XAI Deep Learning Framework for Intelligent Network Intrusion Detection

1st Ms. Ponnila P
Dept. of Artificial Intelligence and Data science
Rajalakshmi Engineering College
Chennai, India
ponnila.p@rajalakshmi.edu.in

2nd Devika C
Dept. of Artificial Intelligence and Data science
Rajalakshmi Engineering College
Chennai, India
221801009@rajalakshmi.edu.in

3rd Keerthika P
Dept. of Artificial Intelligence and Data science
Rajalakshmi Engineering College
Chennai, India
221801027@rajalakshmi.edu.in

4th Lavanya S
Dept.of Artificial Intelligence and Data science
Rajalakshmi Engineering College
Chennai, India
221801028@rajalakshmi.edu.in

Abstract— **Effective network traffic monitoring is essential for ensuring secure communication, minimizing cyber threats, and maintaining uninterrupted service availability in large-scale infrastructures. Traditional network monitoring systems often fall short due to limited scalability, high latency, and the absence of real-time analytics and explainability, resulting in delayed threat detection and inefficient network management. This paper proposes an AI-powered real-time network flow monitoring and intrusion detection framework that integrates Apache Kafka with advanced machine learning and Explainable AI (XAI) techniques to achieve scalable, transparent, and intelligent network analysis. The architecture is composed of multiple layers, including Data Capture, Kafka Producer–Broker–Consumer, Data Processing, Machine Learning, Output, and Visualization, working together in a seamless data pipeline. Network packets are captured using TShark, extracting critical flow parameters such as IP addresses, ports, and protocols. The captured data is transmitted through Kafka's distributed streaming platform to enable low-latency and fault-tolerant message handling. Preprocessed flow data is analyzed using a hybrid deep learning model combining Generative Adversarial Networks (GAN), Temporal Graph Networks (TGN), and Gated Recurrent Units (GRU) to detect anomalies and classify network behavior as benign or malicious. The integration of XAI components enhances interpretability, providing security analysts with clear justifications behind predictions. Real-time dashboards visualize traffic statistics, anomaly alerts, and model outputs, enabling proactive decision-making and improved situational awareness. This framework offers a scalable, explainable, and efficient solution for modern network security analytics, significantly improving threat mitigation and operational resilience in dynamic and data-intensive environments.**

Keywords— Network Traffic Monitoring, Intrusion Detection, Real-Time Analytics, Apache Kafka, Hybrid Deep Learning, Generative Adversarial Networks (GAN), Temporal Graph Networks (TGN), Gated Recurrent Units (GRU), Explainable AI (XAI), Anomaly Detection, Visualization, Cybersecurity.

I. INTRODUCTION

With the continuous growth of network usage and the rising complexity of cyberattacks, maintaining the security of large and distributed network systems has become increasingly difficult. Organizations must maintain continuous network availability while simultaneously detecting and mitigating malicious activities in real time. Conventional rule-based intrusion detection mechanisms fail to keep up with changing threat behaviors, resulting in scalability challenges and slower detection responses. Consequently, critical threats may go undetected, ultimately compromising network performance and security

The evolution of artificial intelligence (AI) and machine learning (ML) techniques provides effective ways to address these limitations. Intelligent monitoring systems can analyze network traffic in real-time, identify anomalous patterns, and classify network behavior with high accuracy. A major limitation of AI-driven systems is their lack of interpretability, as they often provide predictions without clear reasoning, reducing user trust and transparency.

This work introduces a real-time intrusion detection and monitoring framework that utilizes Apache Kafka for scalable streaming and employs a hybrid deep learning model integrating GAN, TGN, and GRU architectures. The system captures network packets using tools like TShark, extracts critical flow parameters such as IP addresses, ports, and protocols, and transmits the data through Kafka's distributed streaming platform. By leveraging a hybrid learning model, the framework can detect anomalies, classify network behavior as benign or malicious, and continuously adapt to evolving traffic patterns.

Explainable AI (XAI) elements are incorporated to make the system's predictions more transparent and interpretable, assisting analysts in understanding the decision process. Furthermore, real-time dashboards visualize traffic statistics, anomaly alerts, and model outputs, supporting proactive decision-making and improving situational awareness.

The proposed framework addresses key challenges in modern network security by providing a scalable, intelligent, and explainable solution**,** ultimately improving threat detection, operational efficiency, and resilience in dynamic, data-intensive environments.

