The ISC Int'l Journal of Information Security
ISeCure

July 2026, Volume 18, Number 2 (pp. 1–14)
http://www.isecure-journal.org

GAT-AID: A Graph Attention-Based Dual-Branch Framework for Scalable Anomaly and Intrusion Detection

Nitin Wankhade [1,*], and Anand Khandare [2]
[1] Thakur College of Engineering & Technology, Computer Engineering Department, Mumbai, Maharashtra, India
[2] Thakur College of Engineering & Technology, Computer Engineering Department, Mumbai, Maharashtra, India

ARTICLE INFO.

Article history:
Received:
Revised:
Accepted:
Published Online:

Keywords:
Anomaly Detection, Autoencoder, Graph Neural Networks, Intrusion Detection Systems, Zero-Day Attacks

Type:

doi:

Abstract

Intrusion Detection Systems (IDS) are vital for defending modern networks against emerging cyber threats, including zero-day attacks. In this article, we introduce GAT-AID (Graph Attention-based Anomaly and Intrusion Detection), an IDS architecture that integrates Graph Attention Networks (GATs), Multi-Layer Perceptron (MLP) classifiers, and Autoencoders. The proposed methodology represents network traffic as a graph, allowing GAT to extract complex node-wise associations across traffic flows. The embeddings generated are further processed through a dual-branch architecture, an MLP-based classifier for identifying known attack types, and an Autoencoder-based anomaly detector for flagging zero-day intrusions. The proposed GAT-AID methodology is evaluated on two widely used benchmark datasets, namely CICIDS2017 and UNSW-NB15. The experiment results demonstrate that it outperforms conventional IDS baselines, including SVM, Random Forest, CNN, and GCN models, achieving higher detection rates, improved robustness against unseen threats, and greater adaptability to evolving network environments. These findings suggest that GAT-AID is an effective and scalable solution for intelligent, real-time intrusion detection.

1 Introduction

Amid the continually evolving domain of information security, ensuring the resilience of interconnected digital environments against network attacks has become a primary challenge. Modern threat methodologies, particularly zero-day vulnerabilities, routinely evade conventional signature-reliant detection paradigms. This persistent challenge calls for the deployment of advanced, behavior-centric analytical frameworks that detect novel incursions solely through abnormal activity profiling.

Traditional ML models struggle to generalize to heterogeneous datasets and lack the ability to derive spatial associations within intricate network traffic. To tackle this issue, graph-based deep learning approaches have emerged as potential options for representing and gaining insight from structured network flow data. Among these, Graph Attention Networks (GATs) have shown greater ability to represent localized interactions by applying attention mechanisms at the node level. However, existing GAT-based approaches often lack scalability, ignore batch dynamics, or fail to handle high-dimensional traffic representations effectively.

* Corresponding author.
Email addresses: nitin.wankhade@gmail.com, anand.khandare@gmail.com
ISSN: 2008-2045

The existing GNN-based IDS using Autoencoder-based approaches has some drawbacks: (1) It does not address the effective zero-day attack detection through unsupervised anomaly scoring, (2) It is inefficient in handling batch-wise high-dimensional and heterogeneous network traffic data, and (3) There is an absence of supervised and unsupervised learning paradigms by using dual-branch architectures in IDS. These gaps necessitate the proposed IDS methodology, GAT-AID-IDS.

Graph Attention Networks (GATs) offer a more adaptable approach to learning node representations from graph-structured data. Rather than applying a fixed method to combine neighbouring information, as traditional Graph Convolutional Networks (GCNs) do, GATs use learned attention scores [1]. This approach enables the models to assign greater importance to relevant neighbouring nodes during feature representations. This promotes the derivation of more intricate, situationally informed feature embeddings. Within the domain of threat identification, wherein inter-node dependencies may frequently evolve or exhibit elevated complexity, this level of adaptive capacity becomes indispensable. Accordingly, our methodological framework incorporates Graph Attention Networks into a bespoke intrusion detection construct, with the objective of augmenting classification fidelity and reinforcing the model's robustness against emergent or hitherto unseen adversarial manifestations. A recent GNN-based intrusion detection framework highlights its effectiveness in modelling traffic behaviours and host-level interactions through graph-based techniques [2].

Graph representation learning identifies complex attack patterns more accurately than traditional approaches. This also demonstrates the GNNs' resilience against adversarial manipulation [3] and their flexibility across diverse cybersecurity applications [4], thereby strengthening their potential for next-generation IDS solutions. Benchmark datasets are important for evaluating the increasing complexity and scale of network attacks. The datasets used in this study capture the current network traffic patterns and include diverse attack behaviors. The comprehensive and contextual nature of the datasets makes them relevant for the development and training of deep learning-based intrusion detection mechanisms.

Recent research has emphasized the potential of graph-based learning to discern the nuances of malicious activities. Traditional methodologies are often inadequate to represent the flow and evolution of threat behaviors. Graph-driven models that incorporate attention mechanisms facilitate a more sophisticated, context-sensitive understanding of interactions within networked environments. Several graph-based malware detection strategies underscore critical architectural choices, Graph Neural Network variants, and the availability of open-access datasets [5]. Their findings underscore the growing significance of graph learning in advancing cybersecurity methodologies [6]. Deep learning with Autoencoders for cyberattack detection and feature extraction provides high accuracy and low false positives [7]. The efficiency of any IDS depends upon overcoming dataset challenges such as data imbalance; a balanced dataset as input is crucial for accurate detection of complex attacks [8].

Extending from this theoretical foundation, the present study proposed GAT-AID, a dual-branch architecture for intrusion detection. The framework combines supervised MLP techniques and unsupervised Autoencoder techniques for simultaneous detection of known attacks and zero-day attacks. In contrast to conventional classifiers, our model extracts node-specific representations from batches of traffic data, enabling detailed, precise detection. The architecture employs a two-layer GAT with dynamic hidden dimensions, multi-head attention, and residual connections to generate node embeddings that encode both local and global traffic behaviors. These embeddings are then passed to two distinct branches: a Multi-Layer Perceptron (MLP) classifier for multi-class classification of known attacks, and an Autoencoder for anomaly detection based on reconstruction error, particularly suited for capturing zero-day attacks.

The contributions of this paper are as follows:

(1) We develop a graph-oriented intrusion detection framework designed to handle large-scale network traffic in dynamic batches.
(2) We build a dual-branch architecture combining MLP and Autoencoder to detect known and novel zero-day attacks.
(3) We propose a GAT-based representation methodology that utilizes multi-head attention and residual learning to enhance identification and generalization.
(4) We assess GAT-AID utilizing two benchmark datasets and contrast its effectiveness with established models such as SVM, Random Forest, CNN, and GCN frameworks, exhibiting enhanced detection accuracy and reducing false positive rates.

