GNN-based Advanced Feature Integration for ICS Anomaly Detection

SHUAIYI L(Y)U, Harbin Institute of Technology, China
KAI WANG*, Harbin Institute of Technology, Weihai, China
YULIANG WEI, Harbin Institute of Technology, Weihai, China
HONGRI LIU, Harbin Institute of Technology, Weihai, China
QILIN FAN, Chongqing University, China
BAILING WANG*, Harbin Institute of Technology, China

Recent adversaries targeting the Industrial Control Systems (ICSs) have started exploiting their sophisticated inherent contextual semantics such as the data associativity among heterogeneous field devices. In light of the subtlety rendered in these semantics, anomalies triggered by such interactions tend to be extremely covert, hence giving rise to extensive challenges in their detection. Driven by the critical demands of securing ICS processes, a Graph-Neural-Network (GNN) based method is presented to tackle these subtle hostilities by leveraging an ICS's advanced contextual features refined from a universal perspective, rather than exclusively following GNN's conventional local aggregation paradigm. Specifically, we design and implement the Graph Sample-and-Integrate Network (GSIN), a general chained framework performing node-level anomaly detection via advanced feature integration, which combines a node's local awareness with the graph's prominent global properties extracted via process-oriented pooling. The proposed GSIN is evaluated on multiple well-known datasets with different kinds of integration configurations, and results demonstrate its superiority consistently on not only anomaly detection performance (e.g., F1 score and AUPRC) but also runtime efficiency over recent representative baselines.

CCS Concepts: • **Security and privacy** → **Intrusion detection systems**; **Formal methods and theory of security**; • **Networks** → *Network architectures*; • **Computing methodologies** → **Neural networks**; *Classification and regression trees.*

Additional Key Words and Phrases: Advanced Feature Pooling, Embedding Integration, Graph Neural Networks, Anomaly Detection, Industrial Control Systems

1 INTRODUCTION

The Industrial Control Systems (ICSs) have aroused significant global security concerns over the last decade [4], [5], [3]. As demands in real-time industrial process control has been growing by the second, the idea of securing the ICSs via physical segregation in real-world applications has been gradually replaced by remote accessibility enabled via Internet connection. Such measures, in spite of facilitating management in a rather positive manner, undesirably expose the ICSs to the diversified malicious interactions over the Internet, a fair amount of which are specifically crafted to compromise the proper functioning of the underlying industrial workflow, causing considerable economic loss or casualty. Thus, in order to timely counter the adversaries targeting the core running

*Corresponding Author

Authors' addresses: Shuaiyi L(y)u, Harbin Institute of Technology, China, lvshuaiyi2568068@163.com; Kai Wang, Harbin Institute of Technology, Weihai, China, dr.wangkai@hit.edu.cn; Yuliang Wei, Harbin Institute of Technology, Weihai, China, wei.yl@hit.edu.cn; Hongri Liu, Harbin Institute of Technology, Weihai, China, liuhr@hit.edu.cn; Qilin Fan, Chongqing University, China, fanqilin@cqu.edu.cn; Bailing Wang, Harbin Institute of Technology, China, wbl@hit.edu.cn.

2157-6904/2023/9-ART $15.00
https://doi.org/10.1145/3620676

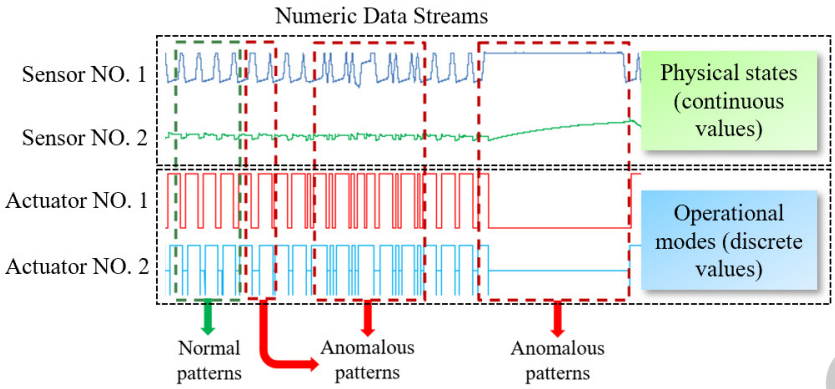


Fig. 1. Numeric streams with respect to process-related field devices

processes, it is imminent to thoroughly investigate the characteristics of the data streams with respect to the process-related field devices, such as sensors and actuators, as well as stipulating a comprehensive scheme to detect the anomalous variational patterns in these streams. Such streams include but are not limited to the following categories: series of physical states measured by the sensors, operational modes of the actuators, etc. (See Fig. 1).

Over the past decades, extensive efforts have been made to secure the ICS processes starting with the traditional approaches aiming at detecting dubious patterns within single field devices or connections [36], [25], [15]. These methods, although easy to deploy, are not capable of coping with recent hostilities that exploit the complex contextual properties among multiple devices, due to the lack of consideration of the ICSs' sophisticated correlational semantics. Therefore, they suffer from serious accuracy issues such as an unacceptably high false alarm rate. To tackle these issues, advanced deep learning based anomaly detection frameworks have emerged at a tremendously rapid rate, including but are not limited to the Convolutional Neural Network (CNN) [1], and Long-Short Term Memory (LSTM) [10] [28], etc. Designed to capture the spatial and temporal correlations within a node's features, these models yield superior detection accuracy as compared to traditional approaches. Nevertheless, as they usually manipilate data points in a way without sufficiently considering their contextual associativity in specific network architectures, their functionality still has potential for improvement.

Recently, the Graph Neural Networks (GNNs) have become a considerably popular trend in the domain of anomaly detection, owing to their powerful ability to capture the contextual semantics in randomly-structured network topologies. These semantics incorporate a diversified range of graph-related features, such as the roles of nodes or edges in the graph, the correlation of nodes and edges, etc. State-of-the-art GNN variations include at least the Graph Convolutional Network (GCN) [21], the Graph Attention Network (GAT) [33], and the Graph Sample-and-Aggregate (Graph-SAGE) [17], etc. These methods conduct local feature aggregation via neighbourhood message passing. Despite being efficient, such a mechanism imposes an attentional bias to a node's immediate surroundings in its contextual awareness. In view of an ICS's heterogeneous nature and its layered structure entailing complex structural semantics that a simple adjacent message aggregation scheme may fail to capture, such a bias is undesirable and potentially deteriorates the model's detection accuracy.

In order to conquer the challenges in accurately extracting complex structural semantics, more advanced methods have been introduced including the Meta-GNN [34], the Graph Deviation Network (GDN) [19], the E-GSAGE [23], etc. Although effective in particular domains, their applicability is strictly limited to the specific type of contextual properties they are designed to discern. As the pool of correlational features in an ICS network is

immense, these approaches rarely make a comprehensive solution that incorporates the majority of the network's core representative features. In an attempt to incorporate the most prominent semantics in anomaly detection tasks, one of our previous works, the Global-Local Integration Network (GLIN) [24] performs feature integration via global pooling. While proved effective, its computation is costly as the scale of the graph increases.

Therefore, in order to address the aforementioned issues, we design and implement a GNN-based anomaly detection scheme named the Graph Sample-and-Integrate Network (GSIN), which deals with node-level state inference tasks by leveraging the graph's prominent properties extracted via partial pooling. The proposed approach is differentiable from existing solutions in that it performs pooling from a subset of a graph's vertices, and that the extracted features make a more appropriate and advanced representation of the graph's contextual profile. In this fashion, not only is the model's anomaly detection performance preserved, but also its runtime efficiency is simultaneously enhanced as the quantity of nodes for pooling and integration is reduced. Specifically, the GSIN comprises: (a) A preprocessor that performs temporal feature encapsulation. It transforms the original data flow into a set of vector expressions to be fed into the subsequent module, each of which representing a specific node's temporal features at a particular time tick of interest. (b) An encoder that creates node embeddings via message passing using the initial vector representations output by the preprocessor, enriching each node with its local awareness. (c) An integrator extracting the graph's universal properties via vector pooling and embedding fusing. As the core building block of the GSIN, the integrator performs semantic extraction and integration using only a subset of the graph's vertices defined via process-oriented pooling, further fertilizing every node with advanced contextual knowledge. (d) A decoder conducting state inference using the nodes' integrated knowledge. It produces labels representing the states of the devices of interest at any applicable time tick.

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

1) We design and develop the GSIN, which performs node-level anomaly detection in ICSs. Apart from the GNN's local aggregation routine, we explore the opportunities of meliorating the model's functionality via advanced contextual feature integration, which is a double-stage process consisting of feature pooling and embedding fusing. Different integration schemes are designed, evaluated and compared using multiple metrics.
2) We evaluate the GSIN over 5 popular open-source datasets (SWaT, WADI, BATADAL, CISS, and CHD), and the model's functionality is comprehensively assessed in terms of multiple metrics.
3) We compare the GSIN's performance against numerous representative baselines. Results demonstrate the GSIN's superiority in terms of both the detection accuracy and its runtime efficiency.

The rest of the paper is organized as follows. Section 2 overviews the literature relevant to ICS anomaly detection; Section 3 defines the problem to be addressed; In Section 4, we provide a detailed description of the GSIN's architecture; Section 5 summarizes the evaluation results and highlights relevant analysis; Section 6 concludes the paper.

2 RELATED WORK

In this section, we overview the literature related to ICS anomaly detection. Our discussion is categorized into Traditional approaches, Classical Machine Learning and Deep Learning methods, and Graph Neural Network methods.

1) **Traditional methods**:

Early efforts in the literature [4] mostly focus on sequential feature mining on data streams with respect to individual devices. Particularly, considering the procedural consistency in typical industrial processes, the idea of detecting anomalies via periodic semantic mining is prevailing, based on which, numerous methods have been proposed. For example, the DFA-based methods [25] [15] convert the devices' numeric sequences into loops of

states and attempt to detect any behaviours that deviate from the loop. A semantic, network-based intrusion detection system [16] is developed that continuously keeps track of process variables to derive variable-specific prediction models as a basis for the inference of future activities. Shortly thereafter, a sequence-aware intrusion detection reference architecture (S-IDS) [6] is presented to detect sequence attacks, a type of semantic attacks that aims at mingling chaotic sequence of events with normal streams. More recently, a software-defined security (SDSec) approach [35] is proposed to prevent the propagation of attack impact among field zones in the ICSs.

Light-weight and simple as they are, the apparent superficiality of features these traditional methods are built on have proved them unsuitable for detecting adversaries exploiting the graph's advanced contextual semantics.

2) **Classical Machine & Deep Learning Methods**:

The demands for improvement in detection accuracy percipitates in-depth feature exploration using machine and deep learning based methodologies. Hybrid intrusion detection systems [1][10][28] are introduced encapsulating the Convolutional Neural Network and Long-short Term Memory Network. Being able to capture both the spatial and temporal characteristics of an entity's features, the hybrid models are demonstrated feasible on the detection of more advanced attacks (such as the zero-day attacks). In addition, auto-encoder methods [18] [13] are designed to detect anomalous data points via sample reconstruction. A GAN-based approach [7] is developed to perform outlier detection using dual auto-encoders. Apart from the aforementioned methods, a kMeans based OCSVM model [2] is proposed for anomaly classification. Various frameworks utilizing Isolation Forest techniques [30] [32] [20] are created to recognize anomalous patterns in network traffics. An automated fault detection method [11] based on the Twin Support Vector Machine (TWSVM) is proposed to enhance the reliability of data-driven condition monitoring in wind turbine operations.

Apparently, by considering high-level features, the aforementioned methods yields superior results in detection accuracy. Nonetheless, without exploring the topological features of the specified industrial processes, the performance of these methods still leaves a lot to be desired.

3) **Graph Neural Network Methods**:

Embedding learning makes up the core functionality of the Graph Neural Networks. Over the past few years, dozens of message passing schemes have been introduced for embedding learning. Graph Attention Network (GAT) [33] assigns trainable weights to a node's neighbourhood to meliorate the model's classification performance. Graph Sample and Aggregate (Graph-SAGE) [17] employs sampling over a node's neighbourhood to enhance the model's operational efficiency. In addition, to leverage high-level contextual properties, more advanced approaches have been proposed [19] [34]. The Graph Deviation Network (GDN) [8] is introduced to explore the correlational properties among sensors in high-dimensional time series data and create GAT-based anomaly detectors using the learnt correlations. Apart from the aforementioned message aggregation schemes on the basis of individual nodes, Meta-GNN [26] is introduced exclusively for the analysis of attributed heterogeneous information network. The model is developed to extract features from meta-graph structured neighbourhood so as to capture higher-order semantic relationships in the network. In order to cover the most prominent semantics in ICSs, the Global-Local Integration Network (GLIN) [21] enriches a node's awareness with advanced contextual knowledge extracted via global pooling. To adapt the GNN-based anomaly detectors to specific industrial processes, the Attributed Heterogeneous Graph Analyzer (AHGA) [27] generates the graph for downstream anomalous pattern recognition using the nodes' process-oriented properties.

