Correlation-aware Spatial-Temporal Graph Learning for Multivariate Time-series Anomaly Detection

Yu Zheng, Huan Yee Koh, Ming Jin, Lianhua Chi, Khoa T. Phan, Shirui Pan, Yi-Ping Phoebe Chen, Wei Xiang

***Abstract*—Multivariate time-series anomaly detection is critically important in many applications, including retail, transportation, power grid, and water treatment plants. Existing approaches for this problem mostly employ either statistical models which cannot capture the non-linear relations well or conventional deep learning models (e.g., CNN and LSTM) that do not explicitly learn the pairwise correlations among variables. To overcome these limitations, we propose a novel method, correlation-aware spatial-temporal graph learning (termed `CST-GL`), for time-series anomaly detection. `CST-GL` explicitly captures the pairwise correlations via a multivariate time series correlation learning module based on which a spatial-temporal graph neural network (STGNN) can be developed. Then, by employing a graph convolution network that exploits one- and multi-hop neighbor information, our STGNN component can encode rich spatial information from complex pairwise dependencies between variables. With a temporal module that consists of dilated convolutional functions, the STGNN can further capture long-range dependence over time. A novel anomaly scoring component is further integrated into `CST-GL` to estimate the degree of an anomaly in a purely unsupervised manner. Experimental results demonstrate that `CST-GL` can detect anomalies effectively in general settings as well as enable early detection across different time delays.**

***Index Terms*—Multivariate Time Series, Anomaly detection, Graph neural networks.**

I. Introduction

RAPID developments in Cyber-Physical Systems (CPS) have resulted in an explosive growth of time-series data collected across industries. In many applications, the CPS implemented generates time-series data from multiple devices or sensors, forming a complex *multivariate time-series*. Importantly, an operator may have thousands to millions of CPS systems, recording a manually unmanageable amount of multivariate time-series data. For example, each server of a cloud infrastructure provider generates multivariate time-series data and many providers may have up to over millions of servers [1]. A similar scale in the CPS system has also been observed in numerous commercial systems and critical infrastructures including power systems, spacecraft [2], engines, transportation, cyber networks [3], and water treatment plants [4]. Relying on human labours to monitor these operations would thus be not only impractical but also impossible.

To enable effective monitoring and warning of large-scale system operations, multivariate time-series anomaly detection has become an important topic. Successful implementation of multivariate time-series anomaly detection model could bring substantial economic and social benefits. For instance, in a water treatment plant [5], hundreds of sensors are installed to monitor water level, flow rates and water quality. A malicious attack may occur by simply turning on a single motorized valve, causing a disastrous cascading effect on the entire water distribution system. Automatically monitoring and detecting these abnormal behaviours can thus provide a fast response, which helps rectify errors, reduce cost, and save lives.

Among the various implementation approaches for detecting anomaly events, *unsupervised* anomaly detection is one of which has attracted the most attention due to the difficulty of obtaining ground-truth anomalies over time. Early approaches typically employed either statistical unsupervised models such as ARIMA/VAR [6] or distance-based approaches [7], [8]. Unfortunately, these methods cannot capture the non-linear spatial and temporal relationship from the multivariate time-series data well. More recently, with the flourish of deep learning (DL), significant advances have been made. For instance, Hundman et al. proposed a Long Short-Term Memory (LSTM) network together with a nonparametric thresholding approach [2] and Su et al. proposed a representation learning-based stochastic recurrent neural network [3] approach to improving the current ability to detect multivariate time-series anomaly events. While the proposed DL frameworks can efficiently scale through high-dimensional multivariate time-series data, they did not explicitly model the underlying pairwise inter-dependence among variable pairs, weakening their capacity in detecting complex anomaly events.

The difficulty of detecting anomaly events in multivariate time-series data lies in the fact that the variable pairs are intricately related. Figure 1 shows a real-world inspired example of multivariate time-series data with six variables where A, B and C represents closely related variable groups. A1 variable is not closely related to other variables and the detection of anomaly events can simply be a significant deviation from past behaviours. B1 and B2 are two inter-related variables that should go up and down together, a deviation from this relationship is thus an anomaly event. On the other hand, C1 always increases with a lag after C2 is switched on (upward spike). The exception to the C1-C2 relationship (grey span of Figure 1) is when C3 is also switched on as C3 decreases

Y. Zheng, L. Chi, K. T. Phan, Y-P. P. Chen, and W. Xiang are with Department of Computer Science and Information Technology, La Trobe University, Melbourne Australia. E-mail: {yu.zheng, l.chi, k.phan, phoebe.chen, w.xiang}@latrobe.edu.au.

H. Y. Koh and M. Jin are with the Department of Data Science and AI, Faculty of IT, Monash University, Clayton, VIC 3800, Australia. E-mail: {ming.jin, huan.koh}@monash.edu.

S. Pan is with School of Information and Communication Technology, Griffith University, Australia. Email: s.pan@griffith.edu.au.

Y. Zheng and H. Y. Koh contributed equally to this work.

L. Chi is the corresponding author.

Manuscript received Jan 3, 2022; revised xx xx, 202x.

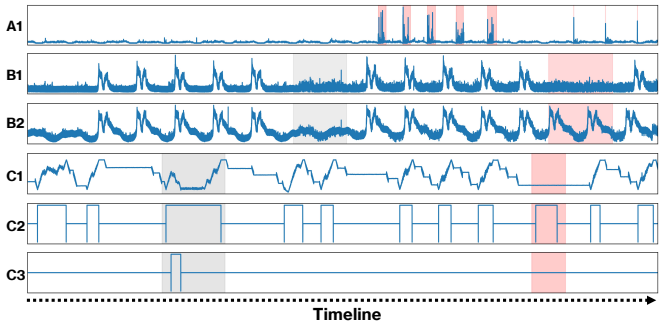

Fig. 1. An example of multivariate time-series data. Red span represents anomaly events while grey span represents time event highlights that are non-anomalous but form a reference to compare the behaviours of anomaly events. The first three examples showcased are from the Server Machine Dataset (SMD) [3], which contains data from internet servers, while the last three are drawn from the WADI, a water treatment plant sensing dataset [5].

C1, creating an offsetting effect on C2. The red span in the C variable groups indicates that an anomaly event has occurred because C1 does not increase despite C2 being switched on and C3 being switched off. While variable pairs that form the multivariate time-series are naturally interdependent, the degree of inter-dependence tells the full story of multivariate time-series data. Further, as shown above, the complexity increases exponentially with the increase in the number of variables. It is thus crucial for an anomaly detection model to not only assume the inter-dependent relationship but to *explicitly learn and capture the pairwise correlations (i.e., degree of spatial dependence) between the variables of a multivariate time-series*.

To explicitly capture the pairwise correlations, a natural way is to model multivariate time series as a graph. For example, by treating each sensor as a node, a sensor graph can be constructed in which the node features are continuously changed over time. With a representative graph, spatial-temporal graph neural networks (STGNNs) can then be employed to tackle the multivariate time series anomaly detection task by explicitly modeling the pairwise correlations via a graph neural network (GNN) module and temporal information via a CNN [9] or RNN [10] module. However, using the generic STGNN models to explicitly model pairwise correlations between variable pairs requires a predefined graph that is often not available in many multivariate time-series data. Consequently, while previous STGNN methods are equipped to capture spatial dependencies, their capacity to learn and construct the relationship between the variables may not always be optimized, especially in cases where a predefined graph is not readily available in many multivariate time-series data.

To address the limitations of generic STGNNs, Graph Deviation Network (GDN) [11] employs a simple graph learning layer to learn and construct the pairwise correlation relationship between the variable pairs. Then, a graph attention network is used to propagate historical information among the variables to forecast the next observation. Anomaly events are subsequently detected based on the magnitude of deviation between forecast and real observations. Nevertheless, in this preliminary study, GDN only captures the spatial dependency in the direct neighbors of each variable, which may cause it to lose important information from high-order (multi-hop) neighbors. Furthermore, it did not explicitly model temporal relations within each univariate time series, which are crucial for characterizing multivariate time series data [12] and thus further compromises the effectiveness of GDN.

Based on the above observations, we summarize the challenges for multivariate time series anomaly detection from the graph neural network perspective as follows.

- **Multivariate time-series correlation learning (Challenge 1).** The underlying correlations among the time-series variable pairs are important for multivariate time series anomaly detection task. How to explicitly capture the pairwise relations among variables to enable spatial-temporal analysis is the first challenge.
- **Spatial-temporal dependency modeling (Challenge 2).** Multivariate time-series analysis requires a deep understanding of spatial-temporal dependency; how to simultaneously capture spatial and temporal dependency remains a challenge for multivariate time series.
- **Anomaly scoring (Challenge 3).** How to estimate the anomaly score in an unsupervised way is the ultimate challenge for multivariate time series anomaly detection.

To address these challenges, we propose a novel algorithm `CST-GL` in this paper. Our theme is to model multivariate time series as a graph and design a spatial-temporal graph neural network to perform forecasting. Based on the forecasting results, the anomaly score can be well estimated, and the anomalies can be detected accordingly. To be more specific, we propose a multivariate time series correlation learning module that can automatically infer the underlying correlation among variables (*for Challenge 1*). Then, a well-designed spatial-temporal graph neural network is presented to model both the spatial and temporal dependency (*for Challenge 2*). The spatial dependence is modeled via a graph convolution network based on the gated mix-hop feature propagation that exploits neighbors from both single and multiple hops to better encode spatial information. The temporal dependence is captured via a temporal convolutional network which incorporates a gating mechanism with temporal convolution functions for long dependence modeling. Based on the forecasting results, we propose an anomaly forecast indicator that performs normalization on the most recent window of historical data and estimates the anomaly scores based on the reconstruction from a simple Principal Component Analysis model (*for Challenge 3*). Experimental results on real datasets demonstrate the superb performance of our method.

The main contributions of this paper are as follows:

- We propose an integrated algorithm for multivariate time series data analysis. Our method seamlessly integrates correlation learning into a spatial-temporal network for multivariate time series.
- We propose a novel algorithm for multivariate time series anomaly detection. Our method can automatically detect anomalies from complex time-series via auto-thresholding and enable early detections effectively.
- We compare our method with *eleven* baselines for both general multivariate time series anomaly detection which

aims to evaluate the overall performance in a whole dataset, as well as early detection of anomalies that needs to detect anomaly as early as possible. Our experimental results demonstrated that our method outperforms all baselines in both settings.
- We conduct a case study to demonstrate that our method not only enables effective anomaly detection but also provides interpretability in real-life applications.

The rest of the paper is structured as follows. Section II reviews the related work. Section III gives the definition of the task. Section IV presents the proposed CST−GL. Section V illustrates our experiments and conclusion in Section VI.

II. Related Work

In this section, we introduce the past work on multivariate time-series anomaly detection and graph neural networks.

A. Anomaly Detection in Multivariate time-series

Detecting anomalies in time-series is a challenging task that has been perennially studied [13], [14], [15]. Historically, statistical models such as ARIMA/VAR [6], PCA [16] and SVM [17] have been applied to detect anomalies in univariate and multivariate time-series. Traditional techniques involving wavelet analysis [18], non-parametric [19], pattern-based [20], [21] and distance-based [7], [8] approach have also been collaboratively implemented. More recently, substantial efforts have been made to advance deep learning approaches for anomaly detection in multivariate time-series data across numerous domains [2], [3], [22]. As argued by [23], [24], this phenomenon has arisen because (a) deep learning frameworks are free from stationary assumptions and can scale through high dimensional temporal data and (b) unlike pattern-based approaches that only detect anomaly events by identifying anomalous sub-sequences, deep learning frameworks can detect anomalous event timestamp-by-timestamp within sequences and are thus well suited for the deployment of real-time streaming anomaly detection systems.

Deep learning models for multivariate time-series anomaly detection are primarily designed using recurrent neural network (RNN) that are combined either with convolutional neural networks (CNN) [22], [25], variational autoencoder (VAE) [26], [3] or Generative Adversarial Networks [27]. The RNN is employed to capture temporal dependencies [28], [29], [2] while the CNN, VAE or GAN is incorporated to capture dependencies among the multivariate variables. Any time-series observations which unexpectedly deviate from the learned temporal and relational dependencies would then be treated as anomalies. However, since CNN, VAE and GAN do not explicitly learn the relationship between the multivariate variables and only encapsulate interactions among variables into a global hidden state, they cannot fully exploit the latent dependencies between the variable pairs [30], [31]. For more research on deep learning for time-series anomaly detection, we refer readers to the most recent survey [32].

B. Graph Learning

Graph learning [33] is a new learning paradigm that enables machine learning for graph data. A key component of this paradigm, graph neural networks, have been widely studied to handle an array of graph-structured data [34]. This includes a well-known subset of methods, namely spatial-temporal graph neural networks, which are typically applied to modeling multivariate time series [35]. In this context, graph structure learning is often involved when prior knowledge of the underlying graph topology is not readily available.

