Expert Systems With Applications 209 (2022) 118345

Contents lists available at ScienceDirect

Expert Systems With Applications

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

Global-local integration for GNN-based anomalous device state detection in industrial control systems

Shuaiyi L(y)u, Kai Wang, Liren Zhang, Bailing Wang *

Faculty of Computing, Harbin Institute of Technology, China

A R T I C L E I N F O

Keywords:
Global-local integration
Global pooling
Graph neural networks
Anomaly detection
Industrial control systems

A B S T R A C T

Anomaly detection are gaining popularity among the research communities for its essential role in securing Industrial Control Systems (ICS). Over the decades, diverse approaches have been proposed to profile anomalous behaviours propagating across the ICS networks. Recent attempts using the Graph Neural Network (GNN) methodologies have enabled state prediction of a device node via encoding its immediate neighbourhood. Such an encoding scheme potentially compromises the model's detection accuracy due to the nodes' biased attention towards their local surroundings. To investigate this issue, we present the **Global-Local Integration Network (GLIN)** that achieves **node-level anomaly detection** by merging a node's local and the network's global expressions. It comprises a preprocessor for graph construction and data transformation, an encoder for node embedding learning, a pooling module producing global representations, an integration module that performs message fusion, and a decoder for label prediction. We develop and evaluate GLIN with 7 global integration schemes and train it over 3 message passing mechanisms. We compare its performance against both classical machine learning and recent deep learning baselines and demonstrate its superiority in terms of multiple popular metrics. Finally, we provide useful insights on the results and suggest promising future work directions.

1. Introduction

Security issues in ICSs have attracted a considerable amount of attention over the previous decades (Asghar et al., 2019). Due to their increasing disclosure to the Internet, these systems have been vulnerable to the malicious events of all categories across the cyberspace, ranging from the well-known attacks such as the Denial-of-Service (DoS) and Device Spoofing, to the subtly crafted zero-day semantic attacks. The study of **anomaly detection** for securing industrial control systems calls for a growing demand due to the ICSs' significance in ensuring proper functioning of all critical infrastructures in the society. These systems' current fragility against hostile interactions has led to a drastic rise in the proposed strategies over the past few years.

Early contributions in the ICS anomaly detection apply regular data mining techniques to the individual network flows. They view certain statistical characteristics as a baseline upon which excessive deviations are detected as potential anomalies. Periodicity has been widely exploited for the construction of numerous classic models including the **Deterministic Finite Automata** (Goldenberg and Wool, 2013) (Markman et al., 2017) series and the **Statechart-based Detectors** (Kleinmann and Wool, 2016). These solutions benefit from their simplicity of implementation as well as their omittable computation cost. However, the fact that the statistical properties serve as a signature has led to an undesired oversensitivity undermining the models' performance. These solutions also suffer from a generalization inadequacy for considering only the visible or inferable properties a flow exhibits. And their application is restricted within a particular local connection due to their incapability of incorporating global analysis.

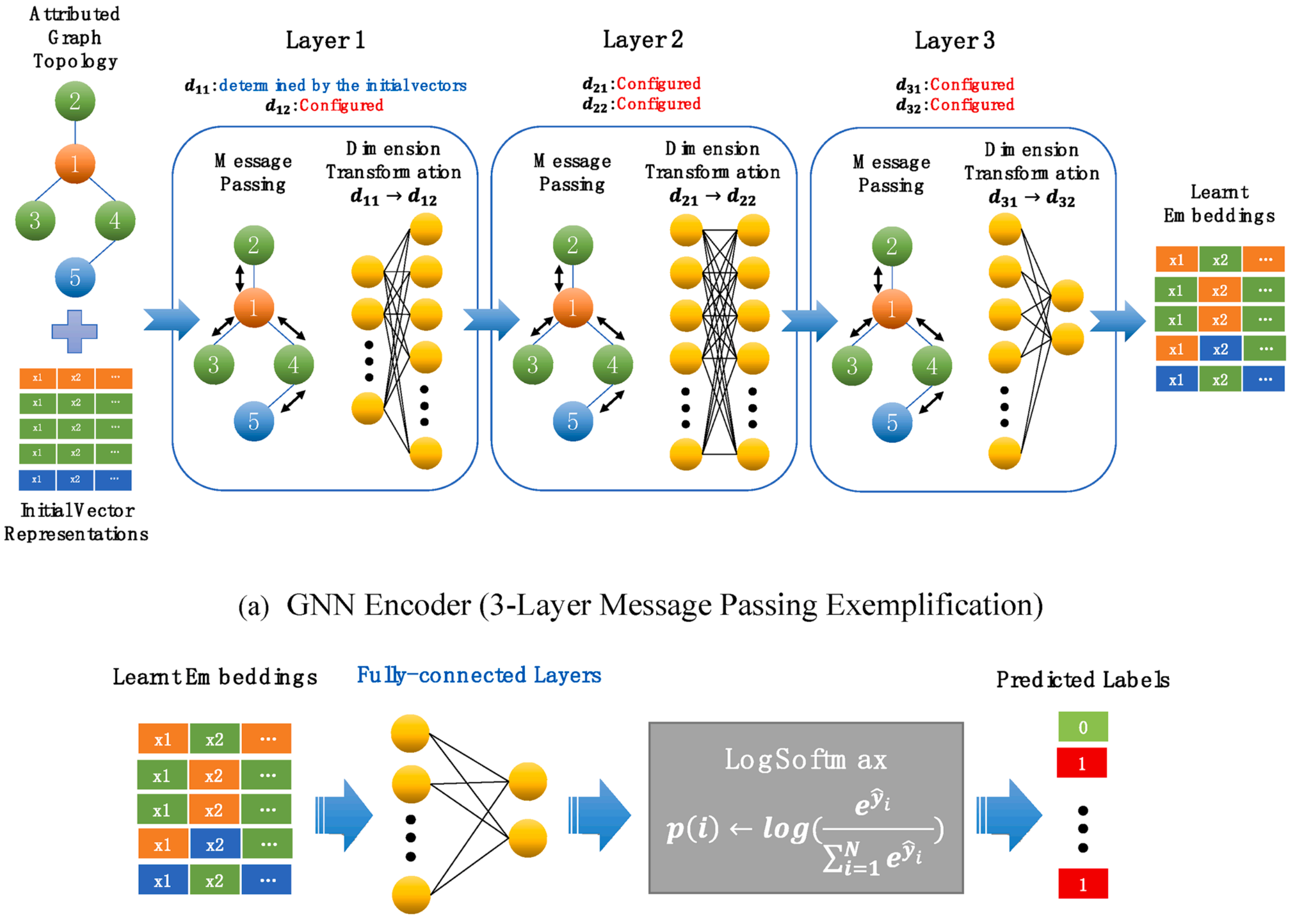
Advances in machine learning have promoted the neural networks to one of the most popular anomaly detection methodologies investigated in the literature. Classical techniques integrate the **Convolutional Neural Networks (CNNs)** and the **Long Short Term Memory modules (LSTMs)** (Dey, 2020) (Abdallah et al., 2021) (Sinha and Manollas, 2020) as their core building components to perform feature extraction over the

Abbreviations: ICS, Industrial Control System; AAE, Adversarial Autoencoder; CNN, Convolutional Neural Network; DFA, Deterministic Finite Automata; HMI, Human Machine Inteface; GAN, Generative Adversarial Network; GAT, Graph Attention Network; GCN, Graph Convolutional Network; GLIN, Global-Local Integration Network; GNN, Graph Neural Network; Graph-SAGE, Graph Sample-and Aggregate; PCA, Principal Component Analysis; PLC, Programmable Logic Controller; SVM, Support Vector Machine; SWaT, Secure Water Treatment; TWSVM, Twin Support Vector Machine; VAE, Variational Autoencoder.

* Corresponding author.

E-mail address: wbl@hit.edu.cn (B. Wang).

https://doi.org/10.1016/j.eswa.2022.118345
Received 27 April 2022; Received in revised form 19 July 2022; Accepted 30 July 2022
Available online 4 August 2022



(a) GNN Encoder (3-Layer Message Passing Exemplification)

(b) GNN Decoder (Node Property Prediction Exemplification)

Fig. 1. General GNN Architecture.

systems' spatial and temporal behaviours. While they surpass the traditional approaches in performance, most of the proposed frameworks either perform general analysis over the entire network or fail to take into account the inter-node relationships when engaged in node-level state prediction.

