Computers & Security 154 (2025) 104417

Contents lists available at ScienceDirect

Computers & Security

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

Full length article

Transformer-based knowledge distillation for explainable intrusion detection system

Nadiah AL-Nomasy [a], Abdulelah Alamri [a], Ahamed Aljuhani [a], Prabhat Kumar [b],*

[a] The College of Computing and Information Technology, University of Tabuk, Tabuk 71491, Saudi Arabia
[b] The Department of Software Engineering, LUT School of Engineering Science, LUT University, 53850 Lappeenranta, Finland

A R T I C L E I N F O

Keywords:
Deep learning
Explainable artificial intelligence (XAI)
Internet of things (ioT)
Intrusion detection system (IDS)
Knowledge distillation (KD)

A B S T R A C T

The rapid expansion of IoT networks has increased the risk of cyber threats, making intrusion detection systems (IDS) critical for maintaining security. However, most of the existing IDS rely on computationally intensive deep learning architectures, rendering them unsuitable for IoT environments with limited resources. Additionally, existing IDS approaches, including those using Knowledge Distillation (KD), often fail to capture the complex temporal dependencies and contextual relationships inherent in IoT traffic, which limits their ability to detect complex multi-stage attacks. Furthermore, these models frequently lack transparency, hindering effective decision-making by security experts. To address these gaps, we propose DistillGuard, a novel IDS framework designed specifically for IoT networks. The proposed framework employs a Transformer-based teacher model, which utilizes a hybrid attention mechanism combining multi-head self-attention (MHSA) and cross-attention layers to effectively capture both temporal and contextual patterns in network traffic. The framework further incorporates a Selective Gradient-Based Knowledge Distillation (SG-KD) process to transfer critical knowledge from the teacher model to a lightweight student model, optimizing performance while reducing computational costs. In addition, 'DistillGuard' integrates gradient contribution heatmaps, layer-wise contribution, and gradient selection impact analysis to provide detailed explanability, enabling security experts to understand which layers contribute to the detection of attacks. Experimental results demonstrate that 'DistillGuard' achieves superior detection accuracy and efficiency compared to existing state-of-the-art IDS models.

1. Introduction

The Internet of Things (IoT) has revolutionized various industries by enabling interconnected smart devices to collect, exchange, and analyze data in real-time. With the number of IoT-connected devices projected to reach 75 billion by 2025 (Pahlevi et al., 2022), the scope of IoT applications continues to expand across domains such as healthcare, industrial automation, smart cities, and transportation. These devices, equipped with sensors and actuators, generate vast amounts of data that support decision-making and enhance automation (Sun et al., 2024).

However, the rapid growth of IoT networks presents significant security challenges. IoT environments are highly vulnerable to cyber threats due to the heterogeneous nature of devices, limited computational resources, and widespread deployment in critical infrastructure (Alrashdi et al., 2019). Attacks such as distributed denial of service (DDoS), man-in-the-middle (MitM), and ransomware pose serious risks to the confidentiality, integrity, and availability of data (Saed and Aljuhani, 2022; Thamer and Alubady, 2021). Ensuring the security of these networks is crucial, as compromised IoT devices can have far-reaching consequences, impacting not only data integrity but also the operational stability of smart systems (Barrera et al., 2023).

One promising solution to address these security concerns is the implementation of Intrusion Detection Systems (IDS) in IoT networks. IDSs can monitor network traffic and detect abnormal behavior, helping to prevent malicious activities before they cause significant damage (Al-Rubaye and Türkben, 2024). Various machine learning and deep learning techniques have been employed in IDS design to improve detection accuracy. However, traditional approaches often suffer from computational complexity, which is impractical for resource-constrained IoT devices (Duan et al., 2024; Aljuhani, 2022). Additionally, existing machine learning and deep learning-based detection models often function as "black boxes", where it is unclear which specific features contribute to a given prediction. This lack of interpretability is particularly problematic in IoT networks, where understanding which features – such

* Corresponding author.
E-mail addresses: 431008527@stu.ut.edu.sa (N. AL-Nomasy), 421000544@stu.ut.edu.sa (A. Alamri), A_Aljuhani@ut.edu.sa (A. Aljuhani), prabhat.kumar@lut.fi (P. Kumar).

https://doi.org/10.1016/j.cose.2025.104417
Received 18 October 2024; Received in revised form 27 February 2025; Accepted 3 March 2025
Available online 13 March 2025

as network traffic patterns, protocol types, or temporal correlations – trigger an alert is crucial for security analysts to make informed decisions (Zhu et al., 2024b). Without insight into the model's reasoning, it becomes difficult to validate or trust the model's predictions, implement targeted mitigation strategies, or adapt the system to evolving threats (Kök et al., 2023).

To overcome these challenges, Knowledge Distillation (KD) has emerged as a powerful method to reduce model complexity by transferring knowledge from a large, complex model (teacher) to a smaller, more efficient model (student) (Tang et al., 2024). In traditional KD approaches, the student model is trained to mimic the output logits of the teacher model, but this often leads to inefficient learning in resource-constrained environments like IoT. Moreover, conventional teacher models, which often rely on basic architectures such as Convolutional Neural Networks (CNNs) or Recurrent Neural Networks (RNNs), struggle to capture the long-range dependencies and complex, multi-dimensional relationships inherent in IoT traffic data (Zhu et al., 2024a). These models typically focus on local patterns, which may not fully represent the intricate temporal and contextual dynamics present in IoT network behavior. As a result, critical information needed for accurate intrusion detection, such as cross-traffic interactions or temporal correlations between events, is often lost during knowledge transfer. This leads to inefficient utilization of computational resources, with the student model learning less relevant features, ultimately resulting in incomplete threat detection and reduced robustness in identifying complex cyberattacks (Wang and Yoon, 2021).

Towards this, we propose 'DistillGuard', a novel Selective Gradient-Based Knowledge Distillation (SG-KD) framework tailored for IoT networks. The proposed framework addresses the limitations of traditional knowledge distillation by incorporating a Transformer-based teacher model that effectively captures long-range temporal and contextual relationships in IoT traffic data. By selectively transferring only the most relevant knowledge to a lightweight student model, 'DistillGuard' ensures high detection accuracy with minimal computational overhead, making it ideal for resource-constrained IoT environments. Additionally, DistillGuard integrates gradient contribution heatmaps, layer-wise contribution analysis, and gradient selection impact evaluation to provide deeper insights into the knowledge transfer process.

1.1. Contribution

- We propose 'DistillGuard', a novel intrusion detection framework for IoT networks, which integrates a Transformer-based teacher model and a Selective Gradient-Based Knowledge Distillation (SG-KD) process.
- First, we design a Transformer-based teacher model that leverages a hybrid attention mechanism, incorporating both multi-head self-attention (MHSA) and cross-attention layers. The MHSA allows the model to focus on multiple aspects of IoT traffic features simultaneously, capturing long-range dependencies within the network data. The cross-attention mechanism enhances the interaction between temporal and network features, such as traffic volume, protocol types, and connection states. This architecture is particularly suited for detecting complex and coordinated cyberattacks in IoT environments, where both temporal sequences and contextual relationships between devices are critical for accurate intrusion detection.
- Second, we propose a novel Selective Gradient-Based Knowledge Distillation (SG-KD) process to optimize knowledge transfer between the Transformer-based teacher model and the lightweight student model. SG-KD leverages the gradients of the teacher's intermediate layers to selectively identify and prioritize the most important features and representations that contribute to accurate detection. Specifically, it computes the gradients of the loss with respect to the teacher's hidden representations to determine the most informative regions of the feature space. These regions are then filtered and transferred to the student model, allowing the student to focus on the most critical patterns and thereby reduces the computational burden on the student model.
- Finally, we integrated gradient contribution heatmaps, layer-wise contribution analysis, and gradient selection impact evaluation to provide deeper insights into the knowledge transfer process. These techniques highlight the most influential layers and gradient distributions within the model, helping security analysts understand how knowledge is distilled and utilized for intrusion detection.

1.2. Paper outline

Section 2 presents the background of intrusion detection and summarizes related works. The proposed 'DistillGuard' and its component is presented in Section 3. The experimental results are discussed in Section 4 and discussion about the contribution in Section 5. Finally, Section 6 summarizes our work with future outline.

2. Background and related work

2.1. Intrusion detection systems and knowledge distillation

Intrusion Detection Systems (IDS) are essential for detecting, analyzing, and preventing unauthorized access or cyberattacks in network infrastructures (Alazawi et al., 2024). Traditional IDS can be categorized into Network Intrusion Detection Systems (NIDS), which monitor network traffic, and Host Intrusion Detection Systems (HIDS), which focus on individual systems such as servers or workstations (Aljuhani et al., 2023). With the rise of more complex attacks, machine learning (ML) and deep learning (DL) techniques have been widely adopted in IDS to automatically detect anomalies and classify potential threats (Barrera et al., 2023), Al-Rubaye and Türkben (2024). However, these ML/DL-based IDS solutions often involve high computational complexity, making them unsuitable for deployment in resource-constrained environments such as IoT networks, where processing power and memory are limited (Abou El Houda et al., 2022).

To mitigate these challenges, Knowledge Distillation (KD) has emerged as an effective method to reduce model complexity while maintaining detection accuracy. KD has been explored in various domains, including cybersecrity (Fu et al., 2023). KD works by transferring knowledge from a larger, more complex teacher model to a smaller, more efficient student model (Tang et al., 2024). The teacher model captures detailed patterns in network traffic, while the student model learns to mimic the teacher's performance in a more compact form. By distilling the teacher's knowledge, the student model is able to retain essential features and perform well even in constrained environments. In IDS applications, KD enables the student model to effectively detect a variety of cyber threats with reduced computational overhead, allowing real-time detection and response in IoT and other resource-limited settings. The synergy between IDS and KD allows for scalable, efficient, and accurate threat detection in modern networks (Wang and Yoon, 2021).

2.2. Current IDS for IoT networks

IDSs have been widely employed in IoT network environments as promising solutions to address various vulnerabilities that could potentially be exploited by intruders in smart networks. For instance, Kalaria et al. (2024) proposed a security framework named 'IoTPredictor' which included anomaly detection, intrusion detection, and firmware updates. The framework was designed to identify security threats in IoT devices. Hidden Markov models (HMMs) were used and integrated with an anomaly detection system to proactively identify cyberattacks in IoT networks. The proposed method effectively identified malicious activities in IoT networks. Hore et al. (2024) proposed an IDS using a sequential deep learning framework. The proposed method included three

Table 1
Overview of Techniques Used in IDS for IoT networks.

Reference	Method name	Contribution	Model interpretability	Suitable for IoT
Kalaria et al. (2024)	IoTPredictor (HMM)	Integrated anomaly detection, intrusion detection, and firmware updates	Lacks transparency in decision-making due to HMM's probabilistic nature	Designed specifically for IoT networks, targeting IoT devices
Hore et al. (2024)	Sequential DNN Framework	Used DNN for classifier and autoencoders for intrusion detection	Lacks interpretability, typical of deep learning models	Requires significant computational power, making it less suited for resource-constrained IoT devices
Jeffrey et al. (2024)	Ensemble Learning (Bagging, Boosting, Voting, Stacking)	Improved accuracy using ensemble learning techniques	Ensemble methods combine several models, making it harder to interpret the final decision	Applicable for IoT environments, but computationally expensive for constrained devices
Zakariyya et al. (2023)	Optimized DNN	Machine learning method optimized for detecting IoT cyberattacks	Deep learning techniques offer limited explainability	Optimized for federated learning settings, minimizing memory and energy consumption for IoT
Ampel et al. (2024)	Multi-Teacher Knowledge Distillation	Combined RoBERTa and CodeBERT for cyber risk management	Knowledge distillation improves efficiency, but explainability remains a challenge	Mainly designed for general cybersecurity, not specifically tailored to IoT environments
Aljuhani et al. (2023)	SaaS-based IDS with PSO	Reduced computational overhead for IoT healthcare systems	The model's decision-making process is complex due to the integration of multiple techniques	Tailored for healthcare IoT, with optimizations like PSO to minimize resource usage
Sahu et al. (2021)	Hybrid CNN-LSTM	Hybrid deep learning for feature extraction and attack detection in IoT	The hybrid architecture adds complexity, limiting transparency	Well-suited for IoT devices due to efficient feature extraction and detection techniques
Mothukuri et al. (2022)	Federated Learning with GRU	Federated learning approach for anomaly detection in IoT networks	GRU models are harder to interpret due to their recurrent nature	Designed for IoT networks with federated learning, reducing the need for centralized data processing

sequential DNN frameworks: one was specified for the classifier, and two were used for autoencoders, implemented to identify cyberattacks in computer and network systems. The proposed method demonstrated its ability to protect network systems from cyber threats. Jeffrey et al. (2024) proposed an anomaly detection approach based on ensemble learning techniques. The proposed method utilized multiple machine learning algorithms to improve detection accuracy. The authors used different ensemble learning techniques such as stacking, voting, boosting, and bagging. When compared to other techniques, the bagging method produced the highest accuracy rate.

Zakariyya et al. (2023) proposed a detection method using a machine learning algorithm for mitigating cyberattacks in IoT networks. The proposed approach used optimized deep neural networks to improve the detection performance of the model. Additionally, the model was tested and evaluated in federated learning settings. The evaluation results showed that the model reduced memory utilization while maintaining accurate detection. Ampel et al. (2024) proposed a multi-teacher knowledge distillation approach that integrated a transformer into a cybersecurity risk management framework. The authors used RoBERTa and CodeBERT with the proposed transformer framework. The method achieved a good accuracy rate compared with existing detection models. Aljuhani et al. (2023) proposed a software-as-a-service (SaaS)-based IDS for IoT healthcare systems. The proposed method used particle swarm optimization (PSO) to minimize computational overhead in IoT devices. The authors employed several ensemble ML and deep learning techniques to detect cyberattacks in IoT networks. The performance results demonstrated that the model could correctly identify threats in IoT networks. Sahu et al. (2021) proposed IoT attack detection using a hybrid deep learning method. The proposed model employed a convolutional neural network to extract features of data. In the attack detection phase, the detection model utilized a long short-term memory algorithm to identify cyberattacks in IoT networks. The experimental results demonstrated that the detection method could effectively identify cyberattacks in IoT networks. Mothukuri et al. (2022) proposed an anomaly detection approach for IoT networks. The proposed method used federated learning to proactively determine anomalies in IoT networks. The gated recurrent units algorithm was applied, and the obtained weights were shared with the server. The proposed approach outperformed traditional machine learning algorithms. Michelena et al. (2023) proposed an intelligent classifier model for mitigating DoS attacks in IoT networks. The proposed method used principal component analysis for feature reduction, and a set of machine learning techniques for data attack detection. The evaluation results showed that the proposed method can effectively detect DoS attacks in IoT networks. Table 1 provides an overview of related IDS techniques for IoT networks, summarizing their key contributions, interpretability, and suitability for IoT environments. This comparison demonstrates how different models address the unique constraints of IoT networks, such as limited computational resources and the need for real-time detection, while also highlighting gaps in interpretability.

