Iraqi Journal of Communications and Computer Networks
e-ISSN: 3105-9252, p-ISSN: 3105-9244
Vol. (1), Iss. (1), September 2025
DOI: https://doi.org/10.58564/IJCCN.1.1.2025.1

https://https://ijccn.net/

Research Article

Green AI-Based Cybersecurity Model for IoT System

Tayseer S. Atia [1*], Ahmed Y. Yousuf [2], Aya R. Hashiem [3]

[1] College of Engineering, Al-Iraqia University, Baghdad, Iraq
[2] College of Engineering, Al-Iraqia University, Baghdad, Iraq
[3] College of Engineering, Al-Iraqia University, Baghdad, Iraq

*Corresponding Author: tayseer.salman@aliraqia.edu.iq

Citation: Tayseer S. Atia, Ahmed Y. Yousuf, & Aya R. Hashiem. (2025). Green AI-Based Cybersecurity Model for IoT System. Iraqi Journal of Communications and Computer Networks (IJCCN), 1(1), 1–13.

ARTICLE INFO

Received: 12 May. 2025
Revised: 22 Jul. 2025
Accepted: 29 Jul. 2025

ABSTRACT

In this manuscript, the Green AI paradigm is employed to develop an Intrusion Detection System (IDS) using Deep Learning. The aim is to overcome the complexity and generalization issues in IoT cyberattack detection solutions by automatically generating Convolution Neural Network (CNN) models and optimizing their settings to ensure their effectiveness on different datasets. The best model is then converted to a sparse model, achieving the Green AI. To generate the CNN models, five activation functions, one normalization, and four regularization parameters are optimized using a three-level tree. The tree of models starts with the root representing the activation function, followed by the normalization level, and then the regularization level. Each time the tree produces a new model structure by changing a node value in activation, normalization, and regularization levels with different combinations, resulting in several branches in the tree. For each tree model, the kernel and filter values are optimized using an exhaustive search to select the best combination among three kernels and three filter values. The findings demonstrate that the proposed method is efficient in generating a model that achieves an accuracy of 95.71 for the MQTT-IoT-IDS2020 dataset and generalizes well on UNSW-NB-2015 and KDD IoT-2017 datasets with 95.87 and 95.91 respectively. Therefore, automatically generating models is a preferred technique for researchers to ensure generalization on different datasets.

Keywords: : Intrusion detection system, cyber- attack, IoT, deep learning, CNN.

1. INTRODUCTION

The Internet of Things (IoT) is a complex network comprised of interconnected computing devices, machinery, objects, fauna, and individuals that possess the ability to directly communicate with one another by transmitting data across a network [1]. The exponential growth of IoT devices and applications over the past decade has resulted in a significant upsurge in cyber-attacks [2]. These cyber-attacks have caused the breach of thousands of unguarded and open-to-public IoT devices [3]. Fig. 1 displays that the number of Internet of Things (IoT) devices being utilized globally will increase to approximately 29 billion by 2030, which is a significant rise from the 9.7 billion devices being used in 2020.

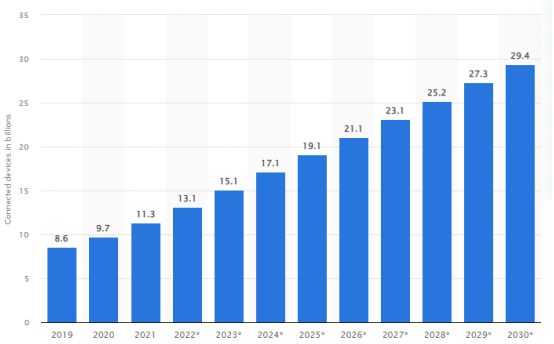

FIGURE 1. Number of IoT-connected devices worldwide from 2019 to 2021, with forecasts from 2022 to 2030 [4]

Researchers have been striving to combat cyber-attacks and breaches for quite some time now. One of the security measures that have been developed to counter such attacks is intrusion detection systems (IDS). Intrusion detection systems employ various techniques, including rule-based, threshold value, state transition diagrams, support vector machines, genetic algorithms, fuzzy logic, artificial neural networks, data mining, and the artificial immune system. However, deep learning is a technique that generates high-level features from low-level ones. Deep learning consists of multiple layers that can execute several processes simultaneously, rendering it independent of human intervention [5]. As a result, many researchers are currently working in the field of Machine Learning (ML).
Researchers have been striving to combat cyber-attacks and breaches for quite some time now. One of the security measures that have been developed to counter such attacks is intrusion detection systems (IDS). Intrusion detection systems employ various techniques, including rule-based, threshold value, state transition diagrams, support vector machines, genetic algorithms, fuzzy logic, artificial neural networks, data mining, and the artificial immune system. However, deep learning is a technique that generates high-level features from low-level ones. Deep learning consists of multiple layers that can execute several processes simultaneously, rendering it independent of human intervention [5]. As a result, many researchers are currently working in the field of Machine Learning (ML).

