Alexandria Engineering Journal 132 (2025) 369–382

Contents lists available at ScienceDirect

Alexandria Engineering Journal

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

Original article

A novel approach for real-time anomaly detection in dynamic computer networks using temporal graph networks and explainable artificial intelligence

Mehmet Ozdem

Turk Telekom, Turgut Ozal Bulvari, 06103 Kule, Aydinlikevler, Altindag/Ankara, Turkiye

ARTICLE INFO

Keywords:
Information security
Explainable Artificial Intelligence
Temporal Graph Networks
Anomaly detection
Real-time network traffic
Network intrusion detection

ABSTRACT

Today, cyber threats are rapidly changing in both diversity and complexity, making traditional defense methods inadequate. This research presents a novel anomaly detection model based on Temporal Graph Networks supported by Explainable Artificial Intelligence in real-time network traffic. Live network data was collected using Wireshark to develop a balanced dataset covering different attack types. The proposed model can effectively identify intrusions with high accuracy by fusing temporal and structural information. Explainable Artificial Intelligence is achieved through a gradient-based feature importance method that quantifies the contribution of numerical and textual features at the classification stage, providing transparency into the reasoning behind attack identification. The model achieved 96.8% accuracy in experiments conducted with a large test dataset and was able to process 50,000 packets with a detection latency of only 1.45 s, demonstrating its effectiveness for real-time deployment. Furthermore, the dataset generated in the study has been openly published on the HuggingFace platform and is available to researchers in similar fields. Using this methodology not only enables security analysts to provide fast and effective detection but also enables more effective threat response by providing explainability. This research represents a significant advancement towards creating autonomous, dynamic, and semantically rich solutions for future cybersecurity systems.

1. Introduction

In today's digital world, network infrastructures are becoming increasingly complex. This is paving the way for cybersecurity threats to diversify and intensify. The rapid proliferation of smart devices is creating new entry points into computer networks. This allows cyber attackers to target diverse targets and exacerbates security threats [1–3].

With increasing data traffic and the rise of multi-layered architectures, traditional security approaches are becoming inadequate, and the need for dynamic, scalable solutions is increasing. In heterogeneous network environments, various technologies such as Long Term Evolution (LTE), Network Functions Virtualization (NFV), and different Internet of Things (IoT) protocols interoperate, creating new requirements for both performance and security [4]. Protocol diversity in these network environments means using lightweight IoT-focused protocols alongside Internet Protocol (IP)-based ones [5]. This allows attackers to develop multi-layered attack vectors and necessitates defense systems to ensure security at both horizontal and vertical levels. Especially in internal threat scenarios such as lateral movement, attackers can evade detection by moving between systems. Therefore, network security should be supported not only by perimeter measures but also by internal segmentation and micro-security policies [6,7].

The tools used in network defense have now moved beyond static structures and are turning to behavioral analysis and artificial intelligence-supported learning methods [8,9]. Deep Packet Inspection (DPI) techniques enable the analysis of network packets at the content level, while Intrusion Detection System (IDS) and Intrusion Prevention System (IPS) stand out with their signature-based and anomaly-based threat detection capabilities [10,11]. Additionally, Multi-Layered Security (MLM) approaches aim to integrate security policies appropriate for each layer of the Open Systems Interconnection (OSI) model. Solutions such as VLAN segmentation and MAC filtering at the data link layer, Transport Layer Security (TLS) and Quick UDP Internet Connections (QUIC) protocols at the transport layer, and Web Application Firewall (WAF) and API security gateways at the application layer should be implemented [12]. Thanks to this multi-layered structure, both external and internal threats can be proactively controlled.

Among advanced threat detection systems, Real-Time Network Monitoring (RTNM) solutions detect unusual behavior in network traffic,

E-mail address: mehmet.ozdem@turktelekom.com.tr.

https://doi.org/10.1016/j.aej.2025.11.001
Received 6 August 2025; Received in revised form 3 October 2025; Accepted 2 November 2025
Available online 7 November 2025

enabling early detection of zero-day attacks. Cyber Threat Intelligence (CTI) systems provide information about threat actors, keeping defense systems constantly up to date [13,14]. The integration of such systems enables a 'preventive defense' approach against cyber threats. Distributed Denial of Service (DDoS) attacks, in particular, cause serious business continuity issues by disrupting the network's serviceability [15]. Attackers typically carry out reflective and amplification-based attacks through botnets; protocols such as Domain Name System (DNS), Network Time Protocol (NTP), and Simple Service Discovery Protocol (SSDP) can be misused for this purpose [16,17]. Against such threats, system resilience can be enhanced using traffic routing algorithms, rate-limiting policies, CDN-based load balancing, and honeypot architectures. Network security today requires a comprehensive architecture that includes multi-layered, adaptive, and AI-supported systems that can not only prevent attacks but also intervene during and after attacks [18–20].

1.1. Problem statement

The increasing sophistication and adaptability of cyber threats seriously limit the effectiveness of traditional attack detection mechanisms. For instance, traditional signature-based IDS systems often fail to detect multi-stage attacks such as lateral movement within enterprise networks, low-frequency data exfiltration, or stealthy DDoS attacks using slow-rate techniques. In such scenarios, Temporal Graph Networks (TGNs) excel by modeling time-dependent relationships between nodes, capturing evolving attack patterns, and detecting anomalies that static models would miss. Combined with Explainable AI, the system can also provide interpretable insights, such as which devices or connections are most affected, enabling faster and more informed incident response. Static signature-based systems and classic rule-based approaches often fail to identify emerging threat vectors, leading to a high rate of false negatives in defense mechanisms [21,22]. The integration of artificial intelligence and machine learning-based solutions into the field of cybersecurity offers hope for overcoming these issues; however, these systems also have some fundamental limitations [23–25]. Although today's AI-powered solutions are equipped with advanced techniques such as behavioral analysis, anomaly detection, threat hunting, and pattern recognition, the constantly changing attack surfaces and dynamic tactics of attackers (e.g., adversarial attacks, zero-day exploits) reduce the effectiveness of these systems [26]. Traditional artificial intelligence models are typically trained on static data, resulting in high response times to real-time attacks and causing latency and low sensitivity in defense systems. Furthermore, datasets used in cybersecurity are mostly imbalanced. In particular, attack types that occur rarely but have critical effects (e.g., APT, DDoS, lateral movement) are not sufficiently represented in training data [27,28]. This reduces the generalization ability of models and increases false positive/negative rates. In addition, feature engineering deficiencies and bottlenecks encountered during the real-time processing of high-volume network traffic weaken the reliability of systems [29]. Another important issue is the insufficient processing of live network data and the lack of visualization-supported attack analyses on this data. Visual methods, such as graph-based threat modeling methods and topological analysis techniques, are critical for understanding attack paths and detecting anomaly clusters [30]. Since most attacks exhibit previously unknown behavior patterns, AI models must be continuously updated and retrained. Therefore, adaptive learning approaches such as transfer learning, online learning, and continual learning are gaining prominence. However, fully integrating these systems into the cybersecurity landscape still requires research. Furthermore, the limited number of studies focusing on real-time attack detection in the literature is another factor limiting developments in this field [31]. Therefore, the need for more robust, low-latency, and continuously improving detection systems has become more critical than ever.

1.2. Main contributions

In this study, live network data were collected using Wireshark to more accurately and time-sensitively detect threats in network traffic, and a rich dataset containing attack patterns was built based on this data. Today, attacks possess not only structural characteristics but also temporal context; therefore, the proposed system is built on the Temporal Graph Network (TGN) architecture, which models both temporal and contextual relationships. TGNs have been shown to perform well on time-sensitive relational data in other domains; their application to real-time network anomaly detection remains limited. Most existing methods are based on static datasets or offline analysis and often lack interpretability. In this study, TGNs are applied directly to live, multi-protocol network traffic, and explainable AI techniques are incorporated to enable accurate and interpretable real-time anomaly detection. TGN is a powerful model capable of capturing complex threat patterns by tracking the relationships between nodes (devices, IP addresses, etc.) in network traffic across time. Additionally, the proposed model integrates Explainable Artificial Intelligence (XAI) components. This allows the user to transparently explain the causes of detected anomalies, strengthening security analysts' decision-making processes. The model not only detects attacks but also explains which network components are at risk, how the attack evolves, and potential propagation paths. The proposed approach directly analyzes live traffic data obtained from packet capture tools such as Wireshark and incorporates a continuously updated learning mechanism for real-time threat identification. The ability to capture attack patterns in temporal data streams enables the system to support high-frequency anomaly detection. Furthermore, the graphical representation used allows for analyzing the evolution of attacks by dynamically reflecting changes in the network topology.

The main contributions of this article can be summarized as follows:

- A rich, diverse, and balanced dataset generated from real-time data collected from live network environments is presented.
- Instead of traditional fixed AI models, a Temporal Graph Network-based system capable of processing temporal and contextual relationships simultaneously has been developed.
- Explainable AI techniques have been integrated to ensure that attack analyses are transparent and understandable.
- The model has the capacity for continuous learning and can reconfigure and update itself against changing threat environments.
- An intuitive visualization and decision support infrastructure is provided for security analysts to more effectively analyze dynamic network structures and attack paths.

This work offers an adaptable, real-time cybersecurity solution that goes beyond attack detection and can also explain the causes of detected anomalies. In this respect, it goes beyond traditional systems, both enhancing operational security and generating high-resolution insights that will contribute to forensic analysis processes.

The remainder of the article is summarized as follows: Section 2 provides a comprehensive review of relevant literature and evaluates the strengths and weaknesses of existing systems. Section 3 covers data preprocessing processes, graph-based modeling approaches, and specialized methods used for cyberattack detection. Section 4 measures the performance of the developed model using various metrics and discusses it in detail through experimental analysis. Finally, Section 5 summarizes the findings, discusses the model's contributions, and offers suggestions for future work (see Table 1).

2. Literature review and comparison

Many approaches have been developed for network-based anomaly detection, and these methods offer different advantages and limitations, particularly in terms of real-time, accuracy, and generalizability.

Table 1
Summary of related works.

