LNNIDS: A Hybrid Liquid Neural Network based IDS for Known and Unknown IoT Attacks

RAKESH KUMAR, Indian Institute of Technology BHU Varanasi, Varanasi, India
MAYANK SWARNKAR, Indian Institute of Technology (BHU) Varanasi, Varanasi, India
MANMEET MUSKAN, Indian Institute of Technology (BHU) Varanasi, Varanasi, India

The rapid expansion of Internet of Things (IoT) networks has introduced new cybersecurity risks, particularly due to their heterogeneous, dynamic, and resource-constrained nature. Intrusion Detection Systems (IDS) are commonly used to detect cyber threats in such environments, but traditional IDSs and many machine learning (ML) or deep learning (DL)-based methods face challenges. These include poor detection of unknown attacks, dependence on large datasets, static feature learning, and an inability to adapt to evolving traffic patterns without frequent retraining. Furthermore, while graph-based IDSs capture relational context, their high computational cost and difficulty in adapting to dynamic IoT networks hinder their scalability. To address these challenges, we propose *LNNIDS*, a novel IDS method that integrates spike encoding with a Hybrid Liquid Neural Network (HLLN) to capture the temporal evolution of IoT attack network traffic. To further enhance accuracy, we introduce *DYNGraph*, a dynamic graph-based method that performs $n \rightarrow 1$ pattern clustering for scalable attack classification. This allows the *LNNIDS* to dynamically group related flow behaviors and adapt to new or unknown attacks without retraining. We evaluated *LNNIDS* on two public IoT datasets. This method achieved 99.50% accuracy and maintained robust performance with only 20% training data. Furthermore, we compared the *LNNIDS* with the seven recent state-of-the-art methods, and we found that *LNNIDS* outperformed them in the detection of known and unknown IoT attacks with 13.45% and 11.99% better accuracy, respectively.

CCS Concepts: • **Security and privacy** → **Intrusion detection systems**; • **Computing Methodologies** → *Neural networks*; • **Networks** → *Traffic classification*; • **Hardware** → Sensor applications and deployments;

Additional Key Words and Phrases: IoT Attacks, liquid neural network, dynamic graph learning, traffic classification, application-layer protocols, application layer protocols, network traces, and bipertight graph

ACM Reference Format:
Rakesh Kumar, Mayank Swarnkar, and Manmeet Muskan. 2025. LNNIDS: A Hybrid Liquid Neural Network based IDS for Known and Unknown IoT Attacks. *ACM Trans. Internet Technol.* 25, 4, Article 26 (November 2025), 31 pages. https://doi.org/10.1145/3771739

1 Introduction

The rapid growth of **Internet of Things** (**IoT**) technology has revolutionized sectors such as smart homes, healthcare, and logistics by enabling seamless control of devices like smartwatches and thermostats. Since IoT devices are expected to grow from 9.7 billion in 2020 to over 29 billion by

Authors' Contact Information: Rakesh Kumar, Indian Institute of Technology BHU Varanasi, Varanasi, Uttar Pradesh, India; e-mail: rakeshkumar.rs.cse21@itbhu.ac.in; Mayank Swarnkar (corresponding author), Indian Institute of Technology (BHU) Varanasi, Varanasi, UP, India; e-mail: mayank.cse@iitbhu.ac.in; Manmeet Muskan, Indian Institute of Technology (BHU) Varanasi, Varanasi, UP, India; e-mail: manmeet.muskan.cse21@iitbhu.ac.in.

ACM 1533-5399/2025/11-ART26
https://doi.org/10.1145/3771739

2030, [1], their widespread deployment introduces significant security risks. These devices are computationally weaker and often lack built-in security mechanisms, making them vulnerable to sophisticated cyber-attacks targeting protocols, firmware, and networks. Integrating IoT with traditional infrastructures has expanded the attack surface, enabling attackers to exploit vulnerabilities and cause financial losses, data breaches, and operational disruptions [8, 38]. With global cybercrime costs expected to reach $10.5 trillion annually by 2025, [31], robust IoT attack detection systems are crucial. Analyzing network traffic patterns to detect complex attack behaviors is essential for protecting smart environments and strengthening IoT security [6, 12, 21, 26]. Traditional methods, such as statistical, **machine learning** (**ML**), **deep learning** (**DL**), and graph-based methods, have been widely used to detect malicious traffic and cyber threats in IoT networks; however, they face significant limitations when dealing with unknown attack patterns. Statistical methods are unable to detect malicious behavior deviating from known attack signatures [13, 39, 45]. Traditional ML methods face challenges with unknown IoT attacks due to their reliance on static data and fixed features, leading to poor adaptability and high false positive rates [15, 25, 45]. DL methods, though capable of learning complex patterns, often lack robustness against unknown IoT attacks and demand large training datasets, making them resource-intensive [17, 24, 40]. Furthermore, detecting unknown IoT attacks is challenging due to the highly dynamic nature of IoT networks; frequent changes in device states, attack behaviors, and communication structures impose high update and computation costs on graph-based methods [22, 27, 47]. Detecting unknown IoT attacks is particularly challenging due to the diversity of devices across different applications and vendors, as well as the frequent additions and removals that create a constantly evolving environment. Moreover, attackers exploit weaknesses in IoT networks to launch attacks such as DDoS, MITM, and backdoor intrusions that constantly change their behavior. These attacks often appear similar to normal traffic during short time periods, making early detection difficult. For example, stealthy port scans or low-rate DoS attacks generate traffic that appears normal due to their low packet rates and minimal flag usage, but suddenly become aggressive. Similarly, malware like Mirai hides within regular traffic patterns before sending large bursts of data. As a result, traditional detection methods, whether statistical, ML, DL, or graph-based, face challenges to accurately detect unknown attacks because they need frequent retraining to keep up with these changing and unpredictable threat behaviors. Additionally, static models are unable to generalize from previously seen signatures of attacks when encountering new signatures or misuse of network protocols, such as incorrectly formatted packets or hidden communications. To tackle these issues, we propose *LNNIDS*, a hybrid liquid neural network–based incremental supervised classifier that accurately identifies both known and unknown IoT attacks by dynamically assigning weights within a specific period for each unknown attack pattern. Unlike traditional intrusion detection methods, which require retraining, this approach enables continuous learning. This approach enhances the effectiveness of our dynamic method when combined with the statistical method *DYNGraph*, particularly when training on a small set of flows and testing on known and unknown IoT attack flows. The key contributions of this work are as follows:

— We propose *LNNIDS*, a hybrid liquid neural network-based supervised classifier, to detect attacks on IoT devices. *LNNIDS* can learn from a small dataset and accurately classify attacks.
— *LNNIDS* utilizes flow-based features, which are encoded into spikes and then fed into a liquid layer to generate dynamic states at specific time steps. These states serve as features for identifying known and unknown attacks on IoT devices.
— We propose *DYNGraph*, a graph-based data structure that stores all attack patterns and performs pattern matching of a test flow against all available attack patterns quickly and accurately. *DYNGraph* performs $n \rightarrow 1$ pattern matching instead of $1 \rightarrow 1$.

— We conducted extensive experiments on two publicly available datasets, achieving an average accuracy of 99.50%. The method also performed better than most closely related state-of-the-art methods.

The remainder of the article is organized as follows. In Section 2, we review related work. Section 3 details our proposed method, *LNNIDS*. In Section 5, we evaluate the performance of *LNNIDS* through experiments and compare it with a state-of-the-art method. Finally, Section 6 concludes the article and outlines future research directions.

2 Related Work

This section provides an overview of the current literature on IoT attack network traffic classification, which can be categorized into five main groups: Statistical Methods, ML-based methods, DL-based methods, and Graph-based methods.

2.0.1 Statistical-Based Methods. Moustafa et al. [34] proposed a one-class statistical method for zero-day attack detection in IoT networks. It utilized 15 flow-based features using PCA and applied a **Gaussian Mixture Model** (**GMM**) with Correntropy to set a detection threshold. The method achieved 99.90% accuracy on 13 attack types but requires large training samples, increasing complexity. Ashraf et al. [14] proposed a statistical method for detecting botnet attacks in IoT networks. It uses eight flow-based features to derive three statistical metrics: outbound traffic size, total traffic, and packet count. A **beta mixture model** (**BMM**) with Correntropy sets detection thresholds, achieving 99.93% accuracy on five botnet attack types. However, it fails to detect attack variations due to limited training data. Velliangiri et al. [43] proposed a statistical method for detecting DDoS attacks in smart networks. This method utilizes 15 packet-based features selected using the Pearson Correlation Coefficient and extracts correlations via Manhattan distance to generate feature distance maps. It utilizes the Hellinger distance to separate normal and malicious network traffic based on a threshold value, achieving 93.7% accuracy in detecting DoS attacks. However, its reliance on statistical thresholds limits adaptability to dynamic traffic. Saiyed et al. [36] proposed a statistical method for detecting low and high-volume DDoS attacks in IoT networks. It extracts eight flow-based features and computes four metrics: distance, entropy, greedy bin packing, and KL divergence. The method then calculates thresholds to detect attack volumes, achieving 90% accuracy in detecting DDoS attacks. However, reliance on statistical thresholds limits its effectiveness against large-scale dynamic attacks. Nie et al. [35] proposed an unsupervised anomaly detection method for IoT traffic using multi-view subspace learning. It utilizes packet header and payload features projected onto a unified low-dimensional subspace to reduce noise and redundancy, with spectral clustering applied for anomaly detection. This method achieved 98.53% accuracy in detecting 20 attacks but faces challenges with real-time deployment due to computational overhead and sensitivity to parameter tuning.

2.0.2 Machine Learning (ML)-Based Methods. Cvitic et al. [19] proposed a method for detecting DDoS attacks in IoT networks using a boosting method. This method utilized 76 out of 83 flow-based features and trained a **Logistic Model Tree** (**LMT**) based boosting model. It achieved an accuracy of 99.99% in detecting three types of DDoS attacks on IoT networks. However, this method faces challenges in generalizing to completely new device types without retraining. Mihoub et al. [33] proposed a ML-based method for detecting and mitigating DoS/DDoS attacks in IoT networks using the Looking-Back method. It utilized 10 flow features and trained four ML models (**Decision Tree** (**DT**), **Random Forest** (**RF**), **Random Tree** (**RT**), **K-Nearest Neighbour** (**KNN**), and two deep learning models (RNN, LSTM). Among these, RF achieved an accuracy of 99.81% in detecting six DoS/DDoS attacks. However, challenges remain in adapting to evolving threats. Kamaldeep et al. [32] proposed a ML-based method for detecting DDoS attacks in IoT networks using feature engineering.

This method extracted 21 time and packet-based features selected by Pearson's correlation method and normalized the dataset using Z-score transformation. Five ML classifiers in which RF achieves 98.8% accuracy in detecting five DDoS attack types. However, this method requires periodic updates to keep pace with evolving threats. El-Sofany et al. [20] proposed a ML-based method for detecting IoT attacks. This method utilizes 23 packet statistics and traffic behavior-based features and trains four ML models along with three ensemble models: RF, **Naive Bayes** (**NB**), DT, **Neural Network** (**NN**), AdaBoost, XGBoost, and Ensemble RF-BPNN. Among these, the Ensemble RF-BPNN achieved 99.9% accuracy in detecting eight types of attacks. However, this method requires improvements to handle complex and dynamic attack traffic. Alsabilah and Rawat [9] proposed a hybrid method for intrusion detection in IoT networks. This method utilizes rule-based features generated by Rough Set Theory and integrates them with XGBoost to handle imbalanced data and heterogeneity in IoT networks. It achieved 99.6% accuracy in detecting three types of IoT attacks. However, this method requires periodic updates to keep pace with evolving threats.

2.0.3 Deep Learning(DL)-Based Methods. Kunang et al. [28] proposed an IoT attack detection method (DAE-DNN) using a pre-trained deep autoencoder and deep neural network with 122 flow features, achieving 95.79% accuracy on 15 attacks. However, it suffers from overfitting on imbalanced data, which limits its performance. Zhou et al. [48] proposed a **Distribution Bias Aware Collaborative GAN** (**DB-CGAN**) for IoT attack detection using 110 flow features with adversarial training among a generator, discriminator, and classifier. It achieved 82.23% accuracy on non-IoT datasets but requires further evaluation for industrial IoT scenarios. Altunay and Albayrak [11] proposed an IoT intrusion detection method using CNN, LSTM, and a hybrid CNN+LSTM with 110 flow features, achieving 99.84% binary and 99.80% multi-class accuracy on 18 attack types, though with high resource consumption. Sharma et al. [37] proposed an IoT intrusion detection system using 39 correlation-selected features with DNN and CNN models, applying LIME and SHAP for interpretability, achieving 81% accuracy on four attack types, but face challenges with minority class imbalance. Wang et al. [46] proposed a **convolutional spiking neural network** (**CSNN**)-based intrusion detection method for IoT networks. Using 64 features selected via the `SelectKBest` method, they encoded each feature into a 4×4 matrix, forming a 32×32 input for the CSNN. This model achieved 98.82% accuracy on the CSE-CIC-IDS2018 dataset and 99.86% on the CIC-DDoS2019 dataset. While efficient, it still falls short of the highest accuracy attained by more traditional, resource-intensive deep neural network models. Hou et al. [23] proposed FTSecureBert and BLWSecureBert for IoT attack detection, achieving 99% and 73.38% accuracy on LUFlow and CIC-IDS datasets, respectively. While FTSecureBert is highly accurate but resource-intensive, BLWSecureBert reduces parameters 3.3× with minimal loss, though misclassification in complex scenarios limits generalizability.

2.0.4 Graph-Based Methods. Chen et al. [18] proposed a Transformer-based architecture with graph learning for multivariate time-series anomaly detection in IoT networks. This method uses 123 temporal features as input for Gumbel-Softmax-based connection learning, influence propagation, and hierarchical dilated convolutions to address temporal dependencies, achieving a 91% F1-score in detecting 56 IoT attacks. However, it faces challenges in modeling weakly correlated features and efficiency tradeoffs in larger graphs. Lo et al. [30] proposed a graph neural network-based intrusion detection system for IoT networks. It constructs graphs using flow-based features and trains E-GraphSAGE to classify network flows as benign or malicious. It achieves 99.99% accuracy for binary classification but faces challenges with detecting the minority class. Altaf et al. [10] proposed a multigraph neural network for IoT intrusion detection using a multi-edge graph where nodes represent devices and edges capture communication flows. This method, which combines Graph Convolutional and Graph Attention Networks, achieved 99.96% accuracy for four attacks

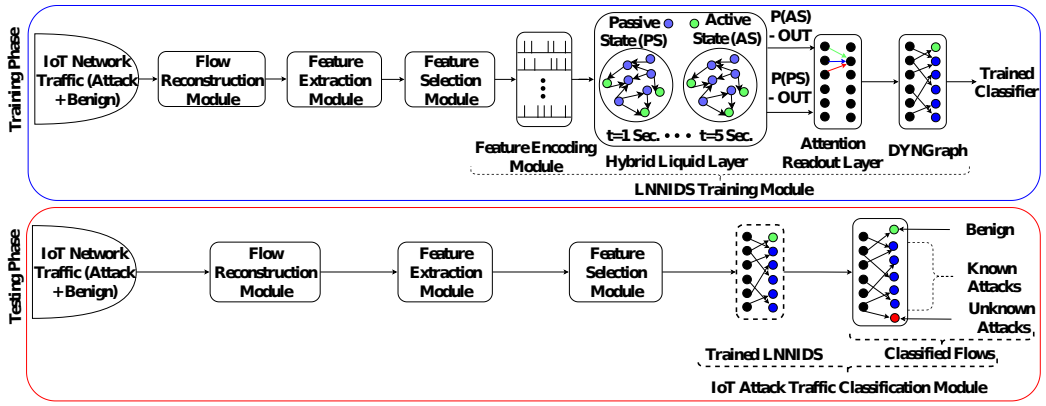


Fig. 1. Architecture of *LNNIDS*.

but suffers from high computational complexity due to its multi-edge structure. Termos et al. [41] proposed a graph-based intrusion detection method for IoT networks. It dynamically selects centrality measures, including degree, closeness, betweenness, eigenvector, and Katz centrality, to construct a graph. This graph is then combined with 80 statistical features to train LSTM and CNN models. LSTM achieved 99.05% accuracy, but the method requires large datasets for effective generalization. Lin et al. [29] proposed a GNN-based intrusion detection method for IoT networks using 43 NetFlow features. It constructs a graph where nodes represent devices and edges denote traffic flows. The model integrates GraphSAGE, global attention, gating mechanisms, and contrastive learning, achieving 98.53% accuracy in detecting four types of attacks. However, it faces difficulties in detecting minority attacks.

