

Article

CID: A Compact Deep Learning Framework for Intrusion Detection Based on Binary Greylag Goose Optimization

Sudeshna Das ¹, Abhishek Majumder ^{1,*} and Sudipta Roy ²

¹ Department of Computer Science and Engineering, Tripura University, Agartala 799022, India; sudeshna.cse@tripurauniv.ac.in

² Department of Computer Science and Engineering, Assam University, Silchar 788011, India; sudipta.roy@aus.ac.in

* Correspondence: abhim@tripurauniv.ac.in

Abstract

The application of Internet of Things-based ecosystems is growing rapidly. Cyber attacks are also increasing at a similar pace. Intrusion detection using deep learning is getting harder as these devices lack enough resources for a large Intrusion Detection System. A compact deep learning-based Intrusion Detection System for IoT, called CID, has been proposed to reduce computational complexity. The proposed CID framework uses MobileNet v1 as the main classification model, and the Binary Greylag Goose Optimization technique is used for feature selection to improve detection while minimizing processing time. On comparing the experimental results, it has been found that the proposed method works better than the baseline methods.

Keywords: Intrusion Detection System; MobileNet v1; Binary Greylag Goose Optimization; Internet of Things

1. Introduction

Cyberattacks have experienced a notable rise in frequency and sophistication, posing a serious threat to present-day digital infrastructure. Numerous industries have experienced significant data theft, extensive system failures, and substantial financial losses due to an increase in large-scale, high-impact attacks [1]. The FBI's Internet Crime Complaint Center reported more than 859,000 cybercrime complaints in 2024, with associated financial losses surpassing USD 16 billion [2,3].

Strong security is necessary to guarantee a secure and trustworthy exchange of information between organizations in today's interconnected business environments. An Intrusion Detection System (IDS) is an adaptable and dynamic tool for preserving system integrity when traditional security measures fail. It is becoming more and more important for security technologies to develop in parallel with these new threats due to the ongoing evolution of cyber threat complexities. An Intrusion Detection System [4] serves a crucial function in detecting anomalies or hostile activities. Shyaa et al. [5] proposed an adaptive IDS using a mode-switching configuration to detect attacks.

An Intrusion Detection System based on anomaly detection techniques is necessary for unseen attacks. It uses deep learning to create models of complicated traffic patterns. Deep learning has demonstrated strong performance across diverse application domains, which makes it compatible for anomaly detection in dynamic network environments. Its self-learning capability is incredibly accurate [6]. This has sparked interest among researchers in



Academic Editor: Amiya Nayak

Received: 22 February 2026

Revised: 30 May 2026

Accepted: 1 June 2026

Published: 25 June 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

using deep learning techniques for IDS. Pawana et al. [7] proposed an IDS for cloud-native environments using deep learning. Huang et al. [8] proposed a cloud-edge collaborative method for detecting DDoS attacks using an LLM architecture. In the LLM, a DDoS attack is treated as a token classification task.

The role of IDS becomes more crucial with the expansion of the IoT ecosystem. Devices in the IoT network have very low computation capacity but are vulnerable to intrusion attempts by adversaries. Conventional IDSs are computationally intensive. So, they can not be deployed for the IoT devices. For solving this problem, a lightweight IDS is essential. Many attempts have been made by researchers to design lightweight IDSs. Wakili et al. [9] proposed an IDS for known and unknown threats in an IoT network based on fusion that adaptively selects the highly reliable result. Ahmim et al. [10] proposed an IDS for IoT that uses a hybrid architecture to detect various types of DDoS attacks. The hybrid model combines various deep learning models. Different properties of the models contribute to the high performance. Wang et al. [11] proposed a lightweight MobileNet v2-based IDS. It integrates transfer learning with hyperparameter optimization. In the proposed CID system, the deep learning model, MobileNet v1 [12,13] has been considered as a classifier. MobileNet v1 provides an effective solution for embedded and mobile applications. Numerous real-world applications extensively use MobileNet v1. The resource requirements have been reduced with the advent of depthwise and pointwise convolutions in the model.

Performance enhancement of deep learning can be achieved through effective feature selection. The feature selection technique identifies the most critical variables, removes unnecessary and redundant ones, and improves the algorithms' predictive capacity. Efficiency and computational complexity can both be enhanced by using an effective feature selection technique. Metaheuristic optimization techniques are efficient in feature engineering. Grandhi and Singh [14] used Gorilla Troops Optimizer for selecting features for performing classification. Vinod et al. [15] proposed an IDS that detects and mitigates cyber-attacks. It uses an Elman Spike Neural Network. Feature selection has been done by integrating Archimedes Optimization with Fennec Fox Optimization. Jayasankar et al. [16] proposed an IDS using the RNN model for an IoT environment. For feature selection, it uses dynamic search fireworks optimization with RNN. Li et al. [17] proposed an adaptive IDS using deep transfer learning and Genetic Algorithm optimization for IoT. The framework used a pre-trained CNN to convert data into images, enabling classification. GA optimizes hyperparameters and a soft voting ensemble is used for predictions. The proposed CID system performs feature selection using a metaheuristic approach named Binary Greylag Goose Optimization (BGGO) [18]. Kenawy and Nima Khodadadi proposed the Greylag Goose Optimization (GGO) in 2023. This algorithm falls under the category of swarm intelligence. It was motivated by the social structure and collective habits of greylag geese in the time of migration. It can prevent local optima and increase the probability of finding the global optimum because of its dynamic grouping and exploration techniques.

Different metaheuristic algorithms have demonstrated their efficiency in resolving intricate optimization issues across many domains. Across many domains, optimization techniques help resolve complex nonconvex problems. The two categories of metaheuristic algorithms are single-solution and population-solution algorithms. Every solution is produced randomly in a single-solution-based algorithm until the best and most efficient one is found. In contrast, population-based algorithms produce a random number of solutions. It updates the values of each solution iteratively. Several iterations are used to develop the optimal solution.

Researchers have developed many metaheuristic algorithms, like Firefly Optimization [19], Artificial Bee Colony (ABC) [20], BGGO, Grey Wolf Optimization (GWO) [21] and Harris Hawks Optimization (HHO) [22]. Among all the algorithms, BGGO stands out as

being especially successful, doing exceptionally well in tasks including feature selection and reducing parameters [23]. The algorithm replicates the greylag goose's foraging movement patterns, such as exploration, exploitation, and flocking behavior. Using random probability, the BGGO algorithm generates a large number of individuals at first. Every individual presents a viable option that could be added to the pool of potential solutions for the problem. To discover better solutions, the goose explorer investigates the nearby present search space. The exploitation team improves the existing solutions. The BGGO algorithm selects the participants with the highest fitness after each iteration. BGGO has achieved good results across many works [24–27].

The work proposes a less resource-intensive model for intrusion detection, the CID system. The approach uses MobileNet v1, which requires less computation and produces precise classification results. BGGO has been used to select features for the MobileNet v1 model. The CID system framework is shown in Figure 1. The figure shows that a raw dataset is first preprocessed. The preprocessed data is fed into Binary Greylag Goose Optimization to extract the most informative features. After selecting the optimal features, a classifier is trained on the reduced dataset to detect intrusions with improved accuracy. This paper's contributions can be summarized as follows:

- A computationally efficient light deep learning-based intrusion detection framework, CID has been proposed that achieves high detection performance while significantly reducing model complexity. By combining CNN-based feature evaluation with MobileNet v1-based classification, the proposed system reduces inference cost without sacrificing accuracy.
- A lightweight Binary Greylag Goose Optimization has been used that effectively reduces feature dimensionality while preserving discriminative information. Compared to conventional feature selection methods, BGGO identifies a compact subset of features that improves classification performance and reduces training time.
- The proposed CID framework shows high capability to generalize across multiple benchmark datasets, namely, NSL-KDD, CICIDS2017, and TON_IoT, indicating robustness to varying traffic distributions and attack types.
- Experimental results show that the proposed approach achieves high detection accuracy and low false alarm rates while significantly lowering computational overhead.



Figure 1. Framework of the proposed CID system.

The remaining portion of this paper is formatted as follows. Related research on Intrusion Detection Systems is reviewed in Section 2. Section 3 describes the foundational components of the proposed approach: MobileNet v1 and Binary Greylag Goose Optimization. The architecture of the CID system is given in Section 4. Section 5 describes the system's performance. Section 6 wraps up the paper by reviewing the important points.

2. Related Work

In this section, some of the recent IDSs have been presented. The section also discusses the IDSs that use optimization techniques.

2.1. Intrusion Detection Techniques

This section presents several efforts on attack detection with deep neural networks. Table 1 provides a comparison of established strategies for identifying IoT-related attacks. Samunnisa et al. (2023) [28] proposed an IDS that employs classification and clustering. It categorizes attack types by utilizing threshold-based functions. The findings were evaluated against two distinct thresholds. Al-Omari et al. (2021) [29] proposed an IDS for identifying cyberattacks. The model has been developed using decision trees, prioritizing security factors. Thockchom et al. (2023) [30] proposed an IDS utilizing ensemble learning. This ensemble model comprises Logistic Regression, Gaussian naive Bayes, and a decision tree. Sarkar et al. (2023) [31] introduced an ensemble method for classifying intrusion detection. Data augmentation is used for rebalancing data. A cascaded meta-specialized classifier framework has been created for classification. Sedhuramalingam et al. [32] proposed an IDS to detect asymmetrical attacks in a wireless sensor network. Using the coyote optimization, the hyperparameters are selected to determine network topologies and network parameters for a deep neural network.

Table 1. Comparison of Intrusion Detection Systems.

Paper	Model and Dataset	Merits	Demerits
Al-Omari et al. 2021 [29]	Model: decision trees. Dataset: UNSW-NB15.	Attacks detected: generic, exploits, analysis, shellcode, DoS, reconnaissance, worms, backdoors, fuzzers.	Deep or complex tree structures require significant computational resources. Static decision tree models have limited adaptability to evolving threats.
Samunnisa et al. 2023 [28]	Model: K-means, GMM, RF, SVM, KNN and SVM. Dataset: NSL-KDD and KDDcup99.	Attacks detected: DoS, probing, U2R and R2L have been detected.	Large hybrid model is computationally heavy and old datasets are considered.
Thockchom et al. 2023 [30]	Model: ensemble model: decision tree and Logistic Regression, Gaussian naive Bayes. Dataset: CIC-IDS2017, KDD Cup 1999 and UNSW-NB15.	Attacks detected: reconnaissance, brute force FTP attack, worms, web attack, DoS attack, backdoors, shellcode, DoS, generic, U2R, brute force SSH attack, fuzzers, analysis, R2L, probing, heartbleed attack, DDoS, infiltration, exploit and botnet.	Combining multiple models in an ensemble increases computational complexity.
Sarkar et al. 2023 [31]	Model: ensemble technique: decision trees, additional trees, Random Forests, naive Bayes, SVM and MLP, Logistic Regression, gradient boosting, K-Nearest Neighbors. Dataset: KDD Cup99 and NSL-KDD.	Attacks detected: R2L, DoS, U2R and probing have been detected.	The cascaded meta-specialist classifier and ensemble structure require additional computational resources. The datasets used here are popular but may not fully represent modern attack patterns.
Ghadami et al. 2025 [33]	Model: parallel convolutional neural network and long short-term memory. Dataset: NSL-KDD, UNSW-NB15	Attacks detected: DoS, probing, U2R, R2L, generic, exploits, analysis, shellcode, DoS, reconnaissance, worms, backdoors, fuzzers.	It requires high computational resources.