Ref.	Tools	Purpose	Method	Analysis and findings
[32]	Intel Berkeley Research Lab (IBRL) WSN Dataset	Detection of anomalies in multimodal wireless sensor networks (WSNs).	Graph Attention Network, Gated Recurrent Unit	Limitations in processing speed for large-scale networks are discussed. The proposed model achieves an F1 score of 0.90, outperforming some GCN-based methods.
[33]	NF-UNSW-NB15-v2, NF-CSE-CIC-IDS2018-v2	Design of a self-supervised GNN-based Network Intrusion Detection System using edge and topological features.	E-GraphSAGE (an edge-feature-based GNN), modified Deep Graph Infomax (DGI), and traditional anomaly detectors (IF, PCA, CBLOF, HBOS).	Anomal-E outperforms baseline methods (GraphSAGE, DGI) and raw feature-based models. Achieves up to 92.35% macro F1-score and 98.77% detection rate under 4% contamination. Demonstrates strong generalization across datasets with no reliance on labeled data.
[34]	DDoS, Tor-nonTor	Design of a GCN-based anomaly detection system using Activity and Event Network model for detecting both volumetric and long-term stealth threats.	Supervised Graph Convolutional Network (GCN) with ReLU activation, log-softmax output; graph snapshots, clustering, and Graph Edit Distance-based analysis.	Achieves 76% accuracy on DDoS and 88% on Tor-nonTor datasets. Combines temporal AEN graph analysis with GCNs, demonstrating effectiveness in detecting real-world network anomalies missed by traditional tools.
[35]	Yelp, Amazon (multi-relational datasets)	Simultaneous resolution of complexity and dynamic nature in attributed heterogeneous networks for anomaly detection.	T2EG (Temporal Enhanced Evolving-graph Generation): Hybrid filter/wrapper feature selection, subgraph embedding, intra-/inter-snapshot link prediction with Gaussian-based KL divergence.	Outperforms state-of-the-art GNN models (GCN, GraphSAGE, GEM, Play2vec) in Amazon datasets, with higher stability in training loss trends. Shows slightly lower accuracy in Yelp due to sparse interactions, but excels in dense, heterogeneous network scenarios.
[36]	MNIST, CIFAR-10, CICIDS2017, KDD Cup 1999, MVTec AD, NSL-KDD	Generic anomaly detection in various domains using CNNs for image, time-series, and network intrusion data.	Convolutional Neural Network (CNN) architectures, including autoencoder-based, one-class CNN, and GANs; include encoder–decoder structure, feature extraction, and reconstruction-based anomaly scoring.	Achieves high accuracy across diverse datasets: 98.95% on MNIST, 99.2% on KDD, 97.5% on CICIDS2017. Demonstrates strong generalization and AUC-ROC scores up to 0.999. Applicable in cybersecurity, industrial inspection, healthcare, and time-series anomaly detection.
[37]	HDFS, BGL	Real-time log anomaly detection on resource-constrained edge devices in IoT environments.	LightLog: A lightweight Temporal Convolutional Network (TCN) model using semantic embedding (Word2Vec + PCA-PPA), pointwise convolutions, and global average pooling for low-complexity deployment.	Achieves F1-scores of 97.0% (HDFS) and 97.2% (BGL) with significantly lower FLOPs, parameter count (544), and model size (139 KB). Outperforms DeepLog, LogAnomaly, and RobustLog in efficiency. Enables real-time, local detection on edge devices while maintaining high accuracy.
[38]	DynAD Framework, Dynamic Network Datasets	Detection of anomalous edges in dynamic networks.	Temporal GCN, Gated Recurrent Unit, Attention Mechanism	The framework has demonstrated challenges in processing large-scale networks due to limited scalability in capturing temporal data. However, DynAD has performed very well in the area of anomaly detection, exhibiting superior AUC values on test datasets.
[39]	Provenance database, Linux Auditd, Windows ETW	Develop an intelligent cyberattack detection system utilizing graph learning.	A heterogeneous graph modeled system assets and events, with GNNs for cyberattack detection.	The proposed graph learning method is compared with traditional rule-based and machine learning models on two real-world datasets. The proposed model is effective in recognizing patterns as advanced rule-based systems.
This paper	Wireshark	Development of an advanced hybrid AI framework utilizing Temporal Graph Networks combined with Explainable AI techniques for real-time anomaly detection in network traffic.	The system leverages TGN to model dynamic graph-structured network data and incorporates XAI methods to provide interpretability of detection results. The model is trained on comprehensive datasets collected via Wireshark and supports live traffic monitoring.	Real-time anomalies are identified by analyzing live network flows with an interpretable hybrid model that captures temporal dependencies and graph relations. The approach achieves 96% accuracy and offers cybersecurity analysts actionable insights through explainable detection outputs.

2.1. Anomaly detection in networks

Network anomaly detection plays a crucial role in maintaining the security and reliability of modern communication infrastructures. Detecting deviations from normal network behavior allows for early identification of potential attacks, failures, or misconfigurations, thereby mitigating risks and ensuring operational continuity.

ReTiNA, developed to overcome the limitations of traditional network anomaly detection methods, provides network-wide situational awareness by detecting anomalies in the correlation processes on individual links and combining these anomalies [40]. This method works in real-time by adaptive correlation estimation and statistical hypothesis testing, without the need for predefined normal behavior. While most methods in the literature suffer from shortcomings such as not being able to work in real time, static modeling, and high specificity that cannot be generalized, ReTiNA overcomes these problems and stands out with its high accuracy and low false positive rates. Magán-Carrión et al. introduced a multivariate statistical monitoring tool called MSNM-Sensor for real-time monitoring and anomaly detection in complex networks and systems [41]. This Principal Component Analysis (PCA)-based approach performs statistical modeling using control limits such as Q-statistics and D-statistics to detect anomalous behavior in network traffic and can integrate both local and hierarchical data sources. However, the fact that the diagnostic module is still under development limits the system's effectiveness in identifying potential attack sources. Fernández Maimó and colleagues propose a MEC-based, deep learning-enabled architecture [42] to provide real-time and autonomous anomaly detection in 5G networks. This system uses deep learning models to analyze network traffic flows and can dynamically allocate resources using management policies. A shortcoming of this study is that the models used in this study have only been tested in a simulation environment. Training the system on real 5G infrastructures and conducting holistic accuracy analyses are left for future work.

Chunhe Ni et al. propose an optimization framework based on edge computing, supported by lightweight machine learning models, to provide an energy-efficient solution for real-time anomaly detection in IoT networks [43]. The proposed system incorporates computation task allocation, dynamic resource allocation, and mechanisms to manage the energy-performance trade-off to maintain high accuracy while reducing energy consumption. However, the study is conducted with specific hardware configurations and limited test scenarios, and large-scale field validations in different IoT environments are left for future research. Senthilraja et al. propose a machine learning-enabled framework to perform anomaly detection via dynamic behavior profiling in software-defined IoT networks [44]. This system detects anomalies by monitoring device behavior in real time and, thanks to its SDN (Software-Defined Networking) architecture, automatically adapts network policies to prevent the spread of threats. A drawback of the proposed framework is that the developed system has only been tested in a simulation environment, and its applicability in real-time networks and long-term security performance is left for future work.

2.2. Temporal and dynamic graph-based approaches

Networks in real-world systems often evolve, with nodes and edges appearing, disappearing, or changing their interactions. Capturing these temporal dynamics is essential for understanding complex patterns and predicting future behavior.

Chen et al. introduce the GAN-powered GraphAttacker framework for generating and testing attacks on graph-based analysis tasks [45]. This framework evaluates how analysis outputs can be affected by developing flexible attack strategies across various tasks. However, the lack of direct application and field testing on real-world systems leaves room for future work to assess the practical validity of the method. Another study aims to develop an intelligent monitoring system for cyberattack detection by combining heterogeneous graph structures and graph neural networks [39]. This approach, supported by Linux Auditd and Windows ETW data, models system entities and events graphically and detects anomalies using GNNs. Although the proposed method demonstrates superior results compared to classical rule-based systems, testing it in different environments and analyzing its long-term security performance should be considered in future studies. In response to the increasing demand for anomaly detection in dynamic graph networks, Mir and colleagues propose an adaptive model called V-GCN [46]. While the study demonstrated high accuracy, F1 score, and low reconstruction error on NSL-KDD and custom-built DynKDD datasets, the model's applicability to large-scale graphs and its generalizability to different domains are left for future research.

2.3. Explainable AI in security

Explainable AI techniques enhance the interpretability of complex models, enabling stakeholders to understand model decisions. This is particularly important in security-critical contexts where trust and accountability are essential.

Gummadi et al. (2024) propose XAI-IoT, an explainable AI framework designed to enhance anomaly detection in IoT systems [47]. The framework combines two main components: (i) anomaly detection using both single and ensemble AI models, and (ii) feature importance analysis employing seven different XAI methods. The proposed framework is rigorously validated on two real-world datasets-one comprising IoT-based manufacturing sensor measurements and the other encompassing N-BaIoT botnet traffic records-thereby evidencing its capacity not only to achieve high anomaly detection performance but also to systematically elucidate the most influential features underlying model decisions. In doing so, the framework advances the transparency and interpretability of AI-driven security mechanisms within IoT environments.

Building on these contributions, the literature as a whole illustrates the growing potential of advanced graph-based and XAI-enabled approaches for anomaly detection in dynamic and heterogeneous networks. Nevertheless, detailed evaluation of such methods across diverse operational environments and large-scale, real-time settings remains a critical requirement for future research. In this study, this gap is aimed to be addressed by synthesizing and advancing the leading methodologies reported in the literature, namely: graph-based modeling of dynamic networks, temporal representation learning through Temporal Graph Networks, and explainable AI techniques. These approaches are integrated within a unified framework specifically designed to enhance both detection performance and interpretability in security-critical network environments.

3. Methodology and implementation

This section comprehensively covers the approach developed for anomaly detection in real-time network traffic. In the first phase, the key challenges encountered during the processing of raw network data are discussed, and the solution techniques implemented to address these challenges are explained. Various transformations and cleansing operations were applied to transform the heterogeneous and noisy network traffic data into a more analyzable form. These preprocessing steps provide a critical foundation for improving the model's accuracy and overall performance. Subsequently, network attacks are detected using an AI-based model [48]. During the training and evaluation of this model, network monitoring tools such as Wireshark were utilized to collect real-time data and support analysis.

