Mathematical Biosciences and Engineering

http://www.aimspress.com/journal/mbe

MBE, 20(8): 13491–13520.
DOI: 10.3934/mbe.2023602
Received: 12 April 2023
Revised: 03 May 2023
Accepted: 21 May 2023
Published: 13 June 2023

Research article

A hybrid deep learning-based intrusion detection system for IoT networks

Noor Wali Khan[1], Mohammed S. Alshehri[2], Muazzam A Khan[1,3], Sultan Almakdi[2], Naghmeh Moradpoor[4], Abdulwahab Alazeb[2], Safi Ullah[1], Naila Naz[1] and Jawad Ahmad[4,*]

[1] Department of Computer Science, Quaid-i-Azam University, Islamabad 44000, Pakistan

[2] Department of Computer Science, College of Computer Science and Information Systems, Najran University, Najran 61441, Saudi Arabia

[3] ICESCO Chair Big Data Analytics and Edge Computing, Quaid-i-Azam University, Islamabad 44000, Pakistan

[4] School of Computing, Engineering & The Built Environment, Edinburgh Napier University, Edinburgh EH10 5DT, UK

* **Correspondence:** Email: J.Ahmad@napier.ac.uk.

Abstract: The Internet of Things (IoT) is a rapidly evolving technology with a wide range of potential applications, but the security of IoT networks remains a major concern. The existing system needs improvement in detecting intrusions in IoT networks. Several researchers have focused on intrusion detection systems (IDS) that address only one layer of the three-layered IoT architecture, which limits their effectiveness in detecting attacks across the entire network. To address these limitations, this paper proposes an intelligent IDS for IoT networks based on deep learning algorithms. The proposed model consists of a recurrent neural network and gated recurrent units (RNN-GRU), which can classify attacks across the physical, network, and application layers. The proposed model is trained and tested using the ToN-IoT dataset, specifically collected for a three-layered IoT system, and includes new types of attacks compared to other publicly available datasets. The performance analysis of the proposed model was carried out by a number of evaluation metrics such as accuracy, precision, recall, and F1-measure. Two optimization techniques, Adam and Adamax, were applied in the evaluation process of the model, and the Adam performance was found to be optimal. Moreover, the proposed model was compared with various advanced deep learning (DL) and traditional machine learning (ML) techniques. The results show that the proposed system achieves an accuracy of 99% for network flow datasets and 98% for application layer datasets, demonstrating its superiority over previous IDS models.

Keywords: IoT; intrusion detection; deep learning; machine learnnig; cyberattacks

1. Introduction

The Internet of Things (IoT) is a network of interconnected physical devices that are embedded with sensors, software, and other technologies that enable them to collect and exchange data over the Internet. IoT is a revolutionary technology that has a signicantlly positive impact on human lives [1, 2]. IoT applications are surprisingly becoming more prevalent in our society, including smart grids, smart retail, smart homes, smart cities, and smart health care [2–4]. The IoT is increasingly being used in a variety of products, including wearables and smart homes, with the aim of enhancing job productivity, living comfort, and entertainment [5]. The IoT architecture consists of three layers (including perception, network, and application layer), as shown in Figure 1. The perception layer is the first layer, and it consists of sensors that track, record or measure environmental factors related to physical events in an environment, temperature, moisture, or levels of pollutants. The network layer is built on top of the perception layer and comprises a variety of switches and routers that facilitate connectivity, enabling the transmission of information shared by various smart devices. The application is the last layer that consists of systems for processing and collecting data from detected objects in order to produce meaningful information that may be used to make decisions. Using this architecture, a number of essential applications have been developed, including smart grids, mobility, distribution channels, agriculture, and healthcare systems [6–8].

It is always necessary to secure these systems in order to prevent cyber attacks that can lead to service disruption, unauthorized access, data tampering, and illegal access [9]. Several researchers have focused on intrusion detection systems (IDS) by utilizing different machine learning (ML) and deep learning (DL) algorithms to detect malicious activities in IoT networks [10].

However, the existing systems are focused on a single layer of the three-layered IoT architecture, which limits their effectiveness in detecting attacks across the entire network. Moreover, the existing systems still need improvements in the detection process of cyberattacks. To overcome these limitations, this paper proposes an intrusion detection framework for IoT networks that covers all three layers in IoT architectures (including the perception layer, network layer, and application layer). The proposed hybrid model consists of a recurrent neural network and gated recurrent units (RNN-GRU), which can analyze sequential data effectively, making it a suitable choice for intrusion detection in IoT environments. In addition, this work utilized data pre-processing techniques such as cleaning, encoding, feature filtering, shuffling, and normalization. The significance of the proposed model is further reinforced by utilizing the ToN-IoT dataset, which was specifically collected for the three-layered IoT architecture. The following are the major contributions of this work:

- This paper proposes a hybrid deep learning intelligent intrusion detection framework for IoT networks. The hybrid model consists of RNN and GRU algorithms.
- This study covers all three layers in the IoT architecture (including perception layer, network layer, and application layer).
- Compared the performance of the proposed scheme with other existing advanced DL and traditional ML techniques.

The remainder of the article is organized as follows. Section 2 discusses IoT layered architecture. Section 3 presents intrusions in IoT networks. IDS and its sub-types are presented in Section 4. Section 5 discusses related work and presents a literature comparison. A step-by-step methodology of the

proposed system is presented in Section 6. Section 7 provides a detailed analysis of the results and a comparison with state-of-the-art models, and also presents the discussion. This work is concluded in Section 8.

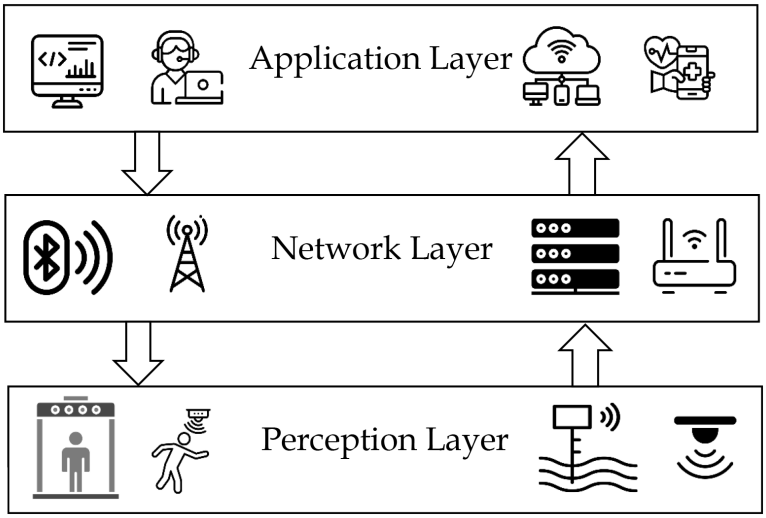

Figure 1. IoT three layers architecture.

2. IoT architecture

IoT networks require a flexible and layered design to link various devices, but there is no universal reference model for IoT architecture. The 3-layered IoT architecture is widely accepted and used in the literature, defining the IoT architecture in terms of the perception, network, and application layers. It provides a clear and concise framework for designing IoT systems that is easier to understand than a 5-layered architecture. However, the choice of architecture depends on the specific use case and requirements of the system being designed. Figure 1 illustrates the three-layer architecture of IoT, consisting of the perception layer, network layer, and application layer. The perception layer consists of sensors, actuators, and other devices that interact with the physical environment. The network layer is responsible for connecting the devices in the perception layer to the cloud or other back-end systems. The application layer is where the data is processed, analyzed, and used to make decisions. The importance of this figure in the present work is that it provides a high-level overview of the IoT architecture, which helps in understanding the data flow and identifying potential attack surfaces in the network. However, there are two more layers: The processing layer and business layer, in a five-layer IoT model [11]. The IoT five-layer architecture is depicted in Figure 2, this figure is important because it provides a more detailed view of the IoT network structure than the three-layer architecture. It helps readers to understand the different layers of the IoT network and how they interact with each other, each layer explains as follows.

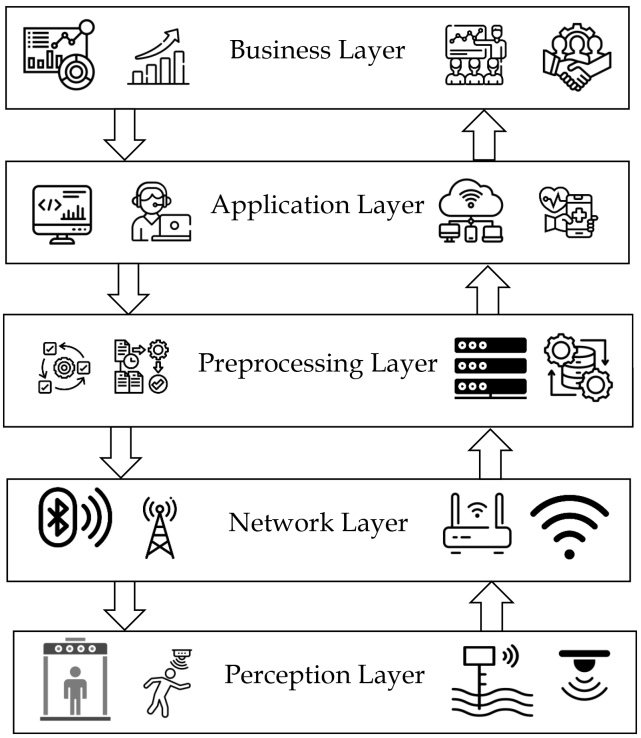

Figure 2. IoT five layered architecture.

2.1. Perception layer

The perception layer has a physical/hardware layer in an IoT architecture which includes real objects of various sizes and forms. This involves components such as sensors, controllers, actuators, and machines [12]. Information gathering, information processing, information storage, and object identification are all handled by this layer. Naming and addressing IoT objects are done by this layer of an IoT model. The devices in the IoT are tracked using digital identities. A Universal Unique Identifier (UUID) is one way to give an object a special identity. Similar addressing techniques, such as IPv6 and IPv4, are utilized to address objects in an IoT network [13].

2.2. Network layer

For transmitting data to processing systems, such as servers, the network layer of an IoT model utilizes a variety of communication methods among physical smart devices. This communication could only be done via secure connections through cables or wireless. This layer can handle a number of technology types and can be extended for the transfer of sensor data. Several communication methods for IoT devices are presented in the existing literature, this includes Wi-Fi, Bluetooth's specialized range, ZigBee for small areas, and 4G and 5G for broad IoT environments [14].

2.3. Preprocessing layer

Middleware is another name for processing layer in an IoT model. Significant amounts of data are sent to this layer via the network layer. After receiving the data, processing layer, as the name suggests, evaluates, analyses, and saves the data in the associated cloud database. Network attacks at this layer affect the security of databases and the cloud. The most well-known threats at this layer are SQL injection (SQLi) attacks, browser attacks, and flooding attacks.