Experimental results show that GAT-AID achieves accurate detection with a minimized false alert rate while maintaining scalability in high-throughput network environments. Its flexibility toward unfamiliar threat categories also makes it a critical component of preventive intrusion defense systems.

2 Related Work

This section reviews existing Graph Neural Network-based IDSs, highlighting the various techniques used for attack detection.

Many conventional ML and DL methods, such as SVMs, Random Forests, CNNs, and Autoencoders used for IDS, have limited capability to learn from complex relational patterns. This is due to their inability to model the structural dependencies within network traffic. GNNs, such as Graph Convolutional Networks (GCNs) and Graph Attention Networks (GATs), address the limitations of learning from graph-structured data by offering improved representations of network interactions.

Despite their promise, existing graph-based IDS studies tend to focus on architectural descriptions rather than on thorough comparative analysis. Critical aspects such as graph construction methodologies, attention mechanism variations, scalability to large heterogeneous traffic, and robustness to data imbalance and zero-day attacks are often underexplored.

Wu *et al.* [9] put forward TCG-IDS, an intrusion detection system that learns without supervision. It uses graphs along with some clever contrastive learning techniques that look at time, asymmetry, and masking to better understand how network traffic changes. Using Graph Neural Networks, the model scored quite high on two datasets, achieving nearly 99.5% on NF-CSE-CIC-IDS2018-V2 and just over 91 % on NF-UNSW-NB15-V2. Plus, it kept false alarms pretty low, around 4% and 3%. The system was built to handle things in real time and designed with zero-trust networks in mind, making it well-suited to situations where you cannot really trust the environment.

Park *et al.* [10] developed a system, G-IDCS, focused on the CAN protocol. Their setup combines a simple threshold detector with machine learning to reduce the number of CAN messages it needs by over 30 times, actually. This approach improved detection accuracy by about 9% compared to earlier graph-based methods. Another nice thing about G-IDCS is that it's more transparent: it produces results that make it easier to understand and investigate attacks. Plus, it stays solid even as attack methods keep changing.

Zhang *et al.* [11] developed a federated graph neural network framework to detect anomalies in CAN systems. The authors created directed graphs from CAN message streams by encoding details such as message content and how often IDS alerts appear. Their method uses a two-step classifier setup: first, a one-class anomaly detector spots unusual behaviors, then a multi-class classifier with an OpenMax layer identifies specific attack types, including ones it has not seen before. Federated learning enables a system to generalize across different vehicles while maintaining data privacy. Their low detection delays, under 3 milliseconds, further support real-time applications and detection of a wide variety of CAN attacks.

Xiao *et al.* [12] proposed a graph-based method to detect intrusions in vehicle networks. They use a Graph Node Attention Network (GNAT) to extract features that account for how CAN packets are ordered over time. Essentially, packets that happen close together are linked, and use attention to highlight key differences between normal and malicious traffic patterns. Subsequently, these features are fed into a random forest classifier, which demonstrates improved attack detection accuracy. The approach also performs effectively across different situations, handling changes both in timing and in how the network messages are structured.

Wang *et al.* [13] proposed K-GetNID, a framework that leverages prior knowledge to detect intrusions early. It models network traffic as graphs with different node types and connections, capturing how these change over time. Using a graph neural network designed for this purpose, the system can learn complex patterns and detect attacks sooner. They also built in a way to adjust detection timing, so it can handle shifts in the data without needing to be retrained all the time. Compared with conventional deep learning models, K-GetNID maintains competitive accuracy while offering improved adaptability to new data and lower reliance on labeled data.

Tran *et al.* [14] proposed a method to prepare network flow data for intrusion detection by transforming it into graphs. This makes it easier for graph neural networks to identify patterns in the traffic. They focus on extracting important features from the flows and building meaningful connections between nodes and edges. Based on their experimental findings, this approach detects intrusions more accurately than some traditional methods. Plus, it is designed to handle larger amounts of data without slowing down, which is crucial for real-world intrusion detection systems.

Saunders *et al.* [15] developed a graph-based intrusion detection system using Graph Convolutional Networks (GCNs) to identify DDoS attacks and classify multiple types of network intrusions. Their method captures both statistical data and the network structure of attacker-victim relationships, enabling the detection of complex attack patterns. The system achieved high accuracy and showed resilience to diverse DDoS attacks, supporting its potential for critical telecom infrastructure.

Lee *et al.* [16] examined intrusion detection through

the lens of how IP addresses behave and interact. The authors used graph neural networks to map relationships in network traffic, helping identify tricky, long-term attacks that might otherwise slip by. To evaluate performance, they ran tests simulating heavy traffic loads. Even with lots of data flowing through, their approach maintains its accuracy, which is promising for real-world networks where traffic can become intense.

Saunder *et al.* [17] took on DDoS detection using Graph Convolutional Networks. Their setup had three hidden layers with 128 neurons each, designed to capture both the network's structure and statistics to distinguish normal from malicious traffic. Testing on a known dataset showed they achieved very high scores—almost perfect across the board. What is impressive is that their model is not only accurate but also capable of handling large and diverse DDoS attacks with ease.

Singh *et al.* [18] introduced a Network Intrusion Detection System (NIDS) that uses a Graph Neural Network (GNN) to classify multiple types of cyberattacks, including R2L, DoS, Probe, and U2R. Their model was trained on an extensive dataset of intrusion attempts and was benchmarked against more traditional machine learning techniques such as decision trees, K-nearest neighbors (KNN), and random forests. Based on their evaluation results measuring accuracy, precision, recall, and F1 score, the GNN model stood out by achieving 91% accuracy, outperforming conventional classifiers. This approach employs a unified model, showing both strong robustness and scalability, while improving detection precision across different categories of network intrusions.

Daoud *et al.* [19] proposed a novel intrusion detection system that mixes knowledge graphs with the usual machine learning and deep learning techniques. The main part of this approach is that it uses the connections in these knowledge graphs to gain better insight into how different pieces of security data relate to each other. For that reason, it can identify tricky, new kinds of attacks that often slip past the usual signature-based methods or simpler ML models. In evaluations, the system significantly reduced alarm rates while effectively detecting zero-day attacks, demonstrating its ability to adapt even as the network environment evolves.

Phung *et al.* [20] unveiled a composite Network Intrusion Detection System that synergizes Graph Attention Networks (GAT) with Particle Swarm Optimization (PSO). The GAT module identifies intricate node-to-node dependencies via an attention-guided learning paradigm, whereas PSO isolates the most discriminative subset of attributes, thereby reducing the dataset's dimensional complexity. Under a 5-fold cross-validation strategy, the proposed architecture achieved higher detection rates, reduced feature-related overhead, and improved adaptability compared with established detection mechanisms.