There are several studies in the literature on deep learning. Kolosnjaji et al. [7] developed a Neural Network (NN) consisting of feedforward and CNN in 2017. They collected malware samples from Virusshare, Maltrieve, and private collections and achieved an evaluation result of 93% precision and recall. In 2017, Wei Wang et al. [8] established an artificial intelligence taxonomy for traffic classification and advanced a CNN approach to malware traffic classification using traffic data as images. The experiment showed that the approach can achieve an accuracy of 99.41% and has a high level of scalability. Wang et al. [9] used a single-dimensional convolution neural network with an end-to-end encrypted traffic categorization technique in 2017. The result displayed a 2-class categorization for non-VPN and VPN traffic with 100% and 99% accuracy, respectively. Xin [10] used four DL models proposed and compared with ML algorithms by Roopak et al. [11] in 2019. Liu et al. [12] suggested a DL-based method for detecting IoT botnets in 2019. The study team used the IoT botnet assault dataset available on the UCI Machine Learning Repository and achieved an accurate differentiation of 99.57% of attack traffic from benign traffic. Ugurlu and Dogru [13] surveyed DL using CNN, RNN, LSTM, DBN, and RBM in 2019. Patidar and Bains [14] created an 8-layer CNN in 2020 to recognize DDoS attacks using the CICIDS2017 datasets, which provided the highest notable precision of roughly 99.38%. Kabir et al. [15] proposed a multiclass CNN model with a mish activation function and Ranger optimizer in their study in 2021, which achieved an accuracy of 98.9% using CIC-IDS-2018 dataset. In 2021, Mendonça [16] suggested a novel IDS based on the Tree-CNN hierarchical algorithm with the Soft-Root-Sign (SRS) activation function, which achieved an average detection accuracy of 0.98 while also significantly lowering execution time (by about 36%). Kumar et al. [17] proposed a hybrid feature selection scheme that combines statistical test-based filter approaches with a metaheuristic approach based on Non-Dominated Sorting Genetic Algorithm (NSGA-II) in 2023 and achieved gains in accuracy up to the maximum (90.48%). Fawzi et al. [18] used the MQTT-IoT-IDS2020 dataset to propose deep learning for the automated detection of brute force assaults on MQTT-IoT networks in 2023 and achieved more than 99% accuracy in differentiating between typical attacks and brute force attacks. Bhayo et al. [19] described a machine learning-based method for detecting DDoS assaults using an SDN-WISE IoT controller to mimic DDoS attack traffic generation in their work in 2023. Saran and Kesswani [20] introduced an Intrusion Detection System (IDS) for identifying multi-class intrusion attacks in the Internet of Things (IoT) using the Message Queuing Telemetry Transport - Internet of Things - Intrusion Detection System dataset (MQTT-IoT-IDS2020) in 2023. The experimented Intrusion Detection System's overall accuracy was 97.76%, 97.80%, 97.58%, 99.98%, and 97.58%, respectively, using the classifiers k-Nearest Neighbour (k-NN), Support Vector Machine (SVM), Naive Bayes (NB), Random Forest (RF), Decision Tree

(DT), and Stochastic Gradient Descent (SGD).

Recent studies[21-23] (Li et al., 2023; Ravi et al., 2024) have optimized sparse CNNs for IoT devices, achieving high accuracy with minimal energy footprints."

Although these models exhibit satisfactory performance on the chosen datasets and environments, the issues pertaining to model design and optimization continue to pose significant challenges in ensuring effective generalization against zero-day attacks. The objective of this paper is to devise a model that strikes a balance between complexity and accuracy. The proposed model will have a low-complexity design and training parameters, while still being capable of achieving generalization and avoiding overfitting. To this end, the following objectives will be pursued:

1. To devise a methodology that automatically generates and optimizes model parameters, thereby enabling the model to effectively generalize across diverse datasets while mitigating overfitting issues.
2. To explore the potential of the latest Keras library for developing a lightweight sparse model, which can effectively address the challenges posed by design complexity and parameter count.
3. To leverage the power of Deep Learning (DL) in developing an Intrusion Detection System (IDS) that embodies the Green AI concept. This IDS will overcome the challenges posed by complexity and generalization in IoT detection by automatically generating and optimizing Convolutional Neural Network (CNN) models that can effectively operate across diverse datasets. The best-performing model will then be converted into a sparse model that embodies the principles of Green AI.

2. METHOD

This paper introduces a novel model, illustrated in Figure 2, that leverages a Convolutional Neural Network (CNN) architecture in conjunction with Long Short-Term Memory (LSTM) for effective feature extraction. The model comprises three convolution layers and three average pooling layers that are followed by an LSTM layer, classification layer, and two fully connected layers. A softmax output layer is employed for the classification stage. The proposed design aims to optimize three key training parameters, namely the activation function, batch normalization, and regularization techniques. These parameters are tuned heuristically by Artificial Intelligence (AI) to generate an efficient model. The flow diagram of the proposed approach is depicted in Figure 3.

