liu, H., Huang, M., and Geng, T. T. Ising-traffic: Using ising machine learning to predict traffic congestion under

uncertainty. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2023.

Raissi, M., Perdikaris, P., and Karniadakis, G. E. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

Sezer, O. B., Gudelek, M. U., and Ozbayoglu, A. M. Financial time series forecasting with deep learning: A systematic literature review: 2005–2019. *Applied Soft Computing*, 90:106181, 2020.

Sun, Y., Li, J., Liu, J., Chow, C., Sun, B., and Wang, R. Using causal discovery for feature selection in multivariate numerical time series. *Machine Learning*, 101(1): 377–395, 2015.

Swendsen, R. and Wang, J. Nonuniversal critical dynamics in monte carlo simulations. *Physical Review Letters*, 58 (2):86–88, 1987.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. Rethinking the inception architecture for computer vision. In *Proceedings of the International Conference on Computer Vision*, pp. 2818–2826, 2016.

Tallec, C. and Ollivier, Y. Can recurrent neural networks warp time? *arXiv preprint arXiv:1804.11188*, 2018.

Tealab, A. Time series forecasting using artificial neural networks methodologies: A systematic review. *Future Computing and Informatics Journal*, 3(2):334–340, 2018.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Vázquez, E. D., Del Ferraro, G., and Ricci-Tersenghi, F. A simple analytical description of the non-stationary dynamics in ising spin systems. *Journal of Statistical Mechanics: Theory and Experiment*, 2017(3):033303, 2017.

Weber, M. F. and Frey, E. Master equations and the theory of stochastic path integrals. *Reports on Progress in Physics*, 80(4):046601, 2017.

Weiss, K., Khoshgoftaar, T. M., and Wang, D. A survey of transfer learning. *Journal of Big data*, 3(1):1–40, 2016.

Wen, Q., Sun, L., Yang, F., Song, X., Gao, J., Wang, X., and Xu, H. Time series data augmentation for deep learning: A survey. *arXiv preprint arXiv:2002.12478*, 2020.

Wissel, C. Manifolds of equivalent path integral solutions of the fokker-planck equation. *Zeitschrift für Physik B Condensed Matter*, 35(2):185–191, 1979.

Yang, E., Ravikumar, P., Allen, G. I., and Liu, Z. Graphical models via univariate exponential family distributions. *The Journal of Machine Learning Research*, 16(1):3813–3847, 2015.

Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. How transferable are features in deep neural networks? *International Conference on Machine Learning*, pp. 3320–3328, 2014.

Zhou, Z., Ma, L., and Liu, H. Trade the event: Corporate events detection for news-based event-driven trading. *arXiv preprint arXiv:2105.12825*, 2021.

Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H., and He, Q. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1):43–76, 2020.

Appendix

A More Related Works

B Theoretical Background: Constructing Multivariate Graphical Models from Univariate Exponential Family

C Proof of Main Text

D Experimental Details of Main Text

E Additional Experiments

A. More Related Works

Feature engineering is the process of constructing and selecting features for use in machine learning models. Compared to data augmentation (Ng et al., 2020; Goodfellow et al., 2014; Doersch et al., 2015; Szegedy et al., 2016), which improves the generalization performance of the model by synthesizing data that is more similar to the test distribution (Bousmalis et al., 2016), feature engineering is an important step in the modeling process focusing on extracting relevant and informative features from the raw data that can be used to train a model, as the quality of the features can have a significant impact on the performance of the model. Recent developments in feature engineering for machine learning include the use of automated techniques for feature selection and construction (Kaul et al., 2017; Khurana et al., 2016; Guyon & Elisseeff, 2003), the use of deep learning techniques for feature learning (Bengio et al., 2013; Le & Mikolov, 2014; Hinton et al., 2012), and the use of transfer learning techniques to adapt pre-trained feature representations to new tasks (Pan & Yang, 2009; Yosinski et al., 2014).

Among all these developments, automated feature engineering, the use of algorithms and computational tools to automatically construct and select features for use in machine learning models (including the use of genetic algorithms (Miikkulainen et al., 2019) to evolve effective feature sets, the use of gradient-based optimization (Jiang et al., 2021) to search for optimal feature combinations, and the use of deep learning techniques (Bengio et al., 2013; Le & Mikolov, 2014) for feature learning), is the primary interest of this paper — as it allows for the handling of large and complex datasets, the discovery of complex feature interactions, and the rapid testing and comparison of different feature sets.

On the other hand, the field of automated feature engineering for time series modeling has also witnessed growing interest in recent times. These methods usually function by extracting a significant number of predefined features from the data, followed by selecting a subset of the most pertinent ones, including univariate (Christ et al., 2018), multivariate time series (De Brabandere et al., 2022) and many others mentioned in Section 1.

However, existing approaches may face limitations due to the issues highlighted in Section 1, such as the majority of features being either too domain-specific or not easily generated based on a fundamental principle. As a complement to these methods, our proposed programmable time series feature engineering framework provide a united perspective from a dynamical physics model, the spin-gas Glauber dynamics. To the best of our knowledge, our framework represents the first theoretically grounded automated feature engineering method for time series, characterized by its comprehensiveness and physics-motivated fundamental principles. Contrary to hand-crafted methods that typically rely on ad-hoc dictionaries and may struggle to transfer between different contexts, feature programming provides significant flexibility and adaptability across various tasks, domains, or applications through appropriate customization.

