International Journal of Applied Mathematics

Volume 38 No. 9s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

AGGCR_NET: AN ADAPTIVE GRADIENT-GATED CONVOLUTION RECURRENT NETWORK WITH AN ADAPTIVE METAHEURISTIC ALGORITHM-BASED IOT DEVICE IDENTIFICATION SYSTEM

[1*]Indira, [2]Dr. A. K. Sampath

[1*]Research Scholar, School of Computer Science and Engineering, Presidency University, Bangalore, India

[2] Professor, School of Computer Science and Engineering, Presidency University, Bangalore, India

Abstract

In the smart environment, the role of Internet of Things (IoT) is inevitable in daily tasks but security and privacy are challenging due to network traffic exposing the vulnerable IoT device to hackers or intruders. This leads to security threats and the demand for developing IoT device identification schemes. Therefore, the proposed model has developed an adaptive gradient-Gated Convolution Recurrent Network (AGGCR_Net) for identifying IoT devices linked to the vulnerable network. In addition, an enhanced feature extraction based on Gaussian Bhattacharyya distance with the entropy provides highly effective features to enhance the classification performance. Moreover, an incorporation of the novel adaptive puma optimization (APO) algorithm provides highly effective feature selection. Finally, Densely Connected GRU in the AGGCR_Net plays the primary role in classification. The Python platform is utilized for implementing the proposed IoT device identification system where performance of proposed model is evaluated in terms of Specificity, accuracy, Sensitivity, precision,MCC, FNR, F-Measure, NPV, G-Mean, and FPR.

Keywords*:* Adaptive gradient, Convolution, Gated Recurrent Unit, Gaussian Bhattacharyya distance, IoT attacks device detection.

1. INTRODUCTION

In computing history, IoT is seen as a fast-emerging paradigm. IoT has advanced significantly across several technology domains. It is the result of the convergence of the internet with hundreds of billions of devices from different systems, such as smart grids, smart cars, smart homes, and smart health care. [1][2]. Understanding the devices connected to a network is necessary for network management. Attacks by insiders may be performed against a vulnerable networked device. Because of this, the need for diverse IoT applications necessitates increased security and protection [3], which in turn calls for precise network traffic classification to identify attacks early and implement appropriate countermeasures [4]. Considering how widely IoT devices are used, malicious manipulations could have a significant impact on the security and resilience of the entire Internet.Specifically, IoT devices require more frequent monitoring to prevent/detect viruses and intrusions because they lack the processing power of conventional networked devices. Securing IoT networks is

Received: August 06, 2025

Volume 38 No. 9s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

now essential due to the rise in IoT threats like Mirai. Over 600,000 machines were affected by Mirai at its height globally [5] [6].

An attacker can also use IoT device recognition and passive network traffic analysis to identify vulnerable IoT devices. Therefore, expanding the uses of IoT requires detecting attacks to shield devices from malicious actions [7]– [9]. The growing prevalence of IoT devices in our culture and their lax security contribute to the interest in developing security solutions for these gadgets. Network security researchers are researching novel security strategies based on deep learning (DL) as internet attacks become more sophisticated and diverse [10]. The emergence of DL offers innovative answers to security issues in large data contexts, whereas standard web attack detection systems reveal vulnerabilities in these settings. In particular, an advanced machine learning technique called DL provides [11] a special ability to automatically extract features from high-volume, high-speed network traffic produced by heterogeneous, networked IoT devices [12] [13]. Furthermore, DL-based security methods are heterogeneity-tolerant since they can autonomously learn diverse characteristics from unstructured data.

As the world's connected device count rises, hackers have discovered numerous ways to get access to company networks [14]. These vulnerabilities could be caused by weak passwords and a lack of robust security standards from manufacturers [15]. Sophisticated IoT attack vectors may occasionally be able to deflect detection in IoT networks due to the absence of standards in production of IoT devices. Because of this, there are now far more possible risks in the domain, but there has also been a substantial amount of study done on threat mitigation techniques [16]. Nevertheless, security risks like DDoS assaults and privacy breaches have gotten worse as a result of widespread use of IoT devices [17][18]. It might be challenging to directly install typical protection techniques like firewalls and anti-virus software in a simple IoT system since it may have a high number of low-power and resource-constrained IoT devices [19]. This makes the system vulnerable to attacks on security. Big data analysis-driven DL algorithms demonstrate an exceptional ability to identify aggressive behavior in high volumes of traffic. The advancement of IoT network security has been aided and facilitated by these deep-learning techniques.

The main aim of this research is to improve detection of IoT-linked devices. In this research, paper proposed an adaptive gradient-gated Convolution Recurrent Network (AGGCR_Net) for identifying IoT devices linked to the vulnerable network.But this method iscomputationally intensive during the high dimensional data collection this difficulty has beenovercome by introducing the adaptive PUMA optimization algorithm. The main aspects of this algorithm are to optimize the relevant feature selection and improves the detection performance. This can ability to handle the large-scale dimensional dataset and thus improve the overall system performance. Moreover, the densely connected GRU which is integrated with the proposed AGGCR_Net method provides better convergence and more efficient detection the IoT attack devices.This paper's contribution is as follows,

Received: August 06, 2025

International Journal of Applied Mathematics

Volume 38 No. 9s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

- Adaptive gradient-gated Convolution Recurrent Network (AGGCR_Net) isthe proposed method, which improves network training for IoT device detection by enhancing performance and convergence speed. It involves convolution and recurrent networks for higher data pre-processing and accurate feature extraction. Data pre-processing information can be managed and specified in IoT networks through a gating mechanism.
- The densely connected layers of AGGCR_Net play a crucial role in enabling high-dimensional feature mapping and intricate feature transformation, which enhances the network's capacity to precisely categorise IoT devices through extensive spatial and temporal information integration.
- Adaptive puma optimizationalgorithm consists of the metaheuristic algorithm which interprets the characteristics of the hunting behavior of pumas. The APO helps to optimize the feature selection whichenhances the ability to extract the relevant feature and to improve the device detection accuracy. This algorithm has effectively balanced the stage of exploitation and the exploration space to effectively find the optimal and the near-optimal solution.
- Densely Connected GRUis an advanced neural network architecture thatcombines the function of DenseNet with the capabilities of Gated Recurrent Units (GRU). This can efficiently perform parallel processing in training large sequential data due to its dense connectivity pattern which can lead to improved computational performance. And reduce the risk of information loss, thus leading to better detection accuracy and robustness**.**

The following sections are prepared as follows: Section 2 explores relevant research and literature reviews, Section 3 introduces the proposed framework, Section 4 provides detailed analysis of the observed results and discussions, and Section 5 offers the final assessment of this study.

2. LITERATURE SURVEY

The literature review of DL-based IoT-linked device detection systems is covered in this chapter. The attack classification models and DL optimization strategies are covered in detail. The research gaps are determined from this analysis of the literature, and in the chapters that follow, research projects are suggested.

In (2021), Arunkumar Thangavelu and E. S. Phalguna Krishna [20] investigated DoS attacks in IoT systems. They gave an example of the assaults and security problems that affect IoT devices. They proposed two techniques, a Firefly optimization algorithm, and a hybrid meta-heuristic lion optimization system, to detect the attack. They conducted the analysis using IoT and NSL-KDD datasets. Their algorithm identified the attacks and reached their maximum performance.

In (2022), Xiuzhang Yang*et al*. [21]built an IDS system with a statistical feature selection model and a knowledge graph. To extract the semantic relationship between the features, a fusion model was created. They classified the long-distance dependence using CNN and BiLSTM. They asserted that the model can identify attacks in real-time by adjusting the weights of various features.

Received: August 06, 2025

Volume 38 No. 9s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

In (2022), Neha Yadav *et al.*[22] designed a system to detect actual intrusions worldwide. To detect network assaults in 5G connections, it was advised to employ an autoencoder in conjunction with a DNN-based attack detection model. The benchmark dataset from UNSW 2015 was used to evaluate the model. The recall, accuracy, and precision, of AE model were enhanced while the detection time was decreased. In terms of attack detection accuracy, this model achieved 99.76%.

In (2021), Segun I. Popoola *et al.* [23] proposed a hybrid DL model to identify attacks in huge datasets. Bot-IoT dataset was used to test model. Bidirectional LSTM Autoencoder was used to reduce dimensionality, and Bidirectional LSTM was used to classify network data. There have been binary and multiclass classifications. They demonstrated how memory space needed to supply massive amounts of network traffic data was greatly decreased by the LAE.

Xiu Kan *et al.* [24] proposed a technique, APSO-CNN, for the identification of attacks on IoT devices by Adaptive Particle Swarm Optimisation Convolutional Neural Network in 2021. Their proposed algorithm is for optimizing parameters of one-dimensional CNN. They determine the fitness due to the composition approach by considering the cross-entropy loss value of the training CNN that was attained. For that purpose, they designed one evaluation methodology that takes into account the predictions and predictive labels while testing the proposed APSO-CNN algorithm. After that, the simulation results demonstrate the utility and reliability associated with the approach by indicating how IoT threats could be detected.

In 2022, Vinayakumar Ravi *et al*. [25] proposed a DL-based approach for attack detection and classification into the appropriate attack categories. In the first step, internal feature representations from GRU DL layers were extracted since the model used kernel-PCA to get the best features. After that, a fully connected network was used in order to perform detection and classification of an attack by fusing the features. From a performance perspective, the proposed feature-fused GRU network had relatively better performance in comparison to the GRU model.

Semih Cakir *et al.* [26] in 2020 proposed a DL-based GRU network model for preventing HF attacks on the RPL protocol of IoT networks. Furthermore, the suggested model might clarify things and inspire fresh approaches to the identification and defense against routing attacks in IoT settings. Effective techniques for detecting and averting RPL attacks in the IoT are DL approaches based on GRU. Scalability will be reduced, though, if the number of nodes is significantly raised.

