Int. J. Advance Soft Compu. Appl, Vol. 18, No. 1, March 2026
Print ISSN: 2710-1274, Online ISSN: 2074-8523
Copyright © Al-Zaytoonah University of Jordan (ZUJ)

Enhanced Intrusion Detection Systems Dataset Synthesis Using Conditional Generative Adversarial Networks with the Adaptive Whale Optimization Algorithm

Ateka H. Alsheyab[1] and Nameer N. El-Emam[2]

[1]Department of Computer Science, College of Information Technology, Amman Arab University, Amman, 11953, Jordan
e-mail: atekaalsheyab22@gmail.com

[2]Department of Computer Science, College of Information Technology, Amman Arab University, Amman, 11953, Jordan
e-mail: n.emam@aau.edu.jo

Abstract

Modern networks, especially in the IoT era, require intrusion detection systems (IDSs) to ensure security and integrity. The growing diversity and data-driven nature of cyberattacks necessitate the generation of high-quality synthetic attack data for training robust detection models. In this domain, Generative Adversarial Networks (GANs) have proven to be a versatile and powerful tool. However, GAN training is unstable, as it often fails to converge and produces suboptimal outputs. To address these challenges, this study proposes a Conditional GAN (CGAN) enhanced with the Adaptive Whale Optimization Algorithm (AWOA). Relax-Alpha is a feedback mechanism that adjusts the generator's updates based on trends in discriminator accuracy. It aims to provide dynamic training control, stabilizing training, and increasing the realism and diversity of synthetic samples. Several adaptation strategies were evaluated, including discrete and continuous alpha adjustments. The model was tested on the CIC IoT 2023 dataset through 5-fold cross-validation. The experiments show strong classification and generation quality; CGAN-AWOA achieved 98.59% accuracy.

Keywords: *Conditional Generative Adversarial Networks, GAN Training Stability, Intrusion Detection Systems, Relax-Alpha, Whale Optimization Algorithm.*

1 Introduction

The rise of the Internet of Things (IoT) has transformed the digital ecosystem by connecting billions of smart devices. According to the IEEE Internet of Things Journal (2023), IoT-connected devices surpassed 14 billion in 2023 and are expected to reach 29 billion by 2030. However, increased connectivity also introduces a range of security vulnerabilities, like Distributed Denial of Service (DDoS) attacks, malware spread, and man-in-the-middle (MITM) attacks [1]. Resource limitations also pose a challenge. Many

Received 5 June 2025; Accepted 15 January 2026

IoT devices have limited processing power, memory, and battery life, which makes it difficult to deploy complex security mechanisms on-device [2]. Intrusion Detection Systems (IDS) need robust, efficient, and scalable systems.

Traditional IDS methods, such as signature-based IDS or anomaly-based IDS, struggle with the dynamic nature of IoT and a lack of labeled data [3]. To address this, the Canadian Institute for Cybersecurity (CIC) introduced the CICIoT2023 dataset. This dataset provides network traffic from a wide range of IoT devices. There are 33 attack types and realistic IoT scenarios. Although CICIoT2023 represents a significant step forward, as it is presented with a large, diverse, and real-time dataset, the researchers emphasize that continuous updates are still necessary due to the emergence of new attack strategies and the evolving IoT threat landscape [4]. Researchers have increasingly explored data augmentation techniques to address these data challenges to expand and balance IoT attack datasets. Using Generative Adversarial Networks (GANs) is a promising approach, as was proposed by Goodfellow et al. (2020b) [5]. GANs' two neural networks in a GAN-based competitive learning process; a generator produces synthetic data, and a discriminator distinguishes real and synthetic samples.

CGANs (Conditional GANs) are further developed from this architecture by incorporating label information, enabling the generation of synthetic data for specific classes or conditions. With CGANs, researchers can generate realistic synthetic attack data that supplements existing datasets, thereby increasing the generalizability and robustness of IDS models [6]. However, training GANs (and therefore CGANs) is difficult, despite their potential. Training GANs is challenging due to instability and mode collapse [7], particularly in complex loss landscapes that traditional optimizers, such as Adam, struggle to handle reliably [6].