B. Theoretical Background: Constructing Multivariate Graphical Models from Univariate Exponential Family

Here, we introduce multivariate graphical model construction using univariate exponential family distributions (Yang et al., 2015), which is used in Section 5 to pinpoint conditions enhancing the modeling capability of our proposed physics model in capturing complex time series.

Suppose $\mathbf{X} = (x_1, \ldots, x_N)$ is a random vector of length N, with each random variable x_i taking values in some set $\mathfrak{X}$. Let $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ be an undirected graph over the set of nodes $\mathcal{V} := [N]$ corresponding to N variables $\{x_i\}_{i=1}^N$. Let $\mathcal{C}$ be the set of cliques (fully-connected subgraphs) of $\mathcal{G}$, and $\{\Phi_c(\mathbf{X}_c)\}_{c\in\mathcal{C}}$ be the set of clique-wise sufficient statistics with $\mathbf{X}_c$ denoting the variables within $\mathcal{C}$. Any distribution over $\mathbf{X}$ within the graphical model family represented by $\mathcal{G}$ has the

exponential family form

$$P(\mathbf{X}) \propto \exp\left\{\sum_{c\in\mathcal{C}} w_c \Phi_c(\mathbf{X}_c)\right\}, \tag{B.1}$$

where $\{w_c\}_{c\in\mathcal{C}}$ are the weights over $\{\Phi_c\}_{c\in\mathcal{C}}$. For $\mathcal{G}$ being only up to pairwise cliques and $\mathfrak{X}$ being binary, (B.1) reduces to the Ising model family $P(\mathbf{X}) \propto \exp\left\{\sum_{i\in\mathcal{V}} w_i\Phi(x_i) + \sum_{(j,k)\in\mathcal{E}} w_{jk}\Phi(x_j, x_k)\right\}$.

Following (Yang et al., 2015), the graphical model (B.1) over multivariate $\mathbf{X}$ can be constructed from a particular choice of univaraite exponential family for each random variable x_i under conditions identified by Lemma B.1 & B.2 below.

Definition B.1 (Univariate Exponential Family)**.** A univariate exponential family is a family of distributions for random variable x,

$$P(x) = \exp\{w\phi(x) + B(x_i) - D(x)\}, \tag{B.2}$$

with w, $\phi(x_i)$, $B(x_i)$, $D(w)$ being the canonical exponential family parameter, *univariate* sufficient statistics, base measure and log-partition function, respectively. The choice of w and $\phi(x_i)$ may vary depending on the particular distribution within the exponential family.

Denoting $\mathcal{N}(i)$ the set of neighbors of node i according to $\mathcal{G}$, we construct the node-conditionals based on above univariate exponential family distribution (B.2) by considering a potential function consisting of a linear combination of up to k-th order products of univariate sufficient statistics $\{\phi(x_j)\}_{j\in\mathcal{N}(i)}$ and $\phi(x_i)$:

$$P\left(x_i|\mathbf{X}_{\mathcal{V}\setminus i}\right) = \exp\left\{\Psi_i\left(\mathbf{X}_{\mathcal{V}\setminus i}\right)\cdot\phi(x_i) + B(x_i) - \overline{D}\left(\mathbf{X}_{\mathcal{V}\setminus i}\right)\right\}, \tag{B.3}$$

where $B(x_i)$ is specified by the univariate exponential family (B.2), $\overline{D}\left(\mathbf{X}_{\mathcal{V}\setminus i}\right) := D\left(\Psi_i\left(\mathbf{X}_{\mathcal{V}\setminus i}\right)\right)$ denotes the conditional log-partition function, and the canonical parameter $\Psi_i\left(\mathbf{X}_{\mathcal{V}\setminus i}\right)$ of the univariate sufficient statistics function $\phi(x_i)$ is given by the following tensor-factorized form

$$\Psi_i\left(\mathbf{X}_{\mathcal{V}\setminus i}\right) = w_i + \sum_{\alpha_2\in\mathcal{N}(i)} w_{i\alpha_2}\phi(x_{\alpha_2}) + \sum_{\alpha_2,\alpha_3\in\mathcal{N}(i)} w_{i\alpha_2\alpha_3}\phi(x_{\alpha_2})\phi(x_{\alpha_3}) + \cdots + \sum_{\alpha_2,\ldots,\alpha_k\in\mathcal{N}(i)} w_{i\alpha_2\ldots\alpha_k}\prod_{j=2}^{k}\phi(x_{\alpha_j}). \tag{B.4}$$

Remark B.1. We remark that (B.4) linearly expands the canonical parameter $\Psi_i\left(\mathbf{X}_{\mathcal{V}\setminus i}\right)$ in terms of products of univaraite sufficient statistics of conditional variables $\{x_{\alpha_l}\}_{l\in\mathcal{V}\setminus i}$ up to $(k-1)$-th order; and the node-conditional (B.3) is a univariate exponential family.

By Hammersley-Clifford theorem (Clifford, 1990) and tenor factorization (Yang et al., 2015; Besag, 1974), (B.3) can be shown to determine the joint distribution $P(\mathbf{X})$ uniquely:

Lemma B.1 (Proposition 1 of (Yang et al., 2015))**.** Given a graph $\mathcal{G}$. Suppose $\mathbf{X} = (x_1, \ldots, x_N)$ is a random vector of size N and its node-conditionals for each node is specified by a univariate exponential family (B.3) that factors according to $\mathcal{G}$. Then its joint distribution sits inside the graphical model family, presented by $\mathcal{G}$, of the form

$$P(\mathbf{X}) = \exp\left\{\sum_{i\in\mathcal{V}} w_i\phi(x_i) + \sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}(i)} w_{ij}\phi(x_i)\phi(x_j) + \cdots + \sum_{i\in\mathcal{V}}\cdots\sum_{\alpha_k\in\mathcal{N}(i)} w_{i\alpha_2\ldots\alpha_k}\phi(x_i)\prod_{l=2}^{k}\phi(\alpha_l) + \sum_{i\in\mathcal{V}} B(x_i) - A(\{w\})\right\}, \tag{B.5}$$

where $A(\{w\})$ is the log-partition function of the joint distribution.