Numerous attack scenarios were implemented on various test networks to create a comprehensive dataset encompassing different types of attacks. Attacks are generally categorized into six main groups:

- **Exploit**: Attacks targeting system vulnerabilities (VulnBot, CVE-2020, SQL Injection, Evil Twin).
- **Probe**: Network reconnaissance attacks (MITM Attack, Port Scan).
- **Flood**: Attacks aimed at consuming bandwidth (Slowloris, User Datagram Protocol (UDP) Flood, SYN Flood, etc.).
- **Malware**: Attacks using malware (Cerber, Agent Tesla, Lokibot, etc.).
- **Brute Force**: Attacks targeting authorization systems (Password Cracking, Redis Brute Force, etc.).
- **Benign**: Normal network traffic without any attacks (temporal and spatial benign samples)

Approximately 5000 packets from each attack type were collected and analyzed in detail using Wireshark [49]. Transmitting these packets to the AI-based model in real time allowed the system to be tested in complex and realistic network environments. By combining different attack categories, a comprehensive and balanced cybersecurity dataset was created for both training and testing.

In this study, raw network traffic data was analyzed by saving it directly in CSV format. This choice is important because it is compatible with widely used data processing tools. As shown in Fig. 1, the attack traffic was first monitored in real time with Wireshark and then automatically converted to Comma-Separated Values (CSV). This approach allowed for direct analysis of the data without the need for additional modules like PyShark or similar. Wireshark's CSV output feature facilitated the organized structure of the data and the comparative evaluation of different attack scenarios. This file structure also facilitated faster processes during model training and live attack detection. While ensuring the diversity of the dataset, special emphasis was placed on core network protocols, as indicated in Table 2. In addition to Transmission Control Protocol (TCP), Internet Control Message Protocol (ICMP), and DNS, other protocols were included to expand the potential attack surface [50]. This aimed to enable the model to learn different protocol behaviors and generalize more robustly. The data collection and preprocessing method increased both transparency and reproducibility, significantly strengthening data quality and analysis modularity.

To increase the transparency of the study and enable other researchers to easily work with the data, all network traffic data used has been made available on the Hugging Face platform. Interested readers can access the dataset via the following link: Hugging Face dataset link.

3.1. Data collection and preprocessing

The dataset consists of examples of various attack types as well as normal (benign) network traffic. All network traffic was collected within a controlled and monitored test network that emulated a small

Table 2
Representative subset of labeled network flows with key features and attack/benign labels.

No	Time	Source IP	Destination IP	Protocol	Size	Info	Label
51	2.412880	192.168.1.105	2.20.134.138	TLSv1.2	2595	Application Data	**benign**
257	175.221991	10.0.2.15	10.0.2.2	SSHv2	166	Client: Encrypted packet (len=100)	**brute_force**
736	8.107803	10.128.0.2	10.0.0.2	HTTP	534	HTTP/1.1 404 Not Found (text/html)	**exploit**
914	71.606031	2.16.119.157	10.0.0.168	TCP	1514	443 > 49701 [ACK] Seq=8866 Ack=696 Win=64128 Len=1460 [TCP PDU reassembled in 919]	**malware**
8862	3.616471	10.0.0.2	10.128.0.2	ICMP	42	Echo (ping) request id=0 × 5c12, seq=43665/37290, ttl=64 (no response found!)	**flood**
5811	244.734036	10.0.2.2	10.0.2.15	TCP	66	5432 > 49746 [ACK] Seq=2 Ack=519 Win=6912 Len=0 TSval=423676 TSecr=1570586924	**probe**

enterprise environment. Various attacks were generated using standard tools (e.g., Metasploit, hping3, Slowloris, malware sandboxes), and all sensitive information was anonymized in compliance with ethical guidelines. The data collection process strictly adhered to institutional standards and best practices for ethical handling of network information, ensuring that no personal or unauthorized user data was exposed. Each attack type and benign traffic are stored in separate CSV files, and each example is labeled with its class [51]. This structure allows the data to be used directly in training and testing.

The dataset covers heterogeneous multi-protocol traffic rather than being limited to TCP/UDP flows. It includes communication over protocols such as HTTP, HTTPS, FTP, SSH, DNS, SMTP, SMB, NTP, and ICMP, which provides a realistic representation of complex network environments. When creating the dataset, a certain number of samples were allocated for each class to ensure a balanced distribution. Specifically, to increase the model's generalization success, 80% of the samples were used for training and 20% for testing. For each attack type, 4000 training and 1000 test samples were used to maintain balance between classes [49]. This distribution is formulated by Eq. (1):

$$\text{Train Size}_{\text{class}} = 4000, \quad \text{Test Size}_{\text{class}} = 1000 \tag{1}$$

Since there are 6 classes in total, 24,000 samples were used for training and 6000 for testing. Numeric columns such as `No`, `Time`, `Length`, `Source`, `Destination`, and `Protocol` were used as direct input features. Additionally, the textual content of the `Information` column was converted into a 500-dimensional word frequency vector using the "CountVectorizer" method [52]. During this process, missing values were filled with empty strings, making the textual data suitable for vectorization. "CountVectorizer" generates a word count matrix by considering the term frequencies in each sample. This approach is widely used in text-based network traffic analysis [53]. The resulting textual features were incorporated into the model input along with the numeric columns. Although the traffic was captured in a controlled, small-scale setup, the use of heterogeneous multi-protocol communication and diverse attack scenarios was explicitly designed to mimic realistic enterprise conditions. While it may not fully capture the volume and variability of ISP-scale traffic, this experimental design aligns with widely adopted practices in security research and provides a reproducible and ethically manageable dataset.

Since numeric columns (`Time`, `Length`) can have values at different scales, these columns are scaled to the range [0, 1] using the "MinMaxScaler" method. This prevents these features from being disproportionately overridden during the learning process. This process was used to create the final feature matrix, as shown in Eq. (2):

$$\mathbf{X}_{\text{final}} = \text{Concat}\left(\text{MinMaxScaler}(\mathbf{X}_{\text{numeric}}), \mathbf{X}_{\text{count}}\right), \quad \mathbf{X}_{\text{numeric}} \in \mathbb{R}^{N\times d_1} \tag{2}$$

Attack type labels in the dataset were converted to numeric values using the `LabelEncoder` method. This process is performed as shown in Eq. (3):

$$\mathbf{y} = \text{LabelEncoder}(\text{attack\_type}) \tag{3}$$

These preprocessing steps ensure that the network traffic data is represented in a consistent and machine-readable format, which is essential for the learning phase. Once the data is numerically encoded and feature vectors are constructed, it becomes suitable for more complex modeling approaches. Given their ability to capture complex structural relationships within network traffic, graph-based models have become increasingly valuable for uncovering hidden dependencies and improving the effectiveness of attack detection [54–56]. In this study, a Temporal Graph Network (TGN)-based architecture is adopted to leverage both the structural and temporal aspects of network interactions.

3.2. *Temporal Graph Network-based architecture*

The developed anomaly detection system is built on the Temporal Graph Network architecture, which stands out for its ability to model time-sensitive relational data. Unlike traditional graph-based methods, TGN enables deeper representation learning by considering not only the structural but also the temporal dimensions of interactions between nodes. In this context, events observed in network traffic are considered as timestamped directed edges. Each connection consists of the components `source node`, `destination node`, `timestamp`, and `edge features` that define the context.

The basic components of the TGN architecture are structured as follows:

– Memory Module: It contains time-updated memory cells for each node. These cells store information from previous interactions, enabling the system to learn contextual relationships over time. This structure allows the system to adapt to new situations without forgetting the influences of past interactions.
– Message Function: It generates messages based on information about the interaction time with the source and destination nodes. This function forms the basis of the learning process that takes past context into account.
– Message Aggregator: It sequentially aggregates all incoming messages related to the same node throughout the time series to create a summary representation to update the node's memory [57]. This mechanism is typically implemented using strategies such as time-aware attention mechanisms or mean pooling.
– Memory Updater: It updates the node's memory based on the collected messages. This process typically utilizes recurrent neural network cells capable of processing sequential data, such as Gated Recurrent Unit (GRU) or Long Short-Term Memory (LSTM) [58].
– Embedding Module: Generates a high-dimensional vector representation (*node embedding*) for each node based on the memory state and current interaction [59]. These representations are passed to the final layers of the model for use in classification tasks.

The model considers each network traffic packet as a time-stamped interaction between a pair of nodes. For example, a TCP connection between a client and a server is modeled as a directed edge between

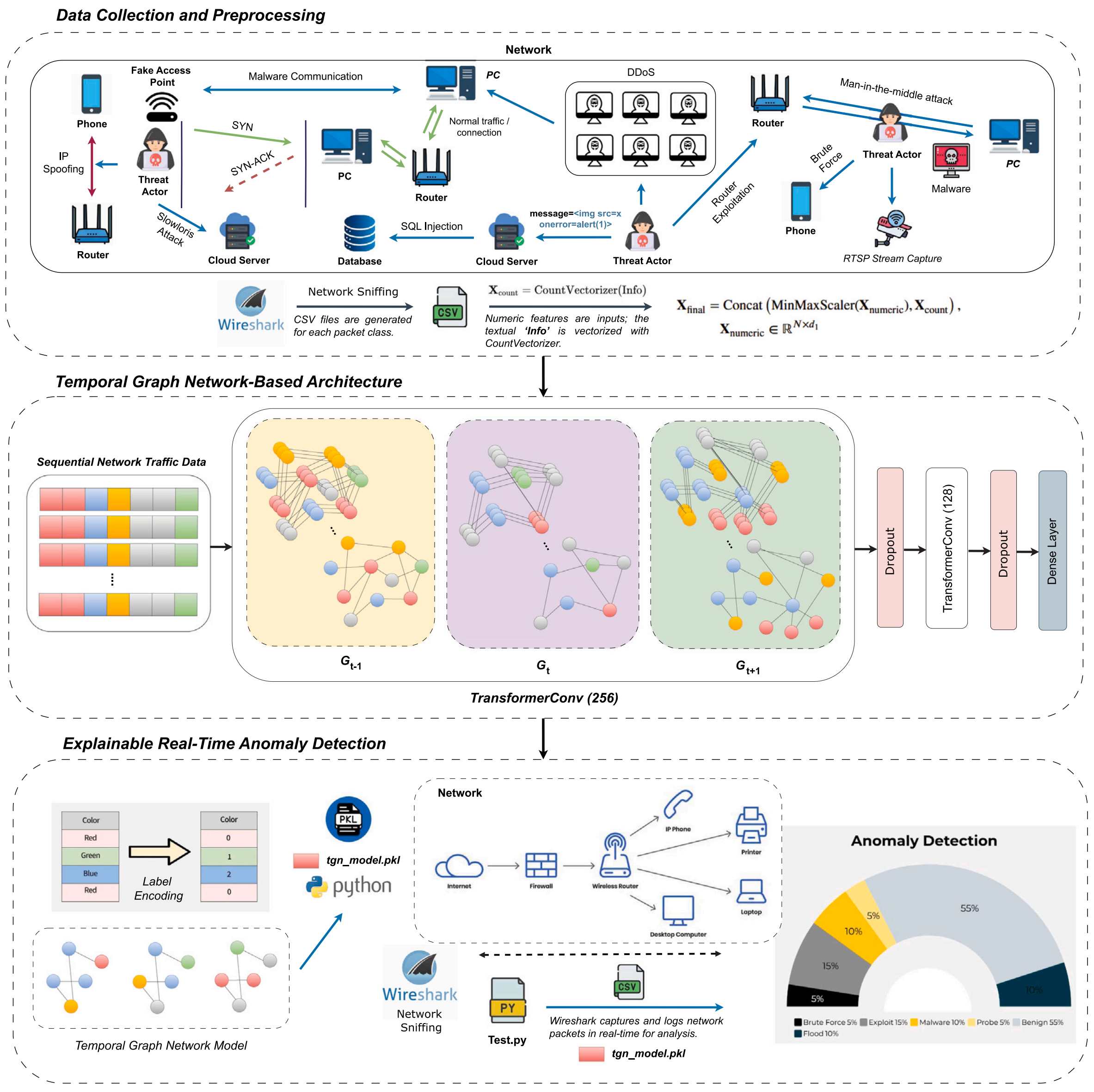

Fig. 1. Detailed flowchart of the proposed approach.