Deep neural networks' exceptional performance has encouraged deep learning applications across a wide range of sectors. However, the widespread adoption of deep learning has been impeded by the possible dangers posed by adversarial samples. Zhou et al. [34] conducted a review of the lifecycle of adversarial attacks and defenses in cybersecurity. Guo et al. [35] proposed an adversarial training method. It consists of two steps: a historical gradient-based adversarial attack and domain-adaptive training to enhance the adversarial robustness. Che et al. [36] proposed a Large Language Model Adversarial Defense technique based on perturbation detection and correction. It improves the performance on adversarial defense tasks and enables accurate, efficient correction of adversarial samples. Guo et al. [37] proposed a bidirectional long short-term memory Kolmogorov–Arnold network that predicts feature values of current data and reconstructs historical data features, enabling consistent bidirectional representation.

2.2. Intrusion Detection Using Metaheuristic Optimization Techniques

In this section, existing works on intrusion detection using metaheuristic optimization techniques have been presented. A comparison of these works has been presented in Table 2. Nasir et al. (2022) [38] provided a systematic review of works published from 2010 to 2020 regarding swarm intelligence methodologies applied to diverse attack surfaces for intrusion detection across multiple domains. The study provides a classification based on the way these swarm intelligence (SI) technologies work in various aspects of the intrusion detection process. Reddy et al. (2024) [39] provide a review of recent developments in swarm intelligence techniques for IoT-based Intrusion Detection Systems. It covers various applications, evaluates comparative performance metrics and highlights directions for future research. The study examined the technological aspects of executing feature selection and parameter optimization in SI. Moreover, it conducts an examination of SI methodologies within the context of IDS in IoT. Donkol et al. (2023) [40] proposed an IDS using a long short-term memory method utilizing an RNN to improve security in IDS. The approach addresses the gradient-clipping problem by utilizing probable point PSO. Alzaqebah et al. (2023) [22] introduced a bio-inspired metaheuristic approach for the detection of multi-stage cyberattacks. To handle the classification complexity, each sub-model is designed to distinguish a single class from the rest. The sub-models leverage an upgraded version of the Harris Hawk Optimization, integrated with an extreme learning machine. Kolukisa et al. (2024) [41] addressed cybersecurity issues in network intrusion detection by introducing a novel method that employs a Logistic Regression model optimized using Artificial Bee Colony Optimization. Srivastava et al. (2024) [42] proposed an IDS that uses a two-stage swarm intelligence technique to manage high-dimensional data. In the first stage, particle swarm optimization identifies critical features while mitigating dataset imbalance. In the second stage, Ant Colony Optimization identifies salient, uncorrelated features. A Genetic Algorithm (GA) optimizes each detection model. Bakro et al. (2024) [43] proposed an IDS for cloud computing, targeting security vulnerabilities and data redundancy. For feature selection, the Grasshopper Optimization Algorithm (GOA) and GA are combined. Random Forest (RF) is trained on these optimized features. Elsaid et al. (2024) [21] proposed an IDS that combines GRU/LSTM with Grey Wolf Optimization for adaptive feature tuning and detection. Kaur et al. (2018) [44] proposed a hybrid IDS that merges K-Means clustering and the Firefly Optimization. The model clusters training data and classifies test data for intrusion detection. Alazab et al. (2024) [45] proposed an IDS using Harris Hawks Optimization to optimize MLP training by tuning weights and biases.

Table 3 provides the comparative analysis of feature selection through metaheuristic optimization in Intrusion Detection System. The table shows various models that use meta-

heuristic optimization techniques for feature selection. Some of the works are lightweight and performed intrusion detection for IoT device.

Table 2. Comparison of Intrusion Detection Systems using metaheuristic optimization.

Reference	Key Features	Merits	Demerits
Elsaid et al. 2024 [21]	Combines GRU/LSTM deep neural networks with Grey Wolf Optimization for adaptive feature tuning and detection. Dataset: NSL-KDD.	U2R, R2L, DoS and probing attacks have been detected.	Although feature selection reduces dimensionality, integrating GWO with large deep learning models like GRU or LSTM introduces computational complexity during the training phase.
Alzaqebah et al. 2023 [22]	Harris Hawk Optimization with an extreme learning machine has been used. Dataset: UNSW-NB15.	Shellcode, backdoors, worms, fuzzers, reconnaissance, generic, exploit, DoS and analysis attacks have been detected.	UNSW-NB15 is a strong benchmark dataset, but it may not fully capture modern attack variations. So, it requires testing on newer datasets.
Kolukisa et al. 2024 [41]	Logistic Regression and Artificial Bee Colony Optimization have been used. Dataset: UNSW-NB15 and NSL-KDD.	Attacks such as probing, shellcode, reconnaissance, DoS, fuzzers, worms, R2L, backdoors, generic, exploit, U2R, and analysis have been detected.	The Artificial Bee Colony algorithm, especially in a parallel implementation, may require significant computational resources.
Srivastava et al. 2024 [42]	PSO, GA, Ant Colony Optimization and XGBoost have been used. Datasets: NSL-KDD, CSE-CIC-IDS2018 and UNSW-NB15.	Attacks such as probing, shellcode, reconnaissance, DoS, fuzzers, worms, R2L, backdoors, generic, U2R, exploit and analysis have been detected.	Use of multiple metaheuristic algorithms increases processing time.
Donkol et al. 2023 [40]	PSO and enhanced LSTM have been used. Datasets: CSE-CIC-IDS2018, CICIDS2017, UNSW-NB15, and BOT.	Attacks such as web attack, heartbleed, reconnaissance, brute force SSH, generic, fuzzers, botnet, backdoors, infiltration, worms, analysis, DDoS, exploit, brute force FTP, DoS, and shellcode have been detected.	LSTM-based models require substantial computational resources, making them less ideal for low-power devices.
Bakro et al. 2024 [43]	Grasshopper Optimization Algorithm, Genetic Algorithm, and Random Forest have been used. Datasets: CIC Bell DNS EXF 2021, UNSW-NB15, CIC-DDoS2019.	Analysis, DoS, exploit, backdoors, generic, DNS exfiltration, reconnaissance, DDoS, worms, fuzzers and shellcode attacks have been detected.	Computational overhead due to dual optimization and RF training.
Yesodha et al. 2024 [46]	Artificial Bee Colony Optimization, fuzzy temporal rules, and CNN have been used. Dataset: NSL-KDD.	U2R, R2L, DoS and probing attacks have been detected.	The FT-ABC-CNN model may perform well on specific datasets or attack types but could struggle to generalize to new, unseen attacks or different WSN configurations without retraining or fine-tuning.
Kaur et al. 2018 [44]	Uses Firefly Optimization for feature selection. K-Means is used for anomaly detection. Dataset: NSL-KDD.	U2R, R2L, DoS and probing attacks have been detected.	The Firefly Algorithm's performance relies on parameters like light intensity, attractiveness, and absorption coefficient; improper tuning may cause poor convergence. K-Means assumes that clusters are the same size and shape, which may not be true for real network traffic distributions. This makes clustering and detection less accurate.

A critical limitation in the literature is the lack of alignment between feature selection mechanisms and the architecture of lightweight classifiers. While MobileNet v1 has been widely recognized for its efficiency in embedded and edge environments, prior IDS studies rarely integrate it with tailored feature selection strategies. This work proposes a jointly optimized IDS framework that tightly couples feature selection and classification. Specifically, a Binary Greylag Goose Optimization is employed for feature selection, which offers improved balance between exploration and exploitation in binary search spaces compared to traditional swarm-based methods. The selected optimal feature subset is then fed into MobileNet v1, a lightweight deep learning architecture chosen for its low computational complexity and suitability for real-time IoT intrusion detection.

Table 3. Comparative analysis of recent works on feature selection in Intrusion Detection Systems.

Reference	Method Type	Classification Model	Datasets Used	Accuracy (%)	Feature Selection Mechanism	Lightweight Design
Vinod et al. 2026 [15]	Deep Learning	Elman Neural Network	UNSW-NB15	96.38%	Archimedes Optimization Algorithm and Fennec Fox Optimization Algorithm	–
Karthikeyan et al. 2025 [47]	Deep Learning	Ensemble of Bidirectional Gated Recurrent Unit, Wasserstein Auto Encoder, and Deep Belief Network	CICIDS-2017 and NSLKDD	99.14%	Adaptive Harris Hawk Optimization	Yes
Chen et al. 2025 [48]	Deep Learning	CNN	Bot-IoT, NSL-KDD, and CICIDS2018	97.8%	Enhanced Pelican Optimization Algorithm	Yes
Al-Shurbaji et al. 2025 [49]	Deep Learning	Generative Adversarial Network, Hybrid CNN-LSTM	BoT-IoT	97%	Particle Swarm Optimization and Gorilla Troops Optimizer	Yes
Nafaa Jabeur 2025 [50]	Machine Learning	XGBoost	METABRIC and KDD datasets	81%	Firefly Algorithm	–
Dharmalingam et al. 2025 [51]	Deep Learning	Convolutional and Auto Encoder	DS2OS and BoT-IoT datasets	97%	Adaptive Eagle Cat Optimization	Yes
Farhan et al. 2025 [4]	Deep Learning	Deep Neural Network	UNSW-NB15	97.93%	Extra Tree Classifier	–
Misrak et al. [52]	Deep Learning	DNN-BiLSTMQ	CIC-IDS2017 and CIC-IoT2023	99.73%	RAL-MIFS	Yes
Sun et al. [53]	Deep Learning	LeNet	NSL-KDD and CICIoV2024	94.93%	XGBoost	Yes
Sharma et al. [54]	Deep Learning	DNN	BoT-IoT, N-BaIoT, UNSW-NB15	98.9%	LFG	Yes

3. Background Techniques

The CID system uses Binary Greylag Goose Optimization and MobileNet v1. Therefore, in this section, these systems have been discussed.

