Encoding Temporal Statistical-space Priors via Augmented Representation under Data Scarcity

Insu Choi[1,*], Woosung Koh[2,*], Gimin Kang[1], Yuntae Jang[2] and Woo Chang Kim[1,†]

[1]Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Republic of Korea
[2]Yonsei University, Seoul, Republic of Korea

Abstract

Woo Cahng KimModeling time series data is a fundamental challenge across various domains due to the intrinsic temporal dimension. Despite significant strides in time series forecasting, high noise-to-signal ratio, non-normality, non-stationarity, and lack of data continue challenging practitioners. To address these, we introduce a simple representation augmentation technique. Our augmented representation acts as a statistical-space prior encoded at each time step. Accordingly, we term our method **S**tatistical-**s**pace **A**ugmented **R**epresentation (**SSAR**). The underlying high-dimensional data-generating process inspires our representation augmentation. We rigorously examine the empirical generalization performance on two data sets with two downstream temporal learning algorithms. Our approach significantly beats all five up-to-date baselines. Furthermore, our approach's modular design facilitates easy adaptation to diverse settings. Lastly, we provide comprehensive theoretical insights throughout the paper to underpin our methodology with a clear and rigorous understanding.

Keywords

Augmented Representation, Spatio-temporal Learning, Information Theory, Time Series Forecasting

1. Introduction

Time series forecasting is crucial across multiple domains such as finance [1], meteorology [2], and manufacturing [3]. Simple time series that are less stochastic and dependent on a tractable number of variables exist. Research primarily targets time series with complex, high-dimensional dependencies. Often, the true set of causal factors, $\mathcal{X}$, is intractable— i.e., unknown or known but impractical to compute. On top of this, complex time series structures, $p(\boldsymbol{y} \in \mathcal{Y} | \boldsymbol{x} \in \mathcal{X})$, often exhibit non-stationarity— challenging modeling.

Initially, methods like the vector autoregressive (VAR) model [4, 5] dominated multivariate forecasting. Extensions like the vector error correction model (VECM) [6] addressed some limitations of VAR models, but assumptions like non-stationarity still posed challenges. Despite their widespread use, these statistical models have caveats, particularly vis-à-vis their underlying statistical property assumptions. Thus, any analysis using these models requires examination of these assumptions—especially the non-stationary assumption, potentially requiring transformations to the data.

In response, neural network-based sequential models have become popular in the past decade. Their main advantage is that a universal function approximator flexibly captures high-dimensional non-linear dependency structures [7]. The most widely tested and verified for time series forecasting are Recurrent Neural Network (RNN) [8] architectures—with flagship examples being Long Short-Term Memory (LSTM) [9] and Gated Recurrent Unit (GRU) [10]. Both LSTM and GRU are part of our baseline.

More recently, with the out-performance of attention mechanism-based models like transformers in other sequential tasks such as natural language processing (NLP) [11] and speech recognition [12], numerous transformer-based time series forecasting models have been developed. Some significant examples include the FEDformer [13], Autoformer [14], Informer [15], Pyraformer [16], and LogTrans [17]. Despite the research interest, a timely oral presentation at the AAAI conference [18] showed strong evidence that Linear, Normalization Linear (NLinear), and Decomposition Linear (DLinear) architectures significantly outperform the aforementioned transformer-based models. This study showed robust multi-variate out-performance across Traffic, Electricity Transformer Temperature (ETT), Electricity, Weather, Exchange Rate, and Influenza-like Illness (ILI) datasets. All three [18]'s models are included in our baseline.

Despite the success of neural-network-based approaches, we observed a lack of literature explicitly targeting the non-stationary and stochastic nature through a simple, theoretically elegant approach.

In response, our contribution to the literature is summarized as follows:

- Develop an easily reproducible augmented representation technique, **SSAR**, that targets modeling complex non-stationary time series
- Clear discussion of the theoretical need for augmenting the input space and why it works well against baselines
- Theoretical discussion of the method's inspiration— the data-generating process of high-dimensional time series structure
- To our knowledge, first to leverage (asymmetric) information-theoretic measures in modeling the statistical-space
- Out-sample improvement vis-à-vis performance and stability against up-to-date baselines: (i) LSTM, (ii) GRU, (iii) Linear, (iv) NLinear, (v) DLinear
- Out-sample empirical results tested on two data sets and two downstream temporal graph learning algorithms
- Present a theoretically unified view with related work, suggesting that **SSAR** implicitly smooths stochastic data

STRL'24: Third International Workshop on Spatio-Temporal Reasoning and Learning, 5 August 2024, Jeju, South Korea
*These authors contributed equally
† Corresponding author
jl.cheivly@kaist.ac.kr (I. Choi); reiss.koh@yonsei.ac.kr (W. Koh); kgm4752@kaist.ac.kr (G. Kang); jytfdsa1218@yonsei.ac.kr (Y. Jang); wkim@kaist.ac.kr (W. C. Kim)

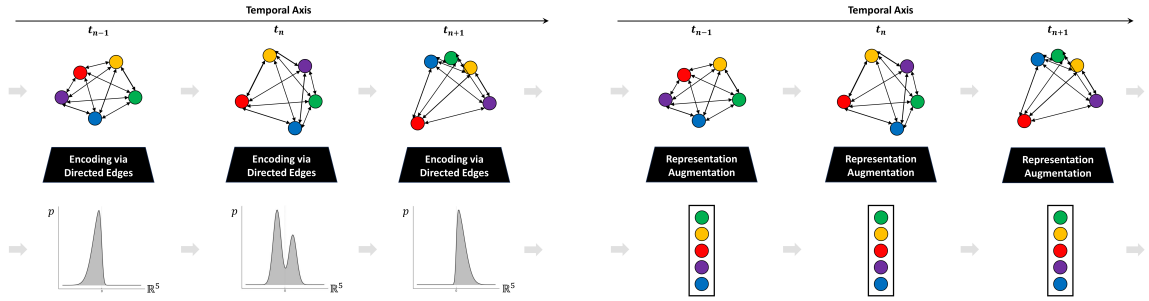

Figure 1: 5 variable example (left: theoretical view, right: representational view)

2. Related Works

Our work is related to temporal graph learning algorithms as our approach transforms a vector-based time series representation into a graph-based one. Then, a downstream graph learning algorithm is inducted to make predictions. Fundamentally, multi-layer perceptrons (MLPs) are incompatible with graph representations. However, graphs naturally represent various real-world phenomena [19]. E.g., social networks [20], chemical molecules [21], and traffic systems [22] inherently possess graphical structures. Graph Neural Networks (GNNs) bridge this gap, enabling learning directly from graphical structures. Contrary to works with predefined edge sets $\mathcal{E}$, we derive $\mathcal{E}$ from historical vertex values $\mathcal{V}$. The closest past work is [23], where they generate Pearson correlation-based $\mathcal{E}$ with $\mathcal{V}$. However, their $\mathcal{E}$ specifically proxies inter-company relations, tailored to their domain. Additionally, their $e \in \mathcal{E}$ are non-directed and symmetric. In contrast, our approach is (i) domain-agnostic, (ii) employs a simple representation augmentation to surpass state-of-the-art, (iii) modular with broad algorithm compatibility, (iv) incorporates directed asymmetric measures for $\mathcal{E}$, and (v) emphasizes theoretical analysis of the augmentation mechanism.

3. Preliminary: Complex Time Series

Modeling complex time series via neural networks presents three key challenges: (i) incomplete modeling, (ii) non-stationarity, and (iii) limited data-generating process access.

Let $p(\mathcal{Y}^t|\mathcal{X}^{t-M})$ be the true probability structure we want to learn. Here, $\mathcal{Y}^t$ is defined by the modeler as the variables of interest. Unlike $\mathcal{Y}^t$, $\mathcal{X}$ is intractable for complex problems as (i) it is too large to be computed realistically, but more pressingly (ii) it is unknown *a priori*. Therefore, we typically use heuristics or empirical evidence to identify $\hat{\mathcal{X}}$. Since we are forecasting, we use lagged values with M indicating the temporal magnitude of the most lagged value. Then, with a learner parameterized by θ, via maximum likelihood estimation we train for $\hat{p}_\theta(\mathcal{Y}^t|\hat{\mathcal{X}}^{t-M})$ where $\hat{p}_\theta(\mathcal{Y}^t|\hat{\mathcal{X}}^{t-M}) \approx p(\mathcal{Y}^t|\hat{\mathcal{X}}^{t-M})$. Often, due to $\mathcal{X}$'s intractability, in vector form, we set $\hat{\boldsymbol{x}}[,:] := \boldsymbol{y}[,:]$, using output-space's lagged values as input-space. We use this heuristic in our study and explain why this is a reasonable assumption in the Appendix. Since $\hat{\boldsymbol{x}}$ is a tractable approximation to the true input-space, we face the partial observation and incomplete modeling problem. This underlies much of the stochasticity and poor performance in forecasting high-dimensional structures. For domains that aggregate information on the global-level—like financial and climate time series, it is fair to assume that $|\mathcal{X}| \longrightarrow \infty$, dramatically raising the difficulty.

On top of this, we have a second, more pervasive challenge—non-stationarity. Non-stationarity is defined as $p^t(\mathcal{Y}|\mathcal{X}) \neq p^{t'}(\mathcal{Y}|\mathcal{X})$ where $t \neq t'$. The cause of non-stationarity could be from partial observability. Figure 1's left diagram summarizes this problem. Note that the distributions are 1-dimensional for a simplified visual depiction. This poses a significant challenge to neural-network-based approximators $\hat{p}_\theta(\mathcal{Y}|\mathcal{X})$ as MLPs—the building block—inherently work on stationary data sets.

The final challenge involves neural-network-based function approximators $F_\theta : \mathcal{X} \mapsto \mathcal{Y}$. The cost for a highly flexible function approximator F_θ is the large $|\theta|$. Consequently, as $|\theta|$ rises, the size of the data set $|\mathcal{D}|$ should also rise, allowing F_θ to generalize out-sample better. I.e., better approximate $p(\mathcal{Y}|\mathcal{X})$. Ideally, $\frac{\partial|D|}{\partial|\theta|} > 0$, but raising $|D|$ arbitrary is often intractable for complex time series. There are cases where reasonable simulators exist for the data-generating process $p(\mathcal{Y}|\mathcal{X})$, especially when $\mathcal{X}$ is tractable and the transition function is well approximated by rules. A representative example is physics simulators in the robotics field [24], where the simulator models the real-world, $p_{sim}(\mathcal{Y}|\mathcal{X}) \approx p(\mathcal{Y}|\mathcal{X})$. Correspondingly, we require a world simulator for complex time series with an intractably high-dimensional data-generating process. Since we have no world simulator, raising $|\mathcal{D}|$ requires time to pass. Therefore, we are restricted with a finite, lacking $\mathcal{D}$.

4. Methodology

4.1. Statistical-space Augmented Representation

In response to these three challenges, we apply our method, **SSAR**. We rigorously examine how **SSAR** overcomes each challenge in Section 4.3. A high-level overview of **SSAR** involves: (i) selecting a statistical measure, (ii) computing this measure $m(\boldsymbol{y}|\boldsymbol{x}, t, w_s)$ for each time t with sliding window w_s, (iii) generate a graph $\mathcal{G}^t$ where vertices $n \in \mathcal{V}^t$ represent variables at t, and weight of edges $w(e)$, where edges $e \in \mathcal{E}^t$, represent $m(\boldsymbol{y}|\boldsymbol{x}, t, w_s)$. Then, with spatiotemporal graph $\boldsymbol{\mathcal{G}} := \bigcup_t \mathcal{G}^t$, any temporal graph learning algorithm that makes temporal node prediction can be applied.

As seen in Figure 1, right, $\boldsymbol{SSAR} : \mathcal{D} \mapsto \boldsymbol{\mathcal{G}}$ where the original time series data $\mathcal{D}$ is in vector form $\boldsymbol{d} \in \mathcal{D}$. The per time

Algorithm 1 SSAR

Input: $\boldsymbol{D}_{raw}, m(\cdot), w_s$
Output: $\boldsymbol{\mathcal{G}} := \{\mathcal{G}_0, ..., \mathcal{G}_{T-w_s-1}\}$
Function ***SSAR***$(\boldsymbol{D}_{raw}, m(\cdot), w_s)$:

$\boldsymbol{D}_{processed} \longleftarrow DataProcess(\boldsymbol{D}_{raw})$
$\mathcal{F} \longleftarrow \boldsymbol{D}_{processed}.get\_feature\_set()$
$T \longleftarrow \boldsymbol{D}_{processed}.get\_total\_timesteps()$
$\boldsymbol{\mathcal{G}} \longleftarrow \{\}$

for $t \in T \backslash \{0, ..., w_s - 1\}$ **do**
  $\mathcal{G}_t \longleftarrow di\text{-}Graph()$
  for $\forall permutation \langle f_i, f_j \rangle \in \mathcal{F}$ **do**
    if $f_i \notin \mathcal{G}_t.nodes$ **then**
      $\mathcal{G}_t.add\_node(f_i)$
    end if
    if $f_j \notin \mathcal{G}_t.nodes$ **then**
      $\mathcal{G}_t.add\_node(f_j)$
    end if
    $weight \longleftarrow |m(\langle f_i, f_j \rangle, w_s, t)|$
    if weight $\neq 0$ **then**
      $\mathcal{G}_t.add\_edge(f_i \rightarrow f_j, weight)$
    end if
  end for
  $\boldsymbol{\mathcal{G}}.append(G_t)$
end for

return $\boldsymbol{\mathcal{G}}$
End Function

step functional view would be $\mathcal{G}^t \leftarrow \boldsymbol{SSAR}(\boldsymbol{d}^{[t-w_s:t]} \in \mathcal{D})$. Algorithm 1 details the pseudo-code for $\boldsymbol{SSAR}(\cdot)$. $\forall t$ $\boldsymbol{d}^t$ is transformed into a weighted, directed graph $\mathcal{G}^t = \langle \mathcal{V}, \mathcal{E}, \mathcal{W} \rangle$ where $\mathcal{V}$ is the set of nodes, $|\mathcal{V}| = N$, $\mathcal{E}$ is the set of directed edges, $|\mathcal{E}| = N^2 - N$, and the weighted adjacency matrix $\mathcal{W} \in \mathbb{R}^{N \times N}$. Here, each node $n \in \mathcal{V}$ represents a variable (scalar) in $\hat{\boldsymbol{x}}$ and $\boldsymbol{y}$. Each $e \in \mathcal{E}$ is a 2-tuple denoted $\langle n_i, n_j \rangle, i \neq j$, with each tuple corresponding to a permutation pair of nodes. $\mathcal{G}^t$'s $|\mathcal{E}| = N^2 - N$ as each permutation pair corresponds to a single directed edge, and nodes cannot direct to themselves. I.e., $\mathcal{G}^t$ is irreflexive. Given that the size of $\mathcal{W}$ is computed excluding the diagonal elements, where $w \in \mathcal{W}$, $w \geq 0 \in \mathbb{R}$, $\mathcal{W}$ is equivalent in size to $\mathcal{E}$ as each e_{ij} maps to a single w_{ij}. I.e., $W : \mathcal{E} \mapsto \mathcal{W}$. Here, $w^t(n_j \rightarrow n_i) \leftarrow m(n_i | n_j, t, w_s)$. An intuitive visualization is available in Figure 2.

