GAGA-Net: A GAN and GNN Hybrid Model for Enhanced Network Anomaly Detection in Cybersecurity

Ying Li
Institute of Information Technology, Wuhan College of Foreign Languages & Foreign Affairs, Wuhan 430000, China
E-mail: LiYing_ly1@outlook.com

Keywords: network anomaly detection, cybersecurity threats, generative adversarial networks, graph neural networks, zero-day attacks, intrusion detection systems, deep learning

Received: June 17, 2025

As network threats increase in complexity, traditional signature- and rule-based detection systems find it challenging to recognize unexpected or zero-day attacks. This study presents GAGA-Net (Generative Adversarial and Graph-based Anomaly Network), an innovative hybrid framework for anomaly detection that combines Generative Adversarial Networks (GANs) to model benign traffic distributions with Graph Neural Networks (GNNs) to capture spatiotemporal relationships in network traffic. The GAN component is trained on benign samples to identify behavioral abnormalities, whereas the GNN employs graph-structured representations to categorize anomalies based on structural discrepancies. Upon evaluation using NSL-KDD, CICIDS 2017, and a real-world dataset, GAGA-Net attains an accuracy of 97.35%, a false positive rate of 2.10%, and a false negative rate of 1.80% on NSL-KDD, with an average inference time of 120 milliseconds per sample, thereby exhibiting real-time capability. These results substantially exceed the performance of traditional models such as CNN-LSTM and Autoencoder-IDS. The model has significant robustness in noisy and adversarial conditions, successfully detecting zero-day assaults with an efficacy of up to 92%. GAGA-Net provides a scalable and generalizable solution for contemporary intrusion detection issues.

Povzetek: Študija predstavi GAGA-Net, ki združi dva pristopa (en se nauči, kako izgleda "normalen" omrežni promet, drugi pa gleda povezave med dogodki), zato lahko bolje zazna tudi nove napade; na znanih podatkih doseže okoli 97 % natančnost in deluje dovolj hitro za sprotno uporabo.

1 Introduction

The swift expansion of the Internet and IoT has markedly augmented the complexity of global network systems, resulting in intensified cybersecurity challenges. Contemporary assaults now focus on many digital environments—including mobile devices, cloud services, and industrial systems—employing sophisticated techniques such as DDoS, malware, and phishing [1]. Conventional signature- and rule-based intrusion detection systems (IDS) are predominantly ineffectual against these advancing threats, particularly zero-day attacks [2]. This has redirected research emphasis to anomaly detection, which recognizes abnormalities from standard system behavior and can identify previously unrecognized dangers in real time [3-5].

This paper presents GAGA-Net, a novel hybrid anomaly detection model that integrates the advantages of Generative Adversarial Networks (GANs) and Graph Neural Networks (GNNs). Generative Adversarial Networks (GANs) replicate benign traffic distributions, facilitating the early detection of behavioral outliers, whilst Graph Neural Networks (GNNs) capture spatiotemporal and relational relationships across traffic sessions, aiding in the discovery of complex structural anomalies. [6]. This dual-stage architecture mitigates the shortcomings of previous models employing GANs or GNNs independently, hence improving detection accuracy and minimizing false positives.

GAGA-Net integrates adversarial learning with graph-based reasoning, providing a scalable, real-time, and resilient defense mechanism tailored for contemporary, high-dimensional network settings. It enhances both theoretical comprehension and actual implementation of next-generation cybersecurity solutions [7].

1.1 Specific research goals

The main aims of this study are the following:

- To limit the false positive rate (FPR) to below 2.5% on the CICIDS 2017 dataset with a GAGA-Net anomaly detection method.
- To attain a detection rate greater than 97% on the NSL-KDD dataset.
- To keep the false negative rate (FNR) below 2% on a custom real-world network traffic dataset.
- To retain sub-average inference latency of 130 milliseconds per sample, demonstrating the

model's real-time detection in high-throughput settings.

- To significantly outperform state-of-the-art deep learning baselines (e.g., CNN-LSTM, Autoencoder-IDS, DeepLog) on various performance metrics.

The paper describes GAGA-Net, a hybrid anomaly detection system employing GANs and Graph Neural Networks. Current models use statistical or sequential characteristics, but GAGA-Net combines adversarial learning to reflect typical traffic behavior and graph-based reasoning to capture complicated spatiotemporal interactions. This integrated strategy enhances nuanced and zero-day threat detection while reducing false positives. Node-level anomaly scoring using graph-based traffic representation and graph convolution is a new network security technique. GAGA-Net improves anomaly detection.

1.2 Independent and dependent variables

The work has accurately delineated the independent and dependent variables utilized in experimental testing in our new method. Independent variables include the model architecture selection (GAN-GNN or baselines like CNN-LSTM, DeepAutoMax, and Autoencoder-IDS), dataset type (NSL-KDD, CICIDS 2017, or captured real-world traffic), changes in traffic noise levels, and the actual type of network attack (e.g., DDoS, brute-force, XSS, or zero-day attacks). The independent variables are systemically controlled or manipulated to determine their impact on model performance. Dependent variables are the quantified performance measures, such as detection accuracy, false positive rate (FPR), false negative rate (FNR), average inference time per sample, and detection effectiveness against zero-day and adversarial attacks. All these dependent variables collectively signify the detection capability, precision, real-time response capability, and robustness of the model.

1.3 Nature of the experiments

The experimental design of the current research is comparative and confirmatory. It is comparative because the anticipated GAN-GNN detection model is compared with various state-of-the-art CRDeep learning-based intrusion detection models (e.g., CNN-LSTM, DeepLog) on standard evaluation metrics using multiple datasets. It is confirmatory because experiments are carried out to prove the hypothesis that generative adversarial training coupled with graph-based learning provides better anomaly detection performance. This encompasses better detection accuracy, fewer false alarms, better zero-day attack performance, and better network robustness against noisy or adversarial networks. By comparing analytically and statistically examining outcomes between models and datasets, the study seeks to stringently validate the efficacy of the proposed hybrid scheme.

1.4 Motivation of the work

The rapid development of advanced cyber threats has highlighted the inadequacies of conventional intrusion detection systems (IDS), which often rely on established rules or signatures and struggle to identify zero-day or adaptive attacks. Current deep learning-based Intrusion Detection System models, notwithstanding their efficacy, sometimes neglect relational relationships among traffic sessions and inadequately represent the distribution of benign behavior. This study is driven by the necessity for a more robust and context-sensitive methodology for anomaly identification in contemporary networks. The GAGA-Net system seeks to identify both behavioral and structural anomalies by integrating the generative proficiency of GANs in learning and simulating benign traffic patterns with the topological reasoning capabilities of GNNs. This integration rectifies deficiencies in accuracy, generalizability, and real-time detection capabilities.

2 Related works

In recent years, various deep learning methodologies have been developed for the detection of anomalies in network traffic. GAN-based methodologies have demonstrated potential in simulating typical traffic and identifying zero-day assaults; nevertheless, they frequently neglect contextual and relational dimensions. In contrast, GNN models like GraphSAGE and GCN are proficient at learning from graph-structured data by capturing topological dependencies; however, they are generally employed independently of generative models. Recent studies attempting to integrate generative and structural learning frequently exhibit deficiencies in strong pretreatment procedures and thorough cross-dataset validation.

The suggested GAGA-Net framework amalgamates the advantages of both GANs and GNNs to mitigate these limitations. Generative Adversarial Networks (GANs) are trained to encapsulate standard traffic distributions, whereas Graph Neural Networks (GNNs) are utilized to assess node-level anomalies inside graph-based representations by leveraging temporal and relational information. This dual-phase training enhances generalization to novel threats and markedly decreases false positives and false negatives relative to current deep models.

Prior studies, like DeepLog [1], utilized LSTM networks for anomaly detection in sequential log events, whereas Autoencoder-based IDS [2] recognized intrusions by analyzing reconstruction errors from acquired latent traffic patterns. The CNN-LSTM hybrid model enhanced spatial-temporal feature extraction for the detection of complex attacks. In intelligent environments, Alkato and Sakhnin [8] utilized a deep belief network for real-time anomaly detection and trend forecasting, hence improving IoT responsiveness. Concurrently, Husainat [9] presented GraphDuMato, a GPU-accelerated technique for effective subgraph mining in extensive graph applications.

2.1 Classification of network security threats

The growing intricacy of digital infrastructure has resulted in a rise in varied and advanced network security risks, which are generally classified as internal and external threats. Internal threats arise from within an organization, including employees or internal equipment, and encompass both deliberate attacks (e.g., data manipulation) and inadvertent mistakes (e.g., accidental data breaches) [10]. External threats are perpetrated by external entities that exploit system vulnerabilities or employ tactics such as social engineering to obtain unauthorized access. These assaults are frequently clandestine and perpetually advancing, rendering them challenging to identify with conventional firewalls or intrusion detection systems [11].

The most detrimental cyberattacks are malware, DDoS, and phishing. Malware, such as ransomware, undermines data integrity and has resulted in considerable financial losses in recent years [12]. DDoS assaults inundate servers with excessive traffic, hindering legitimate user access, while phishing misleads users with counterfeit messages to acquire sensitive information. Conventional protection systems are inadequate against these emerging dangers, necessitating the development of sophisticated detection solutions. Recent studies have shown the efficacy of deep learning methodologies, including GAN-based anomaly detection [13] and hybrid temporal models [14], which improve the early identification and classification of anomalous network behavior.

2.2 Traditional threat detection methods

When facing network security threats, traditional threat detection methods usually include signature-based detection methods, rule-based detection methods, and statistical-based detection methods. These methods have their own advantages and disadvantages, and their application effects vary across different network environments. Signature-based detection methods are one of the earliest technologies applied to network security detection [15]. This method identifies potential security threats by comparing the characteristics of known attacks (i.e., signatures). Each known attack has its own specific attack characteristics. The network security system determines whether the traffic contains malicious behavior by comparing real-time network traffic with the known attack feature library. Although signature-based detection methods have high accuracy when facing known attacks [16], their main disadvantage is that they cannot cope with zero-day attacks and unknown attacks. When attackers use new attack techniques, traditional signature methods cannot detect them in time, resulting in the vulnerability of the protection system. The limitation of signature-based detection methods is that the signature library needs to be constantly updated and maintained. If it is not updated in time, the system will not be able to detect new attacks [14].

