Internet of Things 24 (2023) 100936

Contents lists available at ScienceDirect

Internet of Things

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

Research article

Enhancing IoT network security through deep learning-powered Intrusion Detection System

Shahid Allah Bakhsh [a], Muhammad Almas Khan [b], Fawad Ahmed [a], Mohammed S. Alshehri [c], Hisham Ali [d], Jawad Ahmad [d,*]

[a] *Department of Cyber Security, Pakistan Navy Engineering College, National University of Sciences and Technology, Pakistan*
[b] *Department of Computer Science, FAST National University of Computer and Emerging Sciences, Islamabad, Pakistan*
[c] *Department of Computer Science, College of Computer Science and Information Systems, Najran University, Saudi Arabia*
[d] *School of Computing, Engineering and the Built Environment, Edinburgh Napier University, Edinburgh, UK*

ARTICLE INFO

Keywords:
Internet of Things
Intrusion detection
Cyber threats
Deep learning
Random Neural Network
Long Short Term Memory
Feed Forward Neural Network
Machine Learning
Network security

ABSTRACT

The rapid growth of the Internet of Things (IoT) has brought about a global concern for the security of interconnected devices and networks. This necessitates the use of efficient Intrusion Detection System (IDS) to mitigate cyber threats. Deep learning (DL) techniques provides a promising approach to effectively detect irregularities in network traffic, enhancing IoT network security and reducing cyber threats. In this paper, DL-based IDS is proposed using Feed Forward Neural Networks (FFNN), Long Short-Term Memory (LSTM), and Random Neural Networks (RandNN) to protect IoT networks from cyberattacks. Each DL model has its potential benefit as reported in this paper. For example, the FFNN can handle complex IoT network traffic patterns, while the LSTM is good in capturing long-term dependencies present in the network traffic. With its random connections and flexible dynamics, the RandNN model uses its data learning ability to adapt and learn from network data. These algorithms boost cybersecurity by enabling defense mechanisms against challenging cyber threats and ensuring the security of sensitive data as IoT networks expand. The proposed technique exhibits superior performance when compared with the current state-of-the-art DL-IDS using the CIC-IoT22 dataset. An accuracy of 99.93 % is achieved for the FFNN model, 99.85 % for the LSTM model, and 96.42 % for the RandNN model in detecting intrusion. Moreover, the models have the potential to enhance intrusion detection in IoT networks by generating swift responses to security problems in IoT networks.

1. Introduction

The advancements of the Internet of Things (IoT), cloud computing, computer security, and cyber security has undergone significant progress on a broad scope recently. IoT is a potential paradigm for societal innovation concerned with the Internet and real things, including smart home automation, business applications, smart cities, and environmental monitoring. IoT presents enhanced adaptability and productivity, thereby facilitating the establishment of extensively interconnected frameworks that enable novel services [1]. The advantages are attractive for both industrial and consumer applications. The emergence of the IoT paradigm has been noted to coincide with the development of tailored solutions within the past decades, thereby defining the concepts of Industrial IoT and Industry 4.0 [2]. According to projections, there will be nearly three times as many IoT devices in the world

* Corresponding author.
E-mail address: J.Ahmad@napier.ac.uk (J. Ahmad).

https://doi.org/10.1016/j.iot.2023.100936
Received 12 July 2023; Received in revised form 4 September 2023; Accepted 5 September 2023
Available online 13 September 2023

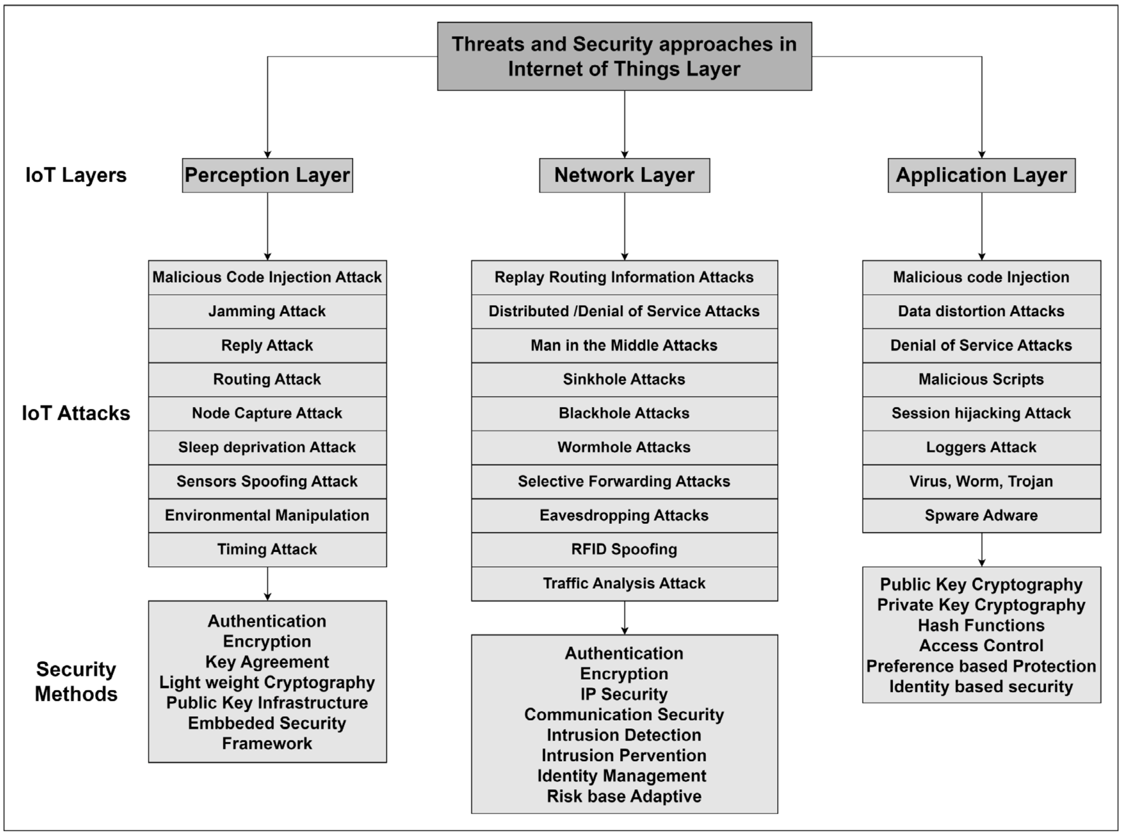


Fig. 1. Security threat and its mitigation in IoT layers [8,9].

in 2030 compared to the 15.14 billion that exist today. There are over 60% of IoT devices used in consumer markets and other corporate sectors. The percentage is expected to stay the same for the next ten years [3].

IoT networks are open and constantly changing topology by nodes joining and leaving the network in real-time. The lack of centralized network management tools makes them vulnerable to security threats. IoT devices have special characteristics, including tiny memory size, restricted data storage, limited power supply, and connection bandwidth [4]. The efficacy of security protocols for IoT infrastructures, in terms of growth and performance is significantly impacted by these limits. As a result, developing an effective intrusion detection system for an IoT network is challenging due to the increased overhead that requires computation power. Cyberattacks are getting more complicated and harder to identify as hackers employ cutting-edge techniques to steal sensitive data while evading detection by IDS. The communication between internetworks is also subject to cybersecurity risks. Because of this, innovative methods are crucial for timely intrusion detection and attack prevention measures. Recently, Machine Learning (ML) and Deep Learning (DL) algorithms have been used for network abnormality detection, intrusion detection, and prevention [5].

Integrating physical objects and sensors enable data exchange through the IoT paradigm. Utilizing technology components in IoT networks has enhanced efficacy in collecting, analyzing, reporting, and projecting data for future planning [6]. Multiple layers comprise an IoT architecture, which identifies, examines, and monitors system consistency. Application, network, and perception are the three levels that make up the basic configuration. The application layer delivers services and programs that are user-specific. The network layer addresses the issues related to inter devices dependability, capacity building of data, energy utilization, and most significantly security aspects while connecting IoT devices to other networks, machinery, and services. The perception layer gathers data from the environment via sensors, actuators, and computing hardware. Considering power conservation, security, and interoperability issues, the physical layer manages operations like signal processing, encryption, and data transfer [7]. Security solutions for each layer of the IoT are broken down and categorized in Fig. 1 with the security threats on each tier. Most IoT architectures includes a wide variety of information, enabling features, and technology expansion across three distinct tiers [8].

For data processing, the Internet connects the IoT network and enables people and organizations to do more with few resources, including time. Along with the advancement in IoT technology, there is also a proportional increase in attacks targeted on IoT systems [10]. IoT-based critical infrastructures are more susceptible to cyber vulnerabilities and are targeted by several attacks. The current state of cyber threats is characterized by growing complexity, persistence and intelligence owing to the rapid evolution of adversarial tactics. Improving the cybersecurity posture of crucial cyber infrastructures is a serious global concern. It is, therefore, crucial to identify cyber risks before implementing effective and robust cybersecurity remedies [9]. Fig. 2 depicts an overview of the

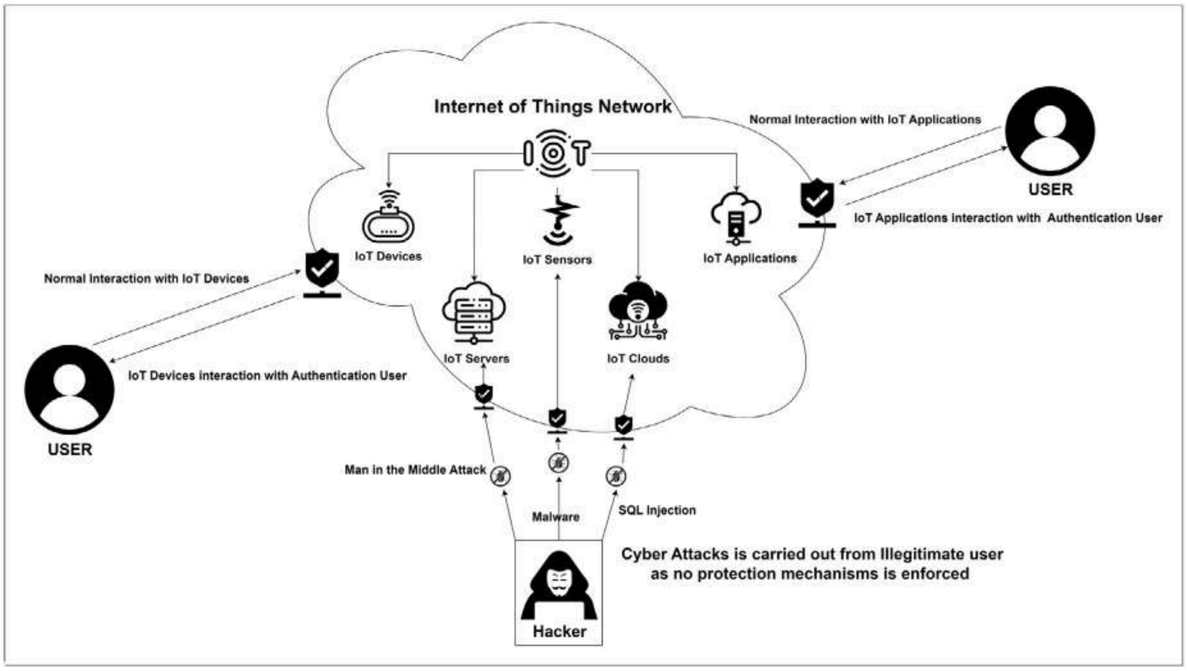

Internet of Things Network
IoT
IoT Devices
IoT Sensors
IoT Applications
IoT Servers
IoT Clouds
Normal Interaction with IoT Applications
IoT Applications interaction with Authentication User
USER
Normal Interaction with IoT Devices
IoT Devices interaction with Authentication User
USER
Man in the Middle Attack
Malware
SQL Injection
Cyber Attacks is carried out from Illegitimate user as no protection mechanisms is enforced
Hacker

Fig. 2. IoT network communicates with authorized and unauthorized users without any protection mechanism.

architecture of the IoT, which interacts openly with authorized and unauthorized users encompassing many components, including but not limited to devices, sensors, servers, actuators, protocols, cloud services, and applications within a network. Identifying the inherent characteristics of the content is challenging, whether normal or malicious, when a user engages with an IoT network. In an IoT network, unauthorized users may carry cyber attacks as no protection mechanisms are enforced.

Expanding network devices increases cyberattacks, making IoT security essential. With the growing use of IoT devices in several industries, including homes, securing the IoT infrastructure that strengthens devices is crucial [11]. IoT devices frequently exhibit constrained processing capabilities, limited memory capacity, and inadequate security features, making them vulnerable to malicious actors. Attackers can penetrate many vulnerable IoT devices, building botnets and interconnected networks of compromised devices under their command, flooding them with excessive network traffic and resulting in major delays. IoT devices with vulnerabilities can be manipulated to obtain unauthorized access [12]. Data breaches occur frequently in all IoT devices, which acquire and transfer sensitive data. It allows attackers to intercept data transfer, manipulate data, and inject malicious commands into the communication channel. The ability to remain undetected for prolonged durations provides a security risk [13]. The invasion of privacy can occur by exploiting IoT devices equipped with cameras or microphones, as the unauthorized transmission of sensitive information can occur. Therefore, strong authentication must be implemented to ensure the security of users and IoT devices in IoT network [14].

IoT intrusion refers to an illicit action or activity that harms the IoT ecosystem that undermines the confidentiality, integrity, or availability of information in any manner. Secure and encrypted communication channels can be established by employing VPNs, ensuring the confidentiality and integrity of the transmitted data [15]. An incursion refers to a situation where access to computer services is impeded by an attack, thereby preventing legitimate users from using such services. An Intrusion Detection System (IDS) is a technological solution, either using software, hardware or both, designed to oversee computer systems and detect malicious activity to maintain system security. Implementing Intrusion Detection and Prevention Systems (IDPS) to monitor network traffic and detect any abnormal activities that signify a security breach is essential [16]. The primary objective of IDS is to identify malicious network traffic and unauthorized computer usage, a task that a traditional firewall cannot accomplish. The two main subcategories of IDS are Signature-based Intrusion Detection Systems (SIDS) and Anomaly-based Intrusion Detection Systems (AIDS) [17]. Methods for intrusion detection are shown in Table 1.

To detect network intrusions, deep learning frameworks have become a popular field. While multiple surveys cover the developing research area on this topic, the literature requires an unbiased comparison of diverse deep learning models, especially in light of new datasets for intrusion detection [18]. In today's environment, cybersecurity is a crucial concern. For example, firewalls have been employed to protect sensitive data whereas IDS look for indications of malicious activity. Significant improvements in techniques such as pattern recognition and anomaly detection have been made due to the expansion of Artificial Intelligence (AI) research rapidly [19]. AI is a promising strategy for addressing cybersecurity threats and ensuring security [20]. It is suggested that implementing an Anomaly-based IDS will be a suitable measure to ensure the security of IoT networks, including servers, clouds, sensors, devices, and applications operating within the infrastructure illustrated in Fig. 3.

Table 1
Methods for IoT network intrusion detection techniques [8,9].

Detection method	Signature based intrusion detection system	Anomaly based intrusion detection system
Benefits	It uses attack signatures to identify known threats. These fingerprints may detect known risks in network traffic or system activity.	It may detect zero-day attacks. It establishes a baseline of normal activity and flags deviations, revealing unusual attack patterns.
	Attack signatures reduce false positives.	It may be used to find new assaults.
	It reacts quickly to known threats since it does not require processing or analysis.	It may detect new threats without signatures. Even if the attack is unknown, it senses abnormalities.
	It involves establishing the IDS with a signature database and monitoring network traffic or system logs.	IDS may detect insider threats when authorized users abuse their abilities or conduct crimes.
	Signature databases provide many signatures for known attack patterns and vulnerabilities, boosting SIDS.	It can learn and update its baseline to reflect system behavior or network dynamics, offering accurate detection.
Drawbacks	It only works against known threats with database signatures. It cannot identify zero-day attacks or new attack methods.	It may falsely notify of departures from normal behavior. False positives may tire security personnel.
	It cannot identify encrypted assaults, reducing its efficacy against such threats.	It may conflate benign and malignant abnormalities. Anomalies may indicate security or behavioral issues.
	If a new attack variation or updated signature is not in the database, resulting in a false negative limits signature detection.	Analyzing network traffic or system records for baseline, deviation may be computationally intensive and require particular hardware.
	It needs frequent signature database upgrades to work. New attack patterns need signature updates in the database.	It cannot process encrypted packets and the attack might go unnoticed and pose a threat.
	It has scalability issues as network traffic increases.	Creating a standard profile for a highly dynamic computer system is difficult.

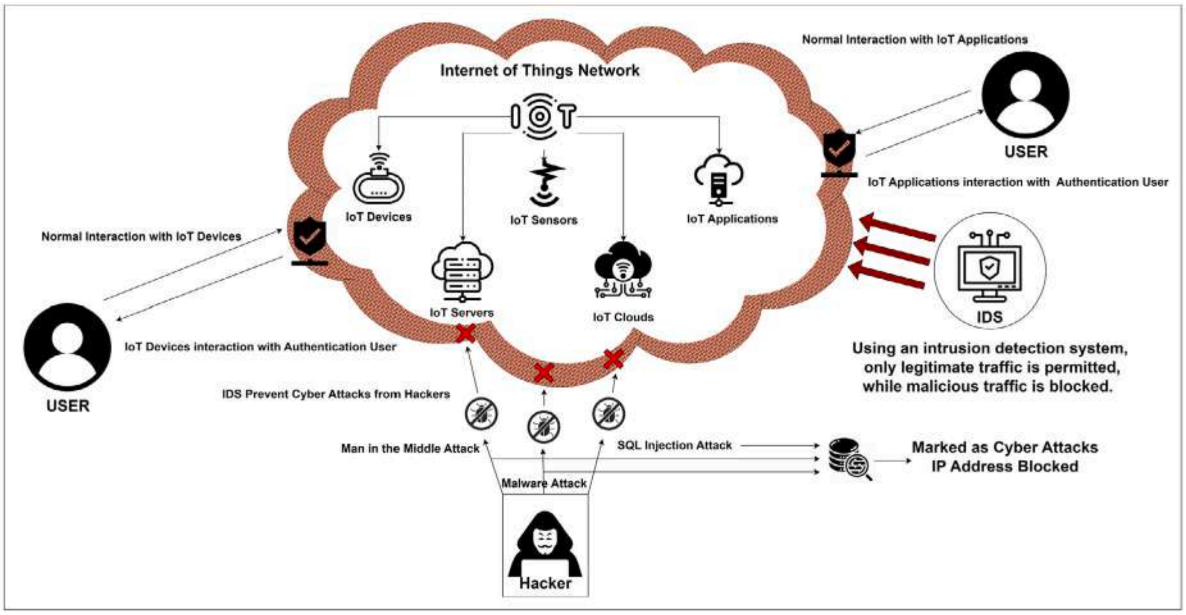

Internet of Things Network
Normal Interaction with IoT Applications
USER
IoT Applications interaction with Authentication User
IoT Devices
IoT Sensors
IoT Applications
Normal Interaction with IoT Devices
IoT Servers
IoT Clouds
IDS
IoT Devices interaction with Authentication User
USER
Using an intrusion detection system, only legitimate traffic is permitted, while malicious traffic is blocked.
IDS Prevent Cyber Attacks from Hackers
Man in the Middle Attack
SQL Injection Attack
Marked as Cyber Attacks IP Address Blocked
Malware Attack
Hacker

Fig. 3. Protecting the IoT network with an intrusion detection system.

In view of the above discussion, researchers have modified neural network topologies to enhance IDS. Deep Neural Networks (DNN) use computational resources to evaluate large amounts of data, find patterns and correlations, and classify the data according to specified criteria. The usage of DNN in IoT security is a potential method for anomaly-based intrusion detection to identify and classify data in a properly trained DNN. In this research, Intrusion Detection System is proposed for the Internet of Things (IoT), employing Deep Neural Networks for real-time data anomaly. This research aims to develop an Intrusion Detection System (IDS) for the Internet of Things (IoT) by utilizing Feed Forward Neural Networks (FFNN), Long Short-Term Memory (LSTM) and Random Neural Networks (RandNN) models. The research introduces a framework for deploying a Deep Learning technique for an Intrusion Detection System (DL-IDS) within the Internet of Things (IoT) networks to integrate cyber security measures and achieve effective intrusion detection performance. The present study involves developing an Intrusion Detection System with a multi-layered Neural

Network to identify distinct categories of attacks commonly observed in the IoT. These include Denial of Service (DoS), Bruteforce, HTTP Flood, UDP Flood, and TCP Flood. Following are the main contribution of the study:

- This work presents a new approach, namely the FFNN, LSTM and RandNN to detect and classify intrusions in IoT networks.
- The efficacy of the proposed scheme is assessed through extensive experimentation on the CIC IoT 2022 dataset, which belongs to the latest generation of IoT datasets.
- Design an Intrusion Detection System framework and identifying the dataset characteristics for IoT networks.
- The RandNN, FFNN, and LSTM efficacy is assessed in binary and multiclass using evaluation metrics.
- This work provides a comprehensive evaluation of the proposed approach compared to established Machine Learning (ML) and Deep Learning (DL) algorithms.

