IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 23 December 2023, accepted 22 February 2024, date of publication 1 March 2024, date of current version 8 March 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3371996

RESEARCH ARTICLE

SmartSentry: Cyber Threat Intelligence in Industrial IoT

SAPNA SADHWANI, URVI KAVAN MODI, RAJA MUTHALAGU, AND PRANAV M. PAWAR
Department of Computer Science, Birla Institute of Technology and Science Pilani, Dubai Campus, Dubai, United Arab Emirates
Corresponding author: Raja Muthalagu (raja.m@dubai.bits-pilani.ac.in)

ABSTRACT While the Internet of Things (IoT) paradigm has transformed connectivity, it has also brought with it previously unheard-of security risks. The categorization of IoT attacks using several machine learning techniques and a deep learning method is the main emphasis of this research. In addition to proposing a binary and multiclass classification framework with Machine Learning (ML) algorithms like Random Forest (RF), Decision tree (DT), Extra Tree Classifier (ETC), Support Vector Machine (SVM), and k-Nearest Neighbor (KNN) and Deep Learning (DL) architectures like Deep Neural Network (DNN), the study assesses a wide range of attack types in IoT environments. Benchmark datasets with real-world IoT attack scenarios, such as Edge-IIoTset, are used for experimentation. Preprocessing is done on the dataset using Principal Componenet Analysis (PCA) for feature selection, Synthetic Minority Oversampling Technique to handle class imbalance and Standard Scaling for feature scaling. These approaches' comparative performance and efficacy are examined. The outcomes indicate how successful the DL model in managing intricate attack patterns and the generalization capabilities of ML algorithms across various attack classes. The DNN model yields the best results, with 100% accuracy for binary classification, 96.15% accuracy for 6-class classification, and 94.68% accuracy for 15-class classification. Further, 10-fold cross validation has been applied to make sure that the model does not overfit. This work contributes to the improvement of IoT security mechanisms by offering insights into the selection of appropriate approaches for binary and multiclass classification of threats.

INDEX TERMS Internet of Things, machine learning, security, intrusion detection system.

I. INTRODUCTION

The development of IoT has brought about a new era of connectedness by connecting systems and gadgets to form a complex network that permeates many facets of contemporary life (refer to Figure1). With continuous communication between devices, infrastructure, and users made possible by this interconnection, industries have witnessed previously unheard-of levels of ease, efficiency, and creativity. Nonetheless, the extensive incorporation of IoT poses a significant security problem because to the widened attack surface and varied ecosystem, which reveal weaknesses that may be used by malevolent entities.

Because so many devices are linked and can be targets of cyberattacks, security in Internet of Things environments has become critical. Strong security measures are made more difficult by the heterogeneous architectures, many communication protocols, resource limitations, and other

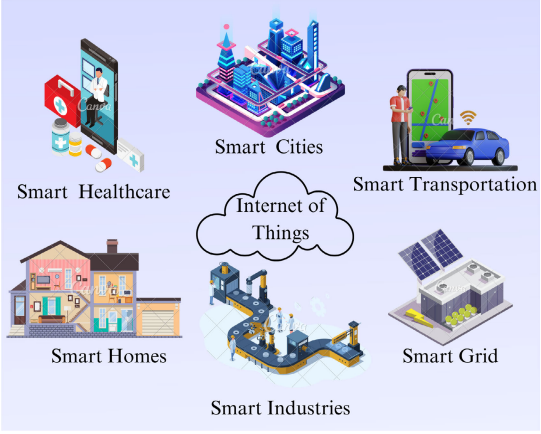


FIGURE 1. Expanse of internet of things.

The associate editor coordinating the review of this manuscript and approving it for publication was Tawfik Al-Hadhrami.

characteristics of IoT devices. The dynamic and decentralized nature of IoT networks is a challenge for traditional security measures, increasing the risk of breaches, data compromises, and interruptions to vital services.

Industrial Internet of Things (IIoT) equipment and systems are vulnerable to serious threats from ransomware, malware, denial-of-service (DoS), and phishing attempts. Ransomware has the ability to encrypt vital systems and demand money in order to unlock them, while malicious software can penetrate IIoT networks and jeopardize the integrity of industrial processes and data. DoS attacks have the ability to overload IIoT infrastructure, disrupting operations and maybe even resulting in equipment failures. Phishing attacks aim to fool administrators or staff into divulging private information or allowing illegal access to IIoT networks and devices. These cyberthreats highlight how crucial it is to have strong cybersecurity measures in place, such as network segmentation, encryption, staff education, and proactive monitoring, in order to protect IIoT settings from abuse and guarantee the dependability and security of industrial processes.

In the face of these obstacles, the use of ML methodologies has attracted interest as a potentially effective means of fortifying security inside IoT networks. With its capacity to infer patterns from data and make choices or predictions without the need for explicit programming, ML presents a strong counter to the dynamic and intricate nature of cyber threats. Within the domain of security, machine learning algorithms have the capacity to evaluate enormous volumes of data, recognize irregularities, and pinpoint patterns suggestive of possible intrusions instantly, thereby enhancing preemptive defensive systems. Nevertheless, there are still a number of serious issues with the current solutions, even with machine learning's promise to strengthen IoT security. The insufficiency of conventional rule-based or signature-based techniques in successfully thwarting complex and dynamic cyberthreats is among the main problems. These techniques are vulnerable to evasion tactics used by contemporary attackers who constantly evolve and come up with new ways to avoid detection since they frequently rely on predetermined rules or patterns.

Moreover, it might be difficult to directly implement computationally demanding machine learning algorithms on many IoT devices due to their resource constraints. It's still difficult to strike a balance between the limited processing power, energy storage of IoT devices and the requirement for strong security measures. Furthermore, the complexity of securing a variety of devices is made worse by the absence of defined security frameworks and protocols throughout the Internet of Things, which can result in inconsistent security implementations and interoperability problems.

This study intends to explore further the integration of machine learning techniques as a preventive security strategy in Internet of Things environments in light of these constraints. It examines the shortcomings of current security paradigms, investigates the potential of machine learning algorithms in threat mitigation, and suggests innovative ways to strengthen IoT system resilience against new cyberthreats.

A. OBJECTIVES

1) GENERAL OBJECTIVE

This paper is aimed to investigate and evaluate the effectiveness of using Machine Learning algorithms to improve the accuracy and efficiency of intrusion detection systems. Based on the type of data available, to understand how different parameters will affect the detection of attack.

2) SPECIFIC OBJECTIVES

The specific objectives of this paper are:

1) Identifying the appropriate ML algorithms and techniques for Intrusion Detection System (IDS) based on the nature of data being analyzed and the characteristics of network or system being protected.
2) Designing and implementing an ML/DL-based intrusion detection system and evaluating its performance in terms of false positive and false negative rates, accuracy, precision, recall, F1-score, test time and train time.
3) Analyzing the impact of various factors on the performance of ML-based IDS, such as size and quality of training data set, ML algorithm and its parameters and the nature of network or system to be protected.
4) Providing recommendations for optimizing performance which includes strategies for improving the accuracy, efficiency, and scalability of the system.

B. SCOPE

The purpose of this work is to offer a thorough assessment of the effectiveness of utilizing machine learning algorithms for intrusion detection and to make a contribution to the creation of intrusion detection systems for Industrial Internet of Things (IIoT) devices that are more effective and efficient.

C. MOTIVATION

The goal of an IDS based on ML is to increase intrusion detection's accuracy and effectiveness, particularly in the face of developing and more advanced cyber threats. As observed from the graph in Figure 2, the number of malware attacks is increasing with each year. The effectiveness of conventional intrusion detection systems, such as signature-based or rule-based systems, may be limited when it comes to identifying new or unidentified threats [31]. While harmful activity may not yet be understood or characterized, ML algorithms can learn from data and spot trends and anomalies that may be suggestive of it. This increases the adaptability and ability of machine learning-based intrusion detection systems to identify previously undiscovered threats. DL techniques can help to understand the complex patterns exhibited by the attack data.

The volume and complexity of data (refer to Figure 3) created by IoT gadgets is another reason for adopting machine

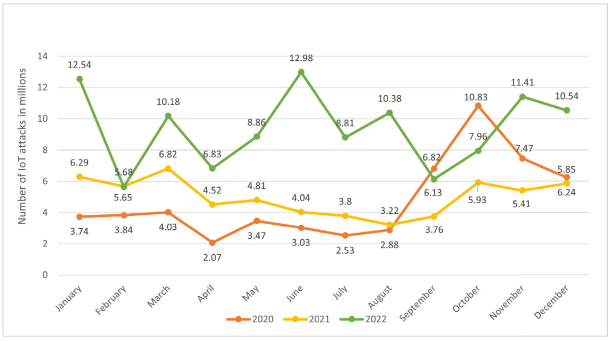


FIGURE 2. Monthly number of IoT malware attacks worldwide from 2020 to 2022 [16].

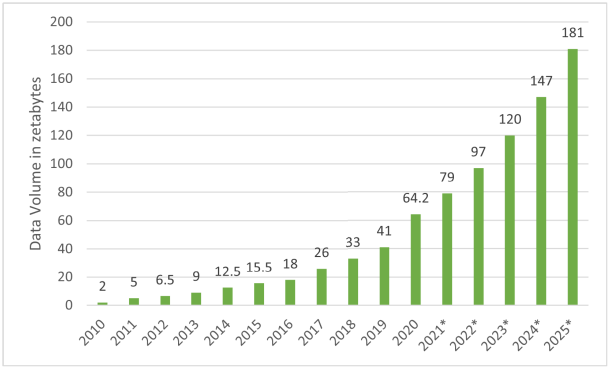


FIGURE 3. Volume of data created, captured, copied and consumed worldwide [17].

learning in intrusion detection [26]. With increasing volume of data, there is a need for security of this data. The volume of data is only going to increase with time. It is not possible to efficiently monitor this and find patterns of attacks manually. Large data sets can be efficiently analyzed by machine learning algorithms, which can also spot trends and abnormalities that traditional intrusion detection techniques or human analysts might miss [30].

D. PROBLEM DESCRIPTION

A lot of IDS are based on the ML approach have been developed [23]. However, they identify limited attacks and the data which they use is not very vast. Also, these systems use a lot of features which may not be necessary for identification [19]. Through this paper, the aim is to evaluate the performance of various machine learning and a DNN model on Edge-IIoTset. IoT devices are resource-constrained and DL techniques have high computational complexity. So, only one DNN model is chosen and its architecture is structured in a way to maintain low complexity at the same time maintaining good efficiency and performance.

E. CONTRIBUTIONS

1) This work provides a detailed review of several previous works done to classify and identify attacks on the Edge-IIoTset.
2) The entire methodology used to carry out the classification has been explained in detail with proper steps and algorithms.
3) Usage of PCA for feature reduction, SMOTE for class imbalance and standard scaling for feature scaling greatly improves the results.
4) Detailed performance analysis has been provided with respect to 2-class, 6-class and 15-class classification of attacks using the Edge-IIoTset.
5) SmartSentry uses various types of machine learning models like RF, DT, SVM, KNN, ETC and a deep learning model, DNN. For binary classification achieved accuracy is 100% for most models. Whereas, for 6- and 15- class, DNN shows the highest accuracy (96.15% and 94.86% respectively).
6) SmartSentry provides high performance measures in very short amount of time without much data loss. In case of DNN, the test times are 20.05s, 18.11s and 27.99s respectively for 2-class,6-class and 15-class respectively. This model exhibits smaller inference times due to lower parameters, lower computational requirements and hence, reduced complexity.
7) High 10-fold cross validation scores prove that the model does not overfit and will generalize on unseen data.

In Section II of this paper, previous work related to Edge-IIoTset dataset has been discussed. Section III discusses the dataset in detail. Various types of attacks are discussed along with which parameters are used to categorize each attack. Section IV discusses the methodology employed, its implementation and framework. In Section V,results of proposed model are discussed and comparisons have been made with other state-of-the-art models. Section VI takes into account a case study of industrial environmental setting where the model can be implemented. SectionVII talks about the conclusion and future work related to this research.

II. RELATED WORKS

The research [1] suggests a new data-set for assessing machine learning-based intrusion detection systems. The Edge-IIoTset dataset was created utilising a specially designed IoT/IIoT test-bed that includes a sizable representative collection of devices, sensors, protocols, and cloud/edge setups. The Edge-IIoTset dataset contains a range of assaults, including malware, man-in-the-middle, and denial-of-service assaults. A wide range of IoT and IIoT applications are covered by the dataset, which is also intended to be realistic and thorough. On the Edge-IIoTset dataset, the authors of the study assess the effectiveness of various machine learning algorithms in both centralised and federated learning modes.

A new IDS paradigm for Internet of Medical Things (IoMT) networks is put out in the research [2]. The decision tree classifier, a machine learning technique that may be used to categorise data, is the foundation of the model. The network traffic data is initially pre-processed by the suggested

IDS model to identify pertinent aspects. The decision tree classifier is then trained using these features. Once trained, the classifier can be used to categorise fresh network traffic data as benign or malicious. The UNSW-NB15 dataset, a publicly accessible dataset of IoMT network traffic, is used by the authors of the research to assess the performance of their suggested IDS model. They discover that their model outperforms other cutting-edge IDS models in terms of accuracy, achieving a figure of 94.23%.

