IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 10 June 2025, accepted 1 July 2025, date of publication 17 July 2025, date of current version 24 July 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3590159

RESEARCH ARTICLE

Bridging Explainability and Security: An XAI-Enhanced Hybrid Deep Learning Framework for IoT Device Identification and Attack Detection

PRABHAV JAIN, ANSHIKA RATHOUR, AASHIMA SHARMA, AND GURPAL SINGH CHHABRA
Department of Computer Science, Thapar Institute of Engineering and Technology, Patiala 147004, India
Corresponding author: Aashima Sharma (aashima.sharma@thapar.edu)

ABSTRACT The rapid expansion of the Internet of Things (IoT) has made accurate and interpretable device identification essential for maintaining network security, managing traffic, and enforcing security policies. However, traditional identification methods, often reliant on handcrafted features or shallow models, struggle to adapt to the highly dynamic and heterogeneous nature of modern IoT environments. They also offer limited transparency, making it difficult for analysts to trust or understand their decisions. To address these challenges, we propose a hybrid machine learning framework that combines deep feature extraction using Convolutional Neural Networks (CNNs) with the robust classification capabilities of XGBoost. This design eliminates the need for manual feature engineering and enhances detection accuracy by learning directly from raw packet-level network data. To ensure interpretability, which is critical in security-sensitive applications, we integrate Explainable AI (XAI) techniques using SHAP values. These explanations help identify the key features that influence each classification decision, increasing trust, and supporting informed actions. We evaluated the proposed system using the CIC IoT2024-DIAD dataset, which includes traffic data from seven device classes and 25 network features. The model achieved strong performance across multiple metrics: 99.92% accuracy, 99.97% F1-Score, 99.62% precision, 99.13% recall, and 99.27% specificity. Regularization and dropout layers were used to mitigate overfitting and ensure generalizability. Unlike existing solutions, our framework balances high accuracy with real-time interpretability, making it scalable, lightweight, and suitable for deployment in resource-constrained IoT settings.

INDEX TERMS Anomaly detection, cybersecurity, device identification, explainable artificial intelligence, IoT detection, machine learning.

I. INTRODUCTION

The Internet of Things (IoT) has revolutionized modern infrastructure, with over 15.14 billion active IoT devices globally as of 2023, a figure projected to surpass 29 billion by 2030 [51]. This surge spans across smart homes, wearables, healthcare, manufacturing, and critical infrastructure, offering immense benefits in automation and intelligent control. However, this explosive growth brings an equally exponential increase in security risks and management complexity due to the sheer heterogeneity of devices and their dynamic communication behaviors.

The associate editor coordinating the review of this manuscript and approving it for publication was Renato Ferrero.

According to Unit 42 (Palo Alto Networks, 2022), 57% of IoT devices are vulnerable to medium or high-severity attacks, and 98% of all IoT traffic remains unencrypted, making devices susceptible to spoofing, manipulation, and takeover [51]. Furthermore, more than 1.5 million IoT-based DDoS attacks were observed in 2022 alone [52], often stemming from unidentified or poorly managed endpoints.

In this context, accurate device identification is critical it forms the foundation for anomaly detection, access control, and threat mitigation. However, traditional rule-based systems are often ineffective in IoT environments due to protocol diversity, device mobility, and evolving traffic patterns.

As shown in Figure 1, research interest in IoT anomaly and device identification surged between 2017 and 2021, with recent trends shifting to more specialized and real-world applications.

To overcome these challenges, this paper presents a hybrid deep learning framework that combines a CNN for feature extraction and XGBoost for classification. The system leverages packet-based and flow-based network attributes such as IP-level metadata, handshake patterns, and HTTPS features to generate a robust behavioral fingerprint for each device.

To promote trust and explainability, we integrate SHAP-based XAI, allowing visualization of feature importance behind classification decisions—an essential step for adoption in security-critical environments.

Extensive experiments on the CICIoT2024-DIAD dataset, a rich benchmark featuring diverse IoT devices and traffic patterns, show that the proposed CNN+XGBoost approach achieves high accuracy and scalability while offering transparent decision-making suited for real-world deployment in resource-constrained networks.

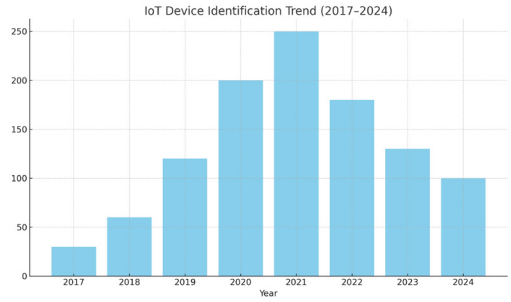


FIGURE 1. Iot device identification trends.

A. MOTIVATION

As the Internet of Things (IoT) ecosystem expands rapidly, the diversity and volume of interconnected devices create significant operational benefits and complex security challenges. These devices, spanning consumer electronics, industrial control systems, and healthcare monitoring equipment, generate unique and dynamic network traffic patterns that are difficult to classify accurately with conventional rule-based methods. The increasing susceptibility of IoT networks to spoofing, manipulation, and malicious intrusions necessitates robust, scalable, and precise device identification mechanisms. However, existing solutions often struggle to handle the heterogeneity and evolving behavior of IoT traffic, limiting their practical applicability. Motivated by the widespread integration of IoT in daily life from smart homes and wearable gadgets to industrial environments and inspired by the rich, real-world traffic scenarios available in the CICIoT2024-DIAD dataset, this work proposes a novel hybrid framework. The proposed system leverages convolutional neural networks for feature extraction and XGBoost for classification, integrated with SHAP-based explainability, to address the critical technical gap in scalable and interpretable IoT device identification. This approach aims to enhance security by accurately fingerprinting devices across diverse, resource-constrained IoT environments.

B. RESEARCH CONTRIBUTION

The main contributions of this research are outlined as follows.:

- This paper proposes a hybrid CNN+XGBoost framework for IoT device identification demonstrating its ability to offer competitive classification performance while significantly reducing computational overhead compared to traditional machine learning models,
- The proposed system extracts both packet-based and flow-based attributes, including HTTPS features, IP-level information, handshake metadata, and User Agent strings, to construct a rich feature set for accurate IoT device fingerprinting. The study integrates t-SNE visualization to provide intuitive insight into the feature space distribution, helping validate the separability of device classes and enhancing model interpretability for network administrators.
- Explainable artificial intelligence technique, SHAP has been deployed to provide interpretability and transparency in the proposed device identification system,
- Performance evaluation of AI models (Adaboost, Random Forest, LightGBM, Logistic Regression,XGBoost, and Extra tree classifier) for device identification using binary and multiclass approaches.
- Analyzing the results on the CIC IoT-DIAD 2024 dataset using various parameters such as accuracy, precision, recall, F1-score, specificity, ROC-AUC, true positive rate, and true negative rate. Furthermore, a detailed attack-wise analysis has been conducted for eight distinct types of attacks

C. ORGANIZATION OF THE PAPER

The rest of the paper is organized as follows:

- **Section II**offers an exhaustive examination of pertinent literature, emphasizing current methodologies in IoT device identification, network traffic analysis, and the utilization of machine learning in IoT security.
- **Section III** outlines the proposed methodology, encompassing data preprocessing steps, feature extraction techniques, and the detailed architecture of the CNN + Hybrid XGBoost model.
- **Section IV** presents the results and discussion, providing an in-depth analysis of the model's performance based on various evaluation metrics and visualizations.
- **Section V** discusses the implementation challenges faced during the study, focusing on system limitations, integration difficulties, and practical constraints encountered during model deployment.
- **Section VI** concludes the paper by summarizing the key findings and their impact on IoT security and device

identification, while also outlining potential directions for future development and research.

II. RELATED WORKS

The literature on IoT device identification and anomaly detection is extensive and multifaceted, reflecting the diverse methodologies and evolving challenges in securing IoT networks. Early works, such as those by Salman et al. [1] and Opoku et al. [2], established the foundation by employing machine learning frameworks to identify IoT devices through their network traffic and to detect irregularities. These studies underscore the necessity of using ML algorithms to decipher the complex patterns inherent in IoT communications, thereby establishing a basis for subsequent research in the field.

A significant body of work has focused on developing models that can differentiate between normal and malicious network behavior. Liu et al. [4] presented a thorough examination of the utilization of machine learning for IoT device detection, emphasizing diverse methodologies ranging from supervised learning techniques to unsupervised systems that operate independently of labeled data. This survey not only cataloged the state-of-the-art methodologies but also identified critical challenges such as data heterogeneity and the dynamic nature of network traffic. Similarly, Gelenbe and Nakıp [5] introduced sequential learning techniques that adapt over time to recognize the evolving patterns of network traffic during botnet attacks, illustrating the potential for real-time threat detection.

Deep learning has become a pivotal factor in enhancing IoT security. Research conducted by Kotak and Elovici [11] and Chowdhury et al. [10] illustrated the effectiveness of deep neural networks in precisely recognizing IoT devices through the acquisition of intricate representations of network traffic. These models have been further refined by incorporating hybrid approaches that combine the strengths of traditional machine learning with the deep feature extraction capabilities of DL, as seen in the works of Santos et al. [12] and Yedilkhan and Smakova [13]. The use of federated learning, as discussed by Wang et al. [31], addresses privacy concerns by enabling distributed training across multiple devices, thus reducing the risk of data breaches while maintaining high accuracy in anomaly detection.

Recent contributions have also focused on the integration of security frameworks with IoT device identification systems. Rabbani et al. [7] and Shahid et al. [8] have merged anomaly detection with device fingerprinting, emphasizing the importance of a unified approach in mitigating cyber threats. These integrated frameworks facilitate not only the early detection of intrusions but also help in the forensic analysis of attacks, as noted by Kumar et al. [22] and Mazhar et al. [23]. The literature also highlights the role of adversarial robustness, with studies like those by Fan et al. [38] and Bao et al. [39] exploring how deep learning models can be fortified against adversarial attacks that attempt to spoof or deceive detection systems. This work explores federated learning-based anomaly detection for distributed IoT networks. Rahmani et al. [50] propose a self-learning adaptive power management scheme that highlights the relevance of intelligent, energy-efficient systems at the edge, aligning with the broader focus on secure and efficient device identification. Several recent works have explored advanced methods for securing and optimizing IoT networks. Ali et al. [54] proposed a blockchain-enabled searchable encryption scheme leveraging neural networks to enhance data confidentiality in industrial IoT healthcare systems. Mohmand et al. [55] presented a machine learning-based approach for accurate classification and prediction of DDoS attacks, demonstrating its effectiveness in network intrusion scenarios. Adil et al. [56] introduced a MAC-AODV-based mutual authentication scheme tailored for constrained IoT environments, ensuring lightweight and secure communication. Khan et al. [57] developed a hybrid framework aimed at enabling seamless and interoperable communication across heterogeneous IoT platforms, addressing interoperability and scalability challenges.

The challenges identified across these studies are manifold. A recurring theme is the heterogeneity of IoT devices, which creates difficulties in developing universally applicable detection models. Moreover, the sheer volume of network traffic generated by IoT ecosystems necessitates scalable and real-time processing solutions. Researchers are actively exploring methods to overcome these hurdles, with proposals ranging from the use of lightweight ML algorithms to advanced distributed learning frameworks. Furthermore, the integration of intrusion detection systems with device identification mechanisms continues to be a focal point, as it offers a promising path toward comprehensive security solutions that can adapt to the dynamic nature of IoT networks. Table 1 encapsulates recent research findings concentrated on IoT device identification and anomaly detection employing diverse AI and machine learning methods. The table highlights each study's objectives, techniques, dataset types, and application scope.

In conclusion, the literature reveals a robust and evolving research field dedicated to enhancing IoT security. From the foundational studies on network traffic analysis to the latest advancements in deep learning and federated learning, researchers are continually pushing the boundaries of what is possible. The dual focus on device identification and anomaly detection is not only critical for safeguarding current IoT infrastructures but also serves as a blueprint for future innovations aimed at securing increasingly complex digital ecosystems.

III. METHODOLOGY

A. DATA DESCRIPTION

The CICIoT2024-DIAD dataset, created by the Canadian Institute for Cybersecurity, supports research in IoT device identification and anomaly detection. This dataset tackles the issues related to the security of Internet of Things (IoT)

TABLE 1. Summary of existing literature.