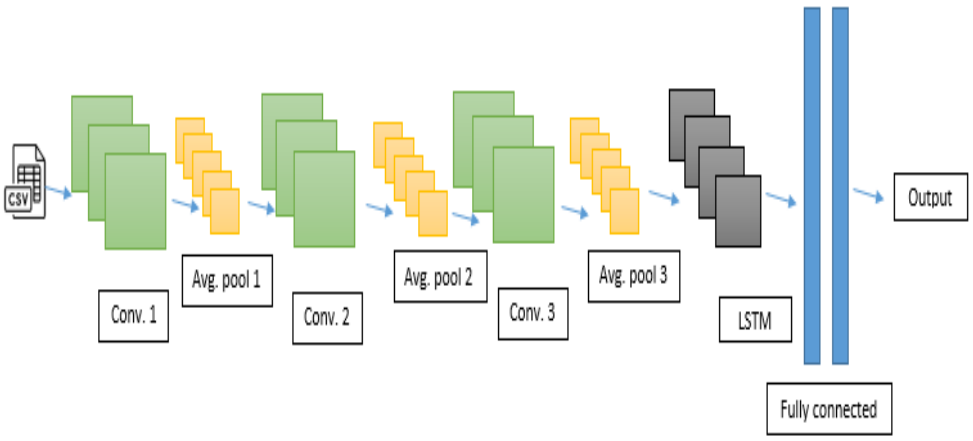

FIGURE 2. The CNN structure of the proposed model.

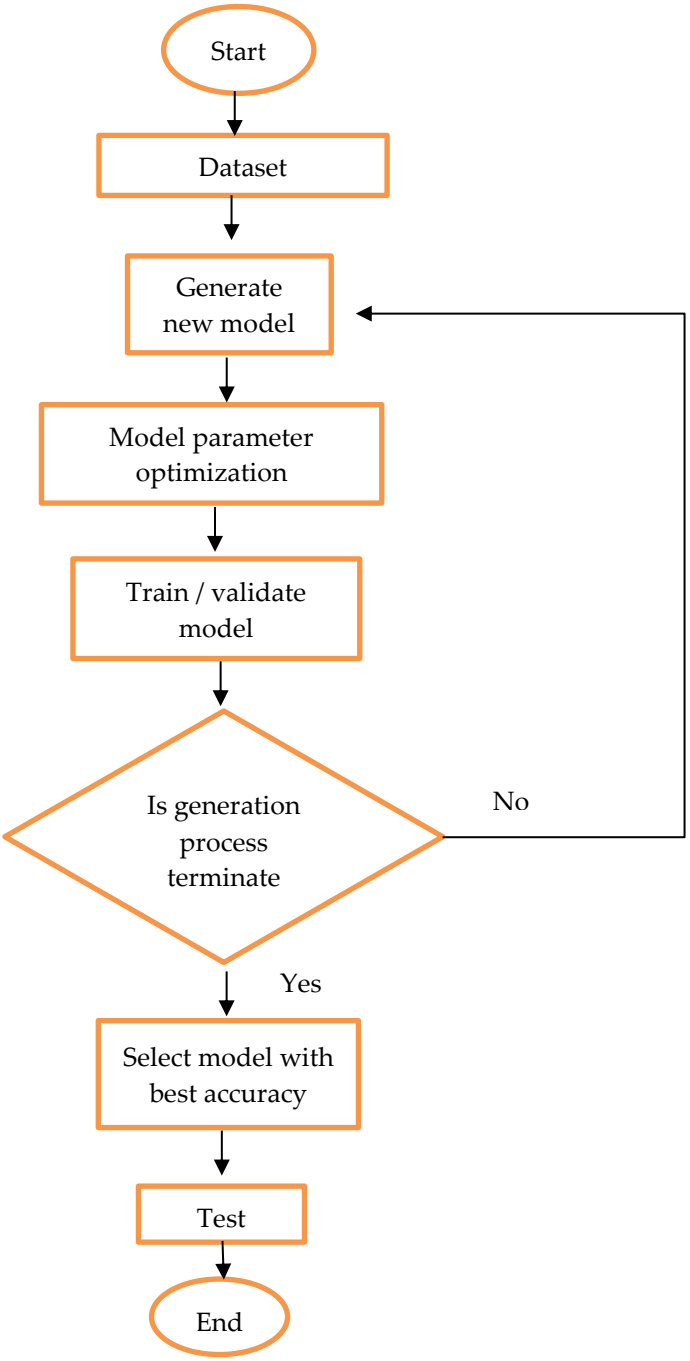

FIGURE 3. The flowchart of the proposed generation process

2.1 Model Generation Process

The crux of the proposed approach involves generating a new model in a depth-first search manner. This process is facilitated by building a tree from top to bottom, as illustrated in Figure 4. The tree is constructed using on/off possibilities and enables the generation of multiple models.