Lemma B.1 states that, if the node-conditional distributions are defined as a univariate exponential family given in B.3, a unique graphical model (joint) distribution is specified by B.5. Interestingly, it can be also shown that, (B.3) and (B.5) are the most general form for pairwise graphical model by construction (Yang et al., 2015).

Reversely, we have the following lemma.

Lemma B.2 (Theorem 2 of (Yang et al., 2015))**.** Suppose $\mathbf{X} = (x_1, \ldots, x_N)$ is a random vector of size N, whose node-conditionals are governed by the exponential family

$$P\left(x_i|\mathbf{X}_{\mathcal{V}\setminus i}\right) = \exp\left\{\Theta\left(\mathbf{X}_{\mathcal{V}\setminus i}\right)\phi(x_i) + B(x_i) - \overline{D}\left(\mathbf{X}_{\mathcal{V}\setminus i}\right)\right\}, \tag{B.6}$$

with canonical parameter function $\Theta\left(\mathbf{X}_{\mathcal{V}\setminus i}\right)$, if the corresponding joint distribution $P(\mathbf{X})$ is a graphical model that factors according to some graph $\mathcal{G}$ with clique-size at most k, then the conditionals (B.6) have the tensor-factorized form given by (B.3) (i.e. $\Theta\left(\mathbf{X}_{\mathcal{V}\setminus i}\right) = \Psi\left(\mathbf{X}_{\mathcal{V}\setminus i}\right)$) , and $P(\mathbf{X})$ takes the form of (B.5).

Lemma B.1 and Lemma B.2 together specify the sufficient conditions for node-conditionals and joint distribution with respect to $\mathcal{G}$ being given by (B.6) and (B.5), respectively: (i) node-conditionals are in a univariate exponential family, and (ii) the joint distribution is a graphical model factored according to $\mathcal{G}$ with clique-size at most k. This construction allows for several advantages, including the capability to model a wide variety of distributions, and closed-form multivariate densities that enable efficient inference and learning with statistical guarantees, see (Yang et al., 2015) for details.

C. Proof of Main Text

Our primary objective in the theory section is to demonstrate that the spin-gas Glauber dynamics introduced in (3.2) (or (5.7) for completeness) can be specified by a univariate exponential family and can be seen as the node-conditional of a multivariate graphical model $\mathcal{G}$, which exhibits desirable properties according to (Yang et al., 2015).

We sketch the three conceptual steps here.

- **Step 1:** We demonstrate that the node-conditional (5.7) is indeed specified by a univariate exponential family, and we identify the sufficient statistics and canonical parameters.
- **Step 2:** We show that the identified canonical parameters of (5.7) (and thus the spin-gas Glauber dynamics (3.2)) take on a tensor-factorized form if the effective local field g is in the form of (5.8).
- **Step 3:** Finally, by applying Lemma B.1 and Lemma B.2 with the tensor-factorized canonical parameters, we complete the proof, illustrating that our model inherits advantageous properties from (Yang et al., 2015) under the identified conditions (5.8).

C.1. Proof of Theorem 5.1

Proof. Recall $\mathcal{G} = (\mathcal{V}_t, \mathcal{V}_{t+\delta t}, \mathcal{E})$, and $\mathbf{X} = (\boldsymbol{\sigma}_t, \boldsymbol{\sigma}_{t+\delta t})$. We first consider the node-conditionals given by the augmented Glauber dynamics, i.e. node-conditionals for nodes $q \in \mathcal{V}_{t+\delta t}$. Starting from expressing the augmented Glauber dynamics (3.2) into node-conditional form with canonical parameter $\Theta\left(\mathbf{X}_{\setminus(i,t+\delta t)}\right)$, by assumption 2.1 (and consequently $\mathcal{E}_{t+\delta t} = \emptyset$, $\mathcal{E}_t \neq \emptyset$ and $\mathcal{E}_{t,t+\delta t} \neq \emptyset$), we have (3.2) taking node-conditional form

$$P(\sigma_{i,t+\delta t}|\boldsymbol{\sigma}_t) = P(\sigma_{i,t+\delta t}|\boldsymbol{\sigma}_{t+\delta t\setminus i}, \boldsymbol{\sigma}_t), \tag{C.1}$$

and therefore

$$\begin{aligned}\frac{\exp\{\sigma_{i,t+\delta t}\Gamma_{i,t}\}}{2\cosh\Gamma_{i,t}} &= \exp\left\{\left(\ln\frac{1}{2}\cdot\Gamma_{i,t}\right)\sigma_{i,t+\delta t} - \ln\left[\frac{1}{2}\left(e^{\Gamma_{i,t}} + e^{-\Gamma_{i,t}} + 1\right)\right]\right\} \\ &= \exp\left\{\Theta\left(\mathbf{X}_{\setminus(i,t+\delta t)}\right)\widetilde{\phi}\left(\sigma_{i,t+\delta t}\right) + B\left(\sigma_{i,t+\delta t}\right) - \overline{D}\left(\boldsymbol{\sigma}_t\right)\right\},\end{aligned}$$

where the last line takes the form of (B.6). Here, the i index specifies a node in $\mathcal{V}_{t+\delta t}$, and the conditional potential function can be decomposed into two parts: canonical parameter $\Theta\left(\mathbf{X}_{\setminus(i,t+\delta t)}\right) = \ln\frac{1}{2}\cdot\Gamma_{i,t}$ and sufficient statistics $\widetilde{\phi}\left(\sigma_{t+\delta t}\right) = \sigma_{i,t+\delta t}$.