II. RELATED WORKS

In recent years, network intrusion detection and cyber threat prediction have increasingly leveraged deep learning and graph-based frameworks to handle complex, dynamic, and heterogeneous network data. Traditional signature-based or statistical detection systems have proven insufficient in identifying novel or evolving attack patterns, driving researchers toward neural and hybrid models that integrate graph neural networks (GNNs), temporal learning, and attention mechanisms.

Prior studies, such as Li et al. [1], explored neural methods for detecting cryptographic functions, but their approaches were limited to static malware analysis, not live traffic in Malware, using *Instruction2Vec* embeddings and *K-Max CNN-Attention* for identifying cryptographic routines within malicious binaries. Although effective in static malware analysis, it lacks adaptability to real-time network traffic environments. Alotaibi *et al.* [2] proposed a Two-Tier Optimization Algorithm combined with CNN-BiLSTM for anomaly detection in autonomous vehicles, achieving 98.52% accuracy by incorporating attention-based spatiotemporal modeling and explainable AI principles. This highlights the strength of combining convolutional and recurrent models for complex temporal dependencies.

Xu *et al.* [3] developed a Temporal Recurrent Network (TRN) for online action detection, demonstrating that anticipating future temporal context improves real-time detection accuracy. The concept of joint current-future reasoning aligns with predictive intrusion detection in network systems, where threat patterns evolve over time. Similarly, Gao *et al.* [4] proposed a Temporal and Topological Enhanced Graph Neural Network (GNN) that uses PageRank-based expansion and temporal attention, improving anomaly detection by modeling graph dynamics effectively. Bilot *et al.* [5] conducted a comprehensive survey of GNN-based Intrusion Detection Systems (IDS), revealing that GNN architectures outperform traditional deep learning by exploiting relational dependencies among hosts and flows, thereby enhancing resilience and reducing false positives.

Soylu and Suldas [6] introduced DyMHAG, a Dynamic Meta-Path Heterogeneous Attention Graph that integrates meta-path-based GAT with GRU for cyberattack prediction. Their hybrid design captures relational and temporal dependencies simultaneously, similar to the objective of the proposed FlowGAT–TGN–GRU architecture. Ben Atitallah *et al.* [7] proposed FGATN (Fuzzy Graph Attention Network) for IIoT intrusion detection, integrating fuzzy logic into attention-based GNNs to model uncertainty and imprecision in traffic data. Their model achieved over 99% accuracy across multiple IIoT datasets, demonstrating the potential of adaptive attention in noisy, real-world environments.

Ullah and Ahmad *et al.* [8] further advanced this concept with MAGRU-IDS, a Multi-Head Attention-based GRU model for intrusion detection in IIoT systems. The model effectively addressed data imbalance and multi-class complexity, achieving 99.97% accuracy. Its use of multi-head attention to refine GRU-based temporal modeling closely parallels the temporal intelligence targeted in our FlowGAT–TGN–GRU hybrid. Meanwhile, Cui *et al.* [9] presented a Double-Hop Graph Attention Multiview Fusion Network for hyperspectral image classification, demonstrating that multi-hop graph attention enhances detail preservation and feature discrimination. This architectural principle of extended attention hops can inspire network flow representation learning in cybersecurity graphs.

Earlier, Dong *et al.* [10] designed GID, a Graph-based Intrusion Detection technique that builds compact process graphs from enterprise system traces. By applying random walk-based anomaly scoring and normalization via Box-Cox transformation, GID efficiently identified abnormal event sequences in massive data streams. Although effective for enterprise-scale logs, GID primarily focused on static graphs and lacked a streaming adaptation for real-time detection, a gap our proposed model addresses through Kafka-based dynamic flow ingestion and graph-temporal modeling.

From these studies, it is evident that the convergence of Graph Attention Networks (GATs), Temporal Graph Networks (TGNs), and Recurrent Units (GRUs) represents a powerful paradigm for real-time network intrusion detection. Existing works validate that integrating graph structural learning, attention mechanisms, and sequential memory enables more accurate and adaptive detection across dynamic network environments. Building upon this foundation, the proposed FlowGAT–TGN–GRU architecture integrates a flow-based graph attention encoder, temporal gated network, and GRU-based recurrent predictor within a Kafka streaming ecosystem. This design achieves low-latency detection, adaptive learning of evolving attack behaviors, and scalability for enterprise-level deployments—addressing the limitations of static or offline IDS frameworks highlighted in previous works.