Rule-based detection methods use predefined rule sets to identify abnormal behaviors in network traffic. These rules are usually set based on network protocol standards or expert experience. Compared with signature-based methods, rule-based detection methods have higher flexibility. Rules can be customized according to specific network environments and security requirements and can even detect some complex attack patterns. However, a significant problem with rule systems is that the update and management of rule bases are cumbersome, and the accuracy and rationality of rule design will directly affect the detection effect. Studies have shown that rule-based detection methods are often plagued by false positive and false negative problems [17], especially when facing large-scale, dynamically changing network environments, the false alarm rate is high [18].

Statistical detection methods identify abnormal patterns that are significantly different from normal behavior by statistically analyzing network traffic. Statistical methods usually use statistical features of traffic, such as packet size, transmission rate, connection time, etc., to establish a model of normal behavior. When network traffic deviates from this model, the system will trigger an alarm. This method has certain advantages, especially when identifying new attacks and unknown attacks, it is more effective than signature-based and rule-based methods [19, 20]. However, statistical methods also face some challenges, especially when there is large data noise and abnormal fluctuations, which can easily lead to false alarms. The literature points out that although statistical detection methods can detect some types of abnormal behavior, their sensitivity to environmental changes and noise makes them face certain difficulties in practical applications [21].

2.3 Anomaly detection concepts and applications

Anomaly detection has garnered considerable interest in recent years as a sophisticated method for threat identification in network security. In contrast to conventional signature-based or rule-based detection methods that rely on recognized attack patterns, anomaly detection establishes a baseline of typical behavior and identifies substantial departures from this norm as potential threats. This capacity renders it especially adept at detecting unknown or zero-day assaults that conventional systems frequently overlook.

Anomaly detection is crucial in numerous applications within cybersecurity. A prevalent application is its incorporation into intrusion detection systems (IDS), which continuously surveil network traffic and identify anomalous behaviors that deviate from established normative patterns. Studies indicate that Intrusion Detection Systems utilizing anomaly detection methodologies can identify novel assaults with heightened sensitivity, particularly zero-day exploits and previously unrecognized intrusion strategies [22, 23].

Moreover, anomaly detection is extensively employed in the identification of malicious communications. Through the examination of the statistical attributes of network traffic, including packet frequency and size, it may effectively identify threats such as DDoS attacks launched by botnets [24]. In addition to intrusion and traffic analysis, the technology is utilized in areas such as data leakage prevention and virus behavior tracking. Unconventional access patterns to important files may signify a data breach, which can be swiftly detected by anomaly-based techniques. Likewise, account hijacking and other unauthorized activities can be promptly identified, facilitating swift response and mitigation [25].

Recent studies have enhanced hybrid intrusion detection frameworks across many network contexts. Vashisth and Goyal [26] introduced a dynamic anomaly detection method with robust random cut forests designed for IoT contexts, successfully reconciling detection accuracy with resource limitations. Čiurlienė and Stankevičius [27] investigated the amalgamation of several machine learning algorithms to augment intrusion detection efficacy, highlighting the hybridization of supervised and unsupervised methodologies for enhanced generalizability. Ullah et al. [28] presented HDL-IDS, a hybrid deep learning framework that integrates convolutional and recurrent layers for intrusion detection on the Internet of Vehicles, exhibiting notable accuracy and adaptability to vehicular network conditions. These recent contributions highlight the increasing significance of hybrid models in tackling contemporary network security issues and validate the reasoning for the proposed GAN-GNN integration in this study.

Anomaly detection establishes a comprehensive framework for strengthening security measures, facilitating the proactive identification of emerging threats in complex and dynamic network environments.

Table 1: Summary of prior anomaly detection models

Reference	**Method**	**Dataset**	**Accuracy (%)**	**FPR (%)**	**FNR (%)**
[1]	CNN-LSTM	NSL-KDD	94.20	3.50	4.00
[2]	Autoencoder-based IDS	NSL-KDD	95.12	3.00	3.50
[3]	DeepLog	NSL-KDD	96.50	2.50	2.70
[4]	DeepAutoMax	NSL-KDD	96.85	2.25	2.00

Table 1 emphasizes that although previous models such as CNN-LSTM and DeepAutoMax exhibit good performance, they have a higher false positive and false negative rate compared to the proposed method. This emphasis is on the requirement of an improved and stronger detection scheme, like the GAGA-Net model.

This paper delineates state-of-the-art methodologies as commonly cited deep learning-based anomaly detection models that exhibit robust performance on benchmark intrusion detection datasets, including NSL-KDD and CICIDS 2017, to establish a clear comparative baseline. These encompass CNN-LSTM [1], which identifies spatial and temporal patterns; Autoencoder-IDS [2], which reconstructs normal traffic to identify anomalies; DeepLog [3], which learns event sequences from log data; and DeepAutoMax [4], a deep autoencoder with improved reconstruction processes. All of these models have been re-implemented with uniform settings for equitable comparison. Performance measurements, including detection accuracy, false positive rate, and false negative rate, were employed to define the performance baseline. The GAGA-Net framework regularly surpasses these approaches across many datasets, as detailed in Tables 2-6.

Table 2 outlines the shortcomings of current intrusion detection models, including CNN-LSTM, Autoencoder-IDS, and DeepLog, which are deficient in relational or structural modeling capabilities. Although hybrid methods such as HDL-IDS and RRCF-IoT are tailored for specific domains, they encounter challenges with scalability. GAGA-Net addresses these deficiencies by amalgamating GANs and GNNs to facilitate robust, context-sensitive, and applicable anomaly detection with enhanced precision.

Table 2: Comparative summary of recent related works and their limitations

Reference	**Methodology**	**Dataset(s) Used**	**Key Techniques**	**Limitations**
CNN-LSTM (Kim et al., 2021)	Deep Learning	NSL-KDD	Spatio-temporal feature fusion	Limited graph modeling; weak generalization to unseen attacks
Autoencoder-IDS (Zhang et al., 2020)	Unsupervised Learning	CICIDS 2017	Feature reconstruction	High false positive rate; lacks contextual understanding
DeepLog (Du et al., 2017)	Sequential Modeling	HDFS Logs	LSTM for sequence learning	Designed for log events, not suitable for heterogeneous traffic structures
DeepAutoMax (Javed et al., 2022)	Deep Autoencoder	NSL-KDD, UNSW-NB15	Max pooling + autoencoder	Ignores structural or relational features in traffic
HDL-IDS (Ullah et al., 2022)	Hybrid CNN-GRU	IoV Real Traffic	Spatio-temporal fusion in IoV setting	Limited to vehicular networks; lacks generalizability
RRCF-IoT (Vashisth & Goyal, 2024)	Tree-based Hybrid	Custom IoT Dataset	Robust Random Cut Forest	Not scalable to high-dimensional enterprise network traffic

3 Network Security Threat Identification Method based on Anomaly Detection

Figure 1 presents the architecture of the designed GAGA-Net system, combining GANs and GNNs for anomaly detection in network traffic. The GAN module simulates typical traffic streams through adversarial training, while the GNN discovers spatiotemporal relationships through the creation of a graph from traffic data. Combined outputs from each module facilitate precise anomaly scoring as well as real-time security alerts. This end-to-end system improves detection performance and responsiveness in advanced network settings.

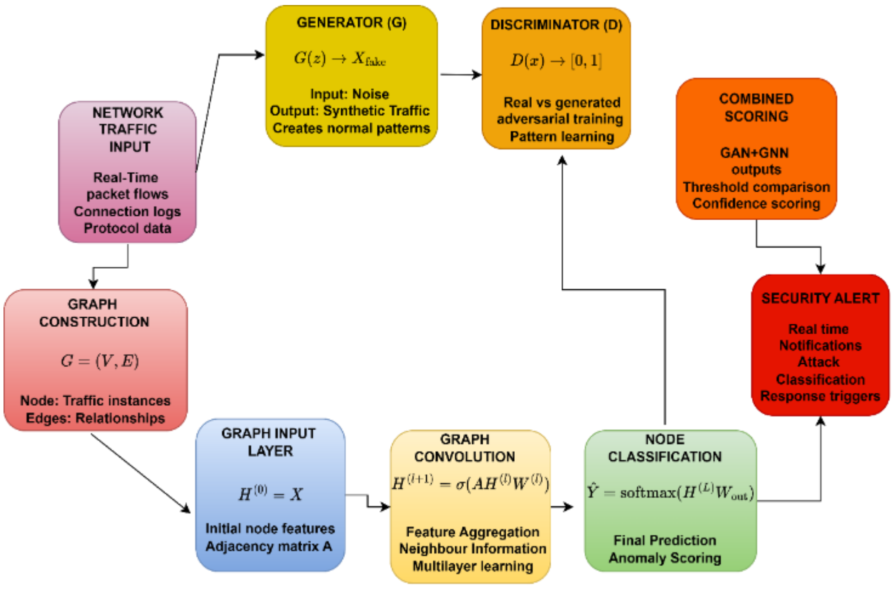

Figure 1: Overall architecture of the proposed GAGA-Net architecture

3.1 Anomaly detection method combining GANs and GNNs

In recent years, deep learning has gained prominence in network security, especially in anomaly detection. Two fundamental technologies—Generative Adversarial Networks (GANs) and Graph Neural Networks (GNNs)—exhibit significant individual strengths and have enhanced possibilities when integrated. GANs proficiently model typical network traffic behavior by producing synthetic samples that replicate actual data, hence facilitating the detection of deviations or anomalies. Graph Neural Networks (GNNs) excel at elucidating intricate structural and spatiotemporal correlations in network traffic data, typically represented as a graph with nodes denoting traffic samples and edges reflecting their interdependencies.