From $\Theta(\mathbf{X}_{\setminus(i,t+\delta t)}) = \ln\frac{1}{2}\cdot\Gamma_{i,t}$ and $\widetilde{\phi}(\sigma_{i,t+\delta t}) = \sigma_{i,t+\delta t}$, we first observe that

$$\Theta(\mathbf{X}_{\setminus(i,t+\delta t)}) = \Theta(\mathbf{X}_{t+\delta t\setminus i}, \boldsymbol{\sigma}_t) = \Theta_i(\boldsymbol{\sigma}_t), \tag{C.2}$$

where, we emphasize that, $\Theta_i(\boldsymbol{\sigma}_t)$ denotes the canonical parameter of the node-conditional of the i-th node in $\mathcal{V}_{t+\delta t}$, and is a function of $\boldsymbol{\sigma}_t$ due to the definition of $\Gamma_{i,t}$.

For the sake of simplicity, we abuse notation by absorbing the $\ln \frac{1}{2}$ factor into $\Gamma_{i,t}$, and write

$$\Theta_i(\boldsymbol{\sigma}_t) = \Gamma_{i,t} = \sum_{j\in\mathcal{V}_t} J_{ij}(t)\sigma_{j,t} + h_i(t) + g_i\left(\frac{\delta\boldsymbol{\sigma}_t}{\delta t}, \frac{\delta}{\delta t}\left(\frac{\delta\boldsymbol{\sigma}_t}{\delta t}\right)\right), \tag{C.3}$$

with $g_i(\cdot)$ satisfying (5.8).

Recall the fact that the univariate sufficient statistics functions $\{\widetilde{\phi}(\sigma_{\alpha_l})\}_{\alpha_l\in\mathcal{N}(i)}$ are now functions of 0th-, 1st- and 2nd-order time derivatives, we expand the $\Psi_i\left(\mathbf{X}_{\setminus(i,t+\delta t)}\right)$ using its definition

$$\Psi_i\left(\mathbf{X}_t\right) = w_i + \sum_{\alpha_2\in\mathcal{N}(i)} \widetilde{w}_{i\alpha_2}\widetilde{\phi}(\sigma_{\alpha_2}) + \sum_{\alpha_2,\alpha_3\in\mathcal{N}(i)} \widetilde{w}_{i\alpha_2\alpha_3}\widetilde{\phi}(\sigma_{\alpha_2})\widetilde{\phi}(\sigma_{\alpha_3}) + \cdots + \sum_{\alpha_2,\ldots,\alpha_k\in\mathcal{N}(i)} \widetilde{w}_{i\alpha_2\ldots\alpha_k}\prod_{j=2}^{k}\widetilde{\phi}(\sigma_{\alpha_j}). \tag{C.4}$$

We immediately read out $w_i = h_i(t)$, and univariate sufficient statistics for node $\alpha_l \in \mathcal{N}(i) \subseteq \mathcal{V}_t$:

$$\widetilde{\phi}(\sigma_{\alpha_l}) \coloneqq \widetilde{\sigma}_{\alpha_l}^{\mathsf{T}} = \left(\sigma_{\alpha_l}, \frac{\delta\sigma_{\alpha_l}}{\delta t}, \frac{\delta}{\delta t}\left(\frac{\delta\sigma_{\alpha_l}}{\delta t}\right)\right)^{\mathsf{T}}, \tag{C.5}$$

with dimension of $\widetilde{w}$ being extended along each index (except i) by a factor of 3, such that, for instance,

$$\begin{aligned}\sum_{\alpha_2\in\mathcal{N}(i)} \widetilde{w}_{i\alpha_2}\widetilde{\phi}(\sigma_{\alpha_2}) &= \sum_{\alpha_2\in\mathcal{N}(i)}\left[w_{i\alpha_2}\sigma_{\alpha_2} + w'_{i\alpha_2}\frac{\delta\sigma_{\alpha_2}}{\delta t} + w''_{i\alpha_2}\frac{\delta}{\delta t}\left(\frac{\delta\sigma_{\alpha_2}}{\delta t}\right)\right] \\ &= \sum_{j\in\mathcal{V}_t} J_{ij}(t)\sigma_{j,t} + \sum_{j\in\mathcal{V}_t} G^{(1)}_{ij}(t)\frac{\delta\sigma_{j,t}}{\delta t} + \sum_{j\in\mathcal{V}_t} G^{(2)}_{ij}(t)\frac{\delta}{\delta t}\left(\frac{\delta\sigma_{j,t}}{\delta t}\right).\end{aligned} \tag{C.6}$$