The Whale Optimization Algorithm (WOA) is a newly developed nature-inspired optimization technique introduced by Seyedali Mirjalili in 2016. WOA is efficient at both global exploration to find the best solution and local exploitation to refine it. Additionally, it can be applied to more complex optimization tasks, such as hyperparameter tuning [9]. However, its standard version may suffer from premature convergence and instability in GAN training. [8].

The proposed Adaptive Whale Optimization Algorithm (AWOA), integrated with the CGAN (CGAN-AWOA) framework, introduces a relaxation factor, denoted alpha, to regulate generator weight updates during training. This factor plays a critical role in controlling the magnitude of parameter adjustments, ensuring that the optimization process remains smooth and stable. By adaptively balancing exploration and exploitation at each iteration, AWOA reduces the risk of abrupt changes in generator weights that can lead to training instability or mode collapse. In the proposed framework, AWOA is employed to optimize the generator, while the Adam optimizer continues to train the discriminator. This hybrid training strategy enhances the generator's capacity to produce high-quality synthetic attack data that more closely resembles real attack traffic compared to conventional CGAN training.

This work makes the following key contributions:

• We propose a feedback-sensitive integration of the Adaptive Whale Optimization Algorithm (AWOA) within the CGAN training loop, where the generator updates are guided directly by discriminator performance rather than using WOA as an external or offline optimizer.

• We introduce a Relax-Alpha mechanism that dynamically controls the magnitude of generator weight updates. The relaxation factor adapts based on training feedback, enabling smoother parameter updates and reducing instability during adversarial learning.

• Unlike direct meta-heuristic updates, the proposed approach combines the current AWOA-derived update with a previous update in a weighted manner, acting as a memory-aware stabilization strategy that mitigates abrupt changes and mode collapse.

• We evaluate the proposed CGAN-AWOA framework using distributional quality metrics (FID and GMSE) and cross-validation-based robustness analysis on the CICIoT2023 dataset, demonstrating improved synthetic data fidelity and stable performance compared to CGAN and CGAN-WOA baselines.

The following sections explore the proposed AWOA integrated with CGAN, outlining its theoretical foundation and experimental evaluation on the CICIoT2023 dataset. The experiments assess the method's effectiveness in generating realistic synthetic attack data and improving IoT intrusion detection systems.

2 Related Work

Vallabhaneni et al. (2024) [9] suggested using a BiGAN to address data imbalance in an IoT intrusion detection system (IDS). Their methodology uses synthetic traffic data to expand training sets, thereby enhancing accuracy, precision, and F1-score in cyberattack detection. This approach is the only one that increases the variety of datasets in IoT settings, unlike the others, which rely solely on existing data.

Kotal et al. (2024) [10] proposed KiNETGAN, a privacy-focused generative adversarial network, for synthesizing network data for distributed intrusion detection systems. It incorporates domain-specific rules and addresses class imbalance by conditionally generating a generator and using a Knowledge Graph as an independent discriminator. This will ensure compliance with privacy rules and enhance the contextual relevance and performance of the intrusion detection.

Chu and Lin (2023) [11] addressed the challenge of classifying IoT attacks in imbalanced datasets using GANs for data augmentation. They observe that their method achieves classification accuracy, precision, recall, and an F1 score higher than 90%. Their approach differs from previous studies, which typically focus on binary classification, but can also support multiclass classification. Earlier techniques outperform, achieving over 98% accuracy in experiments using the BoT-IoT and ToN-IoT datasets.

Prior studies have consistently used CTGAN to synthesize realistic IoT traffic and alleviate class imbalance in IDS training. For instance, Alabsi et al. (2023) [12] design a CTGAN-based IDS for DDoS/DoS detection in IoT networks, reporting improved detection when training on synthetic traffic. Alabdulwahab et al. (2023) [13] build a CTGAN pipeline to generate IoT datasets and enhance ML classifiers on benchmarks such as NSL-KDD and CICIDS2017, aided by feature selection. Complementarily, Wang et al. (2022) [14] propose CTTGAN, a CTGAN variant tailored to expand underrepresented traffic classes and better balance mixed (continuous/categorical) features. Collectively, these CTGAN-based approaches establish conditional tabular GANs as effective baselines for IDS-oriented tabular traffic synthesis; however, they rely on static optimization strategies and do not explicitly address adversarial training instability or dynamic generator control**.**

