International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 18(17s) 2026 pp. 943-957
https://doi.org/10.70917/ijcisim-2026-4813
Article

Graph Neural Network Models for Multi-Stage Cyber Attack Detection in Industrial Control Systems: A Comparative Evaluation Framework Across SCADA-Grade Datasets

Naga Venkata Aswini Pavan Kumar Inguva

Independent researcher, India, Email: pavankumar3266@gmail.com

Abstract: Industrial Control Systems (ICS) and Supervisory Control and Data Acquisition (SCADA) environments are vulnerable to multi-stage cyber attacks that evolve across interconnected devices, protocols, and physical processes. Detection is difficult because each stage of such an attack, evaluated in isolation, may resemble legitimate operational behaviour, while the malicious intent becomes evident only when the structural relationships among assets and the temporal progression of events are considered together. Traditional signature-based, anomaly-based, and classical machine learning approaches typically evaluate individual packets or short windows and therefore struggle with multi-stage campaigns. This paper proposes a comparative evaluation framework for Graph Neural Network (GNN) models, specifically Graph Convolutional Networks (GCN), Graph Attention Networks (GAT), GraphSAGE, and Graph Isomorphism Networks (GIN), applied to multi-stage attack detection in ICS environments. Industrial network traffic and process interactions are represented as graphs in which nodes correspond to controllers, sensors, actuators, engineering workstations, and network devices, and edges capture communication and control relationships. Attack stages are labelled at the node level so that detection becomes a structured classification problem. The paper does not report experimental results. Implementation and empirical benchmarking are planned as future work on publicly available SCADA-grade datasets including SWaT, WADI, HAI, and additional Power System and industrial datasets where available. The contribution is the proposed evaluation framework, the graph representation strategy for ICS environments, and the methodological groundwork for a rigorous future comparison.

Keywords: Industrial Control Systems; SCADA Security; Operational Technology; Cyber-Physical Systems; Multi-Stage Cyber Attack Detection; Graph Neural Networks; Graph Convolutional Network; Graph Attention Network; GraphSAGE; Graph Isomorphism Network; Intrusion Detection System; Critical Infrastructure Protection

1. INTRODUCTION

Industrial Control Systems (ICS) and Supervisory Control and Data Acquisition (SCADA) environments operate critical infrastructure across power grids, water and wastewater treatment, oil and gas, transportation, and manufacturing. These environments are constructed for operational continuity, physical process safety, and long equipment lifetimes rather than for the confidentiality-centric security posture of enterprise information technology (IT) [1], [2]. They typically use legacy communication protocols such as Modbus, DNP3, and IEC 60870-5-104, which were designed with limited or no built-in security features. Many industrial devices have long operational lifecycles, restricted computing resources, and strict real-time performance requirements, which limit the deployment of resource-intensive security mechanisms. Software patching windows are rare or non-existent because interrupting an industrial process can have direct physical, environmental, or safety consequences [3].

The consequences of a successful cyber attack in this setting are correspondingly severe. Documented incidents have caused equipment damage, service disruption, environmental harm, and threats to human safety [4]. High-profile

campaigns have shown that adversaries capable of reaching ICS assets can move laterally, escalate privileges, manipulate sensor readings, override safety systems, and issue malicious commands to programmable logic controllers (PLCs) [5].

A defining characteristic of these campaigns is that they progress in stages. An attacker typically obtains initial access through compromised remote connections, phishing, vulnerable Internet-facing services, or infected engineering workstations. After establishing persistence, the attacker performs reconnaissance to identify controllers, sensors, engineering stations, human-machine interfaces (HMIs), historians, and network topology. Lateral movement across interconnected operational-technology (OT) assets and escalation of privileges follow. In the final stages, malicious commands are issued to PLCs, sensor values are manipulated, safety interlocks are bypassed, or industrial processes are directly disrupted.

Detecting each stage independently is often insufficient. Individual activities such as authentication events, engineering workstation access, or configuration changes may closely resemble legitimate operational behaviour and are routinely permitted by baseline monitoring rules. The malicious pattern emerges only when the events are analysed collectively as an evolving sequence of interconnected actions across multiple assets. Consequently, effective detection requires models that can capture both the structural relationships among industrial assets and the temporal progression of attack stages.

Graph Neural Networks (GNNs) are a natural structural fit for this class of problem. An ICS environment can be modelled as a graph in which nodes represent controllers, PLCs, remote terminal units (RTUs), sensors, actuators, HMIs, engineering workstations, and network devices, and edges capture communication paths, control relationships, protocol interactions, or physical process dependencies. Message passing within a GNN allows each node to learn from its connected neighbours, so the model can capture both local interactions and multi-hop attack propagation. This is a fundamentally different inductive bias from tabular models that treat each event independently or sequence models that emphasise temporal order without explicit structural representation.

This paper proposes a comparative evaluation framework for four widely used GNN architectures — GCN, GAT, GraphSAGE, and GIN — applied to multi-stage attack detection in ICS environments. The framework defines a graph representation for industrial networks, an attack-labelling scheme at the attack-stage level, an evaluation methodology across public SCADA-grade datasets, and a set of baseline comparisons.

Contributions. The specific contributions of this paper are:

1. A proposed comparative evaluation framework for GNN models applied to multi-stage attack detection in ICS environments.

2. A proposed graph representation of ICS networks in which nodes are host-service pairs and edges are observed communication and control relationships, with node and edge features encoding protocol, timing, and process context.

3. A proposed attack-stage labelling scheme that turns multi-stage attack detection into a structured node classification problem rather than a per-flow binary classification.

4. A planned experimental methodology, including datasets, preprocessing, baselines, and metrics, that will enable a rigorous and reproducible comparison of GCN, GAT, GraphSAGE, and GIN under identical conditions.