The subsequent sections of this article are structured in the following manner. Section 2 discusses a literature survey to provide an overview of IoT security challenges and Deep Learning applications in security. Section 3 encompasses the research methodology, which provides a framework for intrusion detection in IoT networks using Deep Learning. It also delineates the implementation platform, dataset description, simulations and discourse of outcomes. Section 4 includes a complete explanation of the proposed Intrusion detection models and an in-depth analysis of mathematical modeling and architectures. For comparison, the theoretical concepts of machine learning are also discussed. Section 5 provides evaluation metrics and experimental setup to analyze binary and multiclass classification results for DL-based IDS. Moreover, a comparative analysis of proposed DL models is provided with traditional ML models and state-of-the-art IDS, emphasizing its applicability and usefulness for IoT systems to identify cyber threats. The paper concludes in Section 6.

2. Literature survey

This section presents an in-depth review of the importance of cyber security in IoT infrastructure by reviewing previous studies and exploring the progress made by applying ML and DL techniques. This highlight the importance of IoT security and its challenges faced due to absence of IDS and emphasis on the need of IDS in IoT networks. The section examines existing research on deep learning-based IDS for IoT applications, focusing on identifying research gap in this area. Many research challenges must be updated to stop cyber attacks and make security solutions for light weight IoT networks. Applications like automobile, health monitoring, autonomous machinery, and smart homes generate a lot of data that requires new security measures. There needs to be more work done on intrusion detection for IoT, even though several academic and industrial researchers have utilized ML and DL techniques to build and deploy Intrusion Detection Systems (IDS) for IoTs during the past years.

2.1. IoT security challenges

IoT aims to improve the quality of life by providing a scope of smart, interconnected devices and applications across multiple domains. The critical challenges faced by IoT devices related to security threats lead to exploring advanced techniques such as DL to increase IoT security. A study in [21] introduced a Convolutional Neural Network (CNN) to enhance the efficiency and security of IoT networks by implementing an anomaly-based IDS. The model can identify intrusion and detect anomalies in traffic using the NID and BoT-IoT datasets, resulting in an accuracy of 99.51% and 92.85%, respectively. Even with the current progress in the IoT industry, a critical requirement exists for extensive research to enhance the threat detection rate of IoT systems. Thus, identifying anomalies in IoT devices is crucial in mitigating attacks and improving security measures.

Researchers faced challenges in detecting malicious attacks against IoT devices due to insufficient feature extraction accuracy from raw network traffic data. To tackle this issue, [22] employed a CNN for automated feature extraction and introduced the IoTFECNN (IoT Features Extraction Convolutional Neural Network). An optimized version of the Capuchin Search Algorithm (CSA) was devised to perform binary multi-objective feature selection with enhanced efficiency. The hybrid methodology known as CNN-BMECapSA-RF was evaluated on TON-IoT and NSL-KDD datasets. The evaluated approach has demonstrated an accuracy of 99.99% and 99.85% detecting 27% and 44% of the features characteristics. Furthermore, it was compared with newer methods and proved to perform better due to its robust classification, effective feature selection, and feature extraction. It evaluates various metrics, including energy consumption, computing overhead costs, and detection delays, within a semi-realistic scenario but does not include anomaly and attack-type detection in IoT.

In recent times, there has been a significant increase in the amount of data transmitted through communication infrastructures due to advancements in technology. Attackers have increased efforts to make networks vulnerable. In [23], an IDS framework is implemented with several Recurrent Neural Networks (RNNs) approaches as LSTM, Simple RNN, and GRU RNNs with an XGBoost feature selection for NSL KDD and UNSW NB15 datasets. The findings indicate that the XGBoost LSTM model shows higher performance, achieving a (Traffic Acquisition Costs) TAC of 88.13%, and XGBoost GRU has a TAC of 86.93% when applied to the NSL KDD dataset for binary and multiclass classification. Moreover, the study showed that the XGBoost Simple RNN achieved a shorter training time. Therefore, the Simple-RNN proved to be the best option with minimal resources.

The optimization of IoT network security requires a focus on lightweight, efficient, and flexible solutions. In [24] a lightweight Random Neural Network is proposed to detect intrusions in the IoT. This methodology is suitable for IoT networks with limited resources due to its improved flexibility and distributed framework. The framework's performance was analyzed on individual classes within the datasets however the enhancement of model performance on minority classes was not addressed. The efficacy of the proposed DnRaNN was evaluated through various metrics, which resulted in attack detection with an accuracy of 99.14%

for binary and an accuracy of 99.05% for multiclass. This model was tested on the ToN_IoT dataset only for IoT security. A field programmable gate array (FPGA)-based accelerator with DnRaNN was used to optimize the effectiveness of the cyberattack detection algorithm in hardware.

2.2. Intrusion detection systems in IoT

The IoT has a lot of potential across multiple sectors. However, techniques for detecting and avoiding security incidents must be developed to implement IoT in various businesses effectively. Several researchers use datasets produced by different organizations to support the development of ML models to predict intrusions. Due to this, not all classes of datasets have the same sample sizes, which is a common issue with datasets. Techniques like random undersampling and oversampling could be more effective in producing reliable results. Researchers in [25] tackle the problem of data imbalance in intrusion detection within the IoT by implementing a specialized loss function. The purpose of this function is to automatically focus on more complex negative cases over simple ones to lower the assigned weights. The effective training of ML models is facilitated by enabling changes in dynamically scaled gradients. This research demonstrates the effectiveness of a DL-based intrusion detection approach utilizing focal loss to analyze three datasets derived from three distinct IoT domains. The FNN Focal technique performed better than the baseline model across the Bot-IoT dataset, achieving 91.55% accuracy. Whereas CNN Focal approach achieved a 24% increase in accuracy compared to the cross-entropy loss function. The scope is however limited to the Focal loss, thereby highlighting the necessity for developing efficient intrusion detection systems.

Using imbalance datasets in intrusion detection poses a significant challenge in developing effective and reliable intrusion detection and classification systems. An ensemble learning methodology has been suggested as a potential solution to solve the class imbalance problem in intrusion detection datasets that employ a Deep Neural Network (DNN)-based IDS [26]. The objective was to address the problem of imbalanced class distribution in intrusion detection datasets while simultaneously obtaining high adaptability. A bagging classifier with a DNN base estimator used class weights to equalize the Deep Neural Network training subsets. This technique is evaluated using four distinct intrusion detection datasets NSL-KDD, UNSW-NB-15, CIC-IDS-2017, and BoT-IoT. This process is statistically analyzed using the Wilcoxon signed-rank test and achieved an accuracy of 98.99% with the BoT-IoT dataset. This research did not use bagging-based ensemble learning to derive an ideal value for the base estimator for improving the efficiency of IDS.

The vulnerability of IoT networks to unauthorized access and manipulation has increased. Hence, establishing IDS is essential in safeguarding IIoT networks. In [27], three deep learning models, namely CNN, LSTM, and a combination of CNN and LSTM, were introduced to detect security vulnerabilities in Industrial Internet of Things (IIoT) networks. The hybrid CNN+LSTM model demonstrated high performance on UNSW-NB15 and X-IIoTID datasets for intrusion detection. The CNN + LSTM architecture achieved an accuracy of 93.21% for binary and 92.91% for multiclass on the UNSW-NB15 dataset. In contrast, the accuracy was 99.84% for binary and 99.80% for multiclass classification on the X-IIoTID dataset. The efficacy of the proposed framework may be augmented through synthetic data production to retrain networks, thereby increasing the accuracy of relevant models for diverse forms of attacks present within the datasets. Researchers have proposed integrating smart frameworks and innovative IDS with AI to address privacy and security issues. The study in [20] analyzed the efficacy of ML and DL methods for protecting the IoT. SVM and RF have been used due to their accuracy in detecting patterns. The techniques, including RNN and extreme gradient boosting (XGBoost), also performed better. The authors in [10] aims to develop an intrusion detection framework for machine learning (IDFML) for IoT networks. This framework utilizes supervised learning techniques. The IDFML has successfully attained a 98.68% accuracy in detecting attacks using a Random Forest classifier.

2.3. Deep learning applications in security

The IoT network comprises several IoT nodes that transmit and receive large amounts of data, thus facilitating advanced intrusion detection techniques. The Attacks in IoTs go undetected for a long, resulting in service disruptions, financial loss, and identity theft. For IoT-enabled services to be reliable, secure, and profitable, IoT devices must be equipped with real-time intrusion detection. A fully connected four-layer network architecture (FCFFN) is presented in [18] to identify malicious traffic that targets connected IoT devices. In the FCFFN architecture, a deep neural network consisting of six layers are fine-tuned to detect intrusion accurately. The proposed deep learning-based IDS quickly identifies attacks on IoT networks like Blackhole, Distributed Denial of Service, Opportunistic Service, Sinkhole, and Workhole. The results show that the system is robust and can automatically identify attacks, with a detection rate of 93.21%, thus improving the safety of IoT networks. This IDS was however been trained using the dataset generated by the experimental system and includes only five types of intrusion.

Many traffic analyses utilize deep learning techniques to classify single flows, which can lead to misclassifying irrelevant flows. Therefore, the utilization of flow sequences is essential for the purpose to carry out traffic analysis. A novel flow-based traffic classifier is presented using the flow transformer technique for anonymity, encrypted and attack traffic for IIoT [28]. The model employs the multi-head attention mechanism to discuss the importance of the flow sequence and uses a feature extraction layer to fully capture the flow sequence's characteristics, achieving an accuracy of 98.5% using the CIC-IoT2022 dataset. The RF-based feature selection method is developed to identify the ideal combination of features, preventing irrelevant features. It aims to select optimal features to mitigate any potential negative impact on classifier performance.

IoT networks enable massive data flows between many devices. Attackers can now compromise the CIA traits of IoT devices and networks. IoT devices generates vast amounts of data that are difficult to analyze in real time, thus making IDS a challenge to build.

A Deep Ensemble-based IDS was presented in [29] using a multi-classifier system known as Lambda architecture that identifies cyber-attacks using LSTM, CNN, and ANN classifiers to maximize batch layer training. The batch layer trains the model, whereas the speed layer of the Lambda architecture makes decisions to improves the ability to make real-time evaluations. LSTM exceeds ANN and CNN-based classifiers in accuracy for Binary Classification. Hybrid Ensemble achieves 99.93% accuracy outperforming individual classifiers in multiclass by reducing processing time. The results indicate that the ensemble methodology shows higher detection accuracy than the direct approach.

Data collection from diverse IoT sensors and the delays faced within the IoT system pose a more complex challenge in identifying abnormal behavior and compromised nodes than in conventional Information Technology (IT) networks. In [30] an approach for detecting anomalies by deep learning by employing a denoising autoencoder to obtain characteristics suitable for the large IoT environment using the DS2OS traffic traces dataset proposed. The classifier utilized the features to differentiate between vulnerable and authentic IoT data. Using denoising autoencoders enables the efficient extraction of features that exhibit strength against noisy characteristics in IoT systems. The study demonstrated that integrating deep learning-based anomaly detection in IoT effectively identifies and mitigates fake data. IoT growth has increased cyber-attack risk, making intrusion detection on IoT networks more plausible due to the potential for unwanted access to many interconnected services. An approach [31] has introduced MFO-RELM, which combines the Mayfly optimization (MFO) with regularized extreme learning machine (RELM) to address the issue of cybersecurity threat identification and categorization in IoT environment. The MFO-RELM methodology demonstrates high efficacy in detecting and assessing cybersecurity vulnerabilities. The MFO-RELM model was applied to the N-BaIoT dataset for testing purposes can identify cybersecurity threats within the Internet of Things (IoT) ecosystem.

2.4. Related work on deep learning-based IDS for IoT

The growing of IoT devices and malicious software, as well as the wide use of encryption technology in IoT communication, has increased the amount of encrypted suspicious traffic between devices, posing a significant threat to IoT cybersecurity. Currently, many traffic detection techniques employed in the IoT need more dataset processing, suboptimal feature extraction, imbalanced data, and limited accuracy in multiclass classification. The vulnerability of IoT networks to threats was addressed by developing a bidirectional CNN-BiLSTM DDoS detection model, as proposed in [32]. The model was evaluated using the CICIDS2017 dataset and included three deep learning algorithms: RNN, LSTM-RNN, and CNN. This research builds a model responsive to different DDoS attacks and can distinguish between malicious and normal traffic to prevent false alarms. The CNN-BiLSTM model has an accuracy of 99.76%. This model examined the architecture of IoT networks and provided possible causes for their security flaws. The model demonstrated its compatibility with the existing IoT network infrastructure. The lack of a genuine testing platform was the primary limitation of this study, which limited the testing's reliability.

The interconnected nature of devices and sensors in the IIoT leads to massive amounts of dynamic and unreliable data. Analyzing and protecting Big data volumes is challenging. The IIoT will increase the vulnerability of industrial systems to cyber attacks, making cyber threat detection more difficult. Anomaly detection has decreased due to cyberattack diversity. According to [33], an ensemble deep LSTM-AE anomaly detection model is utilized to detect irregularities in the IIoT to detect cyber threat. The model is trained on balanced data obtained from unbalanced datasets on two real IIoT datasets. This model outperformed existing ML classifiers and achieved 99.7% accuracy. To safeguard IoT networks from diverse attacks, a study in [34] introduced an innovative IDS that utilizes a filter-based Deep Neural Network (DNN) model. Using Generative Adversarial Networks (GAN), they addressed the issue of class imbalance in the dataset by increasing the number of packets in minority class attack categories in IoT networks. The classifier achieved an accuracy of 85.0% in multiclass for the UNSW-NB15 dataset. However, when using GANs, the classifier achieved a higher accuracy rate of 91.0%. The attained outcomes hold considerable importance within the field of cybersecurity. It can improve accuracy further and minimize errors in false negatives and false positives.

IoT has extensively been employed in automated network systems, significantly influencing various industries recently. IoT comprises many nodes that acquire, retain, and analyze personal data, making them highly vulnerable to attacks from malicious entities. IoT requires assistance to enhance operational efficiency and distinguish subcategories of cyber attacks. The research in [35] proposes a Deep Convolutional Neural Network (DCNN) model that comprises of a convolutional layer, pooling layer, and a fully connected layer designed to detect malicious attacks in IoT networks. The model aims to enhance performance and minimize computational requirements, which can benefit low-power IoT devices. Using the IoTID20 dataset to conduct performance analysis for multiclass subcategories, an accuracy of 77.55% was achieved. Adam, AdaMax, and Nadam optimization techniques were applied in which Nadam optimizer performance was best with all batch sizes. Experiment results show that the proposed method outperformed among existing DL-based algorithms and is more stable.

Any data storage or processing model faces the same problem of fog and cloud security. Once an attack has occurred, it will have destructive impacts on the expansion of the IoT, the Fog, and the cloud. As a result, numerous Fog data protection models have been developed. An intrusion detection system for Fog computing security is presented in [36], which is fully automated using AI to mitigate cyber-attacks. The model was examined using multi-layered Recurrent Neural Networks (RNN) designed to implement Fog computing security near end-users and IoT devices. The tested model has a 98.27% accuracy in detecting Denial-of-Service (DoS) attacks while maintaining a processing overhead of the NSL-KDD dataset.

The increasing number of novel cyber-attacks poses a continual threat to modern IoT networks, requiring the development of advanced defense mechanisms to mitigate zero-day attacks effectively. The limitations of IoT devices are computing power and lack of endurance. Moreover, the diverse cyber-attacks poses a detection challenge that requires continuous defense mechanisms. Researchers has proposed dynamic and distributed frameworks based on DL to detect cyberattacks [37]. The NSL-KDD and BoT-IoT datasets evaluate the performance of the LSTM and FFNN models within the framework. The framework was devised to safeguard IoT networks by utilizing fog layers to detect cyber-attacks. In order to boost detection and classification accuracy while reducing latency, fog layers evaluate data near the edge layer. The proposed distributed system detects cyberattacks by achieving an accuracy of 99.95%. The results indicate that FFNN exhibits a higher performance for the distributed NSL-KDD dataset. The proposed methodology has improved cyber-attack detection in IoT networks that utilize constrained end devices. This research is limited to sufficient data when the number of fog nodes in the framework grows.

2.5. Problem formulation

In future, advancements in anomaly detection within the system will enhance data processing and analyze security, thereby mitigating the risk of major failures in IoT networks. From the above review, it is clear that numerous methods have been proposed for enhancing the security of IoT devices. Multiple research has investigated IoT security and privacy concerns in cybersecurity. Complex models require tremendous resources, whereas lighter models are better suited for deployment on edge devices [20]. Current research has shown that DL needs to be enhanced because it produces misleading results that lead to inefficient procedures and overfits. Based on the above discussion, implementing advanced learning strategies can improve the performance of IDS [38]. Implementing security protocols for both the internal and external components of IoT devices is essential due to the importance of IoT networks. Attackers may launch destructive impacts on IoT-based critical applications. Implementing deep learning models for a better understanding of the decision-making process increases trust in the IDS and assists researchers in understanding and evaluating detections.

In recent years, academic and industrial researchers have used ML and DL techniques to develop and implement IDS for IoT devices. Despite a lot of work done, it is critical to continue developing new and efficient DL algorithms. Therefore, improving security by creating adaptable intrusion detection systems is a challenge. Deep learning-based IDS is a potential solution for detecting patterns of cyber attacks in large volumes of data in IoT networks. This research gap will help build robust DL-based IDS for IoT to detect intrusions. The research aims to develop three DL neural networks capable of detecting intrusion in IoT networks and providing better protection against cyber threats with reduced complexity and classification time in order to enhance accuracy and effectiveness. Therefore, improving IDS in the IoT will improve reliability, efficiency, and credibility. However, additional research is needed to improve intrusion detection efficiently. The motivation for performing this research is to counter cyber threats and overcome IDS limitations by implementing robust security measures within IoT networks.

3. Proposed intrusion detection systems

This section presents a framework for deep learning-based intrusion detection to identify and classify potential vulnerabilities and attacks within IoT networks. The proposed framework is presented for three DL models, followed by a comprehensive evaluation of the tasks performed at each stage and a thorough analysis of the significant contributions of each step. The subsequent discussion comprehensively explains the intrusion detection dataset, hardware, software and libraries employed in this research.

3.1. Proposed framework

The proposed system has been designed to retrieve packets from the data produced by the underlying IoT infrastructure and identify intrusions following sufficient training. The proposed framework entails employing a traffic capture mechanism from an IoT Network, as proposed in [39], to collect the traffic flow of IoT devices. This data is then utilized to emulate the operations of an IoT-enabled Smart Home. The proposed framework involves utilizing a deep neural network trained on a dataset, CIC IoT 2022, consisting of network packets. The CICFlowMeter 4.0 is utilized as a feature extractor to extract relevant features from the packets transmitted through IoT networks [40]. The proposed framework illustrated in Figs. 4 and 5 is a newly developed Intrusion Detection System that employs deep learning techniques to identify anomalies in the IoT networks. The training workflow of the proposed framework depicted in Fig. 4 comprises of five stages: feature extraction, data preprocessing, data balancing, feature selection, and data splitting. The application of a model for training follows these stages. Fig. 5 illustrates integrating Intrusion Detection Systems (IDS) with IoT Networks to identify and predict intrusions before the IoT network. After assessing the effectiveness of various models, the optimal model will be selected for deployment to detect anomalies within IoT network.

Anomaly detection encompasses a range of techniques, such as packet dropping, vulnerability scanning, IP address blocking, physical inspections, and user notification. The model under consideration streamlines the process of choosing algorithms for limited resources of IoT devices by considering actual performance and cost metrics, including accuracy and time cost. Tailored solutions address the distinct requirements of individual users are of paramount importance within the framework of anomaly detection systems for the IoT security. The dataset is subjected to training through various DL models to identify anomalies. Thus, it can be deduced from the results of the model that has been trained. The system can run an algorithm to identify the existence of any malicious packets and conduct a thorough examination of the IoT devices. Hence, the proposed model aims to provide an best solution catering diverse user types, including large firms with abundant resources seeking maximum accuracy and startups prioritizing cost efficiency.

