Journal of Ubiquitous Computing and Communication Technologies (ISSN: 2582-337X)
www.irojournals.com/jucct/

A Framework for Detecting Multiple Cyberattacks in IoT Environments

Yonas Mekonnen[1], Mesfin Kifle[2]

[1]Postgraduation Student, [2]Assistance Professor, Department of Computer Science, Natural and Computational Science, Addis Ababa University, Addis Ababa, Ethiopia

E-mail: [1]yostenact@gmail.com, [2]mesfin.kifle@aau.edu.et

Abstract

The Internet of Things refers to the growing trend of embedding ubiquitous and pervasive computing capabilities through sensor networks and internet connectivity. The growth and expansion of newly evolved cyberattacks, network patterns and heterogenous nature of cyberattacks trend become the warfare across the globe and challenges to apply single layer cyberattacks detection techniques to the Internet of Things. This research work identified the lack of cyberattacks detection framework as the major gap for detection of multiple cyberattacks such as denial of services, distributed denial of services, and multiple attacks while it includes multiple parameters at the same time The proposed framework contains three primary modules; the first module is responsible for capturing and pre-processing the captured data and construction of the model, then the core engine moule orchestrates the detection of cyberattacks. The third module, notifies and displays the results in a dashboard. This research study used multiple parameters including multiple attack classes, network packet patterns, and three scalar types namely no scaler, MinMax, and Standard. Regardless of the defined parameters used minmax scaler followed by standard scaler gives better detection performance than models trained with no scaler. The proposed framework is trained and evaluated with different models including Convolutional Neural Network (CNN), Hybrid, Feed Forward Neural Networks (FFNN), and Long Short-Term Memory (LSTM) that provides a result of 91.42%, 82.75%, 78.38%, and,74.83% detection accuracy respectively where it is observed that CNN model outperforms the optimal results among followed by hybrid and FFNN.

Keywords: IoT Environments, Cyberattacks, Multiple Attack Detection, CNN, LSTM, FFNN

DOI: https://doi.org/10.36548/jucct.2025.1.003
Received: 28.02.2025, received in revised form: 16.03.2025, accepted: 02.04.2025, published: 10.04.2025

1. Introduction

The Internet of Things (IoT) is becoming a new approach that incorporates the Internet into objects, as well as personal and even social life. The Internet of Things (IoT) refers to the growing trend of embedding ubiquitous and pervasive computing capabilities through sensor networks and internet connectivity. IoT devices are growing rapidly, making a big difference in daily lives, helping industries, manufacturing, transportation, healthcare, agriculture, consumers and retailers make important decisions. Despite the significant benefit of IoT to human lives still, the technology is not mature enough to provide secure communication whereas increased connected devices launch large security attacks. The existing research studies' attack detection frameworks are limited to flow which will have a drawback to be identified as an attack, it is necessary to wait for it to be terminated first Additionally, the proposed frameworks [1,2] have been concentrated on identifying a limited number of cyberattacks. Research studies such as [3,4] exclusively focus on one or a small number of pre-trained models in their approaches. Consequently, their attack detection techniques experienced a decline in accuracy, along with a fall in their resilience. The novel proposed solution aims to design a novel framework for detecting multiple (i.e. using network packet patterns or window sizes, one or more machine learning models, multiple scaler types, and applying at least two or more attack categories) application-layer cyberattacks in IoT environments.

This research study has a significant contribution, primarily the proposed framework can be integrated into IoT application-layer attack detection solutions. Secondly, the framework has the potential to be adapted for real-time cyberattacks detection, improving defence against evolving IoT threats and finally, supporting future researchers as an input for further optimization and improvement. In this research, the literature review on attack detection in IoT is given in Section 2. The overall Related Work is discussed in Section 3. Under section 4 the proposed architecture framework is described. Section 5 specifies the performance evaluation and result and analysis. In the end, the conclusion and future of this study is depicted in Section 6.

2. Literature Review

2.1 Elements of IoT

Enabling technologies for the Internet of Things considered in [5] can be grouped into three categories: i) technologies that enable “things” to acquire contextual information, ii) technologies that enable “things” to process contextual information, and iii) technologies to improve security and privacy. The first two categories can be jointly understood as functional building blocks required to build “intelligence” into “things”, which are indeed the features that differentiate the IoT from the usual Internet. The third category is not a function but rather a requirement, without which the penetration of the IoT would be severely reduced.

2.2 Benefits of IoT

IoT by generating actionable data, in turn, can improve efficiency, productivity, management, and quality control regardless of industry [6]. The IoT technologies promise a wide range of benefits across various industries such as manufacturers, agriculture, and infrastructure that benefited significant value from enterprise IoT-enabled applications [7,8]. Furthermore, IoT technologies hold an immense advantage from a medical services perspective an array from wearable fitness, and health monitoring devices to network-enabled medical devices that are expected to transform healthcare systems [9-11].

2.3 IoT Characteristics and Key Factors of IoT Security

Some of the common essential features and characteristics of IoT mentioned [12,13] are, interconnectivity, things-related services, heterogeneity, dynamic change, and economic scale. Implementing IoT is not without its difficulties, there are various factors to IoT security flaws both technical and non-technical, which can be broadly classified into four categories which are technology, system interoperability, dependability, and market readiness [14,15].

2.4 IoT Attacks and Detection Techniques

The fundamental foundational layer of IoT includes the physical, networks, and application layers whereby various attacks are happening across those layers and this study discusses those attacks in each layer as follows.

- **Physical Layer**

The authors [16] discuss the three-level architecture that satisfies IoT's core principles and includes application, network, and perception layers. Under the physical layer, spoofing security issues occurs in SDIoT controllers for which the authors [17] proposed a hybrid countermeasure, which combines initiative-taking and reactive methods to find and stop attacks. The issue of jamming detection [18] within the broader perspective of communication channels. The author [19] proposes a secure cross-device networking protocol to link IoT devices with software and hardware resources.

- **Network Layer**

The research introduces a novel approach to detecting Sybil attacks in wireless sensor networks using Traffic Monitoring (SDTM). The method uses traffic volume and statistical techniques to find malicious nodes using K-means and averaging algorithms [20].

- **Application Layer**

The study aims to analyse major risks for web applications and internet-based services that are universal to multiple web applications of various organizations [21]. An enhanced IoT architecture considers the environment and other factors affecting sensor operation by the researcher [22]. The goal is to improve the architecture by finding or monitoring items and environmental conditions.

2.5 Machine Learning-Based Detection

Some of the common machine learning approaches used in attack detection, as proposed and used by many researchers including in this research study, are: Supervised Learning, which is based on labeled data in a certain way [23]; Unsupervised Learning, where this approach makes use of unlabeled data, and the model marks them on its own depending on the characteristics of the data [24]; and in addition, Semi-Supervised Learning, which combines both supervised and unsupervised methods [25]. Reinforcement Learning is used in an environment that is constantly changing [26], where it follows an environment-driven approach in such a way that it may assist in the correction of errors and behavior in response to changes in the environment [27]. Deep Learning, a machine learning method, may be defined as an approach involving many layers to go through to achieve high-level feature learning from a training set. It is a specific type of feedforward neural network that can convert

the extracted features into models with the utmost accuracy, where the output of the final layer becomes the input of the subsequent layer [28].