The paper does not claim experimental results. Implementation and comparative empirical evaluation are identified as future work.

Paper organisation. Section 2 reviews prior work in ICS attack detection and in GNNs applied to cybersecurity. Section 3 defines the multi-stage attack model. Section 4 presents the proposed comparative evaluation framework. Section 5 describes the datasets under consideration. Section 6 defines the planned experimental design. Section 7 discusses deployment considerations for OT environments. Section 8 states the limitations and open challenges. Section 9 outlines future research directions. Section 10 concludes.

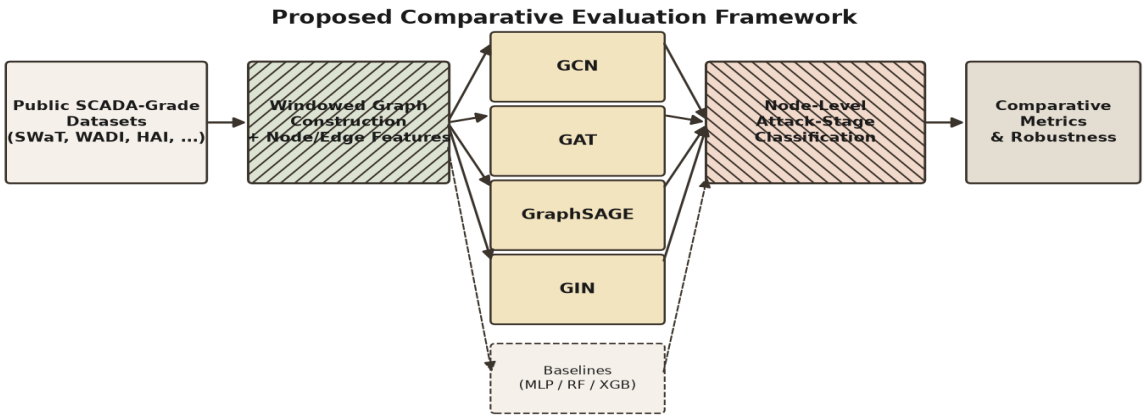

Fig. 1. Proposed comparative evaluation framework. Public SCADA-grade datasets are transformed into windowed graphs and evaluated across four GNN architectures under identical preprocessing, training, and evaluation conditions.

2. Related Work

Table I. ICS / SCADA environments versus enterprise IT: key contrasts relevant to detection

Dimension	**Enterprise IT**	**ICS / SCADA**
Primary security objective	Confidentiality, integrity, availability	Availability and physical process safety first
Protocols	Modern TCP/IP application stacks	Legacy: Modbus, DNP3, IEC 60870-5-104, S7
Patching cadence	Regular; scheduled maintenance windows	Rare; equipment often runs unpatched for years
Impact of a successful attack	Data loss, service disruption, reputational harm	Equipment damage, environmental release, safety-of-life risk
Device compute budget	Ample; supports rich agents	Limited; strict real-time requirements
Detection unit that makes sense	Per-flow / per-endpoint event	Multi-asset subgraph across a temporal window

2.1 Rule-Based and Signature-Based ICS Intrusion Detection

Rule-based intrusion detection systems have been widely deployed in industrial environments, typically using ICS-aware rule sets that match predefined signatures, protocol violations, or known indicators of compromise [6]. These systems are effective against previously observed attacks with low false-positive rates when rules are current, and they remain operationally important in mature OT security programmes. They are known to be less effective against novel or evolving threats and against attackers whose behaviour deviates from previously encoded signatures. Rule maintenance for OT environments is particularly difficult because rule authors must understand both the network protocol and the industrial process context.

2.2 Anomaly Detection on Process and Network Data

Anomaly-based detection monitors process variables, sensor readings, actuator states, and network traffic to identify deviations from baseline operational behaviour [7], [8]. Approaches include statistical process control, one-class classifiers, autoencoders trained to reconstruct normal behaviour, and predictive models that flag residuals above

a threshold. These methods can detect previously unseen attacks and are attractive because they do not require labelled attack data. Their principal weakness is false-positive rate. Legitimate operational changes — a maintenance window, a process transition, a seasonal load pattern — can generate large deviations that resemble malicious activity.

2.3 Classical Machine Learning for ICS Intrusion Detection

Classical machine learning algorithms including Random Forests, Support Vector Machines, Decision Trees, XGBoost, and shallow neural networks have been extensively applied to ICS intrusion detection using engineered statistical or protocol-specific features [9], [10]. These approaches generally outperform pure signature matching on unseen attacks and can leverage domain-specific flow features. Their limitations mirror those of enterprise IT: performance depends heavily on feature engineering, and the models struggle to capture complex temporal and structural relationships that are characteristic of multi-stage industrial attacks.

2.4 Deep Learning for ICS Security

Deep learning has been proposed to address feature-engineering limits and to capture more complex behavioural patterns in industrial environments. CNNs have been applied to spatial representations of process data; recurrent networks including LSTM and GRU have been used to capture temporal dependencies; autoencoders have been used for unsupervised anomaly detection on sensor streams; and Transformer-based models have been explored more recently [11], [12]. Reported detection performance on public benchmarks is frequently favourable. However, most of this work treats each event, each packet, or each short window independently, and does not natively model the structural relationships among interconnected industrial assets that make multi-stage attacks distinctive.