2.3. *Current knowledge distillation approaches*

Wang et al. (2024) introduced an adaptive knowledge distillation (AKD) approach to transfer knowledge from a cloud-based teacher model (T-model) to an edge-based lightweight student model (S-model). The framework dynamically adjusts the distillation temperature to improve the student model's understanding of complex representations. Gou et al. (2022) proposed a collaborative knowledge distillation framework that integrates response-based and relation-based knowledge transfer between peer networks. This approach enables bidirectional transfer, allowing networks to learn collaboratively while enhancing their self-learning capabilities. However, the framework can face challenges in managing output discrepancies between networks during collaboration and in balancing the transfer of response-based and relation-based knowledge to improve generalizability. Zhang et al. (2022) developed a space–time data prediction algorithm using knowledge distillation. This approach compresses a teacher network into lightweight student networks by combining generative adversarial network (GAN) discrimination and teacher outlier elimination (TOE). However, the technique requires careful tuning of GAN parameters, which can introduce additional complexity in deployment for edge devices with limited resources. Zhu et al. (2023) introduced LKD-STNN, a lightweight malicious traffic detection framework for IoT. This method uses knowledge distillation to compress teacher-student networks, retaining detection efficacy while reducing computational overhead. However, the framework emphasizes memory efficiency but may underperform in high-dimensional traffic scenarios with complex attack patterns. Most of these KD techniques transfers all knowledge from the teacher to the student without assessing its relevance, leading to inefficiencies and overburdened student models. Furthermore, these methods lack adversarial robustness, leaving the student model vulnerable to exploitation by adversarial attacks, which are increasingly prevalent in IoT environments. Addressing these shortcomings requires a more selective and secure distillation approach capable of enhancing efficiency and resilience.

3. Proposed DistillGuard framework

In this section, we present the DistillGuard framework and its key components, which are designed to address the specific challenges of Intrusion Detection Systems (IDS) in IoT networks. The framework consists of several critical components: (A) Problem Formulation (B) Data Pre-processing; (C) the Transformer-based Teacher Model; (D) Teacher Model Training; (E) the Student Model; (F) the Selective Gradient-Based Knowledge Distillation (SG-KD) Process and (F) Explainable Artificial Intelligence (XAI). The components of the DistillGuard framework is shown using a flowchart in Fig. 1. Each components are explained in below subsections:

3.1. Problem formulation

Let us consider an IoT network comprising N devices, each generating data streams $\mathbf{X}_n \in \mathbb{R}^{T\times d}$, where T represents the time series length, and d represents the number of features extracted per device. The goal is to develop an IDS that accurately classifies the data streams into C classes: $\mathbf{y}_n \in 0, 1, \dots, C-1$, where each class represents a specific type of traffic or attack, including normal traffic (class 0). Given the diverse and dynamic nature of cyber threats in IoT environments, the development of an effective IDS must address below mentioned critical challenges:

1. *Challenge 1: Computational Constraints:* IoT devices are resource-constrained, with limited memory, processing power, and energy. Let C_n denote the computational capacity of device n. The IDS needs to satisfy the following constraint:

$$\mathcal{F}(\mathbf{X}_n, \mathcal{T}) \le C_n, \quad \forall n \in 1, \dots, N, \tag{1}$$

where $\mathcal{F}(\mathbf{X}_n, \mathcal{T})$ represents the computational complexity of processing input $\mathbf{X}_n$ using the teacher model $\mathcal{T}$.

2. *Challenge 2: Multi-class attack detection:* The teacher model $\mathcal{T}$, a Transformer-based architecture, provides superior detection accuracy across C classes, represented by:

$$\mathcal{A}\mathcal{T} = \mathbb{E}\left[\mathbb{I}(\hat{\mathbf{y}}\mathcal{T} = \mathbf{y}n)\right], \tag{2}$$

where $\mathcal{A}\mathcal{T}$ is the accuracy of the teacher model, $\hat{\mathbf{y}}_{\mathcal{T}}$ is the prediction, and $\mathbb{I}(\cdot)$ is the indicator function. However, deploying $\mathcal{T}$ directly on each IoT device is infeasible due to the constraint C_n. A lightweight model $\mathcal{S}$, the student model, is required such that:

$$\mathcal{F}(\mathbf{X}_n, \mathcal{S}) \ll \mathcal{F}(\mathbf{X}_n, \mathcal{T}). \tag{3}$$

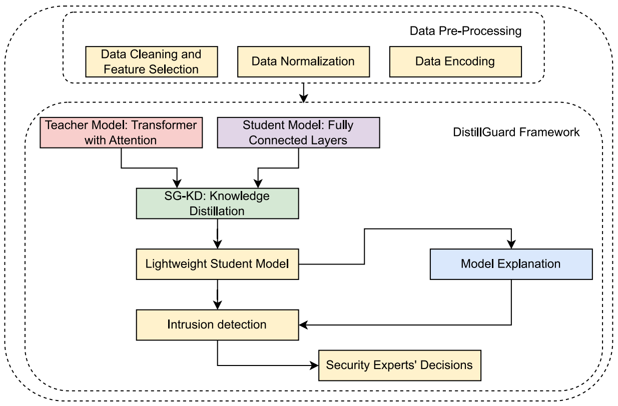

Data Pre-Processing
Data Cleaning and Feature Selection
Data Normalization
Data Encoding
Teacher Model: Transformer with Attention
Student Model: Fully Connected Layers
DistillGuard Framework
SG-KD: Knowledge Distillation
Lightweight Student Model
Model Explanation
Intrusion detection
Security Experts' Decisions

Fig. 1. Flowchart for the proposed DistillGuard framework for Intrusion Detection.

3. Beyond attack detection, the IDS must perform contextual analysis to identify patterns, predict emerging threats, and adapt to evolving attack vectors in real-time. This involves estimating the anomaly score $\mathcal{S}$anomaly for each device n, defined as:

$$\mathcal{S}\text{anomaly}(\mathbf{X}n) = \mathbb{E}[|\mathbf{h}\mathcal{T} - \mathbf{h}_{\mathcal{S}}|^2], \tag{4}$$

where $\mathbf{h}\mathcal{T}$ and $\mathbf{h}_{\mathcal{S}}$ are the feature representations extracted by the teacher and student models, respectively. A high anomaly score indicates potential unseen attacks or deviations from normal behavior, enabling proactive mitigation.

We aim to minimize the loss function $\mathcal{L}_{\text{KD}}$ between the teacher model $\mathcal{T}$ and the student model $\mathcal{S}$, which can be expressed as:

$$\mathcal{L}_{\text{KD}} = \alpha L_{\text{CE}}(\mathbf{y}_n, \hat{\mathbf{y}}_{\mathcal{S}}) + \beta L_{\text{KL}}\left(\sigma\left(\frac{\mathbf{z}_{\mathcal{T}}}{T}\right), \sigma\left(\frac{\mathbf{z}_{\mathcal{S}}}{T}\right)\right) + \gamma L_{\text{anomaly}}(\mathbf{h}_{\mathcal{T}}, \mathbf{h}_{\mathcal{S}}), \tag{5}$$

where: $L_{\text{CE}}(\mathbf{y}_n, \hat{\mathbf{y}}_{\mathcal{S}})$ is the cross-entropy loss for multi-class attack detection between the true labels $\mathbf{y}_n$ and the student's predictions $\hat{\mathbf{y}}_{\mathcal{S}}$. L_{KL} is the Kullback–Leibler (KL) divergence between the softened logits of the teacher model $\mathbf{z}_{\mathcal{T}}$ and those of the student model $\mathbf{z}_{\mathcal{S}}$, with a temperature parameter T:

$$L_{\text{KL}} = \sum_{i=1}^{C} \sigma\left(\frac{z_{\mathcal{T},i}}{T}\right) \log\left(\frac{\sigma\left(\frac{z_{\mathcal{T},i}}{T}\right)}{\sigma\left(\frac{z_{\mathcal{S},i}}{T}\right)}\right), \tag{6}$$

where σ is the softmax function, and C is the number of classes. $L_{\text{anomaly}}(\mathbf{h}_{\mathcal{T}}, \mathbf{h}_{\mathcal{S}})$ measures the difference between the feature representations extracted by the teacher model $\mathbf{h}_{\mathcal{T}}$ and the student model $\mathbf{h}_{\mathcal{S}}$:

$$L_{\text{anomaly}} = \mathbb{E}\left[\|\mathbf{h}_{\mathcal{T}} - \mathbf{h}_{\mathcal{S}}\|^2\right], \tag{7}$$

promoting alignment between the learned representations of the two models. The coefficients α, β, and γ are hyperparameters that balance the contributions of each loss component in training the student model $\mathcal{S}$.

3.2. Data pre-processing

Data preprocessing is essential to prepare the dataset for effectively training machine learning models. In this study, we process the data streams $\mathbf{X}_n \in \mathbb{R}^{T\times d}$ generated by N IoT devices, where T represents the time series length and d is the number of features extracted per device. The aim is to ensure the data is in a suitable format for the development of an Intrusion Detection System (IDS) capable of classifying data into C classes $\mathbf{y}_n \in \{0, 1, \dots, C-1\}$, where each class corresponds to a type of traffic or attack.

3.2.1. Data cleaning and feature selection

Let $\mathbf{X} = \{\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_N\}$ represent the data collected from N IoT devices. Each $\mathbf{X}_n$ contains time-series data of length T with d features. To ensure the completeness of the dataset, we first remove any unnecessary columns and handle missing values:

$$\mathbf{X}_n \leftarrow \text{impute}(\mathbf{X}_n), \quad \forall n \in \{1, \dots, N\}, \tag{8}$$

where impute$(\cdot)$ refers to the selected imputation method, such as mean or median imputation.

For feature selection, we apply the Minimum Redundancy Maximum Relevance (MRMR) algorithm to select the most informative features $\mathbf{X}'_n \in \mathbb{R}^{T\times d'}$, where $d' \ll d$ is the reduced number of features:

$$\text{MRMR} = \max_{\mathbf{X}'_n \subset \mathbf{X}_n} \left(\sum_{j=1}^{d'} \text{Rel}(X_j, \mathbf{y}_n) - \frac{1}{d'(d'-1)} \sum_{j\neq k} \text{blue}(X_j, X_k) \right), \tag{9}$$

where $\text{Rel}(X_j, \mathbf{y}_n)$ measures the relevance of feature X_j with the target variable $\mathbf{y}_n$, and $\text{blue}(X_j, X_k)$ measures redundancy between features X_j and X_k.

3.2.2. Normalization

Normalization is applied to scale features to a common range, improving the performance of learning algorithms. For each feature X_j in $\mathbf{X}'_n$, we use min–max normalization to map values into the range $[0, 1]$:

$$X'_{n,ij} = \frac{X_{n,ij} - \min(X_j)}{\max(X_j) - \min(X_j)}, \quad \forall i \in \{1, \dots, T\},\ j \in \{1, \dots, d'\}, \tag{10}$$

where $\min(X_j)$ and $\max(X_j)$ represent the minimum and maximum values of feature X_j across all time steps.

3.2.3. Encoding

For any categorical features present in $\mathbf{X}'_n$, we employ one-hot encoding to convert them into a binary format suitable for training:

$$\mathbf{O}_{n,ij} = [0, \dots, 1, \dots, 0] \tag{11}$$

where 1 indicates the category of $X_{n,ij}$. This transformation ensures that the categorical data is represented as numerical vectors, allowing the model to learn from these features effectively.

3.2.4. Splitting data

After preprocessing, the dataset is split into training and testing sets to evaluate model performance. The processed feature set $\mathbf{X}' = \{\mathbf{X}'_1, \dots, \mathbf{X}'_N\}$ and corresponding labels $\mathbf{y} = \{\mathbf{y}_1, \dots, \mathbf{y}_N\}$ are divided as follows:

$$\mathbf{X}_{\text{train}}, \mathbf{y}_{\text{train}}, \mathbf{X}_{\text{test}}, \mathbf{y}_{\text{test}} = \text{split}(\mathbf{X}', \mathbf{y}, \text{train\_size} = 0.8), \tag{12}$$

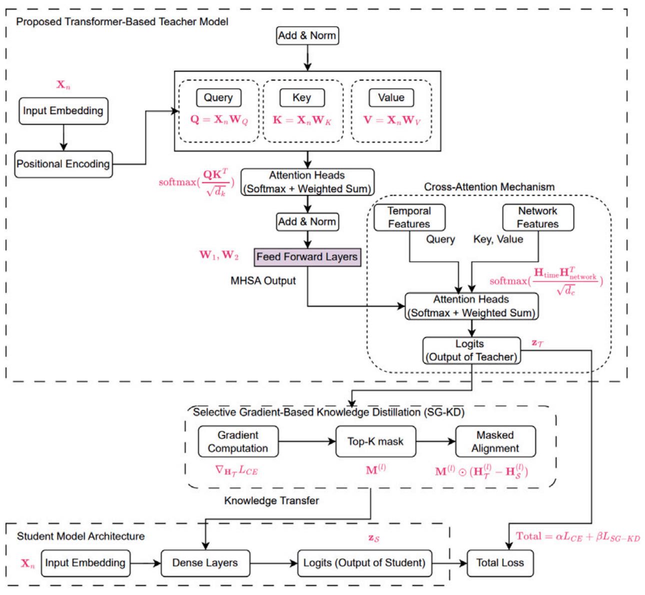
where 80% of the data is used for training and 20% for testing.

3.3. Proposed transformer-based teacher model

The proposed DistillGuard architecture is shown in Fig. 2. The teacher model, denoted as $\mathcal{T}$, utilizes a Transformer architecture to process the input data $\mathbf{X}_n \in \mathbb{R}^{T\times d}$, where T represents the time series length, and d is the number of features extracted per device (Liang et al., 2024). Initially, the input features are projected to a higher-dimensional space:

$$\mathbf{X}'_n = \mathbf{X}_n \mathbf{W}_{\mathcal{T}}^{(proj)} + \mathbf{b}_{\mathcal{T}}^{(proj)}, \tag{13}$$

where $\mathbf{W}_{\mathcal{T}}^{(proj)} \in \mathbb{R}^{d\times d_{proj}}$ and $\mathbf{b}_{\mathcal{T}}^{(proj)} \in \mathbb{R}^{d_{proj}}$ are the weight matrix and bias vector for the projection layer, and d_{proj} is the dimension of the projected space. The Transformer architecture includes a hybrid attention mechanism with both multi-head self-attention (MHSA) and cross-attention layers. The MHSA mechanism is defined as (Chen et al., 2023):



Proposed Transformer-Based Teacher Model
Add & Norm
$\mathbf{X}_n$
Input Embedding
Positional Encoding
Query
$\mathbf{Q} = \mathbf{X}_n\mathbf{W}_Q$
Key
$\mathbf{K} = \mathbf{X}_n\mathbf{W}_K$
Value
$\mathbf{V} = \mathbf{X}_n\mathbf{W}_V$
$\text{softmax}(\frac{\mathbf{QK}^T}{\sqrt{d_k}})$
Attention Heads (Softmax + Weighted Sum)
Add & Norm
$\mathbf{W}_1, \mathbf{W}_2$
Feed Forward Layers
MHSA Output
Cross-Attention Mechanism
Temporal Features
Network Features
Query
Key, Value
$\text{softmax}(\frac{\mathbf{H}_{\text{time}}\mathbf{H}^T_{\text{network}}}{\sqrt{d_c}})$
Attention Heads (Softmax + Weighted Sum)
Logits (Output of Teacher)
$\mathbf{z}_{\mathcal{T}}$
Selective Gradient-Based Knowledge Distillation (SG-KD)
Gradient Computation
Top-K mask
Masked Alignment
$\nabla_{\mathbf{H}_{\mathcal{T}}} L_{CE}$
$\mathbf{M}^{(l)}$
$\mathbf{M}^{(l)} \odot (\mathbf{H}^{(l)}_{\mathcal{T}} - \mathbf{H}^{(l)}_{\mathcal{S}})$
Knowledge Transfer
Student Model Architecture
$\mathbf{z}_{\mathcal{S}}$
$\mathbf{X}_n$
Input Embedding
Dense Layers
Logits (Output of Student)
Total Loss
$\text{Total} = \alpha L_{CE} + \beta L_{SG-KD}$

Fig. 2. Proposed DistillGuard framework for Intrusion Detection.

$$\text{MHSA}_{\mathcal{T}}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Concat}(\text{head}_1, \dots, \text{head}_h)\mathbf{W}_{\mathcal{T}}^{(O)}, \tag{14}$$

with each attention head computed as:

$$\text{head}_i = \text{Attention}_{\mathcal{T}}(\mathbf{QW}_{\mathcal{T},i}^{(Q)}, \mathbf{KW}_{\mathcal{T},i}^{(K)}, \mathbf{VW}_{\mathcal{T},i}^{(V)}), \tag{15}$$

$$\text{Attention}_{\mathcal{T}}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{QK}^\top}{\sqrt{d_k}}\right)\mathbf{V}, \tag{16}$$

where $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$ are the query, key, and value matrices, respectively, and d_k is the dimensionality of the keys.