4.2. Data-generating Process Meta-physics

SSAR is inspired by the meta-physics of the data-generating process of complex time series. The data-generating process refers to $p(\cdot)$. Access to $p(\cdot)$ allows for sampling data $D \sim p(\cdot)$ and approximating $\hat{p}_\theta(\cdot)$ through maximum likelihood estimation based on D. On a different note, this abstracted discussion aims to shed light on how a true $p(\cdot)$ is derived in the real world. I.e., it aims to hypothesize on the mechanisms underlying $p(\cdot)$, then describe how it inspires our approach.

Consider complex time series as described in the preliminary section.

Definition 4.1. Complex time series, causal in nature, is defined as $p(\mathcal{Y}^t | \mathcal{X}^{t-M})$ where $\mathcal{X}$ is intractable—i.e., $|\mathcal{X}| \longrightarrow \infty$.

Similar to the theoretical nature of $p(\cdot)$, the concept of

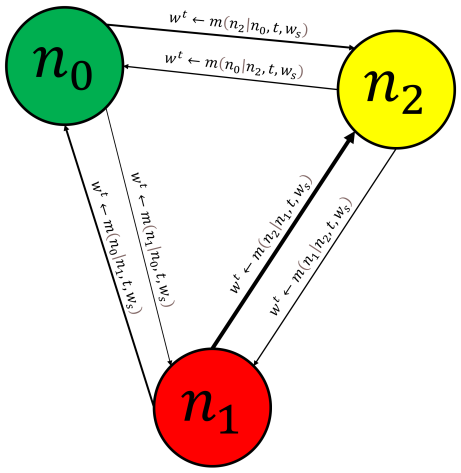

Figure 2: Sample $\mathcal{G}^t$, $N := 3$

$\mathcal{X}$ is also theoretical, given that its variables are human-defined. This implies that an arbitrary degree of granularity may describe $\mathcal{X}$. I.e., $|\mathcal{X}|$ can be arbitrarily raised larger until we reach the smallest units of the physical world. For instance, a high-level event like COVID-19 is an example of $x \in \mathcal{X}$, which can be further broken down into granular events like patient zero's contraction of the virus and so forth. Given $p(\boldsymbol{y}^t | \mathcal{X}^{t-M})$, consider $\mathcal{X}_{meas}, \mathcal{X}_{meas'} \subset \mathcal{X}$, where the former is digitally measured by humans in time series format and the latter comprises the remaining elements. $\mathcal{X}_{meas} \cup \mathcal{X}_{meas'} \equiv \mathcal{X}$ and $\mathcal{X}_{meas} \cap \mathcal{X}_{meas'} = \emptyset$. In the case of learning algorithms that require numerical input and output spaces, naturally, $\boldsymbol{y}^t \in \mathcal{Y} \subseteq \mathcal{X}_{meas}$ and $\hat{\boldsymbol{x}} \in \mathcal{X}_{meas}$. Define any information transfer within $\mathcal{X}_{meas}$ as endogenous and any within $\mathcal{X}_{meas'}$ as exogenous to the system. As not every real-world physical change is digitally tracked, each endogenous change has its roots in some exogenous change. With this backdrop, all numerical variables available to us digitally is a system that absorbs an arbitrary amount of exogenous shocks $\forall t$.

Let $x_{meas'} \in \mathcal{X}_{meas'}$, and $x_{meas} \in \mathcal{X}_{meas}$. Then, a simplified view of the data-generating process can be visualized in Figure 3. Each node at the top of the diagram represents $x_{meas'} \in \mathcal{X}_{meas'}$ while each node at the bottom represents $x_{meas} \in \mathcal{X}_{meas}$. Within the diagram, $|\mathcal{X}_{meas'}| \longrightarrow \infty$ is indicated via "...". Blue and purple edges show causal chains in the real physical world. Each dotted edge represents an exogenous shock to the endogenous system. Non-dotted green and red edges at each time step represent $p(\boldsymbol{y}^t_{meas} | \boldsymbol{x}^{t-1}_{meas})$. However, since $\exists p(\boldsymbol{x}^t_{meas} | \boldsymbol{x}^{t-1}_{meas'})$ which is unknown,

$$p(\boldsymbol{y}^t_{meas} | \boldsymbol{x}^{t-1}_{meas}) = p(\boldsymbol{y}^t_{meas} | p(\boldsymbol{x}^{t-1}_{meas} | \boldsymbol{x}^{t-2}_{meas'})). \quad (1)$$

Under this view, all complex time series are inherently non-stationary and, consequently, incompatible with models assuming stationarity. Consequently, for models that require stationary data, we require some tractable function $f(\cdot)$, $s.t.$,

$$f(\cdot) \approx p(\boldsymbol{x}^{t-1}_{meas} | \boldsymbol{x}^{t-2}_{meas'}). \quad (2)$$

The next section draws inspiration from the inherently directed graphical nature of the data-generating process, as illustrated in Figure 3, to theoretically unpack our method.

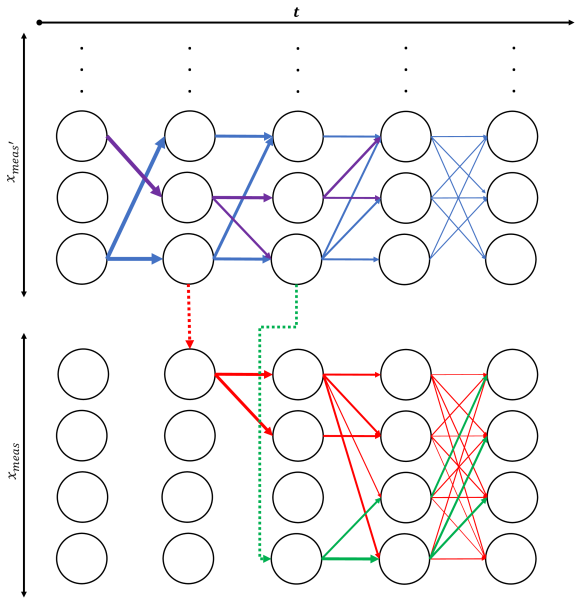

Figure 3: Real-world causal chains

4.3. Prior Encoding: Theoretical View

By the universal approximation theorem [25, 26], any stationary mapping can be approximated by neural networks. MLPs and their subsequent architectural innovations implicitly model high-dimensional statistical spaces.

$$\begin{aligned}\mathbf{O}_0 &:= \alpha(\mathbf{W}_0^T\mathbf{X} + \boldsymbol{b}_0),\\ \mathbf{O}_1 &:= \alpha(\mathbf{W}_1^T\mathbf{O}_0 + \boldsymbol{b}_1),\\ \mathbf{O}_2 &:= \sigma(\mathbf{W}_2^T\mathbf{O}_1 + \boldsymbol{b}_2),\end{aligned} \tag{3}$$

where $\theta = \{\bigcup \mathbf{W}, \bigcup \mathbf{b}\}$, $\mathbf{X}$ is the input tensor, and α, σ are non-linear activations. Given that neural networks are directed graphs, the explicit representation by **SSAR** (Figures 1 and 2) can be implicitly captured by (3). Despite this, we opt for an explicit representation encoded as a Bayesian prior $p(\theta)$. Under the Bayesian view of learning from data,

$$p(\theta|\mathcal{D}) := \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})}. \tag{4}$$

This inductive bias—if accurate, can be helpful for generalized performance when $|\mathcal{D}| \ll \infty$. As noted earlier, complex time series feature finite $\mathcal{D}$, making it challenging to increase its size.

Our prior encoding $\forall t$, as visualized in Figure 1, left, aids learning via overcoming non-stationarity. Since we are learning the distribution $\hat{p}_\theta(\mathcal{Y}^t|\hat{\mathcal{X}}^{t-M})$, to capture the non-stationarity, a natural approach would be to add a second parameter, a regime vector $\boldsymbol{r}$, resulting in learning $\hat{p}_\theta(\mathcal{Y}^t|\hat{\mathcal{X}}^{t-M}, \boldsymbol{r} \leftarrow r(\mathcal{Y}^t|\hat{\mathcal{X}}^{t-M}))$. This involves learning $r_{\theta_r}(\cdot)$. In this case,

$$\hat{p}_\theta(\mathcal{Y}^t|\hat{\mathcal{X}}^{t-M}, \boldsymbol{r} \leftarrow r_{\theta_r}(\mathcal{Y}^t|\hat{\mathcal{X}}^{t-M})), \tag{5}$$

$$\therefore \theta^{tot} := \theta \cup \theta_r, \Rightarrow |\theta^{tot}| > |\theta| \because |\theta_r| \neq \emptyset. \tag{6}$$

Given the small size of $\mathcal{D}$ relative to $\mathcal{X}$, increasing degrees of freedom without further sampling $\mathcal{D} \sim p(\cdot)$ is not ideal.

An ideal alternative is letting a statistical-space relationship at t proxy for $r(\cdot)$—i.e., $m(\boldsymbol{y}^t \in \mathcal{Y}^t|\hat{\boldsymbol{x}} \in \hat{\mathcal{X}}^{t-M}, t, w_s) \approx r(\boldsymbol{y}^t \in \mathcal{Y}^t|\hat{\boldsymbol{x}} \in \hat{\mathcal{X}}^{t-M})$. But, like $r(\boldsymbol{y}^t|\hat{\boldsymbol{x}})$, $m(\boldsymbol{y}^t|\hat{\boldsymbol{x}}, t, w_s)$ is unknown *a priori*. In this case, like $r_{\theta_r}(\cdot)$, we would require a learned approximation $m_{\theta_m}(\cdot)$, raising the size of aggregate parameters.

A reasonable and tractable approximation known *a priori* that does not raise the parameter count is,

$$m(\boldsymbol{y}^{t-1}|\hat{\boldsymbol{x}}, t-1, w_s) \approx m(\boldsymbol{y}^t|\hat{\boldsymbol{x}}, t, w_s) \approx r(\boldsymbol{y}^t|\hat{\boldsymbol{x}}). \tag{7}$$

Assuming sufficient granularity in time steps t, $m(\boldsymbol{y}^{t-1}|\hat{\boldsymbol{x}}, t-1, w_s)$ closely approximates $m(\boldsymbol{y}^t|\hat{\boldsymbol{x}}, t, w_s)$. We hypothesize that the trade-off between parameter count and approximation via $t-1$ is advantageous to the learning system.

Despite identifying a feasible regime-changing approximator, another problem remains. Representing and passing $m(\boldsymbol{y}^{t-1}|\hat{\boldsymbol{x}}, t-1, w_s)$ via Euclidean geometry significantly reduces the spatial information inherent to $m(\boldsymbol{y}^{t-1}|\hat{\boldsymbol{x}}, t-1, w_s)$. A natural representation is graphical, like Figure 3—therefore, we approximate (8) with (9) via (10), (11), and (12). This transformation, which augments the representation, theoretically encapsulates **SSAR**.

$$\hat{p}_\theta(\boldsymbol{y}^t|\hat{\boldsymbol{x}}, \boldsymbol{r} \leftarrow r(\boldsymbol{y}^t|\hat{\boldsymbol{x}})) \approx \tag{8}$$

$$\hat{p}_\theta(\boldsymbol{v}^t \in \mathcal{V}|\boldsymbol{v}^{t-M} \in \mathcal{V}, \boldsymbol{e}^{t-M} \in \mathcal{E}), \tag{9}$$

$$\boldsymbol{v}^t := \boldsymbol{y}^t, \tag{10}$$

$$\boldsymbol{v}^{t-M} := \hat{\boldsymbol{x}}, \tag{11}$$

$$\boldsymbol{e}^{t-M} \approx \hat{\boldsymbol{r}} \approx \boldsymbol{r},$$
$$\text{where } \boldsymbol{e}^{t-M} \leftarrow m(\boldsymbol{y}^{t-1}|\hat{\boldsymbol{x}}, t-1, w_s). \tag{12}$$

4.4. Statistical-space Measures

Six methods are used to compute $m(\boldsymbol{y}^{t-1}|\hat{\boldsymbol{x}}, t-1, w_s)$. The set of measures and corresponding abbreviation $\mathcal{M} :=$ {Pearson correlation: Pearson, Spearman rank correlation: Spearman, Kendall rank correlation: Kendall, Granger causality: GC, Mutual information: MI, Transfer entropy: TE}. This set can be divided into correlation-based $\mathcal{M}^{sym}$ and causal-based $\mathcal{M}^{asym}$ measures, which are symmetric and asymmetric, respectively. $\mathcal{M}^{sym} :=$ {Pearson, Spearman, Kendall} $\subset \mathcal{M}$, $\mathcal{M}^{asym} :=$ {GC, MI, TE} $\subset \mathcal{M}$, $\mathcal{M}^{sym} \cup \mathcal{M}^{asym} \equiv \mathcal{M}$, $\mathcal{M}^{sym} \cap \mathcal{M}^{asym} = \emptyset$. Symmetric measure refer to $m(n_j|n_i) = m(n_i|n_j)\ \forall \langle i, j \rangle \in \mathcal{E}, i \neq j$. Asymmetric refers to the case where $m(n_j|n_i) \neq m(n_i|n_j)$. The asymmetric case is most appropriate for our use case, as it uses only lagged values, making them a proxy for causal effects. Embedding $m^{asym}(\cdot)$ from $\mathcal{M}^{asym}$ as weights is more natural as $m^{asym} : \mathcal{X} \times \mathcal{Y} \mapsto \mathbb{R}_{\geq 0}$. On the other hand, $m^{sym} : \mathcal{X} \times \mathcal{Y} \mapsto [-1, 1]$, therefore, we let $m^{sym} \leftarrow |m^{sym}|$. We empirically test all six.