2.5 Graph Neural Networks in Cybersecurity

Graph Neural Networks have been introduced as a class of models that operate directly on graph-structured data. Kipf and Welling proposed Graph Convolutional Networks (GCN), a spectral formulation that generalises convolution to arbitrary graphs [13]. Velickovic et al. proposed Graph Attention Networks (GAT), which use self-attention to weight neighbour contributions [14]. Hamilton et al. proposed GraphSAGE, an inductive framework that generates node embeddings by sampling and aggregating features from a node's local neighbourhood, enabling generalisation to unseen nodes [15]. Xu et al. proposed the Graph Isomorphism Network (GIN), which is theoretically expressive up to the Weisfeiler-Lehman isomorphism test [16].

GNNs have been applied to cybersecurity tasks including network intrusion detection [17], malware analysis [18], and lateral movement detection in enterprise networks [19]. Their application to ICS and SCADA environments specifically is a smaller but growing literature [20], [21]. This paper positions itself against that body of work by proposing a comparative evaluation framework rather than a new GNN architecture, and by explicitly focusing on multi-stage attack detection through node-level attack-stage labelling.

3. Multi-Stage Attack Model in ICS Environments

A multi-stage cyber attack in an industrial control environment is a sequence of coordinated actions that collectively achieve the attacker's objective. For the purpose of framework design, the paper adopts the following stage decomposition drawn from public frameworks such as the ICS-adapted variants of the intrusion kill chain and MITRE ATT&CK for ICS [22]:

1. Initial access. The attacker obtains an initial foothold, typically through a compromised remote-access channel, a phishing campaign against engineering staff, a vulnerable Internet-facing service, or an infected engineering workstation.

2. Persistence and reconnaissance. The attacker establishes persistence, identifies the assets that comprise the target environment, and maps controllers, sensors, engineering stations, HMIs, historians, and network topology.

3. Lateral movement. The attacker moves across interconnected OT assets, typically crossing network segments and elevating privileges.

4. Command and control. The attacker establishes covert channels to receive instructions and to exfiltrate reconnaissance data.

5. Execution and impact. The attacker issues malicious commands to PLCs, manipulates sensor values, bypasses safety systems, or otherwise disrupts industrial processes with potential physical consequences.

Each stage produces network and process artefacts. However, in isolation many of those artefacts closely resemble legitimate activity. An engineering workstation reading configuration from a PLC is normal during maintenance windows. An HMI issuing a set-point change is normal during scheduled process transitions. Only the collective pattern — a sequence of such events across previously unrelated assets, correlated in time, along paths that legitimate operations do not typically follow — reveals the attack.

The framework design therefore assumes that the detection unit is not a single event but a subgraph of assets and their communications over a rolling temporal window, and that the labelling target is the attack-stage of each node inside that subgraph.

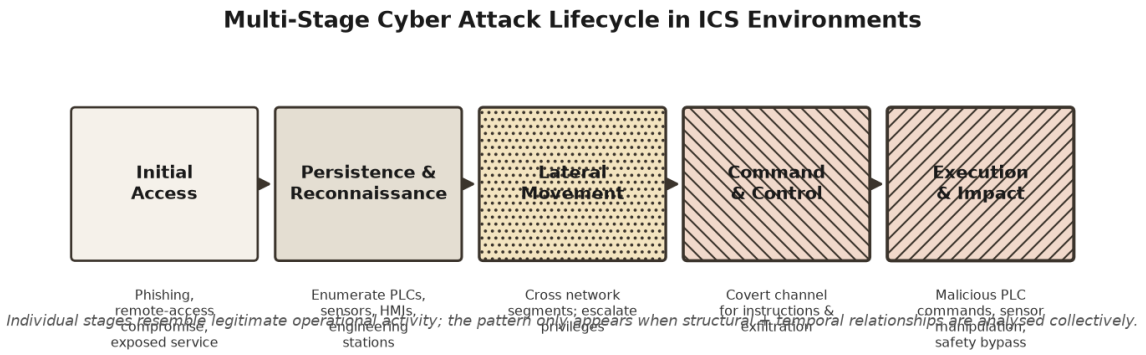

Fig. 2. Multi-stage cyber attack lifecycle in Industrial Control Systems. Detection at each stage in isolation is difficult because individual activities may resemble legitimate operational behaviour.

Table II. Multi-stage attack model in ICS: stages, examples, and detection challenge

Stage	**Typical activity**	**Why isolated detection is hard**
1. Initial access	Phishing, compromised remote access, exposed engineering-workstation service	Individual login / access events resemble legitimate remote maintenance
2. Persistence & reconnaissance	Enumerate PLCs, sensors, HMIs; identify network topology	Scans blend into engineering-workstation traffic and inventory tools
3. Lateral movement	Cross network segments; escalate privileges	Cross-segment traffic can appear on legitimate operational paths
4. Command & control	Establish covert channel for instructions / exfiltration	Long-lived low-volume connections can look like normal telemetry
5. Execution & impact	Malicious PLC writes, sensor manipulation, safety-system bypass	Set-point changes may match legitimate process transitions

