Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/patcog

# Process-Oriented heterogeneous graph learning in GNN-Based ICS anomalous pattern recognition

Shuaiyi L(y)u[a], Kai Wang[b,*], Liren Zhang[b], Bailing Wang[a,*]

[a] Faculty of Computing, Harbin Institute of Technology, China
[b] School of Computer Science and Technology, Harbin Institute of Technology, Weihai, China

A R T I C L E   I N F O

*Article history:*
Received 19 November 2022
Revised 28 March 2023
Accepted 29 April 2023
Available online 2 May 2023

*Keywords:*
Fine-Grained anomaly recognition
Process-Oriented associativity
Heterogeneous graph learning
Industrial control systems

A B S T R A C T

Over the past few years, massive penetrations targeting an Industrial Control System (ICS) network intend to compromise its core industrial processes. So far, numerous advanced methods have been proposed to detect anomalous patterns in the numeric data streams with respect to the heterogeneous field devices involved in the industrial processes. These methods, despite reporting decent results, usually conduct system-wise detection instead of fine-grained anomalous pattern recognition at the device level. Furthermore, lacking explicit consideration of the exclusive process-related features with respect to each differentiated device, the fitness of their application in specified industrial processes is undermined. To tackle these issues, a GNN-based Attributed Heterogeneous Graph Analyzer (the AHGA) is designed to perform device-wise anomalous pattern detection via in-depth process-oriented associativity learning. The AHGA's framework is constructed with four building blocks: a graph processor, a feature analyzer, a link inference decoder, and an anomaly detector. Its performance is assessed and compared against multiple link inference and anomaly detection baselines over 2 popular ICS datasets (SWaT and WADI). Comparative results demonstrate the AHGA's reliability in capturing sophisticated process-oriented relations among heterogeneous devices as well as its effectiveness in boosting the performance of anomalous pattern recognition at device-level granularity.

## 1. Introduction

Security issues in the industrial control systems (ICSs) have aroused considerable attention in recent years [1,2]. Over the past decade, penetrations attempting to compromise the core industrial processes supervised by the ICSs have seen a drastic ascension. In light of the potentially catastrophic outcome such penetrations may lead to (e.g. damage/destruction of critical infrastructures, personnel casualty, economic losses, etc.), deriving reliable detection schemes against these hostile activities in order to secure the ICS processes is of paramount importance universally. As the effect of these penetrations, such as zero-day attacks, can usually be reflected via the numeric variations in the data streams associated with the related field devices, say, the series of sensor readings and actuator operational modes (See Fig. 1), our demand is equivalent to recognizing the deviated and potentially hazardous patterns in these numeric streams.

Anomalous pattern recognition in ICS data streams have been investigated from extensive perspectives ranging from simple classic approaches to advanced machine and deep learning solutions [3]. Leveraging the periodic properties rendered by the ICS traffic, the majority of the classic approaches, such as the DFA series [4], etc. convert the original numeric series into cycles of states, and diagnose any transitional pattern that deviates from the designated flows in the produced cycles. While simple in principle, these solutions are subject to potential state explosion in case the original data exhibits overly convoluted variational patterns. Moreover, these methods only process individual sequences and are hence blind to the in-depth correlational characteristics among different field devices, for which their detection results do not present sufficient credibility.

In order to effectively extract and utilize the in-depth relationships among distinct data streams, Neural Network (NN) based deep learning approaches have been widely investigated for ICS anomalous pattern recognition [5–7]. In general, these methods consist of an input layer that takes in the numeric values associated with all devices as the initial features, a chain of encoding layers or more advanced processing blocks depending on the specific algorithm, and a joint mapping module that produces the predicted labels with respect to the given input (See Fig. 2). Such NN-based models [8,9] are mostly designed to encode both the tem-

[*] Corresponding author.
*E-mail address:* dr.wangkai@hit.edu.cn (K. Wang).

https://doi.org/10.1016/j.patcog.2023.109661
0031-3203/© 2023 Elsevier Ltd. All rights reserved.

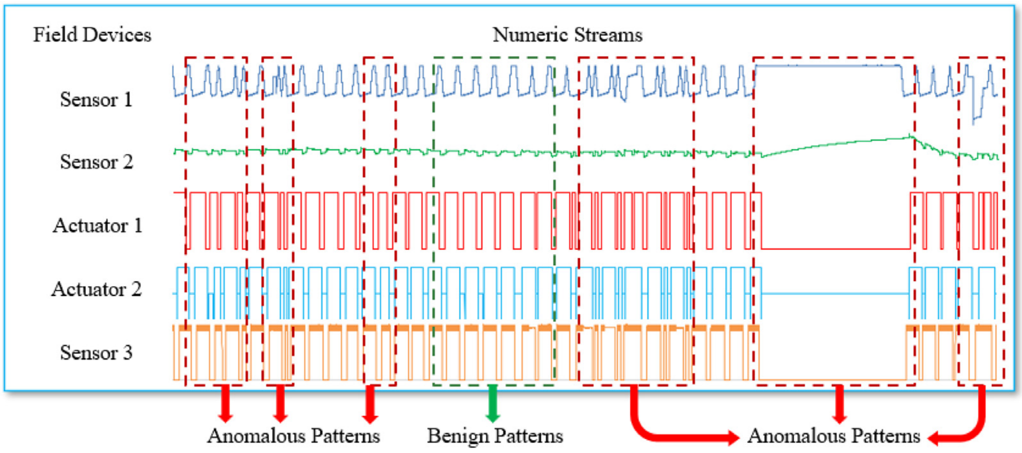


**Fig. 1.** Field device numeric streams with benign and anomalous patterns.

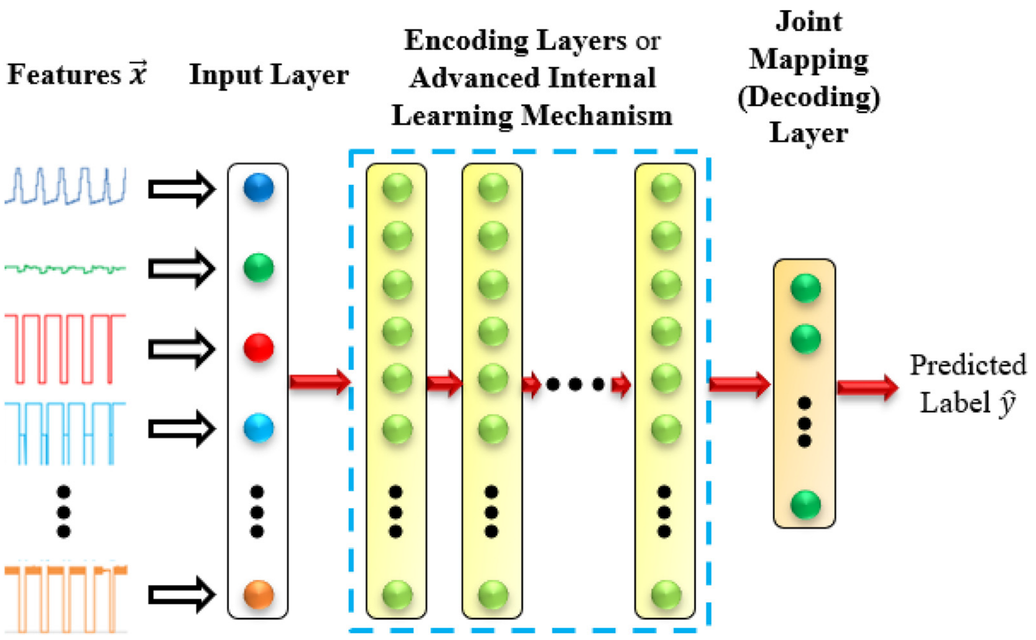

Features $\vec{x}$
Input Layer
Encoding Layers or Advanced Internal Learning Mechanism
Joint Mapping (Decoding) Layer
Predicted Label $\hat{y}$

**Fig. 2.** Outline of neural network ICS anomaly detection scheme (Input: Data streams for all ICS devices. Output: Single system-wise predicted label).

poral and spatial properties among distinct numeric streams, and they report promising results in many metrics. Nevertheless, most of these methods, to the best of our knowledge, inspect system-wise anomalies in an ICS scenario, which means the produced inference results reflect the states of an entire system, rather than an individual device. Such results do not offer valuable insights in accurate retracing and efficient responding to the occurred anomalies.

Recent advances in the Graph Neural Networks (GNNs) have provided massive new opportunities in ICS anomaly detection [10,11], in light of their powerful ability to deal with random graph-structured data [10]. As differentiated from the classic NN approaches, a GNN's input not only includes the device' data streams, but also a graph showing how different devices are connected to each other. Empirically, how well a GNN model performs can be influenced by the quality of the input graph, as it is the core media that defines how the nodes transmit messages to each other. So far, there are extensive GNN methods that aim at developing graph regulating schemes for adversary detection in the ICSs or IoT networks, such as the GDN [12] that dynamically updates the graph based upon the learnt associativity, and the FT-GCN [13] that generates graphs using the profiles of the traffic flows. Effective as they are claimed in their own specified problem, when it comes to securing specific industrial processes, however, their input graphs, typically randomly initialized, do not reflect the devices' explicit relationships in the specified process (For example, the value of a tank level indicator in a water treatment stage is tightly associated with the numeric reading of a flow indicator on a pipe attached to the tank). Potentially, this impairs the quality of the graph upon which subsequent anomalous pattern detection is based.

To address the aforementioned problems in the current NN and GNN-based anomalous pattern recognition for the ICS processes, this paper aims at designing a compound anomaly detector named the Attributed Heterogeneous Graph Analyzer (AHGA), which achieves device-wise (node-level) instead of coarse-grained system-wise anomalous pattern recognition via in-depth process-oriented associativity learning. By applying distributed decoders for every individual node rather than using a joint mapping block, the AHGA can be trained to deduce the states of every device of interest for any applicable jiff, which is a much more challenging task than making inferences for an overall architecture. Furthermore, in order to tackle the inferiority of current GNN approaches that the graph's quality can be negatively affected by existing initialization approaches (such as random initialization), the AHGA is designed to utilize the devices' explicit relations in specific industrial processes as the a priori knowledge for deriving the further sophisticated associativity among heterogeneous devices, and to generate the graph that reflects this advanced associativity. Therefore, with these process-oriented relations considered, the AHGA has a better perception of how different devices in an ICS network are correlated with each other in terms of their roles and functionality, and is hence better tailored to the distinctiveness of disparate industrial processes. To the best of our knowledge, this is one of the first works to leverage the devices' explicit process-oriented relationships to improve the quality of the graph used for ICS anomalous pattern recognition. The AHGA's architecture is composed of the following primary components: (a) A graph processor which initializes the graph structure based upon real world ICS architectures. (b) A feature analyzer that creates a uniform initial vector representation for all heterogeneous devices in the graph, based upon the captured data streams with respect to these devices. In this module, an entropy-based scheme is developed for the derivation of a particular set of statistical properties as part of a vector's entries. (c) A link inference decoder that learns the graph for anomaly detecton, and (d) An anomaly detector that operates on the graph produced by the link inference decoder and conducts node-level anomalous pattern recognition via distributed decoding blocks.