In (2022), Josue Genaro Almaraz-Rivera *et al.*[27] created a novel smart IDS by simulating IoT sensors in a smart home environment and using the Bot-IoT dataset. Models from DL and machine learning (ML) reinforce the system. Class weights were added to the training data to address the imbalanced nature of the dataset; a bigger weight value was assigned to the class with fewer samples. But there's a chance that this method of over-tuning could lead to weights that don't generalize as well as they should.

Received: August 06, 2025

International Journal of Applied Mathematics

Volume 38 No. 9s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

2.1 Challenges:

- In IoT networks, the systems have encountered significant difficulties. To safeguard IoT networks from numerous online threats, research into more advanced solutions is therefore desperately needed [28].
- Device detection performance is negatively impacted by the high complexion of the training period combined with the huge spatial dimensionality of data collection.
- Security systems encounter difficulties when addressing IoT attacks because hackers modify earlier attacks in subtle ways that are invisible to security solutions.
- The impact of interdependent behaviors on IoT security is not well understood by most researchers [29]. However, due to their interdependent behaviours, it is difficult to apply privilege management and static access control strategies to Internet of Things devices or to build a clear protective perimeter for them.
- Researchers must take into account not unbiassed the security vulnerabilities inherent in a particular protocol, but also the possible security threats linked to alternative protocols.

3. PROPOSED METHODOLOGY

In this research, the aim goal is to improve IoT attack detection in linked devices using the proposed AGGCR_Net method based on DL approaches. Initially, the input of the IoT device network traffic datais collected in the dataset named IoT device identification classification dataset [30]. The collected data is fed forward to the pre-processing stage to improve quality of dataset using the three methods likeMissing value imputation, data cleaning, Data Normalization, and standardization. In this pre-processing, data cleaning is performed to reduce the noise and improve the data quality. Missing value imputation goal is to minimize these effects by estimating missing values thus providing a more reliable basis for detecting deviations indicative of attacks. Both normalization and standardization contribute to better model performance by reducing the impact of varying scales and distributions of data features, thereby improving stability and convergence. After,pre-processed:the mean, median, standard deviation, kurtosis, skewness, entropy, and Gaussian Bhattacharyya distance are used to facilitate effective data representation, reduce dimensionality, and extract meaningful features to quantify distributional properties and uncertainty of complex datasets. In this research, the paper proposed the novelty of adaptive PUMA optimization, a feature-selecting algorithm can lead to faster training time which helps to fine tuning capacity to extract pertinent features that ensure to reductionof the overfitting and computational cost of performance. Finally, the selection features are performed with the novel approaches of the proposed AGGCR_Net method. The proposed method is integrated with the densely connected Gated Recurrent Unit (GRU) and Convolutional Neural Network, and that can ability to handle temporal data and optimize learning processes which can lead to faster convergence and better performance in training

Received: August 06, 2025

Volume 38 No. 9s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

network on IoT device detection tasks. The proposed methodology is diagrammatically represented in Figure 1.

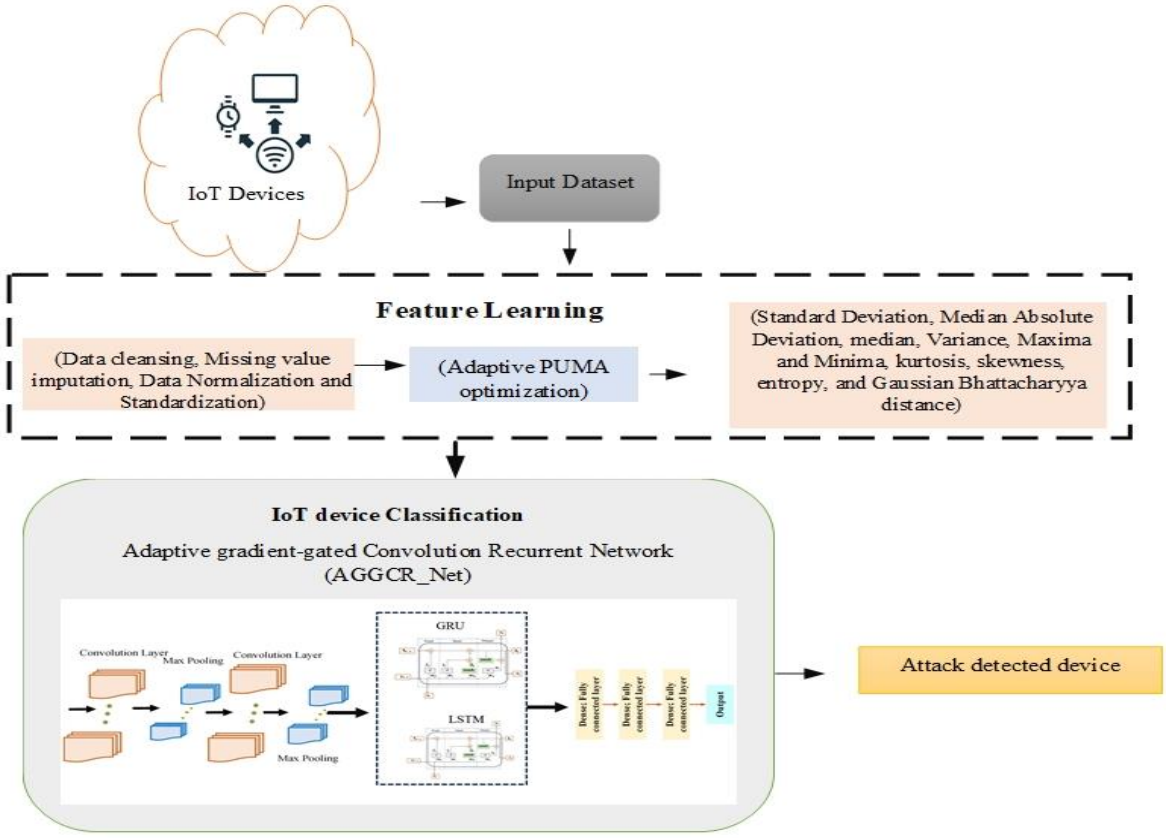


Figure 1: The proposed methodology

3.1 Feature Learning

3.1.1. Data pre-processing

A dataset is pre-processed once it has been collected. It is difficult to evaluate the initial input data and find attacks with all the characteristics that could be either numerical or non-numerical because of the large amount of network traffic. The present study implemented specific pre-processing methodologies, including Missing value imputation, data cleaning, Data Normalization, and Data Standardization, to mitigate the non-numerical feature limitation issue and enhance the quality of the dataset.

a. Data Cleaning

Data cleaning can eliminate unnecessary data and normalize the dataset. Common processes include removing duplicate values and completing any gaps. Both testing set and training set undergo data cleaning in this work. Even though DL methods work on real number vectors, the symbolic properties are converted to numerical values. Regular and anomalous network traffic is separated based on feature identification.

$$y_i' = \log(y_i) \quad (1)$$

Received: August 06, 2025

International Journal of Applied Mathematics

Volume 38 No. 9s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

b. Missing value imputation

Imputation techniques involve filling in the missing data to produce a complete data matrix that may be subjected to DL techniques for analysis. When one value is substituted for a missing data element without offering a clear model for the partially absent data, this technique is known as a single imputation. Mean or median replacement is a commonly employed technique that substitutes the variable mean score for missing data. Equation 2 is the formula for the mean imputation.

$$\bar{y} = \frac{\sum_{i=1}^{k} \hat{y}_i}{k} \quad (2)$$

Assuming that the $i\ th$missing entry is represented by the value $\hat{y}_i$. The noticed values in the column are denoted by y_j. The quantity that is not missing is denoted by k.

c. Data Standardization

Standardization and normalization are two different concepts. It's noteworthy to note that standardization means that 0 is used for the mean and 1 for the standard deviation. A crucial step in pre-processing of data is Data Standardization Process, which is mostly utilized to provide a feature scaling. This ensures that the data are almost on the same scaling, giving each character a comparable weight. Standardization facilitates the processing of data by most DL algorithms. The Z-score normalization approach, which rescales all of the characteristics to ensure that the mean and standard deviation have distribution values between 0 and 1, is applied in this research, as follows

$$y_s = \frac{y - m(y)}{std(y)} \quad (3)$$

where y represents the feature's initial value. Among the feature values, m is their mean. $std$is the standard deviation of the feature values.

d. Data Normalization

It is pre-processing step in which current range is used to create new range. Dataset contains uncertain and partial data; therefore, to improve the quality, the missing data must be repaired by eliminating the irrelevant data. Therefore, the Min Max normalization effectively integrates and normalizes the dataset. Maintaining more variability in forecasting and anticipating paths of a new range requires use of the normalizing strategy. The process of scaling a dataset so that the normalized data values fall between 0 and 1 is referred to as normalization. By subtracting the smaller value from the variable that requires normalization, the normalization is obtained. When the prior result is divided by the latter, lesser value is subtracted from highest value. After normalization, Min-Max scaling is expressed as

$$y'_{ij} = \frac{y_{ij} - y_i^{min}}{y_i^{max} - y_i^{min}} \quad (4)$$

Received: August 06, 2025

Volume 38 No. 9s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

where j is a dataset record, i is a feature in the dataset, and y_i^{max} and y_i^{min} are the feature'sminimum and maximum values of i. As a result, can characterize the distribution and relevance of the related features by mapping the values of each continuous feature into the interval between 0 and 1.