these two IP addresses. Attributes such as protocol type, packet length, TCP flags, and information field within the packet are encoded as edge features. Thanks to this structure, the model learns not only the connection between two nodes but also the connection content and the timeframe in which it occurs. Modeling time-series network interactions in a graph data structure offers significant advantages, particularly in detecting advanced attack patterns (e.g., slow-moving, multi-stage, or time-extended attacks). Therefore, unlike classical CNN or ordinary RNN structures, the TGN architecture can effectively learn temporal connections between nodes within the network and detect even low-frequency but high-impact anomalies.

The architectural representation of the proposed TGN structure, specific to the application, is presented in Fig. 2. The figure shows the time-labeled edges, message generation, memory units, and final embedding vectors as layers. Temporal and structural information are fused at the embedding generation stage rather than at the raw input level. Node memory states, initial feature embeddings, and temporal encodings are combined to produce the final embeddings, ensuring that both time-dependent and relational aspects of network interactions are effectively captured before the classification layer.

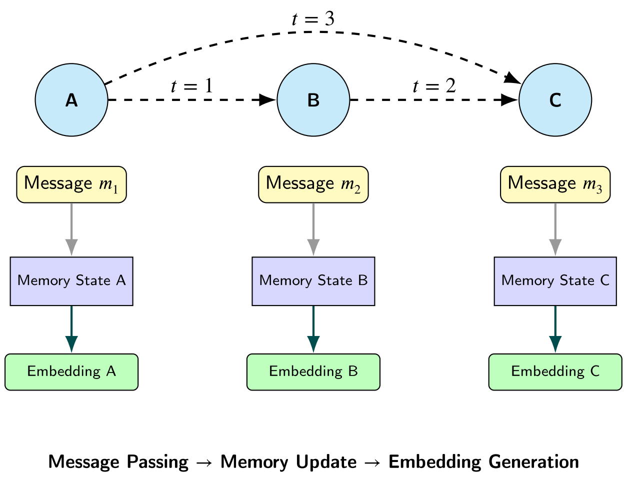

Fig. 2. Temporal Graph Network message passing and embedding for real-time anomaly detection.

The final layer of the model includes a binary or multi-class classifier [60]. This classifier takes embedding vectors as input and predicts whether each sample belongs to a specific attack category or benign traffic. The classification process is primarily performed using a softmax activation function and optimized with a categorical cross-entropy loss function.

Algorithm 1 Temporal Graph Network Model Architecture

```
Require: Feature matrix $X \in \mathbb{R}^{N\times F}$, timestamps $t \in \mathbb{R}^N$, edge list $\mathcal{E}$, edge
    features $E_f$
 1: Initialize: Memory states $\mathcal{M} \leftarrow \text{zeros}(N, H)$, empty message queues $Q_i = \emptyset$
 2: Initial Feature Embedding:
    $H_0 = \text{Linear}_{F\to H}(X), \quad H = 16$
 3: Temporal Encoding:
    $T = \cos(\text{Linear}_{1\to d_t}(t)), \quad d_t = 8$
 4: for each $(i, j, t_k) \in \mathcal{E}$ in temporal order do
 5:     Message Construction:
        $m_{ij}^{(k)} = \text{MLP}([H_0[i], H_0[j], T_k, E_f^{(k)}])$
 6:     Append $m_{ij}^{(k)}$ to $Q_j$
 7: end for
 8: for each node $j$ do
 9:     Aggregate Messages:
        $\tilde{m}_j = \text{TimeAwareAttention}(Q_j)$
10:     Update Memory:
        $\mathcal{M}[j] \leftarrow \text{GRU}(\tilde{m}_j, \mathcal{M}[j])$
11: end for
12: Embedding Generation:
    $Z_0 = \text{MLP}([\mathcal{M}, H_0, T])$
13: TransformerConv Layers:
      First layer with 2 heads (128 dim each):
    $Z_1 = \text{TransformerConv}_{H\to 256}(Z_0, T), \quad Z_1 = \text{Dropout}(Z_1, p = 0.5)$
      Second layer with 1 head to 64-dim:
    $Z_2 = \text{TransformerConv}_{256\to 64}(Z_1, T), \quad Z_2 = \text{Dropout}(Z_2, p = 0.5)$
14: Output Layer:
    $Y = \text{Linear}_{64\to C}(Z_2)$
15: return Logits $Y \in \mathbb{R}^{N\times C}$
```

Algorithm 1 shows the step-by-step architecture of the proposed Temporal Graph Network model. The model starts with a linear layer that transforms the input feature matrix from dimension F to the latent dimension $H = 16$. For temporal information, each timestamp (t) is passed through a learnable linear layer and cosine activation is applied to obtain temporal encodings of dimension $d_t = 8$. To ensure time-aware messaging, the model always generates messages between nodes using the incoming edges in time order. These messages are collected by an attention mechanism that takes into account time differences on a node-by-node basis, and then the node memory is updated with the GRU cell. The updated memory is combined with the initial feature embedding and temporal encodings to generate new node embeddings via MLP. These embeddings are processed by two-stage TransformerConv layers: In the first layer, 256-dimensional intermediate representations with two heads are obtained, and in the second layer, 64-dimensional outputs with a single head are obtained. Regularization is achieved by applying dropouts in both layers. Finally, the resulting 64-dimensional representation is passed through a linear layer with dimensions equal to the number of classes C to produce logit values for anomaly classification. This structure effectively captures both structural and temporal information, improving anomaly detection performance. The model's hyperparameters were systematically explored using a grid search method, and the evaluated ranges along with the selected optimal values are presented in Table 3. This systematic search helped identify optimal combinations of critical parameters such as learning rate, dropout rate, and batch size, thereby improving the model's generalization ability and preventing overfitting. In addition to the described training process, the system allows online adaptation at the level of node memory and embeddings, enabling it to incorporate new interactions in real time without retraining the entire model from scratch. However, full model weight updates still require complete retraining to capture entirely new patterns. This feature supports real-time responsiveness while maintaining model stability.

While the described methodology ensures effective real-time anomaly detection, it is important to recognize potential future challenges. In particular, ML-based anomaly detection models can be vulnerable to adversarial attacks. Carefully crafted perturbations in network traffic may mislead the system, emphasizing the need for robustness and adversarially aware training strategies in subsequent work.

3.3. Explainable real-time anomaly detection

The real-time detection process began by collecting data from live network traffic using Wireshark [61]. Thanks to Wireshark's dynamic packet capture and automatic logging features, new incoming network packets are instantly stored in CSV format, minimizing data loss. This allows for uninterrupted recording of high-volume, constantly changing network traffic. These generated files are monitored by the continuously running test.py script, and each new packet is automatically detected and processed by the system. This allows the data stream to be evaluated in real time, eliminating the need for manual intervention, and the system can instantly adapt to dynamic and updated data.

The test script loads the pre-trained hybrid model (tgn_model.pth) into memory and uses the label_encoder.pkl files to convert class labels in network traffic to numerical values, and the count_vectorizer.pkl files to vectorize textual content based on word frequency. This standardizes different data types and makes them compatible with the model's input format. The `Info` content in the text column is converted using the "CountVectorizer" method, a natural language processing technique. This converts important keywords and patterns in the packet content into numerical form, allowing the model to analyze them more effectively. In addition, numerical features such as `Time` and `Length` are normalized using MinMaxScaler, thus harmonizing data at different scales. This integrated phase ensures that each incoming packet is converted into a consistent and standardized feature vector. The model uses these vectors to classify network traffic and dynamically detect potential threats. This enables real-time network security management, enabling rapid and effective response to various types of attacks. Furthermore, the system's scalability and modular structure allow for easy addition of new classes or features and model updates. This flexibility facilitates adaptation to an ever-changing threat landscape.

Fig. 1 details the data flow and interaction between Wireshark, the test script, and the model, clearly demonstrating the coordinated operation of system components. Furthermore, explainable artificial intelligence (XAI) techniques are integrated into the proposed system to enhance interpretability and trustworthiness. Specifically, feature importance scores are calculated based on the average absolute gradients of the input features with respect to the model's loss, providing a quantitative measure of each feature's contribution to the classification decision. For textual packet information in the "Info" field, the system leverages a CountVectorizer representation, while numerical features such as "Time" and "Length", as well as protocol-based one-hot encoded features, are included in the analysis. The computed gradients are then aggregated per attack class to identify the top three features that most strongly influence predictions, and these features are mapped to interpretable descriptions (e.g., port numbers, protocol names, or

Table 3
Comprehensive hyperparameter settings and preprocessing configurations utilized in the Temporal Graph Network model architecture.