The hyperparameter w_s is inherent to **SSAR**, as it is required to compute $m(\cdot)$. An additional hyperparameter $\exists \forall$ downstream algorithms—M. Scalar M represents the number of previous time steps fed into the model. In our case, M represents the number of historic graphs as $\exists \mathcal{G}^t\ \forall t$. Attaching **SSAR** with a downstream algorithm involves two sliding windows: w_s and M. An intuitive visualization is provided in Figure 4. The computational details $\forall m(\cdot) \in \mathcal{M}$ are available in the Appendix.

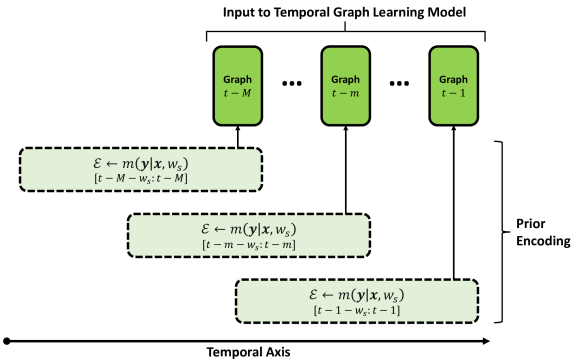

Figure 4: Sliding windows

5. Empirical Study

5.1. Data

To empirically test **SSAR**, we identify representative data sets that fit the definition of complex time series. We chose financial time series, known for their high stochasticity, non-normality, and non-stationarity [27, 28, 29]. Consequently, we sourced two data sets: (i) Inter-category and (ii) Intra-category variables. Inter- and Intra-category data sets exhaustively represent most financial time series. Henceforth, we refer to these data sets as Data Set 1 and 2, respectively. Sourced based on the largest international trading volumes, both data sets serve as representative benchmarks applicable to practitioners. The data sourcing and processing methods are detailed in the Appendix. Notably, extensive preliminary statistical tests, detailed in the Appendix, validate the time series' complexity.

5.2. Experiment Setting

We first apply **SSAR** to each data set. To examine the sensitivity to the hyperparameter w_s we apply **SSAR** $\forall w_s \in \boldsymbol{w}_s := \{20, 30, 40, 50, 60, 70, 80\}$. A minimum w_s of 20 ensures stability in information-theoretic measures. Data sets are split into training, validation, and test sets—0.5×0.7, 0.5×0.3, and 0.5, respectively for Data Set 1, and 0.6, 0.2, and 0.2, respectively for the Data Set 2. These splits simulate potential real-world scenarios.

Five established baselines are included: (i) GRU, (ii) LSTM, (iii) Linear, (iv) NLinear, (v) DLinear, where (iii), (iv), (v) have shown to outperform all state-of-the-art transformer-based architectures. The w_s for baselines corresponds to the temporal dimension size of the input vector. Next, to test the augmented representation, we select two well-known spatio-temporal GNNs—(i) [30]'s Temporal Graph Diffusion Convolution Network (diffusion t-GCN), (ii) [31]'s Temporal Graph Convolution Network (t-GCN). Notably, **SSAR** works with any downstream models that support spatio-temporal data with directed edges and dynamic weights. The number of compatible downstream models is very large. We arbitrarily let diffusion t-GCN be the downstream model for Data Set 1, and t-GCN for Data Set 2.

For ease of replication, we present the tensor operations of diffusion t-GCN for our representation in the Appendix. We do not diverge from the original method proposed by the authors for both downstream models. All experimental design choices, such as splits, downstream models, and sample sizes, were chosen *a priori* and were not changed after inference. Also, each empirical sample is independently trained from a random seed. I.e., no two test samples result from an inference of the same model $\hat{\theta}$.

The objective function J is the mean squared error (MSE) of the prediction of t given $[t\text{-}1 : t\text{-}M]$. For a fair empirical study, we systematically tune hyperparameters $h \in \mathcal{H}$ $\forall \langle w_s$, method, Data Set$\rangle$ in the train and validation set. Rigorous details of the training, validation, and inference process are provided in the Appendix.

5.3. Results and Ablation

We observe highly encouraging results, summarized in Figure 5 and Table 1. In Table 1, each column represents a method, and each row represents the w_s. Sample sizes are one for Data Set 1 and 50 for Data Set 2 for each $\langle$method, $w_s\rangle$ pair. Note that the sample size for the Constant column does not conform to this pattern as Constant weighted edges are not associated with a w_s. However, to match the sample size for each approach, the Constant column presents the 7-sample and 50-sample mean$\pm 1\sigma$ results in Data Sets 1 and 2, respectively.

The approaches are divided into (i) **SSAR**, ours, (ii) baselines, and (iii) ablation. The ablation, Constant, is where edge weights are constant in place of a statistical measure. This setup assesses the utility of graphical structures independent of statistical measures. In Data Set 1, $\forall w_s$ **SSAR** achieved the best results. Notably, a significant improvement from baselines $\rightarrow$ ablation, and another significant improvement from ablation $\rightarrow$ **SSAR**. Moreover, across 42-sample results for all six **SSAR** approaches and w_s, all 35-samples of baselines are beaten with a 100% beat rate.

For Data Set 2, each 50-sample $\langle$method, $w_s\rangle$ combination enables box-and-whisker plot analysis in Figure 5. Each box-and-whisker aggregates across w_s, i.e., they each represent $7 \cdot 50 = 350$ samples. We observe a dramatic improvement in accuracy across **SSAR**-based approaches. The box-and-whisker plot follows the standard, minimum, quartile-1, median, quartile-3, maximum value. The x-axis is intentionally not scaled to include Linear, NLinear, and DLinear outliers. Scaling would significantly reduce legibility. An enlarged version of Figure 5 is in the Appendix.

6. Discussion

6.1. Statistical Analysis

In aggregate, $7 \cdot 12$ (row $\cdot$ column) $= 84$ random seed out-sample results are available for Data Set 1, and $7 \cdot 11 \cdot 50$ (row $\cdot$ column $\cdot$ sample-size) $= 3850$ results are available for **SSAR** and baselines for Data Set 2. An additional 50 samples for the ablation leads to 3900 result samples for Data Set 2.

The statistical analysis is highly encouraging. First, we examine in aggregate whether the mean of **SSAR**s beats the aggregate mean of the baselines. Data Set 1's results are 0.7141 ± 0.0253 (42-samples) and 0.8346 ± 0.0179 (35-samples) for **SSAR**s and baselines, respectively. The T-statistic is -23.9022 (P-val $\longrightarrow$ 0). Data Set 2's results are 0.8652 ± 0.0022 (2100-samples) and 1.2740 ± 1.9097 (1750-samples) for **SSAR**s and baselines, respectively. The T-statistic is -9.8117 (P-val $\longrightarrow$ 0).

Table 1
Test Set Results (MSE)

Data Set 1

w_s	Pearson*	Spearman*	Kendall*	GC*	MI*	TE*	Constant‡	GRU†	LSTM†	Linear†	NLinear†	DLinear†
20	***0.6621***	0.6796	*0.6646*	0.7160	0.7209	0.7169	*0.7519*	0.8166	0.8154	0.8392	0.8775	*0.8338*
30	0.7149	0.6939	**0.668**	0.7237	0.7069	0.713	$\pm$	0.8154	0.8143	*0.8355*	0.8616	0.8370
40	0.7286	***0.6698***	0.7274	0.7361	0.7168	0.7055	*0.0329*	0.8161	0.8140	0.8376	0.8548	0.8368
50	0.7047	0.7082	0.7303	0.7237	***0.6966***	0.7411	$(\pm 1\sigma)$	*0.8150*	*0.8128*	0.8388	0.8540	0.8426
60	0.8144	0.7205	0.7321	**0.7077**	0.7092	0.7079	–	0.8167	0.8165	0.8451	0.8545	0.8435
70	0.7200	0.7529	0.7289	***0.7051***	0.7207	0.7098	–	0.8154	0.8133	0.8468	*0.8519*	0.8519
80	0.7246	0.7162	0.7144	0.7173	0.7194	***0.7038***	–	0.8191	0.8141	0.8477	0.8535	0.8525

Data Set 2 $(\pm 1\sigma)$

w_s	Pearson*	Spearman*	Kendall*	GC*	MI*	TE*
20	0.865570 ± 0.000001	0.869217 ± 0.000007	0.864707 ± 0.000000	***0.864074*±*0.000000***	0.867186 ± 0.000004	0.864379 ± 0.000000
30	***0.864372*±*0.000000***	0.865704 ± 0.000001	0.867756 ± 0.000004	0.864863 ± 0.000000	0.864390 ± 0.000000	0.865536 ± 0.000001
40	0.864416 ± 0.000000	0.864535 ± 0.000000	0.865272 ± 0.000000	***0.864074*±*0.000000***	0.864431 ± 0.000000	0.864167 ± 0.000000
50	0.865261 ± 0.000001	0.865146 ± 0.000000	*0.864197*±*0.000000*	0.865573 ± 0.000000	0.864614 ± 0.000000	***0.864126*±*0.000000***
60	0.864692 ± 0.000000	0.864584 ± 0.000000	0.866039 ± 0.000002	0.864528 ± 0.000000	***0.864271*±*0.000000***	0.864383 ± 0.000000
70	0.865183 ± 0.000001	0.864387 ± 0.000000	0.868527 ± 0.000004	0.867677 ± 0.000006	**0.864294±0.000000**	0.864967 ± 0.000001
80	0.864420 ± 0.000000	*0.864216 ± 0.000000*	0.865633 ± 0.000002	**0.864100±0.000000**	0.866500 ± 0.000002	0.864747 ± 0.000000

w_s	Constant‡	GRU†	LSTM†	Linear†	NLinear†	DLinear†
20	*0.867078±0.000002*	1.073429 ± 0.001211	1.072995 ± 0.000342	*1.085739±0.001977*	1.141603 ± 0.000039	1.185543 ± 0.165922
30	–	*1.073149±0.000304*	*1.072988±0.000395*	1.088698 ± 0.002433	1.126325 ± 0.000044	*1.088895±0.000072*
40	–	1.073579 ± 0.000669	1.073082 ± 0.000406	1.092317 ± 0.001307	1.121533 ± 0.000045	1.094266 ± 0.005221
50	–	1.073659 ± 0.000514	1.073188 ± 0.000435	1.997715 ± 2.352422	1.120787 ± 0.000056	1.115725 ± 0.055661
60	–	1.075270 ± 0.001245	1.073747 ± 0.001143	1.100661 ± 0.006975	3.021245 ± 5.255515	4.500515 ± 9.036276
70	–	1.073497 ± 0.000584	1.073129 ± 0.000399	1.102755 ± 0.007010	1.120726 ± 0.000061	1.101896 ± 0.005430
80	–	1.073587 ± 0.000489	1.073001 ± 0.000308	1.127654 ± 0.019237	*1.120486±0.000057*	1.107615 ± 0.012696

*SSAR: Non-Euclidean input-space, †Baseline: Euclidean input-space, ‡Ablaftion
Bold represents the best result across row, and *italicized* represents the best result across column

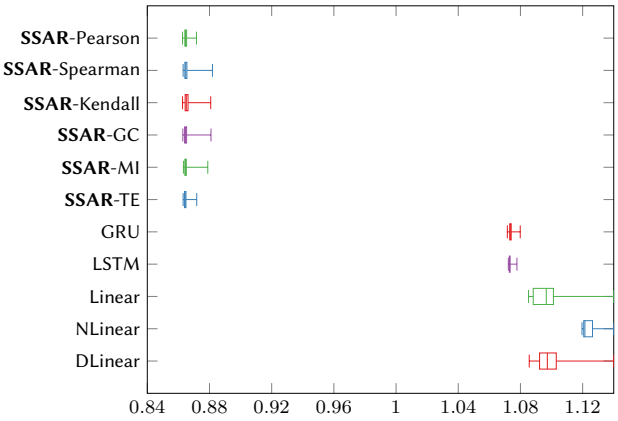

Figure 5: Data Set 2 results (Box-and-Whisker, MSE)

We identify the best-performing baseline to assess **SSAR** more rigorously. Across both datasets, LSTM shows the best mean performance. In Data Set 1, **SSAR**s against LSTM, the T-statistic is -10.3886 (P-val $\longrightarrow$ 0). The T-statistic corresponding to Data Set 2 is -1,770 (P-val $\longrightarrow$ 0). The | T-statistic | rises in Data Set 2 as the variance of LSTM is significantly lower than the baselines' aggregate.

Two ablation studies further examine our contributions. The first study, the constant weighted edge case, has been previously introduced. In Data Set 1, the aggregate mean results going from baselines $\rightarrow$ Constant $\rightarrow$ **SSAR**s is $0.8346 \pm 0.0179 \rightarrow 0.7519 \pm 0.0329 \rightarrow 0.7141 \pm 0.0253$. This corresponds to a 9.91% reduction in MSE from baselines to Constant and a 5.02% reduction from Constant to **SSAR**s. From baselines to **SSAR**s, a 14.43% reduction is observed.

In Data Set 2, going from baselines $\rightarrow$ Constant $\rightarrow$ **SSAR**s is $1.2740 \pm 1.9097 \rightarrow 0.8671 \pm 0.0027 \rightarrow 0.8652 \pm 0.0022$. This corresponds to a 31.94% reduction in MSE from baselines to Constant and a 0.22% reduction from Constant to **SSAR**s. From baselines to **SSAR**s, a 32.09% reduction is observed. Additional study on larger w_s values, with details in the Appendix, shows that statistical significance remains robust.

The second study focuses on adverse outliers in state-of-the-art methods (Linear, NLinear, DLinear). For robustness, we re-examine statistical results after excluding these models' adverse outliers. The results are detailed in the Appendix—and the statistical findings remain unchanged. This observation of significant adverse outliers bodes poorly for the baselines and contrarily emphasizes the stability of our proposed approach. By examining the F-Test on baselines and **SSAR**s, we observe an F-static of 764,534 and a corresponding one-tail F-Critical of 1.08 (P-val $\longrightarrow$ 0). The evidence indicates a significant fall in the variance of **SSAR**s.