In a novel framework called G-IDS, Shahriar et al. (2020) [15] proposed a GAN-assisted approach to enhance intrusion detection systems (IDS) in cyberphysical systems (CPS). The aim is to mitigate difficulties in small and unfair datasets to enhance detection accuracy for various attack styles. This study differs from previous efforts, which have been limited to a specific attack or the use of GANs for adversarial purposes, as it generates generative synthetic data applicable to any Intrusion Detection System (IDS) and makes generative-based methods more robust to novel cyber threats. This study distinguishes itself by focusing carefully on violations of the limitations of emerging CPS technologies, especially data scarcity and imbalance, which are typically neglected in prior work.

Zhao et al. (2024) [22] compared multiple GAN families, including Vanilla GAN, WGAN, and CTGAN, and demonstrated that augmenting IDS training with synthetic traffic significantly improves detection performance, particularly for rare attacks. Similarly, Rahman et al. (2024) [23] showed that IDS models trained entirely on GAN-generated traffic can generalize effectively to real-world datasets with strong accuracy and robustness. While these studies confirm the effectiveness of synthetic traffic generation, they primarily focus on detection performance rather than addressing adversarial training instability or generator update dynamics.

Nadimi-Shahraki (2023) [8] introduced WOA in 2016, inspired by the hunting behavior of humpback whales, making it a prominent recent optimization technique. However, it performs well in complex search spaces, converging faster and achieving better solution quality than other swarm intelligence algorithms. The review also highlights the growing popularity of WOA and its applications across various disciplines, as evidenced by the numerous citations and the expansion of the WOA journal. In addition, the authors describe variants of WOA that address the issues of local optima trapping and low population diversity, thereby improving WOA's performance in both single- and multi-objective optimization. WOA’s advancements make it an essential optimization tool in engineering and IT applications.

Mahmood et al. (2022) [16] emphasized WOA as a powerful and efficient tool for solving complex optimization problems. Inspired by the bubble-net feeding strategy of humpback whales, WOA is recognized for its simplicity, rapid convergence, and minimal parameter requirements, making it a popular choice among researchers. The algorithm has demonstrated competitive performance across a range of fields, including data mining, machine learning, wireless sensor networks, and civil engineering. The paper highlights over eighty developed variants of WOA that enhance its capabilities, solidifying its role as a critical nature-inspired optimization technique.

Brodzicki et al. (2021) [7] demonstrated the role of WOA in optimizing hyperparameters for deep learning models. This research has shown that WOA offers better accuracy and computational efficiency than Grid Search or Random Search methods. To validate, WOA was tested on Fashion-MNIST and Reuters, achieving 89.85% and 80.60% accuracy, respectively, at the expense of increased execution time. To better handle discrete hyperparameter choices, the authors also introduce a 3-dimensional version of

WOA to address a common challenge in deep learning. This study verifies that WOA is a robust optimization tool with the potential to be applied as a hyperparameter tuning tool and encourages future research on more complex scenarios.

Recent studies show that GAN-based IDS models are much more effective at improving detection accuracy because they can generate synthetic data to address class imbalance in IoT networks. These models enhance overall performance in rare attack detection and classification. Among the novelties in the research is the application of AWOA to optimize CGAN weight generation, which is faster and yields better results than classical approaches. Such integration increases data authenticity, reduces false positives, and improves IDS performance in complex IoT environments.

3 Theoretical Aspect of the Proposed Approach

3.1 Adaptive Conditional Generative Adversarial Networks (CGANs)

Conditional Generative Adversarial Networks (CGANs) enhance traditional GANs by introducing side information, such as class labels, into both the generator and discriminator. This allows the model to generate samples conditioned on specific labels or contexts, improving relevance and control over the generated data.

Formally, the generator is defined as a function, as shown in the map at Eq. (1)

$$G{:}\{z,y\} \to \ x\_fake. \tag{1}$$

where:

- z is a random noise vector.
- y is the side information (class, label).
- x_fake is the synthetic data sample generated by G matching the condition y.

The discriminator is defined as a function, as shown in the map at Eq. (2)

$$D{:}\{x,y\} \to [0,1]. \tag{2}$$

where:

- x is an input sample (either real or generated).
- D (x, y) is the discriminator’s output, representing the probability that x is real and corresponds correctly to the condition y.