3.1. Binary Greylag Goose Optimization

In this work, simple Binary Greylag Goose Optimization has been used for limited resource devices. The BGGO algorithm got inspiration from the social dynamics and migratory behavior of the greylag goose. Geese are capable of engaging in extensive migratory flights, sometimes traversing thousands of kilometers in one go. This can be possible due to their collective flight behavior. In this method, each goose represents a candidate feature subset and updates its position through leader-following behavior, interaction with neighboring geese and stochastic perturbations. The algorithm implicitly balances exploration and exploitation without explicitly partitioning the population. The optimization starts with a random population of potential solutions. The objective function is applied to evaluate each solution's quality. It directs the algorithm's search toward the optimal candidate solution. The optimal candidate solution is the highest-performing solution identified within the population.

The BGGO algorithm balances local search and broader global exploration, which lowers the risk of stagnation. After each iteration, the positions of the solutions are updated, and the roles of each member of the swarm are randomly reassigned. This dynamic reshuffling keeps diversity and raises the algorithm's capacity to converge toward a global optimum. Let each solution or goose be represented by a binary vector shown in Equation (1):

$$Y_i = [y_{i1}, y_{i2}, \dots, y_{iD}], \quad y_{ij} \in \{0, 1\} \quad (1)$$

Here, i is for index the goose, $j = 1, 2, \dots, D$ is for the dimension of feature, and D denotes the total count of features.

Velocity is updated using Equations (2) and (3). Binary position is updated using Equation (4). Fitness function is calculated using Equation (5).

$$u_i(t+1) = r_1 \cdot u_i(t) + r_2 \cdot (b_{\text{best}} - b_i(t)) + r_3 \cdot (b_{\text{rand}} - b_i(t)) \quad (2)$$

$$S(u_{ij}) = \frac{1}{1 + e^{-v_{ij}}} \quad (3)$$

Here, $u_i(t)$ is the velocity of goose i at iteration t , b_{best} is the position of the best solution, $b_i(t)$ is the position of the i th goose at iteration t , b_{rand} is the position of a randomly selected solution, and $r_1, r_2, r_3 \in [0, 1]$ are random control parameters.

$$b_{ij}(t+1) = \begin{cases} 1, & \text{if rand} < S(u_{ij}) \\ 0, & \text{otherwise} \end{cases} \quad (4)$$

Here, $b_{ij}(t+1)$ is the position of goose i at iteration $t+1$ and $\text{rand} \in [0, 1]$ is a random number with a uniform distribution.

$$\text{Fitness}(b_i) = \alpha \cdot \text{Error}(b_i) + (1 - \alpha) \cdot \frac{|b_i|}{D} \quad (5)$$

Here, $\text{Error}(b_i)$ is the error that occurs during classification using the selected features. $|b_i|$ represents the number of selected features. $\alpha \in [0, 1]$ is a weight parameter that balances between the error and the selected features.

The key idea in selecting features in BGGO is that it uses a velocity-based mechanism to update binary solutions iteratively. The best solution and random neighbors guide each solution. The solution is then converted into binary decisions using a sigmoid function.

3.2. MobileNet v1

MobileNet v1 is a deep CNN model developed by Google. It is lightweight and efficient. It is mainly designed for light applications. For mobile and embedded vision, it is very effective. MobileNet v1 achieves a significant reduction in model parameters by adopting depthwise separable convolution as its core architectural component. This convolutional technique lowers computational complexity but maintains high classification accuracy. So, MobileNet v1 can be used on devices with limited resources, like smartphones, IoT devices, and edge computing systems.

Depthwise separable convolution splits the standard convolution operation into two steps: depthwise convolution and pointwise convolution. During the depthwise convolution stage, a single spatial filter is used on each input channel separately. This makes it easy to get channel-specific spatial features. Pointwise convolution then uses a 1×1 kernel to linearly combine the outputs across channels, enabling inter-channel feature interaction. This factorized design leads to a considerable reduction in both parameter count and computational cost, thereby enabling MobileNet v1 to deliver strong performance even under strict resource limitations. With a reduction in parameters, this approach also preserves a high level of accuracy. As a result, it is significantly effective in lightweight models designed for mobile and embedded systems.

3.2.1. Depthwise Convolution

In depthwise convolution, a convolutional filter is applied separately to each input channel. This approach is different from standard convolution, where each filter is applied across all input channels simultaneously. Let the input tensor be $\mathbf{I} \in \mathbb{R}^{\mathbf{h}_i \times \mathbf{w}_i \times \mathbb{C}}$, where $\mathbf{h}_i$ is the height and $\mathbf{w}_i$ is the width of the input feature map. $\mathbb{C}$ counts the input channels. For each channel c , there is a dedicated convolution kernel $\mathbf{F}^{(c)} \in \mathbb{R}^{k \times k}$, where k is the kernel size. The output tensor $\mathbf{O} \in \mathbb{R}^{\mathbf{h}_o \times \mathbf{w}_o \times \mathbb{C}}$ is computed using Equation (6):

$$O^{(d)}(x, y) = \sum_{b=0}^{k-1} \sum_{e=0}^{k-1} F^{(d)}(b, e) \cdot I^{(d)}(x + b, y + e), \quad (6)$$

For all $d = 1, 2, \dots, \mathbb{C}$, and for all valid spatial positions (x, y) in the output. Here, $\mathbf{h}_o$ is for height and $\mathbf{w}_o$ is for width of the output feature map. $O^{(d)}(x, y)$ is the output feature map value at spatial position (x, y) for the d -th channel. $F^{(d)}(b, e)$ is the kernel weight at position (b, e) for the d -th channel. $I^{(d)}(x + b, y + e)$ is the input feature map value at position $(x + b, y + e)$ for the d -th channel.

3.2.2. Pointwise Convolution

Pointwise convolution layer applies a 1×1 convolutional filter to compute a linear combination of the outputs from the depthwise convolution, effectively integrating information across channels and generating the final feature map. Each filter processes all channels at a single spatial location and is mainly used to merge channel information. Consider an input tensor $\mathbf{I} \in \mathbb{R}^{\mathbf{h}_i \times \mathbf{w}_i \times \mathbb{C}}$, where $\mathbf{h}_i$ and $\mathbf{w}_i$ represent the spatial dimensions. $\mathbb{C}$ counts the input channels. For F filters, each of size $1 \times 1 \times \mathbb{C}$, the output tensor $\mathbf{O} \in \mathbb{R}^{\mathbf{h}_i \times \mathbf{w}_i \times F}$ is produced.

For each spatial position (x, y) , the output value at channel f is given by Equation (7):

$$O^{(f)}(x, y) = \sum_{c=1}^{\mathbb{C}} V_c^{(f)} \cdot I^{(c)}(x, y) \quad (7)$$

Here, $I^{(c)}(x, y)$ shows the input value at location (x, y) in channel c . $V_c^{(f)}$ is the weight connecting the input channel c to the output channel f . This operation corresponds to the dot product between the filter vector $\mathbf{V}^{(f)} \in \mathbb{R}^{\mathbb{C}}$ and the input vector at location (x, y) across all channels.

4. Proposed Technique

In this section the problem definition and methodology have been discussed.

4.1. Problem Definition

In this work the attacks mentioned in NSL-KDD, CICIDS2017 and TON_IoT datasets have been considered. Table 4 depicts the attack labels used in the datasets.

Table 4. Mapping of original dataset labels to attack categories.

Dataset	Attack Category	Original Labels
NSL-KDD	DoS	smurf, apache2, udpstorm, land, processtable, teardrop, back, mailbomb, pod, neptune
	Probe	nmap, satan, portsweep, ipsweep, saint, mscan
	R2L	snmpguess, warezmaster, imap, xlock, guess_passwd, httptunnel, spy, multihop, phf, ftp_write, xsnoop, named, sendmail, warezclient, snmpgetattack
	U2R	rootkit, perl, sqlattack, buffer_overflow, ps, loadmodule
TON_IoT	DoS	dos
	DDoS	ddos
	Access/Auth	mitm, password
	Malware	ransomware, backdoor
	Web/Injection	injection, xss
CICIDS2017	Probe	scanning
	DoS	DoS slowloris, Heartbleed, DoS Hulk, DoS Slowhttptest, DoS GoldenEye
	DDoS	DDoS
	Infiltration	Infiltration
	Brute Force	FTP-Patator, SSH-Patator
	Probe/Scanning	PortScan, Bot
	Web Attack	Web Attack-Sql Injection, Web Attack-Brute Force, Web Attack-XSS

4.2. Classification Framework

The CID system is modeled as a binary classification model. Here, the input feature vector is mapped to a label $y \in \{0, 1\}$.

4.2.1. Label Encoding

For binary space the mapping function $f(y_{orig})$ is shown in Equation (8):

$$f(y_{orig}) = \begin{cases} 0, & \text{if } y_{orig} \in \{\text{'normal'}, \text{'BENIGN'}\} \\ 1, & \text{if } y_{orig} \in \text{Attack Set} \end{cases} \quad (8)$$

4.2.2. Output Activation and Loss Function

For binary classification, the output layer employs a sigmoid activation function. The Adam optimizer has been used to update the parameters effectively. Binary crossentropy, as stated in Equation (9), has been used as the loss function:

$$\mathcal{L}(\theta) = -\frac{1}{N} \sum_{i=1}^N [s_i \log(\hat{s}_i) + (1 - s_i) \log(1 - \hat{s}_i)] \quad (9)$$

Here,

- N = Number of samples in the dataset or batch.
- s_i = True label for sample i , where $s_i \in \{0, 1\}$.
- $\hat{s}_i$ = Predicted probability for sample i from the model.

4.3. Tabular Data into Image Data Conversion in MobileNet v1

MobileNet v1 is designed to process image data in grid format (height $\times$ width $\times$ channels). Tabular data, on the other hand, is usually a 1D vector of features. To bridge this gap, a series of transformation steps is performed to represent the tabular features as an image. Here's a breakdown of the theoretical process, mirroring the implementation:

- **Padding to Form a Square Image:**
The selected tabular features, which form a 1D vector $[f_1, f_2, \dots, f_n]$, need to be arranged into a square or rectangular grid. Since MobileNet v1 operates on grids, the most common approach is to find the smallest square that can accommodate all n features. This means determining an `image_height` and `image_width` such that `image_height * image_width >= n`. If n is not a perfect square, zero-padding is applied. New dummy features with a value of 0 are added to the 1D feature vector until its length becomes a perfect square. This ensures the features can be reshaped into a symmetric square grid.
- **Reshaping into a Grayscale Image:**
Once the feature vector has been padded to a square length n' , it is reshaped into a 2D matrix of dimensions $\sqrt{n'} \times \sqrt{n'}$. This 2D matrix represents a single-channel image. The values in this single-channel image are the scaled feature values from the original tabular data. Darker or lighter pixels correspond to smaller or larger feature values after normalization. The image shape at this stage is typically `(num_samples, image_height, image_width, 1)`.
- **Resizing to MobileNet v1's Expected Input Dimensions:**
The grayscale image created in the previous step is then resized. MobileNet v1 model converts the input image sizes to 128×128 . This resizing is performed using bilinear interpolation while attempting to preserve the spatial relationships of the features within the image. The image shape remains `(num_samples, MobileNet_H, MobileNet_W, 1)`.
- **Channel Duplication to Form Three Channels:**
MobileNet v1 typically expects three input channels. Since the image is currently one channel, this single channel is duplicated three times. This effectively transforms the image from `(num_samples, H, W, 1)` to `(num_samples, H, W, 3)`.