Generic graph neural networks. Graph neural networks (GNNs) have recently become de facto models to exploit graph data for graph analytics [36], [37], [38], [39], [40], [41]. The core idea of graph neural networks is to employ a *message passing* scheme, which iteratively updates the representation (embedding) of a target node by propagating the representations of neighboring nodes. For instance, GCN [36] updates its node embedding by assigning a predefined weight to each message (embedding) from a neighbor. GAT [37] automatically learns the weight of each neighbor and performs a weighted aggregation to update the target node's representation. Due to the outstanding capacity of modeling inter-relationship of different entities in various domains, GNNs have been widely used in domains and applications including traffic [42], [35], recommender systems [43], drug discovery [44], and anomaly detection [45], [46].

Graph Structure Learning. Learning graph neural network models typically requires a predefined graph structure so that the *message passing* can be performed along with the topological structure. However, in many applications related to time series, the graph structure may be not available and the GNN models are not directly applicable. To overcome this challenge, graph structure learning [47], [48] recently has emerged to automatically learn the graph structure from the data itself. For instance, SUBLIME [48] presents a structure bootstrapping contrastive learning framework to infer the relationship among data. However, these approaches can only be applied to static data. For dynamic data such as time series considered in this paper, these methods cannot be directly applied.

Spatial-Temporal Graph Neural Networks. To extend GNNs for handling dynamic graph-structured data, recent research has delved into spatial-temporal graph neural networks (STGNNs) [49]. These are especially effective in situations where the underlying graph structure remains static, but the features of the nodes undergo dynamic changes over time. A prime example of STGNNs in action is traffic forecasting, where the physical infrastructure such as subway stations and tracks is constant, but the traffic volume fluctuates continuously. Seo et al. [10] proposed a recurrent STGNN which adopts the Long-short Term Memory Networks (LSTMs) [50] and Graph Convolution Network (GCN) [36] as key components to capture temporal and spatial dependencies. Instead, Li et al. [9] proposed a CNN-based method (1D convolution) to capture the capture temporal dependencies and a GCN to capture spatial dependencies. Wu et al. [23] propose a joint graph structure learning and forecasting framework for spatial-temporal modeling. However, these methods typically only

consider general forecasting tasks and they did not exploit anomaly detection from time-series data.

III. Problem Formulation

A multivariate time-series with T successive observations of equal-spaced samples as represented $\mathbf{X} = \{\mathbf{x}^1, \mathbf{x}^2, \cdots, \mathbf{x}^T\}, \mathbf{x}^t \in \mathbb{R}^N$ is composed of N number of univariate time-series $\{\mathbf{x}_1^t, \mathbf{x}_2^t, \cdots, \mathbf{x}_N^t\}$. In a real-time fashion, the multivariate time-series anomaly detection task requires learning of a scoring function, $A(\cdot)$, that outputs an anomaly score to current observation T so that we have $A(\mathbf{x}^{\mathrm{A}}) > A(\mathbf{x}^{\mathrm{M}})$ where $\mathbf{x}^{\mathrm{A}}$ is anomalous observation and $\mathbf{x}^{\mathrm{M}}$ is not. Ideally, the proposed framework should also output a binary label that indicates whether a timestamp is anomalous or not, where $y^T \in \{0, 1\}$ and $y^T = 1$ if the observation $\mathbf{x}^T$ is anomalous.

In this paper, we consider the unsupervised *real-time* anomaly detection task. Firstly, a model is required to learn the normality of a time-series based on a non-anomalous train set. Then, given streaming time-series observations that consists both normal and anomalous observations, the model should detect anomaly events in real time. Under this setting, models can only rely on past observations to make a decision at every timestamp and cannot reverse its previous decisions.

IV. Methodology

In this section, we present the overall framework of `CST-GL` and its detailed designs to detect anomaly events in a multivariate time-series. As shown in Figure 2, our method mainly consists of three main constituents: I. *multivariate time-series correlation learning*, II. *spatial-temporal graph neural network*, and III. *anomaly detection and diagnosis* module.

Given a multivariate time-series, we first propose to exploit the latent associations (i.e., edges) between each univariate time-series (i.e., nodes) explicitly via a pairwise correlation learner, where the learned graph structure together with the historical observations are then encoded by a sandwich-structured spatial-temporal graph neural network to make reliable forecasting. Specifically, we interlace the designed graph and temporal convolutions to capture rich spatial and temporal dependencies respectively. The underlying considerations are two-fold: (1) The potential anomalies in a univariate time-series can be easily identified by referring to its historical observations. For example, a sudden high CPU wattage is likely to trigger the system alert if compared with long-term historical readings. (2) However, for multivariate time-series data, the anomalies in a specific variable may not only associate with its historical observations but also the readings of other variables. A concrete example is traffic networks, where the change of road conditions in a street may cause a serious traffic jam in another one. Thus, it is crucial to model the underlying spatial and temporal dependencies in historical observations to perform precise and stable anomaly detection at each time step. To accomplish this goal, we propose a *anomaly detection and diagnosis* module on the top of *multivariate time-series correlation learning* and *spatial-temporal graph neural network*, where the anomaly score at each time point is derived from the forecasting errors. In other words, we conjecture that time-series anomalies are typically reflected as the mismatch between *anomalous observations* and the forecasting results given by the well-trained spatial-temporal model on *non-anomalous data*.

Further, we argue that for root cause of anomaly events to be identified, pairwise correlations of variables have to be learned and captured by a proposed model. This is because univariates that deviate significantly from past spatial and temporal behaviours may only be symptoms of the root cause and the relationship of pairwise correlation can reveal the root cause variables. As shown later in the experimental section, `CST-GL` identifies root cause of an anomaly events using the well-learned pairwise correlation graph that captures well the inter-dependence relationship between the variable pairs.

In the rest of this section, we introduce the multivariate time-series correlation learning (MTCL) in Subsection IV-A. Then, in Subsection IV-B1 and IV-B2, we illustrate the detailed designs of the proposed spatial-temporal graph neural network (STGNN) in capturing the underlying spatial and temporal clues for accurate forecasting. Finally, we discuss how the proposed anomaly detection and diagnosis module can compute the real-time anomaly score in Subsection IV-C1 and identify the root cause of an anomaly event in Subsection IV-C2.

A. Multivariate Time-Series Correlation Learning

To explicitly enable the modeling of pairwise dependencies among variables in a multivariate time series, we design a correlation learning layer and propose to learn the underlying unknown graph adjacency matrix $\mathbf{A}$ adaptively, where nodes and edges denote variables and their connectivity. Specifically, our detailed formulation is given as follows:

$$\begin{cases} \widetilde{\mathbf{N}}_1 = tanh(\alpha \mathbf{N}_1 \mathbf{W}_1), \\ \widetilde{\mathbf{N}}_2 = tanh(\alpha \mathbf{N}_2 \mathbf{W}_2), \\ \mathbf{A} = ReLU\Big(tanh\big(\alpha(\widetilde{\mathbf{N}}_1 \widetilde{\mathbf{N}}_2^{\mathsf{T}} - \widetilde{\mathbf{N}}_2 \widetilde{\mathbf{N}}_1^{\mathsf{T}})\big)\Big), \end{cases} \tag{1}$$

where $\mathbf{N}_1, \mathbf{N}_2 \in \mathbb{R}^{N \times d}$ are two randomly initialized node embedding matrices, and $\mathbf{W}_1, \mathbf{W}_2 \in \mathbb{R}^{d \times d}$ are two set of trainable parameters. The hyper-parameter α denotes the non-linear activation saturation rate. Compared with our approach, many existing works construct such adjacency matrix by measuring the pairwise distance or similarity between variables in a multivariate time-series, such as Euclidean distance [51] and Cosine similarity [52], resulting in high time and space complexity of $O(N^2)$ [23]. Another significant drawback of existing methods based on distance or similarity metrics is that the learnt pairwise dependencies are symmetric, which is not desired in describing the relations between variables in real-world multivariate time series. For example, the traffic jam on a street may cause the jam on another street but may not vice versa if there are alternative routes. Thus, we expect the learned time series dependencies to be uni-directional. Let $\widetilde{\mathbf{N}}_1$ and $\widetilde{\mathbf{N}}_2$ be two transformed node embedding matrices, the uni-directional property can then be achieved by the subtraction term $\widetilde{\mathbf{N}}_1 \widetilde{\mathbf{N}}_2^{\mathsf{T}} - \widetilde{\mathbf{N}}_2 \widetilde{\mathbf{N}}$ and two nonlinear activation functions, i.e., if A_{ij} is a positive number, then A_{ji} will be zero. The output adjacency matrix $\mathbf{A}$ will have all its elements regularized

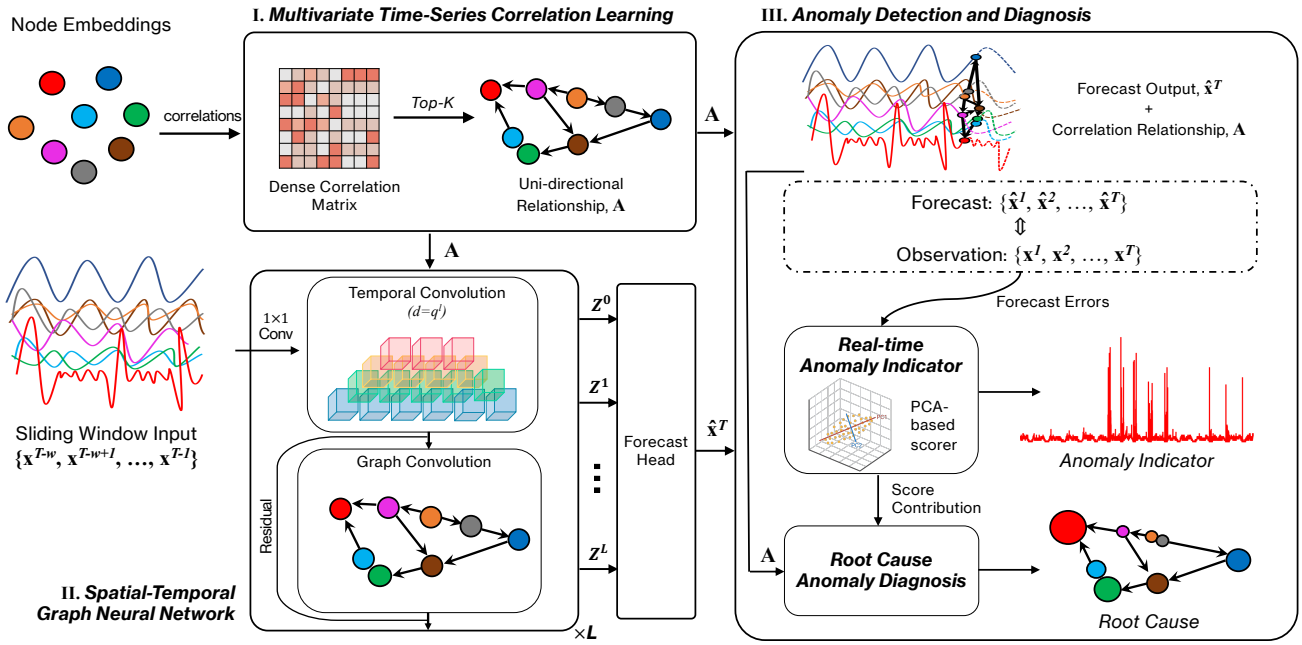

Fig. 2. Overall Framework of CST-GL. **I. MTCL** starts with a randomly initialized node embedding for each multivariate variable, and learn the underlying graph adjacency matrix, A, adaptatively with the entire model in an end-to-end manner. The adjacency matrix, A, will used by the graph convolution networks in the STGNN module. **II. STGNN**'s 1×1 convolution layer projects the sliding window input into the latent space. Then, the temporal and graph convolution networks are interlaced to capture rich spatial and temporal dependencies, representing one spatial. Skip connections, $Z^0 + Z^1 + \ldots + Z^L$, are incorporated to obtain hidden features that encapsulated the spatial-temporal patterns. Finally, the forecast head module projects the hidden features into a one-step forecast output, $\hat{x}^T$. **III. ADD.** a) Real-time Anomaly Indicator: takes in the current one-step forecast result and all observation-forecast pairs computed prior to timestamp T, computes normalized forecast deviation and outputs an anomaly indicator score in real-time using PCA-based scorer. b) Root Cause Anomaly Diagnosis: takes in result from Real-time Anomaly Indicator and learned pairwise correlation, A, from MTCL to enhance CST-GL's interpretability and identify the root causes of anomaly events.

between 0 to 1. To reduce the required computational cost and ease the optimization, we further mask elements with zeros in the learned graph adjacency matrix only except for the top-k closest neighbors of each node to make $\mathbf{A}$ sparse controlled by the hyper-parameter k. Specifically, for i-th row in $\mathbf{A}$, we have the following post-processing:

$$\begin{cases} \mathbf{topk} = argmax(\mathbf{A}[i,:], k), \\ \mathbf{A}[i, -\mathbf{topk}] = 0, \end{cases} \tag{2}$$

where $argmax(\cdot, k)$ returns the indices of top-k largest values in the input vector.