Machine Learning Method FFNN is a widely used ANN that uses a hidden layer to decide the number of neurons in the input and output layers. In real-world scenarios, a network with numerous neurons generates a network, and the output of the FFNN model determines whether an attack is occurring in IoT datasets [29]. LSTMs are a type of RNNs used for collecting data from multiple layers, unlike FFNNs. RNNs are more effective at modelling sequences but face challenges like gradient vanishing and exploding. LSTMs use hidden unit connections to generate output and weight models. This approach is more effective than reinforcement neural networks (RNNs) due to their ability to handle gradient vanishing and exploding issues [29]. A hybrid model used is based on the combination of one or more models such as combination of FFNN, LSTM and/or CNN and LSTM or any other combination of models.

2.6 Machine Learning Methods

Authors [30] propose a method to show security vulnerabilities on IoT devices by integrating both CEP and ML. The accuracy, precision, recall, and F1-score were some of the metrics used to assess the suggested architecture on a dataset of botnet-generated network traffic. When compared to other approaches, the results showed that the suggested architecture was superior at identifying security breaches in IoT. However, the study's unique emphasis is on single attack class type where it might miss other kinds of attacks. Using an ensemble method and distributed JRip (Repeated Incremental Pruning to Produce Error Reduction) and Super-Vector Machine algorithms, this study [4] provides security architecture for IoT devices based on machine learning. The evaluation was conducted using the NSL-KDD dataset. Even though the detection rate and accuracy of 99.71% are high, it may not adhere to network and system signatures or standard interfaces, resulting in incorrect AI decisions and sub-parameter measurements. The author [2] presents a machine learning and stateful SDN architecture for identifying and preventing DDoS attacks in IoT networks using a decision tree technique. The proposed framework had 99.79% accurate results But the study does not provide or explicitly mention detailed information about the types of datasets used for experimentation, which may limit the generalizability of the results. Furthermore, the author [31] proposed a classical machine learning techniques (used Gradient Boosted decision tree, super vector, and random forest classifications to detect the DDoS attacks with an

enormous Botnet such as Mirai and the NSL-KDD dataset which provided the best accuracy of 85.34% of RF model.

2.7 Deep Learning Methods

- **Deep Neural Network**

The study [1] suggests using a middleware system based on a distributed deep neural network framework to find malware attacks on Internet of Things (IoT) devices. These models encompass a wide range of options, such as SVMs, DTs, GBs, NBs, and DNNs, as well as random forests, decision trees, and GBs. With 93% detection accuracy and a 92% f1-score, the results reveal the DNN model performs well, writing down its usefulness for IoT security. However, the study only focuses on physical and perception layer attack detection and does not address application layer attack detections. Additionally, the dataset sample distribution is not balanced and has limited features that create an overfit and need direction to investigate the data imbalance problem.

An IoT intrusion detection system based on a Dense Random Neural Network is described in this study [32]. The aim is to use artificial intelligence and big data analytics to create security solutions for IoT networks that are lightweight, quick, and adaptable. Researchers used the ToN-IoT security dataset and the Dense Random Neural Network algorithm to evaluate the proposed framework and the detection accuracy gives of 99.14% in binary class assessment. The research, however, does not deal with user profiles or behavioural analysis but rather concentrates on IDS traffic analysis and attack detection.

- **Convolutional Neural Network**

The study [33] proposed a framework called Intelligent Intrusion Detection Framework for Multi-Clouds and IoT Environments. The proposed solution keeps a higher true positive rate (TPR) and higher accuracy with less effort, measuring a 95.20% accuracy detection rate. The framework is not flexible enough to oversee multiple attack types.

- **Hybrid Deep Learning Techniques**

The authors [3] offer a method that utilizes multiple deep-learning models to predict damaging attacks on IoT backbone networks using the open-source UNSW-NB15 and NSL-KDD99 datasets. The models formed a convolutional autoencoder (CAE), a hybrid CNN-

LSTM, a long short-term memory, and a convolutional neural network were employed to evaluate the effectiveness of the framework that gives a 98.96% high-value accuracy, MLP, which is part of the suggested framework on finding malicious activity in IoT networks. However, the framework lacks a specified design flow, which means it does not explain how detection works or define the properties necessary to find attacks. In addition, it can be difficult to know what kinds of attacks to look out for because the framework does not specify them.

3. Proposed Work

This section presents a detailed description of the proposed framework for the detection of multiple application-layer cyberattacks in IoT environments. Figure 1 shows the block diagram of the proposed framework mainly three main modules and submodules underneath are designed. The primary module which is data acquisition and pre-processing module is mainly responsible for gathering all necessary data both offline and real-time from the IoT application environment then once data is collected the data pre-processing module cleans up the data, making sure it is consistent, and fills in the gaps if any. For data preparation, this study utilized ML pipeline stages such as the groups of functions like label and/or Indexer. The core engine of the framework, which is the application-layer detection module, mainly focuses on the detection of cyberattack threats in the application layer. The submodules within this engine construct the model: training the framework with selected feature values is the core of the solution, achieved by using different models. Then, the Threat Intelligence module constructs the detection of multiple attacks according to the defined algorithm. The Detection and Analysis module is responsible for the detection of various application-layer attacks, such as botnets, DDoS, DoS, and Mirai; these are the emphasis of this module. The attack simulation module will transmit a simulated attack on the IoT environment when generated.

The final main module is the alert and notification module where shows the results that come from the detection module where it notifies an alert based on the inputs below are the sub-categories that provide further notification in real-time.