After these steps, the tabular data is transformed into a three-channel image format that MobileNet v1 can directly process.

4.4. Methodology

A detailed representation of the workflow of the CID system is provided in Figure 2. A CID system follows some steps. The dataset is first divided into 80% training and 20% testing sets. Categorical features are encoded using one-hot encoding [55], where the encoder is fitted exclusively on the training set and subsequently applied to the test set. Numerical features are normalized using Min-Max scaling [56] as defined in Equation (10), with parameters learned only from the training data and reused for the test data. To address class imbalance, the Synthetic Minority Oversampling Technique (SMOTE) [57] is applied solely to the training set after preprocessing. After applying SMOTE, the class distribution was balanced to an approximately equal ratio between normal and attack samples while maintaining realistic evaluation on the untouched test set. Feature selection using BGGO

is also conducted exclusively on the training data. So, after completing all preprocessing steps on the input data, BGGO was used to generate an optimal feature subset by treating each goose as a binary vector representing a possible subset. The fitness value of each candidate feature subset is evaluated using a CNN classifier. Experiments were conducted over five independent runs. Guided by the best-performing goose, the population evolves through binary position updates to progressively identify optimal feature subsets. After the feature selection process, the MobileNet v1 is trained on the selected features.

$$v' = \frac{v - v_{\min}}{v_{\max} - v_{\min}} \quad (10)$$

Algorithm 1 describes the BGGO algorithm. Symbols used in the algorithm are given in Table 5. BGGO is executed for problems where solutions are represented as binary strings. In the beginning, a population of P geese is formed, where each goose represents a candidate feature subset encoded as a binary vector of length m . The velocity of each goose is initialized to a zero vector. A weighted objective function is used to assess each candidate solution's fitness. The objective function combines the classification error produced by the CNN classifier and the proportion of selected features, which is controlled by a trade-off parameter $\alpha = 0.7$. The random weights r_1 , r_2 , and r_3 are uniformly distributed and are in the range $[0, 1]$. The values will be different every time during execution. The goose with the minimum fitness value is chosen as the leader, which guides the search process. Throughout the iterative cycles, each goose updates its velocity by using its previous momentum with an attraction toward the leader and an interaction with a randomly selected neighbor. These continuous velocity values are subsequently mapped into a discrete probability space via a sigmoid transfer function. It allows stochastic binary updates to the feature subset. By iteratively refining these subsets and updating the leader, the algorithm effectively navigates the high-dimensional search space of datasets to converge on a near-optimal feature configuration. This iterative optimization procedure is carried out repeatedly until the predefined maximum iterations are reached. After which, the final leader represents the optimal feature subset S^* along with its corresponding best fitness value $f(S^*)$.

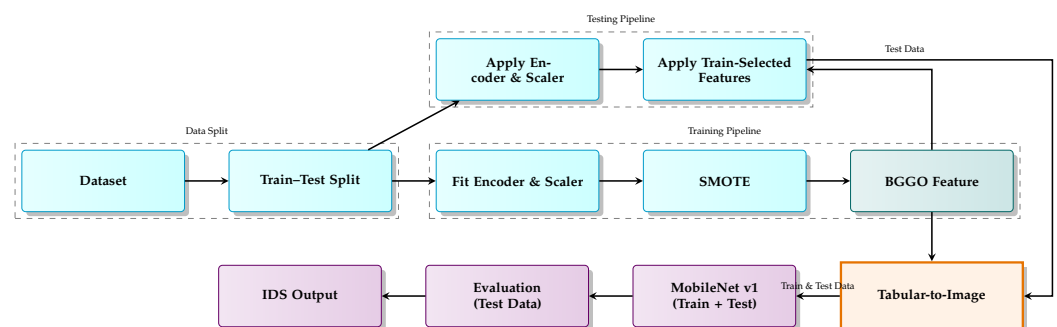


Figure 2. Workflow of CID system.

Finally, the chosen features from the processed training set are used to train the MobileNet v1 classifier. The algorithm's time complexity is computed as $\mathcal{O}(L_{\max} \cdot P \cdot (m + T_f))$. Here m is for dimensionality, P is the size of the population, $L_{\max}$ is the maximum iteration, and T_f is the time to evaluate the fitness using the classifier. Space complexity of the algorithm is computed as $\mathcal{O}(P \cdot m)$. Space complexity shows no additional memory-intensive operations.

Table 5. Description of symbols used in Binary Greylag Goose Optimization (BGGO).

Symbol	Description
$\mathcal{D}$	Dataset containing m features
m	Total number of features in the dataset
P	Population size
$L_{\max}$	Maximum number of iterations
b_i	Binary position vector of the i th goose in $\{0, 1\}^m$
u_i^l	Velocity vector of the i th goose at iteration l
b_{best}	Current best feature subset
$\text{Fitness}(b_i)$	Fitness value of the i th solution
α	Trade-off parameter between classification error and feature reduction
r_1, r_2, r_3	Random coefficients used for velocity update
$S(u_{ij})$	Sigmoid transfer function for the j th dimension
$ b_i $	Number of selected features in solution b_i

Algorithm 1 Feature selection using Binary Greylag Goose Optimization (BGGO)

- 1: **Input:** Dataset $\mathcal{D}$ with m features, population size P , maximum iterations $L_{\max}$, classifier, trade-off parameter α
- 2: **Output:** Optimal feature subset b^* and corresponding fitness value $\text{Fitness}(b^*)$
- 3: Initialize binary population $b_i \in \{0, 1\}^m$, for $i = 1, 2, \dots, P$
- 4: Initialize velocity vectors $u_i = [0, 0, \dots, 0]$
- 5: **for** each goose b_i **do**
- 6: Evaluate fitness:

$$\text{Fitness}(b_i) = \alpha \cdot \text{Error}(b_i) + (1 - \alpha) \cdot \frac{|b_i|}{m}$$

- 7: **end for**
- 8: Determine leader:

$$b_{\text{best}} = \arg \min \text{Fitness}(b_i)$$

- 9: **for** $l = 1$ to $L_{\max}$ **do**
- 10: **for** each goose $i = 1$ to P **do**
- 11: Select a random neighbor b_{rand}
- 12: Update velocity:

$$u_i^{l+1} = r_1 u_i^l + r_2 (b_{\text{best}} - b_i^l) + r_3 (b_{\text{rand}} - b_i^l)$$

- 13: **for** each dimension $j = 1$ to m **do**
- 14: Apply sigmoid function:

$$S(u_{ij}) = \frac{1}{1 + e^{-u_{ij}}}$$

- 15: Update position:

$$b_{ij}^{l+1} = \begin{cases} 1, & \text{if rand} < S(u_{ij}) \\ 0, & \text{otherwise} \end{cases}$$

- 16: **end for**
- 17: Evaluate $\text{Fitness}(b_i^{l+1})$
- 18: **if** $\text{Fitness}(b_i^{l+1}) < \text{Fitness}(b_{\text{best}})$ **then**
- 19: $b_{\text{best}} \leftarrow b_i^{l+1}$
- 20: **end if**
- 21: **end for**
- 22: **end for**
- 23: **return** $b^* = b_{\text{best}}, \text{Fitness}(b^*)$

In MobileNet v1, the data is reshaped into an image-like format. The selected feature vectors are transformed into image representations by reshaping them into square matrices,

where the side length is determined from the total number of features, and zero-padding is applied where necessary to form a complete square grid. The resulting matrices are treated as single-channel images and reshaped accordingly. These images are then resized to a fixed dimension of 128×128 using interpolation. The final transformed data, with shape (num_samples, 128, 128, 1), is fed into MobileNet v1. The MobileNet v1 architecture comprises an input layer. The input layer accepts an image format data that is converted from tabular data. Then come the depthwise and pointwise convolutional layers. Next comes a global average pooling layer. Using global average pooling, the feature maps are configured into a one-dimensional vector for the dense layer. The next layer in the architecture contains one fully connected dense layer. In the dense layer, ReLU6 activation is used to learn nonlinear combinations of the extracted features. In the output, the sigmoid activation function has been used. MobileNet v1 utilized a width multiplier of 0.75. The number of training epochs for MobileNet v1 = 10.

For detecting intrusion, the selected features are used for model training. The five-fold cross-validation is employed on the training data to validate model performance. In the testing phase, the final trained model is applied to the unseen 20% test data.

In the CID system, the computational complexity is analyzed across different stages. During the preprocessing phase, let N denote the number of network traffic instances and m represent the initial number of features. The time complexity for preprocessing is $\mathcal{O}(N \cdot m)$, while the space complexity is $\mathcal{O}(N \cdot m')$, where m' denotes the expanded feature set after one-hot encoding. For the tabular-to-image conversion process, the time complexity is $\mathcal{O}(m)$, and the space complexity is $\mathcal{O}(e^2)$, where $e \times e$ represents the generated image dimension. For the MobileNet v1 model, the overall time complexity is given by $\mathcal{O}\left(\sum_{i=1}^L \left(k^2 \cdot M_i \cdot d_i^2 + M_i \cdot N_i \cdot d_i^2\right)\right)$, where for each layer i , $k \times k$ denotes the kernel size, M_i is the number of input channels, N_i is the number of output channels, and $d_i \times d_i$ represents the spatial dimension of the feature map. The per-sample inference complexity of the CID system is $T_{\text{inference}} = \mathcal{O}(f^* \cdot C)$ where f^* is the number of features selected by Binary Greylag Goose Optimization and C is the computational cost of MobileNet v1.

5. Performance Analysis and Comparison

In this section the performance of the CID system has been analyzed and compared with the baselines.

