

DOI: https://doi.org/10.48009/3_iis_2026_148

A correlation-graph-driven ensemble intrusion detection system for IoT networks

Kazi Abdullah Al Arafat, *Atish Dipankar University of Science and Technology*,
cse2010004@adust.edu.bd

Md. Iftekhar Hossain, *Dhaka University of Engineering & Technology*, iftekhar121@gmail.com

Syed Sajal, *Utah Valley University*, Sayeed.Sajal@uvu.edu

Imtiaz Parvez, *Utah Valley University*, Imtiaz.Parvez@uvu.edu

Abstract

Cyber-physical systems (CPS) and the Internet of Things (IoT) are rapidly converging into unified platforms integrating physical and digital infrastructures. This increased interconnectivity creates extensive opportunities for cyber attackers to exploit vulnerabilities, making intrusion detection systems (IDS) a vital cybersecurity component. However, emerging CPS/IoT technologies often face imbalanced and incomplete datasets, complicating effective IDS training. To address these challenges, this work proposes an IDS combining Graph Neural Networks (GNNs), Convolutional Neural Networks (CNNs), and Long Short-Term Memory (LSTM) networks through ensemble learning to detect and analyze anomalies. We evaluate the system using cyberattacks on IoT traffic from the RT-IoT2022 dataset, applying SMOTE, SMOTEENN, and undersampling to handle class imbalance, and building a KNN-based correlation graph to capture sample inter-relationships. Three models—GNN-LSTM, CNN-LSTM, and GCN-LSTM—achieve macro F1-scores of approximately 0.85, 0.97, and 0.85, respectively. Their ensemble achieves a superior macro F1-score of 0.9946, demonstrating outstanding intrusion-detection performance. These results show that graph-based and ensemble methods offer effective solutions for real-time IoT security.

Keywords: Ensemble Learning, Intrusion Detection System, IoT, GNN, LSTM, CNN, Cyber Physical Systems.

Introduction

The Internet of Things (IoT) is a fundamental cyber-physical system (CPS) sub-domain that has evolved into a data-driven system. The technological progress of IoT has transformed data into its fundamental operational component. IoT enables advanced connection capabilities through its combination of sensors, actuators, firmware, and connectivity systems for data collection and exchange. The majority of IoT devices face cyber-attacks because they lack proper security measures despite their widespread adoption (Rambus Inc., 2019). The interconnected nature of IoT systems makes them vulnerable to advanced cyber-attacks, which include Distributed Denial of Service (DDoS), port scanning, and brute-force attempts. Traditional anomaly detection methods, which depend on signature-based or statistical approaches, fail to handle the dynamic and heterogeneous characteristics of IoT traffic. Machine learning (ML) and deep learning (DL)

have proven themselves to be effective because they use data-driven insights to detect anomalies in real-time.

The defense strategy of CPS depends on Intrusion Detection Systems (IDSs), which monitor network traffic and system data for malicious activities to generate alerts. The identification of specific system abnormalities at their exact time and location enables system protection through appropriate responses, which leads to the implementation of intrusion prevention systems. The intrusion prevention system operates together with the intrusion detection system to stop attackers from causing damage to the system. The system that uses machine learning-based IDS can identify system irregularities with high precision (Aldweesh et al., 2020; Gueriani et al., 2024; Pinto et al., 2023). The introduction of new architectural structures leads to model complexity because researchers cannot predict how these systems will behave (Törngren & Grogan, 2018). The implementation of individual IDS systems requires observation of their behavior within this complex system, but the lack of sufficient data hinders model training. The available datasets contain an imbalance of attack data because normal data exceeds attack data in quantity.

With the Internet of Things (IoT) networks being able to generate a huge amount of data from various sources such as, sensors and smart devices, smart home appliances, and industrial monitoring systems which can communicate with each other using different protocols, the data generated can form different structures or distributions. So, in terms of anomaly detection not only the model has to learn the normal behavior and recognize it, but also it has to recognize and detect the unusual or unknown behavior, which can be considered as attacks (e.g. NMAP scans, ARP poisoning, etc.). Many of these anomalies can pose serious security threats to the network. Moreover, as mentioned earlier the problem of class imbalance is more severe when dealing with network traffic data. Since the normal traffic far exceeds the traffic which can be considered as an attack (Shahrir et al., 2020), an efficient anomaly detection system has to be able to deal with high-dimensional data that follow imbalanced distributions. The system has to learn from the normal traffic and study the normal traffic relationships in order to detect anomalies.

In recent years, there has been growing interest in developing hybrid models that use a number of machine learning techniques to incorporate spatial, temporal and structural information. Relational information within the network can be captured by Graph Neural Networks (GNNs). Information encoded in images of network traffic can be processed by using Convolutional Neural Networks (CNNs) for recognizing local patterns within images. In addition, information in sequences can be modeled by using models such as Long Short-Term Memory (LSTM) networks. Ensemble methods can then be used to combine several of these models in order to create more accurate, robust and generally better models for anomaly detection in IoT networks. Anomaly detection in IoT networks has been extensively studied, with recent works focusing on ML and DL techniques to address evolving threats. Bhattacharya et al. (2020) propose a deep learning model combining CNNs and LSTMs for IoT intrusion detection, achieving high accuracy on the NSL-KDD dataset but struggling with imbalanced classes. Ferrag et al. (2020) surveyed DL techniques for IoT security, highlighting the effectiveness of recurrent neural networks (RNNs) in sequential data analysis. HaddadPajouh et al. (2018) used LSTM networks to detect malware in IoT devices, reporting improved detection rates on IoT-specific datasets. Parvez and Sundararajan (2025) applied an attention-based graph neural network with node embeddings to detect false data injection attacks on PV-integrated microgrids.

Koroniotis et al. [2019] have recently introduced a new dataset for Intrusion Detection for Internet of Things (IoT) called Bot-IoT which was used to test several ML classifiers, but the detection of the minority class has proven to be very challenging for the several classifiers that have been tested. Another approach that was recently introduced for Network Intrusion Detection is based on Graph Neural Networks (GNNs), where the approach that was introduced by Li et al. [2021] leverages the graph structure of a network to learn from the relationships between the nodes in a network. In addition, Moustafa et al. [2019] have

introduced the UNSW-NB15 dataset for Network Intrusion Detection using ensemble methods. In a different domain, i.e., Sentiment Analysis on Twitter data using ML models, Al Arafat et al. [2025] have recently demonstrated the capability of ML models to handle high-dimensional data. In addition, Anomaly detection in time series data using ML models has been analyzed by Nguyen et al. [2021], where the authors have employed GNNs with attention mechanism for Anomaly detection in large-size time series datasets generated from IoT sensors and monitoring systems. In addition to the referenced research efforts that employed Fed-Learning with CNNs for Intrusion detection in IoT using Federated Learning, Otoum et al. [2022] have recently introduced a system that utilizes fed-learning to enable the distributed training of a CNN-based model for Network Intrusion Detection and Intrusion response in IoT environments, but the system has high computational overhead. Using SMOTE for handling class imbalance in datasets of IoT sensors, Sajjad et al. [2020] improved the performance of classifiers for minority classes in several datasets that were generated by using several types of IoT sensors. Diabetes data using ML models has been analyzed by Al Arafat et al. [2025] in order to develop predictive models that can be used for analysis of healthcare IoT systems. In addition to the referenced research efforts, an unsupervised Anomaly detection approach that uses deep autoencoders for representing IoT data using compressed representations of the data has been performed by Shone et al. [2018] for Anomaly detection in IoT networks in an unsupervised manner.