Higher-order product terms can be identified accordingly. Therefore, we arrive the fact that the canonical parameter $\Theta_i(\boldsymbol{\sigma}_t)$ with $g_i(\cdot)$ given by (5.8) is indeed in a tensor-factorized form $\Psi_i(\boldsymbol{\sigma}_t)$. Moreover, the coefficients $\{\widetilde{w}\}$ can all be identified with couplings $\{G\}$. We can therefore write down the extended sufficient statistics as a $(3N+1N)$-dimensional vector

$$\widetilde{\phi}(\mathbf{X}) = \Bigg(\underbrace{\left\{\sigma_{i,t}, \frac{\delta\sigma_{i,t}}{\delta t}, \frac{\delta}{\delta t}\left(\frac{\delta\sigma_{i,t}}{\delta t}\right)\right\}_{i=1}^{N}}_{3N}, \underbrace{\{\sigma_{i,t+\delta t}\}_{i=1}^{N}}_{N}\Bigg)^{\mathsf{T}}$$

and compactify the notation with index $\alpha_l \coloneqq (i,t) \in \mathcal{V}_t$ and $\beta_l \coloneqq (i, t+\delta t) \in \mathcal{V}_{t+\delta t}$,

$$\widetilde{\phi}(\mathbf{X}) = \left(\{\widetilde{\sigma}_{\alpha_l}\}_{l=1}^{N}, \{\sigma_{\beta_l}\}_{l=1}^{N}\right)^{\mathsf{T}}, \tag{C.7}$$

where, $\widetilde{\phi}(\sigma_{\alpha_l}) = \widetilde{\sigma}_{\alpha_l} \coloneqq \left(\sigma_{\alpha_l}, \frac{\delta\sigma_{\alpha_l}}{\delta t}, \frac{\delta}{\delta t}\left(\frac{\delta\sigma_{\alpha_l}}{\delta t}\right)\right)$ for $\alpha_l \in \mathcal{V}_t$, and $\widetilde{\phi}(\sigma_{\alpha_l}) = \sigma_{\beta_l}$ $\beta_l \in \mathcal{V}_{t+\delta t}$, are zero-padded vectors.

Consequently, we can determine the canonical parameter $\Theta_{i,t}$:

$$\begin{aligned}\Psi_i\left(\mathbf{X}_t\right) &= \Psi_i\left(\mathbf{X}_{t+\delta t\setminus i}, \boldsymbol{\sigma}_t\right) = \Gamma_{i,t}\left(\boldsymbol{\sigma}_t\right) \\ &= \sum_{j\in\mathcal{V}_t} J_{ij}(t)\sigma_{j,t} + h_i(t) + g_i\left(\frac{\delta\boldsymbol{\sigma}_t}{\delta t}, \frac{\delta}{\delta t}\left(\frac{\delta\boldsymbol{\sigma}_t}{\delta t}\right)\right) \\ &= w_i + \sum_{\alpha_2\in\mathcal{N}(i)} w_{i\alpha_2}\widetilde{\phi}(\sigma_{\alpha_2}) + \sum_{\alpha_2,\alpha_3\in\mathcal{N}(i)} w_{i\alpha_2\alpha_3}\widetilde{\phi}(\sigma_{\alpha_2})\widetilde{\phi}(\sigma_{\alpha_3}) + \cdots + \sum_{\alpha_2,\ldots,\alpha_k\in\mathcal{N}(i)} w_{i\alpha_2\ldots\alpha_k}\prod_{j=2}^{k}\widetilde{\phi}(\sigma_{\alpha_j}),\end{aligned} \tag{C.8}$$

where $\alpha_l \in \mathcal{N}(i) \subseteq \mathcal{V}_t$ and univariate functions $\{\widetilde{\phi}(\sigma_{\alpha_l})\}_{\alpha_l\in\mathcal{N}(i)}$ are functions of 0th-, 1st- and 2nd-order time derivatives.

Finally, we observe that, since the node-conditionals of nodes $q \notin \mathcal{V}_{t+\delta t}$ follow the standard Glauber dynamics and $G_{t,t+\delta t}$ consists cliques of at most size k, the node-conditionals given by (5.7) and (5.6) satisfy the conditions of Lemma B.2 and hence (5.7) indulges the tensor-factorized form (5.8). Therefore, we complete the proof by applying Lemma B.1 and determining joint distribution $P(\mathbf{X})$ of $\mathcal{G}_{t,t+\delta t}$ uniquely. □

D. Experimental Details of Main Text

D.1. Datasets

We exploit four supervised regression datasets to test the proposed framework.

- **Synthetic Dataset:** To validate our method, we employ a synthetic dataset designed to align with the method's underlying assumptions. We derive this dataset from the Taxi Dataset, replacing the original input features with a mixture of zero-order, first-order, and second-order features, essentially omitting the basic features. For the output values, we opt for a randomly selected subset of zero-order features, such as volatility or exponential smoothing, derived from the foundational basic features, forecasting one step ahead for all input series.
- **Taxi Dataset:** We use the TLC Trip Record Dataset, which has the number of taxi rides for 1000 locations in the form of 30 minute time intervals. We use the current 30-minute interval to forecast the next 30-minute interval.
- **Electricity Dataset:** We use the UCI Electricity Load Diagrams Dataset, which has the electricity consumption of 370 customers in the form of hourly time intervals. We use the current hour to forecast the next hour.
- **Traffic Dataset:** We use the UCI PEM-SF Traffic Dataset holds the occupancy rate of 440 San Francisco Bay Area highways in the form of hourly time intervals. We use the current hour to forecast the next hour.