3.1.2 Feature Extraction

The process of reducing the number of dimensions or features in a dataset after pre-processing is known as feature extraction. By extracting and projecting the relevant and significant data that is scattered throughout the raw input features into fewer features, it seeks to minimise information loss. The next paragraphs explain the feature extraction methods that were employed, as well as mean, median, standard deviation, kurtosis, skewness, entropy, and Gaussian Bhattacharyya distance.

a. Standard Deviation (STD): The standard deviation of a sample class is given by:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{k}(y_i - \bar{y})^2}{k-1}} \quad (5)$$

where k = number of samples, y_i = ith sample value, and y = sample mean.

b. Median Absolute Deviation (MAD): The median absolute deviation is given by:

$$MAD = \sum_{i=1}^{k} \frac{y_{i-MD}}{k} \quad (6)$$

where k = number of samples, y_i = ith sample value and MD = mediansize

c. Median: The median of a class is given by:

$$MD = L_b + E\left(\frac{\frac{N}{2} - f}{f_{md}}\right) \quad (7)$$

where L_b = lower boundary of median class, E = class size, f= cumulative frequency before median class, and f_{md} = frequency of median class.

d. Variance: The variance of a class is given by:

$$V = \sqrt{\sigma} \quad (8)$$

where σ = standard deviation

e. Maxima (Max) and Minima (Min): Provides the minimum andmaximum values of a particular extracted data.

f. Kurtosis: The peakedness or kurtosis is given by:

$$k = \sum (y - \bar{y})^k . f(x) \quad (9)$$

where all values of y are summed over, and $f(x)$ is probability distribution function of y.

g. Skewness: Skewness of a distribution is given by:

$$S = C\left(\frac{y - \bar{y}}{\sqrt{\sigma}}\right)^3 \quad (10)$$

Received: August 06, 2025

International Journal of Applied Mathematics

Volume 38 No. 9s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

where y is the mean and $\sqrt{\sigma}$is the standard deviation

h. Entropy: The statistical unpredictability of a sample is quantified by entropy. The process of creating unbiased random bits from biased sources is known as entropy-based feature extraction.

$$H(y) = \sum_{i=1}^{k} p(y_i) \log_2 \left(\frac{1}{p(y_i)}\right) \quad (11)$$

where $y_i = k$ th sample value and $p(y_i)$ is a probability distribution function.

Then, using the suggested adaptive PUMA optimistic algorithm, these extracted datasets are provided as input for the feature selection process, which may be successfully obtained. To locate the ideal and nearly ideal solution, this algorithm has skilfully balanced the stages of exploitation and exploration space.

i. Gaussian Bhattacharyya Distance

The Gaussian Bhattacharyya distance can be utilized to compare feature distributions extracted from network traffic data in the context of IoT device discovery. The Bhattacharyya distance is a measure of the similarity between two probability distributions.

To extract the features with the highest ability to differentiate between each N_i modulation class, we assume that the target modulation pool, $N = \{N_1 \ldots\ldots N_r\}$, consists of N number of modulations. We take the initial set of d-dimensional characteristics to be $f = \{f_1, \ldots . f_d\}$, from which we shall extract the most diverse features. Assume that the collection of the most successfully extracted features, $e = \{f_1, \ldots . f_e\}$, has, $e \in f$, i.e., $e < d$. A Bhattacharyya distance matrix $GBD_{(d\times e)}$ is generated for all d features and $\binom{r}{2}$ modulation pairings, depending on the probability distribution of each feature in$f$. In its most basic form, GBD is provided by the following equation: GBD for the probability distribution of nth original feature.f_e, for modulation pair N_i and N_j.

$$GBD_{(q(f_n)p(f_n))} = \frac{1}{4}(\sigma_2 - \sigma_1)^h \left(\frac{\sigma^2_{q(f_n)} + \sigma^2_{p(f_n)}}{2}\right)^{-1} (\sigma_2 - \sigma_1) + \frac{1}{2} ln \frac{\left|(\gamma^2_{q(f_n)} + \gamma^2_{p(f_n)})/2\right|}{\left|\sigma^2_{q(f_n)}\right|^{1/2} \left|\sigma^2_{q(f_n)}\right|^{1/2}} \quad (12)$$

The Gaussian Bhattacharyya distance between the $q(f_n)$ and $p(f_n)$ classes are represented by $GBD_{(q(f_n)p(f_n))}$. The $q(f_n), p(f_n)$ classes are two separate training classes, and$\sigma_{q(f_n)}$represents the variance of $q(f_n)$ class and $\gamma_{q(f_n)}$represents the mean of the $q(f_n)$ class. The probability distribution of feature $f_n$for modulation N_i, i.e., $q(f_n) = q(f_n/N_i)$, is thus given by$q(f_n)$ where$n = 1,2, \ldots d$, while the probability distribution of f_n for modulation N_j , i.e., $p(f_n) = q(f_n/N_j)$, is given by $p(f_n)$, where$\forall i \neq j$. For every d feature in f, the GBD for all $\binom{r}{2}$combinations of N_i and N_j pairs are calculated. Upon calculating the Bhattacharya distance matrix $GBD_{(d\times e)}$ for every d feature and every e combination, its columns are arranged in descending order, meaning that the values of the feature $f_n$that have the greatest dissimilarity between their probability distributions $\binom{r}{2}$ thecombination is

Volume 38 No. 9s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

positioned at the top of the GBD matrix.When two classes have equal variances and one feature provides the greatest distance for multiple combinations, this is known as repetition, and it would eliminate the first term of the distance. Only the distribution variances affect this term. This term will be 0 in the case of equal variances and increase in the case of unequal variances. If the variances are inversely proportional to the means and the means are identical, however, the second term will be zero.The final collection of characteristics will consist of all features that correspond to the maximum distance and are unique.

Using an adaptive PUMA optimization, the features are optimally collected and used to generate the feature selection process.

3.1.3 Feature Selection using Adaptive Puma Optimization

The feature selection process for IoT device attack detection in web traffic networks uses metaheuristic algorithms. Metaheuristic algorithms are optimization approaches that help with sophisticated issue-solving that is beyond the scope of conventional methods. Metaheuristic algorithms are useful in the context of FS for finding the ideal subset of features to enhance IoT device detection performance.

Adaptive PUMA optimization, a feature-selecting algorithm that analyzes the traits of puma hunting behavior (a particular sort of feline family that resembles a lion), is presented in this method. The primary goal of the APO algorithm is to maximize the feature selection process, which improves the capacity to extract pertinent features and to increase theIoT attackdevice detection accuracy.

- **Inspiration:** Puma is a large cat species found in Americas, known by names like cougar and mountain lion. Their habitat spans from the Andes Mountains to the Yukon. After the jaguar, the puma is the largest cat in the Americas and is incredibly adaptive, and preys on various animals. Puma’s sprint and hunt for prey by ambushing, passing through covered areas, leaping on prey, suffocating them with fangs, and breaking petite prey's neck on the ground using their powerful fangs.
- **Mathematical Model:** The PO method builds a mathematical model by simulating pumas' natural hunting behaviours. Meta-heuristic optimizers use random solutions to optimise, drawing inspiration from natural occurrences. An innovative method for altering the exploration and exploitation stages is introduced by the PO optimizer algorithm. For optimisation, two distinct methods are employed during the periods of exploration and exploitation. Solutions enter exploitation or exploration phases using a purposeful mechanism. Different approaches inspired by pumas are used in each phase for optimization. Mathematical formulations are presented to clarify the optimization operations of PO.
- **Intelligent Puma (a phase-shifting mechanism):** Pumas hunt in both familiar and unfamiliar areas because they are smart and have good memories. The study explores exploration and exploitation phases of their hunting trips. A sophisticated hyper-heuristic algorithm with heuristic selection is presented, drawing inspiration from pumas. The algorithm's phase change mechanism draws inspiration from cougar hunting strategies, which

Received: August 06, 2025

Volume 38 No. 9s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

involve venturing into uncharted territory. Less experienced cougars attack prey in places that show promise; this was covered in the first-generation section.

- **Inexpert phase:** Puma lacks experience early on and roams its territory while hunting in favorable regions. The Puma algorithm begins by combining exploration and exploitation, with only two functions used in the first three iterations. The procedure requires computing.F_1 and F_2 using particular equations.

$$F_{1L} = s1 \cdot \left(\frac{Sq^1_{CostL}}{S_{tm}}\right) \quad (13)$$

$$F_{1M} = s1 \cdot \left(\frac{Sq^1_{CostM}}{S_{tm}}\right) \quad (14)$$

$$F_{2L} = s2 \cdot \left(\frac{Sq^1_{CostL}+Sq^2_{CostL}+Sq^3_{CostL}}{Sq_{tm1}+Sq_{tm2}+Sq_{tm3}}\right) \quad (15)$$

$$F_{2M} = s2 \cdot \left(\frac{Sq^1_{CostM}+Sq^2_{CostM}+Sq^3_{CostM}}{Sq_{tm1}+Sq_{tm2}+Sq_{tm3}}\right) \quad (16)$$

Eqs. (13) –(16) are used to calculate the variable values of Sq_{Cost} that correspond to each step of exploration and exploitation. In addition, $Sq_{tm}$is a constant-value variable, which in our case is 1. To rank the F_1 and F_2functions, the fixed-value parameters $s1$ and $s2$ need to be set before the optimization process.

$$Sq^1_{CostL} = |Cost^I_{best} - Cost^1_L| \quad (17)$$

$$Sq^2_{CostL} = |Cost^2_L - Cost^1_L| \quad (18)$$

$$Sq^3_{CostL} = |Cost^3_L - Cost^2_L| \quad (19)$$

$$Sq^1_{CostM} = \left|Cost^I_{finest} - Cost^1_M\right| \quad (20)$$

$$Sq^2_{CostM} = |Cost^2_M - Cost^1_M| \quad (21)$$

$$Sq^3_{CostM} = |Cost^3_M - Cost^2_M| \quad (22)$$

$Cost^I_{best}$is cost of best solution in initialization phase with six variables in Eqs. (17) to (22). Calculations are made for $Cost^1_L, Cost^2_L, Cost^3_L, Cost^1_M, Cost^2_M, and\ Cost^3_M$ for exploitation and exploration phases. Following third iteration, the functions F_1 and F_2 are assessed, which results in the choice of either the exploration or exploitation phase. Pumas decides which phase to move forward with based on enjoyable encounters. Eqs. (23) and (24) are utilized to ascertain the points allocated for exploration and exploitation, respectively.

$$Rank_L = (s1 \cdot F_{1L}) + (s2 \cdot F_{2L}) \quad (23)$$

$$Rank_M = (s1 \cdot F_{1M}) + (s2 \cdot F_{2M}) \quad (24)$$

Using Eqs. (26) and (27), two calculations yield phases of exploitation and exploration; $Rank_L$ and $Rank_M$. Exploitation stage is entered if $Rank_M \geq Rank_L$; if not, the exploration phase is performed. After three rounds, each step separately produces solutions that are greater than the population as a whole. After the third iteration, the total cost of the

Received: August 06, 2025

Volume 38 No. 9s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

solutions from the two phases is computed to solve the problem. Current solutions are replaced only when the best solutions from all generated solutions equalize the total population.