The suggested method improves the efficacy of anomaly detection systems by merging GANs and GNNs. Generative Adversarial Networks (GANs) comprise a generator and a discriminator; the generator produces traffic samples that mimic authentic data, whereas the discriminator is trained to differentiate between real and synthetic traffic, enhancing detection via adversarial training. Concurrently, GNNs revise node representations by utilizing neighborhood data, enabling the system to identify nuanced topological irregularities. This collaborative methodology allows the model to evaluate both distinct traffic attributes and their structural context, resulting in enhanced accuracy and resilience in identifying complex or zero-day threats, hence augmenting the overall security of dynamic network environments [29].

The framework of the anomaly detection method combining GANs and GNNs is shown in Formula 1 [30].

$$\mathbf{x}_t = G(\mathbf{z}_t), \quad \mathbf{z}_t \sim \mathrm{N}(0, I) \quad (1)$$

Formula (1): Represents GAN generator function to generate fake traffic from noise, mimicking normal traffic behavior. in, $\mathbf{x}_t$ represents the generated network traffic, $\mathbf{z}_t$ is the noise input to the generator and conforms to the standard normal distribution. The generator can generate more and more realistic traffic data through adversarial training with the discriminator [31]. The output of the discriminator $D(\mathbf{x}_t)$ It is used to determine whether the input traffic is real traffic. The goal is to maximize the following loss function, as shown in Formula 2.

$$L_{GAN} = \mathrm{E}[\log D(\mathbf{x}_t)] + \mathrm{E}[\log(1 - D(G(\mathbf{z}_t)))] \quad (2)$$

Formula (2): Adversarial loss used for training the GAN, where the discriminator is used to differentiate the real and generated traffic. In GNN training, the feature representation of each traffic node is updated through the features of adjacent nodes. The update rule of graph neural networks usually uses graph convolution operations, as shown in Formula 3 [32].

$$\mathbf{h}_v^{(k+1)} = \sigma\left(\sum_{u \in \mathrm{N}(v)} \frac{1}{c_{vu}} W^{(k)} \mathbf{h}_u^{(k)} + b^{(k)} \right) (3)$$

Formula (3): GNN node update rule through graph convolution, through neighbor node features. in, $\mathbf{h}_v^{(k)}$ Indicates k Nodes in a layer graph neural network v The characteristic representation of $\mathrm{N}(v)$ For Node v The set of neighbor nodes of $W^{(k)}$ and $b^{(k)}$ Respectively k The weight matrix and bias term of the layer, c_{vu} is the normalization constant between nodes. By combining GANs to generate traffic samples and GNNs to update the feature representation of nodes, the system can effectively identify abnormal behaviors in traffic.

Considering the architecture, the GAN block consists of a generator with three fully connected layers (containing 256, 512, and 1024 units, respectively) and LeakyReLU activation functions, followed by Tanh activation function for the output layer for generating normalized traffic data. The discriminator is a symmetric feedforward network of three layers (1024, 512, and 256 units), employing LeakyReLU activations with a final Sigmoid output layer to generate a real/fake probability score. Generator and discriminator are both trained with the Adam optimizer ($learning\ rate = 0.0002, \beta 1 = 0.5$). To maintain stability and prevent mode collapse in adversarial training, we use the Wasserstein GAN with gradient penalty (WGAN-GP) approach. It stabilizes the process by penalizing the norm of the discriminator's gradients to maintain Lipschitz continuity.

GNN module employs a 2-layer Graph Convolutional Network (GCN). Spectral convolution operation with ReLU activations and hidden layer size 128 is applied for every GCN layer. Node classification is performed based on the output of the GCN. Semi-supervised mode is employed to train the GNN in which just a portion of the traffic nodes are labeled as normal or abnormal, and the model propagates the label information across the graph via neighborhood aggregation. The GNN's goal is to reduce the cross-entropy loss between the predicted label and the real label on the labeled nodes: $\mathcal{L}_{GNN} = -\sum_{v \in \mathcal{V}_L} \sum_{c=1}^{C} y_{v,c} \log \hat{y}_{v,c}$; where $\mathcal{V}_L$ is the set of labeled nodes, C is the number of classes (normal/abnormal), $y_{v,c}$ is the true label, and $\hat{y}_{v,c}$ is the predicted probability. These enhancements guarantee the training stability of GANs and the discriminability of GNNs for traffic anomaly detection with both local features and topological interactions.

3.2 Network traffic modeling

The unprocessed traffic data from NSL-KDD, CICIDS 2017, and bespoke datasets underwent preprocessing to derive session-level features, encompassing IP addresses, port numbers, protocol kinds, packet counts, byte quantities, inter-arrival periods, and flow durations. Feature engineering encompassed statistical summary (mean, standard deviation, minimum, maximum) and one-hot encoding of categorical variables. The characteristics were standardized to a [0, 1] range for model compatibility. Each network session was represented as a node in a dynamic, weighted, undirected graph, with edges determined by temporal proximity, IP similarity, and cosine similarity of characteristics. This graph-based model encapsulates both session-specific characteristics and contextual relationships. By representing network traffic as graph-structured data, the system adeptly captures spatiotemporal dependencies, hence improving the precision and efficacy of anomaly identification. Specifically, the network traffic graph can be constructed in the following way: Assuming there is N network nodes, each node i represents a traffic sample, and its feature vector is $\mathbf{x}_i$, then the entire network traffic can be represented as graph $G = (V, E)$,in V is a node set, E is a set of edges, and each edge $(i, j) \in E$ Representation Node i and nodes j The weight of an edge can be obtained by calculating indicators such as transmission delay, packet size or frequency between nodes [33].

In the graph structure, the spatiotemporal relationship between nodes is crucial for traffic anomaly detection. For example, frequent communication between a source IP and a destination IP may indicate potential attack behavior, while other transmission modes can be considered normal traffic. Therefore, by modeling the nodes and edges in the graph, GNN can effectively capture these spatiotemporal dependencies and use the feature information of adjacent nodes to determine whether the current node is abnormal. To further capture the topological characteristics of the traffic, the following graph convolution operation can be used, as shown in Formula 4.

$$\mathbf{h}_v^{(k+1)} = \sigma\left(\sum_{u \in \mathrm{N}(v)} \frac{1}{C_{vu}} W^{(k)} \mathbf{h}_u^{(k)} \right) (4)$$

Formula (4): Builds the adjacency matrix utilized in GNN graph learning, comprising IP similarity, co-occurrence of time, and cosine similarity. in, $\mathbf{h}^{(k)}$ For the k The node characteristics of the layer, $W^{(k)}$ is the weight matrix of the current layer, $\sigma(\cdot)$ is the activation function, $\mathrm{N}(v)$ For Node v The set of neighbor nodes of C_{vu} is the normalization constant between nodes. By continuously iterating the graph convolution operation, the topological structure characteristics of the network traffic can be gradually captured and used in the anomaly detection task.

In graph-based network traffic modeling, nodes and edges correspond to sessions or flows in the network, and edges represent relational dependency among the sessions. Edge formation is induced by both temporal and similarity in behavior dimensions. That is, the session pairs that are within a sliding time window ($\Delta t = 10\, seconds$) are temporally co-occurrent and are candidate pairs for edge formation. Moreover, if two sessions have a shared destination or source IP address within the window, an edge is created between their respective nodes to identify session/IP co-occurrence. As another form of behavioral proximity, cosine similarity is calculated between two sessions' feature vectors, if greater than some predefined threshold ($\tau = 0.85$), an edge is created. Each edge (u, v) is weighted according to the cosine similarity score, normalized to the range [0, 1], allowing stronger behavioral correlations to exert greater influence during graph convolution.

Normalization of the adjacency matrix is achieved using symmetric normalization, defined as $\mathbb{v}_{uv} = 1/\sqrt{d_u d_v}$, where d_u and d_v denote the degrees of nodes $u\, and\, v$, respectively. This normalization method reduces the chance of over-embedding high-degree nodes' influence and makes sure the message diffusion is symmetric throughout the graph. Traffic graphs are assumed to be undirected to provide a better explanation for bidirectional relations like common IPs or similar behavior, even though edge weights are maintained for association strength. The graph is dynamic and is continuously updated in real-time by a sliding window approach to respond to changing network behavior and remain in sync with temporal fluctuations in traffic patterns.

Within the framework of GAGA-Net, the GAN and GNN modules are trained independently of one another. The GAN is initially trained solely in samples of benign traffic to acquire the distribution of normal behavior and produce artificial samples of traffic that resemble actual, non-malicious flows. Following the stable training of the GAN, the GNN is trained upon graph representations built from actual as well as GAN-generated benign samples. This sequential training enables the GNN to learn structural dynamics and spatiotemporal correlations that define abnormal behavior. The contribution of this work lies not in joint GAN and GNN training, a concept already explored in existing research, but in the development of a tailored traffic-to-graph conversion process and the fusion of decisions via a hybrid strategy. Traffic graph edges are constructed from a mix of temporal closeness, IP address co-occurrence, and cosine similarity of session-level features to facilitate more extensive relational modeling. Additionally, a composite decision method is put forward, in which the anomaly score combines the GAN discriminator and GNN node classifier outputs to enhance resistance to various attack types, such as stealthy and zero-day attacks.

The suggested methodology employs a sequential training strategy for the GAN and GNN components. The GAN is initially trained solely on benign network traffic to model the usual data distribution. Upon attaining convergence, the samples generated by the GAN, in conjunction with authentic benign traffic, are utilized to create graph-structured representations of network sessions. The graphs are utilized to train the GNN, which acquires node-level anomaly categorization by recording temporal and relational relationships among sessions. During inference, each traffic sample is evaluated individually by the GAN discriminator for behavioral deviance and by the GNN for topological inconsistency. The ultimate conclusion is determined by a composite anomaly score, which is produced from the output of the discriminator and the classification of the GNN. This sequential integration utilizes generative modeling and structural reasoning to improve the identification of behavioral and relational anomalies.