Researchers have recently switched focus to the **Graph Neural Networks (GNNs)** (Veličković et al., 2017) (Velickovic et al., 2019) (Kipf and Welling, 2016a), (Kipf and Welling, 2016b) (Hamilton et al., 2017) for alternative insights due to the GNNs' powerful ability to operate on graph-structured data abstracted from a network topology. Numerous variations have emerged over the past five years, including the **Graph Convolutional Networks (GCNs)** (Kipf and Welling, 2016b), the **Graph Attentional Networks (GATs)** (Veličković et al., 2017) and the **Graph-SAGE** (Hamilton, 2017), etc. Despite their success in improving the performance of node-level state detection by means of direct neighbourhood aggregation, such a mechanism leads to a potential drawback that arises from the nodes' imbalanced awareness of the overall network architecture. Hence, the resulting embeddings may not be sufficient enough to serve as an appropriate basis for subsequent tasks in networks rich in sophisticated global semantics, like the ICSs.

Therefore, we introduce the **Global-Local Integration Network (GLIN)** that achieves anomaly detection in ICSs by merging a node's local and the network's global expressions. The proposed framework has the benefit of producing node embeddings featuring both the node's local characteristics as well as its complex universal associativity, which makes it distinguishable from existing GNN approaches. The conjunction of local and global properties poses an extensively positive influence on the model's performance in anomaly detection. The **GLIN** comprises the following components: **(a)** A **Preprocessor** that transforms the original data flow into the initial vectors used for message passing. **(b)** An **Encoder** for node embedding learning, which performs message passing and maps for all nodes their initial vectors to their corresponding local representations. **(c)** A **Pooling Module** that generates the global expression by incorporating all the nodes' local representations output by the preceding Encoder. **(d)** An **Integration Module** encapsulating the global expression vector into the nodes' local embeddings, and **(e)** A **Decoder** performing label prediction based upon the integrated vectors.

We summarize our key contributions as follows:

1) We design and fabricate the GLIN with multiple classic GNN message passing blocks including GCN, GAT and Graph-SAGE. We create the graph from an ICS's typical layered architecture and initialize the node vectors using the SWaT dataset.
2) We apply and evaluate various pooling and integration schemes on the embeddings produced by the aforementioned GNN message passing blocks. We compare the performances of GLIN models with different configurations against the original GNN model without global integration, using multiple evaluation metrics.
3) We demonstrate GLIN's effectiveness in comparison to several state-of-the-art anomaly detection approaches, and provide further insights and argumentation from various perspectives.

The rest of the paper is structured as below. Section 2 provides an overview of the literature on ICS anomaly detection. Section 3 defines the problem and illustrates the general principles of our proposed framework. Section 4 elaborates on the experimenting procedures and presents the results. Section 5 discusses the results and Section 6 concludes the paper.

2. Preliminaries and related work

In this section, we provide an overview of GNN preliminaries our work is based upon, and discuss current literature relevant to ICS anomaly detection.

2.1. Preliminaries

GNNs are an extensively effective solution to graph related problems such as node property prediction and link inference, and are hence explored as a means of anomaly detection in diverse ranges of scenarios. A GNN model takes in an attributed graph (a graph with node representations) and encodes these representations via message passing, resulting in node embeddings that serve as a basis for subsequent inference tasks.

A typical GNN framework comprises an encoder incorporating multiple hidden layers, and a decoder exclusively tailored to specific tasks. Their general architectures are illustrated in Fig. 1. Each hidden layer performs a single round of message aggregation over the nodes' immediate neighbourhood, and the aggregated vector is updated and possibly shifted to a different dimension before exiting the current block.

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

$$h_v^{(k+1)} \leftarrow UPDATE\left(h_v^{(k)}, AGGREGATE\left(u \in N(v), MESSAGE\left(h_v^{(k)}, h_u^{(k)}, e(u,v)\right)\right)\right) \tag{1}$$

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.

Among all operators, $MESSAGE(.)$ formulates the message from a particular neighbour node u by extracting important information from u and v and the edge connecting them. $AGGREGATE(.)$ 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.

The decoder's function varies among tasks. In node property prediction, for example, the decoder transforms the learnt node embeddings into labels representing the level of existence of a particular feature. In anomaly detection, the labels are usually defined as binary variables with $0'$s denoting normal cases and $1'$s otherwise.

2.2. Related work

We categorize the related work into the following perspectives: traditional ICS anomaly detection methods, machine learning and deep learning ICS anomaly detection methods.

2.2.1. Traditional methods

Traditional anomaly detection methods proposed for industrial control networks range from classic statistical analysis to state automata generation (Goldenberg and Wool, 2013) (Markman et al., 2017) (Yang et al., 2019) (Zhang et al., 2016) (Ao, 2020) (Kleinmann and Wool, 2016) (Kleinmann et al., 2017). They offer lightweight and interpretable solutions to a fairly decent variety of existing security relevant problems. These solutions benefit from multiple application advantages including implementation and deployment simplicity. However, they suffer from a rather large and sometimes unacceptable false detection rate for their incapability of discovering in-depth network characteristics. Details of some of the typical solutions are described below.

An intrusion detection system is presented (Goldenberg and Wool, 2013) that constructs state-based directed-graph models using an HMI-PLC channel's own Deterministic Finite Automation (DFA). It is created on the assumption that the communication flow between two hosts within a fixed channel follows a pattern that repeats itself at constant and well-defined time intervals, thus exhibiting a strong inclination for periodicity. The model is applied to Modbus/TCP traffic and deeply inspects each packet in the network flow. Channels, which are identified with a tuple that comprises host-specific features, are separated and analyzed independently. Application layer features are extracted to contribute to the composition of a state symbol in the model, and the number of states with each period is derived at the end of the DFA modelling process. The model is analyzed and evaluated on its normal state rate, re-transmission rate, miss rate as well as its unknown state rate. The authors claim that the model works perfectly on short and fast periods, but doesn't provide the same level of excellence in performances on long and slow periods.

Burst transmissions are discovered within individual Modbus/TCP communication session (Markman et al., 2017). Evidence provided indicating the presence of semantic messages within each burst, an open-loop DFA model is developed that describes the states and transition functions residing in every message. The model uses a threshold generated with a variance-based method to distinguish packets of different bursts from each other and, distinct from other DFA representations, introduces two boundary states to denote the beginning and the end of a burst. Experiments are conducted to evaluate the model's overall packet recognition rate, accuracy and permissiveness, the final of which the authors suggest a normalized metric to analyze and assess. This model surpasses other DFA models when tested on small systems with bursty channels for its compactness, effectiveness and efficiency. However, as the authors concedes, it tends to exhibit unacceptably high False Alarm Rate when applied to large scale systems.

A software-defined security (SDSec) approach (Yang et al., 2019) is presented to help prevent the propagation of attack impact among field zones in ICSs. The solution is comprised of a hybrid anomaly detection module and a multi-level security response module. The hybrid anomaly detection module extracts critical features from network traffic, and analyses them so as to obtain an overall security level for the zones being inspected. The work also develops a multi-level security response module responsible for attack isolation and mitigation. In this module, all traffic and state anomalies are simply blocked while data anomalies are coped with from the perspective of mitigation. The authors investigate their approach on a typical linear time-invariant system and propose a global mitigation model, in which a switched Kalman filter is adopted to compute the compensation signal against the related attack.

An intrusion detection mechanism (Zhang et al., 2016) is created via exploitation of periodicity and telemetry patterns of network flow within typical SCADA systems. The scheme requires construction of multiple modules that operate together to realize detection of adversaries in 4 categories, known as reconnaissance, response injection, command injection and denial of service. The authors claim that the proposed method bridges the gap that few other algorithms successfully address by inspecting response injection and attacks of other types concurrently. Their simulation results indicate that the suggested mechanism leads to rather low false positive and negative rates and is more efficient than other methods.

In general, the solutions discussed above are easy to implement and exhibit outstanding runtime efficiency. Nevertheless, they suffer from an inadequacy in analyzing a flow's in-depth properties, and their application is constrained within particular localities for the lack of global analysis.

2.2.2. Machine and deep learning methods

As the machine learning techniques advance in the past decade, deep learning based methodologies and application on anomaly detection have been emerging at a dramatic rate and swarming the research community. As opposed to the traditional solutions, deep learning models exploit the network properties via data mining and intensive feature extraction. Hence they achieve superior performance in terms of detection accuracy. However, lack of interpretability continues to be one of their drawbacks despite the recent attempts of model melioration in various fields of application.

2.2.2.1. Classical machine and deep learning methods. To meliorate the reliability of data-driven condition monitoring in wind turbine operations, an automated fault detection approach based on adaptive threshold and the Twin Support Vector Machine (TWSVM) is introduced (Dhiman et al., 2021). The anomaly detection problem is defined as a binary classification task in which samples are labeled via computation of adaptive threshold of univariate series. Then the TWSVM is applied to produce dual hyperplanes for state classification. The model is assessed over the SCADA dataset captured from British window farms and evaluated against various classic methods in terms of false negative and false positive rates. Results suggest the model's improved reliability over existing baselines.