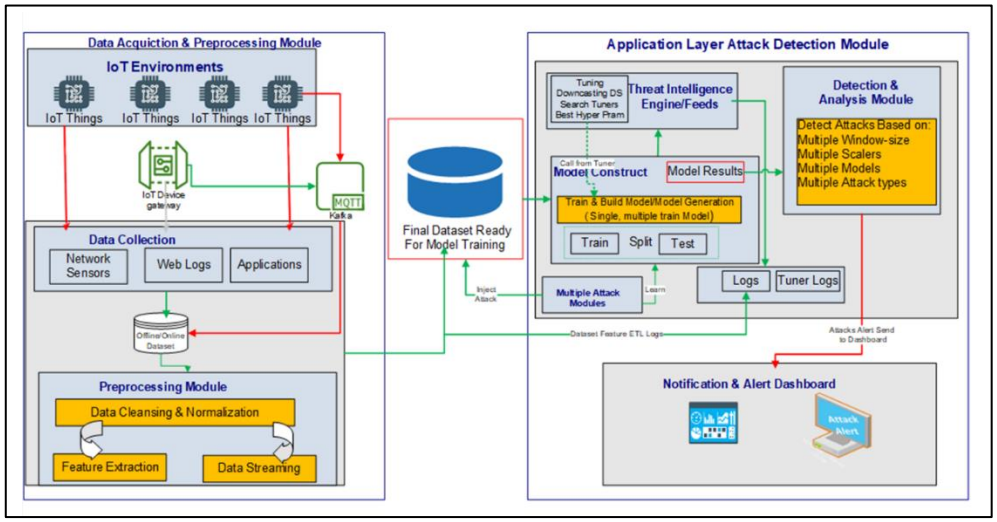

Figure 1. Multiple IoT Cyberattacks Detection Framework

3.1 Methodology

In this research study design science research [34] is applied. The reason behind choosing DSR method is because it focuses on solving real-world problems through the creation of innovative artifacts (the framework in this case). The DSR methodology allows for a more structured approach to both designing and evaluating the artifact which is as important as experimental analysis that provides a solid framework for the iterative process of designing and refining the artifact through continuous testing and evaluation, which is essential to the research. Data is collected based on CICIoT2023 labeled datasets with various application-layer attacks such as DoS, DDoS, Recon, Mirai in real-time, and from the collected a pre-processing technique of null and duplicate values are eliminated, insufficient packet sizes are dropped, constant values and missing values are handled based on mean imputation detail are mentioned in Table 1. Additionally, a feature engineering process is done by extracting important and stable features for effective attack detection. Furthermore, the proposed model trained with several models including CNN, FFNN, LSTM, and Hybrid to capture complex attack patterns with various hyperparameter values are used for the better performance of the results described in Table 1. The simulation platform was developed in Python using libraries such as TensorFlow and Keras for each of the model implementations with their own inputs and parameters.

Table 1. Optimal Hyperparameter Values

Inputs	Models			
	FFNN	**CNN**	**LSTM**	**Hybrid**
Max Trials	10			
Tuner & Train Epoch	10,25			
Dense Units	(64, 256, step=64)			
Dropout	(0.2, 0.5, step=0.1)			
Hidden Layer Activation	ReLU tanh	ReLU tanh		ReLU Tanh
Hidden Layer Count	(1, 3)	3	3	3
Output Method	Dropout	Dense	Dense	Dense
Output Unit	1	(64, 256, step-64)		
Output Activation	Softmax, Sigmoid			
Optimizer	(adam, rmsprop, sgd, dadam, adadelta, adagrad)			
Pool Size L2				2
Kernal size L1				7
Attention Layer & Dimention				96, 2
Num heads				6

Finally, the performance of the attack detection models was evaluated. Given that TP represents True Positives, TN True Negatives, FP False Positives, and FN False Negatives, the matrices used were recall, accuracy, F1-score, and precision-based parameters. Additionally, the models were also evaluated using cross-validation to achieve near-perfect discrimination for selected attacks using the selected model. In this research study, the key summary of qualitative and evaluation metrics used includes

Accuracy: used to measure the overall correctness of the predictions

$$Acc = \frac{TP + TN}{TP + TN + FP + FN}$$

Recall: measure the proportion of true positives among actual positives.

$$Rec = \frac{TP}{TP + FN}$$

Precision: Measure the proportion of true positives among predicted positives.

$$Pre = \frac{TP}{TP + FP}$$

F1-Score: measure the balance between precision and recall.

$$F1 = 2 \times \frac{Pre \times Rec}{Pre + Rec}$$

Cross-validation: involves the process of splitting the dataset to ensure no overfitting happened

Hyperparameter Tuning: Techniques are used to find the best combination

Confusion Matrix: Offers insights into the model's performance by visualizing results.

3.2 Dataset

The proposed framework was trained and evaluated based on the CICIoT2023 original dataset, as well as an 18-version final dataset created using six window sizes (5, 10, 20, 50, 100, and 200) and three scaler types (no scaler, min-max scaler, and standard scaler for each window size), resulting in a total size of 60GB. A 1.1 million balanced dataset was used and divided into an 80% training set, 10% testing set, and 10% validation set, with the following specific sample sizes: train shape: X_train = (879989, 33), y_train = (879989); test shape: X_test = (109990, 33), y_test = (109990); and validation: X_val = (110008, 33), y_val = (110008). The primary reason for the 80/10/10 split is to ensure that the test set adequately represents the data distribution, which helps assess the model's performance on unseen data.

3.3 Data Features and Class Information's

The study utilize features that are categorized as described in Table 2 and identified as raw features, device-specific features (are features that are not relevant for training of model but used for notification purpose only), non-scale features (features that are not be able to scaled), train features (features that are used for training of the model), and scale features (features that are used for scaling purpose). Again, a total of 18 individual attack class information is used which is categorized into DDoS, DoS, Mirai and Recon main categories.

Table 2. Dataset Feature Categories

S. No	Feature Categories	Lists of features
1	Device specific features (7)	Timestamp, src_mac, dst_mac, src_ip, dst_ip, src_port, dst_port
2	Train Features (33)	Protocol_type, fin_flag, syn_flag, rst_flag, psh_flag, ack_flag, urg_flag, icmp, tcp, other, udp, header_length, ttl, duration, sum_packet_size, max_packet_size, avg_packet_size, std_packet_size, min_duration, max_duration, sum_duration, avg_duration, std_duration, fin_count, syn_count, rst_count, psh_count, ack_count, urg_count, tcp_count, udp_count, icmp_count, other_count

3	Scale feature (22)	Train features except (Protocol_type, fin_flag, syn_flag, rst_flag, psh_flag, ack_flag, urg_flag, icmp, tcp, other, udp,)
4	Non-scale features (15)	Timestamp, encoded_label, protocol_type, fin_flag, syn_flag, rst_flag, psh_flag, ack_flag, urg_flag, src_port, dst_port, icmp, tcp, udp, others.

3.4 Model

The proposed model construction uses a different model including CNN, FFNN, LSTM, and Hybrid to represent data for training structured datasets found on the Internet of Things. The model is trained and evaluated with multiple parameter values 18 attack classes, three scaler types namely no scaler, MinMax scaler, and Standard scaler, and multiple network packet patterns /window sizes (seven categories of window sizes range from 5,10,15,20,50,100, and 200). The Algorithm 1 illustrates the Pseudocode for the multiple IoT cyberattacks

Algorithm 1. Pseudocode of Multiple IoT Cyberattacks Detection

Pseudocode Algorithm-1: Multiple IoT Cyberattacks Detection

Input:
- DSnormal = {DS1, DS2, ..., DSn) // Set of IoT Normal Datasets
- Datai // Training data for each dataset Di
- Nadv // Adversarial/attack Count
- A // Attack Method
- D // Detection Mechanism

Output:
- M // Trained Detection Model

Procedures:

1. Sample Generation: // Generate adversarial samples

For i = 1 to Nadv:
    x = Select a Normal IoT sample from DSnormal
    xadv = Apply adversarial attack A on x // xadv = A(x)
    Add (xadv, label(x)) to D_adv // Add to adversarial dataset

2. Model Training

TrainedModel ← EmptyTrainedModel // Initialize the model
  For each IoT dataset Di in DS:
  D_combined = DSnormal ∪ D_adv // Combine normal and adversarial datasets
  FFNNModeli ← TrainFFNN (Datai) // Train FFNN for dataset Di
  CNNModeli ← TrainCNN (Datai) // Train CNN for dataset Di
  LSTMModeli ← TrainLSTM (Datai) // Train DLSTM for dataset Di
  M ← TrainHybrid (CNN+LSTM) // Train Hybrid Model

