Computers, Materials & Continua

Tech Science Press

https://doi.org/10.32604/cmc.2026.084993

ARTICLE

TF-SAGE: Trust Filtered Graph Learning for Stable Internet of Things Intrusion Detection under Adversarial Attacks

Chin-Shiuh Shieh[1], Thanh-Lam Nguyen[1], Thanh-Tuan Nguyen[2,*], Xuan-Huy Nguyen[2], Chau-Tan-Phat Le[2] and Mong-Fong Horng[1,*]

[1]Department of Electronic Engineering, National Kaohsiung University of Science and Technology, Kaohsiung, Taiwan
[2]Department of Electrical and Electronic Engineering, School of Engineering and Technology, Nha Trang University, Khanh Hoa, Vietnam

*Corresponding Authors: Thanh-Tuan Nguyen. Email: tuannt@ntu.edu.vn; Mong-Fong Horng. Email: mfhorng@nkust.edu.tw

Received: 03 May 2026; Accepted: 07 August 2026; Published: 15 September 2026

ABSTRACT: Internet of Things (IoT) intrusion detection systems face increasing pressure from adversarial attacks that can manipulate not only feature vectors but also the relational structure on which graph based models rely. This paper proposes Trust Filtered GraphSAGE (TF-SAGE), a graph based intrusion detection system (IDS) pipeline in which edges are assigned trust scores, filtered before message passing, and coupled with uncertainty aware inference to reduce overconfident decisions under unstable neighborhoods. The model is evaluated on NF-ToN-IoT-v2 as the main benchmark and CICIIoT2025 as an independent confirmation benchmark under the same FSAA and GSAA evaluation protocol. The results show that TF-SAGE is not the top clean score model, yet it maintains substantially stronger stability under attack: on NF-ToN-IoT-v2, it reaches clean macro averaged F1 (Macro-F1) 0.9789, retains 0.9776 under Feature Space Adversarial Attack (FSAA), and achieves 0.9048 with attack success rate (ASR) 0.0941 under GSAA, while the graph baselines degrade more severely. Evidence from calibration and neighborhood recovery further indicates that these gains are mechanistically grounded rather than reducible to a single summary score. These findings position TF-SAGE as a practical resilience oriented design direction for IoT intrusion detection based on graph neural networks (GNNs).

KEYWORDS: Internet of Things intrusion detection; graph neural networks; adversarial stability; trust filtered graph construction; uncertainty calibration; feature space adversarial attack; graph space adversarial attack

1 Introduction

The Internet of Things (IoT) has become an operational infrastructure layer for factories, logistics, healthcare, transportation, and smart homes. This expansion enables automation and continuous monitoring, but it also enlarges the attack surface because IoT systems often operate under limited resources, fragmented protocols, long deployment lifecycles, and partially uncontrolled network environments. Recent bibliometric evidence shows that research on intrusion detection for IoT is growing rapidly in both scale and specialization [1]. Yet, in real deployment, an intrusion detection system (IDS) cannot be judged only by its accuracy on clean data; what matters more is whether it can still produce reliable alerts when traffic patterns shift, malicious behavior is disguised, or communication relations among devices are distorted.

What makes IoT IDS distinct is that many attack indicators do not reside in a single feature, but in the interaction pattern among hosts, protocols, time windows, and connection clusters. If the model relies only

on independent feature vectors, it can easily miss the behavioral structure that often determines whether legitimate and malicious flows remain distinguishable when an attacker imitates benign behavior. Therefore, a useful IoT IDS should represent both feature information and relational context.

Machine learning and deep learning have substantially improved intrusion detection performance, but they have also exposed a weakness in common evaluation practice: clean performance is not equivalent to security value. Recent studies on feature optimization, classifier comparison, and real time IDS pipelines over IoT datasets continue to report strong performance under controlled settings [2,3]. Ensemble systems and attack specific pipelines also show how quickly machine learning/deep learning (ML/DL) IDS research is expanding [4,5]. However, strong benchmark performance does not automatically guarantee defensive value when the input is manipulated. Recent surveys on adversarial machine learning for network intrusion detection systems consistently emphasize that IDS based on machine learning (ML) can be misled by small but purposeful perturbations [6,7]. The central problem is therefore not merely to raise clean F1, but to preserve stable detection behavior under adversarial stress.

In recent years, graph neural networks (GNNs) have emerged as a particularly suitable direction for IDS because network traffic is not a collection of independent samples. The latest survey on GNNs for intrusion detection shows a clear shift from sample level learning toward relation aware and structure aware learning [8]. This shift is supported by positive evidence. Anomal-E shows that edge information and topological structure can be exploited effectively even in a self supervised setting [9]. Altaf et al. and Gao et al. likewise show that graph based learning can improve anomaly detection when topology and attributes are modeled jointly [10,11]. More recently, EMA-IDS highlights the role of edge features and multi hop attention, while Ngo et al. propose attribute based graph construction for IoT IDS [12,13]. In general, graph learning allows the model to represent the geometry of network behavior rather than only the value of each individual record.

However, a major gap in this literature is that most studies focus on the graph backbone while treating graph construction as a reasonable preprocessing step. In graph based IDS, the graph is not given; it is built from similarity thresholds, time windows, node definitions, and relation constraints. If edges are formed incorrectly or neighborhoods are retained poorly, message passing can amplify confusion rather than supply useful context. From a security perspective, graph construction should be treated as an attack surface, not only as preprocessing. This argument is consistent with recent warnings from the adversarial network intrusion detection systems (NIDS) literature [6,7], and becomes even more important in the GNN setting, where perturbations may simultaneously affect features and topology [14]. Moreover, the issue does not stop at attacked accuracy. Hsu et al. show that GNNs can be miscalibrated in ways that are specific to graph data [15]. In deployment oriented literature, work on federated learning and explainable IDS also suggests that reliability is increasingly tied to distributed settings and the need for interpretable decisions [16,17]. This indicates that a graph based IDS should not only be correct, but should also know when its neighborhood is no longer trustworthy.

These observations motivate the central question of this study: can an IDS based on GNNs preserve reliable detection behavior when the graph used for inference is itself exposed to perturbation? The contributions are listed as follows:

(i) TF-SAGE is proposed as a Trust Filtered GraphSAGE framework for IoT IDS, where graph construction is protected through edge persistence, trust filtering, and uncertainty aware inference.
(ii) A dual surface evaluation protocol is introduced to separate Feature Space Adversarial Attack (FSAA) from Graph Space Adversarial Attack (GSAA), making feature driven degradation distinguishable from neighborhood manipulation.

(iii) Experiments on NF-ToN-IoT-v2 and CICIIoT2025 show that protecting graph construction helps graph based IDS maintain more stable detection behavior under adversarial pressure while preserving competitive clean detection.

2 Related Work

The related literature is reviewed along three complementary layers: IoT IDS based on machine learning and deep learning, IDS based on graph representations and graph neural networks, and evaluation directions that move beyond clean performance. The intersection of these layers is not only about which classifier to use, but about whether an IDS can retain its defensive value when data conditions, communication structure, and operational settings become less stable.

2.1 Machine Learning and Deep Learning IDS for IoT

Machine learning and deep learning IDS remain the largest foundation of the current IoT security literature. Recent reviews consistently show that most studies still follow a relatively stable pipeline: extract features from packets or flows, represent the data as vectors, and then apply supervised learning, deep learning, or hybrid learning to classify benign and malicious traffic [1,18]. Studies by Xu et al. and Shi et al. further confirm the continuing importance of feature selection and ensemble learning pipelines for IoT IDS [2,4]. The appeal of this line of work lies in its ease of deployment, reasonable training cost, and straightforward benchmarking on standardized datasets. Thus, even as graph based and robust IDS research expands rapidly, ML/DL IDS still provides the central baseline family in most IoT intrusion detection studies.

Within this family, classical learners and deep models continue to coexist. Berhili et al. show that random forest, decision tree, support vector machine (SVM), logistic regression, and gradient boosting remain widely used because they are relatively fast to train and easier to interpret [19]. Kikissagbe and Adda likewise note that supervised learning and hybrid systems still dominate many NetFlow and traffic feature benchmarks [18]. Musthafa et al. add an important point: class balancing, feature selection, and the way the training set is constructed can materially improve stability under class imbalance [20]. Schroetter et al., in their comparison between neural network based IDS and signature based IDS, further show that the benefit of learning based approaches should be evaluated against realistic traditional baselines rather than assumed by default [21]. In a more deployment oriented direction, Ashraf et al. build an INIDS pipeline over BoT-IoT and compare multiple classifiers to emphasize the role of classifier selection in realistic settings [3]. These results collectively show that IDS performance depends simultaneously on input quality, evaluation protocol, and the learning algorithm, not merely on model architecture. Recent optimization driven IoT IDS work also illustrates the continued emphasis on improving clean detection pipelines. For example, the Decisive Red Fox optimization and descriptive back propagated radial basis function framework reports an optimized IoT IDS design for attack classification [22].

Since 2024, the deep learning literature has expanded rapidly toward richer representation learning. Kim et al. propose a transferable deep learning framework to reduce dataset dependence and improve transferability in IoT IDS [23]. Cui et al. combine residual learning with attention mechanisms to strengthen feature learning and focus on informative traffic signals [24]. Yaras and Dener introduce a hybrid deep learning algorithm, showing that combined architectures remain attractive when attempting to balance accuracy and representational power [25]. Beyond central training, Chaurasia et al. place residual networks into a federated learning framework for industrial Internet of Things (IIoT) [16]. Ali and Al-Sharafi focus on man in the middle (MitM) specific attack mitigation, reflecting a broader movement toward more attack specific and deployment specific IDS design [5]. In another line, Tseng et al. introduce transformer based multiclass intrusion detection on CIC-IoT-2023, showing that the field has clearly shifted from manual

feature engineering toward deeper representation learning [26]. These contributions indicate that the main point of debate is no longer whether deep learning should be used, but which representation learning strategy preserves the most useful information for intrusion detection.

Even so, most studies still evaluate performance in an in domain and label rich setting [18,19]. The reported metrics remain largely Accuracy, Precision, Recall, Macro-F1, and area under the curve (AUC) [23,26]. Even many recent deep models still emphasize clean classification performance under controlled conditions [24,25]. Such settings are useful for comparing classification ability, but they are not sufficient to represent the operational value of IDS in real IoT environments, where traffic evolves with usage patterns, communication among devices, and local network conditions. In addition, many approaches still treat each flow or each sample as a relatively independent classification unit, a limitation repeatedly noted in IoT IDS reviews [1,18]. From the perspective of representation, this point also motivates the move toward graph based modeling [8]. Therefore, ML/DL IDS remains indispensable as a foundation, but precisely because of both its strengths and its limitations, the literature naturally raises the next question: if relations among devices and traffic flows carry meaningful security information, is vector based representation still the only reasonable way to model IDS for IoT?

2.2 Graph Based IDS and Graph Neural Networks

The shift toward graph based IDS is driven by the recognition that network traffic is not a set of independent samples, but a system of interactions with source to destination relations, neighborhood context, and communication structure that changes over time. Zhong et al. show that the GNN based intrusion detection literature has evolved around three major axes: graph construction, network design, and deployment [8]. The attraction of this direction lies in the fact that graph learning models relations, context, and communication structure directly, rather than collapsing the data into fixed vectors at the outset. For IoT, this is particularly valuable because many attack traces emerge through repeated interaction patterns or structural shifts in traffic over a time window.

Empirically, graph based IDS has developed through several complementary lines. Caville et al., with Anomal-E, show that edge information and graph topology can support intrusion and anomaly detection even when labels are scarce, opening a path toward self supervised and weakly supervised IDS [9]. Altaf et al. propose a concatenated multigraph neural network and argue that a single graph is not always sufficient to capture the multiple relation types present in IoT communication [10]. Gao et al. exploit attribute graphs and meta path based GNNs for anomaly traffic detection, emphasizing the benefit of combining topology with node attributes [11]. Ngo et al. approach the problem through attribute based graph construction, showing that how the input graph is formed remains a decisive factor in downstream performance [13]. What unifies these studies is their shift of emphasis from choosing a classifier to constructing a data structure that preserves security relevant relational meaning.

Since 2024, this direction has advanced further in both representational depth and scalability. Tran and Park propose FN-GNN to improve graph embeddings for NIDS [27]. Yin et al. address large scale botnet detection through GraphSAINT based subgraph sampling combined with graph isomorphism networks, emphasizing that scalability and computational cost are central if IDS based on GNNs are to move closer to deployment [28]. Le and Park focus on multiclass attack detection through feature rearrangement, showing that even in multiclass settings the organization of input information can remain a performance bottleneck [29]. Deng and Huang propose EMA-IDS, exploiting edge features and multi hop attention to strengthen information aggregation in NIDS [12]. Together, these contributions suggest that graph based IDS is maturing from proof of concept demonstrations toward the optimization of representation quality, efficiency, and multiple attack discrimination.