3 Proposed Method

This section describes *LNNIDS*, a robust hybrid liquid neural network-based method for identifying known and unknown IoT attacks in a network. It utilizes flow-based features to identify attacks on IoT device networks. *LNNIDS* reconstructs bidirectional flows from IoT traffic traces, extracts flow-based features, and encodes them into time-based spike states, sequences of discrete states that capture the temporal dynamics of the flow features using static weights. These encoded time-based spike states are then input into the **Hybrid Liquid Neural Network** (**HLNN**) to extract dynamic liquid states using the dynamic weights associated with each new encoded spike, thereby enhancing the training process of *LNNIDS*. Afterward, these dynamic liquid states are refined using the readout attention mechanism that assigns weights to the important states. To further improve performance, *DYNGraph* is proposed as a graph-based data structure that stores all attention-based probabilistic attack features generated by the HLNN, enabling fast and accurate classification of known and unknown IoT attacks. The architecture of *LNNIDS* consists of the Training and Testing phases, which are shown in Figure 1, each described in the following subsections.

3.1 Training Phase

This phase consists of four modules: Flow Reconstruction, Feature Extraction, Feature Selection, and *LNNIDS* Training Module. The training phase is shown in Figure 1. The explanation of each module of the training phase is as follows.

3.1.1 Flow Reconstruction Module. This module takes benign and attack IoT network traffic traces and outputs the reconstructed flows from these traces. A flow is a sequence of packets

Table 1. Extracted Features (Statistical-*St*, Protocol-*Pr*, and Behavioral-*Be*)

Feature ID	Feature	Feature ID	Feature	Feature ID	Feature	Feature ID	Feature
$X_1(St)$	fwd_iat_tot	$X_{21}(St)$	fwd_byts_b_avg	$X_{41}(St)$	cwe_flag_count	$X_{61}(St)$	bwd_seg_size_avg
$X_2(St)$	fwd_iat_max	$X_{22}(St)$	bwd_iat_tot	$X_{42}(Pr)$	protocol	$X_{62}(St)$	bwd_pkt_len_mean
$X_3(St)$	fwd_iat_mean	$X_{23}(St)$	bwd_iat_max	$X_{43}(Pr)$	fwd_psh_flags	$X_{63}(St)$	bwd_iat_min
$X_4(Be)$	flow_duration	$X_{24}(St)$	active_min	$X_{44}(St)$	bwd_iat_mean	$X_{64}(St)$	pkt_len_std
$X_5(St)$	flow_iat_max	$X_{25}(St)$	fwd_blk_rate_avg	$X_{45}(Be)$	subflow_bwd_byts	$X_{65}(St)$	pkt_len_var
$X_6(St)$	idle_max	$X_{26}(St)$	fwd_pkts_b_avg	$X_{46}(St)$	totlen_bwd_pkts	$X_{66}(St)$	bwd_pkt_len_std
$X_7(St)$	idle_mean	$X_{27}(St)$	bwd_iat_std	$X_{47}(St)$	fwd_seg_size_avg	$X_{67}(St)$	fwd_pkt_len_max
$X_8(St)$	flow_iat_mean	$X_{28}(St)$	fwd_iat_min	$X_{48}(St)$	fwd_pkt_len_mean	$X_{68}(St)$	pkt_len_max
$X_9(St)$	idle_std	$X_{29}(St)$	idle_min	$X_{49}(St)$	fwd_pkt_len_std	$X_{69}(St)$	init_fwd_win_byts
$X_{10}(Be)$	fwd_pkts_s	$X_{30}(Pr)$	bwd_urg_flags	$X_{50}(St)$	bwd_blk_rate_avg	$X_{70}(St)$	fwd_header_len
$X_{11}(Be)$	flow_pkts_s	$X_{31}(Pr)$	fwd_urg_flags	$X_{51}(St)$	bwd_byts_b_avg	$X_{71}(St)$	init_bwd_win_byts
$X_{12}(Be)$	flow_byts_s	$X_{32}(St)$	ece_flag_cnt	$X_{52}(St)$	bwd_pkts_b_avg	$X_{72}(St)$	subflow_fwd_pkts
$X_{13}(St)$	flow_iat_std	$X_{33}(St)$	fin_flag_cnt	$X_{53}(St)$	flow_iat_min	$X_{73}(St)$	tot_fwd_pkts
$X_{14}(St)$	fwd_iat_std	$X_{34}(St)$	syn_flag_cnt	$X_{54}(St)$	pkt_size_avg	$X_{74}(Pr)$	src_port
$X_{15}(St)$	active_max	$X_{35}(St)$	fwd_seg_size_min	$X_{55}(St)$	pkt_len_mean	$X_{75}(St)$	bwd_pkt_len_max
$X_{16}(St)$	active_mean	$X_{36}(St)$	rst_flag_cnt	$X_{56}(St)$	bwd_header_len	$X_{76}(St)$	bwd_pkt_len_min
$X_{17}(St)$	active_std	$X_{37}(St)$	psh_flag_cnt	$X_{57}(St)$	fwd_act_data_pkts	$X_{77}(St)$	pkt_len_min
$X_{18}(St)$	subflow_fwd_byts	$X_{38}(St)$	ack_flag_cnt	$X_{58}(St)$	subflow_bwd_pkts	$X_{78}(Pr)$	dst_port
$X_{19}(St)$	totlen_fwd_pkts	$X_{39}(St)$	urg_flag_cnt	$X_{59}(St)$	tot_bwd_pkts	$X_{79}(St)$	fwd_pkt_len_min
$X_{20}(Be)$	bwd_pkts_s	$X_{40}(Pr)$	bwd_psh_flags	$X_{60}(St)$	down_up_ratio		

sharing common attributes between hosts, such as source and destination IP addresses, source and destination ports, and the Layer 4 protocol. There are two primary protocols at Layer 4, TCP and UDP, that generate TCP and UDP flows, respectively. A TCP flow consists of all packets transmitted between the establishment and termination of a TCP connection, whereas UDP flows include packets transmitted over UDP that have inter-packet time intervals within a predefined threshold. This threshold is manually tuned and set by the user during the reconstruction of UDP flows. Let us consider a set of incoming packets $P = \{P_1, P_2, P_3, \ldots, P_n\}$. The method organizes them into a set of flows $F = \{F_1, F_2, F_3, \ldots, F_m\}$, where each flow groups packets with common attributes such that $(n \geq m)$.

3.1.2 Feature Extraction Module. This module takes bidirectional flows (F) as input and generates flow-based features ($X = \{\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3, \ldots, \mathbf{X}_d\}$) as output for each flow $F_i \in F$. Once the flows are reconstructed, the method computes a diverse set of features, statistical-based, protocol-based, and behavioral-based are shown in Table 1.

As shown in Table 1, the extracted features include flow duration, packet count, byte count, inter-packet arrival times, flow direction ratio, and payload size distribution. Additionally, transport-layer attributes such as TCP flag counts, connection setup time, and retransmission rates are extracted. Meanwhile, UDP-specific features, including packet loss, inter-packet arrival times, and data rate, are also considered. Furthermore, application-layer features relevant to IoT protocols, such as MQTT, CoAP, and HTTP (i.e., CONNECT, PUBLISH, SUBSCRIBE, GET, POST, etc.), are incorporated to enhance the representation of network traffic behavior. These flow-based features are essential for detecting IoT network attacks by identifying abnormal traffic patterns.

3.1.3 Feature Selection Module. The input to this module consists of a flow-based feature set (i.e., X), and the output is the selected feature subset (i.e., X'). The feature selection process is shown in Figure 2.

As shown in Figure 2, Feature selection employs the standard Wrapper Method, which is applied to the DT to compute relevance scores for the features extracted from IoT attack network traffic. This module iteratively evaluates the relevance score for different feature combinations by applying

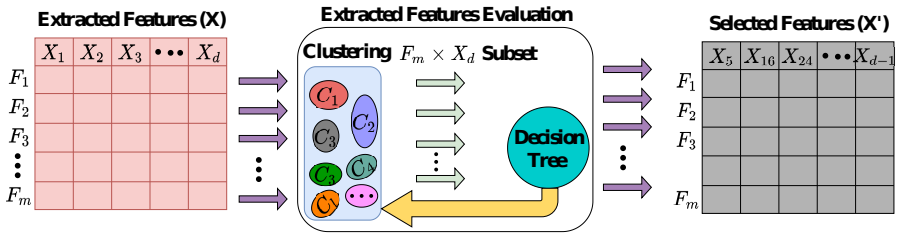


Fig. 2. Feature selection process of *LNNIDS*.

clustering to each combination to assign pseudo-target labels y' that capture the underlying traffic patterns. Given a feature set X and pseudo-target labels y', each feature X_i is evaluated by training a decision algorithm with cross-validation, yielding an accuracy score I by using Equation (1).

$$I(X_i) = \frac{1}{k} \sum_{j=1}^{k} \text{score}_{ij}. \tag{1}$$

where k represents the total number of evaluation models used to compute the average score for the feature X_i. A feature achieves an importance score near 1; it is considered highly relevant, otherwise less relevant. Initially, k-means clustering is applied to all extracted features to generate clusters. Next, subsets are formed for each feature within these clusters. A DT is then used to learn from these subsets and generate importance scores for each feature, and this process is repeated until all clusters have been processed. This design enables the method to discover feature dependencies critical for separating complex IoT traffic types, for instance, distinguishing between high-rate packet bursts in HTTP floods and subtle SYN anomalies in stealthy port scans. Moreover, using DTs only during feature selection prevents overfitting and improves generalizability across evolving IoT environments.

3.1.4 LNNIDS Training Module. This module takes the selected feature set X' from the previous module for each IoT attack as input and produces a trained *LNNIDS* as output. The *LNNIDS* Training Module comprises the following sub-modules: Feature Encoding, Hybrid Liquid Layer, Attention Readout Layer, and *DYNGraph*. These sub-modules are explained below.

(1) Feature Encoding Sub-Module

The input to this sub-module is the selected feature set X', and the output is a time-based spike state (S) generated using the concept from the spike encoding method [16]. Spike encoding maps each feature to a sequence of spikes using static weights at specific time steps, thereby capturing the temporal dependencies of attack behavior. This method efficiently processes real-time IoT traffic with reduced computational complexity. Given the sporadic, event-driven nature of IoT networks, spike encoding offers a sparse and energy-efficient representation, making it ideal for smart environments. It enhances attack detection for threats like DDoS, scanning, and botnets. To illustrate the spike feature encoding S, it is defined $S = \max(0, WX' + b)$ at a specific time-step, Where W is the weight matrix applied to the selected feature set X', with bias b to produce the spiked-encoded states under the following conditions for $Z = WX' + b$: If $Z > 0$, retain the value, If $Z \leq 0$, set it to 0.

Table 2 illustrates the process of encoding selected features into time-based spike series (S), where each flow F_i with its associated selected features X_i is encoded using a static weight matrix and bias at specific time steps to generate the respective spike-encoded state S. This encoding maps input features to a sparse, time-based representation, enabling efficient dynamic traffic analysis in smart environments.

Table 2. Representation of Input Flows with Selected Features and Temporal Spike States

Selected Flow Features (X') and Temporal Spike States S										
Flows	$\mathbf{X}_5$	$\mathbf{X}_{16}$	$\mathbf{X}_{24}$	$\mathbf{X}_{37}$	$\mathbf{X}_{d-1}$	**Time-based Spike States (S)**				
						$S(1s)$	$S(2s)$	$S(3s)$	$S(4s)$	$S(5s)$
F_1	0.1	0.2	0.3	0.4	0.5	2	3	4	3	2
F_2	0.3	0.4	0.5	0.6	0.7	3	4	5	4	3
F_3	0.7	0.8	0.9	1.0	1.1	5	6	7	6	5
⋮			⋮					⋮		
F_m	1.0	1.1	1.2	1.3	1.4	6.5	7.5	8.5	7.5	6.5

(2) Hybrid Liquid Layer (HLL) Sub-Module

The input to this submodule is the time-based spike states (S), and the output is the Active State (AS) and the Passive State (PS). *LNNIDS* utilizes the Liquid State Machine [44] concept to process time-based spike states (S), transforming them into high-dimensional dynamic states. These states transition over time in the form of interconnected AS and PS. The liquid layer captures temporal dependencies by processing combinations of S-states, including pairs, triplets, and higher-order interactions in parallel. These states are initialized with dynamic weights at specific time steps, which control their interactions and enable the adaptive learning of temporal correlations among selected flow-based features. This method improves the processing of diverse malicious IoT traffic, especially from low-interaction devices. The AS and PS states are generated by processing the S-states over time. The AS is computed as the immediate response to the input spikes, while the PS captures the sustained state response. The AS at time t is defined in Equation (2).

$$AS(t) = \tanh\big(\mathbf{W}_{\text{in}} \cdot \mathbf{X_i}(t)\big), \quad PS(t) = \frac{1}{T}\sum_{i=0}^{T-1} AS(t-i), \tag{2}$$

where, $\mathbf{W}_{\text{in}}$ is the weight matrix applied to the input S-states $\mathbf{X_i}(t)$, and the passive state $PS(t)$ is computed by averaging the historical active states over a variable time window T. These states are updated by applying the activation function to the weighted sum of the current active state and the previous state $\mathbf{s}_{\text{prev}}$, yielding the updated state $\mathbf{s}(t)$ as defined in Equation (3).

$$\mathbf{s}(t) = \tanh(\mathbf{W}\text{l} \cdot AS(t) + \mathbf{W}\text{res} \cdot \mathbf{s}_{\text{prev}}(t)), \tag{3}$$

where $AS(t)$ represents the active state at time t derived from the input spike states with $\mathbf{s}_{\text{prev}}(t)$ as the previous HLL layer state. The weight matrices $\mathbf{W}_{\text{l}}$ and $\mathbf{W}_{\text{res}}$ control the current and previous states, respectively, while the activation function $\tanh(\cdot)$ introduces non-linearity. To ensure temporal consistency, liquid states are stored within a tracking time window of variable length T, defined as $\mathbf{s}_{\text{track}}(t) = [\mathbf{s}(t-T+1), \ldots, \mathbf{s}(t)]$, and the final state representation $\mathbf{h}(t) = [AS(t), PS(t)]$. This method efficiently captures time-based variations and dynamic interactions in malicious IoT traffic. The working of HLL is illustrated in Figure 3.