3. Detection

For each IoT dataset Di in DS:

M_detection = ApplyDetection (M, Di) // Apply detection mechanism to the dataset

4. Model Evaluation: //Evaluate the performance of the detection model

Evaluate the hybrid model M_detection using the evaluation dataset D_eval

Compute metrics (Accuracy, Precision, Recall, F1-score)

5. Model Tunning:

- Add the new adversarial samples to D_adv

D_combined ← DSnormal ∪ D_adv // Retrain the model with the updated dataset

- M_detection ← TrainModel(D_combined) // Retrain model
- M_detection = ApplyDetection (M, D) // Apply detection to the updated model

3.5 Experimentation and Development Setup

- **Emulator Environment**

In this research study, we chose docker composer emulator environment for three main reasons initially for ease of simulation of multiple IoT devices at a time, second is for resource optimization; and finally for simplicity of cloning and running the attack simulations in docker environment.

In the emulator environment, all virtual image-based devices are created and ensured to be up and running to simulate the complete cyberattack detection. The process begins with capturing packets from sensor networks. This captured data is then converted to raw data (human-readable format) through a packet processing module, which processes and generates the final version dataset (containing different window sizes and scaler types). Subsequently, this data is used for pre-processing and analysis. The final version dataset is then passed through the model training phase and model tuning for redefinition before being input into the model. Finally, the model makes its predictions, and the inference produces the results.

- **Development Topology**

This research study utilizes a Docker Compose platform to manage diverse types of containerized nodes, as described in Figure 2 below, as an emulator environment based on Ubuntu OS with the following hardware requirements: 16 GB of RAM, Core i7 octa-core processor, and a 1TB SSD. Furthermore, the research employed various methods; for example, MQTT was used for messaging and streaming, a Cassandra database for data storage, and Fast API methods for orchestration and alerts. As shown in Figure 2, the development environment has three sections: the internal network, the external network, and

the communication network. The Internal Network Components consist of the IoT devices, which are directly connected to the external network through the router using the MQTT protocol standard. Communication Network Components: The router is the central communication component between the internal and external networks, utilizing two defined network interfaces: eth0 towards the external network and eth1 towards the internal network. Network Address Translation is configured on the router emulator container to ensure smooth communication among the connected devices.

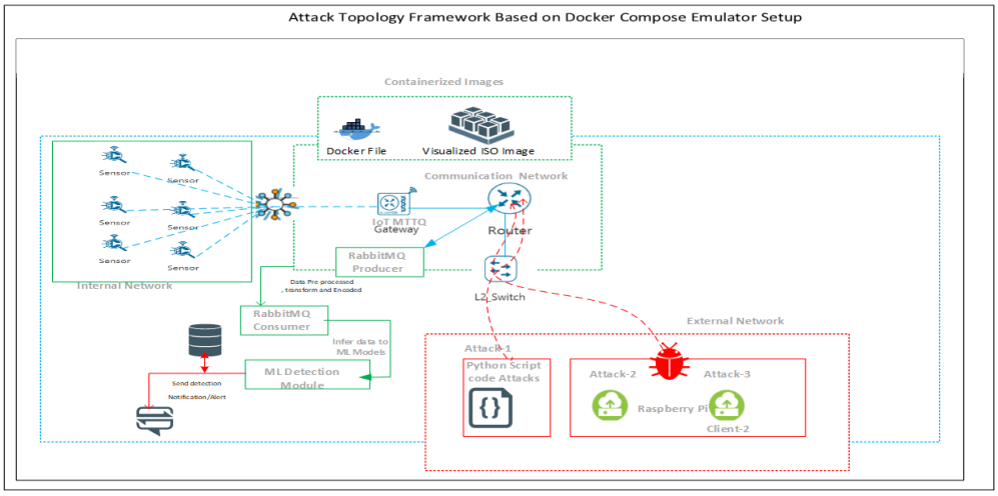

Figure 2. Attack Topology Emulator Setup

External Network Components: Here the external network shows various IoT attack scenarios that inject an attack to the environment which categorized into various categories. A low-level design topology described in Figure 3.

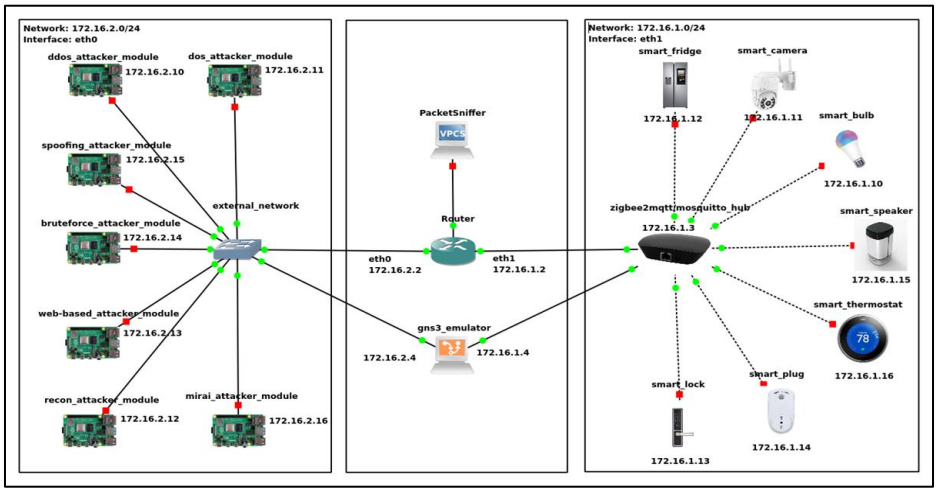

Figure 3. Attack Topology Low-Level Design

- **Tools**

This research study is using a prominent machine learning development framework and tools for large-scale data preparation and model creation shown in Table 3.

Table 3. Tools and Libraries

Library	Purpose	Library	Purpose
Pandas	For tabular, time series etc. data manipulation and analysis	Joblib	Provide lightweight pipelining
Scikit Learn	Provide supervised and unsupervised learning algorithms model evaluation and data preprocessing.	Multiprocessing	For the creation of processes that can run concurrently, leveraging multiple CPU cores.
Numpy	For large, multi-dimensional arrays and matrices.	Tqdm	Monitor the progress of computationally intensive tasks.
Dask	For the efficient parallelization of tasks	Zarr	For working with large, compressed, N-dimensional arrays
Pytorch	For developing and deploying deep learning models.	Logging	Allowing for the recording of events, errors, and other relevant information.
Conda	For managing Python packages and their dependencies	H5py	Allowing for efficient storage and manipulation of large, heterogeneous data.

4. Results and Discussion

This section aims to explore and present a novel proposed framework performance evaluation and analysis of the outcomes based on CICIoT2023 dataset, and its results are compared and discussed in terms of accuracy.