Extensions of deep autoencoders are investigated (Zhou and Paffenroth, 2017) that discover high quality non-linear characteristics while removing outliers simultaneously without access to any clean training data. They propose "Robust Deep Encoder (RDA)" that can effectively filter out random noise. Moreover, they offer generalizations over grouped sparsity norms, differentiating random anomalies from other types of structured corruption, which leads to superior performance in anomaly detection problems.

An adversarial autoencoder (AAE) based approach (Jang et al., 2021) is proposed for fault detection via nonlinear process monitoring. The model merges a variational autoencoder (VAE) and a generative adversarial network (GAN), and is enabled to capture high-dimensional characteristics in data flows that follow disparate or random statistical distributions. With a GAN discriminator involved, the generated features are deemed to be more reliable and effective when utilized in anomaly detection. The AAE model is examined over 21 types of faults over the Tennessee Eastman benchmark and evaluated against multiple anomaly detection baselines. Its superiority is demonstrated in terms of fault detection rate, false alarm rate and fault detection delays.

Hybrid intrusion detection systems (Dey, 2020) (Abdallah et al., 2021) are developed via the incorporation of Convolutional Neural Network and Long-short Term Memory Network. The resulting models are effective on capturing spatial and temporal properties of the network traffic, and are demonstrated feasible on the recognition of zero-day attacks. It is also proved that the combined model achieves a higher detection accuracy than each individual model, and that regularization positively influences CNN's performance in detecting new intrusions. Another hybrid model is proposed (Sinha and Manollas, 2020) for intrusion detection, while it differs from (Dey, 2020) and (Abdallah et al., 2021) in that it combines CNN with a Bi-directional LSTM. This model, as the authors claim, excels many state-of-the-art baselines with a better accuracy and a lower false positive rate.

While these methods outperform the traditional approaches in terms of functionality, most of the proposed frameworks suffer from their insufficient interpretability as to how in-depth features are extracted and utilized to derive desirable results. Moreover, their tuning schemes tend to be relatively obscure and the tuning effects are also not inferable.

2.2.2.2. Graph neural network methods. The literature review (Wu et al., 2021) provides useful insights on GNNs for anomaly detection in Industrial Internet of Things (IIoT). The authors classify existing anomalies into 3 domains, discuss their showcases, and introduce relevant principles of GNN measures taken to address these issues. They also investigate the challenges in current models and shed light on potential future research directions on GNN anomaly detection.

A GNN model detecting node-level anomalies is established (Chen et al., 2020). The authors conduct experiments on the SWaT data, analyze the physical workflow and transform the physical stages into a graph for subsequent training.

A synergistic approach is proposed (Zhao et al., 2021) utilizing data mining to instruct GNN algorithms on how to effectively aggregate local information to obtain global patterns. The authors also provide a novel loss scheme in (Zhao et al., 2020) to assist in the training of GNNs for anomaly-detectable node representations.

A new training scheme is explored (Wang et al., 2021) via decoupling the node representation learning and the classification of states in GNN anomaly detection models. The authors' preliminary study show that decoupled training tends to exceed joint training in terms of performance, but may deteriorate on condition that the behaviour patterns and the label semantics become highly inconsistent. Based on this prior knowledge, they propose Deep Cluster Infomax for node representation while mitigating the bias triggered by the inconsistency. Results prove that the model generated from decoupled training outperforms that from joint training in AUC.

A semi-supervised learning technique (Sankar et al., 2019) is introduced exclusively for the analysis of attributed heterogeneous information network. The authors generalize regular GCNs by introducing a new message aggregation paradigm via *meta*-graph semantics. They develop a novel *meta*-graph convolutional mechanism to extract features from *meta*-graph structured neighbourhood in order to capture higher-order semantic relationships exhibited in the network. They construct Meta-GNN to perform node classification on real-world data sets, achieving significant performance gains over state-of-the-art baselines.

These GNN approaches are successful in benefiting node-level state prediction via direct neighbourhood aggregation. However, this local nature of the proposed aggregation schemes imposes limitations on a node's awareness from a global perspective. Therefore, the resulting node representations may not be optimal for subsequent tasks in networks rich in sophisticated global semantics, like the ICSs.

3. Problem definition and model design

In this section, we formulate the problem to be addressed and comprehensively introduce the concept and logic of the GLIN.

3.1. Problem statement

One of the aspects differentiating the ICSs from regular networks lies in the vulnerability of the underlying physical processes vital to the entire industry. The anomalies in ICSs not only comprises the uncanny phenomena triggered by the wide range of cyber attacks on the network level, but may also result from the sneaky attempts in breaching the regular system operating convention, which potentially leads to chaotic physical states. This work aims at creating a method that can serve as an application in an ICS network for global flow inspection, while performing device-level anomaly detection at any time of interest. In this work, the term anomaly is specifically defined within the scope of physical processes. Hence only the device readings and timestamps are taken into consideration when feature extraction is performed during preprocessing, as discussed in Section 4. That said, the general anomaly detection problem is defined as below:

Definition 1. *Given an industrial field network* $G(V(t), E(t), X(t), Y(t))$, *where* $V(t)$, $E(t)$, $X(t)$ *and* $Y(t)$ *correspond to the set of network's vertices, edges, node (device) reading features and labels at time t, respectively, find model* $M(V(t), E(t), X(t)) = \widehat{Y}(t)$ *such that for* $\forall t, \widehat{Y}(t) = Y(t)$.

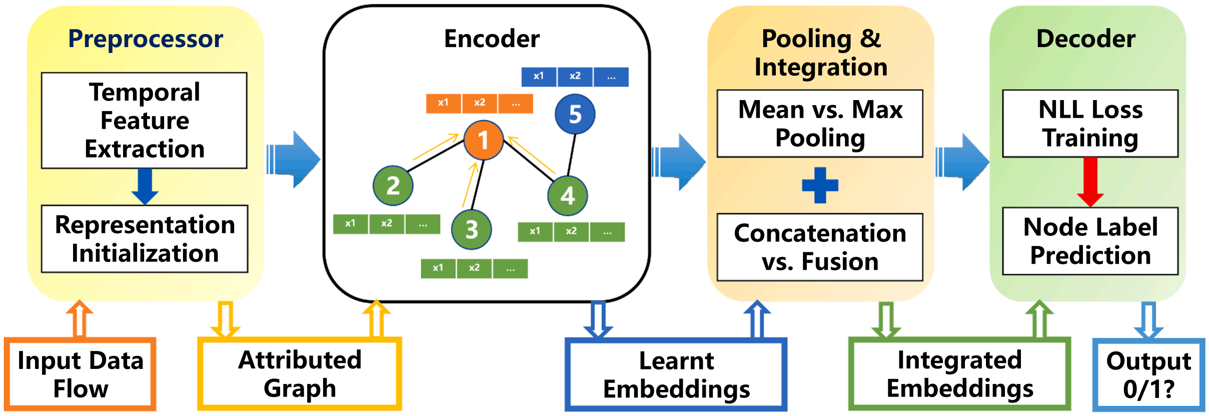


Fig. 2. General GLIN Architecture.

In practice, the nature of structural persistency in industrial control networks prompts the assumption that the graph structure remains stationary over time. Therefore, we streamline the term G by replacing $V(t)$ and $E(t)$ with V and E.

3.2. Model design

We present the GLIN framework to address the problem defined above by integrating local and global representations. The overall architecture is displayed in Fig. 2.

3.2.1. Preprocessor

As clarified in Section 3.1, initial feature vectors embody the physical reading properties of the corresponding devices at some particular moment. Due to the spatial nature of GNN type models, temporal features are usually separately encoded. The Preprocessing Module adopts the Sliding Window Exponential Weighted Average (SW-EWA) algorithm **(see Algorithm 1)** to synthesize all values within a series of time intervals (windows) so that the newly generated data capture the reading's most recent temporal characteristics. These intervals are of a pre-configured size and the values in each interval are mapped accordingly to the entries of a new vector representation in the following manner:

$$X(t) = [X(t,0), X(t,1), \cdots, X(t,m-1)]^{T} t \in T$$

$$X(t,j) = \sum_{i=0}^{n/m} w(t,i,j) \bullet x_0(t-i-j\times(n/m)) for \forall t > n, 0 \le j < m$$

where $X(t)$ is the new vector produced at time t for a specific device, m is the length of $X(t)$, $X(t,j)$ is the value of the j-th entry in $X(t)$, n denotes the window size, T is the set of all the time ticks of interest and $x_0(t)$ represents the original sample at time t. $w(t,i,j)$ regulates the contribution of the original sample at time $t-i-j\times(n/m)$ to the j-th entry in the new vector. Under the assumption that the importance of certain context diminishes with time, the weighing factors in our implementation decay as time backtracks. Therefore, we employ a set of exponential weights in this step of our preprocessing procedure.

$$w(t,i,j) = e^{-\frac{a}{n}(i+j\times(n/m))} \Big/ \sum_{j=0}^{m-1} \sum_{i=0}^{\frac{n}{m}-1} e^{-\frac{a}{n}(i+j\times(n/m))} for \forall t > n, 0 \le j < m$$

Note that a is the coefficient that balances the distribution of all weights in a specific window.