FIGURE 4. Tree model for a generation

Each node in the tree can assume one of two values, either zero or one. A value of zero implies that the node does not exist, whereas a value of one indicates its presence. Execution of the proposed approach commences from the top and proceeds downwards, with all activated possibilities (i.e., those with a value of one) being taken into account at each node. To illustrate the process, let us consider the first path starting from the Activation Function (AF) node, which always has a value of one. We begin by selecting the first type of Rectified Linear Unit (ReLU) and setting the normalization value to zero. We then proceed to take into account all possible values gradually, resulting in a total of four possibilities. This same mechanism is applied to the remaining types of activation functions (PReLU, CReLU, LeakyReLU, and ELU), as depicted in Figure 5. In summary, the total number of possibilities is computed as 20, obtained by multiplying the number of AFs (5) by the number of regularization techniques (4).

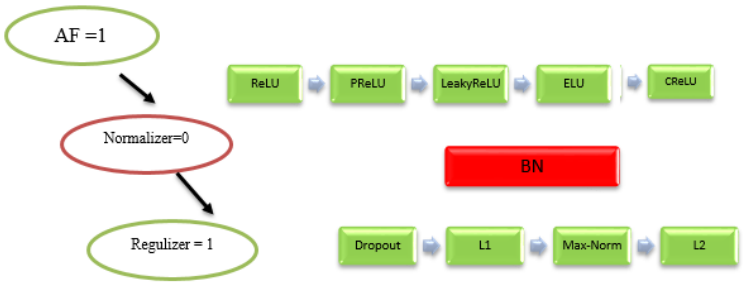

FIGURE 5. The first branch of the tree model

The second path involves setting the normalization value to one and the regularization value to zero. This implies taking into account all possible values of the activation function in combination with all possible values of normalization, as illustrated in Figure 6. The total number of possibilities for this branch is five.

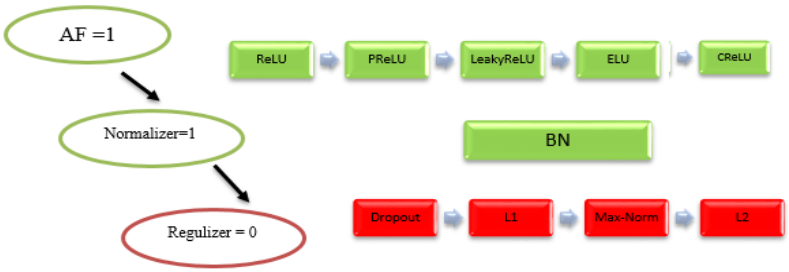

FIGURE 6. The second branch of the tree model

The final path is generated by setting the value of all nodes to one, thereby activating all nodes, as depicted in Figure 7. The total number of possibilities for this path is 20.

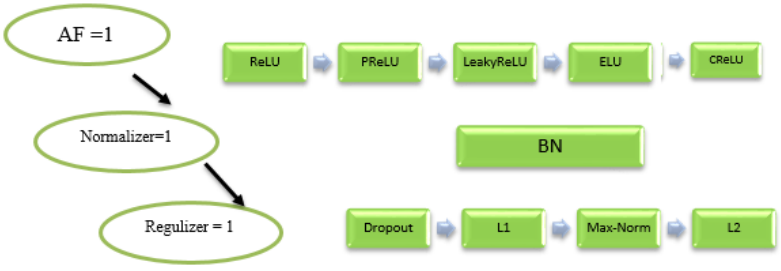

FIGURE 7. The third branch of the tree model

Algorithm 1 Model generation algorithm

Input: loop (1-5) to choose the type of AF,
Choose the type of normalization,
and loop (1-4) to choose the type of regularization
Output: z1, z2, z3
1 for i – 1 to 5 do
2 *z1 – choose one from (ReLU, PReLU, CReLU, LeakyReLU, ELU);*
3 **if n is 1 then**
4 **z2 -** *choose (BN);*
5 **if** r is 1 then
6 **end**
7 **for** k – 1 to 4 do
8 **z3 -** *choose one from (Dropped, max-norm, L1, L2);*
9 **end**
10 **end**
11 end
12 Return *z1*
13 Return *z2*
14 Return *z3.*

2.2 Model Parameter Optimization

The kernel and filter selection stage is crucial in identifying the optimal values from a set of available options. The kernel sizes (1x1, 3x3, and 5x5) enable the capture of patterns at different scales, while the filters (32, 64, and 128) are used to extract features from data samples using a nonlinear function. The exhaustive search approach involves using all probabilities of kernel size and filter types in each generated model. This process is repeated for every model generated.

Algorithm 2 Model Parameter Optimization algorithm

Input: loop (1-3) to choose the type of kernel size,
loop (1-3) to choose the type of filters.
Output: x1, x2.
1 for i – 1 to 3 do
2 ***x1*** *– choose one from (1 × 1, 3 × 3, and 5 × 5);*
3 **for** j – 1 to 3 do
4 **x2 -** *choose one from (32, 64, and 128);*
5 conv (x1, x2)
6 **end**
7 end
8 Return *conv*

2.3 Generate Sparse Model

Upon successful completion of the training and verification phase, the model is subjected to a series of operations aimed at converting it into a sparse model. These operations are designed to reduce the weight and number of variables in the model. All of the optimization strategies employed thus far yield dense models, wherein the majority of parameters are nonzero. To achieve a model that runs faster or takes up less memory, a sparse model is used. This

can be achieved by removing the small weights and setting them to zero.