The key contributions of our work can be summarized as follows:

1) We perform comprehensive preprocessing on the data flows captured on ICS testbeds, and define a multi-dimensional feature extraction scheme to profile heterogeneous field devices. Specifically, we present an entropy-based approach for periodicity mining, which is implemented and validated with the provided device readings.
2) We design the AHGA, a compound framework that achieves fine-grained device-level anomalous pattern recognition via graph learning and distributed decoding. Specifically, the devices' explicit process-oriented relations are used as a basis for the AHGA to infer the in-depth associativity among the devices. This learnt associativity is further utilized to conduct device-wise anomalous pattern recognition via distributed decoding.
3) Using the SWaT and WADI datasets, we evaluate the AHGA's ability to detect anomalous patterns in ICS streams via associativity learning. Comparative results on state-of-the-art baselines have justified the positive effect of utilizing the devices' process-oriented relations on accurate associativity learning and reliable anomalous pattern recognition.

The rest of the paper is structured as follows. Section 2 provides a summary of GNN's preliminaries and an overview of related work regarding correlation analysis in ICSs; In Section 3, we define the problem to be addressed; In Section 4, we explain the structure of AHGA; Section 5 presents the model evaluation results and highlights relevant analysis; Section 6 concludes the article.

## 2. Preliminaries and related work

In this section, the GNNs' fundamentals and related work on ICS anomaly pattern detection are overviewed. For the clarity of presentation, all symbols used in this section and their annotations are displayed in Table 1.

### 2.1. Preliminaries

The GNNs perform message passing on graphs to enrich the nodes with their local information. During each round of message passing, every node in the graph update its representation by aggregating messages from its immediate neighbours. This representation is a contextual mixture of a node's own properties as well as its awareness of its surrounding nodes and edges. Suppose $h_v^{(k+1)}$ is the representation for node $v$ at the $(k+1)$-th GNN layer, it associates with the $k$-th layer in the following manner:

$$h_v^{(k+1)} \leftarrow UPDATE(h_v^{(k)}, AGGR(u \in N(v), MSG(h_v^{(k)}, h_u^{(k)}, e(u, v)))) \tag{1}$$

where $N(v)$ is the set of all nodes directly linked to node $v$, and $e(u, v)$ represents the edge connecting nodes $u$ and $v$.

In (1), $MSG(.)$ formulates the message from a particular neighbour node $u$ by extracting important information from $u$ and $v$ and the edge connecting them. $AGGR(.)$ gathers the messages from all $v$s neighbours in a certain way, and produces an output that is subsequently absorbed in the node $v$s own embedding in the $UPDATE(.)$ operator. Note that there are extensive methods to implement these operators, some of which are introduced in Section 2.2.

### 2.2. Related work

So far, massive deep learning methods have been introduced to explore the in-depth features in data streams to yield desirable event detection results [14–16]. For instance, a deep adversial anomaly detection (DAAD) method [17] is proposed to learn task-specific features capture the marginal distributions of normal data in detecting sequential anomalous patterns. A black-box attack scheme (BBAS) [18] is designed to assist improving the DNN's reliability in detecting adversarial example attacks. In order to detect anomalous patterns in ICS and IoT data streams, numerous approaches are introduced including a compound framework implemented with multi-attentional DNN blocks [19], a weighted random sampling approach using a generalized sampling algorithmic framework [20], two semi-supervised hybrid deep learning methods (AE-GRU and GAN-RNN) [21], an LSTM-based architecture (ClozeLSTM) [22], an unsupervised approach combining neural networks and a one-class objective [23], etc. In addition, to tackle false pattern injection in the ICSs, a robust spatial-temporal detector (AD-RoSM) [24] is developed. A light-weight federated learning based anomaly detector (FATRAF) [25] is designed to detect irregular patterns in time-series data. An ML-based anomaly detector that leverages the system's design knowledge [26] is proposed to improve the model's detection accuracy.

The GNN's application in anomaly pattern recognition has been comprehensively investigated [10,27]. For example, a novel meta-graph based convolutional scheme named Meta-GNN [28] is introduced to extract and incorporate complex local properties in order to capture higher-order semantic relationships in the network. A Deep Cluster Infomax approach is proposed for node representation learning [29] in which representation learning and state classification are separately trained. Specifically designed for ICS anomaly detection scenarios, the Graph Deviation Network (GDN) [12] considers inter-sensor relationships as a key factor in detecting anomalous events via studying a high-dimensional time series. Likewise, the MST-GNN [30] performs feature extraction over multivariate time series data considering the properties of each individual series. In order to create robust low-dimensional representations, a new contrastive-based unsupervised graph representation learning (UGRL) framework [31] is designed that associates downstream tasks to the learning process via contraints. A modified strategy, e-ResGAT [32] is proposed on the basis of the regular GAT via residual learning. In addition, to improve the quality of the features used for message aggregation, the AsGNN [33] is developed to perform feature selection via normalization, and the GLIN

**Table 1**
Symbol Annotations.

| Symbols | Annotations |
|---|---|
| $v, u$ | Nodes in a graph |
| $h_v^{(k)}$ | Vector representation of node $v$ after the $k$-th round of message passing is complete |
| $N(v)$ | $v$'s immediately connected nodes in a the graph |
| $e(u, v)$ | An link connecting nodes $u$ and $v$ |
| $MSG(.)$ | Operator that creates a message between a specified pair of connected nodes |
| $AGGR(.)$ | Operator that aggregates the messages produced between one node and all its connected neighbours |
| $UPDATE(.)$ | Operator that combines a node's own representation and the aggregated message from its neighbours |

**Table 2**
Symbol Annotations.

| Symbols | Annotations |
|---|---|
| $G$ | Graph to learn |
| $V$ | Set of all vertices (devices) in the learnt graph $G$ |
| $E$ | Set of all links in $G$ relecting the learnt relationships among the devices |
| $E_0$ | Set of all links reflecting a priori knowledge |
| $S$ | Set of all relations that may exist, apart from the links repesenting the a priori knowledge |
| $C$ | Set of all conditions defining explicit process-oriented relations among the devices |

[34] is designed to utilize a graph's global properties in improving the model's detection accuracy. Further, to reduce the scale of the graphs for efficient computing, the Graph Reduction Neural Network [35] is proposed to perform structural pattern recognition.

Despite being highly advanced and achieving good results, the aforementioned solutions mostly apply to coarse-grained system-level anomaly detection for which they are incapable of locating anomalous devices, significantly degrading their practicality. Therefore, this work aims at deriving a comprehensive solution to detect anomalous patterns via in-depth associativity learning. Particularly, the proposed method is able to yield results for specific devices instead of deriving system-level anomaly detection.

## 3. Problem statement

In this section, the problem to be addressed as well as some intuitive ideas behind the proposed solution (AHGA) is introduced. To make relevant descriptions clear, all symbols that appear in this section are listed in Table 2 along with their annotations.

As an anomalous pattern recognition solution exhibiting promising prospects, the GNNs enable embedding learning via message passing on a predefined graph topology $G(V, E)$. The purpose of designing the AHGA is to derive a feasible and process-wise interpretable mechanism to produce a proper $G(V, E)$, on which anomalous pattern recognition is conducted. Defining $G(V, E)$ naturally breaks down to determining the node set $V$ and the edge set $E$. In case of abstracting each ICS device to a node in the graph, determining $V$ is trivial as the set of ICS devices related to a particular industrial process usually remains stationary over time, due to the convention that addition or removal of devices is rare in an ICS environment to circumvent undesired influence on the process. Therefore, our first primary task addressed in this paper is to define the link set $E$ so that the graph serves as an appropriate reflection of the sophisticated relationships among heterogeneous ICS devices. That said, this problem is defined as follows:

**Definition 1.** Given an industrial control network $G(V, E_0)$, where $V$ is the set of individual hosts, controllers, or field devices, and $E_0$ the set of known relations (such as direct physical communications) among all elements in $V$. i.e. $\forall e \in E_0, link(e) = 1$ (Operator $link(.)$ returns 1 on $e$s existence or 0 otherwise). Let $S = V \times V \backslash E_0$, and $C$ is the set of field conditions. Find $E \subseteq S$ so that $\forall e \in E, link(e|C) = 1$.

In this paper, we leverage the devices' explicit process-oriented associativity to determine the graph for subsequent anomalous pattern recognition. In a water treatment process, for instance, we may consider that the sensor measuring the water level of a tank and the sensor measuring the water flow in the pipe connected to the tank are associated with each other in their readings. On the other hand, we may not expect much connection between the set of sensors operating in one location and those in another, on the assumption that the processes in disparate geographical locations operate independently. In Definition 1, $C$ is the set of empirically regulated conditions based on the devices' relations exemplified as above. The set does not incorporates all the relationships among the set of nodes $V$. However, it serves as a heuristic for the model to learn a broader variety of relations during the training process. Therefore, the edge set $E$ produced by the AHGA is supposed to cover most or all connections suggested by $C$, as well as other relations not directly defined in it. The produced $E$ is then utilized in downstream anomaly detection tasks.

As the second core task, the AHGA conducts anomalous pattern recognition over the numeric streams with respect to the ICS devices using the produced graph $G(V, E)$. More sophisticated links incorporated in $G$, the AHGA is able to obtain a more in-depth perception of the industrial process, allowing a boost in the results' reliability.

## 4. Model design

The AHGA consists of four core modules respectively named as a graph processor, a feature analyzer, a link inference decoder and an anomaly detector. The graph processor abstracts the a priori knowledge on the devices' relations into a graph topology, which serves as a basis for subsequent graph learning. The feature analyzer creates unified profiles for heterogeneous devices with comprehensive information obtained from numerous respects, including the devices' roles in the industrial process, statistic properties of their numeric streams, etc. The output of the graph processor and the feature analyzer is then passed onto the link inference decoder (implemented as a multi-layer GNN structure) to learn the devices' complex process-oriented relations, with which the AHGA gains a comprehensive view over the industrial process. Finally, these newly learnt relations are utilized by an anomaly detector (a second multi-layer GNN-based module) to recognize anomalous patterns with respect to each device. The overall structure of AHGA