4.1 Detection of Model Performance

Table 4. Model Performance Result

S. No	Algorithm	Window Size	Scaler	Accuracy Score	Precision Score	Recall Score	F1 Score
1	CNN	200	Minmax	0.914245	0.897778	0.914245	0.901111
2	CNN	200	Standard	0.896845	0.911978	0.896845	0.890753

3	CNN	100	Standard	0.856714	0.856623	0.856714	0.855165
4	CNN	100	Minmax	0.840831	0.883632	0.840831	0.820334
5	Hybrid	100	Standard	0.827475	0.82688	0.827475	0.812218

When comparing the detection results described in Figure 4 and Table 4 for a window size of 200 with the min-max scaler to the same window size with the standard scaler, the min-max detection results increased by 1.74%. However, these results decreased by 4.01% when compared to a window size of 100. In addition, when comparing a window size of 100 with the standard scaler to a window size of 200 with the min-max scaler, the results show an increased accuracy detection of 5.75%. Conversely, when comparing the same min-max scaler but with different window sizes (100 and 200), the result drastically decreased by 7.34%. In other words, while increasing the window size generally increased detection accuracy, applying different scaler types within the same window size of 200 showed that the min-max scaler type achieved an accuracy of 0.9142 compared to the standard scaler type's accuracy of 0.8968, representing an increase of 1.74%

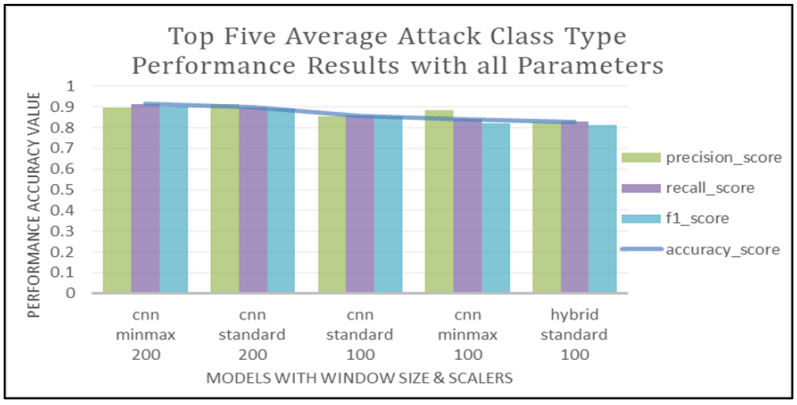

Figure 4. Model Performance Results

4.2 Model Comparison based on Algorithm, Window Size, and Scaler Types

- **CNN and FFNN Model Comparison**

Comparing the individual performance shown in Figure 5, for instance with window sizes of 100 and 200, the average accuracy of the FFNN results is 77.55% and 79.46%, respectively, representing an increase of 1.91% in accuracy. On the other hand, the average accuracy of the CNN model yields 80.46% and 80.60%, respectively, showing a 0.14% increase within the same window size comparison. Furthermore, comparing the detection results between the two algorithms, CNN performs with 2.90% higher detection accuracy than FFNN for a window size of 100 and 1.14% higher for a window size of 200.

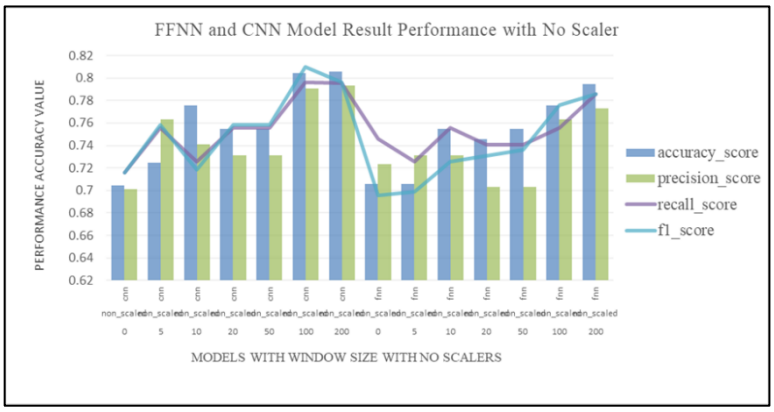

Figure 5. FFNN and CNN Performance with No Scaler

When compared with no scaler type and the min-max scaler within the same window sizes of 100 and 200, the detection score results for the CNN model increased by 3.63% and 10.83%, respectively, and for the FFNN model, the increase was 2.0% and 0.71%, respectively.

As shown below in Figure 6, when comparing the Min-Max and Standard Scalers for the CNN algorithm with window sizes of 100 and 200, the Min-Max detection accuracy decreased by 1.59% and increased by 1.74%, respectively. Similarly, for the FFNN model, the accuracy increased by 0.90% for a window size of 100 and by 1.78% for a window size of 200.

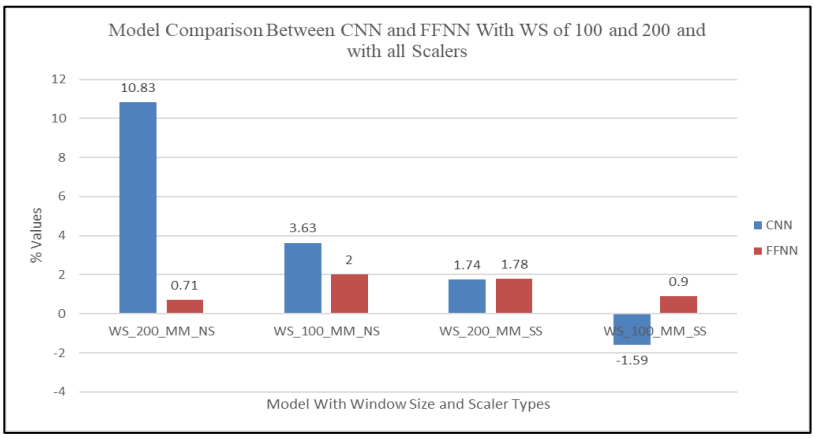

Figure 6. CNN and FFNN model with all Scalers

- **LSTM and FFNN Model Comparison**

When comparing the performance results of the FFNN and LSTM models across all window sizes without applying a scaler, the performance of the FFNN model increased

compared to the LSTM model, as described in Figure 7 below. Considering individual performance, for instance with window sizes of 100 and 200, the average accuracy of the FFNN model yielded 77.55% and 79.46%, respectively, representing a 1.91% increase in accuracy. On the other hand, the accuracy of the LSTM model was 66.56% and 71.06%, respectively, showing a 4.45% increase within the same window size comparison. Furthermore, comparing the detection results between the two algorithms, FFNN achieved 10.99% higher detection accuracy than LSTM for a window size of 100 and 8.4% higher for a window size of 200.