Finally, we discuss the implications of setting w_s. A naïve interpretation might attribute **SSAR**'s improved performance to a larger implicit w_s (based on Figure 4), but this is contradicted by the lack of a significant relationship between w_s and MSE_{test} (Figure 6). Moreoever, if this was true, $\frac{\partial MSE_{test}}{\partial w_s} < 0$. On the contrary, there seems to be no meaningful relationship between w_s and MSE_{test} for the baselines. We present two histograms that summarize $p(min(MSE_{test})|w_s, B \vee S)$, where $B \vee S$ denotes a boolean with some abuse of notation—true: Baseline, false: **SSAR**.

6.2. Theoretical Implications

Initially, the performance improvement in the Constant ablation case appears surprising. Based on the theoretical discussion provided by [32], we show that **SSAR** is not only helpful in modeling the shifting underlying distribution but also implicitly smooths highly stochastic data. These

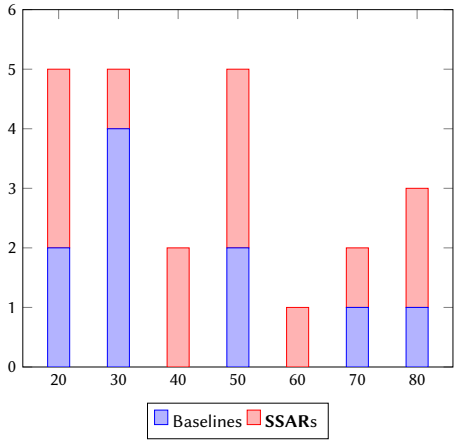

Figure 6: Histogram of minimum MSE given w_s (both data sets)

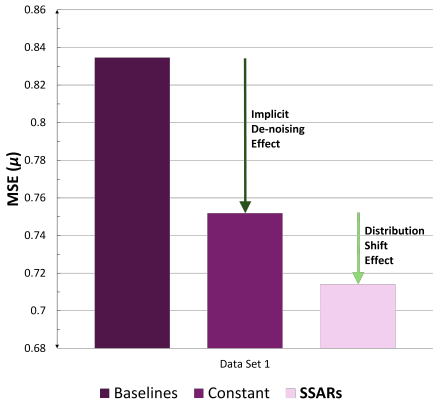

Figure 7: Two effects

effects are visually summarized in Figure 7. [32] shows that when the causal structure is very high-dimensional and therefore highly stochastic, augmenting the training data via smoothing techniques is helpful when noise-to-signal is high. The authors use exponential moving averages to smooth the input and target space. We show that **SSAR** paired with a temporal graph learning algorithm implicitly makes the same augmentations—explaining the improved performance in the Constant ablation case.

Temporal weighted graph learning algorithms for node prediction aggregate neighbouring weights and nodes each node. Afterwards, this new encoding is fed into some neural network with a sequential encoding (e.g., RNNs, Transformers). In this context, $w(\cdot)$ represents edge weights, and θ the learning system's parameters. In its theoretically simplest form, *without loss of generality*, it aggregates the weights of edges incident to the node,

$$\forall nodes,\ \hat{n}_i := n_i + \left[\sum_{e \in \mathcal{I}_i} w(e)\theta_i(e)\right], \qquad (13)$$

where $\hat{n}_i$ is the post-encoding node embedding, $\mathcal{I}_i$ is the set of edges incident to n_i, and θ_i is the learned weight parameter. First, we know that $w(\cdot) \geq 0$, and $\sum_e w(e) > 0$ for both the Constant and **SSAR** case. Then, whether $\hat{n}_i > n_i$ or $\hat{n}_i < n_i$, and magnitude $|\hat{n}_i - n_i|$ is only dependant on parameter $\theta_i(e)$. This implies, $\theta_i(e)$ can learn to de-noise the highly stochastic data. De-noising high noise-to-signal series improves results significantly [32]. Essentially, as long as the Constant weight,

$$w(\cdot) := c \in \mathbb{R}_{\neq 0}, \qquad (14)$$

$\Rightarrow \theta_i$ can implicitly learn to de-noise the input and target space, resulting in improved out-sample performance. This explains why adding no statistical-space prior, but a simple augmented representation with fixed $w(\cdot) := c > 0, \forall w(\cdot)$ resulted in improved performance.

This implicit de-noising partially explains the superior performance of **SSAR**. Remaining improvements are due to approximating Equation (8) with (9). In short, **SSAR** can be decomposed into two effects: (i) **SS**: statistical-space encoding, which tracks the underlying distribution shift, and (ii) **AR**: augmented representation, which allows for a learnable function approximator to implicitly de-noise the stochastic data.

Decomposing **SSAR** into **SS** and **AR**, unlike the clear-cut effects in Figure 7, is challenging. As seen in Equation (13), $\theta_i(e)$ could not only learn to de-noise the data but also implicitly learn the $\boldsymbol{r} \leftarrow r_{\theta_r}(\mathcal{Y}^t|\hat{\mathcal{X}}^{t-M})$ in Equation (5). Also when providng prior $p(\theta)$ in Equation (4) via $e \in \mathcal{E}$, in which it passed through Equation (13), there is no clear way of decomposing the two effects. Thus, while the ablation study aids understanding of **SSAR**'s mechanisms, it is not a rigorous method to quantify the two effects.

6.3. Future Works

Our work, which compares **SSAR** and Euclidean input-space-based state-of-the-art models, can be viewed as two ends of the extreme. Euclidean input-space-based models must learn the underlying non-stationary distribution implicitly, while **SSAR** takes a more deliberate approach.

SSAR explicitly provides a statistical-space approximation $\forall t$, resulting in (i) allowing the neural network to use an approximated regime-vector, and further learn the distribution shift, and (ii) bootstrap the neural network with priors, given that our data is limited. However, in cases where we have access to $\mathcal{D} \sim p(\cdot)$, or $|\mathcal{D}|$ is already sufficiently large, we can hypothesize that a learned statistical space may be beneficial. I.e., implement Equation (5) instead of Equation (9). In this case, the statistical space could be learned implicitly via $\theta : \cdots \times \langle n_i \rightarrow n_j \rangle \times \cdots \mapsto \mathbb{R}$, $i \neq j$ where edge weights are initialized $w^{init}(e) :\neq 0$, in Equation (13). Under the Bayesian view in Equation (4), this would correspond to the prior being a uniform distribution, $p(\theta) := U(\cdot)$. Contrarily, the statistical space could be learned explicitly where the weights of the edges are learned explicitly, $w_\theta : \cdots \times \langle n_i \rightarrow n_j \rangle \times \cdots \mapsto \mathbb{R}_{\geq 0}$, $i \neq j$. This would closely mimic the attention mechanism in transformers.

We encourage future research to explore these middle-ground approaches within the solution space spectrum presented here. A more nuanced study could theoretically and empirically study which method in the spectrum is most ideal under specific degrees of access to $p(\cdot)$, equivalently, the amount of data $\mathcal{D}$ available.

References

[1] D. M. Durairaj, B. K. Mohan, A convolutional neural network based approach to financial time series prediction, Neural Computing and Applications 34 (2022) 13319–13337.

[2] T. Dimri, S. Ahmad, M. Sharif, Time series analysis of climate variables using seasonal arima approach, Journal of Earth System Science 129 (2020) 1–16.

[3] H. D. Nguyen, K. P. Tran, S. Thomassey, M. Hamad, Forecasting and anomaly detection approaches using lstm and lstm autoencoder techniques with the applications in supply chain management, International Journal of Information Management 57 (2021) 102282.

[4] C. W. Granger, Investigating causal relations by econometric models and cross-spectral methods, Econometrica: journal of the Econometric Society (1969) 424–438.

[5] H. Lütkepohl, New introduction to multiple time series analysis, Springer Science & Business Media, 2005.

[6] S. Johansen, Estimation and hypothesis testing of cointegration vectors in gaussian vector autoregressive models, Econometrica: journal of the Econometric Society (1991) 1551–1580.

[7] Z. Liu, Y. Yang, Q. Cai, Neural network as a function approximator and its application in solving differential equations, Applied Mathematics and Mechanics 40 (2019) 237–248.

[8] A. Sherstinsky, Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network, Physica D: Nonlinear Phenomena 404 (2020) 132306.

[9] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural computation 9 (1997) 1735–1780.

[10] K. Cho, B. V. Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, Learning phrase representations using rnn encoder-decoder for statistical machine translation, arXiv preprint arXiv:1406.1078 (2014).

[11] A. Galassi, M. Lippi, P. Torroni, Attention in natural language processing, IEEE transactions on neural networks and learning systems 32 (2020) 4291–4308.

[12] M. Alam, M. D. Samad, L. Vidyaratne, A. Glandon, K. M. Iftekharuddin, Survey on deep neural networks in speech and vision systems, Neurocomputing 417 (2020) 302–321.

[13] T. Zhou, Z. Ma, Q. Wen, X. Wang, L. Sun, R. Jin, FEDformer: Frequency enhanced decomposed transformer for long-term series forecasting, in: Proc. 39th International Conference on Machine Learning (ICML 2022), Baltimore, Maryland, 2022.

[14] H. Wu, J. Xu, J. Wang, M. Long, Autoformer: Decomposition transformers with Auto-Correlation for long-term series forecasting, in: Advances in Neural Information Processing Systems, 2021.

[15] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, W. Zhang, Informer: Beyond efficient transformer for long sequence time-series forecasting, in: Proceedings of the AAAI conference on artificial intelligence, volume 35, 2021, pp. 11106–11115.

[16] S. Liu, H. Yu, C. Liao, J. Li, W. Lin, A. X. Liu, S. Dustdar, Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting, in: International conference on learning representations, 2021.

[17] S. Li, X. Jin, Y. Xuan, X. Zhou, W. Chen, Y.-X. Wang, X. Yan, Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting, Advances in neural information processing systems 32 (2019).

[18] A. Zeng, M. Chen, L. Zhang, Q. Xu, Are transformers effective for time series forecasting?, in: Proceedings of the AAAI conference on artificial intelligence, volume 37, 2023, pp. 11121–11128.

[19] L. Wu, P. Cui, J. Pei, L. Zhao, X. Guo, Graph neural networks: Foundation, frontiers and applications, in: Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD '22, Association for Computing Machinery, New York, NY, USA, 2022, p. 4840–4841. URL: https://doi.org/10.1145/3534678.3542609. doi:10.1145/3534678.3542609.

[20] Q. Cao, H. Shen, J. Gao, B. Wei, X. Cheng, Popularity prediction on social platforms with coupled graph neural networks, in: Proceedings of the 13th International Conference on Web Search and Data Mining, WSDM '20, Association for Computing Machinery, New York, NY, USA, 2020, p. 70–78. URL: https://doi.org/10.1145/3336191.3371834. doi:10.1145/3336191.3371834.

[21] Y. Wang, J. Wang, Z. Cao, A. Barati Farimani, Molecular contrastive learning of representations via graph neural networks, Nature Machine Intelligence 4 (2022) 279–287.

[22] M. Li, Z. Zhu, Spatial-temporal fusion graph neural networks for traffic flow forecasting, in: Proceedings of the AAAI conference on artificial intelligence, volume 35, 2021, pp. 4189–4196.

[23] S. Xiang, D. Cheng, C. Shang, Y. Zhang, Y. Liang, Temporal and heterogeneous graph neural network for financial time series prediction, in: Proceedings of the 31st ACM International Conference on Information & Knowledge Management, 2022, pp. 3584–3593.

[24] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, G. State, Isaac gym: High performance GPU based physics simulation for robot learning, in: Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2), 2021. URL: https://openreview.net/forum?id=fgFBtYgJQX_.

[25] G. Cybenko, Approximation by superpositions of a sigmoidal function, Mathematics of control, signals and systems 2 (1989) 303–314.

[26] K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators, Neural networks 2 (1989) 359–366.

[27] F. Alonso, D. Maldonado, A. Aguilera, J. Roldan, Memristor variability and stochastic physical properties modeling from a multivariate time series approach, Chaos, Solitons & Fractals 143 (2021) 110461.

[28] A. Bastianin, Robust measures of skewness and kurtosis for macroeconomic and financial time series, Applied Economics 52 (2020) 637–670.

[29] S. Liu, K. Wu, C. Jiang, B. Huang, D. Ma, Financial time-series forecasting: Towards synergizing performance and interpretability within a hybrid machine learning approach, arXiv preprint arXiv:2401.00534 (2023).

[30] Y. Li, R. Yu, C. Shahabi, Y. Liu, Diffusion convolutional recurrent neural network: Data-driven traffic forecasting, arXiv preprint arXiv:1707.01926 (2017).

[31] L. Zhao, Y. Song, C. Zhang, Y. Liu, P. Wang, T. Lin,

M. Deng, H. Li, T-gcn: A temporal graph convolutional network for traffic prediction, IEEE transactions on intelligent transportation systems 21 (2019) 3848–3858.
[32] W. Koh, I. Choi, Y. Jang, G. Kang, W. C. Kim, Curriculum learning and imitation learning for model-free control on financial time-series, arXiv preprint arXiv:2311.13326, AAAI 2024 AI for Time Series Analysis (2023).
[33] J. Geweke, Measurement of linear dependence and feedback between multiple time series, Journal of the American Statistical Association 77 (1982) 304–313.
[34] P. Welch, The use of fast fourier transform for the estimation of power spectra: a method based on time averaging over short, modified periodograms, IEEE Transactions on Audio and Electroacoustics 15 (1967) 70–73.
[35] C. E. Shannon, A mathematical theory of communication, The Bell System Technical Journal 27 (1948) 379–423.
[36] T. Schreiber, Measuring information transfer, Physical review letters 85 (2000) 461–464.
[37] L. Barnett, A. B. Barrett, A. K. Seth, Granger causality and transfer entropy are equivalent for gaussian variables, Physical review letters 103 (2009) 238701.
[38] L. Barnett, J. T. Lizier, M. Harré, A. K. Seth, T. Bossomaier, Information flow in a kinetic ising model peaks in the disordered phase, Physical Review Letters 111 (2013) 177203.
[39] Y. Li, R. Yu, C. Shahabi, Y. Liu, Diffusion convolutional recurrent neural network: Data-driven traffic forecasting, in: International Conference on Learning Representations, 2018. URL: https://openreview.net/forum?id=SJiHXGWAZ.
[40] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, B. Ommer, High-resolution image synthesis with latent diffusion models, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, IEEE, 2022, pp. 10684–10695.