Although the solutions above have been demonstrated effective in many applications, they suffer from limitations ranging from the lack of semantic comprehensiveness to insufficiency in runtime conservation. Therefore, as differentiated from previous methods, to tackle the semantic issues, our proposed solution attempts to extract a network's contextual properties from a universal perspective, rather than discriminate against features of specific categories. Moreover, instead of adopting global pooling in existing approaches, the presented framework defines schemes to narrow the pooling range, which presumably makes the process less time-consuming. In summary,

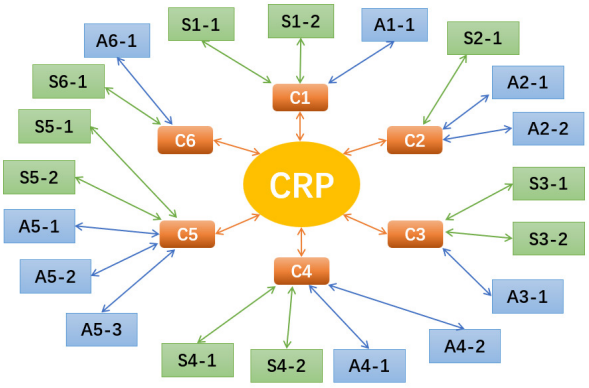


Fig. 2. Star-Shaped Topology (C_i refers to the i-th controller in the system, A_{i-j} and S_{i-j} are the j-th actuator and sensor linked to C_i, respectively)

our work extends the semantic integration idea of encoding a graph's advanced contextual features to ameliorate both the model's anomaly detection accuracy and its runtime efficiency.

3 PROBLEM STATEMENT

As our primary task, constructing a GNN-based model to achieve device-level anomaly detection demands prior investigation in the devices' mutual connectivity, based on which the model's graph input $G(V, E)$ is defined. V and E corresponding to G's vertices (nodes) and edges, building G intuitively breaks down to defining sets V and E.

- **Defining** V:
  Given a heterogeneous ICS network entailing devices of all functions, determining V is inextricably associated with the intended main focus of the model, which in our case, refers to detecting anomalous operational states of all process-related field devices. Therefore, every field device is assigned to V as its member. In addition, to preserve the core topological context of the original ICS networks, all controllers are abstracted into nodes in V as well to channel the field devices. Finally, as the rest of the network is of marginal relevance to our main focus, it is condensed into a single node named the Central Reference Point (CRP), which is also added to V.
- **Defining** E:
  In order to regulate the edge set E, we exclusively consider the physical connectivity among V's elements in the respective ICS network. One way of constructing E is described as follows: With the CRP as the starting point, all controller nodes are directly attached to it, and each controller is connected to the field devices (sensors and actuators) under its supervision. The resulting star-shaped topology is illustrated in Fig. 2. Note that this is not the only plausible way to device the graph for model development. Another scheme may simply omit the CRP and model the graph using only the nodes we are tasked to analyze. See Section 5.1.2 for details.

Graph $G(V, E)$ defined, our core problem is stated as follows (See **Definition 1**):

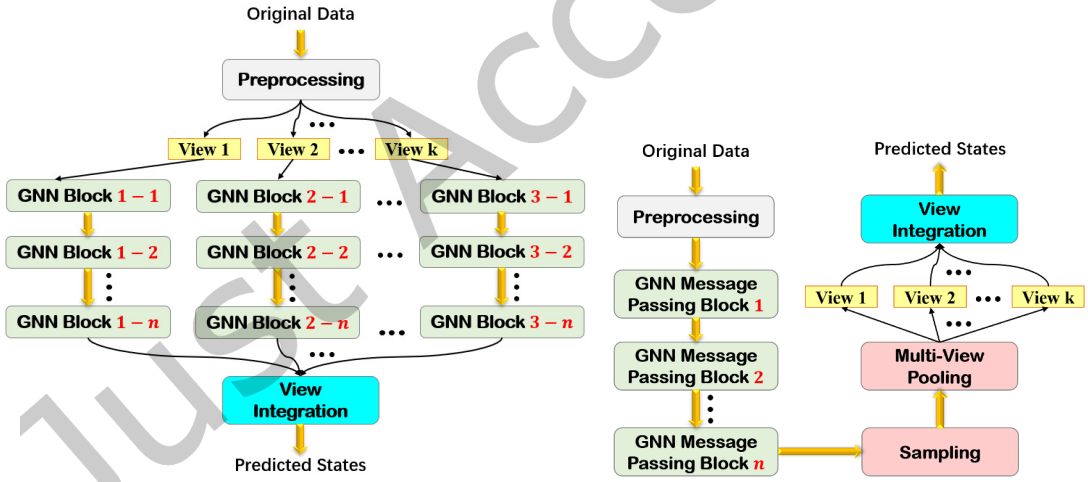
Definition 1. Given the graph $G(V, E)$ and the labeled temporal data sequences $(D(t), Y(t))$ with respect to all nodes in V, create model $M(G, D(t))$ such that for all $t, M(G, D(t)) = Y(t)$.

We conceive our task as a binary classification problem which aims at mapping the original numeric sequences $D(t)$ captured from graph G's node elements to their respective ground-truth operational states $Y(t)$. Specifically, $D(t)$ refers to an encapsulation of all the process-relevant numeric streams with respect to the devices in V (e.g., the numeric values of physical states measured by sensors, or the operational mode of the actuators, etc., see Fig. 1), and $Y(t)$ is a sequence of binary states (0 for normal and 1 otherwise) that reflects the devices' real operational conditions with respect to $D(t)$.

As a potential solution, the GSIN framework proposed in this article is intended to conduct state inference on a fine-grained device-level, and its functionality generalizes over any applicable time tick of interest. As distinguished from current state-of-the-art methods, the GSIN features the following two aspects: 1) multi-view decoding and 2) integration via sampling.

1) **Multi-view decoding:**

The multi-view idea is a paradigm that enables observation of an entity over more than one specified vision. By mining via split channels, the model is intuitively capable of refining information that contains more in-depth and subtle features, and therefore yielding a more universal cognition of the entity, exerting positive influences on the model's functionality. Typical multi-view applications usually separate the original data into disparate channels, each of which is processed with an independent encoding framework. Such approaches usually suffer from a horrendously unfavourable runtime consumption due to an increasing quantity of GNN blocks to be trained. To avert this issue, the GSIN adopts the multi-view concept in its decoding module. It extracts the essences of the information processed by a single encoder via multi-view pooling, groups the essences into distinct channels, treats every channel as an individual view, and performs state inference via view integration (See Fig. 3).



(a) Typical multi-view application (b) GSIN multi-view decoding application

Fig. 3. Stage-wise F1 Performance

2) **View creation via sampling:**

In GSIN, views are generated with the core features pooled from nodes in the node set V. To determine which nodes are to be used for pooling, random sampling is applied to draw a subset of nodes from V instead of using the entire V. Relevant insights are provided as below: As neighbourhood aggregation is usually employed in

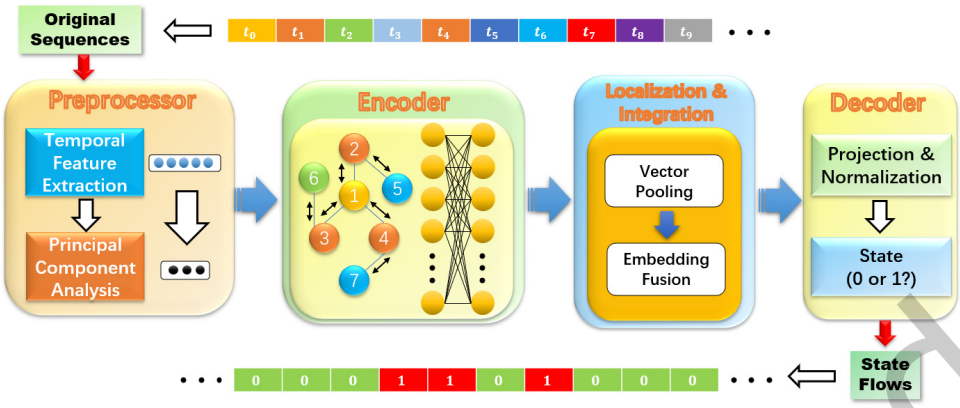


Fig. 4. GSIN Architecture

typical GNN frameworks, the learnt embeddings of adjacent nodes tend to overlap in terms of the contextual information contained. Consequently, pooling from all graph nodes in V may cause a redundancy of certain topological semantics in the resulting pooled messages, negatively impacting the distribution of other core features and affecting the performance of the model that utilizes these features in subsequent inference processes. By introducing sampling, the GSIN is able to moderately remove such redundancy by sparsifying the nodes, which reduces the similarity of the nodes in close proximity. In this fashion, more essential features are preserved in the node embeddings and used by the model to yield more reliable inference results.

4 MODEL DESIGN

In view of current demands for advanced knowledge integration, we design the GSIN, a chained architecture comprising four components, namely a preprocessor, an encoder, an integrator, and a decoder, as shown in Fig. 4. The preprocessor is a temporal condenser that converts the raw data flow into a matrix comprising the initial features to be fed into the encoder, in which these features are updated with local semantics. It is designed to be flexibly configured, and capable of dealing with sequences of all sorts, regardless of the types of information conveyed. The encoder is implemented as a 2-layer GCN block as it is structurally simple and time-conserving. After the encoding step, each node's semantic awareness is further enhanced via incorporation of the graph's high-level contextual characteristics extracted by the integrator, which, as the core building block of the GSIN, performs semantic extraction and integration using only a subset of the graph's vertices defined via process-oriented pooling, rather than the entire topology as in previous methods. Such a measure greatly increases the quality of the node embeddings used for state inference, and thus improves the model's detection performance. Finally, using the representations the integrator produces, the decoder makes inference decisions denoting states of the related devices. The decoding scheme is designed compact and scalable across different number of state types, and is thus favourable to scenarios in which differentiation of multiple anomalous categories is necessary. In this paper, however, only two states are applicable, namely "0" for normal and "1" for abnormal. In general, temporal sequences, while cascading through these building blocks, are converted into state flows denoting device condition at any given timestamp.

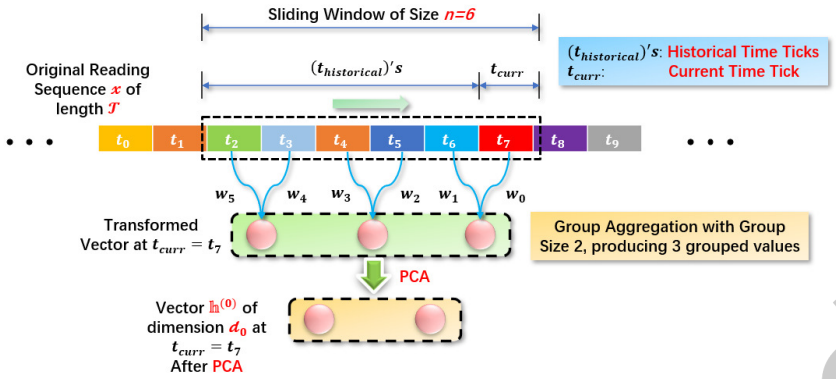


Fig. 5. Sequence-to-Vector Transformation for a Single Node at a Particular Timestamp t_{curr} (**Input**:X_r; **Output**: $H^{(0)}_{r\times d_0}$)

4.1 Preprocessor

To transform the original data into legible input to the GSIN, we develop a preprocessing module that converts the temporal sequences X of shape (r, T) into independent vectors $H^{(0)}$ of shape (r, d_0, T') where r,d_0,T are the device quantity, the dimension of each vector, and the number of original timestamps, respectively. T' is the number of remaining time ticks at the output, whose value depends on the specific data manipulation mechanism. These vectors, derived for all devices, entail sequential characteristics in the current and historical readings at all applicable time ticks. Assuming the impact of historical readings on the current state diminishes with time, we assign exponentially-decaying weights to each reading value before condensing the original data flow via aggregation. To exemplify this, suppose x in X is the original reading sequence of length T captured on a specific device. The transformation process of x is illustrated in Fig. 5.

At the beginning of the process, a window is applied to the original sequence that truncates a series of historical reading snippets ending at the current timestamp of interest. Suppose the window is of size n and moves along the original stream with a step of 1, the numeric relation of T and T' is shown in (1).

$$T' = T - n + 1 \tag{1}$$

In this case, the number of time ticks T' the GSIN is able to process depends on both the length of the original stream T and the designated window size n. In this paper, the value of n is empirically set in accordance with the periodic value that applies to most data streams in a given dataset (e.g. 4300 for SWaT). For the datasets in which cyclicity is vague in the majority of the data sequences, however, a value of 100 is assigned (e.g. BATADAL).