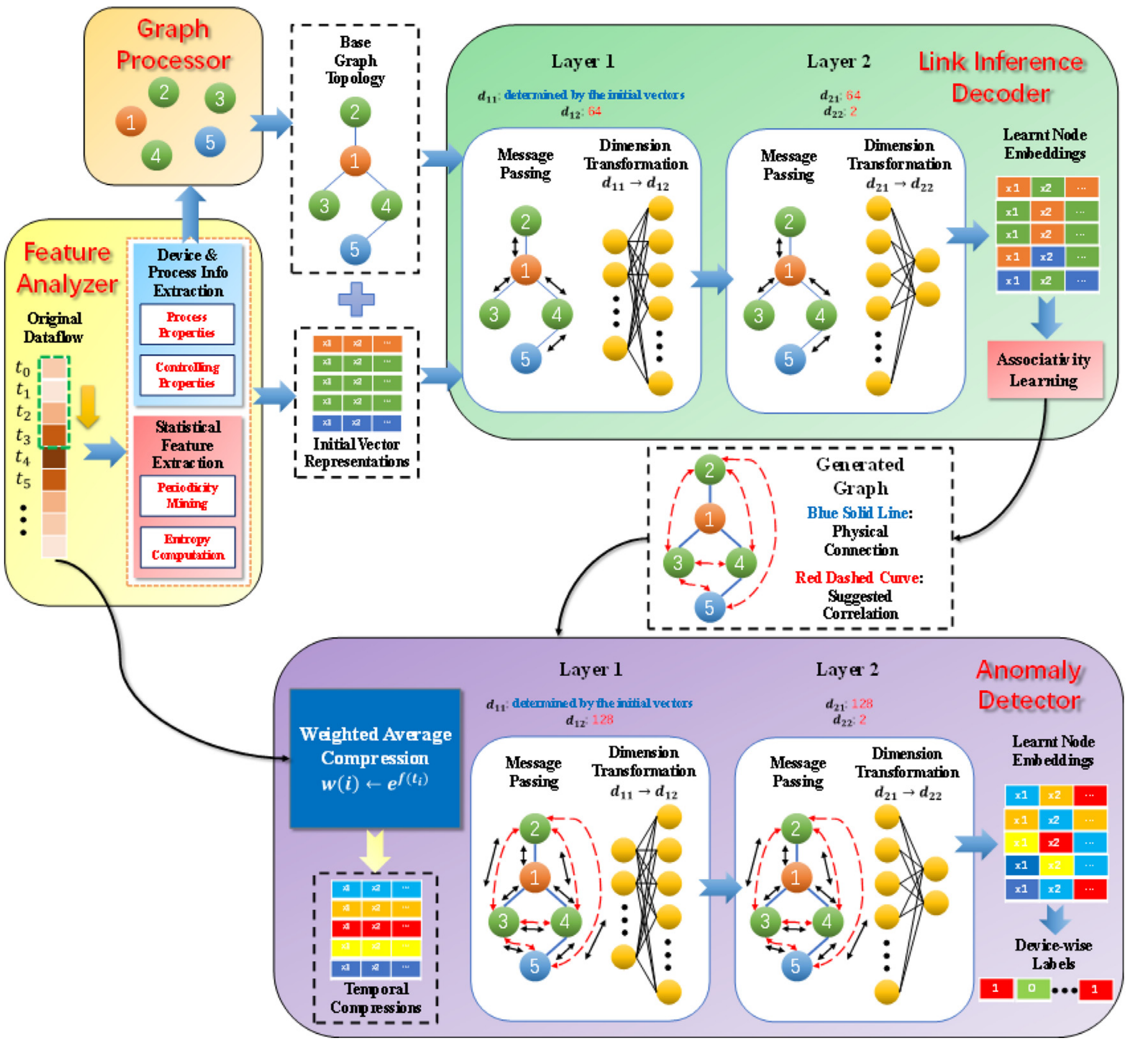


**Fig. 3.** AHGA Structure.

is presented in Fig. 3. Note that in order to give clear presentation, core symbols that appear in this section are listed in Table 3 along with their annotations.

## 4.1. Graph processor

The graph generator constructs a base topology $G(V, E_0)$ exhibiting the devices' explicit process-oriented features on the basis of which subsequent graph learning is conducted. Specifically, 2 types of features are considered: physical communications and numeric stream correlations. In terms of physical communication, for instance, direct data transmission between a sensor and its supervising controller determines a solid physical link of communication between them. As to the stream numeric correlations, on the other hand, a sensor measuring a tank's liquid level is numerically correlated with another sensor that records the flow volume of a pipe linked to this tank.

With both respects taken into account, the edge set $E_0$ of the base topology can be interpreted as follows (see (2)):

$$E_0 = \Psi^{(T)} \cup \Psi^{(C)} \tag{2}$$

where $\Psi^{(T)}$ is the set of all physical communication links and $\Psi^{(C)}$ the set of stream numeric correlations. Equation (2) indicates that link set of the base topology is the union of $\Psi^{(T)}$ and $\Psi^{(C)}$.

To determine the devices' physical links of communication $\Psi^{(T)}$, a general hierarchical topology (see Fig. 4(a)) with multiple layers featuring distinct functions is adopted as a reference. As shown in Fig. 4(a), atop the structure is an aggregation of all remote management systems typically deployed within a corporations internal network, which are usually bounded within the cloud periphery for the purpose of observation or exhibition. This layer is conventionally prohibited to directly interfere with the field process on the lower level. Immediately below it is a layer of controlling units that continuously interact with the peripheral field devices (sensors and actuators) on the bottom layer during operation of an industrial process.

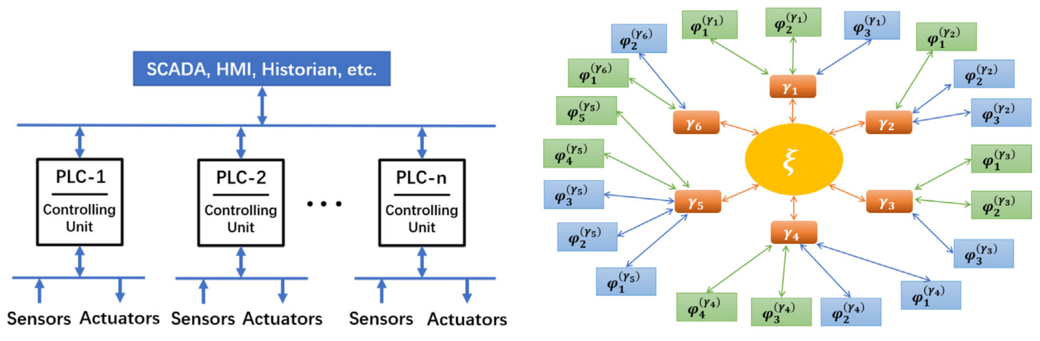
In order to create a general base topology that adapts to the majority of typical ICS architectures in terms of devices' physical communication, we encapsulate everything above the controlling layer into one single node denoted as the central reference point (the CRP, and denoted as $\xi$) while mapping all other devices to their own corresponding nodes. This finalizes the node set $V$ as a set of controllers, field devices, plus the CRP. (See (3))

$$V = \{\xi, \{\gamma_i | i \in [1, n] \cap Z\}, \{\{\varphi_j^{\gamma_i}\} | i \in [1, n] \cap Z\, j \in [1, n_i] \cap Z\}\} \tag{3}$$

Note that in (3), $n$ represents the controllers' quantity, which in our case, equals the number of stages given each stage is assigned with only one controller. $n_i$ is the number of field devices in the $i$-th stage. $\gamma_i$ refers to the controller in the $i$-th stage, and $\varphi_j^{\gamma_i}$ corresponds to the $j$-th field devices supervised by controller $\gamma_i$. Equation (3) defines the explicit composition of the node set $V$.

**Table 3**
Symbol Annotations.

| Subsection | Symbols | Annotations |
|---|---|---|
| 4.1 | $E_0$ | Set of all links reflecting a priori knowledge |
| 4.1 | $\Psi^{(T)}$ | Set of links representing the devices' physical communications |
| 4.1 | $\Psi^{(C)}$ | Set of links relecting how the states of devices are numerically correlated |
| 4.1 | $\xi$ | Node encapsulating the part of an ICS above the controllers' layer (e.g. SCADA, HMI in Fig. 4) |
| 4.1 | $n$ | Controllers' quantity |
| 4.1 | $Z$ | Set of all integers |
| 4.1 | $\gamma_i$ | The controller with respect to the $i$-th stage |
| 4.1 | $\varphi_j^{\gamma_i}$ | The $j$-th field devices supervised by controller $\gamma_i$ |
| 4.1 | $\Omega$ | List of dictionaries reflecting the links between the controllers and field devices |
| 4.1 | $\omega$ | Dictionary in $\Omega$ reflecting the links between the controller and field devices in a specific stage |
| 4.2 | $S$ | Original numeric data stream |
| 4.2 | $N$ | Length of sequence for entropy computation |
| 4.2 | $\beta_{\min}$, $\beta_{\max}$ | Global minima, maxima of a sequence |
| 4.2 | $k$ | Number of intervals devided in range $[\beta_{\min}, \beta_{\max}]$ |
| 4.2 | $\Theta^{(i)}$ | An event in which an element falls within the $i$-th interval of a sequence |
| 4.2 | $\mathrm{E}_i^{(k)}$ | Entropy of event $\Theta^{(k)}$ computed from sequence of length $N$ |
| 4.2 | $\Lambda^{(Var)}$, $\Lambda^{(SD)}$ | Variance, Standard Deviation of sequence entropies |
| 4.2 | $w_0$, $w_{\max}$ | Initial period candidate, upper bound of period value |
| 4.2 | $\delta$, $\delta_w$ | Variation rate of period candidate value, sliding step size of a window on data stream $S$ |
| 4.2 | $\psi$ | Location (starting point) of a specific window on data stream $S$ |
| 4.2 | $P$, $n_P$ | Set of all windows, size of set $P$ |
| 4.2 | $\rho_{\max}$, $\rho_{\min}$ | Averaged period maxima, minima over all windows for data stream $S$ |
| 4.2 | $\mu(\mathrm{E}_i^{(k)})$, $\sigma^2(\mathrm{E}_i^{(k)})$ | Entropy mean, variance over all windows for data stream $S$ |
| 4.2 | $\overrightarrow{e}$ | Initial vector representation of a device before normalization |
| 4.2 | $\overrightarrow{e_G}$, $\overrightarrow{e_C}$, $\overrightarrow{e_M}$ | Sub-vectors in $\overrightarrow{e}$ with respect to the device's General, Control and Measurement features |
| 4.2 | $e_{norm}$ | Initial vector representation after normalization |
| 4.2 | $x$ | Initial vector representation of a device after PCA processing |
| 4.2 | $m$ | $x$'s dimension |
| 4.3 | $v_i$ | Particular node in the node set $V$ |
| 4.3 | $x_i$ | Initial vector representation with respect to node $v_i$ |
| 4.3 | $h_i$ | Learnt embedding with respect to node $v_i$ in the Link Inference Decoder |
| 4.3 | $h_i^{(k)}$ | Learnt embedding with respect to node $v_i$ after the $k$-th message passing step is complete |
| 4.3 | $d_i$ | Degree with respect to node $v_i$ |
| 4.3 | $W^{(k)}$ | Parameter matrix with respect to the $k$-th message passing layer in the Link Inference Decoder |
| 4.3 | $H$ | Embedding matrix containing all learnt embeddings $h_i$'s |
| 4.3 | $A_{ij}$, $p(A_{ij})$ | Event that node $v_i$ and $v_j$ are correlated, probability of $A_{ij}$'s occurence |
| 4.3 | $\mathrm{P}_{\mathrm{link}}$ | Matrix containing all $A_{ij}$'s |
| 4.3 | $\eta$ | Number of message passing layers |
| 4.3 | $ep$, $ep_{\max}$ | Epoch index, configured upper bound of the epoch's quantity |
| 4.4 | $\chi$ | Preprocessed temporal representation of a device, acting as an initial vector for message passing |
| 4.4 | $\mathfrak{S}$ | Interval extracted from original numeric stream for temporal condensation |
| 4.4 | $\mathfrak{N}$, $\mathfrak{n}$ | Length of $\mathfrak{S}$, number of consecutive values in $\mathfrak{S}$ condensed into a single value |
| 4.4 | $a$ | Balancing factor shaping the amplitude of exponential weights |
| 4.4 | $W_{ad}^{(k)}$ | Parameter matrix with respect to the $k$-th message passing layer in the Anomaly Detector |
| 4.4 | $\mathfrak{h}_i$ | Learnt embedding with respect to device $v_i$ in the Anomaly Detector |
| 4.4 | $\hat{y}_i$, $\hat{Y}$ | Label output with respect to device $v_i$, list of all $\hat{y}_i$'s |