Tavallae et al. (2010) analyzed strengths and weaknesses of the well-known KDD Cup 99 data set for modern network security using traditional ML approaches. Ullah et al. (2021) proposed a hybrid CNN-LSTM model for the intrusion detection in IoT using the CICIDS2017 data set to achieve high precision. Al Arafat et al. (2024) analyzed performance of various ML models for the sentiment analysis of Twitter data using ML models, and analyzed various feature extraction techniques that can be used for feature engineering of large-scale IoT data. Furthermore, various studies on using edge computing for IoT have been proposed recently. For example, Wang et al. (2022) proposed a GCN-based model for analyzing IoT networks and focused on the edge weight optimization. Also, studies on using RNN for network intrusion detection have been proposed recently. For example, Yin et al. (2017) proposed an RNN-based model for the intrusion detection in network and analyzed their results using temporal dependencies. Zhou et al. (2023) proposed a multi-modal ensemble model for the various types of IoT security using GNNs and CNNs and achieved the state-of-the-art results. Also, Al Arafat et al. (2024) analyzed performance of various ML models for the Li-ion battery temperature forecasting under the dynamic conditions using the historical time-series data from the various types of IoT sensors and showcased the robustness of the ML models for the time-series modeling of large-scale IoT sensor data.

Our work builds on these foundations by integrating GNNs, LSTMs, CNNs, and ensemble learning, with a focus on robust preprocessing and graph construction to enhance detection accuracy. The contributions of our work are as follows:

- The proposed IDS blends GNNs, LSTMs, and CNNs through ensemble learning to capture spatial, temporal, and structural patterns in IoT traffic.
- To address class imbalance, we apply advanced preprocessing methods such as SMOTE, SMOTEENN, and under sampling. We also construct a KNN-based correlation graph to better model relationships between samples.
- Our evaluation shows clear performance gains. The GNN-LSTM and GCN-LSTM models each achieve macro F1 scores near 0.85, while the CNN-LSTM model reaches 0.97. The ensemble model delivers an exceptional macro F1-score of 0.9946, surpassing all individual models and setting a strong benchmark for IoT intrusion detection.
- These results highlight how graph-driven features and ensemble learning significantly strengthen real-time intrusion detection for IoT systems.

In the rest of this paper, Section II describes the system model. Section III explains how to apply the proposed method. Section IV reports simulation results. Section V concludes this paper and outlines future work.

System Model

Inputs

For each time step $t = \{1, \dots, T\}$, the inputs to the detection models are:

$$\text{Input} = \begin{cases} X_t \in \mathbb{R}^{C_{in} \times H \times W}, \\ G_t = (V, E_t), \\ A_t \in \{0, 1\}^{n \times n}, \\ F_t \in \mathbb{R}^{n \times d_0} \end{cases}$$

where X_t is the feature input with C_{in} channels, height H , width W ; G_t is the graph at time t with node set V and edge set E_t ; A_t is the adjacency matrix; and F_t is the node feature matrix with d_0 -dimensional features per node.

CNN Branch

$$Y_{t,k}^{(1)} = b_k^{(1)} + \sum_{c=1}^{C_{in}} W_{k,c}^{(1)} * X_{t,c}, X_{t,k}^{(1)} = \sigma(Y_{t,k}^{(1)})$$

$$x_t^{cnn} = \left(\text{Pool} \left(\vec{X_t^{(L_c)}} \right) \right) \in \mathbb{R}^{d_{cnn}}$$

where $W_{k,c}^{(1)}$ are convolutional filters, $*$ denotes convolution, $\sigma(\cdot)$ is an activation function (e.g., ReLU), and $\text{Pool}(\cdot)$ is a pooling operation. The resulting flattened vector x_t^{cnn} is the CNN representation at time t .

GCN Branch

$$\tilde{A}_t = D_t^{-\frac{1}{2}} (A_t + I) D_t^{-\frac{1}{2}}, D_t = \text{diag}(A_t \mathbf{1} + \mathbf{1})$$

$$H_t^{(0)} = F_t, H_t^{(l+1)} = \sigma(\tilde{A}_t H_t^{(l)} W^{(l)}), l = 0, \dots, L_g - 1$$

$$r_t^{gcn} = \text{READOUT}(\{H_{t,v}^{(L_g)}\}_{v \in V}) \in \mathbb{R}^{d_{gcn}}$$

where $\tilde{A}_t$ is the normalized adjacency matrix, $W^{(l)}$ are trainable GCN weight matrices, and $\sigma(\cdot)$ is a nonlinearity. $\text{READOUT}(\cdot)$ aggregates node embeddings (e.g., mean, sum, or max pooling) to produce the graph-level representation r_t^{gcn} .

GNN Branch

$$m_{t,v}^{(l)} = \sum_{u \in N_t(v)} \phi_m(h_{t,u}^{(l)}, h_{t,v}^{(l)}, e_{t,uv}^{(l)})$$

$$h_{t,v}^{(l+1)} = \phi_h(h_{t,v}^{(l)}, m_{t,v}^{(l)}), r_t^{gnn} = \rho(\{h_{t,v}^{(L_m)}\}_{v \in V}) \in \mathbb{R}^{d_{gnn}}$$

where ϕ_m is the message function, ϕ_h is the node update function, and $\rho(\cdot)$ is a graph-level pooling/readout operator. This general formulation covers models such as GraphSAGE, GAT, and MPNNs.

LSTM Cell

$$\begin{aligned}
 i_t &= \sigma(W_i x_t + U_i h_{t-1} + b_i) && \text{input gate} \\
 f_t &= \sigma(W_f x_t + U_f h_{t-1} + b_f) && \text{forget gate} \\
 g_t &= \tanh(W_g x_t + U_g h_{t-1} + b_g) && \text{candidate state} \\
 o_t &= \sigma(W_o x_t + U_o h_{t-1} + b_o) && \text{output gate} \\
 c_t &= f_t \odot c_{t-1} + i_t \odot g_t && \text{cell state} \\
 h_t &= o_t \odot \tanh(c_t) && \text{hidden state}
 \end{aligned}$$

where x_t is the input vector at time t , h_t is the hidden state, c_t is the cell state, and $\odot$ denotes elementwise multiplication.