- **Skilled phase**: Pumas decides to switch phases after three generations and choose one for optimization. The scoring is done using three functions (F_1, F_2, F_3). The escalation component is highlighted in the first function, which gives priority to phases of exploration or exploitation. The first function, derived using Eq. (25), places more importance on exploration phase.

$$F_{1repeat}^{M} = s1 \cdot \left| \frac{Cost_c^M - Cost_d^M}{Repeat_{repeat}^M} \right| \quad (25)$$

$$F_{2repeat}^{L} = s1 \cdot \left| \frac{Cost_c^L - C_d^L}{Repeat_{repeat}^L} \right| \quad (26)$$

F_{repeat}^{M} and F_{repeat}^{L}, where $repeat$ is the iteration number, indicating the first function's amount depending on exploration or exploitation in Eqs. (25) and (26). Costs show the optimal solution both before and after the present option is improved. The unselected iterations from prior selection to current one is represented by the numbers $Repeat_{repeat}^{M}$and $Repeat_{repeat}^{L}$. The first function is prioritized by the user-adjustable parameter $s1$, depending on its value.

$$F_{1repeat}^{M} = s2 \cdot \left| \frac{(Cost_{c,1}^M - Cost_{d,1}^M) + (Cost_{c,2}^M - Cost_{d,2}^M) + (Cost_{c,3}^M - Cost_{d,3}^M)}{Repeat_{repeat,1}^M + Repeat_{repeat,2}^M + Repeat_{repeat,3}^M} \right| \quad (27)$$

$$F_{2repeat}^{L} = s2 \cdot \left| \frac{(Cost_{c,1}^L - Cost_{d,1}^L) + (Cost_{c,2}^L - Cost_{d,2}^L) + (Cost_{c,3}^L - Cost_{d,3}^L)}{Repeat_{repeat,1}^L + Repeat_{repeat,2}^L + Repeat_{repeat,3}^L} \right| \quad (28)$$

The emphasis on resonance and performance in the second function helps with the exploitation stage. It is calculated using Eqs. (27) and (28). $F_{1repeat}^{M}$ and $F_{2repeat}^{L}$represent second function in exploitation or exploration, with $repeat$ as the iteration number. Costs are related to the best solutions before and after improvements in selections. $Repeat_{repeat}^{M}$ and $Repeat_{repeat}^{L}$are the unselected iterations in each phase. $s2$ is a parameter determining the adequacy of the second function

$$F_{3itr}^{M} = \begin{cases} if\ selected, F_{3repeat}^{M} = 0 \\ otherwise, F_{3repeat}^{M} + s3 \end{cases} \quad (29)$$

$$F_{3itr}^{L} = \begin{cases} if\ selected, F_{3repeat}^{L} = 0 \\ otherwise, F_{3repeat}^{L} + s3 \end{cases} \quad (30)$$

The selection mechanism's third function emphasizes diversity and provides an opportunity for phases that were not chosen to be chosen. In Eqs. (29) and (30), it is defined. Depending on the iteration number, the function corresponds to the exploitation or exploration stages. If no stage is selected, the function's value rises by parameter $s3$. The

Received: August 06, 2025

Volume 38 No. 9s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

value of $s3$ must be between 0 and 1. The probability of low-score periods increases with a greater c value.$Comp$to the calculation

$$F_{repeat}^{M} = \left(\alpha_{repeat}^{M} \cdot \left(F_{1repeat}M\right)\right) + \left(\alpha_{repeat}^{M} \cdot \left(F_{2repeat}M\right)\right) + \left(\delta_{repeat}^{M} \cdot \left(Calc \cdot F_{3repeat}M\right)\right) \quad (31)$$

$$F_{repeat}^{L} = \left(\alpha_{repeat}^{L} \cdot \left(F_{1repeat}L\right)\right) + \left(\alpha_{repeat}^{L} \cdot \left(F_{2repeat}L\right)\right) + \left(\delta_{repeat}^{L} \cdot \left(Calc \cdot F_{3repeat}L\right)\right) \quad (32)$$

$$Comp = \{\{|Cost_c - Cost_d|\}^{M}, \{|Cost_c - Cost_d|\}^{L}\}, 0 \notin Comp \quad (33)$$

$$\alpha_{repeat}^{L,M} = \begin{cases} if\ F^M > F^L, \alpha^M = 0.99, \alpha^L = \lceil \alpha^L - 0.01, 0.01 \rceil \\ otherwise, \alpha^L = 0.99, \alpha^M = \lceil \alpha^M - 0.01, 0.01 \rceil \end{cases} \quad (34)$$

$$\delta_{repeat}^{M} = 1 - \alpha_{repeat}^{M} \quad (35)$$

$$\delta_{repeat}^{L} = 1 - \alpha_{repeat}^{L} \quad (36)$$

Eqs. (35) and (36) are used to compute the cost of phase transition. The total expenses of every stage are calculated. The search operation involves changes to the parameters $\alpha\ and\ \delta$.

$$\text{If} random_1 > 0.5, Gene_i = random_{dim} * (ub - lb) + lb, Otherwise \quad (37)$$

$$Gene_i = A_{1,i} + i \cdot \left(A_{1,i} - A_{2,i}\right) + i \cdot \left(\left(\left(A_{1,i} - A_{2,i}\right) - \left(A_{3,i} - A_{4,i}\right)\right) + \left(\left(A_{3,i} - A_{4,i}\right) - \left(A_{5,i} - A_{6,i}\right)\right)\right) \quad (38)$$

$$I = 2 \cdot random_2 - 1 \quad (39)$$

Where $Gene_i$ is the generated solution, dim is the dimension, $ub\ and\ lb$ are the upper bound and the lower bound. If cost of exploration exceeds the cost of exploitation, a parameter of α exploitation is penalised.Process is carried out in reverse if the cost of exploitation is larger. The $Comp$ shows the variations in costs between exploration and exploitation upgrades.

$$A_d = \begin{cases} Gene_i, if\ j = j_{random}\ or\ random_3 \leq Pre_{opt} \\ A_{1,i}, otherwise \end{cases} \quad (40)$$

$$nc = 1 - Pre_{opt} \quad (41)$$

$$H = \frac{nc}{n_p} \quad (42)$$

$$\text{If } CostA_d < CostA_{i,} Pre_{opt} = Pre_{opt} + H \quad (43)$$

$Pre_{opt}$is the previous optimizer. Pumas search for food in familiar or new areas, inspiring researchers. Equations help improve solutions for the sorted Puma population. Solutions are updated to maintain diversity and avoid local optimums. Search agents are

Received: August 06, 2025

Volume 38 No. 9s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

sorted by cost initially, prioritizing high-quality solutions. As the parameter increases, solutions with higher cost values explore worse options. Good quality solutions undergo a few changes to avoid local optimality traps. If new production Pumas are not better, a certain equation will not be executed.

$$A_{1,i} = A_d, if\ A_{i,d} < A_{1,i} \tag{44}$$

If newly developed solutions are more affordable, they take the place of the existing ones.

- **Exploitation**

In the exploitation Stage, PO algorithm improves solutions using pumas' hunting behaviors. Pumas either ambush prey or sprint after them. Pumas uses Eq. (48) for running and ambush strategies in the first mode. Fast-running strategy is used if is over 0.5. The ambush strategy includes short and long jumps towards prey.

$$A_d = \begin{cases} if\ random_4 > 0.5, A_d = \frac{\left(\frac{mean(A_i)}{n_p}\right)\cdot A_1^{random} - (-1)^{\beta} \times A_i}{1+(\alpha \cdot random_5)}, otherwise \\ if\ random_6 > K, A_d = PO_{male} + (2 \cdot random_7) \cdot L(randomn_1) \cdot A_2^{random} - A_i, otherwise \\ A_d = (2 \times random_8) \times \frac{(F_1 \cdot random \cdot A(i) + F_2 \cdot (1-random) \cdot PO_{male})}{(2 \cdot random_9 - 1 + randomn_2)} - PO_{male} \end{cases} \tag{45}$$

Where, PO_{male} denotes the male puma. Before performing optimization, the static parameters α,K, and βneed to be changed. The population's best solution is pumamale, and in the process, a variety of random numbers are produced. L, $randomn_1$, $randomn_2$, and P_2^{random}also have functions in the optimization process. The Puma Optimizer Algorithm (POA) is coupled with a Poisson distribution function for adaptation to increase dynamic parameter control and optimization efficiency, hence improving performance even further.If a discrete random variable PD_F has a probability mass function that is described by and has parameter, assume $0 < \lambda < 2$, it is said to have a Poisson distribution in Eq. (46).

$$PD_F(A_i, \lambda) = \frac{\lambda^{A_i} e^{-\lambda}}{A_i!} \tag{46}$$

$$A_d = \begin{cases} if\ random_4 > 0.5, A_d = \frac{\left(\frac{mean(A_{soln})}{n_p}\right)\cdot A_1^{random} - (-1)^{\beta} \times PD_F(A,\lambda)}{1+(\alpha \cdot random_5)}, otherwise \\ if\ random_6 > K, A_d = PO_{male} + (2 \cdot random_7) \cdot L(randomn_1) \cdot A_2^{random} - PD_F(A,\lambda), otherwise \\ A_d = (2 \times random_8) \times \frac{(F_1 \cdot random \cdot PD_F(A,\lambda) + F_2 \cdot (1-random) \cdot PO_{male})}{(2 \cdot random_9 - 1 + randomn_2)} - PO_{male} \end{cases} \tag{47}$$

By combining the Puma Optimizer Algorithm with a Poisson distribution, one can improve overall performance by gaining more dynamic control over parameters and optimising more efficiently through POA-based self-adaptive optimisation.