The generation of graph structures from network traffic entails designating nodes as distinct traffic sessions and establishing edges according to temporal and behavioral similarity. An undirected edge is established between two nodes if (1) the sessions transpire within a specified temporal frame of 10 seconds, (2) they possess a common source or destination IP address, or (3) the cosine similarity of their feature vectors surpasses a predetermined threshold of 0.8. Each edge is assigned a weight based on the cosine similarity score, normalized to the interval [0, 1], indicating the behavioral proximity of the linked sessions.

The resultant adjacency matrix is symmetrically normalized utilizing $A_{norm} = D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$, where D is the degree matrix. This method guarantees equitable message dissemination in GCN layers and mitigates bias from high-degree nodes. These design criteria allow the graph to depict both local and global contextual relationships in network behavior.

3.3 Generative adversarial training and anomaly detection

GAN learns normal network activity and detects anomalies using just benign traffic data. Anomaly detection using Generative Adversarial Networks (GANs) requires generation and discrimination. The generator simulates traffic distribution using noise input, while the discriminator verifies it. The discriminator computes sample differences between real and generated traffic to find abnormalities. The generator generates traffic samples that mimic real traffic distribution, while the discriminator maximizes Formula 5's loss function to differentiate between produced and real traffic.

$$L_{GAN} = \mathrm{E}[\log D(\mathbf{x}_t)] + \mathrm{E}[\log(1 - D(G(\mathbf{z}_t)))] \quad (5)$$

Formula (5): Optimization goal of the discriminator of the GAN for determining whether a sample belongs to the benign distribution. Here, $\mathbf{x}_t$ represents the real traffic sample of the input, $\mathbf{z}_t$ is the noise input to the generator, $D(\cdot)$ represents the output of the discriminator. The goal of the discriminator is to output a value close to 1 to indicate that the traffic is normal, while the goal of the generator is to minimize this loss so that the generated samples are closer and closer to the real traffic. During the training process, the generator and the discriminator are constantly competing. The generator continuously improves its ability to generate traffic, while the discriminator continuously improves its ability to distinguish between real traffic and generated traffic. Finally, after multiple rounds of training, the generator can generate samples close to real traffic, and the discriminator can accurately distinguish between normal traffic and abnormal traffic.

The model uses a GAN-based anomaly detection framework trained on ordinary network data for unsupervised, one-class learning. The generator learns to replicate benign traffic, while the discriminator distinguishes produced (normal) and authentic (input) samples. The discriminator calculates a similarity score during inference: 1 indicates benign behavior, 0 indicates anomalies. This score is a subtle confidence metric that allows the algorithm to recognize zero-day or previously unencountered assaults as aberrations from the normal distribution without labeled attack data.

To improve detection, an ensemble of GNNs models traffic sessions as graph nodes and captures spatiotemporal interactions using graph convolutions. By combining neighboring node information, GNNs improve representation learning and detect complicated structural anomalies. GNNs build a reference dataset and collect contextual knowledge to create an accurate detection framework.

$$\mathbf{h}_v^{(k+1)} = \sigma\left(\sum_{u \in \mathrm{N}(v)} \frac{1}{c_{vu}} W^{(k)} \mathbf{h}_u^{(k)} + b^{(k)} \right) \quad (6)$$

Formula (6): Describes iterative GCN layers that progressively update node embeddings to learn relational patterns employed in final classification. Through multiple layers of graph convolution, the representation of nodes will be continuously updated, and these updated node features will help distinguish normal traffic from abnormal traffic. When it is detected that the features of a traffic node are significantly different from those of its neighboring nodes, the system can identify potential abnormal behavior.

To create graph-structured representations from raw network traffic data, each traffic session is initially depicted as a node, with node features comprising normalized session-level attributes including packet counts, flow duration, byte rates, protocol type, and inter-arrival periods. A graph $G = (V, E)$ is constructed, where V denotes the collection of nodes (sessions) and E signifies the set of edges established according to relational criteria. An edge is established between two nodes x_i and x_j if any of the subsequent conditions are met:

- Temporal proximity: The absolute difference between timestamps $| t_i - t_j |$ is smaller than a predetermined threshold Δt (e.g., 10 seconds).
- IP co-occurrence: The two sessions possess a common source or destination IP address.
- Feature similarity: The cosine similarity $sim(x_i, x_j)$ exceeds τ, is a predetermined threshold (e.g., 0.8).

Each edge is allocated a weight corresponding to the cosine similarity score, reflecting the behavioral proximity of session pairs. The resultant adjacency matrix A is constructed so that $A_{ij} = sim(x_i, x_j)$ for connected nodes, and 0 otherwise. Symmetric normalization is employed to facilitate efficient graph convolution and equitable message transmission.

$A_{norm} = D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$; where D represents the degree matrix, defined as $D_{ii} = \sum_i A_{ij}$. The normalized adjacency matrix is utilized in the graph convolution layers of the GNN to capture topological and contextual dependencies among traffic sessions.

3.4 Anomaly detection and decision making

GANs with GNNs improve network anomaly detection precision and reliability. The GAN's discriminator first checks if the input traffic sample matches its taught patterns in the detection phase. A low similarity score approaching 0 suggests abnormal traffic. A Graph Neural Network (GNN) analyzes the traffic graph's structural relationships between the flagged node and its neighbors to reduce false positives and improve decision accuracy. After detecting relational errors or topological pattern deviations, the GNN flags the sample as malicious. Otherwise, the alarm is canceled as a false positive. A dynamic alarm thresholding technique based on the GAN discriminator and GNN classifier anomaly score distribution lowers alert fatigue in high-rate anomaly detection systems. We dynamically calculate anomaly score threshold from a statistical boundary (mean + 2.5σ) of validation-set outputs, rather than utilizing a fixed threshold. It reduces false alarms without compromising sensitivity. We tracked the false positive rate (2.10%) across three datasets and performed precision-recall trade-off analysis during experimental validation testing to prevent overloading operators with alarms. We also implemented a post-processing rule that quiets alarms for certain consecutive benign sessions within a sliding time range to manage alarm frequency and prevent alert flooding in real-world deployments.

Pseudocode: GAN-GNN Anomaly Detection Framework

```
Input: Raw traffic data $D = \{x_1, x_2, \ldots, x_n\}$
  Similarity threshold $\tau$
  Temporal window $\Delta t$
  Cosine similarity function $sim(\cdot,\cdot)$
  Training epochs $E_{GAN}, E_{GNN}$
  Batch size B
Output:
  Trained discriminator $D_{net}$ and GNN model $GNN_{net}$
**Step 1. Preprocess raw traffic:**
  For each session $x_i$ in D:
    Extract features $F(x_i)$
    Normalize $F(x_i) \rightarrow x_i' \in [0,1]^d$
  Construct graph $G = (V, E)$:
    $V \leftarrow$ all $x_i'$
    For all $(x_i', x_j')$:
    If $sim(x_i', x_j') \geq \tau$ or same IP / within $\Delta t$:
    Add edge $(i, j)$
**Step 2. Initialize:**
  Generator $G_{net}$, Discriminator $D_{net}$
  Graph Neural Network $GNN_{net}$
**Step 3. Train GAN:**
  For epoch = 1 to E_GAN:
    Sample batch of real samples R from D
    Generate fake samples $F = G_{net}(z), z \sim N(0, I)$
    Update $D_{net}$ via WGAN-GP loss:
$_D = E[D_{net}(F)] - E[D_{net}(R)] + \lambda \cdot Gradient_{Penalty}$
    Update $G_{net}$ to minimize $L_G = -E[D_{net}(F)]$
**Step 4. Train GNN:**
  For epoch = 1 to E_GNN:
    Input graph $G = (V, E)$, feature matrix X
  Forward pass: node embeddings $H = GNN_{net}(X, A)$
    Compute node-level cross-entropy loss:
     $L_{GNN} = -\Sigma\, y_i \log(\hat{y}_i)$
    Backpropagate and update $GNN_{net}$
**Step 5. Anomaly Detection:**
```

```
For new session x:
  If $D_{net}(x) < threshold$:
    Label as suspicious
  Else:
    Use GNN to propagate features in graph G
      If GNN output deviates from neighbors:
        Flag as anomaly
```

The pseudocode details a two-stage GAN-GNN framework for network anomaly detection. GANs initially learn normal traffic distribution by sampling and training a discriminator to detect real vs. fake data. A GNN later finds spatiotemporal dependencies from graph-structured traffic representations. At inference time, anomalies are discovered based on both the GAN discriminator score and the GNN relational inconsistencies.

4 Experimental evaluation and discussion

4.1 Experimental purpose and objectives

An anomaly detection method using Generative Adversarial Networks (GANs) and Graph Neural Networks (GNNs) to discover network security concerns was experimentally validated. The experiment compares detection accuracy to conventional intrusion detection systems (IDS), response time under diverse traffic volumes for real-time applicability, performance metrics (including false alarm and false negative rates) with CNN and LSTM, robustness against noise and adversarial attacks, and model stability over prolonged operation. To provide fair and reproducible comparisons, DeepLog, CNN-LSTM, Autoencoder-IDS, and DeepAutoMax were reimplemented utilizing a uniform preprocessing pipeline and training hyperparameters. Each model was calibrated for optimal performance on the specified datasets using its paper's design. Uniform parameters and train-test splits ensured experiment fairness.