B. Spatial-Temporal Graph Neural Network

1) Graph Convolution Network: The spatial correlations between variables play a vital role in reflecting the intrinsic dynamics of multivariate time-series. Towards this, we design a spatial graph convolution layer to effectively pass messages between variables and their neighbors to exploit the underlying spatial patterns, allowing better encoding historical observations to make more precise and stable predictions, thus benefiting the downstream anomaly detection tasks. Similar to SGC [53] and the discrete of MTGODE [12], given an adjacency matrix $\mathbf{A}$ and the input (initial) states $\mathbf{H}_{in}$, we may characterize the graph propagation process as a combination of the feature propagation and linear transformation steps:

$$\begin{cases} \mathbf{H}^{k+1} = \widetilde{\mathbf{A}}\ \mathbf{H}^k, \ k \in \{0, \cdots, K\}, \\ \mathbf{H}_{out} = \mathbf{H}^K\ \mathbf{\Theta}, \end{cases} \tag{3}$$

where K denotes the graph propagation depth, and we have $\mathbf{H}^0 = \mathbf{H}_{in}$. Specifically, $\widetilde{\mathbf{A}}$ in the above equation denotes the normalized adjacency matrix, i.e., $\widetilde{\mathbf{A}} = \widetilde{\mathbf{D}}^{-1}(\mathbf{A} + \mathbf{I})$ and $\widetilde{\mathbf{D}}_{ii} = 1 + \sum_j \mathbf{A}_{ij}$.

However, Equation 3 suffers from two critical limitations. Firstly, although the above feature propagation design allows to recursively propagate latent node states along with a given graph structure, it is inevitable to see that node latent states become indistinguishable, i.e., converge to a single point, or known as over-smoothing, with an increase of the propagation depth K [12]. Secondly, only applying the linear transformation on the last node latent states $\mathbf{H}^K$ may be prone to errors [54], [23]. For example, if there are no correlations between variables in a multivariate time series, the feature propagation step will introduce noises to latent node states by blindly aggregating the neighbouring information. Thus, merely considering the linear transformation of the last propagated states hinders accurately modeling the latent spatial dynamics of a multivariate time series. To address these two limitations, we equipped the vanilla feature propagation in Equation 3 with a gating mechanism and replaced the upcoming linear mapping with an attentive transformation that mixes the information from multiple hops. We have the proposed graph convolution network defined as follows:

$$\begin{cases} \mathbf{H}^{k+1} = \beta\ \mathbf{H}_{in} + (1-\beta)\ \widetilde{\mathbf{A}}\ \mathbf{H}^k, \ k \in \{0, \cdots, K\}, \\ \mathbf{H}_{out} = \sum_{k=0}^{K} \mathbf{H}^k\ \mathbf{\Theta}^k, \end{cases} \tag{4}$$

where β controls to retain how much original node information to avoid the aforementioned over-smoothing issue. Regarding the attentive transformation in Equation 4, we can easily alleviate the problem mentioned in the above example by assigning a relevant large weight to the initial node states $\mathbf{H}^0$ and small weights to $\mathbf{H}^k$, where $k \in \{1, \cdots, K\}$.

As mentioned in Subsection IV-A, the learned pairwise dependencies are uni-directional. Thus, we refactor the final output of the graph convolution network as the summation of two transformations described in Equation 4, where the input latent node states are both $\mathbf{H}_{in}$ but with different adjacency matrices, i.e., $\mathbf{A}$ and $\mathbf{A}^\mathsf{T}$, to incorporate nodes' inflow and outflow information, respectively.

2) Temporal Convolution Network: Solving Equation 4 only allows to model the spatial dynamics at a certain point of time, where the rich temporal clues in multivariate time-series are neglected. To complete this missing information, we devise a simple yet effective temporal convolution network together with our graph convolution network to capture the expressive spatial and temporal patterns in historical observations.

We first introduce the composition of the proposed temporal convolution network, which consists of multiple residual dilated temporal convolution layers to extract and aggregate high-level temporal features in a non-recursive manner to avoid the shortcomings of Recurrent Neural Networks (RNNs), such as time-consuming iteration and gradient explosion [12], [23], [55]. Specifically, given a sequence of historical observations $\mathbf{X} = \{\mathbf{x}^1, \mathbf{x}^2, \cdots, \mathbf{x}^{T\text{-}1}\}$, we have the a temporal convolution layer defined as follows:

$$\mathbf{Z}^{l+1} = \mathcal{T}(\mathbf{Z}^l, Q^{l+1}) + TCN(\mathbf{Z}^l, \mathbf{\Phi}^l),\ l \in \{0, \cdots, L\}, \quad (5)$$

where the outputs of network is $\mathbf{Z}_{out} = \mathbf{Z}^L$, the input states $\mathbf{Z}^0$ are obtained by applying a linear mapping on $\mathbf{X}$, $TCN(\cdot, \mathbf{\Phi}^l)$ is an temporal convolution function parameterized by $\mathbf{\Phi}^l$ at the l-th layer, and $\mathcal{T}(\mathbf{Z}^l, Q^{l+1})$ denotes a truncate function that taking the last Q^{l+1} elements from $\mathbf{Z}^l$ along its sequence length axis. The underlying consideration is that the residual input $\mathbf{Z}^l$ has to be truncated to the length of $TCN(\mathbf{Z}^l, \mathbf{\Phi}^l)$ before adding them together because the sequence length of latent node states shrinks gradually as the underlying temporal information is aggregated after each temporal convolution layers. Specifically, we have $Q^{l+1} = Q^l - r^l \times (k-1)$ and $Q^1 = R - k + 1$, where k, r and R are kernel size, dilation factor, and receptive field (i.e., $R = L(k-1)+1$ and $R = 1+(k-1)(r^L-1)/(r-1)$ when $r = 1$ and $r > 1$). In terms of the design of temporal convolution function $TCN(\cdot, \mathbf{\Phi}^l)$, we follow [23] and adopt a gating mechanism to guide the information flow during the aggregation:

$$\text{TCN}(\mathbf{Z}^l, \mathbf{\Phi}^l) = f_C(\mathbf{Z}^l, \mathbf{\Phi}^l_c) \odot f_\mathcal{G}(\mathbf{Z}^l, \mathbf{\Phi}^l_g), \quad (6)$$

where $f_C(\cdot)$ and $f_\mathcal{G}(\cdot)$ are filtering and gating convolutions, and $\odot$ denotes the element-wise product. Specifically, we define these two convolutions in below:

$$\begin{cases} f_C(\mathbf{Z}^l, \mathbf{\Phi}^l_c) = tanh\big(\mathbf{W}^{1\times n}_{\mathbf{\Phi}^l_c} \star_\Delta \mathbf{Z}^l + \mathbf{b}^{1\times n}_{\mathbf{\Phi}^l_c}\big), \\ f_\mathcal{G}(\mathbf{Z}^l, \mathbf{\Phi}^l_g) = sigmoid\big(\mathbf{W}^{1\times n}_{\mathbf{\Phi}^l_g} \star_\Delta \mathbf{Z}^l + \mathbf{b}^{1\times n}_{\mathbf{\Phi}^l_g}\big). \end{cases} \quad (7)$$

In the above equation, $\star_\Delta$ denotes the dilated convolution operation, where the dilation $\Delta = r^l$. Specifically, to allow the model exploring multi-granular temporal clues and inspired by [23], $f_C(\mathbf{Z}^l, \mathbf{\Phi}^l_c)$ and $f_\mathcal{G}(\mathbf{Z}^l, \mathbf{\Phi}^l_g)$ consists of multiple convolution filters (e.g., $\mathbf{W}^{1\times n}_{\mathbf{\Phi}^l_c}$ and $\mathbf{b}^{1\times n}_{\mathbf{\Phi}^l_c}$) with width $n \in \{2, 3, 6, 7\}$. Since most of multivariate time-series data has some intrinsic periods [23], [55], such as 7, 14, 24, 28, and 30, the combination of the aforementioned kernel widths allows these common periods to be fully covered.

To simultaneously model spatial and temporal dynamics of a sequence of historical observations in a multivariate time-series, we construct a spatial-temporal graph neural network by combining the proposed spatial and temporal convolution networks, where the temporal and spatial convolution layers are interlaced, as shown in Figure 2. More precisely, a layer of the proposed spatial-temporal graph neural network is defined as follows by combining Equation 4 and 5:

$$\mathbf{Z}^{l+1} = \mathcal{T}(\mathbf{Z}^l, Q^{l+1}) + GCN\big(TCN(\mathbf{Z}^l, \mathbf{\Phi}^l), \mathbf{\Theta}\big), \quad (8)$$

where $GCN(\cdot, \mathbf{\Theta})$ and $TCN(\cdot, \mathbf{\Phi}^l)$ are defined in Equation 4 and 6. Finally, we take the output states $\mathbf{Z}_{out}$ to make a single-step-ahead forecasting via a multi-layer perceptron, i.e., $\hat{\mathbf{x}}^T = MLP(\mathbf{Z}_{out}, \mathbf{W}_{mlp})$, which forms a critical evidence to detect anomalies in a multivariate time-series.

C. Anomaly Detection and Diagnosis

1) Real-time Anomaly Indicator: With an effective joint learning of spatial and temporal dependencies from the non-anomalous data, it is expected that the anomalous observations in the test set deviate significantly from the learned patterns. Accordingly, to detect anomalous multivariate observations, we first compute the normalized forecasting deviation for every univariate variable and take the sum of the reconstructed univariate deviations to be the anomalous score for each multivariate observation.

Univariate variables within a multivariate time-series often possess vastly different attributes and scales. Consequently, we independently normalize each univariate deviation to preclude any single variable from dominating the aggregate multivariate deviation value. For every univariate variable, $\mathbf{x}^T_i$, we compute the absolute forecasting error, given by $\mathbf{e}^T_i = \big|\mathbf{x}^T_i - \hat{\mathbf{x}}^T_i\big|$, at the current timestamp T. This error is then normalized:

$$\tilde{\mathbf{e}}^T_i = \frac{\mathbf{e}^T_i - \boldsymbol{\mu}^T_i}{\boldsymbol{\sigma}^T_i}$$

where $\boldsymbol{\mu}^T_i$ and $\boldsymbol{\sigma}^T_i$ are the median and inter-quartile range (IQR) value across error values, $\{\mathbf{e}^{T-W_a}_i, \mathbf{e}^{T-W_a+1}_i, ..., \mathbf{e}^T_i\}$ in a sliding window where W_a represents the window length. Our normalization approach is an extension of [11] where we acquire the median and IQR values through a sliding window rather than from the entire test set observations. This modification allows us to detect anomalies in real-time as normalizing error at time T only relies on past observations and does not require future information as in [11].

After the normalization of each univariate variable, we obtain a multivariate normalized error vector, $\widetilde{\mathbf{E}}^T \in \mathbb{R}^{1\times \boldsymbol{N}}$, for the current timestamp. Although prior research suggests directly taking the summation [56] or maximum [11] to summarize the error vector into a single anomaly score at the current

timestamp, we propose to leverage Principal Component Analysis (PCA) as an intermediate step before aggregating the normalized errors into a final anomaly score.

In particular, after training the spatial-temporal graph neural network module, we compute the normalized errors in the validation set, $\widetilde{\mathbf{E}}_v$. We fit a PCA on the validation normalized errors by finding the validation mean vector $\overline{\mathbf{E}}_v = mean(\widetilde{\mathbf{E}}_v)$, their covariance matrix, $\mathbf{C}_v = cov(\widetilde{\mathbf{E}}_v)$, and the orthogonal eigenvectors, $\mathbf{U}$. $\mathbf{U}$ consists of N orthogonal eigenvectors associated with the N largest eigenvalues in the diagonal matrix, $\mathbf{\Lambda}$, decomposed from $\mathbf{C}_v = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{-1}$. With the fitted PCA, we reconstruct the normalized errors at current timestamp:

$$\begin{cases} \mathbf{P} = (\widetilde{\mathbf{E}}^T - \overline{\mathbf{E}}_v)\mathbf{U}^T \\ \widetilde{\mathbf{P}}, \widetilde{\mathbf{U}} = \mathbf{P}[:, L], \mathbf{U}[:, L] \\ \widetilde{\mathbf{E}}^T_{\text{PCA}} = \widetilde{\mathbf{P}}\,\widetilde{\mathbf{U}}^T + \overline{\mathbf{E}}_v \end{cases} \tag{9}$$

In the above equation, we first apply zero-centering to the normalized error at the current timestamp, $\widetilde{\mathbf{E}}^T$, by subtracting the mean validation error, $\overline{\mathbf{E}}_v$. We then project these results using validation eigenvectors, $\mathbf{U}$. Secondly, we keep only the first L principal components. Finally, we reconstruct the normalized errors, $\widetilde{\mathbf{E}}^T_{\text{PCA}}$, using the reduced L dimensions and revert the zero-centering deduction by adding the validation mean error. We set L as the number of components necessary to achieve a symmetric mean absolute percentage error (sMAPE) of less than 10% on the validation set.