Reference	Year	Objective	AI/ML Model	Data Type(Packet or Flow Based	Class Type	Attacks	IoT Use
Santos et al. [12]	2018	Device ID & traffic classification	Hybrid (ML+DL)	Both	Multi-class	NA	Device Identification & Traffic Classification
Khan et al. [57]	2021	Interoperability	Hybrid	Flow-Based	Binary	NA	IoT Networks
Fan et al. [38]	2020	Device ID with semi-supervised learning	Semi-supervised learning	Both	Multi-class	NA	Device ID
Mohmand et al. [55]	2022	DDoS Detection	ML	Packet-Based	Binary	3	Device Identification
Bao et al.[39]	2021	Adversarial robustness in ID	DL	Both	Multi-class	Multiple	Device Identification
Kotak & Elovici [11]	2021	Device ID using DL	DL	Packet-Based	Multi-class	NA	Device recognition
Kumar et al. [22]	2021	Traffic classification in IoT	ML Algorithms	Flow-Based	Multi-class	Multiple	Network Traffic Classification
Liu et al. [4]	2021	Survey of ML methods	Survey	Both	Binary	Multiple	Device ID
Gelenbe & Nakıp [5]	2022	Botnet detection	Sequential Learning	Flow-Based	Binary	1	Botnet detection
Mazhar et al. [23]	2022	Forensic analysis of IoT	M2M/ML	Both	Binary	Multiple	Forensic Analysis
Salman et al. [1]	2022	Device ID & anomaly detection	ML Framework	Both	Binary/Multi	Multiple	Device ID & Anamoly Detection
Wang et al. [31]	2023	Federated anomaly detection	Federated DL	Both	Binary	Multiple	Anomaly detection
Opoku et al. [2]	2024	Device ID from traffic	ML	Both	Multi-class	Multiple	Device Identification
Yedilkhan & Smakova [13]	2024	Smart home device ID	ML	Both	Multi-class	Multiple	Device Identification
Rabbani et al. [7]	2024	Integrated ID & anomaly detection	Hybrid ML	Both	Binary/Multi	Multiple	Device Identification & Anamoly Detection
Proposed Framework	2025	Robust IoT device identification with explainability	CNN + XGBoost (Hybrid) + SHAP Explainability	Both	Multi-class	Multiple	Device Identification

devices and networks. It offers an extensive compilation of network traffic data, encompassing both normal and attack traffic situations vital for the development and assessment of security models in IoT contexts.A detailed description of the dataset is provided in Table 2 and the details of the features used in the dataset is provided in Table 3.

The dataset includes several key components, enabling robust evaluation of security solutions in IoT contexts:

a) **Multi-Protocol Data**: The dataset includes traffic from various communication protocols such as Wi-Fi and MQTT, representing real-world IoT communication patterns. The data encompasses both benign and malicious traffic, offering a varied array of events for model training.

b) **Attack-Specific Data**: Instances of several attack types, including Denial-of-Service (DoS) and Distributed Denial-of-Service (DDoS) assaults, are included. This enables the detection of malicious activity targeting IoT devices within a TCP/IP network, which is crucial for building intrusion detection models.

TABLE 2. Dataset details.

Title	CCIoT DIoT 2024 Dataset
Reference	https://www.unb.ca/cic/datasets/ciot.html
Normal Behaviour	Includes legitimate traffic from IoT devices under typical conditions
Abnormal Behaviour	Captures a wide range of cyberattacks including DDoS, Dos, Spoofing, Recon, Backdoor Malware, and more
Protocols	MQTT, Wi-Fi, Bluetooth, HTTP, HTTPS, TCP/IP, UDP
Target Classes	28
Total Features	131
Target OS	Windows OS, Linux-based IoT firmware, Android

c) **Time-Series Data**: The dataset includes time-based records reflecting both active and idle states of IoT devices, allowing models to learn temporal dependencies and behaviors that are critical for accurate anomaly detection.

d) **Device-Specific Profiles**: The dataset profiles the behavior of various IoT devices, particularly focusing on Bluetooth devices in healthcare environments. This enables device identification and classification tasks based on network traffic behavior.

The dataset has a diverse array of features, which can be classified into packet-based and flow-based categories:

e) **HTTPS-Specific Features**: Features derived from HTTPS handshakes, including User-Agent strings, are included to facilitate the analysis of secure communication protocols.

f) **Stream, Channel, and Jitter Metrics**: The dataset includes calculated stream and jitter metrics over different time intervals, which are valuable for identifying network anomalies.

g) **Flow-Based Metrics**: The flow-based features capture packet counts, byte counts, inter-arrival times, and other network traffic patterns, providing key insights into the underlying communication behavior.

These features are crucial for training machine learning models, enabling them to effectively identify and mitigate cyber threats in IoT networks.The dataset serves as a significant asset for various applications in IoT security:

h) **IoT Device Identification**: The information facilitates the classification of IoT devices according to their distinctive network traffic patterns, essential for device authentication and access management.

i) **Anomaly Detection**: By providing labeled instances of both normal and attack traffic, this dataset supports the development of models that can detect deviations from expected behaviors, signaling potential security threats.

j) **Security Analytics**: The dataset facilitates the monitoring and response to emerging cyber threats in IoT environments, hence augmenting the overall security posture of IoT networks.

Figure 2 illustrates the distribution of different attack types present in the dataset using a pie chart. It visually highlights the proportion of each attack category, helping to understand the balance and variety of data samples used for analysis. This aids in evaluating the dataset's diversity and potential model bias.

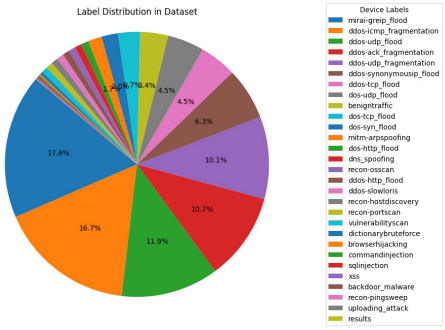


FIGURE 2. Attack distribution in the dataset.

B. DATA LOADING

The raw data for this study was organized across multiple subdirectories within a main dataset folder. Each subdirectory represented a distinct class label. An automated script was employed to efficiently traverse the directory structure, identify all `.csv` files, and load them into memory.

During this process, the script extracted the immediate parent folder name of each CSV file to assign it as the corresponding label for the data. Each CSV file was read using the `pandas` library, and a new `Label` column was appended to indicate its class based on the folder it originated from. Once all individual datasets were labeled, they were concatenated into a single unified DataFrame to form the final structured dataset.

The merged and labeled dataset was saved as `output_labeled.csv` for future use in training and evaluation. This approach ensured that all samples were correctly tagged and consolidated, providing a clean and organized input for the subsequent stages of the machine learning pipeline.

C. DATA PREPROCESSING

Data preprocessing is vital for ensuring model performance and involves several key steps:

TABLE 3. Selected features with their associated names and descriptions.

S.No.	Feature Name	Description
1	inter_arrival_time	Time Gap Between Consecutive Packets
2	stream	Stream Identifier for Grouping Packets
3	channel_30_mean	Mean Value in Channel (30s window)
4	stream_jitter_60_sum	Sum of Jitter in Stream (60s window)
5	stream_60_mean	Mean of Packet Stream (60s window)
6	l3_ip_dst_count	Count of Layer 3 Destination IP Addresses
7	sum_p	Summed Packet Size or Count
8	stream_jitter_30_sum	Sum of Stream Jitter (30s window)
9	src_ip_30_mean	Mean Metrics for Source IP (30s window)
10	src_ip_60_count	Count of Source IP Packets (60s window)
11	average_p	Average Packet Size
12	stream_30_mean	Mean Metrics of Packet Stream (30s window)
13	src_ip_60_mean	Mean of Source IP Features (60s window)
14	channel_60_var	Variance in Communication Channel (60s window)
15	src_ip_10_count	Count of Source IP Metrics (10s window)
16	stream_jitter_5_mean	Mean Jitter of Stream (5s window)
17	most_freq_spot	Most Frequently Occurring Value or Spot
18	channel_60_mean	Mean of Communication Channel (60s window)
19	src_ip_mac_5_count	Count of IP-MAC Pair Metrics (5s window)
20	src_ip_5_mean	Mean of Source IP (5s window)
21	src_ip_mac_60_count	Count of IP-MAC Pair Metrics (60s window)
22	stream_jitter_60_mean	Mean Jitter in Stream (60s window)
23	channel_10_count	Count in Communication Channel (10s window)
24	channel_60_count	Count in Communication Channel (60s window)
25	channel_30_var	Variance of Communication Channel (30s window)

1) **Feature Importance-Based Selection** Following data loading, the merged dataset underwent feature importance-based filtering to reduce dimensionality and improve model efficiency. Owing to the large dataset size, a chunk-wise processing strategy was adopted to mitigate memory constraints and maintain computational feasibility.

a) **Label Encoding:**
Each data chunk was split into features and the target label. Object-type (categorical) features were converted into numerical representations using the `LabelEncoder`. The target class labels were also encoded to maintain uniformity across chunks.

b) **Feature Importance Computation and Bias Prevention:** A `RandomForestClassifier` was independently trained on each chunk, leveraging the model's built-in feature importance attribute to evaluate the contribution of each feature toward classification. The feature importance values from all chunks were aggregated and averaged to compute a global ranking of features. However, to avoid bias and data leakage during evaluation, we applied feature selection strictly within the cross-validation process.
Let $D = \{(x_i, y_i)\}_{i=1}^{N}$ denote the full dataset of N samples, where $x_i \in \mathbb{R}^d$ is the d-dimensional input feature vector and y_i is the class label. The dataset was split into $k = 5$ folds: $\{D_1, D_2, \ldots, D_5\}$. For each iteration $j \in \{1, \ldots, 5\}$:

i) The training set $D_{\text{train}}^{(j)}$ was formed by combining $k-1$ folds, while the remaining fold $D_{\text{val}}^{(j)}$ was held out as validation data.

ii) A Random Forest model $RF^{(j)}$ was trained on $D_{\text{train}}^{(j)}$ using Gini impurity as the splitting criterion. Feature importance was computed as:

$$I_f = \sum_{t \in T_f} \Delta i(t) \tag{1}$$

where I_f is the importance score of feature f, T_f is the set of decision tree nodes that use feature f, and $\Delta i(t)$ is the decrease in impurity caused by node t.

iii) The top-k features (with $k = 25$ in our experiments) were selected from $D_{\text{train}}^{(j)}$.

iv) Both $D_{\text{train}}^{(j)}$ and $D_{\text{val}}^{(j)}$ were transformed using only these selected features, and subsequently passed through the CNN-XGBoost pipeline for training and evaluation.

This ensures that the validation fold $D_{\text{val}}^{(j)}$ remains completely unseen during feature selection. This pipeline guarantees that no information from the validation or test set leaks into the feature selection process, thus maintaining the integrity and fairness of model evaluation.
We also performed an ablation analysis by comparing the results of two approaches:

- **Biased (Incorrect) Feature Selection:** Feature importance computed on the entire dataset before splitting, leading to label leakage.
- **Unbiased (Correct) Feature Selection:** Feature importance computed separately for each training fold.

The biased approach showed a marginal increase in accuracy (0.3%), confirming that it introduces optimistic evaluation. Therefore, we adhered to the unbiased protocol for trustworthy performance reporting.

c) **Top Feature Selection:**
The top 25 features as seen in Table 3, determined by their mean importance scores, were retained for downstream model training. These features were saved to a text file for reproducibility, and their relative significance was visualized using a bar plot. The complete ranked list of features was stored in CSV format for transparency.

Figure 3 presents the top 25 most important features selected based on their contribution to the model's performance. It helps highlight which features have the greatest influence on the classification decisions, guiding efficient feature selection for improved accuracy and reduced complexity.

2) **Correlation Analysis**
Before proceeding to normalization, a correlation matrix was computed for the selected top 25 features. The Pearson correlation coefficient was employed to assess the linear relationship between feature pairs:

$$r_{xy} = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum(x_i - \bar{x})^2}\sqrt{\sum(y_i - \bar{y})^2}} \tag{2}$$

This step ensured that multicollinearity issues were identified and, if necessary, highly correlated redundant features were flagged for potential removal. A heatmap was plotted to visually inspect the inter-feature correlations, offering insight into the relationships among the selected variables. Figure 4 illustrates a correlation heatmap of the dataset features. It highlights the strength and direction of linear relationships between feature pairs, helping to identify multicollinearity and guiding the feature selection process for model training.

3) **Missing Value Treatment:**
Throughout the preprocessing pipeline, missing values were carefully addressed to ensure the integrity and reliability of the dataset. Missing values primarily appeared during the conversion of non-numeric (object-type) columns to numerical formats. If the conversion process encountered an invalid string or an unconvertible value, `NaN` (Not a Number) placeholders were introduced.
To maintain the quality of the dataset, any rows containing `NaN` values were **dropped** immediately after each conversion step. This was applied consistently during both the feature selection and feature scaling phases.
This approach ensured that only complete and valid data points were used for model training, which is critical for avoiding unpredictable behaviors during learning and evaluation. Although imputation techniques can be applied in other contexts, the decision to drop rows was justified in this case due to the large volume of data, which minimized the impact of discarded samples on the overall dataset quality.

4) **Feature Scaling:**
To ensure uniformity across feature magnitudes and to optimize the convergence behavior of machine learning algorithms, the selected features were normalized using `StandardScaler` in a two-step process:

a) **Fitting the Scaler:**
In the first pass, the top 25 features were loaded chunk-wise. Non-numeric columns, if present, were converted to numeric using pandas' coercion method. Rows containing NaN values, which may have resulted from this conversion, were excluded. The `partial_fit()` function of `StandardScaler` was used to iteratively compute the global mean and variance across the entire dataset.

b) **Applying the Transformation:**
In the second pass, the data was reloaded, cleaned using the same procedures, and transformed using the previously fitted scaler. The scaled features were then concatenated and paired with their corresponding class labels for future training and evaluation.