Received: August 06, 2025

International Journal of Applied Mathematics

Volume 38 No. 9s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

3.2 Classification using AGGCR_Net Method

The proposed AGGCR_Net approach is used to classify attacks after the features have been chosen. The suggested technique aids in improving training the network for IoT device identification, resulting in faster convergence and improved attack classification performance. The convolution and recurrent networks used in this method highlight the pace of increased spatial information, which improves the prediction accuracy of IoT devices.

3.2.1 An Adaptive Gradient Method for Gated Convolution Recurrent Network.

An adaptive gradient-based optimization method for the Gated Convolutional Recurrent Network (GCRN) further dynamizes the feature learning to ensure better feature extraction for modeling temporal dependency and, hence, optimal performance of an IoT device identification system. A GCRN is a hybrid model that combines the Recurrent Neural Networks (RNN) and Convolutional Neural Networks (CNN) with gating mechanisms. The architecture inherits all the benefits from the multiple advanced deep models: on one hand, CNNs are robust in feature extraction, and instead, RNNs specialize in sequence modelling. In this architecture, multi-variants of CNN are interconnected with GRU and LSTM. In the multi-variant CNN, Alexnet with multiple Global Average Pooling is come first (c1), Reduced Alexnet with Gated Recurrent Unit (GRU) is come second (c2). The c1 features are concatenated with the c2 features before GRU. And the c3 has the Alexnet with the multiple Global Average Pooling is with the feature of c4. In c4 has the inception based Alexnet with LSTM. The LSTM connects the c4 features and c3 features. And the fully connected layers connect the GRU and LSTM for classification.Therefore, the model becomes considerably efficient in tasks such as time series forecasting, speech recognition, video analysis, and IoT device identification. When it comes to an IoT Device Identification System, the GCRN starts by getting input data from IoT devices, such as patterns of network traffic or series of sensor readings. The input data is processed by the convolutional layers of this CNN to extract local and hierarchical features that capture complex spatial patterns via numerous layers and create a feature map that preserves the input's spatial qualities.

Subsequently these convolutional layers, use gating methods such as Gated Recurrent Units (GRU) to regulate the flow of features and improve the representational capacities of the network.An additional kind of recurrent neural network is the gated recurrent unit. At every time step, to regulate the information flow into the concealed state, GRU employs gating methods. The information entering and leaving the network is managed by the gating mechanisms. An update gate and a reset gate are the two gating mechanisms of a GRU. The reset gate decides how much of the previous hidden state to forget, whereas the update gate specifies how much of the new input the hidden state should use. Updated hidden state will determine the GRU's output.Afterwards, the feature map is passed to the recurrent layers of the RNN, which are designed for extracting temporal dependencies from sequential data.

$$r_t = \sigma(W_r * h_{t-1}, x_t) \tag{48}$$

$$z_t = \sigma(W_z * h_{t-1}, x_t) \tag{49}$$

Received: August 06, 2025

International Journal of Applied Mathematics

Volume 38 No. 9s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

$$\tilde{h}_t = tanh\left(W_h * (r_t * h_{t-1}, x_t)\right) \quad (50)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t \quad (51)$$

The reset gate is formulated in Eq. (48), while the update gate is defined in Eq. (49). The candidate hidden state in Eq. (50), hidden state is expressed in Eq. (51), W_r,W_z, and W_h are the weight matrices, previous hidden state h_{t-1}, x_t, input at time step $t$and the current hidden state h_t are all crucial components of the model.

In applications, one normally uses an LSTM and GRU since both can handle long-term dependencies quite well. These recurrent layers make use of specific gating mechanisms, including input, output, and forget gates, to manage the flow of information across time steps so relevant information is retained while irrelevant information is forever discarded. Finally, the output layer projects the final system output, which might be the identification or classification of the IoT device depending on the application. The AGCCR_Net architecture uses fully connected layers, which are in charge of capturing complex relationships between features after convolutional extraction. Each of these two layers has 4096 neurons to facilitate high-dimensional feature mapping. To improve its capacity to learn from the convolutional layers, it might perform complex transformations on the extracted information. The last layer contains 1000 neurons, probably a classification task, for which AGCCR_Net was originally proposed. All these configurations put together mean that the spatial features are now manipulated toward class probabilities, finishing in a 1000-class output suitable for large-scale image classification tasks. This dense connectivity in fully connected layers supports comprehensive feature integration very important in sophisticated tasks of visual recognition to achieve high accuracy. Ensuring that the spatial features are effectively extracted by the CNNs and that the temporal dependencies are properly captured by the RNNs with the gating mechanism, the GCRN will have a very powerful and versatile network architecture for IoT device identification applications. The update gate is expressed in Eq. (52)

$$z_t^n = \sigma\left(maxout(w_z^n * x_t^n + U_z^n * h_{t-1}^n)\right) \quad (52)$$

A convolution operation is shown by the operator $*$. Likewise, the reset gate

$$r_t^n = \sigma\left(maxout(w_r^n * x_t^n + U_r^n * h_{t-1}^n)\right) \quad (53)$$

Then the update activation gate is

$$\tilde{h}_t^n = tanh\left(maxout\left(w_h^n * x_t^n + U_h^n * (r_t^n \odot h_{t-1}^n)\right)\right) \quad (54)$$

Sigmoid and inflated tangent activation functions of gates are represented by the functions σ $(\cdot)$ and $tanh(\cdot)$ in the above equations, respectively, which indicates an element-wise multiplication. The input x_t^ndenotes a sequence of spectral features at time step t.Filters of single-layer CNNs are represented through model parameters.w_z^n, w_r^n, w_h^n and U_z^n, U_r^n, U_h^n, which are shared through all GCRN time steps.

Received: August 06, 2025

International Journal of Applied Mathematics

Volume 38 No. 9s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

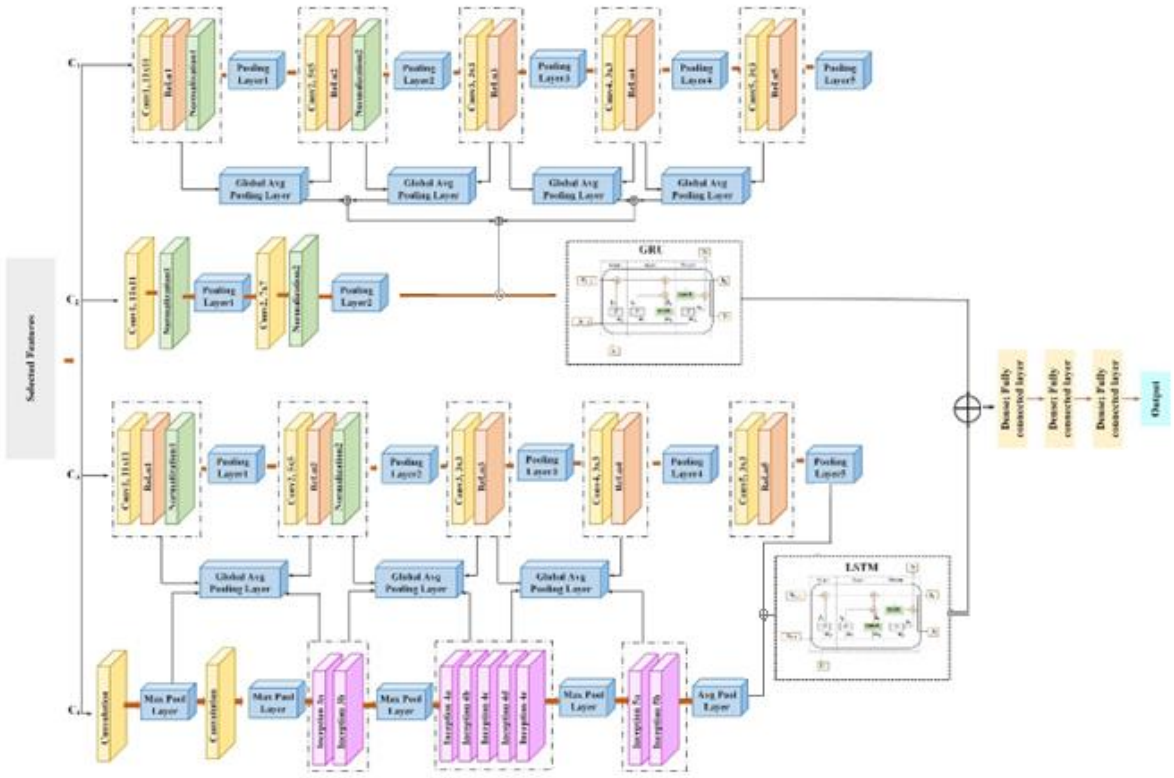


Figure 2: AGGCR_Net

Figure 2 shows AGGCR_Net. Gradient-based optimization methods are one of the optimizers used where neural networks update their weight parameters toward the minimization of a certain loss function. The gradients of the loss function concerning each weight parameter are computed, and then weights are updated accordingly. Normally done by backpropagation and gradient descent. The input gradient and the gradient weight-based parameter are expressed in Eq. (55) and (56) respectively.

$$\frac{\partial L}{\partial x_{c,i}} = \sum_{k=n}^{n} \frac{\partial L}{\partial y_{i+k}} w_{c,k} \tag{55}$$

$$\frac{\partial L}{\partial w_{c,k}} = \sum_{j=0}^{m-1} \frac{\partial L}{\partial y_j} x_{c,j-k} \tag{56}$$

The weight of the network is fine-tuned by using a distribution-based gradient parameter's updation procedure. Here, W denotes the weight matrices. The W is updated by using a gradient parameter-based function. For the weight adaptation, the gradient method is adapted by the Binomial distribution function. The binomial distribution function is in Eq. (57)

$$P(X = k) = \binom{n}{k} p^k (1-p)^{n-k} \tag{57}$$

The updated Binomial distribution function is in Eq. (58)

$$W_{BDF} = \binom{n}{BDF} p^{BDF} (1-p)^{n-BDF} \tag{58}$$

Received: August 06, 2025

International Journal of Applied Mathematics

Volume 38 No. 9s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

The Adaptive Gradient Method is expressed in Eq. (59)

$$\frac{\partial L}{\partial W_{BDF}} = \sum_{j=0}^{m-1} \frac{\partial L}{\partial y_j} W_{BDF} \quad (59)$$

Gradients are adjusted using the Binomial distribution function in an adaptive gradient approach for GCRN. By varying the learning rates as 0.01,0.02, 0.0001, and 0.05 by considering the statistical characteristics of gradients, this method improves training and accelerates the convergence of the network.