As we can see from Figure 3, the time-based spike-encoded states are taken as input by the **Hybrid Liquid Layer** (**HLL**), which produces paired and triplet HLL states at specific time steps, such as [2, 3], [3, 4], and so on for pairs, and [2, 3, 4], [3, 4, 3], and so on for triplets, across dynamic liquid states (e.g., Liquid1–Liquid8) using variable weights (e.g., 1.2, 0.8, etc.). This results in the computation of **Active States** (**AS**), representing immediate responses to shifting traffic behavior. **Passive States** (**PS**) are then derived by averaging recent AS values over a temporal window T, capturing sustained behavioral patterns. These AS and PS values are combined to form the final state $\mathbf{h}(t)$ in the form of paired and triplet states, such as 9.24, 12.80, and so on, enabling *LNNIDS*

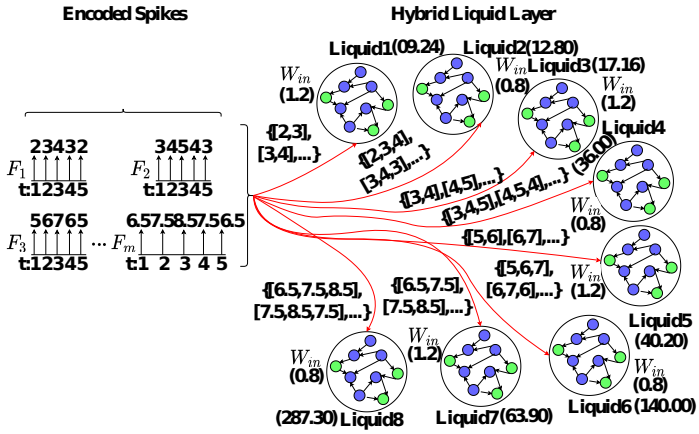


Fig. 3. Working example of hybrid liquid layer for the training of *LNNIDS* classifier.

to model evolving IoT threats, such as multistage Mirai flood attacks with high temporal sensitivity and adaptive precision, making it more effective than static attack detection methods. This dynamic modeling allows early detection of threat escalation and better differentiation between benign spikes and malicious bursts. It also enhances interpretability by clearly tracking how different temporal patterns influence detection outcomes.

(3) Attention Readout Layer Sub-Module

The input to this sub-module is the HLL state vector $h(t)$ (i.e., AS and PS), and the output is the attention-based state W_{Attn}, computed using both pairwise and triplet feature interactions. The use of pairwise and triplet interactions allows the model to capture complex, non-linear relationships between features, effectively automating feature engineering. This is crucial for accurately focusing attention on relevant flow behaviors and detecting sophisticated or previously unseen IoT attacks that simpler, isolated feature representations might miss. The attention readout layer maps HLL states to attention weights, dynamically amplifying important flow representations while down-weighting less informative ones. It performs two tasks: computing attention scores and refining HLL states using these scores. The attention mechanism assigns importance to different HLL states using the softmax function: $\alpha_P = \text{softmax}(W^P_{\text{attn}} h(t))$, $\alpha_T = \text{softmax}(W^T_{\text{attn}} h(t))$, where W^P_{attn} and W^T_{attn} are learnable weights mapping $h(t)$ to attention scores α_P and α_T. The softmax ensures these scores sum to one across the HLL states, forming a probability distribution. The attention scores then adjust the HLL states as follows: $W^P_{\text{Attn}} = W^P_{\text{State}}(\alpha_P \odot h(t))$, $W^T_{\text{Attn}} = W^T_{\text{State}}(\alpha_T \odot h(t))$, where element-wise multiplication ($\odot$) enhances pairwise and triplet features differently. The weighted states are then refined through layers with weights W^P_{State} and W^T_{State}, producing the final representations W^P_{Attn} and W^T_{Attn}. A decision rule is applied in Equation (4) to determine the most relevant attention-based state.

$$W_{\text{Attn}} = \begin{cases} W^P_{\text{Attn}}, & \text{if } \sum \alpha_P > \sum \alpha_T \text{ and } W^P_{\text{Attn}} > W^T_{\text{Attn}} \\ W^T_{\text{Attn}}, & \text{otherwise} \end{cases}, \tag{4}$$

where the total attention scores $\sum \alpha_P$ and $\sum \alpha_T$ are compared with their refined states W^P_{Attn} and W^T_{Attn}. If pairwise interactions contribute higher attention scores and stronger refined states, W^P_{Attn} is chosen; otherwise, triplet interactions W^T_{Attn} are selected. This is important for IoT attack classification, as triplet-based interactions capture higher-order dependencies, such as correlated traffic patterns in DDoS and scanning attacks, better than pairwise interactions. However, pairwise

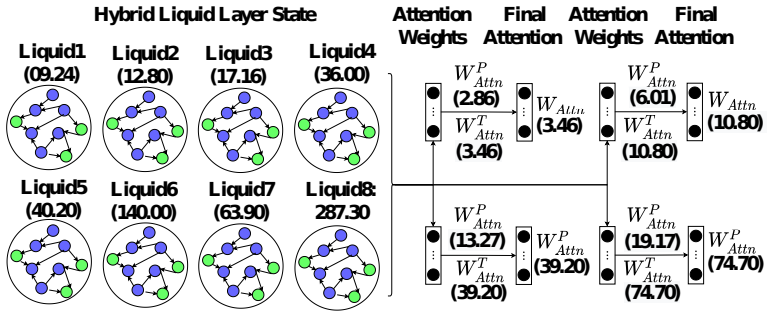


Fig. 4. Working example of adaptive attention layer for the training of *LNNIDS* classifier.

interactions are prioritized if they provide stronger distinguishing properties. This structured decision process helps *LNNIDS* balance pairwise and triplet contributions, improving classification accuracy while maintaining interpretability and efficiency. Additionally, the dynamic switching between pairwise and triplet attention enables the method to adapt to the nature of the observed traffic. For instance, localized traffic surges from a few sources (e.g., port scans) are better captured by pairwise attention, while distributed attack patterns (e.g., DDoS) require broader triplet-based context. This enables the attention mechanism to stay computationally efficient while remaining behaviorally aware. The working of the adaptive attention layer is illustrated in Figure 4.

As we can see from Figure 4, the input HLL state $h(t)$ includes values such as 09.24, 12.80, 17.16, up to 287.30 from Liquid1 to Liquid8. These are fed into both the pairwise and triplet attention readout layers. The softmax attention scores assign relative importance (e.g., 2.86, 3.46, 13.27, 74.70, etc.), which are then element-wise multiplied with the corresponding HLL states. The result is refined through separate layers for pairwise and triplet attention states, producing attention states like 10.80, 19.17, and 39.20. The decision rule in Equation (4) evaluates which stream (pairwise or triplet) provides stronger attention and selects it as the final output W_{Attn}. For example, if the triplet interaction from Liquid6–8 provides higher cumulative scores and refinement strength, it is selected to represent the dominant behavior. This mechanism ensures that critical multi-flow patterns (e.g., DDoS surges or coordinated scans) are prioritized without overburdening computation, making *LNNIDS* behaviorally aware and scalable.

(4) DYNGraph Sub-Module

The input to this sub-module is the attention-based states W_{Attn}, and the output is the trained DYNGraph, which classifies IoT attack traffic by dynamically modeling relationships between attention-based states and attack labels. The DYNGraph $G(V_A \cup V_B, E, \omega)$ consists of two node sets: $V_A = W_{\text{Attn}}$ (where each $v_i \in V_A$ represents an attention-based state) and V_B (predefined classes such as *DoS, Botnet, DDoS*, and *Benign*). Edges $E \subseteq V_A \times V_B$ connect states $v_i \in V_A$ to classes $c_j \in V_B$, with weights $\omega_i(e) \in \mathbb{R}$ (real numbers; $e \in E$) reflecting connection strengths derived from W_{Attn}. Each $v_i \in V_A$ initially forms its own cluster. During iterative refinement, nodes v_i are reassigned to candidate clusters $\mathcal{N}(i) \subseteq V_B$ (e.g., candidate classes like DoS $\in V_B$) by maximizing the gain $\Delta Q(i, C) = Q(\text{assign } v_i \to C) - Q(\text{current})$, where $C \in V_B$ denotes a candidate class. This process continues until loss minimization converges, at which point graph aggregation merges nodes within the same cluster into a single node. The result is an aggregated DYNGraph $G'(V'_A \cup V'_B, E', \omega')$, where: V'_A and V'_B represent merged nodes, Edge weights ω'_{c_p,c_q} quantify connection strengths between clusters $c_p \in V'_A$ and $c_q \in V'_B$. For example, $\omega'_{c_p,c_q} = 6$ indicates a strong connection from cluster c_p to c_q, while $\omega'_{c_p,c_q} = 4$ denotes a weaker one. Iterations proceed until improvements between phases become negligible. The final DYNGraph, defined in

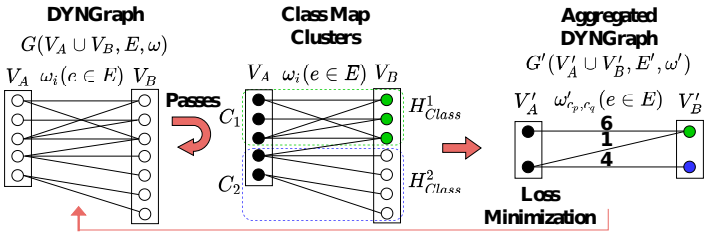


Fig. 5. Working example of *DYNGraph* for the training of *LNNIDS* classifier.

Equation (5), enables streamlined attack detection.

$$H_{\text{DYNGraph}} = W_{\text{final}}(H_{\text{class}} + H'_{\text{state}}), \tag{5}$$

where, H_{class} represents the final class, which is obtained by mapping the attention-based states W_{Attn} (i.e., V_A) to attack classes (V_B) via $H_{\text{class}} = (W_{A\to B} \odot M_{A\to B})^T W_{\text{Attn}}$, where $W_{A\to B}$ is a feature-to-class weight matrix and $M_{A\to B}$ is a connection filter retaining critical $V_A \to V_B$ links (e.g., edge weights $\omega_i(e)$). The refined feature state $H'_{\text{state}} = H_{\text{class}}\,(W_{B\to A} \odot M_{B\to A})$ projects H_{class} back into the feature space using class-to-feature weights $W_{B\to A}$ and feedback filter $M_{B\to A}$, mirroring iterative cluster refinement (via ΔQ) and aggregated weights ω'_{c_p,c_q}. These transformations dynamically optimize relationships between merged clusters, encapsulated in H_{DYNGraph}, which finalizes the DYNGraph for IoT attack classification.

Figure 5 shows the training process of *DYNGraph* for the *LNNIDS* Classifier. The minimization of loss through iterative refinement of $W_{A\to B}, M_{A\to B}, W_{B\to A}, M_{B\to A}$ ensures convergence when cluster improvements stabilize. This dynamic, bidirectional process enables *DYNGraph* to capture complex interdependencies and adapt to evolving IoT attack patterns without requiring manual thresholds. Unlike static classifiers, it dynamically refines attack predictions through feature-to-class interactions, ensuring interpretability while efficiently classifying IoT attack traffic. In IoT networks, attacks like stealthy scans or multistage Mirai-based floods often evolve in stages and overlap temporally, requiring flexible clustering to separate subtle transitions. *DYNGraph* leverages these evolving flow patterns to differentiate benign anomalies from coordinated, progressive threats with improved accuracy.

3.2 Testing Phase

In this phase, the trained *LNNIDS* classifies IoT attack traffic test flows through four modules: Flow Reconstruction, Feature Extraction, Feature Selection, and IoT Attack Traffic Classification. The first three work the same as in training, while the IoT Attack Traffic Classification module is explained in detail below.

3.2.1 IoT Attack Traffic Classification Module. The IoT Attack Traffic Classification module processes new attack traffic flows by first preprocessing them identically to the training phase to generate attention-based states W_{Attn}. These states are input to the pre-trained *DYNGraph*, which evaluates feature-to-class mappings via forward propagation and refines them using its bidirectional feedback mechanism, which is defined in Equation (6). The graph outputs predicted classes: known attacks as $\hat{y}$, and unknown attacks as $\tilde{y}$, where $\tilde{y} = \hat{y}$ if $\hat{y} \in T$ (known classes) or $\tilde{y} = U$ (unknown class) otherwise.

$$\hat{y} = \arg\max\big(H_{\text{DYNGraph}}\big), \tilde{y} = \begin{cases} \hat{y}, & \text{if } \hat{y} \in T, \\ U, & \text{if } \hat{y} \notin T, \end{cases} \tag{6}$$

To test new traffic, it is assigned to clusters in the aggregated graph $G'(V'_A \cup V'_B, E', \omega')$ using hierarchical class maps (H_1, H_2), and the cross-entropy loss Equation (7).

$$L = -\sum_{\substack{c_p \in V'_A \\ c_q \in V'_B}} P_{\text{true}}(c_p, c_q) \log P_{\text{pred}}(c_p, c_q), \tag{7}$$

where P_{true} is the true historical cluster distribution and P_{pred} is the predicted assignment, with $c_p \in V'_A$ and $c_q \in V'_B$ denoting merged clusters in G'. The loss L is compared with the threshold τ, which is selected by analyzing the distribution of loss values L as a predefined upper bound for acceptable consistency. This threshold τ is used for stable clustering and is defined as $\tau = \epsilon \cdot \max \Delta Q(i, C)$, where $\epsilon \in (0, 1)$ controls the tolerance. This adaptive threshold helps the clustering process follow meaningful structural improvements during graph optimization. The loss L is compared with the threshold τ, which is selected by analyzing the distribution of loss values L as a predefined upper bound for acceptable consistency. This threshold τ is used for stable clustering and is defined as $\tau = \epsilon \cdot \max \Delta Q(i, C)$, where $\epsilon \in (0, 1)$ controls the tolerance. This adaptive threshold helps the clustering process follow meaningful structural improvements during graph optimization. In scenarios involving IoT traffic, especially with evolving threats like stealthy port scans or low-rate Mirai floods, traditional fixed thresholds may misclassify transitions or early-stage behaviors. By using τ as a dynamic boundary, the method can separate flows that exhibit gradual deviation from known classes, facilitating the detection of emerging or zero-day attacks. Furthermore, this approach prevents overfitting by avoiding forced assignments when the structural similarity is weak, and enhances generalization by allowing the formation of new clusters only when statistically justified. This threshold balances sensitivity (i.e., detecting unknown attacks) and the retention of known patterns, thereby determining three cases:

- *Matched*: If $(L \leq \tau)$: Update edge weights $\omega'_{c_p,c_q} \leftarrow \omega'_{c_p,c_q} + \Delta\omega$ between clusters $c_p \in V'_A$ and $c_q \in V'_B$.
- *Not Matched*: If $(L > \tau)$: create new clusters (e.g., expand V'_A from 6 to 7 or V'_B from 4 to 5) and re-optimize G' shown in Figure 5.
- *Mismatched*: If ($L \leq \tau$ but violates constraints): Trigger anomaly detection $A = \|\omega'_{c_p,c_q} - \omega_{\text{expected}}\| > \tau$, where $c_p \in V'_A$, $c_q \in V'_B$, to flag deviations.

This process dynamically adapts G' to new attack traffic while preserving relationships between merged clusters (e.g., $\omega'_{c_p,c_q} = 6$ for strong connections).

4 Complexity Analysis

This section describes the asymptotic complexity analysis of *LNNIDS*. The complexity analysis of each module of *LNNIDS* is shown in Table 3.