A. Assumption: $\hat{\boldsymbol{x}}_{[,:]} := \boldsymbol{y}_{[,:]}$

The assumption that the input-space features are equivalent to the output-space features is highly reasonable. Essentially, when training to predict $\boldsymbol{y}$, since $|\mathcal{D}| \gg 0 \Rightarrow \boldsymbol{y}_{[:,]} \gg 0 \ \therefore \exists \hat{\boldsymbol{x}}_{[:,]} \gg 0$. Even if $\hat{\boldsymbol{x}}_{[,:]} \neq \boldsymbol{y}_{[,:]}$, the method and implications presented in this work hold with trivial modifications in the learning system.

B. Data Source

We use two representative data sets for financial markets. The first is an array of major macroeconomic exchange-traded funds (ETFs) and variables, available in Table 2. These variables are representative as they have been chosen based on the largest worldwide trading volumes. This data set examines the effectiveness of our approach across many financial categories (inter-asset-class). The second data set is an array of major commodity futures available in Table 3. Again, these features are chosen beforehand based on the largest worldwide trading volume. This data set examines the effectiveness of our approach within a financial category (intra-asset-class)—commodity futures market.

Table 2
Data Set 1 Description

Variable	Abbreviation	Category
SPDR Gold Trust	GLD	Commodity
U.S. Oil Fund	USO	Commodity
U.S. Dollar Index	USD	Currency
U.S. IG Corporate Bond	LQD	Fixed Income
3M Treasury Yield	3M	Interest Rate
2Y Treasury Yield	2Y	Interest Rate
10Y Treasury Yield	10Y	Interest Rate
Fed Funds Effective Rate	FFEOR	Interest Rate
10Y-3M Spread	10Y-3M	Rate Spread
10Y-2Y Spread	10Y-2Y	Rate Spread
U.S. Real Estate	IYR	Real Estate
CBOE Volatility Index	VIX	Risk
Bull-Bear Spread	BULL_BEAR_SPREAD	Sentiment

Table 3
Data Set 2 Description

Variable	Category
Wheat Futures	Commodity
Corn Futures	Commodity
Copper Futures	Commodity
Silver Futures	Commodity
Gold Futures	Commodity
Platinum Futures	Commodity
Crude Oil Futures	Commodity
Heating Oil Futures	Commodity

Both data sets are easily attainable via public sources. However, we source the data from S&P Capital IQ and Bloomberg for high-quality data that is not adjusted later—to concretely prevent any look-ahead bias. The Bull-Bear Spread is sourced separately from the Investor Sentiment Index of the American Association of Individual Investors (AAII).

The initial time step is set to the date where $\exists$ valid data points $\forall$ variable. Data Set 1's date spans from 2006-04-11 to 2022-07-08 in daily units. Data Set 2's date spans from 1990-01-01 to 2023-06-26 in daily units.

C. Data Processing

The only data processing done from raw data is transforming price data into return (change) data, and pre-processing non-available (nan) data points. We transform market variables to log return, as typical practice in the financial domain. Log return is used instead of regular difference as log allows for computational convenience. Other data points are transformed to the regular difference approach as their data points are much smaller in magnitude, and require higher levels of precision. The pseudo-code for the data processing is available in Algorithm 2.

D. Computing statistical dependencies

Given $\boldsymbol{n}_i := \{i_{t-1}, ..., i_{t-1-w_s}\}$ and $\boldsymbol{n}_j := \{j_{t-1}, ..., j_{t-1-w_s}\}$, the six measures are computed as follows. We remove the superscript t for improved legibility and let $\rho_{\boldsymbol{n}_i,\boldsymbol{n}_j}$, $\rho_{r_{\boldsymbol{n}_i},r_{\boldsymbol{n}_j}}$, $\tau_{r_{\boldsymbol{n}_i},r_{\boldsymbol{n}_j}}$, denotes Pearson correlation, Spearman rank correlation, and Kendell rank correlation, respectively. $r_{\boldsymbol{n}_i}$ denotes rank for time series

Algorithm 2 Data Process

Input: $\boldsymbol{D}_{raw}$
Output: $\boldsymbol{D}_{processed}$
Function DataProcess($\boldsymbol{D}_{raw}$):
$init\ \boldsymbol{D}_{processed}$
$\mathcal{F} \longleftarrow \boldsymbol{D}_{raw}.get\_feature\_set()$

for $f \in \mathcal{F}$ **do**
  if f **is** $MarketVariable$ **then**
    $\forall \boldsymbol{D}_{processed}[f].datapoint[i] \longleftarrow log(\boldsymbol{D}_{raw}[f].datapoint[i]/\boldsymbol{D}_{raw}[f].datapoint[i-1])$
  else if $f.datapoints \neq nan$ **then**
    $\forall \boldsymbol{D}_{processed}[f].datapoint[i] \longleftarrow \boldsymbol{D}_{raw}[f].datapoint[i] - \boldsymbol{D}_{raw}[f].datapoint[i-1]$
  else
    $\forall \boldsymbol{D}_{processed}[f].datapoint[i] \longleftarrow \boldsymbol{D}_{raw}[f].datapoint[i] - \boldsymbol{D}_{raw}[f].datapoint[most\_recent\_non\_nan\_i < i]$
  end if
end for

$\boldsymbol{D}_{processed}[\mathcal{F}].datapoint[0].drop\_timestep()$
return $\boldsymbol{D}_{processed}$
End Function

$\boldsymbol{n}_i$. Then, define each correlation as (15), (16), and (17).

$$\rho_{\boldsymbol{n}_i,\boldsymbol{n}_j} := \frac{\Sigma_t(n_i^t - \bar{\boldsymbol{n}}_i)(n_j^t - \bar{\boldsymbol{n}}_j)}{\sqrt{\Sigma_t(n_i^t - \bar{\boldsymbol{n}}_i)^2}\sqrt{\Sigma_t(n_j^t - \bar{\boldsymbol{n}}_j)^2}}, \quad (15)$$

$$\rho_{r_{\boldsymbol{n}_i},r_{\boldsymbol{n}_j}} := \frac{cov(r_{\boldsymbol{n}_i}, r_{\boldsymbol{n}_j})}{\sigma_{r_{\boldsymbol{n}_i}}\sigma_{r_{\boldsymbol{n}_j}}}, \quad (16)$$

$$\tau_{r_{\boldsymbol{n}_i},r_{\boldsymbol{n}_j}} := \frac{c-d}{\frac{1}{2}w_s(w_s-1)}, \quad (17)$$

where $\bar{\boldsymbol{n}}_i$ denotes the mean of series $\boldsymbol{n}_i$, c is the number of concordant pairs, and d is the number of discordant pairs. A pair $\langle n_i^{t,a}, n_j^{t,a}\rangle$,$\langle n_i^{t,b}, n_j^{t,b}\rangle$ is concordant if the ranks for both elements agree in their order: $(n_i^{t,a} - n_i^{t,b})(n_j^{t,a} - n_j^{t,b}) > 0$, and discordant if they disagree $(n_i^{t,a} - n_i^{t,b})(n_j^{t,a} - n_j^{t,b}) < 0$.

We used Granger causality [4] based on Geweke's method [33]. Geweke's Granger causality (GC) is a frequency-domain approach to Granger causality. Geweke's Granger causality from $\boldsymbol{n}_i$ to $\boldsymbol{n}_j$ is computed by:

$$GC_{\boldsymbol{n}_i \longrightarrow \boldsymbol{n}_j} := \ln\left(\frac{S_{n_j n_j}(f)}{S_{n_j n_j|n_i}(f)}\right), \quad (18)$$

where $S_{n_j n_j}(f)$ is the spectral density of $\boldsymbol{n}_j$ and $S_{n_j n_j|n_i}(f)$ is the spectral density of $\boldsymbol{n}_j$ given $\boldsymbol{n}_i$. We use Welch's method to estimate spectral density as it improves over periodograms in estimating the power spectral density of a signal [34].

We used two information-theoretic measures: Mutual information and Transfer entropy. Mutual information (MI) represents the shared information between two variables, indicating their statistical interdependence [35]. In information theory, the behavior of system $\boldsymbol{n}_i$ can be characterized by the probability distribution $p(\boldsymbol{n}_i)$ or $\log p(\boldsymbol{n}_i)$. This measure is equivalent to the Pearson correlation coefficient if both have a normal distribution. To compute MI between two variables, we need to know the information entropy, which is formulated as follows:

$$H(\boldsymbol{n}_i) := -\sum_{n_i \in \boldsymbol{n}_i} p(n_i)\log_2 p(n_i). \quad (19)$$

Shannon entropy quantifies the information required to select random values from a discrete distribution. The joint (information) entropy can be expressed as:

$$H(\boldsymbol{n}_i, \boldsymbol{n}_j) := -\sum_{n_i \in \boldsymbol{n}_i, n_j \in \boldsymbol{n}_j} p(n_i, n_j)\log_2 p(n_i, n_j). \quad (20)$$

Finally, we can define MI as the quantity of identifying the interaction between subsystems.

$$MI(\boldsymbol{n}_i, \boldsymbol{n}_j) := H(\boldsymbol{n}_i) + H(\boldsymbol{n}_j) - H(\boldsymbol{n}_i, \boldsymbol{n}_j). \quad (21)$$

Following Kvålseth (2017), we use normalized MI (NMI) with range [0, 1] to ensure consistency across measures. The computation is as follows:

$$NMI(\boldsymbol{n}_i, \boldsymbol{n}_j) := \frac{MI(\boldsymbol{n}_i; \boldsymbol{n}_j)}{\min(H(\boldsymbol{n}_i), H(\boldsymbol{n}_j))}. \quad (22)$$

Transfer entropy (TE) is a non-parametric metric leveraging Shannon's entropy, quantifying the amount of information transfer between two variables [36]. Based on conditional MI in Equation (23), we can define the general form of (k, l)-history TE between two sequences $\boldsymbol{n}_i$ and $\boldsymbol{n}_j$ for $\boldsymbol{n}_{i,t}^{(k)} = (\boldsymbol{n}_{i,t}, ., \boldsymbol{n}_{i,t-k+1})$ and $\boldsymbol{n}_{j,t}^{(l)} = (\boldsymbol{n}_{j,t}, ., \boldsymbol{n}_{j,t-l+1})$. It is computed as Equation (24):

$$H(\boldsymbol{n}_j|\boldsymbol{n}_i) := -\sum_{n_j \in \boldsymbol{n}_j, n_i \in \boldsymbol{n}_i} p(n_i, n_j)\log_2 \frac{p(n_i, n_j)}{p(n_i)}. \quad (23)$$

$$TE^{(k,l)}_{\boldsymbol{n}_{i,t} \longrightarrow \boldsymbol{n}_{j,t}}(t) :=$$

$$\sum_{\Omega} p(n_{j,t+1}, n_{i,t}^{(k)}, n_{j,t}^{(l)})\log_2 \frac{p(n_{j,t+1}|n_{i,t}^{(k)}, n_{j,t}^{(l)})}{p(n_{j,t+1}|n_{j,t}^{(l)})}, \quad (24)$$

where $\Omega := \{n_{j,t+1}, n_{i,t}^{(k)}, n_{j,t}^{(l)}\}$, which represents the possible sets of those three values. $TE^{(k,l)}_{\boldsymbol{n}_{i,t} \longrightarrow \boldsymbol{n}_{j,t}}(t)$ is the information about the future state of $\boldsymbol{n}_{j,t}$ which is retrieved by subtracting information retrieved from only $\boldsymbol{n}_{j,t}^{(l)}$, and from information gathered from $\boldsymbol{n}_{i,t}^{(k)}$ and $\boldsymbol{n}_{j,t}^{(l)}$. We set k and l to 1. Under these conditions, the equation for TE with $(1, 1)$-history can be computed as

$$TE^{(1,1)}_{\boldsymbol{n}_{i,t} \longrightarrow \boldsymbol{n}_{j,t}}(t) =$$

Table 4
Descriptive Statistics (Data Set 1)

Statistic	GLD	USO	USD	LQD	3M	2Y	10Y
Mean, μ	0.0003	-0.0003	0.0001	0	-0.0002	-0.0002	-0.0001
Standard deviation, σ	0.0156	0.0156	0.0156	0.0156	0.0156	0.0156	0.0156
Skewness	-0.334	-1.1831	-0.2533	-0.5572	-0.8177	-0.0971	-0.1188
Kurtosis	6.3099	14.8191	4.0016	67.8462	77.2192	11.4556	3.507
Q_0	-0.1255	-0.1912	-0.1394	-0.2696	-0.2663	-0.1529	-0.1432
Q_1	-0.0071	-0.0078	-0.0083	-0.006	-0.0033	-0.0068	-0.0084
Q_2	0.0007	0.0005	0.0003	0.0009	0	0	0
Q_3	0.0083	0.0079	0.0088	0.0066	0.0033	0.0068	0.0084
Q_4	0.1461	0.101	0.0883	0.263	0.2498	0.1291	0.0814

Statistic	FFEOR	10Y-3M	10Y-2Y	IYR	VIX	BULL_BEAR_SPREAD
Mean, μ	-0.0002	0	0	0.0001	0	-0.0001
Standard deviation, σ	0.0156	0.0156	0.0156	0.0156	0.0156	0.0156
Skewness	-0.4796	0.3396	-0.0493	-0.6986	1.0469	-0.0633
Kurtosis	88.5722	13.4647	3.6125	18.3742	5.9128	0.317
Q_0	-0.2175	-0.1226	-0.1122	-0.1889	-0.0705	-0.0538
Q_1	0	-0.0074	-0.008	-0.0051	-0.0088	-0.0102
Q_2	0	0	0	0.0006	-0.0013	0.0002
Q_3	0	0.0074	0.008	0.0059	0.0071	0.0107
Q_4	0.2404	0.1744	0.0841	0.1238	0.1546	0.0561

$$\sum_{\Omega} p(\boldsymbol{n}_{j,t+1}, \boldsymbol{n}_{j,t}, \boldsymbol{n}_{i,t}) \log_2 \frac{p(\boldsymbol{n}_{j,t+1}, \boldsymbol{n}_{i,t}, \boldsymbol{n}_{j,t}) p(\boldsymbol{n}_{j,t})}{p(\boldsymbol{n}_{j,t+1}, \boldsymbol{n}_{j,t}) p(\boldsymbol{n}_{i,t}, \boldsymbol{n}_{j,t})}, \tag{25}$$

where $\Omega = \{\boldsymbol{n}_{j,t+1}, \boldsymbol{n}_{i,t}, \boldsymbol{n}_{j,t}\}$.