Yet the strength of graph based IDS also depends heavily on graph construction. In IDS, graphs may be defined in several ways. One line treats hosts as nodes and flows as edges [10]. Another builds attribute graphs or models multiple relation types across features and entities [11,13]. Each graph definition implicitly encodes a different hypothesis about which relations carry security meaning and which signals should be propagated through message passing. This is why Zhong et al. identify graph construction as a central challenge for the field [8]. More recent work by Tran and Park also indirectly supports this observation, because improvements in embedding or feature arrangement generally matter only if the input graph already captures meaningful structure [27,29]. Therefore, graph based IDS should not be understood as merely replacing a multilayer perceptron (MLP) or convolutional neural network (CNN) with a GNN; it is fundamentally a problem of deciding which edges, neighborhoods, and graph building rules model the underlying IoT communication system correctly.

Accordingly, graph based IDS has made substantial progress in modeling relations, but it is not an automatic solution in every context. If the graph is poorly defined or heavily contaminated, message passing can propagate noise rather than useful context. Moreover, computational cost and deployment on large graphs remain recurring challenges [12,28]. At a broader level, model stability under structural shifts is also repeatedly emphasized [8,27]. Because graph based IDS introduces an additional structural representation layer, evaluation must become deeper as well: we must ask not only whether the model is correct, but whether the learned graph is stable and whether the resulting conclusions retain value when conditions deteriorate. This naturally leads to the next thread of literature on adversarial stability and reliability.

2.3 Adversarial Stability and Reliability Evaluation for IoT IDS

An increasingly important research axis concerns how IDS should be evaluated. Many studies still report Accuracy, Precision, Recall, F1, or AUC on clean data as their primary metrics, but the reality of IoT deployment imposes much broader requirements [6,7]. He et al. synthesize evidence that, in NIDS, carefully designed perturbations can severely degrade performance even when the resulting inputs remain within a valid domain [6]. Sharma and Chen similarly show that both white box and black box attacks can substantially affect intrusion detection systems [7]. High clean performance thus only demonstrates that the model learns well in a narrow setting; it does not guarantee defensive value when traffic, adversaries, or local network conditions change.

From this perspective, recent literature has begun to broaden evaluation from pure classification ability to operational trustworthiness. Schroetter et al. show that many ML based IDS papers still fail to compare against realistic signature based baselines, creating a risk of overstating the benefit of learning based methods [21]. This matters because IDS in practice are constrained by resources, false positives, latency, and maintainability. Closer to deployment, Wardana et al. emphasize trust management, privacy preserving collaboration, and lightweight design in collaborative IoT IDS [30]. Although this line is not identical to adversarial resistance in the narrow sense, it reflects a broader shift in evaluation thinking: an IDS is considered trustworthy not only because it is often correct, but because it behaves stably without imposing excessive operational cost.

For graph based IDS, this issue is even more difficult because perturbations can affect not only features but also topology and neighborhood structure. Zhao et al. show that resistance to adversarial perturbation in GNNs is tightly linked to the stability of graph dynamics [14]. Many observations in Zhong et al.'s survey also make this point clearer, as graph construction is treated as a central bottleneck in GNN based intrusion detection [8]. Therefore, evaluating graph based IDS under attack should not stop at attacked accuracy alone; it must also help answer whether the model fails in feature space, in topology, or in the very logic used to construct edges and neighborhoods.

Beyond stability under attack, the reliability of confidence estimates is becoming an equally important evaluation axis. Hsu et al. show that GNNs suffer from graph specific forms of miscalibration, meaning that model confidence should not be treated as an automatic consequence of high accuracy [15]. For IDS, this is a practical concern because a false alert with unjustifiably high confidence can create substantial operational cost. In a more application oriented direction, Alabbadi and Bajaber propose IoT data stream IDS with explainable artificial intelligence (XAI), reflecting how reliability is increasingly connected to interpretability and auditability [31]. The current literature therefore extends evaluation toward calibration, uncertainty quality, and interpretability, rather than concentrating solely on the final classification output [15,31].

Taken together, recent work suggests a broader evaluation framework for IoT IDS. On the attack stability axis, surveys and adversarial studies show that clean score is insufficient to establish defensive value [6,14]. On the reliability axis, miscalibration and XAI literature continue to emphasize the need to evaluate uncertainty and interpretability [15,17]. On the deployment axis, fair benchmarking, trust management, and operational constraints are increasingly treated as part of the problem itself [21,30]. This framing reconnects the two previous literature streams: from ML/DL IDS to graph based IDS, the key question is no longer only which model scores higher, but which one maintains more trustworthy conclusions when the environment degrades. This provides a natural transition to the proposed method, because the current literature needs not only new models, but also design and evaluation strategies that make IDS results more operationally meaningful.

3 Proposed Method

TF-SAGE is organized around four components: the overall processing logic, trust aware graph construction, uncertainty aware representation learning, and the two attack spaces used to evaluate stability under adversarial stress. The design places graph construction before message passing as a component that must itself be controlled, and it incorporates uncertainty into inference to reduce overconfident decisions.

3.1 TF-SAGE Overview

TF-SAGE is designed to prioritize stability under attack, in which graph construction is not treated as fixed preprocessing but as a vulnerable component that can itself be perturbed and should therefore be controlled. The system input is a normalized traffic window; flows inside the window are mapped to nodes, and the relations among these nodes are inferred rather than imposed through a single hard rule.

The pipeline contains three main blocks. First, TF-SAGE builds a candidate graph from pairwise compatibility among flows. Second, the candidate edges are reexamined through perturbation stability and uncertainty signals, so that fragile neighborhoods are weakened before message passing. Finally, GraphSAGE learns representations over the filtered graph and performs repeated stochastic inference to estimate both the predicted label and the confidence of the decision.

The key difference of TF-SAGE is therefore not a more complex backbone, but a control layer placed before information propagation. If graph construction is wrong, noise can spread across many nodes and create overconfident predictions. TF-SAGE reduces this risk by treating edge trust as an intermediate signal linking raw input, graph structure, and output inference. Fig. 1 summarizes the flow from traffic representation to trust filtering, GraphSAGE inference, and calibrated output.

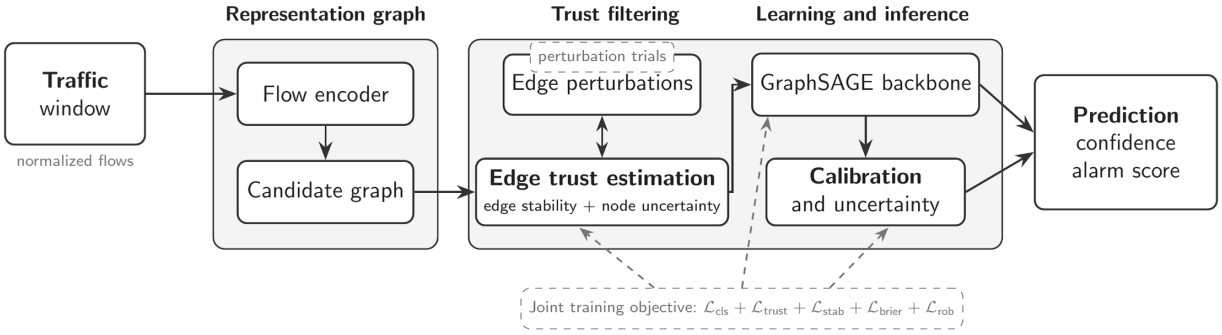


Figure 1: TF-SAGE processing flow.

3.2 Trust Aware Graph Construction for Intrusion Detection

The graph construction block turns a traffic window into a structural hypothesis about relations among flows. For n flows within a normalized window, the data are represented as an attributed graph:

$$G = (V, E_{cand}, X),\ X = [x_1, \ldots, x_n], x_i in R^d \tag{1}$$

In (1), each node corresponds to a flow, X is the feature matrix, and E_{cand} is a candidate edge set that is inferred rather than fixed in advance. Each flow is mapped into a latent space:

$$H = \varphi(X),\ h_i = \varphi(x_i) \tag{2}$$

Within this latent space, TF-SAGE computes compatibility between two flows by combining representational similarity, temporal context, and relational cues. In the implemented TF-SAGE pipeline, $c(i, j)$ denotes the scalar compatibility score returned by the projection scorer from the normalized pair representation of nodes i and j after the learnable projection layer. The candidate kNN edge weight is $w(i, j) = \frac{1}{1+|x_i - x_j|_2}$. The trust term $\tau(i, j)$ denotes a perturbation stability score: over Q admissible perturbation trials, it is the fraction of trials in which the projected compatibility score of edge (i, j) remains above the clean window median score. Thus, $c(i, j)$ ranks candidate compatibility, $w(i, j)$ records distance based neighborhood strength, and $\tau(i, j)$ measures local edge stability before threshold based edge retention.

$$s(i, j) = \alpha \cos(h_i, h_j) + \beta\tau(i, j) + \gamma c(i, j) \tag{3}$$

The score $s(i, j)$ is not interpreted as the probability that an edge is correct, but as a first stage compatibility signal used to select candidate neighbors. The candidate neighborhood and candidate adjacency are then defined as:

$$N_i^{cand} = TopK_i(s(i, *)),\ A_{cand}(i, j) = I[j in N_i^{cand}] \tag{4}$$

The candidate graph may still contain fragile edges. TF-SAGE therefore reevaluates each edge by rebuilding the graph under multiple admissible perturbations and measuring how often the edge persists:

$$t(i, j) = 1/Q sum_{q=1}^{Q} I[(i, j) in BuildGraph(X + \delta^q)],\ \delta^q in \Delta(\varepsilon) \tag{5}$$

Edge filtering is estimated before final prediction from perturbation stability of candidate-edge compatibility scores. Prediction uncertainty is used as an inference-time reliability signal and calibration diagnostic, not as a direct dependency that recomputes the same final graph from its own final prediction. This ordering

avoids a circular interpretation: candidate graph construction precedes trust estimation, trust filtering precedes message passing, and stochastic uncertainty is reported after the filtered representation is obtained.

$$u_{node}(i) = -sum_{yinY} p_i(y) \log p_i(y) \tag{6}$$

To avoid a circular dependency between the final trust graph and predictions produced by that same graph, $p_i(y)$ in (6) is read from a preliminary phase. TF-SAGE first builds the candidate graph A_{cand} through (3)–(5), runs either a warm up pass or uses cached predictions from the previous epoch on this preliminary graph, and then computes u_{node}. Discrete operations such as TopK, the θ_f threshold, and edge reselection are treated as stop gradient control steps within the update, so gradients do not pass through the edge selection decisions.

From perturbation-based persistence, TF-SAGE defines $tau(i, j)$ for each candidate edge and retains edges whose stability exceeds θ_f, while enforcing a minimum local neighborhood k_min so that filtering does not isolate nodes.

$$w(i,j) = t(i,j)exp(-\rho(u_{node}(i) + u_{node}(j))) \tag{7}$$

TF-SAGE then filters edges using a trust threshold and normalizes the adjacency matrix before passing it to GraphSAGE:

$$A_f(i,j) = I[w(i,j) \geq \theta_f]A_{cand}(i,j), A_{norm} = D_f^{-1/2} A_{fD\_f}^{-1/2} \tag{8}$$

These trust scores are used for edge selection and filtering rather than as direct message passing weights; GraphSAGE receives the filtered and normalized graph. The output of this block is the normalized trust graph A_{norm}, in which fragile neighborhoods have already been weakened before representation learning begins. Fig. 2 illustrates the transition from latent nodes to a candidate graph and then to a trust filtered graph.

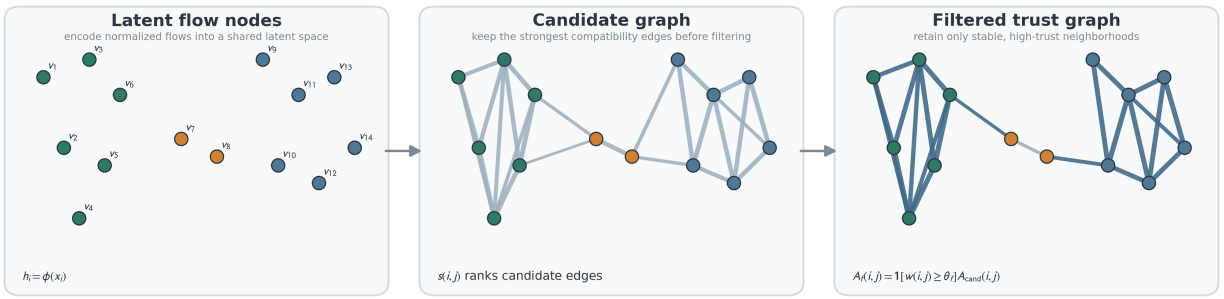


Figure 2: Trust graph construction in TF-SAGE.

3.3 Representation Learning and Uncertainty Aware Inference

Once the trust graph is obtained, TF-SAGE learns node representations by applying GraphSAGE over the filtered structure. At each layer, neighborhood signals are aggregated only through A_{norm}, so message passing no longer propagates over the entire unfiltered candidate graph:

$$Z^0 = H, R^l = A_{norm} Z^{l-1} \tag{9}$$

The node state is then updated from both self information and trust filtered neighborhood information:

$$Z^l = \sigma(Z^{l-1} W_{self}^l + R^l W_{neigh}^l + b^l) \tag{10}$$

This separation is important in IDS because a flow should not be classified entirely based on edges that may themselves be manipulated.