(a) ICS Topology (b) Initial Graph

**Fig. 4.** ICS topology and the abstracted graph.

**Table 4**
Process Oriented Conditions.

| No. | Devices | Correlation |
|---|---|---|
| 1 | FITs from adjacent stages | true |
| 2 | FITs and LIT(LT)s within the same stage | true |
| 3 | Adjacent devices in the piping diagram | true |
| 4 | AITs and other sensors | false |
| 5 | Devices in the same stage exhibiting similar periodic features | true |

The scheme for deriving the devices' physical linkage $\Psi^{(T)}$ is illustrated in Algorithm 1, and this linkage is displayed in Fig. 4(b). The oval yellow node $\xi$ is the central reference point abstracted from all supervisory elements from the upper level. It is surrounded by multiple orange nodes $\gamma$'s representing controllers. Each controller $\gamma$ is associated with numerous field devices $\varphi^{(\gamma)}$'s (green rectangles for sensors and blue for actuators).

**Algorithm 1** Physical Linkage Definition.

```
Input: List of stages $\Omega$; Each stage element $\omega$ is a dictionary mapping every controller $\gamma$ in the stage
to the list $\Phi^{(\gamma)}$ of field devices $\varphi^{(\gamma)'}$s in connection with $\gamma$.
Output: Node set $V$ and physical link set $\Psi^{(T)}$
Ensure:
1: $V \leftarrow \emptyset$, $\Psi^{(T)} \leftarrow \emptyset$
2: Initialize the CRP node as $\xi$
3: $V \leftarrow V + \xi$
4: for each $\omega$ in $\Omega$:
5:     for each controller $\gamma$ in $\omega$:
6:         $\Psi^{(T)} \leftarrow \Psi^{(T)} + <\xi, \gamma>$
7:         $V \leftarrow V + \gamma$
8:         for each field device $\varphi^{(\gamma)}$ connected to $\gamma$:
9:             $\Psi^{(T)} \leftarrow \Psi^{(T)} + <\gamma, \varphi^{(\gamma)}>$
10:            $V \leftarrow V + \varphi^{(\gamma)}$
11:        end for
12:    end for
13: end for
14: return $(V, \Psi^{(T)})$
```

Defining the devices' stream correlations $\Psi^{(C)}$ requires some intuitive observations over the system. Such correlations can be derived from a few general conditions that serve as an overall induction of the devices' numeric associativity inferred from the system's piping and instrumentation diagram. Taking the popular SWaT and WADI datasets as an example, such conditions are summarized in Table 4. Note that minor discrepancies exist in the devices' notation representations (For example, the LITs in SWaT and the LTs in WADI).

In the SWaT and WADI datasets, flow indication transmitters (FITs) are sensors that measure flows in pipes. Since flows aggregate in tanks whose water levels are evaluated by level indication transmitters (LITs), it is intuitively considered that the readings of these two sets of devices are intimately related. In the meantime, we expect the existence of relations among FITs in adjacent stages due to piping interconnections. The analyzer indicator transmitters (AITs) measure the acidity and conductivity of the flow. It is empirically assumed that these inherent chemical properties have little impact on the physical characteristics quantified in FITs and LITs. Therefore, negativity is set as the ground truth value for all pairs with an AIT as one of the elements. Finally, we incorporate periodicity as a link indicator based on the assumption that given one state change causes a variation of another, they share similar cyclic properties. For example, the water level in a tank varies in a periodic fashion due to the cyclic change of flows in its directly linked pipes. Consequently, many links among the devices are assigned a value of 1 denoting their existence and 0 otherwise. A portion (e.g. 40%) of these links are sampled to construct $\Psi^{(C)}$ and the rest is utilized for assessing the quality of the graph during the evaluation process.

With both $\Psi^{(T)}$ and $\Psi^{(C)}$ determined, the construction of the base topology is considered complete, and the base graph is finalized as $G(V, E_0 = \Psi^{(T)} \cup \Psi^{(C)})$.

## *4.2. Feature analyzer*

In a heterogeneous ICS structure covering a variety of devices, difficulties in creating a general representation format for every node in the network reside in the nodes' profile incompatibility. Current solutions tend to circumvent this issue via vector randomization. While this might be effective under specific circumstances and straightforward to implement, it undermines the model's credibility as to how each device's properties impact the model's performance. Therefore, to counter this disadvantage, we design the Feature Analyzer module to produce a unified and interpretable representing paradigm that fuses a node's features from multiple perspectives.

We incorporate attributes of three categories in a node's vector representation, namely the general process, controlling and measurement properties. Taking the sensors as an example, the general process features include the devices' scope of interference; Controlling properties exhibit their position as well as the relationship with their direct neighbourhood in the network; Measurement properties form a set of data characteristics rendered by the readings, which are closely relevant to the respective industrial process. Below is the list of features extracted for all sensors with applicable readings (See Table 5).

Most of the features in Table 5 are fairly obvious to discern in the base topology and easily expressed in one-hot representations (e.g. Given there are 4 stages in an industrial process, vector (0,1,0,0) refers to the 2nd stage as the 2nd entry in the vector is non-zero). However, measurement property extraction necessitates in-depth flow analysis. So far, numerous studies have suggested a decent likelihood of flow periodicity in industrial processes, ranging from the devices' communicating routine to their numeric stream patterns. Therefore, we select a couple of metrics characterizing this periodic pattern along with a few subsidiary data distributive properties as our measurement features, as shown in Table 5.

To specify all measurement properties, periodicity is flagged in a one-hot manner ((1,0,0) for non-constant periodic, (0,1,0) for constant, (0,0,1) for non-periodic). Period values are set to the corresponding period, zero and infinity with respect to periodic, constant and non-periodic streams. The max (min) value is computed as the average of the largest local maxima (smallest local minima) within all consecutive cycles for periodic streams, while defined as the global maxima (minima) for non-periodic streams. The mean and the standard deviation of the information entropy are applicable only to periodic streams and are set to zero and infinity with respect to constant and non-periodic streams. The definition of the information entropy in this context is presented as below:

Table 5
Feature Specification.

| Categories | Features |
|---|---|
| General Process Properties | Device type, Process & Sub-process |
| Controlling Properties | Controllers performing pooling |
| Measurement Properties | Periodicity, Period Value, Period Max(Min),Period Entropy Mean(Variance) |

**Definition 2.** Given a finite sequence of $N$ elements, the sequence's closed range $[\beta_{\min}, \beta_{\max}]$ is partitioned into $k(1 < k < N)$ segments where $\beta_{\min}$ and $\beta_{\max}$ correspond to the sequences global minima and maxima. $\Theta^{(k)}$ corresponds to the event in which an element's value falls within the $k$-th interval. The entropy $\mathrm{E}_i^{(k)}$ is hereby defined as

$$\mathrm{E}_i^{(k)} \leftarrow -\sum_{i=1}^{k} p(\Theta^{(i)})log(p(\Theta^{(i)})) \tag{4}$$

The entropy defined in equation (4) characterizes the distributive properties of a sequence and thus is employed as a measurement feature of sensor reading series. For simplicity, the range is divided uniformly.

In order to determine the period value in cyclic streams, we design and develop the Sliding Window Entropy (SWE). For a perfectly periodic sequence with period $T$, entropy defined in a particular manner obtained within a window of size $\tau = nT(n \in Z^+)$ remains constant. In other words, the entropy suggests zero variance at $\tau$. Such a perfect scenario is rarely spotted in a real ICS setting. Therefore as an alternative, we find the $T$ that produces the most stationary entropy, measured by its Variance $(\Lambda^{(Var)})$ or standard deviation $(\Lambda^{(SD)})$. The overall pipeline is presented in Algorithm 2.

**Algorithm 2** Sequence Period Mining.

```
Input: Data sequence $S$
Output: Period Candidate $T$
Ensure:
1: Initialize the window size $\ell$ as $w_0$, window size upper bound $w_{\max}$, variation step size $\delta$, sliding step
   size $\delta_w$, number of segments as $k$ and the list of entropy $L(\mathrm{E}_i^{(k)})$
2: while $\ell$ is in the interval $[w_0, w_{\max}]$:
3:     $L(\psi) \leftarrow [k \times \delta_w$ for $k \in [0, \left\lfloor \frac{length(S)-\ell}{\delta_w} \right\rfloor) \cap Z]$
4:     for all $\psi$ in $L(\psi)$:
5:         $\varepsilon \leftarrow \mathrm{E}_i^{(k)}$ within window starting at $\psi$ in $S$ (calculated in (4))
6:         $L(\mathrm{E}_i^{(k)}) \leftarrow L(\mathrm{E}_i^{(k)}) + \varepsilon$
7:     end for
8:     perform convexity check on $L(\mathrm{E}_i^{(k)})$: We keep counts of consecutive descents in $L(\mathrm{E}_i^{(k)})$
       and the succeeding rises to distinguish a convex curve from random noisy fluctuations
9:     if $L(\mathrm{E}_i^{(k)})$ shows local convexity:
10:        $T \leftarrow L(\mathrm{E}_i^{(k)})$'s local minima
11:        return $T$
12:    else:
13:        clear $L(\mathrm{E}_i^{(k)})$
14:    end if
15:    $\ell \leftarrow \ell + \delta$
16: end while
17: $T \leftarrow w_{\max}$
18: return $T$
```