After the window is configured, temporal aggregation is carried out to filter out the noises in the streams that may negatively impact the GSIN in capturing and utilizing the variational properties in the original data sequences. Specifically, a grouped averaging scheme is adopted for aggregation in which weights are computed following the scheme in (2) and assigned to the elements in the window to create an aggregated vector.

$$w_i = \frac{e^{-\frac{a}{n}i}}{\sum_{j=0}^{n-1} e^{-\frac{a}{n}i}}, \ i = 0, 2, ..., n-1 \tag{2}$$

In (2), n is the window size and w_i denotes the weight value assigned to the reading i units before the current time tick. a is the coefficient (set to 4 in this paper) that balances the distribution of all weights in the window. Suppose

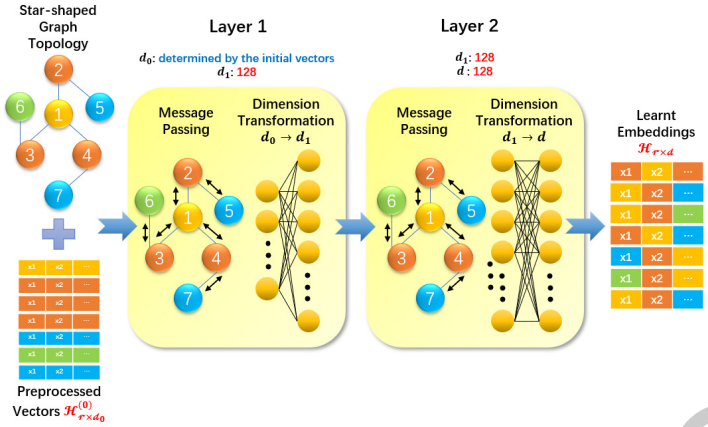


Fig. 6. Encoder Configuration (**Input**: $H^{(0)}_{r\times d_0}$; **Output**: $H_{r\times d}$)

$s_0 = (s_0^{(0)}, s_0^{(1)}, s_0^{(2)}, ..., s_0^{(n-1)})$ and $v_{aggr} = (v_{aggr}^{(0)}, v_{aggr}^{(1)}, v_{aggr}^{(2)}, ..., v_{aggr}^{(m-1)})$ refer to the original window snippet and the aggregated vector, respectively. The aggregation process can be shown as follows (See (3)).

$$v_{aggr}^{(i)} = \sum_{j=(\frac{n}{m})i}^{(\frac{n}{m})(i+1)-1} s_0^{(j)} w_j, \ i = 0, 1, ..., m-1 \tag{3}$$

In (3), the aggregation is completed via grouped averaging and m is the size of the resulting aggregated vector. The value of m indicates the extent to which the original data snippet is condensed. It is desirable that it is not too high or too low in order to preserve a moderate amount of original information after temporal compression. In this paper, we have $m = \frac{1}{100}n$ for large windows (size of 1000 or larger) and $m = \frac{1}{10}n$ for relatively smaller ones (size of within 1000).

The aggregated vector v_{aggr} is subsequently fed into a PCA block, which turns it into a d_0-dimensional row vector $h^{(0)}$. Then the $h^{(0)}$ of all r devices are stacked together, generating $H^{(0)}$ of size $r \times d_0$. Note that this $H^{(0)}$ is exclusive to a single time tick. However, since the number of timestamps only matters in the batching of the dataset, it is omitted in the discussion of the model's structure for the rest of the Section. Hence, $H^{(0)}_{r\times d_0}$ is treated as the input of subsequent modules.

4.2 Encoder

The encoder converts via message passing the vectors $H^{(0)}_{r\times d_0}$ generated by the preprocessing module to convoluted embeddings $H_{r\times d_0}$ rich in local semantics, enabling the new representations to serve as a reflection of the nodes' local awareness, prompting the embeddings to become more informative regarding how the nodes exist in the universal context, which facilitates potential increase in the model's inference (decoding) accuracy. Utilizing the star-shaped topology in Fig. 1, vectors obtained at each specific time tick are grouped into one graph unit for message passing, and multiple graph units may constitute a larger topological version for the purpose of mini-batch training. Our encoder comprises 2 message passing layers whose hyperparameter settings are illustrated in Fig. 6. For simplicity, it employs GCN as its message passing scheme, adopts ReLU as the nonlinear activation operator, and outputs node embeddings of 128 dimensions. Given the trainable weight matrices $W^{(1)}_{d_0\times d_1}$ and $W^{(2)}_{d_1\times d}$, the adjacency matrix $A_{r\times r}$ and its degree matrix D, the message passing procedure is represented in (4), (5) and

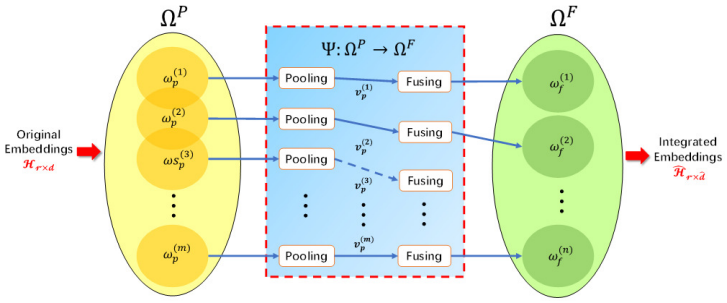


Fig. 7. Integrator Configuration (**Input**: $H_{r\times d}$; **Output**: $\widehat{H}_{r\times \widehat{d}}$; Note: $v_p^{(i)}$ is the pooled vector produced from $\omega_p^{(i)}$)

(6)

$$H_{r\times d_1}^{(1)} \leftarrow \mathrm{ReLU}(\widetilde{A}_{r\times r} H_{r\times d_0}^{(0)} W_{d_0\times d_1}^{(1)}) \tag{4}$$

$$H_{r\times d} \leftarrow \mathrm{ReLU}(\widetilde{A}_{r\times r} H_{r\times d_1}^{(1)} W_{d_1\times d}^{(2)}) \tag{5}$$

$$\widetilde{A}_{r\times r} \leftarrow D^{-0.5} A_{r\times r} D^{-0.5} \tag{6}$$

4.3 Integrator

As opposed to regular GNN paradigm, the GSIN conducts additional embedding integration via vector pooling and embedding fusion (See Fig. 7), transforming $H_{r\times d}$ into $\widehat{H}_{r\times \widehat{d}}$ before preceding to label prediction. This operation differentiates the GSIN from current approaches in that each node's awareness of its surroundings is reinforced with prominent universal properties, while in regular GNN methods, nodes solely broaden their horizon via adjacency aggregation. However, instead of directly considering every node in the production of the global expression that encapsulates these universal properties, the GSIN's focus is to minimize the node sets preserving the graph's most significant global semantics, while balancing the model's detection accuracy and runtime efficiency. Particularly, given a graph $G(V, E)$, the Integrator defines the following: the pooling sets $\Omega^P = \left\{\omega_p^{(i)} | \forall i \leq m, \omega_p^{(i)} \subseteq V\right\}$, the fusing sets $\Omega^F = \{\omega_f^{(j)} | \forall j \leq n, \omega_f^{(j)} \subseteq V, and \bigcup_j \omega_f^{(j)} = V, and \forall k \neq j, \omega_f^{(k)} \cap \omega_f^{(j)} = \emptyset\}$, and the mapping relation $\Psi : \Omega^P \to \Omega^F$. Each set $\omega_p^{(i)}$ in Ω^P performs pooling independently and generates a single embedding to be fused with all nodes in some fusing set $\omega_f^{(j)}$ in Ω^F. Note that m and n refer to the size of Ω^P and Ω^F. Two most frequently used pooling methods are max and mean pooling. For an element set $\omega_p^{(i)}$ in Ω^P with N_1 nodes, let $h_k^{(i)}$ be the embedding vector in $H_{r\times d}$ with respect to the k-th node in $\omega_p^{(i)}$. Thus we have

$$\begin{aligned} v_p^{(i)} &= \sup\{h_k^{(i)}, k \in [1, N_1] \cap Z\} \\ &= [sup\{h_k^{(i)}[0]\}, sup\{h_k^{(i)}[1]\}, ..., sup\{h_k^{(i)}[d-1]\}] \end{aligned} \tag{7}$$

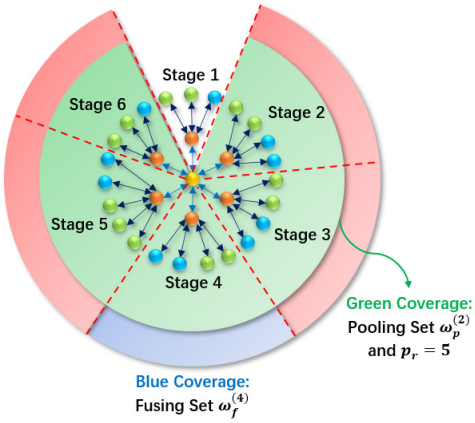


Fig. 8. Stage-wise Integration Settings

for max pooling and

$$\begin{aligned} v_p^{(i)} &= \frac{1}{N_1}\sum_{k=1}^{N_1} h_k^{(i)} \\ &= \frac{1}{N_1}[\sum_{k=1}^{N_1} h_k^{(i)}[0], \sum_{k=1}^{N_1} h_k^{(i)}[1], ..., \sum_{k=1}^{N_1} h_k^{(i)}[d-1]] \end{aligned} \tag{8}$$

for mean pooling where d is the number of columns in H, denoting the dimension of a node embedding. To fuse the pooled embeddings to nodes in Ω^F, weighted averaging and concatenation are most commonly adopted. Suppose the $\Psi : \Omega^P \rightarrow \Omega^F$ mapping of $\omega_p^{(i)}$ yields $\omega_f^{(j)}$, i.e.,

$$\Psi(\omega_p^{(i)}) = \omega_f^{(j)} \tag{9}$$

given $v_p^{(i)}$ the pooled embedding obtained from $\omega_p^{(i)}$, and that $\omega_f^{(j)}$ has N_2 nodes in it, let $h_k^{(j)}$ be the embedding vector in $H_{r\times d}$ with respect to the k-th node in $\omega_f^{(j)}$. Thus, we have for concatenation,

$$\widehat{h}_k^{(j)} = h_k^{(j)} || v_p^{(i)} \tag{10}$$

where $\widehat{h}_k^{(j)}$ is the counterpart of $h_k^{(j)}$ in $\widehat{H}_{r\times\widehat{d}}$, and the symbol || stands for direct appending of $v_p^{(i)}$ to the end of $\omega_f^{(j)}$. For weighted averaging, the following holds,

$$\widehat{h}_k^{(j)} = \alpha h_k^{(j)} + \beta v_p^{(i)} \tag{11}$$

Note that the coefficients α and β can be trained as regular parameters or tuned as hyperparameters. Provided the principles discussed above, it is clear that our goal is to specify the Ω^P, Ω^F and $\Psi : \Omega^P \rightarrow \Omega^F$, as well as the pooling and fusing mechanisms. To streamline the process, max pooling and concatenation are employed, and 2 general schemes are designed to determine Ω^P, Ω^F and $\Psi : \Omega^P \rightarrow \Omega^F$, from the perspectives of stage and layer distributions. Definitions concerning a stage and a layer are provided as below.

Definition 2. A stage in a star-shaped topology is defined as a set of nodes incorporating one controller and all nodes directly attached to it, CRP included. Fig. 8 illustrates how nodes are assigned to their respective stages.

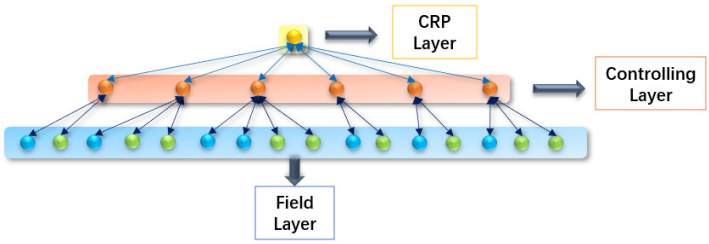


Fig. 9. Layer-wise Integration Settings

Definition 3. A layer in a star-shaped topology is referred to as the set of all nodes whose distances to the CRP are identical (See Fig. 9). As shown in Fig. 2, all controllers are one link away from the CRP, thus, they are in the same layer. Different layers are differentiated via their distance discrepancies.

4.3.1 Stage-wise Integration. To achieve stage-wise integration, all nodes are classified into stages (See Fig. 8), and each stage is a fusing set in Ω^F. In this case, $\omega_f^{(j)} = \{v_1, v_2, ..., v_{n_f^{(j)}}\}$ is the set of all devices in stage j ($n_f^{(j)}$ is the quantity of devices in stage j). To determine Ω^P, we define the term pooling range p_r as the number of consecutive stages that form each $\omega_p^{(i)}$, and therefore

$$\omega_p^{(i)} = \bigcup \forall \omega, \omega \in \{\omega_f^{(k)} | k \in \bigcup_{c=i-1}^{i+(p_r-2)} c \bmod n + 1\} \tag{12}$$

and

$$\Omega^P = \{\omega_p^{(i)} | i \in [1, n] \cap Z\} \tag{13}$$

Note that in this manner, $m = n$, and thus, $\Psi : \Omega^P \rightarrow \Omega^F$ can be intuitively made a bijection, in which each $\omega_p^{(i)}$ in Ω^P is associated with one $\omega_f^{(j)}$ in Ω^F, and vice versa. Specifically, the following mapping is evaluated in our experiments,

$$\Psi(\omega_p^{(i)}) = \omega_f^{((i+\lfloor \frac{p_r}{2} \rfloor - 1) \bmod n+1)} \tag{14}$$

or its inverse

$$\Psi^{-1}(\omega_f^{(j)}) = \omega_p^{((j-\lfloor \frac{p_r}{2} \rfloor + n - 1) \bmod n+1)} \tag{15}$$

This mapping scheme is instantiated in Fig. 8, with $j = 4$ and $p_r = 5$, giving $i = 2$. Thus, $\omega_p^{(i)}$ is the green area starting from Stage 2 covering 5 consecutive stages, and $\omega_f^{(j)}$ is represented by the blue area, which is Stage 4. The general stage-wise integration process is described in Algorithm 1.