Algorithm 4 Sparse Model Algorithm

Input: prepared dataset for training
Output: sparse model

1 Call Conv 1()
2 Check and update weight
3 Return best weight
4 Call Avg. pool 1()
5 Check and update weight
6 Return best weight
7 Call Conv 2()
8 Check and update weight
9 Return the best weight
10 Call Avg. pool 2()
11 Check and update weight
12 Return best weight
13 Call Conv 3()
14 Check and update weight
15 Return best weight
16 Call Avg. pool 3()
17 Check and update weight
18 Return best weight
19 Call fully connected ()
20 Check and update weight
21 Return best weight
22 Call fully connected ()
23 Check and update weight
24 Return best weight
25 Save sparse model

Due to communication vulnerabilities and resource restrictions, IoT devices have become a popular target for attacks. Several methods for detecting IoT threats using DL have been developed. In this thesis, an IoT attack detection CNN model using the MQTT-IoT-IDS2020, UNSW-NB-2015, and KDD IoT datasets is presented. To detect these attacks and create a Green AI detection system, Keras and the TensorFlow library were employed. Findings confirm that ReLU and leaky ReLUs give better runtime performance than ELU and PReLU are suitable for a huge training set, the model becomes a lightning-fast model at runtime when dropping BN and replacing the ELU AF with the leaky ReLU, and the model becomes very sparse by applying Dual Averaging (FTRL) with $\ell_1$regularization. In the future, the researcher will apply the model to IoT devices and measure their performance, this model is the base and can apply to transfer learning to run on unsupervised datasets, merge this model with incremental learning and can use evolutionary learning to generate new CNN structures and optimize the parameters. Also will explore quantum-inspired optimization (Otoom et al., 2025) to further reduce training overhead.

3. RESULTS AND DISCUSSION

3.1 Dataset

The paper utilizes the MQTT-IoT-IDS dataset for training and validating the generated model. To test the model and demonstrate its generalization capabilities, two additional datasets, namely UNSW-NB-2015 and KDD, were

employed.

3.2 Evaluation Metric

The following evaluation metrics [21], are used to assess the efficiency of the proposed model.

$$\textbf{Recall / Sensitivity} = \frac{TP}{TP+FN} \quad \textbf{(a)}$$

$$\textbf{Selectivity} = \frac{TN}{FP+TN} \quad \textbf{(b)}$$

$$\textbf{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \quad \textbf{(c)}$$

$$\textbf{F-Measure} = \frac{2*Precision*Recall}{Precision+Recall} \quad \textbf{(d)}$$

TP is True Positive.
TN is True Negative.
FP is False Positive.
FN is a False Negative.

3.3. Experimental Setup

In this paper, a Python program is used to implement the proposed system on a Hp laptop 11 with 8.00 GB RAM, Windows 11 with the 64-bit operating system, x64-based processor, and a Core(TM) i7-1165G7 @ 2.80GHz 2.70 GHz processor. For the parameters used Learning rate with 0.001, Pool Size with 2, Adam, Admax, and FTRL Optimizer. In the feature extraction model setting, using a filter with a kernel 5*5 and filter 128 for all layers using 100 epochs.

3.4 Training on MQTT-IoT IDS2020

This section summarizes the finding of the model generation process with their parameter optimization for the benchmarked data set. Tables (1, 2, 3, 4, 5, 6, 7, 8, 9, and 10) list these models according to the depth-first search procedure that is proposed in section 2.1.

Table (1) shows the results of the execution of several possibilities on the proposed model using all types of the kernel (1*1, 3*3, and 5*5) and filters (32, 64, and 128) for 100 epochs. With a careful look at the table notes when the size of the kernel is small (1*1), the accuracy results are less than 90%, starting from 80% and increasing by increasing the number of filters as shown in the first three results. Then noticed a significant improvement in accuracy when the filter size in each layer is different as shown in the fourth result. This is because dealing with a small kernel is related to the process of working on the method of ordinary neural networks non-linear communication, and it is noted that the time of execution increases regularly with this in mind that the same process happened when using kernels (3*3) and (5*5).

The last four results test randomly to execute more possibilities as shown in Table 1. The best results are taken when the kernel 5*5 and filter 128 for all layers using 100 epochs. These best parameters are executed in all models.

TABLE 1. Results of the proposed model using ReLU, BN and L1.