To reduce overconfident predictions, TF-SAGE does not rely on a single forward pass. Instead, it generates M stochastic predictions, averages the predictive distributions, and then reads both confidence and uncertainty from this distribution set:

$$P_{mean} = 1/Msum_{m=1}^{M} P^{m} \tag{11}$$

From the averaged distribution, the predicted label and base confidence are defined as:

$$y_{hat\_i} = \underset{yinY}{\text{argmax}}\, P_{mean}(i, y), c_i = \underset{yinY}{\max} P_{mean}(i, y) \tag{12}$$

Two types of uncertainty are estimated from predictive entropy and disagreement across stochastic forward passes:

$$u_{alea}(i) = -\text{sum}_{yinY} P_{mean}(i, y) \log P_{mean}(i, y), u_{epi}(i) = 1/Msum_{m=1}^{M} ||P^{m}(i,) - P_{mean}(i,)||_2^2 \tag{13}$$

In (13), $u_{alea}(i)$ rises when a sample is intrinsically difficult to separate, while $u_{epi}(i)$ rises when repeated inferences remain unstable. These signals help decouple confidence from genuine decision reliability. The alarm score is therefore calibrated as follows:

$$a_i = I[y_{hat\_i} = attack] c_i exp(-etau_{alea}(i) - zetau_{epi}(i)) \tag{14}$$

Fig. 3 illustrates the process from the trust graph, through stochastic GraphSAGE inference, to the averaged predictive distribution and uncertainty signals.

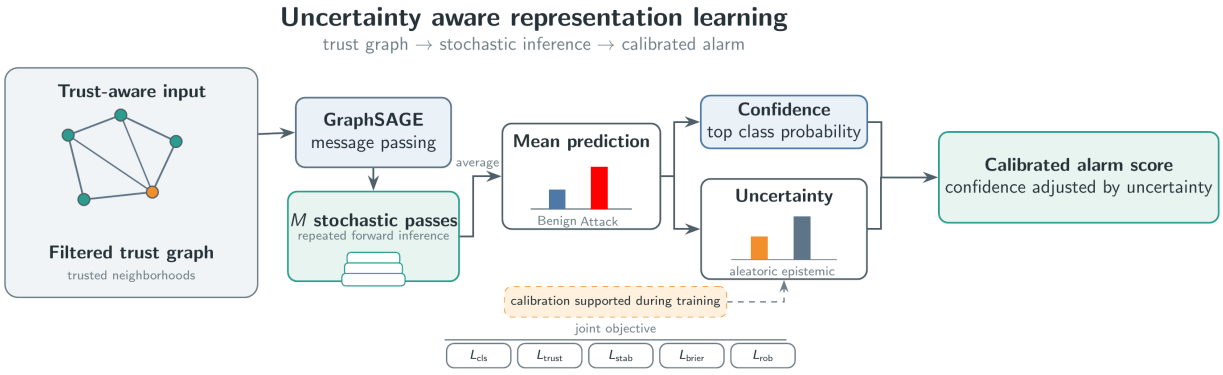


Figure 3: Stochastic inference and alarm calibration.

During training, TF-SAGE optimizes classification, trust consistency, local stability, calibration, and adversarial loss within a single objective:

$$L_{total} = L_{cls} + \lambda_{trust} L_{trust} + \lambda_{stab} L_{stab} + \lambda_{brier} L_{brier} + \lambda_{rob} L_{rob} + \lambda_2 ||\Theta||_2^2 \tag{15}$$

In (15), L_{cls} is cross entropy on labeled clean samples and on admissible adversarial views when they are included in the mini batch. L_{trust} penalizes inconsistency between candidate edge persistence and the edge selection scores retained after filtering, while L_{stab} limits prediction distribution drift between clean and perturbed views. L_{brier} is the Brier score on predictive probabilities for calibration, and L_{rob} is the classification

loss on FSAA or GSAA views generated during training. Gradients are used for the differentiable encoder, GraphSAGE, and classifier branches; TopK, thresholding, and edge selection are kept as stop gradient control operations.

3.4 Adversarial Attack Spaces

To interpret behavior under attack consistently, this study separates two attack spaces at a conceptual level. Feature Space Adversarial Attack (FSAA) operates on the feature vector of each flow, whereas Graph Space Adversarial Attack (GSAA) operates on edges and neighborhoods that govern information propagation in the graph model. These are not implementation details of TF-SAGE itself, but complementary stress tests for two different forms of brittleness in IDS.

FSAA asks whether the model depends too strongly on fragile decision boundaries in feature space. GSAA asks whether graph construction and message passing remain stable when the neighborhood structure is manipulated. The following definitions therefore focus on the three elements required by the evaluation protocol: the perturbation domain, the attack objective, and the attack success rate (ASR).

3.4.1 Feature Space Adversarial Attack

Feature Space Adversarial Attack (FSAA) represents a family of attacks that act directly in the feature space of network flows. The purpose of FSAA is to generate admissible perturbations on the input representation so that malicious samples become more difficult to distinguish from benign behavior while remaining inside the valid data domain. FSAA therefore primarily tests the stability of models that depend strongly on decision boundaries in feature space.

For a flow x_i, FSAA forms an adversarial sample through controlled perturbation over the subset of features allowed to change:

$$x_i^{FSAA} = Proj_X(x_i + m_i\delta_i) \tag{16}$$

The FSAA perturbation domain is restricted by the budget and valid bounds of the chosen features:

$$\Delta_i^{FSAA(\varepsilon_i)} = \{\delta_i : \|D_i(m_i\delta_i)\|_p \le \varepsilon_i, l_i \le x_i + m_i\delta_i \le u_i\} \tag{17}$$

In (16) and (17), m_i is the feature mask, D_i rescales perturbations according to feature magnitude, and l_i and u_i define the valid lower and upper bounds. The attacker selects perturbations that maximize classification loss or reduce the probability assigned to the attack class:

$$\delta_i^* = \text{argmax}_{\delta_i in \Delta_i^{FSAA}} ell(f_{Theta}(x_i + m_i\delta_i, A), y_i) \tag{18}$$

Attack success is measured as the fraction of true attack samples pushed into the benign decision region:

$$ASR_{FSAA} = \text{sum}_i I[y_i = attack] I[y_{hat\_i}(X_{FSAA}^{adv}, A) = benign] / \text{sum}_i I[y_i = attack] \tag{19}$$

FSAA acts in feature space and does not directly edit graph edges. If the graph is reconstructed from perturbed features, structural effects may arise indirectly, but the attacker's original manipulation space remains X. Fig. 4 illustrates this mechanism as an attack sample shifted toward the benign decision region.

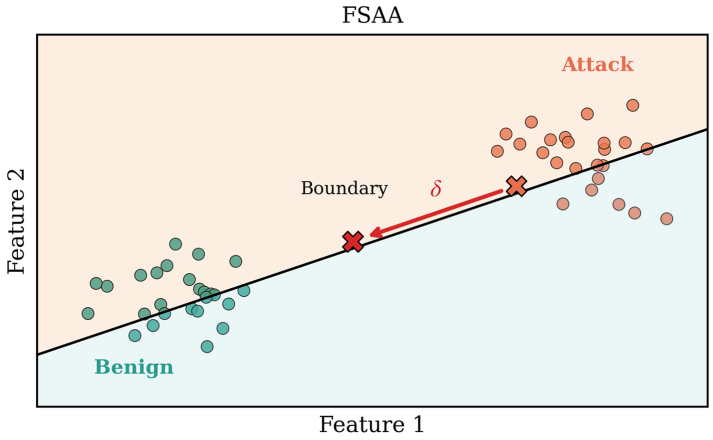


Figure 4: FSAA in feature space.

3.4.2 Graph Space Adversarial Attack

Graph Space Adversarial Attack (GSAA) represents a family of attacks that operate in the structural space used by the graph model for information propagation. Rather than perturbing each feature vector in isolation, GSAA targets the signals that influence edges, neighborhoods, and the relational dependencies inferred during graph construction. GSAA therefore directly tests the central hypothesis of TF-SAGE: graph construction itself is an attack surface that must be controlled before message passing takes place.

Unlike FSAA, GSAA modifies graph structure directly. With an edge edit matrix Δ_A, the adversarial graph and its admissible edit domain are summarized as follows:

$$A_{GSAA}^{adv} = \Pi_G(A + \Delta_A^*) \tag{20}$$

The feasible domain of GSAA is restricted by a normalized graph-space budget ε and a local degree shift constraint:

$$\Omega_\varepsilon^{GSAA} = \{\Delta_A : \Delta_A(i,j) \in \{-1,0,1\}, 0 \le A + \Delta_A \le 1, \sum_{i<j} |\Delta_A(i,j)| \le \lfloor \varepsilon |E_A| \rfloor, |d_i(A + \Delta_A) - d_i(A)| \le \kappa_i, \forall i\} \tag{21}$$

In these equations, Π_G projects the edited matrix back into the valid graph domain, ε is the normalized graph-space perturbation budget, $|E_A| = \sum_{i<j} A(i,j)$ is the number of candidate edges in the clean graph, and $\lfloor \varepsilon |E_A| \rfloor$ converts the normalized budget into the maximum number of allowed edge edits. The term κ_i limits the local degree shift of node i.

$$\Delta_A^* = \underset{\Delta_A \in \Omega_\varepsilon^{GSAA}}{\operatorname{argmax}} \ell(f_\Theta(X, A_{norm}(A + \Delta_A)), y) \tag{22}$$

The attacker chooses edge edits that increase classification loss or reduce the probability of recognizing attack samples:

$$ASR_{GSAA} = \frac{\sum\_iI[y_i = attack]I[\hat{y}_i(X, A_{GSAA}^{adv}) = benign]}{\sum\_iI[y_i = attack]} \tag{23}$$

GSAA is particularly important for graph based IDS because a single misleading edge may influence many nodes through message passing. In addition to ASR, GSAA results are interpreted together with mechanism oriented signals such as neighborhood overlap, trusted neighbor recovery, and trust drop in

order to determine which parts of the neighborhood are broken or restored after trust filtering. Fig. 5 illustrates this process through the insertion of misleading edges, the removal of stable relations, and the retention of more plausible connections after filtering. Given a clean candidate graph, node features, labels, a normalized graph-space budget ε, and a local degree shift bound κ_i, GSAA first identifies structural edits Δ_A that can reduce detection confidence or increase classification loss within the feasible domain $\Omega_{\varepsilon}^{GSAA}$. It then projects the edited graph back into the admissible graph domain through Π_G, applies the same trust filtering and message passing pipeline used during evaluation, and computes attacked Macro-F1 and ASR on the resulting graph.

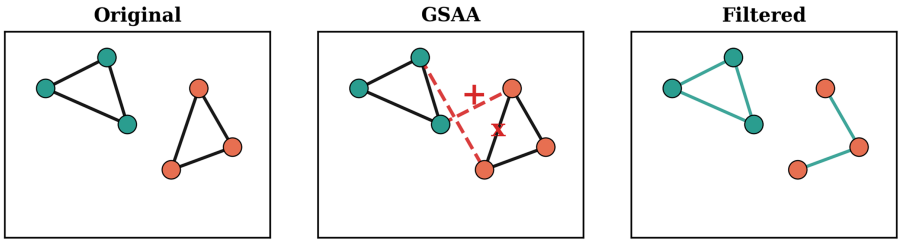


Figure 5: GSAA in graph space.

3.5 Algorithmic Specification of TF-SAGE

Once graph construction, representation learning, and the attack spaces have been defined, TF-SAGE can be specified algorithmically. The purpose of this presentation is to clarify execution order, define the inputs and outputs of each block, and separate methodological logic from experiment specific settings. By design, TF-SAGE is not organized as a deeper or more complicated GNN, but as a structural control pipeline: build a candidate graph, test edge stability, filter fragile neighborhoods, and only then perform representation learning and uncertainty aware inference.

Algorithm 1 specifies trust graph construction. It takes a normalized traffic window, produces latent embeddings, scores pairwise relations among flows, selects candidate neighbors, and then reevaluates these edges under a family of admissible perturbations. Node uncertainty is obtained from warm up predictions or cached predictions from the previous epoch, so trust estimation is not circularly dependent on the final filtered graph in the same update. The output is the normalized adjacency A_{norm} together with trust scores for retained edges. This step treats graph construction as an explicitly controlled operation rather than a fixed preprocessing stage.