Algorithm 1 sequentially amalgamates the following steps:

1) **Defining fusing sets Ω^F (line 2-5)**: In stage-wise integration, each fusing set ω_f is equivalent to the set of all nodes associated with a particular stage in the specified industrial process. Therefore, its members range from the controller node c assigned to this stage as well as all the field devices in its supervision (denoted as *c's children*). All ω_f sets are stored in the fusing set array Ω^F.
2) **Defining pooling sets Ω^P (line 6-9)**: Given the pooling range p_r and the fusing sets Ω^F previously defined, the algorithm maps each ω_f to its respective pooling set ω_p via the inverse of mapping relation $\Psi : \Omega^P \rightarrow \Omega^F$, which is defined in equation (15). Similarly, all ω_p sets are stored in the pooling set array Ω^P.

Algorithm 1. Stage-wise Integration

Input: Learnt embeddings produced by the encoder H, pooling range p_r

Output: Integrated embeddings $\widehat{H}$

Ensure:

1: Fusing sets $\Omega^F \leftarrow$empty set; $\Omega^P \leftarrow$empty set; $\widehat{H} \leftarrow$empty list

2: **for** c in set of controllers: #Defining Ω^F

3: $\quad \omega_f = \bigcup c, c's\ children$

4: $\quad \Omega^F \leftarrow \Omega^F + \omega_f$

5: **end for**

6: **for** ω_f in Ω^F: #Defining Ω^P

7: $\quad \omega_p \leftarrow \Psi^{-1}(\omega_f)$

8: $\quad \Omega^P \leftarrow \Omega^P + \omega_p$

9: **end for**

10:**for** ω_f in Ω^F: #Pooling and Fusing

11: $\quad \omega_p \leftarrow \Psi^{-1}(\omega_f)$

12: $\quad$ Extract the set of embeddings H_p from H with respect to nodes in ω_p.

13: $\quad v_p \leftarrow$maxpooling(H_p)

14: $\quad$ **for** v in ω_f: #Traversing all nodes in the current ω_f

15: $\quad\quad \widehat{H} \leftarrow \widehat{H}$+concat($v$,$v_p$)

16: $\quad$ **end for**

17:**end for**

18:**return** $\widehat{H}$

3) **Pooling and fusing (line 10-17)**: For each fusing set ω_f, the model initially finds its corresponding pooling set ω_p, and locates the embeddings H_p produced by the encoder with respect to the nodes in ω_p. Then the algorithm performs max pooling over H_p and produces an embedding v_p as a reflection of the graph's representative properties. Finally, embedding v_p is fused to all vector representations for nodes in the fusing set ω_f via concatenation. The algorithm outputs the concatenated vectors for subsequent decoding.

4.3.2 Layer-wise Integration. To conduct layer-wise integration, on the other hand, every node is assigned to a layer $\theta^{(i)} \in \Theta$ with Θ the set of all n_l layers in the topology (e.g. The architecture in Fig. 2 can be separated into 3 layers, the CRP, controlling and field layer, as shown in Fig. 9). Each ω_p is produced via sampling, and assuming that nodes in different layers exhibit distinctive influential patterns on the model's decoding functionality, sampling across multiple layers is strictly constrained. Equivalent statement holds as thus: $\forall \omega_p \in \Omega^P, \forall i \neq j, i, j \in [1, n_l] \cap Z, \omega_p \cap \theta^{(i)} = \emptyset\ or\ \omega_p \cap \theta^{(j)} = \emptyset$. Also, for simplicity, the configuration of the fusing sets is streamlined as $\Omega^F = \{V\}$, allowing the effect of layer-wise fusing to span across the entire topology. Moreover, to investigate the influence of different layers on the model's performance, only one ω_p is created at a time and $\Omega^P = \{\omega_p\}$ contains only one element as a result. Algorithm 2 elaborates on the workflow of layer-wise integration procedure.

Algorithm 2. Layer-wise Integration

Input: Learnt embeddings produced by the encoder H

Output: Integrated embeddings $\widehat{H}$

Ensure:

1: Layer list $\Theta \leftarrow$empty list; $\widehat{H} \leftarrow$empty list

2: Breadth-First Search on the specified layered topology (e.g. the structure in Fig. 2), and store each node into its layer $\theta^{(i)}, i \in [1, n_l] \cap Z$.

3: **for** i in $[1, n_l] \cap Z$:

4: $\quad \Theta \leftarrow \Theta + \theta^{(i)}$

5: **end for**

6: Given layer $\theta \in \Theta$ of interest, set $n \in [1, length(\theta)] \cap Z$.

7: $\omega_p \leftarrow sampling(\theta, n)$

8: Extract the set of embeddings H_p from H with respect to nodes in ω_p.

9: $v_p \leftarrow$maxpooling(H_p)

10:**for** v in ω_f:

11: $\quad \widehat{H} \leftarrow \widehat{H}+$concat($v$,$v_p$)

12:**end for**

13:**return** $\widehat{H}$

The main steps of Algorithm 2 can be described as below:

1) **Defining layers (line 2-5)**: Given a hierarchical structure (e.g. Fig. 2), the algorithm dissects it into n_l distinct layers via Breadth-First Search, each of which contains devices of a unique category (e.g. controllers, field devices, etc.). All layers are stored in the layer array Θ.
2) **Sampling (line 6-7)**: After all layers are defined, the GSIN selects the layer θ from Θ to conduct sampling on, configures the number of devices n to perform pooling from, and finally, defines the pooling set ω_p as a subset of n nodes randomly selected from layer θ.
3) **Pooling and fusing (line 8-12)**: In layer-wise integration, the fusing set ω_f is configured as the entire set of nodes in the specified graph. That said, the pooling and fusing processes are stated as follows: For the pooling set ω_p generated previously, the algorithm extracts the vector representations H_p with respect to ω_p from the learnt embeddings H, and then computes embedding v_p by performing max pooling over H_p. Vector v_p is ultimately concatenated to all the learnt embeddings produced by the Encoder.

After integration, regardless of the integration scheme used, $H_{r\times d}$ is transformed into $\widehat{H}_{r\times \widehat{d}}$ which serves as the input to the Decoder.

4.4 Decoder

The decoder takes in the integrated embeddings as input and derives the nodes' state values (0 for normal and 1 otherwise) at the specified time tick (See Fig. 10). It consists of a fully-connected (FC) layer followed by a Softmax block (See (16) and (17)). The FC layer projects the $\widehat{d}$-dimensional vectors onto a 2-dimensional plane, and the

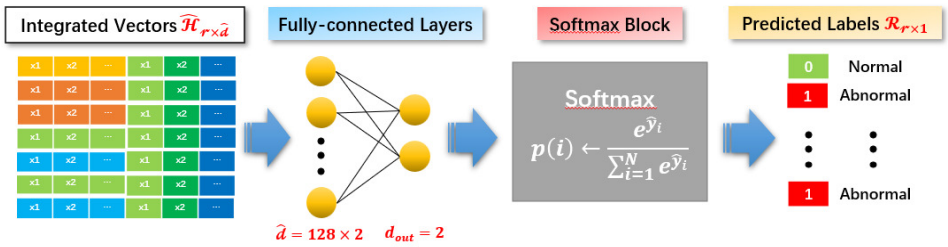


Fig. 10. Decoder Configuration (**Input**: $\widehat{H}_{r\times\widehat{d}}$; **Output**: $R_{r\times1}$)

Softmax operator repositions the projection point on the unit circle via exponential normalization. The ultimate labels are hence determined from the positions of the largest values in the resulting 2-dimensional vectors.

$$H_{r\times2}^{(fc)} \leftarrow fc(\widehat{H}_{r\times\widehat{d}}) \tag{16}$$

or its inverse

$$R_{r\times1} \leftarrow softmax(H_{r\times2}^{(fc)}) \tag{17}$$

5 EVALUATION

The GSIN's functionality is measured on a server running CentOS Linux system (version 7) dedicated to deep learning research. All necessary coding are implemented using PyTorch version 1.7.1+cu110.

5.1 Datasets & Graphs

5.1.1 Datasets. The following datasets are utilized for performance evaluation.

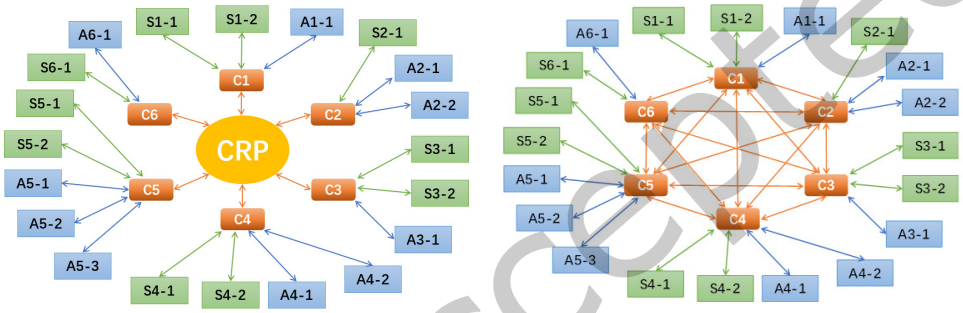
1) **SWaT**[14]: Collected on a six-stage Secure Water Treatment testbed, the SWaT dataset entails the state readings of 51 field devices associated with a water treatment process. It is a popular dataset globally adopted in industrial process related research.
2) **WADI**: This is a dataset obtained over a three-stage Water Distribution testbed. It incorporates the numeric reading sequences with respect to 124 devices across all stages. Similar to the SWaT dataset, the WADI set is also frequently utilized for model measurement designed for analyzing and securing cyber-physical systems. Technical details of the dataset as well as the testbed are available at https://itrust.sutd.edu.sg/itrust-labs-home/itrust-labs_wadi/.
3) **CISS**: The Critical Infrastructure Security Showdown dataset is produced via a series of technology assessment exercises held by iTrust. It keeps record of the numeric states and operational modes from 28 devices over part of the SWaT testbed during the exercises, and labelled only where attacks occur. For more details, please visit https://itrust.sutd.edu.sg/ciss/.
4) **BATADAL**[31]: The BATADAL dataset is a creation of a battle of attack detection algorithms designed for discovering cyber attacks in water distribution systems. 36 readings from 31 devices are maintained in the dataset.
5) **CHD**[29]: Developed by the HCRL, the Car Hacking Dataset is a combination of data snippets recorded in the normal setting along with 4 attack scenarios (DoS, fuzzy, drive gear spoofing, and RPM gauze spoofing). Each scenario is triggered independently and recorded in its own specified dataset. In this paper, the four datasets with attacks are concatenated as one for a more general analysis.

The core characteristics of the datasets are summarized in Table 1.

Table 1. Dataset Characteristics

Dataset	#Controllers	#Devices	#Dimensions Adpoted	Sampling Interval	#Samples	Anomaly Ratio
SWaT	6	51	35	1s	449,920	12.1402%
CISS	6	28	50	1s	115,883	0.0621%
WADI	5	123	74	1s	172,800	5.7737%
BATADAL	9	31	36	1hr	4,177	5.2430%
CHD	N/A	1	9	N/A	16,569,475	14.0712%

5.1.2 Graphs. The GSIN is trained and evaluated on 2 types of topologies as shown in Fig. 11.



(a) 3 Layer Structure (Star-shaped Topology) (b) 2 Layer Structure

Fig. 11. Basic Topologies

The 3-layer structure, also denoted as the star-shaped topology [24] [27], incorporates the CRP node as its top layer. While in the 2-layer structure where the CRP is not applicable, the top layer comprises the controller nodes inter-connected with one another. Note that Fig. 11 merely exhibits the layout and connection patterns of the graph, while the specific quantity of sensors and actuators attached to each controller is configured in accordance to the datasets. The models developed with 2-layered and 3-layered topological structures feature distinct merits and also suffer from different drawbacks.
Models created with a 2-layered graph:

1) **Advantages**: Comparative to a 3-layered architecture, distances among nodes in distinct stages are relatively shortened in the 2-layered topology. Thus, it takes fewer message passing moves for some edge node to learn information from another node located in a distant stage. In the meantime, with the message passing paths shortened, information loss that occurs during message aggregation has been effectively mitigated, for which the quality of the resulting node embeddings produced by the Integrator is improved. Therefore, these embeddings serve as a more solid basis for specific subsequent tasks such as node-level anomaly detection.
2) **Disadvantages**: Although a complete linkage at the controlling layer can be beneficial to the GSIN as explained previously, the other side of the coin, however, lies in the undesirably incremented complexity in the structure of the graph. It yields a more convoluted adjacency matrix and consequently consumes extra

computation resources during the learning process. To tackle this, however, one may consider moderately truncating part of the layers as a countermeasure, in which case the computation cost can be balanced.

Models created with a 3-layered graph:

1) **Advantages**: Instead of a full connection among the controllers, the 3-layered graph introduces a media node (CRP) to link all controllers to each other. With fewer links involved than the 2-layered architecture as shown in Fig. 11, the adjacency relations are relatively streamlined and the GSIN's runtime efficiency is potentially better.