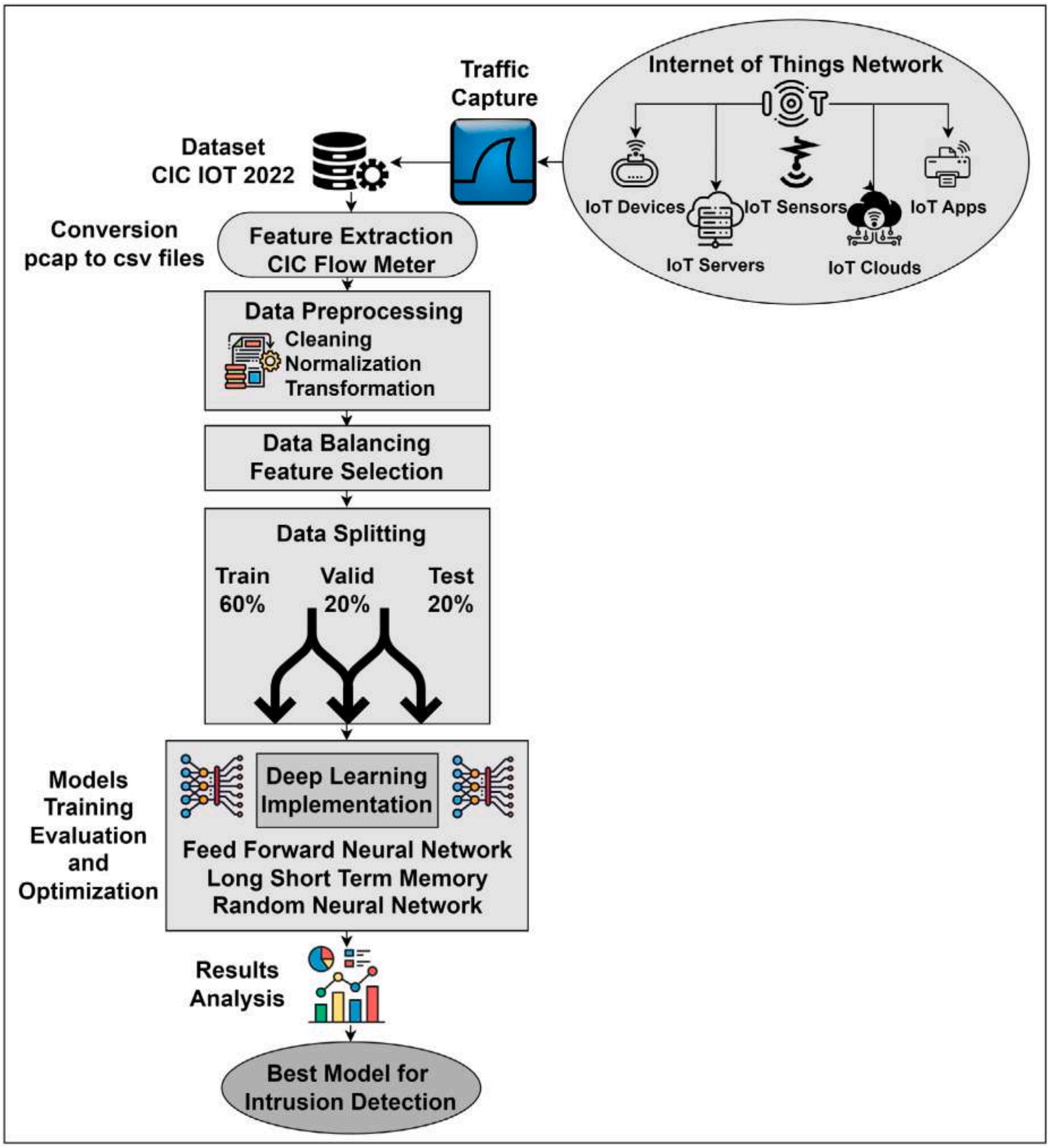


Fig. 4. Workflow of the proposed intrusion detection system for IoT network using DL.

3.2. Workflow description of the proposed framework

The process consists of the following stages including data preprocessing which have importance in data analysis, as it involves cleaning, encoding, feature scaling and extracting the dataset. Data augmentation techniques is used to achieve class balancing and feature selection to identify the most appropriate features. The feature preprocessing encompasses the initial step of dividing the dataset, which is proceeded by the training, validation and testing of DL models.

Data Preprocessing: Initially, the network analyzer tool is employed to gather raw network traffic of IoT devices and extract packet features [39]. Features extraction can be done using CIC Flow Meter 4.0 [40] and convert PCAP (Packet Capture) files into CSV (comma-separated values) files. The dataset comprises 84 features, including 78 numerical values, five categorical values, and one label. The dataset features are subject to cleaning, feature scaling and encoding techniques before given to the DL models.

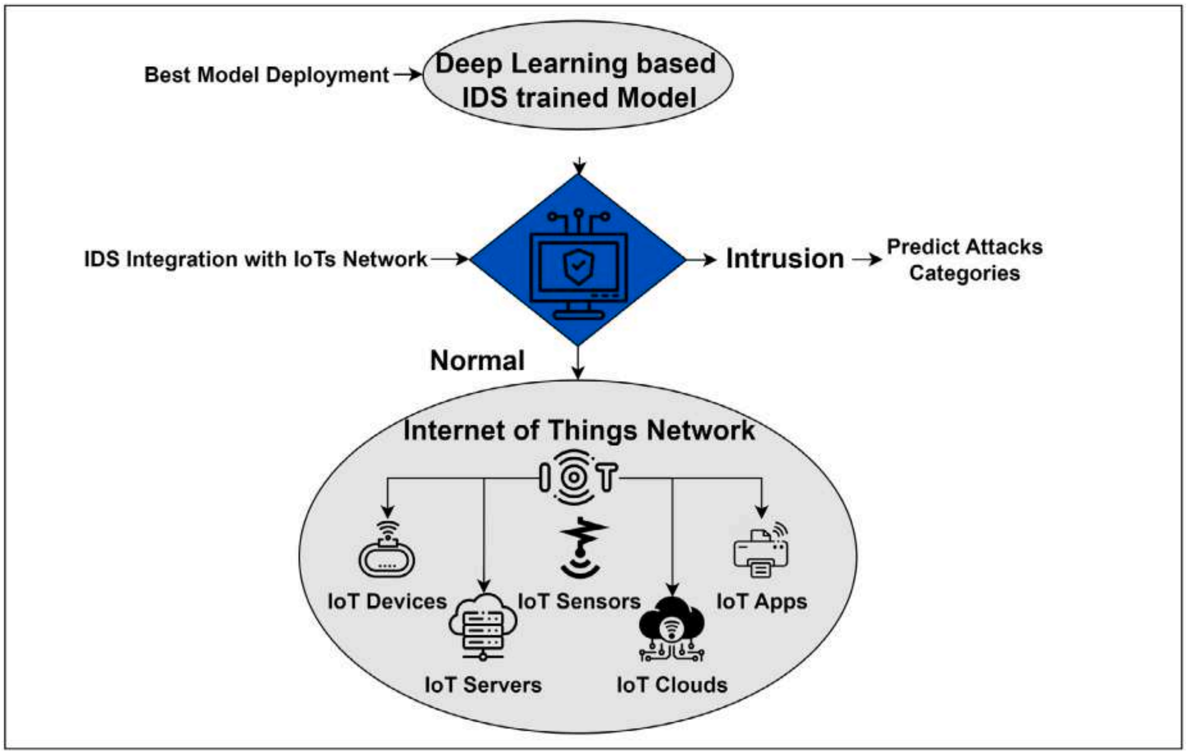

Best Model Deployment
Deep Learning based IDS trained Model
IDS Integration with IoTs Network
Intrusion
Predict Attacks Categories
Normal
Internet of Things Network
IoT Devices
IoT Sensors
IoT Apps
IoT Servers
IoT Clouds

Fig. 5. Workflow of the proposed intrusion detection system for inference in IoT network using DL.

3.2.1. *Data cleaning*

Data cleaning involves the process of identifying and addressing errors, inadequate structure, duplication, or missing values within the dataset. There are various ways through which data duplication or incorrect classification might occur during the integration of different data sources [41]. The lack of a widely accepted methodology for defining each stage of the data cleaning process may lead to anomalies in datasets. The reliability of results and algorithms can be compromised by the presence of inaccurate data, despite achieving accuracy. Establishing a standardized framework for data cleaning process is imperative to ensure accuracy and consistency across all iterations. The CIC IOT 2022 dataset used in this research is cleaned by dropping redundant packets and removing missing and infinite values.

3.2.2. *Categorical encoding*

Converting categorical data into numerical data is commonly called label encoding, a widely used technique in DL. It involves assigning a numerical value to each distinct categorical variable for efficient data processing by algorithms. For example, a dataset absolute features values derived from the set are ‘HTTPFlood’, ‘UDPFlood’, ‘TCPFlood’. This step utilize label encoding to assign numerical values to the features correspondingly. Another method for transforming categorical data into numerical data is One-Hot encoding that is used to represent categorical variables as binary vectors [42]. As mentioned above, this approach generates a binary vector for every category within the variable. The vector comprises zeros for all values except for the index value assigned as one for corresponding class. It allows categorical variables to be used in mathematical algorithms that require numerical input.

- Consider a discrete categorical random variable, denoted as x, with n distinct values $x_1, x_2, \dots, x_n$. The One-Hot encoding process involves a specific value x_i as a vector v, where each element of v is zero except for the ith element, which is assigned a value of 1.

$$v_{ij} = \begin{cases} 1 & \text{if } x_i = x_j, \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

- Let x be a random variable defined over a set $S = \{a, b, c, d, e, f, g, h\}$. We can denote the variables x_1, x_2, x_3, x_4, x_5, x_6, x_7, and x_8 as a, b, c, d, e, f, g, and h, respectively. The One-Hot encoding representation for x comprises of eight binary vectors:

$$\upsilon = \left\{\begin{matrix} (1,0,0,0,0,0,0,0), \\ (0,1,0,0,0,0,0,0), \\ (0,0,1,0,0,0,0,0), \\ (0,0,0,1,0,0,0,0), \\ (0,0,0,0,1,0,0,0), \\ (0,0,0,0,0,1,0,0), \\ (0,0,0,0,0,0,1,0), \\ (0,0,0,0,0,0,0,1). \end{matrix}\right\} \tag{2}$$

Label encoding and One-Hot encoding techniques are used to convert categorical attributes into numeric values. The features namely 'Flow ID', 'Src IP', 'Dst IP', 'Time Stamp', and 'FIN Flag Cnt' are subjected to label encoding which result into integer values of 202163, 876, 2138, 85714 and 8, respectively. The One-Hot encoding approach is employed in neural networks to handle categorical variables efficiently. This implementation is determined by the number of unique levels observed in the variables. In this work, the column 'Label' is transformed into eight columns of normal and attack categories and eliminate label column from the dataset, as shown in Eq. (2).

3.2.3. Feature scaling

Feature scaling is a technique that involves modifying the range of values within a set of features according to a specified range. This method must be used when a feature has a big value but does not affect any other features [41]. This includes standardization and normalization [43]. Normalization is a necessary step to mitigate the issue of significant variances during the merging process. It is achieved by scaling all features to a standard range, thereby enabling optimization. Min-Max normalization is used with a selected range of [0,1] to normalize the values of columns, thereby enabling them to be represented on a uniform scale.

$$x_{norm} = \frac{x - \min(x)}{\max(x) - \min(x)} \times (b - a) + a, \tag{3}$$

where,

x is the original value to be normalized.
$min(x)$ is the smallest column feature value.
$max(x)$ is the largest column feature value.
x_{norm} is the normalized value.
a is the lower bound of desired normalization.
b is the upper bound of desired normalization.

Data Augmentation and Balancing: The problem of imbalanced data occurs when the distribution of samples across different classes is uneven. It leads to a skewed dataset, where one class is imbalanced and may result in a biased model. The training data generally undergoes re-sampling before classification, which augments the number of samples belonging to the minority class, and under-sampling, which reduces the number of samples from the majority class. Imbalanced classification issues are frequently encountered in various datasets because the minority class usually has insufficient data. One approach to rectify class imbalance is the synthesis of additional data from the minority class, thus mitigating the issue of limited data availability.

Synthetic Minority Oversampling Technique (SMOTE) is a widely used approach for generating additional samples in the dataset. The methodology relies on developing data packets that connect a given point with its K-nearest neighboring points [44]. The SMOTE technique generates new samples from the existing dataset to attain a balanced dataset. It is employed to increase the number of instances of the minority categories. Furthermore, an analysis is conducted to evaluate the impact of various factors, including dimensionality reduction, size of the training set, and the number of neighbors (K) on the accuracy of the system. Additionally, a qualitative analysis assesses the variables that affect the results. Following are the steps used to perform SMOTE analysis in this research.

- The techniques involve identifying the samples that pertain to the minority class within the imbalanced dataset. The term "minority class" denotes the category with fewer instances than the "majority class".
- The SMOTE algorithm selects a sample from the minority class within the dataset. The chosen sample is denoted as X.
- Upon obtaining the designated sample X, it is imperative to identify its k nearest neighbors originating from the minority class. The parameter k holds a significant value in determining the count of neighbors to be considered. The set of k nearest neighbors is denoted as NN.
- Synthetic samples NN are produced for each neighbor n by interpolating between the minority sample X, and its neighbor n. A point is randomly selected on the line segment that links X and n to create an artificial specimen. The α parameter, known as the synthetic sample ratio, governs a numerical parameter that takes values within the interval $[0, 1]$. The value of 0 corresponds to the variable X, while 1 corresponds to the variable n.
- The generation of synthetic sample is given by Eq. (4).

$$s = X + \alpha(n - X) \tag{4}$$

- Interpolation is performed on a per-feature basis for every attribute of both X and n.

- The oversampling of a minority sample involves iteratively selecting a model, identifying its closest neighbors, and generating synthetic illustrations. This process is repeated until the dataset is balanced.

Given the imbalanced nature of the dataset, a resampling technique is employed before classification to mitigate the potential issues arising from class imbalance. The dataset is balanced by using the undersampling of the majority class and oversampling of the minority class. The dataset underwent SMOTE analysis, producing resampled data for the X and y variables. The experiment was limited to eight categories, including three Normal classes (Power, Idle, and Interaction) and five Benign classes (HTTP Flood, UDP Flood, TCP Flood, RSTP, and Brute Force). The dataset yielded 476111 packets that underwent SMOTE analysis, encompassing binary and multiclass categories.

Feature Selection: Feature selection is a technique that is used to retain features from data by reducing its dimensionality. It helps to enhance storage efficiency and decrease computational complexity. Due to power limitation of IoT devices, process of designing Intrusion Detection Systems (IDS) classifiers require low-dimensional features in both binary and multiclass classification. Empirical data has demonstrated that this approach has yielded exceptional outcomes. In this research, Principal Component Analysis (PCA) is used to convert the initial dataset into a lower-dimensional subspace while preserving the fundamental characteristics of the original data by selecting and extracting relevant features.

Principal Component Analysis (PCA) is a statistical technique that aims to reduce the number of features in a dataset by using orthogonal combinations of the original parameters that shows high variance [45]. The principal components are independent of one another, eliminating correlated features that contribute insignificantly to decision-making process. The main steps in PCA are computing the mean, standard deviation, covariance, cumulative proportion, eigen-vectors, and eigen-values [46]. The selection of these combinations is based on the significance of the dataset. The principal components is obtained as a linear combinations of the original variables which can capture the highest degree of variance in the dataset. Following are the main steps of PCA:

- Consider the matrix X, which has dimensions $m \times n$ and represents our dataset. Each row of X corresponds to a distinct data point, while each column corresponds to a unique feature. The computation of the mean vector, represented by μ, is shown by Eq. (5).

$$\mu = \frac{1}{m} \sum_{i=1}^{m} X_i . \tag{5}$$

- For obtaining the mean-centered data matrix, denoted by X_i, the mean vector from each data point is subtracted.

$$X_c = X_i - \mu . \tag{6}$$

- The covariance matrix C is calculated by determining the covariance among the characteristics of the data centered around the mean.

$$C = \frac{1}{m} \left(X_c^T \cdot X_c \right) . \tag{7}$$

- An eigenvalue decomposition of the covariance matrix is performed to derive the eigenvectors and eigenvalues. The matrix containing the eigenvectors is referred to as V, while the eigenvalue matrix is denoted as λ.

$$C = V \lambda V^T . \tag{8}$$

- The matrix v is of size $n \times n$ and comprises eigenvectors, representing each column. Additionally, λ is a diagonal matrix with eigenvalues on the diagonal. To construct the matrix, choose the k eigenvectors corresponding to the v_k largest eigenvalues. The vector P contains k elements, denoted as v_1 through v_k.

$$P = [v_1, v_2, \dots, v_k]. \tag{9}$$

- It can be observed that each column denoted by v_i signifies a principal component. A procedure is carried out wherein the mean-centered data is projected into the k principal components to decrease the dimensionality of the data.

$$X_{pca} = X_c P. \tag{10}$$

The dependency among the variables in the dataset can be determined through correlation analysis. A correlation map is generated to identify highly correlated features, and those with values greater than the threshold are removed. When creating a model, features with a high degree of correlation that exceeds a threshold value of 0.95 are selected. The $X_{resampled}$ data is transformed into X_{pca} by dimensionality reduction derived from Eq. (10), where each row represents a data point, and each column represents the principal component. There are 83 features in the original dataset, however PCA retained only 65 binary and 67 multiclass components.

Feature Pre-processing: During the feature pre-processing stage, the processed data is divided into three categories which are training, validation, and testing. The processed data for binary classification has 65 dimensions and the multiclass classification data has 67 dimensions. Both normal and attack types are represented by the labels included in these sets.

Training Validation and Testing: Evaluating the performance of a model is done by dividing the dataset into training, validation, and testing sets, with proportions of 60%, 20%, and 20%, respectively. The processed data is used to train three DL models, including FFNN, LSTM and RandNN. These models are then evaluated using the validation datasets. The accuracy attained through each model is assessed using the test set. The model is designed to categorize a 'Normal' and 'Attack' class for both binary and multiclass classification problems.

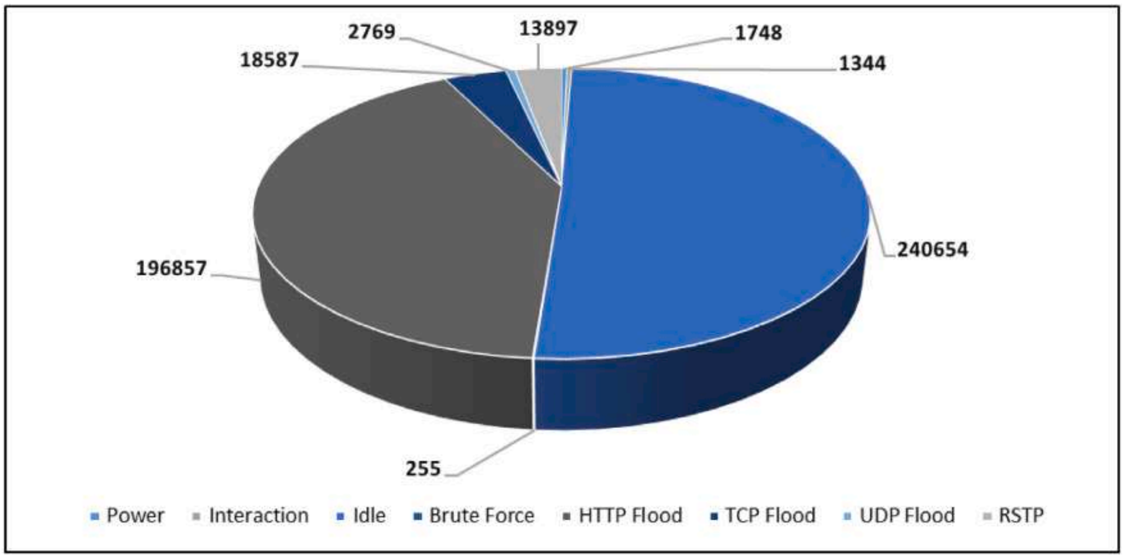


Fig. 6. Dataset's statistical analysis shows packet distribution across various categories.

3.3. Description of the dataset

In this work, the CIC IOT 2022 dataset has been used for intrusion detection. The dataset and testbed arrangement as discussed in [47] can be used for profiling, behavioral analysis, and vulnerability assessment of IoT devices using various protocols. The dataset was captured using network traffic of IoT devices across distinct domains. Audio, Camera, and Home automation devices were used by Canadian Institute for Cybersecurity (CIC) laboratory. Generation of the dataset involved profiling 60 IoT devices connected to the IoT network. The dataset consist of packet headers that were collected from each device during various states, including when it was powered on, idle, active, and during interactions through number of experiments of IoT traffic capture recording in 30 days. Low Orbit Ion Cannon (LOIC) tool was used to launch a denial-of-service (DoS) attack using HTTP, UDP, and TCP Flooding on IoT devices. Hydra and Nmap tools were used for bruteforce and Camera Real Time Streaming Protocol (RTSP) URL attacks for evaluating IoT devices behaviors.

Conversion of Dataset: The CIC IOT 2022 dataset was stored in pcap files. In this research, all collected data of pcap files are converted into csv files to extract features using the CICFlowMeter 4.0 on the Windows 10 operating system. The CICFlowMeter is an open-source software that produces bidirectional flows by analyzing pcap files and extracting relevant characteristics from these flows [48]. It converts network traffic flow into statistical features of the dataset. The CICFlowmeter 4.0 has the ability to select features from a wireshark packets and includes a number of important features and flow timeout [40,49]. This method assigns an arbitrary flow timeout value, such as 60 s for both TCP and UDP packets. Therefore, UDP flows end when a flow timeout occurs while TCP flows end when a connection is broken.

The dataset consists of camera, audio and home automation devices categorized into four distinct normal classes; Power, Idle, Active, and Interaction and five attack classes; HTTP Flood, TCP Flood, UDP Flood, RSTP, and Brute Force. The dataset comprises 84 distinct features and a label that denotes a normal or an attack class. The active class is excluded from this research due to overlap with the idle class. The Idle class is subject to under-sampling by randomly selecting packets for a duration of three days. The dataset comprises 476 111 records, of which 243 746 classify under the Normal label, and 232 365 belong to various attack classes. Figs. 6 presents the distribution of the dataset of each class.