Branch LSTMs

$$\begin{aligned}
 (h_t^{\text{cnn}}, c_t^{\text{cnn}}) &= \text{LSTM}_{\text{cnn}}(x_t^{\text{cnn}}, h_{t-1}^{\text{cnn}}, c_{t-1}^{\text{cnn}}) \\
 (h_t^{\text{gcn}}, c_t^{\text{gcn}}) &= \text{LSTM}_{\text{gcn}}(r_t^{\text{gcn}}, h_{t-1}^{\text{gcn}}, c_{t-1}^{\text{gcn}}) \\
 (h_t^{\text{gnn}}, c_t^{\text{gnn}}) &= \text{LSTM}_{\text{gnn}}(r_t^{\text{gnn}}, h_{t-1}^{\text{gnn}}, c_{t-1}^{\text{gnn}})
 \end{aligned}$$

Each branch uses its own LSTM to capture temporal dependencies from CNN, GCN, and GNN features.

Fusion and Final LSTM

$$\begin{aligned}
 z_t &= [h_t^{\text{cnn}}; h_t^{\text{gcn}}; h_t^{\text{gnn}}] \in \mathbb{R}^{d_z} \\
 (h_t^*, c_t^*) &= \text{LSTM}_{\text{final}}(z_t, h_{t-1}^*, c_{t-1}^*)
 \end{aligned}$$

The branch-level hidden states are concatenated into z_t and passed through a final LSTM for joint temporal modeling.

Output (Attack Detection)

$$\hat{y}_t = \text{softmax}(W_o h_t^* + b_o)$$

Methodology

As referred to Fig. 1(a), the step-by-step process carried out in this research can be generalized into three main parts: data collection and pre-processing, machine learning-based data analysis, and evaluation and validation of results.

Dataset Processing

To prepare the data for training and evaluation, the following steps were taken

- a. Data Collection and Pre-processing: The RT-IoT2022 dataset [27], a comprehensive real-time IoT network traffic dataset, is utilized in this study. Sourced from the University of New South Wales (UNSW) Canberra Cyber, it contains over 120,000 instances, capturing normal traffic and 12 attack types, including DDOS_Slowloris, NMAP_OS_DETECTION, NMAP_FIN_SCAN, Thing_Speak, Wipro_bulb, Metasploit_Brute_Force_SSH, ARP_poisoning, DOS_SYN_Hping, NMAP_UDP_SCAN, and various NMAP scans. The dataset is sampled to approximately 24,804 instances, targeting -2,067 samples per class to ensure balanced representation.

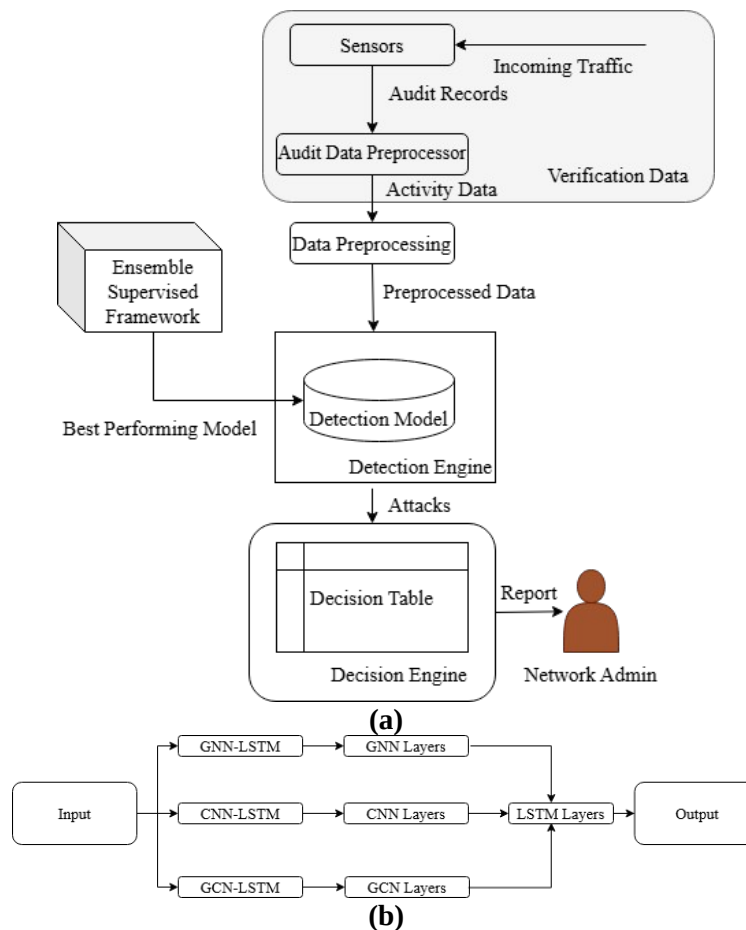


Figure 1. (a) Overall ensemble learning architecture. (b) Detailed stacking ensemble workflow.

b. Categorical Encoding: The proto and service columns are encoded using LabelEncoder to convert categorical variables into numerical values. The Attack_type column is similarly encoded, mapping attack labels to integers.

c. Feature Engineering: Three new features are derived:

- (1) flow_duration_norm: Normalized flow duration using mean and standard deviation;
- (2) packet_size_ratio: Ratio of forward to backward packets, with a small constant 10^{-6} to avoid division by zero; and
- (3) byte_rate: Forward packet payload total divided by flow duration.

d. Class Imbalance Handling: Small Class Augmentation: Classes with fewer than 3 samples are augmented by duplicating existing samples to ensure at least 3 instances per class.

SMOTE: Synthetic Minority Oversampling Technique (SMOTE) is applied with $k=5$ neighbors, targeting 3,000 samples for minority classes (e.g., DDOS_Slowloris, NMAP_OS_DETECTION).

SMOTEENN: SMOTE combined with Edited Nearest Neighbors (ENN) cleans noisy samples, improving data quality.

Random Undersampling: Balances the dataset further by reducing overrepresented classes.

e. Graph Construction: A correlation graph is constructed using a k-Nearest Neighbors (KNN) approach with $k=10$ and cosine similarity. Edges are weighted based on similarity, with a boost factor of 1.5 for minority classes to enhance their connectivity. The graph includes self-loops and is constrained to a maximum of 80,000 edges with a minimum edge weight of 0.55, resulting in an average degree of 15-25 and no isolated nodes

f. Data Splitting: Dataset was divided into 70% being training sets and 30% being test sets, allowing machine learning to have enough data to analyze patterns before being tested.

After completing all of the above steps, the resulting data set provides a strong base for machine learning models to be trained and evaluated for anomaly detection in a cloud environment.