5.1. Assumptions

The experimental design of the proposed intrusion detection framework is based on several key assumptions that define its scope and generalizability. It is assumed that the benchmark datasets NSL-KDD, CICIDS2017, and TON_IoT sufficiently represent real-world network traffic. Specifically, TON_IoT captures contemporary IoT-specific protocols such as MQTT at the edge-to-cloud interface, and the others provide diversity in traditional and modern attack scenarios, thereby serving as reasonable proxies for deployment environments. It is assumed that the training and testing splits are independently and identically distributed, with samples randomly drawn to represent the overall dataset distribution. To prevent class distributions from biasing the learning process, the SMOTE is applied to balance the training data. This introduces the assumption that the synthetic samples generated via linear interpolation accurately map to the true, feasible manifold of malicious network behavior. To mitigate the inherent threats to validity posed by synthetic data—such as artificial variance or overoptimistic performance metrics—the framework strictly enforces that oversampling is executed only after the train–test split. The evaluation partition remains entirely unaltered, ensuring that the model is validated against an authentic, skewed distribution mirroring real-world production traffic. Under these

conditions, it is assumed that the Binary Greylag Goose Optimization algorithm selects a robust, generalized feature subset without overfitting to dataset-specific patterns, enabling the lightweight classifier to effectively detect unseen anomalies in live IoT traffic. The framework assumes deployment in strictly resource-constrained edge environments. The evaluations are conducted on standard computing hardware, and therefore the resource efficiency claims are based on architectural design considerations rather than direct validation under strict hardware constraints. By selecting a lightweight MobileNet v1-based backbone, the system assumes a deployment environment capable of executing depthwise separable convolutions within a tight memory footprint and a limited parameter budget, optimizing CPU cycles to preserve edge-node energy efficiency.

5.2. Dataset

For evaluation, multiple benchmark datasets are utilized, including NSL-KDD [58], CICIDS2017 [59] and TON_IoT [60].

In the NSL-KDD dataset, 41 features are present. The dataset consists of five distinct categories. NSL-KDD is popular in the literature associated with intrusion detection. In the CICIDS2017 dataset, there are 78 different features. The class encompasses attacks of different types. The TON_IoT dataset is made up of both raw and processed data sources. The sources of the data are network traffic statistics, Windows and Linux operating system datasets and telemetry datasets of Internet of Things services. The network dataset in TON_IoT consists of 42 features. For all the evaluating datasets, the attacks and the labels associated with the attacks are given in Table 4.

5.3. Evaluation Metrics

Performance evaluation has been performed using following metrics:

- Precision: Precision can be calculated using Equation (11):

$$\text{Precision} = \frac{P_{\text{true}}}{P_{\text{true}} + P_{\text{false}}} \quad (11)$$

where P_{true} counts the true positives and P_{false} counts the false positives.

- Recall: Recall can be calculated using Equation (12):

$$\text{Recall} = \frac{P_{\text{true}}}{P_{\text{true}} + N_{\text{missed}}} \quad (12)$$

where N_{missed} indicates the number of false negatives.

- Accuracy: Accuracy can be calculated using Equation (13):

$$\text{Accuracy} = \frac{P_{\text{true}} + N_{\text{true}}}{P_{\text{true}} + P_{\text{false}} + N_{\text{missed}} + N_{\text{true}}} \quad (13)$$

where N_{true} refers to the number of true negatives.

- F1 Score: F1 score is calculated using Equation (14):

$$F_1 = \frac{2 \cdot P_{\text{true}}}{2 \cdot P_{\text{true}} + P_{\text{false}} + N_{\text{missed}}} \quad (14)$$

- False Alarm Rate (FAR): FAR evaluates the likelihood of misclassification of benign samples as malicious. It is determined using Equation (15):

$$\text{FAR} = \frac{P_{\text{false}}}{P_{\text{false}} + N_{\text{true}}} \quad (15)$$

5.4. Results and Discussion

The experiments are performed on a Dell machine featuring an Intel Core i5-1135G7 processor and 16 GB RAM. The implementation was done in Python 3.12.7. Evaluation results are depicted in several figures and tables. Figure 3 shows the convergence curve of BGGO across different datasets. Lines represent the mean best fitness, and shaded regions denote the standard deviation ($\pm$ SD) over five runs. The BGGO algorithm was executed for a maximum of 100 iterations. Population size was set to 30. The NSL-KDD dataset converges by 40 iterations, while CIC-IDS2017 required about 85 iterations and the TON_IoT dataset converges at about 70.

Table 6 provides the feature reduction percentage using BGGO. For Table 7, the baseline accuracy values were reproduced under the identical experimental conditions. The table was obtained using the independent 20% test partition. The SVM was configured with an RBF kernel and a regularization parameter $C = 1.0$. For KNN, the optimal number of neighbors was determined to be $k = 5$. The deep learning baselines, CNN and LSTM, were implemented with a fixed learning rate of 0.001, a batch size of 64, and a dropout rate of 0.2. Table 7 provides a comparison of performance in the CID system and other models. The CID system consistently outperforms MobileNet v1, SVM, KNN, LSTM and CNN across all datasets with accuracies of 97.30% in NSL-KDD, 96.89% in CICIDS2017 and 96.30% in TON_IoT. Then, MobileNet v1 achieves a high accuracy of 96.20% in NSL-KDD, 95.30% in CICIDS2017, and 95.80% in TON_IoT. SVM also achieves strong results, 94% accuracy on NSL-KDD due to its effectiveness in high-dimensional spaces, but it struggles with scalability for large datasets like CICIDS2017. LSTM performs well on the CICIDS2017 dataset. KNN's lower performance on NSL-KDD with 91.5% and CICIDS2017 with 91.6% is because it is sensitive to high-dimensional data.

High precision values in the CID system mean a low false positive rate, which makes its deployment less expensive. Using BGGO and MobileNet v1 together helps avoid problems like overfitting and getting stuck in local optima, which are common in deep learning-based IDS models. Also, because MobileNet v1 uses depthwise separable convolutions, it can produce useful representations even on devices with limited resources. BGGO's selected features help MobileNet v1's learning process focus on discriminative patterns, thereby increasing accuracy.

The GRU-GWO technique uses iterations = 60 and wolves = 30 for the NSL-KDD. For CICIDS2017, it uses iterations = 80 and wolves = 30, and in case of TON_IoT iterations = 70 and wolves = 30. The K-Means-Firefly technique uses iterations = 40 and population = 30 for NSL-KDD. For the CICIDS2017 dataset, it uses iterations = 70 and population = 30, and in the case of the TON_IoT dataset iterations = 70 and population = 30. The LR-ABC technique uses iterations = 40 and population = 30 for NSL-KDD. For the CICIDS2017 dataset, it uses iterations = 50 and population = 30, and in the case of the TON_IoT dataset iterations = 50 and population = 30. Harris Hawks-MLP technique uses iterations = 30 with swarm size = 30 for NSL-KDD. For the CICIDS2017 dataset, it uses iterations = 50 and swarm size = 30, and in the case of the TON_IoT dataset iterations = 50 and swarm size = 30.

Table 6. Feature selection using BGGO.

Dataset	Original Features	Selected Features	Reduction (%)
NSL-KDD	41	26	36.6
CICIDS2017	78	35	55.1
TON_IoT	42	28	33.3

Table 7. Performance comparison of different models across NSL-KDD, CICIDS2017, and TON_IoT datasets.

Dataset	Model	Precision (%)	Recall (%)	F1-Score (%)	Accuracy (%)
NSL-KDD	SVM	94.32	93.68	93.99	94.00
	KNN	91.82	91.18	91.47	91.50
	LSTM	93.44	92.96	93.17	93.20
	CNN	92.54	92.66	92.58	92.60
	TabNet	85.92	85.42	85.64	85.67
	MLP	94.46	93.94	94.11	94.20
	MobileNet v1	96.34	96.06	96.17	96.20
	CID System	97.61	97.60	97.60	97.30
CICIDS2017	SVM	92.78	92.22	92.47	92.50
	KNN	91.88	91.32	91.57	91.60
	LSTM	94.48	93.92	94.16	94.20
	CNN	94.16	93.84	93.96	94.00
	TabNet	85.22	84.78	84.96	85.00
	MLP	94.32	93.88	94.09	94.10
	MobileNet v1	95.66	94.94	95.26	95.30
	CID System	94.90	96.63	95.76	96.89
TON_IoT	SVM	94.88	94.52	94.66	94.70
	KNN	94.22	93.78	93.96	94.00
	LSTM	95.28	94.72	94.96	95.00
	CNN	93.52	93.88	93.67	93.70
	TabNet	84.96	84.28	84.58	84.62
	MLP	93.54	92.86	93.07	93.20
	MobileNet v1	95.96	95.64	95.76	95.80
	CID System	99.43	95.99	97.67	96.30

The best-performing results are highlighted in bold.

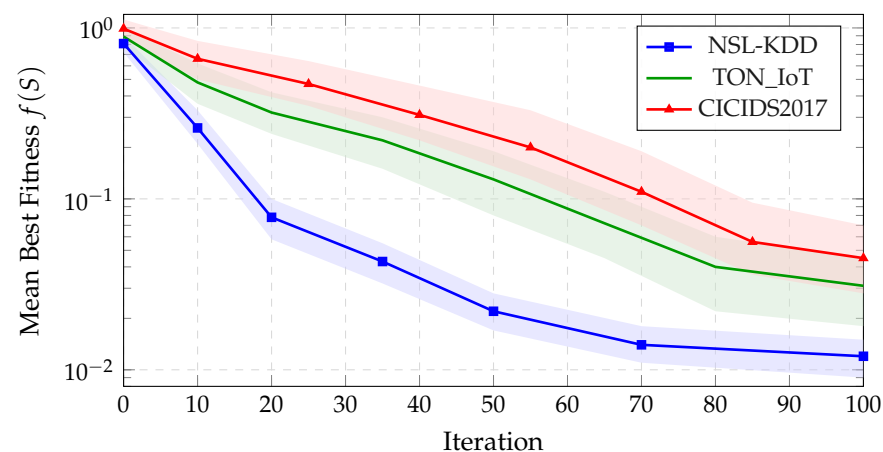
**Figure 3.** Convergence performance of the BGGO-based feature selection framework across datasets.

Table 8 provides the performance statistics of the CID system across the datasets used. To evaluate the model with unseen test set, the complete experiment was repeated over five independent random seeds, and the final reported results correspond to the mean performance across these runs. Table 9 shows the confusion matrix for the CID system for different datasets. The confusion matrices were regenerated using the 20% test sets with their natural class distributions. In Tables 10–12, different IDS techniques are compared. All the optimizations were executed for a maximum of 100 iterations. In the tables, optimization techniques such as Grey Wolf Optimization, Firefly Optimization, Artificial Bee Colony, and Harris Hawks Optimization have been explored for feature selection based on the existing literature. The proposed BGGO-based approach demonstrates superior performance in the considered scenarios. The K-Means with the Firefly method achieves a comparatively low accuracy across the datasets. This reduced performance can be attributed to the limitations of K-Means in modeling non-linear network traffic patterns, along with the restricted capability of the Firefly algorithm to efficiently optimize feature subsets in high-

dimensional search spaces. Figure 4 shows the accuracy comparison of models using different datasets.

Table 8. Performance statistics of the CID system over five independent random seeds.

Dataset	Mean Accuracy (%)	Std. Dev.	Variance
NSL-KDD	97.30	0.22	0.048
CICIDS2017	96.89	0.27	0.073
TON_IoT	96.30	0.24	0.058

Table 9. Confusion matrices of the CID System across different datasets.