This two-step scaling approach guaranteed consistent and memory-efficient normalization, ensuring that the input features conformed to a standard normal distribution (zero mean and unit variance).

Figure 5 shows the dataset after applying feature scaling. It visualizes how the features have been standardized, ensuring that they all have a similar scale, which is crucial for improving the performance and stability of machine learning models, especially for algorithms sensitive to feature magnitudes.

D. Dimensionality Reduction and Visualization

To validate the discriminative potential of the selected features and to gain visual insight into the data structure, t-Distributed Stochastic Neighbor Embedding (t-SNE) was applied.

The t-SNE algorithm reduced the high-dimensional scaled features into a two-dimensional space using the following key parameters:

- **Perplexity:** 30
- **Number of Iterations:** 1000
- **Random State:** 42 (for reproducibility)

The 2D projections were plotted using scatter plots, where each point was colored according to its class label. This visualization helped to assess the natural clustering of the

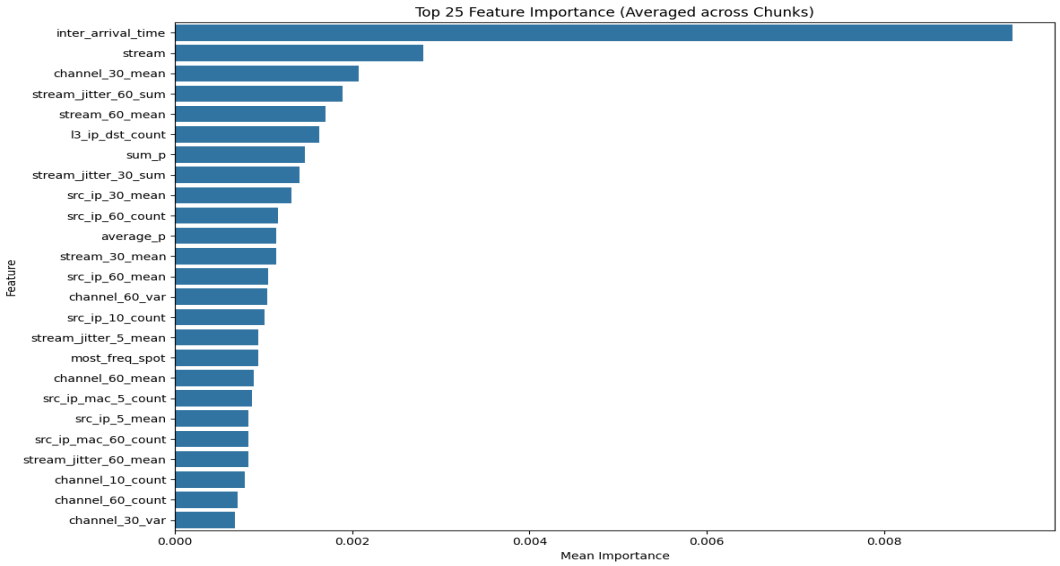


FIGURE 3. Feature importance based feature selection.

TABLE 4. Hyperparameters of different models used.

Model	Hyperparameters
Random Forest	n_estimators=100, max_depth=7, criterion='gini', random_state=42
Extra Trees Classifier	n_estimators=100, max_depth=6, criterion='gini', random_state=42
LightGBM	objective='multiclass', num_class=19, n_estimators=200, learning_rate=0.2, num_leaves=41
AdaBoost	base_estimator=DecisionTreeClassifier(max_depth=2), n_estimators=50, learning_rate=0.1, random_state=None
Logistic Regression	multi_class='multinomial', solver='lbfgs', max_iter=1000, tol=1e-4, C=1.0
XGBoost	n_estimators=100, max_depth=10, objective='multi:softmax', num_class=19, learning_rate=0.1

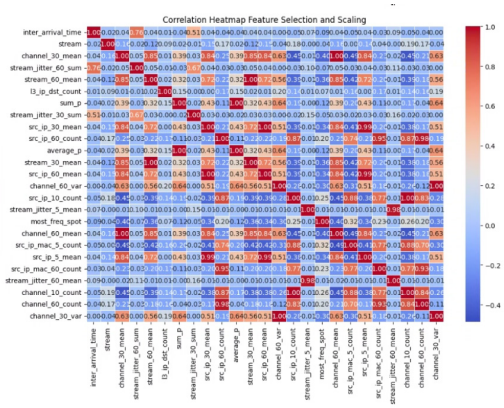


FIGURE 4. Correlation heatmap after data preprocessing.

classes and to verify the effectiveness of the feature selection and scaling processes.

The t-SNE plot provides a visual representation of how the high-dimensional dataset has been reduced as seen in Figure 6, highlighting the underlying structure and separability of the data points. This technique is particularly useful for understanding the relationships between data points and identifying clusters, especially in complex datasets with multiple features.

D. PROPOSED CNN-XGB HYBRID FRAMEWORK FOR DEVICE IDENTIFICATION

Device Identification plays a vital role in network security, particularly in IoT environments, by identifying irregularities in traffic patterns that may signify potential intrusions or malicious behavior. In this work, a hybrid approach was employed that leverages both deep learning and ensemble-based machine learning methods to detect and classify devices within network traffic data to improve detection accuracy and generalization. The process consists of two main stages: training a CNN and testing with XGBoost.

To provide a fair benchmark, we implemented a baseline Multi-Layer Perceptron (MLP) model for comparison.

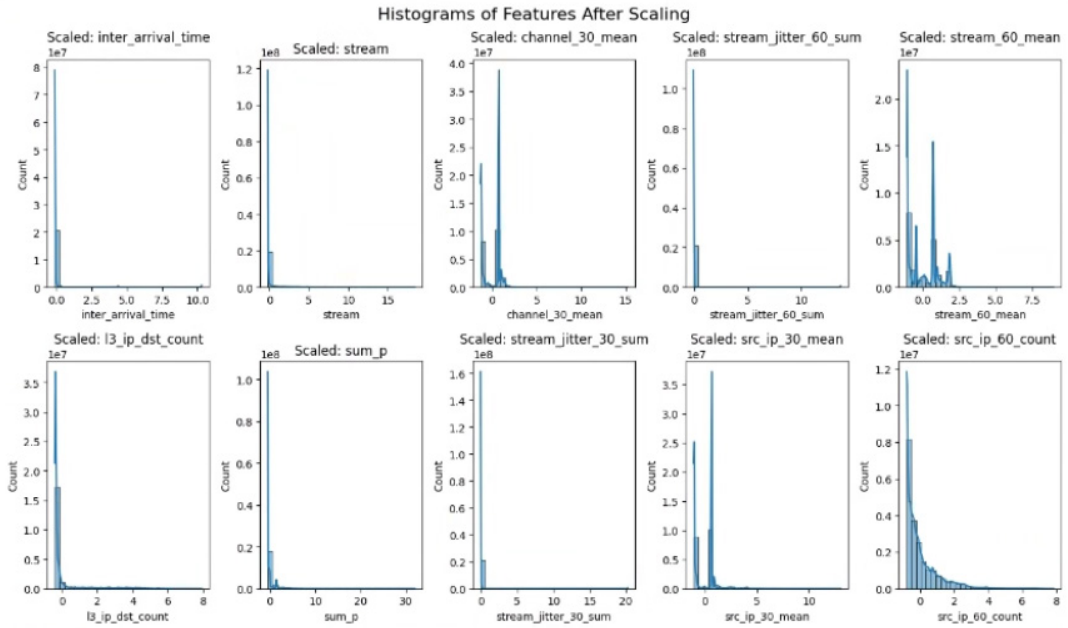


FIGURE 5. Histograms of selected features after Min-Max scaling. Each subplot illustrates the distribution of a specific feature post-scaling.

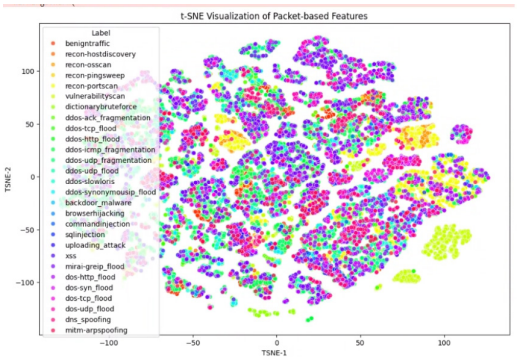


FIGURE 6. t-sne Visualization of packet-based features across all attack classes.

MLPs, while widely used for tabular data, lack the inductive bias needed to capture localized interactions between features. Our empirical evaluation revealed that the CNN model not only outperformed MLP in terms of overall accuracy but also significantly improved recall and F1-scores for rare attack classes such as Backdoor Malware and Command Injection. These findings highlight the representational strength of CNNs in learning complex patterns from tabular cybersecurity datasets where temporal and spatial locality is implicitly embedded in feature arrangements.

Although CNNs are conventionally applied to image or time-series data, recent studies have demonstrated their effectiveness in modeling structured tabular data with spatially relevant features. In this case, several features (e.g., inter-arrival time, payload entropy, and TTL) exhibit locality-sensitive patterns. CNNs enable localized filter operations across adjacent features, capturing hierarchical representations that traditional MLPs fail to model effectively. Empirical comparisons show the CNN-based model achieves higher accuracy and substantially better F1-scores, particularly for minority attack classes, thus justifying its use over MLPs.

As summarized in Table 4, the CNN-XGBoost pipeline significantly outperforms the MLP baseline, especially in detecting rare attacks, demonstrating its superiority in modeling complex dependencies in cybersecurity data for device identification in IoT environments.

a) Stage 1: Training Phase - CNN Although Convolutional Neural Networks (CNNs) are conventionally applied to image or sequential data, we adapted them to process tabular packet-based features by leveraging their ability to learn hierarchical feature representations automatically. The CNN architecture combines fully connected dense layers with batch normalization and dropout layers to stabilize training and prevent overfitting.

1) **CNN Architecture Details:**

- **Input layer:** Accepts the selected top 25 features as input vectors (dimension = 25).
- **Hidden layers:**
  - -- **First hidden layer:** Fully connected dense layer with 64 neurons, activated by ReLU (Rectified Linear Unit).
  - -- **Batch Normalization:** Applied after the first dense layer to normalize activations and accelerate training.
  - -- **Dropout:** Dropout with a rate of 0.3 applied to reduce overfitting.

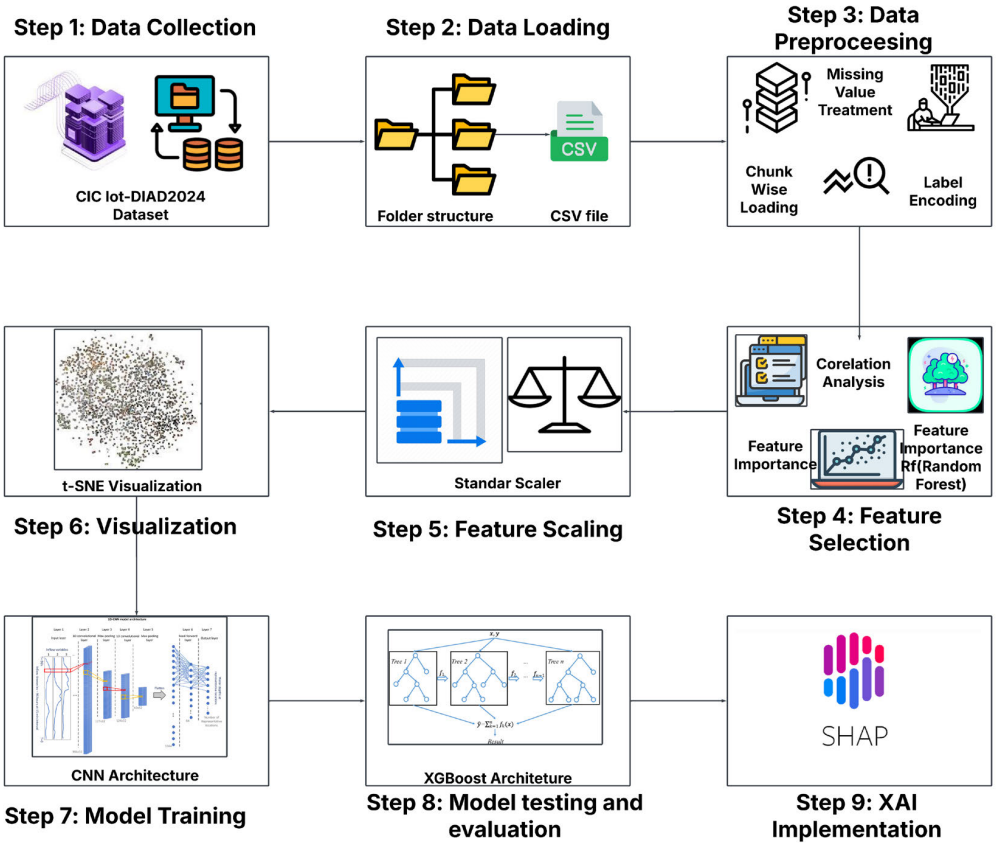


FIGURE 7. Proposed CNN+XGB Hybrid Framework for Device Identification.