4. EXPERIMENTAL RESULT

This section discusses the experiment results using evaluation metrics such as accuracy, precision, F-measure, Sensitivity, Specificity, MCC, NPV, FPR, FNR, and G-mean. The proposed AGGCR_Net is an efficient method for detecting attacks on IoT devices and is executed by using Python. The evaluating results are discussed with the other existing methods including Random Forest, CNN_LSTM, CNN_GRU, SVM, and the proposed PUMA-based AGGCR_Net.

4.1 Performance Metrics

The metrics used to evaluate the classification results have been the following: Accuracy, Precision, F-measure, Recall, Sensitivity, Specificity, MCC, NPV, FPR, FNR, and G-Mean. The considered datasets are somehow unbalanced concerning certain attack classes as has been described in this section. The overall accuracy would have not been a significant parameter. The metrics of this method are as follows

- ***Accuracy:***Accuracy is obtained as a ratio and can be viewed as a combined measure since it's the total number of samples used in assessing the classification model against the sum of true positives and true negatives.

$$Accuracy = \frac{TP+TN}{TP+FN+TN+FP} \quad (60)$$

- ***Precision:*** It is the accuracy proportion of attack samples correctly classified among all samples classified as attacks.

$$Prec = \frac{TP}{TP+FP} \times 100 \quad (61)$$

- ***F-Measure:***Another measure of model accuracy is the F-measure, which connects recall and precision. It is defined as

$$F - score = \frac{TP}{TP+\frac{1}{2}(FP+FN)} \quad (62)$$

- ***Sensitivity:***Sensitivity is defined as the ratio of attack samples that have been correctly classified as malicious to the total attack samples. The exact mathematical formula for calculating sensitivity can be recovered by Eq. 63.

$$Sen = \frac{TP}{TP+FN} \quad (63)$$

Received: August 06, 2025

International Journal of Applied Mathematics

Volume 38 No. 9s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

- ***Specificity:*** It is the proportion of benign samples out of total benign cases that are correctly classified as benign. The specificity is mathematically expressed by Eq. 64.

$$Spec = \frac{TN}{FP+TN} \quad (64)$$

- ***FPR:***It may be defined as the ratio of the total amount of negative data in comparison to that which a share of positive data might have been misclassified as.

$$FPR = \frac{FP}{TN+FP} \quad (65)$$

- ***FNR:***It's the proportion of misclassified cases where a negative label is mistakenly assigned to the total number of cases of the positive label.

$$FNR = \frac{FN}{FN+TP} \quad (66)$$

- ***MCC:*** It is a measure for binary classification quality that comes in very handy in the case of imbalanced data sets. This takes into account true positives, true negatives, false positives, and false negatives in order to come out with a well-balanced metric applicable even in the case when classes are of very different sizes.

$$MCC = \frac{(TP\times TN)-(FP\times FN)}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}} \quad (67)$$

- ***Negative Predictive Value (NPV):***NPV is the percentage of correctly classified cases in the truly negative population. A higher NPV would mean that the model correctly classified true negatives and hence did not raise false negatives. NPV is expressed using the formula:

$$NPV = \frac{TN}{TN+FN} \quad (68)$$

- ***G-Mean:***In the case that performance metrics for both TP and TN are relevant, Eq. (69) yields a measure of the classifier effectiveness for both classes, called the G-Mean.

$$G-mean = \sqrt{TP(1-FP)} \quad (69)$$

A comparative study is conducted between the proposed AGGCR_Net model and existing techniques such as CNN_LSTM, CNN_GRU, SVM, and RF using the IoT device identification classification dataset. Table 1 provides the comparative table for the proposed AGGCR_Net and the existing method by the dataset 1.

Table 1: Comparative table for the proposed AGGCR_Net and the existing method by the Dataset 1

	CNN_LSTM	**CNN_GRU**	**SVM**	**RF**	**AGGCR_Net**
Accuracy	0.8858	0.9234	0.9223	0.9436	0.9704
Precision	0.8534	0.9126	0.9111	0.9167	0.9432
Sensitivity	0.7917	0.9307	0.9213	0.9002	0.9887

Received: August 06, 2025

Volume 38 No. 9s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Specificity	0.86543	0.9167	0.9231	0.9043	0.9419
F-Measure	0.8837	0.9216	0.9162	0.9474	0.9708
MCC	0.795	0.847	0.8438	0.8891	0.9425
NPV	0.7984	0.934	0.932	0.917	0.98786
FPR	0.0976	0.0833	0.0769	0.0957	0.0281
FNR	0.2083	0.0693	0.0787	0.0198	0.0123
G-Mean	0.8345	0.9123	0.9086	0.92345	0.96473

The Table 1 compares the performance of AGGCR_Net with CNN_LSTM, CNN_GRU, SVM, and RF using Dataset 1. AGGCR_Net excels across all metrics: it achieves the highest accuracy (97.04%), precision (94.32%), and sensitivity (98.87%), indicating its superior ability to correctly identify both positive and negative cases. Its specificity is 94.19%, slightly behind CNN_GRU and RF, but still impressive. The F-Measure and MCC are also best for AGGCR_Net, at 97.08% and 0.9425, respectively, highlighting its balanced performance. AGGCR_Net shows the lowest False Positive Rate (2.81%) and False Negative Rate (1.23%), meaning fewer errors in predictions. Its Geometric Mean is highest at 96.47%, reflecting an excellent trade-off between sensitivity and specificity.

Table 2: Comparative table for the proposed AGGCR_Net and the existing method using Dataset 2

	CNN_LSTM	**CNN_GRU**	**SVM**	**RF**	**AGGCR_Net**
Accuracy	0.8769	0.905	0.891	0.9307	0.9756
Precision	0.819	0.9181	0.9135	0.9286	0.9756
Sensitivity	0.9451	0.8218	0.8716	0.9286	0.9756
Specificity	0.8173	0.9199	0.9118	0.9327	0.9756
F-Measure	0.8776	0.8973	0.892	0.9286	0.9756
MCC	0.7629	0.8222	0.783	0.8613	0.9712
NPV	0.9444	0.8448	0.8692	0.9327	0.9756
FPR	0.1827	0.0101	0.0882	0.0673	0.0244
FNR	0.0549	0.1782	0.1284	0.0714	0.0244
G-Mean	0.8543	0.8975	0.86534	0.91233	0.9354

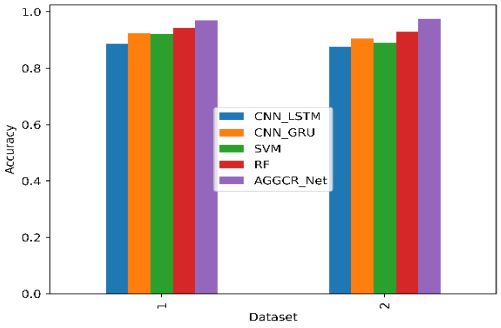
Received: August 06, 2025

International Journal of Applied Mathematics

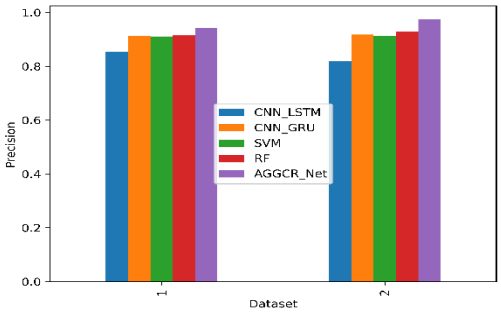
Volume 38 No. 9s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

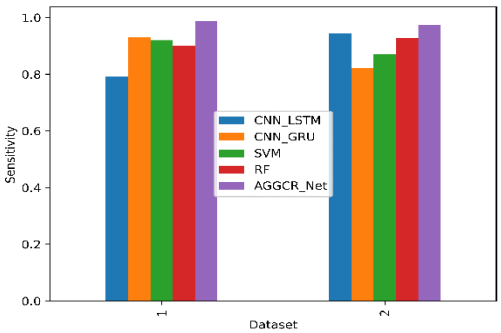
The Table 2 compares AGGCR_Net with CNN_LSTM, CNN_GRU, SVM, and RF using Dataset 2. AGGCR_Net demonstrates superior performance across all metrics, with the highest accuracy (97.56%), precision (97.56%), sensitivity (97.56%), specificity (97.56%), and F-Measure (97.56%). It excels in identifying both positive and negative cases effectively. The MCC is highest for AGGCR_Net at 0.9712, indicating robust performance. Its False Positive Rate (FPR) is the lowest at 2.44%, while its FNR is also the lowest at 2.44%, showing minimal prediction errors. The G-Mean is highest at 93.54%, reflecting a strong balance between sensitivity and specificity.