Seng *et al.* [21] proposed a recommendation model that integrates self-attention with Graph Convolutional Networks (GCN). The GCN helps capture the complex links between users and items, but it struggles to capture how user interests change over time. To address these limitations, they added a self-attention module that better tracks those shifts. They also directly embed rating data into the interaction graph, helping address sparse data and providing a clearer picture of users' long-term likes. In evaluations on popular datasets, the model outperformed existing methods.

Rehman *et al.* [22] introduced Flash, a provenance-based intrusion detection system that treats system logs as graphs and learns detailed node representations using Graph Neural Networks combined with contextual and semantic encoders. One of their key innovations is the embedding recycling method, which speeds up real-time detection by reducing repeated GNN computations. Their model showed high accuracy, strong resistance to advanced persistent threats, and scaling on large-scale, real-world provenance graph datasets.

Sharma *et al.* [23] developed Web Wall, a zero-day intrusion detection system that uses GraphSAGE-based spatial Graph Neural Networks to analyse web traffic from both directions. By extracting features from both flows and individual packets, the model generalizes well to previously unseen attacks without requiring retraining. On the CICIDS2017 and CICIDS2018 datasets, Web Wall achieved over 99.95% accuracy and outperformed traditional intrusion detection systems. It combines anomaly detection with graph embeddings, making it effective in detecting encrypted and zero-day attacks while reducing false positives. Alharbi *et al.* [24] developed a graph-based machine learning methodology to identify botnet activity by analysing interaction patterns among connected hosts.

Despite advances in graph-based IDS, existing studies have several limitations. These include the lack of systematic evaluation of graph construction strategies, attention mechanisms, and scalability to large, heterogeneous network traffic. Furthermore, these approaches fail to address zero-day attack detection through unsupervised methods. This observation motivates the need for integrated frameworks that jointly optimize scalable embedding learning and effective anomaly detection.

Table 1. Comparative overview of representative GNN-based intrusion detection approaches

Study	Architecture	Key findings / limitations
ProAPT (2025)	DRL-based	Limited cross-domain evaluation and lack of real-time scalability
Smart Grid Security [25]	GNN-based	Domain-specific design restricted to electrical infrastructure
Wang et al. (2022)	GCN	Limited effectiveness for zero-day attack detection
Liu et al. (2021)	GAT + AE	Absence of a dual-branch detection mechanism
This work (GAT-AID)	GAT + MLP + AE	Improved zero-day detection performance and scalability

While conventional flow-based methods have been widely used, they often come with drawbacks like heavy processing loads and a limited view of the network's structure. To overcome these issues, graph-based representations incorporating structural attributes have been introduced, showing superior training efficiency, adaptability, and detection accuracy on datasets such as CTU-13 and IoT-23.

In contrast to these studies, GAT-AID proposes a novel dual-branch architecture that combines a supervised MLP and an unsupervised Autoencoder branch on GAT embeddings to address limitations in zero-day detection, scalability, and feature integration. Table 1 presents a comparative analysis of the GNN-based IDS, including its architecture, dataset used, and limitations.

3 Technical Preliminaries

The GAT-AID proposed IDS framework uses several core deep learning and graph processing components, which are briefly defined in this section to ensure clarity for the reader:

(1) **Graph Attention Network (GAT):** GAT is a type of Graph Neural Network (GNN) we have used for modelling the complex, structured relationships within network traffic. It employs a self-attention mechanism to assign learned weights to neighbouring flow records, thereby determining which connections are important for attack detection.

(2) **Multi-Layer Perceptron (MLP):** This is the supervised classifier technique in our dual-branch system. MLP processes the GAT embedding to perform multi-class prediction of known attack types.

(3) **Autoencoder (AE):** The AE serves as an unsupervised network of the GAT-AID frameworks and operates as the second branch of the model, which is used to detect zero-day attacks. It is trained on normal traffic, and patterns with high reconstruction error are flagged as zero-day attacks.

(4) **Softmax Function:** This activationfunction is applied to the MLP's final layer. It converts the raw classification head logits into a probability distribution over all known attack classes, ensuring a consistent level of confidence for each prediction.

(5) **Activation Functions (ReLU / ELU):** These activation functions are applied within the hidden layers of the GAT and MLP to introduce non-linear feature learning. They introduce non-linearity to learn complex relationships, which helps rapid convergence and mitigates the vanishing gradient problem.

4 Proposed Methodology

4.1 Overview of the GAT-AID Pipeline

The GAT-AID pipeline defines the overall workflow for analysing raw network traffic before making intrusion detection decisions. Incoming traffic records are first organized into batch-wise graph representations, allowing relationships among traffic instances within the same observation window to be captured rather than treating each record in isolation. This formulation enables contextual traffic behavior to influence subsequent analysis. The constructed graphs are then processed through successive stages that refine feature representations and support decision-making. At the final stage, the pipeline uses a parallel inference mechanism: one path focuses on known attacks, and the other on deviations from normal traffic behavior. Together, these steps form a practical workflow for large-scale traffic analysis while supporting the detection of both known intrusions and previously unseen attacks.

4.2 Model Architecture

The proposed system, GAT-AID (Graph Attention-based Anomaly and Intrusion Detection), is a hybrid deep learning framework designed to detect both-known and unknown (zero-day) cyberattacks in real-world network traffic. It combines dynamic graph attention encoding, batch-wise graph construction, and two-path detection heads, supervised and unsupervised, optimized for performance across multiple datasets. This methodology comprises six main stages: dataset preprocessing, dynamic graph modelling, attention-based encoding, bifurcated detection modules, baseline benchmarking, and implementation details. The proposed GAT-AID architecture, shown