Algorithm 1: Trust aware graph construction for TF-SAGE

Input: normalized traffic window X, relation metadata M, graph budget k, perturbation family $\Delta(\varepsilon)$, trials Q, threshold θ_f, warm up or cached predictions P_{pre}
Output: normalized trust graph A_{norm} and edge selection scores W

Initialize $A_{cand} \leftarrow 0$, $W \leftarrow 0$
Compute latent representations $H \leftarrow \varphi(X)$
for each node $v_i \in V$ **do**
Compute candidate compatibility scores $s(i,j)$ using Eq. (3)
Select candidate neighbors $N_i^{cand} \leftarrow TopK_i(s(i,*))$
Update $A_{cand}(i,j) \leftarrow I[j \in N_i^{cand}]$
end for
Compute preliminary node uncertainty u_{node} from P_{pre} or a warm up pass on A_{cand}

(Continued)

Algorithm 1 (continued)

for $q = 1, \ldots, Q$ **do**
Sample an admissible perturbation $\delta^q \in \Delta(\varepsilon)$
Rebuild perturbed candidate graph $G^q \leftarrow \text{BuildGraph}(X + \delta^q)$
end for
for each edge $(i, j) \in E_{cand}$ **do**
Estimate edge persistence $t(i, j)$ using Eq. (5)
Read node uncertainty $u_{node}(i)$, $u_{node}(j)$ using Eq. (6)
Compute trust score $w(i, j) \leftarrow t(i, j) \exp(-\rho(u_{node}(i) + u_{node}(j)))$
Filter edge $A_f(i, j) \leftarrow I[w(i, j) \geq \theta_f] A_{cand}(i, j)$
Store $W(i, j) \leftarrow w(i, j)$ as an edge selection score
end for
Normalize $A_{norm} \leftarrow D_f^{-1/2} A_f D_f^{-1/2}$
return A_{norm}, W

The key point of Algorithm 1 is that edge filtering happens before message passing. If an edge appears only because of local noise or random compatibility in latent space, it can be removed before it influences node representations. This reduces the risk that GraphSAGE becomes a mechanism for amplifying noise. In other words, the trust graph is not merely a variation of a kNN graph; it is a structural validation layer that decides which relations are stable enough to be admitted into representation learning.

Algorithm 2 describes the training and inference stage. It uses A_{norm} from Algorithm 1, performs GraphSAGE propagation, generates multiple stochastic predictions, estimates the averaged predictive distribution, and computes uncertainty signals together with the final alarm score. During training, the aggregate objective in (15) ties classification, edge stability, calibration, and adversarial stress into a single optimization process.

Algorithm 2: TF-SAGE training and uncertainty aware inference

Input: node features X, labels Y_L, objective weights λ, stochastic passes M
Output: trained parameters Θ, predictions $\hat{y}_i$, confidence c_i, alarm scores a_i
Initialize model parameters Θ and optimizer state
repeat
Construct or refresh A_{norm} with Algorithm 1 using P_{pre} or warm up predictions
Compute Trust Filtered GraphSAGE states using Eqs. (9) and (10)
Generate stochastic predictions $P^{(m)}$ for $m = 1, \ldots, M$
Average predictions $P_{mean} \leftarrow (1/M) \Sigma_{m=1}^{M} P^{(m)}$
Set $\hat{y}_i$ and c_i using Eq. (12)
Estimate $u_{alea}(i)$ and $u_{epi}(i)$ using Eq. (13)
Evaluate L_{cls}, L_{trust}, L_{stab}, L_{brier}, and L_{rob}
Compute L_{total} using Eq. (15)
Update Θ by descending the gradient of L_{total}
Update P_{pre} from current warm up or clean pass predictions for the next epoch
until validation criterion is satisfied
for each inference window **do**
Construct A_{norm} with Algorithm 1
Run stochastic forward passes and compute P_{mean}
Compute $\hat{y}_i$, c_i, $u_{alea}(i)$, and $u_{epi}(i)$

(Continued)

Algorithm 2 (continued)

18: Compute calibrated alarm score a_i using Eq. (14)
19: **return** class prediction, confidence, and alarm score
20: **end for**

The two algorithms reveal two design anchors of TF-SAGE. The first is that graph construction is controlled before representation learning begins. This is consistent with the fact that, in IDS based on GNNs, the graph is not raw data but a structural hypothesis inferred from network flows. If this hypothesis is wrong, message passing may propagate incorrect relations and generate overconfident predictions. TF-SAGE therefore places trust filtering before the backbone instead of only adding regularization at the end.

The second design anchor is the separation between confidence and uncertainty. Confidence describes the largest predictive probability, whereas uncertainty reflects either class ambiguity or disagreement across stochastic inferences. In IDS, these two quantities should not be conflated. A prediction with high confidence but also high uncertainty is less trustworthy than one that is both confident and stable across repeated inference. The alarm score in (14) directly encodes this principle.

This design also makes mechanism evaluation possible at the component level. Trust filtering is reflected in neighborhood recovery, uncertainty aware inference is reflected in calibration, and the overall effect of the training objective is reflected in attacked Macro-F1 and ASR. As a result, the experimental results do not stop at score comparison; they connect improved outcomes to structural mechanism and model reliability.

The algorithmic specification is intentionally separated from concrete settings such as neighborhood size, the number of perturbation trials, or the number of stochastic forward passes. This separation keeps the method reusable, while specific execution conditions are defined by the evaluation protocol.

4 Threat Model and Evaluation Protocol

The threat model reflects how graph based IDS can be weakened in operation. The adversary does not need to alter the semantic label of a network flow; instead, it introduces admissible perturbations into the input representation or distorts the structural relations used by the model for information propagation. This assumption is well aligned with IoT settings, where even a small shift in traffic features or neighborhood relations may make malicious activity appear more benign.

The two attack surfaces are used to probe two different forms of model dependence. FSAA tests stability when the feature vector is purposefully perturbed, thereby applying direct pressure to the decision boundary in feature space. GSAA tests stability when neighborhood structure is edited, thereby evaluating whether message passing amplifies misleading relations. For TF-SAGE, the point to prove is not only that the model performs well on clean data, but that trust aware graph construction and uncertainty aware inference reduce degradation when both attack surfaces are activated.

The evaluation protocol places behavior under attack at the center. Clean performance confirms the baseline classification ability, whereas behavior under FSAA and GSAA determines the defensive value of the model. The empirical evidence is organized around three axes: comparison against appropriate baselines, degradation trends under larger attack budgets, and mechanism oriented signals from calibration and neighborhood recovery.

4.1 Benchmark, Baselines, and Experimental Configuration

NF-ToN-IoT-v2 is used as the main benchmark because it is the NetFlow v2 variant of ToN-IoT with 43 extended NetFlow features, developed within the standardized NIDS feature framework of Sarhan et al. (2022) [32]. In this study, the data are converted into a binary flow level representation, metadata fields such as IP addresses and labels are removed from model input, and 40 numerical features are retained for graph construction and classification. The evaluation set is stratified with seed 42 and contains 2.118M flows after combining train, validation, and test splits; the attack ratio remains approximately 64.0% across all three splits. This scale provides a sufficiently strong anchor for evaluation under attack while preserving reproducibility across attack settings.

CICIIoT2025 serves as a confirmation benchmark [33]. It is organized from two source families, attack_data and benign_data, and then standardized into the same binary benign/attack target used throughout this study. The evaluation package used here contains 85,721 flows, split into 60,004 training samples, 12,858 validation samples, and 12,859 test samples; the attack rate is held at 41.57% in each split. Although smaller, this benchmark has a different distribution and is therefore useful for checking whether the degradation trend persists beyond NF-ToN-IoT-v2.

Fig. 6 summarizes dataset structure through two side by side charts, highlighting the train/validation/test flow volume and the benign/attack composition of each benchmark. The remaining configuration choices are condensed into Table 1 to keep the protocol concise and auditable.

The comparison models are organized into two families to reflect the different ways in which they consume input data. The tabular family includes Logistic Regression (LR), multilayer perceptron (MLP), and Random Forest (RF), all of which remain common in recent IoT IDS evaluations, notably in Ashraf et al. (2025) [3]. XGBoost is included from the interpretable XGBoost based IoT NIDS line of Hu et al. (2026) [34], while LightGBM represents the gradient boosting and feature selection direction of Chen et al. (2024) [35]. Because these models operate directly on flow level feature vectors, FSAA is the appropriate test for their decision boundary stability in feature space.

The graph baseline family includes graph convolutional network (GCN), GraphSAGE, and graph attention network (GAT). Retaining these backbones is consistent with recent IDS based on GNNs trends summarized by Zhong et al. (2024) [8], while also staying close to newer applied variants such as FN-GNN by Tran and Park (2024) [27], Deep GraphSAGE enhancements by Saidane et al. (2025) [36], BS-GAT by Wang et al. (2025) [37], and GAT based IoT IDS by Ahanger et al. (2025) [38]. Because this family depends on neighborhood structure, GSAA is used to test its sensitivity to structural perturbation.

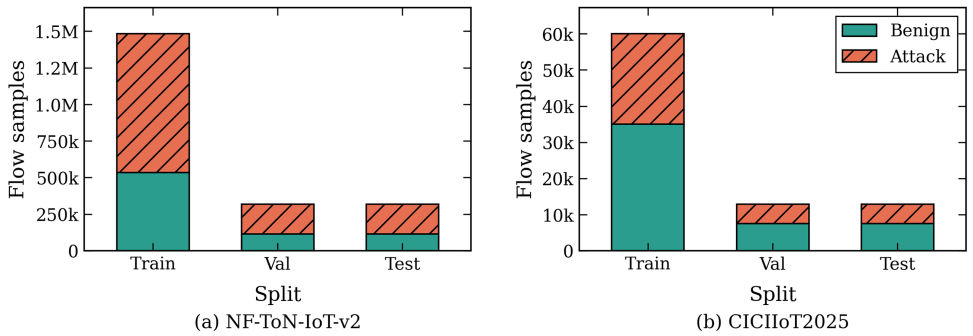


Figure 6: Benchmark scale and class composition.

Table 1: Benchmark and evaluation configuration.

Parameter	NF-ToN-IoT-v2 [32]	CICIIoT2025 [33]
Role	Main benchmark	Independent confirmation benchmark
Flows (train/val/test)	1.482M/317.6k/317.6k	60.0k/12.9k/12.9k
Attack labels	Benign + 9 attacks	Benign/attack
Window length	512 flows	512 flows
Stride	512 flows	512 flows
Graph budget k	10	10
Minimum neighbors k_{min}	4	4
Trust threshold θ_f	0.5	0.5
Headline attack budget ε	0.03	0.03
Attack steps	5	5
Step size	$\varepsilon/4$	$\varepsilon/4$
Trust trials Q	4	4
Uncertainty/stochastic passes T	4	4
Hidden size	128	128
Layers	2	2
Dropout	0.2	0.2

Experimental settings are kept separate from the methodological definition. The runs were executed on a workstation with an NVIDIA RTX 3060 graphics processing unit (GPU); this detail is reported only as minimal reproducibility context and is not used as a comparison variable. Table 1 retains only benchmark information and the main numerical configuration parameters of the pipeline, including window size, stride, k, k_{min}, the threshold θ_f, the headline attack budget, update steps, step size, trust trials, stochastic passes, hidden size, layers, and dropout.

The main values $k = 10$, $k_{min} = 4$, $\theta_f = 0.5$, $Q = 4$, $T = 4$, $\varepsilon \leq 0.03$, and the loss weights were selected using validation behavior, stability under the budget sweep, and computational feasibility on the stated hardware. The same settings are kept across benchmarks unless the profile explicitly changes the data scale, so the reported trends are not the result of per-dataset manual retuning.

For the attack protocol, the budget sweep uses nine ε values from 0 to 0.03. FSAA perturbs only numeric perturbable features and projects the perturbed samples back to valid feature bounds. During FSAA, the candidate edge set is kept fixed, while TF-SAGE recomputes trust filtering from the perturbed features; no direct edge editing is applied. For GSAA, structural edits are applied before trust filtering and message passing, so the attack tests whether the trust filtering stage can suppress unstable or misleading edges before aggregation.

4.2 Reported Metrics and Interpretation Principles

The reported metrics serve two goals: measuring classification quality and measuring degradation under attack. On clean data, Accuracy, Macro-F1, area under the receiver operating characteristic curve (AUROC), and false positive rate (FPR) describe benign/attack discrimination under nominal conditions. Among them, Macro-F1 is prioritized over Accuracy because both benchmarks are class imbalanced and missed attacks carry a more substantial operational cost than a small change in overall correct rate.

For each class $cinY$, Precision measures the fraction of predicted positives that are actually correct:

$$Precision_c = TP_c/(TP_c + FP_c) \tag{24}$$

Recall measures the fraction of true samples from class c that are correctly detected:

$$Recall_c = TP_c/(TP_c + FN_c) \tag{25}$$

F1 balances these two quantities, while Macro-F1 averages class wise F1 to reduce the influence of class imbalance:

$$F1_c = 2Precision_c Recall_c/(Precision_c + Recall_c) \tag{26}$$

$$F1_{macro} = 1/|Y|\text{sum}_{cinY} F1_c \tag{27}$$

Under FSAA and GSAA, the same Macro-F1 computation is applied to adversarial samples in order to obtain attacked Macro-F1. Attack success rate (ASR) directly measures the fraction of true attacks pushed into the benign decision region and is therefore the central defensive metric:

$$ASR = \text{sum}_i I[y_i = attack] I[y^{adv}_{hat\_i} = benign]/\text{sum}_i I[y_i = attack] \tag{28}$$

FPR is used to track false alarms on benign traffic:

$$FPR = FP/(FP + TN) \tag{29}$$

Expected calibration error (ECE) evaluates whether predictive probabilities match empirical correctness. With B confidence bins and B_b the sample set within bin b, ECE is defined as:

$$ECE = \text{sum}^B_{b=1}|B_b|/N|acc(B_b) - conf(B_b)| \tag{30}$$

AUROC is treated as a supporting metric because it reflects ranking quality across thresholds, but attacked Macro-F1 and ASR determine the main conclusion under attack. A model may have strong AUROC while still being unsuitable if ASR remains high under attack, or if FPR rises sharply when the decision threshold is deployed in practice.