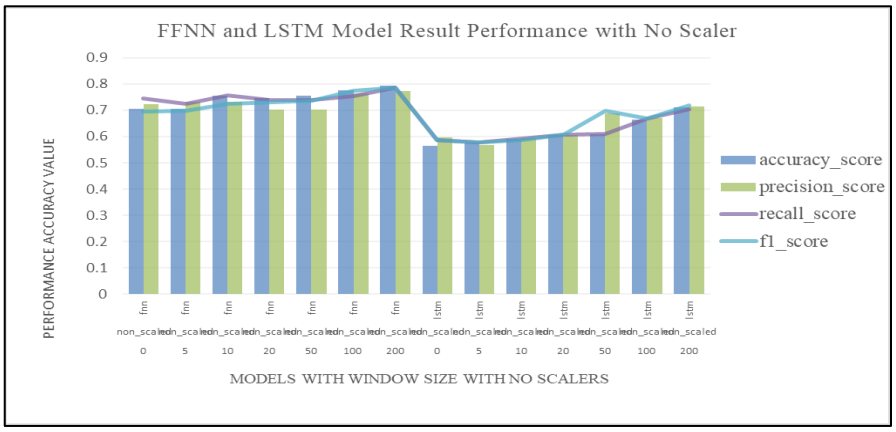

Figure 7. FFNN and LSTM Performance with No Scaler

As shown below in Figure 8, compared to MinMax and Standard Scaler LSTM algorithm the MinMax detection accuracy decreased by -6.43% for window size 100 and increased by 2.26% for window size 200. Again, for FFNN model it increased by 0.90% for window size 100, and 0.78% for a window size of 200. From the overall results analysis of the two models (FFNN and LSTM) it can be concluded that when the window size increases the detection accuracy results in better performance. Applying a scaler type such as MinMax and Standard scaler has a better performance than non-scaler (even if exceptionally LSTM model with no scaler shows better performance with 3.06% increase), the MinMax scaler has better detection accuracy than Standard Scaler and finally, with all parameters FFNN model has a better result than LSTM model which has 0.45% increase, 7.33% increase within MinMax and NS of 100, and MinMax and SS of 100 WS respectively. However, -1.48% decrease was observed for 200 window size between MinMax and standard scaler.

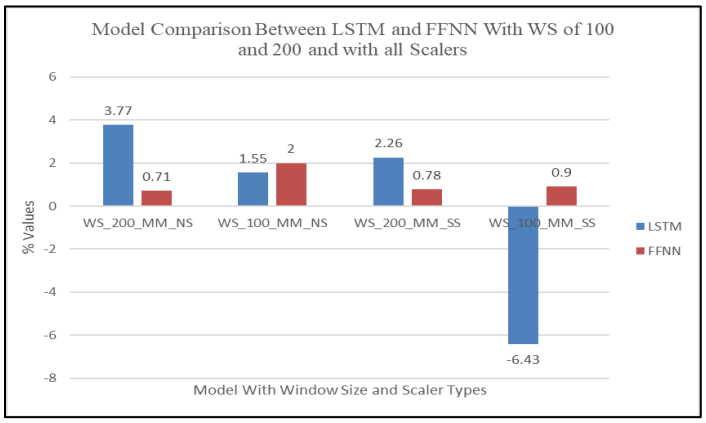

Figure 8. LSTM and FFNN Performance with all Scaler

- **CNN and LSTM Model Comparison**

Comparing the performance results of CNN and LSTM model with all window size's and without applying a scale the performance of CNN model exceeds LSTM model as described in Figure 9 below.

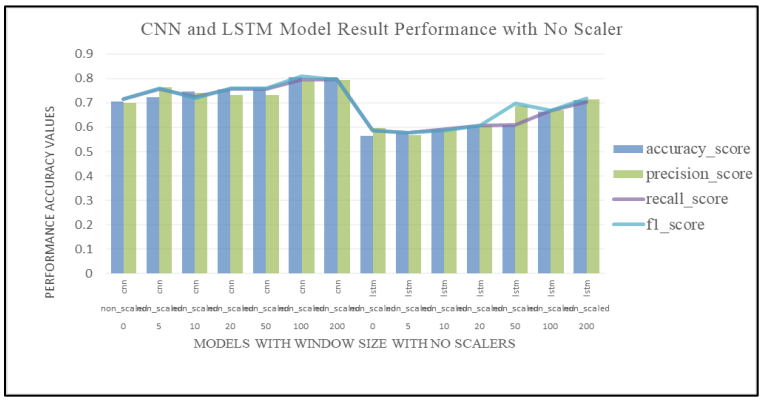

Figure 9. CNN & LSTM Performance with No Scaler

Without scaling, CNN outperformed LSTM in average accuracy for window sizes 100 and 200. CNN's accuracy increased slightly with larger window size, while LSTM's showed a more significant increase. CNN also consistently achieved higher detection accuracy than LSTM for both window sizes. Applying the min-max scaler compared to no scaling improved detection scores for both LSTM and CNN across both window sizes. When comparing min-max and standard scalers, LSTM's accuracy decreased for a window size of 100 but increased for 200, while CNN's accuracy increased compared to no scaling for both window sizes.

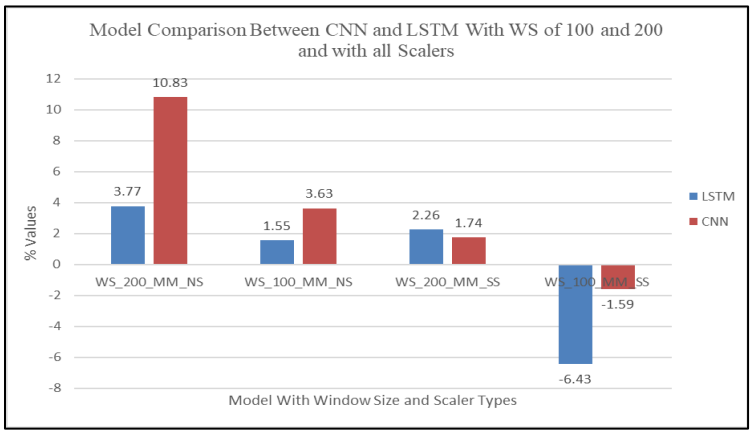

Figure 10. CNN & LSTM Performance with all Scaler

The analysis of CNN and LSTM models in Figure 10 indicates that larger window sizes and the application of scalers (especially Min-Max) improve detection accuracy. CNN generally outperforms LSTM across various configurations. Specifically, CNN shows significant accuracy gains over LSTM with the Min-Max scaler and no scaler at larger window sizes, and with Min-Max and Standard Scaler at a smaller window size. However, a minor decrease in CNN's accuracy was noted in one specific configuration (Min-Max and Standard Scaler at a window size of 200)

- **Hybrid Model Comparison**

When using a standard scaler and a window size of 100, the hybrid model's detection accuracy was lower than CNN's, regardless of the scaler type (standard or min-max) as shown in Figure 11. CNN also outperformed the hybrid model when comparing a 200 window size for CNN against a 100 window size for the hybrid model. With a min-max scaler and a 200 window size, CNN demonstrated superior accuracy, precision, recall, and F1-score compared to the hybrid model, which experienced a significant drop in accuracy but a slight increase in precision.