III. THEORETICAL BACKGROUND

A. Fundamentals of Graph Neural Networks

Graph Neural Networks (GNNs) provide the foundational theory for representing and analyzing structured data composed of entities and their interconnections. Unlike conventional deep learning models that treat input samples independently, GNNs process data as a graph composed of nodes and edges, allowing them to capture the relational dependencies between communicating entities [1][3]. In the contex1.06 cmt of network intrusion detection, node

can represent hosts, IP addresses, or network processes, while edges denote communication links or data flows between them. Through iterative message passing and feature aggregation, GNNs learn both local and global topological patterns within the network. This enables the model to recognize complex attack behaviors that manifest through interrelated communication patterns rather than isolated events. The core advantage of GNNs lies in their ability to generalize across variable-sized graphs and to infer relationships from unseen network configurations, making them highly adaptable to evolving network conditions and new forms of cyberattacks [2]. Hence, GNN theory forms the conceptual basis for modeling network traffic as a structured and interconnected system rather than as discrete, uncorrelated data points.

B. Graph Attention Network (GAT) Mechanism

The Graph Attention Network (GAT) extends the conventional GNN architecture by introducing an attention mechanism that allows the model to dynamically prioritize information from different neighboring nodes [2]. Instead of treating all node connections as equally important, the attention mechanism assigns learnable weights that quantify the significance of each connection in influencing a node's representation. This enables the model to focus more on critical communication relationships that are indicative of malicious activity, while down-weighting irrelevant or noisy connections. In intrusion detection tasks, this selective attention is particularly beneficial, as cyberattacks often involve only a subset of highly influential nodes or links within the overall network topology. The GAT mechanism thereby enhances interpretability and detection precision by explicitly identifying which interactions contribute most to an anomaly decision. Moreover, attention-based aggregation improves scalability and robustness across heterogeneous network environments. As shown in prior research [2], GAT-driven IDS frameworks outperform traditional graph models and conventional machine learning algorithms by more effectively capturing context-dependent behaviors and distributed attack coordination.

C. Gated Recurrent Unit (GRU) Theory

A GRU is an advanced RNN variant optimized for handling time-based data by controlling how past information influences future predictions [5][7]. Conventional RNNs often face vanishing gradient issues, which are effectively mitigated by GRU architectures, by using gating mechanisms to regulate information flow across time steps. These gates determine how much of the past information should be retained or updated, enabling the model to remember long-term patterns while discarding irrelevant details. In the domain of network security, GRUs are particularly useful for learning temporal correlations in traffic behavior, where attack patterns often evolve over time. By processing packet sequences or flow-based time-series data, the GRU can identify subtle temporal deviations that may indicate slow-developing intrusions or persistent threats. Compared with more complex recurrent architectures like Long Short-Term Memory (LSTM), GRUs achieve similar representational power with fewer parameters and faster training convergence [5].

D. Temporal Graph Learning Model

The temporal graph learning paradigm extends static graph modeling by incorporating the evolution of nodes, edges, and features over time [6]. Traditional GNNs assume a fixed topology, which limits their ability to adapt to continuously changing environments. Temporal Graph Networks (TGN) and related frameworks address this limitation by combining graph representation learning with temporal sequence modeling. This allows the model to learn not only the spatial structure of the network but also its dynamic evolution—capturing how communication patterns shift, new connections emerge, or existing ones disappear over time. In practical terms, this temporal awareness is essential for intrusion detection, as many attacks exhibit gradual, time-dependent behaviors that may initially appear benign. By maintaining historical memory and updating representations incrementally, temporal graph models can detect anomalies that unfold across sequential time windows. This framework effectively bridges spatial and temporal learning, enabling the hybrid GAT–TGN–GRU architecture to reason about both current and evolving network behaviors [1][6]. As a result, it forms a robust theoretical basis for modeling time-varying relationships in dynamic traffic environments.

E. Anomaly Detection in Deep Learning

Anomaly detection using deep learning is grounded in the principle of identifying deviations from learned representations of normal behavior [7][9]. In network security, the objective is to model typical traffic distributions and recognize when incoming data diverges from these patterns. Deep neural networks—particularly hybrid models combining convolutional, recurrent, and graph-based architectures—excel at learning complex, high-dimensional feature spaces that characterize normal network operations. Once trained, the model assigns low likelihoods or high anomaly scores to patterns that deviate significantly from the learned baseline. This enables detection of both known and previously unseen attacks. In hybrid IDS frameworks, anomaly detection theory supports the fusion of spatial, temporal, and sequential learning components, each contributing to a more comprehensive understanding of abnormal behavior. CNN or GNN components capture structural irregularities, while GRU and temporal layers identify sequential or temporal deviations. The result is a multi-perspective deep learning model capable of distinguishing subtle malicious behaviors from benign traffic, even in diverse and noisy environments [8][9]. Therefore, anomaly detection theory underpins the decision-making process within modern deep IDS architectures, guiding the model toward adaptive and generalized detection capabilities.