4. Proposed Comparative Evaluation Framework

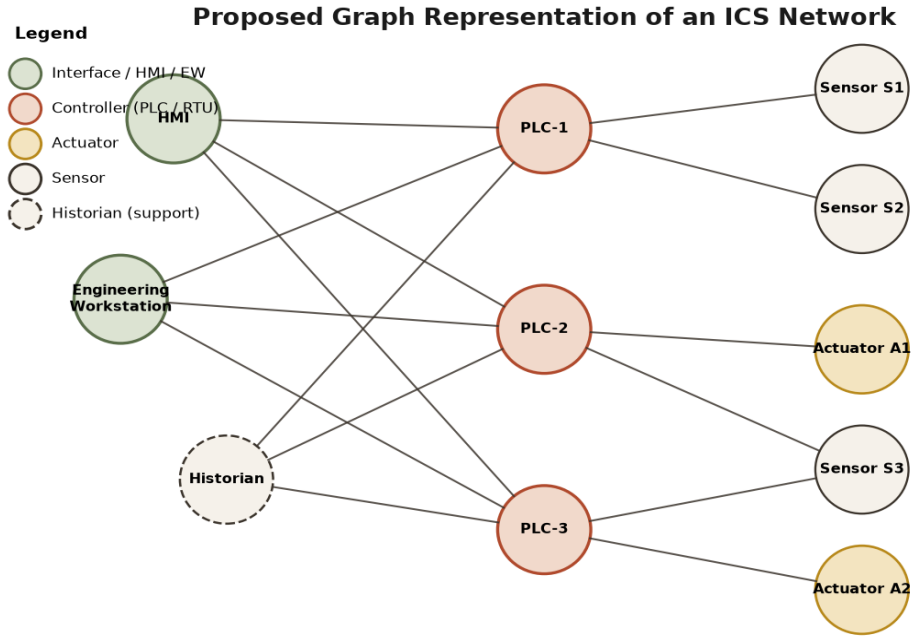

Fig. 3. Proposed graph representation of an ICS network. Nodes correspond to host-service pairs (HMIs, engineering workstations, PLCs, historians, sensors, actuators); edges correspond to observed communication and control relationships within a rolling window.

Table III. Node and edge feature specification for the proposed graph representation

Element	**Feature**	**Description**
Node	Host identity (int)	Stable identifier per host-service pair
Node	Service / protocol identifier	Encoded categorical: Modbus, DNP3, IEC-104, HTTP, ...
Node	Device-role metadata	PLC / RTU / HMI / historian / sensor / actuator (where known)
Edge	Flow count	Sessions observed between the node pair in the window
Edge	Byte count	Aggregate byte volume in the window
Edge	Average packet size	Mean packet size across the session
Edge	Inter-arrival timing statistics	Mean and variance of packet inter-arrival time
Edge	Protocol-specific fields	e.g., Modbus function-code frequencies where captured
Edge	Direction	Initiator-responder relationship

4.1 Framework Objective

The framework's objective is to enable a rigorous and reproducible comparison of four representative GNN architectures — GCN, GAT, GraphSAGE, and GIN — for the task of multi-stage attack detection in ICS environments. The comparison is intended to identify which architectures are most effective under which conditions and to establish a baseline against which future GNN-based ICS detection work can be measured.

4.2 Graph Representation of ICS Networks

The proposed representation constructs, for each rolling temporal window of fixed duration, a directed communication graph G = (V, E) in which:

• Nodes *V* correspond to host-service pairs observed in the industrial network. A single physical device such as a PLC may contribute multiple nodes if it exposes multiple services (for example, a Modbus service and an engineering configuration service). Nodes also cover HMIs, historians, engineering workstations, RTUs, sensors and actuators where they are directly observable on the network, and any network devices whose communications are captured by the monitoring infrastructure.

• Edges *E* correspond to observed session-level communications between node pairs within the window. Edge direction reflects the initiator-responder relationship. Repeated communications between the same node pair within the window are aggregated at the edge level with count and volume features.

• Node features encode host identity (assigned an integer identifier consistent across windows), service or protocol identifier, and any device-role metadata available from the asset inventory.

• Edge features encode protocol, flow count, byte count, average packet size, inter-arrival timing statistics, and any protocol-specific fields captured in the dataset such as Modbus function code frequencies.

The window duration is a design parameter to be tuned during empirical validation. A short window captures fast-moving events but risks fragmenting a multi-stage campaign across multiple graphs. A long window captures more of the campaign but reduces temporal resolution. The framework does not commit to a single window size; it commits to the design principle that the graph is reconstructed periodically over rolling windows.

4.3 Attack-Stage Labelling

The framework proposes labelling attacks at the attack-stage level and at the node level within each window. That is, if an attack event is present in the ground-truth labels of the underlying dataset, the involved nodes for that window carry an attack-stage label drawn from the stage decomposition in Section 3. Nodes not involved in any attack activity in that window carry a benign label.

This turns detection into a node classification task in which the model must predict the attack stage of each node in the graph. Two properties follow. First, per-stage recall and precision can be measured, which is more diagnostic than a single binary detection metric. Second, detection latency for a campaign can be measured as the time between the first attack-related event and the first correctly labelled attack-stage node in the model output.

Where the underlying dataset provides only binary attack labels, the framework proposes to derive attack-stage labels by rule-based mapping from the documented attack scenario. Where the dataset provides only flow-level attack labels, node labels are propagated from the flows that touch each node. This preprocessing step is documented explicitly per dataset so that the labelling procedure is reproducible.

4.4 GNN Architectures Compared

Four representative GNN architectures are proposed for comparison. Each is briefly summarised here in terms of its inductive bias; empirical hyperparameters are deferred to future implementation.

Graph Convolutional Network (GCN) [13]. A GCN aggregates neighbour features by a normalised sum with fixed weights derived from the graph Laplacian. It is efficient and well studied but assumes equal contribution from all neighbours after normalisation.

Graph Attention Network (GAT) [14]. A GAT uses learned attention weights to combine neighbour features. Different neighbours can contribute differently, which is useful when some communication partners are more informative than others for the attack-stage decision.

GraphSAGE [15]. An inductive framework that samples and aggregates features from each node's local neighbourhood, and that generalises to nodes not seen at training time. This inductive property is important for OT environments where new assets are added and where the model must generalise to previously unseen nodes without full retraining.

Graph Isomorphism Network (GIN) [16]. A GIN uses a multi-layer perceptron aggregator that is theoretically as expressive as the Weisfeiler-Lehman graph isomorphism test. This gives it stronger structural discrimination than GCN in principle, which may or may not translate into empirical benefit on ICS graphs.