The main idea of Algorithm 2 is to apply sliding windows to a stream of numeric values and determine the window size that minimizes the entropy's (Defined in Definition 2) standard deviation. The window size is originally set to $w_0$ and increases by step size $\delta$ upon completion of each iteration (stopping at the upper bound $w_{\max}$). During each iteration, the entropy $\mathrm{E}_i^{(k)}$ is computed for all windows and the variance of the entropy sequence $\Lambda^{(Var)}$ is stored in an array. A smaller $\Lambda^{(Var)}$ value indicates a stronger tendency of convergence in the entropy sequence. As the window size increases, if $\Lambda^{(Var)}$ consistently decreases to a local minima (determined via convexity check in line 8), this indicates that the entropy at this setting is the most stationary, implying potential periodicity in the original sequence. The algorithm outputs the corresponding window size as the resulting period value $T$. Once we get $T$, and given the set P of the $T$-length windows defined at all positions $\psi$'s in the stream as regulated in Algorithm 2, the rest of the measurement property values are assigned or calculated as thus:

$$\rho_{\max} = \frac{1}{n_{\mathrm{P}}} \sum_{\forall w \in \mathrm{P}} \sup\{S[w : w + T]\} \tag{5}$$

$$\rho_{\min} = \frac{1}{n_{\mathrm{P}}} \sum_{\forall w \in \mathrm{P}} \inf\{S[w : w + T]\} \tag{6}$$

$$\mu(\mathrm{E}_i^{(k)}) = \frac{1}{n_{\mathrm{P}}} \sum_{\forall w \in \mathrm{P}} \mathrm{E}_i^{(k)} \tag{7}$$

$$\sigma^2(\mathrm{E}_i^{(k)}) = \frac{1}{n_{\mathrm{P}}} \sum_{\forall w \in \mathrm{P}} [\mu(\mathrm{E}_i^{(k)}) - \mathrm{E}_i^{(k)}]^2 \tag{8}$$

Note that the $\rho_{\max}$ in (5) and the $\rho_{\min}$ in (6) represent the averaged period maxima and minima. They are computed as an average of all maximum(or minimum) values with respect to the sliding windows (Note that $\sup\{S[w : w + T]\}$ refers to a window starting at $w$. The $sup(.)$ and $inf(.)$ indicates the upper and lower bounds of the window's numeric range). The $\mu(\mathrm{E}_i^{(k)})$ in (7) and the $\sigma^2(\mathrm{E}_i^{(k)})$ in (8) stand for the mean and variance of the entropies computed for all windows ($n_{\mathrm{P}}$ is the size of window set P).

After all features in Table 5 are ready, they are fused within a single vector $\vec{e}$ (see (9)) as a device's initial vector representation for subsequent graph learning. Taking the SWaT dataset as an example, there are totally 33 entries in this vector (See Fig. 5). The number of entries with respect to the general, controlling and measurement features are 14, 6 and 13. All particular features and the quantity of entries they are assigned are highlighted in red context. For instance, in terms of measurement properties, the following features are specified: measurement type, periodicity and statistic related features. They are each allocated 5, 3 and 5 entries in the vector representation (Particularly, 5 entries are assigned for the one-hot expression of 5 measurement types).

$$\vec{e} = \vec{e_G}||\vec{e_C}||\vec{e_M} \tag{9}$$

Note that $\vec{e_G}$, $\vec{e_C}$ and $\vec{e_M}$ in (9) are sub-vectors with respect to the general, controlling and measurement features defined in Table 5, and the symbol || denotes concatenate operation.

It is notable that the numeric ranges of values with respect to different features can significantly deviate from each other, which might cause the training algorithm to be heavily biased towards the feature values with dominant scales. To circumvent such discrimination, normalization is applied over all the feature values.

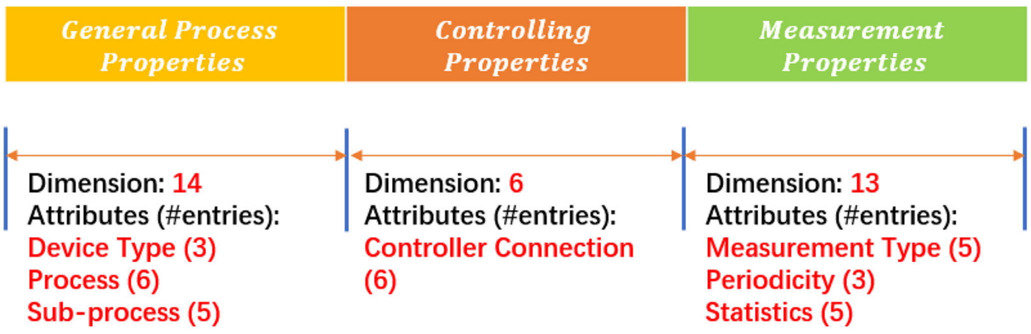


**Fig. 5.** Initial Feature Vector Structure (SWaT Showcase).

For simplicity, we introduce the *arctangent* scheme to achieve normalization (see (10)).

$$e^{(i)}_{norm} \leftarrow \frac{2}{\pi}\arctan(e^{(i)}) \tag{10}$$

In (10), $e^{(i)}$ is the $i$-th element in $\vec{e}$, and $e^{(i)}_{norm}$ is the corresponding normalized value. Note that the value of $i$ does not exceed $e^{(i)}$'s dimension.

By applying normalization, every feature is mapped to some value within the range of (0, 1), hence shrinking the likelihood of commonly experienced training problems such as gradient vanishing.

Finally, the Principal Component Analysis (PCA) is applied to the normalized vector to downsize the initial vector representation $x$ for efficient graph learning (see (11)).

$$x_{[1\times m]} = PCA(e_{norm}) \tag{11}$$

In (11), $x$ is the finalized initial vectors and $m$ refers to $x$'s dimension, which is usually smaller than the size of $e_{norm}$.

## 4.3. Link inference decoder

Given the devices' initial vectors $x$'s the Feature Analyzer produces, there are extensive methods to infer the devices' associativity, such as the Bayesian Network and Support Vector Machine (SVM). However, despite being simple and efficient, these methods lack consideration of the devices' associativity in a specified topological context, and are therefore insufficient for accurate link inference (graph learning) in complex networks including the ICSs. Thus, to guarantee the method's adaptivity to distinctive networks, we employ the GNN approach.

To facilitate efficient graph learning, we utilize the uncomplicated message passing layers (GCN, GAT and GraphSAGE), whose performances are assessed and compared in Section 5. Given all intial vectors $x$'s derived in (11), the message passing layers converts the $x$'s into new embeddings $h$'s. To exemplify this process with a 2-layer GCN using the *ReLU* activation, this transformation for node $v_i$ in graph $G(V, E_0)$ is achieved as thus:

$$\overline{msg_1(v_i, v_j)} = \frac{1}{d_i}\sum_{<v_i,v_j>\in E_0\ \&\ v_i,v_j\in V}\frac{1}{d_j}x_j \tag{12}$$

$$h^{(1)}_i = ReLU\Big(\Big(\overline{msg_1(v_i, v_j)} + x_i\Big)\cdot W^{(1)}, <v_i, v_j>\in E_0\ \&\ v_i, v_j\in V\Big) \tag{13}$$

$$\overline{msg_2(v_i, v_j)} = \frac{1}{d_i}\sum_{<v_i,v_j>\in E_0\ \&\ v_i,v_j\in V}\frac{1}{d_j}h^{(1)}_j \tag{14}$$

$$h_i = ReLU\big(\big(\overline{msg_2(v_i, v_j)} + h^{(1)}_i\big)\cdot W^{(2)}, <v_i, v_j>\in E_0\ \&\ v_i, v_j\in V\big) \tag{15}$$

Equations (12) and (14) regulates how a message from node $v_j$ to node $v_i$ is produced using $v_j$'s vector representation. They show that each message goes through 2 phases of normalization: Phase 1 occurs at the message source $v_j$ in which the original vector is normalized with $v_j$'s out-degree $d_j$, and Phase 2 happens at the destination node $v_i$ which further normalize the message with $v_i$'s in-degree $d_i$. Equations (13) and (15) exhibits how a node $v_i$ updates its own vector with the aggregated messages from its neighbours. Note that $W^{(1)}$ and $W^{(2)}$ are weight matrices with respect to the 2 message passing layers. The shapes of $W^{(1)}$ and $W^{(2)}$ are configured by the input's dimension as well as the number of neurons assigned to each hidden layer.

In our instantiation, we empirically fix the neuron quantity in one message passing layer to 64 to keep the computational cost of the network manageable as well as to maintain a balance between the states of overfitting and underfitting, while allowing for the number of layers to traverse all integer values from 2 to 4. The reason for limiting the number of layers within 2 and 4 is stated as follows: As every field device is directly connected to its controller (see Fig. 4(b)), it takes as few as 2 moves for one peripheral field device to receive messages from another deployed in the same stage, given only one controller is available for each stage. With all controllers linked to a single node (the CRP), it costs as few as 4 moves for messages from one field device to travel to its counterpart in a different stage. As the number of hidden layers is a reflection of a node's scope of visibility over the entire graph with each layer representing one move of message passing, it is suggested and analyzed as a factor shaping the model's performance. For all nodes in the hidden layers, a ReLU activation function is adopted for nonlinear transformation. Finally, the cross entropy loss is adopted for parameter optimization. The training process of the Link Inference Decoder is presented in Algorithm 3.

Note that to efficiently infer the devices' in-depth links using the learnt embeddings, we adopt and compare the following decoding mechanisms:

(a) A Sigmoid function that takes in as input a single pair of learnt embeddings from nodes whose relations are of interest, and outputs a value indicating the odds of linkage, which is accomplished with the scheme below.

$$p(A_{ij}) \leftarrow sigmoid(h_i^T \cdot h_j) \tag{16}$$

In (16), $A_{ij}$ denotes the event that node $i$ and node $j$ are related to each other. $h_i$, $h_j$ are the learnt embeddings for node $i$ and $j$ respectively. Their inner product is fed into the *sigmoid* function generating a value in (0, 1), implying the likelihood of a link's existence. Equilvalent matrix expression shows as thus,

$$\mathrm{P}_{\mathrm{link}} \leftarrow sigmoid(H^T \cdot H) \tag{17}$$

where $\mathrm{P}_{\mathrm{link}}$ is the probability matrix and $H$ is the embedding matrix stacked up with all learnt embeddings.

**Algorithm 3** Link Inference Decoder Training.

```
Input: Base topology $G(V, E_0)$, initial vector representations $x'$s
Output: Learnt graph $G(V, E)$
Ensure:
1: Configure the number of message passing layers $\eta$, upper bound of the number of epochs $ep_{\max}$.
Initialize the epoch index $ep = 0$, the list of weight matrices $[W^{(i)}]$, $i \in [0, \eta) \cap Z$
2: while $ep < ep_{\max}$:
3:     for each weight matrix $W^{(i)}$ with respect to each message passing layer:
4:         $\forall v_i \in V$:
5:             Compute messages from all $v_i'$s neighbours $v_j'$s with current vector representations
               (See (12) or (14))
6:             Aggregate computed messages and update $v_i'$s representation with $W^{(i)}$ and ReLU activation
               (See (13) or (15))
7:     end for
8:     Compute probability matrix $P_{\text{link}}$ (See 17)
9:     Compute cross entropy loss using $P_{\text{link}}$
10:    Perform back propagation
11:    $ep \leftarrow ep + 1$
12: end while
13: Obtain the learnt links $E$ from the finalized $P_{\text{link}}$
14: return $G(V, E)$
```

(b) A Softmax function applied to the output of a fully-connected layer which maps a link embedding to a 2-dimensional vector. Note that a link embedding is computed from the vectors with respect to the link's endpoints via mean and max pooling operations.