Table 3 illustrates that the flow reconstruction module examines each packet header and assigns it to its corresponding flow. Given F flows and P packets, then complexity is $O(F \times P)$. The feature extraction module operates on all bidirectional flows F to extract features X, yielding $O(X \times F)$ complexity. The feature selection module selects X number of features using the DT method, takes $O(X \log F)$ time per flow, and requires $O(F \times X \log F)$ complexity. The feature encoding module transforms X' features into S-sized states at timestamp t, with complexity $O(X' \times S \times t)$. The HLL Layer captures temporal dependencies in $O(S \times t)$ per spiked state. The attention layer computes attention-based m-states, including pairwise and triplet interactions in j-pairs of liquid states of d-size with complexity $O(d \times 2j \times m)$. The computational complexity of *DYNGraph* is dominated by iterative refinement, leading to $O(i \times m \times T)$, where m is the number of attention readout states, T is the set of known attack classes, and i is the number of refinement iterations. Key contributors

Table 3. Module-Wise Complexity Analysis of *LNNIDS*

Module Name	Complexity	Description
Flow Reconstruction	$O(F \times P)$	F - the number of flows, P - the number of packets in the network trace.
Feature Extraction	$O(X \times F)$	X - the number of extracted features per flow F.
Feature Selection	$O(F \times X \log F)$	X - the number of extracted features per flow F using the decision method.
Feature Encoding	$O(X' \times S \times t)$	Converts the fixed number of selected features X' into S-sized states at a given timestamp t.
Hybrid Liquid Layer	$O(S \times t)$	s-sized spiked states at a given timestamp t.
Attention Layer	$O(d \times j \times m)$	Calculates attention over pairs of liquid states j (each of size d) and produces m attention readout states.
DYNGraph	$O(i \times m \times T)$	m is the number of attention readout states, T is the set of known attack classes, and i is the number of refinement iterations.
Classification	$O(i \times m \times T)$	Classifies test flows by extracting features ($O(X)$), inferring attention-based states via *DYNGraph* ($O(i \times m \times T)$), predicting class labels ($O(T)$), and checking for known/unknown classes ($O(1)$). Assigns traffic to clusters and computes cross-entropy loss ($O(m \times T)$), with re-optimization if new clusters are created.

include bipartite graph construction, graph aggregation, and matrix operations, all within $O(m \times T)$. The IoT Attack Traffic classification module processes test flows by extracting features $O(X)$, inferring attention-based states via *DYNGraph* $O(i \times m \times T)$, and predicting class labels $O(T)$. Classification checks ($O(1)$) and cluster assignments ($O(m \times T)$) involve computing cross-entropy loss. If necessary, new clusters are created, requiring graph re-optimization ($O(m \times T)$). The final complexity remains $O(i \times m \times T)$, ensuring efficient classification while adapting dynamically to unknown attacks.

5 Experiments and Results

This section describes the experiments conducted to assess the efficiency of *LNNIDS*. First, we discuss the dataset used in the experiments. Next, we provide evaluation outcomes and sensitivity analysis. Finally, we compare *LNNIDS* with six standard ML, DL, and seven state-of-the-art methods.

5.1 Dataset Description

In our experimental evaluation, we examined the effectiveness and scalability of our *LNNIDS* using two datasets. The first dataset, the ACI IoT dataset [2], consists of publicly available benign and attack IoT network traffic traces. Table 4 provides a detailed overview of this dataset.

As we can see from Table 4, for ease of understanding, we mapped all 11 IoT attacks to unique attack IDs ranging from A_{11} to A_{21}, along with benign traffic labeled as B_1. This dataset contains benign network traffic and attack traces from 21 IoT devices, resulting in 1,013,552 packets and 249,969 total bidirectional flows. The dataset size is 1,531.3 MB. It includes 11 attacks, categorized into four types: MITM Spoofing Attack, Brute Force Attack, DoS Attack, and Recon Attack, alongside benign network traces. The ACI IoT dataset captures early-stage IoT attack behaviors, including port sweeps and brute-force login attempts, which are commonly seen in smart home and office environments. It also includes low-frequency benign activity interspersed with sudden bursts of malicious traffic, offering a valuable contrast for dynamic detection models like *LNNIDS*. We also utilized a second dataset, the CIC IoT dataset 2023 [3], publicly available and described in detail in Table 5.

As we can see from Table 5, for ease of understanding, we mapped all 30 IoT attacks to unique attack IDs ranging from A_{22} to A_{51}, along with benign traffic labeled as B_2. This dataset comprises

Table 4. Public Dataset-1 Description

Attack Name	Attack ID	Attack Type	# of Packets	# of Total Flows	Size (in MB)
ARP Spoofing Attack	A_{11}	MITM Spoofing Attack	0076287	002352	0213.0
Dictionary Attack	A_{12}	Brute Force Attack	0028357	001669	0007.1
DNS Flood Attack	A_{13}	DoS Attack	0189629	070419	0022.9
Host Discovery Attack	A_{14}	Recon Attack	0010477	001213	0002.6
ICMP Flood Attack	A_{15}	DoS Attack	0057675	001234	0460.8
OSScan Attack	A_{16}	Recon Attack	0092975	048586	0008.8
Ping Sweep Attack	A_{17}	Recon Attack	0038404	014733	0005.3
Port Scan Attack	A_{18}	Recon Attack	0199018	096036	0017.7
Slowloris Attack	A_{19}	DoS Attack	0084031	009601	0016.9
SYN Flood Attack	A_{20}	DoS Attack	0084134	001036	0359.5
UDP Flood Attack	A_{21}	DoS Attack	0096062	001086	0257.3
Benign	B_1	Benign	0056503	002004	0159.4
Total	-	-	**1013552**	**249969**	**1531.3**

Table 5. Public Dataset-2 Description

Attack Name	Attack ID	Attack Type	# of Packets	# of Total Flows	Size (in MB)
ACK Fragmentation Attack	A_{22}	DDoS Attack	002235814	01040923	01900.00
ARP Spoofing Attack	A_{23}	Spoofing Attack	000236216	00002796	00237.00
Backdoor Malware Attack	A_{24}	Web-based Attack	000033414	00003135	00010.60
Browser Hijacking Attack	A_{25}	Web-based Attack	000059821	00003656	00034.40
Command Injection Attack	A_{26}	Web-based Attack	000055741	00004148	00028.10
Dictionary Attack	A_{27}	Brute Force Attack	000133138	00008795	00039.50
DNS Spoofing Attack	A_{28}	Spoofing Attack	001812557	00096375	00926.00
Greeth Flood Attack	A_{29}	Mirai Attack	003403855	00011290	01900.00
GREIP Flood Attack	A_{30}	Mirai Attack	003506214	00008410	01900.00
Host Discovery Attack	A_{31}	Recon Attack	001371112	00440124	00239.00
HTTP Flood Attack	A_{32}	DDoS Attack	002881005	00076911	00630.00
HTTP Flood Attack	A_{33}	DoS Attack	003114983	00152094	01380.00
ICMP Flood Attack	A_{34}	DDoS Attack	026883503	00007339	01900.00
ICMP Fragmentation Attack	A_{35}	DDoS Attack	002248644	00017348	01900.00
OS Scan Attack	A_{36}	Recon Attack	000992241	00176153	00325.00
Ping Sweep Attack	A_{37}	Recon Attack	000022943	00001634	00007.87
RSTFIN Flood Attack	A_{38}	DDoS Attack	026530267	06355509	01900.00
SlowLoris Attack	A_{39}	DDoS Attack	002351454	00142834	00683.00
SQL Injection Attack	A_{40}	Web-based Attack	000053462	00006101	00010.20
SYN Flood Attack	A_{41}	DDoS Attack	026038853	09594908	01900.00
SYN Flood Attack	A_{42}	DoS Attack	026405653	01461527	01900.00
SynonymousIP Flood Attack	A_{43}	DDoS Attack	026824105	01001136	01900.00
TCP Flood Attack	A_{44}	DDoS Attack	026729431	06641102	01900.00
UDP Flood Attack	A_{45}	DDoS Attack	026678524	00040158	01900.00
UDP Flood Attack	A_{46}	DoS Attack	014177342	00954900	01900.00
UDP Fragmentation Attack	A_{47}	DDoS Attack	002839217	00012491	01900.00
UDPPlain Attack	A_{48}	Mirai Attack	003637476	00006167	01900.00
Uploading Attack	A_{49}	Web-based Attack	000012939	00001373	00003.38
Vulnerability Scan Attack	A_{50}	Recon Attack	003802533	00448188	01010.00
XSS Attack	A_{51}	Web-based Attack	000040183	00003438	00012.60
Benign	B_2	Benign	002988642	00095145	01900.00
Total	-	-	**238101282**	**28816108**	**34076.65**

Table 6. Dataset Division for Public Dataset-1 (Known vs. Unknown Attacks)

Attack ID (Known)	Training Flows	Attack ID (Unknown)	Training Flows	Testing Flows
A_{11}	470	U_1	-	1,882
A_{12}	334	-	334	1,335
A_{13}	14,084	-	14,084	56,335
A_{14}	243	-	243	970
A_{15}	247	-	247	987
A_{16}	9,717	-	9,717	38,869
A_{17}	2,946	-	2,946	11,787
A_{18}	19,207	U_2	-	76,829
A_{19}	1,920	U_3	-	7,681
A_{20}	207	-	207	829
A_{21}	217	-	217	869
B_1	401	-	401	1,603
Total	**49993**	-	**28,396**	**199,976**

benign and attack network traffic from 105 IoT devices, resulting in 290,433,658 packets and 37,834,052 total bidirectional flows, including both benign and malicious traffic. The dataset size is 38,081.65 MB. It contains 30 types of attacks, categorized into seven types: DDoS Attacks, DoS Attacks, Spoofing Attacks, Web-Based Attacks, Brute Force Attacks, Mirai Attacks, and Recon Attacks, alongside benign network traces. Among these, Mirai-based attacks are particularly prominent and simulate realistic botnet behaviors, involving infected IoT devices that generate periodic high-rate traffic spikes. Additionally, the dataset contains multistage attack patterns where reconnaissance activities are followed by coordinated exploitation, making it valuable for evaluating systems under progressive and evolving threat scenarios.

5.2 Evaluation of *LNNIDS*

This subsection describes the experiments conducted to evaluate the performance of *LNNIDS*. The *LNNIDS* is implemented using Python [5], and flow-based features are extracted from the network trace using CICFlowMeter [4]. For each dataset, 20% of the flows from each IoT attack class are used for training, while the remaining 80% are reserved for testing. The dataset divisions are presented in Tables 6 and 7 for Public Dataset-1 and Public Dataset-2, respectively.

As we can see from Table 6, in Public Dataset-1, unknown IoT attacks are assigned unique identifiers ranging from U_1 to U_3. The dataset includes 28,396 bidirectional training flows excluding unknown attacks, while the testing flows contain these unseen attack types for evaluation.

Similarly, as shown in Table 7, Public Dataset-2 assigns unknown attack identifiers from U_4 to U_7, with 5,701,785 bidirectional flows used for training (excluding the unknowns), and the corresponding testing set includes the remaining flows associated with these unknown attack classes. This structured labeling allows for consistent evaluation of *LNNIDS*'s capability to detect previously unseen IoT threats in both datasets.

5.2.1 Feature Selection. We generate bidirectional flows from benign and attack IoT traces using CICFlowMeter, extracting 79 features. A wrapper-based method then selects the most significant features for both datasets (see Tables 8 and 9).

As we can observe from Table 8, these feature importance scores are computed using a DT within the wrapper method. We evaluate *LNNIDS* with feature sets of 10, 20, 30, 40, 50, 60, 70, and all features. Feature selection enhances both detection accuracy and computational efficiency. The top 10 features capture core network behaviors (e.g., byte/packet counts and flow duration) for lightweight detection of unknown attacks. Increasing to 20 features adds flow statistics and

Table 7. Dataset Division for Public Dataset-2 (Known vs. Unknown Attacks)

Attack ID (Known)	Training Flows	Attack ID (Unknown)	Training Flows	Testing Flows
A_{22}	0208185	–	0208185	00832738
A_{23}	0000559	–	0000559	00002237
A_{24}	0000627	–	0000627	00002508
A_{25}	0000731	–	0000731	00002925
A_{26}	0000830	–	0000830	00003318
A_{27}	0001759	–	0001759	00007036
A_{28}	0019275	–	0019275	00077100
A_{29}	0002258	–	0002258	00009032
A_{30}	0001682	–	0001682	00006728
A_{31}	0088025	–	0088025	00352099
A_{32}	0015382	–	0015382	00061529
A_{33}	0030419	U_4	-	00121675
A_{34}	0001468	–	0001468	00005871
A_{35}	0003470	–	0003470	00013878
A_{36}	0035231	–	0035231	00140922
A_{37}	0000327	–	0000327	00001307
A_{38}	1271102	–	1271102	05084407
A_{39}	0028567	U_5	-	000114267
A_{40}	0001220	U_6	-	00004881
A_{41}	1918982	–	1918982	07675926
A_{42}	0292305	–	0292305	01169222
A_{43}	0200227	–	0200227	00800909
A_{44}	1328220	–	1328220	05312882
A_{45}	0008032	–	0008032	00032126
A_{46}	0190980	–	0190980	00763920
A_{47}	0002498	–	0002498	00009993
A_{48}	0001233	U_7	-	00004934
A_{49}	0000275	–	0000275	00001098
A_{50}	0089638	–	0089638	00358550
A_{51}	0000688	–	0000688	00002750
B_2	0019029	–	0019029	00076116
Total	**5,763,224**	-	**5,701,785**	**23,052,884**

TCP/IP flag indicators, while the top 30 features enhance behavioral analysis against sophisticated attacks, such as Mirai botnet infections. Larger sets (40–70) enhance the detection of stealthy attacks via segment size, port-based attributes, and rate variations, and using all features aids zero-day attack detection at a higher computational cost. Using small flow sets and an attention method reduces complexity, making this approach ideal for benchmarking and ensuring that *LNNIDS* remains scalable and effective. In addition to the wrapper method, we also evaluated filter and embedded methods for feature selection across both datasets. Filter methods rank features using statistical metrics, such as information gain or correlation, but often overlook deeper interactions between features and the model, resulting in weaker selections in complex IoT traffic. Embedded (LASSO) methods are efficient but biased to specific models and miss cross-feature dependencies [7, 42]. In contrast, our wrapper method, which combines clustering with DT-based evaluation, effectively captures structural patterns and non-linear dependencies within attack flows, making it more suitable for dynamic IoT threats. This is reflected in the higher importance scores achieved in Table 8. Analysis of attacks such as ARP spoofing and slow DDoS reveals that low-frequency