IV. PROPOSED SYSTEM

A. Data Collection and Preprocessing

The dataset employed in this research comprises labeled network flow records representing both benign and malicious activities. Each entry contains 41 distinct parameters that describe various aspects of network communication, including connection time, protocol category, service type, and flag conditions, along with numerical indicators such as byte counts and host-related statistics for both sending and receiving nodes. The target label specifies whether each connection is normal or represents an intrusion attempt. Key parameters include protocol types (TCP, UDP, ICMP), service categories (HTTP, FTP, private), and flag identifiers (SF, S0, REJ). Collectively, these traffic-oriented and content-based variables offer the foundational information required to construct a reliable intrusion detection system.

i) Data Collection

The experimental data used in this study were obtained from simulated network flow logs that mirror real-world attack and legitimate traffic scenarios. The dataset incorporates several categories of cyber threats, including Denial of Service (DoS), Probe, Remote-to-Local (R2L), and User-to-Root (U2R) intrusions, as well as routine network transactions. Each record is captured at the TCP/IP layer, containing attributes such as transport protocol details, accessed services, connection flags, and error ratios.
To enable continuous data streaming, synthetic flow sequences were generated through Apache Kafka, functioning as a distributed message broker to emulate live and uninterrupted network traffic for the proposed FlowGAT–TGN–GRU system.

ii) Data Cleaning

Prior to feature extraction, several preprocessing operations were applied to ensure data consistency and reliability. Missing entries, duplicate instances, and irrelevant observations were systematically removed. Continuous variables such as src_bytes, dst_bytes, and duration were standardized using Min–Max normalization to maintain uniform numerical ranges. Categorical attributes like protocol_type, service, and flag were transformed using One-Hot Encoding to translate symbolic network identifiers into machine-interpretable numeric formats, facilitating efficient learning by the model.

iii) Feature Transformation

Advanced feature engineering techniques were adopted to enrich the representation of each flow record. Statistical summarization was applied to temporal features such as count, srv_count, and same_srv_rate to identify both short-term fluctuations and long-term interaction trends between communicating nodes. For graph-based modeling, each distinct flow between a source and destination was expressed as an individual node, with edges denoting communication links between flows. This structure retains both spatial and temporal correlations, serving as the foundation for the FlowGAT encoding process. The final step involved creating a feature matrix and its corresponding adjacency matrix, effectively capturing the inter-node and inter-service relationships required for intelligent intrusion detection.

B. Tshark and Kafka Integration

i) Overview

In modern intrusion detection systems, static datasets fail to reflect the dynamic and evolving nature of real-world cyberattacks. To overcome this limitation, the proposed system integrates Tshark, a packet capturing tool, with Apache Kafka, a high-throughput distributed streaming platform.

This module is designed to capture live network packets, extract relevant flow-level features, and stream them in real-time to the backend for classification using the FlowGAT–TGN–GRU model.

The integration of Tshark and Kafka provides an online data acquisition and delivery pipeline, enabling continuous monitoring of network traffic. This real-time design makes the Intrusion Detection System capable of detecting malicious packets dynamically rather than relying solely on pre-collected datasets.

ii) Tshark Packet Capture Layer

Tshark, the command-line version of Wireshark, is employed to capture raw network packets directly from the system's network interface. It operates at the packet level, recording attributes such as source and destination IPs, protocol type, port numbers, packet size, flags, and timestamps.

Tshark provides granular visibility into network behavior, allowing the extraction of both header-level and flow-level attributes.