(a) NSL-KDD		
	Pred. Attack	Pred. Normal
Actual Attack	12,239	301
Actual Normal	300	9410
(b) CICIDS2017		
	Pred. Attack	Pred. Normal
Actual Attack	48,675	1696
Actual Normal	2614	85,556
(c) TON_IoT		
	Pred. Attack	Pred. Normal
Actual Attack	15,577	650
Actual Normal	90	3683

Table 10. Accuracy comparison of IDS techniques using NSL-KDD dataset.

Technique	Model	Accuracy (%)
[21]	GRU + GWO	93.00
[44]	K-Means + Firefly	71.00
[41]	Logistic Regression + ABC	89.23
[45]	Harris Hawks + MLP	94.00
CID system	BGGO + MobileNet v1	97.30

The best-performing results are highlighted in bold.

Table 11. Accuracy comparison of IDS techniques using CICIDS2017 dataset.

Paper	Model	Accuracy (%)
[21]	GRU + GWO	93.20
[44]	K-Means + Firefly	79.00
[41]	Logistic Regression + ABC	90.00
[45]	Harris Hawks + MLP	94.02
CID system	BGGO + MobileNet v1	96.89

The best-performing results are highlighted in bold.

Table 12. Accuracy comparison of IDS techniques using TON_IoT dataset.

Paper	Model	Accuracy (%)
[21]	GRU + GWO	93.00
[44]	K-Means + Firefly	74.22
[41]	Logistic Regression + ABC	84.19
[45]	Harris Hawks + MLP	93.2
CID system	BGGO + MobileNet v1	96.30

The best-performing results are highlighted in bold.

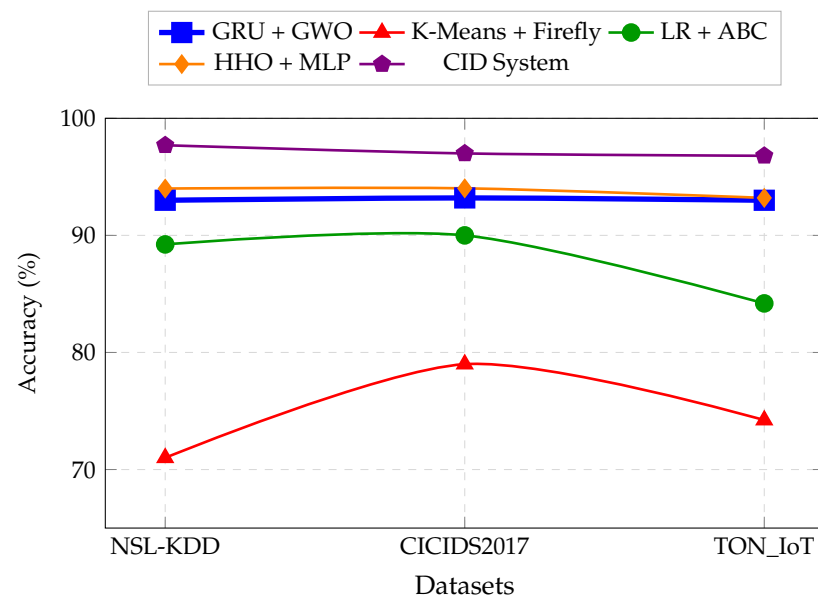


Figure 4. Accuracy comparison of different IDS models.

Table 13 shows the performance of CID system for common attack in NSL-KDD, CICIDS2017 and TON_IoT. The CICIDS2017 and TON_IoT datasets encompass multiple attack categories, including distributed denial-of-service (DDoS) attacks, probing or scanning activities and denial-of-service. The absence of DDoS attacks in NSL-KDD limits its ability to assess detection performance for modern distributed attack scenarios, whereas CICIDS2017 and TON_IoT provide a more comprehensive and realistic testing environment. Therefore, evaluating the CID system across all three datasets ensures that its effectiveness is validated under both traditional and contemporary threat conditions. The CID system used a binary classifier for intrusion detection. The per-attack-type accuracy values are derived using a standard evaluation approach. For each attack category, e.g., DoS, DDoS, Probe, the test dataset is filtered based on ground truth labels. Samples belonging to the selected attack type are considered as the positive class, while normal samples form the negative class. The model's binary outputs are then evaluated on this subset, and accuracy is computed accordingly. Other attack categories are excluded during this process. The proposed approach achieved arithmetic mean detection accuracies of 96.93%, 96.82%, and 96.58% for DoS, DDoS, and probe attacks, respectively, computed across the datasets containing those attack categories, as summarized in Table 13.

Table 13. Classification accuracy evaluation of common attack categories across datasets.

Dataset	Attack Type	DoS Labels	Probe/Scanning Labels	DoS (%)	DDoS (%)	Probe (%)
NSL-KDD	DoS, Probe	smurf, apache2, udpstorm, land, processtable, teardrop, back, mailbomb, pod, neptune	nmap, satan, portsweep, ipsweep, saint, mscan	97.42	–	97.18
CICIDS2017	DoS, DDoS, Probe	Heartbleed, DoS slowloris, DoS Hulk, DoS Slowhttptest, DoS GoldenEye	PortScan, Bot	96.95	97.08	96.63
TON_IoT	DoS, DDoS, Probe	dos	scanning	96.41	96.56	95.94

Table 14 presents the FAR of the CID system. Figure 5 presents the false alarm rate (FAR) comparison of different intrusion detection models across multiple datasets. A detailed observation of the figure shows that the CID system consistently achieves the lowest FAR among all compared methods for each dataset. In particular, the reduction in FAR is more pronounced in datasets with higher feature dimensionality and heterogeneous

traffic patterns, where traditional models tend to generate more false positives. For example, while baseline models exhibit noticeable fluctuations in FAR across datasets, the CID system maintains a relatively stable and minimal false alarm rate. This indicates that the model is less sensitive to dataset variations and generalizes effectively across different network environments. The consistency observed in Figure 5 directly supports the robustness of the proposed approach. The improved FAR performance can be attributed to the integration of BGGO-based feature selection and MobileNet v1 classification. BGGO effectively removes redundant and irrelevant features that typically contribute to incorrect classifications, thereby reducing the likelihood of false alarms. At the same time, MobileNet v1 leverages the refined feature space to learn discriminative patterns, enabling more accurate separation between normal and attack traffic. The proposed model has approximately 5.3 million trainable parameters, and the size of the model is 21 MB. A paired *t*-test was employed to evaluate the statistical significance of the performance differences between the CID system and the two baseline models under identical experimental conditions. Since all models were evaluated on the same datasets using identical train–test partitions and random seed configurations, the observations are considered dependent, making the paired *t*-test appropriate for this analysis. The null hypothesis (H_0) in Equation (16) assumes that there is no significant difference between the mean performances of the compared models, whereas the alternative hypothesis (H_1) in Equation (17) assumes that a significant difference exists. Mathematically, the hypotheses are expressed as

$$H_0 : \mu_d = 0 \quad (16)$$

$$H_1 : \mu_d \neq 0 \quad (17)$$

where μ_d represents the mean difference between paired observations. The paired *t*-test statistic is computed using Equation (18):

$$t = \frac{\bar{d}}{s_d / \sqrt{n}} \quad (18)$$

where $\bar{d}$ denotes the mean of the paired differences, s_d represents the standard deviation of the paired differences, and n is the total number of paired observations. Statistical significance is determined using a significance threshold of $p < 0.05$.

Table 14. False alarm rate of the CID system.

Dataset	FAR (%)
NSL-KDD	3.09
CICIDS2017	2.96
TON_IoT	2.38

The paired *t*-test results presented in Table 15 demonstrate that the CID system consistently outperforms the baseline models, namely Harris Hawks + MLP and GRU + GWO, across all evaluated benchmark datasets. The statistical analysis was conducted using F1-score values obtained from five independent experimental runs under identical dataset partitions and random seed configurations. For all comparisons, the computed *t*-values were substantially higher than the corresponding critical *t*-value of 2.7764, while the obtained *p*-values remained significantly lower than the threshold value of 0.05, confirming that the observed performance improvements are statistically significant.

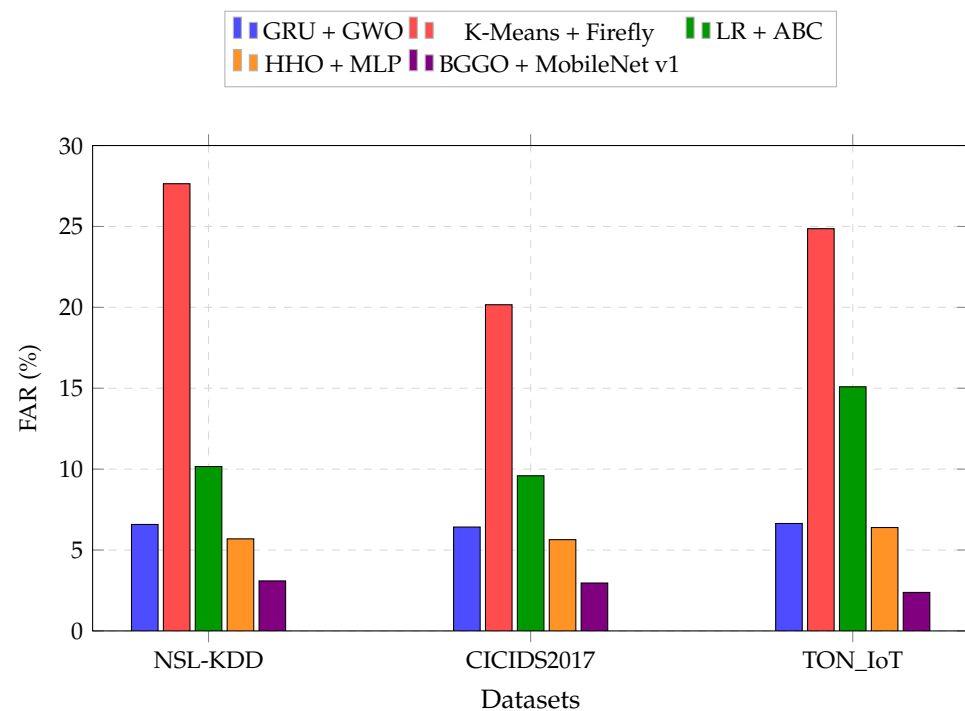


Figure 5. False alarm rate comparison of IDS techniques across datasets.

Table 15. Paired *t*-test results using F1-score values from five independent runs.