The cross-attention mechanism is then applied to capture interactions between different aspects of the data, such as temporal sequences and network features (e.g., ‘src_ip’, ‘dst_ip’, ‘proto’, ‘service’, and ‘duration‘). Let $\mathbf{H}_{\text{time}} \in \mathbb{R}^{T\times d_{proj}}$ be the hidden representation of the temporal sequences and $\mathbf{H}_{\text{network}} \in \mathbb{R}^{N_f\times d_{proj}}$ be the representation of the network-related features, where N_f is the number of selected network features. The cross-attention mechanism is defined as:

$$\text{CrossAttention}_{\mathcal{T}}(\mathbf{H}_{\text{time}}, \mathbf{H}_{\text{network}}) = \text{softmax}\left(\frac{\mathbf{H}_{\text{time}}\mathbf{H}_{\text{network}}^\top}{\sqrt{d_c}}\right)\mathbf{H}_{\text{network}}, \tag{17}$$

where d_c is the dimensionality of the cross-attention, and this mechanism helps to align temporal patterns with network features to better identify potential anomalies or patterns indicative of intrusions.

The output from the cross-attention mechanism is combined with the MHSA output:

$$\mathbf{A}_{\mathcal{T}}^{(L)} = \text{LayerNorm}\left(\mathbf{A}_{\mathcal{T}}^{(L-1)} + \text{MHSA}_{\mathcal{T}}(\mathbf{X}'_n) + \text{CrossAttention}_{\mathcal{T}}(\mathbf{H}_{\text{time}}, \mathbf{H}_{\text{network}})\right), \tag{18}$$

where $\mathbf{A}_{\mathcal{T}}^{(L)}$ represents the final output of the Lth layer of the hybrid attention mechanism after normalization.

The final output $\mathbf{A}_{\mathcal{T}}^{(L)}$ is used to compute logits for multi-class attack detection across C classes:

$$\mathbf{z}_{\mathcal{T}} = \mathbf{A}_{\mathcal{T}}^{(L)}\mathbf{W}_{\mathcal{T}}^{(out)} + \mathbf{b}_{\mathcal{T}}^{(out)}, \tag{19}$$

where $\mathbf{W}_{\mathcal{T}}^{(out)} \in \mathbb{R}^{d_{proj}\times C}$ and $\mathbf{b}_{\mathcal{T}}^{(out)} \in \mathbb{R}^{C}$ are the weight matrix and bias vector for the output layer, respectively.

The logits $\mathbf{z}_{\mathcal{T}}$ are then used to compute the predicted class probabilities via the softmax function:

$$\hat{\mathbf{y}}_{\mathcal{T}} = \sigma(\mathbf{z}_{\mathcal{T}}), \tag{20}$$

where $\sigma(\cdot)$ denotes the softmax function, providing the probability distribution over C classes.

3.4. Teacher model training

The training of the teacher model $\mathcal{T}$ is a critical step in preparing it to extract complex patterns from the data, which can later be transferred to the student model through the knowledge distillation process. The teacher model, based on a Transformer architecture with hybrid attention mechanisms, is trained to classify data streams into C classes, where each class represents a type of network traffic or cyber attack, including normal traffic.

3.4.1. Training data and preprocessing

The training data $\mathbf{X}_{\text{train}}$ and labels $\mathbf{y}_{\text{train}}$ are obtained after the preprocessing steps described earlier, where $\mathbf{X}_{\text{train}} \in \mathbb{R}^{N_{\text{train}}\times T\times d}$ represents the preprocessed time-series input data from N_{train} samples, and $\mathbf{y}_{\text{train}} \in \{0,1,\ldots,C-1\}^{N_{\text{train}}}$ contains the corresponding labels for each sample.

3.4.2. Loss function

To train the teacher model $\mathcal{T}$ for multi-class attack detection, we use the cross-entropy loss, which measures the difference between the true labels $\mathbf{y}_{\text{train}}$ and the predicted probabilities $\hat{\mathbf{y}}_{\mathcal{T}}$. The loss function is defined as:

$$L_{\text{CE}} = -\frac{1}{N_{\text{train}}}\sum_{n=1}^{N_{\text{train}}}\sum_{c=0}^{C-1}\mathbb{I}(y_n = c)\log\hat{y}_{\mathcal{T},n,c}, \tag{21}$$

where $\mathbb{I}(\cdot)$ is the indicator function, y_n is the true label of the nth sample, and $\hat{y}_{\mathcal{T},n,c}$ is the predicted probability that the nth sample belongs to class c as output by the teacher model.

3.4.3. Optimization process

The optimization process aims to minimize the cross-entropy loss L_{CE} using gradient descent-based methods. We use the Adam optimizer, known for its adaptive learning rate and ability to converge faster in deep learning models. The parameter update rule for the teacher model is given by:

$$\theta_{\mathcal{T}} \leftarrow \theta_{\mathcal{T}} - \eta\nabla_{\theta_{\mathcal{T}}}L_{\text{CE}}, \tag{22}$$

where $\theta_{\mathcal{T}}$ represents the parameters of the teacher model, η is the learning rate, and $\nabla_{\theta_{\mathcal{T}}}L_{\text{CE}}$ is the gradient of the loss function with respect to the model parameters.

3.4.4. Regularization and early stopping

To prevent overfitting and ensure the generalization of the teacher model $\mathcal{T}$, regularization techniques such as dropout are applied to the model's layers. Dropout randomly deactivates a proportion of neurons during training, which helps improve the robustness of the model:

$$\text{Dropout}(p) = \begin{cases} 0, & \text{with probability } p, \\ \frac{x}{1-p}, & \text{with probability } 1-p, \end{cases} \tag{23}$$

where p is the dropout rate and x represents the activations of a neuron.

Additionally, we employ early stopping, which monitors the validation loss during training and halts the training process when the loss does not improve for a predefined number of epochs. This helps in avoiding overfitting and saves computational resources.

3.4.5. Model evaluation

After training, the performance of the teacher model $\mathcal{T}$ is evaluated on a separate validation set $\mathbf{X}_{\text{val}}$ and labels $\mathbf{y}_{\text{val}}$ using metrics such as accuracy, precision, recall, and F1-score:

$$\text{Accuracy} = \frac{1}{N_{\text{val}}}\sum_{n=1}^{N_{\text{val}}}\mathbb{I}(\hat{y}_{\mathcal{T},n} = y_n), \tag{24}$$

where N_{val} is the number of validation samples, $\hat{y}_{\mathcal{T},n}$ is the predicted class for the nth validation sample, and y_n is its true class.

These evaluation metrics provide insights into the effectiveness of the teacher model in identifying various types of traffic or attacks in the training data. The well-trained teacher model serves as a basis for transferring knowledge to the student model in the subsequent knowledge distillation process.

3.5. Student model architecture

The student model, denoted as $\mathcal{S}$, is designed to be lightweight and efficient, making it suitable for deployment on resource-constrained IoT devices. It is trained using the knowledge distilled from the teacher model $\mathcal{T}$, enabling it to retain the essential knowledge while significantly reducing computational complexity. The student model's architecture consists of multiple dense layers, designed to mimic the outputs of the teacher model while maintaining a simpler structure.

3.5.1. Model structure

The architecture of the student model $\mathcal{S}$ is comprised of a series of fully connected (dense) layers with non-linear activation functions. The input to the student model is the same preprocessed data $\mathbf{X}_n$ as used for the teacher model, but the model itself is more compact. The output of the student model $\mathbf{y}_{\mathcal{S}}$ is computed as:

$$\mathbf{y}_{\mathcal{S}} = \text{Activation}(\mathbf{X}_n\mathbf{W}_{\mathcal{S}} + \mathbf{b}_{\mathcal{S}}), \tag{25}$$

where $\mathbf{W}_{\mathcal{S}} \in \mathbb{R}^{d\times h}$ is the weight matrix, $\mathbf{b}_{\mathcal{S}} \in \mathbb{R}^{h}$ is the bias vector, d is the input feature dimension, and h is the hidden layer size. The activation function is typically a non-linear function such as ReLU (Rectified Linear Unit):

$$\text{ReLU}(x) = \max(0, x), \tag{26}$$

which allows the model to learn complex representations while maintaining computational efficiency.

3.5.2. Output layer

The final layer of the student model produces logits $\mathbf{z}_{\mathcal{S}}$ for multi-class attack detection:

$$\mathbf{z}_{\mathcal{S}} = \mathbf{H}_{\mathcal{S}}^{(L)}\mathbf{W}_{\mathcal{S}}^{(out)} + \mathbf{b}_{\mathcal{S}}^{(out)}, \tag{27}$$

where $\mathbf{H}_{\mathcal{S}}^{(L)} \in \mathbb{R}^{h\times d_{proj}}$ is the output of the last hidden layer, $\mathbf{W}_{\mathcal{S}}^{(out)} \in \mathbb{R}^{d_{proj}\times C}$ is the weight matrix for the output layer, and $\mathbf{b}_{\mathcal{S}}^{(out)} \in \mathbb{R}^{C}$ is the corresponding bias vector. The logits $\mathbf{z}_{\mathcal{S}}$ are transformed into class probabilities using the softmax function:

$$\hat{\mathbf{y}}_{\mathcal{S}} = \sigma(\mathbf{z}_{\mathcal{S}}), \tag{28}$$

where $\sigma(\cdot)$ denotes the softmax function, providing a probability distribution over the C classes:

$$\sigma(\mathbf{z}_{\mathcal{S}})_c = \frac{\exp(\mathbf{z}_{\mathcal{S},c})}{\sum_{k=0}^{C-1}\exp(\mathbf{z}_{\mathcal{S},k})} \quad \text{for } c \in \{0,\ldots,C-1\}. \tag{29}$$

3.5.3. Model simplicity and efficiency

The student model is designed to be simpler than the teacher model, reducing the number of parameters and computational requirements. This simplicity makes it suitable for real-time deployment on IoT devices that have limited processing power, memory, and energy. The compact design allows the student model to retain the core insights of the teacher model while being capable of quick inference in edge computing environments.

3.5.4. Training objective

The student model $\mathcal{S}$ is trained to mimic the outputs of the teacher model $\mathcal{T}$ through a process called knowledge distillation, where the goal is to minimize the distillation loss. The distillation loss $\mathcal{L}_{\mathrm{KD}}$ typically includes a combination of the cross-entropy loss between the student's predictions and the true labels, and a Kullback–Leibler (KL) divergence term that measures the difference between the softened output distributions of the teacher and student models:

$$\mathcal{L}_{\mathrm{KD}} = \alpha L_{\mathrm{CE}}(\mathbf{y}_{\mathrm{train}}, \hat{\mathbf{y}}_{\mathcal{S}}) + \beta L_{\mathrm{KL}}\left(\sigma\left(\frac{\mathbf{z}_{\mathcal{T}}}{T}\right), \sigma\left(\frac{\mathbf{z}_{\mathcal{S}}}{T}\right)\right), \tag{30}$$

where L_{CE} is the cross-entropy loss, L_{KL} is the Kullback–Leibler divergence, α and β are weighting coefficients, and T is the temperature parameter used to soften the teacher's and student's output distributions.

By using this training objective, the student model is guided to replicate the knowledge captured by the teacher model while maintaining the computational efficiency needed for deployment in resource-constrained IoT environments.

3.6. Selective gradient-based knowledge distillation (SG-KD)

In this work, we propose a novel distillation approach called Selective Gradient-Based Knowledge Distillation (SG-KD). Unlike conventional distillation methods, SG-KD leverages the gradients of the teacher model's loss to identify and selectively transfer the most impactful knowledge to the student model. This approach focuses on critical aspects of the teacher's representations, ensuring that the student model learns efficiently while maintaining high performance. The core idea of SG-KD is to ensure that the student model focuses on the most important features learned by the teacher model. Instead of transferring all knowledge, SG-KD selectively distills information by analyzing the impact of different features on the teacher's predictions. The most significant features, identified using gradient-based selection, are prioritized for knowledge transfer, making the student model more efficient while preserving detection accuracy. This approach reduces computational overhead and enhances adaptability for resource-constrained IoT environments. Below, we describe the Gradient-Based Selection Mechanism, which formally defines how SG-KD selects and transfers knowledge.

3.6.1. Gradient-based selection mechanism

The core idea behind SG-KD is to use the gradients of the teacher model's loss with respect to its hidden representations as a guide for selecting which parts of the teacher's knowledge should be distilled to the student. For each layer l of the teacher model $\mathcal{T}$, we compute the gradient of the cross-entropy loss L_{CE} with respect to the hidden representations:

$$\mathbf{G}_{\mathcal{T}}^{(l)} = \nabla_{\mathbf{H}_{\mathcal{T}}^{(l)}} L_{\mathrm{CE}}(\mathbf{y}, \hat{\mathbf{y}}_{\mathcal{T}}), \tag{31}$$

where $\mathbf{H}_{\mathcal{T}}^{(l)}$ represents the hidden representations of the teacher model at layer l, and $\mathbf{G}_{\mathcal{T}}^{(l)}$ denotes the gradient with respect to these representations.

The gradient $\mathbf{G}_{\mathcal{T}}^{(l)}$ is used to create a selection mask $\mathbf{M}^{(l)}$ that highlights the elements of $\mathbf{H}_{\mathcal{T}}^{(l)}$ that have the most influence on the model's output:

$$\mathbf{M}^{(l)} = \mathrm{TopK}(|\mathbf{G}_{\mathcal{T}}^{(l)}|, k), \tag{32}$$

where $\mathrm{TopK}(\cdot, k)$ selects the indices of the top k absolute gradient values, creating a binary mask that retains the k most influential elements of $\mathbf{H}_{\mathcal{T}}^{(l)}$.

3.6.2. Distillation loss

The distillation loss in SG-KD focuses on aligning the student model's hidden representations with the selectively filtered representations of the teacher model. The loss is defined as:

$$\mathcal{L}_{\text{SG-KD}} = \sum_{l=1}^{L} \|\mathbf{M}^{(l)} \odot (\mathbf{H}_{\mathcal{T}}^{(l)} - \mathbf{H}_{\mathcal{S}}^{(l)})\|_2^2, \tag{33}$$

where $\mathbf{H}_{\mathcal{S}}^{(l)}$ represents the hidden representations of the student model at layer l, and $\odot$ denotes element-wise multiplication with the mask $\mathbf{M}^{(l)}$. This loss ensures that the student model learns from only the most critical aspects of the teacher's knowledge, avoiding the transfer of redundant or less significant information.