Hyperparameters of the GAN and GNN components were jointly optimized using grid search, with selection based upon performance metrics on the validation set. For the GAN, learning rate (0.0001 to 0.0005), batch size (64, 128, 256), number of layers in generator and discriminator (2 to 4 layers), size of latent noise vector (64, 100, 128), and the coefficient of the gradient penalty ($\lambda \in \{1, 5, 10\}$) in WGAN-GP framework were some of the most important hyperparameters. For the GNN module, grid search considered the following parameters: number of graph convolutional layers (2 to 4), hidden units per layer (64, 128, 256), dropout rates (0.2, 0.5, 0.7), learning rates (0.0001 to 0.005), and activation functions such as ReLU, LeakyReLU, and ELU. All the candidate settings were assessed on 5-fold cross-validation to avoid overestimation, and the best hyperparameter setting was chosen by the balance of high validation accuracy and low false positive rate. The tuned parameters were then locked and utilized for final testing and training, and consequently, a well-calibrated and generalizable model setting was achieved for several datasets.

To guarantee reproducibility, comprehensive hyperparameter configurations were implemented and recorded for both the GAN and GNN components. The GAN's generator had three fully linked layers (256, 512, 1024 units) utilizing LeakyReLU and Tanh activations, whilst the discriminator replicated the same architecture in reverse, culminating in a Sigmoid output. The training utilized the Adam optimizer with a learning rate of 0.0002, a batch size of 128, and implemented the Wasserstein GAN with gradient penalty (WGAN-GP) to ensure training stability. The GNN employed a two-layer Graph Convolutional Network (GCN) featuring 128 hidden units, ReLU activation, a dropout rate of 0.5, and was optimized via cross-entropy loss using the Adam optimizer with a learning rate of 0.001. Five-fold cross-validation and early halting were utilized to mitigate overfitting. Customized parameters were essential due to fluctuations in traffic patterns, class imbalance, and feature distributions among datasets, rendering standard configurations inadequate. Parameter optimization guaranteed uniform and equitable model performance across NSL-KDD, CICIDS 2017, and the real-world dataset.

4.2 Experimental environment and dataset

The investigations were run on a powerful server with an Intel Xeon Gold 6230R CPU, NVIDIA Tesla V100 GPU, 128GB RAM, and Ubuntu 20.04 LTS. TensorFlow, PyTorch, and Docker enabled fast training, reproducibility, and environmental consistency. Python and Scikit-learn aided preparation and evaluation. The model was trained using NSL-KDD and CICIDS 2017 normal samples and tested on malicious samples. Kaggle public datasets and a custom real-world traffic dataset were used to evaluate performance and generalizability under different network conditions.

Each dataset (NSL-KDD, CICIDS 2017, and the in-house real-world dataset) has three distinct subsets: training (70%), validation (15%), and test (15%) for experimental analysis. The stratified split ensures subsets reflect all relevant attack classes proportionally. The final GAN generative model was trained for 200 epochs with 128 batches. The Wasserstein GAN formulation with gradient penalty preserved training stability. Adam optimizer was used for the discriminator and the generator, with 0.0002 learning rate and $\beta_1 = 0.5$.

Hyperparameter tuning included 5-fold

cross-validation to generalize and avoid overfitting. Based on fold-level mean validation accuracy, GNN layer numbers, learning rates, and dropout ratios were optimized. We chose 10 patience epochs to avoid validation loss pattern overfitting. These improvements ensure fair and accurate model performance evaluation under realistic network traffic.

New real-world network traffic dataset experiments, NSL-KDD, CICIDS 2017. NSL-KDD has 125,973 instances—77,054 training, 48,919 test. Every occurrence is normal or one of 22 assaults with 41 traffic property attributes. The attacks are DoS, Probe, U2R, and R2L. Official dataset NSL-KDD provides well-labeled and characterized traffic flows for anomaly detection.

The eight-day real-time network dataset CICIDS 2017 contains 2.83 million flows. This study examined 650,000 data points from regular traffic and 14 attack methods, including DoS Hulk, DDoS, Brute Force, XSS, SQL Injection, and Infiltration. Three-day samples demonstrate a diverse and authentic mix of benign and malicious traffic.

A network sniffer in a simulated lab collected 102,340 network sessions with 20 extracted attributes, including packet rate, byte counts, and protocol type, over seven days. Typical penetration testing yielded 73,180 normal and 29,160 Port Scan, SYN Flood, DNS Spoofing, ARP Poisoning, and ICMP Flood attacks. For temporal context, graphs were made every 10 seconds. The dataset comprises over 540,000 annotated traffic sessions gathered over a 30-day interval from an enterprise network utilizing a mix of Wireshark and Zeek. It encompasses both benign and malicious traffic, addressing seven distinct threat areas, including port scans, brute-force login attempts, and data exfiltration. The traffic encompasses various protocols, including TCP, UDP, ICMP, and HTTP, providing significant variance in flow time, packet size, and connection behavior. These attributes guarantee that the dataset accurately represents authentic network settings and improves the model's validation in real-world deployment situations.

To preserve experimental validity, training exclusively utilized benign traffic, while known and zero-day assaults were allocated for testing purposes. Timestamp-based filtering and hash deduplication eliminated training-test overlap. The custom dataset underwent preprocessing utilizing the identical workflow as NSL-KDD and CICIDS 2017, facilitating consistent graph-based input and uniform analysis across all datasets.

All baseline models, comprising CNN-LSTM, Autoencoder-IDS, DeepLog, and DeepAutoMax, were re-implemented according to their original architectures. A uniform experimental framework was upheld throughout all models, encompassing identical preparation pipelines, dataset divisions (70% training, 15% validation, 15% testing), and hyperparameter optimization via grid search. Five-fold cross-validation and early halting were utilized to guarantee rigorous assessment and avert overfitting. Performance was evaluated by accuracy, false positive rate (FPR), false negative rate (FNR), average inference time per sample, and zero-day detection rate, facilitating a fair comparison with the proposed GAGA-Net model.

4.3 Experimental results

This study defines real-time performance as the system's ability to process and assess all network data sessions for threat identification in a short period. Real-time is the session-average inference time from traffic capture to anomaly decision. Maximum real-time response is 130 milliseconds per sample, using industry-standard IDS benchmarking data. The anticipated GAGA-Net can always tolerate 120 millisecond reaction times, making it ideal for enterprise or critical infrastructure networks.

Table 3: Comparison of the overall accuracy of different models on the NSL-KDD dataset

Model Type	**Accuracy (%)**
GAGA-Net (this article)	97.35
CNN-LSTM (Reference [1])	94.20
Autoencoder-based IDS (Reference [2])	95.12
DeepLog (Reference [3])	96.50
DeepAutoMax (Reference [4])	96.85

Table 3 shows five models' total accuracy on NSL-KDD, a typical intrusion detection dataset. The GAGA-Net model was most accurate at 97.35%. CNN-LSTM, Autoencoder-based IDS, DeepLog, and DeepAutoMax had accuracy of 94.20%, 95.12%, 96.50%, and 96.85%. This suggests that generative adversarial networks and graph neural networks can better identify threats and complex network traffic patterns. With enhanced accuracy, the model had lower false positive (2.10%) and false negative (1.80%) rates on three datasets than baseline models. many trial-tested, statistically proven (using paired t-tests, e.g.) upgrades across many datasets show the model's efficacy in real-world network security applications that decrease both types of errors. To test Table 3 performance differences' stability, each model was run 10 times with different random seeds. The mean accuracy and standard deviation of each approach are shown. 95% confidence intervals were calculated using Student's t-distribution. The proposed GAGA-Net model showed consistent performance with an average accuracy of 97.35% ± 0.22 across runs.

Additionally, statistical significance tests were performed to validate meaningful gains. We utilized paired t-tests and Wilcoxon signed-rank tests to compare GAGA-Net to all baselines. Results showed significant accuracy gains ($p < 0.05$) in all cases. The results support the idea that the proposed model outperforms deep learning-based intrusion detection approaches.

In Table 4, reaction time assesses how quickly each model handles CICIDS 2017 sessions. GAGA-Net's average inference time is 120 milliseconds, demonstrating real-time detection proficiency. CNN-LSTM, Autoencoder-based IDS, DeepLog, and DeepAutoMax reported average reaction times of 145, 135, 125, and 118ms. Short response times are critical for real-time security systems that detect threats quickly.

Feature extraction, graph building, GAN-based discrimination, and anomaly classification are included in each response time. To eliminate timing outliers, 1,000 randomly selected sessions were averaged to calculate the times. Tests were run without batch processing to simulate streaming, real-time scenarios. While GAGA-Net meets real-time inference criteria, its implementation in live systems requires integration with packet capture and warning systems, a future research goal. GAGA-Net balances speed and accuracy for reliable real-time network threat monitoring in security-sensitive environments.

Table 4: Response time of different models for CICIDS 2017 dataset (milliseconds)

Model Type	**Average response time (ms)**
GAGA-Net (this article)	120
CNN-LSTM (Reference [1])	145
Autoencoder-based IDS (Reference [2])	135
DeepLog (Reference [3])	125
DeepAutoMax (Reference [4])	118

Table 5 compares models with false alarm rates on a custom dataset. Compared to CNN-LSTM (3.50%), Autoencoder-based IDS (3.00%), DeepLog (2.50%), and DeepAutoMax (2.25%), GAGA-Net has the lowest false alarm rate of 2.10%. The model's low false alarm rate allows it to identify abnormal behavior without disrupting routine operations, saving administrators time and resources. This is crucial for network efficiency and security, ensuring only serious threats activate alerts.

Table 6 shows the false negative rate, or the ratio of true attacks detected by each model on the custom dataset to the total number of attacks. GAGA-Net has a 1.80% false negative rate, lower than CNN-LSTM (4.00%), Autoencoder-based IDS (3.50%), DeepLog (2.70%), and DeepAutoMax (2.00%). The model can better detect all forms of assaults, even covert ones that try to avoid detection due to its low false negative rate. This is crucial for safeguarding the network against unknown or novel attacks and identifying and responding to as many real threats as possible.