2) **Disadvantages**: As the number of message passing steps it takes for a node to become aware of all other nodes is increased by one due to the addition of the CRP node, one more neural layer is required for the respective GSIN to achieve performance close to that of the model trained with a 2-layered structure. In this scenario, the quantity of the trainable parameters may rise and the GSIN's training efficiency is hence impaired. Therefore, hyperparameter tuning tends to be trickier in order to balance the GSIN's functionality and its operational efficiency.

5.2 Metrics

With TP, FP, TN and FN abbreviations with respect to the number of True Positive, False Positive, True Negative and False Negative samples, our evaluation metrics are listed as follows:

1) **Accuracy**: Measures a model's ability to correctly classify samples to the categories they pertain to. Computed as $Accuracy = \frac{TP+TN}{TP+FP+TN+FN}$.

2) **Precision**: Evaluates a model's detection accuracy over the positive samples. This metric is usually of significant value in ICS anomaly detection, given the convention in industrial processes that a system's availability takes priority over any other security paradigm. Computed as $Precision = \frac{TP}{TP+FP}$.

3) **Recall**: Assesses a model's ability to respond to all positive scenarios (i.e. anomalies of all sorts). Computed as $Recall = \frac{TP}{TP+FN}$.

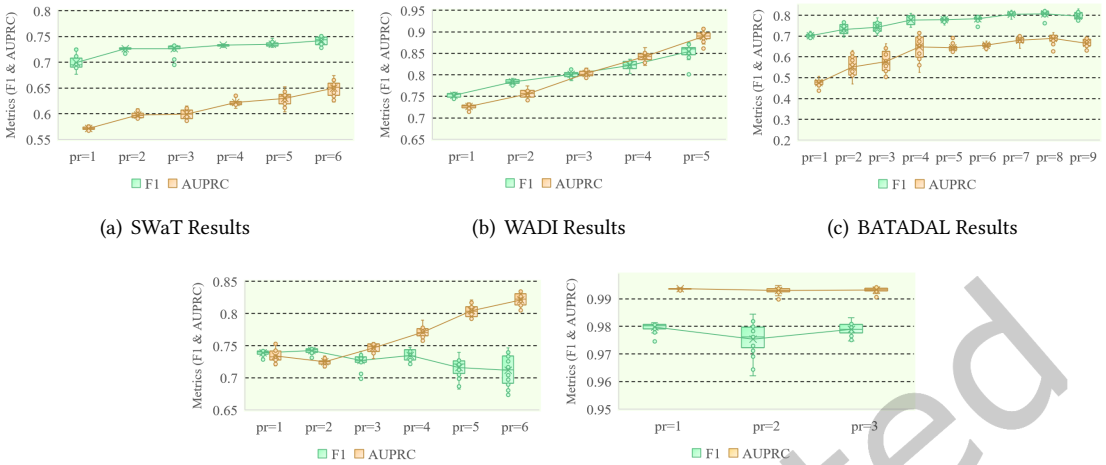
4) **F1 Score**: Compound metric incorporating Precision and Recall. Measures a model's overall detection effectiveness. Computed as $F1 = \frac{2 \times Precision \times Recall}{Precision+Recall}$.

5) **AUROC**: The area under the Receiver Operating Characteristics Curve. Evaluates a model's ability to differentiate positive cases from negative ones.

6) **AUPRC**: The area under the Precision-Recall Curve. Useful metric for the cases in which finding positive examples in unbalanced data is particularly interesting.

5.3 Stage-wise Integration

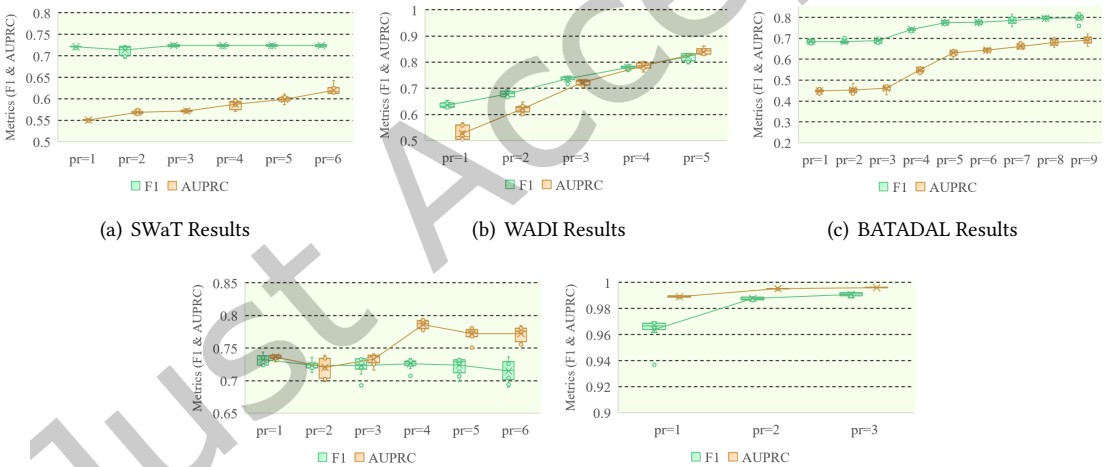
To explore the effect of stage-wise integration on the GSIN, different pooling ranges (denoted as pr) are applied and the respective anomaly detection performances (box plots of the F1 scores and AUPRC values) are recorded in Fig. 12 and Fig. 13. Each box plot is obtained via running the GSIN with a particular parameter settup 20 times in a row, and that other parameter configuration details are as follows: input layer with size that matches the input vectors; hidden layer (one with 128 neurons); output layer with 2 neurons + logSoftmax; $ReLU$ activation. Note that this configuration also applies to the GSINs in Section 5.4 5.5 and 5.6.



(a) SWaT Results (b) WADI Results (c) BATADAL Results

(d) CISS Results (e) CHD Results

Fig. 12. Stage-wise GSIN Performance with 2-Layer Settings



(a) SWaT Results (b) WADI Results (c) BATADAL Results

(d) CISS Results (e) CHD Results

Fig. 13. Stage-wise GSIN Performance with 3-Layer Settings

Our observations stand thus:

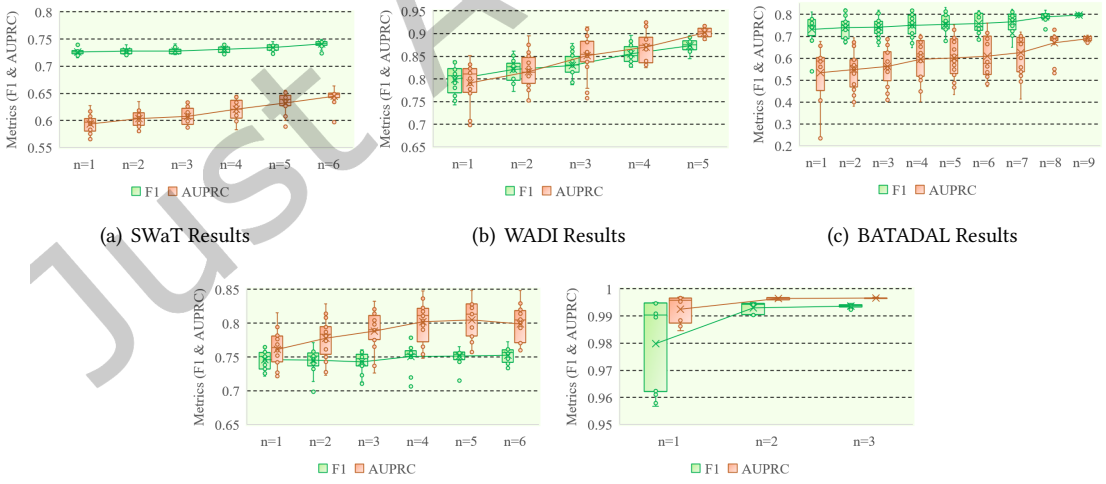
- The GSIN's AUPRC performance rises almost monotonically as the pooling range increases, peaking when the *pr* value reaches its maximum with respect to the specific dataset. No obvious convergence is observed except for the BATADAL, indicating that trading pooling coverage for other benefits (such as operational efficiency) without degrading GSIN's functionality is generally not so viable through stage-wise integration.

This may attribute to the contextual diversity encoded in the embeddings among nodes across different stages. Specifically, although a node's global awareness is enhanced through message passing, its local features still remain dominant, indicating that nodes in the same stage tend to be semantically similar, while the other way around otherwise. Therefore, an embedding pooled from particular stages may not capture all principal characteristics in the entire topology, in which case the potential of improving the GSIN's detection preformance is undesirably reduced.

- The GSIN's functionality varies along with the graph architecture. Our results indicate that given identical pooling configurations, the model's test results with 2 structural layers generally surpass their counterpart with 3. This observation arises from the benefits of the 2-layer structure in message passing, stated as follows: As shown in Fig. 5.1.2, the stage-exclusive properties tend to be more sufficiently mingled in the 2-layer structure during the message passing process as the controller nodes are able to directly exchange information with each other without using a third-party media (as the CRP in the 3-layer structure), which potentially mitigates information loss in the aggregation process. In this scenario, the profiles of the nodes in the 2-layer topology are enriched with more core features across the other stages than in the 3-layer graph, leading to a boost in the GSIN's ability to accurately detect anomolous patterns using these integrated node-level profiles.

5.4 Layer-wise Integration

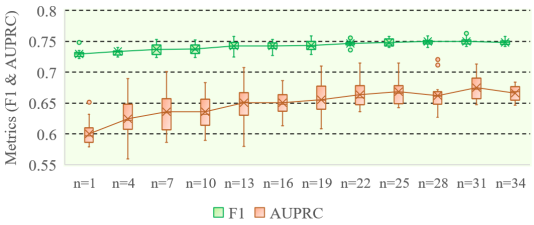
In this section, we examine the effect of layer integration on the GSIN. Similar to Section 5.3, the integration algorithm is also run 20 times for each sampling rate (denoted as n), and the box plots of the GSIN's F1 and AUPRC values obtained over the controlling and field layer with respect to the 2-layer and 3-layer structures are illustrated in Fig. 14 15 16 and 17.



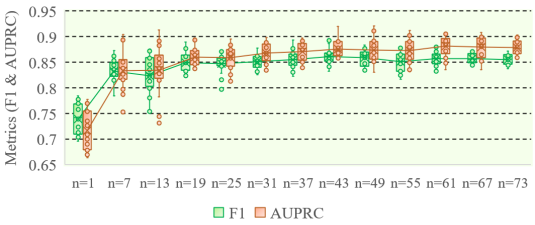
(a) SWaT Results (b) WADI Results (c) BATADAL Results

(d) CISS Results (e) CHD Results

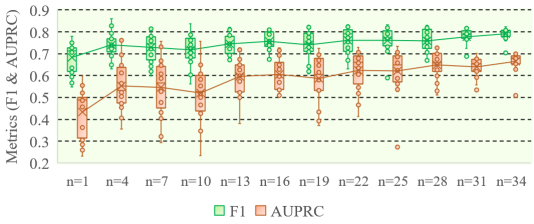
Fig. 14. Controlling Layer Sampling with 2-Layer Settings