Table 8. Feature Importance Scores for Public Dataset-1

Feature	Wrapper	Filter	Embedded	Feature	Wrapper	Filter	Embedded
X_1	0.997143633	0.875673262	0.842409762	X_{41}	0.971736495	0.864844923	0.831581423
X_2	0.992347012	0.875555349	0.842291849	X_{42}	0.971736495	0.864844923	0.831581423
X_3	0.991814928	0.874356567	0.841093067	X_{43}	0.971736495	0.864844923	0.831581423
X_4	0.990962853	0.874356567	0.841093067	X_{44}	0.971004325	0.864844923	0.831581423
X_5	0.990702787	0.873904567	0.840641067	X_{45}	0.970988198	0.864844923	0.831581423
X_6	0.988890551	0.873865263	0.840601763	X_{46}	0.970988198	0.864844923	0.831581423
X_7	0.988798540	0.873688394	0.840424894	X_{47}	0.970912179	0.864844923	0.831581423
X_8	0.988218469	0.873668742	0.840405242	X_{48}	0.970912179	0.864844923	0.831581423
X_9	0.987178324	0.872627177	0.839363677	X_{49}	0.970812188	0.864844923	0.831581423
X_{10}	0.984149962	0.872587873	0.839324373	X_{50}	0.970312296	0.864844923	0.831581423
X_{11}	0.982845801	0.872076917	0.838813417	X_{51}	0.970312296	0.864844923	0.831581423
X_{12}	0.981689646	0.871900047	0.838636547	X_{52}	0.970236286	0.864844923	0.831581423
X_{13}	0.981529571	0.870524396	0.837260896	X_{53}	0.968723925	0.864844923	0.831581423
X_{14}	0.980217380	0.870524396	0.837260896	X_{54}	0.967399645	0.864844923	0.831581423
X_{15}	0.979085311	0.869836571	0.836573071	X_{55}	0.967399645	0.864844923	0.831581423
X_{16}	0.978289202	0.86873605	0.83547255	X_{56}	0.967063672	0.864844923	0.831581423
X_{17}	0.977841145	0.868696746	0.835433246	X_{57}	0.966847753	0.864844923	0.831581423
X_{18}	0.976972964	0.868696746	0.835433246	X_{58}	0.966739633	0.864844923	0.831581423
X_{19}	0.976972964	0.867851703	0.834588203	X_{59}	0.966739633	0.864844923	0.831581423
X_{20}	0.976721013	0.867832051	0.834568551	X_{60}	0.966099609	0.864844923	0.831581423
X_{21}	0.974640807	0.867832051	0.834568551	X_{61}	0.964611394	0.864844923	0.831581423
X_{22}	0.974036687	0.867832051	0.834568551	X_{62}	0.964611394	0.864844923	0.831581423
X_{23}	0.973384624	0.867753442	0.834489942	X_{63}	0.963811335	0.864844923	0.831581423
X_{24}	0.973124545	0.867753442	0.834489942	X_{64}	0.963747189	0.864844923	0.831581423
X_{25}	0.972844580	0.867714138	0.834450638	X_{65}	0.963735188	0.864844923	0.831581423
X_{26}	0.972500557	0.867281791	0.834018291	X_{66}	0.962839184	0.864844923	0.831581423
X_{27}	0.972228467	0.867026313	0.833762813	X_{67}	0.962391055	0.864844923	0.831581423
X_{28}	0.972200346	0.867026313	0.833762813	X_{68}	0.962187031	0.864844923	0.831581423
X_{29}	0.972120361	0.866652922	0.833389422	X_{69}	0.961350894	0.864844923	0.831581423
X_{30}	0.971736495	0.866535009	0.833271509	X_{70}	0.960826868	0.864844923	0.831581423
X_{31}	0.971736495	0.866436748	0.833173248	X_{71}	0.960582915	0.864844923	0.831581423
X_{32}	0.971736495	0.866299183	0.833035683	X_{72}	0.960022750	0.864844923	0.831581423
X_{33}	0.971736495	0.866161618	0.832898118	X_{73}	0.960022750	0.864844923	0.831581423
X_{34}	0.971736495	0.865552401	0.832288901	X_{74}	0.960010940	0.864844923	0.831581423
X_{35}	0.971736495	0.865552401	0.832288901	X_{75}	0.959554734	0.864707358	0.831443858
X_{36}	0.971736495	0.865473792	0.832210292	X_{76}	0.959358762	0.864255358	0.830991858
X_{37}	0.971736495	0.865002140	0.831738640	X_{77}	0.956114299	0.860364231	0.827100731
X_{38}	0.971736495	0.864884227	0.831620727	X_{78}	0.953541957	0.860265971	0.827002471
X_{39}	0.971736495	0.864884227	0.831620727	X_{79}	0.953389921	0.860265971	0.827002471
X_{40}	0.971736495	0.864844923	0.831581423				

features (e.g., `urg_flag_cnt`, `min_pkt_len`) gain value when combined with others. Similarly, Mirai's fluctuating TCP flag behaviors are best detected through compound features captured by the wrapper approach. In edge-constrained IoT setups, selecting only the most structurally diverse features enables *LNNIDS* to remain lightweight while maintaining robust detection. Similar results are observed for Table 9.

5.2.2 Hyper-Parameter Tuning. We identify nine hyperparameters of *LNNIDS* that must be tuned for optimal performance on both datasets. These hyperparameters are shown in Table 10.

Table 9. Feature Importance Scores for Public Dataset-2

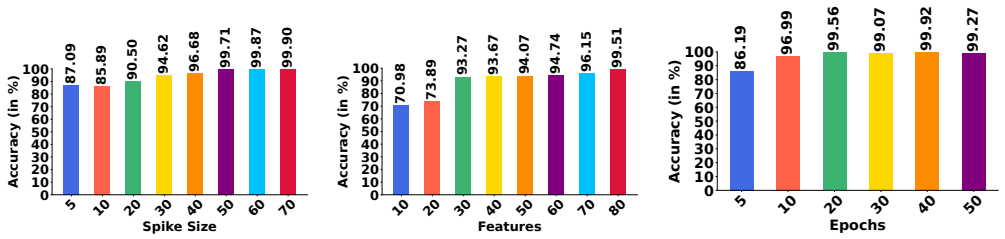
Feature	Wrapper	Filter	Embedded	Feature	Wrapper	Filter	Embedded
X_{18}	0.900931922	0.777410922	0.744147422	X_{19}	0.900931922	0.777410922	0.744147422
X_{56}	0.882461523	0.758940523	0.725677023	X_{4}	0.870541565	0.747020565	0.713757065
X_{57}	0.869934591	0.746413591	0.713150091	X_{1}	0.867751216	0.744230216	0.710966716
X_{73}	0.865937524	0.742416524	0.709153024	X_{72}	0.865937524	0.742416524	0.709153024
X_{22}	0.836547679	0.713026679	0.679763179	X_{16}	0.830793868	0.707272868	0.674009368
X_{7}	0.827645571	0.704124571	0.670861071	X_{11}	0.826638068	0.703117068	0.669853568
X_{9}	0.825874609	0.702353609	0.669090109	X_{3}	0.822897038	0.699376038	0.666112538
X_{2}	0.822498721	0.698977721	0.665714221	X_{23}	0.822173943	0.698652943	0.665389443
X_{8}	0.819886216	0.696365216	0.663101716	X_{28}	0.815433978	0.691912978	0.658649478
X_{6}	0.814604188	0.691083188	0.657819688	X_{5}	0.814509350	0.690988350	0.657724850
X_{44}	0.812707631	0.689186631	0.655923131	X_{20}	0.812359159	0.688838159	0.655574659
X_{10}	0.810507585	0.686986585	0.653723085	X_{14}	0.808840859	0.685319859	0.652056359
X_{12}	0.804440902	0.680919902	0.647656402	X_{14}	0.803423780	0.679902780	0.646639280
X_{8}	0.798900495	0.675379495	0.642115995	X_{15}	0.798793724	0.675272724	0.642009224
X_{17}	0.795666721	0.672145721	0.638882221	X_{45}	0.790972767	0.667451767	0.634188267
X_{46}	0.790972767	0.667451767	0.634188267	X_{24}	0.790230706	0.666709706	0.633446206
X_{27}	0.785071957	0.661550957	0.628287457	X_{78}	0.782596956	0.659075956	0.625812456
X_{60}	0.780029437	0.656508437	0.623244937	X_{79}	0.776556317	0.653035317	0.619771817
X_{77}	0.775487115	0.651966115	0.618702615	X_{71}	0.772507103	0.648986103	0.615722603
X_{63}	0.771537490	0.648016490	0.614752990	X_{64}	0.770525170	0.647004170	0.613740670
X_{68}	0.770155338	0.646634338	0.613370838	X_{51}	0.767495383	0.643974383	0.610710883
X_{76}	0.766016060	0.642495060	0.609231560	X_{37}	0.765572733	0.642051733	0.608788233
X_{41}	0.765572733	0.642051733	0.608788233	X_{42}	0.765572733	0.642051733	0.608788233
X_{32}	0.765572733	0.642051733	0.608788233	X_{39}	0.765572733	0.642051733	0.608788233
X_{38}	0.765572733	0.642051733	0.608788233	X_{35}	0.765572733	0.642051733	0.608788233
X_{36}	0.765572733	0.642051733	0.608788233	X_{43}	0.765572733	0.642051733	0.608788233
X_{33}	0.765572733	0.642051733	0.608788233	X_{30}	0.765572733	0.642051733	0.608788233
X_{31}	0.765572733	0.642051733	0.608788233	X_{34}	0.765572733	0.642051733	0.608788233
X_{40}	0.765572733	0.642051733	0.608788233	X_{52}	0.765572731	0.642051731	0.608788231
X_{50}	0.763465148	0.639944148	0.606680648	X_{61}	0.762085361	0.638564361	0.605300861
X_{62}	0.762085361	0.638564361	0.605300861	X_{55}	0.761634918	0.638113918	0.604850418
X_{54}	0.761634918	0.638113918	0.604850418	X_{21}	0.761535359	0.638014359	0.604750859
X_{58}	0.760169806	0.636648806	0.603385306	X_{59}	0.760169806	0.636648806	0.603385306
X_{29}	0.759088754	0.635567754	0.602304254	X_{66}	0.757628387	0.634107387	0.600843887
X_{64}	0.750698719	0.627177719	0.593914219	X_{65}	0.750691607	0.627170607	0.593907107
X_{49}	0.750250638	0.626729638	0.593466138	X_{75}	0.748240281	0.624719281	0.591455781
X_{48}	0.747628587	0.624107587	0.590844087	X_{47}	0.747628587	0.624107587	0.590844087
X_{25}	0.745871919	0.622350919	0.589087419	X_{74}	0.731901204	0.608380204	0.575116704
X_{21}	0.729748520	0.606227520	0.572964020	X_{56}	0.723466107	0.599945107	0.566681607
X_{69}	0.714307952	0.590786952	0.557523452				

As shown in Table 10, the hyperparameters, their corresponding parameter ranges, and the optimal values selected for each are clearly presented. These optimal configurations enabled fast and accurate IoT attack classification by varying spike size, selected features, and epochs (with other parameters fixed). The best parameters are shown in Figure 6.

As shown in Figure 6(a), we used a spike size of 50 to encode selected features and generate HLL states. Accuracy improves as spike size increases, peaking at size 50 with 99.71% accuracy. Beyond this, computational cost increases with minimal gain. A smaller spike size of 5 leads to poor learning (87.09%) due to coarse temporal resolution, which obscures rapid attack bursts often seen

Table 10. Hyper-Parameter Tuning for *LNNIDS*

Sr. No.	Hyper-parameter	Tuned Value	Parameter Ranges
1	Spike Size	50	0 - Input Size
2	Timestamp (t)	5 sec	1 sec - 5 sec
3	Batch Size	32	32 - 512
4	Epochs	20	0 - 50
5	Optimizer	Adam	SGD, Adam, RMSProp, ADAGRAD
6	Learning Rate	0.001	0.001 - 0.1
7	Loss Function	Entropy Loss	Entropy Loss, MSE Loss, MAE Loss, KLD Loss
8	Edge Map	$\geqslant$ 50%	0 - 100%
9	Node Map	$\geqslant$ 70%	0 - 100%



(a) Spike Size Variations (b) Feature Variations (c) Epoch Variations

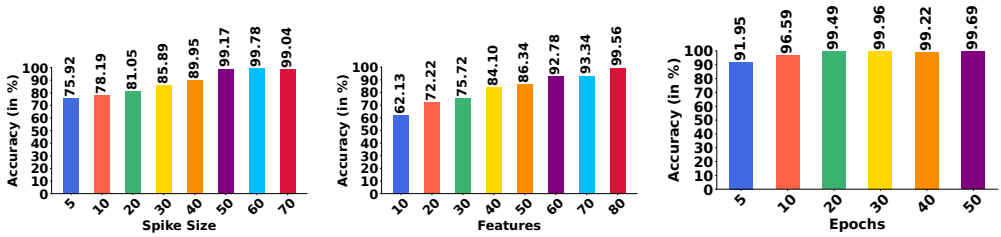
Fig. 6. Hyper-parameter tuning of spike size, features, and epochs for *LNNIDS* on public dataset-1.

in DDoS events. In contrast, a spike size of 50 preserves fine-grained temporal behavior, capturing intermittent malicious traffic. As we can see from Figure 6(b), accuracy also improves with more features, peaking at 79 features. Using only 10 features reduces accuracy to 70.98% due to the loss of critical flow-level indicators, such as packet duration, inter-arrival times, and protocol-specific flags, which are essential for detecting stealthy or low-variance threats. The full feature set captures complex statistical, protocol, and behavioral patterns, ensuring effective generalization across diverse IoT attacks. As shown in Figure 6(c), increasing the number of training epochs improves accuracy, with optimal performance achieved at 20 epochs. Early training enables the model to learn slow-and-low attack behaviors, whereas further training beyond this point leads to diminishing gains and an increased risk of overfitting–memorizing noise, such as benign firmware update traffic. These findings confirm that tuning spike size, feature count, and epochs is critical for balancing detection precision, resource efficiency, and robustness in real-world IoT attack scenarios. Similar experiments on Public Dataset-2 (see Figure 7).

5.2.3 Performance Analysis. After fine-tuning and training *LNNIDS* on a small training set, we evaluated its performance on the test sets of both datasets to detect known and unknown attacks in IoT networks. The detection results for both known and unknown IoT attacks are detailed below.

— *Detection Results for Known Attacks:*
The results are summarized using a confusion matrix for known IoT attacks, and the confusion matrix for Public Dataset-1 is displayed in Table 11.

As we can observe from Table 11, the first column represents the flow-based patterns of each attack on the IoT device's network, identified by its attack ID and benign flows, while the first row shows the original flows associated with each attack and benign flows, each characterized by its attack ID. For example, consider the IoT attack A_{11}, which contains 1,882 flows; among these, 1,796 flows are correctly matched with the flow-based patterns



(a) Spike Size Variations (b) Feature Variations (c) Epoch Variations

Fig. 7. Hyper-parameter tuning of spike size, features, and epochs for *LNNIDS* on public dataset-2.

Table 11. Confusion Matrix for Evaluation of *LNNIDS* on Test Flows of Public Dataset-1 for Known Attacks

	A_{11}	A_{12}	A_{13}	A_{14}	A_{15}	A_{16}	A_{17}	A_{18}	A_{19}	A_{20}	A_{21}	B_1
A_{11} (1882)	1796	22	0	0	0	0	0	0	1	0	0	63
A_{12} (1335)	0	1245	0	0	0	0	0	0	1	0	1	88
A_{13} (56335)	0	0	56213	0	110	0	0	0	0	0	0	12
A_{14} (970)	0	0	0	875	0	90	4	0	1	0	0	0
A_{15} (987)	0	0	25	0	912	0	0	0	50	0	0	0
A_{16} (38869)	0	0	0	5	0	38771	90	3	0	0	0	0
A_{17} (11787)	0	0	0	0	0	15	11739	32	0	0	0	1
A_{18} (76829)	0	0	0	0	0	0	34	76785	0	0	0	10
A_{19} (7681)	0	0	0	0	21	0	0	0	7637	23	0	0
A_{20} (829)	0	0	0	0	0	0	0	0	9	783	37	0
A_{21} (869)	38	2	0	0	0	0	0	0	1	35	793	0
B_1 (1603)	5	1	37	0	0	0	0	40	8	0	0	1512

representing A_{11}, indicating that 95.4% of the attack flows are accurately classified. However, as highlighted in red, some flows are incorrectly matched with the flow-based patterns of IoT attacks A_{12}, A_{19}, and A_{21}, and no flows remain unclassified. Furthermore, a few flows from each IoT attack are mistakenly classified as belonging to other attacks. This misclassification occurs primarily for two reasons: first, overlapping attack flow patterns increase due to the involvement of multiple terminated flows, which contribute to the misclassification; and second, misclassification arises when more than half of the attention-based states exhibit similar patterns. Similar experiments were conducted using Public Dataset-2, and the results are presented as confusion matrices in Tables 12.

— *Detection Results for Unknown Attacks:*
The results are summarized using a confusion matrix for Unknown IoT attacks, and the confusion matrix for Public Dataset-1 is displayed in Table 13.