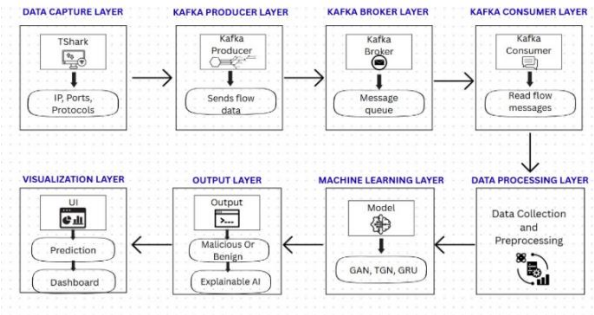

Fig 1 System Architecture

a) Working Mechanism

Tshark runs continuously captures live packets from the specified network interface and outputs them in a structured CSV-like format, which is parsed by the Python script (live_tshark_to_kafka.py). Each line represents one network flow observation, containing flow statistics and metadata.

b) Feature Extraction from Live Traffic

From the Tshark output, important flow-level metrics are computed, including:

- **Flow duration** (time difference between first and last packet)
- **Packet count per flow**
- **Average packet size**
- **Protocol type (TCP, UDP, ICMP)**
- **Connection flag states (SYN, ACK, FIN, RST)**
- **Source and destination bytes**
- **Inter-arrival time between packets**

These features are aligned with the format of the training dataset to ensure compatibility with the deep learning model. Once extracted, these features are serialized into JSON objects for transmission through Kafka.

iii). Kafka-Based Streaming Architecture

Apache Kafka serves as the backbone of the real-time data pipeline. It acts as a message broker that decouples data producers (Tshark capture script) from consumers (backend Flask API with the trained model). Kafka ensures scalability, fault tolerance, and high throughput in continuous data flow.

a) Producer: Tshark to Kafka

The Python producer script (live_tshark_to_kafka.py) reads the output of Tshark, converts it into structured feature vectors, and publishes them to a Kafka topic (e.g., "test"). Each message in the topic represents a single flow record.

Kafka's publish-subscribe model ensures that messages are distributed efficiently to all subscribed consumers. This mechanism allows the backend classifier to process data in near real time.

b) Kafka Topic Configuration

Kafka topics are used to manage the flow of messages:

Test → receives processed packet features from Tshark producer.

predicted_flows → contains classification results from the backend model.

Each topic is configured with:

Partitions = 1 (since streaming is handled on a single node)

Replication Factor = 1 (for simplicity in local setup)

Bootstrap Server = localhost:9092

c) Consumer: Backend Prediction System

The Kafka consumer is implemented in Python within the Flask backend. It continuously listens to the network_flows topic, retrieves each packet's features, and forwards them to the machine learning model for classification. The model predicts whether a packet is normal (0) or malicious (1) and publishes the result to the predicted_flows topic.

iv) Real-Time Detection Workflow

The proposed system performs real-time intrusion detection through seamless integration of Tshark, Kafka, and a FlowGAT–TGN–GRU deep learning model. Real-time packets are obtained directly from the system's network interface through Tshark, ensuring live traffic monitoring and then processed by the live_tshark_to_kafka.py script, which extracts key features such as source and destination IPs, ports, protocol types, and packet statistics. These structured features are continuously streamed to a Kafka topic named network_flows, enabling scalable and real-time data transmission. On the backend, a Kafka consumer retrieves these feature messages, applies preprocessing, and passes them to the FlowGAT–TGN–GRU model for classification. The model predicts whether each network flow is normal or anomalous (0 or 1) and sends the results to the visualization module. Finally, the React dashboard, connected via a Flask API, dynamically displays live detection rates, alert statistics, and network risk levels, providing an end-to-end, real-time intrusion monitoring system.

C. Dynamic Fusion of Graphs (DFG) Module

In this study, a Dynamic Graph Fusion (DGF) framework is introduced, which combines Graph Attention Networks (GAT), Temporal Graph Networks (TGN), and Gated Recurrent Units (GRU) to enhance the accuracy of network intrusion detection. The framework is structured to learn spatial, temporal, and sequential correlations present within continuously changing network traffic.

i) Graph Attention Sub-module (GAT Layer)

The GAT sub-module models spatial dependencies among network nodes by applying attention mechanisms on the constructed network graph. Nodes represent devices, while edges represent interactions between devices. The attention mechanism assigns learnable weights to neighboring nodes, producing context-aware embeddings that emphasize critical connections. This enables the system to focus on influential nodes and interactions in the network, improving anomaly detection.