IDS for IIoT networks are reviewed in the study [7] as they currently stand. The authors talk on the difficulties with intrusion detection in IIoT networks, including the heterogeneity of devices, their resource limitations, and the necessity of many IIoT applications for real-time functionality. In addition, the authors go over the various IDSs that can be applied to IIoT networks, such as signature-based, anomaly-based, and machine learning-based IDSs. They explore which types of IDSs are most suited for various IIoT applications and analyse the benefits and drawbacks of each type of IDS. They contend that machine learning based IDSs are the most effective strategy for IIoT networks, but that before they can be widely used in IIoT networks, a number of issues must be resolved.

A new IDS model for IIoT edge networks is put out in the paper [10]. A hybrid convolutional neural network (CNN) and long short-term memory (LSTM) architecture serves as the model's foundation. The LSTM component of the model learns temporal features from the data, whereas the CNN component of the model learns spatial information from the network traffic data. The model can identify a variety of attacks, including both known and undiscovered threats, thanks to the combination of CNN and LSTM features. The suggested IDS model is also privacy-aware, which means that it does not call for the gathering of any private information. The model suitable for deployment in edge networks, where privacy is a major concern. They discover that when classifying traffic as benign or malicious, their model achieves an accuracy of 97.85%, and when classifying 15 particular attacks, it achieves an accuracy of 97.14%.

A hybrid deep learning-based intrusion detection system designed for the IIoT is presented in this research [13]. It efficiently detects security breaches in IIoT networks by combining Recurrent Neural Networks (RNNs) and Convolutional Neural Networks (CNNs). The hybrid design of the model analyzes both temporal and geographical elements in IIoT data, improving detection accuracy. This technology is a useful tool for preserving the security of industrial environments since it provides a strong defense for IIoT networks against hostile intrusions.

The study [6] suggests a new decentralised and differentially private Federated Learning (FL)-based IDS model for IIoT networks. The model is made up of three parts: a decentralised FL algorithm, a differentially private gradient exchange method, and a key exchange protocol. Even with severe privacy restrictions, the model achieves great accuracy in detecting attacks when tested on a real-world IIoT dataset.

The paper [3] suggests a new architecture for leveraging distributed DNNs in smart IoT ecosystems to identify cyberattacks. The framework is made up of three parts: cloud servers, IoT edge devices, and IoT gateways. To increase scalability and effectiveness, the DNN model is distributed across the IoT gateways. The system achieves great accuracy in identifying cyber-attacks with a low false-positive rate when tested on a real-world IoT dataset.

The paper [4] proposes a unique method using AI to detect cyberattacks in IoT-based smart environments. It consists of a distributed malware detection middle-ware and an AI-enabled malware detection model. The model uses a dataset of known malware and safe IoT traffic, and IoT devices analyze the data. The middle-ware gathers contextual data to increase detection accuracy. The model predicts the traffic's danger or benign nature, notifying the IoT device and system administrator.

A unique FL-based technique for intrusion detection in IIoT networks is proposed in the study [5]. This decentralized approach is well-suited for IoT environments, where centralized data processing may not be feasible due to factors like network connectivity and scalability. Federated learning provides a flexible framework that can be customized to suit different industrial settings, allowing intrusion detection models to adapt and evolve over time without the need for manual intervention or centralized retraining.The suggested method consists of two parts: a global IDS model that is trained on the aggregated model updates from the local IDS models, and a local IDS model that is trained on the local data of each device. The IIoT network's resource-constrained devices are intended to operate the lightweight local IDS paradigm. A more intricate model with a greater detection accuracy is the global IDS model.

An extensive review of the security issues in ML-based IDSs and open issues and challenges for ML-based IoT security under advanced persistent threats (APTs) is given in the paper [8]. The first section of the study covers the security issues that IoT networks face, such as resource limitations, heterogeneity, and dynamic topologies. It is challenging to integrate conventional security solutions with IoT networks because of these issues. The next section of the study presents ML-based IDSs, a potentially effective method for IoT security. From past data, ML-based intrusion detection systems may recognize and identify unusual trends that can be used to identify and prevent assaults. The article then outlines the problems and obstacles that remain for ML-based IoT security in the context of APTs. These are challenging to identify and thwart. Future research areas for ML-based IoT security under APTs are covered in the paper's conclusion.

To detect anomalies in networking security, a unique one-class classifier based on polynomial interpolation is proposed in the study [9]. For the suggested classifier to function, a polynomial interpolation model of the network's

typical traffic patterns must first be built. The limited region including the upper and lower extremities of the polynomials in the training dataset is therefore defined as the normalcy area using this model. Then, if new observations are not within the normalcy range, they are labeled as abnormal. The suggested classifier outperforms rule-based intrusion detection systems like SNORT and traditional one-class classifiers like Extreme Learning Machine and Support Vector Machine models, according to the results.

To safeguard the healthcare system's whole network, a novel intrusion and malware detection system is proposed in the study [11]. The LightGBM-based IDS and the Transformer-based malware detection system (MDS) are its two primary parts. When combined, the two elements provide healthcare systems a complete security solution. Whereas the virus in network traffic is detected by the MDS, intrusions in network traffic are detected by the IDS. After being tested on four distinct datasets, the suggested approach had a 99% accuracy rate. The findings demonstrate that the suggested remedy is a successful means of defending healthcare systems from cyberattacks. Although the suggested solution is still in the early stages of research, it might significantly improve healthcare systems' security.

This study [12] proposes the DEIGASe model, which combines deep extraction with feature selection to handle anomaly detection problems in IoT-based smart cities. Using a stacked auto-encoder, deep extraction is utilized to extract features from the IoT data. The most illuminating characteristics for anomaly detection are then chosen via feature selection. A machine learning model is then trained using the chosen characteristics to identify abnormalities. Three datasets of IoT traffic from smart cities were used to assess the DEIGASe model. The outcomes demonstrated that DEIGASe works better in terms of accuracy and latency than other anomaly detection techniques now in use.

Industrial Control Protocol (ICP) fuzz testing framework MaskFuzzer [14] is built on MaskGAN and can automatically generate adversarial test cases of high quality to reveal vulnerabilities in ICP systems. To use MaskFuzzer, first train a MaskGAN model using a dataset of authentic ICP traffic. Next, we employ the MaskGAN model to produce synthetic ICP traffic that, although visually identical to real data, contains minute perturbations that may lead to vulnerabilities in ICP systems. After being tested on several real-world ICP systems, MaskFuzzer has proven to be successful in locating security holes in these systems.

The authors of the study [15] suggest an IDS for IoV networks based on federated learning. The suggested IDS detects intrusions in the in-vehicle network (IVN) using a convolutional long short-term memory (ConvLSTM) model. The IVN traffic dataset used to train the ConvLSTM model includes both malicious and genuine traffic. To further improve the efficiency of the federated learning process, the authors suggest a client selection method. A subset of cars are chosen by the client selection mechanism for each training cycle, taking into account the variety and caliber of their data. The authors demonstrate that the suggested IDS is efficient at accurately identifying intrusions in the IVN by evaluating it on a simulated IoV network.

The literature survey has been summarised in Table 1. There are many studies which have been aimed at identifying certain attacks with great accuracy. However, none of them classify attacks like DoS, DDoS, Scanning attacks, Man In the Middle (MITM), Injection Attacks and Malware attacks (refer to Table 2) across devices like Ultrasonic sensor, pH sensor, flame sensor, Heart Rate sensor, water sensor, IR receiver sensor and more which are present in Edge-IIoTset. The performance metrics fall greatly when the number of devices or attacks increases. Moreover, based on data collected, it is important to understand which set of features are required to detect which attack.

SmartSentry has high performance metrics in all types of classifications (binary and multiclass). It does not have a complex architecture. As the number of devices or attacks increase, it can easily be trained on those as well.

III. DATASET DESCRIPTION

The extensive and varied Edge-IIoTset dataset was built especially for Edge Computing in IIoT settings. Its goal is to replicate real-world scenarios found in IIoT applications using a vast array of data samples collected from different sensors, machines, and devices deployed in industrial environments. The intricacy and difficulties encountered at the crossroads of Edge Computing and IIoT are captured in this dataset.

In this section, the entire description of Edge-IIoTset has been given. Table 2 describes all the attacks from the dataset.All the various features used (refer to Table 3), test and train observations (refer to Table 4) have been mentioned in this section.

Numerous sensor readings, including those for temperature, pressure, humidity, vibration, energy consumption, and other pertinent characteristics that are essential for monitoring and control in industrial settings, are included in the dataset. It records both typical operating conditions and unusual behaviors that might be indicators of errors, anomalies, or possible security concerns. It does this across a number of time periods. Furthermore, a variety of industrial gear and processes are included in Edge-IIoTset, guaranteeing a broad and diversified dataset for security assessments, anomaly detection, predictive maintenance, and robust analysis.

The dataset consists of two parts, Attack and Normal Data. The Attack part consists of 14 .csv files for each attack type. The Normal part has data collected from 10 different types of sensors in the form of .csv files. For developing this model, all the preprocessing and analyzing has been done on the two other files provided namely: 'ML-EdgeIIoT-dataset.csv' and 'DNN-EdgeIIoT-dataset.csv'

Ten IoT devices have been utilized in the dataset to collect the data on attack and normal circumstances. The devices include industrial sensors like ultrasonic sensor, pH sensor,

TABLE 1. Literature survey.

Ref. No.	Algorithm	Results	Limitations
[1]	DT, RF, SVM, KNN, DNN	Accuracy with DNN 99.99% (2-class), 96.01% (6-class), 94.67% (15-class)	Many resources will be required to maintain and analyse this dataset if size increases
[7]	DNN, Recurrent Neural Network (RNN)	99.99% accuracy	Limited in its adaptability to new attack scenarios
[10]	CNN, LSTM Neural Network	Accuracy 97.85% (2-class), 97.24% (multiclass)	This system is much more complex compared to traditional Intrusion Detection System
[13]	CNN, LSTM Neural Network	100% accuracy with binary classification	System requires significant resources and expertise to optimize the DL models effectively
[6]	2DF-IDS	2DF-IDS shows better metrics	FL increases the computational complexity and requires more powerful hardware
[3]	DNN, SVM, RF, DT, Gradient Boosting (GB), Naïve Bayes(NB)	High Performance scores for DNN in both IoT-23 and Edge-IIoTset	Framework involves a complex architecture. Hence, deploying the solution in real-world environments may prove to be challenging due to hardware and software compatibility and communication protocols
[4]	DNN, SVM, RF, DT, GB, NB	High Performance scores for DNN in both IoT-23 and Edge-IIoTset	It has a complex architecture with multiple components
[5]	CNN, RNN	6% accuracy improvement	Federated learning-based approaches rely on a distributed data architecture, and if the data is not distributed or is highly imbalanced, the performance of the approach could be negatively affected
[8]	Comparison only	-	Paper does not include any empirical evaluation or experimentation to demonstrate the effectiveness of proposed solutions
[9]	Polynomial Interpolation, One Class Classifier	Edge IIoTset proves to be the best dataset while retaining data	MaskFuzzer is only designed for fuzz testing of ICPs and is not applicable for other devices or protocols. It also takes a significant amount of time to train
[11]	Transformer Model and LighGBM Model along with Bayesian Optimization	BERT- based transformer gives best results	These models are resource intensive as they scale for larger healthcare systems, they require more infrastructure and investment in terms of hardware
[12]	Multi Layer Perceptron, XGBoost, KNN	High efficiency and high detection rate	Deep feature extraction limits the inter-pretability of the system, and these features are not easily understood by humans. It also requires regular maintenance to ensure its effectiveness
[14]	Generative Adversarial Network (GAN)	MaskFuzzer has better TIAR	MaskGAN model takes a significant amount of time to train
[15]	ConvLSTM	Good performance in all	Complexity increases because FTL techniques are complex to implement. Use of multiple devices in FTL increases communication overhead

flame sensor, soil moisture sensor, heart rate sensor, sound detection sensor, IR receiver sensor, DHT11 sensor, modbus sensor and water sensor and more.

This dataset consists of 63 features, however, for this research, only 40 of them have been used (refer to Table 3). The rest of the features were host specific, IP and frame features which do not prove to be useful in predictions of attacks. Hence, they have been dropped (refer to Algorithm 3).

The removed features were static in nature, in the sense had no use in classification of attacks as they had the same values throughout.

The file 'ML-EdgeIIoT-dataset.csv' contains a total of 157800 records of attacks and normal samples.Similarly, the file 'DNN-EdgeIIoT-dataset.csv' has a total of 2219201 record samples. The number of records in each file and in each category have been listed in Table 4.

IV. METHODOLOGY AND IMPLEMENTATION

In this section, the entire framework of this approach has been described in detail. It includes all the algorithms, models and test and train statistics used for this approach. Python 3.11 and Jupyter Notebook were used in this paper's implementation

TABLE 2. Attacks classified in edge-IIoTset.