2.4. Application layer

The application layer of an IoT model serves as a global supervisor of the IoT application-specific services. Home automation, intelligent agriculture, smart cities, intelligent healthcare, and smart industry are a few examples of IoT applications that the application layer deals with.

2.5. Business layer

The creation of reliable and effective business models, the management of business regulations, and the deployment of optimal investment are some of the major responsibilities of the business layer in an IoT network. This is based on the fact that business models are created by evaluating the effectiveness of current applications and services. The use of technology and the ways in which services are presented to customers/clients may hence be used to measure the success of the IoT system.

3. Intrusion in IoT network

An IoT-based network still suffers from various problems, such as security issues, that prevent such systems from becoming widely used. Intrusion in IoT networks is a significant concern due to the widespread use of IoT devices in various domains such as healthcare, transportation, smart cities, and industrial automation. These devices are interconnected, and they continuously generate, process, and transmit sensitive data. However, the inherent vulnerabilities of IoT devices and the lack of security measures make them an easy target for cyber attackers.

IoT devices have limited processing power and memory, which makes it difficult to implement traditional security mechanisms such as firewalls, IDS, and antivirus software. Moreover, IoT devices often operate in hostile environments, such as outdoor locations and factories, which make them more susceptible to physical attacks. The combination of these factors makes it challenging to secure IoT devices from intrusions. Security vulnerabilities can be associated with each layer in IoT three/five-layer architecture models.

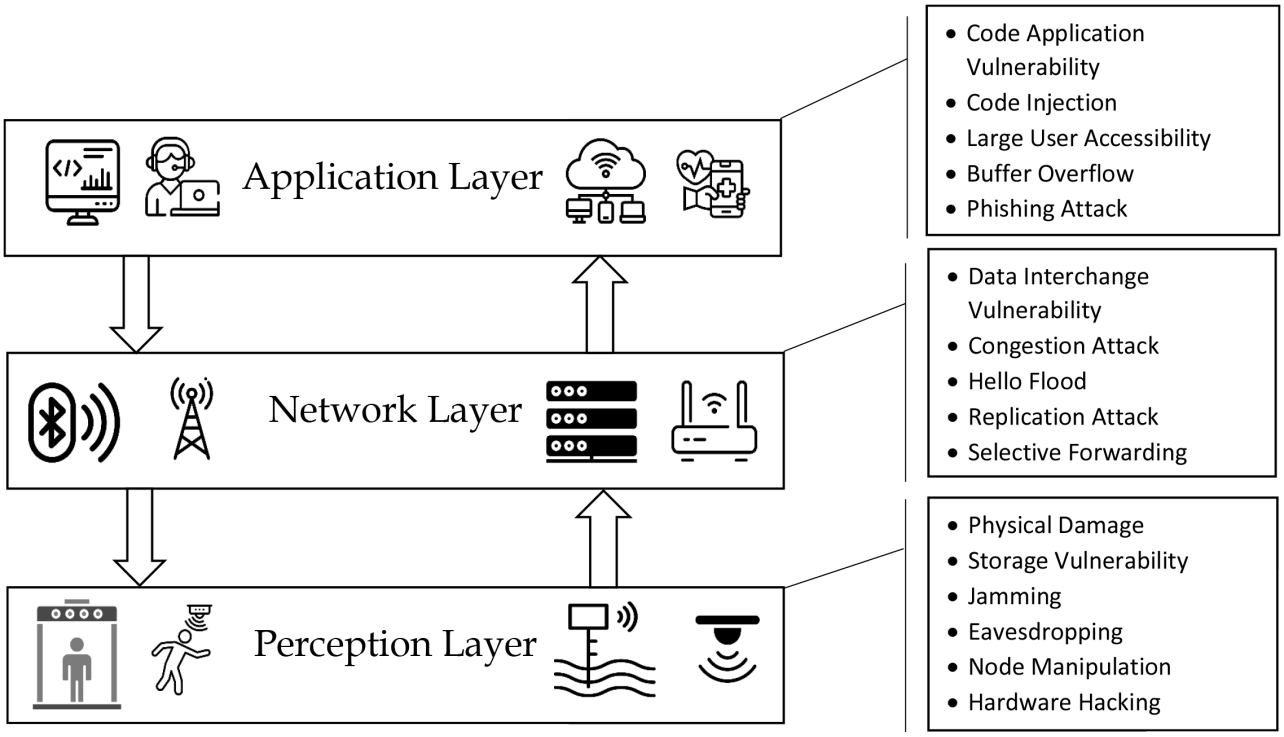

Figure 3. Attacks at layers of IoT.

3.1. Intrusion in IoT Layers

Several attacks associated with different layers of an IoT network are depicted in Figure 3. This figure illustrates the different types of intrusions that can occur in each layer of the IoT network. The perception layer can be attacked by physical tampering, the network layer can be attacked by eavesdropping, denial of service (DoS) attacks, and routing attacks, and the application layer can be attacked by malware and data manipulation attacks. These are presented based on IoT three-layer architecture but can be easily extended to five-layer architecture. For example, attacks such as those stemming from application layer vulnerabilities, e.g., code injection and buffer overflow, network layer vulnerabilities, e.g., congestion and black hole attacks, and perception layer vulnerabilities, e.g., jamming and eavesdropping attacks, are explained as follows.

3.1.1. Application layer attacks

This layer operates on the information received from the network layer in order to deliver user services. Data integrity, data dependability, customer confidentiality, and the capacity to safeguard critical/private/sensitive information are security concerns associated with this layer. As the name suggests, attacks on this layer are application-specific which will have a severe impact on the application programs. The application layer threats include phishing assaults, buffer overflows, manipulation of confidential data, and authentication and authorization attacks [12].

- Code injection: This attack involves injecting malicious content into a system by taking advantage of programming mistakes. In order to launch a code injection attack, scanning is frequently

employed to obtain details about a target system. Attackers take advantage of the public nature of IoT software updates which will lead them to insert malicious code into such devices and take control of an IoT network as a whole. Code injection can be employed for a number of reasons such as data theft, gaining unauthorized access to a system, and worm propagation [15–17].

- Cross-site scripting (XSS): It is a cyber-attack in which a web application's weaknesses are hunted for inserting a malicious script. This suggests that phishing, cookie theft, or network attacks against an entire company might all have an impact on user information [18].
- Buffer overflow: Since a predetermined memory layout is used by many applications to store information and code chunks [16, 17]. In this attack, hackers used a system's security vulnerabilities to violate the limits of the system's code and data block. Buffer overflow is one of the most frequent attacks on applications and software. As an example, an automated production program called WelinTech KinView 6.53 HistrySvr was vulnerable to heap buffer overflow [19].
- Phishing attack: In these attacks, an intruder poses as a genuine user or reputable organization to steal personal/sensitive/critical information using email is phone calls are the most popular tools used for such attacks [17, 20]. DDoS, password, or scanning attacks can be used to enable phishing attacks, although they do not really start phishing attacks.
- Authentication and authorization: The protection of an IoT network's confidentiality and safety depends heavily on the authentication procedure [16]. An attacker can carry out illegal actions, like opening a door in a smart home or stealing your bank account details using Amazon Alexa due to the absence of a flawless authentication system [21].

3.1.2. Network layer attacks

In an IoT network devices connect to the data processing center via the network layer. The intermediate layer of an IoT system, which is also the layer with the greatest vulnerabilities, plays an important role in information coordination. There are various attacks attached to this layer such as wormhole, replication, selective forwarding, hello flood, and black hole attacks. The network layer vulnerabilities and countermeasures have received the most attention in the literature. Some of the attacks associated with this layer are detailed as follows.

- DoS attack: A denial-of-service (DoS)/ distributed denial-of-service (DDoS) attack on a network involves bombarding the target with queries, which creates a lot of communications overhead [16, 17]. This kind of attack will exhaust the victim node and will lead to the legitimate users being unable to access network services [16].
- Sinkhole attack: A sinkhole attack in an IoT network layer is a type of denial-of-service (DoS) attack. In this attack, the hacker used the compromised node to draw data/information from its surrounding nodes [16, 22]. An intruder may deploy a rogue node in such a network to capture network activities and then manipulate sensor information afterward [22].
- Sniffing attack: As the name suggests, in this attack hackers gather network data via sniffer sensors and programs to retrieve important/sensitive/critical information from the network [16, 23].
- Man-in-the-Middle attack: Actual attacks of this kind happen when two susceptible nodes are in communication. In order to interact between two target terminals, the attacker poses as a valid node on one of the nodes [16, 23, 24].
- Replay attack: Attackers listen in on conversations between the sender and receiver to get data.

The transmission couples periodically send the incoming signal, depleting the available communication sources In this attack, hackers eavesdrop on the conversations between the sender and the receiver to steal user information. The transmission couples periodically send the incoming signal, draining the available resources [24, 25], such as computing power and backend database capacity [25]. It can be caused by various types of attacks, including DoS, XSS, and scanning.

3.1.3. Perception layer attacks

Since this layer contains sensors and other data collection devices, sometimes it is also called sensor layer. It's crucial to provide secure connectivity for the devices in this layer. These devices, the majority of which are wireless, are more complicated and have greater potential for security vulnerabilities. This makes it challenging to apply the typical internet security rules to them. The various attacks in this layer include jamming, eavesdropping, node manipulation, and hardware hacking [12, 15].

- Malicious node injection: An attacker can carry out a malicious node injection attack by physically inserting a new malicious node among a group of nodes. The attacker then alters the data and sends false information to other nodes. This type of attack is executed by the attacker through multiple nodes [26].
- Physical damage: Physical damage to IoT system components can also lead to a denial of service (DoS) attack. The attacker may physically access the device, power source, or data flow between devices to cause damage [26].
- Eavesdropping: In this attack, hackers intercept the communication by listening in on the devices and the physical layer nodes [16, 24]. For instance, unauthorized users may utilize an antenna to capture information sent between two parties [12, 25]. A password attack can lead to an eavesdropping attack in the IoT perception layer. An attacker who is able to access IoT devices without authorization can listen in on communications if the passwords are easily guessable or crack-able.
- Tag cloning: In RFID tag cloning, hackers copy the information of an RFID electronic tag or smart card to a cloned tag. The clone tag will then have the same characteristics as the compromised tag. An attacker can achieve this by querying it is deploying infrastructure or by using debugging to gather necessary data [16, 24, 25]. Tag cloning attack in the IoT perception layer is caused by a scanning attack.
- Spoofing attack: In this type of attack, hackers impersonate a legitimate tag in order to gain similar access. It can be triggered by various attacks, such as DDOS, password, XSS, etc. [12, 16, 24]. Additionally, this kind of attack would force nodes to resend their data, thus ramping up network packets. Additionally, it speeds up the node's energy usage which leads to shortening the node's lifespan [12].