Setting of Parameter	Loss %	Accuracy %	Execution Time (h, m, s)
Kernel 1*1 & Filter 32 (all layers)	40.10	83.27	0:18:08
Kernel 1*1 & Filter 64 (all layers)	41.19	83.83	0:21:45
Kernel 1*1 & Filter 128 (all layers)	42.71	82.45	0:29:59
Kernel 1*1 & Filter 32 & Filter 64 & Filter 128	32.14	88.02	0:16:19
Kernel 3*3 & Filter 32 (all layers)	24.37	91.13	0:18:00
Kernel 3*3 & Filter 64 (all layers)	21.29	92.81	0:21:58
Kernel 3*3 & Filter 128 (all layers)	18.62	94.34	0:43:37

Kernel 3*3 & Filter 32 & Filter 64 & Filter 128	20.96	93.24	0:20:11
Kernel 5*5 & Filter 32 (all layers)	25.30	91.94	0:12:51
Kernel 5*5 & Filter 64 (all layers)	19.51	93.18	0:17:37
Kernel 5*5 & Filter 128 (all layers)	15.81	95.00	0:33:37
Kernel 5*5 & Filter 32 & Filter 64 & Filter 128	18.85	93.94	0:32:39
Kernel 1*1 & Filter 32, Kernel 3*3 & Filter 64, Kernel 5*5 & Filter 128	31.48	88.54	0:17:20
Kernel 5*5 & Filter 32, Kernel 3*3 & Filter 64, Kernel 1*1 & Filter 128	25.45	90.52	1:39:45
Kernel 3*3 & Filter 64 & Filter 64 & Filter 128	23.26	93.02	0:21:51
Kernel 5*5 & Filter 64 & Filter 64 & Filter 128	19.03	93.65	0:29:06

Table 2 applied all types of AF and BN and all types of regulizers on the MQTT-IoT-IDS2020 dataset. A careful look at the table shows that the first five results show that the accuracy starts from 91-95%. PReLU gives higher accuracy than ReLU and others but consumes more time than ReLU so ReLU is the best and still gives the best in the rest results, while ELU is unsuitable with BN and dropout.

TABLE 2. Results of the proposed model with BN and regulization

Model	Loss %	Accuracy %	Execution time (h, m, s)
ReLU, BN, L1	15.81	**95.25**	0:33:37
CReLU, BN, L1	16.54	94.13	1:09:03
LeakyReLU, BN, L1	23.06	91.94	0:41:43
PReLU, BN, L1	13.84	95.41	1:51:07
ELU, BN, L1	24.28	91.06	0:39:24
ReLU, BN, L2	20.73	92.02	0:44:05
CReLU, BN, L2	22.92	91.78	1:33:14
LeakyReLU, BN, L2	20.61	92.32	0:34:46
PReLU, BN, L2	14.10	**95.15**	0:39:01
ELU, BN, L2	19.98	92.10	2:00:34
ReLU, BN, max-norm	12.67	95.49	0:44:01
CReLU, BN, max- norm	17.80	94.81	1:28:50
LeakyReLU, BN, max- norm	19.92	92.61	1:27:48
PReLU, BN, max- norm	12.38	**95.71**	0:40:34
ELU, BN, max-norm	19.55	93.30	0:38:40
ReLU, BN, dropout	13.52	95.21	0:35:10
CReLU, BN, dropout	15.67	94.18	1:50:41
LeakyReLU, BN, dropout	21.31	91.86	0:45:47
PReLU, BN, dropout	12.74	**95.47**	0:43:50
ELU, BN, dropout	29.62	88.46	1:12:45

Table 3 applied all types of AF and BN without using regulizers on the same dataset. As shown in the Table below, BN makes the neural network slower predictions. ReLU gives the best results when use with BN.

TABLE 3. Results of the proposed model with BN only

Model	Loss %	Accuracy %	Execution time (h, m, s)
ReLU, BN	13.45	**95.33**	0:35:17
CReLU, BN	15.39	94.41	1:12:56
LeakyReLU, BN	18.98	92.53	0:29:29
PReLU, BN	12.08	95.53	3:12:46
ELU, BN	23.38	91.04	0:39:14

Table 4 applied all types of AF and regulizers without using BN on the same dataset. As shown in the Table below the execution time is less in all results when dropped BN. PReLU still gives the best results with all types of regulizers.

TABLE 4. Results of the proposed model with regulization

Model	Loss %	Accuracy %	Execution time (h, m, s)
ReLU, L1	21.03	91.84	0:26:48
CReLU, L1	2350	91.36	2:10:13
LeakyReLU, L1	26.57	89.41	0:28:19
PReLU, L1	18.22	**93.51**	0:55:04
ELU, L1	30.66	87.81	0:29:03
ReLU, L2	20.61	92.83	1:10:13
CReLU, L2	27.59	89.19	1:18:08
LeakyReLU, L2	23.68	90.80	0:25:01
PReLU, L2	18.08	**93.32**	0:33:29
ELU, L2	33.56	86.25	0:28:49
ReLU, max-norm	17.93	93.70	2:26:48
CReLU, max-norm	31.73	88.62	1:30:15
LeakyReLU, max-norm	49.30	44.94	0:26:15
PReLU, max-norm	15.12	**94.60**	0:45:17
ELU, max-norm	41.15	83.96	2:30:23
ReLU, dropout	18.23	93.86	1:41:59
CReLU, dropout	24.51	90.72	1:30:07
LeakyReLU, dropout	43.12	88.22	0:31:02
PReLU, dropout	49.30	**94.44**	0:37:44
ELU, dropout	31.62	87.28	0:30:00

Applying the test dataset to the proposed models gives 95.87% accuracy with a number of parameters 41166.