Algorithm 1
Sliding Window Exponential Weighted Average (SW-EWA).

```
Input: Readings captured at all time ticks for all applicable devices (sensors and
  actuators). Input has dimension $[\#devices, \#time\_ticks]$
Output: List of aggregated sample vectors for all devices $list_{all}$. Output is of dimension
  $[\#devices, \#time\_ticks - n + 1, m]$
Ensure:
Initialize the window size $n$ and the distribution balancing coefficient $a$
Time index $t \leftarrow n$; Sum $s \leftarrow 0$; List $list_{all} \leftarrow emptylist$
for all devices:
    $list \leftarrow emptylist$
    while $t \le \#time\_ticks$:
        $list_t \leftarrow empty\_list$
        for all $j$ in $[0, m-1]$ (This loop produces an $m$-dimensional vector
  representation for the current device at time tick $t$)
            $e_j \leftarrow 0$
            for all $i$ in $\left[0, \frac{n}{m} - 1\right]$ (Each element in this vector is a fusion of $\frac{n}{m}$
  consecutive values in the input)
                compute weight $w(t,i,j)$
                $e_j \leftarrow e_j + w(t,i,j) \bullet x_0\left(t-i-j\times\left(\frac{n}{m}\right)\right)$
            end for
            $list_t \leftarrow list_t + e_j$
        end for
        $list \leftarrow list + list_t$
        $t \leftarrow t + 1$
    end while
    $list_{all} \leftarrow list_{all} + list$
end for
return $list_{all}$
```

By performing temporal compression, we transform the original reading sequences into their respective graph node features at different time ticks, generating the initial representations for the subsequent training process. Since these features are measured on different scales, we apply tangent normalization on all the features to circumvent potential training bias toward the nodes with dominant magnitude.

$$X_{norm}(t) = arctan(X(t)) t \in T$$

where $X_{norm}(t)$ is the normalized version of $X(t)$.

3.2.2. Encoder

The Preprocessor produces the initial vectors X for message passing. Each of these vectors comes with a particular device and a specific time tick t, and is of dimension m. Given the scale of the time tick pool is tremendously large, in order to leverage our server's computation ability in a reasonable fashion, these time ticks are grouped into mini-batches for further operation. In our implementation, each mini-batch contains vectors from 64 time ticks.

The graph for message passing consists of subgraphs abstracted from

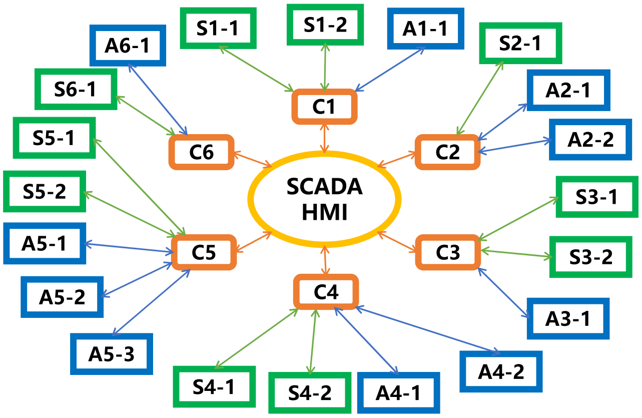


Fig. 3. Star-shaped Subgraph Topology (*Ci* refers to the controller labeled *i*, and $S(A)i-j$ corresponds to the *j*-th Sensor (Actuator) directly linked to the *i*-th Controller).

the real-time network topology captured at all the time ticks in a single batch. It is a time-variant element for dynamic networks with frequent addition or removal of nodes and links. Nevertheless, since little dynamics is conventionally accepted in an ICS network, we assume that our graph remains stationary for all time ticks of interest. Hence, our graph ends up a replication of a subgraph captured at any particular time tick and remains stable across all mini-batches. This subgraph can be defined as a star shaped topology shown in Fig. 3 which is a reflection of the ICSs' typical layered architecture. Let A_s be the adjacency matrix of this subgraph, then the adjacency matrix for the entire graph $\boldsymbol{A}$ is produced via the diagonal concatenation as follows:

$$A \leftarrow \begin{pmatrix} A_s & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & A_s \end{pmatrix}_{k\times k}$$

where k equals the batch size.

The message passing layer (or hidden layer) is empirically configured with a fixed number of nodes, which in our case is set to 128. Double hidden layers are implemented and the ReLU function is adopted for nonlinear activation. Therefore, the number of parameters in the $\boldsymbol{W}$ matrices corresponding to these layers equals 128m and 128^2, respectively. The general embedding learning follows the scheme shown below:

$$H^{(l+1)} \leftarrow \sigma\left(\widetilde{A}H^{(l)}\boldsymbol{W}\right) l \leftarrow 0, 1 and H^{(0)} \leftarrow \boldsymbol{X}$$

where $\widetilde{\boldsymbol{A}}$ is the normalized version of $\boldsymbol{A}$ and $\boldsymbol{l}$ labels the respective hidden layer.

We develop and test our model over multiple convolutional schemes including the **GCN, GAT and Graph-SAGE**. Details are discussed in Section 4.

3.2.3. Pooling module

The Pooling Module constitutes part of the core functionality differentiating the GLIN from existing frameworks. It extracts from individual nodes the graph's universal associative characteristics as content in the global representation, which is combined with the nodes' local expressions in subsequent Integration Modules. The distilled global expression contains representative information that characterizes the whole graph, and is utilized to extensively increase the nodes' global awareness, making the node embeddings informative enough to yield a superior node-level inference accuracy.

Our Pooling Module generates the global expressions for different time ticks by extracting statistical features over the learnt embeddings produced by the Encoder. This is a technique frequently used in the Convolutional Neural Networks (CNNs). Among all the pooling mechanisms, **max** pooling and **average** (or **mean**) pooling are the most popular and thus they are applied to our framework directly. The peripheral algorithm for the pooling operations is described below **(See Algorithm 2)**.

Table 1
Investigated GLIN Pooling & Integration Schemes.

No.	Pooling	Integration
1	Max	Cat.
2	Max	Fusion
3	Mean	Cat.
4	Mean	Fusion
5	Mean & Max	Cat.
6	Mean & Max	Fusion
7	Mean & Max	Fusion-Cat

Note that in the final combination, the Fusion operation is executed on the global vectors before the result is appended to the nodes' local embeddings via the Cat. operation.

Algorithm 2
Batch Global Pooling.