3.4. Hardware, software, and libraries

This work was conducted on Jupyter Notebook application which is an interactive Python-based IDE within the Anaconda distribution for executing and evaluating the proposed methodology [50]. The study utilized Keras, a Python-based deep learning [51]. The hardware used is a Core i7-9700 CPU @3.00 GHz (8 CPUs) processor with a 32 GB RAM. Python 3.10.4 programming language is used along with the open-source module pandas-profiling 3.6.6 for data analysis [52]. Several other factors affect the results of the Deep Learning (DL) algorithm such as batch size, learning rate, and the type of optimizer employed for minimizing the loss function. Optimization helps to minimize the cost function while reducing the resources and effort. After thoroughly evaluating the available alternatives, the ADAM optimizer was selected due to its ability to maximize the categorical cross-entropy loss for multiple classes [53].

4. Experimental configuration of intrusion detection models

This section describes three proposed Deep Learning (DL) models for detecting intrusion and furnishes comprehensive specifications. The following section delineates the mathematics of these models and architectural design of binary and multiclass classification for intrusion detection. The theoretical concepts of the traditional Machine Learning (ML) models will be discussed for comparison with proposed models.

4.1. Proposed model 1 : Feed Forward Neural Network

The first model used in this work to detect intrusion is the Feed Forward Neural Network (FFNN). FFNN has been successfully model used in the number of applications encompasses classification and recognition [54]. FFNN architecture presents the hyperparameter optimization by implementing Scikit learns, RandomizedSearchCV, and KerasClassifier wrapper for the Keras library. Hyperparameter optimization is used to identify the best set of hyperparameters for the given dataset. Four hyperparameters are selected for specifying the neural network's design, which includes the number of hidden layers in the neural network, the number of neurons in each hidden layer, the dropout rate to prevent overfitting and the L2 regularization coefficient. The model is fine-tuned using a suitable set of hyperparameters to enhance classification accuracy and mitigate the risk of overfitting [34]. Dropout regularization technique is used after every hidden layer to mitigate overfitting. The choices for selecting hyperparameter includes a variety of configurations as mentioned below:

$$\begin{aligned}\text{Number of hidden layers:}\quad & [1,2,3]\\ \text{Number of neurons:}\quad & [64,128,256,512]\\ \text{Dropout regularization rate:}\quad & [0.1,0.2,0.3,0.4,0.5]\\ \text{L2 regularization rate:}\quad & [0.001,0.01,0.1,1]\end{aligned}$$

Mathematical Modeling: The mathematical representation of proposed FFNN is derived in the following steps:

- For a given dataset, Let $\mathbf{x}$ be the input vector, the output of the initial hidden layer is represented as:

$$h_1 = \mathrm{ReLU}(W_1 \cdot x + b_1), \tag{11}$$

  where $\mathbf{W}_1$ denotes the weight matrix that connects the input layer to the first hidden layer, $\mathbf{b}_1$ is the bias vector, and ReLU is the activation function that implements the rectified linear unit. The output $\mathbf{h}_1$ computes the output of the first hidden layer.
- The output of the subsequent hidden layers of the neural network is represented by Eq. (12):

$$h_i = \mathrm{ReLU}(W_i \cdot h_{i-1} + b_i), \tag{12}$$

  where i denotes the index of the layer.
- The output layer can be mathematically expressed as:

$$y = \mathrm{softmax}(W_o \cdot h_{\text{num\_hidden\_layers}} + b_o), \tag{13}$$

  where $\mathbf{W}_o$ and $\mathbf{b}_o$ denote the weight matrix and bias vector that connects the final hidden layer and the output layer, respectively.
- The dropout technique involves a process of masking in which a portion of input units is set to zero during the training phase. The masking process for each layer can be represented as $\mathbf{M}_i$, a binary mask that has the same shape as the output of the corresponding layer. Each output layer is obtained through element-wise multiplication with the corresponding mask in the training phase as expressed by Eq. (14):

$$h_i = M_i \cdot \mathrm{ReLU}(W_i \cdot h_{i-1} + b_i). \tag{14}$$

- L2 regularization loss function is used to enhance weight minimization. Each layer's weight matrices are modified according to the loss function. In Eq. (15), the L2 regularization is determined as 0.5 multiplied by the L2 regularization coefficient.

$$L2_{reg} = 0.5 \cdot l2_{reg} \cdot \left(\|W_1\|^2 + \|W_2\|^2 + \cdots + \|W_o\|^2\right). \tag{15}$$

  where the coefficient $l2_{reg}$ is used as the L2 regularization term, and $|\mathbf{W}_i|$ denotes the sum of the Frobenius norms of the weight matrices $\mathbf{W}_i$.

Model Architecture: The input layer consist of 65 features for binary class and 67 features for multiclass classification. The algorithm performs a randomized search over the hyperparameters to determine the best combination to detect intrusions in the dataset. A cross-validation approach using three folds and ten number of iterations was conducted to evaluate the model performance. The model systematically iterates through various configuration of hyperparameters and identify the best combination that maximizes the performance using the RandomizedSearchCV in less processing time using training and validation sets. The automated optimization of the model's architecture and regularization parameters eliminates the need of manual tuning, thereby saving time and effort. Different hyperparameter combinations were tried out during the search process for each layer. After completing the search process, the best configuration of the model hyperparameter is selected.

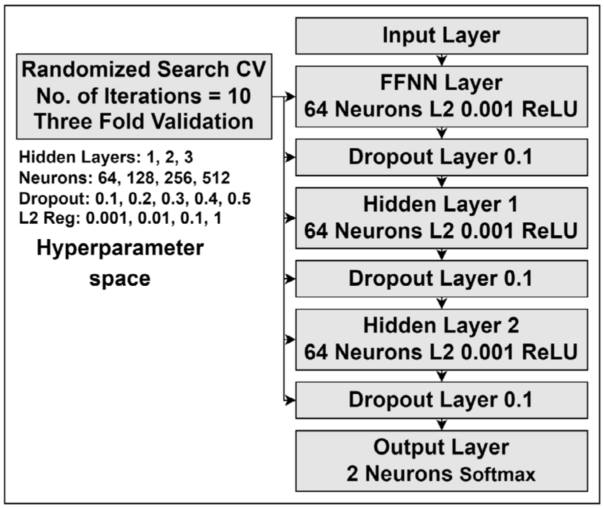


Fig. 7. Architecture of FFNN for binary classification.

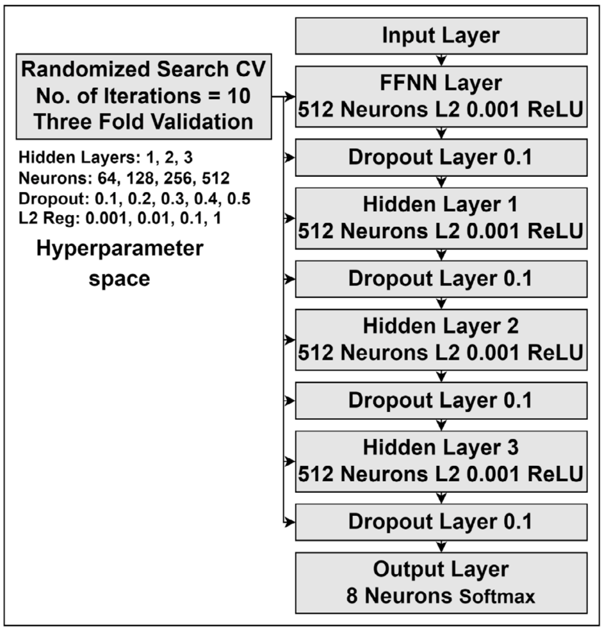


Fig. 8. Architecture of FFNN for multiclass classification.

Figs. 7 and 8 shows the proposed FFNN architecture for binary and multiclass classification. The model comprises three densely connected hidden layers of 512 neurons for multiclass. The model output layer comprises of eight neurons, reflecting various number of categories within the dataset. The model has two hidden layers for binary class, consist of 64 densely connected neurons to the output layer of 2 neurons for each category. The ReLU activation function is used for hidden layers. The Softmax activation function is used in the final layer, which is a widely adopted approach for addressing multiclass problems. The FFNN model uses the Adam optimizer and the categorical cross-entropy loss function to facilitate weight update for training. The model is fine-tuned by implementing the L2 regularization technique, a weight loss of 0.001 and a dropout rate of 0.1 for both binary and multiclass classification. The model's training process involved 100 epochs with 32, 64, and 128 batch sizes. Additionally, various configurations of hidden layers and neurons within each layer were explored.

4.2. Proposed model 2 : Long Short-Term Memory

The Long Short Term Memory (LSTM) algorithm tackles the data preprocessing requirements for tasks including time series, prediction and classification. The proposed LSTM model analyzes complex and large datasets to detect cyber attacks. The model uses LSTMs and dense layers in a sequential structure. LSTM layers collect input data on temporal relationships and long-term trends. The proposed model consists of three LSTM layers, each with different combinations of LSTM units which consist of 16-16-16, 32-32-32, 64-64-64, 128-128-128 for binary class and 128-128-128, 256-128-128, 256-256-128, 256-256-256 for multiclass classification. The LSTM layers are presented with a random normal distribution having a mean of 0.0 and a standard deviation of 1.0 through an initializer. This kind of randomization enhances parameter diversity, which boosts the model's ability to learn complex patterns. A noise initializer has been introduced to improve the learning capabilities of the model and manage data uncertainties efficiently. When LSTM layers are set up, patterns are built up so that sequential data can be sent from one layer to the next. The recurrence and bias initializers have been configured to use the initializer technique to make sure that weight initialization is identical.

Mathematical Modeling: For representing mathematical modeling of the proposed LSTM model, consider an example of the LSTM layer having 256 units at time step t which includes:

- The forget gate at time step t is given by Eq. (16):

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f), \tag{16}$$

where,

$$W_f \in \mathbb{R}^{256\times256},$$
$$b_f \in \mathbb{R}^{256}.$$

- The input gate at time step t is computed as follows:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i), \tag{17}$$

where,

$$W_i \in \mathbb{R}^{256\times256},$$
$$b_i \in \mathbb{R}^{256}.$$

- The candidate hidden state at time step t is computed as:

$$C_t \sim \tanh(W_c \cdot [h_{t-1}, x_t] + b_c), \tag{18}$$

where,

$$W_c \in \mathbb{R}^{256\times256},$$
$$b_c \in \mathbb{R}^{256}.$$

- The cell state at time step t is given as:

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t. \tag{19}$$

- The output gate at time step t is expressed as:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o), \tag{20}$$

where,

$$W_o \in \mathbb{R}^{256\times256},$$
$$b_o \in \mathbb{R}^{256}.$$

- The hidden state at time step t is calculated using Eq. (21):

$$h_t = o_t \odot \tanh(C_t), \tag{21}$$

where,

σ	represents sigmoid activation function.
$\odot$	represents multiplication.
W_f, W_i, W_c, W_o	are weight matrices of each gate.
b_f, b_i, b_c, b_o	are bias vectors of each state.
h_{t-1}	represents the previous hidden state.
x_t	represents the current input.

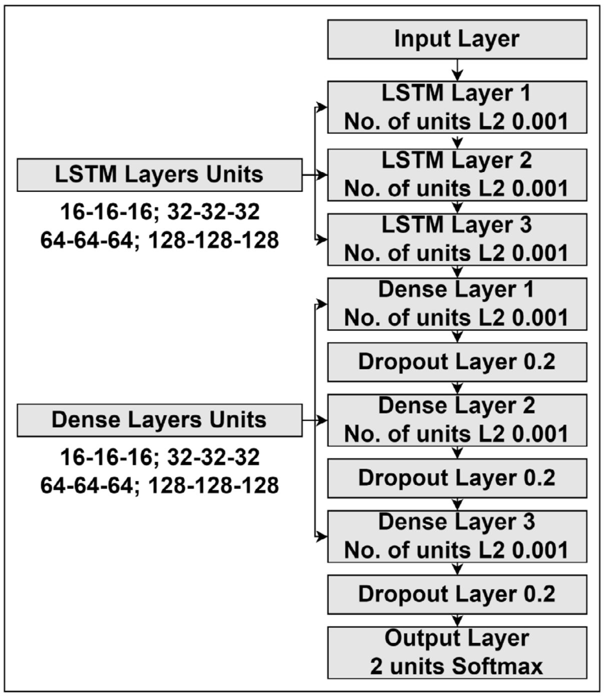

Input Layer
LSTM Layer 1 No. of units L2 0.001
LSTM Layer 2 No. of units L2 0.001
LSTM Layer 3 No. of units L2 0.001
LSTM Layers Units
16-16-16; 32-32-32
64-64-64; 128-128-128
Dense Layer 1 No. of units L2 0.001
Dropout Layer 0.2
Dense Layer 2 No. of units L2 0.001
Dropout Layer 0.2
Dense Layer 3 No. of units L2 0.001
Dropout Layer 0.2
Dense Layers Units
16-16-16; 32-32-32
64-64-64; 128-128-128
Output Layer 2 units Softmax

Fig. 9. Architecture of LSTM for binary classification.

- For consideration, the first Dense Layer will take 256 units. The weighted sum at time step t of the first dense layer z_t is expressed as:

$$z_t = W_z \cdot h_{t-1} + b_z, \tag{22}$$

where,

$$W_z \in \mathbb{R}^{128\times256},$$
$$b_z \in \mathbb{R}^{128}.$$

W_z are weight matrices.
b_z are bias vector.

- The *ReLU* activation function a_t is applied to the weighted sum as follows:

$$a_t = \max(0, z_t). \tag{23}$$

- A dropout layer d_t is used after every dense layer with a rate of 0.2 by applying the regularization technique to the activation values a_t.

$$d_t = \text{dropout}(a_t, 0.2). \tag{24}$$

Model Architecture: The model architecture comprises of three LSTM layers and three dense layers. The number of units in each layer is mentioned in the architecture of LSTM model, illustrated in Figs. 9 and 10 for binary and multiclass classification. The model has the ability to capture complex non-linear relationships within the data by activating the dense layers using the Rectified Linear Unit (ReLU) function. Moreover, the dense layer is subject to kernel regularization with a coefficient of 0.001 and a dropout rate of 0.2 to address overfitting. The ADAM optimization algorithm is utilized for hyperparameter optimization, while categorical cross-entropy function is employed to calculate the loss [55]. During the training phase, the model is provided with input sequences and their corresponding labels. The best architecture consist of 128 units of three LSTM and fully connected dense layers for binary classification and 256 units of three LSTM layers and 128 units of three fully connected dense layers for multiclass classification.

The model input, denoted as x_t, exhibits a dimensionality of 65 for binary and 67 for multiclass, whereas the previous hidden state, represented by h_{t-1}, possesses a dimension of 256 units. The dimensions of the W_f, W_i, W_c, W_o weight matrices can be determined from the number of units in the LSTM layers which is 256. The sum of the dimensions of h_{t-1} and x_t are 321 for binary and 323 for multiclass. Moreover, the dimensions of these matrices would be $(256, 321)$ and $(256, 323)$. The bias vectors b_f, b_i, b_c, b_o must have

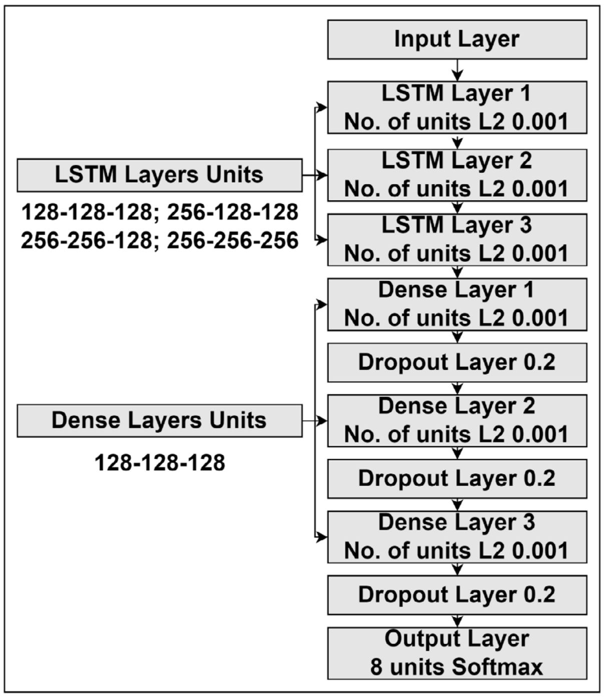


Fig. 10. Architecture of LSTM for multiclass classification.

the same dimensions 256 as the number of units to maintain consistency. The weight matrices W_z of the dense layers would possess dimensions of (128, 256) to establish a connection between the current layer and the previous hidden state. Additionally, the bias vectors b_z has a dimension of 128 corresponding to the number of units. The dropout layer is used after every dense layer, with a dropout rate of 0.2. The training procedure is executed for 100 epochs, utilizing batch sizes of 32, 64, and 128.

4.3. Proposed model 3 : Random Neural Network

The Random Neural Network (RandNN) is an Artificial Neural Network (ANN) that employs the random allocation of connection weights among the neurons. Unlike conventional neural networks with fixed configurations, RandNN has a randomly generated architecture, which provides a unique feature that allows for exploring a large range of network structures. The RandNN comprises interconnected neurons organized into input, hidden, and output layers. The neurons in the RandNN are interconnected via random connections, allowing for data transfer across the network efficiently. RandNN aims to understand the complexities of neuron connections and activation function within a network. The random initialization of weights in RandNN enables applications such as generating random patterns, investigating neural network characteristics, and examining network performance under complex situations. Although random neural networks do not possess the capacity to acquire knowledge from data in the same manner as conventional neural networks, they can still demonstrate exciting behaviors and interactions. A neuron's signal count can be changed using RandNN with positive or negative signals and zero signs. A neuron "fires" when its potential is positive, delivering random positive or negative signals to other neurons. Negative signals establish limitations, while positive signals promote activation [56]. The model has abilities to boost its learning due to the implementation of probability constraints and simplify complex calculations. Because the algorithm is fully distributed, it is perfect for deployment in hardware and IoT devices with minimal resources.

Mathematical Modeling: The mathematical representation of the proposed model makes it easier to understand the intricate random patterns displayed by neurons. The model assumes that there is a system consisting of N neurons, where each individual neuron is exposed to both excitatory and inhibitory signals that originate from external sources. At a given time, a neuron's excitation level is represented by its internal state, denoted as $s_n(t)$. A mathematical model for a RandNN is explained in the following section.

4.3.1. Network description

Let us consider a RandNN model comprising N neurons, where each individual neuron is exposed to both excitatory and inhibitory signals that originate from external sources which may include sensors or other cells. The rate at which excitatory and inhibitory signal received for a given neuron n represented by the symbols $\psi^{+}n$ and $\psi^{-}n$, respectively where n is an element of the set $1, \ldots, N$.

The representation of the internal state of neuron n at time $t \geq 0$ is denoted by $s_n(t)$. When the value of $s_n(t)$ is greater than zero, the occurrence of an inhibitory signal directed toward neuron n during time t results in a decrease of one unit in its internal state, expressed as $s_n(t^+) = s_n(t) - 1$. In cases where $s_n(t) = 0$, the occurrence of an inhibitory signal does not exert any effect. The increment of $s_n(t)$ by $+1$ is a consequence of an excitatory signal. A neuron n is said to be in an excited state if its internal state, represented by $s_n(t)$, is greater than zero. The device is capable of releasing a signal with a probability of $R_n \Delta t$ within the time interval $[t, t + \Delta t]$. $R_n > 0$ denotes the firing rate of a given neuron, while R_n^{-1} represents the average firing delay of the excited nth neuron.

4.3.2. Neuronal interactions

Neuronal interactions can occur through various mechanisms that include:

- When neuron α is excited, as shown by $s_\alpha(t) > 0$, its internal state decreases by one.

$$s_\alpha(t^+) = s_\alpha(t) - 1. \tag{25}$$

- Neuron α can send an excitatory signal to neuron β with probability $\phi^+(\alpha, \beta)$, resulting in:

$$s_\alpha(t^+) = \begin{cases} s_\alpha(t) - 1 \\ s_\alpha(t) + 1 \end{cases} \tag{26}$$

- Neuron α can transmit an inhibitory signal to neuron β, which occurs with a probability of $\phi^-(\alpha, \beta)$, which is leads to:

$$s_\alpha(t^+) = \begin{cases} s_\alpha(t) - 1, & \text{if } s_\alpha(t) > 0 \\ 0, & \text{if } s_\alpha(t) = 0 \end{cases} \tag{27}$$

- Neuron α can activate neuron β with a probability of $\phi(\alpha, \beta)$. This activation function decreases the value of $s_\alpha(t)$ by 1, given that $s_\alpha(t)$ is greater than 0. Therefore,

$$s_\alpha(t^+) = \begin{cases} s_\alpha(t) - 1, & \text{if } s_\alpha(t) > 0 \\ s_\alpha(t) - 1, & \text{otherwise,} \end{cases} \tag{28}$$

If the trigger is activated, there are two possible outcomes:

- Neurons α and β have potential to increase the state of neuron n by $+1$ while decreasing their own states by -1 with a probability of $\delta(\beta, n)$, simultaneously.
- The transition to neuron n is triggered with a probability of $\sigma(\beta$,n) and the above process is repeated.