This measure can be perceived as conditional mutual information, considering a variable's influence as a condition. Also, analogous to the established relationship between the Pearson correlation coefficient and mutual information, an equivalent association can be identified when the two variables comply with the premises of normal distribution [37]. TE measures information flow via uncertainty reduction. "TE from Y to X," translates to the extent Y clarifies the future of X beyond what X can clarify about its own future. Conditional entropy quantifies the requisite information to derive the outcome of a random variable X, given that the value of another random variable Y is known. It is computed as [38]:

E. Descriptive Statistics and Statistical Properties

Tables 4, 5, 6, and 7 summarize the time series's descriptive statistics and statistical property tests. The means and standard deviations clearly indicate the high noise-to-signal ratio—$\mu \approx 0$ and $\sigma >> |\mu|$.

All eight normality statistics strongly indicate non-normality. Most features are non-auto-correlated, and all features are non-stationary. "***" denotes rejection of the null hypothesis of statistical tests at the 0.01 level of significance, "**" at the 0.05 level, and "*" at the 0.1 level.

F. Graph Diffusion Convolutional Network

We implement a t-GCN powered by diffusion convolutional recurrent neural networks (DCRNN) to learn **SSAR**'s spatial and temporal dependency structure [39]. DCRNN shows state-of-the-art performance in modeling traffic dynamics with a spatial and temporal dimension—represented graphically.

The graph signal $\mathcal{X} \in \mathbb{R}^{N \times 1}$ as each node has a single feature. With $\mathcal{X}^t$ representing the signal observed at time t, the diffusion t-GCN learns a function $g(\cdot)$:

$$[\mathcal{X}^{t-M}, \ldots, \mathcal{X}^{t-1}; \mathcal{G}] \xrightarrow{g(\cdot)} [\mathcal{X}^t]. \tag{26}$$

The diffusion process explicitly captures the spatial dimension and its stochastic features. The diffusion process in generative modeling works by encoding information via increasing noise through a Markov process while decoding information via reversing the noise process [40]. The diffusion mechanism here is characterized by a random walk on $\mathcal{G}$ with restart probability $\alpha \in [0, 1]$, and state transition matrix $\boldsymbol{D}_O^{-1}\mathcal{W}$, where $\boldsymbol{D}_O = diag(\mathcal{W}\mathbf{1})$ is the out-degree diagonal matrix, and $\mathbf{1} \in \mathbb{R}^N$ is the all-one vector. The stationary distribution $\mathcal{P} \in \mathbb{R}^{N \times N}$ of the diffusion process can be computed via the closed form:

$$\mathcal{P} := \sum_{k=0}^{K=\infty} \alpha(1-\alpha)^k (\boldsymbol{D}_O^{-1}\mathcal{W})^k. \tag{27}$$

After sufficient time steps, as represented by the summation to infinity, the Markov process converges to $\mathcal{P}$. The intuition is as follows. $\mathcal{P}_{i,:} \in \mathbb{R}^N$ represents the diffusion probability from n_i, i.e., it quantifies the proximity with respect to the node. k denotes the diffusion steps, and K is typically set to a finite natural number as each step is analogous to the filter size in convolution.

As a result, the diffusion convolution over our $\mathcal{X}$ and a filter f_θ is described by:

$$\mathcal{X}_{:,1} \star_G f_\theta := \sum_{k=0}^{K-1} (\theta_{k,1}(\boldsymbol{D}_O^{-1}\mathcal{W})^k + \theta_{k,2}(\boldsymbol{D}_I^{-1}\mathcal{W}^T)^k)\mathcal{X}_{:,1}, \tag{28}$$

where $\theta \in \mathbb{R}^{K \times 2}$ are filter parameters and $\boldsymbol{D}_O^{-1}\mathcal{W}$, $\boldsymbol{D}_I^{-1}\mathcal{W}^T$ are the diffusion process transition matrices with the latter representing the reverse process. A diffusion convolution layer within a neural network architecture would map the

Table 5
Statistical Tests (Data Set 1)

Test	Type	GLD	USO	USD	LQD	3M
Shapiro-Wilk	Normality	0.9412***	0.9106***	0.9674***	0.7205***	0.5102***
D'Agostino K-squared	Normality	611.7238***	1504.0233***	421.8482***	1671.8988***	1886.876***
Lilliefors	Normality	0.0711***	0.0694***	0.0494***	0.1092***	0.2776***
Jarque-Bera	Normality	6834.6006***	38241.7946***	2761.1181***	781940.6002***	1013102.2671***
Kolmogorov–Smirnov	Normality	0.4771***	0.4764***	0.4761***	0.4783***	0.4758***
Anderson-Darling	Normality	44.797***	49.312***	22.4254***	143.6918***	527.6964***
Cramér–von Mises	Normality	327.3755***	327.6465***	327.0075***	329.5083***	332.1592***
Omnibus	Normality	611.7238***	1504.0233***	421.8482***	1671.8988***	1886.876***
Bruesch-Godfrey (5d)	Autocorrelation	0.6548	1.8472	1.8339	1.7377	15.8475***
Ljung-Box (5d)	Autocorrelation	0.6468	1.8631	1.8317	1.7303	16.0909***
Augmented Dicky-Fuller	Stationarity	-64.3747***	-9.0878***	-63.4386***	-12.0401***	-10.498***
Zivot-Andrews	Stationarity	-64.4844***	-9.5335***	-63.5556***	-12.7633***	-11.4003***
Phillips-Perron	Stationarity	-64.7455***	-64.5927***	-63.4453***	-64.147***	-52.8798***
Statistic	Type	2Y	10Y	FFEOR	10Y-3M	10Y-2Y
Shapiro-Wilk	Normality	0.8738***	0.9706***	0.3504***	0.8904***	0.9594***
D'Agostino K-squared	Normality	783.0878***	347.5444***	1720.7889***	918.375***	348.7078***
Lilliefors	Normality	0.1361***	0.0678***	0.3541***	0.09***	0.0969***
Jarque-Bera	Normality	1332463.2869***	30862.2392***	2216.2285***	22288.2101***	2096.6655***
Kolmogorov–Smirnov	Normality	0.474***	0.4772***	0.4764***	0.4753***	0.4765***
Anderson-Darling	Normality	116.837***	22.8665***	894.9531***	60.1568***	38.2637***
Cramér–von Mises	Normality	328.2026***	326.93***	334.2273***	327.8856***	327.0914***
Omnibus	Normality	783.0878***	347.5444***	1720.7889***	918.375***	348.7078***
Bruesch-Godfrey (5d)	Autocorrelation	19.2069***	1.947	44.1751***	20.4973***	4.4516
Ljung-Box (5d)	Autocorrelation	18.6156***	1.9235	45.6069***	20.3085***	4.4324
Augmented Dicky-Fuller	Stationarity	-9.294***	-13.635***	-11.2644***	-9.8368***	-47.7222***
Zivot-Andrews	Stationarity	-10.9044***	-47.8802***	-10.7794***	-14.2945***	-11.688***
Phillips-Perron	Stationarity	-67.0877***	-64.7715***	-74.8074***	-59.8906***	-63.198***
Test	Type	IYR	VIX	BULL_BEAR_SPREAD		
Shapiro-Wilk	Normality	0.8043***	0.936***	0.9975***		
D'Agostino K-squared	Normality	1247.5579***	1034.8942***	15.9253***		
Lilliefors	Normality	0.1303***	0.0785***	0.0166***		
Jarque-Bera	Normality	57660.7643***	6680.5179***	19.6321***		
Kolmogorov–Smirnov	Normality	0.4713***	0.4784***	0.4797***		
Anderson-Darling	Normality	175.1698***	49.7635***	1.8623***		
Cramér–von Mises	Normality	328.9962***	327.3539***	326.375***		
Omnibus	Normality	1247.5579***	1034.8942***	15.9253***		
Bruesch-Godfrey (5d)	Autocorrelation	5.7178	8.9025	90.8848***		
Ljung-Box (5d)	Autocorrelation	5.7007	8.969	84.6422***		
Augmented Dicky-Fuller	Stationarity	-11.7587***	-26.8927***	-16.958***		
Zivot-Andrews	Stationarity	-13.299***	-26.9875***	-16.9997***		
Phillips-Perron	Stationarity	-76.062***	-75.9458***	-17.7063***		

Table 6
Descriptive Statistics (Data Set 2)

Test	Wheat	Corn	Copper	Silver	Gold	Platinum	Crude Oil	Heating Oil
Mean	0.0001	0.0001	0.0002	0.0002	0.0002	0.0001	0.0002	0.0001
Standard deviation	0.0196	0.0171	0.0164	0.0184	0.0102	0.0149	0.0259	0.0239
Skewness	-0.2386	-1.1802	-0.2838	-0.7025	-0.2434	-0.9835	-0.4843	-1.3415
Kurtosis	12.7754	20.1985	4.392	7.324	7.4048	17.7037	20.1242	17.3938
Q_0	-0.2861	-0.2762	-0.1171	-0.1955	-0.0982	-0.2719	-0.4005	-0.3909
Q_1	-0.0111	-0.0085	-0.0083	-0.0079	-0.0044	-0.074	-0.012	-0.0114
Q_2	0	0	0	0.0005	0.0002	0.0005	0.0006	0.0006
Q_3	0.0107	0.0089	0.0088	0.0091	0.0052	0.008	0.0129	0.0124
Q_4	0.233	0.1276	0.1164	0.122	0.0889	0.1272	0.3196	0.1399

signal's feature size to an output of dimension $\mathcal{Q}$. As we are working with a single feature, we denote a parameter tensor as $\Theta \in \mathbb{R}^{Q\times 1\times K\times 2} = [\theta]_{q,1}$. The parameters for the qth output is $\Theta_{q,1} \in \mathbb{R}^{K\times 2}$. In short, the diffusion convolutional layer is described as:

$$\mathcal{H}_{:,q} := a(\mathcal{X}_{:,1} \star_G f_{\Theta_{q,1,:,:}}), \quad for\ q \in \{1,\dots,\mathcal{Q}\}. \tag{29}$$

Where input $\mathcal{X} \in \mathbb{R}^N$ is mapped to output $\mathcal{H} \in \mathbb{R}^{N\times\mathcal{Q}}$, and $a(\cdot)$ is an activation function. With this GCN structure, we can train the network parameters via stochastic gradient

Table 7
Statistical Tests (Data Set 2)

Test	Type	Wheat	Corn	Copper	Silver
Shapiro-Wilk	Normality	0.9389***	0.9174***	0.9552***	0.9241***
D'Agostino K-squared	Normality	1752.8365***	3332.3089***	944.7587***	1783.9423***
Lilliefors	Normality	0.0455***	0.0637***	0.0542***	0.0845***
Jarque-Bera	Normality	57149.8899***	144614.381***	6856.3562***	19445.8744***
Kolmogorov–Smirnov	Normality	0.472***	0.4738***	0.4748***	0.4723***
Anderson-Darling	Normality	49.7856***	89.5722***	67.6838***	127.1812***
Cramér–von Mises	Normality	666.284***	671.1711***	671.5948***	669.1654***
Omnibus	Normality	1752.8365***	3332.3089***	944.7587***	1783.9423***
Bruesch-Godfrey (5d)	Autocorrelation	3.1197	1.0919	0.8877	7.1119
Ljung-Box (5d)	Autocorrelation	3.0741	1.0829	0.8948	7.1953
Augmented Dicky-Fuller	Stationarity	-20.5887***	-88.3374***	-24.3989***	-30.846***
Zivot-Andrews	Stationarity	-20.7714***	-88.3823***	-24.5412***	-31.0458***
Phillips-Perron	Stationarity	-92.3386***	-88.3095***	-96.4925***	-93.9752***
Test	Type	Gold	Platinum	Crude Oil	Heating Oil
Shapiro-Wilk	Normality	0.9289***	0.928***	0.8887***	0.9089***
D'Agostino K-squared	Normality	1305.4712***	2923.7298***	2357.3219***	3450.6279***
Lilliefors	Normality	0.0803***	0.0631***	0.0715***	0.0691***
Jarque-Bera	Normality	19253.8825***	110951.6968***	141944.7626***	108313.6979***
Kolmogorov–Smirnov	Normality	0.4826***	0.4781***	0.4646***	0.4664***
Anderson-Darling	Normality	114.6614***	80.9492***	120.7886***	97.4767***
Cramér–von Mises	Normality	682.8616***	674.6513***	657.7334***	660.199***
Omnibus	Normality	1305.4712***	2923.7298***	2357.3219***	3450.6279***
Bruesch-Godfrey (5d)	Autocorrelation	3.6289	7.7309	16.502***	5.2739
Ljung-Box (5d)	Autocorrelation	3.5897	7.73	16.9965***	5.3533
Augmented Dicky-Fuller	Stationarity	-36.1101***	-15.4566***	-15.5826***	-24.3777***
Zivot-Andrews	Stationarity	-36.3411***	-15.801***	-15.7147***	-24.4921***
Phillips-Perron	Stationarity	-92.7712***	-89.3201***	-92.1408***	-93.4847***

descent.

G. Diffusion Convolutional Gated Recurrent Unit

Next, the temporal dimension is modeled via a GRU, a variant of RNNs that better captures longer-term dependencies. Diffusion convolution replaces standard matrix multiplication in the GRU architecture:

$$\mathbf{r}^t := \sigma(\Theta_r \star_G [\mathcal{X}^t, \mathcal{H}^{t-1}] + \mathbf{b}_r), \tag{30}$$

$$\mathbf{u}^t := \sigma(\Theta_u \star_G [\mathcal{X}^t, \mathcal{H}^{t-1}] + \mathbf{b}_u), \tag{31}$$

$$\mathcal{H}^t := \mathbf{u}^t \odot \mathcal{H}^{t-1} + (1 - \mathbf{u}^t) \odot \mathcal{C}^t, \tag{32}$$

$$\mathcal{C}^t := \tanh(\Theta_\mathcal{C} \star_\mathcal{G} [\mathcal{X}^t, (\mathbf{r}^t \odot \mathcal{H}^{t-1})] + \mathbf{b}_c), \tag{33}$$

where in time step t, $\mathbf{r}^t$, $\mathbf{u}^t$, $\mathcal{X}^t$, $\mathcal{H}^t$ represent the reset gate, update gate, input tensor, and output tensor, respectively. Θ_r, Θ_u, $\Theta_\mathcal{C}$ represent the corresponding filter parameters [30].