Attack	Specific Attack Type	Description
DoS/DDoS	DDoS TCP	An instance of a DDoS attack in which a server is bombarded with TCP packets to render it unusable.
	DDoS UDP	A form of DDoS attack that bombards a server with UDP packets to render it unusable.
	DDoS HTTP	an instance of a DDoS attack where a web server is bombarded with HTTP requests to render it unusable.
	DDoS ICMP	a form of DDoS attack where ICMP packets are swarming a server to create denial of service.
Scanning Attacks	Vulnerability Scanning Attack	Scanning for open ports and services in an effort to find weaknesses in a system or network.
	Port Scanning	technique for locating open services and ports on a system or network.
	OS Fingerprinting	a method of analyzing network data in an effort to determine the operating system installed on a network or machine.
Man in the Middle	ARP Spoofing Attack	a kind of attack where the perpetrator sends fraudulent ARP signals to link their MAC address to the IP address of another device on the network.
	DNS Spoofing Attack	a kind of attack where the perpetrator intercepts DNS communication and provides a fictitious IP address in place of the requested domain name.
Injection Attacks	SQL Injection	an attack that inserts malicious SQL code into a database by taking advantage of flaws in a web application's input validation.
	Uploading Attack	a kind of attack when a perpetrator downloads a harmful file to a server or system in order to obtain unauthorized access.
	Cross-site Scripting (XSS)	a kind of attack in which the perpetrator inserts malicious code into a website in order to steal user information or obtain unauthorized access.
Malware	Password cracking	a password guessing or cracking attempt made to gain unauthorized access to a network or system.
	Backdoor	a covert access point that enables an attacker to circumvent standard authentication and obtain unauthorised access to a system or network.
	Ransomware	malicious software that encrypts a victim's data and demands money in return for the key to unlock them.

Algorithm 1 Pseudocode for Machine Learning Models

Input: File ML-EdgeIIoT-dataset.csv
Output: Prediction for different models
1: Data Preprocessing
2: Dummy Encoding
3: SMOTE for handling imbalance
4: Split into test-train 80-20% split
5: Standard Scaling
6: Feature Selection with PCA
7: Train models like DT, RF, ETC, SVM and KNN
8: Get predictions for the different models
9: Perform 10-fold cross validation

Algorithm 2 Pseudocode for DNN

Input: File DNN-EdgeIIoT-dataset.csv
Output: Prediction for different models
1: Data Preprocessing
2: Dummy Encoding
3: SMOTE for handling imbalance
4: Split into test-train 80-20% split
5: Standard Scaling
6: Feature Selection with PCA
7: Train the DNN model
8: Get predictions for DNN model
9: Perform 10-fold cross validation

of the experiment to deal with the Edge-IIoTset dataset. The Intel(R) Core (TM) i5-1135G7 @ 2.40GHz 2.42GHz and 16 GB RAM are used in this experiment.

A. EXPERIMENTAL ENVIRONMENT

B. DATA ANALYSIS AND PREPROCESSING

Understanding the data helps to filter out unnecessary features which often make the model complex. By performing variable and count analysis all the static features were identified and dropped from the dataset. Static features include the host, IP and frame features which were unnecessary for the prediction models. In this step, null values and duplicates were also removed and a shuffle was performed on the entire dataset (refer to Algorithm 3). Table 5 and Table 6 give a summary of records in each attack category before and after pre-processing is done on the ML and DNN file respectively.

C. DUMMY ENCODING

Dummy encoding, often referred to as one-hot encoding, is a data preparation technique used to transform categorical variables into a numerical format, especially in machine learning

TABLE 3. Features used for classification.

S. No.	Feature Name	Protocol Layer	Datatype	Description
1	arp.opcode	ARP	Unsigned integer	Indicates the type of ARP message being sent or received
2	arp.hw.size	ARP	Unsigned integer	Length of hardware address
3	icmp.checksum	ICMP	Unsigned integer	IVerifies the integrity of an ICMP message
4	icmp.seq.le	ICMP	Unsigned integer	Helps in identifying the order of sent packets
5	http.content_length	HTTP	Unsigned integer	Length of content being sent in an HTTP message
6	http.request.method	HTTP	Character string	Specifies the HTTP method used in HTTP request message
7	http.referer	HTTP	Character string	Indicates URL of the webpage that led to the current request
8	http.request.version	HTTP	Character string	Specifies the version of HTTP being used in an HTTP request message
9	http.response	HTTP	Boolean	Indicates the status code and other information about the response to an HTTP request message
10	tcp.ack	TCP	Unsigned integer	Acknowledges receipt of a TCP segment or sequence of segments
11	tcp.ack_raw	TCP	Unsigned integer	The acknowledgment number in a TCP segment, used for acknowledging receipt of data
12	tcp.checksum	TCP	Label	A value used to verify the integrity of a TCP segment
13	tcp.connection.fin	TCP	Label	A TCP flag used to indicate that a connection should be terminated.
14	tcp.connection.rst	TCP	Label	A TCP flag used to indicate that a connection should be reset
15	tcp.connection.syn	TCP	Label	A TCP flag used to initiate a connection
16	tcp.connection.synack	TCP	Label	A TCP flag used to acknowledge receipt of a SYN flag and initiate a connection
17	tcp.flags	TCP	Label	A field in a TCP segment that specifies various flag controls
18	tcp.flags.ack	TCP	Boolean	A TCP flag used to acknowledge receipt of data
19	tcp.len	TCP	Unsigned integer	The length of a TCP segment in bytes
20	tcp.seq	TCP	Unsigned integer	The sequence number of a TCP segment
21	udp.stream	UDP	Unsigned integer	A sequence of UDP packets between two hosts
22	udp.time_delta	UDP	Time offset	The time difference between two UDP packets in a stream
23	dns.qry.name	DNS	Character string	The domain name being queries in a DNS request
24	dns.qry.name.len	DNS	Unsigned integer	The length of the domain name being queried in a DNS request
25	dns.qry.qu	DNS	Boolean	The query type being used in a DNS request
26	dns.retransmission	DNS	Boolean	A retransmission of a DNS query due to a previous query timeout
27	dns.retransmit.request	DNS	Label	A DNS request for retransmission of a previous query
28	mqtt.conack.flags	MQTT	Unsigned integer	Flags used in an MQTT connection acknowledgement message
29	mqtt.conflag.cleansess	MQTT	Boolean	A flag used in an MQTT connection message to specify whether the session should be cleaned up
30	mqtt.conflags	MQTT	Unsigned integer	Flags used in an MQTT connection message
31	mqtt.hdrflags	MQTT	Unsigned integer	Flags used in an MQTT message header
32	mqtt.len	MQTT	Unsigned integer	The length of an MQTT message in bytes
33	mqtt.msgtype	MQTT	Unsigned integer	The type of MQTT message being sent or received
34	mqtt.proto_len	MQTT	Unsigned integer	The length of the MQTT protocol name in an MQTT message
35	mqtt.protoname	MQTT	Unsigned integer	The name of the MQTT protocol being used in an MQTT message
36	mqtt.topic	MQTT	Character string	The topic being published or subscribed to in an MQTT message
37	mqtt.topic_len	MQTT	Unsigned integer	The length of the topic name in an MQTT message
38	mqtt.ver	MQTT	Unsigned integer	The version of the MQTT protocol being used in an MQTT message
39	Attack_label	NIL	Number	Specifies whether the data sample is attack or normal
40	Attack_type	NIL	Character String	Specifies the type of attack

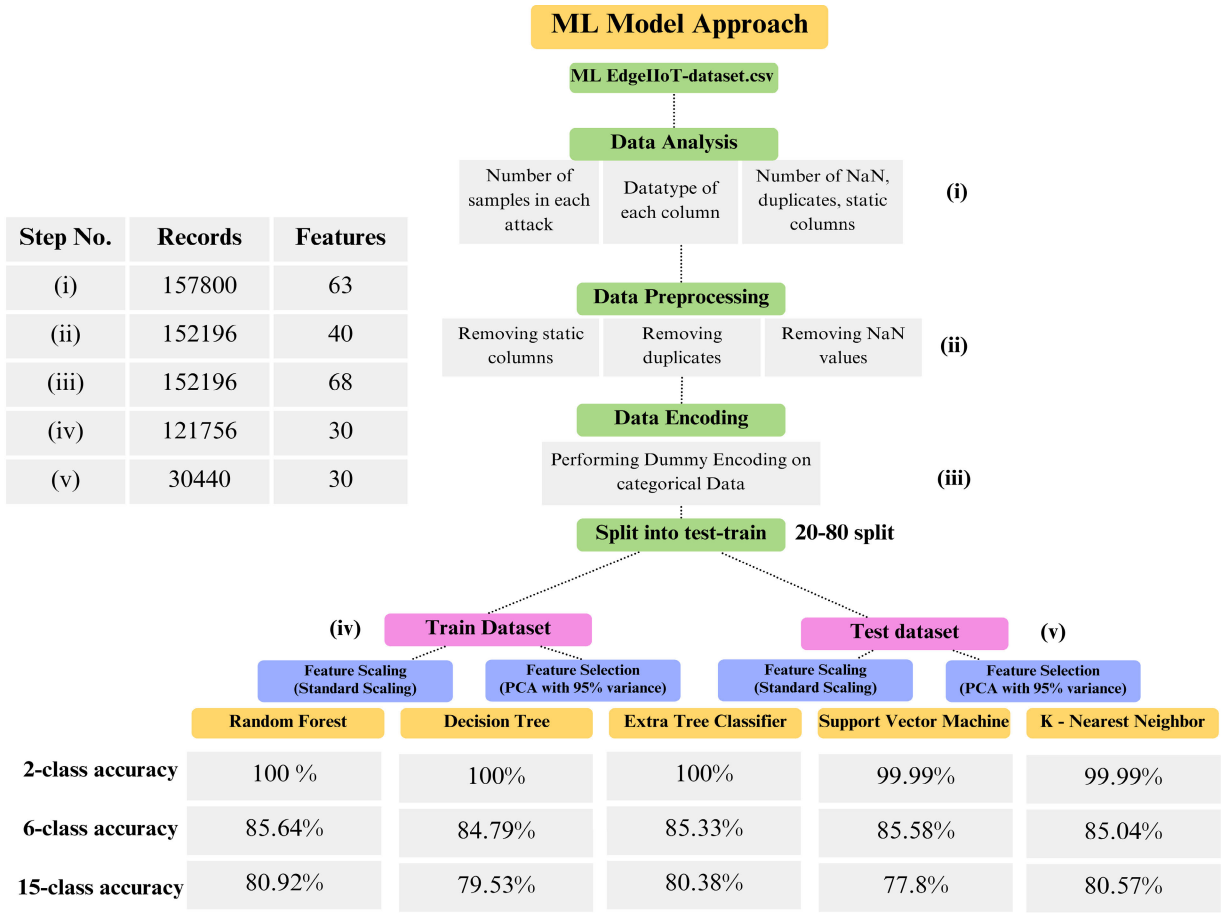
Algorithm 3 Dataset Preprocessing

Input: RawData

Output: PreProcessedData

1: $df \leftarrow RawData$
2: In(df) remove $host\_specific\_features$
3: In(df) remove $ip\_features, frame\_features$
4: In(df) remove NaN_values
5: $PreProcessedData \leftarrow shuffle(df)$

and statistical modeling. In case of categorical values, they cannot be used for analyzing as they are. They need to be converted or augmented to relevant types. For every category of a categorical feature, binary (0 or 1) variables are created using dummy encoding. It generates 'n' binary columns for a categorical variable with 'n' different categories, where each column represents a single category (refer to Algorithm 4). In this model, we have performed Dummy Encoding of 6 features namely http.request.method, http.referer, http.request.version, dns.qry.name.len, mqtt.conack.flags, mqtt.protoname, mqtt.topic. It keeps numerical values from being mistakenly understood as having any kind of meaningful order or magnitude by enabling models to handle categorical data without assuming any ordinal relationship between categories.



Step No.	Records	Features
(i)	157800	63
(ii)	152196	40
(iii)	152196	68
(iv)	121756	30
(v)	30440	30

FIGURE 4. Flowchart of approach for machine learning models.

TABLE 4. Category-wise number of samples in ML file and DL file.