As shown in Table 13, the first column lists the learned flow-based patterns for each IoT attack and benign flow, identified by their corresponding attack IDs, including unknown attack IDs. The first row displays the original flows associated with each attack category. This layout effectively compares the raw input data with the patterns learned by *LNNIDS*, facilitating an understanding of how this method interprets IoT network attack behavior. Notably, the row labeled U_1 corresponds to unknown IoT attacks not used during *LNNIDS*'s training phase. Despite never encountering these attack flows before, *LNNIDS* correctly classified 1,796 out of 1,882 flows (approximately 95.4%), with only a few misclassifications, highlighted

Table 12. Confusion Matrix for Evaluation of LNNIDS on Test Flows of Public Dataset-2 for Known Attacks

	A_{22}	A_{23}	A_{24}	A_{25}	A_{26}	A_{27}	A_{28}	A_{29}	A_{30}	A_{31}	A_{32}	A_{33}	A_{34}	A_{35}	A_{36}	A_{37}	A_{38}	A_{39}	A_{40}	A_{41}	A_{42}	A_{43}	A_{44}	A_{45}	A_{46}	A_{47}	A_{48}	A_{49}	A_{50}	A_{51}	A_{52}
A_{22} (832738)	824359	1780	0	0	0	0	0	1000	1500	2090	0	2007	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0
A_{23} (2237)	5	1971	0	2	0	0	0	121	20	55	0	51	0	0	0	3	0	0	0	4	0	1	0	3	0	0	0	0	0	0	1
A_{24} (2508)	0	0	2202	0	173	0	97	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	5	30	0	0	0	0	0	0	0
A_{25} (2925)	0	0	0	2760	0	0	0	0	161	1	0	2	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A_{26} (3318)	0	0	416	0	2372	0	55	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	7	465	0	0	0	1	0	0	0
A_{27} (7036)	0	0	0	0	0	6720	0	0	0	0	0	0	176	1	0	0	0	0	0	0	0	43	0	0	0	3	2	0	0	1	90
A_{28} (77100)	0	0	82	0	4	0	76570	0	0	0	0	0	0	1	5	0	0	0	13	0	0	0	0	425	0	0	0	0	0	0	0
A_{29} (9032)	8	200	0	2	0	0	0	8526	7	19	0	27	3	0	0	89	0	0	0	17	0	1	0	0	0	133	0	0	0	0	0
A_{30} (6728)	10	0	17	105	0	0	0	1	6410	11	0	170	2	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
A_{31} (352099)	1	419	0	3	0	0	0	2612	3360	345673	0	31	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A_{32} (61529)	0	0	0	0	0	0	4	0	0	0	59279	1680	0	1	56	0	0	50	5	0	41	0	0	1	0	0	0	405	7	0	0
A_{33} (121675)	16	78	0	5	0	0	0	202	1512	26	1200	118603	1	0	0	1	0	0	0	30	0	0	0	0	0	1	0	0	0	0	0
A_{34} (5871)	0	0	0	0	0	208	0	0	0	0	0	0	5361	1	0	1	0	0	0	0	0	64	0	0	0	12	224	0	0	0	0
A_{35} (13878)	0	0	0	0	0	0	0	0	0	0	3	1	0	12795	0	0	0	10	0	0	2	0	0	0	0	1	0	0	630	436	0
A_{36} (140922)	0	0	0	0	220	0	250	0	0	0	590	0	0	2	132854	1500	0	1360	0	0	200	1450	0	250	0	0	0	0	2000	246	0
A_{37} (4412965)	296325	1	0	0	0	0	0	7226	5756	0	0	104536	4647	10809	123715	3608367	0	0	0	68135	52056	0	0	54569	47289	8113	2156	0	0	0	22265
A_{38} (5084407)	1546	676	0	0	0	0	0	2360	0	0	0	1456	0	0	0	0	5078369	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A_{39} (114267)	0	0	0	0	0	0	7	0	0	0	114	0	1	0	16	0	0	110039	0	0	1250	0	0	0	0	0	0	1480	1360	0	0
A_{40} (4881)	0	0	1	0	4	0	175	0	0	0	0	0	0	0	91	0	0	1	4598	0	1	0	0	2	0	0	0	8	0	0	0
A_{41} (7675926)	2133	1445	0	0	0	0	0	8450	0	0	0	12023	0	0	0	0	0	0	0	7651459	0	0	0	0	0	0	0	0	0	0	416
A_{42} (1169222)	1120	0	0	0	0	0	0	0	0	0	1520	0	0	0	1540	0	0	1333	100	2454	1159138	0	0	1136	0	0	0	601	0	0	280
A_{43} (800909)	0	0	0	0	0	2044	0	0	0	0	0	0	2049	0	0	0	0	0	0	0	0	796254	0	0	0	0	0	0	0	0	562
A_{44} (5312882)	9156	0	0	0	0	0	0	0	0	0	550	450	0	0	0	0	0	0	0	8561	7059	0	5286318	80	0	0	0	0	0	0	708
A_{45} (2643463)	0	0	2	0	16	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	2639002	2256	1145	945	0	0	0	88
A_{46} (763920)	0	0	0	0	0	0	0	0	0	0	0	1	1114	0	0	0	0	0	0	0	0	0	0	2514	752318	1919	2417	0	0	2170	1467
A_{47} (9993)	1	7	0	1	0	1	0	235	3	1	0	2	14	0	0	129	1	0	0	0	0	0	0	0	0	9557	14	0	0	0	0
A_{48} (4934)	0	1	0	0	0	39	0	2	0	0	0	0	296	1	0	1	0	0	0	0	0	22	0	0	0	631	3940	0	0	1	0
A_{49} (1098)	0	0	0	0	0	0	0	0	0	33	0	0	0	53	0	0	36	0	0	0	0	0	0	0	0	0	0	976	0	0	0
A_{50} (358550)	0	0	0	0	0	0	1	819	0	0	1280	0	0	1750	714	0	0	2077	0	0	0	0	0	0	0	0	0	1	351536	0	372
A_{51} (2750)	0	0	0	0	0	0	0	0	0	0	0	0	0	62	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2680	8
B_2 (76116)	0	653	0	0	0	667	0	0	0	0	0	0	200	280	0	0	0	0	0	0	0	0	0	0	701	420	0	0	0	1789	71406

Table 13. Confusion Matrix for Evaluation of *LNNIDS* on Test Flows of Public Dataset-1 for Unknown Attacks

	U_1	A_{12}	A_{13}	A_{14}	A_{15}	A_{16}	A_{17}	U_2	U_3	A_{20}	A_{21}	B_1
U_1 (1882)	1753	50	0	0	0	0	0	0	6	5	67	1
A_{12} (1335)	86	1239	0	0	0	0	0	2	5	1	1	1
A_{13} (56335)	0	0	56098	0	85	0	0	5	8	0	0	139
A_{14} (970)	0	0	0	930	0	30	9	0	0	0	0	1
A_{15} (987)	0	0	45	0	866	0	0	0	72	1	1	2
A_{16} (38869)	0	0	0	213	0	38516	139	0	0	0	0	1
A_{17} (11787)	0	0	0	24	0	59	11628	76	0	0	0	0
U_2 (76829)	0	0	2	1	0	17	131	76547	0	0	0	131
U_3 (7681)	1	9	0	0	87	0	0	17	7523	40	4	0
A_{20} (829)	16	1	0	0	2	0	0	0	86	716	8	0
A_{21} (869)	71	6	0	0	0	0	0	0	16	7	769	0
B_1 (1603)	6	0	95	0	2	2	14	17	1	0	0	1466

in red, being erroneously assigned to attack IDs such as A_{12} and B_1. This high accuracy for U_1 underscores *LNNIDS*'s strong generalization capability in identifying patterns even in previously unseen attack scenarios. Moreover, the new weight configurations for unknown attack patterns do not match the attention weights associated with the trained known attack patterns. This deviation indicates that the new test traffic differs significantly from the known traffic, leading to the formation of a new node in the DYNGraph labeled "Unknown Attack1." If multiple unknown attack patterns exist, they are subsequently labeled as "Unknown Attack2," "Unknown Attack3," and so forth. Similar behavior patterns have been observed across various unknown attack classes, highlighting the inherent challenges in accurately detecting novel attacks. Additionally, detecting IoT attacks remains challenging because certain attack patterns closely resemble normal traffic or other legitimate activities, resulting in frequent classification errors. Experiments on Public Dataset-2 (see Table 14) further confirm these findings, thereby strengthening the evidence for *LNNIDS*'s robustness across diverse datasets. Collectively, these observations validate the effectiveness of our approach and emphasize its potential for real-world deployment in dynamic and evolving IoT environments.

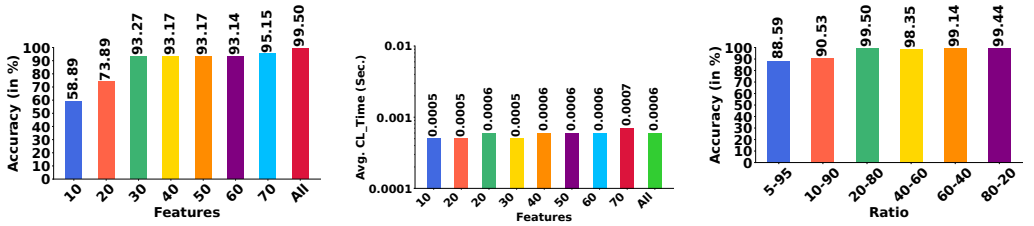
5.3 Sensitivity Analysis

In this subsection, we analyze how *LNNIDS* responds to different feature selections and flow distributions. We test the method using various top-ranked features, measuring both accuracy and classification time. Additionally, we evaluate its performance on flow sets with varying training-to-testing ratios using the complete feature set. Furthermore, we assess the scalability and resource efficiency of *LNNIDS*. These experiments help us understand the sensitivity of *LNNIDS* to input variability and its robustness across different operating conditions. This also provides insights into how well this method generalizes under constrained conditions, especially in edge-based IoT environments. Figure 8 presents the performance results for Public Dataset-1.

As shown in Figure 12(a), in our experiments, *LNNIDS* achieved low accuracy with only 10 or 20 features, because these limited feature sets fail to capture all the critical statistical, protocol, and behavioral details needed to distinguish complex IoT attack patterns. For instance, during a DDoS attack on a network of smart cameras, subtle differences in packet inter-arrival times and data volume key indicators of the attack are missed when only a few features are used, leading to misclassification. This is especially true for features related to timing (e.g., `flow_iat_mean`),

Table 14. Confusion Matrix for Evaluation of LNNIDS on Test Flows of Public Dataset-2 for Unknown Attacks

	A22	*A23*	*A24*	*A25*	*A26*	*A27*	*A28*	*A29*	*A30*	*A31*	*A32*	*U4*	*A34*	*A35*	*A36*	*A37*	*A38*	*U5*	*U6*	*A41*	*A42*	*A43*	*A44*	*A45*	*A46*	*A47*	*U7*	*A49*	*A50*	*A51*	*B2*
A22 (832738)	798752	4726	0	0	0	0	0	5246	8390	6545	0	5390	0	0	0	0	0	0	3689	0	0	0	0	0	0	0	0	0	0	0	0
A23 (2237)	5	1924	0	2	0	0	0	121	20	55	0	60	0	0	0	3	0	0	0	4	0	1	0	3	0	0	0	0	0	0	39
A24 (2508)	0	0	2145	0	203	0	97	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	5	57	0	0	0	0	0	0	0
A25 (2925)	0	0	0	2694	0	0	0	0	227	1	0	2	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A26 (3318)	0	0	182	0	2956	0	5	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	7	165	0	0	0	1	0	0	0
A27 (7036)	0	0	0	0	0	6683	0	0	0	0	0	0	176	1	0	0	0	0	0	0	0	43	0	0	0	3	2	0	0	1	127
A28 (77100)	0	0	528	0	50	0	74074	0	0	0	0	0	0	1	90	0	0	0	80	0	0	0	0	1425	0	0	0	0	0	0	852
A29 (9032)	8	200	0	2	0	0	0	8279	54	19	0	27	3	0	0	189	0	0	0	17	0	1	0	0	0	233	0	0	0	0	0
A30 (6728)	10	0	17	105	0	0	0	125	6285	11	0	171	2	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
A31 (352099)	3360	1	0	8	0	0	0	2612	619	344968	0	531	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A32 (61529)	0	0	0	0	0	0	4	0	0	0	58126	458	0	1	56	0	0	150	5	0	41	0	0	1	0	0	0	2680	7	0	0
U4 (121675)	16	78	0	5	0	0	0	202	3012	1026	5232	111105	1	0	0	1	0	0	0	996	0	0	0	0	0	1	0	0	0	0	0
A34 (5871)	0	0	0	0	0	208	0	0	0	0	0	0	5114	47	0	1	0	0	0	0	0	164	0	0	0	12	325	0	0	0	0
A35 (13878)	0	0	0	0	0	0	0	0	0	0	3	1	0	12766	0	0	0	10	0	0	2	0	0	0	0	1	0	0	659	436	0
A36 (140922)	0	0	0	0	220	0	250	0	0	0	590	0	0	2	130274	1885	0	1360	0	0	200	2215	0	200	0	0	0	0	2180	0	1546
A37 (4412965)	247464	1836	0	0	0	0	0	8226	6156	0	0	125647	1	10809	123715	3527528	0	0	0	97135	88056	0	0	75569	57289	8113	3156	0	0	0	32265
A38 (5084407)	22546	1276	0	0	0	0	0	7360	0	0	0	19456	0	0	0	0	5015132	0	0	0	0	0	0	0	0	0	0	0	0	0	18637
U5 (114267)	0	0	0	0	0	0	7	0	0	0	414	0	1	0	516	0	0	106625	0	0	1850	0	0	0	0	0	0	2480	2374	0	0
U6 (4881)	0	0	1	0	4	0	259	0	0	0	0	0	0	0	191	0	0	1	4414	0	1	0	0	2	0	0	0	8	0	0	0
A41 (7675926)	63920	1445	0	0	0	0	0	8023	0	0	0	22450	0	0	0	0	0	0	0	7551672	0	0	0	0	0	0	0	0	0	0	28416
A42 (1169222)	1120	0	0	0	0	0	0	0	0	0	1520	0	0	0	1540	0	0	1333	100	2939	1156653	0	0	601	0	0	0	1136	0	0	2280
A43 (800909)	0	0	0	0	0	3044	0	0	0	0	0	0	3049	0	0	0	0	0	0	0	0	777983	0	0	0	0	0	0	0	0	16833
A44 (5312882)	9156	0	0	0	0	0	0	0	0	0	550	450	0	0	0	0	0	0	0	28561	37059	0	5221899	80	0	0	0	0	0	0	15127
A45 (2643462)	0	0	2	0	16	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	2586472	24786	7145	2945	0	0	0	22088
A46 (763920)	0	0	0	0	0	0	0	0	0	0	0	1	1114	0	0	0	0	0	0	0	0	0	0	2514	746063	1919	2417	0	0	2170	7722
A47 (9993)	1	7	0	1	0	1	0	327	3	1	0	2	14	0	0	229	1	0	0	0	0	0	0	0	0	9265	141	0	0	0	0
U7 (4934)	0	1	0	0	0	39	0	2	0	0	0	0	82	1	0	1	0	0	0	0	0	22	0	0	0	330	4456	0	0	0	0
A49 (1071)	0	0	0	0	0	0	0	0	0	0	33	0	0	0	67	0	0	36	0	0	0	0	0	0	0	0	0	935	0	0	0
A50 (358550)	0	0	0	0	0	0	1	819	0	0	1280	0	0	1750	714	0	0	2077	0	0	0	0	0	0	0	0	0	1	347292	0	4616
A51 (2750)	0	0	0	0	0	0	0	0	0	0	0	0	0	62	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2626	62
B2 (76116)	0	653	0	0	0	667	0	0	0	0	0	0	200	280	0	0	0	0	0	0	0	0	0	0	701	420	0	0	0	1218	71977



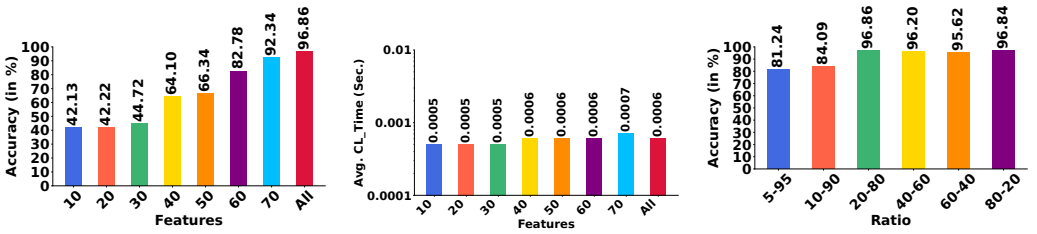
(a) Accuracy vs Features (b) CL_Time vs Features (c) Accuracy vs Ratio