With the reconstructed normalized error, we compute the final anomaly score at current timestamp by taking the L1 distance between the denoised and original normalized errors as the final anomaly score:

$$A(T) = \|\widetilde{\mathbf{E}}^T_{\text{PCA}} - \widetilde{\mathbf{E}}^T\|_1 \tag{10}$$

The incorporation of PCA addresses the fundamental problem posed by anomalies: the anomalous node variables have the potential to introduce bias into the learned embeddings within a neural network module, inadvertently affecting the forecast across all dimensions. This effect, corroborated by previous research [57], often leads to an unwarranted increase in forecast errors in variable nodes that are otherwise unaffected. Even in the absence of anomaly events, certain variable nodes may sporadically experience an upsurge in errors due to random fluctuations [11]. Such fluctuations can set off a cascade of effects across all nodes, echoing the impact of an actual anomalous event and potentially resulting in false positives. This unintended effect contributes to the degradation of accuracy in anomaly detection and diagnosis.

Previous methodologies have attempted to resolve this issue with the utilization of Markov Chain Monte Carlo (MCMC) imputation [1], [58]. This approach, however, is inefficient. In contrast, we propose the application of PCA to resolve these issues. PCA can efficiently project the normalized errors at current timestamp, $\widetilde{\mathbf{E}}^T$, onto the principal components of the validation errors, and subsequently reconstruct them as, $\widetilde{\mathbf{E}}^T_{\text{PCA}}$. This process effectively dampens common noise variations and pinpoints variables that contribute significantly to anomaly events. This identification is made possible because the variables that cannot be accurately reconstructed are more likely the true contributors to the anomaly events, thereby offering a more accurate depiction of the anomaly event itself.

Though PCA is capable of mitigating the inherent noise for more accurate representation, it is still the ability of STGNN to capture spatial-temporal patterns that holds the key to a comprehensive solution. The joint implementation of STGNN and PCA is instrumental in detecting and diagnosis anomalies, as we demonstrate in Section V: Experimental Study.

Last but not least, an anomaly indicator that can well signify the abnormality of a timestamp observation helps in informing system operators and determining an appropriate threshold by human experts to classify and detect anomalies. Nevertheless, for industrial operations that involve over thousands of multivariate time-series with distinct attributes such as warehousing robots [59], this approach does not scale well. To automate the threshold selection process, we classify current observation in the test set as anomalous if $A(T)$ in test set observation exceed the maximum $A(t)$ of all observations in the validation set. This non-parametric approach relies on CST-GL's ability in sufficiently capturing the spatial and temporal dependencies of a multivariate time-series data, so that any observations that exceeds the maximum anomaly score during normal time period (i.e., validation data) are in fact anomalies while those that do not exceed the maximum value are not anomalies.

2) Root Cause Anomaly Diagnosis: Since the final anomaly score is calculated as the linear combination of reconstruction errors, we can identify the root cause of anomalous events by ranking the univariate variables that contribute most significantly to the anomaly score. In practical scenarios, we determine the percentage contribution of each univariate variable to the final anomaly score. This approach would provide a more detailed perspective, allowing operators to more effectively identify the root cause of anomalous events.

In some cases, the top ranked variables that most contribute to the anomaly score may not be the root causes but are merely the symptoms [11], [24]. When the top ranked contributors are identified not to be the root cause, we further search for the variables that are most related to the top ranked contributors by aggregating the anomaly contribution scores of one-hop distance neighbors:

$$R_i(T) = \sum_{j \in N(i)} A_i(T) \tag{11}$$

where $A_i(T)$ represents the absolute error for the univariate node i as per Equation 10. $N(i)$ represents the neighbors of the univariate node i, which is based on the learned relation between variable pairs from the MTCL module.

As demonstrated in our experiments in Section V, this two-pronged approach ensures the systematic identification and diagnosis of (a) variables that exhibit abnormal behavior, and (b) variables closely related to these abnormal variables, as potential root causes of an anomaly event. The choice between directly ranking the variables based on the error contribution or based on the one-hop distance neighbors will largely depend on the nature of the anomalies.

V. EXPERIMENTAL STUDY

In this section, we conduct experiments to explore CST-GL's capabilities by answering following questions:

- **Overall Detection Performance.** Does our framework outperform baseline methods in the unsupervised, real-time anomaly detection task? How do the individual modules within CST-GL each contribute specifically to its ability to achieve anomaly detection and diagnosis?
- **Early Detection Performance.** Can CST-GL be adapted and generalized to commercial systems where early detection of anomaly events is often paramount?
- **Interpretability & Case Study.** Would CST-GL benefit system operators in detecting and diagnosing multivariate time-series anomaly events in an interpretable manner?

A. Experimental Settings

In this subsection, we introduce the settings of our experiments, including datasets, baseline methods and parameter settings and computing infrastructures.

1) Datasets: We evaluate CST-GL on three widely used benchmark datasets for multivariate time-series anomaly detection: SWaT, WADI and SMD. The statistics of these datasets are demonstrated in Table I, and the detailed descriptions are given as follows:

TABLE I
THE STATISTICS OF THE DATASETS.

Dataset	♯ channels	♯ train	♯ test	anomalies
SWaT	51	47,515	44,986	11.97%
WADI	127	118,795	17,275	5.99%
SMD	38	304,168	304,174	5.84%

- **SWaT [4]** is a scaled-down version of a real-world industrial water treatment plant initiated by Singapore's Public Utility Board. The dataset comprises 7 days of normal operations (train data) and 4 days of attack scenarios (test data). The anomaly labels represent the attacks that are conducted at different intervals in the test set.
- **WADI [5]** is an extension of the SWaT dataset with a larger number of water pipelines, storage, and treatment systems, representing a more complete and realistic water treatment dataset [11]. The train set of WADI is two weeks of normal operation while the test set is a 2 days attack scenario. Following the original author's implementation [11], we removed the first 21,600 samples and down-sampled SWaT and WADI to one measurement every 10 seconds by taking the median values.
- **SMD [3]** is a real-world server machine dataset collected by a large Internet company. SMD contains time-series data of servers, each with 38 multivariate variables. It is divided into train and test sets of equal size. The original SMD dataset did not have any preprocessing applied to remove servers experiencing concept drift. This was subsequently addressed by the original authors in [1] to remove servers suffering from concept drift. Following the subsequent work, the reported results in this section takes the average scores computed for the 12 servers that do not suffer from concept drift.

2) Baselines: We compare our CST-GL with five standard multi-dimensional anomaly detection methods that do not take temporal dependencies into consideration and six recently proposed frameworks designed specifically for multivariate time-series anomaly detection. Baseline descriptions and implementation details are provided in the Appendix.

The five standard multi-dimensional anomaly detection methods are Raw Signal [24], **PCA**, **AutoEncoder**, **Kmeans** and **DAGMM** [60]. **Raw Signal** is a simple baseline model that reconstructs any signal to zero, resulting in an error equivalent to the normalized signals themselves. Using the normalized signals, a Gaussian scoring function is employed to compute the negative log-likelihood of observing these signal values at each timestamp. This model provides insights into the nature and difficulty of the benchmark dataset.

The six state-of-the-art frameworks for multivariate time-series anomaly detection are **LSTM-VAE** [26], **OmniAnomaly** [3], **USAD** [61], **MTAD-GAT** [56], **GDN** [11] and **InterFusion** [1]. Notably, **InterFusion**, an extension of **OmniAnomaly**, is the state-of-the-art RNN framework, while **MTAD-GAT** and **GDN** are the state-of-the-art GNN baselines for the multivariate time-series anomaly detection task.

3) Parameter Settings: We train our model for 20 epochs with a batch size of 64, Adam optimizer is applied to optimize CST-GL with learning rate of 3×10^{-4} and $(\beta_1, \beta_2) = (0.9, 0.999)$. Following previous works [3], [1], validation set ratio for SWaT, WADI and SMD are 0.1, 0.1 and 0.3 respectively. We set sliding window length, w, to be 5, 5, and 100 for SWaT, WADI and SMD as suggested by the original papers [3], [11]. We define the hyperparameter search space as shown in Appendix, and select the hyperparameters that achieve lowest average root-mean-square error in the validation set.

After hyperparameter search, the MTCL module has neighbour size, k, set to be 15, 30 and 10 for SWaT, WADI and SMD respectively. Across all datasets, the correlation learning module has a node dimension of 256, the retain ratio of 0.1 and saturation rate of 20. The graph convolution network and the temporal convolution network modules both have 16 output dimensions. The skip connection layers all have 32 output dimensions. We use 2 graph and temporal module layers. Lastly, for the number of principal components, we set it automatically based on the number required to achieve less than 10% sMAPE on the validation set.

4) Computing Infrastructures: Our proposed learning framework is implemented using PyTorch 1.7.0. The computation of F1 score, ROC and PRC is acquired by Scikit-learn. All experiments are conducted on a personal computer with Ubuntu 20.04 OS, with an NVIDIA Tesla T4 GPU, a 2.20GHz Intel Xeon CPU, and 12.7 GB RAM. For model comparison with a single and five experimental runs, we use seed 0 and 0-4 respectively. The empirical computational complexity for all methods requiring non-trivial training costs is detailed in the Appendix.

B. Overall Detection Performance

As many baseline methods do not incorporate a threshold selection mechanism [60], [26], [61], [1], we compare

TABLE II
AVERAGE AUC PERFORMANCE (± STANDARD DEVIATION) OF FIVE EXPERIMENTAL RUNS ON THREE BENCHMARK DATASETS. THE BEST AND SECOND BEST PERFORMING METHOD IN EACH EXPERIMENT IS IN BOLD AND UNDERLINED RESPECTIVELY.

	SWaT		**WADI**		**SMD**	
	ROC	PRC	ROC	PRC	ROC	PRC
Raw Signal	0.8218 (0.0000)	0.5661 (0.0000)	0.6544 (0.0000)	0.1117 (0.0000)	0.7295 (0.0000)	0.1775 (0.0000)
PCA	0.8257 (0.0000)	0.7298 (0.0000)	0.5597 (0.0000)	0.2731 (0.0000)	0.6742 (0.0000)	0.2189 (0.0000)
AutoEncoder	0.8311 (0.0088)	0.7224 (0.0094)	0.5291 (0.0285)	0.2210 (0.0205)	0.8270 (0.0008)	0.4388 (0.0046)
Kmeans	0.7391 (0.0000)	0.2418 (0.0000)	0.6030 (0.0000)	0.1158 (0.0000)	0.5855 (0.0000)	0.1308 (0.0000)
DAGMM	0.7219 (0.0473)	0.2630 (0.0932)	0.5375 (0.0388)	0.1315 (0.0396)	0.7489 (0.0111)	0.2522 (0.0095)
LSTM-VAE	0.8016 (0.0016)	0.6936 (0.0047)	0.5165 (0.0322)	0.1486 (0.0476)	0.7802 (0.0105)	0.3153 (0.0306)
OmniAnomaly	0.8256 (0.0211)	0.7061 (0.0066)	0.5520 (0.0092)	0.2184 (0.0023)	0.8265 (0.0114)	0.4575 (0.0197)
USAD	0.8213 (0.0056)	0.7087 (0.0055)	0.5535 (0.0103)	0.1945 (0.0008)	0.7888 (0.0077)	0.4686 (0.0011)
MTAD-GAT	0.8261 (0.0040)	0.7176 (0.0043)	0.4119 (0.0259)	0.0729 (0.0013)	0.8576 (0.0035)	0.5057 (0.0082)
GDN	0.8124 (0.0177)	0.7135 (0.0035)	0.4725 (0.0056)	0.0521 (0.0070)	0.8443 (0.0150)	0.4684 (0.0142)
InterFusion	0.8409 (0.0132)	0.6970 (0.0844)	0.6388 (0.0311)	0.3775 (0.0319)	0.8374 (0.0215)	0.4265 (0.0351)
CST-GL	**0.8520 (0.0022)**	**0.7628 (0.0032)**	**0.8283 (0.0179)**	**0.5477 (0.0197)**	**0.8604 (0.0131)**	**0.5132 (0.0273)**

model performances using the Receiver Operating Characteristic (ROC) and Precision-Recall Curve (PRC) Area Under the Curve (AUC) scores by treating every timestamp as an independent observation to be classified as an anomaly or not. Under this pointwise approach, a model is required to predict the occurrence of anomaly events across the entire time-series, including when they have started and ended. The closer the ROC and PRC score is to 1, the better a model is at scoring and differentiating anomalous and non-anomalous time points. For comparison between

1) Baseline Comparison: The PRC and AUC results are summarized in Table II and we observe that:

- **Proposed Framework.** CST-GL showed superior performances against all the other baselines with an average outperformance of 7.16 and 8.30 percentage points against the next best baseline for the ROC and PRC scores respectively. It also achieved high performance with relatively low variability and, in the case of WADI's PRC values, the performance gain is greater than 45% when compared to the next best result. The experimental result in Table II demonstrates CST-GL's superior performance in providing a representative anomaly indicator to inform and alert system operators. It also aids experts in deciding on an appropriate threshold for human intervention as the anomaly scores for anomalous and non-anomalous timepoints are well separated.
- **Temporal Dependency.** On average, baseline methods that consider temporal information achieve higher ROC and PRC results, validating that temporal information is paramount for detecting anomalies in multivariate time-series. The importance of effective learning of temporal cues is also evident by the performance of GDN, which did not address the temporal dependencies between time-series observations directly. Despite explicitly learning spatial correlation between multivariate variable pairs, the GDN model is less effective when adapted to the unsupervised, *real-time* anomaly detection task.
- **Spatial Pairwise Correlation.** As LSTM-VAE, OmniAnomaly and USAD do not directly capture the underlying pairwise inter-dependence among the multivariate time-series variables, they performed poorer than InterFusion and CST-GL. Similar to our framework, InterFusion directly addresses the spatial-temporal dependencies by learning dual-view latent embeddings. Nonetheless, as InterFusion's latent embedding only encapsulates spatial correlation within a global hidden state, they do not explicitly model the relationships between variable pairs. We conjuncture that successful capturing of spatial correlation dependencies requires an *explicit* graphical modeling of relationships between the multivariate variables as it evidently improves the effectiveness of a time-series anomaly detection model.