This conditioning is typically performed by directly concatenating y with the input vector or by embedding y into a higher-dimensional space before integrating it with the rest of the input [17].

Mathematically, the objective function of a CGAN modifies the original minimax loss of standard GANs to incorporate the conditional information as follows in Eq. (3), [19]

$$\min_{G}\max_{D} V\,(D,G) = E_{x\sim p_{data}(x|y)}[\log D\,(x|y)] + E_{z\sim p_z(z)}\left[\log\left(1 - D\big(G(z|y)\big)\right)\right] \tag{3}$$

where:

- Y represents conditional information (i.e., labels such as standard or attack types),
- G(z|y) represents the Generator output conditioned on z and y,

- D(x|y) is the Discriminator's probability prediction for x, given the condition y.

Fig. 1 illustrates the CGAN architecture, showing how the Generator generates conditioned data, while the Discriminator evaluates the generated data based on the data and condition y.

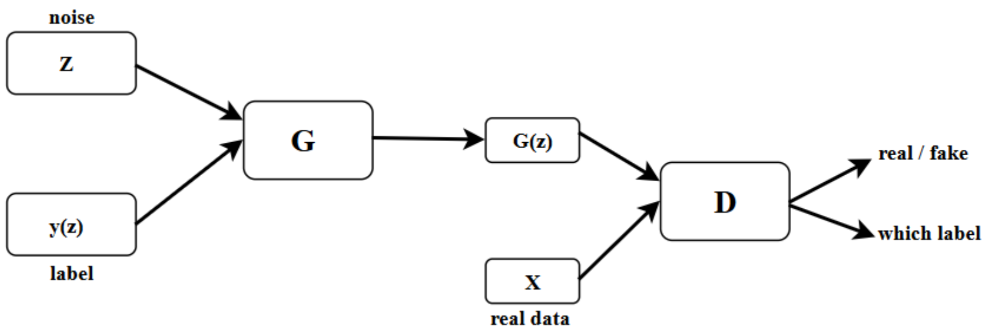


Fig 1. CGAN architecture [18]

A standard Binary Cross-Entropy (BCE) loss is used for both networks because it is suitable for binary classification, where D outputs a probability indicating whether a given sample is real or fake. The BCE loss for D is driven by correctly predicting 1 for real samples and 0 for generated (fake) samples. Conversely, G aims to conduct sampling so that the probability of D incorrectly classifying fake samples as real is maximized, thus minimizing the BCE loss from the Discriminator's perspective to fool it. The BCE loss function is formally defined as in Eq. (4):

$$L\_BCE\ (D(x), y) = -[log\ (1 - D(x))\ (1 - y) + ylog(D(x))] \tag{4}$$

where D(x) represents the Discriminator's output for input x, and y is the true label (1) for real and (0) for fake samples.

BCE provides a stable training signal for both G and D, enabling the networks to improve their performance throughout adversarial training gradually.

A CGAN is the architecture of a standard GAN enhanced with additional auxiliary information for both the generator and discriminator. The noise vector z is concatenated with the conditional input y and fed into a series of fully connected (Linear) layers with ReLU activations in the Generator so that the output G(z|, y) follows the task condition specified. The Discriminator consists of the following: concatenate input sample (real or generated) with y, leverage fully connected layers with Leaky ReLU activation functions, use Dropout layers for regularization, and output a final Sigmoid activation to give a probability score. Therefore, this architectural conditioning enables the CGAN to generate realistic and 'conditionally controlled' data.

3.2 Adaptive Whale Optimization Algorithm (AWOA)

The AWOA is based on humpback whale bubble-net hunting behavior and is widely used to solve optimization problems. Instead of trying many solutions, as in the PSO approach, WOA agents alternately explore (searching for new solutions) and exploit (refining the best-known solution), mimicking whales' hunting strategy.

3.2.1 Encircling Prey (Exploitation)

WOA starts by encircling prey, which mimics the whale's movement toward the best solution. This behavior is modeled as in Eq. (5) [19]:

$$X(t+1) = X^\wedge * \ (t) - A \times |C \times X^\wedge * \ (t) - X(t)| \tag{5}$$

Where:

- X(t) is the whale's current position,
- X^* (t) is the position of the best solution found so far.
- A and C are coefficient vectors that control movement.