4. Intrusion detection system

Intrusion detection systems (IDS) are frequently used to maintain track of network activity, identify malicious behaviour, evaluate its seriousness, and alert the administrator. People who steal data or cause intrusions are referred to as intruders. The main invaders are Misfeasor (legitimate user/insider), Masquerader (unauthorized user/outsider), and clandestine User

(insider/outsider) [1, 2, 27]. The existing literature has described various types of IDS based on different criteria such as information source, analysis process, detection technique, learning method, and knowledge base. Figure 4 illustrates the different types of intrusion detection systems (IDS) that can be used to detect network intrusions in the IoT network. The types of IDS include signature-based IDS, anomaly-based IDS, and hybrid IDS.

4.1. Information source based IDS

Host-based, network-based, and hybrid-based IDS are the three primary categories of IDS depending on information sources [28]. When implemented on a local host, the host-based intrusion detection system (HIDS) may monitor and find intrusions. It relies on the file system, system calls, and network events of the host to look for intrusions and notifies the system when the intruder tries to engage in any harmful action. This type of IDS is only able to monitor attacks against the system that they are installed on. Network-based intrusion detection system (NIDS) finds attacks by examining network-related information such as network packets, broadcast domains of network protocols, traffic volume, and IP addresses. A hybrid intrusion detection system (HIDS) is a combination of multiple IDS types. The main goal in implementing HIDS is to achieve a real-time intrusion detection mechanism [2, 29].

4.2. Detection based IDS

In principle, the two primary methods of intrusion detection systems are signature-based and anomaly-based.

4.2.1. Anomaly based intrusion detection

An anomaly-based IDS also referred to as behaviour-based detection, mimics the behaviour of users, computer systems, and networks. The intrusion detection system uses a variety of ways to build a baseline from typical behavior patterns. Any major departure of the examined pattern from the norm is marked as an anomaly. The term "intrusion" refers to a device's activity that deviates from its usual behaviour [2, 29, 30].

4.2.2. Signature based intrusion detection

An intrusion detection system that uses different signatures for anomaly detection is called signature-based IDS. As the name implies, a signature-based IDS uses a database to record the signatures or patterns of known attacks. To create a signature, the packet's properties are extracted. When an activity is detected, the IDS looks at its signature and compares it to the patterns that have already been saved in the database. If the pattern of the captured network packet matches any known signatures an alarm is set out to the system administrator. Although signature-based IDS are highly good at seeing known attacks, it is ineffective at spotting unknown attacks (also known as zero-day attacks/threats) [2, 30].

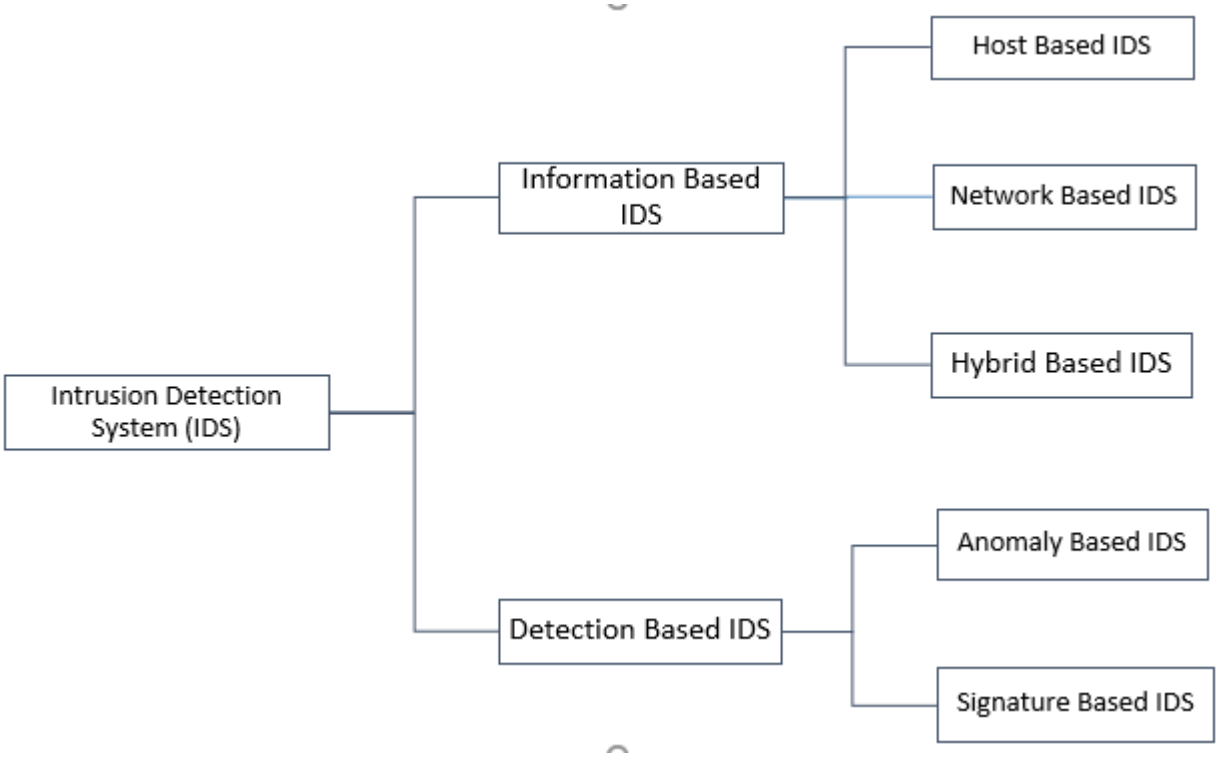

Figure 4. Classification of intrusion detection systems.

IDS is essential in an attack detection process. When such behavior is found, the administrator may also receive notifications. Various ML and DL techniques may be applied to effectively manage and categorize intrusions. Modified K-Means, support vector machine (SVM), decision tree (DT) [31], principle component analysis (PCA) [32], logistic regression (LR), random forest (RF) [33], linear discriminant analysis (LDA) [34, 35], and artificial neural network (ANN) [35]. Additionally, there are classifier ensembles such as extreme gradient boosting (XGB) [36], gradient boosted machine (GBM), and extremely randomized trees (ETC) [37]. These are a few of the ML methods utilized for IDSs. Similarly, DL approaches such as convolutional neural network (CNN) [38, 39], recurrent neural network (RNN) [39–43], long short term memory (LSTM) [39, 43, 44], and gated recurrent unit (GRU) [39, 42, 43, 45] are also used in IDSs. However, the DL technique has outperformed ML, especially for large datasets. Due to their support for high-dimensional characteristics, DL algorithms are appropriate for certain IoT devices that generate significant amounts of data. Additionally, DL algorithms perform better than machine learning algorithms for rapid and accurate automatic identifying of unwanted attacks. Although the major objectives of DL IoT data processing are security and privacy, it has also been extensively employed for smart city applications, big data analytics, video processing, voice processing, and image processing [46, 47]. Moreover, deep learning models outperform classical machine learning techniques in terms of performance and accuracy because they can independently extract features from large volumes of data [44].

Recent studies have also shown the effectiveness of deep learning algorithms, such as convolutional neural networks (CNNs) [38,39] and recurrent neural networks (RNNs) [39–42], for intrusion detection

in IoT networks. However, to the best of our knowledge, there is limited research on the use of a hybrid of RNN and GRU for intrusion detection in IoT networks. Our proposed model combines the strengths of these two deep learning algorithms and achieves superior performance in detecting different types of attacks in IoT networks. As we utilized the hybrid of two deep learning models, RNN and GRU, both have the ability to process sequential data in real-time, which is important in the context of IoT systems where data is generated in real-time and in a continuous stream. These models can analyze patterns in the data and detect anomalies in real-time, which is critical for timely responses to potential security threats. Additionally the use of data engineering techniques (cleaning, normalization) can further improve the performance of the model and make it more practical in real-time applications. These techniques can help reduce noise in the data, improve model accuracy, and speed up model training and prediction.

An RNN model is created to determine the sequential properties of data and then use the patterns to forecast future events. The RNN models have an internal memory that constantly recalls the results of calculations made in earlier phases. Any length of input may be handled by RNN. The model size stays the same, regardless of how big the input is. RNNs' internal memory, which enables them to remember important data about the input they received, is what allows them to predict the future with such accuracy. As a result, they are the preferred algorithm for sequential data, including time series, speech, text, financial data, audio, video, and a wide variety of other forms. The vanishing gradient problem is a significant flaw in the RNN model. LSTM and GRU are used to tackle the RNN's vanishing gradient problem. To establish whether the technique is more effective at resolving the vanishing gradient problem, LSTM and GRU are compared for the initial phase of developing the model in this study. The findings indicate that the accuracy of LSTM and GRU is about equal. However, compared to the model with LSTM, the model with GRU required fewer training iterations and less time for RNN to converge. RNN with GRU was consequently more appropriate for real-time scenarios. GRU employs what is referred to as the update gate and reset gate to address the RNN's problem with vanishing gradients and also the response time of GRU is fewer [48]. This paper proposes a novel approach to intrusion detection in IoT networks using a hybrid of deep learning and intelligent intrusion detection framework that utilizes RNN and GRU algorithms. This approach can analyze sequential data, making it suitable for intrusion detection in IoT environments where data is often collected sequentially. Additionally, we explain the data pre-processing techniques we used, including cleaning, encoding, feature filtering, shuffling, and normalization, which improve the model's performance.

The proposed model aims to categorize attacks across all layers of the 3-layered IoT architecture, namely the physical, network, and application layers. Intrusion detection in IoT requires analyzing data from all layers to detect any potential threats. To validate the effectiveness of the proposed model, we conducted experiments on the ToN-IoT dataset, which was specifically gathered for the three-layered IoT architecture, making our results more reliable and applicable in real-world scenarios. Our results demonstrate that the proposed hybrid model outperforms the baseline models in terms of accuracy, precision, and recall, providing better security for IoT environments. This contribution is significant for industries that rely on IoT devices, such as healthcare, finance, and logistics, as well as for the broader security of critical infrastructure. Overall, our contribution is the proposal of a novel and effective intrusion detection model that covers all layers of the IoT architecture, analyzes sequential data, and employs hybrid deep learning algorithms.

5. Related works

IoT has received a lot of academic attention across the world. IoT applications are surprisingly becoming more prevalent in all spheres of society, including smart grids, smart retail, smart homes, smart cities, and health care [7]. It is always necessary to secure these systems to prevent service disruption, unauthorized access, and possible cyberattacks such as tampering, illegal access, or others in order to guarantee adequate accuracy of the data and the appropriate operation of the processes in them. The development of IDS frequently employs machine learning and deep learning (ML/DL) [43]. An intrusion detection system (IDS) is a security tool used to identify and stop unauthorized network access or attacks. An IDS is used in the context of IoT networks to shield connected devices and systems from potential threats that can jeopardize their security and functioning. It is impossible to overestimate the significance of IDS in IoT networks, which are increasingly widespread in our daily lives and frequently involved in vital infrastructure like healthcare, transportation, and energy systems. Network administrators can utilize intrusion detection systems to discover security risks in real time, respond to them, and get crucial information that can be used to stop and mitigate possible attacks.