Hyperparameter	Detailed description	Specified value
Computational Device	Hardware accelerator designated for model training and inference operations.	CUDA-enabled GPU
Number of Target Classes	Cardinality of distinct attack categories encompassed within the classification task.	6
Objective Function	Loss criterion employed to optimize the predictive capacity of the model during training.	CrossEntropyLoss
Initial Learning Rate	Step size hyperparameter controlling the magnitude of parameter adjustments per optimization iteration.	5.0×10^{-5}
Mini-batch Size	Number of training samples processed concurrently within a single forward/backward propagation cycle.	128
Weight Decay Coefficient	Regularization parameter incorporated within Adam optimizer to mitigate overfitting via penalization of large weights.	1×10^{-4}
Dropout Probability	Stochastic regularization rate applied within Transformer convolutional layers to prevent model overfitting.	0.5
Total Training Epochs	Number of complete iterations over the entire training dataset.	100
Input Feature Vector Dimensionality	Aggregated dimensionality of concatenated numeric and textual feature representations fed into the model.	502
Hidden Layer Dimensionality	Number of neurons in the model's hidden linear transformation layer, controlling representation capacity.	16
CountVectorizer Maximum Features	Upper bound on the number of unique tokens extracted from the textual `Info` field via bag-of-words encoding.	500
Feature Normalization Technique	Methodology applied to scale numeric features within a predefined range to ensure consistent input distribution.	Min–Max Scaling

packet characteristics) to facilitate human understanding. This approach enables a class-wise explanation of the TGN model's predictions, revealing which network traffic characteristics are decisive for each type of attack. By incorporating these XAI methods, the system not only achieves high detection performance but also provides transparency, allowing cybersecurity experts to validate the model's reasoning, reduce potential false alarms, and make more informed operational decisions. While the integration of the explainability module introduces additional computations for gradient-based feature importance calculations, the system maintains real-time performance. The XAI computations are efficiently parallelized and performed on small batches of incoming packets, ensuring that the processing delay per packet remains negligible. In practice, this allows the model to provide interpretable predictions without compromising the responsiveness required for live network traffic monitoring.

Despite its strong performance, the system has limitations. Stealthy or low-frequency attacks, resembling benign traffic, may go undetected, while the growing use of encrypted protocols (e.g., TLS, SSH) reduces visibility of features. Variability in benign traffic can also lead to false positives. Moreover, although the Temporal Graph Network achieves high accuracy, its complexity limits interpretability. XAI techniques help mitigate this, but balancing transparency with performance remains an open challenge for future work. Concept drift may occur as network traffic patterns evolve, potentially reducing model accuracy if the system is not periodically retrained or updated. Additionally, adversarial attacks could manipulate packet characteristics to evade detection, highlighting the need for robust defenses and continuous monitoring. Recognizing these risks ensures that the system's limitations are transparent and guides future improvements for secure and reliable real-world deployment.

4. Experimental results and evaluation

This section provides a detailed evaluation of the performance of the proposed TGN-based and XAI-enabled model for anomaly detection in real-time network traffic. The model's ability to analyze temporal and structural patterns in real time is measured using accuracy and loss curves, the area under the Receiver Operating Characteristic (ROC), and a confusion matrix that visualizes the success rates for each class. The findings demonstrate that the model performs attack detection with high accuracy and, thanks to the XAI module, can explain decision-making processes transparently. Furthermore, the model demonstrates strong discrimination performance, particularly for attack types that are difficult to distinguish, such as probe, brute-force, and exploit. This demonstrates that the model offers a reliable solution not only in terms of overall accuracy but also in terms of class-based effectiveness.

To visually and numerically evaluate the model's performance, the results presented in Figs. 4 and 3 were examined in detail. The training process's performance is visualized in Fig. 3 using loss and accuracy curves. The high loss values observed in the early stages decreased over time as the model minimized its errors and reached stable low loss levels on both the training and test datasets. After approximately 100 epochs, the model achieved 98% accuracy on the training data and 97% accuracy on the test data. The parallelism between test accuracy and training accuracy indicates that the model successfully minimized the overfitting problem. The rapid early drop in the loss curves supports the model's optimization with an effective learning strategy.

The confusion matrix in Fig. 4 reveals that the model can classify samples belonging to six different classes (benign, brute force, exploit, flood, malware, probe) with high accuracy. The highest confusion rate is observed between the benign and malware classes, which is generally due to the partial overlap of the data patterns between these two classes. The model's overall classification performance is satisfactory, even among classes that are difficult to distinguish, such as probe and brute force. The model also demonstrated nearly error-free classification success in flood and exploit attacks. The results demonstrate the model's high sensitivity and accuracy in distinguishing normal from abnormal behavior in network traffic.

The multi-class ROC curve presented in Fig. 4 quantitatively demonstrates the model's discriminatory performance for each attack type. The Area Under the Curve (AUC) value was 1.00 for the brute force, exploit, flood, and probe classes, demonstrating almost perfect discrimination in these classes. The AUC values for the benign and malware classes were 0.99, respectively. These results clearly demonstrate the model's success in maximizing the true positive rate and minimizing the false positive rate. In other words, the proposed TGN-based and XAI-enhanced model demonstrated superior performance in anomaly

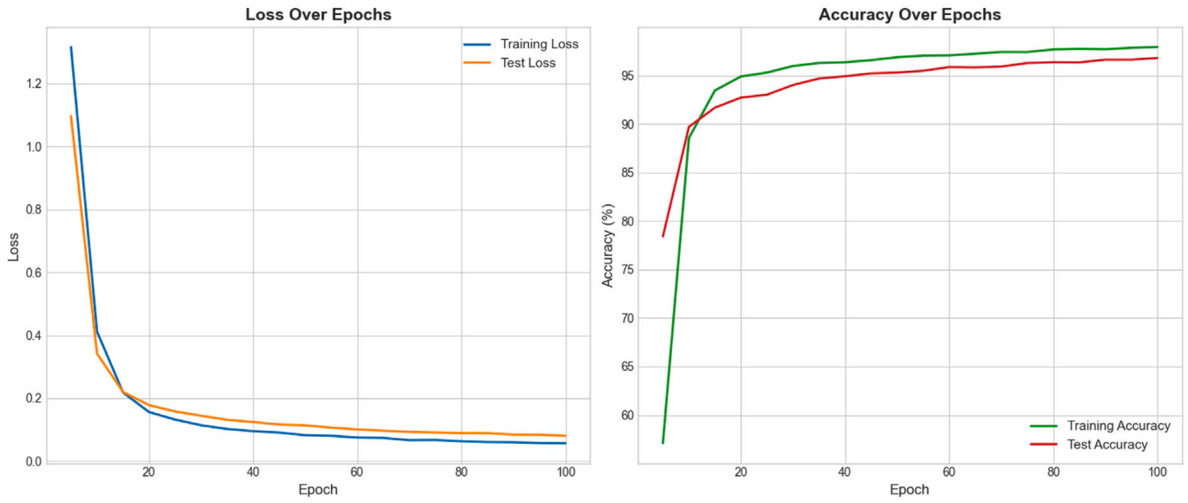

Fig. 3. Training and test loss and accuracy curves for the proposed model.

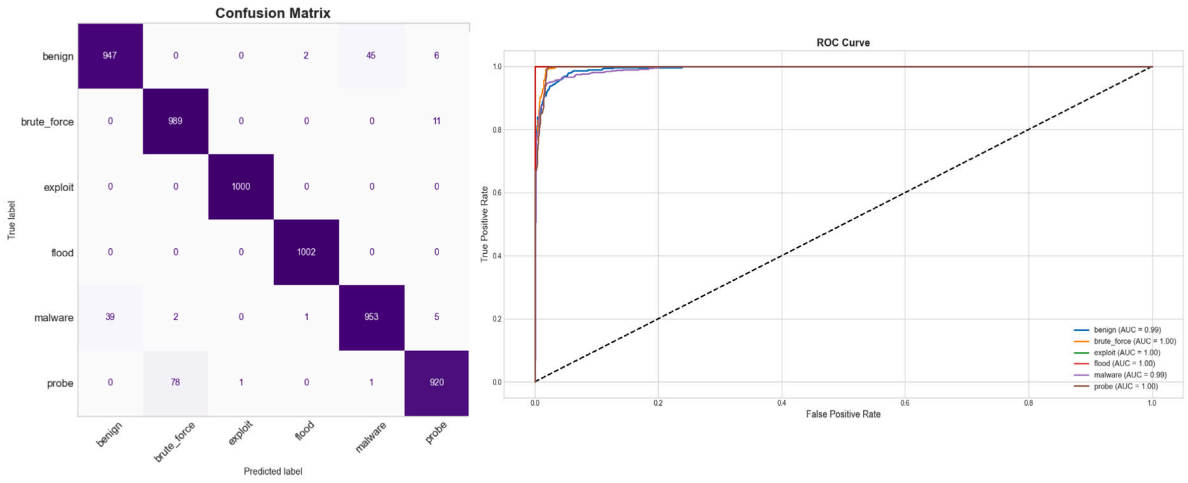

Fig. 4. Evaluation of the proposed model's classification performance using the confusion matrix and ROC curves across all classes.

Table 4
Comparison of performance metrics for various models.

Model	Accuracy	F1-score	Precision	Recall	Specificity
KNN	0.785	0.786	0.794	0.785	0.957
Random Forest	0.938	0.938	0.939	0.937	0.987
XGBoost	0.749	0.700	0.838	0.749	0.950
RoBERTa	0.961	0.961	0.962	0.961	0.992
LSTM	0.953	0.954	0.956	0.953	0.990
GraphSAGE	0.933	0.933	0.933	0.933	0.986
GCN	0.942	0.942	0.943	0.942	0.988
GRU	0.955	0.955	0.956	0.955	0.991
GCN+CNN	0.954	0.953	0.955	0.954	0.990
Graph Transformer	0.951	0.951	0.952	0.951	0.990
GIN	0.954	0.954	0.955	0.954	0.991
TGN	**0.9682**	**0.9681**	**0.9687**	**0.9682**	**0.9936**

detection in real-time network traffic by simultaneously analyzing temporal and structural features. The model provided a successful solution not only in terms of overall accuracy but also in terms of its ability to distinguish and explain each attack type. With the XAI module, the model's decision-making mechanism became transparent and interpretable, providing a significant advantage for the model's integration into cybersecurity systems. In addition, it is evaluated that the model can be used as a reliable building block in real-time threat detection systems thanks to its low latency, high accuracy rates, and class-based stability.