Calibration, neighborhood overlap, trust drop, and adversarial retention add a mechanism oriented layer to the classification metrics. If attacked Macro-F1 improves but the filtered neighborhood remains unstable, the trust filtering argument would still be weak. Conversely, when lower degradation under attack is accompanied by better calibration and stronger neighborhood recovery, the result becomes more mechanically grounded.

5 Experimental Results

5.1 Clean Detection Performance

Table 2 reports clean in domain performance on NF-ToN-IoT-v2 for all baselines used in the later sections. All models achieve strong nominal classification quality, but the ordering reveals different nominal advantages across tabular and graph based classifiers. In the tabular family, XGBoost, LightGBM, and RF reach Macro-F1 values of 0.9948, 0.9968, and 0.9968, respectively, reflecting their ability to exploit clear

decision boundaries in normalized flow level data. MLP also maintains high Recall (0.9889), but its FPR of 0.0616 suggests a noticeably higher false alarm cost.

Table 2: Clean performance on NF-ToN-IoT-v2.

Model	Accuracy	Precision	Recall	Macro-F1	AUROC	FPR
LR [3]	0.8131	0.8843	0.8147	0.8481	0.8720	0.1897
MLP [3]	0.9707	0.9662	0.9889	0.9774	0.9950	0.0616
XGBoost [34]	0.9933	0.9928	0.9967	0.9948	0.9995	0.0128
LightGBM [35]	0.9959	0.9957	0.9979	0.9968	0.9997	0.0077
RF [3]	0.9959	0.9960	0.9977	0.9968	0.9988	0.0072
GCN [8]	0.9739	0.9783	0.9809	0.9796	0.9965	0.0386
GraphSAGE [36]	0.9658	0.9825	0.9637	0.9730	0.9961	0.0306
GAT [38]	0.9635	0.9692	0.9739	0.9716	0.9900	0.0550
TF-SAGE	0.9729	0.9746	0.9833	0.9789	0.9939	0.0455

Within the graph based group, GCN reaches Macro-F1 0.9796, TF-SAGE reaches 0.9789, GraphSAGE reaches 0.9730, and GAT reaches 0.9716. TF-SAGE does not outperform the strongest clean baselines, yet it remains competitive even after adding trust filtering and uncertainty aware inference. Its AUROC of 0.9939 indicates sufficiently strong ranking quality, while its FPR of 0.0455 marks a tradeoff that would matter in operational environments sensitive to false alarms.

These results place TF-SAGE in a reasonable position: the model is not designed to maximize clean score alone, but its clean classification base is strong enough for the gaps under FSAA and GSAA to be interpreted as changes in stability and reliability rather than artifacts of weak clean performance.

The three slices in Fig. 7 clarify the structure of the clean results for a representative subset of models: Accuracy, Macro-F1, and receiver operating characteristic (ROC). The first two panels keep a shared scale so that small nominal differences are not exaggerated, while the ROC panel adds a zoomed inset over the low FPR, high true positive rate (TPR) region because all curves are already concentrated near the upper left corner. Table 2 remains the full coverage source when all baselines need to be compared.

XGBoost remains the strongest clean reference, while GCN, GraphSAGE, and MLP follow closely on the nominal panels. TF-SAGE does not lead absolutely on clean data, but it still preserves competitive ranking quality in the ROC panel. This clean starting point is strong enough for the later gaps under attack to be interpreted as adversarial stability and reliability evidence rather than as artifacts of a weak nominal baseline.

5.2 Stability under FSAA and GSAA

FSAA and GSAA examine two different adversarial surfaces of graph based IDS. FSAA acts directly in feature space, where decision boundaries may be shifted by small but purposeful perturbations. GSAA acts in graph space, where edge insertions and removals can alter neighborhoods and propagate misleading signals through message passing. The two scenarios are therefore complementary: FSAA measures stability under feature manipulation, whereas GSAA measures stability of the model's structural dependence.

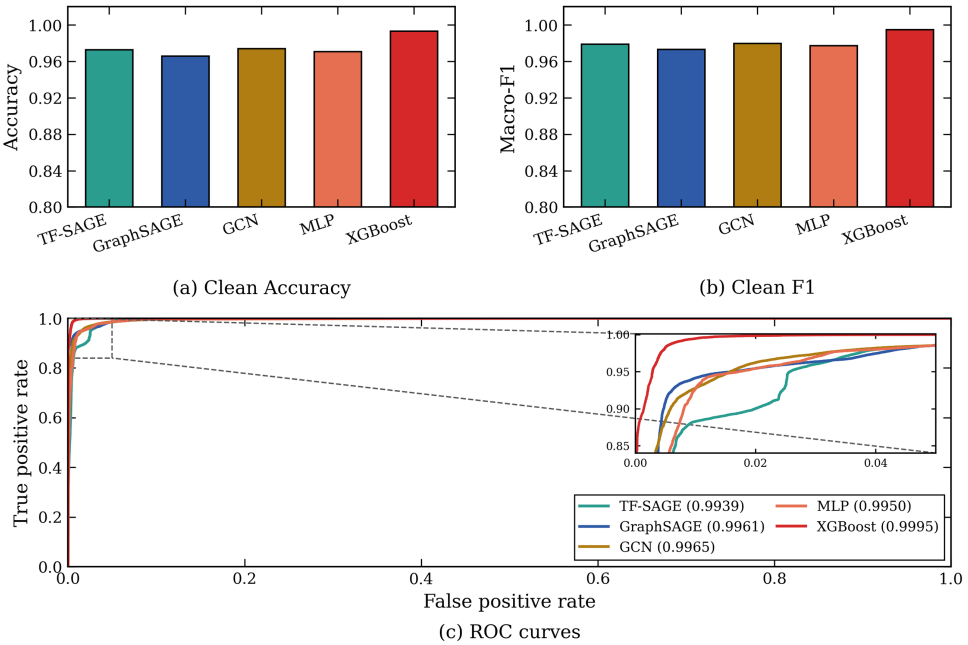


Figure 7: Clean performance on NF-ToN-IoT-v2.

On NF-ToN-IoT-v2, TF-SAGE preserves Macro-F1 0.9776 under FSAA with ASR 0.0048, a decrease of only 0.0013 from its clean Macro-F1 of 0.9789. This mild degradation contrasts sharply with the tabular models that are otherwise very strong on clean data. XGBoost drops from Macro-F1 0.9948 to 0.6169 with ASR 0.4983; LightGBM and RF drop from clean Macro-F1 0.9968 to 0.6882 and 0.6861, with ASR 0.3115 and 0.3070, respectively. LR starts from a lower clean Macro-F1 (0.8481) but still yields ASR 0.2916, indicating that a considerable fraction of attack samples can be pushed into the benign decision region. Clean nominal advantage therefore does not translate automatically into resistance to degradation when features are adversarially perturbed.

These FSAA results should be interpreted in relation to the training setup. TF-SAGE is evaluated as a complete robustness pipeline that includes the robust training term L_{rob}, whereas the tabular baselines in Table 3 are trained only on clean samples. Therefore, the comparison supports the conclusion that the full TF-SAGE pipeline remains stable under the specified FSAA setting, but it should not be read as evidence that the graph architecture alone accounts for the full performance gap. A natural follow-up control would be an adversarially trained MLP, which would allow the effect of robust training to be compared more directly across model families.

Table 3: Results under FSAA on NF-ToN-IoT-v2.

Model	Macro-F1 Clean	Macro-F1 FSAA	ΔMacro-F1	ASR
LR [3]	0.8481	0.7995	0.0486	0.2916
XGBoost [34]	0.9948	0.6169	0.3779	0.4983
LightGBM [35]	0.9968	0.6882	0.3086	0.3115
RF [3]	0.9968	0.6861	0.3107	0.3070
TF-SAGE	0.9789	0.9776	0.0013	0.0048

Under GSAA, the contrast shifts to structure dependent behavior. TF-SAGE reaches attacked Macro-F1 0.9048 with ASR 0.0941, whereas GraphSAGE, GCN, and GAT achieve attacked Macro-F1 values of 0.8156, 0.7964, and 0.6992, respectively. This gap shows that a standard GNN backbone does not automatically protect IDS when the neighborhood is manipulated. A single misleading edge may distort many nodes through repeated message passing, whereas trust filtering reduces the chance that fragile relations are allowed to propagate in the first place.

The six panels in Fig. 8 place FSAA and GSAA results side by side, allowing attacked Macro-F1, ASR, and Macro-F1 drop to be compared under a fixed model order.

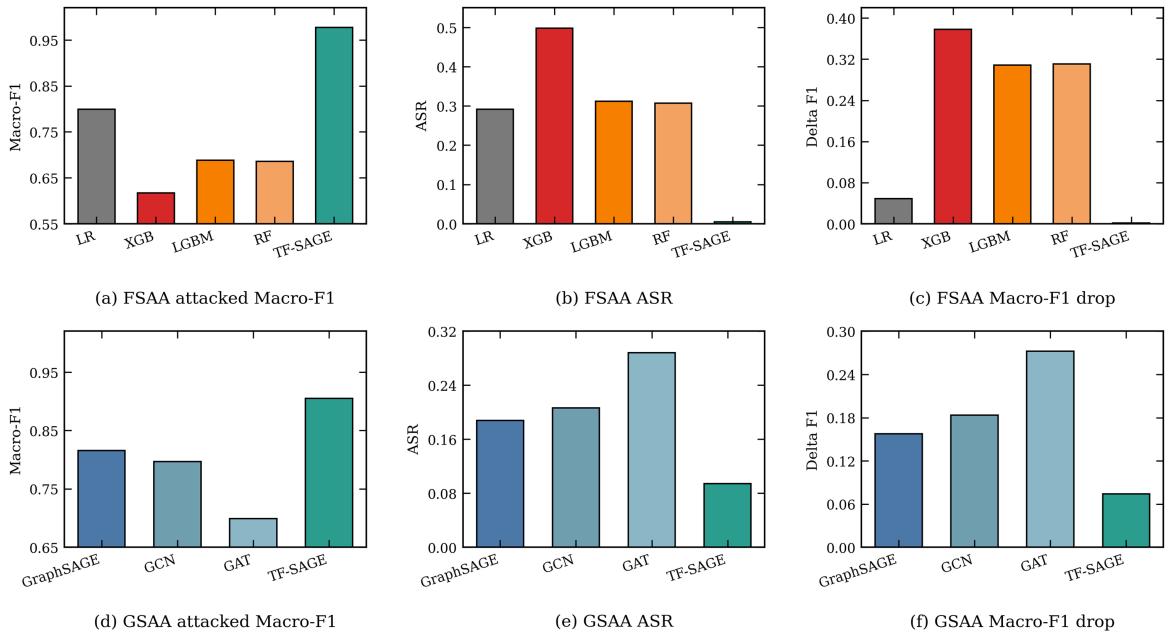


Figure 8: Degradation under FSAA and GSAA on NF-ToN-IoT-v2.

The shared pattern across FSAA and GSAA is that TF-SAGE does not rely on a single signal source. When features are perturbed, decisions are still supported by trust filtered neighborhoods and uncertainty aware inference; when graph connectivity is edited, low trust edges are weakened before they can be amplified through message passing.

Tables 3 and 4 summarize the two complementary attack surfaces. Table 3 focuses on FSAA for tabular classifiers and TF-SAGE, whereas Table 4 focuses on GSAA for graph based models and TF-SAGE. In both cases, the TF-SAGE advantage is reflected not only in attacked Macro-F1, but also in the lower fraction of attack samples pushed into the benign class.

Table 4: Results under GSAA on NF-ToN-IoT-v2.

Model	Macro-F1 Clean	Macro-F1 GSAA	ΔMacro-F1	ASR
GraphSAGE [36]	0.9730	0.8156	0.1574	0.1877
GCN [8]	0.9796	0.7964	0.1832	0.2065
GAT [38]	0.9716	0.6992	0.2724	0.2879
TF-SAGE	0.9789	0.9048	0.0741	0.0941

5.3 Sensitivity to Attack Budget

To examine degradation trends under increasing perturbation budgets, we vary epsilon over nine levels from 0 to 0.03. Table 5 reports the budget sensitivity comparison between TF-SAGE and plain GraphSAGE, while Fig. 9 extends the same budget range to a representative model group. This analysis complements the fixed GSAA evaluation in Table 4 by showing how attacked Macro-F1 and ASR change as epsilon increases. Table 4 reports a fixed GSAA evaluation under a structural attack configuration used for direct comparison among graph based models. Table 5 serves a different purpose: it reports a budget sensitivity analysis in which ε is varied to examine degradation trends. Therefore, the numerical differences between Tables 4 and 5 arise from their different evaluation roles and attack configurations. The two tables should be read as complementary evaluations rather than as repeated results from the same GSAA setting.