The creation of intrusion detection systems (IDS) for Internet of Things (IoT) networks has become more and more common in recent years. These algorithms have shown to be successful at identifying patterns and anomalies in network traffic that might point to a possible intrusion. Such algorithms' use for intrusion detection in IoT networks has been addressed in a number of studies. To investigate the security aspects of IoT networks different ML algorithms like DT, RF, NB, etc., and DL algorithms such as LSTM, RNN, CNN, GRU were proposed [54] which we cover in the subsequent paragraphs. However, the DL approach has performed better in terms of accuracy than ML, particularly for huge datasets. Researchers creating ML algorithms must be clever and only extract features that can enhance the model since choosing features takes a lot of time and they won't know which features are valuable until they train and test their model. Thus, working with datasets with various dimensions is a problem for ML because it is not always easy to extract the most predictive features [49]. Furthermore, deep learning models are more accurate and perform better than traditional machine learning methods since they can independently extract characteristics from huge amounts of data [44].

Authors in [7] utilized an ensemble-based voting classifier, where the final forecast was obtained by combining the conventional ML algorithm with voting on its predictions. A study in [8] suggest that IoT devices become more capable of spotting anomalies in IoT networks when they use a stacking-based ensemble model. To increase the ensemble-based IDS's effectiveness and precision, [27] utilized ensemble-based IDS with XGBoost, which improves the accuracy.

In [50], authors discussed an overview of several machine learning techniques that can be employed to spot any harmful or out-of-the-ordinary data and present the most effective approach for two datasets, the first dataset was created from data exchanged between sensors and the second is UNSW-NB15. In [37], they examine the potential of machine learning classifying methods to protect their IoT from DoS attacks. The Classifiers are evaluated using well-known datasets such as CIDDS-001, UNSW-NB15, and NSL-KDD. To achieve ML features for cyber-security, including IoT, some cyber security experts have modified deep learning components. Aside from that, DL can develop new features that allow for problem-solving without the involvement of a person [51].

Table 1. Literature comparison.

Reference	Model/ Attack Detection Mechanism	Dataset	IoT Layer Targeted	Evaluation Measures
[7]	Ensemble Voting Classifier	ToN-IoT Telemetry Dataset	IoT Perception Layer	Accuracy, Precision, Recall, F1-Score
[8]	Stacking based Ensemble Model	ToN-IoT Telemetry Dataset	IoT Perception Layer	Accuracy, Precision, Recall, F1-Score
[27]	Ensemble Based IDS using XGBoost	KDDCup99	IoT Network Layer	Accuracy, Precision, Recall, F1-Score, Confusion Matrix
[37]	ML Classifiers (RF, AB, XGB, GBM, ETC)	CIDDS-001, UNSW-NB 15, NSL-KDD	IoT Network Layer	Accuracy, False positive Rate, AUC, Sensitivity
[53]	Deep Convolutional Neural Network	IODIT20	IoT Perception Layer	Accuracy, Precision, Recall, F1-Score
[54]	DL-IDS(SMO with SDPN)	NSL-KDD	IoT Network Layer	Accuracy, Precision, Recall, F1-Score
[53]	Hybrid DL Based Model (LSTM-GRU)	CIC DOS, CI-CIDS 2017 and CSE-CIC IDS 2018	X	Accuracy, Precision, Recall, F1-Score
[45]	ReliefF-Based IDS (MNN, WKNN, SVM)	ToN-IoT Windows 10 Dataset	IoT Application Layer	Confusion Matrix, Accuracy, Precision, Recall, F1-Score
[57]	DT , NB	TON-IoT Network Flow Dataset	IoT Network Layer	Confusion Matrix
[58]	GraphSage Based IDS	ToN-IoT Network Flow Dataset	IoT Network Layer	Accuracy, Precision, Recall, F1-Score
[59]	DFF, CNN, RNN, DT, LR, NB	UNSW-NB15, CSE CIC-IDS2018, ToN-IoT Network Flow	IoT Network Layer	Accuracy, F1-Score, Detection Rate, False Alarm Rate
[60]	CNN -Based IDS	ToN-IoT Network Flow Dataset	IoT Network Layer	Accuracy, Precision, Recall, Loss, F1-Score
Proposed Study	**RNN-GRU**	**ToN-IoT (Telemetry dataset, Network Flow dataset, windows 7 &10 dataset)**	**IoT Perception Layer IoT Network Layer IoT Application Layer**	**Accuracy, Precision, Recall, F1-Score**

Models:- RF - Random Forest. AB - Adaa Boost. XGB- Extreme Gradient Boost.
GBM - Gradient Boosted Machine. ETC - Extremely Randomized Trees.
SMO - Spider Monkey Optimization. SDPN - Stacked Deep Polynomial Network.
LSTM- Long Short-Term Memory. GRU - Gated Recurrent Unit.
MNN- Medium Neural Network. WKNN- Weighted KNN. SVM - Support Vector Machine.

Authors in [52] presented a DL technique to find anomalies on IoT nodes in a household environment. In order to boost efficiency while using less computing power an IDS based on a deep convolutional neural network is proposed in [53]. In order to identify malicious attacks in IoT contexts, [54] presents a novel DL-based intrusion detection system to address the issues of protecting IoT nodes. The spider monkey optimization (SMO) algorithm and the stacked deep polynomial network (SDPN) are combined in their suggested model to obtain the best detection and recognized rates. SMO chooses the best attributes from the datasets, and SDPN categorises the output as normal or abnormal. Authors in [55] presents a novel ReliefF-IDS that uses ML and DL methods to uncover new attacks that existing systems are unable to prevent in an IoT network. [56] recommended using DL to identify intrusions using recurrent neural networks (RNN-IDS). They show through their experimental results that RNN-IDS is ideally suited for producing IDS with good accuracy and that it outperforms conventional ML both binary and multi-class techniques.

We are committed to studying various intrusion detection systems, or IDS, in-depth in this work in order to identify unusual activity in IoT networks at various layers. Although previous studies have proposed ML and DL algorithms for intrusion detection in IoT networks, they have not considered the three-layered architecture of IoT, and have focused on only one layer of the IoT architecture and have not addressed the challenges of detecting intrusions across multiple layers, which is the focus of our proposed hybrid approach. Moreover, previous studies have either focused on ML or DL, but not a combination of both, as proposed in our work. Therefore, our proposed approach fills the gap in the literature by addressing the shortcomings of the previous studies. Additionally, we can provide a

comparative analysis of the performance of our proposed hybrid approach with the previous studies. This will help to further demonstrate the superiority of our proposed approach. We proposed a hybrid and intelligent intrusion detection framework by using two DL algorithms: RNN and GRU which outperforms the baseline models. We conclude this section with a tabular review of the most current and pertinent findings that have been published in the literature and are shown in Table 1.

6. The proposed framework

IDS employs various approaches and frameworks to detect attacks, which are the essential elements of modern cyber-security techniques. IoT IDS has recently benefited from advancements in artificial intelligence (AI), including ML and DL approaches [61]. The RNN has proven to perform well with a sequence of data that has different input lengths. It remembers the past and therefore centers its decisions on what it has learned from the past. This indicates that RNNs utilize the data from their previous state as a source for their most recent prediction. It is possible to continue this procedure for an unlimited number of iterations in order to enable a network to transmit data across time using its hidden state [56].

RNN is the earliest algorithm that remembers its input because of its internal memory, which makes it perfect for machine learning problems requiring sequential data. The vanishing gradient problem, a significant flaw in the RNN model, hinders it to be correct. LSTM and GRU are used to tackle the RNN's vanishing gradient problem. GRU employs what is referred to as an update gate and reset gate to address the conventional RNN's problem with vanishing gradients. Additionally, in comparison with RNN, the response time of GRU is less. GRUs are an enhanced form of the traditional recurrent neural network (RNN) [48]. To establish which technique is more effective in resolving the vanishing gradient problem, LSTM and GRU are compared for the initial phase of the proposed model development in this study. The findings indicate that the accuracy of LSTM and GRU is about equal. However, compared to the model with LSTM, the model with GRU required fewer training iterations and less time for RNNs to converge. RNN with GRU was consequently more appropriate for real-time scenarios. Therefore, we proposed a three-layer hybrid deep learning model by combining RNN and GRU. We first initialize our Sequential model before including the layers. RNN is the first layer, dense with activation function relu is the second layer, and GRU with softmax as the activation function is the third layer that we added to our model as shown in Figure 5. This figure illustrates the proposed model for intrusion detection in the IoT network. It shows the different steps involved in the proposed model, including preprocessing the dataset, splitting the dataset into training and testing datasets, training the model using RNN, Dense, and GRU layers, and testing the model using the testing dataset. This figure is important because it provides readers with an overview of the proposed model and the steps involved in the model.

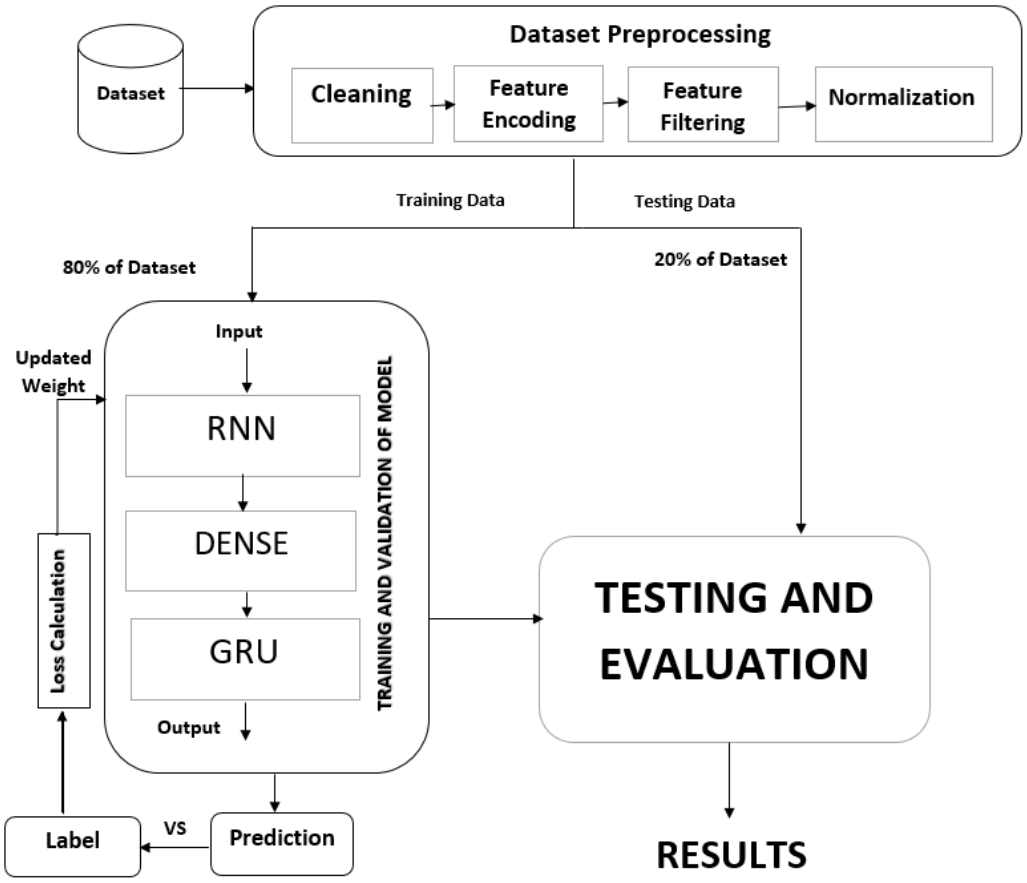

Figure 5. Proposed model.