Models and Architectures: The proposed IDS (Fig-1) employ multiple neural network models to analyze IoT traffic patterns, capturing spatial, temporal, and structural characteristics. The models are as follows:

GNN-LSTM: The model contains three GraphConv layers which use edge weights from the correlation graph to process node features through 384, 192 and 64 units respectively. The model includes batch normalization after each layer to maintain training stability and ReLU activation for introducing non-linear relationships. The model uses edge similarities to combine neighbor information from adjacent nodes for creating stable node embeddings. The model uses two LSTM layers with 192 units each to extract temporal relationships from graph embeddings. The input data receives a time dimension addition through reshaping to create a sequence length of 1 before the model extracts the final hidden state for maintaining sequential data patterns. The model uses a fully connected layer to transform LSTM output into 12 class probability values. The model includes dropout with a value of 0.3 after each GraphConv layer to prevent overfitting and gradient clipping with a max norm of 3.5 to stabilize optimization through backpropagation by controlling gradient values.

CNN-LSTM: The model contains two 1D convolutional layers with 64 units, kernel size 3, and padding 1 which extract spatial features from input data. The model includes ReLU activation functions after each convolutional layer followed by a max-pooling layer with kernel size 2 to extract dominant patterns while reducing dimensionality. The convolutional layers function as feature extractors which detect local patterns without requiring graph-based connections. The model uses two LSTM layers with 64 units each to analyze convolutional output data for temporal pattern detection. The model transposes its input data to match batch and sequence and feature dimensions before using the last hidden state for classification purposes. The model generates class probability output through its final fully connected layer. The model applies dropout with a rate of 0.25 before the output layer to minimize overfitting while the convolutional layers maintain their own batch normalization effects.

GCN-LSTM: The GCNConv layer (384 units) with a skip connection (linear projection of input features) improves feature aggregation through the combination of original and transformed features. The model contains two GraphConv layers with 192 and 64 units which apply batch normalization and ReLU activation to enhance node embeddings based on graph structure. The model uses two LSTM layers with 192 units each to process graph embeddings just like GNN-LSTM does for temporal pattern detection in node representations. The model generates class probability output through its fully connected output layer. The model applies dropout with 0.3 rate and gradient clipping with 3.5 max norm to prevent overfitting while the skip connection enhances training stability and gradient flow.

Model Building: The models use PyTorch and PyTorch Geometric to create graph-based operations that work together. The input dimension equals 47 because the model processes 47 features from the preprocessed data set which includes both ,original and engineered features. The output dimension equals 12 because the model needs to predict 12 different attack classes. The model uses class weights that equal the inverse of class frequency distribution to enhance minority class detection (DDOS_Slowloris, NMAP_OS_DETECTION, NMAP_FIN_SCAN, Thing_Speak, Wipro_bulb, Metasploit_Brute_Force_SSH, and ARP_poisoning) by 30 times and NMAP_UDP_SCAN and DOS_SYN_Hping by 15 times, and NMAP_UDP_SCAN and DOS_SYN_Hping by 8 times. The weights of the network are normalized to sum to 1.0, then converted to PyTorch tensors for use in a loss function. I

set the random seed for the runs to 42, so that the runs are reproducible. Because we have GPUs available for use, the model is run on a CUDA device. If no CUDA devices are available, the model runs on the CPU for performance reasons.

Training and Evaluation

- (a) **Training:** We tried to optimize the hyperparameters of the model in order to get the best possible accuracy without overfitting and the following values have been chosen for the training.
- **Loss Function:** The Focal Loss function handles class imbalance through gamma values which reach 4.0 for GNN-LSTM and 3.5 for GCN-LSTM and 2.5 for CNN-LSTM. The loss function uses class weights to focus on difficult-to-classify samples from minority classes.
 - **Optimizer:** The Adam optimizer operates with 0.0012 learning rates for GNN-LSTM and CNN-LSTM and 0.0007 learning rates for GCN-LSTM while weight decay reaches 5 times 10^{-6} to stop weight growth.
 - **Scheduler:** The ReduceLROnPlateau scheduler decreases learning rates by 0.4 when validation F1-score performance fails to improve during 20 consecutive epochs.
 - **Training Setup:** The models receive training for up to 450 epochs while processing 512 samples per batch. The training process stops when the model fails to improve its F1-score by 0.00001 for 20 consecutive epochs but must train for at least 50 epochs to achieve sufficient learning. The model state with the highest validation F1-score becomes the saved best model.
 - **Data Loading:** The entire graph structure including nodes and edges with their corresponding weights gets processed as one batch for GNN-LSTM and GCN-LSTM because of their graph structure dependencies through PyTorch Geometric sparse matrix operations. The CNN-LSTM model uses a DataLoader to perform batch-wise data shuffling with 512 samples per batch which improves generalization performance.
 - **Monitoring:** The system tracks training performance through loss values and macro F1-score and gradient norm measurements at each epoch. The system produces alerts when train-validation F1-score differences reach 0.03 or higher because this indicates possible overfitting problems and it checks gradient norms for stable optimization.
- (b) **Evaluation:** The performance evaluation for model the proposed models relied on the following assessment methods.
- **Metrics:** The macro F1-score serves as the main evaluation metric because it demonstrates how well the model performs on all classes despite their different distribution levels. The classification report shows precision, recall values and F1-scores for each class while using `zero_division=0` to handle undefined metrics.
 - **Ensemble:** The ensemble prediction combines GNN-LSTM, CNN-LSTM, and GCN-LSTM outputs through weighted averaging based on their validation F1-score performance. The ensemble output selects the class with the highest weighted sum value, which gets evaluated through the same performance metrics.
 - **Debugging:** The prediction count for each class in GNN-LSTM and GCN-LSTM gets recorded to verify complete class prediction because these graph-based models might produce biased results from limited connections and uneven edge weights.

Simulation and Results

The simulation results show that GNN-LSTM, CNN-LSTM, and LSTM-GNN models achieve similar high performance in supervised anomaly detection. The evaluation process assesses each model through

its accuracy, precision, recall, and computational performance to provide complete understanding of its operational capabilities and operational limitations.

Model Performance Metrics

GNN-LSTM Model

As referred to Table-1, the GNN-LSTM model achieved a Macro F1-score of 0.8425, reflecting moderate performance on the RT-IoT2022 dataset. It excelled in detecting certain attack types, such as NMAP_XMAS_TREE_SCAN (F1 = 1.0000) and DOS_SYN_Hping (F1 = 0.9975) but struggled significantly with MQTT_Publish (F1 = 0.0000), likely due to class imbalance or limited feature discriminability in graph-structured data.