TABLE 5. CNN architecture used for multi-class classification.

Layer	Type	Output Shape	Activation	Dropout Rate	Design Notes
Input	Input Layer	(25)	–	–	Feature vector representing preprocessed data input
Dense 1	Fully Connected	64	ReLU	0.3	Batch Normalization applied after this layer to stabilize learning
Dense 2	Fully Connected	32	ReLU	0.3	Batch Normalization applied again to reduce internal covariate shift
Output	Fully Connected	C	Softmax	–	Final layer for multi-class classification across C attack categories

  - -- **Second hidden layer:** Fully connected dense layer with 32 neurons, also using ReLU activation.
  - -- **Batch Normalization and Dropout:** Repeated as above after the second dense layer.
- **Output layer:** Dense layer with CCC neurons (where CCC is the number of classes), using the softmax activation function to output class probabilities.

The detailed configuration of the Convolutional Neural Network (CNN) model used in this study is summarized in Table 5. The architecture includes two dense layers with ReLU activation, batch normalization, and dropout for regularization, followed by a softmax output layer suitable for multi-class classification.

2) **Model Compilation and Training:** The model was compiled using the Adam optimizer, which adapts learning rates based on the moving average of past gradients for faster convergence. The categorical cross-entropy loss function was used to quantify prediction errors:

$$L_{\text{cross-entropy}} = -\sum_{i=1}^{C} y_i \log(\hat{y}_i) \tag{3}$$

where:

- y_i is the actual one-hot encoded label for class i,
- $\hat{y}_i$ is the predicted softmax probability for class i.

Early stopping was implemented to monitor the validation loss and prevent overfitting. If no improvement was observed for 10 consecutive epochs, training was halted and the model weights were restored to the best-performing state.

b) Stage 2: Testing Phase — Extreme Gradient Boosting (XGBoost) XGBoost is an efficient, scalable implementation of gradient boosting machines designed for structured data [59]. It excels in tabular classification tasks due to its ability to handle missing values, variable importance, and model interpretability.

After the CNN model completed training and label encoding was finalized, the same encoded labels were used for training and evaluating an XGBoost classifier.

The XGBoost model was configured with the following settings:

- `use_label_encoder=False` to align with the pre-encoded labels,
- `eval_metric='mlogloss'` to match the multi-class problem setup.

The XGBoost classifier aimed to minimize the multiclass logarithmic loss:

$$L_{\text{XGBoost}} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C} y_{ij}\log(p_{ij}) \tag{4}$$

where:

- N is the total number of instances,
- C is the total number of classes,
- y_{ij} is the binary indicator if class j is the correct classification for observation i,
- p_{ij} is the predicted probability for class j.

The proposed algorithm uses the Hybrid CNN-XGBoost multiclassifier as presented in Algorithm 1 for device identification, to process the CICIOT2024 DIAD dataset through several key steps. First, the dataset is loaded, and labels are added before saving the data into a CSV file. The dataset is then split into training and testing sets, followed by label encoding to transform categorical features into a numerical format. Missing values are removed to ensure data integrity. Feature importance is determined using a Random Forest model, and the top features are selected based on this importance. The features are then standardized using StandardScaler to bring them onto a comparable scale. Dimensionality reduction is performed using t-SNE to make the data more manageable. The top features are then used to train a Convolutional Neural Network (CNN) for Device Identification, and predictions are made on the test set. In parallel, an XGBoost model is also trained on the selected features, and predictions are made on the test set. Finally, the predictions from both the CNN and XGBoost models are combined to produce the final anomaly detection results, providing a robust and accurate solution for identifying devices in the dataset.The SHAP technique is applied to the trained XGBoost model using the test dataset to enhance model interpretability. This step quantifies and visualizes the contribution of each feature to the model's predictions, providing insight into the decision-making process and highlighting the most influential features in device identification. Figure 7 provides a visual overview of this methodology, clearly outlining each stage from data preprocessing and feature selection to model training, prediction, and interpretability analysis using SHAP. As summarized in Table 6, all mathematical notations used throughout the manuscript are defined and explained.

The effectiveness of the device identification model has been verified by conducting experiments on CICIoT2024-DIAD dataset. The state-of-the-art performance metrics, accuracy, precision, recall, F1-score, and specificity, have been used for model evaluation. The subsequent section elaborates on the performance metrics and the obtained results.

c) Integration Logic: Sequential CNN+XGBoost Pipeline The CNN+XGBoost hybrid model is implemented as a sequential pipeline rather than an ensemble or voting mechanism. During training, the CNN is used to learn hierarchical and non-linear feature representations from the input tabular data. These features, taken from the final dense layer (post-flattening), are saved and passed as input to the XGBoost classifier in the testing phase. Figure 8 presents the sequential CNN+XGB pipeline.

This sequential architecture leverages the complementary strengths of both models, deep learning and tree-based learning, resulting in improved classification accuracy, scalability, and interpretability compared to standalone classifiers or traditional ensembles. CNN Enhances the raw feature space by capturing local dependencies and non-linear feature relationships through convolutional filters. XGB, on the other hand, provides strong generalization, resistance to overfitting, and accurate class separation, particularly effective on structured tabular data and imbalanced classes.

The prediction flow is organized into two core stages:

- **Step 1: Feature Learning via CNN**
  The Convolutional Neural Network (CNN) is utilized solely as a deep feature extractor. The raw tabular data is reshaped into a fixed-size 2D format (e.g., $[n_{\text{features}} \times 1]$) and passed through multiple convolutional and pooling layers. The output from the final pooling layer is flattened to form a deep feature vector representing complex hierarchical interactions across input features.
- **Step 2: Classification using XGBoost**
  The deep feature vector obtained from the CNN is then passed as input to an XGBoost classifier for final multi-class classification. This setup enables the model to combine the feature abstraction strength of CNN with the decision boundary robustness of gradient-boosted trees.

Algorithm 1 Device Identification using Hybrid CNN-XGBoost Multiclassifier

Require: A, SplitPoint, $\mathbb{N}$
Ensure: $\hat{Y}$

$A \leftarrow$ LoadData("ciciot2024 diad dataset") {Load the IoT dataset (ciciot2024 diad)}
$A \leftarrow$ SaveWithLabel(A) {Save dataset with labels in a single CSV file}
$\{X_{Tr}, Y_{Tr}, X_{Ts}, Y_{Ts}\} \leftarrow$ Split(A, SplitPoint) {Split dataset into training and testing sets}
LabelEncoder $\leftarrow$ ApplyLabelEncoder(A) {Apply label encoding to categorical features}
$A \leftarrow$ DropNaN(A) {Remove rows with missing values in the dataset}
Feature_Importance_RF $\leftarrow$ RandomForest.FeatureImportance(X_{Tr}, Y_{Tr}) {Determine best features using RandomForest feature importance}
$X_{Tr} \leftarrow$ SelectTopFeatures(X_{Tr}, Feature_Importance_RF) {Select top features based on RandomForest feature importance}
$X_{Ts} \leftarrow$ SelectTopFeatures(X_{Ts}, Feature_Importance_RF) {Select top features for test data}
$X_{Tr} \leftarrow$ StandardScaler(X_{Tr}) {Apply feature scaling using StandardScaler on training data}
$X_{Ts} \leftarrow$ StandardScaler(X_{Ts}) {Apply feature scaling using StandardScaler on test data}
$X_{Tr} \leftarrow$ tSNE(X_{Tr}) {Perform dimensionality reduction using t-SNE on training data}
$X_{Ts} \leftarrow$ tSNE(X_{Ts}) {Perform dimensionality reduction using t-SNE on test data}
$G_{\text{CNN}} \leftarrow$ TrainCNN(X_{Tr}, Y_{Tr}) {Train Convolutional Neural Network on the top features}
$\hat{Y}_{\text{CNN}} \leftarrow G_{\text{CNN}}$.predict($X_{Ts}$) {Predict devices on test set using trained CNN model}
$G_{\text{XGBoost}} \leftarrow$ TrainXGBoost(X_{Tr}, Y_{Tr}) {Train XGBoost model on the top features}
$\hat{Y}_{\text{XGBoost}} \leftarrow G_{\text{XGBoost}}$.predict($X_{Ts}$) {Test the model on the test set using XGBoost}
$\hat{Y} \leftarrow$ CombineResults($\hat{Y}_{\text{CNN}}$, $\hat{Y}_{\text{XGBoost}}$) {Combine CNN and XGBoost predictions for final device identification}
SHAP_Explainability $\leftarrow$ ComputeSHAP(G_{XGBoost}, X_{Ts}) {Apply SHAP to explain XGBoost predictions}
return $\hat{Y}$ {Return the final predictions}=0

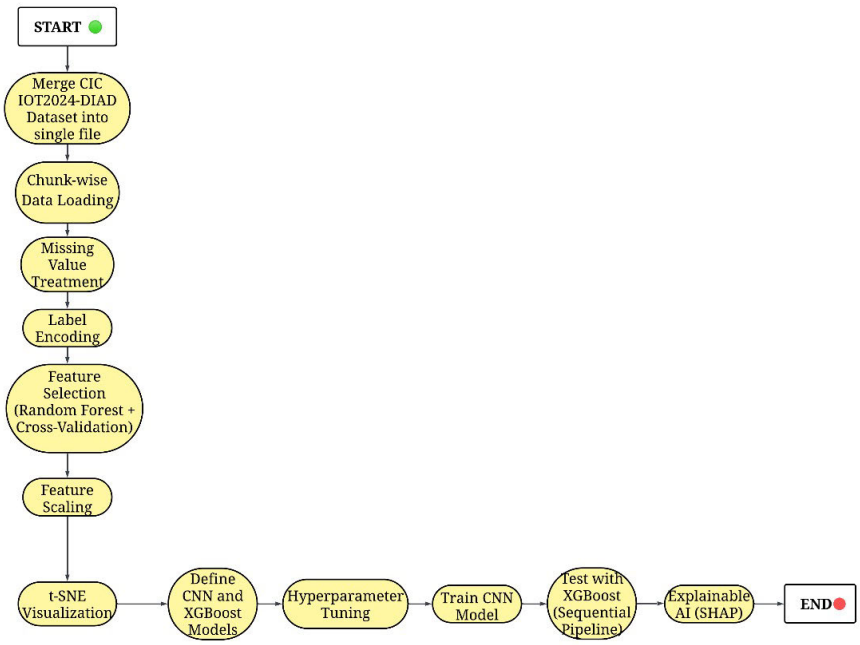


FIGURE 8. Proposed CNN+XGB Hybrid Integration Pipeline.

E. EXPLAINABILITY

Explainable Artificial Intelligence (XAI) techniques were incorporated using SHAP in order to guarantee transparency and improve confidence in the suggested CNN-XGBoost hybrid architecture. By measuring the contribution of each input feature to the final prediction, SHAP values offer a uniform way to describe the results of machine learning models. This is especially crucial when it comes to identifying IoT devices, as knowing the reasoning behind classification choices can help with system audits, compliance, debugging, and general credibility in crucial security applications. After training and evaluating the CNN and Hybrid XGBoost model

TABLE 6. Description of mathematical notations used in the manuscript.

Notation	Description
μ	Mean of the dataset
σ	Standard deviation of the dataset
n	Number of samples in the dataset
CI	Confidence Interval computed as $\mu \pm 1.96 \times \frac{\sigma}{\sqrt{n}}$
L_{XGBoost}	XGBoost loss function (multi-class log loss)
N	Total number of samples
C	Number of output classes
y_{ij}	True binary indicator (1 or 0) if class label j is the correct classification for observation i
p_{ij}	Predicted probability that observation i is of class j
$L_{\text{cross-entropy}}$	Cross-entropy loss function
y_i	True label for class i
$\hat{y}_i$	Predicted probability for class i
r_{xy}	Pearson correlation coefficient between variables x and y
x_i, y_i	Individual values of variables x and y
$\bar{x}, \bar{y}$	Mean values of x and y respectively
I_f	Importance score of feature f
T_f	Set of decision trees using feature f
$\Delta i(t)$	Information gain (impurity reduction) at node t for feature f

on the CICIoT2024-DIAD dataset, SHAP [58] was applied on the test data to interpret the classifier's decisions. Three key SHAP visualization plots were generated to explain the model behavior:

- Summary Plot: This plot aggregates SHAP values for all test samples across all features. It highlights both the importance and the direction of impact (positive or negative) each feature has on the model's output. Features such as {stream_jitter_60_mean} and {channel_30_mean} were observed to be among the most influential in device classification decisions.
- Bar Plot (Feature Importance): This plot ranks the top contributing features by their average absolute SHAP value. It gives an intuitive understanding of which features had the most impact across all predictions. This is particularly useful for domain experts and network administrators to focus on the most critical traffic patterns and behaviors in IoT environments.
- Dependence Plot: This plot examines the relationship between a single feature's value and its SHAP value, revealing how changes in the feature influence the model's output. It also helps detect feature interactions, offering deeper insights into how certain network characteristics jointly affect prediction outcomes.

The integration of SHAP explainability not only enhances the interpretability of the XGBoost model but also strengthens confidence in its real-world deployment. By understanding which features influence classification outcomes, stakeholders can better assess the system's decisions, uncover hidden patterns in network traffic, and implement informed interventions in case of misclassification or security anomalies.