Each architecture is implemented under identical preprocessing, identical graph construction, identical labelling, identical training partitions, and identical evaluation, so that any observed performance differences are attributable to the model family rather than to preprocessing or evaluation choices.

Table IV. Graph Neural Network architectures compared under the proposed framework

Architecture	**Aggregation mechanism**	**Distinctive property**	**Trade-off**
GCN [Kipf & Welling, 2017]	Normalised weighted sum of neighbour features	Simple, efficient, well-studied baseline	Assumes equal contribution from neighbours after normalisation
GAT [Velickovic et al., 2018]	Learned attention weights over neighbours	Different neighbours can contribute differently	Higher compute; attention weights require tuning
GraphSAGE [Hamilton et al., 2017]	Sample + aggregate features from local neighbourhood	Inductive; generalises to nodes unseen at training	Sampling introduces variance; batch design matters
GIN [Xu et al., 2019]	Sum aggregator + MLP update	Provably expressive up to Weisfeiler-Lehman test	Theoretical expressiveness does not always convert to empirical gain

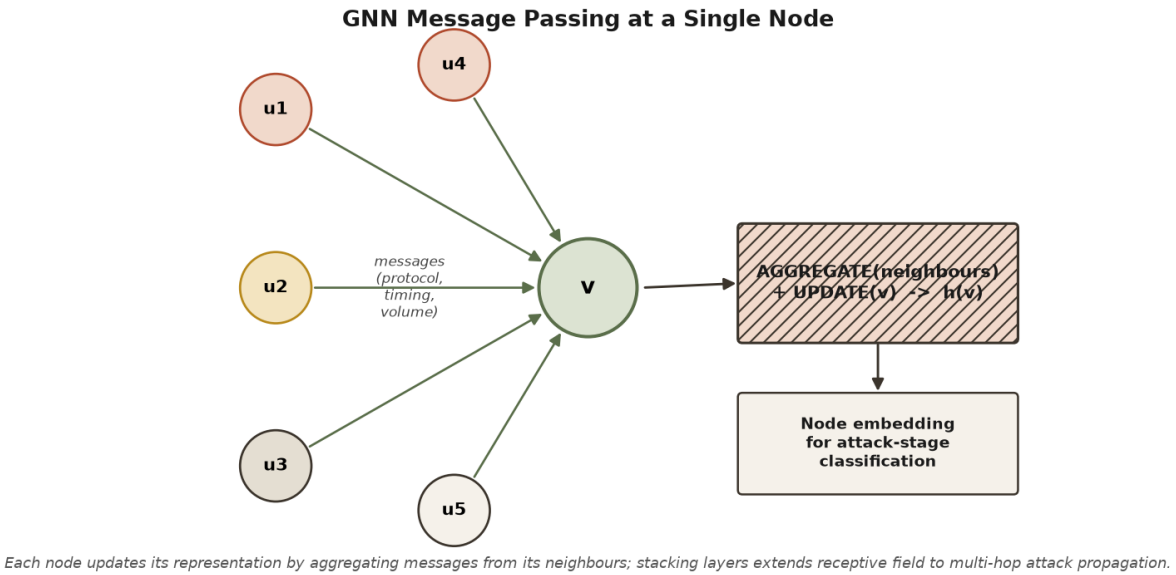

Fig. 4. GNN message passing at a single node. Each node updates its representation by aggregating features from its neighbours; stacking layers extends the receptive field to multi-hop attack propagation.

4.5 Baseline Models

Two categories of baselines are proposed. Non-graph deep learning baselines include a standalone MLP over the aggregated flow-feature representation, a CNN, and an LSTM over the sequence of events in each window. These baselines allow the framework to isolate the benefit contributed specifically by the graph structural representation. Classical machine learning baselines include Random Forest and XGBoost over engineered flow features. These baselines establish the level of performance obtainable without any deep or graph representation.

5. Datasets Under Consideration

Four publicly available SCADA-grade datasets are proposed for the future comparative evaluation. Each dataset is selected because it is widely used in the ICS security literature, because it captures realistic industrial process and network data, and because the attack scenarios it labels are documented in sufficient detail to support attack-stage labelling.

Table V. Public SCADA-grade datasets planned for the comparative study

Dataset	**Testbed / origin**	**Documented attack coverage**	**Notes**
SWaT	Water treatment testbed (iTrust, SUTD)	Sensor-value manipulation, malicious PLC commands, multi-scenario	Canonical benchmark in ICS security research
WADI	Water distribution testbed (iTrust)	Multi-scenario attacks; documented	Larger than SWaT; useful for graph-size scaling
HAI	Hardware-in-the-loop augmented ICS (NSR, Korea)	Cross-subsystem attack scenarios	Well-suited to multi-stage evaluation

Power system + additional	Mississippi State University collection and similar	Various power-system disturbance and attack cases	Considered where public and adequately labelled

5.1 Secure Water Treatment (SWaT)

The Secure Water Treatment (SWaT) dataset was produced by the iTrust Centre at the Singapore University of Technology and Design [23]. It captures data from a water treatment testbed that includes six process stages and multiple PLCs, sensors, and actuators. The dataset covers approximately eleven days of continuous operation, of which seven days are normal and four days include a documented set of attacks. Attacks include manipulation of sensor readings, injection of malicious commands to PLCs, and other cyber-physical scenarios. SWaT is widely considered a canonical benchmark for ICS security research.

5.2 Water Distribution (WADI)

The WADI dataset was produced by the same iTrust group and captures a water distribution testbed larger than SWaT [24]. It provides sensor and actuator data as well as network traffic and includes documented attack scenarios. Its larger scale relative to SWaT is useful for assessing how GNN models perform as the graph size grows.