2) Ablation Study: We conduct an ablation study on SWaT and WADI to validate how various modules of CST-GL contribute to its multivariate time-series anomaly detection performance. We implement different variants of CST-GL with modifications to the following modules:

- **w/o MTCL:** CST-GL without Multivariate time-series Correlation Learning. We replace the learned adjacency matrix, **A**, with a complete digraph adjacency matrix and remove MTCL.
- **w/o GCN:** CST-GL without the Graph Convolution Network. We remove the GCN module (including MTCL) and replace it with a linear layer.
- **mod. TCN:** CST-GL with modified Temporal Convolution Network. We modify TCN to nullify its ability in capturing multi-granular temporal clues by replacing the multi-convolution filters with a single 1x1 filter.
- **w/o PCA:** CST-GL without the PCA-based anomaly scoring module. We replace our the PCA module with standard Gaussian scoring function [24]. The Gaussian scoring function would correspond to the Raw Signal in Table II, but the input for this function is the forecast error from the STGNN in CST-GL.
- **w/o STGNN:** CST-GL without the STGNN. This is equivalent to the PCA model in Table II.

Focusing on MTCL module, we see a drop in performance this module is removed (**w/o MTCL**), and a a complete digraph adjacency matrix is used for modelling interactions between variables. Importantly, the degradation of performance is notably more pronounced for the WADI dataset. We hypothesize that the noise from unimportant neighbouring nodes is more

TABLE III
ABLATION STUDY - AVERAGE AUC PERFORMANCE
(± STANDARD DEVIATION)

	SWaT		WADI	
	ROC	PRC	ROC	PRC
CST-GL	**0.8520 (0.0022)**	**0.7628 (0.0032)**	**0.8283 (0.0179)**	**0.5477 (0.0197)**
w/o MTCL	0.8457 (0.0181)	0.7218 (0.0240)	0.7832 (0.0046)	0.4739 (0.0145)
w/o GCN	0.8401 (0.0037)	0.6800 (0.0459)	0.7854 (0.0064)	0.4688 (0.0103)
mod. TCN	0.8446 (0.0049)	0.7324 (0.0124)	0.7849 (0.0104)	0.4886 (0.0148)
w/o PCA	0.8610 (0.0092)	0.7509 (0.0152)	0.7017 (0.0375)	0.4105 (0.0625)
w/o STGNN	0.8257 (0.0000)	0.7298 (0.0000)	0.5597 (0.0000)	0.2731 (0.0000)

pronounced when the GCN propagate information among the variables under the WADI with 127 number of multivariate variables, as compared to SWaT that only has 51.

Next, we scrutinize the effects of modifications to the STGNN. We observe that the exclusion of the GCN module (**w/o GCN**) significantly degrades the anomaly detection results. This is consistent with previous studies [1], [11] as modelling of the pairwise correlations among variables can enable information flow among the interdependent univariate variable nodes, thereby improving the performance of detecting anomaly events. Conversely, when we modify the TCN (**mod. TCN)** within CST-GL, it also leads to decline in performance. This can be attributed to the fact that the temporal dependency of the multivariate time-series data is less effectively captured.

When the PCA-based anomaly scoring module is replaced by a Gaussian scoring function [24] (**w/o PCA**), we note a reduction in performance in the WADI dataset. This performance drop can be attributed to the Gaussian function's lack of robust denoising capabilities, an area where PCA excels. Despite this, the CST-GL still outperforms the established baselines.

Finally, the removal of the STGNN (**w/o STGNN**), leaving only PCA model, significantly reduces performance. This underscores the crucial role of the STGNN. While PCA can lessen inherent noise for improved representation, it is the capacity of the STGNN to recognize spatial-temporal patterns that forms a comprehensive solution for anomaly detection.

3) Automatic Thresholding Mechanism: Our framework incorporates an automatic thresholding mechanism where the maximum anomaly score in the validation set is taken as the threshold without a need for human experts in determining the optimal threshold. Table IV shows the best F1 score achieved through an enumerative search of global optimal threshold against the F1 score of our automatic thresholding mechanism.

The non-parametric threshold selection of CST-GL, despite its simplicity, effectively captures the spatial-temporal dependencies of multivariate time-series during the normal period (i.e., training set). This capability allows for a notable degree of separation between anomalous and non-anomalous timepoints in the test set, as evidenced by promising F1 scores. However, it is important to note that the effectiveness of the automatic threshold is most pronounced on the SWaT dataset, and exhibits some performance drops on WADI and SMD. Moving forward, we aim to refine the thresholding process to close the gap between the automatically determined threshold and the threshold determined using best-F1 scores across a broader range of scenarios.

TABLE IV
F1 SCORE OF AUTOMATIC THRESHOLD VS. OPTIMAL THRESHOLD.

Dataset	Automatic	Optimal
SWaT	0.7486 (0.0071)	0.7529 (0.0014)
WADI	0.4927 (0.0487)	0.5711 (0.0112)
SMD	0.4065 (0.0283)	0.5225 (0.0011)

C. Early Detection Performance

As time-series anomaly events usually form contiguous anomaly segments, previous works have argued that detecting anomalies within any subset of a ground truth anomaly segment is sufficient in real-world scenarios. Based on this notion, they evaluated multivariate time-series anomaly detection models using point-adjusted (PA) approach [3], [61], [1]. Under this approach, if any timestamp in a contiguous anomaly segment with M_a timestamps are correctly detected as an anomaly, the PA approach considers the entire anomaly segment as correctly predicted with M_a true positives [57]. However, since any detection within a contiguous anomaly segment is treated equally, *the PA approach does not reward early detections in an anomalous segment* [24], [62], [63]. Nevertheless, early detection of anomaly events is often crucial in a wide range of practical applications and a model which can detect anomaly events early will have significant value in real-world settings [64].

To evaluate early detection ability of CST-GL and baseline methods, we adopt the metric suggested by [65], where detection of contiguous anomaly segment is only treated as true positives, if and only if an anomaly point is detected correctly and its timestamp is at most δ steps after the first anomaly of the contiguous anomaly segment. For example, $\delta = 0$ would equate to identifying an anomaly segment as early as possible without any delays and $\delta = 60$ for a time-series with second-interval would equate to detecting anomaly segment within a minute after the first anomaly timestamp. As δ becomes sufficiently large (i.e., the delay constraint is removed), the results of the early detection PA approach will be the same as the original PA approach.

In this work, we evaluate models' early detection ability with delay 0, 1, 5, 10, 20, 30 and 60 minutes. Following previous work [3], [61], [1] in computing model's anomaly scoring ability, we report the best F1 score for each delay. Based on Table V, VI and VII, we observe the following:

- **Immediate Detection.** CST-GL showed a substantial advantage against the next baseline when $\delta = 0$ where the performance improvement is 86.27%, 43.44% and 2.75% for SWaT, WADI and SMD respectively. This indicates that our model significantly outperform the simple and state-of-the-art baselines in early detection of multivariate time-series anomaly events.
- **Practicality.** On all three benchmark datasets, our proposed framework performed best across all delays, δ, with the exception of 5 and 10 minutes under the SMD dataset. While model performance gaps decrease as δ is increased, our framework remains state-of-the-art even when delay is set to be 60 minutes. These results suggest that our

TABLE V
SWAT - BEST F1 RESULTS AT DIFFERENT DELAYS.
THE BEST AND SECOND BEST PERFORMING METHOD IN EACH EXPERIMENT IS IN BOLD AND UNDERLINED RESPECTIVELY.

Methods	No delay	1 min	5 min	10 min	20 min	30 min	60 min
Raw Signal	0.2444	0.2661	0.2713	0.2729	0.2911	0.2967	0.7652
PCA	0.3049	0.3264	0.3608	0.4610	0.4684	0.4684	0.6874
AutoEncoder	0.3219	0.3264	0.3369	0.3453	0.3453	0.7473	0.7750
Kmeans	0.0557	0.0689	0.0864	0.1062	0.1280	0.1443	0.4751
DAGMM	0.3738	0.3969	0.4120	0.4152	0.4578	0.6822	0.6822
LSTM-VAE	0.3048	0.3088	0.3100	0.3100	0.3144	0.3194	0.5604
OmniAnomaly	0.3200	0.3217	0.3234	0.3256	0.3295	0.3334	0.5800
USAD	0.3072	0.3169	0.3218	0.3294	0.3382	0.5208	0.5240
MTAD-GAT	0.2192	0.2238	0.4043	0.4632	0.4938	0.7999	0.7999
GDN	0.3052	0.3125	0.3134	0.3152	0.3162	0.3182	0.3183
InterFusion	0.4067	0.4803	0.5139	0.5406	0.5465	0.6031	0.6031
CST-GL	**0.7576**	**0.7705**	**0.7953**	**0.7972**	**0.8093**	**0.8489**	**0.8507**

TABLE VI
WADI - BEST F1 RESULTS AT DIFFERENT DELAYS.
THE BEST AND SECOND BEST PERFORMING METHOD IN EACH EXPERIMENT IS IN BOLD AND UNDERLINED RESPECTIVELY.

Methods	No delay	1 min	5 min	10 min	20 min	30 min	60 min
Raw Signal	0.1948	0.2463	0.3009	0.5306	0.5306	0.5306	0.5306
PCA	0.1172	0.2692	0.2816	0.3144	0.3144	0.3144	0.3144
AutoEncoder	0.2307	0.3892	0.3892	0.3892	0.3892	0.3892	0.3892
Kmeans	0.0258	0.1239	0.3575	0.4814	0.4814	0.4814	0.4814
DAGMM	0.1368	0.2715	0.2715	0.2715	0.2715	0.4745	0.4745
LSTM-VAE	0.1300	0.1806	0.2712	0.2726	0.3549	0.3549	0.3549
OmniAnomaly	0.2167	0.2957	0.2957	0.3549	0.3549	0.3549	0.3549
USAD	0.1280	0.2204	0.2714	0.2728	0.3469	0.3469	0.3469
MTAD-GAT	0.1212	0.1286	0.1317	0.1691	0.4109	0.4109	0.4109
GDN	0.1454	0.1539	0.2576	0.2946	0.2946	0.2946	0.2946
InterFusion	0.2921	0.3740	0.3754	0.3754	0.5704	0.5774	0.5774
CST-GL	**0.4190**	**0.5796**	**0.7716**	**0.7716**	**0.7716**	**0.7716**	**0.7716**

TABLE VII
SMD - BEST F1 RESULTS AT DIFFERENT DELAYS.
THE BEST AND SECOND BEST PERFORMING METHOD IN EACH EXPERIMENT IS IN BOLD AND UNDERLINED RESPECTIVELY.