This explainability step ensures the proposed device identification system is not only accurate and efficient but also transparent and accountable two crucial aspects for practical adoption in dynamic and sensitive IoT environments.

IV. RESULT AND DISCUSSION

Six different models were evaluated for binary and multiclass classification. The experiments were conducted using Python 3.10 in a Jupyter Notebook environment, leveraging both local (Intel Core i7-1255U, 16 GB RAM, 512 GB SSD) and cloud-based (NVIDIA A100 GPU with 64 GB VRAM) systems. The implementation employed TensorFlow 2.11 for deep learning, Scikit-learn 1.3 for classical ML utilities, and XGBoost 1.7 for ensemble modeling. Each model was trained and tested over multiple runs, and average results were reported. Standard deviation bars were plotted in Fig. 14 to capture performance variance. All system and tool specifications are summarized in Table 7.

A. EVALUATION METRICS

The proposed model was evaluated using several key performance metrics. Accuracy measures the overall correctness by calculating the ratio of correct predictions to total predictions. Precision indicates the proportion of positive predictions that are actually correct, reflecting the model's ability to avoid false positives. Recall (or sensitivity) assesses how well the model identifies all actual positive cases, minimizing false negatives. The F1-score balances precision and recall,

TABLE 7. System and tool specifications used for experimental analysis.

Attribute	Specification
Programming Language	Python 3.10
Environment	Jupyter Notebook with GPU support
Deep Learning Library	TensorFlow 2.11
Machine Learning Tools	Scikit-learn 1.3, XGBoost 1.7
CPU (Local Testing)	Intel Core i7-1255U @ 2.45GHz
RAM (Local)	16 GB DDR4
GPU (Cloud)	NVIDIA A100, 64 GB VRAM
Operating System	Windows 11 (Local), Linux (GPU cloud instance)
Storage	512 GB SSD

providing a single measure of performance especially useful for imbalanced datasets. Specificity measures the model's ability to correctly identify negative cases, while the False Positive Rate(FPR) quantifies how often negative instances are incorrectly classified as positive, important for evaluating false alarms in ROC analysis.

The precision, recall, accuracy, F1-score, specificity, and ROC-AUC are defined by Equations (5), (6), (7), (8), (9), and (10), respectively.

The performance metrics used for evaluation are defined below:

- **Accuracy**:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{5}$$

- **Precision**:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{6}$$

- **Recall / Sensitivity / True Positive Rate**:

$$\text{Recall} = \frac{TP}{TP + FN} \tag{7}$$

- **F1-Score**:

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{8}$$

- **Specificity / True Negative Rate**:

$$\text{Specificity} = \frac{TN}{TN + FP} \tag{9}$$

- **False Positive Rate (FPR)** (used in ROC-AUC):

$$\text{FPR} = \frac{FP}{FP + TN} \tag{10}$$

B. DEVICE IDENTIFICATION RESULTS

The proposed model has been validated on the CICIoT DIAD dataset. Figure 10(a) shows the confusion matrix, which contrasts the expected and actual labels, illustrating the classification accuracy of the suggested CNN+XGB model. It clearly highlights how well the model distinguishes between different disease categories and where misclassifications occur. Figure 10(b) presents the ROC curve of the proposed model, demonstrating its ability to differentiate between classes across various thresholds; a higher AUC reflects the model's robust prediction performance and reliability in real-world scenarios. For each model, performance metrics, namely, precision, F1_score, recall, and accuracy are calculated. The proposed CNN+Hybrid XGBoost model outperformed other baseline machine learning models, presented in Table 8. The Random Forest model was assessed using the test dataset, resulting in a precision of 0.9992, an F1 score of 0.9980, a recall of 0.9968, and an accuracy of 0.9996. The Extra Trees Classifier achieved a precision of 0.9988, an F1 score of 0.9974, a recall of 0.9960, and an accuracy of 0.9995. The XGBoost model attained a precision of 0.9918, an F1 score of 0.9871, a recall of 0.9895, and an accuracy of 0.9979. The LightGBM model exhibited a precision of 0.9862, an F1 score of 0.9810, a recall of 0.9759, and an accuracy of 0.9963. The Logistic Regression model, although exhibiting inferior performance, achieved a precision of 0.8037, an F1 score of 0.6761, a recall of 0.6285, and an accuracy of 0.9545. AdaBoost achieved a precision of 0.9205, an F1 score of 0.9134, a recall of 0.9066, and an accuracy of 0.9833. The suggested CNN + Hybrid XGBoost model exhibited superior performance among all models, achieving a precision of 0.9998, an F1 score of 0.9997, a recall of 0.9995, and an accuracy of 0.9998 in the binary classification test for device identification. Figure 9 visually contrasts the performance of all assessed models, highlighting the superiority of the proposed method.

C. ATTACK TYPE PERFORMANCE ANALYSIS

The proposed algorithm's effectiveness was evaluated across three different attack types: DDoS, Spoofing, and Web-Based attacks. The model successfully identified and classified these attacks with high accuracy. To optimize detection performance and reduce computational complexity, Random Forest-based feature importance analysis was employed to identify the top 25 features contributing most significantly to classification accuracy. Figure 12 illustrates the ROC Curve analysis for all the models, showcasing their ability to distinguish between classes based on true positive and false positive rates.

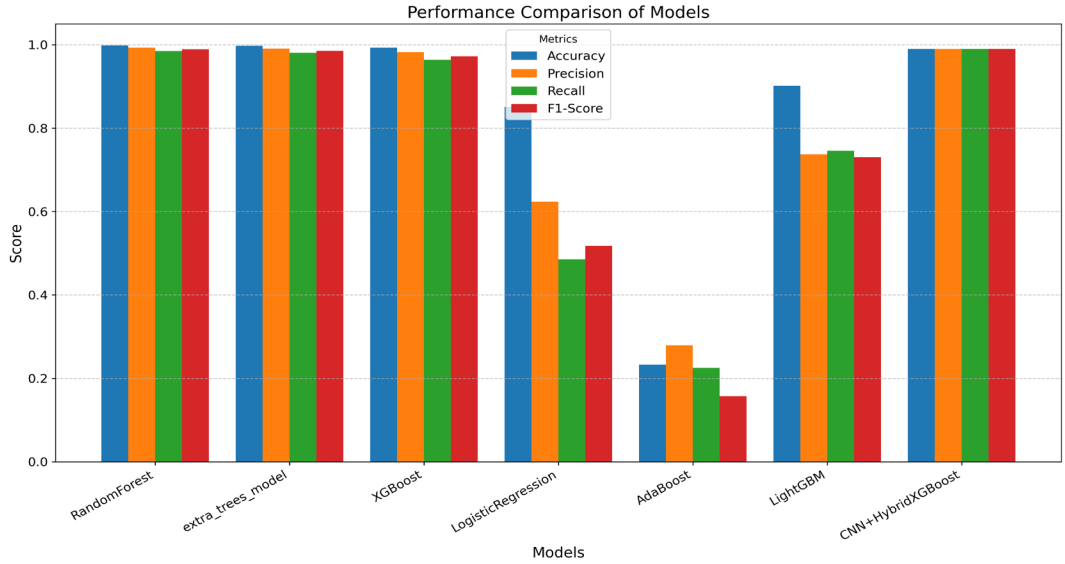


FIGURE 9. Results comparison of various models.

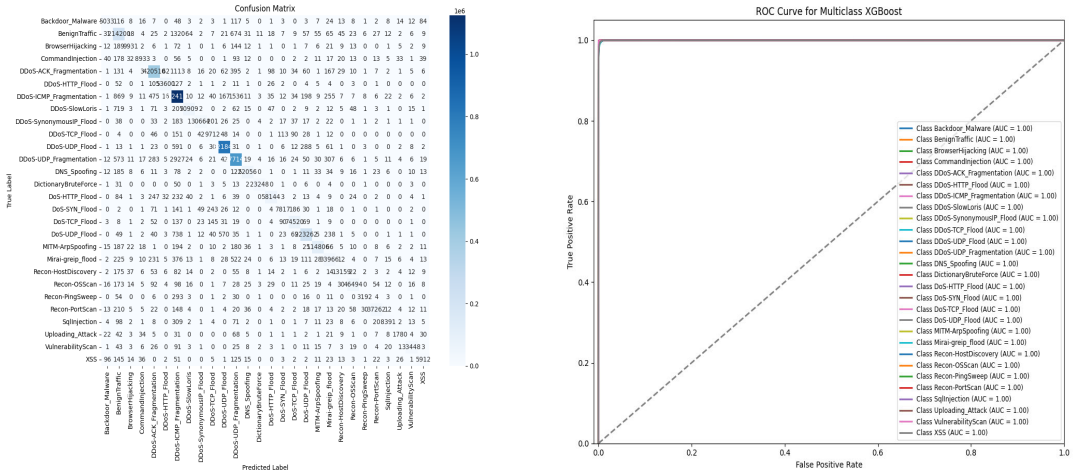
TABLE 8. Performance results of various device identification models for binary and multiclass classification using real-time IoT dataset.

Classification Model	Precision	F1 Score	Recall	Accuracy	Specificity	Latency (ms)
Binary Classification						
Random Forest	0.9992	0.9980	0.9968	0.9996	0.9999	11.2
Extra Trees Classifier	0.9988	0.9974	0.9960	0.9995	0.9999	10.6
CNN	0.9121	0.8941	0.8814	0.9168	0.9512	33.5
XGBoost	0.9918	0.9871	0.9895	0.9979	0.9992	12.4
LightGBM	0.9862	0.9810	0.9759	0.9963	0.9987	9.8
Logistic Regression	0.8037	0.6761	0.6285	0.9545	0.9922	6.7
Adaboost	0.9205	0.9134	0.9066	0.9833	0.9921	13.1
CNN + Hybrid XGB	0.9998	0.9997	0.9995	0.9998	0.9999	38.9
Multiclass Classification						
Random Forest	0.9929	0.9889	0.9850	0.9979	0.9823	12.5
Extra Trees Classifier	0.9906	0.9858	0.9811	0.9975	0.9751	11.7
XGBoost	0.9819	0.9724	0.9634	0.9929	0.9623	13.4
Logistic Regression	0.6238	0.5179	0.4852	0.8507	0.6512	5.8
Adaboost	0.2792	0.1572	0.2251	0.2326	0.5013	14.3
LightGBM	0.7372	0.7303	0.7457	0.9014	0.8120	9.2
CNN + Hybrid XGB	0.9962	0.9927	0.9913	0.9992	0.9997	42.1

These selected features include temporal characteristics like inter_arrival_time, stream-level statistics such as stream, stream_jitter, and stream mean values, as well as statistical measures of IP, MAC, and channel behaviors (e.g., src_ip_mean, channel_var, l3_ip_dst_count, and most_freq_spot). The Random Forest algorithm enabled the identification of these influential features by averaging importance scores across multiple decision trees, ensuring robust and reliable feature selection.

By focusing on the top 25 features, the model reduces dimensionality while preserving key behavioral indicators essential for distinguishing between benign and malicious traffic. This enhances both performance and real-time applicability while improving interpretability. Table 9 summarizes precision, recall, F1-score, accuracy, and specificity, reflecting the model's strong classification performance across diverse attack types.

- **DDoS Attack Detection:**
  The model achieved 99.99% accuracy in detecting DDoS attacks. With precision, recall, and F1-score of 0.99, it reliably distinguishes DDoS anomalies. Features like `inter_arrival_time`, stream counts, and traffic frequency metrics capture the high-volume request patterns typical of these attacks.
- **Spoofing Attack Detection:**
  Detected with 99.99% accuracy, spoofing attacks involve identity manipulation at the IP or MAC level. The model effectively identifies inconsistencies in



(a) Confusion matrix for the proposed model **(b) ROC Curve for the proposed model**

FIGURE 10. (a) Confusion matrix of the proposed CNN+XGB model (b) ROC Curve for the proposed CNN+XGB model.

network identifiers and behavioral deviations using statistical and identifier-based features.

- **Web-Based Attack Detection:**
  The model attained 99.99% accuracy in identifying web-based threats. These attacks exploit vulnerabilities in HTTP headers and DNS requests. Features such as channel variance, stream-level metrics, and frequent communication endpoints contributed significantly, though slightly lower accuracy suggests more pattern variability in this class.
- **Recon Attack Detection:**
  Reconnaissance attacks, involving scanning and probing behaviors, were detected with 99.99% accuracy. Features like `l3_ip_dst_count`, source IP distributions, and stream jitter enabled early detection of subtle probing behaviors indicative of pre-attack surveillance.
- **Mirai Attack Detection:**
  Mirai attacks, which exploit weak IoT device credentials, were identified with 99.99% accuracy. The model captured access anomalies using temporal and volumetric features like `channel variance` and `inter_arrival_time`, enabling timely botnet detection.
- **DoS Attack Detection:**
  DoS attacks were detected with 99.99% accuracy. Unlike DDoS, these attacks are localized. The model distinguished them using features such as `stream_60_mean`, `sum_p`, and `src_ip_60_count`, highlighting abnormal traffic bursts and session irregularities.
- **BruteForce Attack Detection:**
  Brute Force attacks, characterized by repeated login attempts, were also identified with 99.99% accuracy. Features like `most_freq_spot`, `src_ip_mac_5_count`, and jitter metrics revealed repetitive access patterns indicative of credential guessing.