H. Training And Inference Method

The pseudo-code for the training and inference pipeline is available in Algorithms 3, 4, 5, and 6. The $RandomGridSearch(\cdot)$ in Algorithm 3 is done with 260 random seed trials with 13 parallel CPU cores.

The $\mathcal{H}_{searchspace}$ for the GCNs are as follows.

- Input Size: [8, 9, ..., 30]
- Hidden Layer Size: [8, 16, ..., 120]
- Learning Rate: $[1e^{-1}, 1e^{-2}, ..., 1e^{-6}]$
- Epochs: [2, 3, ..., 30]
- k: [1, 2, ..., 6] (only for linear measures)

The $\mathcal{H}_{searchspace}$ for all baselines are as follows. The input size does not need tuning as they are w_s.

- Hidden Layers Size: [8, 16, ..., 128] (nonapplicable to Linear, NLinear, DLinear)
- Learning Rate: $[1e^{-1}, 1e^{-2}, ..., 1e^{-6}]$
- Epochs: [5, 10, ..., 30]

The tuned hyperparameters for each data set are presented in Tables 8, 9, 10, and 11.

The diffusion t-GCN has five hyperparameters: (i) input vector size M , (ii) hidden layer size, (iii) diffusion steps (filter size), k (iv) learning rate, and (v) training epochs. The k for the set of non-linear causal measures, $\mathcal{M}^{asym}$, is set to 1 as the sparsity in $w(e) > 0$ causes computational errors. This makes the hyperparameter count for $m(\cdot) \in \mathcal{M}^{asym}$, four. The output vector size is set to one as the network predicts one time step in the future. The hyperparameters are equivalently optimized $\forall\langle m(\cdot), w_s\rangle$ combination. The same approach is taken for t-GCN but excludes the hyperparameter k as it is not part of the model.

I. Data Set 2 Test Set Quartile Results

The results in Table 12 are for $w_s \in \{20, ..., 80\}$ in aggregate, corresponding to the main text's Figure 5. Figure 8 is Figure 5 of the main text enlarged for better legibility. We note that the Constant case is excluded as its smaller sample size does not allow for fair statistical comparison.

Algorithm 3 Training t-GCNs

Input: $\boldsymbol{\mathcal{G}}$, $\mathcal{M} := \{m_0(\cdot), ..., m_K(\cdot)\}$, $\mathcal{W} := \{w_s^0, ..., w_s^L\}$, $\mathcal{H}_{searchspace}$, $split\_ratio, test\_sample\_count$, $tuining\_sample\_count$
Output: $\hat{\boldsymbol{\theta}}$
Function TrainGCN($\boldsymbol{\mathcal{G}}$, $\mathcal{M}$, $\mathcal{W}$, $\mathcal{H}_{searchspace}$, $split\_ratio, test\_sample\_count, tuining\_sample\_count$):

$\mathcal{G}_{train}, \mathcal{G}_{validation}, \mathcal{G}_{test} \longleftarrow split\_ratio(\boldsymbol{\mathcal{G}})$

for $\forall m_k(\cdot) \in \mathcal{M}$ **do**
  for $\forall w_s^l \in \mathcal{W}$ **do**
    $samples_k^l \longleftarrow RandomGridSearch(\mathcal{H}_{searchspace}, \mathcal{G}_{train}, \mathcal{G}_{validation}, m_k(\cdot), w_s^l, tuining\_sample\_count)$
    $\hat{\mathcal{H}} \longleftarrow arg\_min(samples_k^l.MSE_{validation})$
    $\mathcal{G}_{train} \longleftarrow Concat(\mathcal{G}_{train}, \mathcal{G}_{validation})$
    for $0, 1, ..., test\_sample\_count - 1$ **do**
      $\hat{\theta} \longleftarrow TrainModel(\mathcal{G}_{train}, \hat{\mathcal{H}}, AdamOptimizer)$
      $\hat{\boldsymbol{\theta}}.add(\hat{\theta}, m(\cdot)_k, w_s^l)$
    end for
  end for
end for

return $\hat{\boldsymbol{\theta}}$
End Function

Algorithm 4 Training Baselines

Input: $\boldsymbol{D}_{processed}$, $\lambda := \{Model_0(\cdot), ..., Model_M(\cdot)\}$, $\mathcal{W} := \{w_s^0, ..., w_s^L\}$, $\mathcal{H}_{searchspace}$, $split\_ratio, test\_sample\_count$
Output: $\hat{\boldsymbol{\theta}}$
Function TrainBaselines($\boldsymbol{D}_{processed}$, λ, $\mathcal{W}$, $\mathcal{H}_{searchspace}$, $split\_ratio, test\_sample\_count$):

$\boldsymbol{D}_{train}, \boldsymbol{D}_{validation}, \boldsymbol{D}_{test} \longleftarrow split\_ratio(\boldsymbol{D}_{processed})$

for $\forall Model_m(\cdot) \in \lambda$ **do**
  for $\forall w_s^l \in \mathcal{W}$ **do**
    $samples_k^l \longleftarrow GlobalSearch(\mathcal{H}_{searchspace}, \boldsymbol{D}_{train}, \boldsymbol{D}_{validation}, Model_m, w_s^l)$
    $\hat{\mathcal{H}} \longleftarrow arg\_min(samples_k^l.MSE_{validation})$
    $\boldsymbol{D}_{train} \longleftarrow Concat(\boldsymbol{D}_{train}, \boldsymbol{D}_{validation})$
    for $0, 1, ..., test\_sample\_count - 1$ **do**
      $\hat{\theta} \longleftarrow TrainModel(\boldsymbol{D}_{train}, \hat{\mathcal{H}}, AdamOptimizer)$
      $\hat{\boldsymbol{\theta}}.add(\hat{\theta}, Model_m, w_s^l)$
    end for
  end for
end for

return $\hat{\boldsymbol{\theta}}$
End Function

Algorithm 5 t-GCN Inference

Input: $\boldsymbol{\mathcal{G}}$, $\mathcal{M} := \{m_0(\cdot), ..., m_K(\cdot)\}$, $\mathcal{W} := \{w_s^0, ..., w_s^L\}$, $\hat{\boldsymbol{\theta}}$, $split\_ratio, test\_sample\_count$
Output: $\mathbf{MSE}_{test}$
Function InferenceGCN($\boldsymbol{\mathcal{G}}$, $\mathcal{M}$, $\mathcal{W}$, $\hat{\boldsymbol{\theta}}$, $split\_ratio, test\_sample\_count$):

$\mathcal{G}_{train}, \mathcal{G}_{validation}, \mathcal{G}_{test} \longleftarrow split\_ratio(\boldsymbol{\mathcal{G}})$

for $\forall m(\cdot)_k \in \mathcal{M}$ **do**
  for $\forall w_s^l \in \mathcal{W}$ **do**
    for $0, 1, ..., test\_sample\_count - 1$ **do**
      $MSE_{test} \longleftarrow Inference(\mathcal{G}_{test}, \hat{\boldsymbol{\theta}})$
      $\mathbf{MSE}_{test}.add(MSE_{test}, m(\cdot)_k, w_s^l)$
    end for
  end for
end for

return $\mathbf{MSE}_{test}$ **End Function**

Table 8
Data Set 1 Tuned Hyperparameters, $\hat{\mathcal{H}}$

Epochs												
w_s	Pearson*	Spearman*	Kendall*	GC*	MI*	TE*	Constant‡	GRU†	LSTM†	Linear†	NLinear†	DLinear†
20	28	16	24	5	2	4	29	5	5	10	20	10
30	14	20	16	3	27	27	6	5	5	10	10	5
40	21	21	22	9	9	27	7	5	5	5	10	5
50	20	29	23	6	14	4	10	5	5	10	10	5
60	8	5	29	26	23	12	17	5	5	10	10	5
70	15	4	2	7	4	3	5	5	5	10	10	5
80	3	12	4	10	12	2	7	5	5	10	10	5
Learning Rate												
w_s	Pearson*	Spearman*	Kendall*	GC*	MI*	TE*	Constant‡	GRU†	LSTM†	Linear†	NLinear†	DLinear†
20	$1e^{-06}$	$1e^{-05}$	$1e^{-05}$	0.001	0.01	0.001	$1e^{-06}$	0.001	0.001	0.01	0.1	0.001
30	0.0001	$1e^{-05}$	$1e^{-06}$	0.01	0.1	0.1	0.01	0.001	0.001	0.001	0.01	0.001
40	0.0001	$1e^{-05}$	0.0001	$1e^{-06}$	$1e^{-06}$	0.1	$1e^{-06}$	0.001	0.001	0.001	0.01	0.001
50	0.1	0.1	0.01	$1e^{-06}$	$1e^{-06}$	$1e^{-06}$	$1e^{-06}$	0.001	0.001	0.001	0.001	0.001
60	$1e^{-06}$	$1e^{-06}$	0.1	0.1	0.1	0.1	0.1	0.001	0.001	0.001	0.001	0.001
70	0.01	0.01	$1e^{-06}$	$1e^{-06}$	0.01	$1e^{-06}$	$1e^{-06}$	0.001	0.001	0.001	0.001	0.001
80	0.01	0.01	0.01	0.01	0.01	0.1	$1e^{-06}$	0.001	0.001	0.001	0.001	0.001
Hidden Layer(s) Size												
w_s	Pearson*	Spearman*	Kendall*	GC*	MI*	TE*	Constant‡	GRU†	LSTM†	Linear†	NLinear†	DLinear†
20	120	96	72	88	16	120	16	128	64	—	—	—
30	112	112	56	32	16	8	16	128	64	—	—	—
40	72	32	40	32	80	8	48	64	128	—	—	—
50	32	16	8	96	24	80	64	32	64	—	—	—
60	80	96	8	16	16	8	16	32	128	—	—	—
70	16	112	80	80	16	72	24	128	64	—	—	—
80	16	8	24	56	8	24	16	32	128	—	—	—

*__SSAR__: Non-Euclidean input-space, †Baseline: Euclidean input-space, ‡Ablation

Table 9
Data Set 1 Tuned Hyperparameters, $\hat{\mathcal{H}}$ (t-GCN-only)

Input Size							
w_s	Pearson	Spearman	Kendall	GC	MI	TE	Constant
20	27	30	28	29	30	27	29
30	30	30	27	21	30	28	17
40	30	27	30	28	29	23	27
50	19	13	14	18	19	17	24
60	24	30	30	30	29	29	29
70	30	30	25	28	30	29	25
80	25	27	30	28	29	29	18
k							
w_s	Pearson	Spearman	Kendall	GC	MI	TE	Constant
20	2	2	3	—	—	—	—
30	2	2	3	—	—	—	—
40	2	2	2	—	—	—	—
50	1	1	1	—	—	—	—
60	4	3	1	—	—	—	—
70	2	2	5	—	—	—	—
80	1	1	1	—	—	—	—

J. Larger w_s

The statistical analysis for larger w_s is equally encouraging. In reference to Table 13, first, we examine in aggregate whether **SSAR**s beats the baselines. The aggregate $\mu \pm 1\sigma$ for Data Set 2 are 0.8664 ± 0.0060 (600-samples) and 2.0326 ± 9.0136 (500-samples) for **SSAR**s and baselines, respectively. The T-statistic is -3.1695, corresponding to a one-sided p-value of 0.0008.

To rigorously assess **SSAR**, we identify the best-performing baseline. Here, GRU performs best when taking the mean value. The T-statistic performance against GRU is -344.66 (P-val $\longrightarrow$ 0). |T-statistic| rises as the variance of GRU is significantly lower than the aggregate. In conclusion, the results hold even when raising the w_s.

K. Second Ablation

Data Set 1 has no outliers due to the lower sample size. Therefore, we analyze the results after controlling for outliers in Data Set 2. First, we identify outliers as $Q_3 + 3 \cdot IQR > MSE_i, Q1 - 3 \cdot IQR < MSE_i$, where Q_n represents the nth quartile, IQR represents Inter Quartile Range,

Table 10
Data Set 2 Tuned Hyperparameters, $\hat{\mathcal{H}}$

Epochs												
w_s	Pearson*	Spearman*	Kendall*	GC*	MI*	TE*	Constant‡	GRU†	LSTM†	Linear†	NLinear†	DLinear†
20	20	16	24	5	2	4	29	10	30	20	10	30
30	19	20	16	3	27	27	6	10	30	20	10	10
40	28	21	22	9	9	27	7	20	10	20	10	30
50	8	29	23	6	14	4	10	10	10	30	10	20
60	29	5	29	26	23	12	17	5	30	20	20	30
70	9	4	2	7	4	3	5	10	10	20	10	20
80	30	12	4	10	12	2	7	20	20	30	10	20
Learning Rate												
w_s	Pearson*	Spearman*	Kendall*	GC*	MI*	TE*	Constant‡	GRU†	LSTM†	Linear†	NLinear†	DLinear†
20	$1e^{-06}$	$1e^{-05}$	$1e^{-05}$	0.001	0.01	0.001	$1e^{-06}$	0.01	0.0001	0.001	0.001	0.1
30	$1e^{-06}$	$1e^{-05}$	$1e^{-06}$	0.01	0.1	0.1	0.1	0.0001	0.0001	0.001	0.001	0.001
40	$1e^{-06}$	$1e^{-05}$	0.0001	$1e^{-06}$	$1e^{-06}$	0.1	$1e^{-06}$	0.0001	0.0001	0.001	0.001	0.001
50	$1e^{-06}$	0.1	0.01	$1e^{-06}$	$1e^{-06}$	$1e^{-06}$	$1e^{-06}$	0.0001	0.01	0.1	0.001	0.01
60	$1e^{-06}$	$1e^{-06}$	0.1	0.1	0.1	0.1	0.1	0.001	0.01	0.001	0.1	0.1
70	$1e^{-06}$	0.01	$1e^{-06}$	$1e^{-06}$	0.01	$1e^{-06}$	$1e^{-06}$	0.0001	0.0001	0.001	0.001	0.1
80	$1e^{-06}$	0.01	0.01	0.01	0.01	0.1	$1e^{-06}$	0.0001	0.0001	0.0001	0.001	0.001
Hidden Layer(s) Size												
w_s	Pearson*	Spearman*	Kendall*	GC*	MI*	TE*	Constant‡	GRU†	LSTM†	Linear†	NLinear†	DLinear†
20	56	96	72	88	16	120	16	32	16	—	—	—
30	96	112	56	32	16	8	16	128	32	—	—	—
40	112	32	40	32	80	8	48	32	16	—	—	—
50	88	16	8	96	24	80	64	32	128	—	—	—
60	64	96	8	16	16	8	16	16	8	—	—	—
70	96	112	80	80	16	72	24	32	32	—	—	—
80	96	8	24	56	8	24	16	32	32	—	—	—

*__SSAR__: Non-Euclidean input-space, †Baseline: Euclidean input-space, ‡Ablation

Table 11
Data Set 2 Tuned Hyperparameters, $\hat{\mathcal{H}}$ (t-GCN-only)

Input Size							
w_s	Pearson	Spearman	Kendall	GC	MI	TE	Constant
20	16	30	28	29	30	27	29
30	16	30	27	21	30	28	17
40	16	27	30	28	29	23	27
50	16	13	14	18	19	17	24
60	17	30	30	30	29	29	29
70	16	30	25	28	30	29	25
80	16	27	30	28	29	29	18

and MSE_i is a MSE data point. We observe that all outliers are adverse, i.e., $Q_3 + 3 \cdot IQR > MSE_i$. This is expected, as a low MSE outlier would be numerically impossible since MSE > 0. Therefore, all outliers worsen performance and sharply reduce the stability of the learning system. The outlier study is done, including larger w_s tested in Appendix J. We summarize the identified outliers in Table 14.