Methods	No delay	1 min	5 min	10 min	20 min	30 min	60 min
Raw Signal	0.4127	0.4648	0.5381	0.6757	0.7834	0.8068	0.8965
PCA	0.3102	0.3762	0.4637	0.5828	0.5956	0.6399	0.7000
AutoEncoder	0.4401	0.5444	0.6258	0.7279	0.7670	0.8004	0.8194
Kmeans	0.1179	0.1472	0.1923	0.1986	0.2407	0.2407	0.3527
DAGMM	0.3304	0.4117	0.5652	0.6145	0.6779	0.7060	0.7721
LSTM-VAE	0.3475	0.3639	0.3812	0.4033	0.5793	0.5989	0.6658
OmniAnomaly	0.4649	0.5217	0.6352	0.8215	0.8288	0.8457	0.9093
USAD	0.3592	0.4570	0.7036	0.7555	0.7677	0.0892	0.8484
MTAD-GAT	0.4688	0.5359	0.6544	0.7695	0.8355	0.8503	0.9088
GDN	0.4734	0.5120	0.5427	0.7191	0.7269	0.7727	0.8533
InterFusion	0.4275	0.4958	0.6309	0.7392	0.7778	0.8013	0.8954
CST-GL	**0.5070**	**0.6234**	**0.7917**	**0.8420**	**0.8548**	**0.9095**	**0.9335**

anomaly detection model has the greatest ability at detecting not only anomalous events that require immediate attention but also anomalous events that is less hurried. CST-GL can thus be potentially applied across a wide-range of practical applications and is dependable in a diverse range of real-world operational requirements.

- **Baseline Comparison.** Consistent with overall detection results, InterFusion that learns the temporal dependencies and inter-dependence between univariate variables achieved the second best results in the early anomaly detection task. We further observe that other baseline methods which do not address both dependencies have greater variability across different delays and datasets, validating that effective learning of temporal and pairwise inter-dependence univariate time-series helps in generalizing a detection model across different tasks and datasets.

D. Root Cause Anomaly Diagnosis

In accordance with the approach suggested by Garg et al. [24], we gauge the anomaly diagnosis performance of all models using the Root-cause top 3 metric (RC-Top3). The RC-Top3 measures instances where at least one of the genuine

TABLE VIII
PERFORMANCE ON ROOT CAUSE DIAGNOSIS.
AVERAGE RC-TOP3 (± STANDARD DEVIATION).

Methods	SWaT	WaDI	SMD
PCA	0.3714 (0.0000)	0.3846 (0.0000)	0.6960 (0.0000)
AutoEncoder	0.3543 (0.0433)	0.5230 (0.0644)	0.7376 (0.0226)
Kmeans	0.3714 (0.0000)	0.4615 (0.0000)	0.6475 (0.0000)
DAGMM	0.1429 (0.0535)	0.1538 (0.0000)	0.4933 (0.0681)
LSTM-VAE	0.3143 (0.0626)	0.4615 (0.1288)	0.5186 (0.0136)
OmniAnomaly	0.3714 (0.0000)	0.4768 (0.0344)	**0.8608 (0.0044)**
USAD	0.0400 (0.0139)	0.1846 (0.0377)	0.4380 (0.0225)
MTAD-GAT	0.3829 (0.0433)	0.5385 (0.0543)	0.6894 (0.0093)
GDN	0.3047 (0.0719)	0.4923 (0.0421)	0.7307 (0.0444)
InterFusion	0.3086 (0.0313)	0.5846 (0.0422)	0.4568 (0.0169)
+ MCMC	0.2171 (0.0433)	0.5077 (0.0421)	0.7747 (0.0132)
Raw Signal	0.1143 (0.0000)	0.4615 (0.0000)	0.8107 (0.0000)
+ MTCL-Graph	0.1200 (0.0114)	0.5231 (0.0576)	0.6723 (0.0460)
CST-GL	0.4286 (0.0452)	0.4307 (0.0377)	0.8532 (0.0151)
+ MTCL-Graph	**0.4914 (0.0313)**	**0.6154 (0.0308)**	0.7776 (0.0430)

causes is identified among the top three causes as determined by the detection model. For all models, we provide the mean performance along with its standard deviation. Since InterFusion utilizes MCMC imputation on the original reconstruction to diagnose root causes, we present results both with and without MCMC imputation. As argued by the original author [1], anomalous node variables have the potential to introduce bias into the learned embeddings within their network module, which could create undesirable noise. MCMC imputation can help to dampen this noise, similar to the role of the PCA-based scorer in CST−GL.

For CST−GL, we report the diagnosis performance using the ranking derived from the PCA-based method, as well as the ranking obtained after aggregating the anomaly scores from its one-hop neighbour (CST−GL +MTCL-Graph). The latter approach leverages the learned relationships between variables from the MTCL module to diagnose root causes. This approach considers that anomalous behavior exhibited by some variables may merely be symptomatic, while the root cause could be attributed to closely related variables. To further assess the benefits of MTCL, we also present the diagnostic results of the raw signal using the MTCL-Graph (Raw Signal+MTCL-Graph). This serves to evaluate the effectiveness of MTCL in facilitating the diagnosis of anomalies, even when the raw signal alone is used.

As evidenced in Table VIII, CST−GL excels in identifying root causes of anomalies on SWaT and WADI, using MTCL-Graph. It comes a close second to OmniAnomaly on SMD without MTCL-Graph. As detailed in Section IV-C2, the decision to diagnose root cause based directly on error contribution or one-hop distance neighbors using MTCL-Graph depends on the anomaly characteristics. SWaT and WADI often have symptomatic variables that exhibit abnormal behaviours but not causative [11], [24]. These variables were influenced by true root causes that exhibit normal behaviours. Thus, unearthing the true root causes requires the explicit identification of variables closely associated with symptomatic variables. This task is effectively achieved with our MTCL module.

Contrastingly, in SMD, variables that display abnormal behaviors are indeed the root causes themselves. As such, the MTCL-Graph does not provide any added benefit; instead, it is more advantageous to directly evaluate the anomaly score from the PCA-based scorer. These observations align with the diagnosis results of InterFusion. When MCMC imputation is employed, the effects of anomalous noise are mitigated, enabling InterFusion to accurately diagnose the root causes in SMD. However, MCMC imputation does not enhance results in SWaT and WADI, as it reduces the impacts of abnormal behaviors transferred to related variables. Consequently, MCMC imputation softens the anomalous effects on the actual root cause variables that do not themselves exhibit abnormal behaviors. To consolidate, we also demonstrate that MTCL-Graph improves the ability in diagnosing the root cause directly from raw signals alone in SWaT and WADI, but not in SMD.

On the whole, CST−GL with the aid of MTCL-Graph provides a comprehensive and actionable tool for for operators in detecting and diagnosing anomaly events.

E. Case Study in Practice

To showcase CST−GL's implementation under real-world scenarios, we conduct time-series anomaly detection case studies on WADI's Water Distribution System where the root cause of the anomaly event is known.

a) Background: The water distribution process in WADI is segmented into three sub-processes: P1, P2 and P3. P1 involves water intake and water quality management, P2 takes in water from P1 and supplies it to the consumers and P3 returns excess water back to P1. To monitor and automate the system effectively, 127 sensors are installed. The sensors within each sub-process are intimately linked to monitor and automate the water distribution sub-process. Nonetheless, any attacks on a single sub-process will have a cascading effect on the entire water distribution system. In this experimental setting, CST−GL is required to detect malicious attacks by ill-intentioned parties that have access to the system control from October 9, to October 11, 2017. CST−GL is provided with 14 days of normal multi-sensors data from September 25, to October 9, 2017, to train the model. No labels or information related to the attacks are given during training and CST−GL is required to detect the anomalies in an unsupervised manner.

b) Stealth Attack: At 10:55 a.m. on December 10, 2017, the attacker launch a 29-minute stealthy attack on WADI to drain an elevated reservoir by changing the reading seen by water quality sensor, 1_AIT_001 (i.e., the root cause of attack). Further, the attacker cleverly manipulates the root cause sensor to make the event undetectable. Consequently, determining the root cause of this attack is non-trivial given that the attacker has extensive knowledge about the WADI system and is deliberately hiding the root cause.

c) Proposed Framework In Action: The following describes CST−GL's real-time anomaly detection mechanisms.

- **Automated Early Detection.** Relying on the automated thresholding mechanism, CST−GL alerted the human operators at 10:58 a.m. (less than 4 minutes after the

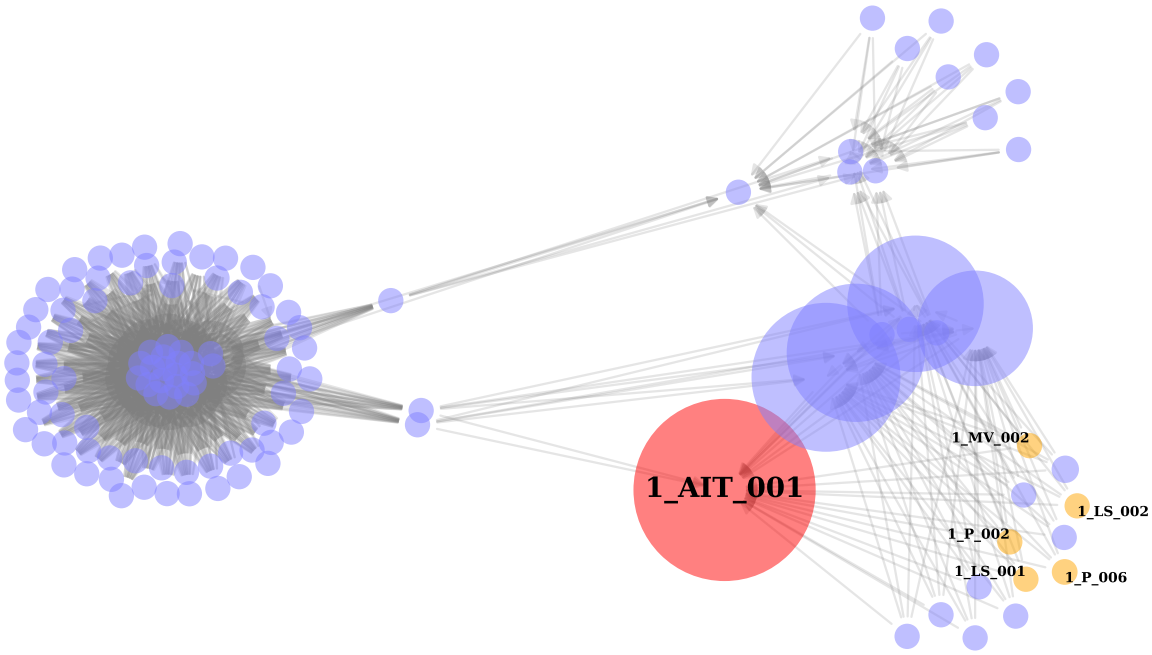

Fig. 3. Root cause analysis of stealth attack on Water Distribution System. Size of nodes represents the computed ranking of nodes as described in section IV-C2. Red node represents the highest ranked sensor and orange nodes represent the top 5 sensors that contributed the majority of anomaly score during the first 5 minutes of attack. Apart from the highest-ranked red node, four blue nodes indicate other potential sources of the anomaly, with their sizes signifying relative importance. However, the primary node associated with the attack is the first red one, not the four blue nodes. The directed arrows represent the uni-directional relationships that CST-GL has learned using the MTCL module and correspond to spatial dependencies between different sensors.

attack) that a possible anomaly event has occurred and emergency intervention is required. During the attack period, the anomaly score remained high, continuously warning operators about the urgency of the attack event.

- **Root Cause Identification with Learned Relation.** Looking at CST-GL's system outputs, the human operators see 5 sensors from sub-process P1 to contribute the substantial majority of anomaly scores during this period: 1_MV_002, 1_P_002, 1_P_006, 1_LS_001 and 1_LS_002. After inspecting all 5 sensors, it is found that they are not the root cause but are merely the symptoms of this attack. Nevertheless, they are very likely to be related to the root cause of the stealthy attack event. Thus, the sensors most closely related to the five sensors are immediately ranked based on the CST-GL's learned relation between variable pairs. The aggregated scores of one-hop distance neighbors in sub-process P1 ranks 1_AIT_001 as the sensor most associated with the 5 aforementioned sensors, as illustrated in Figure 3. The root cause of the attack is thus successfully identified after inspecting merely 6 out of 127 sensors.
- **Informativeness with Pointwise Detection.** The original WADI dataset assumes no human intervention and the 29-minute stealth attack ended at 11:24 a.m. While CST-GL continues to inform the human operator that an anomaly event is ongoing after this period due to imperfect prediction and lag effects from the stealth attack, it is able to provide continuous affirmations after 11:26 a.m. that the attack has ended (i.e., the timestamps after this period are labeled as non-anomalous), less than 2 minutes of lag time. This not only allows system operator to make decisions with informed knowledge but also direct efforts on exploring the data within the most relevant time frame to thoroughly understand the anomaly events that have already occurred.

In another WADI anomaly event, a flow sensor, 1_FIT_001, is attacked via false readings. To detect the root cause of this attack is again non-trivial because the false readings are within the normal range of this sensor [11]. Following the implementation above, CST-GL is able to alert system operators that an anomalous event has occurred after just *10 seconds* of the attack. Similarly, the top sensors that contributed to the anomaly score are again found not to be the the root cause. Through aggregated scores of the learned relations between sensors, CST-GL ranked the root cause sensor, 1_FIT_001, as the third most possible sensor to be the root cause, correctly identifying the root cause after inspecting 8 out of 127 sensors. Lastly, CST-GL informs that the anomaly event has ended within the precision of ±1 minute.

d) Summary: The case studies demonstrate CST-GL's ability in (1) detecting anomalous event early, (2) significantly reducing the search range for human operators to identify

the root cause by localizing the relevant variables and (3) informing operators about the duration of anomaly events with reasonable precision. Importantly, it also illustrates that joint learning of spatial-temporal and pairwise correlation relational dependencies can help a multivariate time-series anomaly detection model to detect and diagnose anomaly events, significantly reduce the destructive impact of such events on industrial systems.