This strategy ensures the whale progressively moves closer to the best solution, refining the search space. Fig. 2 (a) demonstrates how the whale contracts its encircling behavior around the optimal solution.

3.2.2 Bubble-Net Spiral Hunting (Exploration)

Whales also employ bubble-net spiral hunting to explore the solution space further. This mechanism is expressed as in Eq. (6) [19]:

$$X(t+1) = |X^\wedge * \ (t) - X(t)| \times e^\wedge(b \times l) \times cos\ (2\pi \times l) + X^\wedge * \ (t) \tag{6}$$

Where:

- b: is a constant that defines the logarithmic spiral's shape,
- l: A random number in the range [−1, 1] introduces randomness in the angle of the spiral and is critical for mimicking the natural variability in whale hunting paths.

The spiral movement enables the algorithm to explore new regions of the search space, balancing exploration and exploitation. Fig.2 (b) illustrates the whales' spiral path as they explore potential solutions.

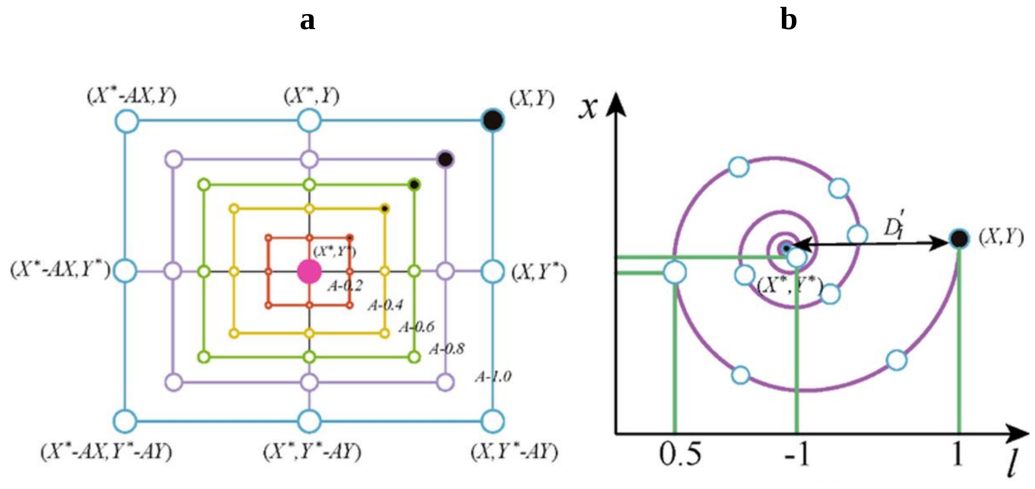


Fig 2. Bubble-Net Hunting Strategy in WOA: Shrinking Encircling Mechanism (left) and Spiral Path Exploration (right) [7]

3.3 Time Complexity of the Proposed CGAN-AWOA Algorithm

Following standard complexity reporting in metaheuristic-based optimization, let p denote the AWOA population size, T the number of AWOA iterations, n the number of training samples, d the feature dimension, Eg the number of adversarial epochs, and w the number of generator parameters. The additional cost introduced by AWOA is confined to the generator update. For each mini-batch, AWOA evaluates p candidate solutions over T iterations, where each evaluation requires forward computations through the generator and discriminator and a relaxation update over w parameters. Therefore, the dominant training-time complexity scales as $O(Eg \times\ n \times\ d \times\ p \times\ T)$.

In contrast, discriminator training remains Adam-based and scales linearly with the data pass, i.e., $O\ (Eg \times n \times d)$. Since AWOA is applied only to the generator and with bounded p and T, the added overhead is controlled and suitable for offline synthetic data generation for IDS augmentation.