In summary, the Link Inference Decoder learns a complex graph reflecting the devices' sophisticated process-oriented associativity. It outputs a new set of edges $E$ that contains not only the a priori knowledge in the base topology, but the newly learnt in-depth relations as well. At this stage, the graph $G(V, E_0)$ the Graph Processor outputs is updated to $G(V, E)$.

## 4.4. Anomaly detector

Using the devices' explicit process-oriented associativity, the Link Inference Decoder produces a graph whose links represent the convoluted relations among the devices in an ICS network. In order to accurately detect anomalous patterns hidden in the numeric data streams with respect to these devices via leveraging the information rendered by the produced graph, a second GNN-based module is built. This module takes in the graph and the nodes' preprocessed temporal expressions (with respect to any jiff of interest) derived from the original data flow, produces embeddings via message passing, and maps the learnt embeddings to their respective labels denoting whether an anomalous pattern is detected. Distinguished from many methods implemented with composite frameworks, this scheme considers both temporal and spatial features with a single architecture, and hence is expected to be both accurate and efficient in anomalous pattern recognition.

To create the preprocessed temporal expression $\chi$ for a specific node $v_i \in G(V, E)$ at a particular time tick $t$, we take the $v_i$'s original numeric stream $S$, and employ a weighted averaging scheme to compress the historical data snippet within a specified interval of length $\mathfrak{N}$ ending at this time tick $t$, denoted as $\mathfrak{S} = S[t - \mathfrak{N} + 1 : t]$. Assuming the influence of a historical numeric value decays with time, an exponential weighted average scheme is implemented so that the produced expression better reflects the temporal characteristics of the original data flow (see (18)). The size of the interval $\mathfrak{N}$ is intuitively configured to be large enough to cover the most necessary temporal sequential features, which is 4300 for SWaT (period values for most periodic flows) and 5000 for WADI (empirically set and subject to tuning). Then every $\mathfrak{n}$(set to 100 in this work) consecutive values in this interval are condensed into a single one via weighted averaging. In this scenario, the preprocessed temporal representations are of size $\frac{\mathfrak{N}}{\mathfrak{n}} = 43$ and 50, with respect to SWaT and WADI datasets.

$$\chi[i] \leftarrow \sum_{j=i\mathfrak{n}}^{(i+1)\mathfrak{n}-1} e^{-a(\mathfrak{n}-j)} \mathfrak{S}[j], \ i \in [0, \frac{\mathfrak{N}}{\mathfrak{n}}) \cap Z \tag{18}$$

Formula (18) shows how each of $\chi$'s elements is computed. In (18), values of $\mathfrak{n}$ consecutive time ticks are integrated as one to achieve temporal compression. Note that the coefficient $a$ is a balancing factor shaping the amplitude of the exponential weights, which is set as 4 in this work.

Using the graph $G(V, E)$ obtained from the Link Inference Decoder and the devices' preprocessed temporal representations $\chi$'s, the Anomaly Detector conducts embedding learning via message passing. Similar to the Link Inference Decoder, we adopt efficient GNN blocks (GCN, GAT and GraphSAGE) as our message aggregators. The number of neurons in each hidden layer is set to 128, and the number of layers varies from 2 to 4. A ReLU is employed for nonlinear activation and cross entropy loss is adopted for training. Instead of being fused in a joint decoder, every learnt embedding $\mathfrak{h}$ is propelled into its exclusive Softmax layer to derive a label $\hat{y}$ (0's for normal and1's otherwise). In this fashion, each label is expected to denote the state of a particular device at any applicable time of interest, and hence device-level anomalous pattern recognition is achieved via distributed decoding. The Anomaly Detector's training process is described in Algorithm 4.

**Algorithm 4** Anomaly Detector Training.

```
Input: Learnt graph $G(V, E)$, preprocessed temporal representations $\chi'$s for all devices
Output: Derived states $\hat{Y}$
Ensure:
1: Configure the number of message passing layers $\eta$, upper bound of the number of epochs $ep_{\max}$.
Initialize the epoch index $ep = 0$, the list of weight matrices $[W_{ad}^{(i)}]$, $i \in [0, \eta) \cap Z$
Initialize $\hat{Y} \leftarrow$ empty list
2: while $ep < ep_{\max}$:
3:     for each weight matrix $W_{ad}^{(i)}$ with respect to each message passing layer:
4:         $\forall v_i \in V$:
5:             Compute messages from all $v_i'$s neighbours $v_j'$s with current vector representations
6:             Aggregate computed messages and update $v_i'$s representation with $W_{ad}^{(i)}$ and ReLU activation
7:     end for
8:     for each learnt embedding $\mathfrak{h}_i$ with respect to device $v_i$:
9:         $\hat{y}_i \leftarrow softmax(\mathfrak{h}_i)$
10:        $\hat{Y} \leftarrow \hat{Y} + \hat{y}_i$
11:    end for
12:    Compute cross entropy loss using $\hat{Y}$
13:    Perform back propagation
14:    $ep \leftarrow ep + 1$
15: end while
16: return $\hat{Y}$
```

**Table 6**
Dataset Characteristics.

| Dataset | #Stages | #Field Devices | Duration with Attack | #Samples | Anomaly Ratio after Temporal Condensation |
|---|---|---|---|---|---|
| SWaT | 6 | 51 | 4 days | 449,919 | 0.3238 |
| WADI | 5 | 123 | 2 days | 172,801 | 0.3159 |

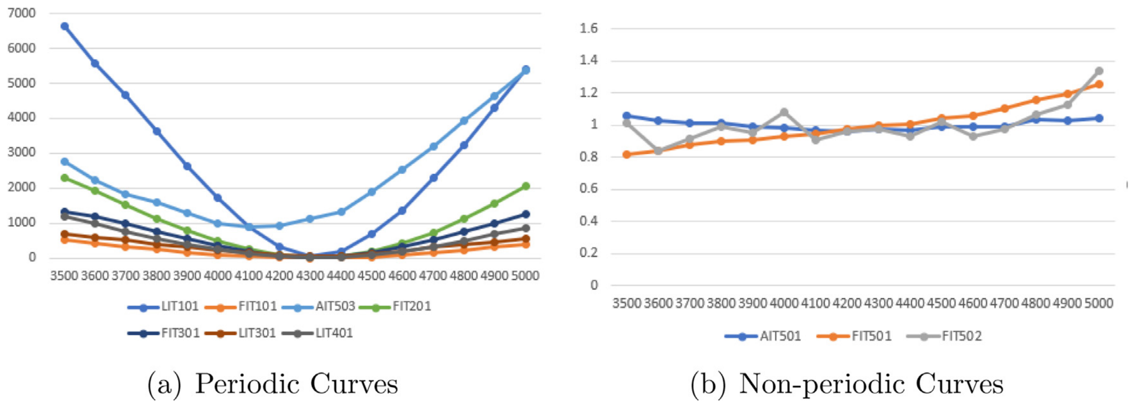


**Fig. 6.** $\Lambda^{(Var)} - T$ Curves (SWaT Showcases).

## 5. Evaluation

Evaluation is conducted over the following open source datasets: the Secure Water Treatment dataset (SWaT) and the Water Distribution dataset (WADI). Both datasets are developed on testbeds built for the simulation of multi-stage industrial processes (water treatment for SWaT and water distribution for WADI). Each dataset contains columns of numeric streams, each of which associated with a particular device. The values in every column are sampled with a 1-second interval for both datasets. The core features of the datasets are presented in Table 6.

### *5.1. Feature extractor evaluation*

As illustrated in Section 4, we introduce an IE-based methodology to derive the cyclical properties in a device's numeric stream. Periodicity is extrapolated and specific period values are observed at the point where IE's standard deviation falls to a local valley. The variance curves $\Lambda^{(Var)} - T$ of devices exhibiting periodic and non-periodic properties are presented in Fig. 6. We observe that the local convexity of the $\Lambda^{(Var)} - T$ curves for particular devices (e.g. LIT-101) is apparent in Fig. 6, and thus the period values with respect to these devices are determined as the local minima of the curves and serve as a basis for the computation of other periodic features. The LIT-101 in SWaT, for example, takes 4300 as its approximated period.

### *5.2. Link inference accuracy*

The Link Inference Decoder is implemented and trained with multiple types of classic GNN blocks: GCN, GAT and G-SAGE. During the training process, the loss converges rapidly for all three GNN scenarios (GCN, GAT and G-SAGE). Taking the GCN as an example, by tuning the number of GCN layers (2, 3, 4 layers) and performing a 20% dropout, its loss curves are exemplified in Fig. 7.

The performances of the Link Inference Decoders trained with different encoding and decoding blocks are compared against the link prediction baselines below: (a) the Bayesian Classifier, (b) the Nonlinear Support Vector Classifier (SVC), (c) the principal component analysis (PCA) and (d) the Cosine Similarity Analysis. The results are shown in Table 7.

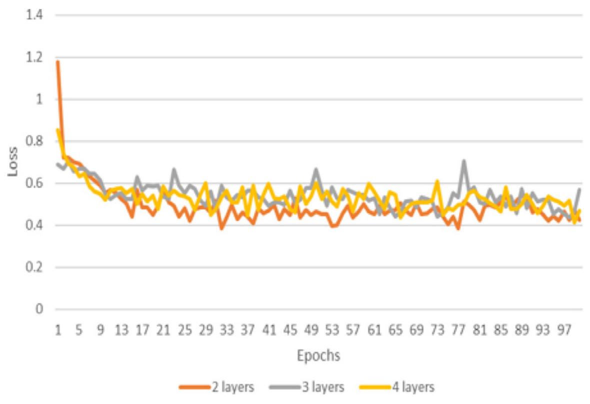


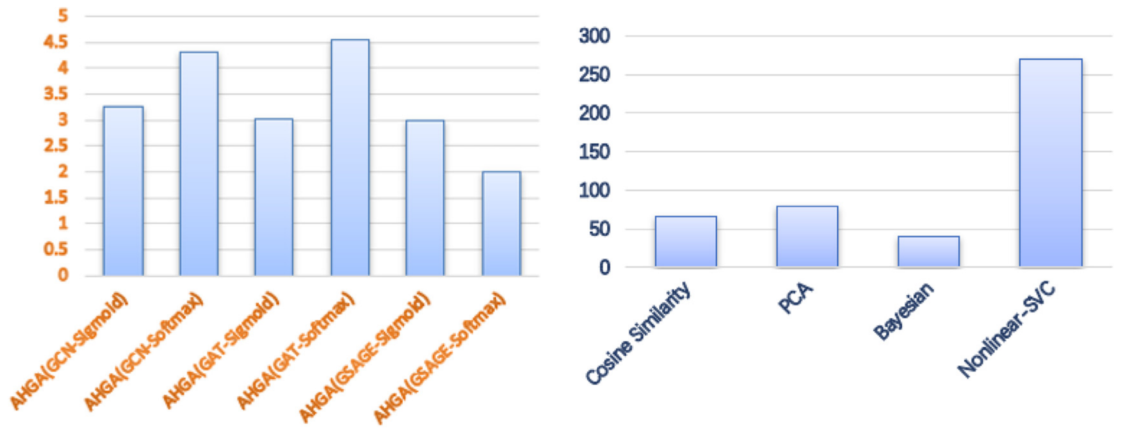
**Fig. 7.** Link Inference Decoder Training Loss Curves.