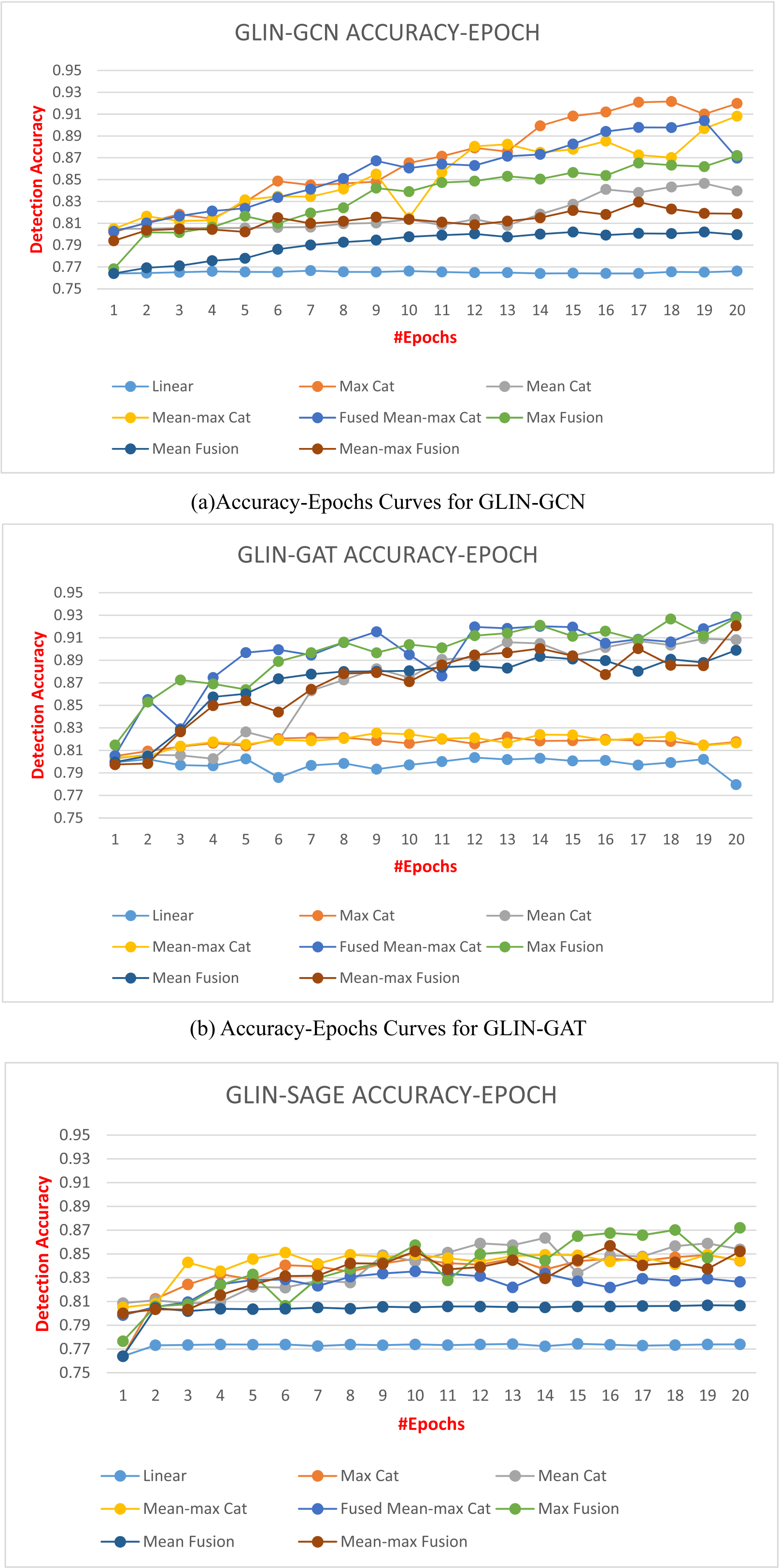
```
Input: Embedding Matrix $\boldsymbol{H}$ obtained from encoding the given batch $\boldsymbol{B}$
Output: List of pooling vectors $list_p$
Ensure:
$list_{pmax}, list_{pmean} \leftarrow emptylist, emptylist$
for all time ticks $\boldsymbol{t}$ in $\boldsymbol{B}$:
    $H_t \leftarrow matrixstackedfromnodeembeddingsw.r.ttimetickt$
    if max pooling configured:
        $v_{max} \leftarrow maxpooling(H_t)$
        $list_{pmax} \leftarrow list_{pmax} + v_{max}$
    end if
    if mean pooling configured:
        $v_{mean} \leftarrow meanpooling(H_t)$
        $list_{pmean} \leftarrow list_{pmean} + v_{mean}$
    end if
end for
$list_p \leftarrow [list_{pmax}, list_{pmean}]$
return $list_{all}$
```

3.2.4. Integration module

The Integration Module enables another unique functionality that makes the GLIN outstanding and distinguishable. This component encapsulates the nodes' local characteristics with their universal graph-level awareness, which is accomplished via combination of a node's embedding learnt from the Encoder and the global graph expression produced by the Pooling Module. By applying this integration, all nodes' knowledge spans over the entire graph, creating expressions that contain a node's local representative information as well as its sophisticated associativity with the rest of the graph, thus posing a positive influence on the model's performance in node-level state inference.

Similar to global pooling, the integration module also involves two means of operations, namely vector concatenation (or "cat" for short) and vector fusion. For concatenation, we append any applicable pooled global vectors to the rear of a node's embedding. For example, a "max-cat" operation combines a node's own vector representation with the global expression generated via max pooling, doubling the vector's dimension in the meantime. The fusion operation is defined as a weighted average operation over the local and global embeddings, which in our case, these vectors are treated equally. For example, a "mean-max-fusion" operation computes the element-wise average of the node's own embedding and the global vectors produced via the mean and max pooling step. Unlike the Cat. operation, the vector's dimension remains unchanged after fusion.

In our implementation, we develop and evaluate 7 pooling-integration combinations as listed below (TABLE 1).



(a)Accuracy-Epochs Curves for GLIN-GCN

(b) Accuracy-Epochs Curves for GLIN-GAT

(c) Accuracy-Epochs Curves for GLIN-SAGE

Fig. 4. Validation accuracy on GLIN with different message passing layers, applying all global integration schemes.

3.2.5. Decoder

The Decoder maps the integrated node representation to an N-dimensional vector $\widehat{y}$ with a fully-connected layer, where N stands for the number of classes. Then the Logarithm-Softmax function is applied to compute the logarithmic probability $lp(i)(1 \leq i \leq N)$ of all the classes a specific node is associated with.

$$lp(i) \leftarrow \log\left(\frac{e^{\widehat{y_i}}}{\sum_{j=1}^{N} e^{\widehat{y_j}}}\right) 1 \leq i \leq N$$

where $\widehat{y}_i$ refers to the i-th entry in $\widehat{y}$.

During the training process, instead of employing the binary cross-entropy loss function, we adopt the NLLLoss to achieve more efficient coordination with the Logarithm-Softmax Decoder. In this circumstance, we preserve the model's ability to scale along with the number of classes requiring differentiation.

4. Evaluation

In this section, we illustrate the evaluation details and present the results. All our evaluation processes are run over two sets of data extracted from the SWaT dataset. One dataset, referred to as the **Cross-Neg Dataset** includes the negative samples and the positive ones obtained around the point an anomaly commences. The other one, denoted as the **Pos-Neg Dataset**, covers the negative samples as well as the positive ones captured a while after an anomaly occurs. Each dataset totally contains 860,160 samples.

4.1. Metrics

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

Accuracy: Assesses a model's ability to classify the samples to their ground-truth categories.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

Precision: Measures a model's accuracy over the detected positive samples. This metric is usually of critical significance in evaluating ICS auditing and security systems, given the convention that availability takes priority when it comes to industrial processes.

$$Precision = \frac{TP}{TP + FP}$$

Recall: Evaluates a model's generality over all positive samples (i.e. anomalies of all sorts).

$$Recall = \frac{TP}{TP + FN}$$

F1 Score: Compiled metric incorporating **Precision** and **Recall**. Measures a model's overall effectiveness in detecting positive samples.

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

4.2. Validation results

2-layered GLINs with multiple message passing schemes are implemented, configured and trained as described in Section 3. Their Validation "**Accuracy-Epoch**" curves are plotted in Fig. 4.

Rapid accuracy convergence occurs in most of the scenarios, as illustrated in Fig. 3. The same applies for other metrics as well, while the respective figures are omitted.

Table 2
Test Results for all implemented GLIN frameworks. Each entry encapsulates 2 values separated by '|' corresponding to the 2 datasets defined at the beginning of Section 4. i.e. the format "Cross-Neg result | Pos-Neg result".