Dataset	Comparison	t-Value	Critical t-Value	p-Value
NSL-KDD	CID vs GRU + GWO	17.84	2.7764	<0.001
	CID vs Harris Hawks + MLP	11.62	2.7764	<0.001
CICIDS2017	CID vs GRU + GWO	13.47	2.7764	<0.001
	CID vs Harris Hawks + MLP	9.28	2.7764	0.0007
TON_IoT	CID vs GRU + GWO	15.91	2.7764	<0.001
	CID vs Harris Hawks + MLP	10.84	2.7764	0.0004

6. Conclusions and Future Work

The study proposes an intrusion detection framework based on deep learning, known as the CID system, which demonstrates potential suitability for IoT deployment. The proposed method uses MobileNet v1 as the main classification model to analyze and categorize network traffic. It uses Binary Greylag Goose Optimization to choose the best features. BGGO helps the classifier focus on the most useful features by choosing the best subset of features. This improves detection accuracy and reduces computational overhead. The BGGO technique uses leader-guided search, neighborhood influence, and random variation. These parameters are used to speed up convergence and lower the chance of premature convergence. MobileNet v1's compact architecture significantly lowers computational complexity. This makes it suitable for deep learning-based Intrusion Detection Systems. The CID system consistently outperforms other models across the three benchmark datasets. From a practical perspective, the low FAR delivered by the CID system is crucial. By maintaining a low FAR across the datasets, the CID system significantly reduces the operational noise. It ensures the critical security warnings are prioritized. Future research will focus on deploying the advanced BGGO algorithm. The simplified BGGO used in this study provides an effective and low-complexity method for binary feature selection, but the lack of organized V-formation dynamics may hamper the cooperative

search behavior. Advanced BGGO can be implemented in different MobileNet versions to compare the resource utilization. In the real world, it is essential to deploy robust intrusion detection directly on resource-constrained devices and edge devices. Deep learning models give highly accurate outputs but have large sizes. They require significant computational resources. The CID system can be deployed in an IoT network. But for training the model in an IoT environment, techniques such as weight pruning and quantization can be used. Quantization and pruning can reduce latency and power consumption. These techniques compress the model, resulting in reduced training time and memory. IoT environments are dynamic in nature. Currently, the CID system relies on static training datasets. Future research can integrate adaptive online learning for dynamically changing IoT environments. Drift detection mechanisms can also be incorporated to continuously monitor variations in traffic distributions and classification performance. When significant drift is detected, the system can trigger periodic retraining or incremental model updates using newly observed traffic samples to maintain detection accuracy without requiring complete retraining from scratch. Although this work considers a binary classification setting, future research can explore multi-class intrusion detection to distinguish between different attack categories and provide more fine-grained threat analysis. The robustness of the model against adversarial conditions remains an important concern. Future studies can investigate the resilience of the proposed BGGO-based feature selection and deep learning framework under adversarial attacks and noisy environments to ensure reliability in real-world deployments. The current implementation has been evaluated in a simulation environment. The proposed framework exhibits compact characteristics on standard computing hardware; however, in the future, the experiments can be performed on constrained edge device platforms such as Raspberry Pi. The proposed approach can be validated on more recent datasets, such as CIC-IoT-2023 and Edge-IIoTset, to further demonstrate its generalization capability across evolving cyber-attack landscapes.

Author Contributions: S.D. contributed to the conceptualization of the study, methodology development, data curation, and preparation of the original draft. A.M. contributed to conceptualization, reviewing, and editing of the manuscript. S.R. contributed to manuscript editing. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding

Data Availability Statement: The datasets used are publicly available datasets and are cited in the paper.

Acknowledgments: The authors acknowledge the facilities and support provided by Mobile Computing Lab, Tripura University and Operating System Lab, Tripura Institute of Technology during the completion of this work.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Chowdhury, R.; Sen, S.; Goswami, A.; Purkait, S.; Saha, B. An implementation of bi-phase network intrusion detection system by using real-time traffic analysis. *Expert Syst. Appl.* **2023**, *224*, 119831. [CrossRef]
2. FBI Internet Crime Complaint Center (IC3). 2024 Internet Crime Report. 2024. Available online: https://www.ic3.gov/AnnualReport/Reports/2024_IC3Report.pdf (accessed on 28 May 2026).
3. Abdullah, M.; Nawaz, M.M.; Saleem, B.; Zahra, M.; Ashfaq, E.b.; Muhammad, Z. Evolution Cybercrime—Key Trends, Cybersecurity Threats, and Mitigation Strategies from Historical Data. *Analytics* **2025**, *4*, 25. [CrossRef]
4. Farhan, M.; Waheed Ud Din, H.; Ullah, S.; Hussain, M.S.; Khan, M.A.; Mazhar, T.; Khattak, U.F.; Jaghdam, I.H. Network-based intrusion detection using deep learning technique. *Sci. Rep.* **2025**, *15*, 25550. [CrossRef] [PubMed]

5. Shyaa, M.A.; Ibrahim, N.F.; Zainol, Z.B.; Abdullah, R.; Anbar, M.; Alzubaidi, L. IGPC-MSOS: A Knowledge-Preserving Transfer Learning Framework with Dynamic Mode-Switching for Handling Concept Drift in Network Intrusion Detection Systems. *Knowl.-Based Syst.* **2026**, *337*, 115361. [\[CrossRef\]](#)
6. Diro, A.A.; Chilamkurti, N. Distributed attack detection scheme using deep learning approach for Internet of Things. *Future Gener. Comput. Syst.* **2018**, *82*, 761–768. [\[CrossRef\]](#)
7. Pawana, I.W.A.J.; Abella, V.; Lastre, J.K.; Ko, Y.; You, I. Enhancing Roaming Security in Cloud-Native 5G Core Network through Deep Learning-Based Intrusion Detection System. *Comput. Model. Eng. Sci.* **2025**, *145*, 2733. [\[CrossRef\]](#)
8. Huang, A.; Yan, J.; Fan, X.; Zhou, H. Multi-Scenario Cloud-Edge Collaborative DDoS Detection in LLM-Enabled AIoT. *IEEE Trans. Netw. Sci. Eng.* **2025**, *13*, 3790–3809. [\[CrossRef\]](#)
9. Wakili, A.; Bakkali, S. ZeroDefense: An adaptive hybrid fusion-based intrusion detection system for zero-day threat detection in IoT networks. *J. Electron. Sci. Technol.* **2026**, *24*, 100345. [\[CrossRef\]](#)
10. Ahmim, A.; Maazouzi, F.; Ahmim, M.; Namane, S.; Dhaou, I.B. Distributed denial of service attack detection for the Internet of Things using hybrid deep learning model. *IEEE Access* **2023**, *11*, 119862–119875. [\[CrossRef\]](#)
11. Wang, Y.; Qin, G.; Zou, M.; Liang, Y.; Wang, G.; Wang, K.; Feng, Y.; Zhang, Z. A lightweight intrusion detection system for internet of vehicles based on transfer learning and MobileNetV2 with hyper-parameter optimization. *Multimed. Tools Appl.* **2024**, *83*, 22347–22369. [\[CrossRef\]](#)
12. Wang, B.; Yu, L.; Zhang, B. AL-MobileNet: A novel model for 2D gesture recognition in intelligent cockpit based on multi-modal data. *Artif. Intell. Rev.* **2024**, *57*, 282. [\[CrossRef\]](#)
13. Huang, K.; Xian, R.; Xian, M.; Wang, H.; Ni, L. A comprehensive intrusion detection method for the internet of vehicles based on federated learning architecture. *Comput. Secur.* **2024**, *147*, 104067. [\[CrossRef\]](#)
14. Grandhi, A.; Singh, S.K. Interrelated dynamic biased feature selection and classification model using enhanced gorilla troops optimizer for intrusion detection. *Alex. Eng. J.* **2025**, *114*, 312–330. [\[CrossRef\]](#)
15. Vinod, D.; Prasad, M. Enhancing Network Security: A Novel Intrusion Detection System Utilizing Dual-Optimization Techniques for Feature Selection and Classification. *Comput. Netw.* **2026**, *277*, 112021. [\[CrossRef\]](#)
16. Jayasankar, T.; Kiruba Buri, R.; Maheswaravenkatesh, P. Intrusion detection system using metaheuristic fireworks optimization based feature selection with deep learning on Internet of Things environment. *J. Forecast.* **2024**, *43*, 415–428. [\[CrossRef\]](#)
17. Li, J.; Othman, M.S.; Ying, X.; Hassan, D.S.; Chen, H.; Yusuf, L.M. Adaptive NetFlow IIoT Intrusion Detection With Deep Transfer Learning, Genetic Optimization, and Ensemble Methods for Network Management. *IEEE Trans. Netw. Serv. Manag.* **2025**, *23*, 681–698. [\[CrossRef\]](#)
18. El-Kenawy, E.S.M.; Khodadadi, N.; Mirjalili, S.; Abdelhamid, A.A.; Eid, M.M.; Ibrahim, A. Greylag goose optimization: Nature-inspired optimization algorithm. *Expert Syst. Appl.* **2024**, *238*, 122147. [\[CrossRef\]](#)
19. Ghasemi, M.; kadkhoda Mohammadi, S.; Zare, M.; Mirjalili, S.; Gil, M.; Hemmati, R. A new firefly algorithm with improved global exploration and convergence with application to engineering optimization. *Decis. Anal. J.* **2022**, *5*, 100125. [\[CrossRef\]](#)
20. Katipoğlu, O.M.; Mohammadi, B.; Keblouti, M. Bee-inspired insights: Unleashing the potential of artificial bee colony optimized hybrid neural networks for enhanced groundwater level time series prediction. *Environ. Monit. Assess.* **2024**, *196*, 724. [\[CrossRef\]](#) [\[PubMed\]](#)
21. Elsaid, S.A.; Shehab, E.; Mattar, A.M.; Azar, A.T.; Hameed, I.A. Hybrid intrusion detection models based on GWO optimized deep learning. *Discov. Appl. Sci.* **2024**, *6*, 531. [\[CrossRef\]](#)
22. Alzaqebah, A.; Aljarah, I.; Al-Kadi, O. A hierarchical intrusion detection system based on extreme learning machine and nature-inspired optimization. *Comput. Secur.* **2023**, *124*, 102957. [\[CrossRef\]](#)
23. Saravanan, S.; Kumar, R.S.; Balakumar, P.; Prabakaran, N. Optimal power harvesting under partial shading: Binary Greylag Goose optimization for reconfiguration and Machine learning-Based fault diagnosis in solar PV arrays. *Energy Convers. Manag.* **2025**, *333*, 119808. [\[CrossRef\]](#)
24. Elkenawy, E.S.M.; Alhussan, A.A.; Khafaga, D.S.; Tarek, Z.; Elshewey, A.M. Greylag goose optimization and multilayer perceptron for enhancing lung cancer classification. *Sci. Rep.* **2024**, *14*, 23784. [\[CrossRef\]](#) [\[PubMed\]](#)
25. Khosrowshahi, H.N.; Aghdasi, H.S.; Salehpour, P. A refined Greylag Goose optimization method for effective IoT service allocation in edge computing systems. *Sci. Rep.* **2025**, *15*, 15729. [\[CrossRef\]](#) [\[PubMed\]](#)
26. Wang, L.; Yao, Y.; Yang, Y.; Zang, Z.; Zhang, X.; Zhang, Y.; Yu, Z. Novel Greylag Goose Optimization Algorithm with Evolutionary Game Theory (EGGO). *Biomimetics* **2025**, *10*, 545. [\[CrossRef\]](#) [\[PubMed\]](#)
27. Ghorbal, A.B.; Grine, A.; Eid, M.M.; El-Kenawy, E.S.M. Greylag Goose Optimization and Deep Learning-Based Electrohysterogram Signal Analysis for Preterm Birth Risk Prediction. *Comput. Model. Eng. Sci. (CMES)* **2025**, *144*, 2001–2028. [\[CrossRef\]](#)
28. Samunnisa, K.; Kumar, G.S.V.; Madhavi, K. Intrusion detection system in distributed cloud computing: Hybrid clustering and classification methods. *Meas. Sens.* **2023**, *25*, 100612. [\[CrossRef\]](#)
29. Al-Omari, M.; Rawashdeh, M.; Qutaishat, F.; Alshirah, M.; Ababneh, N. An intelligent tree-based intrusion detection model for cyber security. *J. Netw. Syst. Manag.* **2021**, *29*, 20. [\[CrossRef\]](#)