Extended Features. Let N be the number variate in the multivariate time series, K be the number of extended features and T be the sequence length. We use the default setting of the feature program to generate $K = 45$ features from each dataset (which has one feature for each variate). Namely, the data formats of the basic and extended features are $(N, 1, T)$ and (N, K, T), respectively.

To generate the extended features, we utilize all of the fundamental operators on the output of the previous order at each order's feature calculation. For the 0th order features, we applied smoothing using the window operator with lookback size of [7, 25]. For the 1st order features, we applied the window, difference, and shift operators with lookback size of [7, 25]. For the 2nd order features, we applied the window, difference, and shift operators with lookback size of [7, 25]. Our final extended features set is composed of the basic features and the all of the generated features at each order.

Noisiness. To see the noisiness of each dataset, we compute the Temporal Signal-to-Noise Ratio (TSNR) for all datasets.

Table 3. Assessment of Dataset Noisiness using Temporal Signal-to-Noise Ratio (TSNR) for each Dataset. The taxi dataset emerges as the noisiest, and our experimental findings highlight the increased significance of extended features in handling such noisy datasets.

	Synthetic	Taxi	Electric	Traffic
TSNR	1.60	1.27	2.98	1.32

D.2. Hyperparameter Search

Hyperparameter optimization is conducted via random search for 100 iterations.

- `learning_rate`: 0.01, 0.001, 0.0001, 0.00001
- `batch_size`: 64, 128, 256, 512, `feature_dim`
- `hidden_size`: 64, 128, 512, 1024, 2048
- `num_epochs`: we use early stopping.

D.3. Implementation Details

For each experimental setting, we set the batch size to equal the number of variables for the corresponding dataset.

- **Architectural Details:** For the MLP model, the architecture is composed of 3 layers, with a hidden size of 512, and ReLU activation function. For the CNN model, the architecture is composed of 1 fully connected layer with hidden size of 512 and ReLU activation function, and 2 convolutional layers with kernel size of 3. For the LSTM model, the architecture is composed of 2 hidden layers and a hidden size of 512.

- **Training Details:** We use an Adam optimizer with learning rate `lr` $= 10^{-5}$ for training. The coefficients of Adam optimizer, betas, are set to $(0.9, 0.999)$.
- **Platforms:** The GPUs and CPUs used to conduct experiments are NVIDIA GEFORCE RTX 2080 Ti and INTEL XEON SILVER 4214 @ 2.20GHz.

E. Additional Experiments

To complement the experiments presented in Section 6, we carry out further investigations that include more advanced models, practical scenarios, illustrative examples of customizing the feature program, and ablation studies to demonstrate the impact of operators on the generated features.

E.1. Evaluating Feature Programming Across Diverse Deep Learning Models

We conduct further experiments using a variety of widely-used time series forecasting models, with an emphasis on standard one-step-ahead predictions. In addition, we also incorporate multi-horizon prediction task experiments to address practical scenarios.

Models.

- **XGBoost** (Chen et al., 2015)
- **LightGBM** (Ke et al., 2017)
- **Transformer** (Vaswani et al., 2017)
- **Temporal Fusion Transformer (TFT)** (Lim et al., 2021)
- **Temporal Convolution Network (TCN)** (Chen et al., 2020)
- **N-BEATS** (Oreshkin et al., 2019)

We utilize the `DART` [7] package to implement XGBoost, LightGBM, Transformer, TFT, TCN, and N-BEATS.

Data. We employ the easiest dataset and the most challenging dataset identified in Section 6 and specified in Appendix D.1 — the synthetic and taxi datasets.

Experiment Settings. For each of the datasets, we do an 80/20 train/test split to get our training data (in-sample) and testing data (out-of-sample). For all experiments, we conduct 5 runs and report the average performance. For the models, we stick to the common default configurations used in literature so as for a fair comparison.

Problem Setting: One-Step-Ahead Prediction with Extended Features. We conduct one-step-ahead predictions using above models and assess their performance by comparing results obtained with and without the inclusion of extended features. We examine the standard one-step-ahead time series regression problem with a lookback size of $T = 20$, which corresponds to a sequence length of 20. As illustrated in Table 4, our extended features yield consistent performance enhancements in prediction tasks when used as model inputs for a range of machine learning (XGBoost and LightGBM) and deep neural network time series prediction models (Transformer, TCN, TFT and N-BEATS).

Multi-Horizon Prediction with Extended Features. We further investigate multi-horizon prediction tasks with horizon sizes of 1, 2, 3, 5, 10 and 20, utilizing a lookback size of $T = 20$. The models applied for multi-horizon prediction include Transformer, TFT, TCN[8], N-BEATS. These experiments are conducted using the synthetic dataset, and reported in Table 5. The results showcase the effectiveness of the generated features in various multi-horizon prediction settings. Particularly, when forecasting the next 20 values using a sequence of length 20 (arguably the toughest task when only relying on basic

[7] https://unit8co.github.io/darts/

[8] The current version of `DART` package requires that the output length is strictly smaller than the input length. As a result, we are unable to report the 20-step horizon results for TCN. Instead, we have included the results for a 19-step horizon and labeled them with * in Table 5.

Table 4. Performance Comparison of Common Time Series Models with and without Extended Features on Synthetic and Taxi Datasets. The table demonstrates the performance enhancements consistently achieved in one-step-ahead prediction tasks using a lookback size of $T = 20$ when extended features are utilized as inputs for various machine learning (XGBoost and LightGBM) and deep neural network time series prediction models (Transformer, TCN, TFT and N-BEATS).