3.6.3. Combined training objective

In the knowledge distillation process, the teacher model $\mathcal{T}$ is pre-trained to accurately classify the input data using its own cross-entropy loss function. During the distillation phase, the student model $\mathcal{S}$ is trained using a combination of two loss functions:

$$\mathcal{L}_{\mathrm{Total}} = \alpha L_{\mathrm{CE}}(\mathbf{y}, \hat{\mathbf{y}}_{\mathcal{S}}) + \beta \mathcal{L}_{\text{SG-KD}}, \tag{34}$$

where $L_{\mathrm{CE}}(\mathbf{y}, \hat{\mathbf{y}}_{\mathcal{S}})$ is the cross-entropy loss that ensures the student model learns to predict the true labels accurately. $\mathcal{L}_{\text{SG-KD}}$ is the selective gradient-based distillation loss, which helps the student model align its intermediate representations with those of the pre-trained teacher model. α and β are hyperparameters that balance the contribution of the attack detection and distillation losses. The SG-KD approach effectively transfers knowledge from a complex pre-trained teacher model to a more efficient student model, tailoring it to the resource constraints of IoT devices.

4. Performance evaluation

In this section, we present a comprehensive evaluation of the proposed DistillGuard framework. The performance evaluation includes the following key components: experimental setup, evaluation metrics, dataset description, result analysis, explainability through XAI for understanding feature impact, and a comparison with state-of-the-art intrusion detection systems (IDS). For a consistent and fair evaluation, we re-implemented state-of-the-art IDS models in our experimental environment, ensuring they followed our standardized preprocessing pipeline and training setup. While the original implementations may have used different feature selection methods, hyperparameter tuning strategies, or training conditions, our approach maintains uniformity across all models, including DistillGuard. This ensures that all models were trained and evaluated under identical conditions, allowing for a direct and unbiased performance comparison. The details of each component are outlined below. Each of them are listed below:

4.1. Experimental setup

The performance evaluation was conducted using Jupyter Notebook and Anaconda to manage and execute Python code. The tests were performed on a machine running Windows 11, equipped with an Intel Core i7 8750H processor, 16 GB Dual-Channel DDR4 RAM, and an NVIDIA GeForce RTX 2070 with Max-Q Design graphics card. The experimental evaluation used several Python libraries, including `scikit-learn`, `pandas`, `numpy`, `keras`, `tensorflow`, `memory_profiler`, `gc`, and `matplotlib`.

Algorithm 1: Training and Evaluation Procedure of DistillGuard for IoT Intrusion Detection

```
Input  : Time-series data $\mathbf{X}_n \in \mathbb{R}^{T\times d}$ for $N$ IoT devices
Output: Predicted class $\hat{\mathbf{y}}_n$ for each device $n$

1  Step 1: Load Data
2      Load time-series input data $\mathbf{X}_n$ and corresponding labels $\mathbf{y}_n$
3  Step 2: Data Preprocessing
4      Check for missing values, encode categorical features, and
         normalize features using Min-Max scaling
5  Step 3: Feature Selection with MRMR
6      Initialize the MRMR algorithm and select relevant features using
         the equations
7  Step 4: Split Data into Training and Testing Sets
8      Perform train–test split with an appropriate ratio
9  Step 5: Hyperparameter Tuning with Keras Tuner
10 Step 6: Train the Teacher Model $\mathcal{T}$
11     Build a Transformer-based teacher model using the selected
         features $\mathbf{X}'_n$
12     Train $\mathcal{T}$ on clean data and evaluate on the testing set
13 Step 7: Adversarial Training on the Teacher Model
14     Generate adversarial samples using FGSM
15     Fine-tune the Teacher model $\mathcal{T}$ using both clean and adversarial
         samples to improve robustness
16 Step 8: Train the Student Model $\mathcal{S}$ Using SG-KD
17     while expected accuracy not met do
18         foreach epoch do
19             Compute gradient-based masks $\mathbf{M}^{(l)}$ from teacher model:
20                 $\mathbf{M}^{(l)} = \mathrm{TopK}(|\nabla_{\mathbf{H}_{\mathcal{T}}^{(l)}} L_{\mathrm{CE}}(\mathbf{y}_n, \hat{\mathbf{y}}_{\mathcal{T}})|, k)$
21             Calculate the SG-KD loss:
22                 $\mathcal{L}_{\text{SG-KD}} = \sum_{l=1}^{L} \|\mathbf{M}^{(l)} \odot \mathbf{H}_{\mathcal{T}}^{(l)} - \mathbf{M}^{(l)} \odot \mathbf{H}_{\mathcal{S}}^{(l)}\|_2^2$
23             Update student model parameters by minimizing the
                 total loss:
24                 $\mathcal{L}_{\text{Total}} = \alpha L_{\mathrm{CE}}(\mathbf{y}_n, \hat{\mathbf{y}}_{\mathcal{S}}) + \beta \mathcal{L}_{\text{SG-KD}}$
25             Train the student model $\mathcal{S}$ on the selected features $\mathbf{X}'_n$
26     Evaluate the student model $\mathcal{S}$ on the testing set
27 Step 9: Apply Model Compression Techniques (Pruning and
     Quantization)
28     Apply pruning to reduce the size of the Teacher and Student
         models
29     Perform quantization to further optimize model memory and
         computational efficiency
30     Fine-tune both models after pruning and quantization to recover
         potential accuracy loss
31 Step 10: Evaluate and Interpret the Compressed Student Model
32     Test the pruned and quantized Student model on the testing set
         and compare its performance to the original models
33     Use XAI tools like gradient contribution heatmaps to understand
         the model
34     Analyze feature contributions to ensure transparency and trust
         in predictions
35 Output: Final Predicted Classes $\hat{\mathbf{y}}_n$ and Explanation for
     Predictions
```

4.2. Model architecture

The Teacher model in DistillGuard is a Transformer-based architecture designed to capture both temporal and contextual dependencies in network traffic. This model has 6 transformer layers, each containing 8 attention heads in the MHSA mechanism. The hidden dimension for each layer is set to 512 neurons, and the feed-forward layers consist of 2048 neurons. The cross-attention mechanism is designed to capture interactions between temporal sequences and network features (e.g., *src_ip*, *dst_ip*, *proto*, *service*, *duration*). The hidden representations of temporal sequences and network features are projected to a dimensionality of $d_{proj} = 512$, with the number of selected network features $N_f = 5$. The cross-attention operates over these representations, with a dimensionality of the cross-attention mechanism, d_c, set to 512. This allows effective alignment between the temporal patterns and network features for detecting anomalies. The output from the cross-attention mechanism is combined with the output from the MHSA layer. Layer normalization is applied at each layer to stabilize training and prevent overfitting. The final output is computed using a softmax function, which transforms the logits into probabilities for multi-class attack detection across $C = 10$ classes. The weight matrix for the output layer has dimensions 512×10. The model uses a dropout rate of 0.1 to reduce overfitting. The learning rate for training the Teacher model is set to 0.001, and it is optimized using the Adam optimizer. The model is trained for 50 epochs with a batch size of 32. The Student model in DistillGuard is designed for efficiency and is comprised of a series of fully connected (dense) layers. This consists of 3 fully connected (dense) layers. The input feature dimension is $d = 512$, and each dense layer uses $h = 256$ neurons. The activation function for all layers is the ReLU (Rectified Linear Unit), defined as $\mathrm{ReLU}(x) = \max(0, x)$, which ensures non-linearity while maintaining computational efficiency. The Student model is trained using SG-KD. The distillation temperature is set to 2.0, and the loss function combines cross-entropy from the soft targets of the Teacher model and the hard targets from the ground truth. The learning rate for the Student model is 0.0005, optimized using the Adam optimizer. The Student model is trained for 50 epochs with a batch size of 64. We used Grid search to find optimal parameters. The final dense layer of the Student model produces logits $\mathbf{z}_{\mathcal{S}}$ for multi-class attack detection, using a softmax function to compute class probabilities. The weight matrix for the output layer has dimensions 256×10, where 10 (for instance in ToN-IoT) corresponds to the number of output classes.

4.3. Adversarial robustness analysis

To evaluate the robustness of DistillGuard under adversarial conditions, we employed the *Fast Gradient Sign Method (FGSM)*, a widely used technique for generating adversarial examples. FGSM perturbs the input data by leveraging the gradient of the model's loss function with respect to the input features, creating adversarial samples aimed at deceiving the model.

FGSM Formulation: Given an input $\mathbf{x}$, a true label y, and a neural network with parameters $\boldsymbol{\theta}$, the adversarial example $\mathbf{x}_{\text{adv}}$ is computed as:

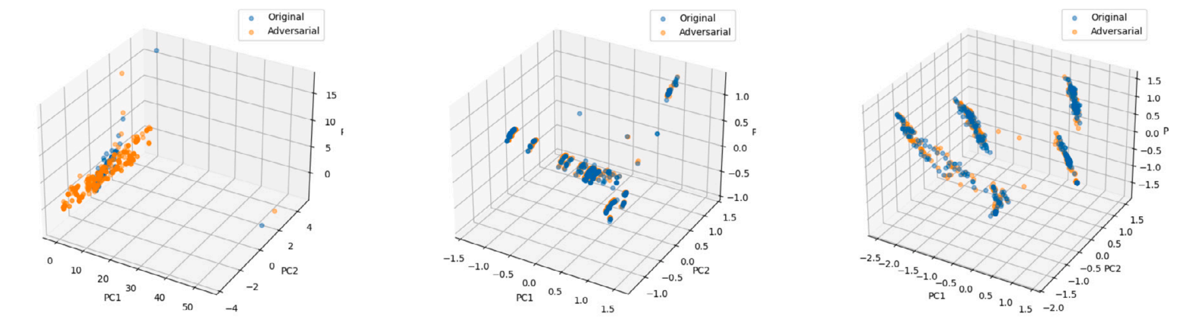
$$\mathbf{x}_{\text{adv}} = \mathbf{x} + \epsilon \cdot \operatorname{sign}\left(\nabla_{\mathbf{x}} J(\boldsymbol{\theta}, \mathbf{x}, y)\right),$$

where, ϵ is the perturbation magnitude, a small scalar value controlling the strength of the attack, $\nabla_{\mathbf{x}} J(\boldsymbol{\theta}, \mathbf{x}, y)$ represents the gradient of the loss function J with respect to the input $\mathbf{x}$, and $\operatorname{sign}(\cdot)$ extracts the sign of the gradient, ensuring uniform perturbation across all input dimensions.

Implementation Details: For this evaluation: We selected a range of ϵ values to simulate varying levels of adversarial perturbations, from subtle changes ($\epsilon = 0.01$) to more aggressive ones ($\epsilon = 0.1$). Adversarial examples were generated for both binary and multi-class attack detection tasks, using the same dataset split as the primary evaluation. The perturbations were applied directly to the input features while ensuring they remained within valid data ranges (e.g., normalizing inputs to maintain feature integrity).

4.4. Evaluation metrics

The performance of the constructed classifiers is evaluated using the testing set. In the domain of IDS, several key metrics are commonly employed to measure the effectiveness of attack detection models. These metrics are based on the Confusion Matrix (CM), which provides a comprehensive analysis of the classifier's ability to distinguish between normal and attack network traffic. The confusion matrix consists of the following terms: (1) *True Negatives (TN)*: Instances where the classifier correctly identifies normal traffic (0) and the actual label is also normal (0). (2) *False Positives (FP)*: Instances where the classifier



(a) Comparison between original and adversarial samples (ToN-IoT)

(b) Comparison between original and adversarial samples (RT-IoT)

(c) Comparison between original and adversarial samples (NbaIoT)

Fig. 3. Comparison of original and adversarial samples for ToN-IoT, RT-IoT, and N-BaIoT datasets.

incorrectly labels normal traffic (0) as an attack (1). (3) *False Negatives (FN)*: Instances where the classifier fails to detect an attack, labeling it as normal (0), while the actual label is attack (1). (4) *True Positives (TP)*: Instances where the classifier correctly identifies attack traffic (1) and the actual label is also an attack (1). Using the values from the confusion matrix, several important evaluation metrics are derived to assess classifier performance, including accuracy, precision, recall, specificity, and F-measure:

- **Accuracy**: This metric represents the overall effectiveness of the classifier by calculating the proportion of correctly classified network samples (both normal and attack) relative to the total number of samples. It is computed as:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{35}$$

  A higher accuracy value indicates that the classifier is effectively distinguishing between normal and attack traffic. The ideal accuracy value of 1 (or 100%) is achieved when all samples are correctly classified.
- **Precision**: Also known as the Positive Predictive Value, precision measures the proportion of correctly predicted attack samples out of all samples that were predicted as attacks. It answers the question: "When the classifier predicts an attack, how often is it correct?" It is calculated as:

$$Precision = \frac{TP}{TP + FP} \tag{36}$$

- **Recall (Sensitivity)**: This metric measures the classifier's ability to correctly identify actual attack samples out of all attack samples present. It answers the question: "When there is an actual attack, how often does the classifier correctly detect it?" It is calculated as:

$$Recall = \frac{TP}{TP + FN} \tag{37}$$

- **Specificity**: Specificity measures the classifier's ability to correctly identify normal traffic, that is, to avoid falsely labeling normal traffic as an attack. It is computed as:

$$Specificity = \frac{TN}{TN + FP} \tag{38}$$

- **F-Measure**: The F-measure (or F1-score) is the harmonic mean of precision and recall, providing a single metric that balances both aspects, especially when dealing with imbalanced datasets. It is calculated as:

$$F\text{-measure} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{39}$$

4.5. Dataset description

The experiments are conducted using three publicly available datasets: the ToN-IoT, RT-IoT, and N-BaIoT datasets, which are widely used in the research community for evaluating machine learning-based intrusion detection systems (IDS). The ToN-IoT dataset covers a broad range of network traffic scenarios, including normal behavior and various modern cyberattacks, such as Backdoor, DDoS, DoS, Injection, Password attacks, Ransomware, MITM (Man-in-the-Middle), Scanning, XSS, and Normal traffic. This dataset is highly regarded for its realistic simulation of attack patterns and regular network behavior, making it a valuable resource for developing and testing IDS models. It serves as an extensive benchmark to assess the effectiveness of machine learning models in detecting and mitigating cyber threats. In addition to ToN-IoT, the RT-IoT dataset is also utilized, providing a different range of attack scenarios. The RT-IoT dataset includes attack classes such as ARP_Poisoning, DDOS_Slowloris, DOS_SYN_Hping, MQTT_Publish, Metasploit_Brute_Force, and various NMAP-based attacks (e.g., NMAP_FIN_SCAN, NMAP_OS_DETECT, NMAP_TCP_SCAN, NMAP_UDP_SCAN, and NMAP_XMAS_TREE). Additionally, it contains benign activities like Thing_Speak and Wipro_Bulb traffic. This dataset offers a complementary perspective, as it focuses on a wide variety of attack types specific to IoT environments, providing a robust framework for evaluating the adaptability and precision of IDS models across diverse scenarios. Furthermore, the N-BaIoT dataset is incorporated to evaluate the frameworks performance in detecting botnet attacks in IoT networks. This dataset includes traffic from compromised IoT devices infected with the Mirai and Gafgyt malware, covering attack types such as Mirai_ACK, Mirai_Scan, Mirai_SYN, Mirai_UDP, and Gafgyt_Combo, Gafgyt_Junk, Gafgyt_Scan, Gafgyt_TCP, and Gafgyt_UDP. The N-BaIoT dataset is particularly valuable for assessing the systems ability to handle large-scale, heterogeneous IoT traffic and detect sophisticated botnet activities. Together, the ToN_IoT, RT-IoT, and N-BaIoT datasets enable comprehensive testing, ensuring that the IDS can effectively detect both traditional and IoT-specific cyber threats across a wide range of scenarios.