Moving forward, our aim is to enhance CST-GL for a broader range of applications by addressing the issues of concept drift and missing values. In terms of concept drift, we plan to implement mechanisms that can detect and quantify the magnitude of data drift, thus facilitating necessary adjustments to the model in line with evolving data distributions [66], [67]. For handling missing values, we intend to assess the robustness of `CST-GL` by employing standard interpolation and imputation algorithms [68]. Furthermore, we aspire to incorporate spatial-temporal graph controlled differential equations [12], inherently suited to scenarios involving missing values.

VI. Conclusion

In this work, we proposed a novel framework for multivariate time series anomaly detection. Our model, `CST-GL`, explicitly learns pairwise correlations between variables pairs of multivariate time-series data, jointly capture spatial-temporal dependencies and effectively detect anomaly events when the behaviour of time-series data deviate from the non-anomalous patterns. Experiments on three real-world datasets showed that `CST-GL` outperformed eleven baselines in general and early detection settings. `CST-GL` also enables interpretation and root cause diagnosis of anomaly events in multivariate time-series data, paving the way for STGNN-based methods to be implemented in real-world applications. In the future, we will study generalizability of `CST-GL` in dynamic and missing values scenarios together with the trustworthiness of our GNN model [34] through the perspectives of robustness and explainability. We will also look into how large language models can enhance graph learning [69] for time series data.

Appendix

Our appendix primarily provides details of the experimental settings to ensure the reproducibility of our work. **A1. Implementation of Baseline**, details the hyperparameters of the baselines we reproduced in our work, while **A2. Empirical Computational Complexity** provides information about the time complexities of the baselines and our model, `CST-GL`. Lastly, **A3. `CST-GL` Hyperparameter Search Space** outlines the search space that we use to set the hyperparameters of `CST-GL`, based on the combination of parameters that achieve the lowest average Root-Mean-Square-Error (RMSE) in the validation set.

A1. Implementation of Baseline

- **Raw Signal** [24] is a trivial baseline model that reconstructs any signal to zero, resulting in an error that equates to the normalized signals themselves. On the normalized signals, a Gaussian scoring function is utilized to compute the negative log-likelihood of observing these signal values in each timestamp. This baseline is reproduced using the code provided in the Github repository: https://github.com/astha-chem/mvts-ano-eval. We use the dynamic gaussian scoring function (**Gauss-D**) or the 'univar_gaussian' option in the fit_scores_distribution function provided in the repository.
- **PCA** assigns an anomaly score for each timestamp based on reconstruction error. In particular, we fit PCA on the training data, including the validation data, to obtain the mean and eigenvectors. During real-time anomaly detection testing, we project the multi-dimensional input onto a low-dimensional space, and reconstruct them back again to find the root-mean-square reconstruction error. For the number of principal components, we set it automatically based on the number required to achieve less than 10% sMAPE.
- **AutoEncoder** independently assigns an anomalous score to each observation by tracking the reconstruction error using an encoder-decoder framework. The encoder is a two-layer multilayer perceptron with the dimensions being [input_dimension, 50 and 20], and the decoder is also a a two-layer multilayer perceptron with the dimensions [20, 50 and input_dimension]. Similar to PCA, we train the AutoEncoder on the training data, including the validation data. During real-time anomaly detection testing, we apply AutoEncoder for computing the root-mean-square reconstruction error as anomaly scores at each timestamp.
- **Kmeans** treats each observation as independent points, and generate multiple clusters using the training data. To determine the number of cluster, K, we use Silhouette score and we search K from 0 to 20. During real-time anomaly detection testing, we calculate the distance between multivariate observation and the centroid of its closest corresponding cluster. The computed L2 distance is used as the anomaly score for detecting anomalies.
- **DAGMM** [60] joints Autoencoders and Gaussian Mixture Model to attain anomaly score using reconstruction errors generated from a low-dimensional representation. To reproduce their results on our settings, we use the Github repository:https://github.com/tnakae/DAGMM. We set the dimensions as [20, 10, 5, 1] for the compression network, and as [5, 2] for the estimation network. We set dropout ratio as 0.5. The rest of the parameters follow the default settings. Similar to PCA and AutoEncoder, we train on the training data, including the validation data. During real-time anomaly detection testing, DAGMM predicts the energy of the observation with the more energy suggesting that it more likely to be an anomaly.
- **LSTM-VAE** [26] replaces the feed-forward neural networks in the VAE with a long short-term memory (LSTM) to capture the temporal dependency of time-series data. Nevertheless, the stochasticity of variables modeled by VAE is without temporal dependence. To reproduce the results for LSTM-VAE, we use the code from Github repository: https://github.com/

lin-shuyu/VAE-LSTM-for-anomaly-detection. The hidden dimension of the network is set as 10 and number of epoch for training as 20. The window size is set as 5, 5 and 100 for SWaT, WADI and SMD, respectively. During real-time anomaly detection testing, the anomaly score is based on reconstruction errors at each timestamp.

- **OmniAnomaly** [3] adopts the stochastic variable connection technique, OmniAnomaly's recurrent neural network explicitly models the temporal dependencies between stochastic variables. The anomaly score is the posterior reconstruction probability of each input. Each timestamp is classified as either anomalous or non-anomalous using the Peaks-Over-Threshold method [19]. To reproduce the results from OmniAnomaly, we use the code from Github repository: https://github.com/NetManAIOps/OmniAnomaly. Following the default hyperparameters, we set the z hidden dimension as 3, RNN hidden dimension as 500, normalizing flow layers as 20, and number of epoch for training as 20. The window size is set as 5, 5 and 100 for SWaT, WADI and SMD, respectively. During real-time anomaly detection testing, the anomaly score is based on inverse of reconstruction probability at each timestamp.
- **USAD** [61] is an autoencoder with encoder-decoder architecture that is trained in an adversarial manner to combine the advantages of autoencoders and adversarial training. To reproduce the results from USAD, we use the code from Github repository: https://github.com/manigalati/usad. USAD utilizes one encoder network and two decoder networks. In accordance with the default setting, all networks are three-layer multilayer perceptrons, with the hidden dimension being one-half and one-quarter of the original input dimension respectively. We train USAD over 250 epochs. The window size is set as 5, 5 and 100 for SWaT, WADI and SMD, respectively. During real-time anomaly detection testing, the anomaly score is derived from the reconstruction error at each timestamp.
- **MTAD-GAT** [56] is an attention-based graph neural network that implicitly learns dependence relationships between the multivariate variables by assuming a complete graph between the variables. It computes both reconstruction and forecast errors to detect anomalies. To reproduce the results from MTAD-GAT, we use the code from Github repository: https://github.com/ML4ITS/mtad-gat-pytorch. The Graph Attention Networks used to model spatial and temporal cues consist of a single layer. The initial convolution layer possesses a kernel size of 7, while the number of GRU layers is also set to one, having a hidden dimension of 150. The forecast output module is designed with three hidden layers, each with hidden dimensions of 150. In contrast, the reconstruction output module contains only one hidden layer with a hidden dimension of 150. We train the MTAD-GAT over 50 epochs with a dropout rate of 0.3. The window size is set as 5, 5 and 100 for SWaT, WADI and SMD, respectively. During real-time anomaly detection testing, the anomaly score is computed based on the reconstruction and forecast error at each timestamp.
- **GDN** [11] is an attention-based graph neural network that explicitly learns dependence relationships between the multivariate variables and computes forecast errors by leveraging these relationships as anomaly scores. To reproduce the results from GDN, we use the code from Github repository: https://github.com/d-ailin/GDN. Following the default hyperparameters for WADI (SWaT), we set the embedding vector for the graph learning module to 128 (64), the number of neighbors, k, to 30 (15), and the dimension of hidden layers to 128 (64) neurons. For SMD, we set the hyperparameters to match those of SWaT. We train GDN using 50 epochs with early stopping at 10 epochs. When calculating the deviations, the original GDN model inadvertently incorporates future information into the current timestamp by normalizing errors using the full test set's median values. To rectify this, we replace this median value with the median value from the validation set. The window size is set as 5, 5 and 100 for SWaT, WADI and SMD, respectively. During real-time anomaly detection testing, the anomaly score is determined based on the normalized forecast error at each timestamp.
- **InterFusion** [1] explicitly learns a low-dimensional that captures inter-metric (i.e., the relationship between each univariate variable) and temporal dependency for a sequence of multivariate time-series. The anomalous score is the reconstruction probability. To reproduce the results from InterFusion, we use the code from Github repository: https://github.com/zhhlee/InterFusion. As the repository contain the parameters for each of the setting we used in this study and each setting uses a different configuration, we refer the readers to the repository for details of hyperparameters. During real-time anomaly detection testing, the anomaly score is based on inverse of reconstruction probability at each timestamp.

A2. Empirical Computational Complexity

The table below details the time complexities of all the models. Simple baselines, namely RawSignal, PCA and Kmeans, have negligible implementation time and are thus excluded from the table:

TABLE IX
AVERAGE TRAINING TIME PER EPOCH AND IN TOTAL (EPOCH/TOTAL).

Methods	SWaT	WaDI	SMD
AutoEncoder	0.2min/0.11hr	0.4min/0.35hr	0.3min/0.16hr
DAGMM	0.1min/0.24hr	0.2min/0.79hr	0.1min/0.05hr
LSTM-VAE	0.9min/0.31hr	2.2min/0.72hr	1.1min/0.53hr
OmniAnomaly	1.3min/0.42hr	4.6min/1.53hr	4.3min/1.43hr
USAD	0.1min/0.09hr	0.2min/0.33hr	0.1min/0.02hr
MTAD-GAT	0.7min/0.35hr	4.5min/2.25hr	0.4min/0.23hr
GDN	0.4min/0.31hr	1.2min/0.77hr	0.3min/0.09hr
InterFusion	7.8min/1.95hr	15.9min/3.98hr	3.5min/0.96hr
CST-GL	1.3min/0.43hr	4.1min/1.37hr	0.8min/0.26hr

A3. `CST-GL` Hyperparameter Search Space

We define the hyperparameter search space as shown in the

table below, and select the hyperparameters that achieve lowest average root-mean-square error in the validation set.

TABLE X
THE HYPERPARAMETER OPTIONS WE SEARCHED THROUGH FOR CST-GL ON VALIDATION SET.

Hyperparameter	Search Space
MTCL node dimension	64,128,256,512
MTCL retain ratio	0.05,0.1,0.2,0.3
MTCL saturation rate, α	5,10,20,30
MTCL neighbour size, k	10, 15, 20, 30, 50
Number of TCN Layers	1,2,3
TCN output dimension	16,32,64
Number of GCN Layers	1,2,3
GCN output dimension	16,32,64

REFERENCES

[1] Z. Li, Y. Zhao, J. Han, Y. Su, R. Jiao, X. Wen, and D. Pei, "Multivariate time series anomaly detection and interpretation using hierarchical inter-metric and temporal embedding," in *KDD*, 2021, pp. 3220–3230.

[2] K. Hundman, V. Constantinou, C. Laporte, I. Colwell, and T. Söderström, "Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding," in *KDD*, 2018, pp. 387–395.

[3] Y. Su, Y. Zhao, C. Niu, R. Liu, W. Sun, and D. Pei, "Robust anomaly detection for multivariate time series through stochastic recurrent neural network," in *KDD*, 2019, pp. 2828–2837.

[4] A. P. Mathur and N. O. Tippenhauer, "Swat: A water treatment testbed for research and training on ics security," in *CySWATER*, 2016, pp. 31–36.

[5] C. M. Ahmed, V. R. Palleti, and A. P. Mathur, "Wadi: a water distribution testbed for research in the design of secure cyber physical systems," in *CySWATER*, 2017, pp. 25–28.

[6] Q. Yu, L. Jibin, and L. Jiang, "An improved arima-based traffic anomaly detection algorithm for wireless sensor networks," *Intl. J Of Distrib. Sens. Net.*, vol. 12, no. 1, p. 9653230, 2016.

[7] E. Keogh, J. Lin, and A. Fu, "Hot sax: Efficiently finding the most unusual time series subsequence," in *ICDM*, 2005.

[8] X. Wang, J. Lin, N. Patel, and M. Braun, "Exact variable-length anomaly detection algorithm for univariate and multivariate time series," *DMKD*, vol. 32, no. 6, pp. 1806–1844, 2018.