Table 5 shows that as the budget increases, both models degrade, but at very different rates. From the clean anchor to epsilon = 0.03, TF-SAGE drops from Macro-F1 0.9789 to 0.9738 and ASR rises from 0.0000 to 0.0076. Over the same range, plain GraphSAGE drops from Macro-F1 0.9730 to 0.9582 and ASR rises from 0.0000 to 0.0194. If only the nonzero segment from epsilon = 0.0075 to 0.03 is considered, TF-SAGE loses 0.0038 Macro-F1 while GraphSAGE loses 0.0126; the ASR increase is 0.0056 for TF-SAGE but 0.0165 for GraphSAGE.

Table 5: Controlled perturbation budget sweep on NF-ToN-IoT-v2.

ε	Model	Accuracy	Macro-F1	AUROC	ASR
0.00000	TF-SAGE	0.9729	0.9789	0.9939	0.0000
	GraphSAGE [36]	0.9658	0.9730	0.9961	0.0000
0.00375	TF-SAGE	0.9721	0.9783	0.9933	0.0010
	GraphSAGE [36]	0.9644	0.9720	0.9954	0.0014
0.00750	TF-SAGE	0.9712	0.9776	0.9927	0.0020
	GraphSAGE [36]	0.9630	0.9708	0.9945	0.0029
0.01125	TF-SAGE	0.9702	0.9768	0.9921	0.0031
	GraphSAGE [36]	0.9615	0.9696	0.9934	0.0045
0.01500	TF-SAGE	0.9694	0.9762	0.9915	0.0040
	GraphSAGE [36]	0.9600	0.9685	0.9920	0.0061
0.01875	TF-SAGE	0.9686	0.9756	0.9907	0.0049
	GraphSAGE [36]	0.9581	0.9669	0.9903	0.0081
0.02250	TF-SAGE	0.9678	0.9750	0.9899	0.0058
	GraphSAGE [36]	0.9556	0.9649	0.9883	0.0108
0.02625	TF-SAGE	0.9670	0.9744	0.9891	0.0067
	GraphSAGE [36]	0.9516	0.9617	0.9858	0.0149
0.03000	TF-SAGE	0.9663	0.9738	0.9883	0.0076
	GraphSAGE [36]	0.9473	0.9582	0.9828	0.0194

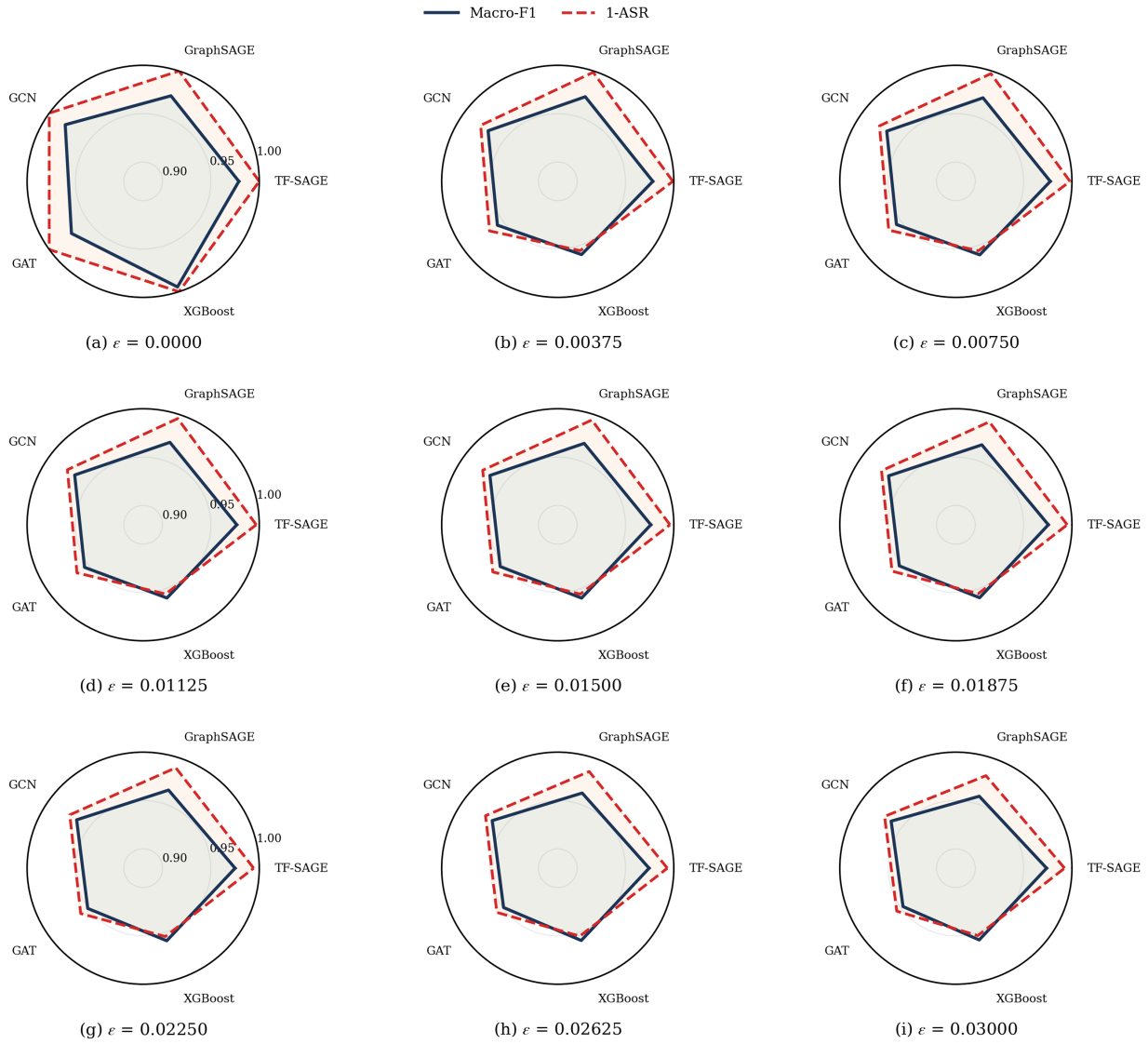


Figure 9: Controlled perturbation budget profile of the five models.

The key point is not only that TF-SAGE remains better at every value of epsilon, but that the gap itself widens under stronger perturbation. The Macro-F1 gap between TF-SAGE and GraphSAGE grows from 0.0068 at epsilon = 0.0075 to 0.0156 at epsilon = 0.03, while the ASR gap grows from 0.0009 to 0.0118. This pattern complements the main attack results: the benefit of TF-SAGE is not limited to a favorable operating point, but persists as the perturbation budget increases in this auxiliary sweep.

Fig. 9 uses the same nine budget levels but expands the visualization to five models, using attacked Macro-F1 and 1-ASR as the two plotted axes.

In the clean panel ($\varepsilon = 0$), XGBoost provides the strongest nominal starting point, with GCN and TF-SAGE immediately behind it. As ε increases, TF-SAGE maintains higher Macro-F1 and lower ASR: its Macro-F1 decreases only from 0.9789 to 0.9738, while 1-ASR remains close to 0.9924 even at the largest budget. The gap to GraphSAGE, GAT, and XGBoost widens near the end of the sweep. This pattern complements the earlier attack results: the TF-SAGE advantage is not tied to a single favorable budget within the auxiliary sweep, but remains visible across the full perturbation range under study.

5.4 Mechanism Evidence: Calibration and Neighborhood Recovery

The mechanism analysis focuses on two signals close to how TF-SAGE forms its decisions: the reliability of predictive probabilities and the stability of neighborhoods after trust filtering. Attacked Macro-F1 and ASR have already described the overall performance loss; the more important question here is why the model continues to behave stably when both confidence and input structure are stressed by attack.

The TF-SAGE reliability diagram in Fig. 10 shows where confidence diverges from empirical accuracy across the probability range, rather than reducing the evidence to a single aggregate statistic.

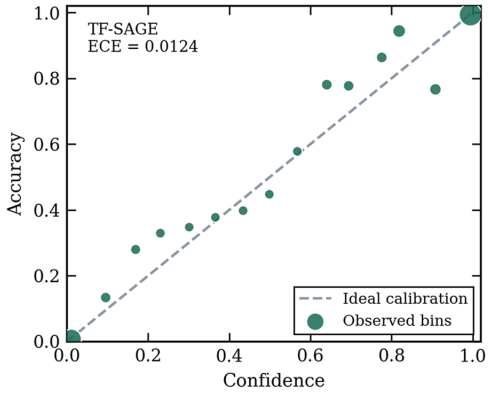


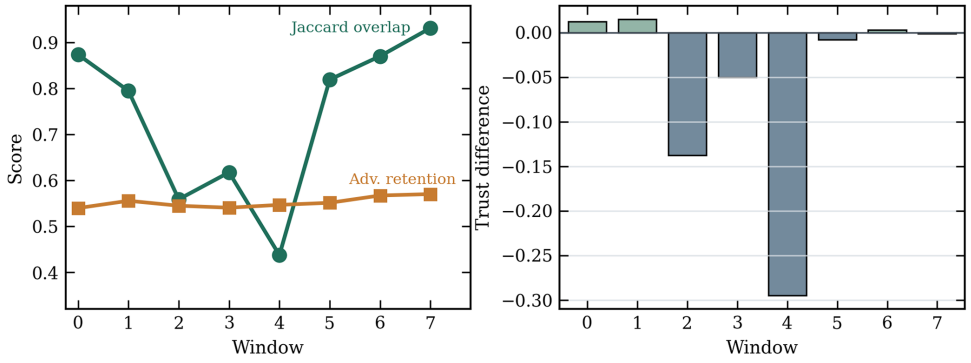
Figure 10: TF-SAGE reliability diagram.

With ECE = 0.0124 over 317,440 samples, the reliability signal is extracted from a sufficiently large validation slice rather than from a few isolated examples. Most predictive mass remains concentrated at low confidence and very high confidence extremes where the observed accuracy still tracks the reported probability closely. This indicates that TF-SAGE's predicted probabilities remain operationally informative, rather than merely accompanying high accuracy.

The remaining mismatch is concentrated mainly in middle to high confidence bins such as the 0.60–0.93 region, where the boundary between benign and attack traffic becomes more ambiguous. This mismatch pattern is less concerning than calibration failure across the full probability range, because it suggests that uncertainty aware inference still carries discriminative value exactly where the cases become difficult. For IDS deployment, that point matters almost as much as attacked Macro-F1 itself: a model may retain strong scores under attack while still becoming operationally risky if it turns overconfident in ambiguous regions.

The second mechanism dimension lies in the neighborhood dynamics under GSAA. Fig. 11 provides a window level view of the overlap between clean and filtered neighborhoods, as well as the proportion of candidate edges retained after trust filtering reorganizes the structure. This evidence explains filter behavior at the window level rather than replacing the full scale benchmark results.

Avg Jaccard overlap = 0.7379 indicates that a substantial core of the clean neighborhood still survives after GSAA, while Avg adversarial retention = 0.5520 shows that trust filtering does not react by deleting the graph wholesale. More than half of the candidate edges are still retained after refiltering, which means TF-SAGE tries to preserve enough structure for message passing to remain useful rather than defending itself by cutting relations indiscriminately.



(a) Jaccard overlap and adversarial-edge retention (b) Trust-score shift

Figure 11: Neighborhood recovery under GSAA.

Table 6 evaluates the component removals under the same GSAA setting used in Table 4. The No-robust-objective row evaluates the role of adversarial training by removing the robust objective from the training loss. The No-Trust setting evaluates trust filtering by disabling trust based edge rejection, implemented by setting $\theta_f = 0$ while keeping the remaining graph construction pipeline unchanged. The No-stability-regularization row evaluates the stability and calibration related part of the uncertainty aware reasoning path by removing the regularizer that constrains local representation drift under perturbation. The full TF-SAGE model remains the strongest setting among the evaluated variants, with attacked Macro-F1 of 0.9048 and ASR of 0.0941. Among the ablations, the No-Trust setting produces the largest degradation, reducing attacked accuracy to 0.7062 and attacked Macro-F1 to 0.7845, while increasing ASR to 0.2557. These results show that trust filtered graph construction makes a clear contribution under this GSAA setting, while adversarial training and stability regularization also contribute to the overall robustness of the complete TF-SAGE pipeline.

Table 6: Component ablation under direct GSAA.

Variant	Accuracy Clean	Macro-F1 Clean	AUROC Clean	Accuracy GSAA	Macro-F1 GSAA	ASR
TF-SAGE (full)	0.9729	0.9789	0.9939	0.8867	0.9048	0.0941
No-robust objective	0.9242	0.9411	0.9699	0.7217	0.7861	0.2329
No-Trust setting	0.9374	0.9519	0.9767	0.7062	0.7845	0.2557
No-stability regularization	0.9268	0.9431	0.9723	0.7538	0.8120	0.2001

These direct GSAA ablation results support the design of TF-SAGE as a coupled robustness pipeline rather than a collection of interchangeable modules. Robust training improves adversarial consistency, stability regularization supports uncertainty-aware local smoothness and calibration-facing behavior, and trust-filtered graph construction preserves reliable message-passing neighborhoods under perturbation. The ablation evidence therefore suggests that the observed robustness is not explained by the GraphSAGE backbone alone, but by the interaction between trust-aware graph construction, adversarially regularized learning, and stability-aware inference.