Fig. 8. Sensitivity analysis of *LNNIDS* with varying features and flow ratios for public dataset-1.

protocol-specific flags (e.g., `syn_flag_cnt`), or flow behavior (e.g., `flow_pkts_s`), which are essential for differentiating stealthy or multi-stage attacks like SYN floods or UDP amplification attacks. As more features are added, the proposed method gains a richer representation of IoT attack behavior, resulting in significantly higher accuracy (99.5%) when all features are used, capturing both known and unknown attacks. The improved accuracy is not simply a marginal increase, but a reflection of a more comprehensive behavioral understanding, enabling the method to reliably distinguish between subtle variations in attack patterns and normal IoT communication. Despite the increased feature set, we can see from Figure 12(b) the average classification time (i.e., Avg. *CL_Time*) remains very low (around 0.0006 seconds per flow, due to the efficiency of the dynamic weight assignment in the hybrid liquid layer architecture and the *DYNGraph*'s effective representation of refined liquid states that capture dynamic and evolving attack patterns. For example, in an industrial IoT network with continuous sensor data, such rapid classification enables near real-time detection of anomalies without introducing delays. However, classification time fluctuates (e.g., 0.0006 at 20 features) due to the matching of pair and triplet interactions in the HLL. Moreover, we can also see from Figure 8(c) the 20:80 training-to-testing ratio offers an optimal balance by providing sufficient training data for robust learning while maintaining a realistic evaluation scenario for sporadic port scans or stealthy data exfiltration, achieved through effective time-based spike encoding and a hybrid liquid layer architecture that captures both known and unknown attack patterns. As further demonstrated in our experiments, *LNNIDS* achieved an accuracy of 88.59% on Public Dataset-1 and 81.24% on Public Dataset-2 using only 5% training data. While these results confirm the method's viability in low-data regimes, the accuracy is lower compared with 20% training (99.50% and 96.86% respectively), due to increased misclassifications. These misclassifications are primarily caused by the overlap between attack classes such as stealthy port scans, low-rate DoS, and slow password guessing, which often resemble benign traffic over short time windows. In such cases, the system lacks exposure to sufficient temporal and statistical diversity to correctly interpret subtle behavioral shifts in features like `flow_duration`, `idle_mean`, and `fwd_pkts_s`. However, as the training size increases, *LNNIDS* develops a stronger ability to capture and differentiate these evolving patterns, significantly reducing false positives and enhancing overall robustness across heterogeneous IoT traffic scenarios. This is a key improvement direction for *LNNIDS* that we aim at strengthening in future versions. Similar experiments were conducted using Public Dataset-2, and the results are presented as confusion matrices in Figure 9.

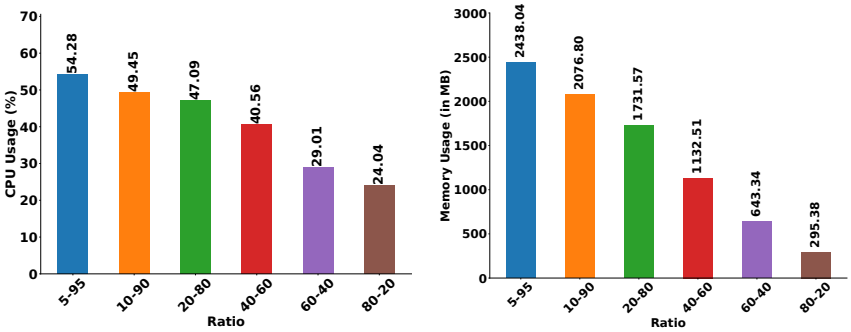
To further evaluate the scalability and resource efficiency of *LNNIDS*, we conducted a detailed analysis of CPU and memory usage during the testing phase across both datasets. For Public Dataset-1 (ACI IoT, See in Figure 10(a) and 10(b)),

As we can see from Figure 10(a), CPU usage peaked at 54.28% at the 5–95% train-test split and gradually reduced to 24.04% at the 80–20 split, while memory usage followed a similar trend, as seen in Figure 10(b), decreasing from 2438.04 MB to 295.38 MB. This shows that *LNNIDS* operates with



(a) Accuracy vs Features (b) CL_Time vs Features (c) Accuracy vs Ratio

Fig. 9. Sensitivity analysis of *LNNIDS* with varying features and flow ratios for public dataset-2.

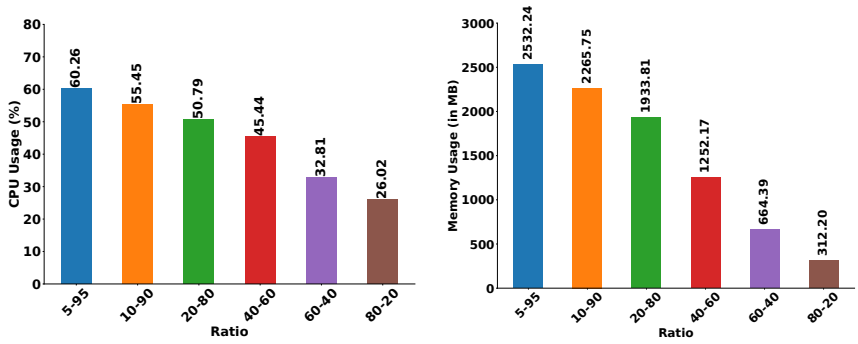


(a) CPU Usage vs Ratio (b) Memory Usage vs Ratio

Fig. 10. Resource usage vs train-test ratio of *LNNIDS* on public dataset-1.

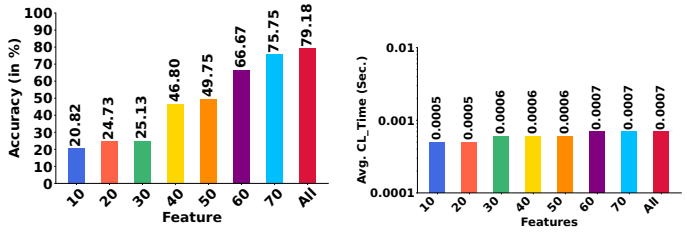
efficient resource utilization even in moderate-scale IoT environments. These results validate the method's lightweight nature and demonstrate that even with a limited training set, *LNNIDS* avoids overloading the computational resources. It maintains stability and responsiveness, which is critical for real-time intrusion detection at the edge. Additionally, the consistent drop in memory and CPU usage as the training ratio increases suggests effective model generalization and optimization across traffic scales. This is especially important in heterogeneous IoT networks, where resource-constrained nodes must process varying traffic volumes without latency spikes. The ability of *LNNIDS* to scale gracefully with data availability underlines its practical viability for deployment in low-power IoT edge settings. In contrast, for Public Dataset-2 (CIC IoT 2023), see Figures 11(a) and 11(b).

As we can see from Figure 11(a), Public Dataset-2 contains over 37 million flows and represents a much denser, high-volume traffic profile. CPU usage peaked at 60.26% and dropped to 26.02% across increasing training ratios. We can also see from Figure 11(b) that the corresponding memory usage ranged from 2532.24 MB to 312.20 MB. These higher resource demands reflect the greater behavioral diversity, protocol heterogeneity, and class imbalance found in complex IoT attack traffic, where features like `flow_pkts_s`, `idle_mean`, and `ack_flag_cnt` fluctuate rapidly. Despite this, *LNNIDS* demonstrated a stable resource profile, with no memory overflows or CPU bottlenecks observed even while processing all test flows. This performance is largely enabled by the use of *DYNGraph*'s dynamic flow clustering, which reduces redundant comparisons and attention-based filtering that processes only behaviorally significant flow interactions, such as coordinated DDoS spikes or stealthy scans. We used modest hardware, including an 11th Generation Intel Core i7-2.8 GHz 64-bit processor with 16 GB RAM, for running our experiments. In the future, we plan to



(a) CPU Usage vs Ratio (b) Memory Usage vs Ratio

Fig. 11. Resource usage vs train-test ratio of *LNNIDS* on public dataset-2.



(a) Accuracy vs Features (b) CL_Time vs Features

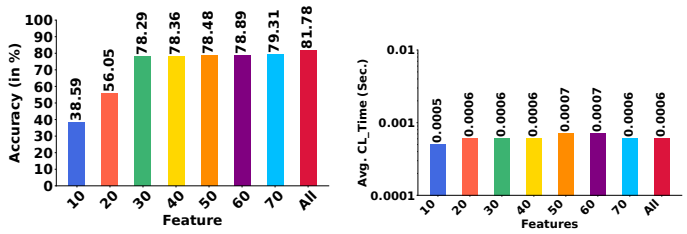
Fig. 12. Cross-platform performance analysis of *LNNIDS* with varying features using public dataset-1 for training and public dataset-2 for testing.

further reduce resource consumption through techniques like lightweight graph approximations, selective attention pruning, and quantization-aware inference. These improvements will help make *LNNIDS* even more suitable for deployment in constrained IoT edge environments.

5.4 Cross-Platform Performance Analysis

In this subsection, we analyze the cross-platform performance of *LNNIDS*. For the first setting, *LNNIDS* uses the entire Public Dataset-1 for training and the entire Public Dataset-2 for testing. For the second setting, the datasets are swapped. The results of this cross-platform analysis are shown in Figures 12 and 13.

As we can see from Figure 12(a), when Public Dataset-1 is used for training and Public Dataset-2 for testing, the accuracy starts with 20.82% with only 10 features and increases steadily, peaking near 79.18% with all features. This trend indicates that lower feature counts limit the model's ability to capture critical IoT attack traffic characteristics such as `flow_pkts_s`, `idle_mean`, `ack_flag_cnt`, and inter-arrival patterns, which often differ significantly across datasets due to device-specific and protocol-level variations, which are essential for detecting diverse attacks across different environments, whereas adding more features enables *LNNIDS* to better generalize to unseen attack behaviors in Dataset-2. The overall accuracy in this cross-platform case is lower than that in the same-platform testing because Public Dataset-1 and Dataset-2 differ in device types, firmware versions, and protocol usage, resulting in variations in packet length distributions, timing behaviors, and encryption overhead. Such differences lead to a domain shift, where patterns learned from Dataset-1 do not fully align with Dataset-2's attack traffic profiles. As shown in Figure 12(b), the



(a) Accuracy vs Features (b) CL_Time vs Features

Fig. 13. Cross-platform performance analysis of *LNNIDS* with varying features using public dataset-2 for training and public dataset-1 for testing.

average classification time per flow remains nearly constant at approximately 0.0005–0.0007 seconds across 10–all feature counts, demonstrating that *LNNIDS* maintains real-time detection capability without latency penalties even when using the full feature set.

In contrast, as we can see from Figure 13(a), when Public Dataset-2 is used for training and Public Dataset-1 for testing, the accuracy begins at 38.59% for 10 features and rises quickly to 81.78% with all features. This suggests that while Dataset-2 offers richer training diversity, features such as `flow_pkts_s`, `idle_mean`, and `ack_flag_cnt`, introduced after the first 20 features, are crucial for accurately mapping to Dataset-1's attack patterns. The performance gap between the two settings stems from Dataset-2's broader range of attack types and protocol-layer combinations (e.g., mixed MQTT and HTTP flooding), which improves generalization when enough features are available. In contrast, Dataset-1's attack traffic is more uniform, with less variation in inter-arrival times. These key features make it harder for a model trained solely on it to adapt without the full feature set. As shown in Figure 13(b), classification time stays consistent around 0.0005–0.0006 seconds regardless of 10 – All feature count, confirming *LNNIDS*'s scalability and suitability for real-time, zero-day IoT attack detection in cross-platform scenarios.

5.5 Performance Comparison

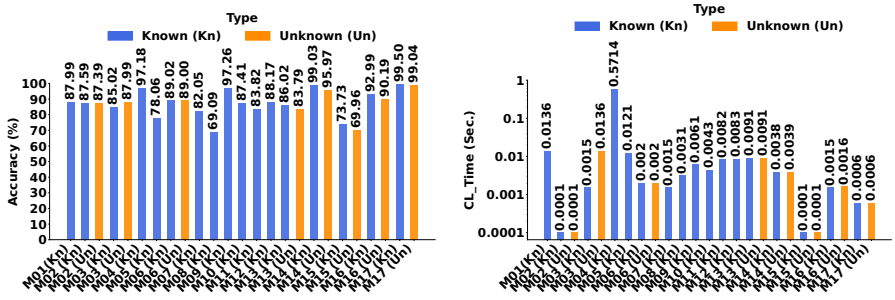
This subsection compares our *LNNIDS* method with six traditional ML and DL approaches: Bagging, DT, **K-Nearest Neighbor (KNN)**, **Light Gradient Boosting Machine (LGBM)**, NB, and **Artificial Neural Network (ANN)** to highlight its strengths in IoT attack detection. Additionally, *LNNIDS* is evaluated against seven state-of-the-art methods for detecting both known and unknown attacks. For clarity, state-of-the-art methods are denoted as "STA," known attacks as "kn, and unknown attacks as "Un." These abbreviations are summarized in Table 15.

As we can see from Table 15, which presents the abbreviations used for the comparison of *LNNIDS* with ML, DL, and STA methods for both Known (Kn) and Unknown (Un) attacks, traditional methods such as Bagging (DT), DT, KNN, LGBM, NB, and ANN are primarily designed for known attacks and lack adaptability to evolving threats. Meanwhile, several STA methods, including GDLC, RF, RF-BPNN, XGB, CNN+DNN, LSTM+CNN, BLWSecureBert, DB-CGAN, DAE-DNN, and CSNN, attempt to handle both known and unknown attacks but still rely on static architectures. Among these STA methods, some are bio-inspired, such as CNN+DNN, LSTM+CNN, BLWSecure-Bert, DB-CGAN, and DAE-DNN, which use static architectures. In addition, the CSNN method is also bio-inspired but is based on a spiking neural network architecture, offering event-driven processing capabilities. Their results are illustrated in Figures 14 and 15 using Public Dataset-1 and Dataset-2.

As we can see from Figure 14, using Public Dataset-1, *LNNIDS* achieves 99.50% accuracy for Kn attacks and 99.04% for Un attacks, outperforming DB-CGAN (99.03% / 95.97%), DAE-DNN (73.73%

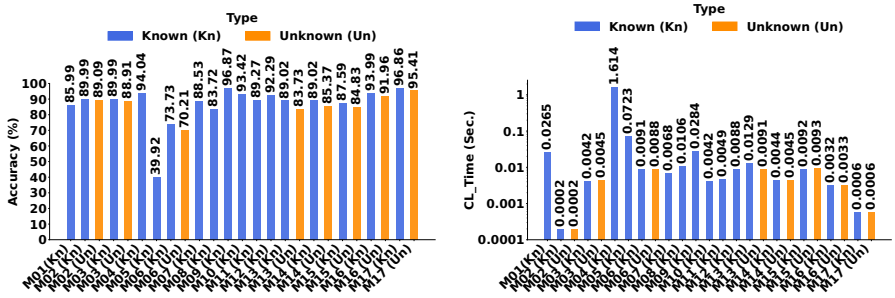
Table 15. Abbreviation for Comparison of *LNNIDS* with ML, DL, and STA Methods for Known(Kn) and Unknown(Un) Attacks

Method ID	State-of-the-Art (STA)	Method Used	Attack Traffic
M_1	Standard Method	Bagging (DT)	Kn
M_2	Standard Method	DT	Kn & Un
M_3	Termos et al. [41]	GDLC	Kn & Un
M_4	Standard Method	KNN	Kn
M_5	Standard Method	LGBM	Kn
M_6	Standard Method	NB	Kn & Un
M_7	Kamaldeep et al. [32]	RF	Kn
M_8	El-Sofany et al. [20]	RF-BPNN	Kn
M_9	Alsabilah and Rawat [9]	XGB	Kn
M_{10}	Standard Method	ANN	Kn
M_{11}	Sharma et al. [37]	CNN + DNN	Kn
M_{12}	Altunay and Albayrak [11]	LSTM + CNN	Kn
M_{13}	Hou et al. [23]	BLWSecureBert	Kn and Un
M_{14}	Zhou et al. [48]	DB-CGAN	Kn and Un
M_{15}	Kunang et al. [28]	DAE-DNN	Kn and Un
M_{16}	Wang et al.[46]	CSNN	Kn and Un
M_{17}	*LNNIDS*	LNN + Graph	Kn & Un



(a) Accuracy vs STA Methods (b) Avg. CL_Time vs STA Methods

Fig. 14. Performance comparison of *LNNIDS* with ML and DL and STA methods with public dataset-1.