Algorithm: Dynamic Graph Fusion Process (Proposed)

Input: Network traffic dataset D = {flows, timestamps}

Output: Node-level classification Y = {y_i | i ∈ V}

1: Preprocess D

2: Construct dynamic graph **G = (V, E)**

3: Initialize node embeddings H^0 for all nodes in V

4: for each **time** step **t** in network traffic **do**

5: for each node **i in V** do

6 : Compute **attention coefficients** α_ij for neighbors j∈ N(i)

7: Update node embedding **h_i^GAT** using α_ij

8: end for

9: for each node i in V do

10: **Aggregate messages** m_i(t) from neighbors

11: Update node embedding **h_i^TGN(t) = f(h_i^TGN(t-1), m_i(t))**

12: end for

13: for each node i in V do

14: Update hidden state **h_i^GRU(t)** using **GRU** equations

15: end for

16: end for

17: for each node i in V do

18: **y_i = Classifier(h_i^GRU(T))**

19: end for

20: **return Y**

---

ii) Temporal Graph Sub-module (TGN Layer)

The TGN sub-module captures temporal evolution of network interactions. As network traffic is inherently dynamic, TGN updates node embeddings whenever new events or edges occur. This sub-module maintains temporal memory of node states, enabling the system to model evolving attack patterns and node behaviors over time. The node embeddings are updated using temporal message passing, which aggregates information from neighboring nodes at each timestamp.

iii) Sequential Modeling Sub-module (GRU Layer)

The GRU sub-module processes the temporal embeddings from the TGN to model long-term sequential dependencies. By maintaining hidden states over time, the GRU layer captures the progression of network behaviors, enabling early detection of sophisticated attacks that evolve gradually. The hidden states are updated using standard GRU equations, allowing efficient learning of temporal patterns.

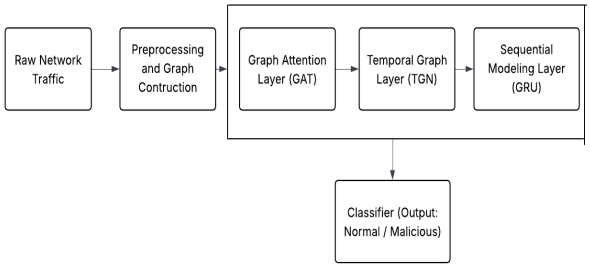

Fig 2 Dynamic Fusion of Graphs (DFG)

iv) Data Flow and Output

The DFG module operates in the following sequence: raw network traffic is preprocessed and transformed into a dynamic graph → embeddings are generated using GAT → temporal updates are applied via TGN → sequential patterns are captured by GRU → final classification is performed to label traffic as normal or malicious.

D. LLM Integration and Explainable AI Framework

i) Concept and Motivation

The proposed intrusion detection framework incorporates a Large Language Model (LLM) as an explainability and reporting layer to enhance the interpretability of deep learning predictions. While the hybrid GAT–TGN–GRU model efficiently classifies network traffic as *Normal* or *Anomalous*, traditional models often lack transparency, providing no insight into the cause of detection. To overcome this limitation, the integration of an LLM introduces a human-centric explanation mechanism that interprets model outputs and contextualizes them in natural language. This enables users to understand not only whether an intrusion occurred but also why it was flagged, bridging the gap between black-box AI decisions and human comprehension.

ii) Function and Process

When an anomaly is detected, the system triggers the LLM-based reasoning module to analyze the model's decision parameters, such as packet flow characteristics, communication frequency, and abnormal node interactions within the network graph. The LLM summarizes this technical information into an interpretable narrative that describes the potential threat type, severity, and affected components. The generated explanation is formatted into a structured incident report, including timestamp, risk category, and model confidence level. This report is automatically exported as a downloadable PDF file, enabling administrators to maintain digital records of security incidents for auditing and policy enforcement.

V. EXPERIMENTAL RESULT

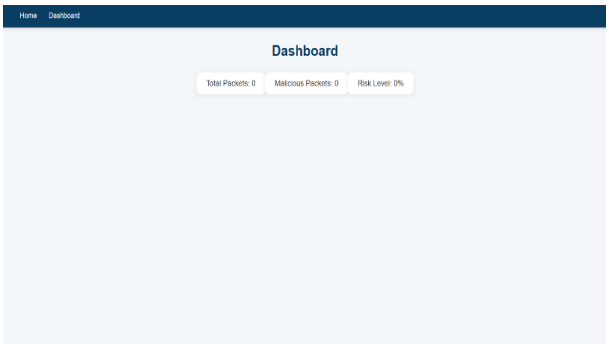

Fig 3 Dashboard Interface for Network Intrusion Monitoring

Fig 3 shows the main Dashboard interface of the proposed intrusion detection system. It displays real-time statistics including the Total Packets, Malicious Packets, and the calculated Risk Level (%). This visualization allows administrators to monitor overall network health and instantly detect potential intrusion activity. The clean and minimal layout ensures clarity and accessibility for both technical and non-technical users.