4.6. Result analysis

In this subsection, we present a detailed analysis of the attack detection performance for both multi-class and binary attack detection tasks. The result evaluation is based on performance metrics such as precision, recall, F1-score, and confusion matrices, which provide insights into the model's strengths and weaknesses across different attack types.

4.6.1. Adversarial plot analysis

Fig. 3 shows a 3D scatter plot of the original data (blue points) and the adversarial samples (orange points) for ToN-IoT (Fig. 3(a)), RT-IoT (Fig. 3(b)) and N-BaIoT (Fig. 3(c)) datasets. The adversarial samples are generated using the Fast Gradient Sign Method (FGSM), which perturbs the input minimally to mimic the original data while causing misattack detection. The data is projected onto the first three principal components (PC1, PC2, and PC3) to simplify the high-dimensional data

Table 2
Performance evaluation of teacher and student models (ToN-IoT dataset).

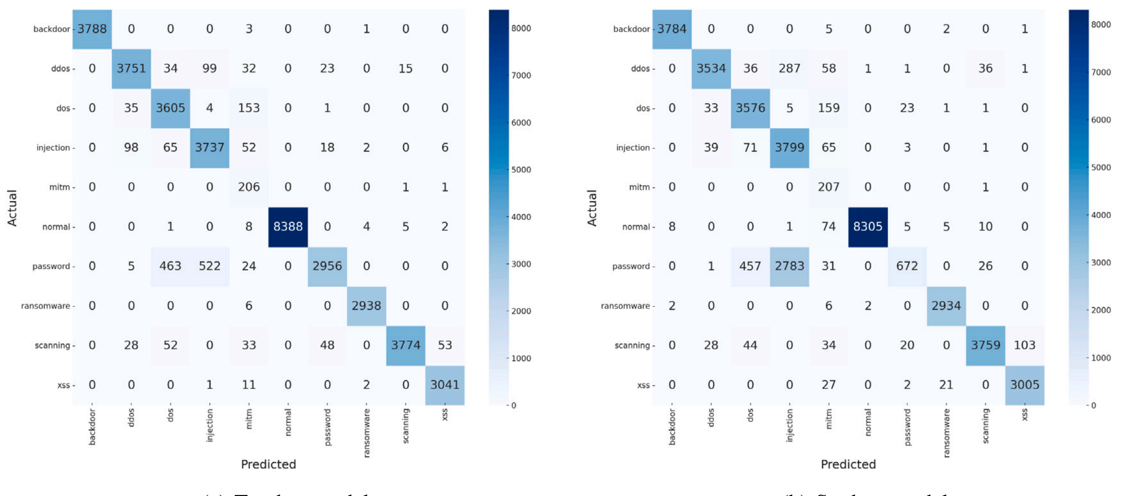
Model	Parameter	Backdoor	DDoS	DoS	Injection	MITM	Normal	Password	Ransomware	Scanning	XSS
Teacher Model	Precision	100.00	95.76	85.43	85.65	39.02	100.00	97.05	99.69	99.45	98.00
	Recall	99.89	94.87	94.92	93.94	99.04	99.76	74.46	99.80	94.63	99.54
	F1-score	99.95	95.31	89.92	89.61	55.98	99.88	84.26	99.75	96.98	98.77
Student Model	Precision	99.74	97.22	85.47	55.26	31.08	99.96	92.56	99.02	98.04	96.62
	Recall	99.79	89.38	94.15	95.50	99.52	98.77	16.93	99.66	94.26	98.36
	F1-score	99.76	93.13	89.60	70.01	47.37	99.37	28.62	99.34	96.11	97.49

Table 3
Performance evaluation of teacher and student models (RT-IoT dataset).

Model	Parameter	ARP _poisoning	DDOS _Slowloris	DOS _SYN _Hping	MQTT _Publish	Metasploit _Brute _Force	NMAP _FIN _SCAN	NMAP _OS _DETECT	NMAP _TCP _scan	NMAP _UDP _SCAN	NMAP _XMAS _TREE	Thing _Speak	Wipro _bulb
Teacher Model	Precision	95.50	75.91	100.00	100.00	100.00	57.14	100.00	99.50	100.00	100.00	97.14	93.62
	Recall	97.23	97.20	100.00	99.76	85.71	66.67	100.00	99.50	92.28	99.75	96.49	86.27
	F1-score	96.36	85.25	100.00	99.88	92.31	61.54	100.00	99.50	95.98	99.88	96.81	89.80
Student Model	Precision	92.45	75.91	99.99	100.00	100.00	66.67	100.00	100.00	99.79	100.00	96.94	100.00
	Recall	97.23	97.20	100.00	99.76	85.71	66.67	100.00	98.00	92.28	99.75	93.71	80.39
	F1-score	94.78	85.25	100.00	99.88	92.31	66.67	100.00	98.99	95.89	99.88	95.30	89.13

Table 4
Performance evaluation of teacher and student models (N-BaIoT dataset).

Model	Parameter	Benign	Gafgyt _Combo	Gafgyt _Junk	Gafgyt _Scan	Gafgyt _TCP	Gafgyt _UDP	Mirai _ACK	Mirai _Scan	Mirai _SYN	Mirai _UDP	Mirai _UDPPlain
Teacher Model	Precision	99.89	99.48	95.11	99.95	76.19	49.87	99.99	100.00	99.98	100.00	99.96
	Recall	99.92	97.41	98.90	100.00	00.08	99.96	99.94	99.99	99.97	99.99	100.00
	F1-score	99.90	98.44	96.97	99.97	00.15	66.54	99.97	100.00	99.97	100.00	99.98
Student Model	Precision	99.92	97.78	98.43	99.90	65.22	49.86	99.99	100.00	99.97	99.99	99.94
	Recall	99.86	99.21	95.39	99.98	00.07	99.95	99.91	99.99	99.96	99.99	99.99
	F1-score	99.89	98.49	96.89	99.94	00.14	66.53	99.95	100.00	99.96	99.99	99.96



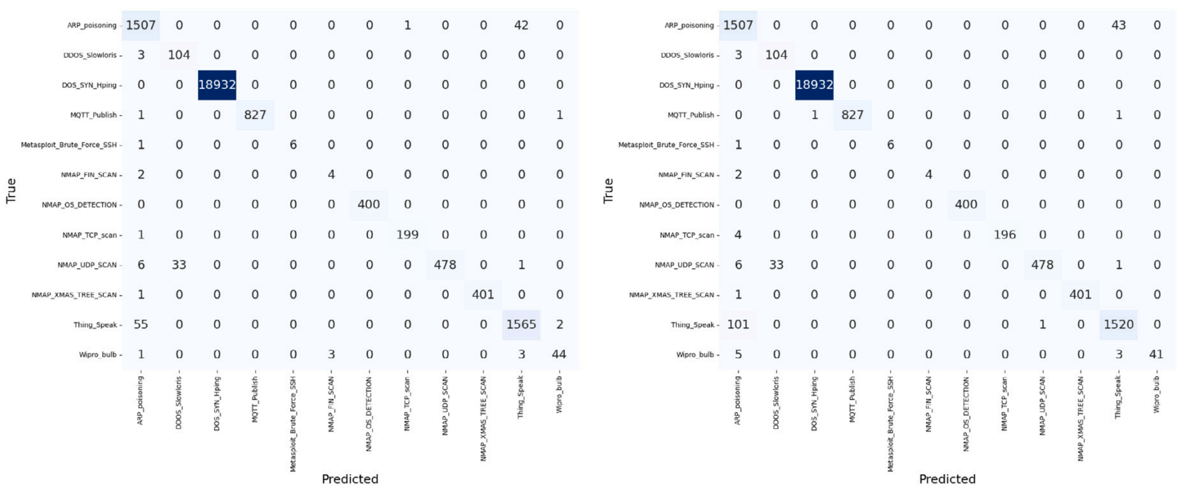
(a) Teacher model (b) Student model

Fig. 4. Confusion matrices for multi-attack detection on the ToN-IoT dataset.

and allow for easy comparison. The plot demonstrates that FGSM-generated adversarial samples are very close to the original data, making them challenging to detect. While the points are similar in the first two components (PC1 and PC2), slight differences in PC3 suggest that the perturbations may primarily affect less important features. This illustrates how FGSM exploits model vulnerabilities to create adversarial samples that are effective yet difficult to identify.

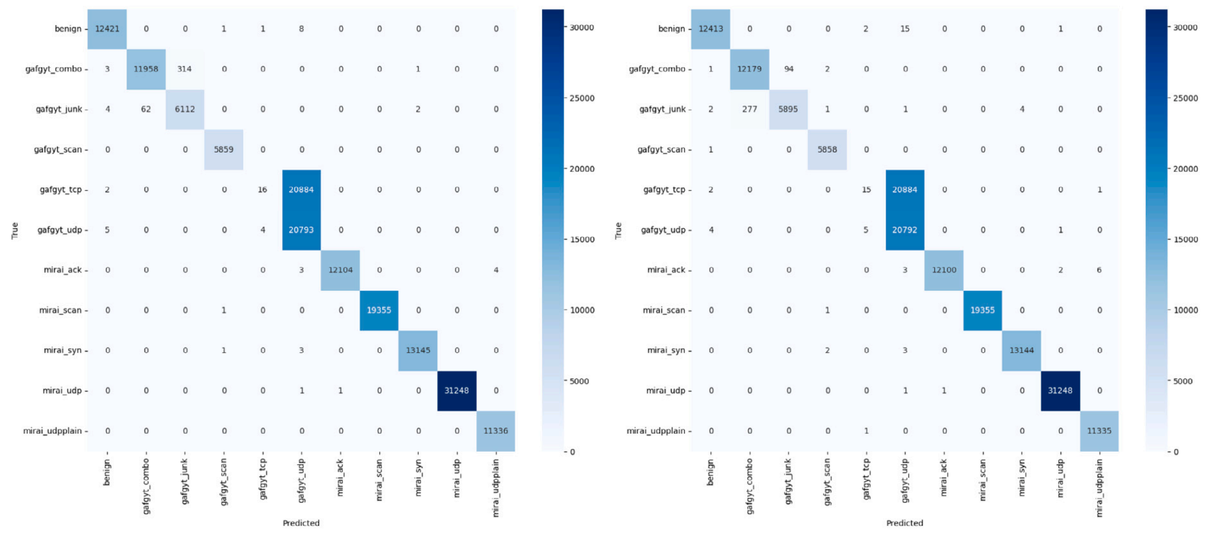
4.6.2. Multi-class attack detection

For multi-class attack detection, Tables 2 and 3 show how KD from the Teacher model improves the training of the Student model for multi-class attack detection. As shown in these tables, the Teacher model generally provides better performance compared to the Student model. *Ton-IoT Dataset:* For example, in the ddos and xss attack classes, the Teacher model yielded precision rates of 95.76% and 98.00%, respectively, while the Student model performed slightly higher precision rates of 97.22% and slightly lower 96.62%. For the ransomware attack class, the Teacher model obtained precision and recall rates of 99.69% and 99.80%, respectively, while the Student model attained slightly lower results with precision and recall rates of 99.02% and 99.66%, respectively. The backdoor attack class had the highest precision rate (100%) in the Teacher model. For the normal class, the Teacher model achieved precision and recall rates of 100% and 99.76%, respectively, while the Student model performed slightly lower in precision and recall rates of 99.96% and 98.77%. RT-IoT Dataset: Similarly, in the



(a) Teacher model (b) Student model

Fig. 5. Confusion matrices for multi-attack detection on the RT-IoT dataset.



(a) Teacher model (b) Student model

Fig. 6. Confusion matrices for multi-attack detection on the N-BaIoT dataset.

RT-IoT dataset, the Teacher model generally outperforms the Student model across various attack classes. For instance, in the ARP_poisoning and DDOS_Slowloris classes, the Teacher model achieved precision rates of 95.50% and 75.91%, respectively, compared to the Student model's precision rates of 92.45% and 75.91%. In the DOS_SYN_Hping class, the Teacher model obtained perfect precision and recall rates of 100.00% and 100.00%, respectively, while the Student model achieved precision and recall rates of 99.99% and 100.00%, respectively. For the MQTT_Publish and Metasploit_Brute_Force classes, both models achieved 100.00% precision, with the Teacher model maintaining higher recall rates of 99.76% and 85.71%, compared to the Student model's recall rates of 99.76% and 85.71%. In the NMAP-related classes, such as NMAP_FIN_SCAN, NMAP_OS_Detect, NMAP_TCP_Scan, NMAP_UDP_Scan, and NMAP_XMAS_Tree, the Teacher model consistently achieved higher precision and recall rates. For example, in the NMAP_FIN_SCAN class, the Teacher model had a precision of 57.14% and recall of 66.67%, whereas the Student model improved the precision to 66.67% while maintaining the same recall rate of 66.67%. Additionally, in the Thing_Speak and Wipro_Bulb classes, the Teacher model achieved precision rates of 97.14% and 93.62%, respectively, while the Student model attained precision rates of 96.94% and 100.00%, respectively. For recall rates in these classes, the Teacher model achieved 96.49% and 86.27%, while the Student model achieved 93.71% and 80.39%, respectively. N-BaIoT Dataset: As shown in Table 4, the Teacher model also demonstrates superior performance on the N-BaIoT dataset. For the Gafgyt_Combo and Gafgyt_TCP attack classes, the Teacher model achieved precision rates of 99.48% and 76.19%, respectively, compared to the Student model's 97.78% and 65.22%. In the Mirai_ACK and Mirai_UDP classes, the Teacher model obtained precision rates of 99.99% and 100.00%, while the Student model achieved 99.99% and 99.99%, respectively. Notably, the Teacher model achieved perfect 100.00% recall for the Gafgyt_Scan class, compared to the Student model's 99.98%. For the Gafgyt_Junk class, however, the Student model improved precision to 98.43% versus the Teacher's 95.11%, while maintaining comparable recall rates. In the Mirai_UDP-Plain class, both models achieved near-perfect performance, with the Teacher model attaining 99.96% precision and 100.00% recall, and the Student model achieving 99.94% precision and 99.99% recall. The low recall rates for the Gafgyt_TCP class (0.08% for Teacher and 0.07%

Table 5
Comparison of evaluation metrics for ToN-IoT and RT-IoT and N-BaIoT datasets, including performance under normal and adversarial conditions (multi-attack detection).