The architecture of the RNN consists of a sequence of repeating neural network modules that operate on a sequence of inputs. At each time step t, the RNN takes an input x_t and a hidden state h_t from the previous time step and produces an output o_t and a new hidden state h_{t+1}. This process is mathematically represented in Eqs (6.1) and (6.2).

$$h_{t+1} = f(W_{xh} * x_t + W_{hh} * h_t + b) \tag{6.1}$$

$$o_t = g(W_{oh} * h_{t+1} + b_o) \tag{6.2}$$

where W_xh, W_hh, and W_{oh} are the weight matrices for the input-to-hidden, hidden-to-hidden, and hidden-to-output connections, respectively, b and b_o are the bias terms. f and g are the activation functions used in the hidden and output layers, respectively. The dense layer connects all the neurons of the previous layer to the current layer. The activation function used in the dense layer is ReLU, which is defined in Eq (6.3). The output of the dense layer is represented mathematically in Eq (6.4).

$$f(x) = max(0, x) \tag{6.3}$$

$$h = f(W_h * x + b) \tag{6.4}$$

where W_h is the weight matrix for the dense layer, x is the input to the dense layer, and b is biased. The architecture of the GRU is similar to that of the RNN, but it uses a gating mechanism that allows the model to selectively update or forget information from the previous time step. The GRU has two gates, a reset gate and an update gate, which control the flow of information. The activation function used in the GRU is softmax, which is used to convert the output of the GRU into a probability distribution over the possible outputs. The GRU operations are mathematically represented in Eqs (6.5)–(6.8).

$$z = sigmoid(W_{xz} * x + W_{hz} * h + b_z) \tag{6.5}$$

$$r = sigmoid(W_{xr} * x + W_{hr} * h + b_r) \tag{6.6}$$

$$h' = tanh(W_{xh} * x + W_{hh} * (r * h) + b_h) \tag{6.7}$$

$$h = z * h + (1 - z) * h' \tag{6.8}$$

where W_{xz}, W_{hz}, W_{xr}, W_{hr}, W_{xh}, W_{hh} are the weight matrices for the input-to-update gate, hidden-to-update gate, input-to-reset gate, hidden-to-reset gate, input-to-hidden, and hidden-to-hidden connections, respectively. b_x, b_h, b_r, and b_z are the bias terms.

The input data to the model is a sequence of features that are extracted from the IoT network traffic. These features can include source and destination IP addresses, port numbers, packet size, and protocol type. Before feeding the data into the model, it is preprocessed, including normalization and scaling. The data is then divided into training, validation, and testing datasets.

During the model's training process, the weights are updated using backpropagation, which leverages gradient descent to minimize the loss function. The loss function evaluates the disparity between the predicted output and the actual output. The gradients are calculated with respect to the weights, and the weights are updated using the Adam optimization function. This can be achieved by analyzing data from each of the different layers and utilizing this information to identify patterns and anomalies that could signify an intrusion. The workflow of the model is as follows:

- To find patterns and anomalies, the first layer, RNN evaluates sequential data from the physical, network, and application layers.
- The output of the first layer is processed by the second layer, a dense layer, that learns features from it and uses them to identify patterns and anomalies.
- In order to further enhance the detection of patterns and anomalies, the third and final layer, GRU, analyzes the output of the second layer and learns long-term dependencies in the data.

The number of inputs to these layers varies depending on the features of the datasets for the three layers of the IoT network. For instance, in the telemetry dataset, we remove the date, time, and timestamp, and feed the remaining four features to the model. In the network dataset, we feed the source port, destination port, IP address, and other features to the model. Similarly, for the OS log files (such as Windows 10 or Windows 7), features are selected using the ReliefF method and provided to the model.

6.1. Dataset

The dataset is created by assembling data from a variety of sources. IoT-based datasets are important for assessing IoT security and several researchers utilize them for their experiments. Many datasets are provided in the existing research for measuring the performance of IDS. For example, DARPA (1998), KDD-CUP99 (1998), NSL-KDD (2009), BOT-IOT(2018), CIC-IDS (2017), UNSWNB (2015), ISCX (2012), CIC-IDS (2017), and TON-IoT (2020). The existing datasets don't adequately capture crucial IoT/IIoT properties. For example, many IoT/IIoT services do not have a variety of normal and attack events, nor do they incorporate heterogeneous data sources. Furthermore, these datasets are not intended for three-layer architecture, in particular, [1, 62]. We used TON-IOT datasets for the evaluation because this dataset is specifically designed for three-layer architecture. The data in this dataset was acquired from the telemetry data of IoT/IIoT services, Windows and Linux operating system datasets, and network traffic datasets. They were created under a variety of attack scenarios. Moreover, the data within these datasets are distinct from one another. This is because the collected datasets are coming from many sources [1, 2]. The utilized TON_IoT dataset statistics is shown in the Table 2.

The TON IoT dataset was created by the cyber range and IoT labs at UNSW Canberra using a realistic model of a regular-size network in Australia. It comprises telemetry data of IoT/IIoT services, network traffic, and pperating systems logs of the IoT network. The comparison of the Ton-IoT dataset is presented in Table 3.

Table 2. Dataset statistics.

IoT Layers	Dataset Files	Total Record	Normal Instances	Scanning Attack	DOS / DDOS Attack	Ransom-ware Attack	Backdoor Attack	Injection Attack	XSS Attack	Password Attack	Total Attacks
Physical Layer	IoT Fridge Sensor	59,944	35,000	0	5000	2902	5000	5000	2042	5000	24,944
Physical Layer	IoT Garage Door Sensor	59,587	35,000	529	5000	2902	5000	5000	1156	5000	24,587
Physical Layer	IoT GPS Tracker	58,960	35,000	550	5000	2833	5000	5000	577	5000	23,960
Physical Layer	IoT Modbus Sensor	51,106	35,000	529	0	0	5000	5000	577	5000	16,106
Physical Layer	IoT Motion Light Sensor	59,488	35,000	1775	5000	2264	5000	5000	449	5000	24,488
Physical Layer	IoT Thermostat Sensor	52,774	35,000	61	0	2264	5000	5000	449	5000	17,774
Physical Layer	IoT Weather Sensor	59,260	35,000	529	5000	2865	5000	5000	866	5000	24,260
Network layer	Network Flow	461,043	300,000	18,615	19,986	13,738	18,710	19,964	14,516	19,847	161,043
Application Layer	windows 10	21,104	10,000	0	4608	15	447	612	1269	3628	11,104
Application Layer	windows 7	15,980	10,000	226	2134	82	1779	998	4	757	5980

Table 3. Comparison of the proposed dataset with existing state of art dataset.

Dataset	Attack Labels	Diverse Attack Scenarios	Separate Dataset for Each IoT Layer	Year
KDDCUP99	Yes	No	No	1998
NSL-KDD	Yes	No	No	1998
LWSNDR	Yes	No	No	2010
UNSW-NB15	Yes	Yes	No	2015
AWID	Yes	Yes	No	2015
ISCX	Yes	Yes	No	2017
UNSW-IOT	Yes	Yes	No	2018
BOT-IOT	Yes	Yes	No	2019
RPL-NIDDS17	Yes	Yes	No	2017
CICIDS2017	Yes	Yes	No	2017
ToN-IoT	**Yes**	**Yes**	**Yes**	2020

6.2. Preprocessing of datasets

Before applying any machine learning approach to the dataset, it is essential to preprocess the raw data to ensure high accuracy and improve the machine learning process. In this study, we performed several preprocessing steps on the ToN-IoT dataset, including removing duplicates, handling missing values, and scaling the data to ensure all features are on the same scale. These steps were applied to ensure the dataset is appropriately formatted and suitable for training and testing our proposed mode. To prepare the ToN-IoT dataset for the evaluation of the proposed model, we go through the following steps:

- **Cleaning:** Before training an ML model, a dataset must be examined for blank and undefinable entries. Any records with empty or undefinable values will cause problems during model training. There are several methods for cleaning datasets with empty and undefinable entries. In this experiment, we employed the ToN-IoT datasets that included physical layer/telemetry, network traffic, and OS logs. We leveraged Python libraries, including Pandas and NumPy, to verify the datasets for missing and/or infinite values as part of the cleaning process. Boolean functions were used to check for missing and/or unending values. False indicates that the dataset is clean, whereas true indicates the presence of missing or infinite values. The first layer of the dataset did not contain any empty or undefined entries, whereas the second and third layers of the datasets had missing value records. To address the issue of missing values, we filled in the missing values using the median of each feature.
- **Features encoding:** Feature encoding is the process of transforming categorical attributes into numerical ones. The label encoding (LE) technique is employed where each categorical attribute value is converted to numeric values.
- **Features filtering:** Each dataset has a different set of features, and certain features may cause over-fitting or under-fitting, leading to a negative impact on the model's performance. Therefore, some features should be excluded from the dataset. Another issue is the time complexity

associated with these features, particularly for datasets with numerous features, including insignificant ones that cannot affect the output label. For feature selection, there are primarily three techniques: intrinsic, wrapper, and filter methods. For the telemetry dataset, we remove the date, time, and timestamp features. According to [62], these three features may contribute to the issue of over-fitting while the dataset is being trained using ML algorithms. Timestamp, source port, destination port, and IP address were removed from the network layer data of the dataset because they had the potential to cause over-fitting of ML algorithms during the training phase. For the OS log files, such as Windows 10 or Windows 7, the features were selected using the ReliefF method. The Windows 7 dataset contained 133 features, including two additional class-label features for either normal or attack data. The twenty-five most important features were selected using the ReliefF method. Similarly, the Windows 10 dataset contained 125 features along with the class label and attack type parameters [62]. Twenty important features were selected for the Windows 10 dataset using the ReliefF method, which were then fed into the model.