4.3.3. Activation probability

The probability of activation for neuron n, indicated by γ_n, can be calculated as follows:

$$\gamma_n = \epsilon_n^+ R_n + \epsilon_n^-, \tag{29}$$

where,

$$\epsilon_n^+ = \psi_n^+ + \sum_{\beta=1, \beta \neq n}^{N} R_\beta \gamma_\beta \phi^+(\beta, n) + \sum_m \mu_n R_{\alpha_1}^{\beta=1,m-1} \gamma_{\alpha_\beta} \phi(\alpha_\beta, \alpha_{\beta+1}) \gamma_n \delta(\alpha_{\beta+1}, n)$$

$$\epsilon_n^- = \psi_n^- + \sum_{\beta=1, \beta \neq n}^{N} R_\beta \gamma_\beta \phi^-(\beta, n) + \sum_m \mu_n R_{\alpha_1}^{\beta=1,m-1} \gamma_{\alpha_\beta} \phi(\alpha_\beta, \alpha_{\beta+1}) \gamma_n \sigma(\alpha_{\beta+1}, n)$$

The activation threshold ϵ_n^+ and ϵ_n^- for neuron n is calculated, including their firing rates and probability with various interactions. The thresholds affect neuron excitation and RandNN dynamics.

4.3.4. Activation function

The activation function $\xi(\eta)$ is a important component in RandNN, where η represent the input. This is the summation of the product of the input values η_n, and their weights γ_n.

$$\Xi(\eta) = \sum_n \gamma_n \eta_n. \tag{30}$$

The activation function integrates the activation probabilities and external input to compute the overall activation of the network.

Model Architecture: The architecture consists of input and output neurons and the positive and negative weight associated with each neuron. The model integrates clusters, which represent extremely connected cells. Each cluster possesses a unique group of neurons and can receive inhibitory input from cell units outside the cluster. Fig. 11 shows the RandNN architecture which consist of input, hidden and output layers. The proposed model determine the activation probability for a single cell inside a cluster considering excitatory and inhibitory inputs. It depend on firing rates, connection weights, and input values. Backpropagation through time (BPTT) was used to implement the algorithms, which defined unique features and initialization weights techniques

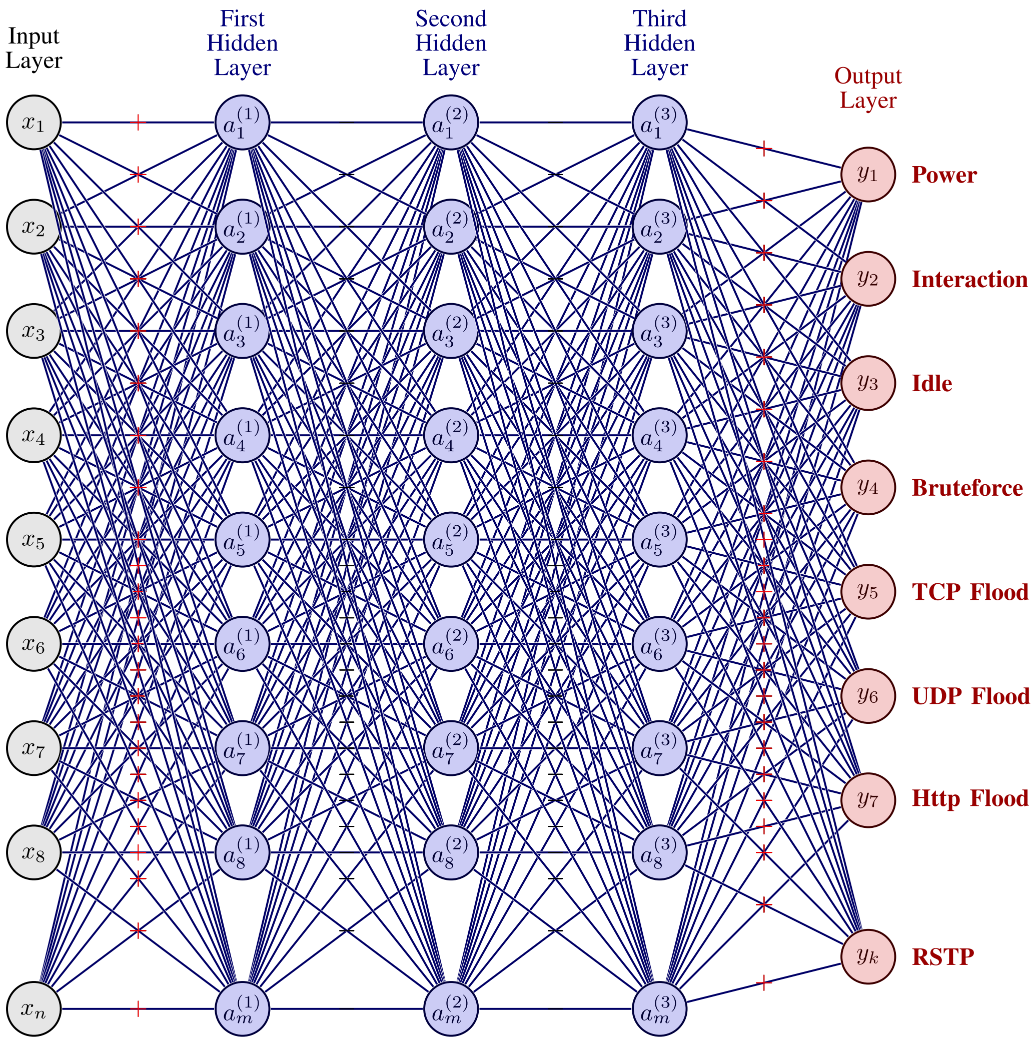


Fig. 11. Architecture of Random Neural Network.

using the input neurons, connection matrices, learning rate and number of epochs. The algorithm helps in storing and retrieving weights, the computation of neuron firing rates and determine loss or accuracy.

The SequentialRNN model is used with a modification in rnnsim library for training a RandNN to detect intrusion [56]. The architecture comprises of a dual-layered structure with 4 and 6 hidden layers for binary and multiclass classification. The model uses a learning rate of 1.0, 0.1, and 0.01 with 100 epochs. The input X, represent the dimension of 65 for binary and 67 for multiclass. The predictive output is determined through recurrent updates of the firing rates of neurons, which is determine by the connectivity and weight matrices. The model considers the neurons interaction, firing rates, internal states, excitatory and inhibitory signal and activation probabilities.

4.4. Theoretical concepts of ML models

For comparison with the proposed Neural Networks, the theoretical concepts of the Naive Bayes, Decision Tree, K Nearest Neighbor, and Random Forest models are presented. The models also explore its architecture with mathematical equations.

Naive Bayes: Naive text classification uses Bayes due to its simplicity. Naive Bayes (NB) assumes that the features are independent and normally distributed and calculates the probability of a data point belonging to a particular class on the Bayes theorem [57]. Using NB for IDS may be challenging due to the feature independence assumption. It evaluates each attribute independently, which may result in inaccurate classifications. Despite its simplicity, the NB analyzes attribute patterns to identify potential threats. This model yields significant data but may fail to capture complex attack patterns. It may overlook characteristic correlations. NB simplify the likelihood by assuming feature independence as:

$$P(X|C=c) = P(x_1|C=c) \cdot P(x_2|C=c) \quad \cdot \ldots \cdot \quad \cdot P(x_n|C=c).$$

where,

- $x_1 - x_n$ are instance's properties.
- Observing feature X_i for class c is $P(X_i|C = c)$.
- A new instance is classified by the class with the highest posterior likelihood.

NB assumes feature independence, which may not be true in real life. When characteristics are interact, this assumption may lead to inaccurate predictions. Since, it calculates probabilities using training data, NB is vulnerable to lack of data and imbalanced class distributions.

Decision Tree: The Decision Tree (DT) model employs a hierarchical approach to partition the data and enable accurate predictions of dataset [58]. This supervised learning technique can solve classification and regression problems. The DT model creates a hierarchical structure with nodes and class labels representing leaf node properties. The decision tree's main purpose is:

- Identify the optimal property and allocate it to the root node of the tree.
- Create subsets of the training data consisting of data with identical attribute values for each subset.
- Repeat the above steps until a final node is reached.

In dataset analysis, selecting the best attribute to serve as the root node of a decision tree is a crucial task. Information gain is used for categorical categories, while the Gini index is better for continuous attributes. Entropy or Gini impurity can be used to calculate information gain define by Eq. (31):

$$\text{Information Gain} = \text{Entropy(p)} - \frac{1}{N}\sum_{i=1}^{N} p_i \cdot \text{Entropy(c}_i\text{)} \tag{31}$$

where,

- N is the number of children.
- p_i is the percentage of occurrences in child i.
- Entropy(p) and Entropy(c$_i$) are the entropies of the parent and child nodes.

The decision rule is essential for determining each node's attribute and threshold used for data partitioning. The equation for the decision rule that compares the attribute value of a given instance to a certain threshold and assigns the sample to the left or right child node is as follows:
if, attribute_value $\leq$ threshold
If the condition is met, go to the left child node; otherwise, go to the right. In binary classification, the determination of the class label can be achieved by assigning the majority class. So, the class label is chosen by selecting the majority class present in the leaf node. The diversity of decision trees is dependent on the implementation criteria used for evaluating and dividing nodes.

Random Forest: Random Forest (RF) is a supervised learning algorithm for classification tasks. The ensemble model uses a variety of trees to predict a certain target variable [59]. The algorithm receives a set of n samples as input and generates many decision trees using a subset of the available input features. Following that, a majority voting technique is performed on the results of each tree to generate the forecast for the target variable. The RF operating mechanism is as follows:

- Choose k features at random from a list of m features in the data, where k is significantly less than m.
- This algorithm finds the best division for a given collection of k attributes.
- Divide the node into its child nodes using the most effective split.
- Continue iterating the above process until an end node is reached.
- A forest of trees can be made by repeating the above steps.

To derive the mathematical representation, consider a dataset D with n instances of m features using Eq. (32), where X represents the feature and y represents class labels.

$$D = (X_1, y_1), (X_2, y_2), \dots, (X_n, y_n). \tag{32}$$

The RF approach employs bootstrapping to construct several decision trees by randomly picking subsets from the dataset. Construction of decision trees includes selecting a random subset of features, b, from B decision trees (where $b = 1, 2, \dots, B$). The act of voting consists of each decision tree b contributing a prediction for the class label of a new instance X_{test}. The predicted class label for tree b can be denoted as:

$$\hat{y}_b(X_{test}). \tag{33}$$

The RF method combines the predictions of multiple decision trees to reduce overfitting. The ensemble of decision trees is randomized by training each tree on a different bootstrap dataset and a subset of features. The RF predict class label via majority voting, in which the individual decision tree predictions are evaluated collectively. The final prediction is determined by selecting the class label with the most votes, formulated as:

$$\hat{y}_{\text{rf}}(X_{\text{test}}) = \arg\max_i \sum_u [\hat{y}_b(X_{\text{test}}) = i] \tag{34}$$

where $i = 1, 2, \dots, K$ denotes the number of class labels in the dataset.

K-Nearest Neighbor: The K-Nearest Neighbor (KNN) algorithm is an ML technique that is based on the principles of supervised learning [60]. The KNN algorithm classifies a new data point based on its similarity to the existing data points. The algorithm optimizes the new data points by choosing the most similar category efficiently. This algorithm is considered non-parametric because it makes no assumptions about the analyzed data. The "lazy learner" algorithm delays training set learning until classification is achieved. The KNN algorithm holds the dataset during training. After that, the algorithm classifies new data into the closest category.

Consider a scenario involving the detection of credit card fraud. The KNN algorithm compared the attributes of a novel transaction to those of previously classified malicious and normal transactions. The model determines whether a transaction is illegitimate or legitimate by calculating the similarity between it and its nearest neighbors. If the majority of the transaction's nearest neighbors are identified as illegitimate the KNN algorithm labels the transaction as malicious. This approach enables the KNN algorithm to recognize patterns in credit card transactions, resulting in the detection of fraud. KNN uses d to discover X_{test} for k closest neighbors in the dataset D. X_{test} is projected class label shown by Eq. (35) where $i = 1, 2, \dots, K$ is the dataset's class label count.

$$C(X_{\text{test}}) = \arg\max_{i} \sum_{r} [C(X_r) = i]. \tag{35}$$

This algorithm involves the computation of distances between the new instance and all instances in the dataset, resulting in a computationally intensive process. Inaccurate values of k can result in either overfitting or underfitting. Specific attributes have greater significance than others in practical scenarios like intrusion detection. However, the KNN algorithm assumes that all features possess equal importance. The KNN algorithm generates distances based on feature magnitude, making it dependent on feature size. Significant features may impact the computation of distance. When using this algorithm, the challenge is handling datasets with imbalanced class distributions. If the percentage of the majority class is too high, it can compromise the accuracy of predictions.

5. Experimental setup and results

In this section, a number of experimental results are presented to evaluate the three proposed DL models discussed in Section 4 to assess their capacity for generalization. The efficacy of the proposed DL models is determined by extracting the metrics to evaluate the classification performance. Furthermore, a comparison with ML and other proven approaches described in the existing literature is also presented.

5.1. Evaluation metrics

The proposed models are evaluated through assessment metrics, such as accuracy, precision, recall, and F1 score, commonly employed in various domains, including threat detection. The following sections describes each metric, followed by formulas and mathematical justifications.

Accuracy: The term "accuracy" refers to the degree of correctness displayed by a model's predictions. The accuracy rate is the proportion of correctly classified instances (including true positives and negatives) to the total number of instances. It is calculated using Eq. (36):

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}, \tag{36}$$

where,

- True Positive (TP): The count of correctly identified instances as positive (i.e., threats).
- True Negative (TN): The count of correctly identified instances as negative (i.e., non-threats).
- False Positive (FP): The count of instances incorrectly classified as positive (i.e., non-threats misclassified as threats).
- False Negative (FN): The count of instances that are incorrectly classified as negative (i.e., threats misclassified as non-threats).

Precision: The concept of precision refers to the degree of accuracy of positive predictions. The true positive rate is the proportion of accurately predicted positive instances relative to the total number of positive instances predicted, including both true and false positives. The formula for precision is given as:

$$\text{Precision} = \frac{TP}{TP + FP}. \tag{37}$$

Recall: Recall, also known as sensitivity or true positive rate, measures the number of accurately predicted positive instances (true positives) relative to all true positive instances (true positives and false negatives). The mathematical expression is given by Eq. (38):

$$\text{Recall} = \frac{TP}{TP + FN}. \tag{38}$$

F1 Score: The F1 score is a metric that combines precision and recall. As a result, achieving an equilibrium between the two measurements. The average of precision and recall helps when precision and recall have uneven choices. It is frequently used in ML and statistical analysis to evaluate the performance of a classification model. The mathematical expression is shown by Eq. (39):

$$\text{F1 Score} = \frac{2 \cdot (\text{Precision} \cdot \text{Recall})}{\text{Precision} + \text{Recall}}. \tag{39}$$

The F1 score provides an unique value that highlights the combined performance of precision and recall.

Table 2
Specifications of Feed Forward Neural Network model layers.

Model	Binary class	Multi class	Input layer	Hidden layer		Dropout layer		Output layer	
				Size	Neuron	Size	Activation function	Size	Activation function
FFNN	✓		65	2	64	0.1	ReLU	2	Softmax
FFNN		✓	67	3	512	0.1	ReLU	8	Softmax

Table 3
Evaluation metrics of Feed Forward Neural Network model for binary classification.

Model	Batch size	Training accuracy	Testing accuracy	Training time	Testing time	Precision	Recall	F1 score
FFNN	**128**	**0.9984**	**0.9993**	**530.89**	**2.8**	**0.9993**	**0.9993**	**0.9993**
FFNN	64	0.9981	0.9989	850.78	2.89	0.9989	0.9989	0.9989
FFNN	32	0.9976	0.9990	1500.21	2.95	0.9990	0.9990	0.9990

Table 4
Evaluation metrics of Feed Forward Neural Network model for multiclass classification.

Model	Batch size	Training accuracy	Testing accuracy	Training time	Testing accuracy	Precision	Recall	F1 score
FFNN	**128**	**0.9797**	**0.9872**	**10708.47**	**30.32**	**0.9873**	**0.9872**	**0.9872**
FFNN	64	0.9748	0.9810	18045.39	30.96	0.9814	0.9810	0.9809
FFNN	32	0.9733	0.9697	35280.79	30.21	0.9708	0.9697	0.9698

5.2. *Result analysis of Feed Forward Neural Network*

Results of FFNN for binary and multiclass classification problems framework demonstrate the impact of various parameters on its performance. The best binary classification configuration is achieved using 64 neurons in the two hidden layers, a dropout rate of 0.1 and a regularization rate of 0.001. With a batch size of 128, the FFNN achieved the maximum accuracy of 99.93%, precision of 99.93%, recall of 99.93%, and F1-score of 99.93%. The best configuration for multiclass is achieved using 512 neurons in three hidden layers, a dropout rate of 0.1 and a regularization rate of 0.001. With a batch size of 128, the FFNN achieved the maximum accuracy of 98.72%, precision of 98.73%, recall of 98.72%, and F1-score of 98.72%. The results highlight the importance of modifying these parameters to enhance the FFNN's efficiency in binary and multiclass, increasing its potential for adaptation and predictions. The specifications of the FFNN are summarized in Table 2, and the metrics outcomes are evaluated in Tables 3 and 4. Figs. 12(a) and 12(b) depict the accuracy and loss plots for various batch sizes for binary and multiclass, respectively.

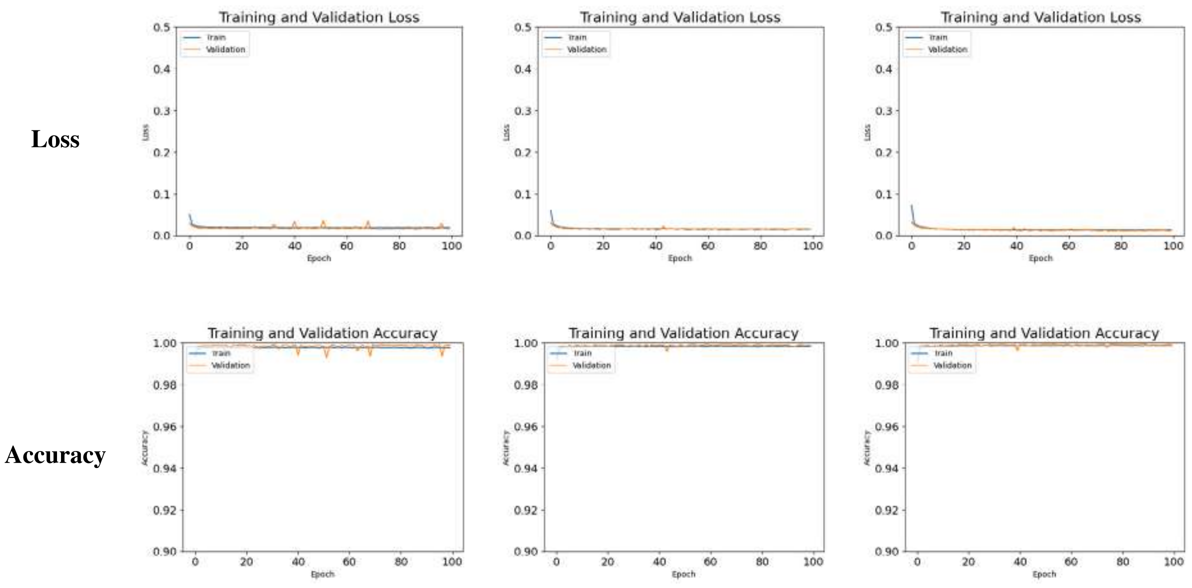
5.3. *Result analysis of Long Short-Term Memory*

This research shows that different configurations have a significant impact when using LSTM layers for binary and multiclass classification tasks. Table 5 displays the LSTM model's specifications. Various configurations for the binary class were evaluated on 16-16-16, 32-32-32, 64-64-64, and 128-128-128 with 32, 64 and 128 batch sizes. LSTM 16-16-16 achieved an accuracy of 99.78% with a 32 batch size, LSTM 128-128-128 attained an accuracy of 99.89% with a 64 batch size, and LSTM 16-16-16 achieved an accuracy of 99.85% with a 128 batch size, respectively. Among all tested configurations, the LSTM 128-128-128 with 64 batch size configuration demonstrated the highest accuracy. Increasing the number of LSTM layers allows for a more comprehensive description of temporal inter dependencies in data. Adding additional layers to the LSTM model facilitates the detection of more complex patterns and prolonged dependencies, resulting in enhanced efficiency and accuracy. The LSTM model results for all four configurations with 32, 64, and 128 batch sizes for binary classification are shown in Table 6. Furthermore, the accuracy and loss plots of the four LSTM models are shown in Figs. 13(a), 13(b), 14(a) and 14(b). The proposed LSTM model demonstrates an increased ability to handle various data uncertainty patterns.