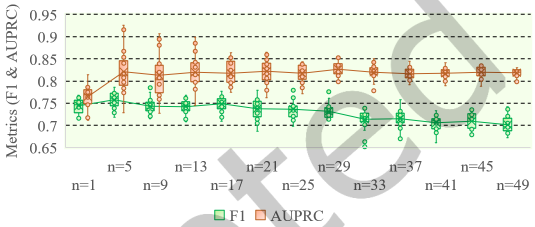
(a) SWaT Results



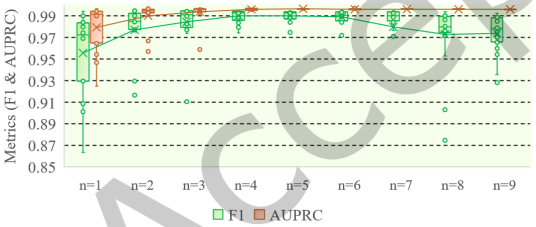
(b) WADI Results



(c) BATADAL Results

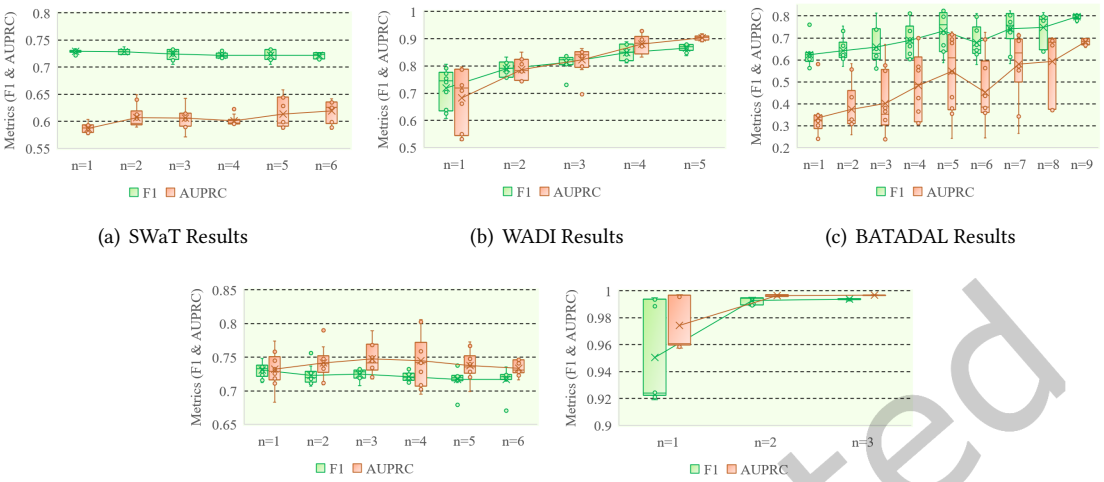


(d) CISS Results



(e) CHD Results

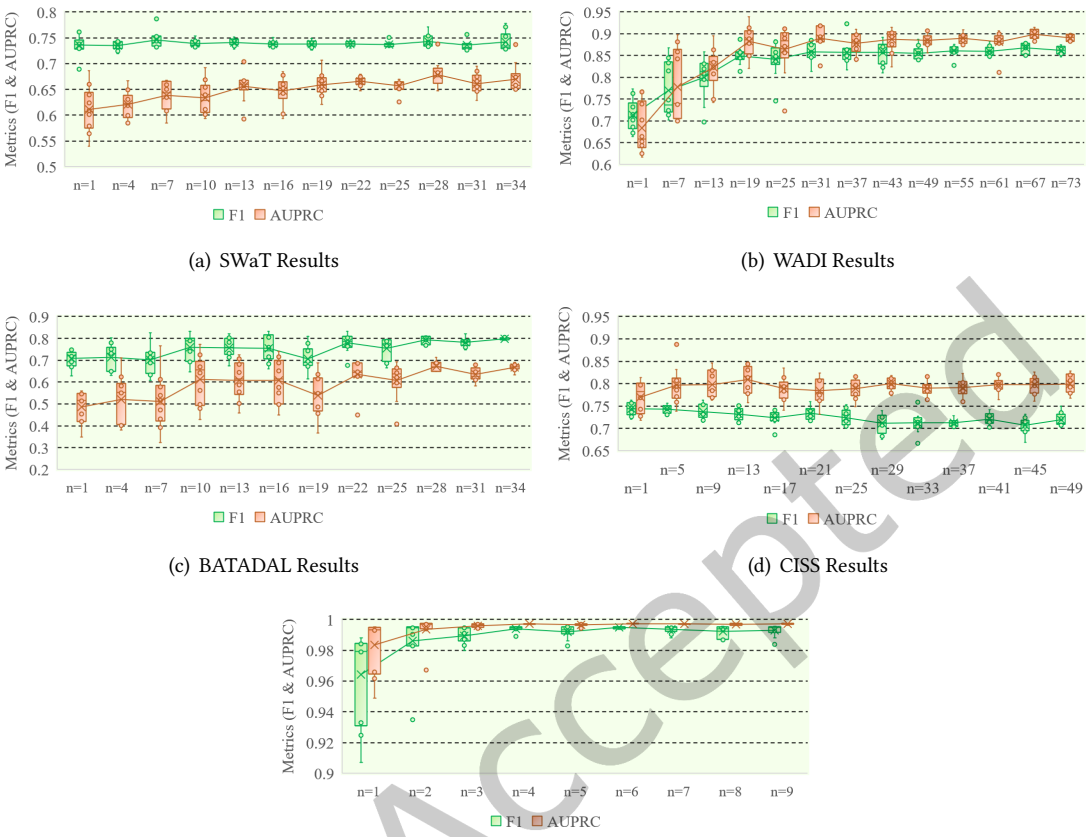
Fig. 15. Field Layer Sampling with 2-Layer Settings



(a) SWaT Results (b) WADI Results (c) BATADAL Results

(d) CISS Results (e) CHD Results

Fig. 16. Controlling Layer Sampling with 3-Layer Settings



(a) SWaT Results

(b) WADI Results

(c) BATADAL Results

(d) CISS Results

(e) CHD Results

Fig. 17. Field Layer Sampling with 3-Layer Settings

With n the number of sampled nodes for integration, it is perceivable that the GSIN's AUPRC tends to saturate above a particular n value. For example, the value of AUPRC generally ceases to increase to a higher level as n reaches 31 or higher in the WADI dataset, and the same applies to the CHD dataset when n exceeds 4, etc. This leads to our inference that although participation of all nodes may intuitively benefit integration with the greatest amount of representative features possible, it is probably unnecessary to do so when similar results can be achieved with fewer nodes taken into consideration, due to semantic overlapping among nodes in close proximity. In fact, parts of the figures actually show decrement of the GSIN's detection functionality as the sampling rate approaches its supremum (e.g. the AUPRC for the CISS dataset in the 3-layer setting, and the F1 for the CHD dataset in the 2-layer setting, etc.). This sheds light on our conjecture that information overlapping may trigger semantic redundancy that negatively impacts the message distribution in a node's embedding, and consequently deteriorates the GSIN's anomaly detection accuracy.

Based upon the results shown above, one is able to see the optimal integration option for the GSIN. The model's functionality generally saturates at around $n = 25$, $n = 37$, $n = 25$, $n = 13$, and $n = 4$ with respect to the SWaT, WADI, BATADAL, CISS, and CHD datasets with field layer sampling adopted, implying that the embedding

Table 2. Mapping of Device Labels

Dataset	Label Mapping
SWaT	Controllers: 1-6; Stage 1: 7-10; Stage 2: 11-15; Stage 3: 16-23; Stage 4: 24-28; Stage 5: 29-39; Stage 6: 40-41
WADI	Controllers: 1-5; Stage 1: 6-20; Stage 2: 21-30; Stage 3: 31-42; Stage 4: 43-72; Stage 5: 73-79
BATADAL	Controllers: 1-9; Stage 1: 10-15; Stage 2: 16; Stage 3: 17-30; Stage 4: 31; Stage 5: 32-41; Stage 6: 42; Stage 7: 43; Stage 8: 44; Stage 9: 45
CISS	Controllers: 1-6; Stage 1: 7-11; Stage 2: 12-19; Stage 3: 20-29; Stage 4: 30-36; Stage 5: 37-53; Stage 6: 54-56
CAN	Grouping Nodes: 1-3; Group 1: 4-6; Group 2: 7-9; Group 3: 10-12

created from a uniformly pooled subset of field layer nodes is semantically rich enough to maximize the GSIN's anomaly detection performance.

5.5 Individual Device Influence

To further investigate how each device in the network may impact the GSIN's functionality via integration, models conducting individual sampling are created in which only one device is selected for pooling. To quantitize such individual influences, a new metric – the Individual Influence Score (IIS) is introduced as follows:

$$IIS(n) = \frac{\Psi^{(n)} - \mu}{\sigma} \tag{18}$$

where n is the device label, $\Psi^{(n)}$ the metric score (F1 or AUPRC) of the GSIN that conducts pooling over the single device labelled n, and with N the number of devices in the graph, Ψ's mean and standard deviation are computed as (19) and (20):

$$\mu = \overline{\Psi} = \frac{1}{N}\sum_{n=1}^{N}\Psi^{(n)} \tag{19}$$

and

$$\sigma = \overline{(\Psi^{(n)} - \mu)^2} = \frac{1}{N}\sum_{n=1}^{N}(\Psi^{(n)} - \mu)^2 \tag{20}$$

Fig. 18 19 20 21 22 present the histograms of the IISs with respect to all devices for all datasets. The x-axis values denote the device labels in the graph, and the y-axis values represent the respective IISs. To derive better implications over the results, the mapping relations of all the devices to their labels are provided in Table 2. Taking the SWaT dataset as an exemplification, there are 6 controllers labelled from 0 to 5, as well as 6 stages, each of which associated with a couple of field devices. Specifically, devices number 6-9 belong to Stage 1; devices number 10-14 are members of Stage 2; devices labelled 15-22 are correlated with Stage 3; devices 23-27 are assigned to Stage 4; devices 28-38 are incorporated in Stage 5 and devices 39-40 are related to Stage 6.

Here are our observations:

- One can conclude from the figures that there is a noticeable resemblance in the IIS scores obtained from pooling the field devices in the same stage. For instance, the IISs computed for the devices from Stage 5 in

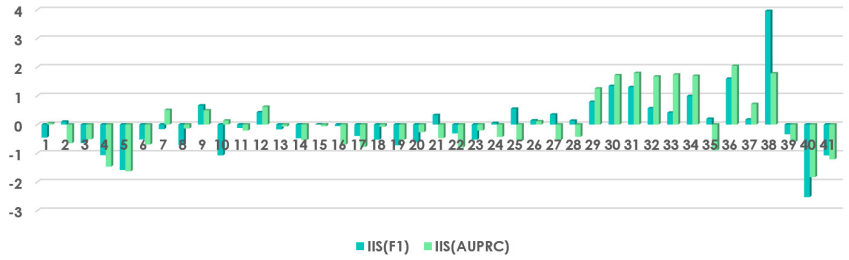


Fig. 18. Individual Influence Scores (IIS) for SWaT

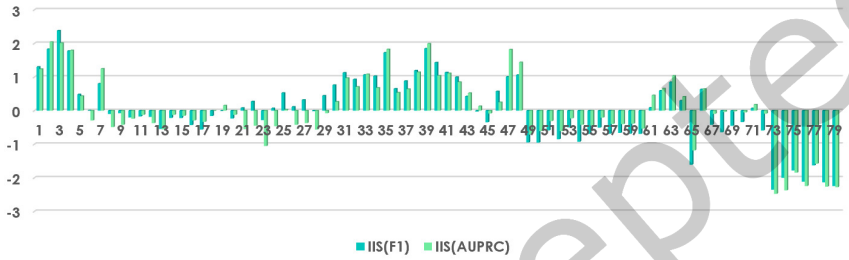


Fig. 19. Individual Influence Scores (IIS for WADI)

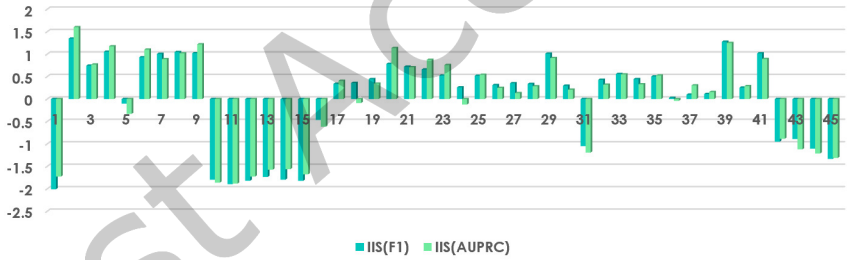


Fig. 20. Individual Influence Scores (IIS) for BATADAL

the WADI dataset are numerically similar to each other. The same can be said to Stage 1 in the BATADAL dataset, Stage 2 in the CISS dataset, etc.. The reason for this numeric uniformity lies in the topological role consistency among all nodes in the same stage in terms of how messages are aggregated, and this consistency leads to similarity in the semantic composition of their vector representations. Therefore, the embeddings of nodes within a particular stage tend to be numerically similar and hence it is expected their contributions to the GSIN's performance are not far from equivalent.

- The influences of nodes to the GSIN from disparate stages in the same industrial process may differ tremendously from each other. For example, the IIS discrepancy for the devices in Stage 1 and 3 in the BATADAL dataset is rather easy to discern, and the differences between the results obtained from Stage 3 and 5 in the WADI dataset are also fairly apparent. We speculate that this might result from the devices'

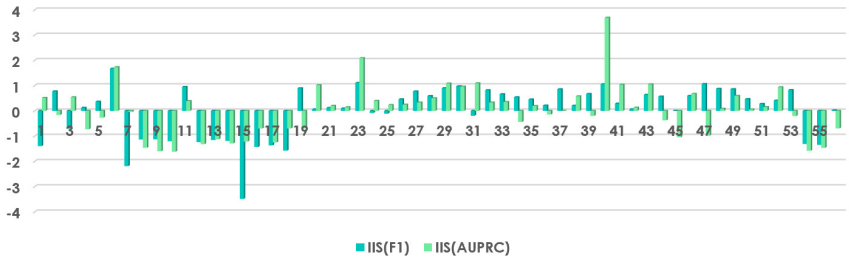


Fig. 21. Individual Influence Scores (IIS) for CISS

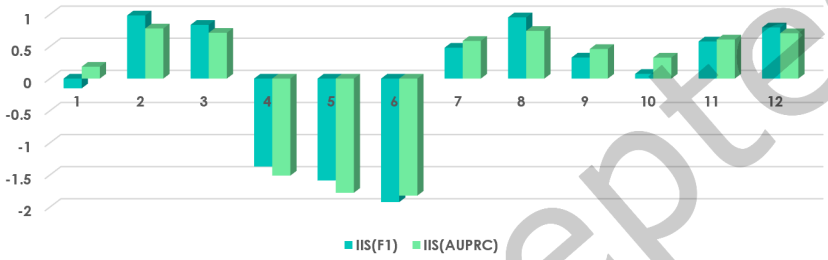


Fig. 22. Individual Influence Scores (IIS) for CHD

distinctiveness in their local topological context. As the set of immediate neighbours from which messages are aggregated differ drastically across stages, embeddings of nodes from different stages are typically unique numerically as they exhibit a node's distinct local awareness. Therefore, it is not surprising to spot a dramatic deviation between the nodes' influences to the GSIN's functionality from disparate stages. On the other hand, it is conjectured that the numeric properties of the reading streams with respect to different field devices also shapes the overall contribution of a particular node to the GSIN's performance, though further research is required to deeply understand the inherent principles.