- **Balancing strategies:** Each dataset comprises numerous records. We need to ensure that each class has the same amount of occurrences or small changes before training and testing the model. Under-sampling and over-sampling are the two main strategies for balancing the number of instances in each class. For balancing the network dataset and OS log files dataset, we utilized synthetic minority oversampling technique (SMOTE) method [63, 64]. This is applied to attack classes. The SMOTE technique eliminates the over-fitting problem by offering artificial minority class samples. For the scientific community in unbalanced classification, SMOTE pre-processing approach became a pioneer. In ML and data mining, SMOTE is regarded as one of the most important data pre-processing/sampling methods [63].
- **Normalization:** Scaling, mapping, and other pre-processing techniques are examples of normalization. It involves transforming the data to create a new data range based on an existing one. Normalization can be quite beneficial for prediction or forecasting purposes as it helps to remove any variations or biases in the data, making it easier for the model to learn the underlying patterns and relationships [65]. In the normalization process, data is adjusted to fit within a specific range, typically between 0 and 1. If the dataset already has values in a similar range, normalization may not be necessary. However, for datasets with values that are vastly different in range, such as one feature having values between 0 and 1 while another has values between 100 and 1000, normalization becomes crucial to ensure the model is properly trained. One of the most commonly used techniques for normalization is the Min-Max Scaler approach [66]. In this paper, we standardized values between 0 and 1 using this approach to scale our research data.

7. Results and discussion

The experimental findings and comparisons of the suggested model with state-of-the-art approaches in several layers are presented in this section.

7.1. Experimental setup

The proposed model's effectiveness was tested on a machine used for testing that had an Intel (R) Core (TM) i5 CPU with a clock speed of 3.20 GHz, 8 GB of RAM, and was running on the Windows

10 Pro operating system. The model was implemented using Python 3.0 programming language in VScode and Jupyter notebook with ipynb support. VScode is a popular code editor, and Jupyter notebook is an interactive computing environment that allows for easy experimentation with code. The fact that the model outputs results after each cell of the code is important because it allows for easy debugging and troubleshooting. TensorFlow's Keras package was used for implementing the model. Keras is a high-level neural network API that makes it easy to build and train deep learning models. The use of Keras simplifies the implementation of the model and can save time and effort.

7.2. Evaluation procedure

True positive (TP), true negative (TN), false positive (FP), and false negative (FN) values are used to evaluate the performance of a model. In addition to these values, existing research has used precision, recall, F1-score, and accuracy to evaluate the performance of the model. Therefore, we have used all these metrics to evaluate our proposed model and compare our results with existing work.

7.3. Results

Our proposed model was tested through three experiments. The first experiment focused on the Telemetry or physical layer dataset. The second experiment examined the network traffic or network layer dataset of ToN-IoT, while the third experiment evaluated the OS log files of both Windows 10 and Windows 7 datasets. The model was trained over ten epochs, utilizing the binary-cross entropy function for loss. We used two optimization functions including Adam and Adamax optimizers to test the proposed model.

7.3.1. Binary classification results on perception layer data of ToN-IoT Dataset

The Perception layer plays a critical role in gathering data from a variety of IoT devices and sensors. To assess the performance of the proposed model for binary classification of cyberattacks, we utilized the Perception layer data of the ToN-IoT dataset, which includes data from seven different IoT devices. To evaluate the proposed model's performance on heterogeneous sensor data, we combined the seven individual sensor data into a single CSV file, and the results for the merged file are shown alongside the individual files in Table 4. The results indicate that Adam performs better than Adamax. Table 5 compares the binary classification results of the proposed model with the most recent approaches. According to the results, the proposed Hybrid model outperforms existing models for the majority of the data files from individual IoT sensors.

Table 4. Binary classification results on IoT perception layer data files.

Dataset Files	Optimizer	Accuracy	Precision	Recall	F-Measure
IoT Fridge Sensor	Adam	1.0	1.0	1.0	1.0
	Adamax	1.0	1.0	1.0	1.0
IoT Garage Door Sensor	Adam	1.0	1.0	1.0	1.0
	Adamax	1.0	1.0	1.0	1.0
IoT GPS Tracker Sensor	Adam	1.0	1.0	1.0	1.0
	Adamax	1.0	1.0	1.0	1.0
IoT Modbus Sensor	Adam	0.85	0.85	0.8	0.78
	Adamax	0.83	0.82	0.79	0.76
IoT Motion Light Sensor	Adam	0.8	0.8	0.78	0.8
	Adamax	0.75	0.78	0.75	0.74
IoT Thermostat Sensor	Adam	0.98	0.99	0.98	0.98
	Adamax	0.93	0.95	0.91	0.93
IoT Weather Sensor	Adam	0.8	0.83	0.86	0.84
	Adamax	0.78	0.79	0.82	0.81

7.3.2. Binary classification results on network layer data of ToN-IoT dataset

The proposed hybrid model has been evaluated for binary class classification on the network layer data of the ToN-IoT dataset. We conducted experiments using the Adam and Adamax optimization functions, and the assessment results are presented in Table 6. Based on the results evaluations, the Adam optimizer performance is optimum for the selected dataset. For the network layer dataset, the proposed model delivers excellent classification performance, achieving a 99% accuracy, 98% precision, 99% recall, and 98% F1-Measure score. Comparison of the proposed model with the existing solutions, as shown in Table 7. The results indicate that the proposed hybrid model outperforms existing techniques.

7.3.3. Binary classification results on application layer data of ToN-IoT Dataset

The proposed hybrid model has been evaluated for binary classification on the application layer data of the ToN-IoT dataset. The evaluation results of our experiments using the Adam and Adamax optimization functions are presented in Table 8. The results show that the model achieved optimal performance on Adam optimizer with 98% accuracy, 98% precision, 97% recall, and 98% F1-Measure score for the windows 10 log files. Similarly, for the windows 7 log files dataset, the binary classification performance achieved 84% accuracy, 85% precision, 84% recall, and 87% F1-Measure. The performance of the Adam optimizer was found optimal for windows 10 data files based on the result assessments. The findings of the evaluation indicate that the Adam optimizer yields the best results for the selected dataset. Table 9 show binary classification results of the proposed model are compared with those of the latest approaches. The outcomes indicate that the proposed hybrid model performance is optimum as compared to the existing techniques.

Table 5. Results comparison with existing models on binary classification of perception layer dataset.

Dataset Files	Reference	Models	Accuracy	Precision	Recall	F-Measure
IoT Fridge Sensor	[4] , [24]	LR	0.57	0.34	0.58	0.43
		LDA	0.77	0.79	0.77	0.77
		RF	0.97	0.97	0.97	0.97
		NB	0.5	0.53	0.51	0.51
		SVM	0.81	0.86	0.82	0.8
		LSTM	1	1	1	1
		Proposed Study	1	1	1	1
IoT Garage Door	[4] , [24]	LR	0.86	0.88	0.86	0.87
		LDA	0.86	0.88	0.86	0.87
		RF	0.85	0.85	0.85	0.85
		NB	0.84	0.86	0.85	0.86
		SVM	0.86	0.88	0.87	0.87
		LSTM	0.87	0.89	0.88	0.88
		Proposed Study	1	1	1	1
IoT GPS Tracker	[4] , [24]	LR	0.86	0.88	0.86	0.87
		LDA	0.86	0.88	0.86	0.87
		RF	0.85	0.85	0.85	0.85
		NB	0.84	0.86	0.85	0.86
		SVM	0.86	0.88	0.87	0.87
		LSTM	0.87	0.89	0.88	0.88
		Proposed Study	1	1	1	1
IoT Modbus Sensor	[4] , [24]	LR	0.67	0.46	0.68	0.55
		LDA	0.67	0.46	0.68	0.55
		RF	0.97	0.98	0.98	0.98
		NB	0.67	0.46	0.68	0.55
		SVM	0.67	0.46	0.68	0.55
		LSTM	0.68	0.47	0.68	0.55
		Proposed Study	0.85	0.8	0.78	0.8
IoT Motion Light	[4] , [24]	LR	0.58	0.34	0.59	0.43
		LDA	0.58	0.34	0.59	0.43
		RF	0.58	0.34	0.59	0.43
		NB	0.58	0.34	0.59	0.43
		SVM	0.58	0.34	0.59	0.43
		LSTM	0.59	0.35	0.59	0.44
		Proposed Study	0.8	0.78	0.78	0.8
IoT Thermostat Sen	[4] , [24]	LR	0.86	0.88	0.86	0.87
		LDA	0.86	0.88	0.86	0.87
		RF	0.85	0.85	0.85	0.85
		NB	0.84	0.86	0.85	0.86
		SVM	0.86	0.88	0.87	0.87
		LSTM	0.87	0.89	0.88	0.88
		Proposed Study	0.91	0.91	0.92	0.9
IoT Weather Sen	[4] , [24]	LR	0.58	0.6	0.59	0.53
		LDA	0.6	0.59	0.6	0.53
		RF	0.84	0.84	0.84	0.84
		NB	0.69	0.72	0.69	0.67
		SVM	0.63	0.68	0.63	0.55
		LSTM	0.82	0.82	0.81	0.8
		Proposed Study	0.99	0.97	0.96	0.95
Combined	[4] , [24]	LR	0.61	0.37	0.61	0.46
Data Files		LDA	0.68	0.74	0.68	0.62
		RF	0.85	0.87	0.85	0.85
		NB	0.62	0.63	0.62	0.51
		SVM	0.61	0.37	0.61	0.46
		LSTM	0.81	0.83	0.81	0.8
		Proposed Study	0.83	0.86	0.86	0.83

Table 6. Binary classification results on IoT network traffic data files.

Dataset Files	Optimizer	Accuracy	Precision	Recall	F1-Measure
Network Flow	Adam	0.99	0.98	0.98	0.99
Dataset	Adamax	0.97	0.96	0.95	0.96

Table 7. Comparison of network dataset results with existing models.

Dataset Files	Reference	Models	Accuracy	Precision	Recall	F-Measure
Network Traffic	[25]	DT	0.97	0.96	0.96	0.95
		RF	0.96	0.95	0.97	0.93
		KNN	0.97	0.98	0.95	0.94
		XGB	0.98	0.97	0.94	0.92
		Proposed Study	0.99	0.99	0.98	0.97

Table 8. Binary classification results of application layer datasets.

Dataset Files	Optimizer	Accuracy	Precision	Recall	F1-Measure
Windows 10	Adam	0.98	0.98	0.97	0.98
	Adamax	0.94	0.95	0.94	0.95
Windows 7	Adam	0.99	0.95	0.94	0.96
	Adamax	0.94	0.89	0.86	0.84

Table 9. Comparison of application layer datasets results with existing models.