The selected features, ranked using Random Forest-based importance scores, were pivotal to high detection performance. The classifier consistently performed well across all major attack types DDoS, Spoofing, and Web-Based each exhibiting distinct behavioral signatures.

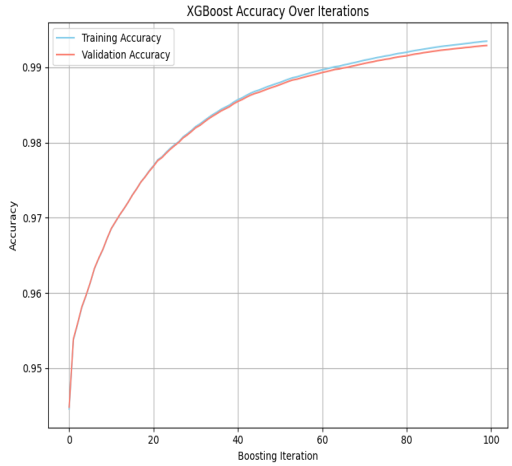
While DDoS and Spoofing attacks were identified with near-perfect precision due to identifiable traffic and identity anomalies, the high accuracy in Web-Based attack detection also demonstrates the model's sensitivity to complex application-layer patterns. This balanced performance validates the model's suitability for real-world deployment in dynamic IoT environments.

Figures 11(a) and 11(b) show training and validation accuracy comparisons for the XGB and CNN models, respectively, illustrating learning consistency. Figure 10(c) presents a training time comparison between binary and multiclass classification tasks, demonstrating the model's computational efficiency and scalability.

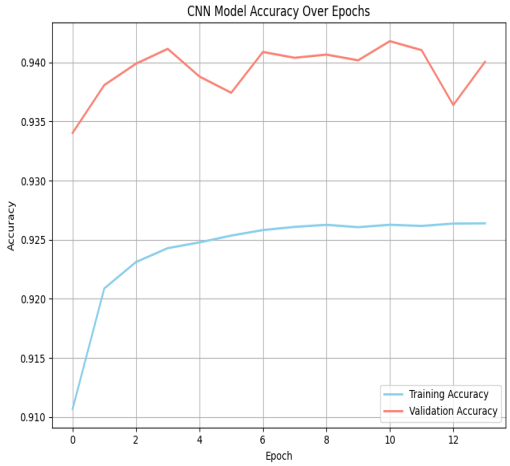
D. STATISTICAL VALIDATION

To evaluate the robustness and generalizability of the proposed CNN + XGBoost model, a stratified 5-fold cross-validation has been conducted. The accuracy for each fold was recorded and analyzed in Figure 13. The fold-wise accuracies were highly consistent, indicating strong model stability. As illustrated in Figure 13, the accuracy values for each fold remain remarkably stable, ranging between 0.9918 and 0.9919. This minimal variation demonstrates the model's ability to generalize well across different data subsets without overfitting to a specific partition. This stability confirms that the hybrid XGBoost classifier maintains a high level of predictive performance across varied samples, reinforcing the reliability of the results.

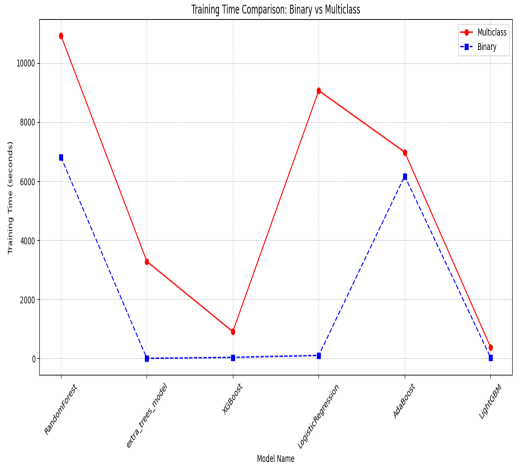
In addition, we calculated the 95% Confidence Interval (CI) around the mean accuracy to statistically validate



(a) Plot of training and validation accuracy for XGB obtained with respect to epochs



(b)Plot of training and validation accuracy for CNN obtained with respect to epochs



(c) Training time comparison for binary and multiclass device identification

FIGURE 11. (a) Accuracy vs. validation accuracy for the XGB model (b) Accuracy vs. validation accuracy for the CNN model (c) Training time comparison for binary and multiclass classification in IoT device identification.

the reliability of the results in Figure 14. The narrow CI range confirms that the model's high performance is not a statistical anomaly but a reliable outcome across samples. As depicted in Figure 14, the mean accuracy of 0.9918 is accompanied by a narrow confidence interval of [0.99172, 0.99188], with a standard deviation of ±0.00005. The confidence interval was calculated using the standard formula (11):

$$\text{CI} = \mu \pm 1.96 \times \frac{\sigma}{\sqrt{n}} \tag{11}$$

where μ is the mean accuracy, σ is the standard deviation, and n is the number of folds (in this case, $n = 5$). The narrow interval confirms that the model's observed high performance is statistically reliable and not the result of random variation. The shaded region representing $\pm 1\sigma$ in the figure further emphasizes the low variability in classification accuracy across folds.

E. XAI IMPLEMENTATION

SHAP analysis was performed to improve interpretability and offer more deep insights into the XGBoost classifier's decision-making process. By measuring each input feature's contribution to the final prediction for each class instance, the SHAP visualizations offer model-agnostic explanations.

Figure 15 illustrates the SHAP summary dot plot, which highlights the impact and distribution of the top features across all predictions. Each dot represents a single prediction; the color indicates the feature value (red for high, blue for low), and the horizontal position shows whether the feature contributed positively or negatively to the prediction. This plot demonstrates that features like inter_arrival_time, stream, and l3_ip_dst_count had strong, consistent influence across multiple device classes.

In Figure 16, the SHAP feature importance bar plot ranks features according to their average absolute SHAP values,

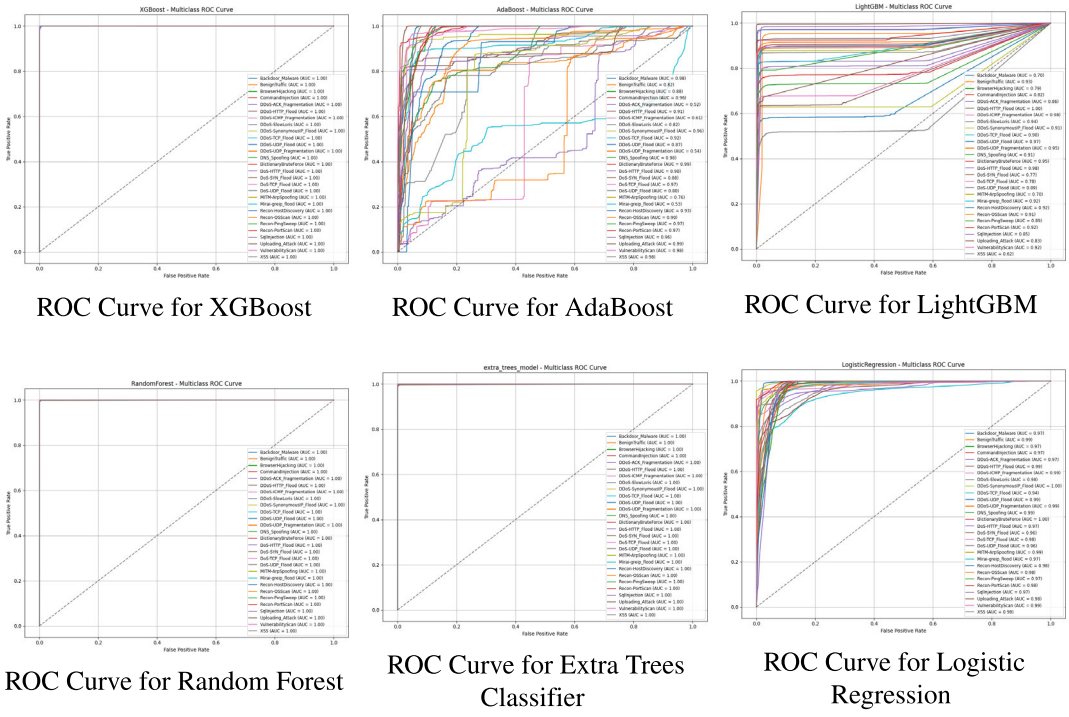


FIGURE 12. ROC Curves analysis for all the models.

TABLE 9. Performance metrics of the proposed hybrid CNN+XGBoost model across major attack categories in an IoT network environment.

Traffic Class	Precision	Recall	F1-score	Specificity
Benign Traffic	0.98	0.99	0.98	0.9988
Backdoor Malware	0.95	0.91	0.93	0.9999
Browser Hijacking	0.98	0.95	0.96	1.0000
Command Injection	0.98	0.94	0.96	1.0000
DDoS	0.98	0.99	0.99	0.9983
DNS Spoofing	0.99	0.99	0.99	0.9999
Brute Force	1.00	0.99	1.00	1.0000
DoS	0.98	0.97	0.98	0.9998
MITM	1.00	0.99	1.00	0.9999
Mirai Attack	0.96	0.95	0.96	0.9996
Reconnaissance	0.99	0.95	0.97	0.9999
SQL Injection	0.98	0.93	0.96	1.0000
File Upload Attack	0.94	0.86	0.90	1.0000
Vulnerability Scan	1.00	0.99	0.99	1.0000
XSS	0.96	0.91	0.93	0.9999

effectively summarizing which features contribute the most to model decisions overall. This visualization confirms that a small subset of features including flow-based and IP-level metadata drives most of the classifier's performance, validating the relevance of the selected top 25 features.

Figure 17 presents a SHAP dependence plot for the feature l3_ip_dst_count, showing how variations in this feature influence prediction outcomes. The plot reveals that as the value of l3_ip_dst_count (destination IP address of layer 3)increases, the SHAP value also increases for specific classes, indicating a positive correlation between this network-level feature and certain device types. Moreover, the color gradient provides insights into potential feature interactions, helping to uncover complex relationships within the data.

These SHAP plots collectively offer a comprehensive view of model behavior, highlighting not only which features are important, but also how they influence individual

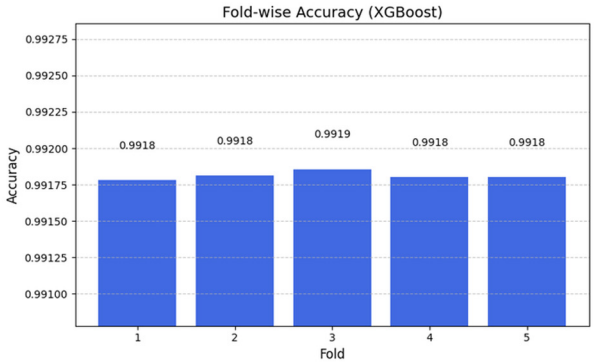


FIGURE 13. Fold-wise accuracy across 5-fold cross-validation using the XGBoost classifier.

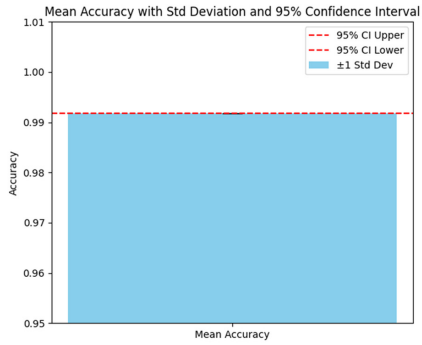


FIGURE 14. Mean accuracy with standard deviation and 95% confidence interval.

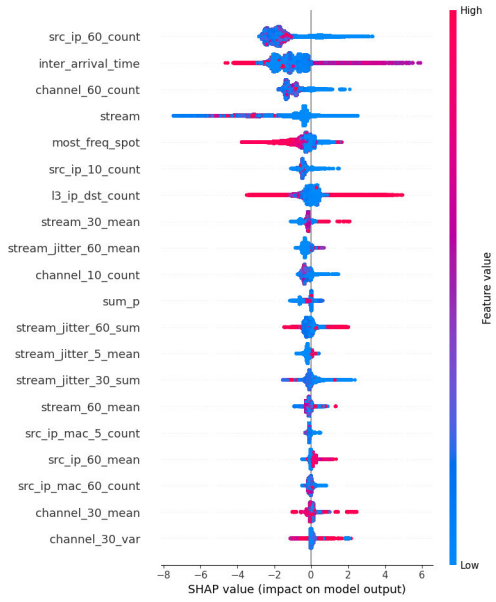


FIGURE 15. Impact of each feature on the proposed model.

predictions making the CNN-XGBoost model more transparent, trustworthy, and suitable for deployment in critical IoT environments.