Finally, to generate a sparse model satisfying the green AI principle we utilize L1 regulizer, removing BN, and using LeakyReLU and FTRL. For generalization testing this model is trained and validated UNSW-NB-2015 and KDD-2017, Table 5 lists the results.

TABLE 5. Results of all sparser models using all datasets.

Datasets	Accuracy %
MQTT-IoT-IDS2020	97.44
UNSW-NB-2015	96.64
KDD-2017	99.95

3.5 Comparative to Related Works

A comparative analysis has been conducted among the proposed model and the most related works in the literature as given in Table 6. Looking at the table below notice that the proposed models are best with their parameters and structures.

TABLE 6. Comparison of Evaluation parameters with other models.

Resources	No. of the dataset used	No. of the proposed model	Is the model static or dynamic?	Accuracy
Proposed model	Three datasets 1) MQTT-IoT-IDS2020 2) UNSW-NB 2015 3) KDD-2017	One model with five AF (ReLU, CReLU, PReLU, LeakyReLU, and ELU)	Dynamic	**Model 1** (MQTT-IoT-IDS2020 with 97.44 %, UNSW-NB 2015 with 96.64%, KDD-2017 with 99.95 %)
[13]	One dataset (UNSW-NB15)	Two models (RNN, LSTM)	static	95.71%
	Two datasets (NSL-KDD, UNSW-NB15)	Five models (SVM, MLP, RBM, SAE, WND)	static	SVM 81.5% MLP 86.7% RBM 86.1% SAE 88.2% WND 91.2%
	KDD CUP 99	Three models (DNN, CNN, LSTM)	Static	DNN 99.94% CNN 98.7% LSTM 99.5%
	Two datasets (KDDCUP 99, UNSW-NB15)	One model (DSAE)	Static	KDD CUP99 99.931% UNSW-NB15 89.711%
[15]	CIC-IDS-2018	One model	Static	98.9 %
[14]	CICIDS2017	Two models	Static	Model 1 99.10%. Model 2 99.38%
[16]	Three datasets 1) captures from the university campus 2) from a medium size company 3) CICIDS2017.	One model with three AF (ReLU, Softmax, and SRS	Static	98%
[17]	One dataset ToN-IoT dataset	hybrid feature selection scheme that combines statistical test-based filter approaches, such as Chi-Square (2), (PCC), and (MI), with a metaheuristic approach based on (NSGA-II).		90.48%
[18]	MQTT-IoT-IDS2020	DL		99%
[19]	DDoS attack traffic	Naive Bayes (NB), Decision Tree (DT), and Support Vector Machine (SVM) algorithms		NB with 97.4% SVM with 96.1% DT with 98.1%

[20]	MQTT-IoT-IDS2020	k-Nearest Neighbour (k-NN), Support Vector Machine (SVM), Naive Bayes (NB), Random Forest (RF), Decision Tree (DT), and Stochastic Gradient Descent (SGD)		k-NN with 97.76% SVM with 97.80% NB with 97.58% RF with 99.98% DT with 99.98% SGD with 97.58%
[21]	CIC-IDS2017	Federated lightweight CNN-LSTM	dynamic	98.2%
[22]	UNSW-NB15 and MQTT datasets	GreenID-Net	Sparse	96.8%
[23]	KDD IoT-2017	AutoSec	Automated	97.3%

4. CONCLUSION

Due to communication vulnerabilities and resource restrictions, IoT devices have become a popular target for attacks. Several methods for detecting IoT threats using DL have been developed. In this thesis, an IoT attack detection CNN model using the MQTT-IoT-IDS2020, UNSW-NB-2015, and KDD IoT datasets is presented. To detect these attacks and create a Green AI detection system, Keras and the TensorFlow library were employed. Findings confirm that ReLU and leaky ReLUs give better runtime performance than ELU and PReLU are suitable for a huge training set, the model becomes a lightning-fast model at runtime when dropping BN and replacing the ELU AF with the leaky ReLU, and the model becomes very sparse by applying Dual Averaging (FTRL) with ℓ1regularization. In the future, the researcher will apply the model to IoT devices and measure their performance, this model is the base and can apply to transfer learning to run on unsupervised datasets, merge this model with incremental learning and can use evolutionary learning to generate new CNN structures and optimize the parameters. Also will explore quantum-inspired optimization (Otoom et al., 2025) to further reduce training overhead.