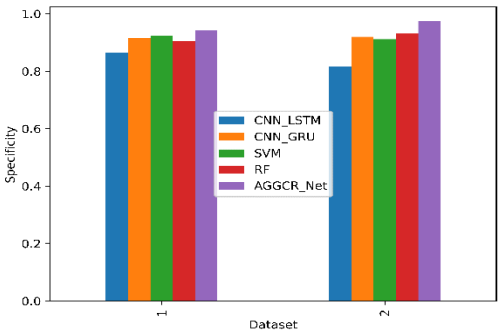
Accuracy



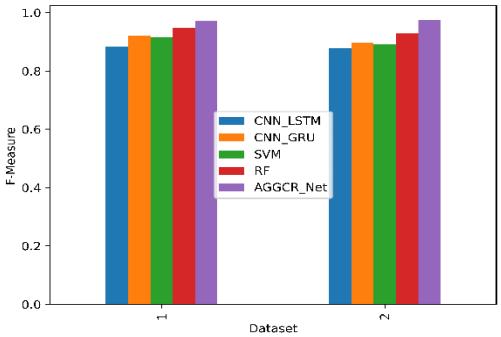
Precision



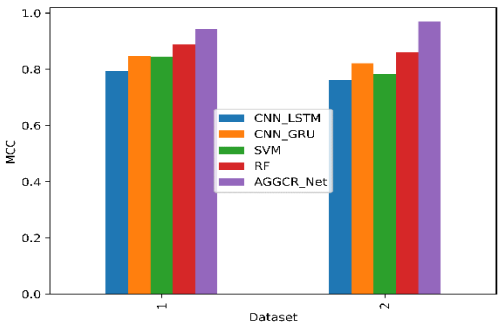
Sensitivity



Specificity



F-Measure



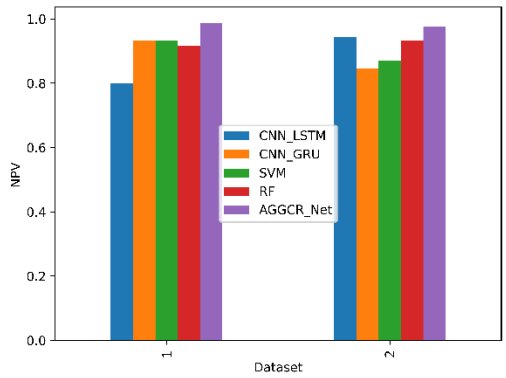
MCC

Received: August 06, 2025

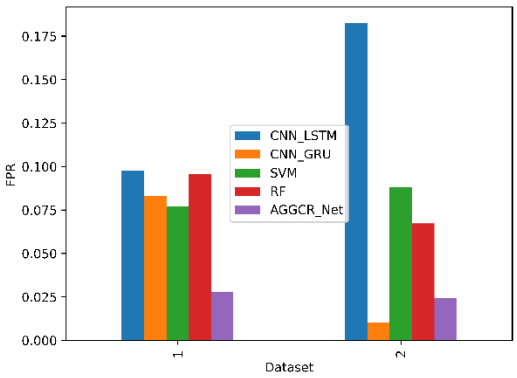
International Journal of Applied Mathematics

Volume 38 No. 9s, 2025

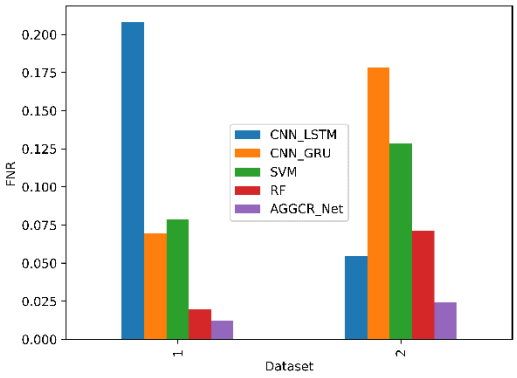
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)



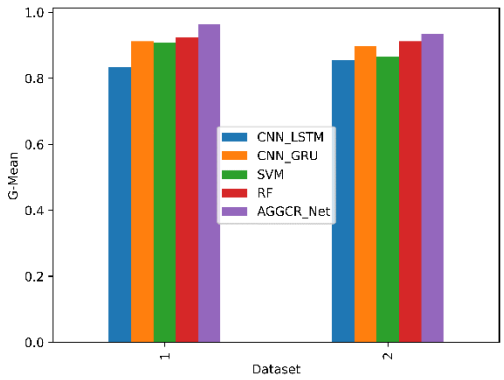
NPV



FPR



FNR



G-Mean

Figure 3 (a)-(j): Graphical representation of the performance metrics compared with existing works and proposed by the two datasets.

Figure 3 (a)-(j) demonstrates graphical representation of performance metrics of the existing works with the proposed method. Performance metrics such asSpecificity, Precision, Sensitivity, Accuracy, F-measure, MCC, FNR, NPV, FPR, and G-mean are compared with different methods like CNN_LSTM, CNN_GRU, SVM, RF, and AGGCR_Net.

Table 3: Comparison metrices for the Aalto Dataset

	Precision	**Recall**	**F1 Score**	**Accuracy**
HueSwitch	0.97	0.97	0.89	0.98
HueBridge	0.89	0.99	0.97	0.99
D-LinkHomeHub	1	0.98	0.94	0.97

Received: August 06, 2025

International Journal of Applied Mathematics

Volume 38 No. 9s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

D-LinkSwitch	0.96	1	0.98	0.98
D-LinkSensor	0.95	1	0.87	0.96
D-LinkWaterSensor	0.98	0.99	0.97	0.98
WeMoLink	0.9	0.96	0.95	0.99
D-LinkSiren	0.91	0.96	0.92	0.94
D-LinkCam	0.98	0.97	0.96	0.98
WeMoInsightSwitch	0.96	0.98	0.97	0.99
WeMoSwitch	0.89	0.92	0.98	0.97
Lightify	0.91	0.95	0.99	0.98
D-LinkDoorSensor	0.92	0.94	0.98	0.99
EdimaxPlug1101W	0.93	1	0.97	0.97
D-LinkDayCam	0.96	0.99	0.98	0.98
EdimaxPlug2101W	1	0.94	0.93	0.96
EdimaxCam	0.94	1	0.9	0.98
Withings	0.95	0.96	0.99	0.99
EdnetGateway	0.96	0.95	0.96	0.98
TP-LinkPlugHS100	0.93	0.94	0.97	0.97
TP-LinkPlugHS110	0.93	0.97	0.99	0.98
HomeMaticPlug	0.94	1	0.98	0.95
MAXGateway	0.89	0.99	0.97	0.96
Aria	0.91	0.94	0.97	0.98
EdnetCam	0.97	0.98	0.98	0.99
SmarterCoffee	0.95	0.9	0.98	0.94
iKettle2	0.92	0.96	0.98	0.93

Table 3 shows performance metrics for several devices from the Aalto Dataset. HueBridge and WeMoInsightSwitch had the highest accuracy of 0.99, with HueBridge leading in recall (0.99) and WeMoInsightSwitch in F1 Score (0.97). D-LinkHomeHub performs well in precision (1.00) and F1 Score (0.94), although D-LinkSiren has a good recall (0.96) but a lesser accuracy (0.94). Despite their high recall rates, EdimaxPlug2101W and HomeMaticPlug have lesser accuracies (0.96 and 0.95, respectively).

Received: August 06, 2025

International Journal of Applied Mathematics

Volume 38 No. 9s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Table 4: Comparison metrices for the UNSW IoTDevID Dataset

	Precision	**Recall**	**F1 Score**	**Accuracy**
Non-IoT	0.97	1	0.98	0.97
Nest Dropcam	0.99	0.98	1	0.98
Insteon Camera	0.98	0.97	0.92	0.98
Samsung SmartCam	0.97	1	0.95	0.98
Phillip Hue Lightbulb	0.96	0.98	0.98	0.98
Google Chromecast	0.97	1	0.98	0.98
Netatmo weather station	0.96	0.99	0.98	0.97
Dropcam	0.98	0.96	0.97	0.98
Belkin Wemo switch	1	0.98	0.98	0.98
Light Bulbs LiFX Smart Bulb	0.98	0.96	0.99	0.98
Ring Door Bell	0.97	0.9	0.96	0.96
Awair air quality monitor	1	0.92	1	0.99
PIX-STAR Photo-frame	0.98	1	0.99	0.99
HP Printer	0.97	0.98	0.98	0.97
Amazon Echo	0.96	0.99	1	0.98
TPLink Router Bridge LAN	0.98	0.97	0.99	0.99
August Doorbell Cam	0.97	0.98	0.97	0.97
Netatmo Welcome	0.98	0.93	0.98	0.99
Belkin Camera	0.99	0.94	0.97	0.95
TP-Link Day Night Cloud camera	0.98	1	0.99	0.99
Canary Camera	0.94	0.96	0.98	0.98
TP-Link Smart plug	1	0.99	0.97	0.96
Triby Speaker	0.98	0.98	0.95	0.99
Withings Aura smart sleep sensor	0.97	1	0.99	0.99
Belkin wemo motion sensor	0.98	0.96	0.97	0.97
Smart Things	0.95	0.98	1	0.98
Withings Smart Baby Monitor	0.98	0.97	0.97	0.99

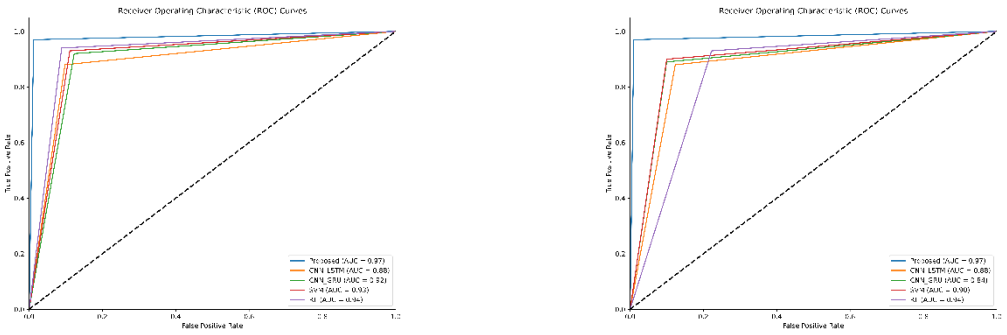
Received: August 06, 2025

Volume 38 No. 9s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

iHome	0.93	0.95	0.97	0.98
Withings Smart scale	0.98	0.96	0.96	0.99
NEST Protect smoke alarm	0.98	0.99	0.97	0.97
unknown maybe cam	0.98	0.95	1	0.98
Hello Barbie	1	0.97	0.97	0.99
Blipcare Blood Pressure meter	0.98	0.96	0.98	0.99

Table 4 displays performance metrics for several devices from the UNSW IoTDevID Dataset. Awair air quality monitor and PIX-STAR Photo-frame have the best accuracy (0.99) and good F1 scores (1.00). The Belkin Wemo switch and TP-Link Day Night Cloud camera both achieve excellent accuracy (0.98) and strong F1 Scores (0.98 and 0.99, respectively). Belkin Camera has good precision (0.99) but low accuracy (0.95). Overall, the dataset performs well with high precision and recall for most devices, however others, such as Ring Door Bell, have significantly lower accuracy (0.96).