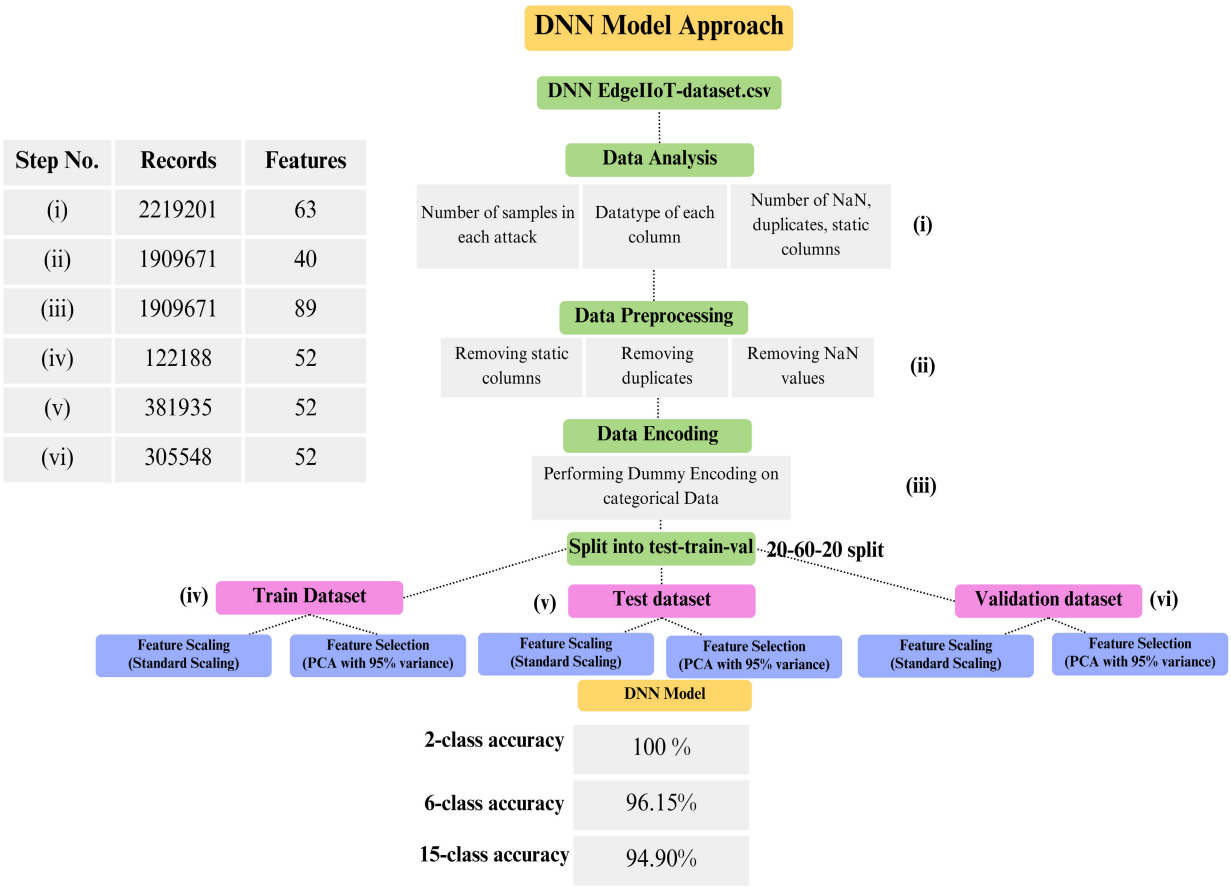
Category	Samples in ML file	Samples in DNN File
Normal	24301	1615643
DDoS_UDP	14498	121568
DDoS_ICMP	14090	116436
DDoS_HTTP	10561	49911
DDoS_TCP	10247	50062
Vulnerability_Scanner	10076	50110
Port_Scanning	10071	22564
Fingerprinting	1001	1001
MITM	1214	1214
SQL_Injection	10311	51203
Uploading	10269	37634
XSS	10052	15915
Password	9989	51053
Backdoor	10195	24862
Ransomware	10925	10925

TABLE 5. Number of samples before and after preprocessing ML-EdgeIIoT-dataset.csv.

Category	Before pre-processing	After pre-processing
Normal	24301	24101
DDoS_UDP	14498	14498
DDoS_ICMP	14090	13096
DDoS_HTTP	10561	10495
DDoS_TCP	10247	10247
Vulnerability_Scanner	10076	10062
Port_Scanning	10071	8921
Fingerprinting	1001	853
MITM	1214	358
SQL_Injection	10311	10282
Uploading	10269	10214
XSS	10052	9543
Password	9989	9972
Backdoor	10195	9865
Ransomware	10925	9689

D. HANDLING IMBALANCE

In machine learning, handling unbalanced datasets is critical, particularly when one class greatly outnumbers another, resulting in skewed models that favor the dominant class. A common technique for addressing class imbalance is the Synthetic Minority Over-sampling Technique (SMOTE), which involves oversampling the minority class to produce a dataset that is more balanced. SMOTE employs interpolation between positively aligned instances to create new instances by concentrating on the feature space. By interpolating new instances along the line segments connecting pre-existing minority class samples, SMOTE creates synthetic samples for the minority class (refer to Algorithm 5). By producing



Step No.	Records	Features
(i)	2219201	63
(ii)	1909671	40
(iii)	1909671	89
(iv)	122188	52
(v)	381935	52
(vi)	305548	52

FIGURE 5. Flowchart of approach for deep neural network.

TABLE 6. Number of samples before and after preprocessing DNN-EdgeIIoT-dataset.csv.

Category	Before pre-processing	After pre-processing
Normal	1615643	1363998
DDoS_UDP	121568	121567
DDoS_ICMP	116436	67939
DDoS_HTTP	49911	48544
DDoS_TCP	50062	50062
Vulnerability_Scanner	50110	50026
Port_Scanning	22564	19977
Fingerprinting	1001	853
MITM	1214	358
SQL_Injection	51203	50826
Uploading	37634	36807
XSS	15915	15066
Password	50153	49933
Backdoor	24862	24026
Ransomware	10925	9689

artificial examples that mimic actual minority class situations, this technique aids in increasing the representation of the minority class. This technique ensures that the models are not overfitting.

E. SPLITTING IN TEST-TRAIN

Here, the dataset is split into 20-80% of test-train. Performing this step before Feature Scaling and Feature Selection avoids leaking of data and thus prevents overfitting of the model.

F. FEATURE SCALING

Feature scaling is a technique used to normalize or standardize the range of independent variables or features in a dataset to ensure that they are on a similar scale, which can improve the performance and accuracy of machine learning models. It can also help avoid bias towards variables with larger scales or ranges. In this approach, Standard Scaling has been performed on the data. Each feature's values are scaled to have a standard deviation of one and are centered around a mean of zero in standard scaling (refer to Algorithm 6). By guaranteeing that every feature has a mean of 0 and a standard deviation of 1, standard scaling essentially brings all features to a single scale. It helps certain machine learning algorithms perform better, especially those that rely on distance metrics (e.g., K-Nearest Neighbors, Support Vector Machines), keeps features with larger magnitudes from dominating those with smaller magnitudes, and helps algorithms that are sensitive to the scale of features converge faster during training.

G. FEATURE SELECTION

To enhance model performance, lower computational cost, and avoid overfitting, a subset of relevant features from a larger collection of features in a dataset are chosen through the feature selection process. This procedure involves

Algorithm 4 Dummy Encoding

Input: A dataset with categorical variables to be dummy encoded.
Output: A new dataset with one-hot encoded binary columns.

Create an empty list to store the names of the new one-hot encoded columns.
for all categorical variables in the dataset **do**
  for all distinct categories in the variable **do**
    Create a new binary column for the category.
    for all rows in the dataset **do**
      if value in the original variable matches the category **then**
        Assign 1 to the new column
      else
        Assign 0 to the new column
      end if
    end for
    Append the new column name to the list of one-hot encoded column names.
  end for
end for
Drop the original categorical columns from the dataset.
Concatenate the new one-hot encoded columns with the dataset.

Algorithm 5 SMOTE for Imbalanced Classes

Input: T, N, k
Output: S

if $N < 100$ **then**
  Randomise T minority class samples
  $S = (N/100) * T$
  $N = 100$
end if
$N = (int)(N/100)$
for $i = 1$ to T **do**
  Compute k nearest neighbours for i
  $Populate(N, i, nnarray)$
end for
while $N \neq 0$ **do**
  $nn = random(1, k)$
  for $attr \leftarrow 1$ to $numattr$ **do**
    $dif = Sample[nnarray[nn]][attr] - Sample[i][numattr]$
    $gap = random(0, 1)$
    $Synthetic[newindex][attr] = Sample[i][attr] + gap * dif$
  end for
  $newindex++$
  $N = N - 1$
end while

Algorithm 6 Standard Scaling

Input: Dataset, X
Output: Standard Scaled Dataset, Z

Compute mean, μ_j and Standard Deviation, σ_j
$\mu_j = \frac{1}{n}\sum_{i=1}^{n} x_i^j$
$\sigma_j = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i^j - \mu_j)^2}$
For each feature, $x \in X$
$Z_i^j = (x_i^j - \mu_j)/\sigma_j$
$Z = [z_1, z_2, \ldots, z_n]$

deleting redundant, irrelevant, or noisy features. It aims to improve model accuracy and efficiency, while also improving model interpretability and generalization ability. In order to minimize the number of features (or dimensions) in a dataset while preserving the greatest amount of variance or information, Principal Component Analysis (PCA) is a dimensionality reduction approach that is often used in machine learning and data analysis. Even while PCA's primary purpose is to reduce dimensionality in data, it also indirectly helps with feature selection by highlighting the most crucial elements or characteristics that have a substantial impact on data variance. By making it possible to identify the most informative features that significantly contribute to the variability of the dataset, PCA makes feature selection easier (refer to Algorithm 7). By concentrating on the most pertinent information in the data, this dimensionality reduction helps minimize the curse of dimensionality, simplify computational complexity, and may even enhance the effectiveness of machine learning algorithms.

H. MODEL BUILDING

Now, all these pre-processed data is used to build various models like Decision Tree, Random Forest, Extra Tree Classifier, Support Vector Machine, K-Nearest Neighbor and Deep Neural Network. These specific models have been chosen, because some have been known to be robust against noisy data, while some are good for generalization. They all have trade-offs. So, by testing all of these side-by-side, it becomes easier to understand which would outperform all with minimum trade-offs.

1) DECISION TREE CLASSIFIER (DT)

A well-liked machine learning approach for classification and regression applications is the decision tree. It looks like a flowchart, with branches standing in for potential outcomes and nodes for decisions or features. The technique builds a tree structure by recursively dividing the data according to the most informative attributes. This enables it to move from the root node of the tree to a leaf node, making predictions. Decision trees are useful for both novices and specialists in data analysis since they are simple to comprehend and analyze. Additionally, they serve as a basis for more intricate ensemble techniques like Random Forests, which increases their capacity for prediction. The limitation of DT is that it is prone to overfitting very easily.

Algorithm 7 PCA Algorithm

Input: Feature set X with dimension dim
Output: k-dimensional feature set X'

Compute covariance matrix π
while $i \leq dim$ **do**
  while $j \leq dim$ **do**
    $\mu_i \leftarrow$ Sample mean of feature i
    $\mu_j \leftarrow$ Sample mean of feature j
    $\sigma_{ij} = \frac{1}{n} \sum_{k=1}^{n} (x_i^k - \mu_i)(x_j^k - \mu_j)$
    $j = j + 1$
  end while
  $i = i + 1$
end while
Split π into eigenvalues and eigenvectors
Calculate cumulative explained variance(CEV)
if $CEV \geq 0.95$ **then**
  construct projection matrix w
end if
Apply W to transform input X
Return X'

2) RANDOM FOREST CLASSIFIER (RF)

Several decision trees are combined in a Random Forest Classifier, a highly effective ensemble machine learning model that lowers overfitting and increases prediction accuracy.It works by building a forest of decision trees, each of which is trained using a different subset of the characteristics and data. The final prediction is created during classification by combining the outputs of each separate tree. In addition to being resilient, Random Forests can handle numerical and categorical input and are less likely to overfit. They are renowned for their remarkable performance, interpretability, and feature importance ranking in complicated datasets and are widely utilized for a variety of applications, including regression and classification problems.It is however, less interpretable compared to individual decsision trees.

3) EXTRA TREE CLASSIFIER (ETC)

Extremely Randomized Trees, or the Extra Tree Classifier for short, is an ensemble machine learning technique closely linked to the Random Forest. It takes random subsets of the data and creates numerous decision trees, each with a different twist. By choosing random split points for features, Extra Trees increase the degree of randomization and provide even lower bias and variation than conventional Random Forests. This method lowers the chance of overfitting while increasing model variety. Extra Trees work well with high-dimensional data and are computationally efficient. With their reliable performance, they are frequently employed in classification jobs and come in handy in difficult feature selection situations.ETCs can be less accurate due to the randomness in node splits.

4) SUPPORT VECTOR MACHINE (SVM)

A potent supervised machine learning approach for regression and classification problems is called Support Vector Machine (SVM). SVM looks for the best hyperplane to maximize the space between data points belonging to various classes. Its goal is to minimize classification mistakes and establish a well-defined decision boundary. With the introduction of kernel functions, SVM can handle both linear and nonlinear data, making it adaptable for a range of uses. SVMs are frequently utilized in domains including image recognition, text classification, and finance because they are reliable and efficient for handling high-dimensional data. They are widely used in machine learning due to their robust generalization and capacity to handle complicated datasets.They can, however, be computationally expensive with large datasets and non-linear kernels.