5.3 HAI Dataset

The HAI (Hardware-In-the-Loop-based Augmented ICS) dataset was produced by the National Security Research Institute of the Republic of Korea and captures a testbed based on hardware-in-the-loop simulation combining power generation, pumped-storage hydropower, and boiler control processes [25]. It includes documented attack scenarios that span multiple sub-systems and is well suited to multi-stage evaluation.

5.4 Additional SCADA and Power System Datasets

Additional publicly available SCADA datasets are considered where they satisfy the requirements of documented attack scenarios and sufficient labelling detail. These include Power System datasets from the Mississippi State University collection and any additional industrial datasets that become available during the empirical study.

5.5 Enterprise Industrial Traffic

Where organisational approval and data governance permit, the framework may be evaluated on anonymised enterprise industrial network traffic collected from operational environments. At present, no proprietary industrial dataset has been obtained or analysed for this study.

6. Planned Experimental Design

Because the framework is at the design stage, this section defines the planned experimental methodology so that future implementation is reproducible and directly comparable across datasets and architectures.

Table VI. Planned baseline models against which each GNN architecture is compared

Baseline	**Type**	**Role in the comparison**
Multi-Layer Perceptron	Non-graph deep learning	Isolates the benefit of graph structure vs pure feature learning
1-D CNN	Non-graph deep learning	Represents temporal-local pattern learning without graph inductive bias
LSTM	Non-graph deep learning	Represents temporal-sequence learning without graph inductive bias

Random Forest	Classical ML	Non-deep baseline over engineered flow features
XGBoost	Classical ML (gradient boosting)	Strong classical baseline over engineered flow features

Table VII. Planned evaluation metrics for the comparative GNN study

Category	**Metric**	**Purpose**
Detection	Per-attack-stage precision / recall / F1	Diagnostic per stage rather than one binary label
Detection	Macro-averaged precision / recall / F1	Overall performance across attack stages
Detection	False-positive rate on benign class	Operational cost proxy
Detection	ROC-AUC (where applicable)	Threshold-independent separability
Latency	Detection latency per attack	Time from first attack event to first correct attack-stage node output
Robustness	Performance under node-dropout	Simulates device outage / partial visibility
Robustness	Performance under edge-dropout	Simulates missing communication observation
Transfer	Cross-dataset generalisation	Train on one testbed, evaluate on another
Compute	Training time, inference time, memory footprint	Deployment feasibility

6.1 Preprocessing

For each dataset, preprocessing comprises the removal of incomplete or duplicate records, imputation of missing values where appropriate, categorical encoding of protocol and service identifiers, numerical normalisation, and temporal partitioning into training, validation, and testing subsets using a reproducible seeded protocol. Class-imbalance handling is applied at training time only, using either resampling or cost-sensitive loss functions, and its choice is documented per experiment.

6.2 Graph Construction

Graphs are constructed per rolling window as described in Section 4.2. The window duration is a hyperparameter tuned on the validation partition. The graph construction pipeline is implemented once and reused across all four GNN architectures and all four datasets so that comparisons are conducted under identical structural preprocessing.

6.3 Training Protocol

Each GNN architecture is trained under an identical training protocol using the same optimiser, the same maximum number of epochs, and the same early-stopping criterion applied to validation loss. Architecture-specific hyperparameters, including the number of layers, hidden dimension, and (for GAT) the number of attention heads, are tuned on the validation partition using a bounded grid search. Baseline models are trained under equivalent protocols adapted to their architecture.

6.4 Evaluation Metrics

The planned evaluation reports the following metrics for each combination of GNN architecture and dataset:

Detection performance. Per-attack-stage precision, recall, and F1-score; overall macro-averaged precision, recall, and F1-score; false-positive rate on the benign class; and ROC-AUC where applicable.

Detection latency. For each attack instance, the time between the first attack-related event and the first correctly labelled attack-stage node in the model output. This is the operational latency that matters in a critical-infrastructure setting.

Robustness. Performance under node-dropout, in which a random subset of nodes is removed from the graph at inference time to simulate device outage or partial visibility. Performance under edge-dropout is measured analogously.

Cross-dataset transfer. Train on one dataset and evaluate on another. This measures the extent to which a GNN model trained on one industrial testbed generalises to a different testbed.

Computational cost. Training time, inference time per graph, and memory footprint at inference.

6.5 Statistical Reporting

Each experimental configuration is repeated with multiple random seeds and reported with mean and standard deviation. Statistical significance of pairwise differences between GNN architectures on the same dataset is reported using appropriate non-parametric tests. Where a difference is not statistically significant, the paper will say so explicitly.

7. Deployment Considerations for OT Environments

Deployment of a GNN-based multi-stage attack detection system in an operational ICS environment must respect several constraints that distinguish OT from enterprise IT.

Availability first. Detection infrastructure must not interfere with the industrial process. A detection component that produces false-positive commands into the control network is worse than no detection at all. The framework is therefore designed to operate in a passive monitoring mode, receiving mirrored traffic and process telemetry rather than being placed in-line.

Hardware constraints. Many OT environments cannot host GPU-accelerated inference near the process. The framework accommodates this by allowing inference to be executed on a dedicated server placed near the industrial network with either CPU-only or GPU-accelerated configuration, depending on graph size and refresh cadence. Exact requirements will be established during empirical validation.

Protocol coverage. The framework relies on being able to parse industrial protocols including Modbus, DNP3, and IEC 60870-5-104 well enough to extract per-flow features. This requires either the use of an existing OT-aware sensor or the addition of a lightweight parsing layer. Neither is a research contribution of the framework; both are engineering prerequisites for deployment.