30. Thockchom, N.; Singh, M.M.; Nandi, U. A novel ensemble learning-based model for network intrusion detection. *Complex Intell. Syst.* **2023**, *9*, 5693–5714. [\[CrossRef\]](#)
31. Sarkar, A.; Sharma, H.S.; Singh, M.M. A supervised machine learning-based solution for efficient network intrusion detection using ensemble learning based on hyperparameter optimization. *Int. J. Inf. Technol.* **2023**, *15*, 423–434. [\[CrossRef\]](#)
32. Sedhuramalingam, K.; Saravanakumar, N. A novel optimal deep learning approach for designing intrusion detection system in wireless sensor networks. *Egypt. Inform. J.* **2024**, *27*, 100522. [\[CrossRef\]](#)
33. Ghadami, R. An intrusion detection system in the Internet of Things with deep learning and an improved arithmetic optimization algorithm (AOA) and sine cosine algorithm (SCA). *Sci. Rep.* **2025**, *15*, 38156. [\[CrossRef\]](#) [\[PubMed\]](#)
34. Zhou, S.; Liu, C.; Ye, D.; Zhu, T.; Zhou, W.; Yu, P.S. Adversarial attacks and defenses in deep learning: From a perspective of cybersecurity. *ACM Comput. Surv.* **2022**, *55*, 1–39. [\[CrossRef\]](#)
35. Guo, R.; Chen, Q.; Liu, H.; Wang, W. Adversarial robustness enhancement for deep learning-based soft sensors: An adversarial training strategy using historical gradients and domain adaptation. *Sensors* **2024**, *24*, 3909. [\[CrossRef\]](#) [\[PubMed\]](#)
36. Che, L.; Wu, C.; Hou, Y. Large Language Model Text Adversarial Defense Method Based on Disturbance Detection and Error Correction. *Electronics* **2025**, *14*, 2267. [\[CrossRef\]](#)
37. Guo, R.; Li, A.; Liu, H. An Adversarial Attack Detection Method Based on Bidirectional Consistency Discrimination for Deep Learning-Based Soft Sensors. In *Proceedings of the 2025 CAA Symposium on Fault Detection, Supervision, and Safety for Technical Processes (SAFEPROCESS)*; IEEE: Piscataway, NJ, USA, 2025; pp. 1–6. [\[CrossRef\]](#)
38. Nasir, M.H.; Khan, S.A.; Khan, M.M.; Fatima, M. Swarm intelligence inspired intrusion detection systems-a systematic literature review. *Comput. Netw.* **2022**, *205*, 108708. [\[CrossRef\]](#)
39. Reddy, D.K.K.; Nayak, J.; Behera, H.; Shanmuganathan, V.; Viriyasitavat, W.; Dhiman, G. A systematic literature review on swarm intelligence based intrusion detection system: Past, present and future. *Arch. Comput. Methods Eng.* **2024**, *31*, 2717–2784. [\[CrossRef\]](#)
40. Donkol, A.A.E.B.; Hafez, A.G.; Hussein, A.I.; Mabrook, M.M. Optimization of intrusion detection using likely point PSO and enhanced LSTM-RNN hybrid technique in communication networks. *IEEE Access* **2023**, *11*, 9469–9482. [\[CrossRef\]](#)
41. Kolukisa, B.; Dedetürk, B.K.; Hacilar, H.; Gungor, V.C. An efficient network intrusion detection approach based on logistic regression model and parallel artificial bee colony algorithm. *Comput. Stand. Interfaces* **2024**, *89*, 103808. [\[CrossRef\]](#)
42. Srivastava, A.; Sinha, D. PSO-ACO-based bi-phase lightweight intrusion detection system combined with GA optimized ensemble classifiers. *Clust. Comput.* **2024**, *27*, 14835–14890. [\[CrossRef\]](#)
43. Bakro, M.; Kumar, R.R.; Husain, M.; Ashraf, Z.; Ali, A.; Yaqoob, S.I.; Ahmed, M.N.; Parveen, N. Building a cloud-IDS by hybrid bio-inspired feature selection algorithms along with random forest model. *IEEE Access* **2024**, *12*, 8846–8874. [\[CrossRef\]](#)
44. Kaur, A.; Pal, S.K.; Singh, A.P. Hybridization of K-means and firefly algorithm for intrusion detection system. *Int. J. Syst. Assur. Eng. Manag.* **2018**, *9*, 901–910. [\[CrossRef\]](#)
45. Alazab, M.; Khurma, R.A.; Castillo, P.A.; Abu-Salih, B.; Martín, A.; Camacho, D. An effective networks intrusion detection approach based on hybrid Harris Hawks and multi-layer perceptron. *Egypt. Inform. J.* **2024**, *25*, 100423. [\[CrossRef\]](#)
46. Yesodha, K.; Krishnamurthy, M.; Selvi, M.; Kannan, A. Intrusion detection system extended CNN and artificial bee colony optimization in wireless sensor networks. *Peer-to-Peer Netw. Appl.* **2024**, *17*, 1237–1262. [\[CrossRef\]](#)
47. Karthikeyan, M.; Brindha, R.; Vianny, M.M.; Vaitheeshwaran, V.; Bachute, M.; Mishra, S.; Dash, B.B. Integration of metaheuristic based feature selection with ensemble representation learning models for privacy aware cyberattack detection in IoT environments. *Sci. Rep.* **2025**, *15*, 22887. [\[CrossRef\]](#) [\[PubMed\]](#)
48. Chen, Y.; Guo, Y.; Gao, Y.; Liu, B. A novel lightweight deep learning framework using enhanced pelican optimization for efficient cyberattack detection in the Internet of Things environments. *J. Eng. Appl. Sci.* **2025**, *72*, 69. [\[CrossRef\]](#)
49. Al-Shurbaji, T.; Anbar, M.; Manickam, S.; Al-Amiedy, T.A.; Mukhaini, G.A.; Hashim, H.; Farsi, M.; Atlam, E.S. BoT-EnsIDS: Approach for detecting IoT Botnet attacks leveraging bio-inspired based ensemble feature selection and hybrid deep learning model. *Alex. Eng. J.* **2025**, *129*, 744–767. [\[CrossRef\]](#)
50. Jabeur, N. FireBoost: A new Bio-Inspired Approach for Feature selection based on Firefly Algorithm and Optimized XGBoost. *Intell. Syst. Appl.* **2025**, *29*, 200613. [\[CrossRef\]](#)
51. Dharmalingam, M.; Subramaniam, K.; M, A.; Nandhagopal, N. Diverse attack detection in IoT using hybrid deep convolutional with capsule auto encoder for intrusion detection model. *J. Parallel Distrib. Comput.* **2025**, *208*, 105190. [\[CrossRef\]](#)
52. Misrak, S.F.; Melaku, H.M. Lightweight intrusion detection system for IoT with improved feature engineering and advanced dynamic quantization. *Discov. Internet Things* **2025**, *5*, 97. [\[CrossRef\]](#)
53. Sun, Y.; Wang, Z. Intrusion detection in IoT and wireless networks using image-based neural network classification. *Appl. Soft Comput.* **2025**, *177*, 113236. [\[CrossRef\]](#)
54. Sharma, K.P.; Nagpal, T.; Vora, T.; Yadav, A.; Abdullah, M.I.; Jayaprakash, B.; Kashyap, A.; Sridevi, G.; Bhowmik, A.; Bukate, B.B. Interpretable intrusion detection for IoT environments using a self-attention-based explainable AI framework. *Sci. Rep.* **2025**, *15*, 39937. [\[CrossRef\]](#) [\[PubMed\]](#)

55. Rezvan, M.R.; Sorkhi, A.G.; Pirgazi, J.; Kallehbasti, M.M.P. AdvanceSplice: Integrating N-gram one-hot encoding and ensemble modeling for enhanced accuracy. *Biomed. Signal Process. Control* **2024**, *92*, 106017. [[CrossRef](#)]
56. Kim, Y.S.; Kim, M.K.; Fu, N.; Liu, J.; Wang, J.; Srebric, J. Investigating the impact of data normalization methods on predicting electricity consumption in a building using different artificial neural network models. *Sustain. Cities Soc.* **2025**, *118*, 105570. [[CrossRef](#)]
57. Elreedy, D.; Atiya, A.F. A comprehensive analysis of synthetic minority oversampling technique (SMOTE) for handling class imbalance. *Inf. Sci.* **2019**, *505*, 32–64. [[CrossRef](#)]
58. Alrayes, F.S.; Zakariah, M.; Amin, S.U.; Khan, Z.I.; Alqurni, J.S. CNN Channel Attention Intrusion Detection System Using NSL-KDD Dataset. *Comput. Mater. Contin.* **2024**, *79*, 4319–4347. [[CrossRef](#)]
59. Sharafaldin, I.; Lashkari, A.H.; Ghorbani, A.A. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In *International Conference on Information Systems Security and Privacy*; SciTePress (Science and Technology Publications): Setúbal, Portugal, 2018; Volume 1, pp. 108–116. [[CrossRef](#)]
60. Alsaedi, A.; Moustafa, N.; Tari, Z.; Mahmood, A.; Anwar, A. TON_IoT telemetry dataset: A new generation dataset of IoT and IIoT for data-driven intrusion detection systems. *IEEE Access* **2020**, *8*, 165130–165150. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.