5) K-NEAREST NEIGHBOUR (KNN)

A straightforward and efficient supervised learning approach for regression and classification is K-Nearest Neighbors (KNN). In order to provide predictions, it searches the feature space for the K data points (neighbors) that are closest to the query point. The class of the query point is determined for categorization by the majority class among the neighbors. The technique takes the average of the K nearest neighbors' goal values for regression. KNN may be used to many sorts of data and is non-parametric and intuitive. The selection of K, data scaling, and high-dimensional data, however, could have consequences. Given these difficulties, KNN is a popular choice in pattern recognition and recommendation systems because to its ease of use and practicality. KNN can be computationally expensive with large datasets and many neighbours. It also requires careful normalization to avoid bias towards features with larger scales.

6) DEEP NEURAL NETWORK (DNN)

A subclass of artificial neural networks known as Deep Neural Networks (DNNs) are distinguished by having numerous hidden layers situated between the input and output layers. The network can extract complex and abstract patterns from the data thanks to these hidden layers. DNNs form the basis of deep learning and have transformed machine learning, resulting in advances in computer vision, natural language processing, and several other fields. Activation functions and back-propagation are used to modify model parameters during training. DNNs are capable of modeling intricate connections, but they overfit easily and need a lot of data and processing power. Despite difficulties, they greatly improved AI capabilities and are useful in many applications, including voice and picture recognition.

I. VALIDATION PHASE

The method of K-fold cross-validation is used to assess prediction models (refer to Algorithm 8). There are k folds, or subsets, inside the dataset. A distinct fold is used as the

TABLE 7. Model hyperparameters for 2-class classification.

Machine Learning Classifier	Hyperparameter Values
Random Forest	criterion: gini, min_samples_leaf: 1, min_samples_split: 2, n_estimators: 100
Decision Tree	criterion:gini, min_samples_leaf:1, min_samples_split:2,
Extra Tree Classifier	criterion: gini, min_samples_leaf: 1, min_samples_split: 2, n_estimators: 100
Support Vector Machine	C: 1.0, degree: 3, kernel: linear
K-Nearest Neighbour	leaf_size: 30, metric: minkowski, n_neighbors: 5, p: 2
Deep Neural Network	layers:3, input_dim:50, hidden_neurons:16, output_dim:2, activation:relu, softmax, optimizer:adam, epochs : 10

TABLE 8. Model hyperparameters for 6-class classification.

Machine Learning Classifier	Hyperparameter Values
Random Forest	min_samples_leaf:1, min_samples_split:2, n_estimators:100, n_jobs:None
Decision Tree	criterion:gini, min_samples_leaf:1, min_samples_split:2
Extra Tree Classifier	criterion:gini, min_samples_leaf:1, min_samples_split:2, n_estimators:100
Support Vector Machine	C:1.0, degree:3, kernel:linear
K-Nearest Neighbour	metric:minkowski, n_neighbors:5, p:2
Deep Neural Network	layers:5, input_dim:48, hidden_neurons:96, output_dim:6, activation:relu, softmax, optimizer:adam, epochs : 5

TABLE 9. Model hyperparameters for 15-class classification.

Machine Learning Classifier	Hyperparameter Values
Random Forest	max_depth:15, criterion:gini,min_samples_split:2, min_samples_leaf:1
Decision Tree	criterion:gini, min_samples_leaf:1,min_samples_split:2
Extra Tree Classifier	criterion:gini, min_samples_leaf:1, min_samples_split:2
Support Vector Machine	C:15.0, kernel:linear
K-Nearest Neighbour	metric: minkowski, n_neighbors: 3
Deep Neural Network	layers:5, input_dim:49, hidden_neurons:96, output_dim:15, activation:relu, softmax, optimizer:adam, epochs:5

validation set for each of the k training and evaluation cycles of the model. The generalization performance of the model is estimated by averaging the performance indicators from each fold.

J. PREDICTING DATA

The different models mentioned above (refer to Section IV-H are built using different hyperparameters (refer to Tables 7, 8 and 9).

K. PERFORMANCE EVALUATION

To assess the effectiveness of the created models, a variety of performance indicators including Accuracy, Precision, Recall, F1-score, and confusion matrix are utilized. In this investigation, DNN produced the best outcomes across all three categorization classes. The following part will detail the findings.

$$acc = \frac{TP_{att} + TN_{norm}}{TP_{att} + TN_{norm} + FP_{norm} + FN_{att}} \quad (1)$$

$$Pr = \frac{TP_{att}}{TP_{att} + FP_{norm}} \quad (2)$$

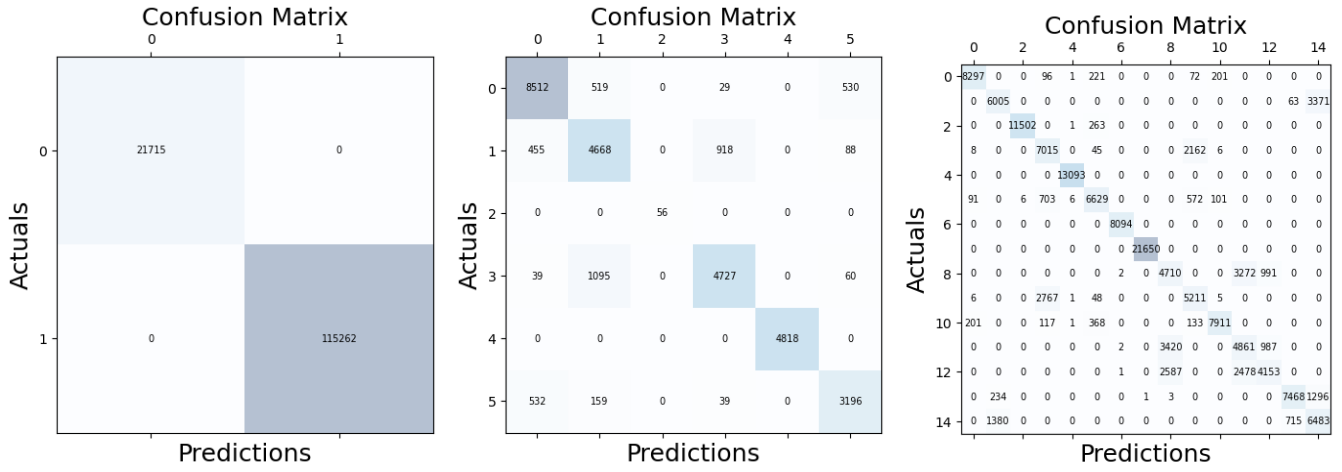
$$Rc = \frac{TP_{att}}{TP_{att} + FN_{att}} \quad (3)$$

$$F1 - score = 2 \times \frac{Pr \times Rc}{Pr + Rc} \quad (4)$$

Here, acc is Accuracy, Pr is Precision, Rc is Recall. TP_{att} is the True Positives (correct Attacks detected), TN_{norm} is True Negatives (correct Normals detected), FP_{norm} is False Positives (Attacks detected when it Normal) and FP_{att} is False Positives (Normals detected when it is Attack).

V. RESULTS AND DISCUSSION

In this section, results have been discussed with respect to confusion matrices (refer to Section V-A), performance metrics of 2-class, 6-class and 15-class (refer to Section V-B), AUC-ROC curves (refer to Section V-C), k-fold cross validation (refer to Section V-D), Time Complexity Analysis (refer to Section V-E) and comparison to state-of-the-art-models (refer to Section V-F).



(a) Confusion matrix for 2-class (b) Confusion matrix for 6-class (c) Confusion matrix for 15-class

FIGURE 6. Confusion matrices for random forest classifier.

Algorithm 8 10-Fold Cross Validation

Input: Balanced Dataset (*data*)
Output: Classifier performance parameters

```
create(classifier)
data_split(1to10) ← to_10_folds(data)
per_met ← null
for i=1 to 10 do
    data_train ← null
    j ← 1
    while j ≤ 10 do
        if j ≠ i then
            data_train ← append(data_train, data_split(j))
        end if
    end while
    train(classifier, data_train)
    test(classifier, data(i))
    per_met ← performance_metrics(classifier)
end for
print per_met
```

A. CONFUSION MATRICES

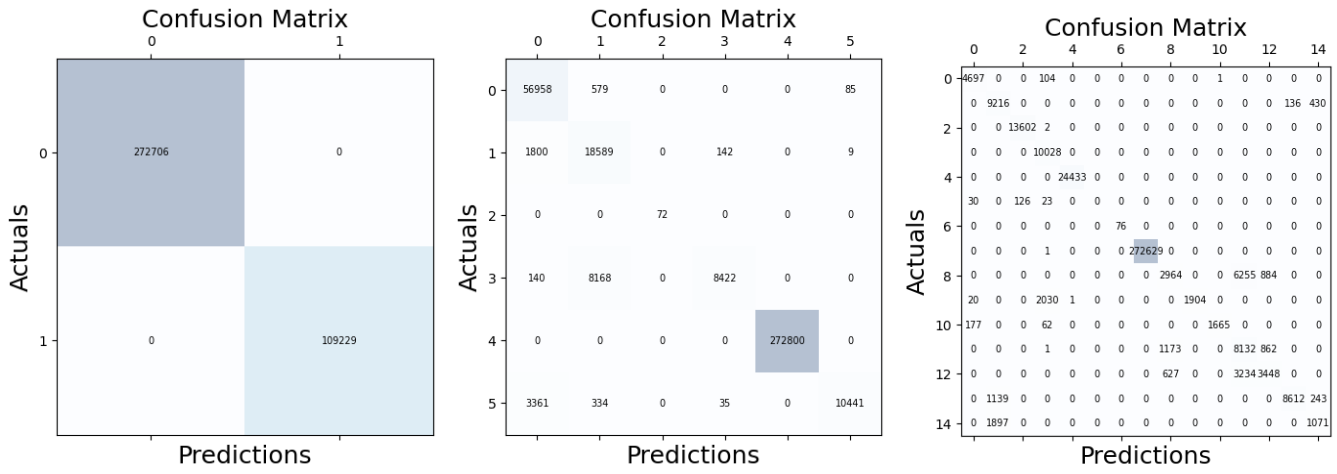
A confusion matrix is a table that is used to assess how well a classification system performs. It offers a thorough synopsis of how a model's predictions relate to the real ground truth for various classes. Understanding a classifier's accuracy, precision, recall, and other performance indicators is made easier with the help of this matrix. In this section, confusion matrices are shown for Random Forest Classifier, as it was the best performing among other machine learning models and for the DNN model. The confusion matrices are shown for 2-class, 6-class and 15-class classification (refer to Figure 6 and Figure 7). For 6-class classification, 0,1,2,3,4,5 refer to DDoS, Injection, Man-in-the-Middle (MITM), Malware, Normal and Scanning respectively. For 15-class classification, 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14 refer to Backdoor, DDoS_HTTP, DDoS_ICMP, DDoS_TCP, DDoS_UDP, Fingerprinting, MITM, Normal, Password, Port_Scanning, Ransomware, SQL_Injection, Uploading, Vulnerability_Scanner and XSS respectively. In case of 2-class, a perfect confusion matrix has been achieved with zero false positives and false negatives. For 6-class, the model faces some difficulty to detect class 3 which is Malware attacks being incorrectly classified as Injection attacks. Similarily, for DNN, a perfect confusion matrix has been achieved for 2-class. In case of 6-class and 15-class classification, some false positives are there. However, upon considering other performance metrics and cross-validation scores, the model is robust.

B. PERFORMANCE METRICS

For 2-class, almost all models give 100% results in all performance metrics (refer to Table 10). Achieving high values for performance metrics other than Accuracy, shows that these are not overfitting and are reliable in their efficiency. For 6-class and 15-class, DNN shows the best results (refer to Table 11 and Table 12).

Figure 8 and Table 13 presents the accuracy of all models in all classes.

The trend of the model's accuracy throughout epochs-iterations or runs through the complete dataset during training—is shown on accuracy graphs. As training continues, they demonstrate the model's accuracy in predicting outcomes on training and validation datasets. In an ideal scenario, training and validation accuracy would both trend higher, indicating that the model is learning efficiently and neither overfitting nor underfitting. Differences in the accuracy of training and validation might point to problems like underfitting (both accuracies stay low) or overfitting (high training accuracy but lower validation accuracy). From Figure 9(a), it is seen that accuracy shoots up in the 4 th epoch and validation accuracy is 1 throughout for 2-class. In case



(a) Confusion matrix for 2-class

(b) Confusion matrix for 6-class

(c) Confusion matrix for 15-class

FIGURE 7. Confusion matrices for DNN model.

TABLE 10. Performance Metrics for 2-class classification.