Analyst workflow integration. Alerts produced by the framework are intended to integrate into whichever monitoring platform the OT organisation already uses. The alert format includes the predicted attack-stage label per node, a confidence score, the subgraph in which the attack was detected, and enough context for an OT analyst to correlate with process alarms and to initiate the appropriate response.

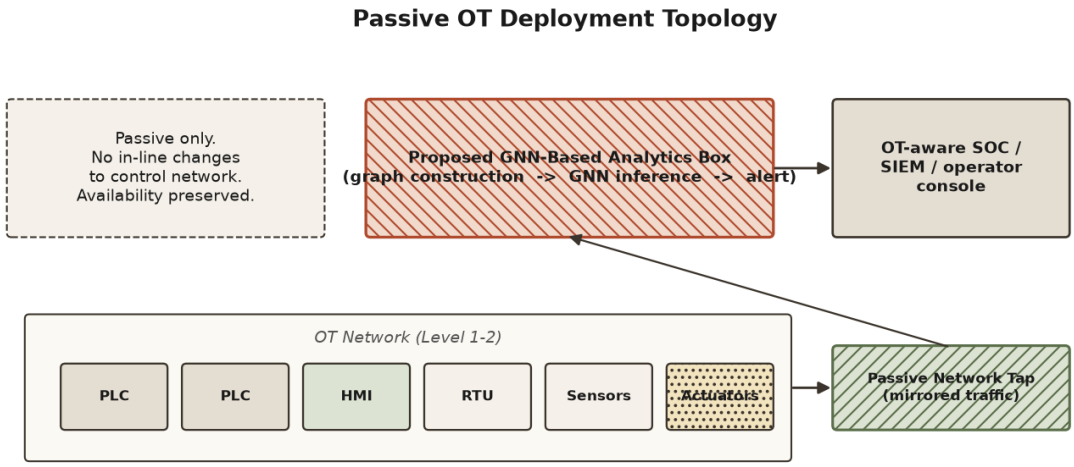

Fig. 5. Passive OT deployment topology. The analytics component consumes mirrored traffic from a passive network tap; it is not placed in-line with the control network.

8. Limitations and Open Challenges

The framework, and any empirical work that follows from it, must acknowledge several limitations.

Dataset representativeness. Public SCADA benchmark datasets, including SWaT, WADI, and HAI, are produced from testbeds. They are the best datasets available for reproducible research, but they do not fully capture the diversity, scale, and long-tail behaviour of operational industrial environments. Cross-testbed transfer measurements partially address this concern but do not eliminate it.

Concept drift. Industrial process behaviour evolves over time as equipment ages, as maintenance schedules change, as the mix of connected assets grows, and as operator practice changes. A model trained on historical traffic will lose accuracy. Drift detection, periodic retraining, and version-controlled model validation are needed as operational practice.

Adversarial evasion. Adversaries may deliberately shape traffic to appear benign to a GNN-based detector. Adversarial robustness of GNNs is an active research area and must be evaluated explicitly. The framework does not claim adversarial robustness at the outset; it defines a robustness evaluation protocol in Section 6.4.

Zero-day and open-set attacks. A supervised classifier may recognise unusual patterns but is likely to assign an unseen attack technique to the closest known attack stage. Open-set recognition and uncertainty estimation are complementary directions.

Graph construction is a design choice. The choice of what a node is, what an edge is, and what features they carry is a design choice that shapes every downstream result. The framework fixes this choice at the specification level and applies it consistently across all four GNN architectures and all four datasets, but a different graph representation is an entirely valid alternative research direction.

Attack-stage labelling depends on ground truth. Where the underlying dataset does not directly provide attack-stage labels, the mapping to attack stages is a labelling step whose accuracy affects every downstream measurement. The framework requires that this labelling step be documented per dataset.

Computational cost at scale. GNN inference cost grows with graph size and connectivity. A very large industrial environment with high inter-asset communication density may require sampling-based inference (as in GraphSAGE)

rather than full-graph inference. The framework anticipates this and includes computational-cost measurements in the evaluation.

9. Future Research Directions

The immediate next step is to implement the proposed framework and to conduct the planned comparative evaluation across the four GNN architectures and the four public SCADA-grade datasets. Beyond that initial empirical study, the following directions are identified:

1. Extension to temporal GNN variants that model the evolution of the industrial graph across windows explicitly, rather than reconstructing the graph independently per window.

2. Evaluation on cross-testbed transfer scenarios in which a model trained on one industrial process is evaluated on a different process, to quantify how far GNN-learned representations generalise.

3. Integration with anomaly-based process-variable detectors, so that structural network-side detection and behavioural process-side detection can be fused.

4. Development of adversarial evaluation protocols specifically for GNN-based ICS detectors, including graph-structure perturbations and feature perturbations that respect industrial process constraints.

5. Investigation of open-set recognition and uncertainty estimation on the attack-stage label to reduce misclassification of previously unseen attack techniques.

6. Prototype integration with an OT-aware monitoring platform to measure end-to-end operational latency, analyst-facing throughput, and correct-escalation rate on realistic response workflows.

10. Conclusion

This paper has proposed a comparative evaluation framework for Graph Neural Network models applied to multi-stage cyber attack detection in Industrial Control Systems. The framework defines a graph representation of ICS networks in which nodes are host-service pairs and edges are observed communication and control relationships, a node-level attack-stage labelling scheme that turns detection into a structured classification problem, and a comparative evaluation methodology for GCN, GAT, GraphSAGE, and GIN under identical preprocessing, training, and evaluation conditions. Four publicly available SCADA-grade datasets, including SWaT, WADI, and HAI, are identified for the planned empirical study. The paper does not report experimental results; implementation and comparative empirical evaluation are identified as future work. The contribution is methodological: a defined, reproducible framework against which future GNN-based ICS detection work can be developed and measured.