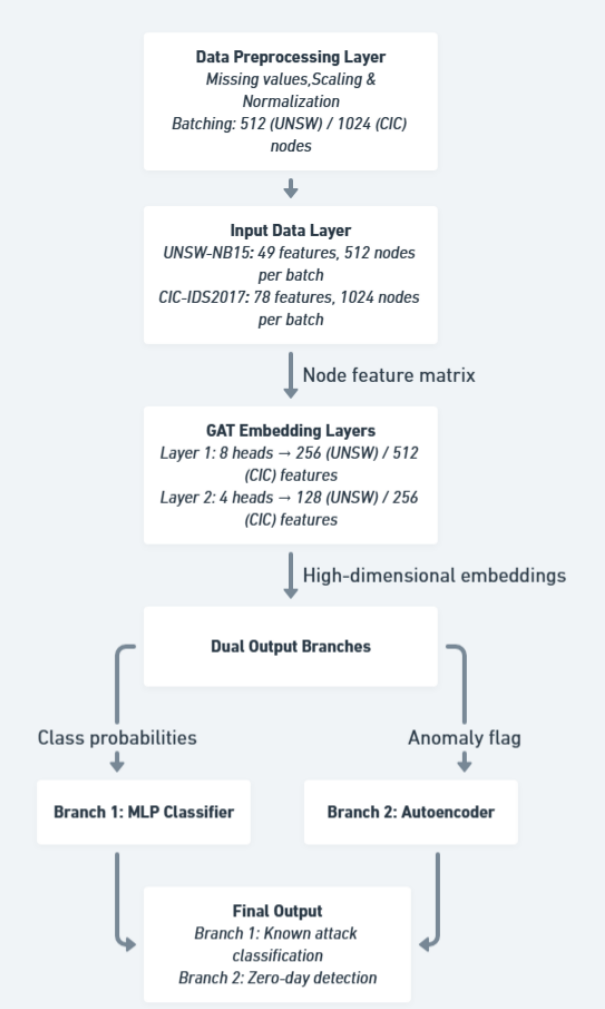

Figure 1. Proposed Graph Attention-based Anomaly and Intrusion Detection (GAT-AID) Architecture

in Figure 1, comprises a dual-branch attack-detection system that leverages both supervised and unsupervised learning. After the GAT backbone, a multilayer perceptron (MLP) classifier serves as the first branch, comprising three fully connected layers with ReLU activations. The final layer applies a Softmax function for multi-class classification to detect known attacks. The second branch consists of an Autoencoder that performs unsupervised anomaly detection with a bottleneck representation size of 32. This branch reconstructs input node features and flags deviations via reconstruction error, enabling the detection of unknown or novel attacks.

The overall architecture of the proposed GAT-AID framework is illustrated in Figure 1.

4.3 Dataset and Preprocessing

The Data Preprocessing Layer constitutes the first stage in handling raw traffic before the GAT model. This stage is essential for two key reasons: first, to clean, organize, and structure the data to reduce memory overhead; and second, to significantly improve training efficiency. For UNSW-NB15, 500,000 records are divided into 512-node batches, producing 977 subsets, while the same number of CICIDS 2017 samples are grouped into 1024-node batches, yielding 488 subsets. To mitigate ordering bias, samples are randomly shuffled across epochs. Each training step processes a single batch to the GAT input layer, where the batch is encoded as a node-feature matrix: [512 nodes × 49 features] for UNSW-NB15 and [1024 nodes × 78 features] for CICIDS 2017. The preprocessing also includes feature alignment and padding, enabling consistent integration of datasets with different feature sizes.

4.4 Input Layer

Following preprocessing, the input layer standardizes the incoming batches from multiple datasets, performing feature alignment and normalization. Each batch is processed sequentially and examined for missing or misaligned features. Missing values are padded with zeros or mean values to maintain a uniform matrix structure. Numeric features are normalized to the [0,1] range to limit feature imbalance and stabilize training. The batch is passed forward without reshaping, ensuring it is ready for downstream GAT hidden layers. Notably, the input layer neither modifies the batch size nor performs mini-batch processing; it operates strictly on the batch provided by the preprocessing layer.

4.5 first GAT hidden layer

The initial hidden layer of the GAT extracts features using an attention mechanism. To ensure training stability and to preserve information from the original node's features, our system incorporates both layer normalization and residual connections. Each dataset batch is processed independently, preventing memory issues and maintaining processing consistency. In the case of UNSW-NB15, a batch has dimensions of [512 nodes × 49 features], whereas CICIDS2017 provides [1024 nodes × 78 features]. The hidden representations are expanded to 256 dimensions for UNSW-NB15 and 512 for CICIDS 2017. To capture different patterns of node interactions, the model applies eight attention heads in parallel. Nonlinearity is introduced using the Exponential Linear Unit (ELU) activation function, which helps prevent dying neurons. Meanwhile, layer normalization controls gradient growth, and the residual pathway ensures that the base node features remain intact. The result of this stage is a set of richer node embeddings-[512 nodes × 256 features] per batch for UNSW-NB15 and [1024 nodes ×

512 features] per batch for CICIDS 2017.

4.6 The second GAT hidden layer

The second GAT hidden layer builds on the first by narrowing the embeddings and focusing on traffic behaviors that matter most. For UNSW-NB15, the input dimensions are [512 nodes × 256 features], and for CICIDS 2017, they are [1024 nodes × 512 features]. At this point, the dimensions are reduced to 128 for UNSW-NB15 and 256 for CICIDS 2017, thereby retaining essential behavioral traits while trimming redundancy. Four attention heads run in parallel, each capturing different aspects of traffic variation, from sudden spikes to subtle protocol changes. The ReLU activation function helps the model converge more quickly, and layer normalization addresses scaling issues across batches. Residual paths are still active, carrying forward the baseline node information, so critical data is lost. By the end of this layer, the model produces compressed but richer embeddings: [512 nodes × 128 features] for UNSW-NB15 and [1024 nodes × 256 features] for CICIDS 2017. These embeddings are then passed to a classifier or an anomaly detector.

4.7 Hyperparameter Selection and Justification

The hyperparameter choices in GAT-AID are strategically designed to balance model expressiveness, computational efficiency, and detection performance. First, selecting eight attention heads in the first GAT layer enables the model to simultaneously capture diverse traffic patterns, from packet flow variations to protocol anomalies, without excessive computational overhead. Eight heads provide sufficient representational diversity to help achieve reasonable training time compared to deeper or wider variants. The second GAT layer uses four attention heads, reflecting the diminishing returns principle: after initial feature extraction, fewer heads suffice to focus on the most salient behavioral characteristics, reducing redundancy while maintaining discriminative power. The first hidden layer uses the Exponential Linear Unit (ELU) activation function to mitigate the dying-receptor problem and maintain stable gradient flow. In the second layer, ReLU is preferred because the first layer has already captured diverse features and supports faster convergence at this stage.

Hidden dimension choices (256 for UNSW-NB15 and 512 for CICIDS2017 in Layer 1; 128 and 256 in Layer 2) are justified by dataset complexity and feature heterogeneity. CICIDS2017 has higher dimensionality (78 features vs. 49), necessitating larger intermediate representations to preserve information diversity. The progressive dimensionality reduction (512→256, 256→128 for CICIDS2017) follows the principle of hierarchical feature abstraction, progressively filtering redundant information while retaining essential attack signatures. Layer normalization and residual connections at each stage stabilize this reduction process, preventing information loss and enabling effective gradient propagation.

Training employs a learning rate of 0.001 across all experiments, identified through preliminary experiments as the optimal balance between convergence speed and stability. Higher learning rates (≥ 0.01) cause early epochs to diverge, whereas lower rates (< 0.001) result in slow training without performance improvement. Adam optimizer is chosen over Stochastic Gradient Descent (SGD) due to its adaptive learning rate, which automatically adjusts step sizes per parameter. This approach is particularly beneficial for heterogeneous network traffic datasets where gradient magnitudes vary significantly across feature dimensions. The Adam configuration with a learning rate of 0.001 ensures consistent convergence within 50–100 epochs across both datasets.