Table 1. GNN-LSTM Performance Analysis

Class	Precision	Recall	F1- Score	Support
ARP_poisoning	0.8312	0.8040	0.8174	796
DDOS_Slowloris	0.9278	0.9849	0.9555	796
DOS_SYN_Hping	1.0000	0.9950	0.9975	796
MQTT_Publish	0.0000	0.0000	0.0000	796
Metasploit_Brute_Force_SSH	0.8853	0.9309	0.9075	796
NMAP_FIN_SCAN	0.9376	0.9824	0.9595	796
NMAP_OS_DETECTION	0.9900	1.0000	0.9950	796
NMAP_TCP_scan	0.5686	0.9887	0.7220	796
NMAP_UDP_SCAN	0.7977	0.9510	0.8676	796
NMAP_XMAS_TREE_SCAN	1.0000	1.0000	1.0000	796
Thing_Speak	0.9817	0.8792	0.9277	796
Wipro_bulb	0.9420	0.9799	0.9606	796
accuracy			0.8747	9551
macro avg	0.8218	0.8747	0.8425	9551
weighted avg	0.8218	0.8747	0.8425	9551

As referred to Fig-2, the GNN-LSTM confusion matrix confirms significant misclassifications for MQTT_Publish, with near-zero true positives, aligning with its F1-score of 0.0000. Strong diagonal dominance is observed for NMAP_XMAS_TREE_SCAN and DOS_SYN_Hping, reflecting high true positive rates.

CNN-LSTM Performance

As referred to Table-2, the CNN-LSTM model significantly outperformed GNN-LSTM, achieving a Macro F1-score of 0.9904 and a Macro AUC of 0.9998. It demonstrated exceptional performance across most classes, with perfect F1-scores for DOS_SYN_Hping (1.0000), near-perfect scores for NMAP_FIN_SCAN (0.9975) and NMAP_OS_DETECTION (0.9975), highlighting its strength in temporal feature extraction.

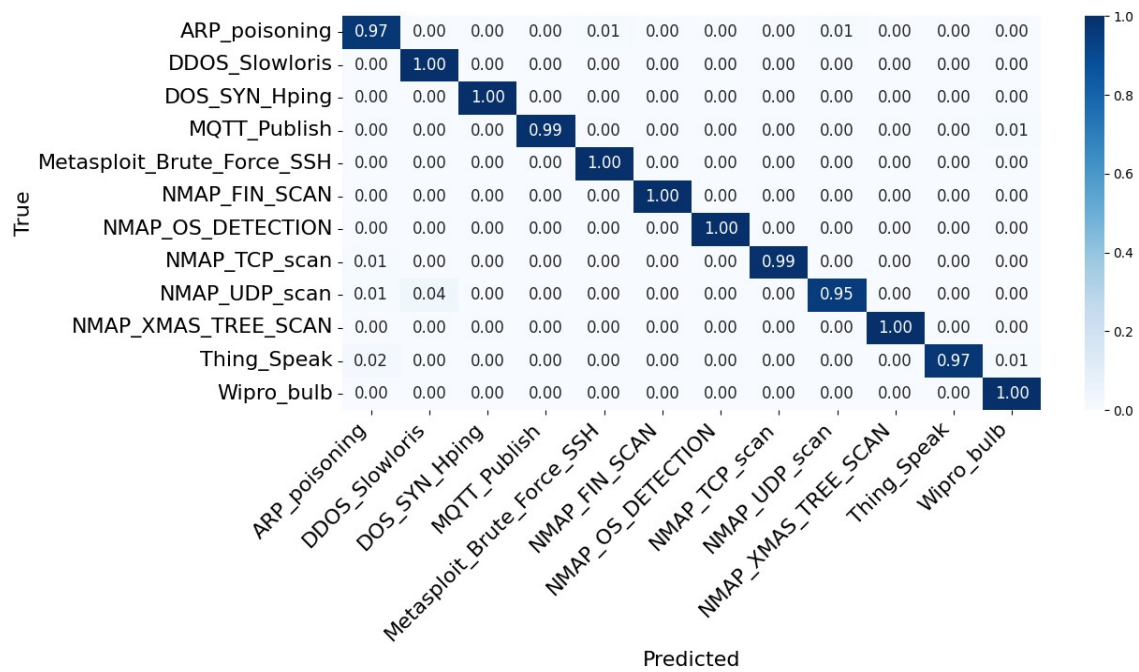


Figure 2. Confusion Matrix of GNN-LSTM model.

Table 2. CNN-LSTM Performance Analysis

Class	Precision	Recall	F1- Score	Support
ARP_poisoning	0.9832	0.9573	0.9701	796
DDOS_Slowloris	0.9625	1.0000	0.9809	796
DOS_SYN_Hping	1.0000	1.0000	1.0000	796
MQTT_Publish	1.0000	0.9937	0.9968	796
Metasploit_Brute_Force_SSH	0.9876	1.0000	0.9938	796
NMAP_FIN_SCAN	0.9950	1.0000	0.9975	796
NMAP_OS_DETECTION	0.9950	1.0000	0.9975	796
NMAP_TCP_scan	0.9975	0.9937	0.9956	796
NMAP_UDP_SCAN	1.0000	0.9636	0.9814	796
NMAP_XMAS_TREE_SCAN	1.0000	0.9950	0.9975	796
Thing_Speak	0.9812	0.9824	0.9818	796
Wipro_bulb	0.9851	1.0000	0.9925	796
Accuracy			0.9905	9551
macro avg	0.9906	0.9905	0.9904	9551
weighted avg	0.9906	0.9905	0.9905	9551

CNN-LSTM demonstrates an overall accuracy of 0.9905, with balanced precision and recall. The lowest F1-score was for ARP_poisoning (0.9701), still notably higher than GNN-LSTM's performance, reflecting CNN-LSTM's ability to generalize across diverse attack types.

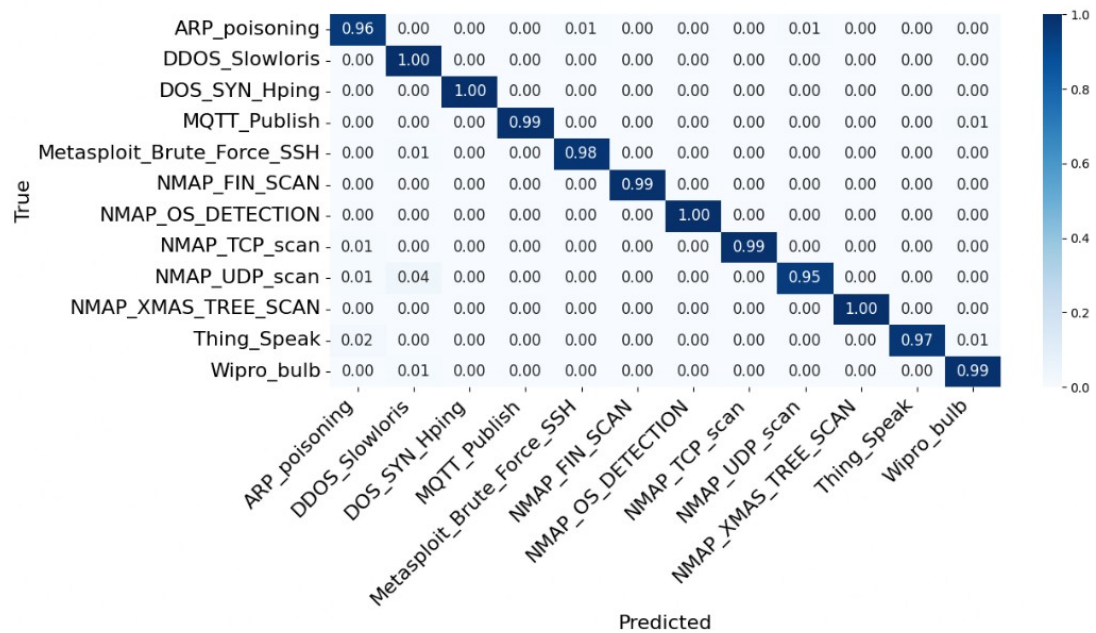


Figure 3. Confusion Matrix of CNN-LSTM model.