**Table 7**
Link Inference Baseline Comparison (Note: The message passing block adopted in the link inference decoder is shown in parenthesis. e.g. AHGA (GCN) indicates that GCN blocks are implemented).

| Methods | Decoder | Accuracy | Precision | Recall | F1 |
|---|---|---|---|---|---|
| AHGA(GCN) | Sigmoid | 0.7753 | 0.6929 | **0.9887** | 0.8148 |
| | Softmax | 0.7977 | 0.7431 | 0.9101 | 0.8181 |
| AHGA(GAT) | Sigmoid | 0.7921 | 0.7241 | 0.9438 | **0.8195** |
| | Softmax | 0.7753 | 0.7059 | 0.9438 | 0.8077 |
| AHGA(GSAGE) | Sigmoid | 0.7809 | 0.7049 | 0.9663 | 0.8152 |
| | Softmax | 0.7865 | **0.7525** | 0.8539 | 0.8000 |
| PCA | N/A | 0.7564 | 0.2449 | 0.0723 | 0.1116 |
| Nonlinear-SVC | N/A | 0.7398 | 0.1000 | 0.0938 | 0.0968 |
| Bayesian | N/A | **0.8517** | 0.6731 | 0.5072 | 0.5785 |
| Cosine Similarity | N/A | 0.8452 | 0.5776 | 0.3121 | 0.4052 |

Our observations are as follows:

1) In terms of decoding mechanisms, the AHGA models implementing sigmoid inference decoders tend to exhibit an outstanding rate of recall (mostly above 0.9), while suffering from a relatively inferior precision (approximately between 0.60 and 0.75). This indicates that with sigmoid decoders employed, the AHGA is capable of preserving the majority of the a priori linkage knowledge suggested in the base topology, learning most of the connectivity reflected by the process-oriented conditions



(a) Time Consumption for the AHGAs (in $ms$) (b) Time Consumption for Other Baselines (in $ms$)

**Fig. 8.** Correlation Inference Efficiency.

(which leads to a high recall), as well as implying new in-depth associations the aforementioned conditions fail to cover, which is why the precision rate is relatively low. From the perspective of link learning, this result is desirable as additional sophisticated links are successfully learnt from the devices' explicit process-oriented relations. The AHGAs employing Softmax decoders, on the other hand, produce a more balanced precision/recall output. Nevertheless, their ability to generalize over the a priori knowledge is not as great compared to the models using sigmoid decoders, as reflected by their rates of recall.

2) With all heterogeneous devices profiled via fusion of specific properties into their vector representations, the model's interpretability is improved, as every vector expression explicitly reflects how some device functions in specified industrial processes, as well as how it impacts other devices in the network. The learnt graph, in this case, is better tailored to the industrial processes, rather than some random topology which cannot be interpreted. It is implied that this graph provides a more reliable basis for subsequent anomalous pattern recognition.
3) Among all the baselines displayed in Table 7, the AHGA approaches generally outperform the rest with an F1 gain of approximately 22.15%. This discrepancy resides in the methods' operational principles. The Cosine Similarity, for instance, takes the devices' vectorized profiles and directly performs pair-wise similarity computation without considering any relational characteristics in specified industrial processes. The Bayesian and Nonlinear-SVC, on the other hand, fabricates link vectors using the devices' profiles and conduct classification over all the links. Similarly, these approaches treat each link as an independent entity and do not leverage the devices' process-specific associativity. Therefore, it is claimed that the AHGA is a better fit to deriving relations in specified industrial scenarios.

### 5.3. Link inference efficiency

In order to assess the AHGA's graph learning efficiency, we measure the amount of test time consumed with respect to all AHGA variations and other baseline methods (See Fig. 8).

Observation implies that the Softmax-based AHGAs are approximately 33% less efficient with a decoding time of around 4.5 msecs than the ones using a sigmoid decoder, which takes about 3 msecs. The Graph-SAGE being an exception, it is intuitively assumed that the Softmax decoder should work longer due to the additional fully-connected layer as well as the required exponential computation in the Softmax function. In comparison with other baselines, the AHGAs are very efficient at test phase despite their complex training workflows. This may arise from the difference between the AHGAs and other baselines stated as thus: Once trained, the AHGAs enable link inference via only a few steps of operations on the matrix level, which is minor compared to other methods that have to perform element-wise computations (e.g. the pair-wise similarity computation of 2 specific node vectors in cosine similarity, as well as the element-wise input processing in Bayesian and Nonlinear-SVC).

### 5.4. Anomaly detection accuracy

Implemented with distinct message passing blocks in the Anomaly Detector, the AHGA's ability to recognize anomalous patterns is evaluated against the baselines as follows: (a) K-Means, (b) Regular Graph Convolutional Network (GCN), (c) Graph Attention Network (GAT), (d) Graph Sample and Aggregate Model (GraphSAGE), (e) Topology Adaptive GCN (TAGCN), (f) Flow-Topology GCN (FT-GCN), (g) Isolation Forest and (h) One-class SVM (OCSVM). As applicable, the graph inputs for all GNN baselines are initialized as Fig. 4. The results are summarized in Table 8.

The following insights are obvious:

1) The AHGAs outperform their GNN counterparts (GCN, GAT, and GSAGE) in most evaluation metrics. For example, the AHGA using GSAGE detector achieves an F1 gain of approximately 4.47% (SWaT) and 13.64% (WADI) over the regular GSAGE model. It is speculated that this improvement arises from the addition of the Link Inference Decoder which incorporates the devices' sophisticated associativity in the graph used for anomalous pattern recognition. In this graph, any pair of devices deemed related with each other are directly linked. In this case, the availability of information from related devices is drastically increased as it takes only one message passing step to obtain, and therefore, this information can be more directly and sufficiently encoded in a node's embedding vector used by the AHGA to detect anomalous patterns. As to the GCN, GAT and GSAGE where related devices are not immediately connected, a node's embedding vector suffers from a loss of in-depth correlational information due to the redundant message passing iterations across irrelevant nodes, and therefore leads to less desirable pattern recognition results.

**Table 8**
Anomaly Detection Baseline Comparison.

| Methods | SWaT | | | | WADI | | | |
|---|---|---|---|---|---|---|---|---|
| | Accuracy | Recall | Precision | F1 | Accuracy | Recall | Precision | F1 |
| AHGA(GCN)-GCN | 0.7838 | 0.6165 | 0.8009 | 0.6967 | 0.8207 | 0.7330 | 0.8138 | 0.7713 |
| AHGA(GCN)-GAT | 0.7883 | 0.6072 | 0.8855 | 0.7204 | 0.7495 | 0.5794 | **0.8292** | 0.6822 |
| AHGA(GCN)-GSAGE | **0.8198** | **0.6675** | **0.8918** | **0.7635** | **0.8325** | **0.7589** | 0.8196 | **0.7881** |
| GCN | 0.7687 | 0.5733 | 0.8587 | 0.6875 | 0.7189 | 0.5260 | 0.7859 | 0.6302 |
| GAT | 0.7888 | 0.6106 | 0.8704 | 0.7177 | 0.7327 | 0.5499 | 0.8161 | 0.6570 |
| GSAGE | 0.7887 | 0.6103 | 0.8741 | 0.7188 | 0.7295 | 0.5442 | 0.8122 | 0.6517 |
| TAGCN | 0.7630 | 0.5538 | 0.8692 | 0.6766 | 0.7277 | 0.5372 | 0.8115 | 0.6465 |
| FT-GCN | 0.7641 | 0.5587 | 0.8754 | 0.6820 | 0.7345 | 0.5383 | 0.7528 | 0.6277 |
| OCSVM | 0.6519 | 0.5275 | 0.5321 | 0.5298 | 0.6466 | 0.5352 | 0.5371 | 0.5361 |
| Isolation Forest | 0.6278 | 0.5271 | 0.5270 | 0.5270 | 0.6172 | 0.5393 | 0.5402 | 0.5398 |
| K-Means | 0.6344 | 0.5295 | 0.5304 | 0.5299 | 0.6145 | 0.5122 | 0.5136 | 0.5129 |

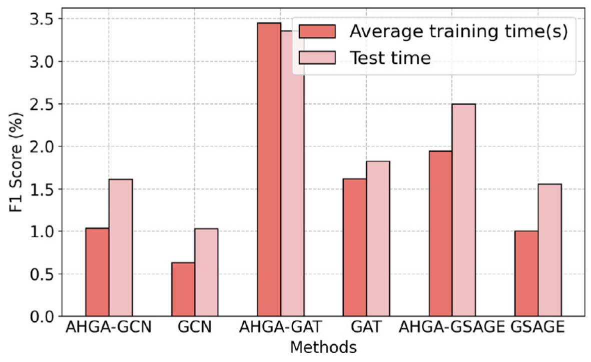


**Fig. 9.** Anomaly Detection Efficiency.

2) In comparison with TAGCN and FT-GCN, the AHGA also achieves superior results. In TAGCN, multiple convolutional kernels are adopted to extract a node's local features from neighbours of distinct distances. Although this enables the TAGCN to update a device's profile with information of distant nodes, the amount of such information obtained in this manner is not as sufficient as direct linkage implemented in the AHGA given identical message passing settings. Similar statements apply to the FT-GCN which adopts multiple TAGCN channels as its core encoding mechanism. Even though the FT-GCN's overall performance is slightly better than the TAGCN for its multi-view setting, the correlations among distinct devices are not sufficiently learnt compared to the AHGAs.
3) The AHGAs' performance also surpasses the outlier detectors (OCSVM, Isolation forest) and the K-Means clustering method, in that the profile of each device is directly used for boundary computation in OCSVM, IForest and K-Means, with no device-wise relationships taken into account. In this fashion, these methods do not effectively adapt to the ICS scenarios in which devices are highly associated.

### *5.5. Anomaly detection efficiency*

Finally, we assess AHGA's runtime efficiency in anomalous pattern recognition (training/test time consumption) and the results are illustrated in Fig. 9. It is apparent that the AHGAs are half as efficient as their counterpart GNN models in runtime. This results from the difference in the graph's linkage complexity. As the AHGAs has more links in the graph that represent the devices' in-depth process-oriented relations, it is not surprising that they are more time-consuming than the GNN baselines without graph learning. However, it is a drawback that needs to be resolved to meet the real-time requirements of ICS anomalous pattern detectors.