Dataset Files	Reference	Models	Accuracy	Precision	Recall	F-Measure
Windows 7	[12]	KNN	0.92	0.68	0.95	0.79
		SVM	0.75	0.006	0.006	0.006
		Proposed Study	0.98	0.97	0.98	0.97
Windows 10	[19]	RF	0.9	0.94	0.92	0.91
		NB	0.92	0.82	0.85	0.87
		DT	0.93	0.94	0.92	0.9
		Proposed Study	0.88	0.94	0.96	0.9

7.3.4. Results of multi-class classification in the IoT perception layer

The proposed model is employed for multi-class classification using the Perception layer data files of the ToN-IoT datasets. The ToN-IoT dataset includes a feature known as "type", which contains 7 unique names for IoT attacks. The type of attack that occurs is determined through multi-classifications. The results of the multi-classification for perception-layer data files are

presented in Table 10, indicating that the Adam optimizer outperforms Adamax.

Table 10. Multi-class classification results of perception layer dataset.

Dataset Files	Optimizer	Accuracy	Precision	Recall	F-Measure
IoT Fridge Sensor	Adam	0.78	0.79	0.76	0.68
	Adamax	0.75	0.72	0.7	0.62
IoT Garage Door	Adam	0.85	0.83	0.85	0.85
	Adamax	0.78	0.77	0.79	0.76
IoT GPS Tracker	Adam	0.95	0.95	0.95	0.95
	Adamax	0.88	0.87	0.85	0.85
IoT Modbus Sensor	Adam	0.98	0.97	0.97	0.98
	Adamax	0.91	0.9	0.91	0.89
IoT Motion Light	Adam	0.76	0.75	0.78	0.76
	Adamax	0.72	0.69	0.71	0.68
IoT Thermostat	Adam	0.86	0.87	0.88	0.87
	Adamax	0.76	0.8	0.81	0.79
IoT Weather	Adam	0.98	0.99	0.99	0.97
	Adamax	0.95	0.96	0.93	0.94

7.3.5. Results of multi-class classification in the IoT network layer

The proposed hybrid model was evaluated for multi-class classification on the network layer data of the ToN-IoT dataset. We conducted experiments using the Adam and Adamax optimization functions and present the assessment results in Table 11. Based on the evaluation results, it is observed that the Adam optimizer outperforms the Adamax optimizer for the selected dataset. The proposed model delivers excellent multi-class classification performance for the network layer dataset, achieving 99% accuracy, 98% precision, 98% recall, and 97% F1-Measure score.

Table 11. Multi-classification results of network traffic.

Dataset Files	Optimizer	Accuracy	Precision	Recall	F-Measure
Network Traffic	Adam	0.99	0.99	0.98	0.97
	Adamax	0.96	0.95	0.97	0.93

7.3.6. Results of multi-class classification in the IoT application layer

Table 12 presents the proposed model results on application layer data of the utilized dataset. The results indicate that our proposed model achieved optimal results using the Adam optimizer, delivering excellent multi-class classification performance for the windows 7 log files dataset in the application layer. The model achieved an 88% accuracy, 86% precision, 97% recall, and 88% F1-Measure score. Similarly, for the windows 10 log files dataset, the multi-class classification performance achieved 84% accuracy, 85% precision, 84% recall, and 87% F1-Measure.

Table 12. Multi classification results of application layer datasets.

Dataset Files	Optimizer	Accuracy	Precision	Recall	F-Measure
Windows 7	Adam	0.88	0.86	0.87	0.88
	Adamax	0.81	0.8	0.79	0.8
windows 10	Adam	0.84	0.85	0.84	0.87
	Adamax	0.78	0.79	0.78	0.79

7.4. Discussion

In order to evaluate the performance of our proposed system, we conducted a comparative analysis with existing research that was developed for the same ToN-IoT 2020 dataset. To assess the effectiveness of our model, we employed several evaluation criteria, including accuracy, precision, recall, and f1-measure, and compared the results with those obtained by existing systems from the literature.

8. Conclusions

This study proposes a novel intrusion detection system for IoT based on a hybrid RNN and GRU model. The proposed approach is validated on the ToN-IoT datasets, demonstrating its effectiveness in detecting a range of common attacks and outperforming existing state-of-the-art approaches. This work has important implications for the field of intrusion detection in IoT, as it showcases the potential of DL techniques for improving the accuracy and effectiveness of existing systems. By optimizing the proposed approach through extensive data engineering, we achieved significant improvements in performance, which could have important implications for real-world applications of intrusion detection in IoT. However, there are still some limitations to the proposed approach that should be addressed in future research. For example, the proposed system heavily relies on the ToN-IoT dataset, which may not fully capture the range of attacks that might be encountered in real-world scenarios.

Use of AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgments

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).

Conflict of interest

The authors declare there is no conflict of interest.

References

1. K. Elgazzar, H. Khalil, T. Alghamdi, A. Badr, G. Abdelkader, A. Elewah, et al., Revisiting the internet of things: New trends, opportunities and grand challenges, *Front. Internet Things*, **1** (2022), 1–7. https://doi.org/10.3389/friot.2022.1073780
2. V. Terzieva, S. Ilchev, K. Todorova, The role of Internet of Things in smart education, *IFAC-PapersOnLine*, **55** (2022), 108–113. https://doi.org/10.1016/j.ifacol.2022.08.057
3. S. Tanwar, N. Gupta, C. Iwendi, K. Kumar, M. Alenezi, Next generation IoT and blockchain integration, *J. Sens.*, **2022** (2022). https://doi.org/10.1155/2022/9077348
4. L. K. Ramasamy, F. Khan, M. Shah, B. Prasad, C. Iwendi, C. Biamba, Secure smart wearable computing through artificial intelligence-enabled internet of things and cyber-physical systems for health monitoring, *Sensors*, **22** (2022), 1076. https://doi.org/10.3390/s22031076
5. Y. Cao, S. Miraba, S. Rafiei, A. Ghabussi, F. Bokaei, S. Baharom, et al., Economic application of structural health monitoring and internet of things in efficiency of building information modeling, *Smart Struct. Syst.*, **26** (2020), 559–573. https://doi.org/10.12989/sss.2020.26.5.559
6. C. Iwendi, G. Wang, Combined power generation and electricity storage device using deep learning and internet of things technologies, *Energy Rep.*, **8** (2022), 5016–5025. https://doi.org/10.1016/j.egyr.2022.02.304
7. M. Khan, M. Khattk, S. Latif, A. Shah, M. Ur Rehman, W. Boulila, et al., Voting classifier-based intrusion detection for IoT networks, in *Advances on Smart and Soft Computing*, Springer, (2022), 313–328. https://doi.org/10.1007/9789811655593_26
8. N. Naz, M. Khan, S. Alsuhibany, M. Diyan, Z. Tan, M. Khan, et al., Ensemble learning-based IDS for sensors telemetry data in IoT networks, *Math. Biosci. Eng.*, **19** (2022), 10550–10580. https://doi.org/10.3934/mbe.2022493
9. M. A. Razzaq, S. H. Gill, M. A. Qureshi, S. Ullah, Security issues in the Internet of Things (IoT): A comprehensive study, *Int. J. Adv. Comput. Sci. Appl.*, **8** (2017). https://doi.org/10.14569/ijacsa.2017.080650
10. G. Joshi, W. Kim, Survey, nomenclature and comparison of reader anti-collision protocols in RFID, *IETE Tech. Rev.*, **25** (2008), 234–243. https://doi.org/10.4103/0256-4602.44659
11. S. Al-Qaseemi, H. Almulhim, M. Almulhim, S. Chaudhry, IoT architecture challenges and issues: Lack of standardization, in *2016 Future Technologies Conference (FTC)*, (2016), 731–738. https://doi.org/10.1109/FTC.2016.7821686
12. R. Mahmoud, T. Yousuf, F. Aloul, I. Zualkernan, Internet of things (IoT) security: Current status, challenges and prospective measures, in *2015 10th International Conference for Internet Technology and Secured Transactions (ICITST)*, (2015), 336–341. https://doi.org/10.1109/ICITST.2015.7412116
13. V. Kumar, M. Devi, P. Raja, P. Kanmani, S. Velayutham, S. Sengan, et al., Design of peer-to-peer protocol with sensible and secure IoT communication for future internet architecture, *Microprocess. Microsyst.*, **78** (2020), 103216. https://doi.org/10.1016/j.micpro.2020.103216
14. K. Tajziehchi, A. Ghabussi, H. Alizadeh, Control and optimization against earthquake by using genetic algorithm, *J. Appl. Eng. Sci.*, **8** (2018), 73–78. https://doi.org/10.2478/JAES-2018-0010

15. M. Wu, T. Lu, F. Ling, J. Sun, H. Du, Research on the architecture of Internet of Things, in *2010 3rd International Conference on Advanced Computer Theory and Engineering (ICACTE)*, (2010), V5-484-V5-487. https://doi.org/10.1109/ICACTE.2010.5579493

16. K. Mohamed, IoT physical layer: sensors, actuators, controllers and programming, in *The Era of Internet of Things*, Springer, (2019), 21–47. https://doi.org/10.1007/978-3-030-18133-8_2

17. I. Ahmad, M. Niazy, R. Ziar, S. Khan, Survey on IoT: security threats and applications, *J. Rob. Control*, **2** (2021), 42–46. https://doi.org/10.18196/jrc.2150

18. G. E. Rodríguez, J. G. Torres, P. Flores, D. E. Benavides, Cross-site scripting (XSS) attacks and mitigation and Blockchain Integration, *Comput. Networks*, **166** (2020), 106960. https://doi.org/10.1016/j.comnet.2019.106960

19. K. Chen, S. Zhang, Z. Li, Y. Zhang, Q. Deng, S. Ray, et al., Internet-of-Things security and vulnerabilities: Taxonomy, challenges, and practice, *J. Hardware Syst. Secur.*, **2** (2018), 97–110. https://doi.org/10.1007/s41635-017-0029-7

20. B. Thakur, S. Chaudhary, Content sniffing attack detection in client and server side: A survey, *Int. J. Adv. Comput. Res.*, **3** (2013).

21. E. Fernandes, J. Jung, A. Prakash, Security analysis of emerging smart home applications, in *2016 IEEE Symposium on Security and Privacy (SP)*, (2016), 636–654. https://doi.org/10.1109/SP.2016.44

22. A. Sastry, S. Sulthana, S. Vagdevi, Security threats in wireless sensor networks in each layer, *Int. J. Adv. Comput. Res.*, **4** (2013), 1657–1661.