Supporting this general evaluation, Table 4 presents a comparative analysis of different machine learning and deep learning-based models in terms of various performance metrics. Metrics such as accuracy, F1-score, precision, recall, and specificity in the table reveal both the overall success of the models and their class-based stability. According to the results, the proposed TGN model demonstrated the highest success in all metrics. With an accuracy rate of 96.82%, its correct prediction capacity is quite high, and with an F1-score of 96.81%, a balance between precision and sensitivity is achieved. A precision value of 96.87% indicates that the model has a low false positive rate, while

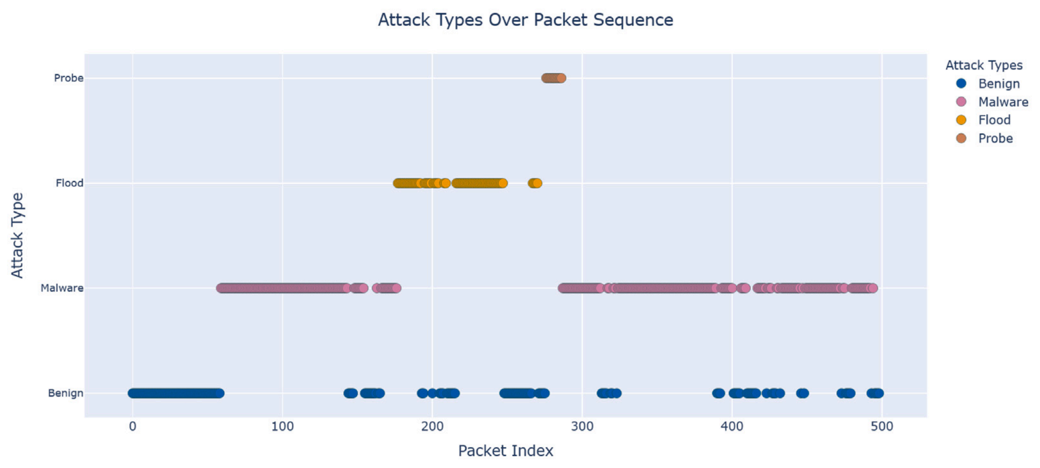

Fig. 5. Distribution of attack types in time according to packet order.

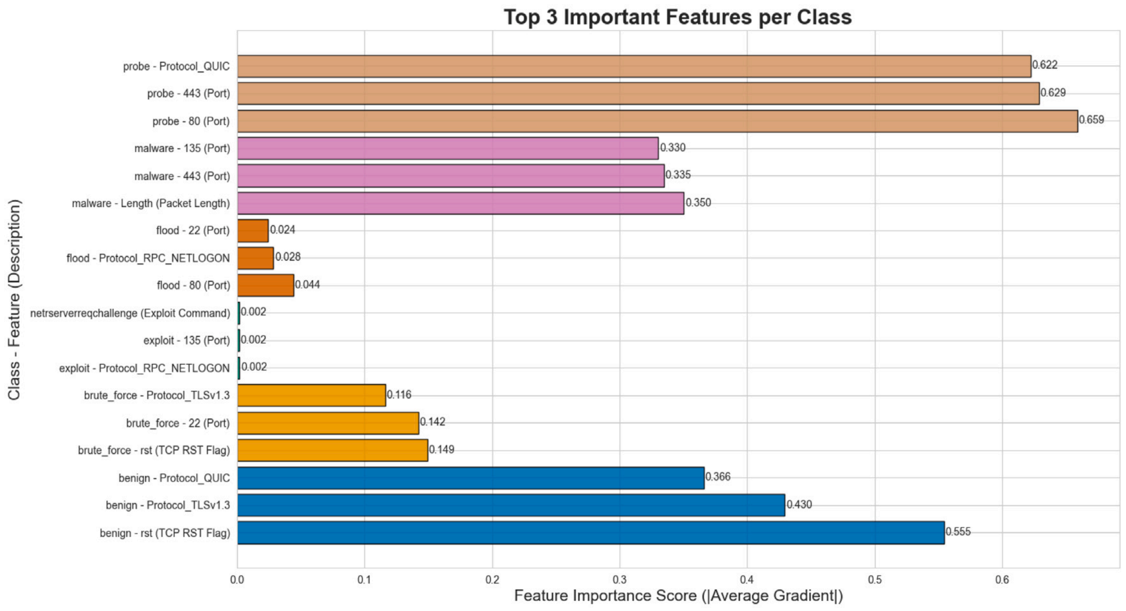

Fig. 6. Top 3 features per class from the TGN model with XAI-based explainability.

a sensitivity of 96.82% indicates that it successfully detects the vast majority of attacks. A highly high specificity value of 99.36% is critical in demonstrating that normal traffic is not mistakenly perceived as a threat.

Another test dataset used effectively consists of a total of 500 daily packets containing samples of benign failures, as well as malware and flood attacks. The packets were analyzed according to their original sequence number over the time axis, and as illustrated in Fig. 5, the distribution of different attack types over time was visualized. The visualization shows that malware samples are clustered within specific intervals, while flood attacks appear as dense and short-lived bursts in the packet sequence. Benign traffic, on the other hand, exhibits a more widespread and irregular distribution, clearly distinguishable from the clusters of attack traffic. A small number of probe instances were also observed; however, in a dataset with multiple complex classes, such occurrences may represent inherent weaknesses or limitations in classification. Overall, time series-based visualization and sequence information provide meaningful indicators that can assist in the rapid classification and prioritization of attack types in real-time monitoring applications.

While other deep learning models capable of processing sequential data, such as RoBERTa, LSTM, and GRU, have also demonstrated high performance, TGN appears to surpass these models due to its ability to handle both temporal and structural context. Graph-based models, GraphSAGE and GCN, while successfully processing structural information, were not as successful as TGN due to their inability to adequately account for time-dependent patterns. Traditional methods such as KNN, Random Forest, and XGBoost, on the other hand, exhibited relatively poor performance in terms of accuracy and F1-score. The XGBoost model, in particular, exhibited high precision but a low

Table 5
Ablation showing the effect of different proposed model components and configurations on model accuracy.

Model component	Accuracy
CountVectorizer (max_features=200)	0.938
CountVectorizer (max_features=500)	0.968
CountVectorizer (max_features=2000)	0.959
Without Dropout Layer	0.955
Without TransformerConv2	0.962
(Conv1: 32× 2 → Conv2: 16× 1)	0.940
(Conv1: 32× 2 → Conv2: 32× 1)	0.952
(Conv1: 64× 2 → Conv2: 64× 1)	0.961

F1-score, demonstrating its unbalanced predictive power. In light of all this data, the TGN model appears to excel not only in its overall success rates but also in critical areas such as accurate attack classification, protection of normal traffic, and interpretability. This comprehensive success profile clearly demonstrates that the model is a strong candidate for integration into real-time threat detection systems. To better understand the factors behind this strong performance, an ablation study was conducted to examine how each component and configuration of the model contributes to its overall accuracy. Table 5 presents an ablation study evaluating the impact of different components and configurations of the proposed model on model accuracy. The results highlight how variations in the TransformerConv layer dimensions, the presence or absence of the dropout layer, and other modifications affect overall performance. Increasing the number of features in both Conv1 and Conv2 layers led to the highest accuracy, while removing Conv2 or the dropout layer resulted in a slight performance drop. These findings indicate that both convolutional layers and dropout play an important role in stabilizing training and enhancing predictive performance. Overall, the study provides a clear understanding of how each model component contributes to the final results.

As shown in Fig. 6, to clarify the model's decision-making mechanism and demonstrate its explainability (XAI), importance scores based on the average gradient values of the three most important features for each attack class were determined and visualized. This analysis clearly reveals the traffic characteristics by which the model distinguishes different attack types. The findings indicate that the TGN model's decision-making process is neither random nor meaningless, but is shaped by signals appropriate to the characteristic behaviors of the attack types. In particular, the identification of Port 80, Port 443, and Protocol_QUIC as the features with the highest importance scores in probe attacks is entirely consistent with the fact that scanning attacks are often conducted on common HTTP/HTTPS ports or modern protocols. This demonstrates the model's ability to accurately identify potential threat surfaces. The prominence of distinguishing features such as Packet Length and Port 135 in malware traffic demonstrates that the model accurately reflects the unique packet sizes of malware and their tendency to use ports associated with protocols such as Server Message Block (SMB). To further demonstrate the practical utility of the explainability analysis, we highlight how these findings can assist human analysts in real-world settings. For instance, in the case of probe attacks, the model identifies Port 80, Port 443, and Protocol_QUIC as dominant features. This directly informs analysts that abnormal scanning behavior often targets widely used web protocols, which aligns with common adversarial tactics and supports prioritizing monitoring on these services. Similarly, for malware traffic, the high importance of Packet Length and Port 135 indicates that malicious activity tends to exploit specific protocol behaviors (e.g., SMB) and exhibit abnormal payload sizes, which are well-known forensic indicators. In the benign class, the prominence of standard TLS features provides confidence that the model distinguishes normal encrypted traffic patterns from malicious misuse. During our internal evaluation, analysts reported that these feature-level insights increased their confidence in the system's outputs and reduced the time spent correlating alerts with protocol-level evidence. Thus, the explainability framework not only clarifies model behavior but also directly enhances its usability in operational cybersecurity environments.

In flood attacks, features such as Port 22, Port 80, and Protocol_RPC_NETLOGON demonstrate that the model can distinguish between high-density connection attempts concentrated on specific protocols and ports. However, the lower importance scores in the flood class compared to the other classes indicate that this attack type has more subtle and variable signals at the traffic level. In the exploit class, the significantly lower importance scores (around 0.002) indicate that the model uses limited signals for this attack type; this is a significant indicator that additional feature engineering is necessary for exploit detection. The TCP RST flag and Port 22 were among the most important features for brute-force attacks, demonstrating that the model correctly interprets frequent connection resets due to numerous failed login attempts. Finally, the high importance scores of modern secure communication protocols such as Protocol_QUIC and Protocol_TLSv1.3 in the benign traffic class clearly demonstrate the model's ability to recognize normal user traffic in accordance with current internet protocols. Taken as a whole, these findings demonstrate that the features used by the TGN model to distinguish attack types are consistent with both expectations in the network security literature and real-world attack scenarios. The model not only achieves high accuracy rates but also strongly supports explainable AI objectives by meaningfully and interpretably demonstrating which features are distinctive for each attack type. Building on this strong foundation, the real-time anomaly detection system has been developed to meet critical performance requirements.

Table 6
Real-time detection latency, throughput, and memory usage. (Throughput is computed as *Processed Packets/Latency*).

Processed packets	Latency (s)	Throughput	Memory (MB)
500	0.32	1562.50	37
50,000	1.45	34 482.76	423
500,000	12.56	39 744.04	3856

Table 7
Performance analysis of the model in attack detection on cross-datasets.