REFERENCES

1. R. Langner, "Stuxnet: Dissecting a cyberwarfare weapon," IEEE Security & Privacy, vol. 9, no. 3, pp. 49–51, 2011.
2. E. D. Knapp and J. T. Langill, Industrial Network Security: Securing Critical Infrastructure Networks for Smart Grid, SCADA, and Other Industrial Control Systems, 2nd ed. Waltham, MA, USA: Syngress, 2014.
3. K. Stouffer, V. Pillitteri, S. Lightman, M. Abrams, and A. Hahn, "Guide to Industrial Control Systems (ICS) Security," NIST Special Publication 800-82 Rev. 2, 2015.
4. R. M. Lee, M. J. Assante, and T. Conway, "Analysis of the cyber attack on the Ukrainian power grid," Electricity Information Sharing and Analysis Center (E-ISAC), Tech. Rep., 2016.
5. A. Nicholson, S. Webber, S. Dyer, T. Patel, and H. Janicke, "SCADA security in the light of cyber-warfare," Computers & Security, vol. 31, no. 4, pp. 418–436, 2012.
6. S. Cheung, B. Dutertre, M. Fong, U. Lindqvist, K. Skinner, and A. Valdes, "Using model-based intrusion detection for SCADA networks," in Proc. SCADA Security Scientific Symp., 2007.
7. A. Carcano, A. Coletta, M. Guglielmi, M. Masera, I. N. Fovino, and A. Trombetta, "A multidimensional critical state analysis for detecting intrusions in SCADA systems," IEEE Trans. Industrial Informatics, vol. 7, no. 2, pp. 179–186, 2011.
8. J. Goh, S. Adepu, M. Tan, and Z. S. Lee, "Anomaly detection in cyber physical systems using recurrent neural networks," in Proc. IEEE Int. Symp. High Assurance Systems Engineering (HASE), 2017, pp. 140–145.

9. M. Kravchik and A. Shabtai, "Detecting cyber attacks in industrial control systems using convolutional neural networks," in Proc. Workshop on Cyber-Physical Systems Security and PrivaCy (CPS-SPC), 2018, pp. 72–83.
10. Y. Perez-Riverol et al., "Machine learning applications for industrial control system security: A survey," in various venues; see representative survey Y. Hu et al., "A survey of intrusion detection for in-vehicle networks," IEEE Trans. Intelligent Transportation Systems, 2020.
11. W. Aoudi, M. Iturbe, and M. Almgren, "Truth will out: Departure-based process-level detection of stealthy attacks on control systems," in Proc. ACM SIGSAC Conf. Computer and Communications Security, 2018, pp. 817–831.
12. C. M. Ahmed, V. R. Palleti, and A. P. Mathur, "WADI: A water distribution testbed for research in the design of secure cyber physical systems," in Proc. Workshop on Cyber-Physical Systems for Smart Water Networks (CySWATER), 2017, pp. 25–28.
13. T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in Proc. Int. Conf. Learning Representations (ICLR), 2017.
14. P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," in Proc. Int. Conf. Learning Representations (ICLR), 2018.
15. W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in Advances in Neural Information Processing Systems (NeurIPS), vol. 30, 2017.
16. K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" in Proc. Int. Conf. Learning Representations (ICLR), 2019.
17. W. W. Lo, S. Layeghy, M. Sarhan, M. Gallagher, and M. Portmann, "E-GraphSAGE: A graph neural network based intrusion detection system for IoT," in Proc. IEEE/IFIP Network Operations and Management Symposium (NOMS), 2022, pp. 1–9.
18. Y. Wang et al., "Heterogeneous graph attention network for malware detection," in various security venues; representative work N. Marastoni, R. Giacobazzi, and M. Dalla Preda, "Data augmentation and transfer learning to classify malware images in a deep learning context," Journal of Computer Virology and Hacking Techniques, vol. 17, no. 4, pp. 279–297, 2021.
19. G. Kwon, C. Kim, S. Lee, and H. Kim, "GraphSAGE-based lateral movement detection," in various security venues.
20. M. Kravchik and A. Shabtai, "Efficient cyber attack detection in industrial control systems using lightweight neural networks and PCA," IEEE Trans. Dependable and Secure Computing, vol. 19, no. 4, pp. 2179–2197, 2022.
21. W. Aoudi and M. Almgren, "A framework for determining robust context-aware attack-detection thresholds for cyber-physical systems," in Proc. Annual Computer Security Applications Conference (ACSAC), 2021.
22. MITRE Corporation, "MITRE ATT&CK for Industrial Control Systems (ATT&CK for ICS)," MITRE Technical Report, 2020. [Online]. Available: https://collaborate.mitre.org/attackics/
23. J. Goh, S. Adepu, K. N. Junejo, and A. P. Mathur, "A dataset to support research in the design of secure water treatment systems," in Proc. Int. Conf. Critical Information Infrastructures Security (CRITIS), 2016, pp. 88–99.
24. C. M. Ahmed, V. R. Palleti, and A. P. Mathur, "WADI: A water distribution testbed" (see [12]).
25. H.-K. Shin, W. Lee, J.-H. Yun, and H. Kim, "HAI 1.0: HIL-based Augmented ICS security dataset," in Proc. USENIX Workshop on Cyber Security Experimentation and Test (CSET), 2020. Author declaration: This paper presents a proposed comparative evaluation framework for Graph Neural Network models applied to multi-stage cyber attack detection in Industrial Control Systems. No experimental implementation, prototype training, or empirical benchmarking was completed as part of this manuscript. Comparative empirical evaluation on public SCADA-grade datasets is identified as future work. All performance-related claims are reserved for that future experimental study.