5. REFERENCES

1. Daibo, Masahiro. "Toroidal vector-potential transformer." 2017 Eleventh International Conference on Sensing Technology (ICST). IEEE, 2017.
2. W. Zhou, et al., " The effect of IoT new features on security and privacy: New threats, existing solutions, and challenges yet to be solved," IEEE Internet Things J., vol. 6, no. 2, pp. 1606_1616, Apr. 2019
3. M. M. Najafabadi, et al., ``Deep learning applications and challenges in big data analytics," J. Big Data, vol. 2, no. 1, Dec. 2015, doi: 10.1186/s40537-014-0007-7.
4. J. Lansky et al., "Deep Learning-Based Intrusion Detection Systems: A Systematic Review," in IEEE Access, vol. 9, pp. 101574-101599, 2021, doi: 10.1109/ACCESS.2021.3097247.
5. M. Uğurlu and İ. A. Doğru, "A Survey on Deep Learning Based Intrusion Detection System," 2019 4th International Conference on Computer Science and Engineering (UBMK), 2019, pp. 223-228, doi: 10.1109/UBMK.2019.8907206.
6. Otoum, Yazan, Dandan Liu, and Amiya Nayak. "DL-IDS: a deep learning–based intrusion detection framework for securing IoT." Transactions on Emerging Telecommunications Technologies 33.3 (2022): e3803.
7. Kolosnjaji, Bojan, et al. "Empowering convolutional networks for malware classification and analysis." 2017 International Joint Conference on Neural Networks (IJCNN). IEEE, 2017.
8. Wang, Wei, et al. "Malware traffic classification using convolutional neural network for representation learning." 2017 International conference on information networking (ICOIN). IEEE, 2017.
9. Wang, Wei, et al. "End-to-end encrypted traffic classification with one-dimensional convolution neural networks." 2017 IEEE international conference on intelligence and security informatics (ISI). IEEE, 2017.
10. Xin, Y., et al. "and C. Wang." Machine learning and deep learning methods for cybersecurity. IEEE Access 6 (2018): 35365-35381.
11. Roopak, Monika, Gui Yun Tian, and Jonathon Chambers. "Deep learning models for cyber security in IoT networks." 2019 IEEE 9th annual computing and communication workshop and conference (CCWC). IEEE, 2019.

12. Liu, Junyi, Shiyue Liu, and Sihua Zhang. "Detection of IoT botnet based on deep learning." 2019 Chinese Control Conference (CCC). IEEE, 2019.
13. Uğurlu, Mesut, and İbrahim Alper Doğru. "A survey on deep learning based intrusion detection system." 2019 4th International Conference on Computer Science and Engineering (UBMK). IEEE, 2019.
14. Patidar, Sanjay, and Inderpreet Singh Bains. "Web security in IoT networks using deep learning model." 2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT). IEEE, 2020.
15. Kabir, Solaiman, et al. "A Convolutional Neural Network based Model with Improved Activation Function and Optimizer for Effective Intrusion Detection and Classification." 2021 International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE). IEEE, 2021.
16. Mendonça, Robson V., et al. "Intrusion detection system based on fast hierarchical deep convolutional neural network." IEEE Access 9 (2021): 61024-61034.
17. Arun Kumar Dey, Govind P. Gupta, Satya Prakash Sahu, Hybrid Meta-Heuristic based Feature Selection Mechanism for Cyber-Attack Detection in IoT-enabled Networks, Procedia Computer Science, Volume 218, 2023, Pages 318-327, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2023.01.014.
18. Ahmed Fawzi Otoom, Wafa' Eleisah, Emad E. Abdallah, Deep Learning for Accurate Detection of Brute Force attacks on IoT Networks, Procedia Computer Science, Volume 220, 2023, Pages 291-298, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2023.03.038.
19. Jalal Bhayo, Syed Attique Shah, Sufian Hameed, Awais Ahmed, Jamal Nasir, Dirk Draheim, Towards a machine learning-based framework for DDOS attack detection in software-defined IoT (SD-IoT) networks, Engineering Applications of Artificial Intelligence, Volume 123, Part C, 2023,106432, ISSN 0952-1976, https://doi.org/10.1016/j.engappai.2023.106432.
20. Naveen Saran, Nishtha Kesswani, A comparative study of supervised Machine Learning classifiers for Intrusion Detection in Internet of Things, Procedia Computer Science, Volume 218, 2023, Pages 2049-2057, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2023.01.181.
21. Li, Y., Xu, Y., Liu, Z., Hou, H., Zheng, Y., Xin, Y., ... & Zhao, Y. (2023). Robust IoT intrusion detection with lightweight neural networks and federated learning. IEEE Internet of Things Journal, 10(9), 7609–7621.
22. Ravi, V., Pham, T. D., & Alazab, M. (2024). GreenID-Net: Energy-efficient sparse CNN for real-time IoT intrusion detection. Sustainable Computing: Informatics and Systems, 42, 100926.
23. Khan, F. A., Gumaei, A., Al-Rakhami, M., & Boukiff, S. (2024). AutoSec: Automated CNN architecture search for IoT intrusion detection. Journal of Network and Computer Applications, 224, 103881.
24. Otoom, M., Alrawashdeh, T., & Alzu'bi, D. (2025). Quantum-inspired optimization for green AI-based intrusion detection in IoT. Future Generation Computer Systems, 158, 241–253.