4.8 The Output Layer

The Output Layer uses a dual-branch architecture, allowing it to classify known attacks and simultaneously detect zero-day anomalies. The first branch relies on a Multi-Layer Perceptron (MLP) to process the refined embeddings for multi-class classification. This MLP is relatively shallow with just two hidden layers with 128 and 64 neurons. Both use ReLU, and the last layer employs Softmax, which converts the output into a probability distribution across ten classes for UNSW-NB15 and fifteen classes for CICIDS 2017. Categorical cross-entropy drives the optimization in this branch.

4.9 The second branch

The second branch of our model uses an Autoencoder(AE) to detect zero-day anomalies. Unlike the classification branch, this branch operates on higher-dimensional embeddings, making it better suited for identifying patterns it has not seen before and adapting to changes in network behavior. Along with the MLP branch, the Autoencoder produces a binary flag for each node in a batch - [512 nodes × 1 flag] for UNSW-NB15 and [1024 nodes × 1 flag] for CICIDS2017.

At the batch level, both outputs are combined: the MLP labels (known attacks) and the Autoencoder anomaly signals. This fusion enhances robustness, as the system captures both known and unseen attacks.

Overall, the framework processes data in a memory-efficient, batch-wise fashion, enabling scalability. The combination of GAT-based embeddings, attention, and the dual-branch output provides a flexible setup that detects not only well-studied attacks but also zero-day threats.

To detect zero-day attacks, the Autoencoder (AE) employs the reconstruction error (RE) computed for each traffic record indexed by i. The reconstruction error RE_i is calculated using the Mean Squared Error (MSE) between the original GAT embedding $\mathbf{Z}_i$ and its reconstructed representation $\hat{\mathbf{Z}}_i$, which is formally defined as:

$$RE_i = \left\| \mathbf{Z}_i - \hat{\mathbf{Z}}_i \right\|^2$$

The binary anomaly indicator $A_i \in \{0, 1\}$ is determined by comparing the reconstruction error RE_i with a learned threshold τ, as defined by:

$$A_i = \begin{cases} 1, & \text{if } RE_i > \tau \quad \text{(zero-day anomaly)} \\ 0, & \text{if } RE_i \leq \tau \quad \text{(normal traffic)} \end{cases}$$

4.10 Dual-Branch Decision Fusion (DBDF) and Final Classification

The supervised attack indicator K_i is derived from the output of the MLP classifier and indicates whether a network flow is identified as a known attack category.

$$K_i = \begin{cases} 1, & \text{if the predicted class} \neq \text{“Normal”} \\ 0, & \text{if the predicted class} = \text{“Normal”} \end{cases}$$

The final decision of the proposed GAT-AID framework is obtained by jointly considering the supervised attack flag K_i and the unsupervised anomaly indicator A_i. The supervised branch is responsible for identifying known attack classes, while the unsupervised branch detects previously unseen zero-day anomalies.

$$C_i = \begin{cases} \text{“Known Attack”}, & \text{if } K_i = 1 \\ \text{“Zero-Day Anomaly”}, & \text{if } K_i = 0 \text{ and } A_i = 1 \\ \text{“Normal”}, & \text{if } K_i = 0 \text{ and } A_i = 0 \end{cases}$$

The dual-branch decision fusion operates independently at the batch level, ensuring scalable and high-speed processing while combining supervised and unsupervised detection criteria for robust and reliable intrusion detection.

4.11 Validation and Baseline Comparison Strategy

To ensure methodological rigor and fair performance evaluation, we employ a comprehensive validation framework.

Train–Test Split

Each dataset is partitioned into 80% training and 20% testing subsets, a standard practice in machine learning that provides sufficient training data while maintaining adequate test coverage for robust generalization assessment. The split is performed chronologically for temporal datasets (CICIDS2017) to prevent temporal data leakage and ensure the model is tested on unseen future traffic patterns.

Cross-Validation Strategy

To mitigate variance from random splits, 5-fold stratified cross-validation is used, ensuring each fold maintains the original class distribution. This approach strengthens confidence in reported metrics and provides per-fold performance variation estimates. The mean and standard deviation across folds are reported for all metrics, enabling uncertainty quantification of reported results.

Baseline Comparison

The proposed GAT-AID framework is evaluated against several representative intrusion detection baselines, encompassing statistical, ensemble-based, deep learning, and graph-based approaches.

- **Support Vector Machine (SVM):** A traditional statistical classifier used as a non-neural baseline. Hyperparameters are selected via a three-fold grid search on the training set, with $C \in \{0.1, 1, 10, 100\}$ and kernel functions $\{\text{linear}, \text{RBF}, \text{polynomial}\}$.
- **Random Forest (RF):** Ensemble method representing tree-based approaches that are robust to feature scaling. The hyperparameter grid includes $n_{\text{estimators}} \in \{50, 100, 200\}$ and $\text{max\_depth} \in \{10, 20, \text{None}\}$. Optimal parameters are determined using three-fold grid search.
- **Convolutional Neural Network (CNN):** A deep learning baseline designed to capture spatial feature correlations. The architecture consists of two convolutional layers with 32 and 64 filters, followed by two fully connected layers with 128 and 64 neurons, respectively. The model is trained using Adam optimizer with a learning rate of 0.001 for 100 epochs.
- **Graph Convolutional Network (GCN):** A state-of-the-art graph neural network baseline that enables direct comparison with the proposed GAT-based approach. The architecture comprises two GCN layers with 256 and 128 hidden dimensions, respectively, employing ReLU activation. Training is performed using Adam optimizer with a learning rate of 0.001 for 100

epochs.

5 Experimental Setup

5.1 Computational Environment

All experiments were conducted on a high-performance computing system equipped with a modern high-end GPU (e.g., NVIDIA RTX 4090 or A100, with 24–80 GB of VRAM), 64 GB system memory, and the Ubuntu 22.04 operating system. The model uses PyTorch Geometric 2.4 and PyTorch 2.1. The computational trade-offs between GAT-AID and baseline approaches were evaluated by recording the training times for each model, including data handling and preprocessing.

5.2 Model Training Configuration

The proposed network is trained using Adam optimizer with an initial learning rate of 0.001. The supervised classification branch employs the categorical cross-entropy loss function, while the Autoencoder branch minimizes the mean squared error (MSE) between the original feature representation and its reconstruction. A batch size of 128 graph samples is used consistently across all training runs.