Attack type	Malware dataset	Brute force dataset
Malware	23,225	51
Flood	35	0
Exploit	49	19
Benign	91	17
Probe	9	4
Brute Force	33	4932

In real-time anomaly detection, maintaining low latency and ensuring system scalability against increasing data loads are crucial. As shown in Table 6, the proposed system has been designed and tested to meet these requirements. The results demonstrate that the system can perform fast and effective detection by keeping latencies to a minimum, even under heavy traffic. Furthermore, thanks to its scalable architecture, it can seamlessly handle increasing packet flows, thus providing reliable performance in live network environments. These features demonstrate that the system is suitable for practical applications in real-time anomaly detection processes. In addition to the average values reported in Table 6, the variance and maximum observed delays were also examined. Since each packet follows the same processing pipeline, latency fluctuations are negligible, and the worst-case delays remain very close to the averages. For example, the maximum observed latency was 0.35 s for 500 packets and 13.1 s for 500,000 packets, which are almost identical to their corresponding averages. This confirms that the proposed system does not suffer from sporadic spikes and maintains stable latency, a critical requirement for real-time deployment.

The generalization capability of the developed model was evaluated through cross-dataset tests. As shown in Table 7, the model was able to classify both malware and brute-force attacks with high accuracy across different datasets. The model's classification performance is particularly notable in malware and brute force attacks, correctly identifying 23,225 malware packages and 4932 Brute Force packages. The model also demonstrated reasonable performance across other attack types. These results demonstrate the model's robust ability to recognize new attack types and its effective generalization to diverse datasets.

The TGN-based model developed within the scope of this study not only demonstrated high success rates but also made a significant contribution to the field of explainable artificial intelligence by revealing which network traffic characteristics play a decisive role in the attack detection process. The ability to transparently analyze the model's decision-making mechanisms not only increases model reliability but also allows cybersecurity experts to make more informed interpretations of system outputs. To further evaluate the quality of explanations, the outputs of the XAI module were examined using a combination of domain knowledge and proxy metrics. Specifically, the model highlights the network traffic features that most influence its predictions, and these explanations were compared against established cybersecurity knowledge and known attack patterns, verifying that the identified features align with critical indicators of malicious activity. Additionally, proxy metrics such as feature importance rankings and consistency of explanations across similar attack scenarios were used to quantitatively assess the reliability and stability of the explanations. Together, these findings demonstrate that the proposed approach is not limited to technical success but also provides meaningful, interpretable, and trustworthy insights at the operational level. In this respect, the model stands out as a holistic AI solution capable of simultaneously meeting both high accuracy and explainability requirements in real-time threat detection systems.

5. Conclusion

The emphasis of this research is on the successful implementation of a Temporal Graph Network (TGN)-based model, which also uses an Explainable Artificial Intelligence (XAI) module, to provide anomaly detection for real-time network traffic analysis. The model is proposed to outperform traditional methods because it can learn and explain graph structures for timestamped directed relationships due to network traffic. The TGN architecture effectively captures both temporal and structural relationships between network nodes, while the XAI module quantifies feature importance for each attack class, providing transparent and interpretable decision-making. In experiments conducted in real network environments, the model successfully classified attack traffic collected using Wireshark and identified attack types classified into six different categories with high accuracy. The model's 96.82% accuracy, particularly for difficult-to-distinguish attack types such as probes, brute force attacks, and exploits, demonstrates the system's strong generalization capabilities. The model's ability to work with time-series data enabled it to identify temporal patterns in network attacks successfully. Additionally, the classification performance of the model was compared with other traditional and deep learning-based methods like XGBoost, and it was found that the proposed TGN model outperformed all key metrics, including F1 score, precision, recall, and specificity. By explicitly incorporating temporal information, the TGN model outperformed GCN, GraphSAGE, LSTM, and GRU models, particularly in accurately detecting time-dependent attack patterns such as probes and brute-force attempts. As part of the model's real-time operation, network packets captured in real time with Wireshark were integrated into the test scenario. While the proposed model demonstrates high accuracy for most attack types, extremely low-frequency or stealthy attacks may still present challenges. Incorporating additional context-aware features or adaptive thresholding could further enhance the detection of such subtle anomalies. Although the system currently analyzes network traffic directly from Wireshark captures and CSV exports, its modular design allows outputs to be exported in standard formats (e.g., JSON or Syslog) for integration with SIEM platforms and intrusion prevention systems, enabling real-time alerts and enhanced situational awareness. The dataset used in this study is openly available on the Hugging Face platform, encouraging other researchers to produce reproducible work. Thus, the study not only makes a theoretical contribution but also provides a framework that will benefit the community in line with the open science approach. The proposed methodology presents an innovative view into cybersecurity, by extending beyond detecting an attack on a network, it gives temporal characteristics of its impacts on the network. In this regard, the study has made practical and useful contributions to the academic literature on network security.

Abbreviations

AUC	Area Under the Curve
CSV	Comma-Separated Values
CTI	Cyber Threat Intelligence
DDoS	Distributed Denial of Service
DNS	Domain Name System
DPI	Deep Packet Inspection
GRU	Gated Recurrent Unit
ICMP	Internet Control Message Protocol
IDS	Intrusion Detection System
IoT	Internet of Things
IP	Internet Protocol
IPS	Intrusion Prevention System
LSTM	Long Short-Term Memory
LTE	Long Term Evolution
MLM	Multi-Layered Security
NTP	Network Time Protocol
NFV	Network Functions Virtualization
OSI	Open Systems Interconnection
PCA	Principal Component Analysis
QUIC	Quick UDP Internet Connections
ROC	Receiver Operating Characteristic
RTNM	Real-Time Network Monitoring
SMB	Server Message Block
SSDP	Simple Service Discovery Protocol
TCP	Transmission Control Protocol
TGN	Temporal Graph Network
TGNs	Temporal Graph Networks
TLS	Transport Layer Security
UDP	User Datagram Protocol
WAF	Web Application Firewall
XAI	Explainable Artificial Intelligence

Declaration of competing interest

The author declares that there are no known competing financial interests or personal relationships that could appear to influence the work reported in this paper.