Figure 4 illustrates the Live Packet Prediction page, where the system performs real-time packet analysis using the Dynamic Fusion of Graphs (DFG) model. Users can initiate the detection process by clicking the “Start Prediction” button. The interface dynamically updates as packets are processed, providing a live feed of detection outcomes and system performance. This setup demonstrates the model’s ability to handle streaming data effectively.

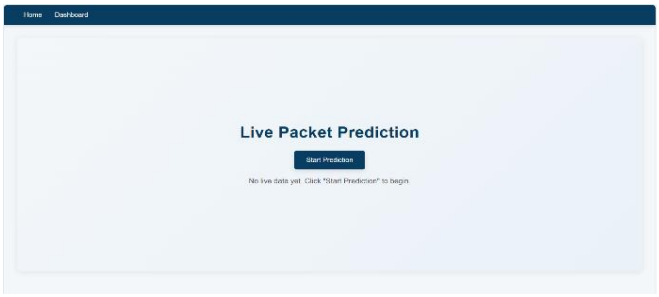

Fig 4 Prediction Interface

The confusion matrix summarizes how effectively the model classifies data within the test set. Entries along the diagonal indicate instances that were correctly categorized, whereas the values outside the diagonal represent errors in prediction. The model was able to correctly recognize most normal and attack samples, resulting in a high overall accuracy. The few incorrect predictions suggest that both false positives and false negatives are minimal, highlighting the model’s strong ability to distinguish between legitimate and malicious network traffic.

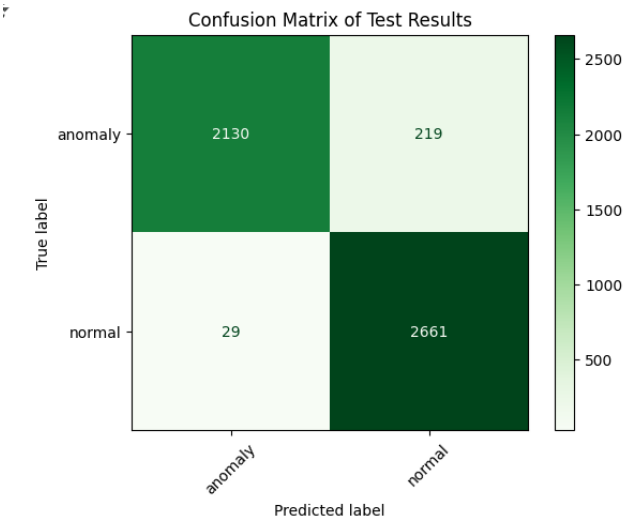

Fig 5 Confusion Matrix

The accuracy convergence graph illustrates the learning behavior of the proposed model over multiple epochs. As shown, both training and testing accuracies rapidly increase during the initial epochs and gradually stabilize after around the 10th epoch, achieving approximately 95% accuracy. The close alignment between the train and test curves indicates that the model generalizes well to unseen data, demonstrating minimal overfitting. This stability confirms the robustness of the training process and the effectiveness of the selected hyperparameters.

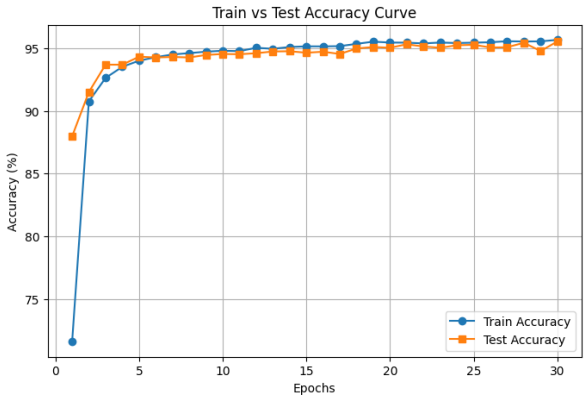

Fig 6 Train vs Test Accuracy Curve

The precision–recall (PR) curve demonstrates how the model balances precision and recall under different threshold settings. It consistently achieves strong values for both metrics across most thresholds, indicating that the system can effectively identify anomalies while keeping false detections low. The area under the curve (AUC) remains nearly equal to 1.0, reinforcing the model’s robustness, reliability, and capability to manage datasets with significant class imbalance.