To mitigate overfitting and improve generalization performance, early stopping based on validation loss is applied with a patience of 10 epochs. In addition, regularization strategies include dropout with a rate of 0.3 and ℓ_2 weight decay with a coefficient of 1×10^{-4} applied to all trainable parameters.

To assess statistical robustness and reduce variance due to random initialization, model training is repeated for three independent runs using different random seeds, and the reported results reflect averaged performance across these runs.

5.3 Baseline Models and Hyperparameter Tuning

GAT-AID is benchmarked against four established intrusion detection approaches. Hyperparameter tuning for all baselines follows a consistent grid search protocol on the training set to represent their best achievable performance, preventing algorithmic bias.

5.4 Baseline Models and Hyperparameter Tuning

To ensure a fair and comprehensive evaluation, the proposed GAT-AID framework is compared against a diverse set of representative baseline models, encompassing statistical, ensemble-based, deep learning, and graph-based intrusion detection approaches. For all baselines, hyperparameters are tuned exclusively on the training data to avoid information leakage.

- **Support Vector Machine (SVM):** A classical statistical learning baseline serving as a non-neural reference model. The hyperparameter grid includes $C \in \{0.1, 1, 10, 100\}$ and kernel functions $\{\text{linear}, \text{RBF}, \text{polynomial}\}$. Hyperparameter selection is performed using a three-fold grid search on the training set, with the optimal configuration determined based on the cross-validation F1-score.
- **Random Forest (RF):** An ensemble-based baseline representing tree-based classification techniques that are robust to feature scaling. The hyperparameter search space consists of $n_{\text{estimators}} \in \{50, 100, 200\}$ and $\text{max\_depth} \in \{10, 20, \text{None}\}$. Optimal parameters are identified through three-fold grid search on the training data. The model is implemented using *scikit-learn* (version 1.0).
- **Convolutional Neural Network (CNN):** A deep learning baseline designed to capture spatial feature correlations. The network architecture comprises two convolutional layers with 32 and 64 filters, respectively, using a kernel size of 3×3, followed by a max-pooling layer and two fully connected layers with 128 and 64 neurons. A Softmax layer is employed for final classification. The model is trained using Adam optimizer with a learning rate of 0.001 for 100 epochs and a batch size of 512. The implementation is based on the PyTorch framework.
- **Graph Convolutional Network (GCN):** A graph-based deep learning baseline enabling direct comparison with the proposed GAT-based methodology. The architecture consists of two GCN layers with 256 and 128 hidden dimensions, respectively, employing ReLU activation, followed by a Softmax output layer. Training is conducted using Adam optimizer with a learning rate of 0.001 for 100 epochs and a batch size of 512. The implementation is carried out using PyTorch in conjunction with PyTorch Geometric (PyG).

6 Datasets

This study evaluates the proposed method using two widely recognized intrusion detection datasets: CICIDS 2017 and UNSW-NB15.

CICIDS 2017 dataset, developed by the Canadian Institute for Cybersecurity, captures realistic modern network traffic with both benign and various malicious behaviors, including Distributed Denial-of-Service (DDoS), Port Scans, Brute Force attacks, Botnets, and Web-based exploits. It includes approximately 2.8 million labelled samples across 15 classes,

consisting of one normal category and 14 attack types. A total of 78 relevant features were extracted from the raw traffic data, with fields such as timestamp and flow ID excluded to avoid redundancy and information leakage.

The UNSW-NB15 dataset, created by the Australian Cyber Security Centre (ACSC), simulates contemporary traffic by blending legitimate flows with nine malicious categories, including Shellcode, Fuzzers, Backdoors, and Exploits. It contains approximately 2.5 million samples across 10 classes, including one normal class and nine attack types. After preprocessing, 49 numerical and categorical features were retained for analysis.

For balanced evaluation, equal-sized subsets of 500,000 samples each were selected from both datasets, containing an even mix of normal and attack traffic to prevent model bias. Table 2 summarizes the key statistics:

The CICIDS2017 dataset primarily focuses on application-layer and high-volume attacks, whereas the UNSW-NB15 dataset includes subtler threats such as reconnaissance and backdoor attacks. Prior to model training, numerical features were normalized using Min-Max scaling, and categorical variables were encoded. Non-predictive fields such as timestamps and IDs were removed to prevent information leakage.

7 Evaluation Metrics

To comprehensively evaluate GAT-AID's detection performance, we employ a suite of metrics capturing different aspects of intrusion detection effectiveness:

Classification Metrics

The detection performance for known attack categories, handled by the supervised MLP branch, is evaluated using the following classification metrics:

- **Accuracy:** Measures the correct predictions across all classes, indicating balanced performance.
- **Precision:** Measures the true positives among predicted positives. This is very important for minimizing false alarms in production IDS.
- **Recall (Sensitivity):** Measures true positives among actual positives. This is important to ensure that missed attacks are kept to a minimum.
- **F1-Score:** Harmonic mean of Precision and Recall, providing a single metric for imbalanced datasets.

Anomaly Detection Metrics

To assess zero-day attack detection performance via the Autoencoder-based unsupervised branch, the following anomaly detection metrics are considered:

- **False Positive Rate (FPR):** Measures the rate at which normal traffic is flagged as anomalous, which has an immediate impact on operational usability.
- **False Negative Rate (FNR):** 1. Measures missed attacks. These metrics indicate how well the detection system works against new threats.

The anomaly detection threshold is derived from the reconstruction error distribution of normal traffic in the validation set. Specifically, the threshold is set at the 95^{th} percentile of reconstruction errors, ensuring that approximately 95% of normal samples are correctly labeled as benign while keeping the false positive rate close to 5%. The same threshold is applied during testing to maintain consistency in performance evaluation.

Combined Performance Metric: The area under the receiver operating characteristic curve (ROC–AUC) is employed as a threshold-independent metric to quantify the trade-off between the true positive rate and false positive rate across varying decision thresholds.

Computational Efficiency Metrics: To evaluate the practical feasibility of the proposed framework in real-time network environments, computational efficiency is assessed using training time (measured in seconds per epoch) and inference latency (measured in milliseconds per batch). These metrics provide insight into the scalability and deployment readiness of the system.

7.1 Validation Strategy

A rigorous validation strategy is adopted to ensure robustness, fairness, and statistical reliability in the evaluation of the proposed GAT-AID framework. The key components of the validation protocol are summarized below.