5.6 Runtime Consumption

In this section, we evaluate the GSIN's runtime efficiency against its closest couterpart, the GLIN, which features global pooling instead of sampling. Both models are of the same GNN architecture as described in Section 5.3. Layer-wise integration is applied to the GSIN with sampling rates of 25, 37, 25, 21, and 4 with respect to SWaT, WADI, BATADAL, CISS and CHD. The training and test time consumption of the two methods for all datasets are displayed in Table 3.

The results in Table 3 show that the GSIN is no less than 5.8384% more efficient than the GLIN in training, and it also renders a desirable improvement in the model's runtime consumption during test phase in all datasets apart from the CHD. This indicates that sampling can have a meliorating effect on the model's runtime cost. As the set of nodes to be processed during integration is streamlined, it is expected the pooling process becomes less time consuming to an extent depending on the actual sampling rate configured in the GSIN. It is suggested that the minimum sampling rate should be selected at which the GSIN's functionality and operational efficiency are appropriately balanced as required in specific industrial scenarios.

Table 3. Runtime Consumption

Dataset	Models	Training (s)	Test (s)	Efficiency Gain (%)
SWaT	GLIN	2.3129	17.3887	Train: +57.0943
	GSIN	1.4723	11.8294	Test: +46.9956
WADI	GLIN	1.4752	11.3276	Train: +172.6802
	GSIN	0.5410	4.9696	Test: +127.9379
BATADAL	GLIN	0.4159	0.6395	Train: +87.9349
	GSIN	0.2213	0.3802	Test: +68.2009
CISS	GLIN	0.7258	5.8436	Train: +108.4434
	GSIN	0.3482	3.6740	Test: +59.0528
CHD	GLIN	2.4636	16.6507	Train: +5.8384
	GSIN	2.3277	18.7411	Test: -11.1541

Table 4. SWaT Dataset Anomaly Detection Baseline Comparison (%)

Methods	Accuracy	Recall	Precision	F1	AUROC	AUPRC
GSIN (Ours)	80.25	64.37	**89.27**	**74.80**	**77.87**	**67.63**
TAGCN	76.87	57.20	87.41	69.15	69.97	52.81
GAT	75.69	56.52	75.32	64.58	67.42	51.51
E-GSAGE	78.47	60.53	87.22	71.47	71.74	56.58
GLIN	**80.34**	**64.76**	88.19	74.68	75.80	66.06
FT-GCN	76.02	55.11	87.22	67.54	65.35	60.78
kNN	79.44	63.03	83.64	71.89	77.56	64.05
AE	67.80	51.02	51.76	51.39	58.32	30.65
SO-GAAL	72.96	50.05	54.53	52.19	58.57	32.08
MO-GAAL	65.34	54.28	54.63	54.46	53.32	30.34
Replicator-NN	62.20	53.15	53.02	53.08	57.08	31.37
OCSVM	64.67	52.53	52.89	52.71	45.93	27.89
Isolation Forest	63.54	53.72	53.72	53.72	58.39	32.46
LOF	65.85	56.92	56.90	56.91	58.75	33.12

5.7 Baseline Comparison

Finally, we compare the GSIN's performance against multiple existing baselines ranging from similar GNN methods to other machine and deep learning approaches, as listed below.

- **GSIN**: Graph Sample-and-Integrate Network. Proposed method in this article.

Table 5. WADI Dataset Anomaly Detection Baseline Comparison (%)

Methods	Accuracy	Recall	Precision	F1	AUROC	AUPRC
GSIN (Ours)	**88.28**	**88.14**	**85.33**	**86.71**	**94.91**	**89.12**
TAGCN	71.90	52.67	78.25	62.96	65.25	45.77
GAT	70.76	50.47	81.56	62.35	68.61	50.37
E-GSAGE	74.30	61.52	69.87	65.43	74.47	57.66
GLIN	86.24	87.18	83.15	85.12	94.50	88.39
FT-GCN	72.19	52.98	75.22	62.17	63.52	46.12
kNN	72.45	56.29	66.76	61.08	67.54	47.48
AE	67.18	52.24	54.22	53.21	56.42	34.48
SO-GAAL	70.52	50.13	63.56	56.05	55.00	34.52
MO-GAAL	70.55	50.18	65.50	56.83	49.83	34.67
Replicator-NN	60.30	51.92	51.95	51.93	55.47	32.67
OCSVM	62.70	53.48	53.70	53.59	51.50	32.22
Isolation Forest	61.28	53.48	53.46	53.47	56.17	33.10
LOF	60.76	53.12	53.15	53.13	55.48	34.28

Implementation details: 1 input layer (size matches preprocessed vectors); 1 hidden layer (128 neurons); output layer (2 classes); *ReLU* activation; max pooling + field layer sampling (n equals 25, 37, 25, 21, and 4 with respect to SWaT, WADI, BATADAL, CISS and CHD).

- **TAGCN**[12]: Systematic approach to perform convolutions on graphs using learnable filters.
  Implementation details: 1 input layer (size matches preprocessed vectors); 3 hidden layer (128 neurons); output layer (2 classes); number of kernels $k = 3$; trainable coefficients.
- **GAT**[33]: Attention-based message aggregation scheme. Features attention computation using vertice profiles.
  Implementation details: 1 input layer (size matches preprocessed vectors); 1 hidden layer (128 neurons); output layer (2 classes); *ReLU* activation.
- **E-GSAGE**[23]: Employs edge profile incorporation during message passing.
  Implementation details: 1 input layer (size matches preprocessed vectors); 3 hidden layer (128 neurons); output layer (2 classes); *ReLU* activation; edge profiles (end point mean pooling); full sampling.
- **GLIN**[24]: Features global-local semantic pooling and integration. Enhances global awareness for accurate decoding.
  Implementation details: 1 input layer (size matches preprocessed vectors); 1 hidden layer (128 neurons); output layer (2 classes); *ReLU* activation; max pooling + concatenate integration.
- **FT-GCN**[9]: A novel approach for label-limited IoT network intrusion detection. We attempt to transfer and evaluate the model in the specified industrial scenarios.

Table 6. BATADAL Dataset Anomaly Detection Baseline Comparison (%)

Methods	Accuracy	Recall	Precision	F1	AUROC	AUPRC
GSIN (Ours)	**91.89**	**76.92**	87.06	**81.67**	**84.60**	**70.12**
TAGCN	86.08	50.31	**93.03**	65.30	64.76	22.49
GAT	87.34	54.97	91.85	68.78	65.70	32.15
E-GSAGE	87.89	58.32	85.16	69.23	71.19	38.27
GLIN	91.63	75.53	87.11	80.91	83.48	67.57
FT-GCN	85.60	50.00	42.80	46.12	61.43	58.95
kNN	86.21	50.53	84.64	63.28	66.80	25.14
AE	62.27	56.57	53.38	54.93	51.79	15.98
SO-GAAL	86.02	50.12	88.84	64.09	50.12	52.95
MO-GAAL	86.01	50.13	76.80	60.66	50.13	40.94
Replicator-NN	61.41	56.96	53.53	55.19	62.35	18.41
OCSVM	78.88	52.56	53.81	53.18	43.42	19.91
Isolation Forest	77.52	54.27	54.18	54.22	64.16	20.80
LOF	75.60	54.44	54.44	54.44	60.61	22.97

Implementation details: 2 parallel TAGCN frameworks, each of which has 1 input layer (size matches preprocessed vectors); 3 hidden layer (128 neurons); output layer (2 classes); number of kernels $k = 3$; trainable coefficients; fully-connected mapping layer.

- **kNN**: Classic distance-based classification method.
  Implementation details: *n_neighbours* is set as the number of nodes in the topology associated with the particular dataset (42, 80, 57, 46, and 12 with respect to SWaT, WADI, CISS, BATADAL and CHD).
- **AE**: Classic outlier detection method featuring profile reconstruction.
  Implementation details: 1 input layer (size matches preprocessed vectors); 1 encoding layer (32 neurons for SWaT and WADI, 16 for CISS, 8 for BATADAL and CHD); 1 decoding layer (size matches input layer); *ReLU* activation.
- **SO-GAAL**[22]: Anomaly detection via outlier sampling. Characterizes informative sample generation and boundary learning.
  Implementation details: 1 generator (3 layers in total, number of neurons in each layer matches preprocessed vectors); 1 discriminator (composed of 1 input layer, 1 hidden layer of size 128, and 1 output layer of size 2); *ReLU* activation.
- **MO-GAAL**[22]: Same principle as the SO-GAAL, except using multiple generators to create outliers.
  Implementation details: k generators ($k = 2$; each generator has 3 layers in total, and the number of neurons in each layer matches preprocessed vectors); 1 discriminator (composed of 1 input layer, 1 hidden layer of size 128, and 1 output layer of size 2); mini-batch size equals $64/k$; *ReLU* activation.
- **Replicator-NN**[18]: AE-based outlier detector. Featuring a multi-layer perceptron that detects anomalies based on reconstruction errors.

Table 7. CISS Dataset Anomaly Detection Baseline Comparison (%)

Methods	Accuracy	Recall	Precision	F1	AUROC	AUPRC
GSIN (Ours)	**74.94**	**73.23**	73.69	**73.46**	**81.23**	**84.03**
TAGCN	71.19	64.19	83.36	72.53	67.88	69.16
GAT	69.38	64.52	70.10	67.19	66.41	69.44
E-GSAGE	71.76	65.20	78.52	71.25	65.92	68.46
GLIN	73,63	71.21	72.39	71.80	79.30	82.34
FT-GCN	71.97	65.22	**84.02**	73.44	66.86	68.53
kNN	73.27	68.44	76.17	72.10	75.46	79.66
AE	39.11	40.23	40.39	40.31	43.55	59.26
SO-GAAL	68.69	63.81	69.09	66.35	64.11	66.80
MO-GAAL	60.95	60.21	59.95	60.08	60.39	67.18
Replicator-NN	54.50	56.48	56.43	56.45	64.04	68.03
OCSVM	55.08	57.88	57.77	57.83	49.74	61.87
Isolation Forest	61.44	59.89	59.90	59.90	62.54	67.12
LOF	57.69	57.85	57.38	57.62	56.35	68.16

Implementation details: 1 input layer (size matches preprocessed vectors); 2 encoding layer (size of encoding layer 1: 32 neurons for SWaT and WADI, 16 for CISS, 8 for BATADAL and CHD; size of encoding layer 2: half of layer 1's size); 2 decoding layers (size of decoding layer 1: equivalent to encoding layer 1; size of decoding layer 2: equals the size of input); $a_2 = a_4 = 1$; $a_3 = 100$; $N = 4$.

- **OCSVM**: State-of-the-art novelty detection method. Flexible in learning sophisticated nonlinear boundaries. Implementation details: *nu* equals the ratio of anomalies for the respective dataset; $kernel$ = "rbf"; $gamma$ = "$auto$".
- **Isolation Forest**: Efficient outlier detector using feature splitting. Applicable to large volume data. Implementation details: $contamination$ is configured as the ratio of anomalies with respect to the specific dataset; $n\_estimators = 1000$.
- **LOF**: Detects outlier via local density deviation computation. Assumes that outliers exhibit a noticeably lower density than their neighbouring data points.
  Implementation details: $n\_neighbours$ equals the number of nodes in the graph (42, 80, 57, 46, and 12 with respect to SWaT, WADI, CISS, BATADAL and CHD); $contamination$ is set as the ratio of anomalies for the corresponding dataset; $novelty = True$; $n\_jobs = -1$.

The test set results with respect to all datasets are summarized in Table 4 5 6 7 8.

In general, despite the existence of a few exceptions, the GSIN surpasses the baseline methods in almost all the metrics, such as a marginal F1 gain of 0.12%, 1.59%, 0.76%, 0.02% and 0.22% with respect to the SWaT, WADI, BATADAL, CISS, and CHD datasets, as well as an improvement of 1.57%, 0.73%, 2.55%, 1.69% and 0.12% in AUPRC. Based on these results, our observations and analysis are stated as below.