More importantly, the trust score shift is not uniform across windows: some windows lose trust clearly, some remain nearly unchanged, and some even rise slightly after refiltering. This pattern fits the role of TrustFilter better than a rigid thresholding rule. It suggests that the filter responds to local neighborhood

conditions rather than forcing every window into the same degree of contraction. This does not imply that every attacked neighborhood is preserved equally; rather, TF-SAGE retains a sufficiently large structural core while redistributing trust under graph disturbance.

These two evidence layers support the same mechanism argument from complementary directions. The reliability diagram shows that the combined objective in Eq. (15) does not drive the model into overconfident probability outputs, while the neighborhood recovery slice shows that trust filtering in Algorithm 1 does not destroy structure but reselects a more stable neighborhood core before message passing begins. The calibration and neighborhood recovery analyses complement the component ablation in Table 6 by showing how TF-SAGE maintains both prediction reliability and neighborhood stability under attack. The performance gap under attack therefore has a more direct explanation: TF-SAGE controls both the reliability of its predictions and the quality of the neighborhood structure passed into representation learning.

5.5 Independent Confirmation on CICIIoT2025

Although NF-ToN-IoT-v2 is the main benchmark, CICIIoT2025 provides an independent confirmation benchmark with a different and more difficult distribution. The purpose of this benchmark is to examine whether the stability advantage of TF-SAGE remains visible beyond the main dataset under the same FSAA and GSAA evaluation logic. Under FSAA, TF-SAGE reaches attacked Macro-F1 0.8308 and ASR 0.1016, whereas the tabular baselines degrade more strongly: XGBoost falls to 0.7655 with ASR 0.2145, LightGBM to 0.6742 with ASR 0.3775, and RF to 0.6902 with ASR 0.3375. This reproduces the same pattern seen on the main benchmark: nominal clean advantage does not automatically translate into stability when feature space perturbation is introduced.

On the GSAA side, CICIIoT2025 further confirms the structural stability advantage of TF-SAGE. GraphSAGE, GCN, and GAT drop to attacked Macro-F1 values of 0.7528, 0.6108, and 0.7518, whereas TF-SAGE maintains 0.8797 with ASR only 0.0351. This gap indicates that when neighborhood structure is distorted, TF-SAGE preserves substantially more attack detection behavior than standard graph backbones. The auxiliary benchmark therefore reinforces the same central argument obtained from NF-ToN-IoT-v2: the value of TF-SAGE lies in controlling graph construction and message passing under perturbation, not merely in clean performance.

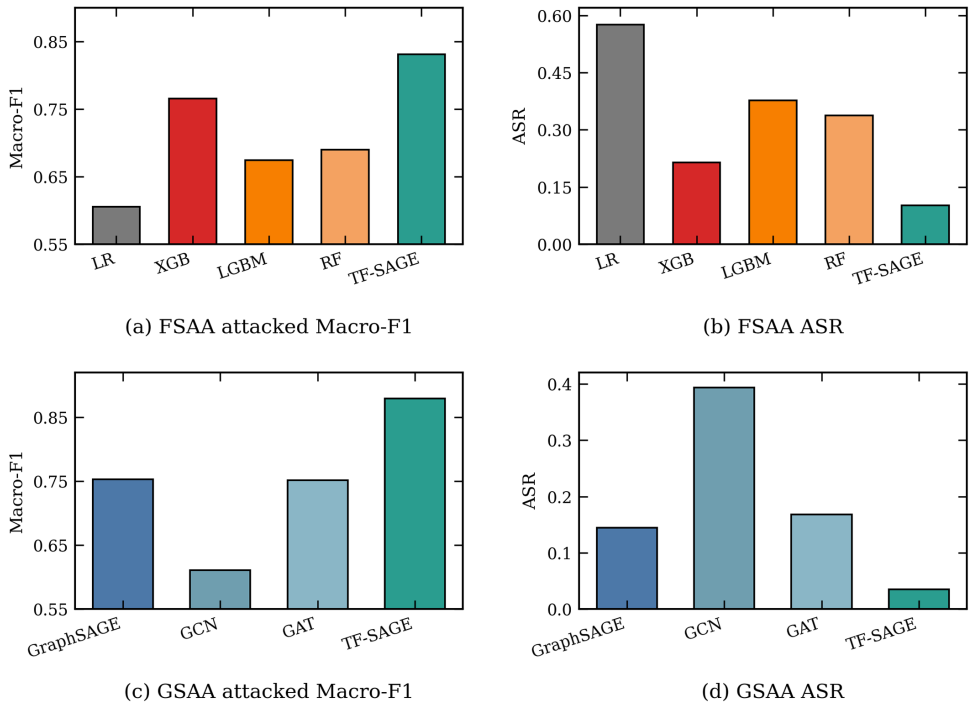
Table 7 reports the CICIIoT2025 confirmation results under the same evaluation logic used for the main benchmark. FSAA evaluates feature space perturbation for tabular models and TF-SAGE, whereas GSAA evaluates structural perturbation for graph based models and TF-SAGE. This setup allows CICIIoT2025 to serve as an independent benchmark for checking whether the stability advantage of TF-SAGE persists beyond NF-ToN-IoT-v2.

Fig. 12 presents the same result package in a compact 2×2 layout, with the two upper panels for FSAA and the two lower panels for GSAA. This layout makes it easier to check the consistency of the trend on the confirmation benchmark without shifting the focus away from the main results.

Under FSAA, TF-SAGE preserves stronger attacked Macro-F1 and lower ASR than the tabular baselines. Under GSAA, the separation from standard graph backbones is also clear, with TF-SAGE retaining higher attacked Macro-F1 and substantially lower ASR. These results show that the same stability pattern observed on NF-ToN-IoT-v2 also appears on CICIIoT2025, supporting the conclusion that TF-SAGE improves robustness under both feature space and graph space perturbations.

Table 7: Confirmation results on CICIIoT2025.

Attack Space	Model	Macro-F1 Clean	Macro-F1 Adv.	ΔMacro-F1	ASR
FSAA	LR [3]	0.8658	0.6052	0.2606	0.5754
FSAA	XGBoost [34]	0.9523	0.7655	0.1868	0.2145
FSAA	LightGBM [35]	0.9552	0.6742	0.2810	0.3775
FSAA	RF [3]	0.9630	0.6902	0.2728	0.3375
FSAA	TF-SAGE	0.9254	0.8308	0.0946	0.1016
GSAA	GraphSAGE [36]	0.9031	0.7528	0.1503	0.1449
GSAA	GCN [8]	0.9273	0.6108	0.3165	0.3934
GSAA	GAT [38]	0.9463	0.7518	0.1945	0.1679
GSAA	TF-SAGE	0.9168	0.8797	0.0371	0.0351



(a) FSAA attacked Macro-F1

(b) FSAA ASR

(c) GSAA attacked Macro-F1

(d) GSAA ASR

Figure 12: Confirmation results on CICIIoT2025.

6 Discussion

The experimental results point to an important conclusion: in IoT IDS, clean score is a necessary condition, but not a sufficient one. On the main benchmark, tabular baselines may still achieve very high Accuracy and AUROC on clean data, yet when feature space is actively manipulated through FSAA, their attacked Macro-F1 degradation and ASR increase become much larger than those of TF-SAGE. This means that an IDS judged only by clean score may still lack defensive value precisely at the attack surface most relevant to real operation. To provide a statistical reliability check without mixing different checkpoints or evaluation subsets, Table 8 reports Wilson confidence intervals for selected ASR values already reported in Tables 3, 4, and 7. The intervals are computed from aggregate attack flip counts and therefore quantify

uncertainty in the reported attack success rates. They do not estimate training-seed variance or support unpaired statistical-superiority claims.

Table 8: Wilson confidence intervals for selected ASR outcome.

Dataset	Scenario	Model	Metric	Estimate	95% CI
NF-ToN-IoT-v2	FSAA	TF-SAGE	ASR	0.0048	[0.0046, 0.0051]
NF-ToN-IoT-v2	GSAA	TF-SAGE	ASR	0.0941	[0.0918, 0.0964]
NF-ToN-IoT-v2	GSAA	GraphSAGE [36]	ASR	0.1877	[0.1847, 0.1908]
CICIIoT2025	FSAA	TF-SAGE	ASR	0.1016	[0.1002, 0.1029]
CICIIoT2025	GSAA	TF-SAGE	ASR	0.0351	[0.0323, 0.0381]
CICIIoT2025	GSAA	GraphSAGE [36]	ASR	0.1449	[0.1394, 0.1506]

The resulting intervals are narrow for the selected ASR outcomes, suggesting that the reported attack success estimates are not driven by small test samples or unstable aggregate flip counts. On NF-ToN-IoT-v2, the GSAA ASR interval of TF-SAGE remains clearly below that of GraphSAGE, which reinforces the observed robustness pattern at the flip-rate level under the evaluated GSAA setting. At the same time, these results are interpreted conservatively: they complement the main tables by bounding finite-sample uncertainty, but they do not replace a future multi-seed variance study.

The GSAA results push this argument further by showing that graph construction is not a neutral preprocessing step, but a decisive component of model stability. When the neighborhood is purposefully edited, GraphSAGE, GCN, and GAT all degrade more severely in attacked Macro-F1 and yield higher ASR, whereas TF-SAGE preserves a narrower performance loss. This gap is consistent with the mechanism evidence: predictive probabilities remain well calibrated at scale, while trust filtering does not cut the graph aggressively but preserves a substantial structural core after the neighborhood is edited. In this sense, the main contribution of TF-SAGE is not making the backbone deeper, but transforming graph construction from a heuristic into a verifiable structural control layer.

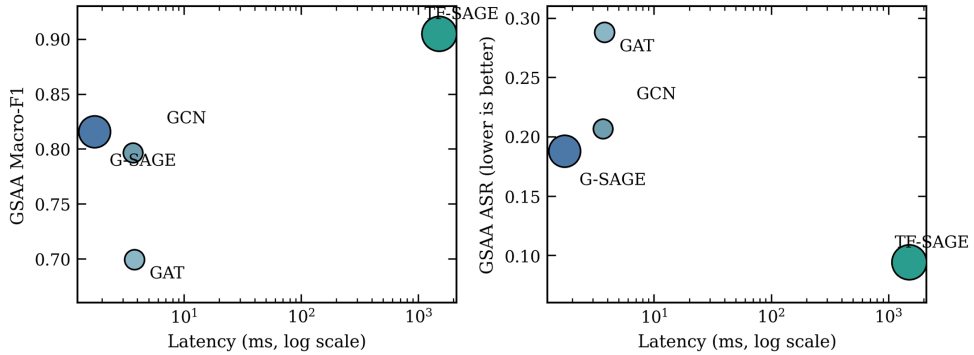
The robustness results should be interpreted within the FSAA and GSAA threat models defined in this study. FSAA examines whether the IDS remains stable when flow features are perturbed, whereas GSAA examines whether graph based inference remains stable when neighborhood relations are perturbed. These evaluations support the robustness claims within the defined feature space and graph space settings, but they should not be read as a complete assessment of adversarial robustness for GNN based IDS. A broader evaluation with additional adaptive GNN attack baselines remains as direction for future work.

The deployment tradeoff is shown by placing attacked Macro-F1 and ASR next to latency within the graph model family.

If latency alone is considered, TF-SAGE requires more computation because graph updating, trust filtering, and stochastic inference add extra processing to the inference pipeline. Fig. 13 places this cost next to the GSAA robustness results, where TF-SAGE achieves attacked Macro-F1 of 0.9048 and ASR of 0.0941.

These results should therefore be interpreted as an operational tradeoff rather than a cost free improvement. Faster graph baselines such as GraphSAGE [36] and GCN [8] retain a speed advantage, but they show greater sensitivity to structural corruption under GSAA. From a practical perspective, TF-SAGE is more suitable for IDS layers where stability under attack matters more than minimum per-flow latency, such as centralized monitoring gateways, edge analysis clusters with stable compute budgets, or settings in which false negatives on malicious traffic are more costly than the added computation. In contrast, very low-latency

environments may still favor lighter models, especially when they are deployed as one layer within a broader defense stack. The practical choice therefore depends on how a deployment balances latency, reliability, and acceptable security loss.



(a) Macro-F1 retention versus latency (b) Attack success versus latency

Figure 13: GSAA Macro-F1 and inference cost tradeoff.

In deployment, the uncertainty output is intended to support analyst triage rather than to replace analyst judgment. High-confidence malicious predictions can be prioritized for rapid review or automated response when site policy allows it, while low-confidence or high-entropy cases can be routed to manual inspection, correlated with additional telemetry, or held for delayed confirmation. In this role, uncertainty helps security teams avoid treating ambiguous predictions as fully reliable alerts and allocate review effort to traffic windows that require closer inspection. This operational interpretation also defines the scope of the present classification setting. Although both NF-ToN-IoT-v2 and CICIIoT2025 contain multiple attack families, the experiments in this paper focus on binary benign-vs.-attack detection. This setting is suitable for examining whether graph construction, message passing, and uncertainty reporting remain stable when the IDS is exposed to feature space and graph space perturbations. Multiclass attack-family attribution is a valuable deployment task, but it requires a different label definition, class-imbalance treatment, and robustness evaluation protocol. The present results should therefore be interpreted as evidence for binary intrusion detection under the evaluated FSAA and GSAA settings, while multiclass robustness analysis remains a future extension.