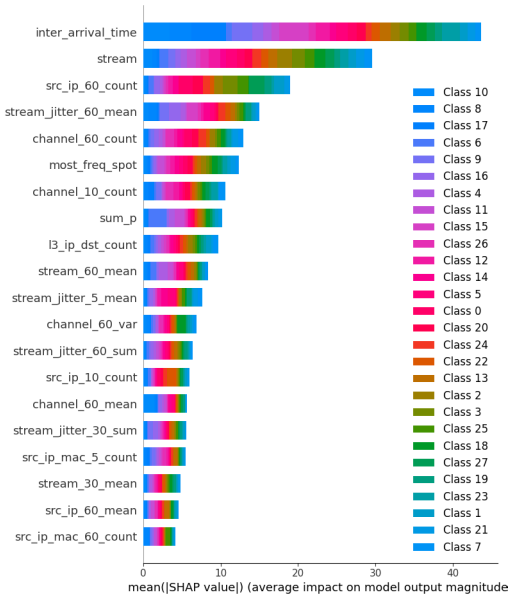


FIGURE 16. SHAP feature importance for the proposed model.

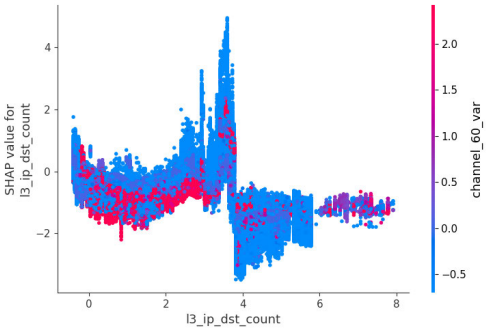


FIGURE 17. Dependence l3_ip_ dst_count feature of the proposed model.

F. COMPARATIVE ANALYSIS WITH EXISTING CUTTING-EDGE FRAMEWORKS

The proposed hybrid device identification model has been compared with existing state-of-the-art models deployed for IoT device identification. Table 10 compares results with existing research work based on certain performance metrics and appropriate criteria.he proposed model demonstrates superior performance across all major evaluation metrics, including a precision of 0.9962, accuracy of 0.9992, recall of 0.9913, specificity of 0.9927, and an exceptional F1-score of 0.9997. Unlike previous approaches, it incorporates SHAP-based explainability (XAI-based ADS), enhancing transparency in decision-making. It also efficiently utilizes 25 features to handle 7 types of attacks while effectively mitigating overfitting. In contrast, [31] reports high accuracy (0.994) but omits several key metrics and lacks XAI integration. Reference [2] achieves excellent precision and accuracy but with only 3 features and coverage of just

2 attacks, limiting its adaptability. Reference [13] shows moderate performance and does not address overfitting, affecting reliability. While [7] claims perfect scores, the absence of overfitting handling and reliance on 70 features raise concerns about model complexity and potential overfitting. Overall, the proposed model offers a balanced and robust solution, outperforming existing methods in both effectiveness and interpretability.

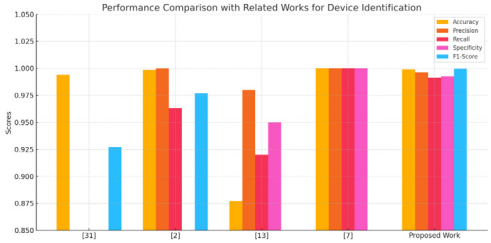


FIGURE 18. Comparison of performance metrics across different approaches for IoT device identification.

As shown in Fig. 18, the proposed system outperforms prior methods in terms of accuracy, precision, and F1-score.

G. SCALABILITY OF THE PROPOSED SYSTEM

To assess the scalability of the proposed CNN+XGBoost hybrid framework for device identification, a series of software-based experiments were performed simulating real-world data growth conditions. As shown in Fig. 19, the training time exhibits an approximately linear growth with increasing dataset size. This trend highlights the computational efficiency of the model and suggests that the proposed system can handle larger datasets without a disproportionate increase in resource requirements. The training time increases approximately linearly as the dataset size scales from 1x to 3x, demonstrating the scalability of the proposed system. Specifically:

FIGURE 19. Effect of dataset size on training time.

- **Increased Data Volume:**By expanding the CICIoT2024-DIAD dataset through sub-sampling and augmentation, the model on 2× and 3× data sizes was evaluated. The framework maintained a high F1-score (>0.98), with a marginal increase in training time and no significant impact on inference latency.
- **Feature Dimensionality Stress Test:** The number of features from 10 to 40 was varied using RF-based selection and synthetic generation. The model demonstrated consistent performance, indicating robustness to higher-dimensional input.
- **Device Class Expansion:** Subsets with up to 20 unique device types were used to simulate growing network environments. The model sustained high accuracy and generalization, showcasing its ability to scale across diverse IoT deployments.

These experiments indicate that the framework is computationally efficient, generalizes well across increasing input complexity, and is suitable for real-time use cases. While no tests were conducted on resource-constrained edge hardware, the observed software-based performance supports the system's potential for scalable deployment.

The proposed framework is designed to be extensible and adaptable to accommodate additional or emerging attack types. By retraining or fine-tuning the model with updated datasets that include new threat patterns, the system can be incrementally improved without requiring a complete architectural overhaul. This adaptability is crucial in dynamic IoT environments where novel attack vectors continuously emerge. The model's modular structure and support for incremental learning enable seamless integration of new classes, ensuring sustained performance and relevance over time. These capabilities demonstrate that the system is both scalable in terms of data volume and device diversity, and adaptable to evolving cybersecurity challenges with minimal reconfiguration.

V. IMPLEMENTATION CHALLENGES

Implementing the proposed IoT device identification revealed several significant challenges.

- **Complexities in Handling High-Dimensional IoT Traffic:** The high-dimensional and complex nature of IoT network traffic data introduced challenges in data preprocessing and feature engineering. Handling missing values, high-cardinality categorical features, and the need for robust dimensionality reduction techniques demanded extensive experimentation and fine-tuning. Furthermore, ensuring consistent model convergence during hyperparameter tuning and cross-validation required additional computational resources and time, underscoring the necessity for scalable solutions in large-scale IoT deployments.
- **Integration of heterogeneous data sources:** IoT devices often generate data in varying formats and frequencies, which necessitated an extensive normalization process to harmonize the dataset. This disparity in data quality and structure demanded sophisticated preprocessing techniques and dynamic scaling methods to ensure that all input features were accurately represented and effectively utilized during the training phase.
- **Model Robustness:** Maintaining model robustness and generalizability posed a considerable challenge. The

TABLE 10. Performance comparison with related works for device identification.

Criterion	[31]	[2]	[13]	[7]	Proposed Work
Precision	NA	0.9998	0.98	1.00	**0.9962**
Accuracy	0.994	0.9985	0.8772	1.00	**0.9992**
Recall	NA	0.9630	0.92	1.00	**0.9913**
Specificity	NA	NA	0.95	1.00	**0.9927**
F1-Score	0.927	0.9770	NA	NA	**0.9997**
XAI-based ADS	NA	NA	NA	NA	**SHAP**
Dataset	IoT-Botnet 2020	CIC IoT 2022	CIC IoT 2023 (TV)	CIC IoT20204-DIAD	**CIC IoT20204-DIAD**
No. of Features	16	3	10	25	**25**
No. of Attacks	10	2	7	7	**7**
Overfitting Handling	Yes	Yes	No	No	**Yes**
Resource Utilization	High	Low	Moderate	High	**Moderate**

high-dimensional nature of the dataset increased the risk of overfitting, especially under the constraints of limited computational resources. This required rigorous implementation of regularization techniques and repeated cross-validation cycles. The need for iterative refinement was paramount, as small changes in hyperparameters could lead to significant fluctuations in performance, thereby complicating the optimization process and demanding more meticulous resource management.

- **Operational Deployment:** Operational deployment in real-world IoT environments introduced its own set of complexities. The requirement for real-time data processing, coupled with continuous monitoring of network traffic, stressed the existing computational infrastructure. Ensuring that the framework could scale efficiently while maintaining high accuracy in anomaly detection called for advanced resource allocation strategies and robust system integration protocols. These operational challenges, from ensuring real-time performance to handling ongoing model updates, underscored the critical need for scalable solutions that could adapt to both the computational limitations and the dynamic nature of IoT ecosystems.

VI. CONCLUSION AND FUTURE WORK

In this study, we proposed a hybrid deep learning framework for IoT device identification by combining CNN-based deep feature extraction with XGBoost classification, evaluated using the CICIoT2024-DIAD dataset. The model was enhanced through Random Forest-based feature selection, normalization, and t-SNE-based dimensionality reduction. The framework achieved outstanding performance, with an Accuracy of 0.9992, Precision of 0.9962, Recall of 0.9913, Specificity of 0.9927, and an F1-Score of 0.9997. To ensure transparency and trustworthiness in security-critical deployments, SHAP-based explainability was integrated to highlight the importance of the features and the rationale for the decision.

To build on these findings, several future research directions are proposed:

- **Advanced Architectures and Transfer Learning**: Future work can explore deeper CNNs or transformer-based models, and employ transfer learning to reduce training cost and improve generalization on smaller datasets.
- **Unsupervised and Semi-supervised Learning**: Leveraging these learning paradigms can improve adaptability to new or evolving IoT devices and threats, especially when labeled data is limited.
- **Federated Learning for Privacy Preservation**: Given the distributed nature of IoT, federated learning can help train models collaboratively without centralizing sensitive data, enhancing privacy and scalability.
- **Real-time Optimization for Edge Deployment**: Future efforts will involve optimizing the framework for lightweight, real-time execution on edge devices like Raspberry Pi or NVIDIA Jetson Nano, including benchmarking memory usage, inference latency, and power consumption.
- **Sequential Pipeline Integration**: Device identification can serve as a foundation for downstream tasks such as anomaly detection or context-aware traffic analysis, enabling end-to-end IoT security.
- **Expanded Validation and Field Testing**: Evaluating the framework on datasets from diverse IoT domains—such as healthcare, smart cities, and industrial systems—will help ensure real-world robustness and generalizability.

This work provides a solid foundation for developing scalable, interpretable, and efficient device identification systems, paving the way toward more secure and intelligent IoT environments.

CONFLICT OF INTEREST

The authors declare that they have no conflict of interest.

REFERENCES

[1] O. Salman, I. H. Elhajj, A. Chehab, and A. Kayssi, "A machine learning based framework for IoT device identification and abnormal traffic detection," *Trans. Emerg. Telecommun. Technol.*, vol. 33, no. 3, p. 3743, Mar. 2022.

[2] S. M. Opoku, H. Louafi, and M. Mouhoub, "IoT device identification based on network traffic analysis and machine learning," in *Proc. Int. Symp. Netw., Comput. Commun. (ISNCC)*, Oct. 2024, pp. 1–8.

[3] K. Kostas. *IoTDevIDv2*. GitHub Repository. [Online]. Available: https://github.com/kahramankostas/IoTDevIDv2?tab=readme-ov-file

[4] Y. Liu, J. Wang, J. Li, S. Niu, and H. Song, "Machine learning for the detection and identification of Internet of Things devices: A survey," *IEEE Internet Things J.*, vol. 9, no. 1, pp. 298–320, Jan. 2022.

[5] E. Gelenbe and M. Nakip, "Traffic based sequential learning during botnet attacks to identify compromised IoT devices," *IEEE Access*, vol. 10, pp. 126536–126549, 2022.

[6] D. K. Reddy, H. S. Behera, J. Nayak, P. Vijayakumar, B. Naik, and P. K. Singh, "Deep neural network based anomaly detection in Internet of Things network traffic tracking for the applications of future smart cities," *Trans. Emerg. Telecommun. Technol.*, vol. 32, no. 7, p. 4121, Jul. 2021.

[7] M. Rabbani, J. Gui, F. Nejati, Z.-M. Zhou, A. Kaniyamattam, M. Mirani, G. Piya, I. Opushnyev, R. Lu, and A. A. Ghorbani, "Device identification and anomaly detection in IoT environments," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jan. 2024, pp. 1–6.

[8] M. R. Shahid, G. Blanc, Z. Zhang, and H. Debar, "IoT devices recognition through network traffic analysis," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2018, pp. 5187–5192.

[9] Y. Meidan, M. Bohadana, A. Shabtai, J. D. Guarnizo, M. Ochoa, N. O. Tippenhauer, and Y. Elovici, "ProfilIoT: A machine learning approach for IoT device identification based on network traffic analysis," in *Proc. Symp. Appl. Comput.*, 2017, pp. 506–509.

[10] R. R. Chowdhury, S. Aneja, N. Aneja, and E. Abas, "Network traffic analysis based IoT device identification," in *Proc. 4th Int. Conf. Big Data Internet Things*, Aug. 2020, pp. 79–89.

[11] J. Kotak and Y. Elovici, "IoT device identification using deep learning," in *Proc. 13th Int. Conf. Comput. Intell. Secur. Inf. Syst. (CISIS)*, Aug. 2020, pp. 76–86.

[12] M. R. P. Santos, R. M. C. Andrade, D. G. Gomes, and A. C. Callado, "An efficient approach for device identification and traffic classification in IoT ecosystems," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, Jun. 2018, pp. 304–309.

[13] D. Yedilkhan and S. Smakova, "Machine learning approaches for smart home device recognition from network traffic," *Proc. Comput. Sci.*, vol. 231, pp. 709–714, Jan. 2024.