Table 8. CHD Dataset Anomaly Detection Baseline Comparison (%)

Methods	Accuracy	Recall	Precision	F1	AUROC	AUPRC
GSIN (Ours)	**99.48**	**99.13**	**99.62**	**99.37**	**99.78**	**99.70**
TAGCN	76.92	60.51	85.85	70.99	89.26	79.61
GAT	90.06	84.06	91.73	87.73	94.97	90.64
E-GSAGE	99.30	98.84	99.46	99.15	99.68	99.58
GLIN	98.80	98.33	98.76	98.54	99.69	99.54
FT-GCN	75.98	58.72	87.31	70.22	75.64	68.89
kNN	87.06	79.50	88.62	83.82	92.85	88.19
AE	65.34	60.30	59.52	59.91	72.76	52.55
SO-GAAL	58.51	67.26	64.24	65.72	68.41	34.52
MO-GAAL	73.28	50.70	65.47	57.14	69.02	46.27
Replicator-NN	54.15	67.22	68.52	67.87	77.47	56.24
OCSVM	70.27	58.38	61.82	60.05	51.06	40.40
Isolation Forest	73.01	58.18	66.97	62.26	47.59	40.08
LOF	72.93	67.82	67.46	67.64	76.29	53.08

- As differentiated from most of the GNN baselines (i.e. TAGCN, GAT, E-GSAGE, FT-GCN), the GSIN enhances a node's awareness over the entire picture via encapsulating the graph's representative features into the node's vector representation. In this fashion, the GSIN is able to better exploit the relational properties among nodes that are topologically distant from each other, and utilize them effectively in deriving more reliable detection results.
- As compared to the GLIN that conducts pooling over all the nodes in the graph, the GSIN performs sampling before the integration process. On one hand, the sampling operation streamlines the set of nodes for subsequent feature pooling, potentially preserving a portion of computational resources consumed in global pooling, making the algorithm more efficient. On the other hand, as nodes within close proximity tend to overlap in terms of contextual semantics due to the neighbouring aggregation paradigm employed in message passing, incorporating all nodes during integration may introduce redundant information in the resulting global representation. Such redundancy may unbalance the feature composition of the node embeddings with repetitive info, exerting limitations on their ability to absorb extra distinct information. By applying sampling, this issue is effectively alleviated as the distances among the nodes are enlarged. In this case, semantic overlapping is reduced and the pooled features are a more appropriate reflection of the graph's core properties. Therefore, the quality of the integrated node embeddings is improved, and the GSIN's functionality is ameliorated.
- State-of-the-art outlier detection baselines (i.e. kNN, OCSVM, Isolation Forest, LOF) treats every input sample as an independent data point, and attempt to directly discriminate anomalous samples using the given features. Such principle, though straightforward to implement, are not very effective when dealing with fine-grained node-level anomaly detection issues in which data points are typically indifferentiable numerically. The proposed GSIN manages to mitigate this issue via global feature integration, which makes

each data point not only an exhibition of some node's exclusive properties, but also a reflection of its associativity with the rest of the data points. In this manner, the density of the data points are moderately decreased as external features are taken into account, and as a result, this leads to a rather significant boost in the model's anomaly detection performance.

- Autoencoder based methods (i.e. AE, Replicator-NN) are designed to encode and reconstruct the input samples and distinguish outliers based upon their reconstruction errors. These benchmarks may also suffer from the numeric resemblance issues when it comes to node-level anomaly detection, as reconstruction error alone may not provide sufficient gauges to differentiate data points that are almost identical to each other. Such problems can be addressed by the GSIN as stated above. Generative adversarial learning (i.e. SO-GAAL, MO-GAAL), on the other hand, generates informative noise samples to support anomaly classification. However, as the statistical characteristics of the distributions from which noise samples are obtained may dramatically deviates from the actual anomalous data points, boundary learning may still be challenging, especially on a fine-grained level where a fairly large portion of the outliers are numerically close to the noraml samples. However, one may consider utilizing the distribution of the specific input as the bases for noise generation, which may navigate the model to learn better boundaries.

6 CONCLUSION

In this paper, we investigate the feasibility of improving a GNN model's node-level inference functionality, via integration of the graph's universal properties. We explore the means to extract the set of nodes that produces the optimal expression encapsulating the graph's most representative features. Specifically, we design and develop the GSIN, a generic framework comprising a preprocessor, an encoder, an integration module, and a decoder. The GSIN takes in the original data flow captured from all devices of interest and outputs the state labels corresponding to each device at any applicable time tick. It outperforms current baselines with an AUPRC gain of 1.57%, 0.73%, 2.55%, 1.69% and 0.12% with respect to the SWaT, WADI, BATADAL, CISS, and CHD datasets. Moreover, the sampling function makes the GSIN 5.8384% more efficient in training compared to its counterpart performing global integration. Our future work involves an in-depth analysis of different factors shaping each individual device's influence on the GSIN's performance. Moreover, from the application's perspective, we plan on evaluating the feasibility and reliability of deploying the GSIN in real-time systems running diverse industrial processes.

ACKNOWLEDGMENTS

This work is supported by the National Key R&D Program of China (2020YFB2009502), the National Natural Science Foundation of China (No. 62272129) and the Double First-Class Scientific Research Funds of HIT (No. IDGA1010200107). The SWaT, WADI, BATADAL and CISS datasets are provided by iTrust Centre for Research in Cyber Security (https://itrust.sutd.edu.sg/dataset/), and the CHD dataset is developed by the HCRL (https://ocslab.hksecurity.net/Datasets/).

REFERENCES

[1] M. Abdallah, N. An Le Khac, H. Jahromi, and A. Delia Jurcut. 2021. A Hybrid CNN-LSTM Based Approach for Anomaly Detection Systems in SDNs. In *16th International Conference on Availability, Reliability and Security*. 1–7.

[2] Loai Abedalla, Murad Badarna, Waleed Khalifa, and Malik Yousef. 2019. K–means based one-class svm classifier. In *International Conference on Database and Expert Systems Applications*. Springer, 45–53.

[3] M. AlMedires and M. AlMaiah. 2021. Cybersecurity in Industrial Control System (ICS). In *2021 International Conference on Information Technology (ICIT)*.

[4] M. R. Asghar, Q. Hu, and S. Zeadally. 2019. Cybersecurity in industrial control systems: Issues, technologies, and challenges. *Computer Networks* 165, 106946 (2019). https://doi.org/10.1016/j.comnet.2019.106946

[5] R. R. R. Barbosa, R. Sadre, and A. Pras. 2012. A first look into SCADA network traffic. In *2012 IEEE Network Operations and Management Symposium*. IEEE, 518–521.

[6] M. Caselli, E. Zambon, and F. Kargl. 2015. Sequence-aware intrusion detection in industrial control systems. In *1st ACM Workshop on Cyber-Physical System Security*. 13–24.
[7] Lei Chen, Yuan Li, Xingye Deng, Zhaohua Liu, Mingyang Lv, and Hongqiang Zhang. 2022. Dual Auto-Encoder GAN-Based Anomaly Detection for Industrial Control System. *Applied Sciences* 12, 10 (2022), 4986.
[8] A. Deng and B. Hooi. 2020. Graph neural network-based anomaly detection in multivariate time series. In *AAAI Conference on Artificial Intelligence*, Vol. 35. 4027–4035.
[9] Xiaoheng Deng, Jincai Zhu, Xinjun Pei, Lan Zhang, Zhen Ling, and Kaiping Xue. 2022. Flow Topology-based Graph Convolutional Network for Intrusion Detection in Label-Limited IoT Networks. *IEEE Transactions on Network and Service Management* (2022).
[10] A. Dey. 2020. Deep IDS: A deep learning approach for Intrusion detection based on IDS 2018. In *2020 2nd International Conference on Sustainable Technologies for Industry 4.0 (STI)*. IEEE, 1–5.
[11] H. S. Dhiman, D. Deb, S. M. Muyeen, and I. Kamwa. 2021. Wind turbine gearbox anomaly detection based on adaptive threshold and twin support vector machines. *IEEE Transactions on Energy Conversion* 36(4) (2021), 3462–3469.
[12] Jian Du, Shanghang Zhang, Guanhang Wu, José MF Moura, and Soummya Kar. 2017. Topology adaptive graph convolutional networks. *arXiv preprint arXiv:1710.10370* (2017).
[13] Daniel Fährmann, Naser Damer, Florian Kirchbuchner, and Arjan Kuijper. 2022. Lightweight long short-term memory variational auto-encoder for multivariate time series anomaly detection in industrial control systems. *Sensors* 22, 8 (2022), 2886.
[14] Jonathan Goh, Sridhar Adepu, Khurum Nazir Junejo, and Aditya Mathur. 2016. A dataset to support research in the design of secure water treatment systems. In *International conference on critical information infrastructures security*. Springer, 88–99.
[15] N. Goldenberg and A. Wool. 2013. Accurate modeling of Modbus/TCP for intrusion detection in SCADA systems. *International journal of critical infrastructure protection* 6(2) (2013), 63–75.
[16] D. Hadžiosmanović, R. Sommer, E. Zambon, and P. H. Hartel. 2014. Through the eye of the PLC: semantic security monitoring for industrial processes. In *30th Annual Computer Security Applications Conference*. 126–135.
[17] W. Hamilton, Z. Ying, and J. Leskovec. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems* 30 (2017).
[18] Simon Hawkins, Hongxing He, Graham Williams, and Rohan Baxter. 2002. Outlier detection using replicator neural networks. In *Data Warehousing and Knowledge Discovery: 4th International Conference, DaWaK 2002 Aix-en-Provence, France, September 4–6, 2002 Proceedings 4*. Springer, 170–180.
[19] Z. Hu, Y. Dong, K. Wang, K. W. Chang, and Y. Sun. 2020. Gpt-gnn: Generative pre-training of graph neural networks. In *26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1857–1867.
[20] Paweł Karczmarek, Adam Kiersztyn, and Witold Pedrycz. 2020. n-ary isolation forest: An experimental comparative analysis. In *International Conference on Artificial Intelligence and Soft Computing*. Springer, 188–198.
[21] T. N. Kipf and M. Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).
[22] Yezheng Liu, Zhe Li, Chong Zhou, Yuanchun Jiang, Jianshan Sun, Meng Wang, and Xiangnan He. 2019. Generative adversarial active learning for unsupervised outlier detection. *IEEE Transactions on Knowledge and Data Engineering* 32, 8 (2019), 1517–1528.
[23] Wai Weng Lo, Siamak Layeghy, Mohanad Sarhan, Marcus Gallagher, and Marius Portmann. 2022. E-graphsage: A graph neural network based intrusion detection system for iot. In *NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 1–9.
[24] S. L(y)u, K. Wang, L. Zhang, and B. Wang. 2022. Global-local integration for GNN-based anomalous device state detection in industrial control systems. *Expert Systems with Applications* 209, 118345 (2022).
[25] C. Markman, A. Wool, and A. A. Cardenas. 2017. A new burst-DFA model for SCADA anomaly detection. In *2017 Workshop on Cyber-Physical Systems Security and PrivaCy*. 1–12.
[26] A. Sankar, X. Zhang, and K. C. C. Chang. 2019. Meta-GNN: Metagraph neural network for semi-supervised learning in attributed heterogeneous information networks. In *2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*. 137–144.
[27] L Shuaiyi, Kai Wang, Liren Zhang, and Bailing Wang. 2023. Process-Oriented heterogeneous graph learning in GNN-Based ICS anomalous pattern recognition. *Pattern Recognition* 141 (2023), 109661.
[28] J. Sinha and M. Manollas. 2020. Efficient deep CNN-BILSTM model for network intrusion detection. In *2020 3rd International Conference on Artificial Intelligence and Pattern Recognition*. 223–231.
[29] HM Song, J Woo, and HK Kim. 2018. Can network Intrusion datasets.
[30] Xiaoling Tao, Yang Peng, Feng Zhao, Peichao Zhao, and Yong Wang. 2018. A parallel algorithm for network traffic anomaly detection based on Isolation Forest. *International Journal of Distributed Sensor Networks* 14, 11 (2018), 1550147718814471.
[31] Riccardo Taormina, Stefano Galelli, Nils Ole Tippenhauer, Elad Salomons, Avi Ostfeld, Demetrios G. Eliades, Mohsen Aghashahi, Raanju Sundararajan, Mohsen Pourahmadi, M. Katherine Banks, B. M. Brentan, M. Herrera, Amin Rasekh, Enrique Campbell, I. Montalvo, G. Lima, J. Izquierdo, Kelsey Haddad, Nikolaos Gatsis, Ahmad Taha, Saravanakumar Lakshmanan Somasundaram, D. Ayala-Cabrera, Sarin E. Chandy, Bruce Campbell, Pratim Biswas, Cynthia S. Lo, D. Manzi, E. Luvizotto, Jr, Zachary A. Barker, Marcio Giacomoni,

M. Fayzul K. Pasha, M. Ehsan Shafiee, Ahmed A. Abokifa, Mashor Housh, Bijay Kc, and Ziv Ohar. 2018. The Battle Of The Attack Detection Algorithms: Disclosing Cyber Attacks On Water Distribution Networks. *Journal of Water Resources Planning and Management* 144, 8 (Aug. 2018), 04018048. https://doi.org/10.1061/(ASCE)WR.1943-5452.0000969

[32] Maurras Ulbricht Togbe, Mariam Barry, Aliou Boly, Yousra Chabchoub, Raja Chiky, Jacob Montiel, and Vinh-Thuy Tran. 2020. Anomaly detection for data streams based on isolation forest using scikit-multiflow. In *International Conference on Computational Science and Its Applications*. Springer, 15–30.

[33] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).

[34] Y. Wang, J. Zhang, S. Guo, H. Yin, C. Li, and H. Chen. 2021. 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*. 1239–1248.

[35] J. Yang, C. Zhou, Y. C. Tian, and S. H. Yang. 2019. A software-defined security approach for securing field zones in industrial control systems. *IEEE Access* 7 (2019), 87002–87016.

[36] J. Zhang, S. Gan, X. Liu, and P. Zhu. 2016. Intrusion detection in SCADA systems by traffic periodicity and telemetry analysis. In *2016 IEEE Symposium on Computers and Communication (ISCC)*. IEEE, 318–325.