(a) Accuracy vs STA Methods (b) Avg. CL_Time vs STA Methods

Fig. 15. Performance comparison of *LNNIDS* with ML and DL and STA methods with public dataset-2.

/ 69.96%), and CSNN (92.99% / 90.19%). Similarly, on Public Dataset-2 (Figure 15), *LNNIDS* attains 96.86% for Kn attacks and 95.41% for Un attacks, exceeding DB-CGAN (89.02% / 85.37%), DAE-DNN (87.59% / 84.83%), and CSNN (93.99% / 91.96%). This improvement is attributed to *LNNIDS*'s hybrid liquid neural network (HLL) and attention-based graph learning, which dynamically adapts to evolving traffic patterns, unlike static architectures in DAE-DNN and fixed spiking encodings in CSNN, or the learned static distributions in generative models like DB-CGAN. In terms of classification latency, *LNNIDS* maintains a consistent advantage: on Dataset-1, the average classification time (Avg. CL_Time) per flow is 0.0006s for both Kn and Un attacks, compared with DB-CGAN (0.0038s / 0.0039s), DAE-DNN (0.0001s / 0.0001s), and CSNN (0.0015s / 0.0016s). On Dataset-2, *LNNIDS* further achieves 0.0006s for both Kn and Un attacks, outperforming DB-CGAN (0.0044s / 0.0045s), DAE-DNN (0.0092s / 0.0093s), and CSNN (0.0032s / 0.0033s). These results emphasize that while generative and spiking neural network-based models can perform zero-day detection, their iterative sampling (DB-CGAN) or event-driven spike processing (CSNN) introduce latency, making them less optimal for strict real-time IoT intrusion detection. Overall, including DB-CGAN, DAE-DNN, and CSNN strengthens the comparative evaluation across generative, static deep, and bio-inspired architectures, with *LNNIDS* consistently achieving higher accuracy and lower classification time, validating its suitability for evolving threat detection in resource-constrained IoT environments.

6 Conclusion and Future Work

This article introduces *LNNIDS*, a supervised method for classifying attacks on IoT device network traffic by analyzing their flow-based features. *LNNIDS* utilizes bidirectional flows extracted from IoT attack network traffic traces and subsequently extracts relevant flow-based features. A wrapper method based on a relevance score is employed for feature selection. Using spike encoding, these features are transformed into spike states suitable for the HLL method, which captures the dynamic patterns of known and unknown IoT attack traffic through an attention-based approach. The IoT attacks are subsequently classified using *DYNGraph*, which matches the feature embedding patterns of known and unknown attacks. We tested *LNNIDS* on a public dataset, achieving an accuracy of 99% and an average recall of 96%. We also compared *LNNIDS* with traditional ML/DL and STA methods, demonstrating that our proposed method achieved a higher recall rate while requiring smaller training sets and larger testing sets. Despite its high accuracy, a limitation of *LNNIDS* is that *LNNIDS* still faces challenges in differentiating between attacks with similar traffic patterns when only a few flows are available for training. Future work could further enhance *LNNIDS* to improve IoT attack classification and evaluate its performance in real-time attack scenarios.

References

[1] 2023. Number of Internet of Things (IoT) Connected Devices Worldwide from 2019 to 2030. (2023). Retrieved from https://www.statista.com/statistics/1183457/iot-connected-devices-world

[2] 2025. A real-time dataset and benchmark for large-scale attacks in IoT environment. Retrieved from https://www.unb.ca/cic/datasets/iotdataset-2023.html

[3] 2025. A real-time dataset and benchmark for large-scale attacks in IoT environment. Retrieved from https://ieee-dataport.org/documents/edge-iiotset-new-comprehensive-realistic-cyber-security-dataset-iot-and-iiot-applications

[4] 2025. Retrieved from https://www.unb.ca/cic/research/applications.html#CICFlowMeter

[5] 2025. Python. (2025). Retrieved from https://www.python

[6] Tariq Ahamed Ahanger, Abdullah Aljumah, and Mohammed Atiquzzaman. 2022. State-of-the-art survey of artificial intelligent techniques for IoT security. *Computer Networks* 206, 23 (2022), 108771–108800.

[7] Naveed Ahmed, Md Asri Ngadi, Muhammad Siraj Rathore, and Azhar Mahmood. 2025. PCM-RF a hybrid feature selection mechanism for intrusion detection system in IoT. *Security and Privacy* 8, 6 (2025), e499–e516.

[8] Saleh Mohamed Alhidaifi, Muhammad Rizwan Asghar, and Imran Shafique Ansari. 2024. A survey on cyber resilience: Key strategies, research challenges, and future directions. *ACM Computing Surveys* 56, 12 (2024), 1–48.
[9] Nasser Alsabilah and Danda B. Rawat. 2025. Joint rough set theory and XGBoost based learning for network intrusion detection system. *IEEE Internet of Things Journal* 12, 7 (2025), 1–8.
[10] Tanzeela Altaf, Xu Wang, Wei Ni, Guangsheng Yu, Ren Ping Liu, and Robin Braun. 2023. A new concatenated multigraph neural network for IoT intrusion detection. *Internet of Things* 22, 6 (2023), 100818–100832.
[11] Hakan Can Altunay and Zafer Albayrak. 2023. A Hybrid CNN+ LSTM-based Intrusion Detection System for Industrial IoT Networks. *Engineering Science and Technology, an International Journal* 38, 10 (2023), 101322–101334.
[12] Ons Aouedi, Thai-Hoc Vu, Alessio Sacco, Dinh C. Nguyen, Kandaraj Piamrat, Guido Marchetto, and Quoc-Viet Pham. 2024. A survey on intelligent internet of things: Applications, security, privacy, and future directions. *IEEE Communications Surveys & Tutorials* 26, 2 (2024), 1–56.
[13] Sarhad Arisdakessian, Omar Abdel Wahab, Azzam Mourad, Hadi Otrok, and Mohsen Guizani. 2022. A survey on IoT intrusion detection: Federated learning, game theory, social psychology, and explainable AI as future directions. *IEEE Internet of Things Journal* 10, 5 (2022), 4059–4092.
[14] Javed Ashraf, Marwa Keshk, Nour Moustafa, Mohamed Abdel-Basset, Hasnat Khurshid, Asim D. Bakhshi, and Reham R. Mostafa. 2021. IoTBoT-IDS: A novel statistical learning-enabled botnet detection framework for protecting networks of smart cities. *Sustainable Cities and Society* 72, 11 (2021), 103041–103052.
[15] Shahid Allah Bakhsh, Muhammad Almas Khan, Fawad Ahmed, Mohammed S. Alshehri, Hisham Ali, and Jawad Ahmad. 2023. Enhancing IoT network security through deep learning-powered intrusion detection system. *Internet of Things* 24, 6 (2023), 100936–100971.
[16] Sander M. Bohte, Joost N. Kok, and Han La Poutre. 2002. Error-backpropagation in temporally encoded networks of spiking neurons. *Neurocomputing* 48, 14 (2002), 17–37.
[17] Mingcan Cen, Frank Jiang, Xingsheng Qin, Qinghong Jiang, and Robin Doss. 2024. Ransomware early detection: A survey. *Computer Networks* 239, 25 (2024), 110138–110157.
[18] Zekai Chen, Dingshuo Chen, Xiao Zhang, Zixuan Yuan, and Xiuzhen Cheng. 2021. Learning graph structures with transformer for multivariate time-series anomaly detection in IoT. *IEEE Internet of Things Journal* 9, 12 (2021), 9179–9189.
[19] Ivan Cvitić, Dragan Perakovic, Brij B. Gupta, and Kim-Kwang Raymond Choo. 2021. Boosting-based DDoS detection in internet of things systems. *IEEE Internet of Things Journal* 9, 3 (2021), 2109–2123.
[20] Hosam El-Sofany, Samir A. El-Seoud, Omar H. Karam, and Belgacem Bouallegue. 2024. Using machine learning algorithms to enhance IoT system security. *Scientific Reports* 14 (2024), 12077–12095.
[21] Anselme Herman Eyeleko and Tao Feng. 2023. A critical overview of industrial internet of things security and privacy issues using a layer-based hacking scenario. *IEEE Internet of Things Journal* 10, 24 (2023), 21917–21941.
[22] Smarajit Ghosh. 2025. Network traffic analysis based on cybersecurity intrusion detection through an effective automated separate guided attention federated graph neural network. *Applied Soft Computing* 169, 24 (2025), 112603–112623.
[23] Shuling Hou, Gaoshang Xiao, and Huiying Zhou. 2025. High-performance network attack detection in unknown scenarios based on improved vertical model. *Cluster Computing* 28, 16 (2025), 62–77.
[24] Fatima Hussain, Rasheed Hussain, Syed Ali Hassan, and Ekram Hossain. 2020. Machine learning in IoT security: Current solutions and future challenges. *IEEE Communications Surveys & Tutorials* 22, 3 (2020), 1686–1721.
[25] Muhammad Muhammad Inuwa and Resul Das. 2024. A comparative analysis of various machine learning methods for anomaly detection in cyber attacks on IoT networks. *Internet of Things* 26, 7 (2024), 101162–101179.
[26] Saeid Jamshidi, Amin Nikanjam, Kawser Wazed Nafi, Foutse Khomh, and Rasoul Rasta. 2025. Application of deep reinforcement learning for intrusion detection in internet of things: A systematic review. *Internet of Things* 31, 8 (2025), 101531–101559.
[27] Weiwei Jiang. 2022. Graph-based deep learning for communication networks: A survey. *Computer Communications* 185, 43 (2022), 40–54.
[28] Yesi Novaria Kunang, Siti Nurmaini, Deris Stiawan, and Bhakti Yudho Suprapto. 2021. Attack classification of an intrusion detection system using deep learning and hyperparameter optimization. *Journal of Information Security and Applications* 58, 9 (2021), 102804–102818.
[29] Lieqing Lin, Qi Zhong, Jiasheng Qiu, and Zhenyu Liang. 2025. E-GRACL: An IoT intrusion detection system based on graph neural networks. *The Journal of Supercomputing* 81, 16 (2025), 42.
[30] Wai Weng Lo, Siamak Layeghy, Mohanad Sarhan, Marcus Gallagher, and Marius Portmann. 2022. E-GraphSAGE: A graph neural network based intrusion detection system for IoT. In *Proceeding of 16th IEEE/IFIP Network Operations and Management Symposium (NOMS'22)*. 1–9.
[31] Cybercrime Magazine. 2025. Cybercrime To Cost The World $10.5 Trillion Annually By 2025. (2025). Retrieved from https://cybersecurityventures.com/hackerpocalypse-cybercrime-report-2016/

[32] Manisha Malik, Maitreyee Dutta, et al. 2023. Feature engineering and machine learning framework for DDoS attack detection in the standardized internet of things. *IEEE Internet of Things Journal* 10 (2023), 8658–8669.
[33] Alaeddine Mihoub, Ouissem Ben Fredj, Omar Cheikhrouhou, Abdelouahid Derhab, and Moez Krichen. 2022. Denial of service attack detection and mitigation for internet of things using looking-back-enabled machine learning techniques. *Computers & Electrical Engineering* 98, 47 (2022), 107716–107726.
[34] Nour Moustafa, Marwa Keshk, Kim-Kwang Raymond Choo, Timothy Lynar, Seyit Camtepe, and Monica Whitty. 2021. DAD: A distributed anomaly detection system using ensemble one-class statistical learning in edge networks. *Future Generation Computer Systems* 118, 36 (2021), 240–251.
[35] Fengyuan Nie, Weiwei Liu, Guangjie Liu, Bo Gao, Jianan Huang, Wen Tian, and Chau Yuen. 2025. Empowering anomaly detection in IoT traffic through multi-view subspace learning. *IEEE Internet of Things Journal* 14, 11 (2025), 1–15.
[36] Makhduma F. Saiyed and Irfan Al-Anbagi. 2023. Flow and unified information-based DDoS attack detection system for multi-topology IoT networks. *Internet of Things* 24, 6 (2023), 100976–100998.
[37] Bhawana Sharma, Lokesh Sharma, Chhagan Lal, and Satyabrata Roy. 2024. Explainable artificial intelligence for intrusion detection in IoT networks: A deep learning based approach. *Expert Systems with Applications* 238, 34 (2024), 121751–121766.
[38] Paweł Smaga. 2025. Profiling the victim-cyber risk in commercial banks. *Computers & Security* 150, 42 (2025), 104274–104287.
[39] Hamid Tahaei, Firdaus Afifi, Adeleh Asemi, Faiz Zaki, and Nor Badrul Anuar. 2020. The rise of traffic classification in IoT networks: A survey. *Journal of Network and Computer Applications* 154, 24 (2020), 102538.
[40] Akbar Telikani, Arupa Sarkar, Bo Du, Fendy Santoso, Jun Shen, Jun Yan, Jianming Yong, and Emily Yap. 2025. Unmanned aerial vehicle-aided intelligent transportation systems: Vision, challenges, and opportunities. *IEEE Communications Surveys & Tutorials* 27, 25 (2025), 1–50.
[41] Mortada Termos, Zakariya Ghalmane, Ahmad Fadlallah, Ali Jaber, Mourad Zghal, et al. 2024. GDLC: A new graph deep learning framework based on centrality measures for intrusion detection in IoT networks. *Internet of Things* 26, 7 (2024), 101214–101234.
[42] Dipti Theng and Kishor K. Bhoyar. 2024. Feature selection techniques for machine learning: A survey of more than two decades of research. *Knowledge and Information Systems* 66, 12 (2024), 1575–1637.
[43] S Velliangiri, Bhuvaneswari Amma NG, and Nam-Kyun Baik. 2023. Detection of DoS attacks in smart city networks with feature distance maps: A statistical approach. *IEEE Internet of Things Journal* 10, 21 (2023), 18853–18860.
[44] David Verstraeten, Benjamin Schrauwen, Dirk Stroobandt, and Jan Van Campenhout. 2005. Isolated word recognition with the liquid state machine: A case study. *Inform. Process. Lett.* 95, 6 (2005), 521–528.
[45] Supongmen Walling and Sibesh Lodh. 2025. An extensive review of machine learning and deep learning techniques on network intrusion detection for IoT. *Transactions on Emerging Telecommunications Technologies* 36, 2 (2025), e70064–e70092.
[46] Zhen Wang, Fuad A. Ghaleb, Anazida Zainal, Maheyzah Md Siraj, and Xing Lu. 2024. An efficient intrusion detection model based on convolutional spiking neural network. *Scientific Reports* 14 (2024), 7054–7073.
[47] Meihui Zhong, Mingwei Lin, Chao Zhang, and Zeshui Xu. 2024. A Survey on Graph Neural Networks for Intrusion Detection Systems: Methods, Trends and Challenges. *Computers & Security* 141, 18 (2024), 103821–103837.
[48] Xiaokang Zhou, Yiyong Hu, Jiayi Wu, Wei Liang, Jianhua Ma, and Qun Jin. 2022. Distribution bias aware collaborative generative adversarial network for imbalanced deep learning in industrial IoT. *IEEE Transactions on Industrial Informatics* 19, 1 (2022), 570–580.

Received 15 March 2025; revised 14 August 2025; accepted 8 October 2025