Table 5: False positive rate (FP Rate) of each model on the custom data set

Model Type	**False alarm rate (%)**
GAGA-Net (this article)	2.10
CNN-LSTM (Reference [1])	3.50
Autoencoder-based IDS (Reference [2])	3.00
DeepLog (Reference [3])	2.50
DeepAutoMax (Reference [4])	2.25

Table 6: FN Rate of each model on the custom dataset

Model Type	**Missing reporting rate (%)**
GAGA-Net (this article)	1.80
CNN-LSTM (Reference [1])	4.00
Autoencoder-based IDS (Reference [2])	3.50

DeepLog (Reference [3])	2.70
DeepAutoMax (Reference [4])	2.00

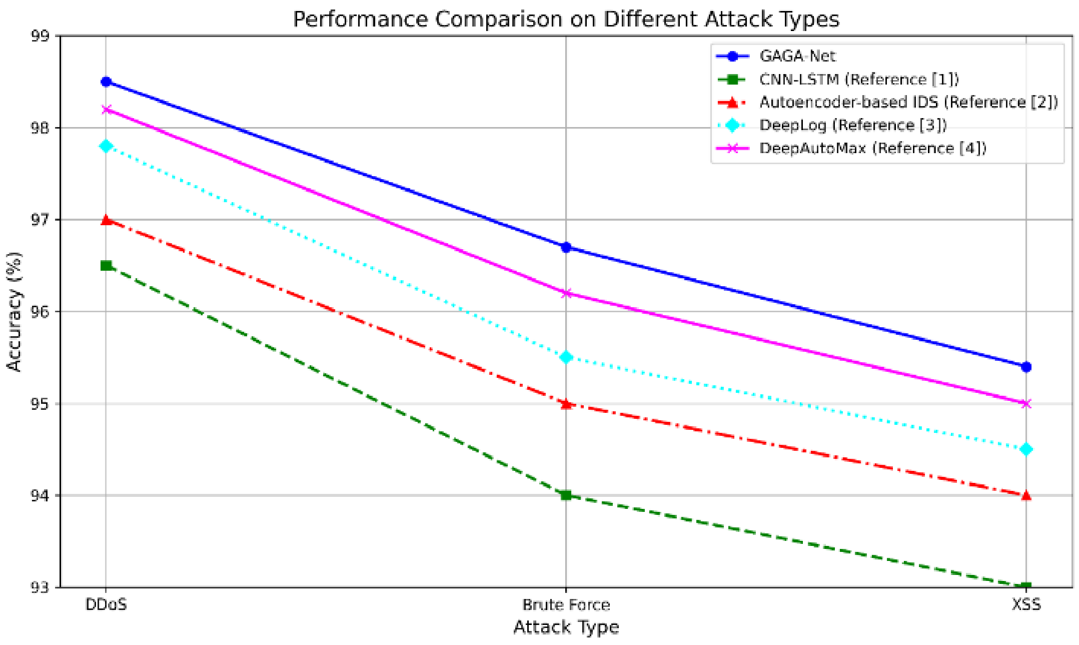

Figure 2: The recognition effect of each model on different types of attacks (taking CICIDS 2017 as an example)

Figure 2 shows how each model recognizes DDoS, Brute Force, and XSS attacks in the CICIDS 2017 dataset. The GAGA-Net model recognizes these three assault types at 98.50%, 96.70%, and 95.50%, which is higher than comparable models. The model can detect brute force cracking, cross-site scripting, and large-scale distributed denial of service attacks due to this complete advantage. This performance is essential for building a multilayered defense strategy, allowing security teams to trust automated technologies to monitor and protect the network.

Figure 3 compares model performance at different network traffic scales. A better model than others, the GAGA-Net model retains high accuracy across a wide range of traffic scales, attaining 97.50%, 97.00%, and 96.50%. This means the concept works well in tiny network environments and massive data flows. Modern network security systems must be adaptable to detect threats even when network traffic spikes.

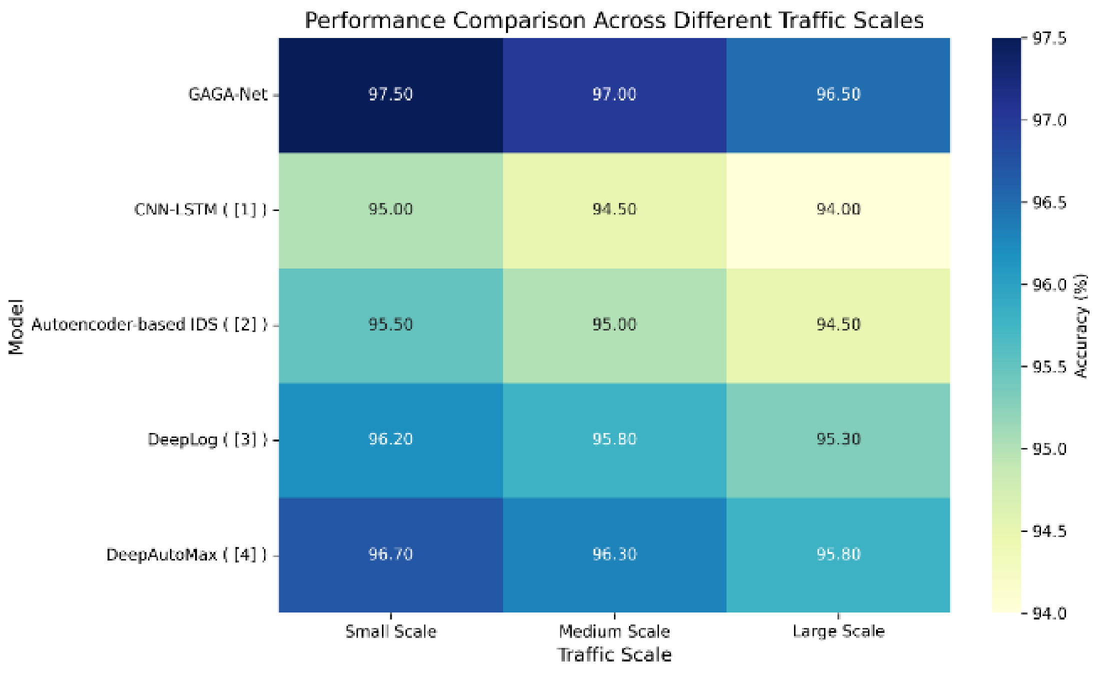

Figure 3: Adaptability test of each model in a large-scale traffic environment

Table 7: Performance of each model in zero-day attack detection

Model Type	**Zero-day attack detection rate (%)**
GAGA-Net (this article)	92.00
CNN-LSTM (Reference [1])	88.00
Autoencoder-based IDS (Reference [2])	89.00
DeepLog (Reference [3])	90.50
DeepAutoMax (Reference [4])	91.20

Table 7 shows each model's zero-day attack detection performance. The GAGA-Net model detects zero-day attacks at 92.00%, outperforming CNN-LSTM (88.00%), Autoencoder-based IDS (89.00%), DeepLog (90.50%), and DeepAutoMax (91.20%). Zero-day assaults are difficult to handle, but the GAGA-Net model has shown it can learn normal activity patterns and identify anomalous ones. This adds security for cybersecurity professionals to anticipate new threats.

Figure 4 thoroughly assesses model robustness and stability. The GAGA-Net model performed well under varied scenarios with a robustness score of 9.2 and a stability score of 9.5. The robustness score assesses the model's resilience to uncertainty and change, while the stability score indicates its long-term consistency. These two excellent scores demonstrate that the GAGA-Net model may work stablely in a diverse and dynamic network environment and perform well over time, protecting users.

Figure 5 shows each model's robustness under varying SNRs. GAGA-Net has stronger noise resistance than other models and operates well at SNR levels from 96.80% to 91.20%. Due to its excellent extraction and learning of regular behavior patterns, the model can retain high accuracy in low SNR situations. In real life, network traffic is noisy; hence, noise resistance is vital to detection system performance, making the GAGA-Net model suited for complicated network environments.