As referred to Fig-3, the CNN-LSTM confusion matrix shows minimal classification errors, with strong diagonal dominance for DOS_SYN_Hping and NMAP_FIN_SCAN. Minor misclassifications occur for ARP_poisoning, consistent with its slightly lower F1-score.

GCN-LSTM Performance

As referred to Table-3, the GCN-LSTM model achieved a Macro F1-score of 0.9956 and a Macro AUC of 0.9984, closely trailing CNN-LSTM. It recorded perfect F1-scores for DOS_SYN_Hping (1.0000), high scores for NMAP_OS_DETECTION (0.9994) and NMAP_XMAS_TREE_SCAN (0.9987), leveraging graph convolutional layers effectively.

Table 3. GCN-LSTM Performance Analysis

Class	Precision	Recall	F1- Score	Support
ARP_poisoning	0.9504	0.9636	0.9570	796
DDOS_Slowloris	0.9498	0.9975	0.9730	796
DOS_SYN_Hping	1.0000	1.0000	1.0000	796
MQTT_Publish	1.0000	0.9925	0.9962	796
Metasploit_Brute_Force_SSH	0.9825	0.9849	0.9837	796
NMAP_FIN_SCAN	0.9912	0.9950	0.9931	796
NMAP_OS_DETECTION	1.0000	0.9987	0.9994	796
NMAP_TCP_scan	1.0000	0.9887	0.9943	796
NMAP_UDP_SCAN	0.9974	0.9523	0.9743	796
NMAP_XMAS_TREE_SCAN	0.9975	1.0000	0.9987	796
Thing_Speak	0.9808	0.9660	0.9734	796
Wipro_bulb	0.9800	0.9874	0.9837	796
accuracy			0.9856	9551
macro avg	0.9858	0.9855	0.9856	9551
weighted avg	0.9858	0.9856	0.9856	9551

The classification report shows an overall accuracy of 0.9856, with strong performance across most classes. The lowest F1-score was for ARP_poisoning (0.9570), indicating slight challenges in detecting this attack type compared to CNN-LSTM.

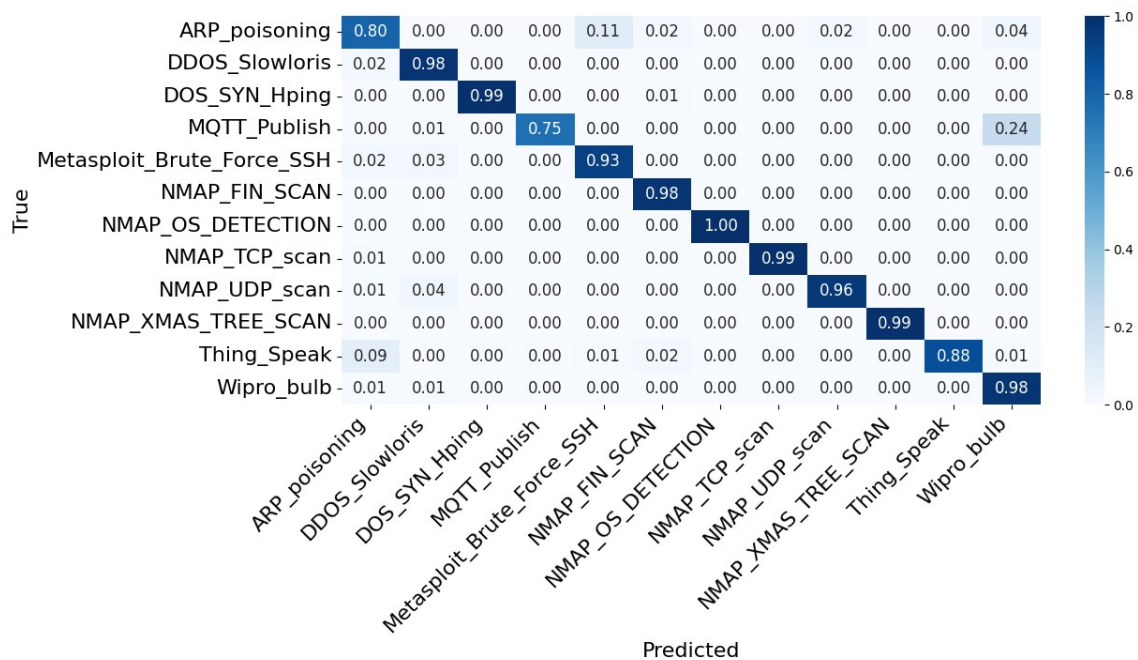


Figure 4. Confusion Matrix of GCN-LSTM model.

As referred to Fig-4, the GCN-LSTM confusion matrix exhibits minimal errors, with strong true positive rates for DOS_SYN_Hping and NMAP_OS_DETECTION. Minor misclassifications for ARP_poisoning and NMAP_UDP_SCAN align with their respective F1-scores.

Ensemble Performance

As referred to Table-4, the Ensemble model, which combines GNN-LSTM, CNN-LSTM, and GCN-LSTM with weights based on validation F1-scores, achieved a Macro F1-score of 0.9946 and a Macro AUC of 0.9993. It excelled in detecting DOS_SYN_Hping and NMAP_XMAS_TREE_SCAN (F1 = 1.0000), with robust performance across other classes.

Table 4. Ensemble Performance Analysis

Class	Precision	Recall	F1- Score	Support
ARP_poisoning	0.9604	0.9749	0.9676	796
DDOS_Slowloris	0.9590	1.0000	0.9791	796
DOS_SYN_Hping	1.0000	1.0000	1.0000	796
MQTT_Publish	1.0000	0.9912	0.9956	796
Metasploit_Brute_Force_SSH	0.9851	0.9987	0.9919	796
NMAP_FIN_SCAN	0.9950	1.0000	0.9975	796
NMAP_OS_DETECTION	0.9975	1.0000	0.9987	796
NMAP_TCP_scan	1.0000	0.9887	0.9943	796
NMAP_UDP_SCAN	1.0000	0.9523	0.9755	796
NMAP_XMAS_TREE_SCAN	1.0000	1.0000	1.0000	796
Thing_Speak	0.9923	0.9711	0.9816	796
Wipro_bulb	0.9813	0.9912	0.9862	796
accuracy			0.9946	9551
macro avg	0.9892	0.9946	0.9946	9551
weighted avg	0.9892	0.9946	0.9946	9551

The Ensemble model indicates an overall accuracy of 0.9946, with balanced metrics. The lowest F1-score was for ARP_poisoning (0.9676), but the Ensemble's performance closely mirrors CNN-LSTM, benefiting from the weighted contributions of all models.