The LSTM configurations used in the multiclass are 128-128-128, 256-128-128, 256-256-128, and 256-256-256. The results showed that the 256-256-256 LSTM model achieved 98.19% accuracy with a batch size of 32, while the 256-128-128 LSTM models attained 98.19% accuracy with a batch size of 64. Furthermore, the 256-256-256 LSTM models achieved an accuracy of 98.60% with batch sizes of 128. As a result, the configuration of 256-256-256 showed maximum accuracy. The results of the LSTM model in various configurations for multiclass are presented in Table 7. Furthermore, Figs. 15(a), 15(b), 16(a) and 16(b) show the accuracy and loss plots for the four LSTM models for various batch sizes. The analysis predicts that different LSTM layer configurations has improved model efficiency.

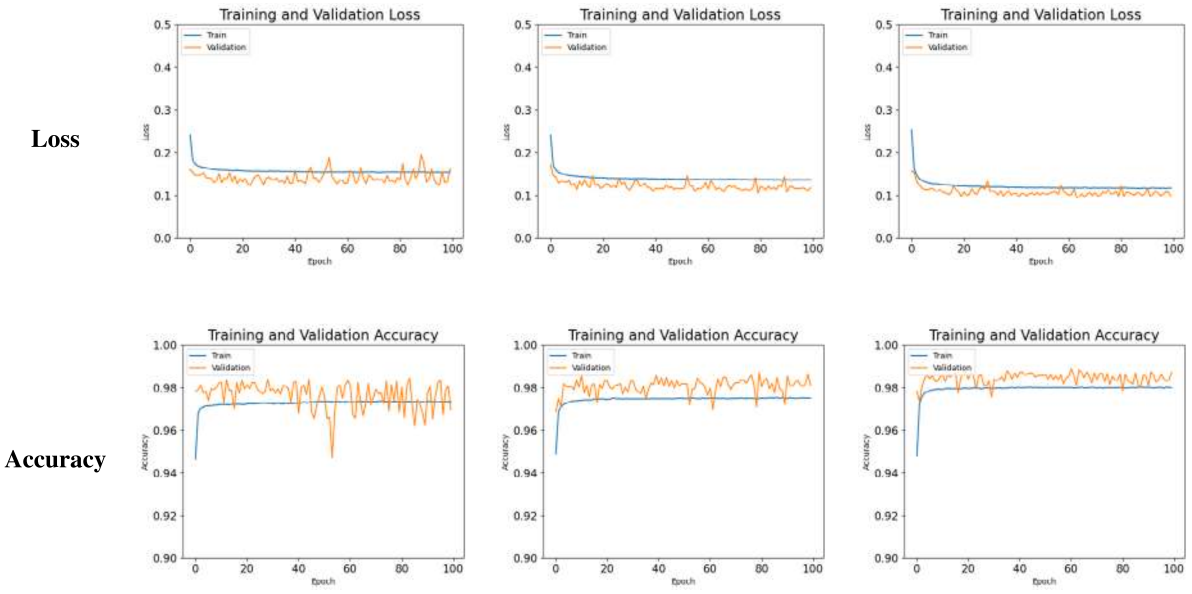
5.4. *Result analysis of Random Neural Network*

Random connections in RandNN have hidden potential to capture complex dependencies. Experiment were conducted to examine the effect of changing the number of hidden layers and learning rates in the RandNN architecture. The sizes of the hidden layers determine the connectivity between neurons, which facilitates the smooth access of data throughout the network. The framework provides techniques for incorporating additional layers into the model and integrating layer-to-layer connections. It also has the



(a) Batch Size 32 (b) Batch Size 64 (c) Batch Size 128

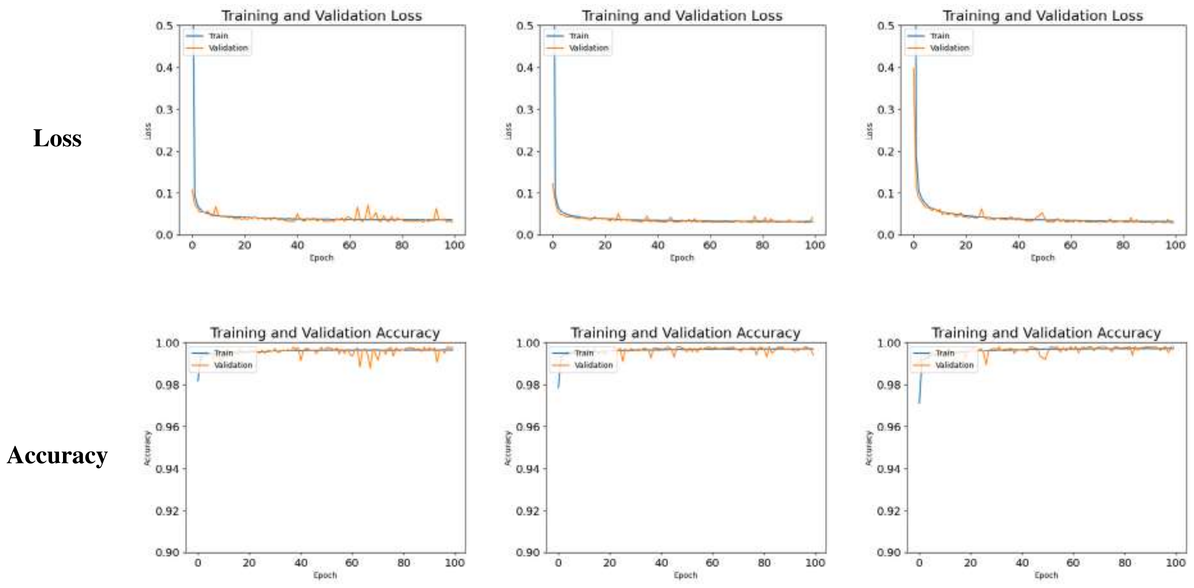
(a) FFNN model plots for binary classification.



(a) Batch Size 32 (b) Batch Size 64 (c) Batch Size 128

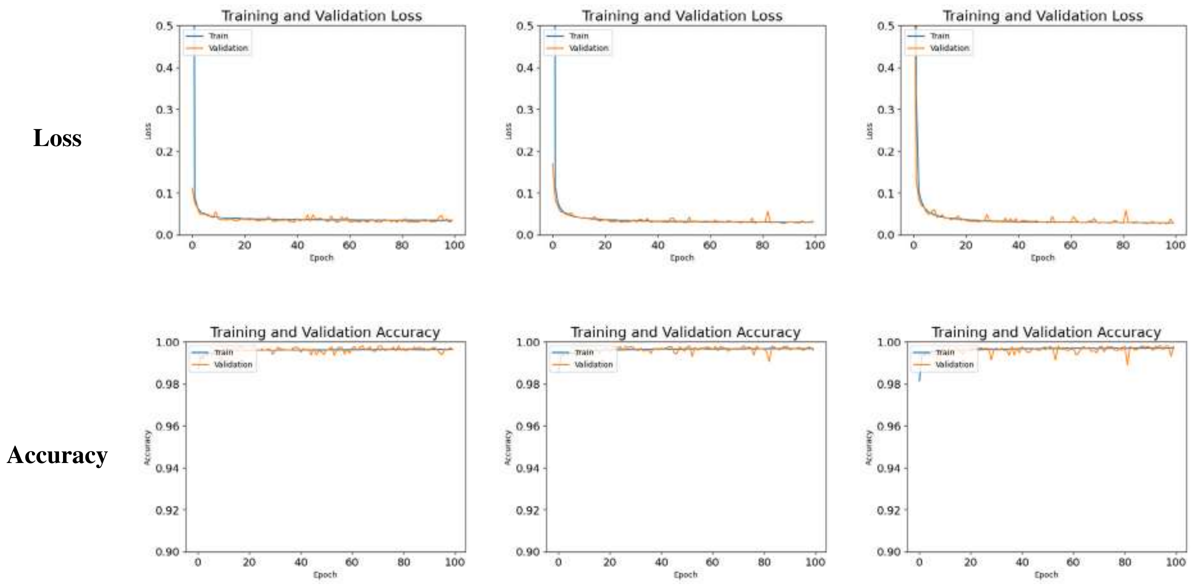
(b) FFNN model plots for multiclass classification.

Fig. 12. Feed Forward Neural Network plots of different batch sizes for binary and multiclass classification.



(a) Batch Size 32 (b) Batch Size 64 (c) Batch Size 128

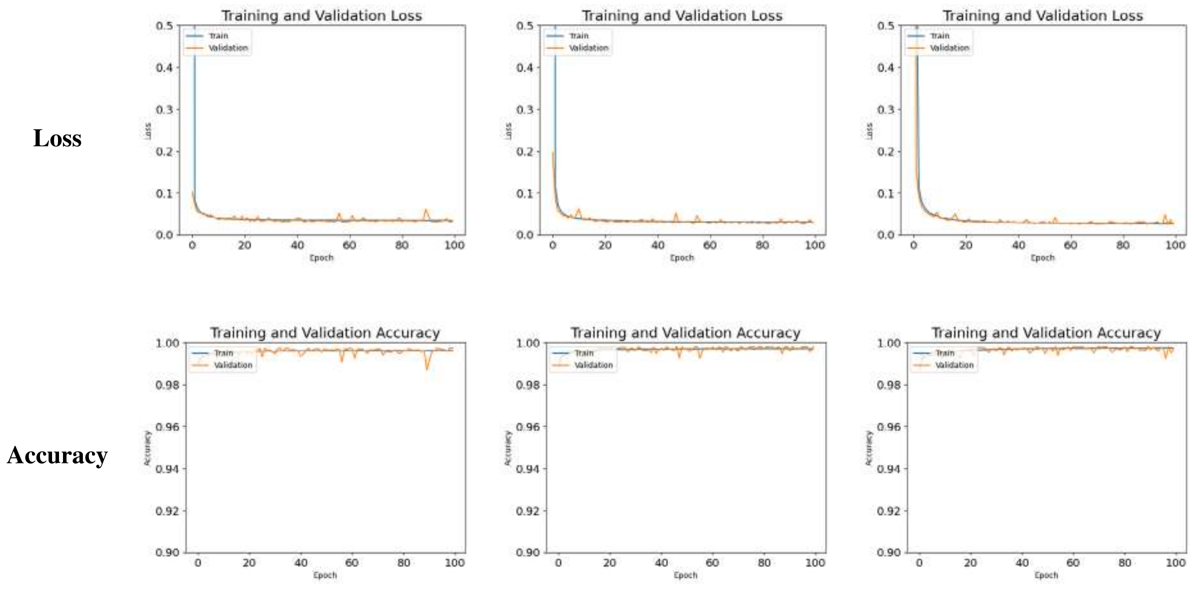
(a) LSTM 16-16-16 model plots for binary classification.



(a) Batch Size 32 (b) Batch Size 64 (c) Batch Size 128

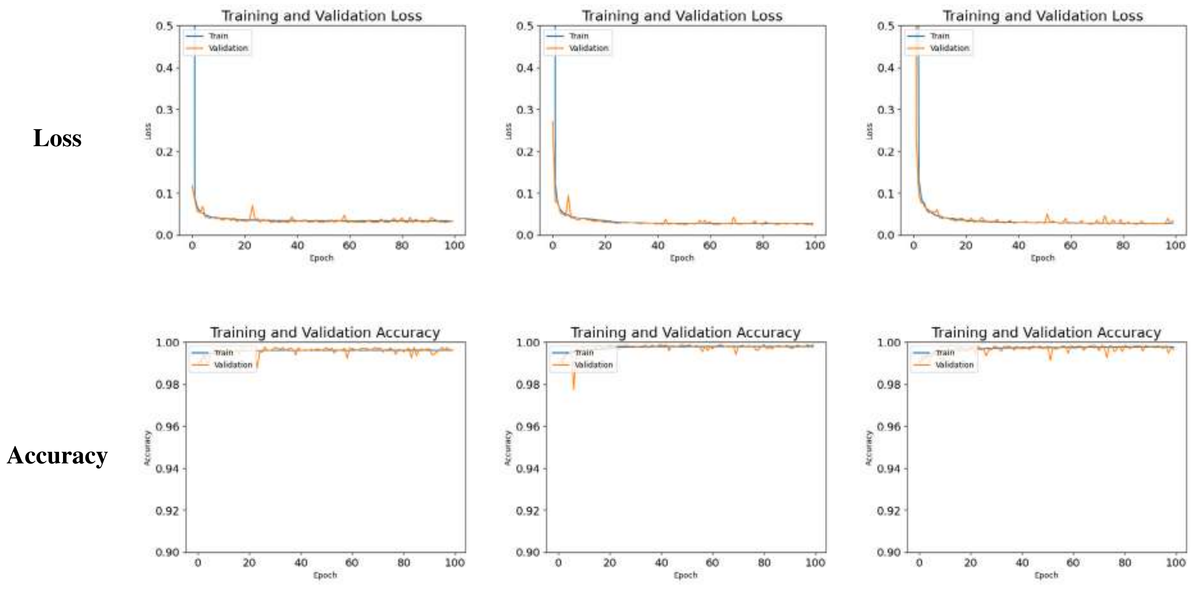
(b) LSTM 32-32-32 model plots for binary classification.

Fig. 13. Long Short-Term Memory plots of different batch sizes for binary classification.



(a) Batch Size 32 (b) Batch Size 64 (c) Batch Size 128

(a) LSTM 64-64-64 model plots for binary classification.



(a) Batch Size 32 (b) Batch Size 64 (c) Batch Size 128

(b) LSTM 128-128-128 model plots for binary classification.

Fig. 14. Long Short-Term Memory plots of different batch sizes for binary classification.

Table 5
Specifications of Long Short-Term Memory model layers.

Model	Binary class	Multi class	Input layer	LSTM layer		Dense layer		Dropout layer		Output layer	
				Size	Units	Size	Units	Size	Activation function	Size	Activation function
LSTM	✓		65	3	16-16-16	3	16	0.2	ReLU	2	Softmax
LSTM	✓		65	3	32-32-32	3	32	0.2	ReLU	2	Softmax
LSTM	✓		65	3	64-64-64	3	64	0.2	ReLU	2	Softmax
LSTM	✓		65	3	128-128-128	3	128	0.2	ReLU	2	Softmax
LSTM		✓	67	3	128-128-128	3	128	0.2	ReLU	8	Softmax
LSTM		✓	67	3	256-128-128	3	128	0.2	ReLU	8	Softmax
LSTM		✓	67	3	256-256-128	3	128	0.2	ReLU	8	Softmax
LSTM		✓	67	3	256-256-256	3	128	0.2	ReLU	8	Softmax

Table 6
Evaluation metrics of Long Short-Term Memory model for binary classification.

Model	Batch size	Training accuracy	Testing accuracy	Training time	Testing time	Precision	Recall	F1 score
LSTM 16-16-16	32	0.9965	0.9978	3944.10	4.74	0.9978	0.9978	0.9978
LSTM 32-32-32	32	0.9965	0.9967	3521.66	4.61	0.9967	0.9967	0.9967
LSTM 64-64-64	32	0.9961	0.9976	3960.71	5.18	0.9976	0.9976	0.9976
LSTM 128-128-128	32	0.9962	0.9968	21 337.72	27.43	0.9968	0.9968	0.9968
LSTM 16-16-16	64	0.9969	0.9949	1767.96	4.28	0.9949	0.9949	0.9949
LSTM 32-32-32	64	0.9967	0.9963	1915.67	4.63	0.9963	0.9963	0.9963
LSTM 64-64-64	64	0.9970	0.9979	2750.24	5.26	0.9979	0.9979	0.9979
LSTM 128-128-128	**64**	**0.9978**	**0.9989**	**8714.29**	**12.72**	**0.9989**	**0.9989**	**0.9989**
LSTM 16-16-16	128	0.9971	0.9985	1212.13	4.49	0.9985	0.9985	0.9985
LSTM 32-32-32	128	0.9969	0.9981	1140.05	4.90	0.9981	0.9981	0.9981
LSTM 64-64-64	128	0.9972	0.9975	1781.01	5.94	0.9976	0.9975	0.9975
LSTM 128-128-128	128	0.9975	0.9974	5287.01	12.43	0.9974	0.9974	0.9974

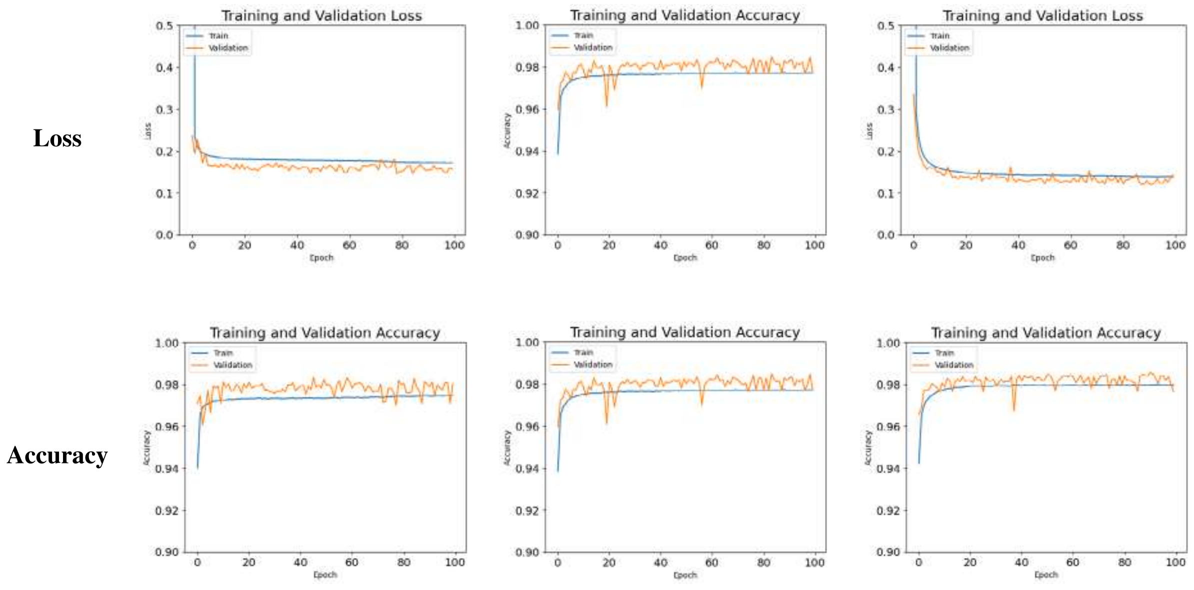
Table 7
Evaluation metrics of Long Short-Term Memory model for multiclass classification.

Model	Batch size	Training accuracy	Testing accuracy	Training time	Testing time	Precision	Recall	F1 score
LSTM 128-128-128	32	0.9746	0.9808	20 269.04	20.57	0.9808	0.9807	0.9808
LSTM 256-128-128	32	0.9750	0.9814	26 422.93	26.94	0.9813	0.9814	0.9813
LSTM 256-256-128	32	0.9737	0.9711	41 768.47	35.68	0.9813	0.9811	0.9809
LSTM 256-256-256	32	0.9749	0.9819	55 464.11	40.05	0.9819	0.9819	0.9818
LSTM 128-128-128	64	0.9771	0.9779	16 800.93	27.04	0.9783	0.9779	0.9776
LSTM 256-128-128	64	0.9776	0.9819	15 124.16	27.85	0.9822	0.9819	0.9818
LSTM 256-256-128	64	0.9776	0.9817	26 894.90	40.65	0.9819	0.9817	0.9818
LSTM 256-256-256	64	0.9765	0.9814	30 606.32	43.20	0.9815	0.9814	0.9813
LSTM 128-128-128	128	0.9798	0.9765	5775.76	20.74	0.9769	0.9765	0.9764
LSTM 256-128-128	128	0.9801	0.9835	8341.94	28.32	0.9837	0.9835	0.9834
LSTM 256-256-128	128	0.9795	0.9825	12 584.59	34.44	0.9828	0.9825	0.9824
LSTM 256-256-256	**128**	**0.9812**	**0.9860**	**18724.71**	**45.14**	**0.9860**	**0.9860**	**0.9859**

ability for added functionality to save and load weights. The specifications of the RandNN model are presented in Table 8. Four and six hidden layers were used to evaluate the model's performance for binary and multiclass classification. Table 9 shows the results of the RandNN model for binary classification with 1.0, 0.1, and 0.01 learning rates (LR). The model attained the highest accuracy of 96.42%, precision of 96.42%, recall of 96.42%, and F1-score of 96.42% for binary classification with 0.1 LR after 100 epochs. Using four hidden layers and a LR of 0.1 improved the model's performance by capturing more complex patterns. Table 10 displays the multiclass classification evaluation metrics of the RandNN model under various LR. For multiclass, the model has achieved an accuracy of 92.89%, precision of 89.07%, recall of 92.89%, and F1-score of 90.82% with a 1.0 LR. Figs. 17(a) and 17(b) show accuracy and loss plots for binary and multiclass problems with varying LR. These findings demonstrate the effect of LR on the performance of RandNN.