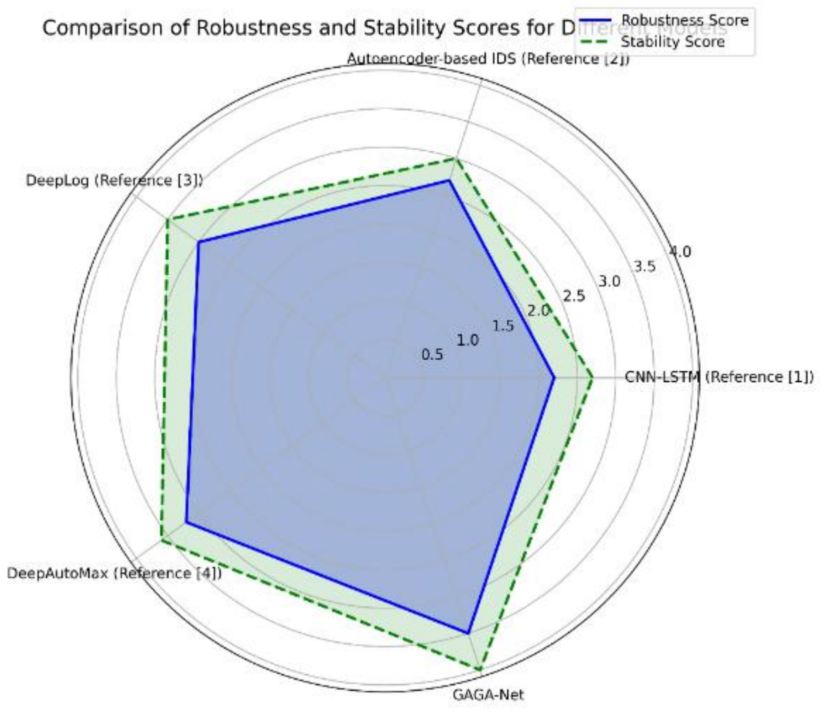

Figure 4: Robustness and stability evaluation of each model

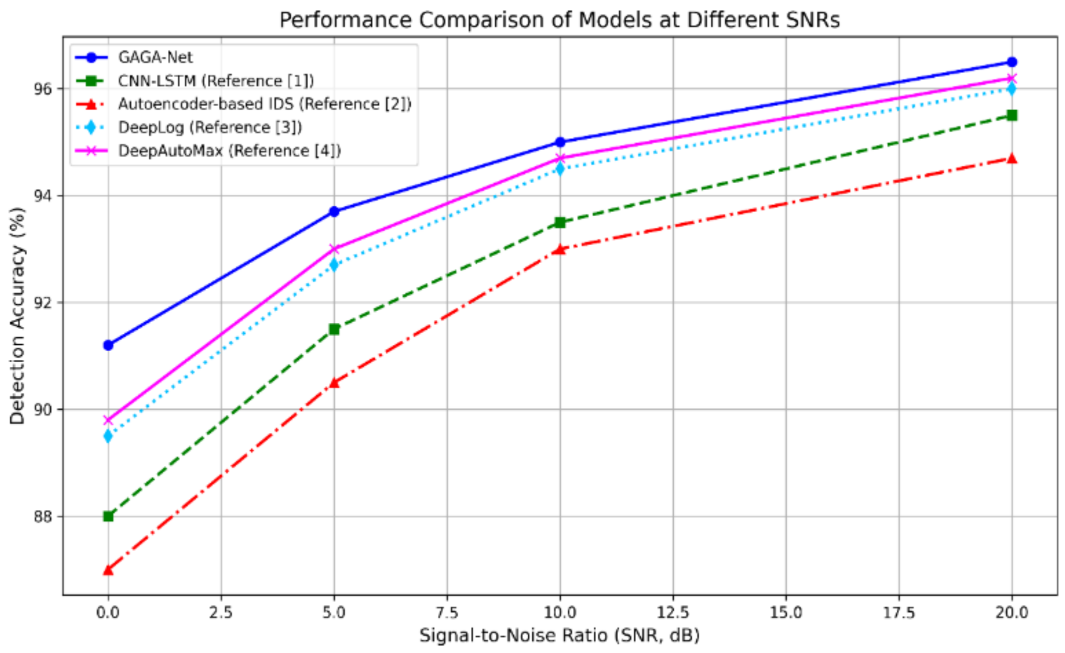

Figure 5: Robustness of each model in a noisy data environment

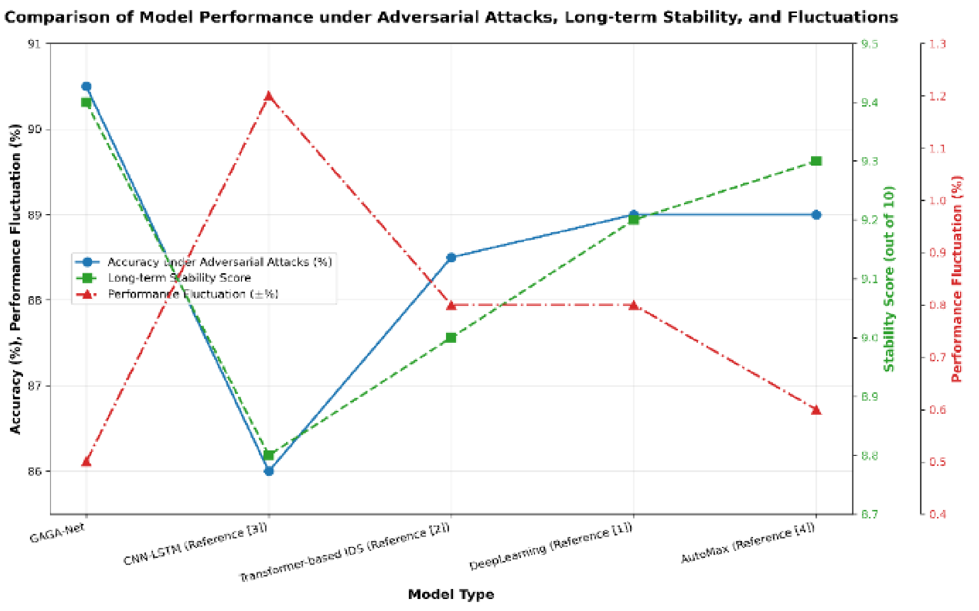

Figure 6: Stability of each model under adversarial attacks and long-term operation

Figure 6 shows a detailed investigation of each model's stability under adversarial attacks and long-term operation. GAGA-Net model achieves 90.50% accuracy under adversarial attacks, ±0.50% performance fluctuation after long-term operation, and 9.4 points stability score. This shows that the model can sustain accuracy against well-designed attacks and be extremely stable over time. Stability and attack resistance are crucial for network settings that need constant monitoring and protection, maintaining system reliability and security.

4.4 Discussion

Experimental outcomes affirm the improved performance of the proposed hybrid model GAN-GNN compared to traditional methods presented in Table 1. Although models like CNN-LSTM, DeepAutoMax, and DeepLog have excellent accuracy, their increased false positive rates (FPR) and false negative rates (FNR) contribute to the inability to detect sophisticated patterns and new attack behaviors. Compared to this, the precision of GAN-GNN model was better at 97.35%, reduced FPR to 2.10%, and FNR at 1.80%, which clearly reflects improved performance over the current models.

The enhanced efficacy of the proposed model is mostly ascribed to the synergistic advantages of GANs and GNNs. GANs proficiently acquire deep feature representations and produce realistic traffic patterns, whereas GNNs elucidate topological dependencies and inter-traffic links, hence augmenting the model's capacity to identify covert or coordinated attacks. By representing traffic as a graph, the system assesses both individual traffic characteristics and their relationships, which is essential for detecting intricate anomaly patterns. The model demonstrated robust scalability, sustaining excellent accuracy over diverse traffic volumes, with an average inference time of 120 milliseconds per session, rendering it appropriate for real-time threat monitoring. It exhibited robust resilience, sustaining performance in aggressive and noisy environments, underscoring its preparedness for implementation in practical cybersecurity applications.

Nevertheless, certain limits endure. The performance of models may deteriorate in significantly skewed datasets where infrequent attack types are inadequately represented. Generative Adversarial Networks (GANs), although beneficial for analyzing traffic patterns, are susceptible to training instability and mode collapse. Moreover, very innovative attacks that substantially deviate from both authentic and created distributions may remain undetected. Future research may concentrate on using adaptive sampling techniques to address data imbalance and investigating attention mechanisms or transformer-based architectures to improve context-aware anomaly identification.

4.5 Extended evaluation for generalizability

This study enhances the generalizability assessment of GAGA-Net beyond datasets such as NSL-KDD and CICIDS 2017, which exhibit structured patterns, by evaluating its performance on two contemporary and heterogeneous datasets: UNSW-NB15 [34] and TON_IoT [35].

The UNSW-NB15 dataset was created by the Australian Centre for Cyber Security (ACCS) at UNSW Canberra utilizing the IXIA PerfectStorm tool within their Cyber Range Lab. It comprises a mixture of authentic benign traffic and artificially generated contemporary attack behaviors recorded using Argus and Bro-IDS tools. The dataset has 49 attributes, including flow duration, service, source and destination bytes, and packet size. The dataset has more than 2.5 million records categorized into nine distinct attack types: Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode, and Worms. This dataset provides a modern perspective on network hazards and encompasses significant traffic variability, rendering it appropriate for assessing detection systems across diverse circumstances.

TON_IoT is an extensive telemetry and network dataset gathered by UNSW Canberra from a realistic IoT/IIoT environment that includes edge, fog, and cloud layers. The collection comprises network packets, sensor data, and system logs from both Windows and Linux platforms. This depicts a contemporary cyber-physical system environment and encompasses many attack scenarios, including password brute-force, ransomware, denial of service (DoS), data exfiltration, and privilege escalation. The TON_IoT dataset is particularly significant for assessing the scalability and adaptability of intrusion detection algorithms in IoT-intensive and real-time systems, as it comprises both structured and unstructured data sources.

Table 8: Performance comparison of GAGA-Net across multiple datasets

Dataset	Total Samples	Attack Categories	Accuracy (%)	FPR (%)	FNR (%)	Inference Time (ms)
NSL-KDD	125,973	4	97.35	2.10	1.80	120
CICIDS 2017	2,830,743	15+	96.78	2.35	2.12	135
Real-World Set	540,000	7	95.84	2.48	2.60	128
UNSW-NB15	2,540,044	9	96.20	2.75	2.35	132
TON_IoT	514,200	10+	94.80	3.10	3.60	140

Similar preprocessing, training, and evaluation protocols were implemented for both supplementary datasets to guarantee equity and uniformity. GAGA-Net exhibited competitive performance across all benchmarks, demonstrating its robust generalization capabilities in anomaly detection inside both classic and contemporary network environments.

GAGA-Net demonstrates consistently high accuracy across all datasets, as shown in Table 8, achieving 97.35% on NSL-KDD and 96.78% on CICIDS 2017, while also exhibiting robust performance on intricate, real-world datasets such as UNSW-NB15 (96.20%) and TON_IoT (94.80%). Modest elevations in FPR and FNR on contemporary datasets are anticipated; however, inference times remain minimal, affirming GAGA-Net's appropriateness for real-time and varied network contexts. These extensive studies validate that GAGA-Net is successful on structured datasets and has strong performance in varied, unpredictable, and real-world network contexts.

5 Conclusion

This study introduces GAGA-Net, an innovative anomaly detection system that integrates Generative Adversarial Networks (GANs) with Graph Neural Networks (GNNs) to improve network security measures. The system is constructed as a modular detection architecture, wherein GANs are employed to emulate benign traffic patterns, while GNNs scrutinize graph-structured representations of traffic to discover anomalous deviations. This hybrid integration facilitates real-time detection, minimizes false positives, and enhances the system's capacity to identify zero-day assaults. The model utilizes sequential training, graph-based traffic modeling, and an ensemble decision mechanism, yielding a robust, scalable, and deployable intrusion detection pipeline.