Algorithm	Metric	Normal(0)	Attack(1)
DT	Pr	1.0	1.0
	Rc	1.0	1.0
	F1	1.0	1.0
RF	Pr	1.0	1.0
	Rc	1.0	1.0
	F1	1.0	1.0
ETC	Pr	1.0	1.0
	Rc	1.0	1.0
	F1	1.0	1.0
SVM	Pr	1.0	1.0
	Rc	1.0	1.0
	F1	1.0	1.0
KNN	Pr	1.0	1.0
	Rc	1.0	1.0
	F1	1.0	1.0
DNN	Pr	**1.0**	**1.0**
	Rc	**1.0**	**1.0**
	F1	**1.0**	**1.0**

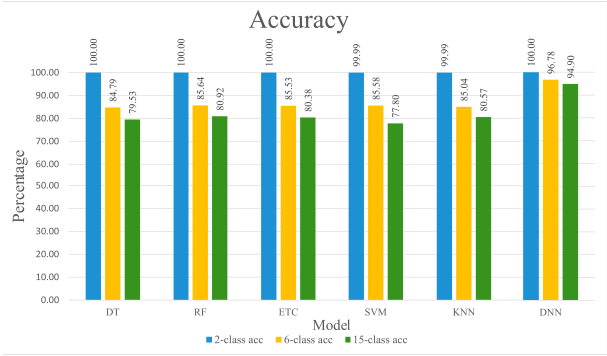


FIGURE 8. Accuracy for all models and all classes.

of 6-class, training accuracy shows a steep slope from first to second epoch (from 96.11 to 96.14), and then gradually increases till fifth epoch (till 96.16). Validation accuracy experiences a similar scenario, going from 96.10 in the first epoch to 96.13 in the fifth epoch (refer to Figure 10(a)). For 15-class, training accuracy gradually increase from 94.88 in first epoch to 94.94 in the fifth epoch. For validation accuracy, a steep slope is noticed from second to fourth epoch. In the fifth epoch, validation accuracy slightly reduces (refer to Figure 11(a)).

The trend of the model's loss function over epochs is displayed in loss graphs. In comparison to forecasts, the loss function expresses how well the model approximates the true values. Better model performance is shown by lower loss values. Just as with accuracy graphs, it's important to watch the training and validation loss trend. Both should ideally decline across epochs to show that the model is learning efficiently. Divergence between validation and training losses might point to problems such as underfitting or overfitting. For 2-class, training loss drops drastically in the second epoch and continues to remain low throughout (refer to Figure 9(b)). The validation loss, remains low-almost negligible thorughout the five epochs. The same scenario can be seen in case 6-class (refer to Figure 10(b)) and 15-class (refer to Figure 11(b)), where training loss drops significantly in second epoch and validation loss remains low.

From the graphs, it is seen that the graphs are converging, so the model is neither underfitting nor overfitting. A consistent and close trend observed in losses and accuracies indicates good generalization of data.

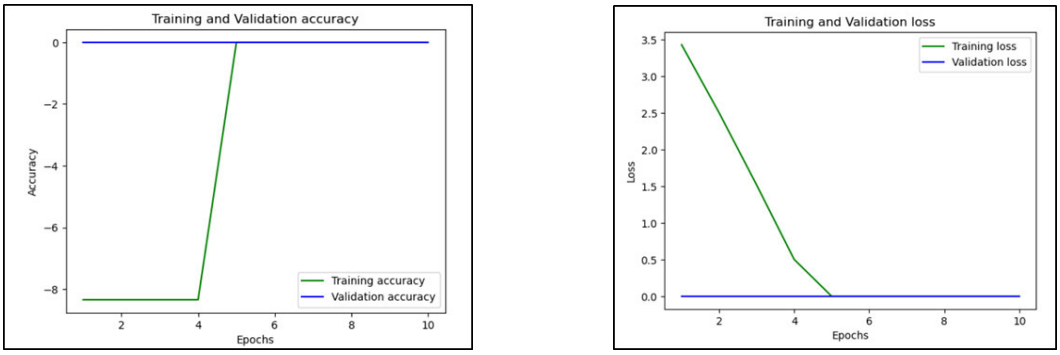
C. AUC-ROC CURVES

One popular statistic for assessing the effectiveness of binary classification algorithms is the AUC-ROC (Area Under the Receiver Operating Characteristic Curve). It assesses a model's capacity to discriminate between positive and negative classes over a range of threshold values.

Plotting the True Positive Rate (TPR) versus the False Positive Rate (FPR) for various categorization thresholds is

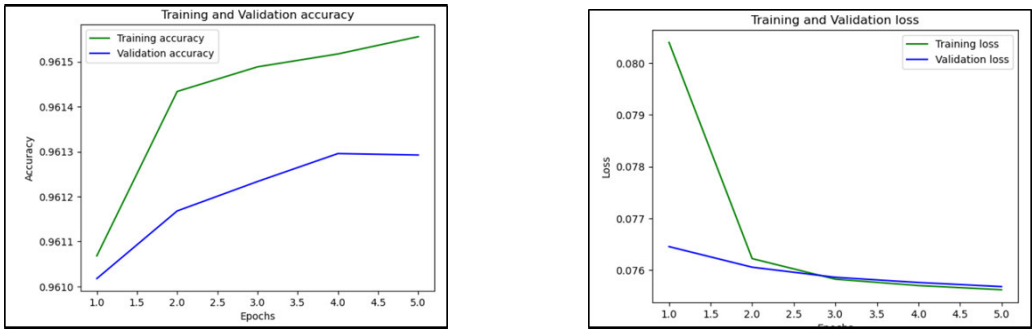
TABLE 11. Performance Metrics for 6-class classification.

Model	Metrics	Normal	DDoS	Injection	MITM	Malware	Scanning
DT	Pr	1.00	0.89	0.73	1.00	0.81	0.81
	Rc	1.00	0.89	0.73	1.00	0.81	0.82
	F1	1.00	0.89	0.73	1.00	0.81	0.81
RF	Pr	1.00	0.90	0.72	1.00	0.85	0.83
	Rc	1.00	0.88	0.79	1.00	0.85	0.83
	F1	1.00	0.89	0.75	1.00	0.82	0.83
ETC	Pr	1.00	0.89	0.72	1.00	0.83	0.82
	Rc	1.00	0.89	0.76	1.00	0.80	0.81
	F1	1.00	0.89	0.74	1.00	0.81	0.82
SVM	Pr	1.00	0.96	0.67	1.00	0.97	0.75
	Rc	1.00	0.83	0.94	1.00	0.69	0.85
	F1	1.00	0.89	0.78	1.00	0.81	0.80
KNN	Pr	1.00	0.88	0.70	1.00	0.86	0.82
	Rc	1.00	0.90	0.77	1.00	0.77	0.79
	F1	1.00	0.89	0.74	1.00	0.81	0.81
DNN	Pr	**1.00**	**0.92**	**0.67**	**1.00**	**0.98**	**1.00**
	Rc	**1.00**	**0.99**	**0.91**	**0.96**	**0.50**	**0.74**
	F1	**1.00**	**0.95**	**0.77**	**0.98**	**0.66**	**0.85**



(a) DNN Accuracy (b) DNN Loss

FIGURE 9. Accuracy and loss graphs for 2-class classification.



(a) DNN Accuracy (b) DNN Loss

FIGURE 10. Accuracy and loss graphs for 6-class classification.

known as the ROC curve. FPR is the ratio of false positives to all negatives, whereas TPR is frequently referred to as recall or sensitivity. A distinct threshold value is represented by each point on the ROC curve. The whole two-dimensional area under the ROC curve is measured by AUC. An AUC of 1.0 for a perfect classifier would mean that, at any threshold, it achieves complete classification between positive and negative classes. An AUC of 0.5 for a random classifier would suggest no capacity to discriminate apart from chance. All the classifiers in this research have AUC of 1.0, indicating they are perfect classifiers (refer to Figure 12).

D. K-FOLD CROSS VALIDATION

One method for assessing machine learning models is K-fold cross-validation. The process is dividing the dataset into K subsets, using K-1 subsets for model training, and using the remaining subset for testing. Every subset serves as the test set once during the K repetitions of this operation. This ensures that every data point is used for both training and testing, maximizing the use of available data. It lessens overfitting, encourages generalization, and aids in evaluating a model's performance. The findings are averaged over K rounds, giving a more accurate evaluation

of a model's capabilities. Since the model is evaluated on multiple different test sets, k-fold cross-validation provides a more realistic assessment of the model's generalization performance. It helps in determining how well the model performs on unseen data, which is crucial for assessing its robustness. 10-fold cross validation has been used on all models in this research (refer to Table 14). The consistency observed in the 10-fold cross validation scores, shows the robustness and its ability generalize unseen data. Highest scores have been observed in case of DNN for 2-class (1.00), 6-class (0.96) and 15-class (0.94) which also coincide with its accuracies achieved.

E. TIME COMPLEXITY ANALYSIS

Other than performance metrics, another important aspect to evaluate a model is it's time complexity. If a model performs well but takes an unreasonable amount of time, it is not feasible to deploy such a model. Especially, in cases like intrusion detection, where attacks need to be detected with milliseconds, high time complexities can prove to be very costly. Larger datasets and more complicated models may be handled more effectively and scalable by algorithms with reduced time complexity. Determining hardware requirements and model training timeframes, as well as efficiently allocating computing resources, are all aided by an understanding of temporal complexity. This analysis helps determine which method, given the size of the dataset and available processing resources, is best suited for a given task. The Table 15 gives the train and test time complexities of all models used in SmartSentry. From the table, Decision Tree shows the best time complexities for both testing and training. Table 16 provides a concise view of the test and train times of the different algorithms used. The DNN model, inspite of having high complexities, their testing times are significantly low. Since, the DNN model have otherwise shown the best performance metrics, it is a good choice for an intrusion detection system. The machine learning algorithms like DT, RF have low testing and training times, but their other performance metrics makes them a poor choice for an IDS.This is because, DT is prone to overfitting and the results given by RF are not that interpretable. They will not be able to generalize on unseen data.

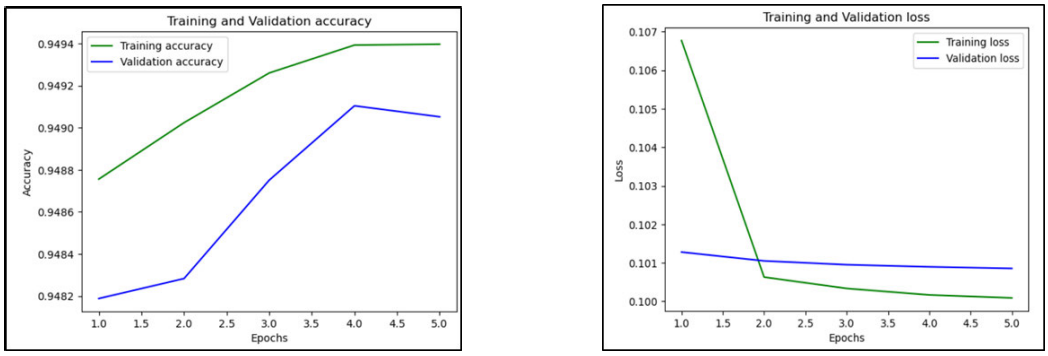
F. COMPARATIVE ANALYSIS

In this section, comparison has been made of the proposed model to the previous studies done in this area. Comparison has been made in terms of (1) Best performing state-of-the-art models and types of classification with their performance metrics (refer to Table 17) (2) Method employed by the different state-of-the-art models in their studies (refer to Table18).

The model proposed in [1] is completely outperformed by the proposed model in terms of all performance measures. Moreover, this paper does not take into account test time, train time and cross validation scores of the models. Paper [3] and [4] have been written by the same authors. This paper

TABLE 12. Performance Metrics for 15-class classification.

Algorithm	Metric	Backdoor	Ddos_HTTP	DDoS_ICMP	DDoS_TCP	DDoS_UDP	Fingerprinting	MITM	Normal	Password	Port_scanning	Ransomware	SQL_Injection	Uploading	Vulnerability_Scanner	XSS
DT	Pr	0.90	0.69	0.99	0.71	1.00	0.79	1.00	1.00	0.45	0.70	0.89	0.47	0.51	0.83	0.59
	Rc	0.92	0.66	0.99	0.73	1.00	0.81	1.00	1.00	0.45	0.65	0.88	0.46	0.51	0.84	0.60
	F1	0.91	0.67	0.99	0.72	1.00	0.80	1.00	1.00	0.45	0.68	0.89	0.47	0.51	0.84	0.60
RF	Pr	0.96	0.79	1.00	0.66	1.00	0.88	1.00	1.00	0.44	0.64	0.96	0.46	0.68	0.91	0.58
	Rc	0.93	0.64	0.98	0.76	1.00	0.82	1.00	1.00	0.52	0.65	0.91	0.52	0.46	0.83	0.76
	F1	0.95	0.70	0.99	0.70	1.00	0.85	1.00	1.00	0.48	0.64	0.93	0.49	0.54	0.87	0.66
ETC	Pr	0.94	0.69	1.00	0.73	1.00	0.83	1.00	1.00	0.44	0.70	0.93	0.46	0.54	0.83	0.59
	Rc	0.94	0.68	0.98	0.77	1.00	0.81	1.00	1.00	0.49	0.71	0.92	0.46	0.48	0.84	0.58
	F1	0.94	0.69	0.99	0.75	1.00	0.82	1.00	1.00	0.46	0.70	0.92	0.46	0.51	0.84	0.59
SVM	Pr	0.63	0.87	1.00	1.00	1.00	0.86	1.00	1.00	0.42	0.57	0.74	0.47	0.66	0.85	0.58
	Rc	0.78	0.61	0.98	0.58	1.00	0.84	1.00	1.00	0.71	0.99	0.42	0.34	0.39	0.85	0.78
	F1	0.70	0.71	0.99	0.73	1.00	0.85	1.00	1.00	0.52	0.73	0.53	0.39	0.49	0.85	0.67
KNN	Pr	0.92	0.68	1.00	0.77	1.00	0.82	1.00	1.00	0.43	0.74	0.95	0.50	0.60	0.87	0.65
	Rc	0.94	0.73	0.99	0.79	1.00	0.66	1.00	1.00	0.55	0.74	0.93	0.43	0.52	0.84	0.62
	F1	0.93	0.70	1.00	0.78	1.00	0.73	1.00	1.00	0.48	0.74	0.94	0.46	0.55	0.86	0.64
DNN	**Pr**	**0.96**	**0.73**	**1.00**	**0.82**	**1.00**	**0.45**	**1.00**	**1.00**	**0.45**	**1.00**	**0.96**	**0.91**	**0.68**	**0.96**	**0.56**
	Rc	**0.97**	**0.95**	**1.00**	**1.00**	**1.00**	**0.29**	**1.00**	**1.00**	**0.91**	**0.49**	**0.89**	**0.19**	**0.48**	**0.85**	**0.25**
	F1	**0.96**	**0.82**	**1.00**	**0.90**	**1.00**	**0.35**	**1.00**	**1.00**	**0.60**	**0.66**	**0.92**	**0.31**	**0.56**	**0.90**	**0.35**



(a) DNN Accuracy (b) DNN Loss

FIGURE 11. Accuracy and loss graphs for 15-class classification.