Metric	Dataset	XGBoost		LightGBM		Transformer		TCN		TFT		N-BEATS	
		Basic	Extended	Basic	Extended	Basic	Extended	Basic	Extended	Basic	Extended	Basic	Extended
R^2 Score %	Synthetic	99.12	**99.34**	98.92	**99.34**	96.46	**96.86**	98.87	**99.37**	96.21	**97.60**	99.26	**99.41**
	Taxi	77.34	**78.15**	77.05	**81.01**	72.64	**73.70**	75.36	**77.70**	54.58	**61.31**	76.82	**79.94**
Pearson Correlation %	Synthetic	99.56	**99.67**	99.46	**99.67**	**99.11**	99.08	99.44	**99.69**	98.41	**99.19**	97.63	**99.73**
	Taxi	87.96	**88.42**	87.80	**88.97**	85.54	**86.83**	86.87	**88.26**	76.65	**79.65**	87.84	**89.61**

features), the features generated by our method have demonstrated significant improvements in prediction performance across all models, suggesting that the extended features possess a greater amount of autoregressive information compared to basic features.

Table 5. Evaluation of Feature Quality for Multi-Horizon Prediction using Synthetic Dataset. This table presents the performance of the Transformer, TFT, TCN and N-BEATS models on multi-horizon prediction tasks with horizon sizes of 1, 2, 3, 5, 10 and 20, benchmarked on the synthetic dataset. The results showcase the effectiveness of the generated features in various multi-horizon prediction settings. Particularly, when predicting a full-length sequence (using a sequence of length 20 to predict the next 20 values), our generated features have demonstrated considerable prediction improvements across all models, suggesting that the extended features possess a greater amount of autoregressive information compared to basic features.

Model	Metric	1-Step Horizon		2-Step Horizon		3-Step Horizon		5-Step Horizon		10-Step Horizon		20-Step Horizon	
		Basic	Extended	Basic	Extended	Basic	Extended	Basic	Extended	Basic	Extended	Basic	Extended
Transformer	R^2 Score %	96.46	**96.86**	94.47	**96.58**	94.07	**96.92**	91.03	**92.78**	73.11	**81.17**	54.46	**73.33**
	Pearson Correlation %	**99.11**	99.08	98.02	**98.63**	97.11	**98.48**	95.42	**96.40**	85.76	**90.41**	74.31	**86.20**
TFT	R^2 Score %	96.21	**97.60**	95.81	**96.86**	93.99	**95.11**	84.76	**92.29**	60.87	**78.60**	35.35	**64.91**
	Pearson Correlation %	98.41	**99.19**	97.92	**98.57**	97.30	**98.02**	93.39	**96.11**	81.14	**88.75**	66.13	**80.85**
TCN	R^2 Score %	98.87	**99.37**	96.03	**97.98**	92.41	**95.75**	79.91	**88.32**	41.91	**66.18**	17.18*	**52.50***
	Pearson Correlation %	99.44	**99.69**	98.02	**99.01**	96.20	**97.86**	89.51	**94.11**	65.51	**81.41**	46.08*	**73.19***
N-BEATS	R^2 Score %	99.26	**99.41**	98.39	**98.87**	97.26	**98.00**	93.17	**95.69**	78.09	**86.97**	57.06	**74.65**
	Pearson Correlation %	97.63	**99.73**	99.20	**99.44**	98.62	**99.07**	96.62	**97.84**	88.90	**93.28**	75.80	**86.44**

Simple Models Using Extended Features Achieve Comparable Performance to SOTA Models. As demonstrated in Table 4 and Table 5, utilizing extended features as inputs for simple models (e.g. Transformer) allows them to reach performance levels comparable to state-of-the-art models (TCN, TFT, N-BEATS) that rely solely on raw features.

E.2. Customizing Feature Programs: Three Examples of Customized Feature Programs

Our framework forms the crux of customizable feature engineering for multivariate time series. It's highly adaptable and doesn't depend on the model's structure. The production of extended features depends solely on the custom design of the pre-programmed operation module and the feature template. This proves especially advantageous when deep learning models struggle to extract specific features or when users wish to ensure that a certain aspect of the data is considered by the model. In this section, we present three illustrative examples to showcase the customization procedure that may be required in real-world scenarios. These examples show how to tailor the feature program to resemble known hand-crafted features. This corresponds to common practical scenarios where we have some insights into the types of features that might be particularly informative for the specific task, domain, or application at hand. For demonstration purposes, we adopt a simplified setting that employs straightforward and commonly known hand-crafted features (Momentum `MoM`, Bias `Bias` and Absolute Energy `AbsEnergy`) as the targets features for resembling.

Momentum $\texttt{MoM}[t, \Delta\tau]$**.** As mentioned in (Gu et al., 2020), one of the simplest common hand-crafted feature for time series is the momentum feature, which refers to the rate of change in a time series over a specified period. It is a measure of the series recent performance, which is often used to identify trending time series that are expected to continue their movements in the same direction.

Let $\Delta\tau$ be the lookback size of the momentum feature. The momentum feature is defined as:

$$\texttt{MoM}[t, \Delta\tau] = \frac{x_t - x_{t-\Delta\tau}}{x_{t-\Delta\tau}}. \tag{E.1}$$

It calculates the percentage change in the time series values over the past $\Delta\tau$ time steps. A positive momentum value indicates that the time series value has increased during that period, while a negative value means the value has decreased.