References

[1] Ö. Aslan, S.S. Aktuğ, M. Ozkan-Okay, A.A. Yilmaz, E. Akin, A comprehensive review of cyber security vulnerabilities, threats, attacks, and solutions, Electronics 12 (6) (2023) 1333.
[2] Z. Huang, Z. Wang, Assessing the robustness of physical networks under attack uncertainty, Reliab. Eng. Syst. Saf. 262 (2025) 111231, http://dx.doi.org/10.1016/j.ress.2025.111231.
[3] Y. Li, Q. Liu, A comprehensive review study of cyber-attacks and cyber security; emerging trends and recent developments, Energy Rep. 7 (2021) 8176–8186.
[4] B.B. Gupta, M. Quamara, An overview of internet of things (IoT): Architectural aspects, challenges, and protocols, Concurr. Comput.: Pr. Exp. 32 (21) (2020) e4946.
[5] M.M. Inuwa, R. Das, A comparative analysis of various machine learning methods for anomaly detection in cyber attacks on IoT networks, Internet Things 26 (2024) 101162, http://dx.doi.org/10.1016/j.iot.2024.101162.
[6] M. Baykara, R. Das, A novel honeypot based security approach for real-time intrusion detection and prevention systems, J. Inf. Secur. Appl. 41 (2018) 103–116, http://dx.doi.org/10.1016/j.jisa.2018.06.004.
[7] H. Ahmetoglu, R. Das, A comprehensive review on detection of cyber-attacks: Data sets, methods, challenges, and future research directions, Internet Things 20 (2022) 100615, http://dx.doi.org/10.1016/j.iot.2022.100615.
[8] H. Gao, R. Xin, P. Chen, et al., Memory-augment graph transformer based unsupervised detection model for identifying performance anomalies in highly-dynamic cloud environments, J. Cloud Comput. 14 (40) (2025) http://dx.doi.org/10.1186/s13677-025-00766-5.
[9] B. Tu, T. Zhou, B. Liu, Y. He, J. Li, A. Plaza, Multi-scale autoencoder suppression strategy for hyperspectral image anomaly detection, IEEE Trans. Image Process. (2025) http://dx.doi.org/10.1109/TIP.2025.3595408, 1–1, Advance online publication.
[10] Y. Otoum, A. Nayak, As-ids: Anomaly and signature based ids for the internet of things, J. Netw. Syst. Manage. 29 (3) (2021) 23.
[11] I. Martins, J.S. Resende, P.R. Sousa, S. Silva, L. Antunes, J. Gama, Host-based IDS: A review and open issues of an anomaly detection system in IoT, Future Gener. Comput. Syst. 133 (2022) 95–113.
[12] H. Alturkistani, S. Chuprat, Artificial intelligence and large language models in advancing cyber threat intelligence: A systematic literature review, Available at SSRN 4903071.
[13] N. Sun, M. Ding, J. Jiang, W. Xu, X. Mo, Y. Tai, J. Zhang, Cyber threat intelligence mining for proactive cybersecurity defense: A survey and new perspectives, IEEE Commun. Surv. Tutor. 25 (3) (2023) 1748–1774.
[14] A. Alshamrani, S. Myneni, A. Chowdhary, D. Huang, A survey on advanced persistent threats: Techniques, solutions, challenges, and research opportunities, IEEE Commun. Surv. Tutor. 21 (2) (2019) 1851–1877.
[15] M. Yue, H. Yan, R. Han, Z. Wu, A DDoS attack detection method based on IQR and DFFCNN in SDN, J. Netw. Comput. Appl. 240 (2025) 104203, http://dx.doi.org/10.1016/j.jnca.2025.104203.
[16] M. Anagnostopoulos, S. Lagos, G. Kambourakis, Large-scale empirical evaluation of DNS and SSDP amplification attacks, J. Inf. Secur. Appl. 66 (2022) 103168.
[17] D. Demirol, R. Das, D. Hanbay, A novel approach for cyber threat analysis systems using BERT model from cyber threat intelligence data, Symmetry 17 (4) (2025) 587, http://dx.doi.org/10.3390/sym17040587.
[18] S. Lakshmanan, U. Maheswari Gnaniyan Ponnusamy, S. Andi, An efficient ddos attack detection using chaos henry gas solubility optimization weight initialization based rectified linear unit, Cybern. Syst. (2023) 1–28.
[19] A. Ali, M.A. Khan, K. Farid, S.S. Akbar, A. Ilyas, T.M. Ghazal, H. Al Hamadi, The effect of artificial intelligence on cybersecurity, in: 2023 International Conference on Business Analytics for Technology and Security, ICBATS, IEEE, 2023, pp. 1–7.
[20] Z. Zhang, H. Ning, F. Shi, F. Farha, Y. Xu, J. Xu, F. Zhang, K.-K.R. Choo, Artificial intelligence in cyber security: research advances, challenges, and opportunities, Artif. Intell. Rev. (2022) 1–25.
[21] S. Zhou, Z. He, X. Chen, W. Chang, An anomaly detection method for UAV based on wavelet decomposition and stacked denoising autoencoder, Aerospace 11 (5) (2024) 393, http://dx.doi.org/10.3390/aerospace11050393.
[22] Y. Qiao, T. Wang, J. Lü, K. Liu, TEMPO: Time-evolving multi-period observational anomaly detection method for space probes, Chin. J. Aeronaut. 38 (9) (2025) 103426, http://dx.doi.org/10.1016/j.cja.2025.103426.
[23] Y. Gong, H. Yao, A. Nallanathan, Intelligent sensing, communication, computation and caching for satellite-ground integrated networks, IEEE Netw. (2024).
[24] A. Yılmaz, R. Das, A novel hybrid approach combining GCN and GAT for effective anomaly detection from firewall logs in campus networks, Comput. Netw. 259 (2025) 111082.
[25] Y. Qiao, J. Lü, T. Wang, K. Liu, B. Zhang, H. Snoussi, A multihead attention self-supervised representation model for industrial sensors anomaly detection, IEEE Trans. Ind. Informatics 20 (2) (2024) 2190–2199, http://dx.doi.org/10.1109/TII.2023.3280337.
[26] T. Sowmya, E.M. Anita, A comprehensive review of AI based intrusion detection system, Meas.: Sensors 28 (2023) 100827.
[27] M.O. Kaya, M. Ozdem, R. Das, A novel approach for graph-based real-time anomaly detection from dynamic network data listened by wireshark, EAI Endorsed Trans. Ind. Networks Intell. Syst. 12 (2) (2025).
[28] Y. Liu, W. Li, X. Dong, Z. Ren, Resilient formation tracking for networked swarm systems under malicious data deception attacks, Internat. J. Robust Nonlinear Control 35 (6) (2025) 2043–2052.
[29] G. Fernandes, J.J. Rodrigues, L.F. Carvalho, J.F. Al-Muhtadi, M.L. Proença, A comprehensive survey on network anomaly detection, Telecommun. Syst. 70 (2019) 447–489.
[30] R. Das, M. Soylu, A key review on graph data science: The power of graphs in scientific studies, Chemometr. Intell. Lab. Syst. 240 (104896) (2023).
[31] S. Ahmad, S. Jha, A. Alam, M. Alharbi, J. Nazeer, [Retracted] Analysis of intrusion detection approaches for network traffic anomalies with comparative analysis on botnets (2008–2020), Secur. Commun. Networks 2022 (1) (2022) 9199703.
[32] Q. Zhang, M. Ye, X. Deng, A novel anomaly detection method for multimodal WSN data flow via a dynamic graph neural network, Connect. Sci. 34 (1) (2022) 1609–1637.
[33] E. Caville, W.W. Lo, S. Layeghy, M. Portmann, Anomal-E: A self-supervised network intrusion detection system based on graph neural networks, Knowl.-Based Syst. 258 (2022) 110030.
[34] P. Kisanga, I. Woungang, I. Traore, G.H. Carvalho, Network anomaly detection using a graph neural network, in: 2023 International Conference on Computing, Networking and Communications, ICNC, IEEE, 2023, pp. 61–65.
[35] J. Kim, K. Kim, G.-y. Jeon, M.M. Sohn, Temporal patterns discovery of evolving graphs for graph neural network (GNN)-based anomaly detection in heterogeneous networks, J. Internet Serv. Inf. Secur. 12 (1) (2022) 72–82.
[36] M. Jain, A. Shah, Anomaly detection using convolutional neural networks (CNN), ESP Int. J. Adv. Comput. Technol. (ESP-IJACT) 2 (3) (2024) 12–22.
[37] Z. Wang, J. Tian, H. Fang, L. Chen, J. Qin, LightLog: A lightweight temporal convolutional network for log anomaly detection on the edge, Comput. Netw. 203 (2022) 108616.
[38] D. Zhu, Y. Ma, Y. Liu, A flexible attentive temporal graph networks for anomaly detection in dynamic networks, in: 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), IEEE, 2020, pp. 870–875.
[39] M. Lv, C. Dong, T. Chen, T. Zhu, Q. Song, Y. Fan, A heterogeneous graph learning model for cyber-attack detection, 2021.
[40] J. Noble, N. Adams, Real-time dynamic network anomaly detection, IEEE Intell. Syst. 33 (2) (2018) 5–18.
[41] R. Magán-Carrión, J. Camacho, G. Maciá-Fernández, Á. Ruíz-Zafra, Multivariate statistical network monitoring–sensor: an effective tool for real-time monitoring and anomaly detection in complex networks and systems, Int. J. Distrib. Sens. Networks 16 (5) (2020) 1550147720921309.
[42] L. Fernández Maimó, A. Huertas Celdrán, M. Gil Pérez, F.J. García Clemente, G. Martínez Pérez, Dynamic management of a deep learning-based anomaly detection system for 5G networks, J. Ambient. Intell. Humaniz. Comput. 10 (2019) 3083–3097.
[43] C. Ni, J. Wu, H. Wang, Energy-aware edge computing optimization for real-time anomaly detection in IoT networks, Appl. Comput. Eng. 139 (2025) 42–53.
[44] K. Palaniappan, B. Duraipandi, U.M. Balasubramanian, et al., Dynamic behavioral profiling for anomaly detection in software-defined IoT networks: A machine learning approach, Peer-To-Peer Netw. Appl. 17 (4) (2024) 2450–2469.
[45] J. Chen, D. Zhang, Z. Ming, K. Huang, W. Jiang, C. Cui, GraphAttacker: A general multi-task graph attack framework, IEEE Trans. Netw. Sci. Eng. 9 (2) (2022) 577–595.
[46] A.A. Mir, M.F. Zuhairi, S. Musa, A. Namoun, Adaptive anomaly detection in dynamic graph networks, in: 2024 International Visualization, Informatics and Technology Conference, IVIT, IEEE, 2024, pp. 200–206.
[47] A.N. Gummadi, J.C. Napier, M. Abdallah, XAI-IoT: an explainable AI framework for enhancing anomaly detection in IoT systems, IEEE Access 12 (2024) 71024–71054.
[48] C.S. Babu, et al., Adaptive AI for dynamic cybersecurity systems: Enhancing protection in a rapidly evolving digital landscape, in: Principles and Applications of Adaptive Artificial Intelligence, IGI Global, 2024, pp. 52–72.
[49] M.O. Kaya, M. Ozdem, R. Das, A new hybrid approach combining GCN and LSTM for real-time anomaly detection from dynamic computer network data, Comput. Netw. (2025) 111372.
[50] M.O. Kaya, H.A. Dagdogen, M. Ozdem, R. Das, An effective new penetration testing approach to detect web attacks on web applications, Expert Syst. Appl. (2026) 129623, http://dx.doi.org/10.1016/j.eswa.2025.129623, Online first.
[51] H. Gebrye, Y. Wang, F. Li, Traffic data extraction and labeling for machine learning based attack detection in IoT networks, Int. J. Mach. Learn. Cybern. 14 (7) (2023) 2317–2332.
[52] M. Ahmed, M.N. Uddin, Cyber attack detection method based on nlp and ensemble learning approach, in: 2020 23rd International Conference on Computer and Information Technology, ICCIT, IEEE, 2020, pp. 1–6.
[53] B.J. Jansen, K.K. Aldous, J. Salminen, H. Almerekhi, S.-g. Jung, Data preprocessing, in: Understanding Audiences, Customers, and Users Via Analytics: An Introduction to the Employment of Web, Social, and Other Types of Digital People Data, Springer, 2023, pp. 65–75.

[54] M. Soylu, R. Das, Prediction and graph visualization of cyber attacks using graph attention networks, Comput. Secur. 157 (2025) 104534, http://dx.doi.org/10.1016/j.cose.2025.104534.

[55] C. Huang, C. Gao, M. Li, Y. Li, X. Wang, Y. Jiang, X. Huang, Correlation information enhanced graph anomaly detection via hypergraph transformation, IEEE Trans. Cybern. 55 (6) (2025) 2865–2878, http://dx.doi.org/10.1109/TCYB.2025.3558941.

[56] M. Soylu, R. Das, A hybrid graph neural network model for predicting cyber attacks from heterogeneous and dynamic network data, IEEE Access 13 (2025) 151512–151526, http://dx.doi.org/10.1109/ACCESS.2025.3603403.

[57] A. Riker, E. Cerqueira, M. Curado, E. Monteiro, A two-tier adaptive data aggregation approach for m2m group-communication, IEEE Sensors J. 16 (3) (2015) 823–835.

[58] S. Kasmaiee, S. Kasmaiee, M. Homayounpour, Correcting spelling mistakes in Persian texts with rules and deep learning methods, Sci. Rep. 13 (2023) 19945.

[59] M. Yamada, V. D'Amario, K. Takemoto, X. Boix, T. Sasaki, Transformer module networks for systematic generalization in visual question answering, IEEE Trans. Pattern Anal. Mach. Intell. (2024).

[60] C. Iwendi, S. Khan, J.H. Anajemba, M. Mittal, M. Alenezi, M. Alazab, The use of ensemble models for multiple class and binary class classification for improving intrusion detection systems, Sensors 20 (9) (2020) 2559.

[61] R. Das, G. Tuna, Packet tracing and analysis of network cameras with wireshark, in: 2017 5th International Symposium on Digital Forensic and Security, ISDFS, IEEE, 2017, pp. 1–6.

Mehmet Ozdem received his BS degree from Başkent University, Department of Electrical and Electronics Engineering in 2002, his MS degree from Middle East Technical University, Institute of Informatics in 2007, and his Ph.D. degree from Gazi University, Institute of Science, Department of Electrical and Electronics Engineering in 2021. He has more than 20 years of experience in the Telecommunications industry. He worked in many groups (investment, planning, operation, quality assurance, and architecture teams). He currently works as the Network Architecture & Quality Assurance Director and Head of R&D Centers, following responsibilities in Turk Telekom. He is Vice President of ITU (International Telecommunication Union) SG12 and QSDG organizations. He is also a board member of the Wireless Broadband Alliance. He lectures part-time at universities and holds international certifications, including CCIE, PMP, and ITIL.