TABLE 13. Accuracy comparison of all models.

Algorithm	2-class	6-class	15-class
DT	100.00	84.79	79.53
RF	100.00	85.64	80.92
ETC	100.00	85.33	80.38
SVM	99.99	85.58	77.80
KNN	99.99	85.04	80.57
DNN	**100.00**	**96.78**	**94.90**

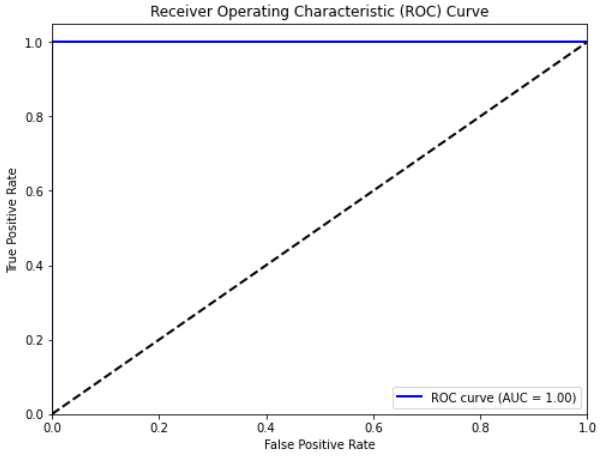


FIGURE 12. AUC-ROC curve for 2-class classifiers.

TABLE 14. 10-fold cross validation scores.

Model	2-class accuracy	6-class accuracy	15-class accuracy
DT	1.00	0.95	0.93
RF	1.00	0.96	0.94
ETC	1.00	0.95	0.93
SVM	1.00	0.95	0.93
KNN	1.00	0.95	0.93
DNN	1.00	0.96	0.94

TABLE 15. Train and test time complexity of models used.

Model	Train Time Complexity	Test Time Complexity
Random Forest	O(k'*n*log(n)*m)	O(m*k')
Decision Tree	O(n*log(n)*m)	O(m)
Extra Tree Classifier	O(k'*m*n*log(n))	O(k'*n*log(n))
Support Vector Machine	$O(n^2)$	O(n'*m)
K-Nearest Neighbour	O(k*n*m)	O(n*m)
Deep Neural Network	O(n*p*q)	O(n*p)

TABLE 16. Train and test time for all ML Models (in seconds).

Algorithm	2-class		6-class		15-class	
	Train	Test	Train	Test	Train	Test
DT	0.19	0.01	7.48	0.0075	11.08	0.0066
RF	4.21	0.45	64.08	0.62	57.43	0.55
ETC	1.14	0.53	17.48	0.74	17.63	0.94
SVM	0.32	0.21	764.77	102.44	1748.23	99.26
KNN	0.0053	176.29	0.09	93.71	0.01	57.44
DNN	715.36	20.05	373.88	18.11	367.58	27.99

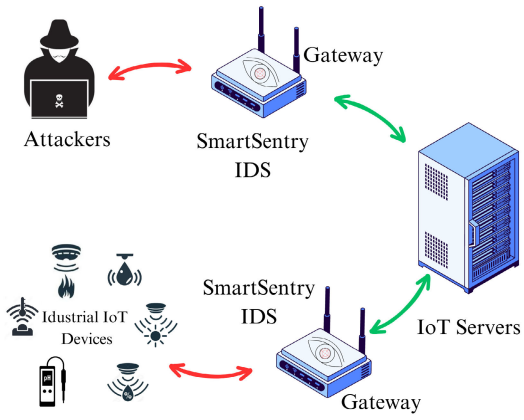


FIGURE 13. SmartSentry in industrial IoT environment.

does not perform binary classification and has not performed cross-validation to ensure that the model is not overfitting. Moreover, test and train times have not been mentioned. This paper [6], has a good accuracy, but its other metrics suggest that the model may not generalize well to unseen data i.e. the model could be overfitting. Lack of validation scores is also a major limitation. In paper [9], the authors suggest a one-class classifier. Research has only been done on binary classification. Proposed model, outperforms this study in binary classification with a DNN model by approximately 3% greater accuracy. Research [10] presents a CNN-LSTM model for binary and multiclass classification. It has a greater accuracy than the DNN model proposed in this research. However, upon comparing other performance metrics like F1-score, it has low values. It is seen that the hybrid CNN-LSTM model is overfitting and will not be able to generalize

TABLE 17. Comparison to other models based on performance metrics.

Classification Type	Ref. No.	[1]	[3]	[4]	[6]	[9]	[10]	[12]	**Proposed Model**
2- class	Best Algorithm	DNN	-	-	-	Poly BR	CNN-LSTM	Xgboost	**DNN**
	Acc	99.99	-	-	-	97.27	97.85	100.00	**100.00**
	Pr	99.98	-	-	-	96.03	98.13	100.00	**100.00**
	Rc	100.00	-	-	-	94.82	97.59	100.00	**100.00**
	F1	99.98	-	-	-	-	97.81	100.00	**100.00**
	Train Time	-	-	-	-	-	-	2	**715.36**
	Test Time	-	-	-	-	-	-	0.077	**20.05**
	Cross Validation	-	-	-	-	-	-	-	**1.00**
	AUC	-	-	-	-	-	-	100.00	**100.00**
6-class	Best Algorithm	DNN	-	-	-	-	-	-	**DNN**
	Acc	96.01	-	-	-	-	-	-	**96.78**
	Pr	91.83	-	-	-	-	-	-	**96.82**
	Rc	84.16	-	-	-	-	-	-	**96.16**
	F1	87.66	-	-	-	-	-	-	**95.97**
	Train Time	-	-	-	-	-	-	-	**373.88**
	Test Time	-	-	-	-	-	-	-	**18.11**
	Cross Validation	-	-	-	-	-	-	-	**0.96**
15-class	Best Algorithm	DNN	DNN	DNN	Centralized Model	-	CNN-LSTM	-	**DNN**
	Acc	94.67	94.00	94.00	94.84	-	97.14	-	**94.90**
	Pr	80.20	98.00	98.00	79.00	-	82.32	-	**95.33**
	Rc	77.00	89.00	89.00	84.00	-	72.66	-	**94.90**
	F1	77.33	93.00	93.00	79.00	-	74.62	-	**94.62**
	Train Time	-	-	-	-	-	-	-	**367.58**
	Test Time	-	-	-	-	-	-	-	**27.99**
	Cross Validation	-	-	-	-	-	-	-	**0.94**

TABLE 18. Comparison to other models based on methodology used.

Approach	Edge-IIoTseT [1]	AI Enabled Middleware [4]	Hybrid CNN-LSTM [10]	Hybrid Deep Learning [13]	DEIGASe [12]	Proposed Model
Dummy Encoding	✓	×	×	✓	×	✓
Balancing Technique (SMOTE)	✓	✓	×	×	×	✓
Feature Scaling (Standard Scaling)	✓	×	×	✓	×	✓
Feature Selection (PCA)	×	×	×	×	✓	✓
Cross validation (10-fold)	×	×	×	×	✓	✓

to unseen data. In [12], a deep extraction approach has been combined with feature selection. Although, the results suggest zero FPR, work has not been done on 6-class or 15-class classification. Moreover, no results are provided to make sure that the model does not overfit the data.

The proposed model, hence, outperforms the compared exisiting state-of-the-art models in terms of performance metrics like Accuracy, Precision, Recall, F1-score, Train Time, Test Time, AUC-ROC Curve. Moreover, 10-fold cross validation scores suggest that the model does not overfit the data.

VI. CASE STUDY: PROTECTING ENVIRONMENTAL SENSORS IN INDUSTRIAL SETTINGS THROUGH SMARTSENTRY

The previous sections of this research discuss the methodology and performance of the proposed model. This section gives an insight on how it can be implemented in real-life scenarios. Taking the example of an industrial setting having various environmental sensors like Flame sensor, Water sensor, Temperature sensor, Humidity sensor, pH sensor, Ultrasonic sensor.

By continually monitoring conditions, environmental sensors play vital roles in assuring safety and compliance in industrial settings. These sensors do, however, provide a growing danger of data manipulation, system breach, and safety hazards due to their increased susceptibility to cyber-attacks. Tough industrial settings make these weaknesses worse. An inventive cybersecurity solution designed for environmental sensors is SmartSentry. SmartSentry protects sensor networks against cyberattacks by using real-time monitoring and intrusion detection. Its powerful security procedures and tamper-proof design efficiently decrease risks in environmental monitoring systems by protecting industrial operations from cyber threats, guaranteeing the integrity of sensor data, and preventing unauthorized access. Integrating SmartSentry with the Gateway ensures that the servers remain protected in both cases- whether it be while collecting data from the sensors, or while someone tries to attack the servers.

VII. CONCLUSION AND FUTURE WORK

Expanding on the discussed approach, the development of SmartSentry with high-performance metrics across various classification models showcases its potential in enhancing network security. Notably, the utilization of Deep Neural Networks (DNN) yielded the best results among all the models employed (100% accuracy for binary classification). This success demonstrates the effectiveness of deep learning techniques in accurately identifying and mitigating potential threats. To further improve the model's performance, future research can explore the application of GridSearchCV, a technique that systematically tunes hyperparameters across multiple models. By fine-tuning the parameters, better accuracy can be achieved, leading to more reliable intrusion detection. Additionally, investigating different feature selection techniques can enhance the model's efficiency. By identifying the most relevant and informative features, the models can focus on key indicators of intrusion, improving their overall performance. Moreover, the application of this approach extends beyond traditional network security. As illustrated in the case study, it can be applied to smart city models, protecting devices within the infrastructure against various cyberattacks. This highlights the versatility and potential of the proposed approach in securing complex and interconnected systems.

In conclusion, the ongoing research and development in intrusion detection systems holds great promise for enhancing network security. By incorporating advanced techniques, such as RNN, CNN and hyperparameter tuning, and exploring different feature selection methods, the model can continue to evolve and effectively counter emerging cyber threats. Although this study has shed light on how IoT assaults are classified, it also opens up a number of new research directions. To test the resilience and efficacy of the same strategy, simulating it on several datasets is a viable avenue. To test how successfully our IDS responds to real-world situations, it can also be integrated into genuine IoT devices and assaults may be replicated. The more assaults the model is trained on, the more it will be able to generalize to new forms of data. Filling up these research voids will help safeguard the integrity of data gathered from IIoT devices in addition to enhancing cyber security.

REFERENCES

[1] M. A. Ferrag, O. Friha, D. Hamouda, L. Maglaras, and H. Janicke, "Edge-IIoTset: A new comprehensive realistic cyber security dataset of IoT and IIoT applications for centralized and federated learning," *IEEE Access*, vol. 10, pp. 40281–40306, 2022.

[2] K. Gupta, D. K. Sharma, K. Datta Gupta, and A. Kumar, "A tree classifier based network intrusion detection model for Internet of medical things," *Comput. Electr. Eng.*, vol. 102, Sep. 2022, Art. no. 108158.

[3] G. Bhandari, A. Lyth, A. Shalaginov, and T.-M. Grønli, "Distributed deep neural-network-based middleware for cyber-attacks detection in smart IoT ecosystem: A novel framework and performance evaluation approach," *Electronics*, vol. 12, no. 2, p. 298, Jan. 2023.

[4] G. P. Bhandari, A. Lyth, A. Shalaginov, and T.-M. Grønli, "Artificial intelligence enabled middleware for distributed cyberattacks detection in IoT-based smart environments," in *Proc. IEEE Int. Conf. Big Data*, Dec. 2022, pp. 3023–3032.