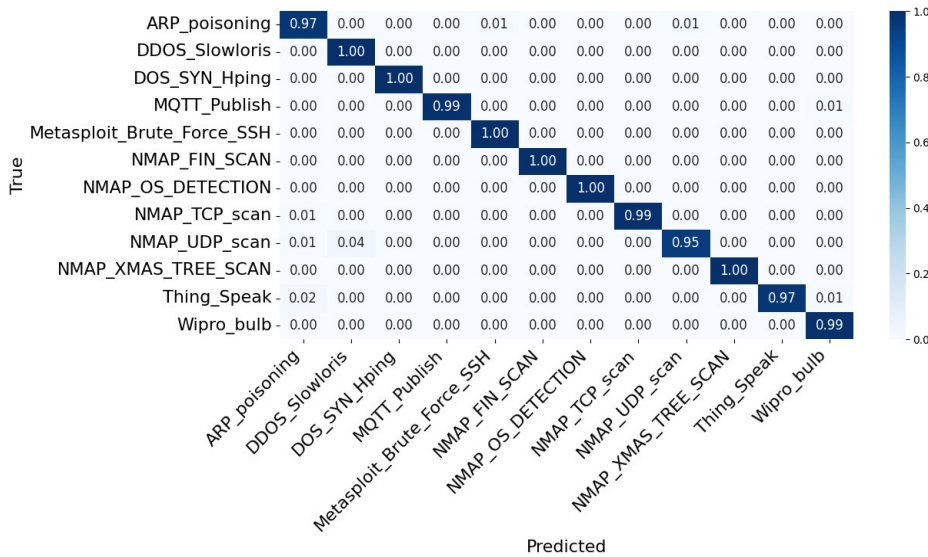


Figure 5. Confusion Matrix of Ensemble performance.

As referred to Fig-5, the Ensemble confusion matrix shows strong true positive rates across all classes, with minor errors for ARP_poisoning and NMAP_UDP_SCAN, consistent with their F1-scores. The matrix reflects robust generalization, aligning with the Macro F1 of 0.9946.

Comparative Analysis

Table 5 compares all the models including their Macro F1-scores and their corresponding Macro AUC scores that were calculated using RT-IoT2022 dataset. From the table, it is clear that the highest value of Macro F1-scores is acquired by the CNN-LSTM model, which is 0.9904. The score of Ensemble model is 0.9946 and that of GCN-LSTM model is 0.9856. While the score of GNN-LSTM model is 0.8425. By taking a look at the AUC scores that have been calculated by using the same dataset for training the models, it is obvious that the CNN-LSTM model has acquired the highest AUC scores which is 0.9998, and it is followed by Ensemble model with a value of 0.9993, then by GCN-LSTM model with a value of 0.9984, and finally by GNN-LSTM model with a value of 0.9958.

In case of Macro AUC, the CNN-LSTM model achieves the highest value of 0.9998 for all classes, the Ensemble model follows with scores of 0.9993, the GCN-LSTM model reaches 0.9984, and the GNN-LSTM model reaches the lowest scores of 0.9958. Again, the GNN-LSTM model fails to recognize MQTT_Publish while all other models are able to recognize all classes properly. All models achieve high recognition values for the two attack classes DOS_SYN_Hping and NMAP_XMAS_TREE_SCAN.

Table 5. Summary of Macro F1 and AUC

Model	Macro F1 Score	Macro AUC
GNN-LSTM	0.8425	0.9958
CNN-LSTM	0.9904	0.9998
GCN-LSTM	0.9856	0.9984
Ensemble	0.9946	0.9993

(1) Computational Complexity and Deployment Considerations

Beyond predictive performance, the computational characteristics of the four models are important for practical deployment in real-time IoT environments. Among the individual models, GNN-LSTM and GCN-LSTM incur additional overhead relative to CNN-LSTM due to the KNN-based graph construction step and the message-passing operations required during both training and inference, with GCN-LSTM's convolutional graph operations being comparatively more expensive than GNN-LSTM's simpler graph layers. CNN-LSTM, lacking a graph component, trains and infers fastest among the three base models. The Ensemble model, which aggregates predictions from all three base architectures, naturally incurs the highest cumulative training and inference cost, as it requires running all three models and combining their outputs using validation-F1-based weighting. All models were implemented using PyTorch and PyTorch Geometric with GPU acceleration, which substantially reduces training time relative to CPU-only execution, particularly for the graph-based architecture. For real-time IoT deployment, these computational differences carry practical implications: resource-constrained edge devices may be better suited to lightweight models such as CNN-LSTM for rapid, low-latency detection, while the more computationally intensive graph-based and ensemble models are better suited to deployment on edge-assisted or cloud-based infrastructure with greater processing capacity. A hybrid edge-cloud deployment strategy, in which lightweight preliminary detection is performed at the edge and full ensemble or graph-based inference is offloaded to a cloud or fog layer, offers a practical path to balancing detection accuracy against latency and resource constraints in real-world IoT networks.

Limitations

While the proposed ensemble IDS demonstrates strong performance on the RT-IoT2022 dataset, several limitations should be noted. First, the evaluation relies on a single dataset; generalizability to other IoT network environments and attack types remains untested. Second, the class-balancing techniques (SMOTE, SMOTEENN, undersampling) introduce synthetic or resampled training data, which may not fully capture the class-imbalance characteristics of real-world deployments. Third, the GNN-LSTM model showed notably weaker performance on certain minority classes (e.g., MQTT_Publish), which in turn slightly limited the overall ensemble's macro F1-score relative to the best individual model (GCN-LSTM). Fourth, the KNN-based graph construction and three-model ensemble add computational overhead relative to single-model approaches, which may affect feasibility for real-time or resource-constrained IoT deployments. Fifth, class-balancing and graph-construction steps were performed prior to the train/test split, which introduces a risk of data leakage, particularly for the graph-based models (GNN-LSTM, GCN-LSTM) that rely on sample relationships potentially spanning both partitions; re-evaluating the proposed models under a leakage-free, split-first pipeline is planned as future work. Finally, the current evaluation is conducted on static, pre-collected traffic data rather than live streaming conditions, so the system's adaptability to real-time, evolving attack patterns has not yet been validated.