23. D. Welch, S. Lathrop, Wireless security threat taxonomy, in *IEEE Systems, Man and Cybernetics SocietyInformation Assurance Workshop*, (2003), 76–83. https://doi.org/10.1109/SMCSIA.2003.1232404

24. J. Cho, S. Yeo, S. Kim, Securing against brute-force attack: A hash-based RFID mutual authentication protocol using a secret value, *Comput. Commun.*, **34** (2011), 391–397. https://doi.org/10.1016/j.comcom.2010.02.029

25. A. Mitrokotsa, M. Rieback, A. Tanenbaum, Classification of RFID attacks, in *Proceedings of the 2nd International Workshop on RFID Technology-Concepts, Applications, Challenges (ICEIS 2008) - IWRT*, SciTePress, 2008. https://doi.org/10.1587/transinf.E93.D.518

26. J. Deogirikar, A. Vidhate, Security attacks in IoT: A survey, in *2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC)*, (2017), 32–37. https://doi.org/10.1587/transinf.E93.D.518

27. B. Bhati, G. Chugh, F. Al-Turjman, N. Bhati, An improved ensemble based intrusion detection technique using XGBoost, *Trans. Emerging Telecommun. Technol.*, **32** (2021), e4076. https://doi.org/10.1002/ett.4076

28. H. Bostani, M. Sheikhan, Hybrid of anomaly-based and specification-based IDS for Internet of Things using unsupervised OPF based on MapReduce approach, *Comput. Commun.*, **98** (2017), 52–71. https://doi.org/10.1016/j.comcom.2016.12.001

29. J. Singh, M. Nene, A survey on machine learning techniques for intrusion detection systems, *Int. J. Adv. Res. Comput. Commun. Eng.*, **2** (2013), 4349–4355.

30. Y. Otoum, A. Nayak, As-ids: Anomaly and signature based ids for the internet of things, *J. Network Syst. Manage.*, **29** (2021). https://doi.org/10.1007/s10922-021-09589-6

31. M. Ferrag, L. Maglaras, A. Ahmim, M. Derdour, H. Janicke, Rdtids: Rules and decision tree-based intrusion detection system for internet-of-things networks, *Future Internet*, **12** (2020). https://doi.org/10.3390/fi12030044

32. Sharipuddin, B. Purnama, Kurniabudi, E. A. Winanto, D. Stiawan, D. Hanapi, et al., Features extraction on IoT intrusion detection system using principal components analysis (PCA), in *2020 7th International Conference on Electrical Engineering, Computer Sciences and Informatics (EECSI)*, (2020), 114–118. https://doi.org/10.23919/EECSI50503.2020.9251292

33. A. Hussein, P. Falcarin, A. Sadiq, Enhancement performance of random forest algorithm via one hot encoding for IoT IDS, *Periodicals Eng. Nat. Sci.*, **9** (2021), 579–591. http://dx.doi.org/10.21533/pen.v9i3.2204

34. T. Saranya, S. Sridevi, C. Deisy, T. D. Chung, M. K. A. Khan, Performance analysis of machine learning algorithms in intrusion detection system: a review, *Procedia Comput. Sci.*, **171** (2020), 1251–1260. https://doi.org/10.1016/j.procs.2020.04.133

35. D. Zheng, Z. Hong, N. Wang, P. Chen, An improved LDA-based ELM classification for intrusion detection algorithm in IoT application, *Sensors*, **20** (2020), 1706. https://doi.org/10.3390/s20061706

36. J. Vitorino, R. Andrade, I. Praca, O. Sousa, E. Maia, A comparative analysis of machine learning techniques for iot intrusion detection, *arXiv preprint*, (2022), arXiv:2111.13149. https://doi.org/10.48550/arXiv.2111.13149

37. A. Verma, V. Ranga, Machine Learning intrusion detection systems for IoT applications, *Wireless Pers. Commun.*, **111** (2020), 2287–2310. https://doi.org/10.1007/s11277-019-06986-8

38. X. Kan, Y. Fan, Z. Fang, L. Cao, N. Xiong, D. Yang, et al., A novel IoT network intrusion detection approach based on adaptive particle swarm optimization convolutional neural network, *Inf. Sci.*, **568** (2021), 147–162. https://doi.org/10.1016/j.ins.2021.03.060

39. A. Banaamah, I. Ahmad, Intrusion Detection in IoT Using Deep Learning, *Sensors*, **22** (2022), 8417. https://doi.org/10.3390/s22218417

40. M. Almiani, A. AbuGhazleh, A. Al-Rahayfeh, S. Atiewi, A. Razaque, Deep recurrent neural network for IoT intrusion detection system, *Simul. Modell. Pract. Theory*, **101** (2020), 102031. https://doi.org/10.1016/j.simpat.2019.102031

41. I. Ullah, Q. Mahmoud, Design and development of RNN anomaly detection model for IoT networks, *IEEE Access*, **10** (2022), 62722–62750. https://doi.org/10.1109/ACCESS.2022.3176317

42. S. Park, H. Park, Y. J. Choi, RNN-based prediction for network intrusion detection, in *2020 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, (2020), 572–574. https://doi.org/10.1109/ICAIIC48513.2020.9065249

43. S. M. Kasongo, A deep learning technique for intrusion detection system using a Recurrent Neural Networks based framework, *Comput. Commun.*, **199** (2023), 113–125. https://doi.org/10.1016/j.comcom.2022.12.010

44. X. Wang, X. Lu, A host-based anomaly detection framework using XGBoost and LSTM for IoT devices, *Wireless Commun. Mobile Comput.*, **2020** (2020), 1251–1260. https://doi.org/10.1155/2020/8838571
45. S. Ullah, M. A. Khan, J. Ahmad, S. Jamal, Z. Huma, M. T. Hassan, et al., HDL-IDS: a hybrid deep learning architecture for intrusion detection in the Internet of Vehicle, *Sensors*, **22** (2022), 1340. https://doi.org/10.3390/s22041340
46. K. Tajziehchi, A. Ghabussi, H. Alizadeh, Control and optimization against earthquake by using genetic algorithm, *J. Appl. Eng. Sci.*, **8** (2018), 73–78. https://doi.org/10.2478/jaes-2018-0010
47. X. Ma, L. Foong, A. Morasaei, A. Ghabussi, Z. Lyu, Swarm-based hybridizations of neural network for predicting the concrete strength, *Smart Struct. Syst.*, **26** (2020), 241–251. https://doi.org/10.12989/sss.2020.26.2.241
48. A. Chawla, B. Lee, S. Fallon, P. Jacob, Host based intrusion detection system with combined CNN/RNN model, in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, Springer, (2018), 149–158. https://doi.org/10.1007/978-3-030-13453-2_12
49. S. Bini, Artificial intelligence, machine learning, deep learning, and cognitive computing: what do these terms mean and how will they impact health care, *J. Arthroplasty*, **33** (2018), 2358–2361. https://doi.org/10.1016/j.arth.2018.02.067
50. A. Arko, S. Khan, A. Preety, M. Biswas, *Anomaly Detection In IoT Using Machine Learning Algorithms*, Thesis, Brac University, 2019. http://hdl.handle.net/10361/12776
51. T. Ghazal, M. Hasan, M. Alshurideh, H. Alzoubi, M. Ahmad, S. AKbar, et al., IoT for smart cities: Machine learning approaches in smart healthcare—A review, *Future Internet*, **13** (2021), 218. https://doi.org/10.3390/fi13080218
52. O. Brun, Y. Yin, E. Gelenbe, Deep learning with dense random neural networks for detecting attacks against IoT-connected home environments, *Procedia Comput. Sci.*, **134** (2018), 458–463. https://doi.org/10.1016/j.procs.2018.07.183
53. S. Ullah, J. Ahmad, M. Khan, E. Alkhammash, M. Hadjouni, Y. Ghadi, et al., A new intrusion detection system for the Internet of Things via deep convolutional neural network and feature engineering, *Sensors*, **22** (2022), 3607. https://doi.org/10.3390/s22103607
54. Y. Otoum, D. Liu, A. Nayak, DL-IDS: a deep learning–based intrusion detection framework for securing IoT, *Trans. Emerging Telecommun. Technol., Wiley Online Library*, **33** (2022). https://doi.org/10.1002/ett.3803
55. R. H. Mohamed, F. A. Mosa, R. A. Sadek, Efficient intrusion detection system for IoT environment, *Int. J. Adv. Comput. Sci. Appl.*, **13** (2022). https://doi.org/10.14569/IJACSA.2022.0130467
56. Y. Yang, L. Wu, G. Yin, L. Li, H. Zhao, A survey on security and privacy issues in Internet-of-Things, *IEEE Internet Things J.*, **4** (2017), 1250–1258. https://doi.org/10.1109/JIOT.2017.2694844

57. T. Ariffin, S. Abdullah, F. Fauzi, M. Z. Murah, IoT attacks and mitigation plan: A preliminary study with Machine Learning Algorithms, in *2022 International Conference on Business Analytics for Technology and Security (ICBATS)*, (2022), 1–6. https://doi.org/10.1109/ICBATS54253.2022.9758933
58. W. W. Lo, S. Layeghy, M. Sarhan, M. Gallagher, M. Portmann, E-graphsage: A graph neural network based intrusion detection system for iot, in *NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*, (2022), 1–9. https://doi.org/10.1109/NOMS54207.2022.9789878
59. M. Sarhan, S. Layeghy, N. Moustafa, M. Gallagher, M. Portmann, Feature extraction for machine learning-based intrusion detection in IoT networks, in press, 2022. https://doi.org/10.1016/j.dcan.2022.08.012
60. I. Idrissi, M. Azizi, O. Moussaoui, Accelerating the update of a DL-based IDS for IoT using deep transfer learning, *J. Electr. Eng. Comput. Sci.*, **23** (2021). https://doi.org/10.11591/ijeecs.v23.i2.pp1059-1067
61. 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). https://doi.org/10.1186/s42400-021-00077-7
62. A. Alsaedi, N. Moustafa, Z. Tari, A. Mahmood, A. Anwar, TON_IoT telemetry dataset: A new generation dataset of IoT and IIoT for data-driven intrusion detection systems, *IEEE Access*, **8** (2020), 165130–165150. https://doi.org/10.1109/ACCESS.2020.3022862
63. F. Hilario, A. Luis, S. López, F. Herrera, N. V. Chawla, SMOTE for Learning from imbalanced data: Progress and challenges, marking the 15-year anniversary, *J. Artif. Intell. Res.*, **61** (2018). https://doi.org/10.1613/jair.1.11192
64. H. Han, W. Wang, B. Mao, Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning, in *International Conference on Intelligent Computing*, Springer, (2005), 878–887.
65. S. G. K. Patro, K. K. Sahu, Normalization: A preprocessing stage, *arXiv preprint*, (2015), arXiv:1503.06462. https://doi.org/10.48550/arXiv.1503.06462
66. A. Bahri, Y. Li, On a min-max procedure for the existence of a positive solution for certain scalar field equations in RN, *Rev. Mat. Iberoam.*, **6** (1990), 1–15. https://doi.org/10.4171/RMI/92