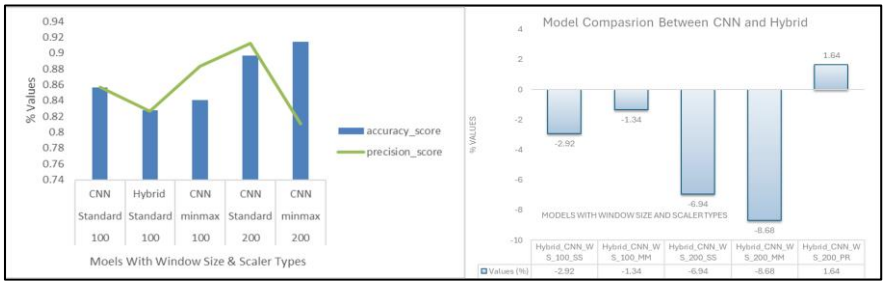

Figure 11. Hybrid and CNN Performance

4.3 Overall Performance Results based on Multiple Attributes

The proposed detection framework applied multiple attributes for cyberattack detection, and the resulting findings demonstrate promising and outperformed performance based on the applied attributes, which include window size, multiple attack classes, and different scaler types. Figure 12 below shows the overall percentage values comparison.

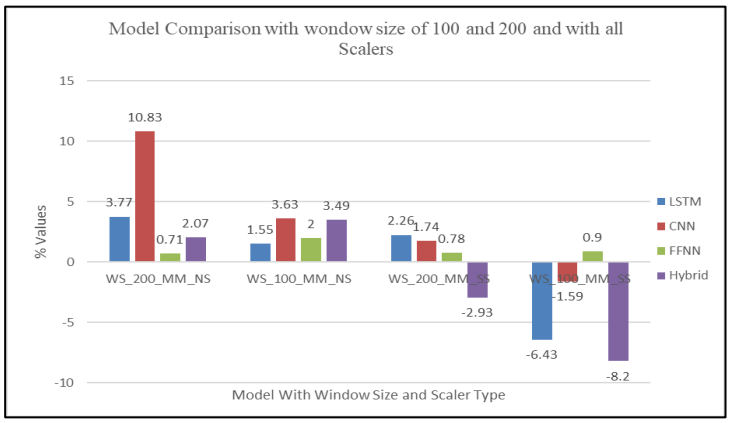

Figure 12. Overall Model Performance with all Scales

The analysis indicates that applying the Min-Max scaler generally leads to better detection accuracy for CNN compared to using no scaler across window sizes 100 and 200. CNN consistently outperformed LSTM, Hybrid, and FFNN models when using the Min-Max scaler. While Min-Max generally outperformed the Standard Scaler for LSTM, CNN, and FFNN at a window size of 200, the Standard Scaler yielded better performance than Min-Max for the Hybrid model at the same window size.

4.4 Individual Attack Class Performance Results

From the individual attack detection point of view, with window sizes of both 100 and 200 (as shown in the Table 5 below) and using both min-max and standard scalers, the CNN and hybrid models outperformed, achieving almost 100% detection precision for attack classes 1 to 4, which are Ddos_ack_fragmentation, Ddos_icmp_flood, Ddos_icmp_fragmentation, Ddos_pshack_flood, and Ddos_rstfin_flood

CNN (WS_200 Standard Scaler)

Attack Class	Precision	Recall	F1-score
0	1.00	1.00	1.00
1	1.00	1.00	1.00
2	1.00	1.00	1.00
3	1.00	1.00	1.00
4	1.00	1.00	1.00
5	1.00	0.98	0.98
6	0.93	0.88	0.91
7	0.911	0.92	0.92
8	0.969	0.97	0.97
9	0.5816	0.5584	0.5296
10	1.00	0.97	0.97
11	0.9039	0.95	0.92
12	0.8843	0.8	0.84
13	0.9321	0.92	0.9244
14	0.8035	0.589	0.67
15	0.6756	0.856	0.7552
16	1.00	0.96	0.98
17	0.8194	0.82	0.824

(a)

CNN (WS_200 MinMax Scaler)

Attack Class	Precision	Recall	F1-score
0	0.99	1.00	1.00
1	1.00	1.00	1.00
2	1.00	1.00	1.00
3	1.00	1.00	1.00
4	1.00	0.98	1.00
5	1.00	0.9689	1.00
6	0.93	0.83	0.88
7	0.87	0.93	0.9
8	0.9	0.98	0.94
9	0.75	0.83	0.79
10	1.00	0.96	1.00
11	0.97	0.92	0.92
12	0.9	0.87	0.88
13	0.86	0.79	0.82
14	0.89	0.92	0.91
15	0.93	0.89	0.91
16	1.00	1.00	1.00
17	0.17	0.6	0.27

(b)

Table 5. (a) (b) Attack Class Performance

While DoS attack detection performed relatively well, it was still less accurate than individual DDoS attack detection. Mirai and Recon attack detection generally performed poorly, except for the hybrid model achieving perfect precision for a specific Mirai UDP flood attack with a 100 window size and both scalers. This variability in performance is attributed to the size of the datasets for each individual attack class. Figure 13 illustrates the detailed classification performance of the CNN model under specific conditions (200 window size, standard scaling).

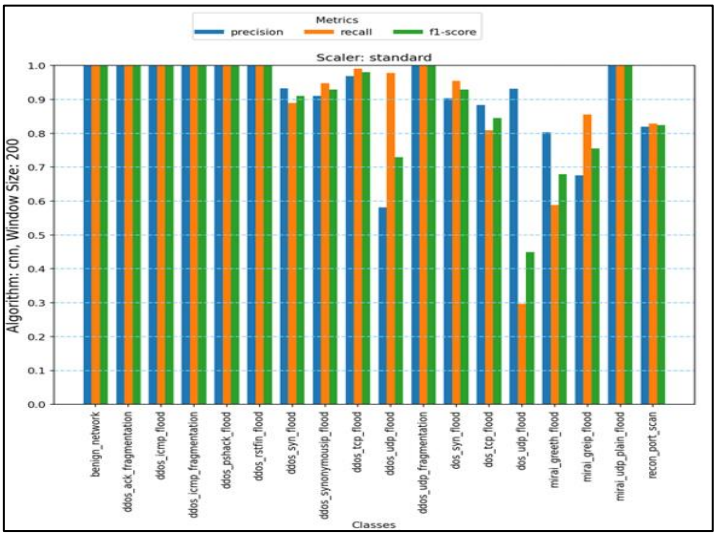

Figure 13. CNN WS_200 Individual Attack Class Representation

5. Conclusion

Recognizing the challenges in cyberattack detection within heterogeneous IoT environments due to diverse devices, evolving threats, and varied data, this research proposes a novel framework employing multiple parameters like attack classes and scalers. Evaluation across CNN, Hybrid, FFNN, and LSTM models revealed that using Min-Max and Standard scalers improved detection, with CNN achieving the highest accuracy (91.42%). Future work will focus on tuning the framework with more deep learning models and parameters, incorporating additional attack types such as, web attacks, and integrating multiple IoT datasets to enhance its ability to detect a broader range of threats.

References

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

[2] W. L. Khedr, A. E. Gouda and E. R. Mohamed, "FMDADM: A multi-layer ddos attack detection and mitigation framework using Machine Learning for stateful SDN-based IOT Networks," IEEE Access, vol. 11, 2023. 28934-28954.