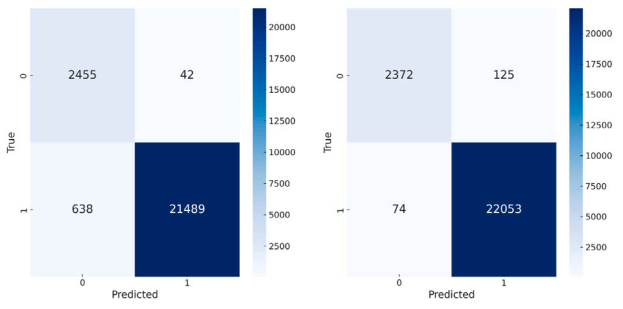
Dataset	Metric	Teacher model	Teacher (Adv) (Pruned+Quant)	Student model	Student KD (From Pruned+Quant)
ToN-IoT	Parameters	630,000	218,767	13,000	8299
	Accuracy	94.98	85.42	88.13	81.56
	F1-score	95.08	84.71	86.57	79.34
	Recall	94.98	85.12	88.13	81.12
	Precision	95.73	86.18	91.85	82.27
	Testing Time (s)	3.5552	3.7421	2.4203	2.6145
	Memory Usage	35.00MB	17.914MB	2.80MB	2.00MB
RT-IoT	Parameters	502,500	283,692	16,000	10,092
	Accuracy	99.36	92.01	99.15	90.34
	F1-score	93.11	89.45	93.17	87.12
	Recall	93.40	89.78	92.56	87.43
	Precision	93.23	89.23	94.31	88.51
	Testing Time (s)	4.6160	4.8623	3.6110	3.7842
	Memory Usage	38.00MB	15.32MB	1.59MB	1.00MB
N-BaIoT	Parameters	3 745 803	1 055 744	16 523	8262
	Accuracy	87.14	87.06	87.13	86.93
	F1-score	87.45	87.20	87.43	86.99
	Recall	90.56	90.15	90.39	90.20
	Precision	92.77	91.28	91.91	93.12
	Testing Time (s)	12.4220	5.3523	4.6110	2.6537
	Memory Usage	75.43MB	40.12MB	2.50MB	1.00MB

Table 6
Comparison of evaluation metrics for ToN-IoT and RT-IoT and N-BaIoT datasets, including performance under normal and adversarial conditions (binary attack detection).

Dataset	Metric	Teacher model	Teacher (Adv) (Pruned+Quant)	Student model	Student KD (From Pruned+Quant)
ToN-IoT	Parameters	499,762	133,792	1410	435
	Accuracy	96.0262	96.181	96.4333	96.1548
	F1-score	96.0124	96.1529	96.4054	96.1313
	Recall	96.0262	96.181	96.4333	96.1548
	Precision	96.0045	96.1509	96.4072	96.1259
	Training Time	193.53 s	18.60 s	51.57 s	16.74 s
	Inference Time (s)	1.43 s	2.151094s	0.52 s	0.643505 s
	Memory Usage	35.91MB	32.999KB	2.31MB	5.506KB
RT-IoT	Parameters	483,218	143,881	3138	1138
	Accuracy	98.6355	93.7842	98.8142	93.1544
	F1-score	98.7165	93.6093	98.8226	93.0361
	Recall	98.6883	93.7842	98.8142	93.1544
	Precision	98.7966	94.0622	98.8367	93.1719
	Training Time	134.42 s	11.86 s	35.74 s	4.45 s
	Inference Time (s)	2.34 s	0.820308 s	0.229328 s	0.223848 s
	Memory Usage	27.32MB	33.001KB	1.59MB	12.266KB
N-BaIoT	Parameters	3 741 186	1 055 744	15 362	7548
	Accuracy	99.98	99.98	99.98	99.98
	F1-score	99.92	99.89	99.87	99.87
	Recall	99.93	99.86	99.80	99.80
	Precision	99.89	99.86	99.94	99.94
	Training Time	687.99 s	284.86 s	80.32 s	30.11 s
	Inference Time (s)	19.81 s	4.78 s	2.95 s	1.92 s
	Memory Usage	45.76MB	6.95MB	1.7MB	0.02MB

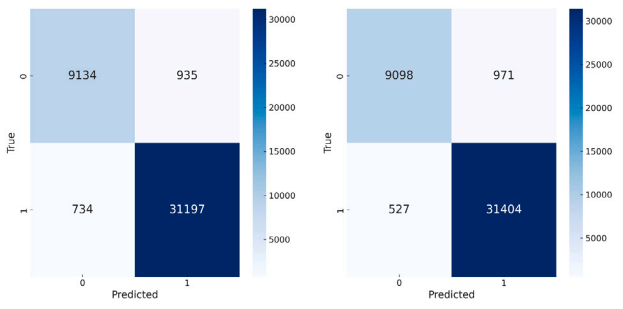
for Student) highlight attack detection challenges specific to this attack type.

The confusion matrices in Fig. 4, 5, 6 show how well the Teacher and Student models classify data in the RT-IoT2022, ToN-IoT, and N-BaIoT datasets. For the ToN-IoT dataset, the Teacher model (Fig. 4(a)) performs well overall. It correctly identifies 3605 cases of the DoS attack and 8388 cases of normal traffic. However, it makes some mistakes, such as 65 Injection cases being classified as Scanning and 44 Scanning cases being misclassified as Password attacks. The Student model (Fig. 4(b)) has similar performance, correctly classifying 3576 DoS cases. However, it misclassifies 71 Injection cases as Scanning and 44 Scanning cases as Password attacks. Despite these small errors, the Student model delivers good results and is more efficient. For the RT-IoT2022 dataset, the Teacher model (Fig. 5(a)) performs well in identifying most types of attacks. For example, it correctly classifies 18,932 cases of the DOS_SYN_Hping attack and 1565 cases of the Thing_Speak attack. However, some mistakes happen, like 33 cases of NMAP_UDP_SCAN being misclassified as DDOS_Slowloris and 55 cases of Thing_Speak being misclassified. The Student model (Fig. 5(b)) shows similar results to the Teacher model. It also correctly classifies 18,932 cases of DOS_SYN_Hping but fewer cases of Thing_Speak (1520). Some errors, like 33 cases of NMAP_UDP_SCAN and 101 cases of Thing_Speak being misclassified, are present. Although slightly less accurate in some areas, the Student model still works well and uses fewer resources, making it suitable for low-power environments. For the N-BaIoT dataset, the Teacher model (Fig. 6(a)) demonstrates strong performance, with high correct attack detections such as 12,421 benign samples and 11,958 Gafgyt_Combo attacks. However, it misclassifies some instances, such as 314 Gafgyt_Combo samples as Gafgyt_Junk and 8 benign samples as Gafgyt_UDP. The Student model achieves comparable results, correctly classifying 12,413 benign samples and improving Gafgyt_Combo detection to 12,179 correct cases. While the Student model (Fig. 6(b)) reduces misattack detections for Gafgyt_Combo (94



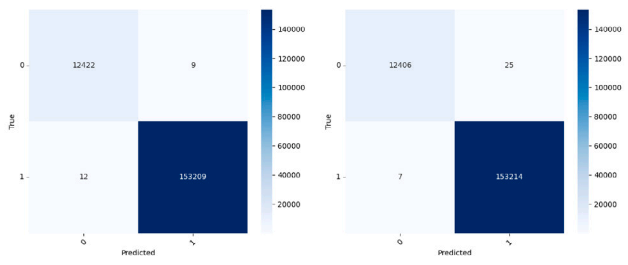
(a) Teacher model (b) Student model

Fig. 7. Confusion matrices for binary attack detection on the ToN-IoT dataset.



(a) Teacher model (b) Student model

Fig. 8. Confusion matrices for binary attack detection on the RT-IoT dataset.



(a) Teacher model (b) Student model

Fig. 9. Confusion matrices for binary attack detection on the N-BaIoT dataset.

misclassified as Gafgyt_Junk vs. 314 in the Teacher model), it introduces minor errors, such as 277 Gafgyt_Junk samples misclassified as Gafgyt_Combo and 15 benign samples misclassified as Gafgyt_UDP. Despite these trade-offs, the Student model maintains robust performance for critical classes like Mirai_UDP (31,248 correct attack detections) and Mirai_UDPPlain (11,335 correct), while remaining resource-efficient for IoT deployments

Table 5 further evaluates the performance metrics between the Teacher and Student models on the ToN-IoT, RT-IoT, and N-BaIoT datasets. ToN-IoT Dataset: The Teacher model achieved an accuracy of 94.98%, while the Student model reached an accuracy of 88.13%, reflecting a decrease of 6.85%. The F1 score, precision, and recall of the Teacher model were 95.08%, 95.73%, and 94.98%, respectively, while the Student model reported an F1 score of 86.57%, precision of 91.85%, and recall of 88.13%. Despite the decrease in these metrics, the Student model significantly improved in terms of efficiency. The testing time for the Teacher model was approximately 3.5552 s, while the Student model required only 2.4203 s, reflecting a 31.04% reduction in testing time. *RT-IoT Dataset:* In contrast, the RT-IoT dataset shows that the Teacher model achieved an accuracy of 99.36%, while the Student model achieved a slightly lower accuracy of 99.15%, indicating a minimal decrease of 0.21%. The F1 score for the Teacher model was 93.11%, compared to 93.17% for the Student model, showing a marginal improvement of 0.06%. Precision metrics favored the Student model, with the Teacher model at 93.23% and the Student model at 94.31%, marking an improvement of 1.08%. However, recall was slightly higher in the Teacher model at 93.40% compared to the Student model's 92.56%, resulting in a decrease of 0.84%. Regarding efficiency, the testing time for the Teacher model was approximately 4.6160 s, whereas the Student model required 3.6110 s, demonstrating a 21.77% reduction in testing time. *N-BaIoT Dataset:* On the N-BaIoT dataset, the Teacher model achieved an accuracy of 87.14%, while the Student model achieved a comparable accuracy of 87.13%, showing a negligible difference of 0.01%. The F1 score for the Teacher model was 87.45%, compared to 87.43% for the Student model, reflecting a minimal decrease of 0.02%. Precision metrics were slightly higher for the Teacher model at 92.77%, while the Student model achieved 91.91%. Recall was also marginally higher for the Teacher model at 90.56% compared to the Student model's 90.39%. In terms of efficiency, the testing time for the Teacher model was approximately 12.4220 s, while the Student model required only 4.6110 s, demonstrating a significant reduction of 62.89% in testing time.

4.6.3. Binary-class attack detection

Table 6 evaluates the performance metrics of the Teacher and Student models, including pruned and quantized versions, on both the ToN-IoT, RT-IoT, and N-BaIoT datasets *ToN-IoT Dataset:* The Teacher model achieved an accuracy of 96.03%, while the Student model reached an accuracy of 96.43%, reflecting an improvement of 0.40%. The F1 score, precision, and recall of the Teacher model were 96.01%, 96.00%, and 96.03%, respectively, while the Student model reported an F1 score of 96.40%, precision of 96.40%, and recall of 96.43%. In terms of efficiency, the Teacher model required approximately 193.53 s of training time and 1.43 s of inference time, compared to the Student model's 51.57 s of training time and 0.52 s of inference time, reflecting significant efficiency gains. The Teacher model utilized 499,762 parameters and 35.91 MB of memory, while the Student model required only 1410 parameters and 2.31 MB of memory. *RT-IoT Dataset:* The Teacher model achieved an accuracy of 98.64%, while the Student model achieved an accuracy of 98.81%, showing an improvement of 0.17%. The F1 score for the Teacher model was 98.72%, compared to 98.82% for the Student model, reflecting a slight improvement of 0.10%. Precision metrics also favored the Student model, with the Teacher model at 98.79% and the Student model at 98.83%, marking an improvement of 0.04%. Recall was slightly higher in the Student model at 98.81% compared to the Teacher model's 98.68%, resulting in an improvement of 0.13%. Efficiency gains were also notable, with the Teacher model requiring approximately 134.42 s of training time and 2.34 s of inference time, while the Student model required 35.74 s of training time and 0.229 s of inference time. The Teacher model utilized 483,218 parameters and 27.32 MB of memory, compared to the Student model's 3138 parameters and 1.59 MB of memory. *N-BaIoT Dataset:* Both models achieved identical accuracy of 99.98% on the N-BaIoT dataset. However, the Teacher model slightly outperformed the Student model in F1-score (99.92% vs. 99.87%) and recall (99.93% vs. 99.80%), while the Student model showed marginally higher precision (99.94% vs. 99.89%). The Student model demonstrated substantial efficiency improvements, requiring only 80.32 s of training time and 2.95 s of inference time compared to the Teacher model's 687.99 s and 19.81 s, representing reductions of 88.33% and 85.11%, respectively. Memory usage was significantly lower for the Student model (1.7MB) compared to the Teacher model (45.76MB), with the pruned+quantized Student variant using only 0.02MB. These results maintain the trend where the Student model achieves near-equivalent performance to the Teacher model while offering dramatic efficiency enhancements,

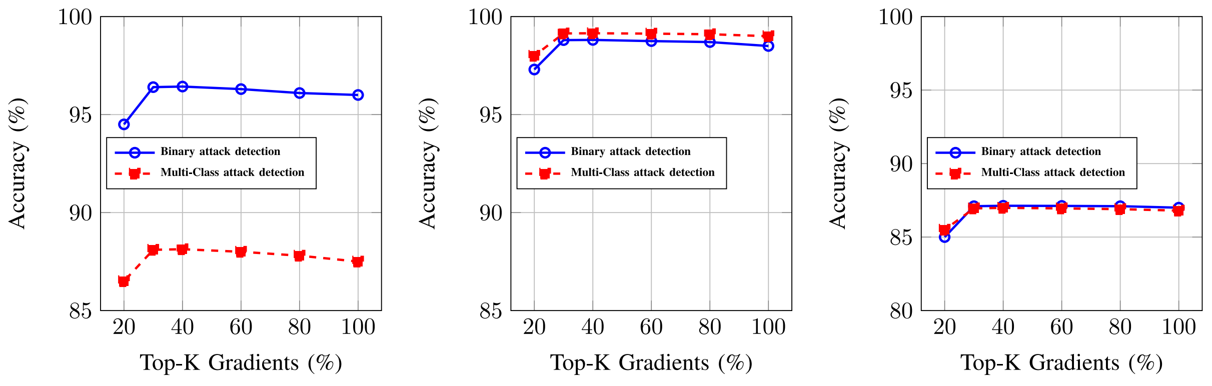
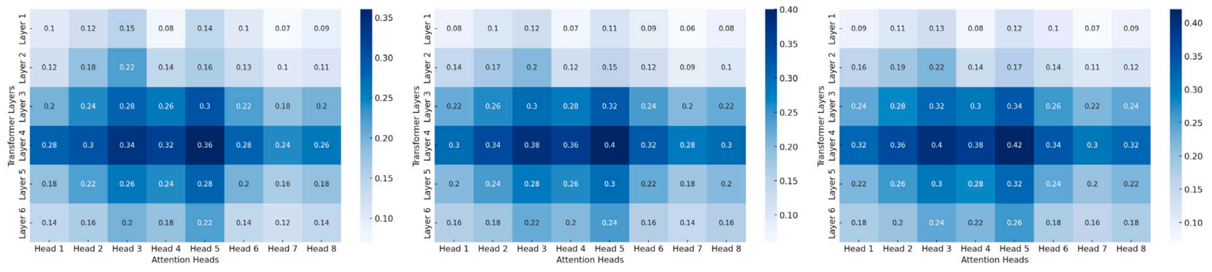


Fig. 10. Accuracy impact of different gradient selection levels for binary and multi-class attack detection across ToN-IoT, RT-IoT, and N-BaIoT datasets. The best performance is achieved at 30%–40% gradient selection, demonstrating that additional gradients provide diminishing returns.