To resemble $\texttt{MoM}[t, \Delta\tau]$ with the feature programming framework, we set the feature template and operation module as follows:

1. Set feature template:
   (a) Set both `0th basic feature`, and `2nd basic feature` to be empty.
   (b) Incorporate `ratio feature`, defined as $\texttt{ratio}[a, b] := a/b$, into `1st basic feature`.
2. Pre-program the operation module:
   - **0th order:**
     (a) Perform `Shift[`$x_t, \Delta\tau$`]` and obtain $\{x_{t-\Delta\tau}\}$.
     (b) Perform `Difference[`$x_t, x_{t-\Delta\tau}$`]` and obtain 1st-order series $\{x_t - x_{t-\Delta\tau}\}$.
   - **1th order**:
     (a) Pass $a = \{x_{t-\Delta\tau}\}$ and $b = \{x_t - x_{t-\Delta\tau}\}$ as `1st basic feature` (where $\texttt{ratio}[a, b]$ is in.)
     (b) The output from `1st basic feature` gives the resembled $\texttt{MoM}[t, \Delta\tau]$.

It is important to note that the computational trajectory presented here specifically focuses on the output feature that resembles $\texttt{MoM}[t, \Delta\tau]$. The final set of extended features may include other distinct extended features.

BIAS $\texttt{Bias}\,[t, \Delta\tau]$**.** The second example we considered is the bias feature, which indicated the current trend of the time series and the potential reversals. Let $\texttt{SMA}[t, \Delta\tau] := \sum_{i=t-\Delta\tau+1}^{t} x_i/\Delta\tau$ be the sample moving average of lookback size $\Delta\tau$. The bias feature is defined as

$$\texttt{Bias}\,[t, \Delta\tau] := \frac{x_t - \texttt{SMA}[t, \Delta\tau]}{\texttt{SMA}[t, \Delta\tau]}. \tag{E.2}$$

To resemble $\texttt{Bias}\,[t, \Delta\tau,]$, we set the feature template and operation module as follows:

1. Set feature template:
   (a) Incorporate `SMA feature`, defined as $\texttt{SMA}[t, \Delta\tau]$, into `0th basic feature`.
   (b) Incorporate `ratio feature` into `1st basic feature`.
   (c) Set `2nd basic feature` to be empty.
2. Pre-program the operation module:
   - **0th order:**
     (a) Perform `Difference[`$x_t, \texttt{SMA}[t, \Delta\tau]$`]` and obtain 1st-order series $\{x_t - \texttt{SMA}[t, \Delta\tau]\}$.
   - **1th order**:
     (a) Pass $a = \{x_t, \texttt{SMA}[t, \Delta\tau]\}$ and $b = \{\texttt{SMA}[t, \Delta\tau]\}$ as `1st basic feature` (where $\texttt{ratio}[a, b]$ is in.)
     (b) The output from `1st basic feature` gives the resembled $\texttt{Bais}[t, \Delta\tau]$.

Absolute Energy $\texttt{AbsEnergy}[t, \Delta\tau]$**.** Following (Christ et al., 2018), the third example we considered is the absolute energy feature which is often used in signal processing, particularly for identifying the energy content of a signal, and it can also be useful in time series analysis for understanding the overall strength or magnitude of the data. The absolute energy feature is defined as

$$\texttt{AbsEnergy}[t, \Delta\tau] := \sum_{i=t-\Delta\tau+1}^{t} x_i^2. \tag{E.3}$$

To resemble $\texttt{AbsEnergy}[t, \Delta\tau]$, we set the feature template and operation module as follows:

1. Set feature template:
   (a) Incorporate `square feature`, $\texttt{square}[a] \coloneqq a^2$, into `0th basic feature`.
   (b) Incorporate `sum feature`, $\texttt{sum}[a_t, \Delta\tau] \coloneqq \sum_{i=t-\Delta\tau+1}^{t} a_i$, into `1st basic feature`.
   (c) Set `2nd basic feature` to be empty.
2. Pre-program the operation module:
   - **0th order:**
     (a) Do nothing.
   - **1th order**:
     (a) Pass $a = \{x_t^2\}$ (from `0th basic feature`) as `1st basic feature` (where $\texttt{sum}[a_t, \Delta\tau]$ is in.)
     (b) The output from `1st basic feature` gives the resembled $\texttt{AbsEnergy}[t, \Delta\tau]$.

Evaluating the Resemblance of Hand-Crafted Features. From the results above, by construction, we can achieve an *exact* resemblance of the given hand-crafted features by appropriately customizing the feature program. Following the same procedure, more complex features can also be programmed and resembled. Generally, to evaluate the quality of the resembled features, we follow the feature generation flow (path in the computational graph) and identify the specific feature output that resembles the target hand-crafted features. In our case, we have $\texttt{MoM}[t, \Delta\tau]$, $\texttt{Bais}[t, \Delta\tau]$, and $\texttt{AbsEnergy}[t, \Delta\tau]$ all perfectly resembled, with both R^2 and Pearson correlation equal to 1 when comparing the resembled features to the hand-crafted features. For the comparison baselines, we use `MoM` from (Gu et al., 2020), `Bias` from the `PandasTA`[9] package, and `AbsEnergy` from (Christ et al., 2018).

We would like to stress that achieving such exact resemblance is not generally applicable in practice (as it requires prior knowledge of the features to use), but the customization process for the feature program is indeed general. It is always possible to program desired (hand-crafted) features as long as they can be represented as programmable functions.

[9] https://github.com/twopirllc/pandas-ta