[3] Nagisetty and G. P. Gupta, "Framework for detection of malicious activities in IOT networks using Keras Deep Learning Library," 2019 3rd International Conference on Computing Methodologies and Communication (ICCMC), 2019

[4] M. Bagaa, T. Taleb, J. Bernabe and A. Skarmeta, "A Machine Learning Security Framework for IOT systems," IEEE Access, vol. 8, 2020. 114066-114077.

[5] O. Vermesan, P. Friess., “Internet of Things from Research and Innovation to Market Deployment” Denmark: River Publishers, 2014, 10

[6] V. Rigworo Kebande, "Industrial internet of things (IIoT) forensics: Challenges, opportunities, and future directions," 2023.

[7] Ullah, S. M. Anwar, J. Li, L. Nadeem, T. Mahmood, A. Rehman and T. Saba, "Smart cities: The role of internet of things and machine learning in realizing a data-centric smart environment," Complex Intelligent Systems, vol. 10, no. 1, 2023. 1607-1637.

[8] S. Syed, D. Sierra-Sosa, A. Kumar and A. Elmaghraby, "IoT in Smart Cities: A Survey of Technologies, practices and challenges," Smart Cities, vol. 4, no. 2, 2021. 429-475.

[9] X. Yang, X. Wang, X. Li, D. Gu, C. Liang, K. Li, G. Zhang and J. Zhong, "Exploring emerging IoT Technologies in smart health research: A knowledge graph analysis," BMC Medical Informatics and Decision Making, vol. 20, no. 1, 2020.

[10] Y. Cheng, X. Zhao, J. Wu, H. Liu, Y. Zhao, M. Al Shurafa and I. Lee, "Research on the smart medical system based on Nb-IOT Technology," Mobile Information Systems.2021.1-10.

[11] Rejeb, K. Rejeb, H. Treiblmaier, A. Appolloni, S. Alghamdi, Y. Alhasawi and M. Iranmanesh, "The internet of things (IoT) in Healthcare: Taking Stock and moving forward," Internet of Things, vol. 22, 2023.

[12] P. High, "Forbes," Forbs, 25 November 2014. [Online]. Available: https://www.forbes.com/sites/peterhigh/2013/10/14/gartner-top-10-strategic-technology-trends-for-2014/. [Accessed 25 August 2023].

[13] P. Friess and O. Vermesan, "Internet of things applications - from research and innovation to market deployment," 2022.

[14] R. Khan, M. Kashif, R. H. Jhaveri, R. Raut, T. Saba and S. A. Bahaj, "Deep learning for intrusion detection and security of internet of things (IOT): Current analysis, challenges, and possible solutions," Security and Communication Networks,2022. 1-15.

[15] M. Catillo, A. Pecchia and U. Villano, "Botnet detection in the internet of things through all-in-one deep autoencoding," Proceedings of the 17th International Conference on Availability, Reliability and Security, 2022.

[16] M. Burhan, R. A. Rehman, B. Khan and B.-S. Kim, "IoT elements, layered architectures and security issues: A comprehensive survey," Sensors, vol. 18, no. 9, 2018.

[17] T.-H. Nguyen and M. Yoo, "A hybrid prevention method for eavesdropping attack by link spoofing in software-defined internet of things controllers," International Journal of Distributed Sensor Networks, vol. 13, no. 11, 2017.

[18] Wood, J. Stankovic and S. Son, "Jam: A jammed-area mapping service for Sensor Networks," Proceedings. 2003 International Symposium on System-on-Chip (IEEE Cat. No.03EX748).

[19] D. Puthal, S. Nepal, R. Ranjan and J. Chen, "Threats to networking cloud and edge datacenters in the internet of things," IEEE Cloud Computing, vol. 3, no. 3, 64-71, 2016.

[20] S. Golestani Najafabadi, H. Naji and A. Mahani, "Sybil attack detection: Improving security of WSNS for Smart Power Grid Application," 2013 Smart Grid Conference (SGC), 2013.

[21] G. S and G. B.B., "Cross-Site Scripting (XSS) attacks and defense mechanisms: classification and state-of-the-art," Int J Syst Assur Eng Manag, vol. 8, no. 1, 2017. 512-530.

[22] D. D., "Improved Layered Architecture for Internet of Things," Int. J. Comput. Acad. Res (IJCAR), vol. 4, 2015. 214-223.

[23] M. Hasan, M. M. Islam, M. I. Zarif and M. Hashem, "Attack and anomaly detection in IOT sensors in IoT sites using machine learning approaches," Internet of Things, vol. 7, 2019.

[24] Haldorai, A. Ramu and M. Suriya, "Organization internet of things (IoTs): Supervised, unsupervised, and reinforcement learning," Business Intelligence for Enterprise Internet of Things, 2020. 27-53.

[25] S. Rathore and J. H. Park, "Semi-supervised Learning Based Distributed Attack Detection Framework for IoT," Applied Soft Computing, vol. 72, 2018. 79-89.

[26] M. Lopez-Martin, B. Carro and A. Sanchez-Esguevillas, "Application of deep reinforcement learning to intrusion detection for supervised problems," Expert Systems with Applications, , 2020. 141.

[27] K. Yang, J. Ren, Y. Zhu and W. Zhang, "Active learning for wireless IoT intrusion detection," IEEE Wireless Communications, vol. 25, no. 6, 2018. 19-25.

[28] Bhati and C. S. Rai, "Analysis of support vector machine-based intrusion detection techniques," Arabian Journal for Science and Engineering, vol. 45, no. 4, pp., 2019.

[29] O. Jullian, B. Otero, E. Rodriguez, N. Gutierrez, H. Antona and R. Canal, "Deep-learning based detection for cyber-attacks in IoT Networks: A distributed attack detection framework," Journal of Network and Systems Management, vol. 31, no. 2, 2023. 2371-2383.

[30] J. Roldán, J. Boubeta-Puig, J. Luis Martínez and G. Ortiz, "Integrating complex event processing and machine learning: An intelligent architecture for detecting IOT security attacks," Expert Systems with Applications, vol. 149, 2020.

[31] M. Anwer, S. M. Khan, M. U. Farooq and W. Waseemullah, "Attack detection in IoT using machine learning," Engineering, Technology & Applied Science Research, vol. 11, no. 3, 7273-7278, 2021.

[32] S. Latif, Z. E. Huma, S. S. Jamal, F. Ahmed, J. Ahmad, A. Zahid, K. Dashtipour, M. U. Aftab, M. Ahmad and Q. H. Abbasi, "Intrusion Detection Framework for the internet of things using a dense random neural network," IEEE Transactions on Industrial Informatics, vol. 18, no. 9, 6435-6444, 2022.

[33] S. M. Nizamudeen, "Intelligent intrusion detection framework for multi-clouds – IoT environment using swarm-based deep learning classifier," Journal of Cloud Computing, 2023.

[34] Vom Brocke, J., Hevner, A. and Maedche, A., 2020. Introduction to design science research. Design science research. Cases, 1-13.