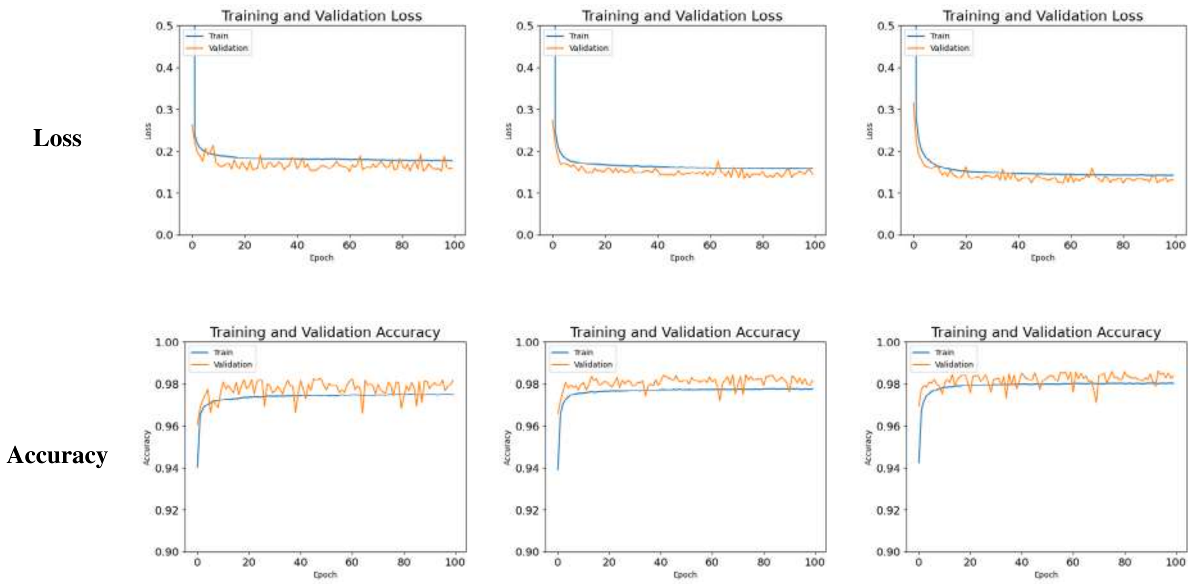
5.5. Comparison of proposed models with ML algorithms and DL based state-of-the-art IDS

In this section, results of the proposed DL models are compared with traditional ML models and a number of state-of-the-art IDS models. Tables 12 and 13 show classification results of the proposed DL models and their comparison with traditional ML techniques; such as Naive Bayes (NB), Decision Tree (DT), Random Forest (RF), and K Nearest Neighbor (KNN) algorithms. For binary and multiclass cases, the proposed FFNN model outperformed the ML models and the two proposed LSTM and RandNN models. As compared to LSTM and RandNN models, FFNN model boosts performance and learns complex non-linear correlations between the input features and output classes by adjusting hyperparameters. All ML models are evaluated using 5-fold cross-validation for training and assessing the model yield evaluation metrics presented in Tables 14 and 15, respectively. Although the accuracy of



(a) Batch Size 32 (b) Batch Size 64 (c) Batch Size 128

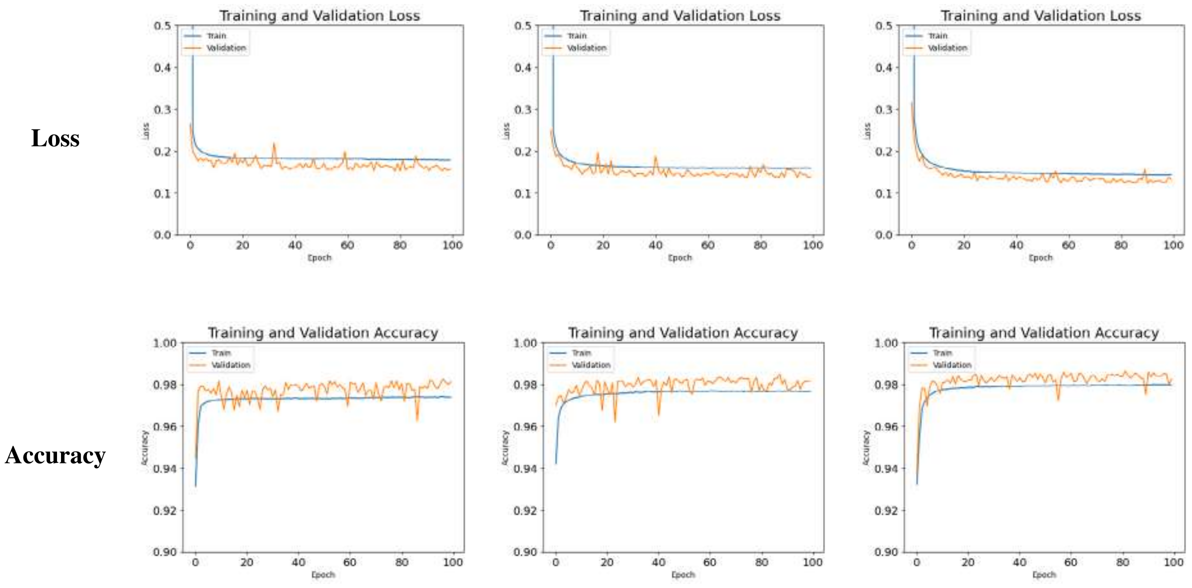
(a) LSTM 128-128-128 model plots for multiclass classification.



(a) Batch Size 32 (b) Batch Size 64 (c) Batch Size 128

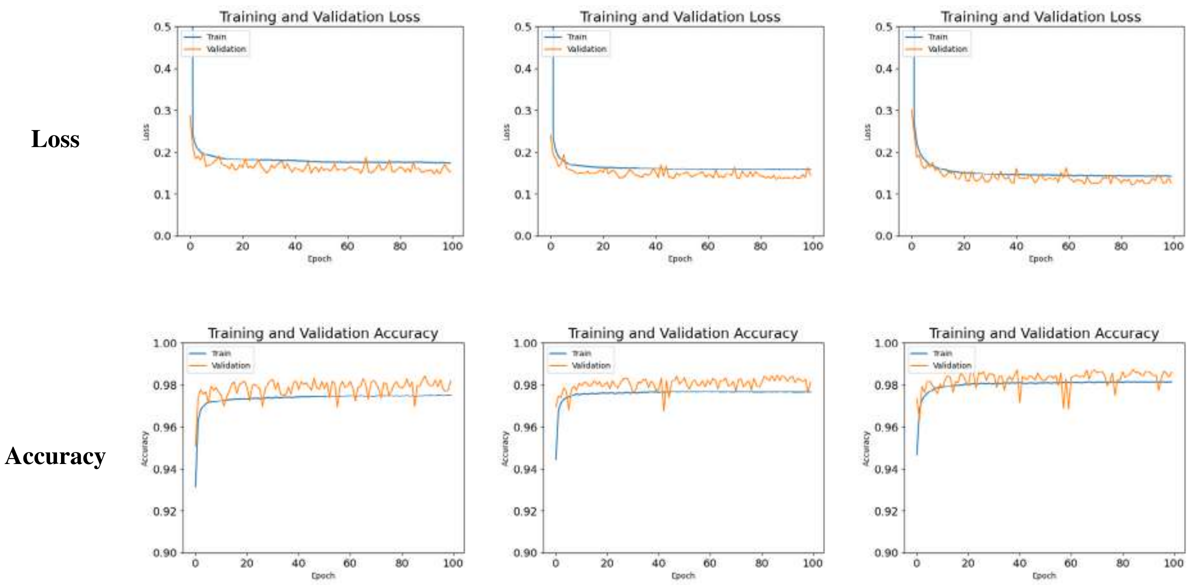
(b) LSTM 256-128-128 model plots for multiclass classification.

Fig. 15. Long Short-Term Memory plots of different batch sizes for multiclass classification.



(a) Batch Size 32 (b) Batch Size 64 (c) Batch Size 128

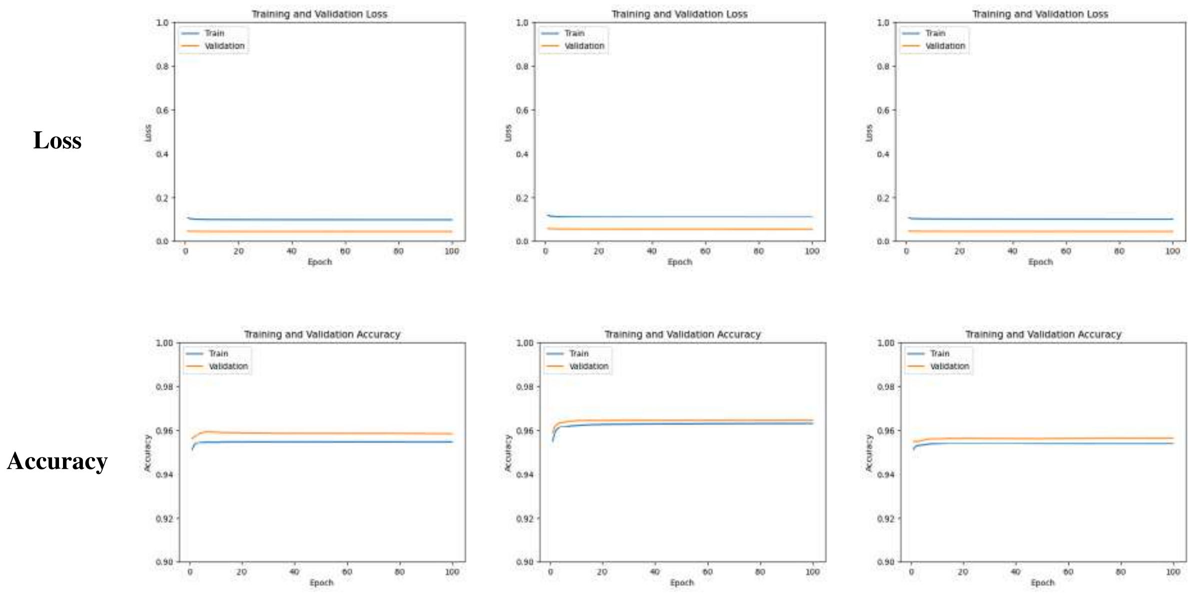
(a) LSTM 256-256-128 model plots for multiclass classification.



(a) Batch Size 32 (b) Batch Size 64 (c) Batch Size 128

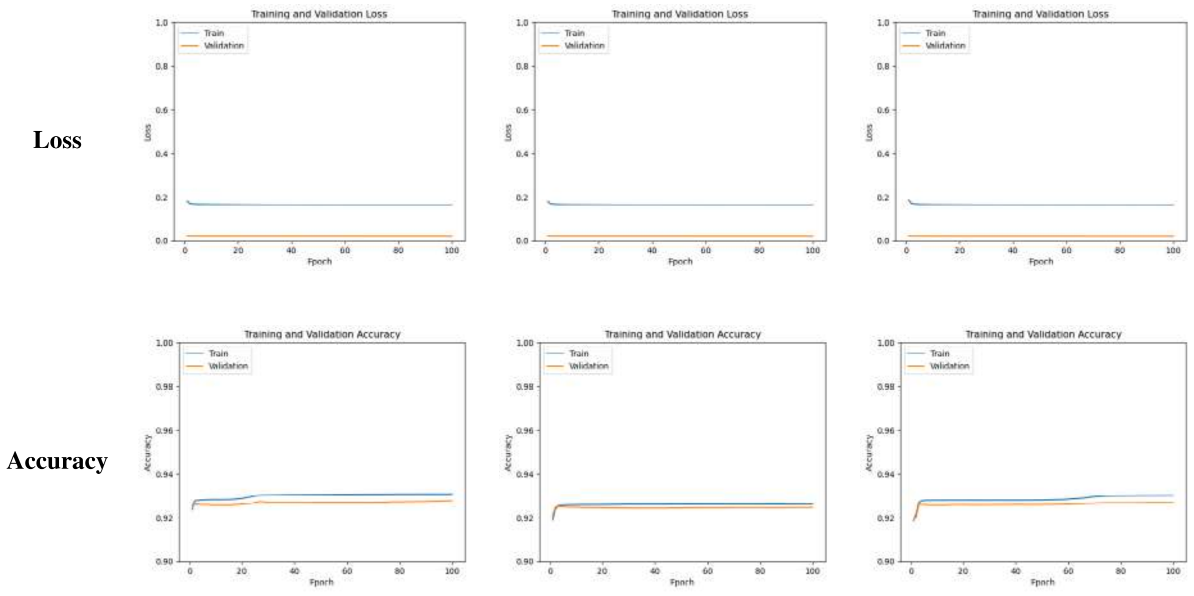
(b) LSTM 256-256-256 model plots for multiclass classification.

Fig. 16. Long Short-Term Memory plots of different batch sizes for multiclass classification.



(a) Learning Rate 1.0 (b) Learning Rate 0.1 (c) Learning Rate 0.01

(a) RandNN model plots for binary classification.



(a) Learning Rate 1.0 (b) Learning Rate 0.1 (c) Learning Rate 0.01

(b) RandNN model plots for multiclass classification.

Fig. 17. Random Neural Network plots of different learning rates for binary and multiclass classification.

Table 8
Specifications of Random Neural Network model layers.

Model	Binary class	Multi class	Input neuron	Hidden layer		Weight initialization	Output layer	
				Size	Learning rate		Size	Firing rate
RandNN	✓		65	4	1.0	0.2	2	0.1
RandNN	✓		65	4	0.1	0.2	2	0.1
RandNN	✓		65	4	0.01	0.2	2	0.1
RandNN		✓	67	6	1.0	0.2	8	0.1
RandNN		✓	67	6	0.1	0.2	8	0.1
RandNN		✓	67	6	0.01	0.2	8	0.1

Table 9
Evaluation metrics of Random Neural Network model for binary classification.

Model	LR	Training accuracy	Testing accuracy	Training time	Testing time	Precision	Recall	F1 score
RandNN	1.0	0.9548	0.9588	459 092.01	509.16	0.9588	0.9588	0.9588
RandNN	**0.1**	**0.9631**	**0.9642**	**461 375.44**	**509.04**	**0.9642**	**0.9642**	**0.9642**
RandNN	0.01	0.9538	0.9566	514 495.55	591.12	0.9567	0.9566	0.9566

Table 10
Evaluation metrics of Random Neural Network model for multiclass classification.

Model	LR	Training accuracy	Testing accuracy	Training time	Testing time	Precision	Recall	F1 score
RandNN	**1.0**	**0.9306**	**0.9289**	**922 337.77**	**628.72**	**0.8907**	**0.9289**	**0.9082**
RandNN	0.1	0.9262	0.9261	938 497.95	634.95	0.9167	0.9261	0.9054
RandNN	0.01	0.9301	0.9280	925 653.00	624.96	0.8899	0.9280	0.9073

NB is only 69.67% for binary class and 58.26% for multiclass, it is known for its ease of use, speed, and effectiveness with high-dimensional data. However, it makes the simplified assumption of features becoming independent, which may require improvement with interdependent features. The NB model has the lowest accuracy. The DT technique achieved 99.76% accuracy for binary classification and 98.33% for multiclass classification. However, it is subject to overfitting and needs assistance interpreting complex interconnections.

RF improved prediction and accuracy to 99.87% for binary and 98.52% for multiclass by combining several DT but at the cost of increased computation complexity and reduce ability to interpreted. Despite its simplicity and high accuracy of 99.75% for binary and 97.51% for multiclass, KNN is highly sensitive to computation costs, neighbor selection, and dimensionality. The NB approach is a scalable and computationally effective solution for high-dimensional datasets, whereas DT algorithms have overfitting issues. The RF approach reduced overfitting and enhanced model adaptability. The KNN technique performs exceptionally well in clustering-based classification applications. The issues of overfitting, dimensionality, and hyperparameter sensitivity can potentially lead to challenges. Although the RF achieves greater accuracy, execution takes a long time. As mentioned above, the limitations of the ML algorithms include the need for high quality training data, the risk of overfitting, the lack of interoperability and the dynamic adjustment of hyperparameters. Considering data privacy, algorithm reliability, and potential biases is critical while using these models.

Various architectures of Deep Neural Networks (DNN) are constructed with varying dense hidden layers, and neuron counts in each layer to enhance accuracy. The FFNN model is capable of estimating a wide variety of complex functions and nonlinear relationship between inputs and outputs features. Integrating FFNN improves performance and allows it to learn complex relationships using hyperparameter configurations. The optimization of the FFNN was achieved through the selection of neurons, hidden layers, and regularization strength. As compared to FFNN, LSTM layers performs better in handling sequential data due to the ability to simulate long-term dependencies and accurately capture complex temporal dynamics in IoTs. The weights initialization allow LSTM for a more adaptive and exploratory approach to model such relationships. Random connections in RandNN allow for a creative approach, revealing the network's hidden potential to capture complex dependencies that other network topologies might miss. The RandNN architecture achieves the best outcome, incorporating the number of hidden layers with different learning rates. Random weight initialization enables a more flexible approach to analyzing complex relationships. The results showed that the accuracy acquired from the CIC IoT 2022 dataset for detecting intrusions in the IoT network, are greater than those obtained from other datasets. Therefore, the CIC IoT 2022 dataset possesses a clearly defined architecture suited for complex IoT networks.

The comparative analysis of proposed models with state-of-the-art DL-based IDS models is shown in Table 11. To demonstrate the effectiveness of our proposed approach, we examined state-of-the-art DL models from recent years of literature. In [25], a trained neural network intrusion detection system with main focus on the loss function uses three datasets. Although the accuracy was 98.95%, but the F1 score was 68.48% using WUSTL-IIoT-2021 dataset. After training, the model did well in one class but not with other. In [28], a multi-head attention mechanism and feature extraction layer help Flow Transformer to evaluate complex traffic. The authors experimental results show that CNN, LSTM, or Autoencoder to Flow Transformer do not improve classifier performance. Traditional ML algorithms have limited performance with Flow Transformer when used for complicated network data. Using ensemble learning, [26] addresses the class imbalance problem in ML. The bagging classifier utilizes a DNN as its

Table 11
Comparison of the proposed deep learning models with state-of-the-art IDS.

Reference	Classifier	Dataset	Accuracy	Precision	Recall	F1 score
[25]	FNN-Focal CNN-Focal	WUSTL-IIoT-2021	0.9326 0.9308	0.9524 0.9423	0.737 0.734	0.8011 0.7963
	FNN-Focal CNN-Focal	WUSTL-EHMS-2020	0.9895 0.9821	0.7722 0.8854	0.641 0.665	0.6848 0.705
	FNN-Focal CNN-Focal	Bot-IoT	0.9155 0.8677	0.5559 0.6165	0.638 0.633	0.5784 0.5853
[27]	CNN + LSTM	UNSW-NB15	0.929	–	–	–
[24]	RaNN	ToN_IoT	0.9905	–	–	–
[34]	DNN GAN-DNN	UNSW-NB15	0.84 0.91	0.83 0.918	0.834 0.91	0.83 0.912
[26]	DNN	NSL-KDD UNSW-NB15 CIC -IDS-2017 BOT-IOT	0.989 0.967 0.9874 0.9899	0.999 0.989 0.9977 0.989	0.987 0.987 1.0 0.913	0.9931 0.9878 0.9986 0.9495
[32]	RNN CNN-BiLSTM LSTM CNN	CICIDS2017	0.9958 0.9976 0.9973 0.9882	0.987 0.989 0.911 0.977	0.983 0.992 0.935 0.983	0.985 0.991 0.991 0.98
[30]	DNN	DS2OS Traffic	0.949	–	–	–
[37]	FFNN LSTM	NSL-KDD	0.9867 0.9644	0.983 0.9574	0.992 0.977	– –
	FFNN LSTM	BoT-IoT	0.9997 0.9995	0.9995 0.999	1.0 1.0	– –
[36]	RNN	NSL-KDD	0.9218	0.9023	–	0.9029
[10]	RF XGB ET	IoTID20	0.9868 0.9867 0.9845	– – –	– – –	– – –
[35]	DCNN	IoTID20	0.9812 0.7755	0.9713 0.7876	0.978 0.734	0.9746 0.76
[18]	FCFFN	IoT	0.9374	0.9371	0.938	0.9347
[29]	Ensemble ANN CNN LSTM	IoT-23	0.996 0.969 0.97 0.982	– – – –	– – – –	– – – –
[41]	DT XGBOOST ANN XGBOOST LR XGBOOST KNN XGBOOST SVM XGBOOST	UNSW-NB15	0.9085 0.8439 0.7764 0.8446 0.6089	– – – – –	– – – – –	– – – – –
[43]	DAE + DNN	NSL-KDD	0.8333	0.8602	0.833	0.8333
	DAE-DNN	CSE-CIC-ID2018	0.9579	0.9538	0.958	0.9511
[28]	Flow Transformer	SJTU-AN21 ISCXVPN2016 CIC-IoT2022	0.86 0.952 0.985	0.868 0.953 0.985	– – –	0.855 0.952 0.984
Proposed	**FFNN** **LSTM** **RandNN**	**CIC-IoT2022**	**0.9993** **0.9989** **0.9642**	**0.9993** **0.9989** **0.9642**	**0.9993** **0.9989** **0.9642**	**0.9993** **0.9989** **0.9642**

base estimator. The proposed IDS performs better with ensemble bagging learning and balanced distribution with class weights. The proposed strategy outperforms class imbalance strategies in four historical intrusion detection datasets.