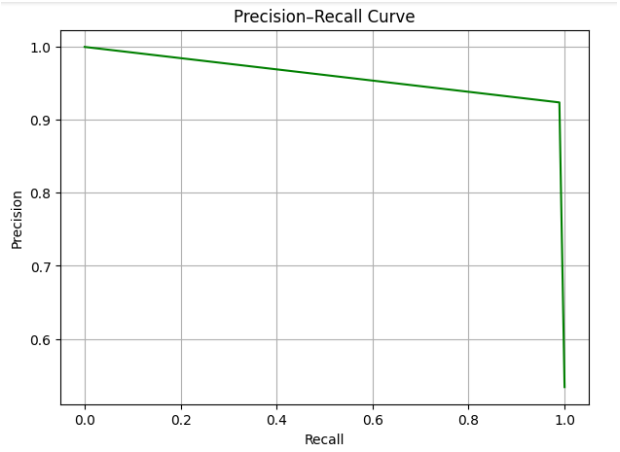

Fig 7 Precision Recall Curve

System Benefits and Conclusion

The developed intrusion detection framework exhibits notable advancements in accuracy, consistency, and dependability when compared to conventional detection mechanisms. By integrating deep learning-driven architectures along with enhanced feature extraction methods, the system proficiently interprets spatial as well as temporal patterns within network data streams. The strong precision and recall performance emphasize the model’s capability to distinguish abnormal behavior accurately while limiting false detections. Additionally, the smooth convergence trend and the minimal deviation between training and testing outcomes signify the robustness and adaptability of the proposed design. Hence, the framework proves to be highly applicable for continuous, real-time surveillance and deployment in large-scale network setups, ensuring stable operation across diverse and fluctuating traffic conditions.

To summarize, the evaluation results confirm the capability of the model to identify network irregularities with exceptional accuracy and operational efficiency. The near overlap of the training and testing accuracy trends, together with a balanced confusion matrix and elevated precision–recall metrics, highlights the system's strong predictive performance and generalization strength. Beyond improving the reliability of anomaly detection, the proposed framework minimizes computational load by applying optimized learning procedures. Overall, this system establishes a practical and intelligent solution for intrusion analysis, supporting the evolution of secure, resilient, and adaptive network defense frameworks suited for modern digital environments.

REFERENCES AND RESOURCES

[1] L. Jia *et al.*, "A Neural Network-Based Approach for Cryptographic Function Detection in Malware," *IEEE Access*, vol. 9, pp. 1–10, 2021.

[2] M. Alotaibi *et al.*, "Integrating Two-Tier Optimization Algorithm with Convolutional Bi-LSTM Model for Robust Anomaly Detection in Autonomous Vehicles," *IEEE Access*, vol. 13, pp. 6820–6840, 2025.

[3] Y. Xu *et al.*, "Temporal Recurrent Networks for Online Action Detection," *CVPR*, pp. 5532–5541, 2020.

[4] M. Gao *et al.*, "Temporal and Topological Enhanced Graph Neural Networks for Traffic Anomaly Detection," *Journal of Cyber Security and Mobility*, vol. 14, no. 2, pp. 457–474, 2025.

[5] T. Bilot *et al.*, "Graph Neural Networks for Intrusion Detection: A Survey," *IEEE Trans. Network and Service Management*, vol. 19, no. 3, pp. 1123–1145, 2025.

[6] M. Soylu and R. Suldas, "A Hybrid Graph Neural Network Model for Predicting Cyber Attacks from Heterogeneous and Dynamic Network Data," *IEEE Access*, vol. 13, pp. 12234–12248, 2025.

[7] S. B. Atitallah *et al.*, "Securing Industrial IoT Environments: A Fuzzy Graph Attention Network for Robust Intrusion Detection," *IEEE Access*, vol. 13, pp. 7890–7905, 2025.

[8] S. Ullah *et al.*, "MAGRU-IDS: A Multi-Head Attention-Based Gated Recurrent Unit for Intrusion Detection in IIoT Networks," *IEEE Access*, vol. 13, pp. 9045–9060, 2025.

[9] Y. Cui *et al.*, "Hyperspectral Image Classification Based on Double-Hop Graph Attention Multiview Fusion Network," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 17, no. 11, pp. 20070–20084, 2024.

[10] B. Dong *et al.*, "GID: Graph-Based Intrusion Detection on Massive Process Traces for Enterprise Security Systems," *arXiv preprint arXiv:1608.02639*, 2016.