[5] M. M. Rashid, S. U. Khan, F. Eusufzai, M. A. Redwan, S. R. Sabuj, and M. Elsharief, "A federated learning-based approach for improving intrusion detection in industrial Internet of Things networks," *Network*, vol. 3, no. 1, pp. 158–179, Jan. 2023.

[6] O. Friha, M. A. Ferrag, M. Benbouzid, T. Berghout, B. Kantarci, and K.-K.-R. Choo, "2DF-IDS: Decentralized and differentially private federated learning-based intrusion detection system for industrial IoT," *Comput. Secur.*, vol. 127, Apr. 2023, Art. no. 103097.

[7] O. Cheikhrouhou, O. B. Fredj, N. Atitallah, and S. Hellal, "Intrusion detection in industrial IoT," in *Proc. 15th Int. Conf. Secur. Inf. Netw. (SIN)*, Nov. 2022, pp. 01–04, doi: 10.1109/SIN56466.2022.9970535.

[8] Z. Chen, J. Liu, Y. Shen, M. Simsek, B. Kantarci, H. T. Mouftah, and P. Djukic, "Machine learning-enabled IoT security: Open issues and challenges under advanced persistent threats," *ACM Comput. Surveys*, vol. 55, no. 5, pp. 1–37, May 2023, doi: 10.1145/3530812.

[9] P. Dini, A. Begni, S. Ciavarella, E. De Paoli, G. Fiorelli, C. Silvestro, and S. Saponara, "Design and testing novel one-class classifier based on polynomial interpolation with application to networking security," *IEEE Access*, vol. 10, pp. 67910–67924, 2022, doi: 10.1109/ACCESS.2022.3186026. https://doi.org/10.1109/access.2022.3186026.

[10] E. M. D. Elias, V. S. Carriel, G. W. De Oliveira, A. L. Dos Santos, M. Nogueira, R. H. Junior, and D. M. Batista, "A hybrid CNN-LSTM model for IIoT edge privacy-aware intrusion detection," in *Proc. IEEE Latin-American Conf. Commun. (LATINCOM)*, Nov. 2022, pp. 1–6, doi: 10.1109/LATINCOM56090.2022.10000468.

[11] A. Ghourabi, "A security model based on LightGBM and transformer to protect healthcare systems from cyberattacks," *IEEE Access*, vol. 10, pp. 48890–48903, 2022, doi: 10.1109/ACCESS.2022.3172432. https://doi.org/10.1109/access.2022.3172432.

[12] C. Hazman, S. Benkirane, and M. Azrour. (2022). *DEIGASe: Deep Extraction and Information Gain for an Optimal Anomaly Detection in IoT-based Smart Cities*. [Online]. Available: https://doi.org/10.21203/rs.3.rs-2141835/v1

[13] A. Khacha, R. Saadouni, Y. Harbi, and Z. Aliouat, "Hybrid deep learning-based intrusion detection system for industrial Internet of Things," in *Proc. 5th Int. Symp. Informat. Appl. (ISIA)*, Nov. 2022, pp. 1–6, doi: 10.1109/ISIA55826.2022.9993487.

[14] W. Sun, B. Zhang, J. Ding, and M. Tang, "MaskFuzzer: A MaskGAN-based industrial control protocol fuzz testing framework," in *Proc. IEEE Int. Conf. Smart Internet Things (SmartIoT)*, Aug. 2022, pp. 51–57, doi: 10.1109/SmartIoT55134.2022.00018.

[15] J. Yang, J. Hu, and T. Yu, "Federated AI-enabled in-vehicle network intrusion detection for Internet of Vehicles," *Electronics*, vol. 11, no. 22, p. 3658, Nov. 2022, doi: 10.3390/electronics11223658.

[16] Published by Ani Petrosyan A.6. (2022). *Monthly Number of IoT Attacks Global 2022*. [Online]. Available: https://www.statista.com/statistics/1322216/worldwide-internet-of-things-attacks/

[17] O. Djuraskovic. (2023). *30+ Big Data Statistics-Amount of Data Generated in the World*. [Online]. Available: https://firstsiteguide.com/big-data-stats/

[18] I. Tareq, B. M. Elbagoury, S. El-Regaily, and E.-S.-M. El-Horbaty, "Analysis of ToN-IoT, UNW-NB15, and edge-IIoT datasets using DL in cybersecurity for IoT," *Appl. Sci.*, vol. 12, no. 19, p. 9572, Sep. 2022.

[19] S. Haque, F. El-Moussa, N. Komninos, and R. Muttukrishnan, "Identification of important features at different IoT layers for dynamic attack detection," in *Proc. IEEE IEEE 9th Intl. Conf. Big Data Secur. Cloud Intl. Conf. High Perform. Smart Comput.*, May 2023, pp. 84–90.

[20] W. I. Khedr, A. E. Gouda, and E. R. Mohamed, "FMDADM: A multi-layer DDoS attack detection and mitigation framework using machine learning for stateful SDN-based IoT networks," *IEEE Access*, vol. 11, pp. 28934–28954, 2023.

[21] L. Arya and G. P. Gupta, "Ensemble filter-based feature selection model for cyber attack detection in industrial Internet of Things," in *Proc. 9th Int. Conf. Adv. Comput. Commun. Syst. (ICACCS)*, vol. 1, Mar. 2023, pp. 834–840.

[22] Z. A. El Houda, B. Brik, A. Ksentini, and L. Khoukhi, "A MEC-based architecture to secure IoT applications using federated deep learning," *IEEE Internet Things Mag.*, vol. 6, no. 1, pp. 60–63, Mar. 2023.

[23] M. Alsharif and D. B. Rawat, "Machine learning enabled intrusion detection for edge devices in the Internet of Things," in *Proc. IEEE 13th Annu. Comput. Commun. Workshop Conf. (CCWC)*, Mar. 2023, pp. 0361–0367.
[24] S. Li, X. Xu, G. Zhou, Y. Wang, Z. Li, Y. Zhao, and W. Zhao, "Cybersecurity status assessment of cloud manufacturing systems based on semiquantitative information," *IEEE Access*, vol. 11, pp. 43458–43471, 2023.
[25] K. Tomar, K. Bisht, K. Joshi, and R. Katarya, "Cyber attack detection in IoT using deep learning techniques," in *Proc. 6th Int. Conf. Inf. Syst. Comput. Netw. (ISCON)*, Mar. 2023, pp. 1–6.
[26] J. Kim, J. Park, and J.-H. Lee, "Analysis of recent IIoT security technology trends in a smart factory environment," in *Proc. Int. Conf. Artif. Intell. Inf. Commun. (ICAIIC)*, Feb. 2023, pp. 840–845.
[27] H. A. Madni, R. M. Umer, and G. L. Foresti, "Blockchain-based swarm learning for the mitigation of gradient leakage in federated learning," *IEEE Access*, vol. 11, pp. 16549–16556, 2023.
[28] S. Kantimahanthi, J. V. D. Prasad, S. Chanamolu, and K. Kommaraju, "Machine learning approaches in cyber attack detection and characterization in IoT enabled cyber-physical systems," in *Proc. Int. Conf. Intell. Data Commun. Technol. Internet Things (IDCIoT)*, Jan. 2023, pp. 136–142.
[29] C. Avais Hanif, M. Ali Mughal, M. Attique Khan, U. Tariq, Y. Jin Kim, and J.-H. Cha, "Human gait recognition based on sequential deep learning and best features selection," *Comput., Mater. Continua*, vol. 75, no. 3, pp. 5123–5140, 2023.
[30] A. Pinto, L.-C. Herrera, Y. Donoso, and J. A. Gutierrez, "Survey on intrusion detection systems based on machine learning techniques for the protection of critical infrastructure," *Sensors*, vol. 23, no. 5, p. 2415, Feb. 2023.
[31] M. Nuaimi, L. C. Fourati, and B. B. Hamed, "Intelligent approaches toward intrusion detection systems for industrial Internet of Things: A systematic comprehensive review," *J. Netw. Comput. Appl.*, vol. 215, Jun. 2023, Art. no. 103637.
[32] B. Kaur, S. Dadkhah, F. Shoeleh, E. C. P. Neto, P. Xiong, S. Iqbal, P. Lamontagne, S. Ray, and A. A. Ghorbani, "Internet of Things (IoT) security dataset evolution: Challenges and future directions," *Internet Things*, vol. 22, Jul. 2023, Art. no. 100780.
[33] V. Hnamte and J. Hussain, "DCNNBiLSTM: An efficient hybrid deep learning-based intrusion detection system," *Telematics Informat. Rep.*, vol. 10, Jun. 2023, Art. no. 100053.
[34] W. Ding, M. Abdel-Basset, and R. Mohamed, "DeepAK-IoT: An effective deep learning model for cyberattack detection in IoT networks," *Inf. Sci.*, vol. 634, pp. 157–171, Jul. 2023.
[35] J. Liu, Y. Tang, H. Zhao, X. Wang, F. Li, and J. Zhang, "CPS attack detection under limited local information in cyber security: An ensemble multi-node multi-class classification approach," *ACM Trans. Sensor Netw.*, vol. 20, no. 2, pp. 1–27, Mar. 2024.
[36] C. Hazman, A. Guezzaz, S. Benkirane, and M. Azrour, "LIDS-SIoEL: Intrusion detection framework for IoT-based smart environments security using ensemble learning," *Cluster Comput.*, vol. 26, no. 6, pp. 4069–4083, Dec. 2023.
[37] E. Gyamfi and A. Jurcut, "Intrusion detection in Internet of Things systems: A review on design approaches leveraging multi-access edge computing, machine learning, and datasets," *Sensors*, vol. 22, no. 10, p. 3744, May 2022.
[38] E. Chatzoglou, G. Kambourakis, C. Smiliotopoulos, and C. Kolias, "Best of both worlds: Detecting application layer attacks through 802.11 and non-802.11 features," *Sensors*, vol. 22, no. 15, p. 5633, Jul. 2022.
[39] M. Douiba, S. Benkirane, A. Guezzaz, and M. Azrour, "An improved anomaly detection model for IoT security using decision tree and gradient boosting," *J. Supercomput.*, vol. 79, no. 3, pp. 3392–3411, Feb. 2023.

SAPNA SADHWANI received the B.E. and M.E. degrees in computer engineering from Mody University, India, in 2012 and 2014, respectively. She is currently pursuing the Doctor of Philosophy degree. From 2015 to 2016, she was a Lecturer with Mody University. She has been a Lecturer with the Birla Institute of Technology and Science Pilani, Dubai, United Arab Emirates, since 2018. Her research interests include cyber security, artificial intelligence, and blockchain.

URVI KAVAN MODI is currently pursuing the Bachelor of Engineering degree in computer science with the Birla Institute of Technology and Science Pilani, Dubai, United Arab Emirates. Her research interests include machine learning, network security, and IoPTS.

RAJA MUTHALAGU was a Postdoctoral Research Fellow with the Air Traffic Management Research Institute, Nanyang Technological University, Singapore, from 2014 to 2015. He is currently an Associate Professor with the Birla Institute of Technology and Science Pilani, Dubai Campus, Dubai, United Arab Emirates. He has published more than 55 research papers in reputed journals and conferences. His research mostly focuses on applying intelligent techniques for detecting and mitigating a security attack in the IoT, SDN, and other computer networks. His research interests include wireless communications, signal processing, aeronautical communications, and cyber security. He was a recipient of the Canadian Commonwealth Scholarship Award 2010 for the Graduate Student Exchange Program in the Department of Electrical and Computer Engineering, University of Saskatchewan, Saskatoon, SK, Canada.

PRANAV M. PAWAR received the degree in computer engineering from Dr. Babasaheb Ambedkar Technological University, Maharashtra, India, in 2005, the master's degree in computer engineering from Pune University, in 2007, and the Ph.D. degree in wireless communication from Aalborg University, Denmark, in 2016. He was an Associate Professor with MIT ADT University, Pune, India, from 2018 to 2019, and an Associate Professor with the Department of Information Technology, STES's Smt. Kashibai Navale College of Engineering, Pune, from 2008 to 2018. From 2006 to 2007, he was a System Executive with POS-IPC, Pune. He is currently an Assistant Professor with the Department of Computer Science, Birla Institute of Technology and Science (BITS) Pilani, Dubai. Before BITS Pilani, he was a Postdoctoral Fellow with Bar-Ilan University, Israel, from March 2019 to October 2020, in the area of wireless communication and deep learning. He is a IBM DB2 and IBM RAD certified professional and completed NPTEL certification in different subjects. He received recognition from Infosys Technologies Ltd. for his contribution to the Campus Connect Program and also received different funding for research and attending conferences at the international level. He published more than 40 papers at the national and international levels. His research interests include energy efficient MAC for WSN, QoS in WSN, wireless security, green technology, computer architecture, database management systems, and bio informatics. He was a recipient of an Outstanding Postdoctoral Fellowship from Israel Planning and Budgeting Committee. His Ph.D. thesis received a nomination for the Best Thesis Award from Aalborg University.