The widespread adoption of deep learning models can be attributed to their capacity to acquire complex patterns. In [27], CNN+LSTM based IDS is trained on the UNSW-NB15 dataset and obtained an accuracy of 93.21%. The author shows that an IIoT network achieved high detection accuracy but significant loss during training, validation, and testing. The author in [18] proposed a system performed by keeping RNNs in the right position in the network, thus reducing loss and training time. A DL-IDS for IoT devices covered five types of intrusion only and achieved an accuracy of only 93.74%. For the model to be network-adaptive, training of deep IDS is required before deployment. Therefore, each IoT network must be trained using a large dataset from existing IoT networks and adopt feature engineering to employ the model properly. The comparative analysis shows that the proposed FFNN outperformed well in terms of Accuracy, Precision, Recall and F1 score. In contrast to FFNN, LSTM architectures are recognized for their efficacy in sequential data processing. Moreover, RandNN possesses a randomized architecture because of the ability to

Table 12
Comparison of the proposed DL models with ML algorithms for binary classification.

Model	NB	DT	RF	KNN	FFNN	LSTM	RandNN
Accuracy	0.6967	0.9976	0.9987	0.9975	**0.9993**	0.9989	0.9642
Precision	0.7574	0.9976	0.9987	0.9975	**0.9993**	0.9989	0.9642
Recall	0.6967	0.9975	0.9987	0.9975	**0.9993**	0.9989	0.9642
F1 score	0.6778	0.9975	0.9987	0.9975	**0.9993**	0.9989	0.9642

Table 13
Comparison of the proposed DL models with ML algorithms for multiclass classification.

Model	NB	DT	RF	KNN	FFNN	LSTM	RandNN
Accuracy	0.5826	0.9833	0.9852	0.9751	**0.9872**	0.9860	0.9289
Precision	0.6362	0.9833	0.9851	0.9751	**0.9873**	0.9860	0.8907
Recall	0.5826	0.9833	0.9851	0.9751	**0.9872**	0.9860	0.9289
F1 score	0.5763	0.9832	0.9850	0.9751	**0.9872**	0.9860	0.9082

Table 14
Cross validation of machine learning algorithms for binary classification.

Cross validation	Evaluation metrics	Naive Bayes	Decision Tree	Random Forest	K Nearest Neighbor
Fold 1	Accuracy	0.6881	0.9970	0.9985	0.9972
	Precision	0.7454	0.9971	0.9984	0.9972
	Recall	0.6881	0.9970	0.9985	0.9972
	F1 score	0.6687	0.9970	0.9984	0.9972
Fold 2	Accuracy	0.6876	0.9978	0.9989	0.9975
	Precision	0.7450	0.9979	0.9989	0.9975
	Recall	0.6876	0.9980	0.9989	0.9975
	F1 score	0.6681	0.9979	0.9989	0.9974
Fold 3	Accuracy	0.6888	0.9972	0.9985	0.9976
	Precision	0.7514	0.9972	0.9985	0.9976
	Recall	0.6888	0.9972	0.9985	0.9976
	F1 score	0.6680	0.9972	0.9985	0.9976
Fold 4	Accuracy	0.6988	0.9971	0.9982	0.9978
	Precision	0.7514	0.9972	0.9981	0.9978
	Recall	0.6888	0.9972	0.9982	0.9978
	F1 score	0.6680	0.9972	0.9981	0.9978
Fold 5	Accuracy	0.6990	0.9875	0.9987	0.9975
	Precision	0.7630	0.9975	0.9987	0.9975
	Recall	0.6890	0.9975	0.9987	0.9975
	F1 score	0.6679	0.9974	0.9987	0.9975

handle nonlinear relationships and interactions between features effectively. However, acquiring large amounts of labeled data and computational resources is sometimes necessary. Conventional ML models are efficient, scalable, and easy to interpret, but only in certain circumstances. Researchers can benefit from an extensive review of DL and traditional ML tradeoffs. Researchers can use the study findings to understand better which framework will work best for their specific needs, data and available computing power.

6. Conclusion and future work

This paper demonstrates the efficacy of deep learning-based IDS employing FFNN, LSTM, and RandNN models in countering cyber threats in IoT environments. The RandNN model exhibits its potential due to random connections, thus facilitating a more exploratory approach, enabling the model to capture intricate dependencies and uncovering insights that might not be apparent in other architectures. The LSTM model shows strong suitability in addressing prolonged dependencies and effectively capturing temporal dynamics within the IoT data. Finally, the FFNN model has demonstrated superior results when compared with the proposed RandNN and LSTM models, and other state-of-the-art IDS techniques. Moreover, it has the ability to enhance the overall performance of an IDS by learning complex associations between input characteristics and output classifications. The proposed IDS framework enables the extraction and classification of features in a versatile manner, thereby facilitating the efficient identification of diverse cyber threats in IoT environments. The models can adjust to an IoT network's ever-changing and dynamic characteristics, thereby contributing to proactive mitigation of cyber threats. The results in this paper reflects compelling insights into the performance of various classifiers in effectively identifying a wide range of cyberattacks. The challenges associated with research in this field encompass several key aspects. These include the need for extensive and diverse datasets to ensure the optimal training of models and the requirement for computational resources capable of effectively managing the complexity of IoT data.

With the increase in security challenges encountered by IoT, there could be numerous areas for future research that can augment the security of IoT ecosystems. One prospective avenue for future investigation involves exploring the utilization of DL

Table 15
Cross validation of machine learning algorithms for multiclass classification.

Cross validation	Evaluation metrics	Naive Byes	Decision Tree	Random Forest	K Nearest Neighbors
Fold 1	Accuracy	0.5813	0.9833	0.9848	0.9749
	Precision	0.6347	0.9830	0.9847	0.9748
	Recall	0.5813	0.9833	0.9847	0.9749
	F1 score	0.5754	0.9832	0.9847	0.9748
Fold 2	Accuracy	0.5824	0.9835	0.9850	0.9754
	Precision	0.6355	0.9834	0.9850	0.9753
	Recall	0.5824	0.9834	0.9849	0.9754
	F1 score	0.5757	0.9833	0.9850	0.9753
Fold 3	Accuracy	0.5803	0.9832	0.9852	0.9751
	Precision	0.5745	0.9833	0.9852	0.9751
	Recall	0.5826	0.9833	0.9851	0.9751
	F1 score	0.6360	0.9833	0.9852	0.9750
Fold 4	Accuracy	0.5826	0.9833	0.9852	0.9753
	Precision	0.6360	0.9834	0.9851	0.9753
	Recall	0.5826	0.9835	0.9852	0.9753
	F1 score	0.5763	0.9832	0.9850	0.9753
Fold 5	Accuracy	0.5866	0.9833	0.9854	0.9750
	Precision	0.6394	0.9834	0.9854	0.9750
	Recall	0.5866	0.9833	0.9854	0.9750
	F1 score	0.5798	0.9833	0.9854	0.9750

methodologies in the development of more resilient and sophisticated IDS for IoT. This can help to detect and prevent advanced intrusion attempts with increased precision and effectiveness through the utilization of DL algorithms, such as Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and Transformers. Moreover, incorporating federated learning techniques, which enable collaborative and privacy-preserving training of models on IoT devices can enhance intrusion detection capabilities by incorporating scalability and diversity. Ensemble techniques and hybrid architectures can be studied for better intrusion detection and advanced cyber threats. In addition, it is imperative to investigate adversarial defense mechanisms to safeguard DL-based IDS against emerging attack vectors. Exploring these areas could lead to a substantial contribution to achieve IoT security and enhancing the overall reliability of IoT implementations in the foreseeable future.

Declaration of competing interest

The authors have no conflicts of interest to declare. All coauthors have seen and agree with the contents of the manuscript and there is no conflict of interest to report. We certify that the submission is original work and is not under review at any other publication.

Data availability

Data will be made available on request.

Acknowledgment

The authors are thankful to the Deanship of Scientific Research at Najran University for funding this work under the Research Groups Funding program grant code (NU/RG/SERC/12/3).

References

[1] A. Djenna, S. Harous, D.E. Saidouni, Internet of things meet internet of threats: New concern cyber security issues of critical cyber infrastructure, Appl. Sci. 11 (10) (2021) 4580.
[2] E. Sisinni, A. Saifullah, S. Han, U. Jennehag, M. Gidlund, Industrial internet of things: Challenges, opportunities, and directions, IEEE Trans. Ind. Inform. 14 (11) (2018) 4724–4734, http://dx.doi.org/10.1109/TII.2018.2852491.
[3] L.S. Vailshery, Statista, 2022, [Online]. Available: https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/, Accessed on May 20, 2023.
[4] Y. Lu, L. Da Xu, Internet of things (IoT) cybersecurity research: A review of current research topics, IEEE Internet Things J. 6 (2) (2018) 2103–2115.
[5] A.R. Khan, M. Kashif, R.H. Jhaveri, R. Raut, T. Saba, S.A. Bahaj, Deep learning for intrusion detection and security of Internet of things (IoT): current analysis, challenges, and possible solutions, Secur. Commun. Netw. 2022 (2022).
[6] I. Ullah, Q.H. Mahmoud, Design and development of RNN anomaly detection model for IoT networks, IEEE Access 10 (2022) 62722–62750, http://dx.doi.org/10.1109/ACCESS.2022.3176317.
[7] P. Sethi, S.R. Sarangi, Internet of things: architectures, protocols, and applications, J. Electr. Comput. Eng. 2017 (2017).
[8] N. Abosata, S. Al-Rubaye, G. Inalhan, C. Emmanouilidis, Internet of things for system integrity: A comprehensive survey on security, attacks and countermeasures for industrial applications, Sensors 21 (11) (2021) 3654.
[9] M. Burhan, R.A. Rehman, B. Khan, B.-S. Kim, IoT elements, layered architectures and security issues: A comprehensive survey, sensors 18 (9) (2018) 2796.

[10] S. Bajpai, K. Sharma, B.K. Chaurasia, Intrusion detection framework in IoT networks, SN Comput. Sci. 4 (4) (2023) 350.
[11] M.H.P. Rizi, S.A.H. Seno, A systematic review of technologies and solutions to improve security and privacy protection of citizens in the smart city, Internet Things (2022) 100584.
[12] D. Nanthiya, P. Keerthika, S. Gopal, S. Kayalvizhi, T. Raja, R.S. Priya, SVM based ddos attack detection in IoT using Iot-23 botnet dataset, in: 2021 Innovations in Power and Advanced Computing Technologies (I-PACT), IEEE, 2021, pp. 1–7.
[13] M. Ibrahim, A. Continella, A. Bianchi, Aot-attack on things: A security analysis of IoT firmware updates, in: 2023 IEEE 8th European Symposium on Security and Privacy (EuroS&P), 2023.
[14] T. Suleski, M. Ahmed, W. Yang, E. Wang, A review of multi-factor authentication in the Internet of Healthcare Things, Digit. Health 9 (2023) 20552076231177144.
[15] M.I. Zakaria, M.N. Norizan, M.M. Isa, M.F. Jamlos, M. Mustapa, Comparative analysis on virtual private network in the internet of things gateways, Indones. J. Electr. Eng. Comput. Sci. 28 (1) (2022) 488–497.
[16] A. Javadpour, P. Pinto, F. Ja'fari, W. Zhang, DMAIDPS: a distributed multi-agent intrusion detection and prevention system for cloud IoT environments, Cluster Comput. 26 (1) (2023) 367–384.
[17] A. Khraisat, A. Alazab, A critical review of intrusion detection systems in the internet of things: techniques, deployment strategy, validation strategy, attacks, public datasets and challenges, Cybersecurity 4 (2021) 1–27.
[18] A. Awajan, A novel deep learning-based intrusion detection system for IoT networks, Computers 12 (2) (2023) 34.
[19] N. Yadav, S. Pande, A. Khamparia, D. Gupta, Intrusion detection system on IoT with 5G network using deep learning, Wirel. Commun. Mob. Comput. 2022 (2022) 1–13.
[20] M. Abdullahi, Y. Baashar, H. Alhussian, A. Alwadain, N. Aziz, L.F. Capretz, S.J. Abdulkadir, Detecting cybersecurity attacks in internet of things using artificial intelligence methods: A systematic literature review, Electronics 11 (2) (2022) 198.
[21] T. Saba, A. Rehman, T. Sadad, H. Kolivand, S.A. Bahaj, Anomaly-based intrusion detection system for IoT networks through deep learning model, Comput. Electr. Eng. 99 (2022) 107810.
[22] H. Asgharzadeh, A. Ghaffari, M. Masdari, F.S. Gharehchopogh, Anomaly-based intrusion detection system in the Internet of Things using a convolutional neural network and multi-objective enhanced capuchin search algorithm, J. Parallel Distrib. Comput. (2023).
[23] S.M. Kasongo, A deep learning technique for intrusion detection system using a Recurrent Neural Networks based framework, Comput. Commun. 199 (2023) 113–125.
[24] S. Latif, Z. e Huma, S.S. Jamal, F. Ahmed, J. Ahmad, A. Zahid, K. Dashtipour, M.U. Aftab, M. Ahmad, Q.H. Abbasi, Intrusion detection framework for the internet of things using a dense random neural network, IEEE Trans. Ind. Inform. 18 (9) (2021) 6435–6444.
[25] A.S. Dina, A. Siddique, D. Manivannan, A deep learning approach for intrusion detection in Internet of Things using focal loss function, Internet Things (2023) 100699.
[26] A. Thakkar, R. Lohiya, Attack classification of imbalanced intrusion data for IoT network using ensemble learning-based deep neural network, IEEE Internet Things J. (2023).
[27] H.C. Altunay, Z. Albayrak, A hybrid CNN+ LSTMbased intrusion detection system for industrial IoT networks, Eng. Sci. Technol. Int. J. 38 (2023) 101322.
[28] R. Zhao, Y. Huang, X. Deng, Y. Shi, J. Li, Z. Huang, Y. Wang, Z. Xue, A novel traffic classifier with attention mechanism for industrial internet of things, IEEE Trans. Ind. Inform. (2023) 1–12, http://dx.doi.org/10.1109/TII.2023.3241689.
[29] R. Alghamdi, M. Bellaiche, An ensemble deep learning based IDS for IoT using Lambda architecture, Cybersecurity 6 (1) (2023) 5.
[30] A. Abusitta, G.H. de Carvalho, O.A. Wahab, T. Halabi, B.C. Fung, S. Al Mamoori, Deep learning-enabled anomaly detection for IoT systems, Internet Things 21 (2023) 100656.
[31] F. Alrowais, S. Althahabi, S.S. Alotaibi, A. Mohamed, M.A. Hamza, R. Marzouk, Automated machine learning enabled cybersecurity threat detection in internet of things environment, Comput. Syst. Sci. Eng. 45 (1) (2023) 687–700.
[32] F.M. Aswad, A.M.S. Ahmed, N.A.M. Alhammadi, B.A. Khalaf, S.A. Mostafa, Deep learning in distributed denial-of-service attacks detection method for internet of things networks, J. Intell. Syst. 32 (1) (2023).
[33] A. Yazdinejad, M. Kazemi, R.M. Parizi, A. Dehghantanha, H. Karimipour, An ensemble deep learning model for cyber threat hunting in industrial internet of things, Digit. Commun. Netw. 9 (1) (2023) 101–110.
[34] B. Sharma, L. Sharma, C. Lal, S. Roy, Anomaly based network intrusion detection for IoT attacks using deep learning technique, Comput. Electr. Eng. 107 (2023) 108626.
[35] S. Ullah, J. Ahmad, M.A. Khan, E.H. Alkhammash, M. Hadjouni, Y.Y. Ghadi, F. Saeed, N. Pitropakis, A new intrusion detection system for the internet of things via deep convolutional neural network and feature engineering, Sensors 22 (10) (2022) 3607.
[36] M. Almiani, A. AbuGhazleh, A. Al-Rahayfeh, S. Atiewi, A. Razaque, Deep recurrent neural network for IoT intrusion detection system, Simul. Model. Pract. Theory 101 (2020) 102031.
[37] O. Jullian, B. Otero, E. Rodriguez, N. Gutierrez, H. Antona, R. Canal, Deep-learning based detection for cyber-attacks in IoT networks: A distributed attack detection framework, J. Netw. Syst. Manage. 31 (2) (2023) 33.
[38] Y.K. Saheed, A.I. Abiodun, S. Misra, M.K. Holone, R. Colomo-Palacios, A machine learning-based intrusion detection for detecting internet of things network attacks, Alex. Eng. J. 61 (12) (2022) 9395–9409.
[39] S. Dadkhah, H. Mahdikhani, P.K. Danso, A. Zohourian, K.A. Truong, A.A. Ghorbani, Towards the development of a realistic multidimensional IoT profiling dataset, in: 2022 19th Annual International Conference on Privacy, Security & Trust (PST), IEEE, 2022, pp. 1–11.
[40] A.H. Lashkari, G. Draper-Gil, M.S.I. Mamun, A.A. Ghorbani, et al., Characterization of tor traffic using time based features, in: ICISSp, 2017, pp. 253–262.
[41] S.M. Kasongo, Y. Sun, Performance analysis of intrusion detection systems using a feature selection method on the UNSW-NB15 dataset, J. Big Data 7 (2020) 1–20.
[42] J.T. Hancock, T.M. Khoshgoftaar, Survey on categorical data for neural networks, J. Big Data 7 (1) (2020) 1–41.
[43] Y.N. Kunang, S. Nurmaini, D. Stiawan, B.Y. Suprapto, Attack classification of an intrusion detection system using deep learning and hyperparameter optimization, J. Inf. Secur. Appl. 58 (2021) 102804.
[44] D. Elreedy, A.F. Atiya, A comprehensive analysis of synthetic minority oversampling technique (SMOTE) for handling class imbalance, Inform. Sci. 505 (2019) 32–64.
[45] R. Abdulhammed, M. Faezipour, H. Musafer, A. Abuzneid, Efficient network intrusion detection using pca-based dimensionality reduction of features, in: 2019 International Symposium on Networks, Computers and Communications (ISNCC), IEEE, 2019, pp. 1–6.
[46] S. Bhattacharya, P.K.R. Maddikunta, R. Kaluri, S. Singh, T.R. Gadekallu, M. Alazab, U. Tariq, A novel PCA-firefly based xgboost classification model for intrusion detection in networks using GPU, Electronics 9 (2) (2020) 219.
[47] S. Dadkhah, H. Mahdikhani, P.K. Danso, A. Zohourian, K.A. Truong, A.A. Ghorbani, Towards the development of a realistic multidimensional IoT profiling dataset, in: 2022 19th Annual International Conference on Privacy, Security & Trust (PST), 2022, pp. 1–11, http://dx.doi.org/10.1109/PST55820.2022.9851966.
[48] A. Shimokawa, Codeberg, 2022, [Online]. Available: https://codeberg.org/iortega/TCPDUMP_and_CICFlowMeter/src/branch/master/CICFlowMeters/CICFlowMeter-4.0/, Accessed on May 18, 2023.
[49] G. Draper-Gil, A.H. Lashkari, M.S.I. Mamun, A.A. Ghorbani, Characterization of encrypted and vpn traffic using time-related, in: Proceedings of the 2nd International Conference on Information Systems Security and Privacy (ICISSP), 2016, pp. 407–414.

[50] T. Kluyver, B. Ragan-Kelley, F. Pérez, B.E. Granger, M. Bussonnier, J. Frederic, K. Kelley, J.B. Hamrick, J. Grout, S. Corlay, et al., Jupyter Notebooks-A Publishing Format for Reproducible Computational Workflows, Vol. 2016, 2016.
[51] A. Géron, Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow, O'Reilly Media, Inc., 2022.
[52] N. Silaparasetty, Machine Learning Concepts with Python and the Jupyter Notebook Environment: Using Tensorflow 2.0, Springer, 2020.
[53] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2014, arXiv preprint arXiv:1412.6980.
[54] G. Arulampalam, A. Bouzerdoum, A generalized feedforward neural network architecture for classification and regression, Neural Netw. 16 (5–6) (2003) 561–568.
[55] Z. Zhang, M. Sabuncu, Generalized cross entropy loss for training deep neural networks with noisy labels, Adv. Neural Inf. Process. Syst. 31 (2018).
[56] E. Gelenbe, Random neural networks with negative and positive signals and product form solution, Neural Comput. 1 (4) (1989) 502–510.
[57] K.M. Leung, et al., Naive bayesian classifier, Polytech. Univ. Dep. Comput. Sci./Financ. Risk Eng. 2007 (2007) 123–156.
[58] Y.-Y. Song, L. Ying, Decision tree methods: applications for classification and prediction, Shanghai Arch. Psychiatry 27 (2) (2015) 130.
[59] G. Biau, E. Scornet, A random forest guided tour, Test 25 (2016) 197–227.
[60] P. Cunningham, S.J. Delany, K-nearest neighbour classifiers-A tutorial, ACM Comput. Surv. (CSUR) 54 (6) (2021) 1–25.