Model	Accuracy(%)	Recall(%)	Precision(%)	F1 Score(%)
GCN-Linear	65.27 | 76.13	59.48 | 50.36	69.55 | 75.87	64.12 | 60.54
GCN-Max Cat.	93.16 | 96.10	93.47 | 93.57	92.76 | 95.60	93.11 | 94.53
GCN-Max Fusion	75.15 | 84.33	70.18 | 67.42	85.06 | 91.26	76.91 | 77.55
GCN-Mean Cat.	86.52 | 85.80	86.50 | 75.89	86.04 | 82.76	86.27 | 78.41
GCN-Mean Fusion	76.80 | 79.93	76.19 | 58.54	76.13 | 86.84	76.16 | 69.94
GCN-FMM Cat.	87.11 | 92.82	84.70 | 93.55	90.36 | 88.78	87.44 | 90.76
GCN-MM Cat.	87.41 | 91.73	85.14 | 92.48	90.27 | 87.37	87.63 | 89.43
GCN-MM Fusion	86.70 | 79.76	84.17 | 58.02	90.21 | 87.95	87.09 | 69.92
GAT-Linear	66.01 | 77.95	60.31 | 54.17	70.70 | 87.20	65.09 | 66.83
GAT-Max Cat.	89.21 | 81.50	89.37 | 63.85	88.76 | 81.42	89.07 | 71.55
GAT-Max Fusion	90.19 | 93.43	89.95 | 88.47	89.89 | 93.13	89.92 | 90.74
GAT-Mean Cat.	87.49 |91.89	85.85 | 84.79	88.72 | 92.52	87.26 | 88.48
GAT-Mean Fusion	85.23 | 90.21	83.62 | 81.19	86.03 | 91.42	84.81 | 86.00
GAT-FMM Cat.	91.60 | 94.80	90.66 | 90.19	92.12 | 95.41	91.39 | 92.73
GAT-MM Cat.	91.38 | 81.64	90.13 | 63.18	92.42 | 84.10	91.26 | 72.15
GAT-MM Fusion	88.73 | 91.78	87.60 | 85.02	89.25 | 91.88	88.42 | 88.31
SAGE-Linear	61.75 | 76.99	54.63 | 52.15	68.59 | 85.84	60.82 | 64.88
SAGE-Max Cat.	85.65 | 82.97	83.03 | 67.05	89.18 | 83.07	86.00 | 74.21
SAGE-Max Fusion	93.05 | 89.47	92.96 | 84.33	92.78 | 86.19	92.87 | 85.25
SAGE-Mean Cat.	91.41 | 86.89	91.41 | 73.16	91.04 | 91.36	91.23 | 81.25
SAGE-Mean Fusion	85.08 | 80.23	84.13 | 59.19	85.00 | 87.05	84.56 | 70.47
SAGE-FMM Cat.	93.73 | 80.10	93.05 | 58.55	94.09 | 89.62	93.57 | 70.83
SAGE-MM Cat.	85.31 | 84.19	82.38 | 71.69	89.88 | 81.61	85.97 | 76.33
SAGE-MM Fusion	83.29 | 88.81	79.96 | 90.29	88.79 | 87.66	84.15 | 83.82

4.3. Test results

Test results are shown in Table 2 (The term "FMM" and "MM" in the "Model" column is abbreviation of "Fused Mean-max" and "Mean-max", respectively).

Performance discrepancies among different types of global-integration schemes are conveniently visible in Fig. 5.

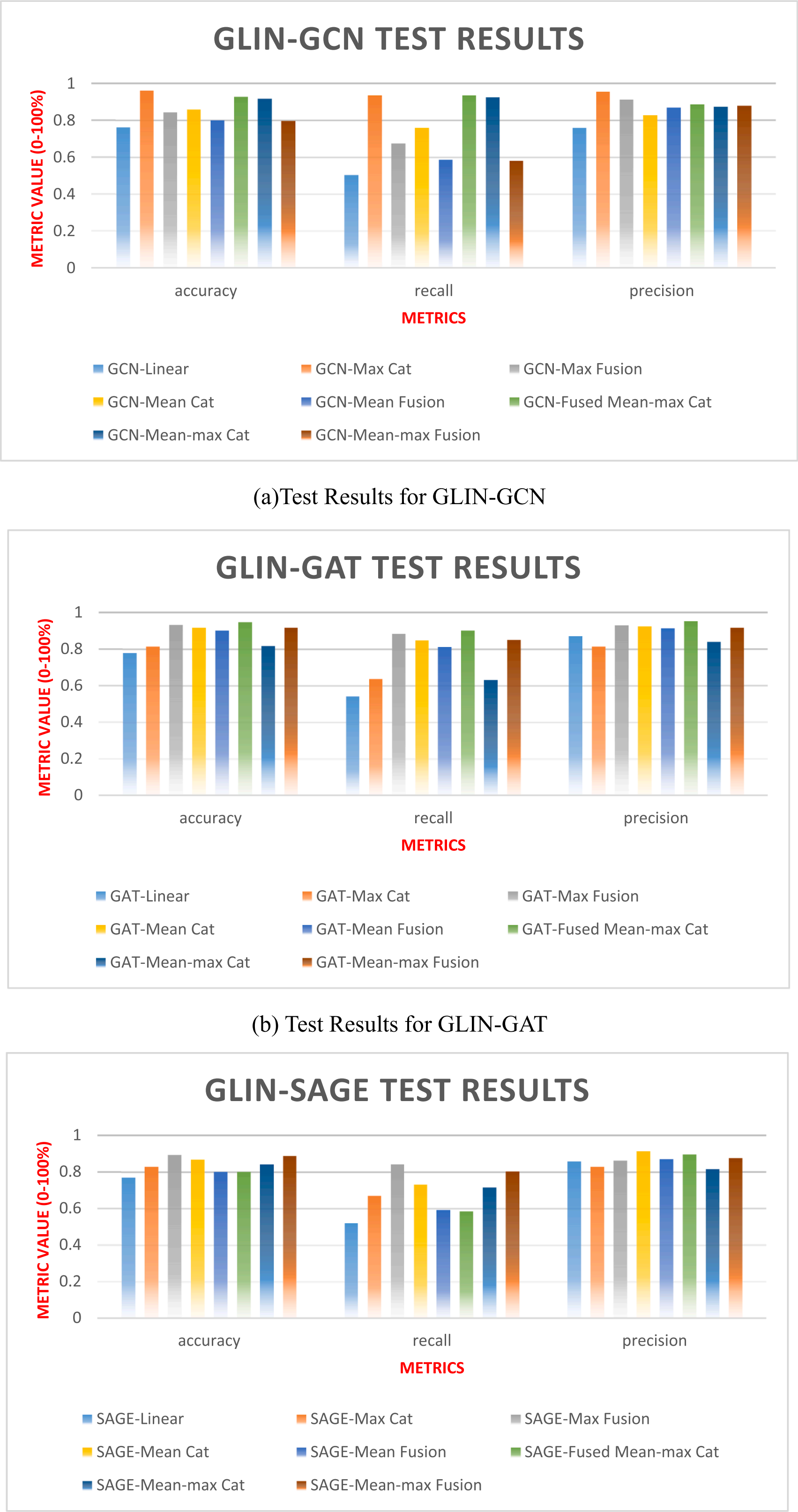
4.4. Performance comparison

We evaluate the performances of 3 types of GLIN framework against the following baselines:

GCN: Neighbours are equally treated during message aggregation, and both the adjacency matrix and the input vectors are normalized.

GAT: Weights are assigned to every neighbour during message aggregation, and their values are trained along with the parameters among the hidden layers.

Graph-SAGE: Nodes are sampled for message passing, and the aggregated neighbour messages are directly concatenated to the self-



(a)Test Results for GLIN-GCN

(b) Test Results for GLIN-GAT

(c) Test Results for GLIN-SAGE

Fig. 5. Global-local integration testing performance results.

Table 3
Baseline results comparison.

Model	Accuracy(%)	Recall(%)	Precision(%)	F1 Score(%)
GLIN-GCN	93.16 | 96.10	93.47 | 93.57	92.76 | 95.60	93.11 | 94.53
GLIN-GAT	91.60 | 94.80	90.66 | 90.19	92.12 | 95.41	91.39 | 92.73
GLIN-SAGE	93.05 | 89.47	92.96 | 84.33	92.78 | 86.19	92.87 | 85.25
PCA-GLIN-GCN	**96.29 | 96.86**	**96.55 | 94.63**	**95.97 | 96.68**	**96.26 | 95.64**
GCN	65.27 | 76.13	59.48 | 50.36	69.55 | 75.87	64.12 | 60.54
GAT	66.01 | 77.95	60.31 | 54.17	70.70 | 87.20	65.09 | 66.83
Graph-SAGE	61.75 | 76.99	54.63 | 52.15	68.59 | 85.84	60.82 | 64.88
CNN-Local	63.10 | 65.71	65.81 | 54.59	66.60 | 54.32	66.20 | 54.45
CNN-Global	88.35 | 91.97	88.35 | 87.19	90.55 | 93.11	89.44 | 90.05
Linear-NN	63.38 | 76.60	63.42 | 51.71	63.07 | 75.44	63.25 | 61.36
IForest	58.65 | 59.74	57.48 | 57.40	57.48 | 55.55	57.48 | 56.46
One-Class SVM	58.21 | 58.14	55.35 | 53.49	55.98 | 52.67	55.66 | 53.08
PCA-NN	64.31 | 76.25	59.83 | 50.60	64.16 | 80.69	61.92 | 62.20
PCA-CNN	62.35 | 76.15	60.11 | 50.38	60.82 | 79.84	60.46 | 61.78
PCA-GCN	69.61 | 81.14	65.52 | 62.91	71.40 | 81.24	68.34 | 70.91

representing vectors.

CNN-Local: 1-dimensional convolutional model with 2 Conv. and 1 Pooling layer. Trained and tested on node-level samples. i.e. Each input sample is a preprocessed vector of length $\boldsymbol{m}$ corresponding to a certain device at a specific time tick.