- **Train–Test Split:** Each dataset is partitioned into 80% training data and 20% testing data. For datasets exhibiting temporal characteristics, a chronological split is applied to prevent temporal data leakage. This ensures that the evaluation is conducted on previously unseen future traffic patterns, thereby reflecting realistic deployment scenarios.
- **Cross-Validation:** A stratified five-fold cross-validation strategy is employed to preserve the original class distribution (benign versus attack categories) within each fold. Evaluation metrics

Table 2. Summary of Reduced IDS Datasets

Dataset	Total Samples	Normal Samples	Attack Samples	Features	Attack Classes	Major Attack Types
CICIDS2017	500,000	250,000	250,000	80	10	DoS, DDoS, PortScan, Brute Force, Web Attack
UNSW-NB15	500,000	250,000	250,000	49	6	Fuzzers, Exploits, Backdoor, Reconnaissance, Generic

are computed independently for each fold, and the mean and standard deviation across folds are reported to enable uncertainty quantification.

- **Fair Comparison Protocol:** All baseline models are trained and evaluated using identical train–test splits and cross-validation folds. This protocol ensures a fair and direct comparison between the proposed GAT-AID framework and competing approaches.
- **Statistical Significance Testing:** McNemar's test is used to assess the statistical significance of performance differences between GAT-AID and baseline models, with a significance level of $\alpha = 0.05$. This test is well-suited for paired comparisons of classification outcomes under cross-validated experimental settings. Additionally, 95% confidence intervals are estimated via bootstrap resampling with 1000 iterations to quantify uncertainty in all reported performance metrics.

8 Results and Discussion

We evaluated the GAT-AID model on CICIDS 2017 and UNSW-NB15 benchmark datasets. The objective was to examine detection accuracy, zero-day attack detection, and computational efficiency. To ensure a fair evaluation, each test was repeated five times. The results were then averaged, and the standard deviation was included to show consistency. A paired t-test confirmed that the model's improvements were statistically significant ($p < 0.05$).

8.1 Performance Comparison with Baseline Models

To understand how GAT-AID performs relative to several established methods, we compared it with Random Forest (RF), Support Vector Machine (SVM), Convolutional Neural Network (CNN), and Graph Convolutional Network (GCN).

While SVM struggles with large, complex data, it remains a useful benchmark because it has been widely used in intrusion detection research for years.

As shown in Table 3, GAT-AID outperformed the compared models, achieving 98.4% on accuracy and 98.0% on F1-score. The improvements over CNN and GCN are statistically significant ($p < 0.05$), suggesting the attention-based graph-based models.SVMs perform comparatively poorly; results demonstrate their limitations on modern, large-scale datasets, supporting the case for more advanced techniques.

Table 3. Performance Comparison on CICIDS2017 Dataset

Model	Accuracy	Precision	Recall	F1-score
Random Forest	94.8 ± 0.3%	93.6 ± 0.5%	92.4 ± 0.4%	93.0 ± 0.4%
SVM	91.2 ± 0.4%	90.1 ± 0.5%	89.7 ± 0.6%	89.9 ± 0.5%
CNN	96.1 ± 0.3%	95.3 ± 0.4%	95.0 ± 0.4%	95.1 ± 0.3%
GCN	96.8 ± 0.2%	95.9 ± 0.3%	96.0 ± 0.3%	95.9 ± 0.3%
Proposed GAT-ADT	**98.4 ± 0.2%**	**98.0 ± 0.3%**	**97.9 ± 0.3%**	**98.0 ± 0.2%**

Table 4. Zero-Day Attack Detection Performance Comparison

Model	AUC–ROC	FPR	FNR
CNN	0.93	0.07	0.08
GCN	0.95	0.06	0.07
GAT + Autoencoder	**0.98**	**0.03**	**0.02**

These results further support the adoption of advanced graph-based techniques, such as GAT-AID, for robust and scalable intrusion detection.

8.2 Zero-Day Attack Detection

We evaluated zero-day attack detection by intentionally excluding selected attack categories during training and using an autoencoder-based anomaly detection module. Table 4 summarizes the zero-day Detection Performance. Table 4 summarizes the zero-day detection performance. The proposed approach achieved an AUC of 0.98, with both the false positive rate (FPR) and false negative rate (FNR) reduced by over 50% compared to the CNN baseline. These results demonstrate strong resilience against unseen attacks.

8.3 Computational Efficiency

Table 5 presents training time, inference time, and memory footprint. While GAT introduces a slight

Table 5. Computational Overhead

Model	Training Time (sec)	Inference Time (ms/sample)	Memory (MB)
CNN	250	1.2	500
GCN	320	1.4	550
Proposed GAT	**410**	**1.8**	**600**

Table 6. Comparison with State-of-the-Art Models

Model (Year)	Accuracy	AUC
IDS-GCN (2022)	96.8%	0.95
CNN-IDS (2023)	96.1%	0.94
RNN-AE (2023)	95.9%	0.93
Proposed GAT-AID	**98.4%**	**0.98**

computational overhead, its accuracy improvements justify this trade-off. GAT can process more than 500 packets per second on an optimized GPU, enabling near real-time IDS deployment despite the overhead.

8.4 Comparison with State-of-the-Art Models

Table 6 benchmarks the performance GAT-AID against recent IDS approaches.

The proposed GAT-AID framework improves detection accuracy by at least 1.6%, and achieves an increase of 0.03 in AUC compared to existing models. These results indicate the effectiveness of GAT-AID in addressing challenges associated with modern intrusion detection systems.

9 Conclusion

This study proposed GAT-AID, a dual-branch intrusion detection framework that integrates a multi-head Graph Attention Network (GAT) with a supervised Multi-Layer Perceptron (MLP) classifier and an unsupervised autoencoder. The proposed approach effectively integrates both topological dependency with semantic feature representation across network flows to enable accurate modelling of complex network patterns.

Experiment evaluations conducted on the CICIDS 2017 and UNSW-NB15 datasets demonstrated strong detection of both known and zero-day attacks. This proposed method outperforms conventional classifiers and GCN-based approaches. The GAT module effectively models complex inter-flow connections with attention-weighted message transmission. The proposed approach improves the identification of rare and advanced attack vectors. The result confirms its effectiveness in dynamic conditions, including zero-day attack detection. Moreover, the dual-branch architecture supports both classification and anomaly detection, resulting in a scalable and robust intrusion detection framework.

Despite these encouraging results, several works remain to be done in the future. The current static graph design based on KNN similarity could be extended to adaptive or learnable graph topologies that evolve with network patterns. However, GAT-AID demonstrates strong detection capability; it faces challenges when deployed for real-time use in high-volume network environments. This limitation can be addressed using a hierarchical or graph-sampling GNN architecture. Extending the framework to encrypted traffic analysis and multi-source telemetry inputs may improve overall detection performance.

Acknowledgment

The authors would like to thank the Department of Computer Engineering, Thakur College of Engineering and Technology, Mumbai, for providing the necessary facilities and support to carry out this research work. The authors are also thankful to their colleagues for useful discussions and suggestions during different stages of the work. The availability of computational resources and benchmark datasets helped in conducting the experiments. The authors would also like to thank the reviewers for their comments, which helped in improving the paper.