The confirmation results on CICIIoT2025 further suggest that the attack stability interpretation is not tied to a single benchmark. Under the same evaluation logic, TF-SAGE continues to preserve smaller degradation and lower ASR after the data distribution changes, strengthening the central argument that graph construction should be explicitly controlled in graph based IDS. The present experiments evaluate adversarial stability under feature space and graph space perturbations, which is related to robustness but is not the same as zero-day or unseen attack-family generalization. Therefore, the reported results should not be read as evidence that attack families absent from training can be detected reliably. Zero-day intrusion detection requires a dedicated problem formulation, including an unseen-family evaluation protocol, appropriate training objectives, and model mechanisms designed to recognize traffic patterns that are not represented in the known attack classes. Developing and evaluating such zero-day oriented graph based IDS models remains an important direction for future work. More broadly, graph based IoT IDS should be evaluated by placing clean quality, feature space stability, graph space stability, predictive reliability, and unseen-family generalization in a coherent evaluation framework; if any of these axes is missing, the resulting conclusion about defensive value may still be incomplete.

7 Conclusion

This study presents TF-SAGE as a graph based IDS framework that protects graph construction rather than treating it as a fixed preprocessing step. Across NF-ToN-IoT-v2 and CICIIoT2025, the results show that clean detection performance alone is not sufficient to characterize IDS value. Although TF-SAGE is not always the top clean score model, it maintains more stable behavior than the evaluated baselines under both FSAA and GSAA, with lower degradation and lower ASR under the defined threat models.

The mechanism evidence further supports this interpretation. The budget sweep shows that the stability pattern persists as the perturbation budget increases, while calibration and neighborhood recovery analyses indicate that the observed robustness is not reducible to a single summary score. By applying trust filtering before message passing and coupling it with uncertainty aware inference, TF-SAGE reduces dependence on fragile neighborhoods and avoids treating unstable predictions as fully reliable decisions. These findings suggest that graph based IDS should be evaluated not only by nominal accuracy, but also by how well graph formation, information propagation, and prediction reliability remain stable under attack.

However, there are some limitations to the scope of the current evaluation scope. The experiments are conducted under the defined FSAA and GSAA protocols and focus on binary benign-vs.-attack detection; they do not establish complete adversarial robustness, multiclass attack-family attribution, or zero-day detection. Future work should therefore extend the evaluation to broader adaptive GNN attacks, larger and more diverse benchmarks, multiclass and unseen-family protocols, multi-seed variance analysis, and finer grained studies of uncertainty use in analyst triage. These directions would help determine whether TF-SAGE and its successors can serve not only as high performing models in one experimental setting, but also as a broader design framework for resilience oriented graph based IDS.

Acknowledgement: Not applicable.

Funding Statement: This research was partly supported by the National Science and Technology Council, Taiwan, with grant numbers NSTC 115-2622-8-992-005-TD1 and NSTC 114-2622-8-992-007-TD1.

Author Contributions: Chin-Shiuh Shieh supervised the study. Thanh-Tuan Nguyen and Thanh-Lam Nguyen developed the methodology, conducted the experiments, curated the data, and prepared the original draft. Mong-Fong Horng contributed to formal analysis and manuscript review. Xuan-Huy Nguyen and Chau-Tan-Phat Le contributed to validation, data checking, and manuscript preparation. All authors reviewed and approved the final version of the manuscript.

Availability of Data and Materials: The NF-ToN-IoT-v2 and CICIIoT2025 benchmarks used in this study are publicly available from their original sources. The processed tables, figure assets, and manuscript-supporting materials used in this study are available from the corresponding authors upon reasonable request.

Ethics Approval: Not applicable.

Conflicts of Interest: Given his role as Editorial Board Members and Guest Editor of this journal, Chin-Shiuh Shieh had no involvement in the peer review of this article and had no access to information regarding its peer review. Given his role as Guest Editor of this journal, Thanh-Tuan Nguyen had no involvement in the peer review of this article and had no access to information regarding its peer review. Full responsibility for the editorial process for this article was delegated to another journal editor. The authors declare no other conflicts of interest.

References

1. Goranin N, Hora SK, Čenys HA. A bibliometric review of intrusion detection research in IoT: evolution, collaboration, and emerging trends. Electronics. 2024;13(16):3210. doi:10.3390/electronics13163210.
2. Xu B, Sun L, Mao X, Ding R, Liu C. IoT intrusion detection system based on machine learning. Electronics. 2023;12(20):4289. doi:10.3390/electronics12204289.
3. Ashraf J, Raza GM, Kim BS, Wahid A, Kim HY. Making a real-time IoT network intrusion-detection system (INIDS) using a realistic BoT-IoT dataset with multiple machine-learning classifiers. Appl Sci. 2025;15(4):2043. doi:10.3390/app15042043.
4. Shi L, Yang Q, Gao L, Ge H. An ensemble system for machine learning IoT intrusion detection based on enhanced artificial hummingbird algorithm. J Supercomput. 2024;81(1):110. doi:10.1007/s11227-024-06475-1.
5. Ali MA, Al-Sharafi SAH. Intrusion detection in IoT networks using machine learning and deep learning approaches for MitM attack mitigation. Discov Internet Things. 2025;5(1):48. doi:10.1007/s43926-025-00104-w.
6. He K, Kim DD, Asghar MR. Adversarial machine learning for network intrusion detection systems: a comprehensive survey. IEEE Commun Surv Tutorials. 2023;25(1):538–66. doi:10.1109/comst.2022.3233793.
7. Sharma S, Chen Z. A systematic study of adversarial attacks against network intrusion detection systems. Electronics. 2024;13(24):5030. doi:10.3390/electronics13245030.
8. Zhong M, Lin M, Zhang C, Xu Z. A survey on graph neural networks for intrusion detection systems: methods, trends and challenges. Comput Secur. 2024;141(3):103821. doi:10.1016/j.cose.2024.103821.
9. Caville E, Lo WW, Layeghy S, Portmann M. Anomal-E: a self-supervised network intrusion detection system based on graph neural networks. Knowl Based Syst. 2022;258(1):110030. doi:10.1016/j.knosys.2022.110030.
10. Altaf T, Wang X, Ni W, Yu G, Liu RP, Braun R. A new concatenated multigraph neural network for IoT intrusion detection. Internet Things. 2023;22(1):100818. doi:10.1016/j.iot.2023.100818.
11. Gao M, Wu L, Li Q, Chen W. Anomaly traffic detection in IoT security using graph neural networks. J Inf Secur Appl. 2023;76(5):103532. doi:10.1016/j.jisa.2023.103532.
12. Deng P, Huang Y. Edge-featured multi-hop attention graph neural network for intrusion detection system. Comput Secur. 2025;148(4):104132. doi:10.1016/j.cose.2024.104132.
13. Ngo T, Yin J, Ge YF, Wang H. Optimizing IoT intrusion detection—a graph neural network approach with attribute-based graph construction. Information. 2025;16(6):499. doi:10.3390/info16060499.
14. Zhao K, Kang Q, Song Y, She R, Wang S, Tay WP. Adversarial robustness in graph neural networks: a Hamiltonian approach. In: Proceedings of the Neural Information Processing Systems 36; 2023 Dec 10–16; New Orleans, LA, USA. doi:10.52202/075280-0148.
15. Hsu HH, Shen Y, Tomani C, Cremers D. What makes graph neural networks miscalibrated? In: Proceedings of the Neural Information Processing Systems 35; 2022 Nov 28–Dec 9; New Orleans, LA, USA. doi:10.52202/068431-1001.
16. Chaurasia N, Ram M, Verma P, Mehta N, Bharot N. A federated learning approach to network intrusion detection using residual networks in industrial IoT networks. J Supercomput. 2024;80(13):18325–46. doi:10.1007/s11227-024-06153-2.
17. Arreche O, Guntur T, Abdallah M. XAI-IDS: toward proposing an explainable artificial intelligence framework for enhancing network intrusion detection systems. Appl Sci. 2024;14(10):4170. doi:10.3390/app14104170.
18. Kikissagbe BR, Adda M. Machine learning-based intrusion detection methods in IoT systems: a comprehensive review. Electronics. 2024;13(18):3601. doi:10.3390/electronics13183601.
19. Berhili M, Chaieb O, Benabdellah M. Intrusion detection systems in IoT based on machine learning: a state of the art. Procedia Comput Sci. 2024;251:99–107. doi:10.1016/j.procs.2024.11.089.
20. Musthafa MB, Huda S, Kodera Y, Ali MA, Araki S, Mwaura J, et al. Optimizing IoT intrusion detection using balanced class distribution, feature selection, and ensemble machine learning techniques. Sensors. 2024;24(13):4293. doi:10.3390/s24134293.
21. Schrötter M, Niemann A, Schnor B. A comparison of neural-network-based intrusion detection against signature-based detection in IoT networks. Information. 2024;15(3):164. doi:10.3390/info15030164.

22. Rabie OBJ, Selvarajan S, Hasanin T, Alshareef AM, Yogesh CK, Uddin M. A novel IoT intrusion detection framework using decisive red fox optimization and descriptive back propagated radial basis function models. Sci Rep. 2024;14(1):386. doi:10.1038/s41598-024-51154-z.
23. Kim H, Park S, Hong H, Park J, Kim S. A transferable deep learning framework for improving the accuracy of Internet of Things intrusion detection. Future Internet. 2024;16(3):80. doi:10.3390/fi16030080.
24. Cui B, Chai Y, Yang Z, Li K. Intrusion detection in IoT using deep residual networks with attention mechanisms. Future Internet. 2024;16(7):255. doi:10.3390/fi16070255.
25. Yaras S, Dener M. IoT-based intrusion detection system using new hybrid deep learning algorithm. Electronics. 2024;13(6):1053. doi:10.3390/electronics13061053.
26. Tseng SM, Wang YQ, Wang YC. Multi-class intrusion detection based on transformer for IoT networks using CIC-IoT-2023 dataset. Future Internet. 2024;16(8):284. doi:10.3390/fi16080284.
27. Tran DH, Park M. FN-GNN: a novel graph embedding approach for enhancing graph neural networks in network intrusion detection systems. Appl Sci. 2024;14(16):6932. doi:10.3390/app14166932.
28. Yin L, Chen W, Luo X, Yang H. Efficient large-scale IoT botnet detection through GraphSAINT-based subgraph sampling and graph isomorphism network. Mathematics. 2024;12(9):1315. doi:10.3390/math12091315.
29. Le HD, Park M. Enhancing multi-class attack detection in graph neural network through feature rearrangement. Electronics. 2024;13(12):2404. doi:10.3390/electronics13122404.
30. Wardana AA, Kołaczek G, Sukarno P. Lightweight, trust-managing, and privacy-preserving collaborative intrusion detection for Internet of Things. Appl Sci. 2024;14(10):4109. doi:10.3390/app14104109.
31. Alabbadi A, Bajaber F. An intrusion detection system over the IoT data streams using eXplainable Artificial Intelligence (XAI). Sensors. 2025;25(3):847. doi:10.3390/s25030847.
32. Sarhan M, Layeghy S, Portmann M. Towards a standard feature set for network intrusion detection system datasets. Mob Netw Appl. 2022;27(1):357–70. doi:10.1007/s11036-021-01843-0.
33. Firouzi A, Dadkhah S, Maret SA, Ghorbani AA. DataSense: a real-time sensor-based benchmark dataset for attack analysis in IIoT with multi-objective feature selection. Electronics. 2025;14(20):4095. doi:10.3390/electronics14204095.
34. Hu Y, Xiao K, Luo L, Chen L. An XGBoost-based intrusion detection framework with interpretability analysis for IoT networks. Appl Sci. 2026;16(2):980. doi:10.3390/app16020980.
35. Chen W, Yang H, Yin L, Luo X. Large-scale IoT attack detection scheme based on LightGBM and feature selection using an improved salp swarm algorithm. Sci Rep. 2024;14(1):19165. doi:10.1038/s41598-024-69968-2.
36. Saidane S, Telch F, Shahin K, Granelli F. Deep GraphSAGE enhancements for intrusion detection: analyzing attention mechanisms and GCN integration. J Inf Secur Appl. 2025;90(5):104013. doi:10.1016/j.jisa.2025.104013.
37. Wang Y, Han Z, Du Y, Li J, He X. BS-GAT: a network intrusion detection system based on graph neural network for edge computing. Cybersecurity. 2025;8(1):27. doi:10.1186/s42400-024-00296-8.
38. Ahanger AS, Khan SM, Masoodi F, Salau AO. Advanced intrusion detection in Internet of Things using graph attention networks. Sci Rep. 2025;15(1):9831. doi:10.1038/s41598-025-94624-8.