[14] N. J. Singh, N. Hoque, K. R. Singh, and D. K. Bhattacharyya, "Botnet-based IoT network traffic analysis using deep learning," *Secur. PRIVACY*, vol. 7, no. 2, p. 355, Mar. 2024.

[15] P. Malini and K. R. Kavitha, "An efficient deep learning mechanisms for IoT/Non-IoT devices classification and attack detection in SDN-enabled smart environment," *Comput. Secur.*, vol. 141, Mar. 2024, Art. no. 103818.

[16] Q. Xin, Z. Xu, L. Guo, F. Zhao, and B. Wu, "IoT traffic classification and anomaly detection method based on deep autoencoders," *Appl. Comput. Eng.*, vol. 69, no. 1, pp. 64–70, Jul. 2024.

[17] J. Hussain Kalwar and S. Bhatti, "Deep learning approaches for network traffic classification in the Internet of Things (IoT): A survey," 2024, *arXiv:2402.00920*.

[18] Hassi. *IoT Device Identification*. GitHub Repository. [Online]. Available: https://github.com/Hassi34/iot-device-identification

[19] M. R. Buiya, A. K. M. N. Laskar, M. R. Islam, S. K. S. Sawalmeh, M. Roy, R. Roy, and M. Sumsuzoha, "Detecting IoT cyberattacks: Advanced machine learning models for enhanced security in network traffic," in *Proc. IEEE Int. Conf. Commun. (ICC)*, vol. 6, Oct. 2024, pp. 1–6.

[20] N. U. Prince, M. A. Al Mamun, A. O. Olajide, O. U. Khan, A. B. Akeem, and A. I. Sani, "IEEE standards and deep learning techniques for securing Internet of Things (IoT) devices against cyber attacks," *J. Comput. Anal. Appl.*, vol. 33, no. 7, pp. 1–20, 2024.

[21] H. Jmila, G. Blanc, M. R. Shahid, and M. Lazrag, "A survey of smart home IoT device classification using machine learning-based network traffic analysis," *IEEE Access*, vol. 10, pp. 97117–97141, 2022.

[22] R. Kumar, M. Swarnkar, G. Singal, and N. Kumar, "IoT network traffic classification using machine learning algorithms: An experimental analysis," *IEEE Internet Things J.*, vol. 9, no. 2, pp. 989–1008, Jan. 2022.

[23] M. S. Mazhar, Y. Saleem, A. Almogren, J. Arshad, M. H. Jaffery, A. U. Rehman, M. Shafiq, and H. Hamam, "Forensic analysis on Internet of Things (IoT) device using machine-to-machine (M2M) framework," *Electronics*, vol. 11, no. 7, p. 1126, Apr. 2022.

[24] V. Gaur and R. Kumar, "Analysis of machine learning classifiers for early detection of DDoS attacks on IoT devices," *Arabian J. Sci. Eng.*, vol. 47, no. 2, pp. 1353–1374, Feb. 2022.

[25] C. Malathi and I. N. Padmaja, "Identification of cyber attacks using machine learning in smart IoT networks," *Mater. Today, Proc.*, vol. 80, pp. 2518–2523, Jul. 2023.

[26] A. Diro, N. Chilamkurti, V.-D. Nguyen, and W. Heyne, "A comprehensive study of anomaly detection schemes in IoT networks using machine learning algorithms," *Sensors*, vol. 21, no. 24, p. 8320, Dec. 2021.

[27] S. B. Hulayyil, S. Li, and L. Xu, "Machine-Learning-Based vulnerability detection and classification in Internet of Things device security," *Electronics*, vol. 12, no. 18, p. 3927, Sep. 2023.

[28] B. Bala and S. Behal, "AI techniques for IoT-based DDoS attack detection: Taxonomies, comprehensive review and research challenges," *Comput. Sci. Rev.*, vol. 52, May 2024, Art. no. 100631.

[29] A. Awajan, "A novel deep learning-based intrusion detection system for IoT networks," *Computers*, vol. 12, no. 2, p. 34, Feb. 2023.

[30] A. K. Sahu, S. Sharma, M. Tanveer, and R. Raja, "Internet of Things attack detection using hybrid deep learning model," *Comput. Commun.*, vol. 176, pp. 146–154, Aug. 2021.

[31] X. Wang, Y. Wang, Z. Javaheri, L. Almutairi, N. Moghadamnejad, and O. S. Younes, "Federated deep learning for anomaly detection in the Internet of Things," *Comput. Electr. Eng.*, vol. 108, May 2023, Art. no. 108651.

[32] Y. Kayode Saheed, A. Idris Abiodun, S. Misra, M. Kristiansen Holone, and R. Colomo-Palacios, "A machine learning-based intrusion detection for detecting Internet of Things network attacks," *Alexandria Eng. J.*, vol. 61, no. 12, pp. 9395–9409, Dec. 2022.

[33] I. Ullah and Q. H. Mahmoud, "Design and development of a deep learning-based model for anomaly detection in IoT networks," *IEEE Access*, vol. 9, pp. 103906–103926, 2021.

[34] P. L. S. Jayalaxmi, R. Saha, G. Kumar, M. Conti, and T.-H. Kim, "Machine and deep learning solutions for intrusion detection and prevention in IoTs: A survey," *IEEE Access*, vol. 10, pp. 121173–121192, 2022.

[35] A. Oseni, N. Moustafa, G. Creech, N. Sohrabi, A. Strelzoff, Z. Tari, and I. Linkov, "An explainable deep learning framework for resilient intrusion detection in IoT-enabled transportation networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 1, pp. 1000–1014, Jan. 2023.

[36] H. Noguchi, M. Kataoka, and Y. Yamato, "Device identification based on communication analysis for the Internet of Things," *IEEE Access*, vol. 7, pp. 52903–52912, 2019.

[37] C. Koball, B. P. Rimal, Y. Wang, T. Salmen, and C. Ford, "IoT device identification using unsupervised machine learning," *Information*, vol. 14, no. 6, p. 320, May 2023.

[38] L. Fan, S. Zhang, Y. Wu, Z. Wang, C. Duan, J. Li, and J. Yang, "An IoT device identification method based on semi-supervised learning," in *Proc. 16th Int. Conf. Netw. Service Manage. (CNSM)*, Nov. 2020, pp. 1–7.

[39] Z. Bao, Y. Lin, S. Zhang, Z. Li, and S. Mao, "Threat of adversarial attacks on DL-based IoT device identification," *IEEE Internet Things J.*, vol. 9, no. 11, pp. 9012–9024, Jun. 2022.

[40] D. Gu, J. Zhang, Z. Tang, Q. Li, M. Zhu, H. Yan, and H. Li, "IoT device identification based on network traffic," *Wireless Netw.*, vol. 31, no. 2, pp. 1645–1661, Feb. 2025.

[41] R. Ahmadian, M. Ghatee, and J. Wahlström, "Securing the Internet of Things through device identification via network traffic analysis," in *Proc. 10th Int. Conf. Signal Process. Intell. Syst. (ICSPIS)*, Dec. 2024, pp. 132–138.

[42] D. Arnold, M. Gromov, and J. Saniie, "Network traffic visualization coupled with convolutional neural networks for enhanced IoT botnet detection," *IEEE Access*, vol. 12, pp. 73547–73560, 2024.

[43] S. Hizal, U. Cavusoglu, and D. Akgun, "A novel deep learning-based intrusion detection system for IoT DDoS security," *Internet Things*, vol. 28, Dec. 2024, Art. no. 101336.

[44] M. M. Salim, D. Camacho, and J. H. Park, "Digital twin and federated learning enabled cyberthreat detection system for IoT networks," *Future Gener. Comput. Syst.*, vol. 161, pp. 701–713, Dec. 2024.

[45] A. A. Wardana, G. Kołaczek, A. Warzyński, and P. Sukarno, "Ensemble averaging deep neural network for botnet detection in heterogeneous Internet of Things devices," *Sci. Rep.*, vol. 14, no. 1, p. 3878, Feb. 2024.

[46] A. Pakmehr, A. Aßmuth, N. Taheri, and A. Ghaffari, "DDoS attack detection techniques in IoT networks: A survey," *Cluster Comput.*, vol. 27, no. 10, pp. 14637–14668, Dec. 2024.

[47] B. Wu, P. Gysel, D. Mon Divakaran, and M. Gurusamy, "ZEST: Attention-based zero-shot learning for unseen IoT device classification," in *Proc. IEEE Netw. Operations Manage. Symp.*, May 2024, pp. 1–9.

[48] A. I. Jony and A. K. B. Arnob, "A long short-term memory based approach for detecting cyber attacks in IoT using CIC-IoT2023 dataset," *J. Edge Comput.*, vol. 3, no. 1, pp. 28–42, May 2024.
[49] G. Sri vidhya and R. Nagarajan, "A novel bidirectional LSTM model for network intrusion detection in SDN-IoT network," *Computing*, vol. 106, no. 8, pp. 2613–2642, Aug. 2024.
[50] A. M. Rahmani, A. Haider, K. Moghaddasi, F. S. Gharehchopogh, K. Aurangzeb, C. Qiu, and M. Hosseinzadeh, "Self-learning adaptive power management scheme for energy-efficient IoT-MEC systems using soft actor-critic algorithm," *Internet Things*, vol. 31, Mar. 2025, Art. no. 101587.
[51] *Number of Connected IoT Devices Worldwide From 2019 to 2030*, Statista, Hamburg, Germany, Sep. 2023.
[52] *Unit 42 IoT Threat Report*, Palo Alto Netw., 2020.
[53] *Threat Intelligence Report, 1H 2023*, NETSCOUT, 2023.
[54] A. Ali, M. A. Almaiah, F. Hajjej, M. F. Pasha, O. H. Fang, R. Khan, J. Teo, and M. Zakarya, "An industrial IoT-based blockchain-enabled secure searchable encryption approach for healthcare systems using neural network," *Sensors*, vol. 22, no. 2, p. 572, Jan. 2022.
[55] M. I. Mohmand, H. Hussain, A. A. Khan, U. Ullah, M. Zakarya, A. Ahmed, M. Raza, I. U. Rahman, and M. Haleem, "A machine learning-based classification and prediction technique for DDoS attacks," *IEEE Access*, vol. 10, pp. 21443–21454, 2022.
[56] M. Adil, R. Khan, M. A. Almaiah, M. Al-Zahrani, M. Zakarya, M. S. Amjad, and R. Ahmed, "MAC-AODV based mutual authentication scheme for constraint oriented networks," *IEEE Access*, vol. 8, pp. 44459–44469, 2020.
[57] R. Khan, I. Ali, M. A. Jan, M. Zakarya, M. A. Khan, S. S. Alshamrani, and M. Guizani, "A hybrid approach for seamless and interoperable communication in the Internet of Things," *IEEE Netw.*, vol. 35, no. 6, pp. 202–208, Nov. 2021.
[58] S. Lundberg and S. Lee, "A unified approach to interpreting model predictions," in *Proc. Adv. Neural Inf. Process. Syst.*, Jan. 2017, pp. 4765–4774.
[59] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2016, pp. 785–794.
[60] R. Shwartz-Ziv and A. Armon, "Tabular data: Deep learning is not all you need," 2022, *arXiv:2012.03225*.

PRABHAV JAIN is currently pursuing the bachelor's degree in computer science with the Thapar Institute of Engineering and Technology. His technical expertise spans Python, C++, JavaScript, and tools, such as AWS, Docker, and Wireshark. He has completed notable academic projects in machine learning and data analysis, such as wine quality prediction, android malware detection, and climate trend analysis. He is certified in AWS Cloud Foundations and Cloud Developing, showcasing his skills in cloud architecture and DevOps.

ANSHIKA RATHOUR is currently pursuing the bachelor's degree in computer science with the Thapar Institute of Engineering and Technology, with a strong foundation in full-stack development, cloud computing, and cybersecurity. She is proficient in Python, C++, JavaScript, and modern frameworks, such as React.js and Node.js. She brings technical depth across diverse domains, including AI, machine learning, and database systems. She is certified by AWS and Cisco. She is passionate about building secure, scalable tech solutions, and constantly explores new innovations in cloud architecture and ethical hacking.

AASHIMA SHARMA received the Ph.D. degree in computer science and engineering, reflecting her deep expertise and dedication to the field. She is currently an Assistant Professor with the Department of Computer Science and Engineering, Thapar Institute of Engineering and Technology, Punjab. She has authored numerous research papers published in reputed journals and international conferences, contributing significantly to advancements in these domains. Her research interests include blockchain, the Internet of Medical Things (IoMT), network security, and healthcare applications.

GURPAL SINGH CHHABRA received the bachelor's degree in computer application from HNB Garhwal University, in 2004, the master's degree in computer application from Punjab Technical University, in 2006, and the master's degree in software engineering and the Ph.D. degree specializing in cyber security and forensics from Thapar Institute of Engineering and Technology, in 2009 and 2019, respectively. He is currently an Assistant Professor with the Thapar Institute of Engineering and Technology, over 14 years. His research interests include network security, big data analytics, artificial intelligence, and cyber forensics.