Dataset 1 Dataset 2

Figure 4:ROC Curves for the proposed method

The ROC curve of the proposed AGGCR_Net model for attacks detected in IoT-linked devices is shown in Figure 4, which is drawn between false positive and true positive values by using Datasets 1 and 2 respectively.

5. CONCLUSION

The proposed AGGCR_Net model shows notable improvements in the detection of attack devices connected to the IoT in intelligent settings. The steps being performed are fed into the dataset. The processes that comprise the framework are pre-processing feature extraction and classification. Strong classification performance is achieved by the system by utilizing an adaptive gradient-gated Gated Convolution Recurrent Network (AGGCR_Net),

Received: August 06, 2025

Volume 38 No. 9s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

enhanced feature extraction with Gaussian Bhattacharyya distance with entropy, and new adaptive puma optimization (APO) for feature selection. Densely Connected GRU integrated to enhance the classification performance while compared with other approaches such as CNN_LSTM, CNN_GRU, SVM, and RF, it achieved superior convergence in the classification of IoT device identification when used as performance measures. Dataset 1 has an accuracy of 0.9704%, whereas Dataset 2 has an accuracy of 0.9756%. The method, which is based on the Python platform, solves serious security issues brought about by weak IoT devices, to more effective and safer IoT deployments in smart environments.

DECLARATIONS:

DATA AVAILABILITY STATEMENT

All the data is collected from the simulation reports of the software and tools used by the authors. Authors are working on implementing the same using real world data with appropriate permissions.

FUNDING

No fund received for this project

CONFLICTS OF INTEREST

The authors declare that they have no conflict of interest.

ETHICAL APPROVAL AND HUMAN PARTICIPATION

No ethics approval is required.

REFERENCE:

[1] Sahu, Amiya Kumar, Suraj Sharma, Mohammad Tanveer, and Rohit Raja. "Internet of Things attack detection using hybrid Deep Learning Model." Computer Communications 176 (2021): 146-154.

[2] Samy, Ahmed, Haining Yu, and Hongli Zhang. "Fog-based attack detection framework for internet of things using deep learning." Ieee Access 8 (2020): 74571-74585.

[3] Vu, Ly, Quang Uy Nguyen, Diep N. Nguyen, Dinh Thai Hoang, and Eryk Dutkiewicz. "Deep transfer learning for IoT attack detection." IEEE Access 8 (2020): 107335-107344.

[4] Alkahtani, Hasan, and Theyazn HH Aldhyani. "Botnet Attack Detection by Using CNN-LSTM Model for Internet of Things Applications." *Security and Communication Networks* 2021, no. 1 (2021): 3806459.

[5] Parra, Gonzalo De La Torre, Paul Rad, Kim-Kwang Raymond Choo, and Nicole Beebe. "Detecting Internet of Things attacks using distributed deep learning." Journal of Network and Computer Applications 163 (2020): 102662.

[6] Popoola, Segun I., Bamidele Adebisi, Mohammad Hammoudeh, Guan Gui, and Haris Gacanin. "Hybrid deep learning for botnet attack detection in the internet-of-things networks." IEEE Internet of Things Journal 8, no. 6 (2020): 4944-4956.

Received: August 06, 2025

Volume 38 No. 9s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

[7] Luo, Chaochao, Zhiyuan Tan, Geyong Min, Jie Gan, Wei Shi, and Zhihong Tian. "A novel web attack detection system for internet of things via ensemble classification." *IEEE Transactions on Industrial Informatics* 17, no. 8 (2020): 5810-5818.

[8] Al-kahtani, Mohammad S., Zahid Mehmood, Tariq Sadad, Islam Zada, Gauhar Ali, and Mohammed ElAffendi. "Intrusion detection in the Internet of Things using fusion of GRU-LSTM deep learning model." Intelligent Automation & Soft Computing 37, no. 2 (2023).

[9] Shahin, Mohammad, F. Frank Chen, Hamed Bouzary, Ali Hosseinzadeh, and Rasoul Rashidifar. "A novel fully convolutional neural network approach for detection and classification of attacks on industrial IoT devices in smart manufacturing systems." The International Journal of Advanced Manufacturing Technology 123, no. 5 (2022): 2017-2029.

[10] Pecori, Riccardo, Amin Tayebi, Armando Vannucci, and Luca Veltri. "IoT Attack detection with deep learning analysis." In *2020 International Joint Conference on Neural Networks (IJCNN)*, pp. 1-8. IEEE, 2020.

[11] M. Ali Al-Garadi, A. Mohamed, A. Al-Ali, X. Du, and M. Guizani, ‘‘A survey of machine and deep learning methods for Internet of Things (IoT) security,’’ CoRR, vol. abs/1807.11023, 2018, arXiv:1807.11023. [Online]. Available: http://arxiv.org/abs/1807.11023.

[12] R. J. Tom, S. Sankaranarayanan, and J. J. Rodrigues, “Smart energy management and demand reduction by consumers and utilities in an iotfog-based power distribution system,” IEEE Internet of Things Journal, vol. 6, no. 5, pp. 7386–7394, 2019.

[13] L. Chettri and R. Bera, “A comprehensive survey on internet of things (iot) towards 5g wireless systems,” IEEE Internet of Things Journal, vol. 7, no. 1, pp. 16–32, 2020.

[14] S. Chavhan, D. Gupta, B. Chandana, A. Khanna, and J. J. Rodrigues, “Iot-based context-aware intelligent public transport system in a metropolitan area,” IEEE Internet of Things Journal, 2020.

[15] A. Khraisat, I. Gondal, P. Vamplew, J. Kamruzzaman, and A. Alazab, “A novel ensemble of hybrid intrusion detection system for detecting internet of things attacks,” Electronics, vol. 8, no. 11, p. 1210, 2019.

[16] Abbas, Sidra, Imen Bouazzi, Stephen Ojo, Abdullah Al Hejaili, Gabriel Avelino Sampedro, Ahmad Almadhor, and Michal Gregus. "Evaluating deep learning variants for cyber-attacks detection and multi-class classification in IoT networks." PeerJ Computer Science 10 (2024): e1793.

[17] F. Y. Yavuz, D. Ünal, and E. Gül, ‘‘Deep learning for detection of routing attacks in the Internet of Things,’’ Int. J. Comput. Intell. Syst., vol. 12, no. 1, pp. 39–58, Nov. 2018.

[18] Yaras, Sami, and Murat Dener. "IoT-Based Intrusion Detection System Using New Hybrid Deep Learning Algorithm." Electronics 13, no. 6 (2024): 1053.

[19] Zhang, Xinyu, Long Li, Lina Pu, Jing Yang, Zichen Wang, Rong Fu, and Zhipeng Jiang. "Deep Learning-based Malicious Energy Attack Detection in Sustainable IoT Network." In 2024 International Conference on Computing, Networking and Communications (ICNC), pp. 417-422. IEEE, 2024.

[20] Krishna, ES & Thangavelu, A 2021, ‘Attack detection in IoT devices using hybrid metaheuristic lion optimization algorithm and firefly

Volume 38 No. 9s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

optimization algorithm', International Journal of System Assurance Engineering and Management, DOI: 10.1007/s13198-021-01150-7.

[21] Xiuzhang Yang, Guojun Peng, Dongni Zhang & Yangqi Lv 2022, 'An Enhanced Intrusion Detection System for IoT Networks Based on Deep Learning and Knowledge Graph', Security and Communication Networks, DOI: https://doi.org/10.1155/2022/4748528.

[22] Neha Yadav, Sagar Pande, Aditya Khamparia & Deepak Gupta 2022, 'Intrusion Detection System on IoT with 5G Network Using Deep Learning', Hindawi Wireless Communications and Mobile Computing, DOI: https://doi.org/10.1155/2022/9304689.

[23] Popoola, S, Adebisi, B, Hammoudeh, M, Gui, G & Gacanin, H 2021, 'Hybrid Deep Learning for Botnet Attack Detection in the Internet-ofThings Networks'. IEEE Internet Things Journal, vol. 8, pp. 4944–4956.

[24] Kan, Xiu, Yixuan Fan, Zhijun Fang, Le Cao, Neal N. Xiong, Dan Yang, and Xuan Li. "A novel IoT network intrusion detection approach based on adaptive particle swarm optimization convolutional neural network." Information Sciences 568 (2021): 147-162.

[25] Ravi, V, Chaganti, R & Alazab, M 2022, 'Deep Learning Feature FusionApproach for an Intrusion Detection System in SDN-Based IoTNetworks', IEEE Internet of Things Magazine, vol. 5, no. 2, pp. 24-29.

[26] Almaraz-Rivera, Josue Genaro, Jesus Arturo Perez-Diaz, and Jose Antonio Cantoral-Ceballos. "Transport and application layer DDoS attacks detection to IoT devices by using machine learning and deep learning models." Sensors 22, no. 9 (2022): 3367.

[27] Cakir, Semih, Sinan Toklu, and Nesibe Yalcin. "RPL attack detection and prevention in the Internet of Things networks using a GRU based deep learning." IEEE Access 8 (2020): 183678-183689.

[28] Singh, N. Joychandra, Nazrul Hoque, Kh Robindro Singh, and Dhruba K. Bhattacharyya. "Botnet-based IoT network traffic analysis using deep learning." Security and Privacy 7, no. 2 (2024): e355.

[29] Jony, Akinul Islam, and Arjun Kumar Bose Arnob. "A long short-term memory-based approach for detecting cyber-attacks in IoT using CIC-IoT2023 dataset." Journal of Edge Computing 3, no. 1 (2024): 28-42.

[30] Dataset accessed fromhttps://github.com/kahramankostas/IoTDevIDv2/blob/main/CSVs.rar

Received: August 06, 2025