(a) Gradient contribution in ToN-IoT dataset (b) Gradient contribution in RT-IoT dataset (c) Gradient Contribution in N-BaIoT dataset

Fig. 11. Gradient Contribution Heatmaps for different datasets. Darker regions indicate higher contribution of specific transformer layers and attention heads in detecting network anomalies. SG-KD selectively distills high-impact gradients while pruning less significant ones.

particularly critical for high-throughput IoT deployments. These results demonstrate that while the Student model achieves comparable or better performance metrics across the datasets, it also offers significant improvements in efficiency, making it an excellent choice for resource-constrained IoT environments.

Figs. 7, 8, and 9 present the confusion matrices for the binary attack detection performance of the Teacher and Student models on the ToN-IoT, RT-IoT, and N-BaIoT datasets. For the ToN-IoT dataset, the Teacher model (Fig. 7(a)) correctly classifies 9134 normal instances and 31,197 attack instances, with 734 normal instances misclassified as attacks and 935 attack instances misclassified as normal. The Student model (Fig. 7(b)) shows similar performance, correctly classifying 9098 normal instances and 31,404 attack instances, while misclassifying 527 normal instances as attacks and 971 attack instances as normal. The results indicate that the Student model achieves slightly better performance in handling attack traffic while maintaining competitive overall accuracy compared to the Teacher model. For the RT-IoT dataset, the Teacher model (Fig. 8(a)) correctly classifies 2455 normal instances and 21,489 attack instances, with 638 normal instances misclassified as attacks and 42 attack instances misclassified as normal. The Student model (Fig. 8(b)) correctly classifies 2372 normal instances and 22,053 attack instances, with only 74 normal instances misclassified as attacks and 125 attack instances misclassified as normal. For the N-BaIoT dataset, the Teacher model (Fig. 9(a)) correctly classifies 12,422 benign instances and 153,209 attack instances, with 9 benign instances misclassified as attacks and 12 attack instances misclassified as benign. The Student model (Fig. 9(b)) demonstrates improved precision for attack detection, correctly classifying 153,214 attack instances (a slight increase of 5) while misclassifying only 7 attack instances as benign (a reduction of 5 compared to the Teacher). However, the Student model misclassifies 25 benign instances as attacks (versus 9 in the Teacher model), reflecting a trade-off in benign traffic handling.. These results highlight that the Student model performs better than the Teacher model in reducing misattack detection for normal traffic while maintaining a comparable level of accuracy for attack traffic.

4.7. XAI for understanding the DistillGuard model

Fig. 10 illustrates the impact of gradient selection on attack detection accuracy across ToN-IoT, RT-IoT, and N-BaIoT datasets. The results demonstrate that selecting 30%–40% of the most important gradients provides the best trade-off between accuracy and computational efficiency. Beyond this point, using additional gradients yields diminishing returns, while selecting fewer gradients causes a notable drop in accuracy. This validates the effectiveness of the proposed SG-KD, which intelligently filters essential gradients, reducing computational overhead while preserving model performance. The consistency of this trend across multiple datasets highlights the generalization capability of SG-KD, making it highly suitable for real-time intrusion detection in resource-constrained IoT environments.

Fig. 11 illustrates the gradient contribution heatmaps across ToN-IoT, RT-IoT, and N-BaIoT datasets. The heatmaps highlight the importance of specific transformer layers and attention heads in extracting key intrusion detection features. Notably, in all datasets, mid-layer attention heads (Layers 3 and 4) exhibit the highest contributions, reinforcing the importance of intermediate feature representations for attack detection. In contrast, early layers (Layer 1 and 2) contribute less, as they primarily capture lower-level patterns that are not as relevant for classification. Additionally, variations in gradient intensity across datasets confirm that each dataset's attack patterns require different levels of attention focus. The RT-IoT dataset demonstrates greater variability in gradient distribution, likely due to its more diverse set of attack categories, whereas the N-BaIoT dataset emphasizes deep-layer feature contributions, suggesting a higher dependency on long-term network behavior analysis. These findings validate the efficiency of Selective Gradient-Based Knowledge Distillation (SG-KD) in transferring only high-impact knowledge while filtering out less informative gradients. We present these heatmaps for multi-attack detection rather than binary classification because multi-class scenarios pose a greater challenge in distinguishing between diverse attack

Table 7
Binary attack detection performance of IDS models across ToN-IoT, RT-IoT, and N-BaIoT datasets.

Reference	Method name	ToN-IoT				RT-IoT				N-BaIoT			
		Accuracy	Precision	Recall	F1-score	Accuracy	Precision	Recall	F1-score	Accuracy	Precision	Recall	F1-score
Kalaria et al. (2024)	IoTPredictor (HMM)	93.00	92.00	91.00	91.50	94.00	93.00	92.00	92.50	85.10	84.00	83.00	83.50
Hore et al. (2024)	Sequential DNN Framework	93.50	92.40	91.30	91.80	94.50	93.30	92.20	92.70	85.70	84.10	83.20	82.90
Jeffrey et al. (2024)	Ensemble Learning (LSTM-GRU)	94.00	93.00	92.00	92.50	95.00	94.00	93.00	93.50	86.20	85.10	84.30	84.00
Zakariyya et al. (2023)	Optimized DNN	94.50	93.40	92.30	92.80	95.50	94.30	93.20	93.70	86.80	85.50	84.50	84.30
Sahu et al. (2021)	Hybrid CNN-LSTM	94.00	92.90	93.10	93.00	95.00	93.90	93.10	93.50	87.00	86.00	85.90	85.40
Proposed Model	Teacher	96.02	96.00	96.02	96.01	98.63	98.79	98.68	98.71	87.14	92.77	90.56	87.45
Proposed Model	Student	96.43	96.40	96.43	96.40	98.81	98.83	98.81	98.82	87.13	91.91	90.39	87.43

Table 8
Multi-class attack detection performance of IDS models across ToN-IoT, RT-IoT, and N-BaIoT datasets.

Reference	Method name	ToN-IoT				RT-IoT				N-BaIoT			
		Accuracy	Precision	Recall	F1-score	Accuracy	Precision	Recall	F1-score	Accuracy	Precision	Recall	F1-score
Kalaria et al. (2024)	IoTPredictor (HMM)	84.50	83.85	83.13	83.45	85.50	84.75	83.93	84.32	80.20	79.80	78.50	78.10
Hore et al. (2024)	Sequential DNN Framework	84.00	83.45	82.73	83.10	85.00	84.15	83.43	83.85	81.30	80.00	79.20	78.80
Jeffrey et al. (2024)	Ensemble Learning (LSTM-GRU)	85.50	84.85	84.13	84.45	86.50	85.75	84.93	85.32	82.10	81.20	80.00	79.80
Zakariyya et al. (2023)	Optimized DNN	86.00	85.35	84.63	84.95	87.00	86.15	85.43	85.85	83.50	82.30	81.10	80.90
Sahu et al. (2021)	Hybrid CNN-LSTM	86.50	85.75	85.03	85.35	87.50	86.75	85.93	86.12	84.20	83.00	82.10	81.50
Proposed Model	Teacher	94.98	95.73	94.98	95.08	99.36	93.23	93.40	93.11	86.93	93.12	90.20	86.99
Proposed Model	Student	88.13	91.85	88.13	86.57	99.15	94.31	92.56	93.17	86.99	93.00	90.20	86.99

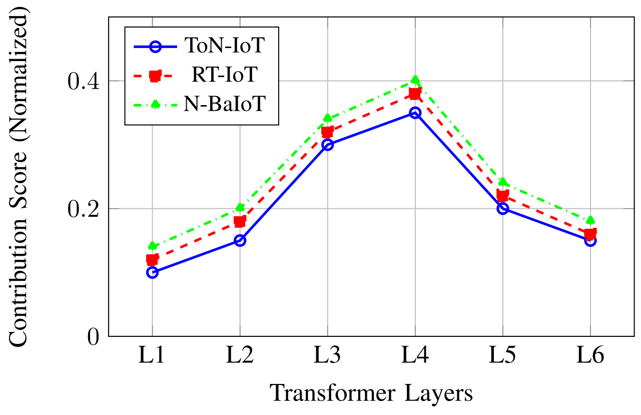


Fig. 12. Layer-wise Contribution of Distilled Knowledge for ToN-IoT, RT-IoT, and N-BaIoT datasets.

types, requiring more precised feature extraction and gradient selection strategies, which directly impact the effectiveness of SG-KD. Fig. 12 presents the layer-wise contribution of distilled knowledge across ToN-IoT, RT-IoT, and N-BaIoT datasets. The results demonstrate a consistent trend: Layers 3 and 4 contribute the most to the knowledge distillation process, indicating that the most valuable attack patterns reside in mid-level feature representations. This aligns with the transformer architecture's ability to capture both local and global dependencies, with intermediate layers playing a central role in understanding attack patterns. Meanwhile, Layer 5 and Layer 6 contribute less, suggesting that deeper layers focus on highly abstract features that may not be necessary for efficient knowledge transfer. The N-BaIoT dataset exhibits the highest deep-layer contributions, likely due to its need for behavioral anomaly detection over extended sequences. These insights validate SG-KD's capability to prioritize critical knowledge from the teacher model while significantly reducing computational overhead in the student model.

4.8. Comparison with state-of-the-art techniques

4.8.1. Comparison with existing IDS for IoT network

Tables 7 and 8, presents a detailed comparison of the proposed models (Teacher and Student) with state-of-the-art intrusion detection systems (IDS) in IoT networks across both ToN-IoT, RT-IoT and N-BaIoT datasets for binary and multi-class attack detection tasks. The proposed Teacher model demonstrates superior performance on the ToN-IoT dataset, achieving an accuracy of 96.02% for binary attack detection and 94.98% for multi-class attack detection. Similarly, the Teacher model achieves 98.63% accuracy for binary attack detection and 99.36% for multi-class attack detection on the RT-IoT dataset. The Student model achieves competitive results, with 96.43% and 88.13% accuracy for binary and multi-class attack detection, respectively, on the ToN-IoT dataset, and 98.81% and 99.15% accuracy for binary and multi-class attack detection, respectively, on the RT-IoT dataset. In addition to accuracy, the proposed models excel in other metrics. For binary attack detection on ToN-IoT, the Student model achieves a precision of 96.40% and a recall of 96.43%, while on RT-IoT, it achieves a precision of 98.83% and a recall of 98.81%. In comparison, other models, such as the Hybrid CNN-LSTM (Sahu et al., 2021), achieve only 93.00% accuracy for binary attack detection and 86.50% accuracy for multi-class attack detection on ToN-IoT. Similarly, the Optimized DNN (Zakariyya et al., 2023) achieves 94.50% accuracy for binary attack detection and 86.00% for multi-class attack detection on ToN-IoT, while its performance on RT-IoT is 95.50% and 87.00% for binary and multi-class attack detection, respectively. Similarly, high performance is seen using N-BaIoT dataset against other approaches.

We have also compared the parameters and memory usage of the proposed DistillGuard against state-of-the-art approaches in Tables 9 and 10. For binary attack detection, the Teacher model utilizes 499,762 parameters with a memory usage of 35.91 MB on ToN-IoT and 483,218 parameters with 27.32 MB on RT-IoT. The Student model, in contrast, requires only 1410 parameters and 2.31 MB of memory on ToN-IoT and 3138 parameters with 1.59 MB on RT-IoT. For multi-class attack detection, the Teacher model scales to 630,000 parameters with 35.00 MB of memory on ToN-IoT and 502,500 parameters with 38.00 MB on RT-IoT, while the Student model uses 13,000 parameters with 2.80 MB on ToN-IoT and 16,000 parameters with 2.00 MB on RT-IoT. Similarly, using N-BaIoT dataset student model uses very parameters (16,523 i.e., in binary and 20,000 i.e., in multi-class) and memory (2.50 i.e., in binary and 3.00 i.e., in multi-class). Overall, the results indicate that the proposed Teacher and Student models provide higher attack detection performance while maintaining lower computational requirements across both datasets, making them highly effective and efficient solutions for intrusion detection in resource-constrained IoT environments.

Table 9
Comparison of IDS model parameters for Binary and Multi-Class attack detection across ToN-IoT, RT-IoT, and N-BaIoT datasets.

Reference	Method name	ToN-IoT		RT-IoT		N-BaIoT	
		Binary	Multi-Class	Binary	Multi-Class	Binary	Multi-Class
Kalaria et al. (2024)	IoTPredictor (HMM)	120,123	155,123	220,000	255,023	280,500	320,000
Hore et al. (2024)	Sequential DNN Framework	220,231	265,231	260,230	295,230	300,000	340,000
Jeffrey et al. (2024)	Ensemble Learning (LSTM-GRU)	160,876	190,876	190,800	220,000	255,000	280,000
Zakariyya et al. (2023)	Optimized DNN	200,569	240,569	240,500	275,000	270,000	310,000
Sahu et al. (2021)	Hybrid CNN-LSTM	300,450	320,450	320,000	355,400	350,000	380,000
Proposed Model	Teacher	499,762	630,000	483,218	502,500	3,745,803	4,000,000
Proposed Model	Student	1410	13,000	3138	16,000	16,523	20,000

Table 10
Comparison of IDS model memory usage (MB) for Binary and Multi-Class attack detection across ToN-IoT, RT-IoT, and N-BaIoT datasets.

Reference	Method name	ToN-IoT		RT-IoT		N-BaIoT	
		Binary	Multi-Class	Binary	Multi-Class	Binary	Multi-Class
Kalaria et al. (2024)	IoTPredictor (HMM)	12.10	14.10	13.80	16.20	15.50	18.00
Hore et al. (2024)	Sequential DNN Framework	22.50	27.50	26.50	31.50	28.50	34.50
Jeffrey et al. (2024)	Ensemble Learning (LSTM-GRU)	18.10	21.50	20.50	24.50	23.50	27.50
Zakariyya et al. (2023)	Optimized DNN	21.50	25.50	24.50	28.50	26.00	30.00
Sahu et al. (2021)	Hybrid CNN-LSTM	30.00	33.00	34.00	38.00	38.00	41.00
Proposed Model	Teacher	35.91	35.00	27.32	38.00	75.43	80.00
Proposed Model	Student	2.31	2.80	1.59	2.00	2.50	3.00

Table 11
Comparison of the proposed DistillGuard with other KD techniques based on accuracy, precision, recall, F1-score, and number of parameters for binary and multi-class attack detection across ToN-IoT, RT-IoT, and N-BaIoT datasets.