References

[1] N. Lin, L. Wu, Y. Liu, L. Xu, Z. Lin, T. Huang, and H. Liu. Dynamic malware detection method based on graph neural networks. In *Proceedings of the 8th World Conference on Computer Communication Technology (WCCCT)*, pages 309–315, April 2025.

[2] Z. Guo. Big data mining algorithm for malicious behavior feature extraction and application in network security. In *Proceedings of the 2nd International Conference on Big Data, Computing Intelligence, and Applications (BDCIA)*, volume 13550, pages 270–277, March 2025.

[3] Y. Wang, Z. Han, Y. Du, J. Li, and X. He. Bs-gat: A network intrusion detection system based on graph neural network for edge computing. *Cybersecurity*, 8(1):27, 2025.

[4] X. Deng, J. Zhu, X. Pei, L. Zhang, Z. Ling, and K. Xue. Flow topology-based graph convolutional network for intrusion detection in label-limited iot networks. *IEEE Transactions on Network and Service Management*, 20(1):684–696, March 2023.

[5] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. Graph attention networks. In *International Conference*

on Learning Representations (ICLR), Toulon, France, April 2018.

[6] T. Bilot, N. El Madhoun, K. Al Agha, and A. Zouaoui. Graph neural networks for intrusion detection: A survey. *IEEE Access*, 11:49114–49139, 2023.

[7] A. Salehi, S. Ahmadi, and M. R. Aref. A semi-supervised ids for cyber-physical systems using a deep learning approach. *ISeCure*, 15(3):43–50, 2023.

[8] H. Shadabfar, M. Dehghan, and B. Sadeghian. Dsrl-apt-2023: A new synthetic dataset for advanced persistent threats. *ISeCure*, 17(2):107–116, 2025.

[9] C. Wu, J. Sun, J. Chen, M. Alazab, Y. Liu, and Y. Xiang. Tcg-ids: Robust network intrusion detection via temporal contrastive graph learning. *IEEE Transactions on Information Forensics and Security*, 20:1475–1486, 2025.

[10] S. B. Park, H. J. Jo, and D. H. Lee. G-idcs: Graph-based intrusion detection and classification system for can protocol. *IEEE Access*, 11:39213–39227, 2023.

[11] H. Zhang, K. Zeng, and S. Lin. Federated graph neural network for fast anomaly detection in controller area networks. *IEEE Transactions on Information Forensics and Security*, 18:1566–1579, 2023.

[12] J. Xiao, H. Chen, and F. Zhong. A novel feature extraction framework using a graph node attention network for in-vehicle network intrusion detection. *IEEE Systems Journal*, 18(1):150–161, 2024.

[13] M. Wang, N. Yang, and N. Weng. K-getnid: Knowledge-guided graphs for early and transferable network intrusion detection. *IEEE Transactions on Information Forensics and Security*, 2024. In press.

[14] D. H. Tran and M. Park. Graph embedding for a graph neural network in an intrusion detection system. In *Proceedings of the International Conference on Information Networking (ICOIN)*, pages 395–397, January 2024.

[15] B. J. Saunders, R. E. de Grande, G. H. S. Carvalho, and I. Woungang. Deep graph learning for ddos detection and multi-class classification ids. In *IEEE International Conference on Cyber Security and Resilience (CSR)*, pages 96–100, September 2024.

[16] S. W. Lee, J. Y. Lee, and T. J. Lee. Graph neural networks for network intrusion detection: An ip behavioral analysis perspective. In *Silicon Valley Cybersecurity Conference (SVCC)*, pages 1–4, June 2024.

[17] B. J. Saunders, P. Kisanga, G. H. S. Carvalho, and I. Woungang. A graph convolutional network-based ddos detection model. In *IEEE International Systems Conference (SysCon)*, pages 1–5, April 2024.

[18] S. P. Singh, B. Abhay, A. Kumar, P. Agrawal, D. Singh, and V. Ghosh. An efficient approach using a graphical neural network for a network intrusion detection system. In *International Conference on Computing Communication and Networking Technologies (ICCCNT)*, pages 1–11, June 2024.

[19] M. A. Daoud, S. A. M. Mostefaoui, H. M. Meghazi, O. Labbadi, C. Zenina, and A. Bouguessa. Advanced intrusion detection systems leveraging knowledge graph-based techniques. In *International Conference on Systems and Control (ICSC)*, pages 424–428, November 2024.

[20] N. A. T. Phung, A. Nguyen, and M. T. Nguyen. Network intrusion detection system using a graph attention network and particle swarm optimization. In *International Conference on Advanced Technologies for Communications (ATC)*, pages 378–382, October 2024.

[21] K. Kim, M. Sohn, and J. Kim. Cross-domain explainable recommendation using graph convolutional networks and a topic model. *IEEE Access*, 13:118310–118323, 2025.

[22] M. U. Rehman, H. Ahmadi, and W. U. Hassan. Flash: A comprehensive approach to intrusion detection via provenance graph representation learning. In *IEEE Symposium on Security and Privacy (SP)*, pages 3552–3570, May 2024.

[23] N. Sharma, M. Swarnkar, and B. Mondal. Webwall: Zero-day attack detection in web traffic using a spatial graph neural network. In *IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*, pages 1–6, December 2024.

[24] A. Alharbi and K. Alsubhi. Botnet detection approach using graph-based machine learning. *IEEE Access*, 9:99166–99180, 2021.

[25] M. Dehghan and E. Khosravain. Smart grid security: Proactive prediction of advanced persistent threats. *Computer and Knowledge Engineering*, 8(2):25–40, 2025.

Nitin Wankhade Nitin Wankhade is a Research Scholar pursuing a Ph.D. in Computer Engineering from the Computer Engineering Department at Thakur College of Engineering and Technology, Mumbai. He is also working as an Assistant Professor in the Computer Engineering Department at NMIET, Pune. He has more than 18 years of combined academic experience and is involved in teaching, research, and project guidance. His research focuses mainly on deep learning, natural language processing, and the application of graph neural networks in cybersecurity. Over the years, he has guided many undergraduate and postgraduate students in areas related to computer science and artificial intelligence.

Dr. Anand Khandare Dr. Anand Khandare is working as an Associate Professor in the Computer Engineering Department at Thakur College of Engineering and Technology, Mumbai. He completed his Ph.D. in Computer Science and Engineering from Sant Gadge Baba Amravati University in 2019. Over the last 20 years, he has been teaching, mentoring students, and conducting research. His prime areas of interest include AI, ML, and programming. Dr. Khandare has also worked on research projects funded by government bodies and has offered consultancy in data science and related fields. He continues to guide many undergraduate and postgraduate students in their research and project work.