CNN-Global: The same network configuration apart from the input dimension. Each sample in this case is a fused vector of all applicable devices at a specific time tick. Note that in this situation, metrics are evaluated on the graph level.

Linear-NN: Double-layer fully-connected network.

Isolated Forest: Unsupervised outlier detector. Featured for its random forest generation process and runtime efficiency.

One-Class SVM: Unsupervised outlier detector. Novelty detection discovering rare events. Featured for its ability to cope with extremely unbalanced data.

PCA: Technique for redundancy elimination. Implemented with succeeding NN frameworks.

The evaluation results are exhibited in Table 3:

4.5. Runtime comparison

The training and test runtime is illustrated in Fig. 6. It can be inferred that the additional pooling and integration process produces an extra time consumption of 45.46–91.28% for training, and 1.25–57.14% for testing. Nonetheless, the GLINs still outperform the CNN-Global method which provides the same level of excellence in terms of all the accuracy metrics evaluated in TABLE 3, by reducing the testing runtime by rates of 55.56% (GLIN-GCN), 5.56% (GLIN-GAT) and 38.89% (GLIN-GSAGE). Note that the training and test time consumptions for Isolation Forest and One-Class SVM exhibit a rather prominent scaling range discrepancy in contrast to all other methods evaluated. Specifically, it takes the Isolation Forest 2 min and 35 s to train, and 32.7 s to complete the testing process. As for the One-Class SVM, it takes up to 20 h 27 min and 25 s to train, and testing costs 1 h 49 min and 27 s. The aforementioned results are not included in Fig. 6 due to this scaling difference.

5. Analysis and discussion on experimental results

This section provides some insights on the results shown in Section 4.

5.1. Result interpretation & baseline comparison

The GLIN converts the temporal sequences obtained from all field devices associated with the physical process (water treatment as in the SWaT dataset), and maps them into labels representing the devices' state of operation. That provided, this model, once deployed, is capable of simultaneously differentiating anomalous behavioural patterns with respect to all devices (controllers, sensors, actuators, etc.) in a real-time fashion, given that all it needs to determine a device's state at a specific time tick t is simply a window of historical values ending at t. With security gateways or flow auditing systems performing Deep Packet Inspection on the original packet flow, the required temporal sequences are easily obtained, and the labels produced by the GLIN can be utilized for downstream tasks such as network risk analysis and packet filtering.

Regarding the results summarized in section 4, our observations are highlighted as follows:

The GLINs present a significant gain over their corresponding GNN prototypes (GCN, GAT and Graph-SAGE). The F1 scores, for example, are at least 20.37% superior. This enhancement attributes to the GLINs' global feature incorporation mechanism, which balances each node's overall contribution to the integration of an individual node's vector representation. The integrated embeddings take into account both global and local semantics, featuring a node's uniqueness

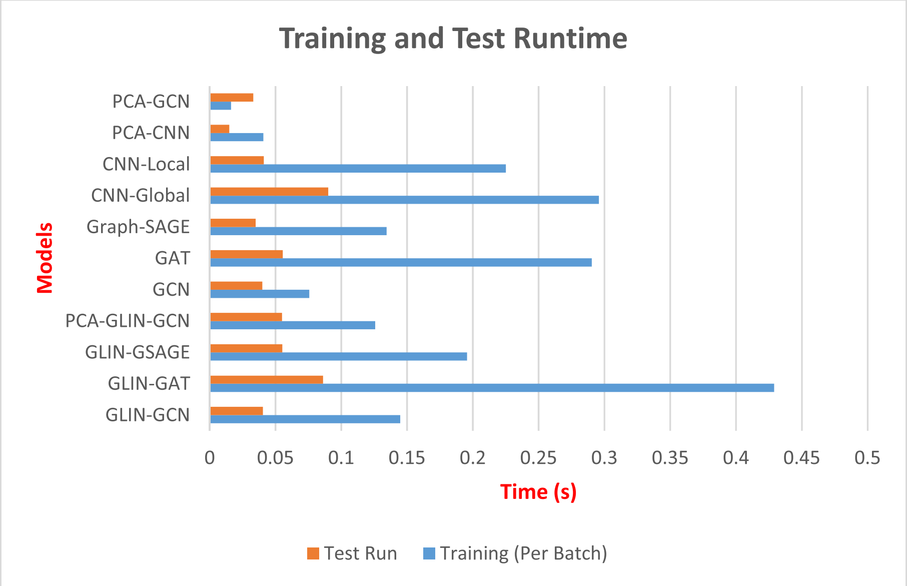


Fig. 6. Training and Test Runtime Evaluation (Note: Test time is appropriately scaled (5 ×) in order to clarify discrepancies among all models).

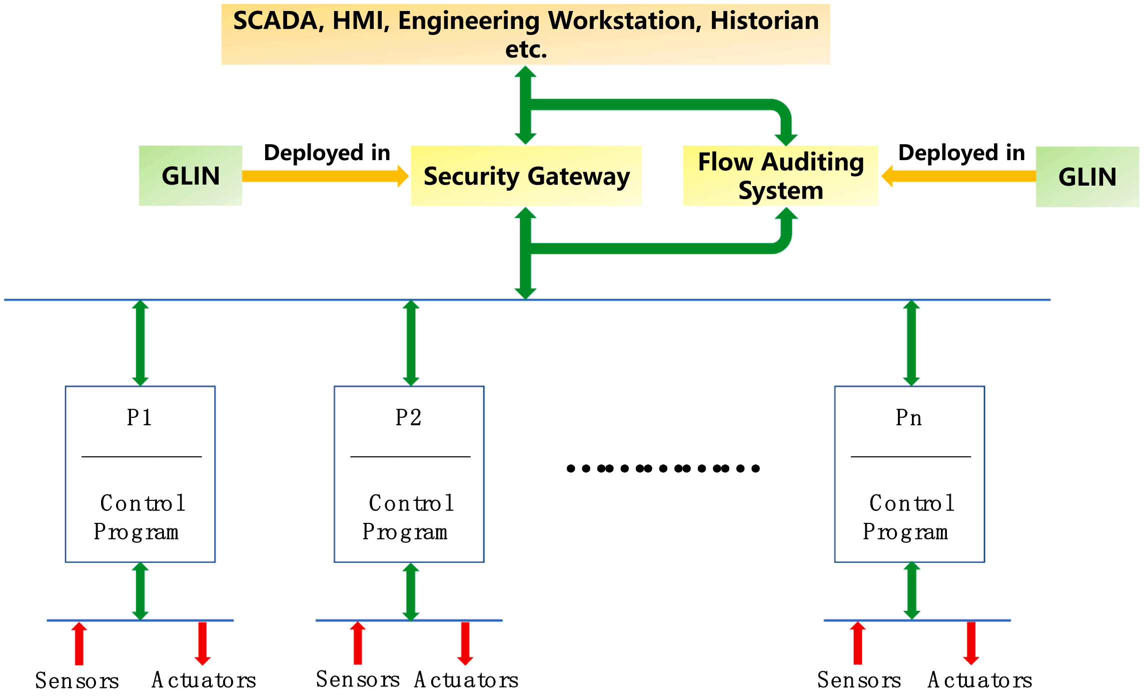


Fig. 7. GLIN Deployment.

and characterizing its sophisticated association with all distant nodes. This prompts the effectiveness of GLIN to perform node-level state prediction for all nodes are knowledgeable enough with respect to both itself and the global setting it resides in. Furthermore, PCA leads to extra performance enhancement since it creates more compact and effective representations via redundancy elimination.

The GLINs' performances excel several other state-of-the-art deep learning based classifiers (CNN, Linear-NN), with an F1 gain of no less than 5.59%. The GLINs are developed on the basis of a general GNN architecture, and thus inherit most, if not all, of the properties of a GNN framework. They are able to utilize in the label prediction process node-level relational properties, which other classifiers such as CNN and Linear-NN usually fail to capture. Note that we evaluate 2 types of CNN models, one that performs label inference with nodes' initial vector representations (CNN-Local). As shown in Table 4, the results are far from perfect compared to the GLIN. This demonstrates the boosting effect of a node's global awareness towards the node-level property prediction. The other model (CNN-Global) fuses all initial vectors within a particular time tick, completes prediction with the fused vectors and ends up with much more desirable results. However, owing to embedding fusion, representations of all nodes corresponding to the same time tick become completely indistinguishable. Therefore, CNN-Global cannot perform label prediction on the same granularity level as the GLINs.