## 6. Conclusion

In this work, we investigate the problem of anomalous pattern recognition in the data streams of the ICS devices. A comprehensive framework named the AHGA is proposed, which integrates a graph processor, a feature analyzer, a link inference decoder and an anomaly detector. The AHGA has merit compared to other models in that it captures the in-depth associativity among heterogeneous ICS devices using the explicit process-oriented relations among them. This associativity exhibits more interpretability given it is derived from the devices' explicit features reflecting how they function in the specified industrial processes as well as how they may influence each other. Using the learnt associativity, the devices are better aware of how they are related to the rest of the network, which serve as a more reliable basis for anomalous pattern recognition. Furthermore, in terms of granularity, the AHGA is able to conduct device-wise anomalous pattern detection, differentiated from most of the NN baselines operating on the coarse-grained system's level. Our evaluation demonstrates the AHGA's superiority in anomalous pattern detection, with an F1 gain of 4.47% and 13.17% over the current baselines with respect to the SWaT and WADI datasets. The drawback of the AHGA lies in its extra runtime consumption caused by the increasing complexity of the learnt graph reflecting the devices' in-depth relations. Moreover, as the a priori knowledge for different industrial processes can be rather distinctive, it is challenging to create to a set of conditions that generalizes over all processes. To address these deficiencies, our future work involves studying the commonality of disparate industrial scenarios, and attempting to derive a general set of conditions for graph learning that applies to all or the majority of the types of industrial processes. In addition, we shall develop schemes of re-regulating message passing paths and reducing the quantity of nodes (via clustering or other approaches) to optimize the AHGA's runtime efficiency.

## Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Data availability

The data is available on request to iTrust, who creates the datasets used in the paper.

## Acknowledgement

This work is supported by the National Key R&D Program of China (2020YFB2009502) and the National Natural Science Foundation of China (62272129). Dataset is provided by iTrust Centre for Research in Cyber Security (https://itrust.sutd.edu.sg/dataset/).

## References

[1] M. AlMedires, M. AlMaiah, Cybersecurity in industrial control system (ICS), in: 2021 International Conference on Information Technology (ICIT), IEEE, 2021, pp. 640–647.
[2] M. Mbow, K. Sakurai, H. Koide, Advances in adversarial attacks and defenses in intrusion detection system: a survey, in: Science of Cyber Security-SciSec 2022 Workshops: AI-CryptoSec, TA-BC-NFT, and MathSci-Qsafe 2022, Matsue, Japan, August 10–12, 2022, Revised Selected Papers, Springer, 2023, pp. 196–212.
[3] J. Yu, Y. Zhang, Challenges and opportunities of deep learning-based process fault detection and diagnosis: a review, Neural Comput. Appl. 35 (1) (2023) 211–252.
[4] J. Yang, C. Zhou, Y.-C. Tian, S.-H. Yang, A software-defined security approach for securing field zones in industrial control systems, IEEE Access 7 (2019) 87002–87016.
[5] A.M.Y. Koay, R.K.L. Ko, H. Hettema, K. Radke, Machine learning in industrial control system (ICS) security: current landscape, opportunities and challenges, J. Intell. Inf. Syst. (2022) 1–29.
[6] B. Kim, M.A. Alawami, E. Kim, S. Oh, J. Park, H. Kim, A comparative study of time series anomaly detection models for industrial control systems, Sensors 23 (3) (2023) 1310.
[7] Z. Qian, K. Huang, Q.-F. Wang, X.-Y. Zhang, A survey of robust adversarial training in pattern recognition: fundamental, theory, and methodologies, Pattern Recognit. 131 (2022) 108889.
[8] M. Abdallah, N. An Le Khac, H. Jahromi, A. Delia Jurcut, A hybrid CNN-LSTM based approach for anomaly detection systems in SDNs, in: The 16th International Conference on Availability, Reliability and Security, 2021, pp. 1–7.
[9] J. Sinha, M. Manollas, Efficient deep CNN-BILSTM model for network intrusion detection, in: Proceedings of the 2020 3rd International Conference on Artificial Intelligence and Pattern Recognition, 2020, pp. 223–231.
[10] Y. Wu, H.-N. Dai, H. Tang, Graph neural networks for anomaly detection in industrial internet of things, IEEE Internet Things J. (2021).
[11] H. Kim, B.S. Lee, W.-Y. Shin, S. Lim, Graph anomaly detection with graph neural networks: current status and challenges, IEEE Access (2022).
[12] A. Deng, B. Hooi, Graph neural network-based anomaly detection in multivariate time series, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 35, 2021, pp. 4027–4035.
[13] X. Deng, J. Zhu, X. Pei, L. Zhang, Z. Ling, K. Xue, Flow topology-based graph convolutional network for intrusion detection in label-limited iot networks, IEEE Trans. Netw. Serv. Manage. (2022).
[14] M.A. Umer, K.N. Junejo, M.T. Jilani, A.P. Mathur, Machine learning for intrusion detection in industrial control systems: applications, challenges, and recommendations, Int. J. Crit. Infrastruct. Prot. (2022) 100516.
[15] P. Arora, B. Kaur, M.A. Teixeira, Security in industrial control systems using machine learning algorithms: an overview, ICT Anal. Appl. (2022) 359–368.
[16] B.A. Tama, S.Y. Lee, S. Lee, A systematic mapping study and empirical comparison of data-driven intrusion detection techniques in industrial control networks, Arch. Comput. Methods Eng. 29 (7) (2022) 5353–5380.
[17] X. Zhang, J. Mu, X. Zhang, H. Liu, L. Zong, Y. Li, Deep anomaly detection with self-supervised learning and adversarial training, Pattern Recognit. 121 (2022) 108234.
[18] J. Shen, N. Robertson, Bbas: towards large scale effective ensemble adversarial attacks against deep neural network learning, Inf. Sci. (Ny) 569 (2021) 469–478.
[19] J.-R. Jiang, Y.-T. Lin, Deep learning anomaly classification using multi-attention residual blocks for industrial control systems, Sensors 22 (23) (2022) 9084.
[20] C. Karras, A. Karras, S. Sioutas, Pattern recognition and event detection on iot data-streams, arXiv preprint arXiv:2203.01114 (2022).
[21] A. Dairi, F. Harrou, B. Bouyeddou, S.-M. Senouci, Y. Sun, Semi-supervised deep learning-driven anomaly detection schemes for cyber-attack detection in smart grids, in: Power Systems Cybersecurity: Methods, Concepts, and Best Practices, Springer, 2023, pp. 265–295.
[22] S. Rao, M. Ghaderi, H. Zhang, CloudPAD: managed anomaly detection for ICS, in: Proceedings of the 4th Workshop on CPS & IoT Security and Privacy, 2022, pp. 55–61.
[23] E.A. Boateng, J.W. Bruce, D.A. Talbert, Anomaly detection for a water treatment system based on one-class neural network, IEEE Access 10 (2022) 115179–115191.
[24] S. Li, J. Liu, Z. Pan, S. Lv, S. Si, L. Sun, Anomaly detection based on robust spatial-temporal modeling for industrial control systems, in: 2022 IEEE 19th International Conference on Mobile Ad Hoc and Smart Systems (MASS), IEEE, 2022, pp. 355–363.
[25] H.T. Truong, B.P. Ta, Q.A. Le, D.M. Nguyen, C.T. Le, H.X. Nguyen, H.T. Do, H.T. Nguyen, K.P. Tran, Light-weight federated learning-based anomaly detection for time-series data in industrial control systems, Comput. Ind. 140 (2022) 103692.
[26] D.C.L. Sung, G.R. MR, A.P. Mathur, Design-knowledge in learning plant dynamics for detecting process anomalies in water treatment plants, Comput. Secur. 113 (2022) 102532.
[27] J. Serey, M. Alfaro, G. Fuertes, M. Vargas, C. Durán, R. Ternero, R. Rivera, J. Sabattin, Pattern recognition and deep learning technologies, enablers of industry 4.0, and their role in engineering research, Symmetry (Basel) 15 (2) (2023) 535.
[28] A. Sankar, X. Zhang, K.C.-C. Chang, Meta-GNN: metagraph neural network for semi-supervised learning in attributed heterogeneous information networks, in: Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, 2019, pp. 137–144.
[29] Y. Wang, J. Zhang, S. Guo, H. Yin, C. Li, H. Chen, Decoupling representation learning and classification for gnn-based anomaly detection, in: Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval, 2021, pp. 1239–1248.
[30] Z. Ning, Z. Jiang, H. Miao, L. Wang, Mst-gnn: a multi-scale temporal-enhanced graph neural network for anomaly detection in multivariate time series, in: Web and Big Data: 6th International Joint Conference, APWeb-WAIM 2022, Nanjing, China, November 25–27, 2022, Proceedings, Part I, Springer, 2023, pp. 382–390.
[31] L. Peng, Y. Mo, J. Xu, J. Shen, X. Shi, X. Li, H.T. Shen, X. Zhu, Grlc: graph representation learning with constraints, IEEE Trans. Neural Netw. Learn. Syst. (2023).
[32] L. Chang, P. Branco, Graph-based solutions with residuals for intrusion detection: the modified e-graphsage and e-resgat algorithms, arXiv preprint arXiv:2111.13597 (2021).
[33] B. Jiang, B. Wang, B. Luo, Sparse norm regularized attribute selection for graph neural networks, Pattern Recognit. 137 (2023) 109265.
[34] L. Shuaiyi, K. Wang, L. Zhang, B. Wang, Global-local integration for GNN-based anomalous device state detection in industrial control systems, Expert Syst. Appl. 209 (2022) 118345.
[35] A. Gillioz, K. Riesen, Graph reduction neural networks for structural pattern recognition, in: Structural, Syntactic, and Statistical Pattern Recognition: Joint IAPR International Workshops, S+ SSPR 2022, Montreal, QC, Canada, August 26–27, 2022, Proceedings, Springer, 2023, pp. 64–73.

**Shuaiyi L(y)u** received his M.S.E degree in Electrical and Electronic Engineering from the University of Pennsylvania, the US. He is currently a Ph.D candidate in Cybersecurity at Harbin Institute of Technology (HIT), China. His research is focused on anomaly detection model design and optimization in industrial control systems (ICSs).

**Kai Wang** is currently an Associate Professor with Faculty of Computing, Harbin Institute of Technology (HIT), China. Before joined HIT, he was a Postdoctor in Computer Science and Technology at Tsinghua University. He received his B.S. and Ph.D. degree from Beijing Jiaotong University. His current research interest is mainly on In-vehicle Intrusion Detection, V2X Security, Deep Learning, etc. He has published more than 30 papers in prestigious international journals and conferences (e.g., ACM Transactions on Internet Technology, IEEE Systems Journal, IEEE Network), and serves as the TPC Member and technical reviewer for many important international conferences and journals (e.g., ACM Computing Surveys, IEEE Transactions on Industrial Informatics, IEEE Internet of Things Journal, IEEE HotICN 2020). He is a CCF Senior Member.

**Liren Zhang** received his Bachelor of Engineering in Information Security from Harbin Institute of Technology, China. He is currently a master student in Computer Technology at Harbin Institute of Technology (HIT) in China. His research focuses on multi-view learning-based anomaly detection for industrial control systems (ICS).

**Bailing Wang** is currently an Professor with Faculty of Computing, Harbin Institute of Technology (HIT), China. He received his Ph.D. degree from the School of Computer Science and Technology, Harbin Institute of Technology, in 2006. His main research interest is mainly on information content security, industrial control network security, V2X Security, etc. He has published more than 80 papers in prestigious international journals and conferences, and has selected for the China national talent plan.