We examine the results post-outlier-removal in Table 15. First, we examine in aggregate whether **SSAR**s beats the baselines. The aggregate $\mu \pm 1\sigma$ is 0.8654 ± 0.0023 (2700-samples) and 1.1025 ± 0.0320 (2250-samples) for **SSAR**s and baselines, respectively. The T-statistic is -383.82 (P-val $\longrightarrow 0$).

To more rigorously assess the out-performance of our approach, we identify the best-performing baseline. Here, LSTM performs best when taking the mean MSE. Against

Table 12
Test Set Result Quartiles (MSE)

Data Set 2											
Quartiles	Pearson*	Spearman*	Kendall*	GC*	MI*	TE*	GRU†	LSTM†	Linear†	NLinear†	DLinear†
Q_0	0.8627	0.8630	0.8627	0.8629	0.8632	0.8630	1.0716	1.0722	1.0852	1.1196	1.0857
Q_1	0.8642	0.8642	0.8643	0.8640	0.8642	0.8640	1.0731	1.0729	1.0882	1.1207	1.0923
Q_2	0.8645	0.8647	0.8651	0.8645	0.8645	0.8643	1.0735	1.0730	1.0966	1.1215	1.0973
Q_3	0.8653	0.8655	0.8662	0.8653	0.8651	0.8649	1.0740	1.0734	1.1012	1.1263	1.1030
Q_4	0.8716	0.8819	0.8808	0.8810	0.8789	0.8718	1.0799	1.0777	9.4546	20.5654	33.3744

*__SSAR__: Non-Euclidean input-space, †Baseline: Euclidean input-space

Algorithm 6 Baselines Inference

Input: $\boldsymbol{D}_{processed}$, $\lambda := \{Model_0(\cdot), ..., Model_M(\cdot)\}$, $\mathcal{W} := \{w_s^0, ..., w_s^L\}$, $\hat{\boldsymbol{\theta}}$, $split\_ratio, test\_sample\_count$
Output: $\textbf{MSE}_{test}$
Function InferenceBaselines($\boldsymbol{D}_{processed}$, λ, $\mathcal{W}$, $\hat{\boldsymbol{\theta}}$, $split\_ratio, test\_sample\_count$):

$\boldsymbol{D}_{train}, \boldsymbol{D}_{validation}, \boldsymbol{D}_{test} \longleftarrow split\_ratio(\boldsymbol{D}_{processed})$

for $\forall Model_m(\cdot) \in \lambda$ **do**
  for $\forall w_s^l \in \mathcal{W}$ **do**
    for $0, 1, ..., test\_sample\_count - 1$ **do**
      $MSE_{test} \longleftarrow Inference(\boldsymbol{D}_{test}, \hat{\boldsymbol{\theta}})$
      $\textbf{MSE}_{test}.add(MSE_{test}, Model_m, w_s^l)$
    end for
  end for
end for

return $\textbf{MSE}_{test}$ **End Function**

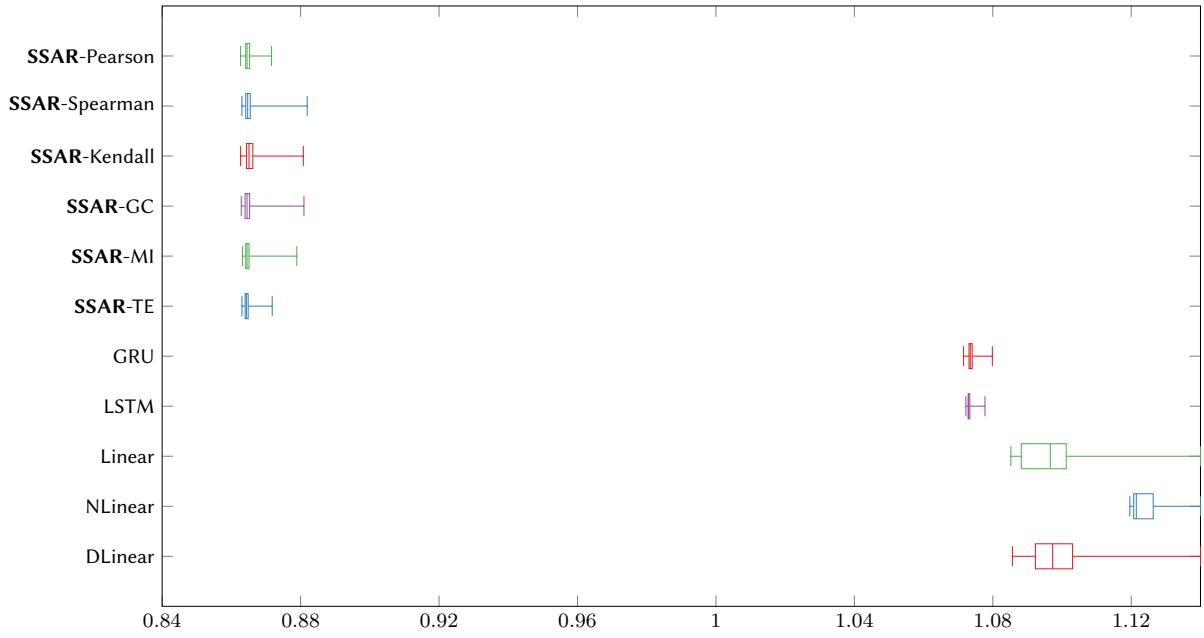

Figure 8: Data Set 2 Test Set (Box-and-Whisker, MSE)

LSTM, the T-statistic is -1,905 (P-val $\longrightarrow$ 0). Correspondingly, we conclude that the results hold even when the adverse outliers in the baselines are removed.

L. Complexity and Scalability

The complexity of our representation can be described in two steps: computing the (i) Statistical-space matrix and then (ii) generating the graph. Consistent with the main

Table 13
Random-seed Test Set Results (MSE)

	Data Set 2 ($\pm 1\sigma$)					
w_s	Pearson*	Spearman*	Kendall*	GC*	MI*	TE*
90	0.8741 ± 0.0000	0.8768 ± 0.0000	0.8643 ± 0.0000	0.8648 ± 0.0000	0.8648 ± 0.0000	**0.8643±0.0000**
100	0.8644 ± 0.0000	0.8648 ± 0.0000	0.8649 ± 0.0000	0.8651 ± 0.0000	0.8647 ± 0.0000	**0.8641±0.0000**
w_s	Constant‡	GRU†	LSTM†	Linear†	NLinear†	DLinear†
90	0.8671±0.0000	1.0734 ± 0.0005	1.0750 ± 0.0058	1.1320 ± 0.1128	1.1229 ± 0.0015	1.1084 ± 0.0040
100	—	1.0736 ± 0.0006	1.0740 ± 0.0011	10.4293 ± 27.3390	1.1255 ± 0.0001	1.1121 ± 0.0100

*SSAR: Non-Euclidean input-space, †Baseline: Euclidean input-space, ‡Ablation
Bold represents the best result across row

Table 14
Outliers

Model	w_s	Count	Mean MSE
Linear	50	6	8.0023
Linear	100	7	67.2112
NLinear	60	7	14.4714
DLinear	60	8	22.2426

Table 15
Random-seed Test Set Results, Excluding Outliers (MSE)

				Data Set 2 ($\pm 1\sigma$)		
w_s	Pearson*	Spearman*	Kendall*	GC*	MI*	TE*
20	0.865570 ± 0.000001	0.869217 ± 0.000007	0.864707 ± 0.000000	***0.864074±0.000000***	0.867186 ± 0.000004	0.864379 ± 0.000000
30	***0.864372±0.000000***	0.865704 ± 0.000001	0.867756 ± 0.000004	0.864863 ± 0.000000	0.864390 ± 0.000000	0.865536 ± 0.000001
40	0.864416 ± 0.000000	0.864535 ± 0.000000	0.865272 ± 0.000000	***0.864074±0.000000***	0.864431 ± 0.000000	0.864167 ± 0.0000000
50	0.865261 ± 0.000001	0.865146 ± 0.000000	*0.864197±0.000000*	0.865573 ± 0.000000	0.864614 ± 0.000000	**0.864126±0.000000**
60	0.864692 ± 0.000000	0.864584 ± 0.000000	0.866039 ± 0.000002	0.864528 ± 0.000000	***0.864271±0.000000***	0.864383 ± 0.000000
70	0.865183 ± 0.000001	0.864387 ± 0.000000	0.868527 ± 0.000004	0.867677 ± 0.000006	**0.864294±0.000000**	0.864967 ± 0.000001
80	0.864420 ± 0.000000	*0.864216±0.000000*	0.865633 ± 0.000002	**0.864100±0.000000**	0.866500 ± 0.000002	0.864747 ± 0.000000
90	0.874072 ± 0.000025	0.876404 ± 0.000038	0.864294 ± 0.000000	0.864844 ± 0.000001	0.864818 ± 0.000000	**0.864264±0.000000**
100	0.864424 ± 0.000000	0.864796 ± 0.000000	0.864930 ± 0.000000	0.865146 ± 0.000000	0.864677 ± 0.000000	***0.864123± 0.000000***
w_s	Constant‡	GRU†	LSTM†	Linear†	NLinear†	DLinear†
20	*0.867078±0.000002*	1.073429 ± 0.001211	1.072995 ± 0.000342	*1.085739± 0.001977*	1.141603 ± 0.000039	1.185543 ± 0.165922
30	—	*1.073149±0.000304*	*1.072988±0.000395*	1.088698 ± 0.002433	1.126325 ± 0.000044	*1.088895± 0.000072*
40	—	1.073579 ± 0.000669	1.073082 ± 0.000406	1.092317 ± 0.001307	1.121533 ± 0.000045	1.094266 ± 0.005221
50	—	1.073659 ± 0.000514	1.073188 ± 0.000435	1.178907 ± 0.212394	1.120787 ± 0.000056	1.115725 ± 0.055661
60	—	1.075270 ± 0.001245	1.073747 ± 0.001143	1.100661 ± 0.006975	1.157271 ± 0.101624	1.121063 ± 0.052763
70	—	1.073497 ± 0.000584	1.073129 ± 0.000399	1.102755 ± 0.007010	1.120726 ± 0.000061	1.101896 ± 0.005430
80	—	1.073587 ± 0.000489	1.073001 ± 0.000308	1.127654 ± 0.019237	*1.120486± 0.000057*	1.107615 ± 0.012696
90	—	1.073363 ± 0.000518	1.075029 ± 0.005778	1.132033 ± 0.112828	1.122947 ± 0.001541	1.108393 ± 0.003993
100	—	1.073579 ± 0.000604	1.073994 ± 0.001135	1.185696 ± 0.254091	1.125477 ± 0.000060	1.112132 ± 0.009970

*__SSAR__: Non-Euclidean input-space, †Baseline: Euclidean input-space, ‡Ablation
Bold represents the best result across row, and *italicized* represents the best result across column

text, N denotes the number of features, and T denotes total samples, i.e., time steps. k denotes the number of bins for MI and TE. Table 16 summarizes the time and space complexity for step (i). Each complexity value is multiplied by N^2 corresponding to each edge, i.e., the directed pair.

The time complexity of generating the temporal graph representation is $O(T \times N^2)$. The corresponding space complexity is $O(T \times N^2)$ if stored in an adjacency matrix and $O(T \times (N + |E|))$ if stored in an adjacency list, where $|E|$ is the size of the directed edge list. **SSAR** is highly scalable in both the temporal and feature dimensions, given that the computed measures are provided. By using a finer discrete time step, T can easily rise. However, the complexity rises linearly $w.r.t.$ T for both the time and space complexity. Despite rising non-linearly, N^2 $w.r.t.$ N we note that $N \ll T$. This pattern will hold when scaling to larger data sets to avoid overfitting.

We used Nvidia GTX 4070 Ti and Nvidia GTX 2080 Ti as our GPUs for the baselines that can leverage high-core count parallel computing. We always used a single GPU system for each computational task. We used commonly available 6 to 32 virtual CPU core systems. Lastly, we used systems with 30 to 32 GB of RAM. Despite a total of 5084 random seed (ablations and baselines included) training and inference experiments, our total time spent running experiments was within two weeks. We approximate that with five parallel systems, each with 5 CPU cores for the GCNs and 5 CPU cores and a CUDA-enabled GPU for baselines, all empirical studies can be conservatively replicated within ten days. We expect our implementation to have no scaling challenges in modern AI clusters.

Table 16
Computational Complexity of Measures

Measure, $m(\cdot)$	Time Complexity	Space Complexity
Pearson Correlation	$O(T \times N^2)$	$O(T \times N^2)$
Spearman Rank Correlation	$O(TlogT \times N^2)$	$O(T \times N^2)$
Kendall Rank Correlation	$O(TlogT \times N^2)$	$O(T \times N^2)$
Mutual Information	$O(TlogT \times N^2)$	$O(k^2 \times N^2)$
Granger Causality	$O(T^2logT \times N^2)$	$O(T \times N^2)$
Transfer Entropy	$O(T^3 \times N^2)$	$O(k^2 \times N^2)$