[9] B. Li, X. Li, Z. Zhang, and F. Wu, "Spatio-temporal graph routing for skeleton-based action recognition," in *AAAI*, 2019, pp. 8561–8568.

[10] Y. Seo, M. Defferrard, P. Vandergheynst, and X. Bresson, "Structured sequence modeling with graph convolutional recurrent networks," in *ICONIP*, 2018, pp. 362–373.

[11] A. Deng and B. Hooi, "Graph neural network-based anomaly detection in multivariate time series," in *AAAI*, 2021, pp. 4027–4035.

[12] M. Jin, Y. Zheng, Y.-F. Li, S. Chen, B. Yang, and S. Pan, "Multivariate time series forecasting with dynamic graph neural odes," *IEEE TKDE*, 2022.

[13] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Computing Surveys*, vol. 41, no. 3, pp. 1–58, 2009.

[14] R. Chalapathy and S. Chawla, "Deep learning for anomaly detection: A survey," *arXiv preprint*, vol. abs/1901.03407, 2019.

[15] A. Blázquez-García, A. Conde, U. Mori, and J. A. Lozano, "A review on outlier/anomaly detection in time series data," *ACM Computing Surveys*, vol. 54, no. 3, pp. 1–33, 2021.

[16] S. Li and J. Wen, "A model-based fault detection and diagnostic methodology based on pca method and wavelet transform," *Energy and Buildings*, vol. 68, pp. 63–71, 2014.

[17] L. M. Manevitz and M. Yousef, "One-class svms for document classification," *JMLR*, vol. 2, no. Dec, pp. 139–154, 2001.

[18] W. Lu and A. A. Ghorbani, "Network anomaly detection based on wavelet analysis," *EURASIP J. Adv. Signal Process*, vol. 2009, pp. 1–16, 2008.

[19] A. Siffer, P. Fouque, A. Termier, and C. Largouët, "Anomaly detection in streams with extreme value theory," in *KDD*, 2017, pp. 1067–1075.

[20] L. Feremans, V. Vercruyssen, B. Cule, W. Meert, and B. Goethals, "Pattern-based anomaly detection in mixed-type time series," in *ECML PKDD*, 2019, pp. 240–256.

[21] C.-C. M. Yeh, Y. Zhu, L. Ulanova, N. Begum, Y. Ding, H. A. Dau, D. F. Silva, A. Mueen, and E. Keogh, "Matrix profile i: all pairs similarity joins for time series: a unifying view that includes motifs, discords and shapelets," in *ICDM*, 2016, pp. 1317–1322.

[22] C. Zhang, D. Song, Y. Chen, X. Feng, C. Lumezanu, W. Cheng, J. Ni, B. Zong, H. Chen, and N. V. Chawla, "A deep neural network for unsupervised anomaly detection and diagnosis in multivariate time series data," in *AAAI*, 2019, pp. 1409–1416.

[23] Z. Wu, S. Pan, G. Long, J. Jiang, X. Chang, and C. Zhang, "Connecting the dots: Multivariate time series forecasting with graph neural networks," in *KDD*, 2020, pp. 753–763.

[24] A. Garg, W. Zhang, J. Samaran, R. Savitha, and C.-S. Foo, "An evaluation of anomaly detection and diagnosis in multivariate time series," *IEEE TNNLS*, 2021.

[25] T. Tayeh, S. Aburakhia, R. Myers, and A. Shami, "An attention-based convlstm autoencoder with dynamic thresholding for unsupervised anomaly detection in multivariate time series," *Machine Learning and Knowledge Extraction*, vol. 4, no. 2, pp. 350–370, 2022.

[26] D. Park, Y. Hoshi, and C. C. Kemp, "A multimodal anomaly detector for robot-assisted feeding using an lstm-based variational autoencoder," *IEEE Robot*, vol. 3, no. 3, pp. 1544–1551, 2018.

[27] D. Li, D. Chen, B. Jin, L. Shi, J. Goh, and S.-K. Ng, "Mad-gan: Multivariate anomaly detection for time series data with generative adversarial networks," in *ICANN*, 2019, pp. 703–716.

[28] S. Chauhan and L. Vig, "Anomaly detection in ecg time signals via deep long short-term memory networks," in *DSAA*, 2015, pp. 1–7.

[29] P. Malhotra, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal, and G. Shroff, "Lstm-based encoder-decoder for multi-sensor anomaly detection," *arXiv preprint*, vol. abs/1607.00148, 2016.

[30] W. Chen, L. Tian, B. Chen, L. Dai, Z. Duan, and M. Zhou, "Deep variational graph convolutional recurrent network for multivariate time series anomaly detection," in *ICML*, vol. 162, 2022, pp. 3621–3633.

[31] S. Han and S. S. Woo, "Learning sparse latent graph representations for anomaly detection in multivariate time series," in *KDD*, 2022, pp. 2977–2986.

[32] Z. Z. Darban, G. I. Webb, S. Pan, C. C. Aggarwal, and M. Salehi, "Deep learning for time series anomaly detection: A survey," *ArXiv preprint*, vol. abs/2211.05244, 2022.

[33] F. Xia, K. Sun, S. Yu, A. Aziz, L. Wan, S. Pan, and H. Liu, "Graph learning: A survey," *IEEE TAI*, vol. 2, no. 2, pp. 109–127, 2021.

[34] H. Zhang, B. Wu, X. Yuan, S. Pan, H. Tong, and J. Pei, "Trustworthy graph neural networks: Aspects, methods and trends," *ArXiv preprint*, vol. abs/2205.07424, 2022.

[35] K. Zhang, Q. Wen, C. Zhang, R. Cai, M. Jin, Y. Liu, J. Zhang, Y. Liang, G. Pang, D. Song *et al.*, "Self-supervised learning for time series analysis: Taxonomy, progress, and prospects," *ArXiv preprint*, vol. abs/2306.10125, 2023.

[36] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *ICLR*, 2017.

[37] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *ICLR*, 2018.

[38] X. Liu, M. Yan, L. Deng, G. Li, X. Ye, D. Fan, S. Pan, and Y. Xie, "Survey on graph neural network acceleration: An algorithmic perspective," in *IJCAI-22*, 2022, pp. 5521–5529, survey Track.

[39] X. Zheng, Y. Liu, S. Pan, M. Zhang, D. Jin, and P. S. Yu, "Graph neural networks for graphs with heterophily: A survey," *ArXiv preprint*, vol. abs/2202.07082, 2022.

[40] M. Jin, Y. Zheng, Y.-F. Li, C. Gong, C. Zhou, and S. Pan, "Multi-scale contrastive siamese networks for self-supervised graph representation learning," in *IJCAI*, 2021.

[41] T. Huang, T. Chen, M. Fang, V. Menkovski, J. Zhao, L. Yin, Y. Pei, D. C. Mocanu, Z. Wang, M. Pechenizkiy *et al.*, "You can have better graph neural networks by not training weights at all: Finding untrained gnns tickets," in *LoG*. PMLR, 2022, pp. 8–1.

[42] Z. Wu, D. Zheng, S. Pan, Q. Gan, G. Long, and G. Karypis, "Traversenet: Unifying space and time in message passing for traffic forecasting," *IEEE TNNLS*, 2022.

[43] M. Jin, Y.-F. Li, and S. Pan, "Neural temporal walks: Motif-aware representation learning on continuous-time dynamic graphs," in *NeurIPS*, 2022.

[44] A. T. N. Nguyen, D. T. N. Nguyen, H. Y. Koh, J. Toskov, W. MacLean, A. Xu, D. Zhang, G. I. Webb, L. T. May, and M. L. Halls, "The application of artificial intelligence to accelerate g protein-coupled receptor drug discovery," *British Journal of Pharmacology*, 2023.

[45] Y. Zheng, M. Jin, Y. Liu, L. Chi, K. T. Phan, S. Pan, and Y.-P. P. Chen, "From unsupervised to few-shot graph anomaly detection: A multi-scale

contrastive learning approach," *ArXiv preprint*, vol. abs/2202.05525, 2022.
[46] Y. Zheng, M. Jin, Y. Liu, L. Chi, K. T. Phan, and Y.-P. P. Chen, "Generative and contrastive self-supervised learning for graph anomaly detection," *IEEE TKDE*, 2021.
[47] W. Jin, Y. Ma, X. Liu, X. Tang, S. Wang, and J. Tang, "Graph structure learning for robust graph neural networks," in *KDD*, 2020, pp. 66–74.
[48] Y. Liu, Y. Zheng, D. Zhang, H. Chen, H. Peng, and S. Pan, "Towards unsupervised deep graph structure learning," in *WWW*, 2022.
[49] M. Jin, H. Y. Koh, Q. Wen, D. Zambon, C. Alippi, G. I. Webb, I. King, and S. Pan, "A survey on graph neural networks for time series: Forecasting, classification, imputation, and anomaly detection," *ArXiv preprint*, vol. abs/2307.03759, 2023.
[50] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
[51] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," in *ICLR*, 2018.
[52] Y. Li, N. Zhong, D. Taniar, and H. Zhang, "Mcgnet+: an improved motor imagery classification based on cosine similarity," *Brain Informatics*, vol. 9, no. 1, pp. 1–11, 2022.
[53] F. Wu, A. Souza, T. Zhang, C. Fifty, T. Yu, and K. Weinberger, "Simplifying graph convolutional networks," in *ICML*. PMLR, 2019, pp. 6861–6871.
[54] S. Abu-El-Haija, B. Perozzi, A. Kapoor, N. Alipourfard, K. Lerman, H. Harutyunyan, G. V. Steeg, and A. Galstyan, "Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing," in *ICML*, 2019.
[55] Z. Wu, S. Pan, G. Long, J. Jiang, and C. Zhang, "Graph wavenet for deep spatial-temporal graph modeling," in *IJCAI*, 2019, pp. 1907–1913.
[56] H. Zhao, Y. Wang, J. Duan, C. Huang, D. Cao, Y. Tong, B. Xu, J. Bai, J. Tong, and Q. Zhang, "Multivariate time-series anomaly detection via graph attention network," in *ICDM*, 2020, pp. 841–850.
[57] H. Xu, W. Chen, N. Zhao, Z. Li, J. Bu, Z. Li, Y. Liu, Y. Zhao, D. Pei, Y. Feng, J. Chen, Z. Wang, and H. Qiao, "Unsupervised anomaly detection via variational auto-encoder for seasonal kpis in web applications," in *WWW*, 2018, pp. 187–196.
[58] D. J. Rezende, S. Mohamed, and D. Wierstra, "Stochastic backpropagation and approximate inference in deep generative models," in *ICML*, 2014, pp. 1278–1286.
[59] T. Chen, X. Liu, B. Xia, W. Wang, and Y. Lai, "Unsupervised anomaly detection of industrial robots using sliding-window convolutional variational autoencoder," *IEEE Access*, vol. 8, pp. 47 072–47 081, 2020.
[60] B. Zong, Q. Song, M. R. Min, W. Cheng, C. Lumezanu, D. Cho, and H. Chen, "Deep autoencoding gaussian mixture model for unsupervised anomaly detection," in *ICLR*, 2018.
[61] J. Audibert, P. Michiardi, F. Guyard, S. Marti, and M. A. Zuluaga, "USAD: unsupervised anomaly detection on multivariate time series," in *KDD*, 2020, pp. 3395–3404.
[62] S. Kim, K. Choi, H. Choi, B. Lee, and S. Yoon, "Towards a rigorous evaluation of time-series anomaly detection," in *AAAI*, 2022, pp. 7194–7201.
[63] R. Wu and E. Keogh, "Current time series anomaly detection benchmarks are flawed and are creating the illusion of progress," *IEEE TKDE*, 2021.
[64] S. Ahmad, A. Lavin, S. Purdy, and Z. Agha, "Unsupervised real-time anomaly detection for streaming data," *Neurocomputing*, vol. 262, pp. 134–147, 2017.
[65] H. Ren, B. Xu, Y. Wang, C. Yi, C. Huang, X. Kou, T. Xing, M. Yang, J. Tong, and Q. Zhang, "Time-series anomaly detection service at microsoft," in *KDD*, 2019, pp. 3009–3017.
[66] G. I. Webb, R. Hyde, H. Cao, H. L. Nguyen, and F. Petitjean, "Characterizing concept drift," *DMKD*, vol. 30, no. 4, pp. 964–994, 2016.
[67] I. Goldenberg and G. I. Webb, "Pca-based drift and shift quantification framework for multidimensional data," *Knowledge and Information Systems*, vol. 62, no. 7, pp. 2835–2854, 2020.
[68] L. Beretta and A. Santaniello, "Nearest neighbor imputation algorithms: a critical evaluation," *BMC Medical Inform. Decis. Mak.*, vol. 16, no. 3, pp. 197–208, 2016.
[69] S. Pan, L. Luo, Y. Wang, C. Chen, J. Wang, and X. Wu, "Unifying large language models and knowledge graphs: A roadmap," *ArXiv preprint*, vol. abs/2306.08302, 2023.