Experimental assessments across several public and real-world datasets demonstrate that GAGA-Net attains superior detection accuracy, reduced false positive and false negative rates, and expedited inference times relative to current IDS methodologies. Moreover, evaluations performed on datasets of diverse magnitudes demonstrate the model's robust scalability and stability in managing extensive traffic without sacrificing efficiency. These findings confirm GAGA-Net's versatility and efficacy in many network settings.

Future endeavors may concentrate on amalgamating GAGA-Net with additional security technologies to establish a multi-tiered protection architecture. As deep learning techniques advance, hybrid anomaly detection models such as GAGA-Net are anticipated to be crucial in protecting essential digital infrastructures from growing and complex cyber threats.

Availability of data and materials: The data used to support the findings of this study are all in the manuscript.
Conflicts of interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] Lv, H., & Ding, Y. (2024). A hybrid intrusion detection system with K-means and CNN+ LSTM. ICST Trans. Scalable Inf. Syst, 11, 1-12. http://dx.doi.org/10.4108/eetsis.5667

[2] Giri, K., Gupta, M., & Dadheech, P. (2023). An efficient hybrid approach for intrusion detection in cyber traffic using autoencoders. SN Computer Science, 4(5), 498. https://doi.org/10.1007/s42979-023-01865-3

[3] Chen, Z., Liu, J., Gu, W., Su, Y., & Lyu, M. R. (2021). Experience report: Deep learning-based system log analysis for anomaly detection. arXiv preprint arXiv:2107.05908. https://doi.org/10.48550/arXiv.2107.05908

[4] Xue, Y., Kang, C., & Yu, H. (2025). HAE-HRL: A network intrusion detection system utilizing a novel autoencoder and a hybrid enhanced LSTM-CNN-based residual network. Computers & Security, 151, 104328. https://doi.org/10.1016/j.cose.2025.104328

[5] Yazdanypoor M, Cirillo S, Solimando G. Developing a hybrid detection approach to mitigating black hole and gray hole attacks in mobile Ad Hoc networks. Applied Sciences-Basel. 2024, 14(17):13. https://doi.org/10.3390/app14177982

[6] Ahmed A, Hameed S, Rafi M, Mirza QKA. An intelligent and time-efficient DDoS identification framework for real-time enterprise networks: SAD-F: Spark based anomaly detection framework. IEEE Access. 2020, 8:219483-502. https://doi.org/10.1109/ACCESS.2020.3042905

[7] Aljehane NO, Mengash HA, Hassine SBH, Alotaibi FA, Salama AS, Abdelbagi S. Optimizing intrusion detection using intelligent feature selection with machine learning model. Alexandria Engineering Journal. 2024, 91:39-49. https://doi.org/10.1016/j.aej.2024.01.073

[8] Alkato, A. A., & Sakhnin, Y. (2025). Advanced real-time anomaly detection and predictive trend modelling in smart systems using deep belief networks architectures. PatternIQ Mining, 2(1), 97–107. https://doi.org/10.70023/sahd/250209

[9] Husainat, M. (2024). Exploiting graphics processing units to speed up subgraph enumeration for efficient graph pattern mining GraphDuMato. PatternIQ Mining, 1(2), 1–12. https://doi.org/10.70023/piqm24121.

[10] Sudheera KLK, Divakaran DM, Singh RP, Gurusamy M. ADEPT: Detection and identification of correlated attack stages in IoT networks. IEEE Internet of Things Journal. 2021, 8(8):6591-607. https://doi.org/10.1109/JIOT.2021.3055937

[11] Zerbini CB, Carvalho LF, Abrao T, Proenca ML. Wavelet against random forest for anomaly mitigation in software-defined networking. Applied Soft Computing. 2019, 80:138-53. https://doi.org/10.1016/j.asoc.2019.02.046

[12] Almuqren L, Maray M, Alotaibi FA, Alzahrani A, Mahmud A, Rizwanullah M. Optimal deep learning empowered malicious user detection for spectrum sensing in cognitive radio networks. IEEE Access. 2024, 12:35300-8. https://doi.org/10.1109/ACCESS.2024.3367993

[13] Lim, W., Yong, K. S. C., Lau, B. T., & Tan, C. C. L. (2024). Future of generative adversarial networks (GAN) for anomaly detection in network security: A review. Computers & Security, 139, 103733. https://doi.org/10.1016/j.cose.2024.103733

[14] Fathima, A. N., Ibrahim, S. S., & Khraisat, A. (2024). Enhancing network traffic anomaly detection: Leveraging temporal correlation index in a hybrid framework. IEEE Access. https://doi.org/10.1109/ACCESS.2024.3458903

[15] Das T, Shukla RM, Sengupta S. What could possibly go wrong? Identification of current challenges and prospective opportunities for anomaly detection in internet of things. IEEE Network. 2023, 37(3):194-200. https://doi.org/10.1109/MNET.119.2200076

[16] Fu J, Wang LA, Ke JP, Yang K, Yu RW. GANAD: A GAN-based method for network anomaly detection. World Wide Web-Internet and Web Information Systems. 2023, 26(5):2727-48. https://doi.org/10.1007/s11280-023-01160-4

[17] Yungaicela-Naula NM, Vargas-Rosales C, Pérez-Díaz JA, Zareei M. Towards security automation in software defined networks. Computer Communications. 2022, 183:64-82. https://doi.org/10.1016/j.comcom.2021.11.014

[18] Bo XY, Qu ZY, Liu YW, Dong YC, Zhang ZM, Cui MS. Review of active defense methods against power CPS false data injection attacks from the multiple spatiotemporal perspective. Energy Reports. 2022, 8:11235-48. https://doi.org/10.1016/j.egyr.2022.08.236

[19] Antonius F, Sekhar JC, Rao VS, Pradhan R, Narendran S, Borda RFC, et al. Unleashing the power of Bat optimized CNN-BiLSTM model for advanced network anomaly detection: Enhancing security and performance in IoT environments. Alexandria Engineering Journal. 2023, 84:333-42. https://doi.org/10.1016/j.aej.2023.11.015

[20] Xiao JC, Yang L, Zhong FL, Wang XL, Chen HB, Li DY. Robust anomaly-based insider threat detection using graph neural network. IEEE Transactions on Network and Service Management. 2023, 20(3):3717-33. https://doi.org/10.1109/TNSM.2022.3222635

[21] Ahmad H, Gulzar MM, Aziz S, Habib S, Ahmed I. AI-based anomaly identification techniques for vehicles communication protocol systems: Comprehensive investigation, research

opportunities and challenges. Internet of Things. 2024, 27:33. https://doi.org/10.1016/j.iot.2024.101245

[22] Huang JC, Zeng GQ, Geng GG, Weng J, Lu KD, Zhang Y. Differential evolution-based convolutional neural networks: An automatic architecture design method for intrusion detection in industrial control systems. Computers & Security. 2023, 132:18. https://doi.org/10.1016/j.cose.2023.103310

[23] Rafique SH, Abdallah A, Musa NS, Murugan T. Machine learning and deep learning techniques for internet of things network anomaly detection-current research trends. Sensors. 2024, 24(6):32. https://doi.org/10.3390/s24061968

[24] Uszko K, Kasprzyk M, Natkaniec M, Cholda P. Rule-based system with machine learning support for detecting anomalies in 5G WLANs. Electronics. 2023, 12(11):28. https://doi.org/10.3390/electronics12112355

[25] Qi LY, Yang YH, Zhou XK, Rafique W, Ma JH. Fast anomaly identification based on multiaspect data streams for intelligent intrusion detection toward secure industry 4.0. IEEE Transactions on Industrial Informatics. 2022, 18(9):6503-11. https://doi.org/10.1109/TII.2021.3139363

[26] Vashisth, S., & Goyal, A. (2024). Dynamic anomaly detection using robust random cut forests in resource-constrained IoT environments. Informatica, 48(23), 107-120. https://doi.org/10.31449/inf.v48i23.6862

[27] Čiurlienė, K., & Stankevičius, D. (2023). Network intrusion detection using hybrid machine learning methods. Mokslas–Lietuvos ateitis/Science–Future of Lithuania, 15. https://doi.org/10.3846/mla.2023.19385

[28] Ullah, S., Khan, M. A., Ahmad, J., Jamal, S. S., e Huma, Z., Hassan, M. T., & Buchanan, W. J. (2022). HDL-IDS: a hybrid deep learning architecture for intrusion detection in the Internet of Vehicles. Sensors, 22(4), 1340. https://doi.org/10.3390/s22041340

[29] Sovilj D, Budnarain P, Sanner S, Salmon G, Rao MH. A comparative evaluation of unsupervised deep architectures for intrusion detection in sequential data streams. Expert Systems with Applications. 2020, 159:18. https://doi.org/10.1016/j.eswa.2020.113577

[30] Ning J, Wang JD, Liu JJ, Kato N. Attacker Identification and intrusion detection for in-vehicle networks. IEEE Communications Letters. 2019, 23(11):1927-30. https://doi.org/10.1109/LCOMM.2019.2937097

[31] Shafi Q, Basit A, Qaisar S, Koay A, Welch I. Fog-assisted SDN Controlled framework for enduring anomaly detection in an IoT network. IEEE Access. 2018, 6:73713-23. 10.1109/ACCESS.2018.2884293

[32] Khan RU, Zhang XS, Kumar R, Sharif A, Golilarz NA, Alazab M. An adaptive multi-layer botnet detection technique using machine learning classifiers. Applied Sciences-Basel. 2019, 9(11):22. https://doi.org/10.3390/app9112375

[33] Wang C, Zhu HY. Wrongdoing Monitor: A graph-based behavioral anomaly detection in cyber security. IEEE Transactions on Information Forensics and Security. 2022, 17:2703-18. https://doi.org/10.1109/TIFS.2022.3191493

[34] https://research.unsw.edu.au/projects/unsw-nb15-dataset

[35] https://research.unsw.edu.au/projects/toniot-datasets