Dataset	Method	Parameters		Binary classification				Multi-class classification			
		Binary	Multi-Class	Accuracy	Precision	Recall	F1-score	Accuracy	Precision	Recall	F1-score
ToN-IoT	Proposed DistillGuard	1410	13,000	96.43	96.40	96.43	96.40	88.13	91.85	88.13	86.57
	AKD (Wang et al., 2024)	50,000	55,000	95.80	95.50	95.60	95.55	87.50	91.00	87.00	85.90
	Collaborative KD (Gou et al., 2022)	55,000	58,000	95.60	95.20	95.30	95.25	87.00	90.50	86.50	85.50
	GAN KD (Zhang et al., 2022)	60,000	65,000	95.30	94.90	95.00	94.95	86.80	90.30	86.30	85.30
	LKD-STNN (Zhu et al., 2023)	40,000	45,000	94.80	94.50	94.60	94.55	86.20	89.90	85.90	85.00
RT-IoT	Proposed DistillGuard	3138	16,000	98.81	98.83	98.81	98.82	99.15	94.31	92.56	93.17
	AKD (Wang et al., 2024)	55,000	65,000	97.90	97.50	97.60	97.55	98.50	93.80	91.90	92.80
	Collaborative KD (Gou et al., 2022)	58,000	68,000	97.60	97.20	97.30	97.25	98.10	93.50	91.50	92.30
	GAN KD (Zhang et al., 2022)	64,000	69,000	97.30	96.90	97.00	96.95	97.80	93.20	91.20	91.90
	LKD-STNN (Zhu et al., 2023)	56,000	55,000	96.80	96.50	96.60	96.55	97.40	92.90	90.90	91.50
N-BaIoT	Proposed DistillGuard	8262	15,362	87.13	91.91	90.39	87.43	86.99	93.00	90.20	86.99
	AKD (Wang et al., 2024)	57,000	68,000	86.20	89.50	88.90	86.50	86.00	91.50	89.90	86.50
	Collaborative KD (Gou et al., 2022)	60,000	64,000	85.80	88.90	88.30	85.90	85.50	91.00	89.40	85.90
	GAN KD (Zhang et al., 2022)	66,000	71,000	85.50	88.60	88.00	85.60	85.20	90.80	89.20	85.60
	LKD-STNN (Zhu et al., 2023)	57,000	59,000	85.00	88.00	87.50	85.20	84.90	90.50	88.90	85.30

4.8.2. Comparison with existing knowledge distillation approaches for IoT network

Table 11 presents a detailed comparison of the proposed DistillGuard framework against existing KD methods, including Adaptive Knowledge Distillation (AKD) (Wang et al., 2024), Collaborative KD (Gou et al., 2022), GAN-based KD (Zhang et al., 2022), and LKD-STNN (Zhu et al., 2023) under binary and multi-class attack detection across the ToN-IoT, RT-IoT, and N-BaIoT datasets. The results demonstrate that DistillGuard achieves superior accuracy while maintaining a significantly lower model size compared to other KD techniques. Notably, for the RT-IoT dataset, DistillGuard attains 98.81% accuracy for binary classification and 99.15% for multi-class detection with only 3138 and 16,000 parameters, respectively, while AKD and Collaborative KD require over 50,000 parameters for comparable performance. A similar trend is observed in ToN-IoT and N-BaIoT datasets, where DistillGuard consistently outperforms alternative KD techniques in classification accuracy while reducing computational overhead. Furthermore, DistillGuard significantly improves efficiency, with up to 5× fewer parameters than traditional KD techniques, making it more suitable for deployment in resource-constrained IoT environments. The ability to achieve high classification performance with fewer parameters indicates that the proposed SG-KD effectively transfers only the most relevant knowledge from the teacher to the student model, reducing redundancy and improving inference speed. In contrast, alternative KD techniques such as GAN-based KD and LKD-STNN exhibit marginally lower classification performance while requiring considerably more parameters. This suggests that their knowledge transfer processes may introduce unnecessary complexity, which can be impractical for real-time IoT security applications.

5. Discussion

The proposed DistillGuard framework demonstrates superior performance compared to state-of-the-art Intrusion Detection Systems (IDSs) due to its ability to balance detection accuracy and computational efficiency while ensuring strong generalization across diverse IoT environments. The key factor behind this performance lies in our hybrid Transformer-based teacher model, which effectively captures temporal dependencies and contextual relationships in network traffic using a multi-head self-attention (MHSA) and cross-attention mechanism. These components allow the model to adapt to varied network conditions, attack patterns, and heterogeneous IoT infrastructures, ensuring robustness beyond a single dataset. Unlike traditional IDS models, which often rely on static feature extraction or basic deep learning architectures (e.g., CNNs or DNNs), our Transformer-based teacher model dynamically learns temporal variations and contextual relationships in attack data. This allows the model to generalize well across different IoT environments and identify complex attack strategies, such as those found in N-BaIoT (Mirai and Gafgyt botnet attacks), ToN-IoT (DDoS, MITM, Ransomware), and RT-IoT (Slowloris, MQTT-based attacks, NMAP scanning techniques). Furthermore, SG-KD plays a crucial role in ensuring that high-performance IDS models can be deployed on resource-constrained IoT devices without significant degradation in accuracy. Instead of transferring all learned knowledge from the teacher model to the student, SG-KD selectively distills only the most informative aspects, prioritizing high-impact gradient representations. This reduces redundancy, lowers memory and computational requirements, and ensures that the student model remains lightweight yet effective. Our experimental evaluation supports this claim, showing that the student model—trained via SG-KD—achieves comparable detection performance to the teacher model while significantly reducing model size and memory usage. The trade-offs between model accuracy and efficiency are well-managed, ensuring that even low-power IoT devices can utilize the DistillGuard framework for real-time intrusion detection. To further explain the effectiveness of our model, we leverage Explainable AI (XAI) techniques to analyze knowledge transfer patterns in SG-KD. The Gradient Contribution Heatmaps illustrate that mid-layer attention heads contribute the most to attack detection, ensuring that only the most relevant features are distilled to the student model. Similarly, the Layer-wise Contribution Analysis confirms that knowledge transfer is concentrated in Layers 3 and 4, validating the efficiency of SG-KD in selecting high-impact gradients while minimizing unnecessary computations. Finally, our Gradient Selection Impact analysis highlights that selecting only the top 30%–40% of gradients achieves the best trade-off between accuracy and efficiency, demonstrating the advantage of our selective knowledge transfer approach. These insights validate DistillGuard's interpretability, ensuring that its decisions remain transparent while optimizing resource efficiency for IoT security. While our proposed approach optimizes IDS performance for edge-based IoT security, deploying DistillGuard on real-world IoT environments introduces several challenges: (1) Real-Time Performance & Low Latency and (2) Scalability in Large IoT Deployments. The SG-KD-driven student model is specifically designed to be lightweight, significantly reducing computational complexity. However, to further enhance real-time performance, deploying DistillGuard on hardware accelerators such as Edge TPUs, FPGAs, and NVIDIA Jetson devices can optimize computational efficiency.

6. Conclusion

In this work, we presented DistillGuard, a novel intrusion detection system (IDS) framework designed for efficient and accurate network security monitoring in IoT environments. The proposed approach leverages a Transformer-based teacher model that effectively captures both temporal dependencies and contextual relationships in network traffic using a hybrid attention mechanism. To enhance efficiency while maintaining detection performance, we introduced Selective Gradient-Based Knowledge Distillation (SG-KD), which selectively transfers high-impact knowledge from the teacher model to a lightweight student model. This process significantly reduces computational overhead, making DistillGuard well-suited for resource-constrained IoT devices. Our experimental results demonstrate that DistillGuard achieves superior accuracy and resource efficiency compared to state-of-the-art IDS models, particularly in both binary and multi-class attack detection tasks. The results validate that SG-KD efficiently eliminates redundant knowledge while preserving critical attack detection capabilities and model efficiency, requiring up to 5× fewer parameters while maintaining high classification performance across ToN-IoT, RT-IoT, and N-BaIoT datasets. To understand the model explanation, we integrated XAI-driven analysis of the knowledge distillation process. Our Gradient Contribution Heatmaps reveal that mid-layer attention heads play a crucial role in detecting attacks, guiding the SG-KD process to transfer only the most informative knowledge. Furthermore, the Layer-wise Contribution Analysis confirms that knowledge transfer is concentrated in Layers 3 and 4, ensuring efficient learning for the student model. Finally, the Gradient Selection Impact evaluation shows that selecting the top 30%–40% of gradients achieves the optimal balance between accuracy and efficiency, reinforcing the effectiveness of our selective knowledge transfer approach. Future work will focus on extending DistillGuard to handle more complex multi-stage attacks and adversarial threats, further improving its adaptability to evolving cybersecurity risks. Additionally, we aim to integrate federated learning to enhance privacy and scalability in distributed IoT environments.

CRediT authorship contribution statement

Nadiah AL-Nomasy: Writing – original draft, Investigation, Data curation. **Abdulelah Alamri:** Writing – original draft, Visualization, Formal analysis. **Ahamed Aljuhani:** Writing – original draft, Validation, Supervision, Methodology. **Prabhat Kumar:** Writing – original draft, Validation, Supervision, Methodology, Investigation, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

Abou El Houda, Z., Brik, B., Senouci, S.-M., 2022. A novel IoT-based explainable deep learning framework for intrusion detection systems. IEEE Internet Things Mag. 5 (2), 20–23.

Al-Rubaye, R.H.K., Türkben, A.K., 2024. Using artificial intelligence to evaluating detection of cybersecurity threats in ad hoc networks. Babylon. J. Netw. 2024, 45–56.

Alazawi, S.A.H., Abdulbaqi, H.A., Ali, A.H., 2024. CNN-based intrusion detection software for network operating system environment. Babylon. J. Internet Things 2024, 79–86.

Aljuhani, A., 2022. IDS-chain: A collaborative intrusion detection framework empowered blockchain for Internet of Medical Things. In: 2022 IEEE Cloud Summit. IEEE, pp. 57–62.

Aljuhani, A., Alamri, A., Kumar, P., Jolfaei, A., 2023. An intelligent and explainable SAAS-based intrusion detection system for resource-constrained IoMT. IEEE Internet Things J..

Alrashdi, I., Alqazzaz, A., Alharthi, R., Aloufi, E., Zohdy, M.A., Ming, H., 2019. FBAD: Fog-based attack detection for IoT healthcare in smart cities. In: 2019 IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference. UEMCON, IEEE, pp. 0515–0522.

Ampel, B.M., Samtani, S., Zhu, H., Chen, H., Nunamaker, Jr., J.F., 2024. Improving threat mitigation through a cybersecurity risk management framework: A computational design science approach. J. Manage. Inf. Syst. 41 (1), 236–265.

Barrera, D., Bellman, C., Van Oorschot, P., 2023. Security best practices: A critical analysis using IoT as a case study. ACM Trans. Priv. Secur. 26 (2), http://dx.doi.org/10.1145/3563392.

Chen, J., Song, L., Cai, S., Xie, H., Yin, S., Ahmad, B., 2023. TLS-MHSA: An efficient detection model for encrypted malicious traffic based on multi-head self-attention mechanism. ACM Trans. Priv. Secur. 26 (4), 1–21.

Duan, C., Li, S., Lin, H., Chen, W., Song, G., Li, C., Yang, J., Wang, Z., 2024. IoTa: Fine-grained traffic monitoring for IoT devices via fully packet-level models. IEEE Trans. Dependable Secur. Comput. 21 (4), 3931–3947.

Fu, M., Nguyen, V., Tantithamthavorn, C.K., Le, T., Phung, D., 2023. VulExplainer: A transformer-based hierarchical distillation for explaining vulnerability types. IEEE Trans. Softw. Eng. 49 (10), 4550–4565.

Gou, J., Sun, L., Yu, B., Du, L., Ramamohanarao, K., Tao, D., 2022. Collaborative knowledge distillation via multiknowledge transfer. IEEE Trans. Neural Netw. Learn. Syst..

Hore, S., Ghadermazi, J., Shah, A., Bastian, N.D., 2024. A sequential deep learning framework for a robust and resilient network intrusion detection system. Comput. Secur. 103928.

Jeffrey, N., Tan, Q., Villar, J.R., 2024. Using ensemble learning for anomaly detection in cyber–physical systems. Electronics 13 (7), [Online]. Available: https://www.mdpi.com/2079-9292/13/7/1391.

Kalaria, R., Kayes, A., Rahayu, W., Pardede, E., Salehi, S.A., 2024. IoTPredictor: A security framework for predicting IoT device behaviours and detecting malicious devices against cyber attacks. Comput. Secur. 104037.

Kök, İ., Okay, F.Y., Muyanlı, Ö., Ödemir, S., 2023. Explainable Artificial Intelligence (XAI) for Internet of Things: A survey. IEEE Internet Things J. 10 (16), 14764–14779.

Liang, P., Yang, L., Xiong, Z., Zhang, X., Liu, G., 2024. Multilevel intrusion detection based on transformer and wavelet transform for IoT data security. IEEE Internet Things J. 11 (15), 25613–25624.

Michelena, Á., Aveleira Mata, J.A., Jove, E., Alaiz Moretón, H., Quintián, H., Calvo-Rolle, J.L., 2023. Development of an intelligent classifier model for denial of service attack detection. Int. J. Interact. Multimed. Artif. Intell. 8 (3), 33–42.

Mothukuri, V., Khare, P., Parizi, R.M., Pouriyeh, S., Dehghantanha, A., Srivastava, G., 2022. Federated-learning-based anomaly detection for IoT security attacks. IEEE Internet Things J. 9 (4), 2545–2554.

Pahlevi, R.R., Suryani, V., Nuha, H.H., Yasirandi, R., 2022. Secure two-factor authentication for iot device. In: 2022 10th International Conference on Information and Communication Technology. ICoICT, IEEE, pp. 407–412.

Saed, M., Aljuhani, A., 2022. Detection of man in the middle attack using machine learning. In: 2022 2nd International Conference on Computing and Information Technology. ICCIT, pp. 388–393.

Sahu, A.K., Sharma, S., Tanveer, M., Raja, R., 2021. Internet of Things attack detection using hybrid deep learning model. Comput. Commun. 176, 146–154.

Sun, P., Shen, S., Wan, Y., Wu, Z., Fang, Z., Gao, X.-z., 2024. A survey of IoT privacy security: Architecture, technology, challenges, and trends. IEEE Internet Things J. 1.

Tang, L., Kou, E., Zhang, W., Wu, Q., Chen, Q., 2024. IoT-FKGDL-SL: Anomaly detection framework integrating knowledge distillation and a swarm learning for 5G IoT. IEEE Internet Things J. 1.

Thamer, N., Alubady, R., 2021. A survey of ransomware attacks for healthcare systems: Risks, challenges, solutions and opportunity of research. In: 2021 1st Babylon International Conference on Information Technology and Science. BICITS, IEEE, pp. 210–216.

Wang, L., Yoon, K.-J., 2021. Knowledge distillation and student-teacher learning for visual intelligence: A review and new outlooks. IEEE Trans. Pattern Anal. Mach. Intell. 44 (6), 3048–3068.

Wang, Y., Yu, Z., Wu, J., Wang, C., Zhou, Q., Hu, J., 2024. Adaptive knowledge distillation based lightweight intelligent fault diagnosis framework in IoT edge computing. IEEE Internet Things J..

Zakariyya, I., Kalutarage, H., Al-Kadri, M.O., 2023. Towards a robust, effective and resource efficient machine learning technique for IoT security monitoring. Comput. Secur. 133, 103388.

Zhang, Y., Xing, Y., Liu, Y., Zhang, T., 2022. Distillation knowledge-based space-time data prediction on industrial IoT edge devices. Ad Hoc Netw. 137, 102984.

Zhu, S., Xu, X., Zhao, J., Xiao, F., 2023. Lkd-stnn: A lightweight malicious traffic detection method for Internet of Things based on knowledge distillation. IEEE Internet Things J..

Zhu, S., Xu, X., Zhao, J., Xiao, F., 2024a. LKD-STNN: A lightweight malicious traffic detection method for Internet of Things based on knowledge distillation. IEEE Internet Things J. 11 (4), 6438–6453.

Zhu, T., Ying, J., Chen, T., Xiong, C., Cheng, W., Yuan, Q., Zheng, A., Lv, M., Chen, Y., 2024b. Nip in the bud: Forecasting and interpreting post-exploitation attacks in real-time through cyber threat intelligence reports. IEEE Trans. Dependable Secur. Comput. 1–18.