The GLINs tremendously surpass the unsupervised anomaly detectors evaluated in this work (Isolation Forest, One-Class SVM) in terms of all evaluation metrics. These two methods produce hyperspheres to differentiate rare or abnormal cases from the normal ones. They treat each feature vector individually, and hence do not consider the temporal and spatial associativity during the model fitting process as the GLINs do. Also, the Isolation Forest algorithm operates upon existing attributes in the vectors, giving rise to performance deterioration due to a lack of in-depth feature analysis.

5.2. Influence of integrations

The effect of integrations on models' performance varies among combinations of different message aggregation techniques and integration schemes. Overall, the co-existence of the mean and max pooling operations tends to enhance a model's classification accuracy and precision to the greatest extent, while no general patterns are observed in recall. This suggests the positive influence of global message complexity on a model's functionality, which should be elaborated on in future study.

As for individual pooling schemes, GCN benefits greatly more from the max pooling method while GAT coordinates better with mean pooling. This presumably arises from the differences in the adjacency manipulation executed in the original frameworks. For instance, GCN's indiscrimination in message aggregation smoothens the prominent characters each node holds. Thus an complemental stress on these characters, which the max pooling operation offers, may offset the negative smoothing impact attributed to the original GCN. The same principle may apply to GAT as well, while the situation is simply the opposite. GAT absorbs a lot of individuality during message passing via weight assignment, in the meantime impairing the generality a system exhibits. This can be compensated via the application of mean pooling.

The effect of concatenation and fusion on the original models also varies. Concatenation leads to a boost in GCN's performance while fusion ameliorates GAT's detection accuracy. This actually corresponds to the influence of max pooling to a GCN model and the effect of mean pooling to a GAT framework, respectively. Concatenation preserves all relevant properties while fusion suppresses the prominence, supplementing what is lacking attention in the original methodologies.

5.3. Sensitivity against anomalies

Test results yielded from the **Cross-Neg** dataset suggest that the GLIN series are capable of differentiating anomalous activities at a rather early stage, indicating a much prompter response to potentially malicious behaviours than their counterpart approaches. Exemplified with GCN, one can easily infer that the proposed GLIN method excels the regular GCN in every adopted metric by 20–30%. This all-around improvement demonstrates the effectiveness of global integration over peripheral learning, contributing to a higher level of unambiguity in the definition of normal activities.

5.4. Limitations

The integration process attaches the same global message block to the embeddings of nodes in the same batch. This shortens the distance among these representations, in the meantime leading to potential over-smoothing issues, raising difficulties in distinguishing abnormal devices in the same time tick. Max pooling, in particular, directly appends the same global expressions to the end of the original vectors. This drastically expands the vector's dimension as well as the similarity among different nodes to an extent determined by the number of pooling mechanisms involved. Relevant details should be investigated in future work.

5.5. Deployment strategies

With security gateways and flow auditing systems equipped in real ICS networks, the GLIN can be deployed in these security devices and run as a cooperative function in coordination with existing operations (See Fig. 7). Specifically, the GLIN can work as a separate plugin looped by the primary protocol stack whenever necessary, or alternatively, it can be initiated as a continuous thread, serving as part of the main Deep Packet Inspection (DPI) process. In the former scenario, received packets are accumulated and stored before being transmitted to the GLIN upon call, while the latter requires the GLIN to perform analysis on real-time data flow.

6. Conclusion

In this work, we investigate the influence of global expressions on the performance of GNN models dedicated to device anomaly detection in ICSs. We present the GLIN, a framework that achieves node-level anomaly detection via global and local message integration. The GLIN comprises a preprocessor, an encoder, a pooling module, an integration module, and a decoder. The model is unique in that it achieves node-level state inference via global incorporation. We evaluate the GLIN against multiple existing frameworks using various popular metrics and results have proven GLIN's superiority over the current baselines, with an F1 gain of no less than 5.59%. Finally, we present possible application and deployment schemes of the GLIN in real ICSs. Our future work involves investigating how the GLIN should be implemented in diverse ICS environment, and studying the possibility of enhancing its runtime efficiency in order to adapt to real-time applications.

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

This paper uses an opensource dataset named SWaT, which is available on request. Details can be found in https://itrust.sutd.edu.sg.

Acknowledgements

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

References

Abdallah, M., An Le Khac, N., Jahromi, H., & Delia Jurcut, A. (2021, August). A Hybrid CNN-LSTM Based Approach for Anomaly Detection Systems in SDNs. In The 16th International Conference on Availability, Reliability and Security (pp. 1-7).

Ao, Q. (2020). An intrusion detection method for industrial control system against stealthy attack. In *In 2020 7th International Conference on Dependable Systems and Their Applications (DSA)* (pp. 157–161). IEEE.

Asghar, M. R., Hu, Q., & Zeadally, S. (2019). Cybersecurity in industrial control systems: Issues, technologies, and challenges. *Computer Networks, 165*, Article 106946.

Chen, L., Kuang, X., Xu, A., Yang, Y., & Suo, S. (2020). Anomaly Detection on Time-series Logs for Industrial Network. In *In 2020 3rd International Conference on Smart BlockChain (SmartBlock)* (pp. 81–86). IEEE.

Dey, A. (2020). Deep IDS: A deep learning approach for Intrusion detection based on IDS 2018. In *In 2020 2nd International Conference on Sustainable Technologies for Industry 4.0 (STI)* (pp. 1–5). IEEE.

Dhiman, H. S., Deb, D., Muyeen, S. M., & Kamwa, I. (2021). Wind turbine gearbox anomaly detection based on adaptive threshold and twin support vector machines. *IEEE Transactions on Energy Conversion, 36*(4), 3462–3469.

Goldenberg, N., & Wool, A. (2013). Accurate modeling of Modbus/TCP for intrusion detection in SCADA systems. *International Journal of Critical Infrastructure Protection, 6*(2), 63–75.

Hamilton, W., Ying, Z., & Leskovec, J. (2017). Inductive representation learning on large graphs. Advances in neural information processing systems. 30.

Jang, K., Hong, S., Kim, M., Na, J., & Moon, I. (2021). Adversarial autoencoder based feature learning for fault detection in industrial processes. *IEEE Transactions on Industrial Informatics, 18*(2), 827–834.

Kipf, T. N., & Welling, M. (2016a). Variational graph auto-encoders. arXiv preprint arXiv: 1611.07308.

Kipf, T. N., & Welling, M. (2016b). Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907.

Kleinmann, A., Amichay, O., Wool, A., Tenenbaum, D., Bar, O., & Lev, L. (2017). Stealthy deception attacks against SCADA systems. In Computer Security (pp. 93-109). Springer, Cham.

Kleinmann, A., & Wool, A. (2016, October). Automatic construction of statechart-based anomaly detection models for multi-threaded scada via spectral analysis. In Proceedings of the 2nd ACM Workshop on Cyber-Physical Systems Security and Privacy (pp. 1-12).

Markman, C., Wool, A., & Cardenas, A. A. (2017, November). A new burst-DFA model for SCADA anomaly detection. In Proceedings of the 2017 Workshop on Cyber-Physical Systems Security and PrivaCy (pp. 1-12).

Sankar, A., Zhang, X., & Chang, K. C. C. (2019, August). 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 (pp. 137-144).

Sinha, J., & Manollas, M. (2020, June). Efficient deep CNN-BILSTM model for network intrusion detection. In Proceedings of the 2020 3rd International Conference on Artificial Intelligence and Pattern Recognition (pp. 223-231).

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., & Bengio, Y. (2017). Graph attention networks. arXiv preprint arXiv:1710.10903.

Velickovic, P., Fedus, W., Hamilton, W. L., Liò, P., Bengio, Y., & Hjelm, R. D. (2019). *Deep Graph Infomax. ICLR (Poster), 2*(3), 4.

Wang, Y., Zhang, J., Guo, S., Yin, H., Li, C., & Chen, H. (2021, July). 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 (pp. 1239-1248).

Wu, Y., Dai, H. N., & Tang, H. (2021). Graph neural networks for anomaly detection in industrial internet of things. *IEEE Internet of Things Journal.*

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

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

Zhao, T., Deng, C., Yu, K., Jiang, T., Wang, D., & Jiang, M. (2020, October). Error-bounded graph anomaly loss for GNNs. In Proceedings of the 29th ACM International Conference on Information & Knowledge Management (pp. 1873-1882).

Zhao, T., Jiang, T., Shah, N., & Jiang, M. (2021). A synergistic approach for graph anomaly detection with pattern mining and feature learning. IEEE Transactions on Neural Networks and Learning Systems.

Zhou, C., & Paffenroth, R. C. (2017, August). Anomaly detection with robust deep autoencoders. In Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining. (pp. 665-674).