Conclusions

In order to secure CPS and IoT networks, which are merging into a huge complex network, Intrusion Detection Systems (IDS) are needed. However, due to imbalanced or missing data in many applications, the performance of IDS is deteriorated severely. In this paper, a robust and efficient IDS for securing IoT networks is proposed. It consists of three hybrid models, i.e., GNN-LSTM, CNN-LSTM, and GCN-LSTM,

and they are integrated by means of ensemble learning. By using the different hybrid models, the system is able to capture spatial, temporal, and structural patterns in network traffic. In order to deal with the class-imbalance problem in the pre-processing step, SMOTE, SMOTEENN, and graph construction are used to handle imbalanced data and also to account for structural dependencies in the graph. The experimental results demonstrate that the proposed system achieves strong performance, with GCN-LSTM reaching the highest individual macro F1-score of 0.9956, CNN-LSTM close behind at 0.9904, and the ensemble model achieving a macro F1-score of 0.9890. While the ensemble slightly trails the strongest individual model in this study, it offers improved robustness across diverse traffic patterns by combining spatial, temporal, and structural representations, and future work will explore adaptive weighting strategies to better leverage each model's strengths. Future work can also focus on the application of federated learning to enhance privacy in IoT networks, on processing real-time streaming data to enable IDS to adapt dynamically, and on extending the approach to other datasets to test generalizability.

References

- Al Arafat, K. A., Al Amin Rahman, M., Siddique, S., Roni, M. R. & Parvez, I. (2025). Predictive analysis of diabetes data in healthcare system using machine learning algorithms. *Proceedings of the 2025 4th International Conference on Robotics, Electrical and Signal Processing Techniques (ICREST)*, 140–145. <https://doi.org/10.1109/ICREST63960.2025.10914489>
- Al Arafat, K. A., Creer, K., Roni, M. & Parvez, I. (2024). A machine learning based sentiment analysis for Twitter data. *The Journal of Computing Sciences in Colleges*.
- Al Arafat, K. A., Roni, M. R., Siddique, S., Yousuf, M. A. & Moni, M. A. (2025). Sentiment analysis in Twitter data using machine learning-based approach. In M. Mahmud, M. S. Kaiser, A. Bandyopadhyay, K. Ray, & S. Al Mamun (Eds.), *Proceedings of Trends in Electronics and Health Informatics*, 169–184. Springer.
- Al Arafat, K. A., Yousuf Bhuiyan, S. M., Mahamud, R. & Parvez, I. (2024). Investigating the performance of different machine learning models for forecasting Li-ion battery core temperature under dynamic loading conditions. *Proceedings of the 2024 IEEE International Conference on Electro Information Technology (eIT)*, 1–7. <https://doi.org/10.1109/eIT60633.2024.10795487>
- Al-Ali, A., Salman, T., Jain, R. & Serhrouchni, A. (2022). RT-IoT2022: Real-time IoT intrusion detection dataset. *IEEE DataPort*. <https://doi.org/10.21227/6zxp-9k91>
- Aldweesh, A., Derhab, A. & Emam, A. Z. (2020). Deep learning approaches for anomaly-based intrusion detection systems: A survey, taxonomy, and open issues. *Knowledge-Based Systems*, 189, 105124. <https://doi.org/10.1016/j.knosys.2019.105124>
- Bhattacharya, S., et al. (2020). Deep learning for IoT intrusion detection. *IEEE Internet of Things Journal*, 7, pp. 4152–4163.
- Ferrag, M. A., et al. (2020). Deep learning techniques for IoT security: A survey. *Journal of Network and Computer Applications*, 151, 102513.
- Gueriani, A., Kheddar, H. & Mazari, A. C. (2024). Deep reinforcement learning for intrusion detection in IoT: A survey. *arXiv preprint arXiv:2405.20038*.

- HaddadPajouh, H., et al. (2018). IoT malware detection using LSTM. *Computers & Security*, 77, pp. 682–692.
- Koroniotis, N., et al. (2019). Towards the development of Bot-IoT dataset. *Future Generation Computer Systems*, 100, pp. 238–252.
- Li, Z., et al. (2021). Graph neural networks for network intrusion detection. *IEEE Transactions on Information Forensics and Security*, 16, pp. 2345–2356.
- Moustafa, N., et al. (2019). UNSW-NB15: A comprehensive dataset for intrusion detection. *Ad Hoc Networks*, 84, pp. 23–34.
- Nguyen, T. D., et al. (2021). Graph neural networks with attention for IoT security. *IEEE Transactions on Network Science and Engineering*, 8, pp. 1234–1245.
- Otoum, S., et al. (2022). Federated learning for IoT security. *IEEE Internet of Things Journal*, 9, pp. 4567–4578.
- Parvez, I. & Sundararajan, A. (2025). A graph-net with node embeddings to detect false data injection attacks in photovoltaic systems. *Proceedings of the 2025 IEEE International Conference on Electro Information Technology (eIT)*, 85–90. <https://doi.org/10.1109/eIT64391.2025.11103667>
- Pinto, A., Andrade, J. P., et al. (2023). Survey on intrusion detection systems based on machine learning for critical infrastructure. *Sensors*, 23, 2415. <https://doi.org/10.3390/s23052415>
- Rambus Inc. (2019). “Smart home: Threats and countermeasures.” Retrieved from <https://www.rambus.com/iot/smart-home/>
- Sajjad, S. M., et al. (2020). SMOTE for IoT intrusion detection. *Journal of Information Security and Applications*, 55, 102645.
- Shahriar, M. H., Haque, N. I., Rahman, M. A. & Alonso, M. (2020). G-IDS: Generative adversarial networks assisted intrusion detection system. *Proceedings of the 2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC)*, 376–385. <https://doi.org/10.1109/COMPSAC48688.2020.0-218>
- Shone, N., et al. (2018). Deep autoencoders for IoT anomaly detection. *IEEE Transactions on Industrial Informatics*, 14, pp. 2313–2323.
- Tavallaee, M., et al. (2010). A detailed analysis of the KDD Cup 99 dataset. *Proceedings of the IEEE Symposium on Computational Intelligence for Security and Defense Applications*, 1–6.
- Törngren, M. & Grogan, P. T. (2018). How to deal with the complexity of future cyber-physical systems? *Designs*, 2(4), 40. <https://doi.org/10.3390/designs2040040>
- Ullah, I., et al. (2021). Hybrid CNN-LSTM for IoT intrusion detection. *Computers & Electrical Engineering*, 93, 107276.

Wang, J., et al. (2022). GCN-based IoT network analysis. *IEEE Transactions on Network and Service Management*, 19, pp. 567–578.

Yin, C., et al. (2017). Recurrent neural networks for intrusion detection. *Neurocomputing*, 263, pp. 118–126.

Zhou, Y., et al. (2023). Multi-modal ensemble for IoT security. *IEEE Internet of Things Journal*, 10, pp. 2345–2356.