4 The Proposed Method

4.1 System architecture

The Generator optimization is performed iteratively across the training epochs. Fig. 3 illustrates that the generator takes in two inputs per epoch: selected random noise from the latent space and a label vector with one position set to 1. Subsequently, synthetic feature vectors are generated by processing these inputs through fully connected layers. The Discriminator then evaluates these generated samples and calculates the classification loss. The WOA produces a set of candidate generator weight updates based on the Discriminator's feedback. The generator is updated using a dynamic relaxation mechanism with a fitness function to select the best candidate. This cyclical approach of generation, evaluation, candidate exploration, and relaxed updating is repeated throughout all training epochs, progressively enhancing the generator's ability to synthesize realistic samples while maintaining adversarial stability.

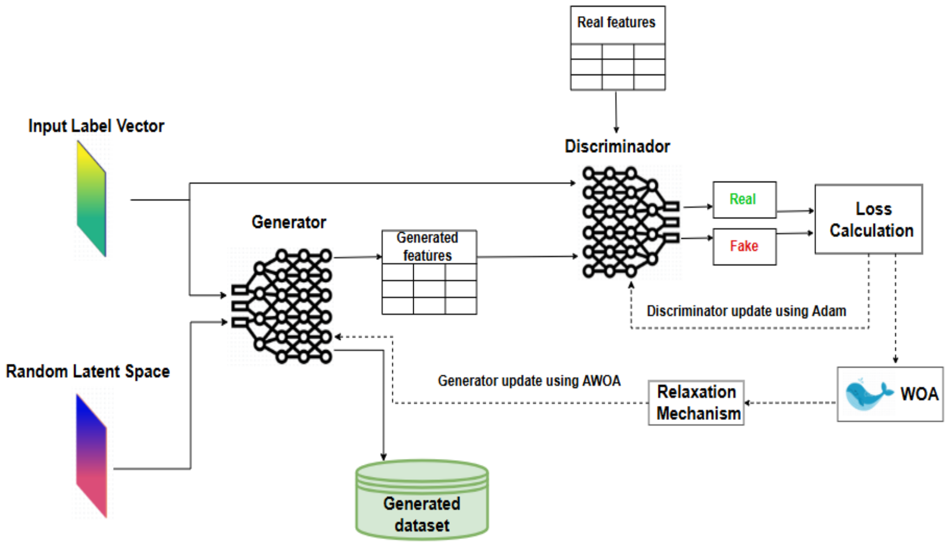
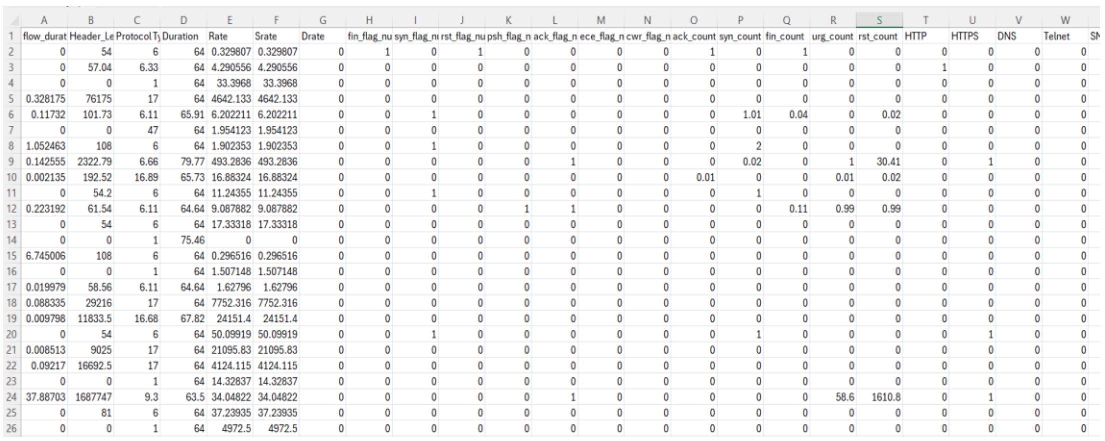


Fig 3. The architecture of the proposed CGAN-AWOA model

4.2 Dataset Collection

This study utilizes the CIC IoT Dataset 2023, officially published by the Canadian Institute for Cybersecurity (CIC) at the University of New Brunswick (UNB CIC). For practical implementation, the dataset was accessed from Kaggle (Dogra, 2023). Network flow records are categorized across 34 different classes, including DDoS attacks, reconnaissance activities, malware intrusions, and benign traffic. Forty-six statistical and protocol-specific features are associated with the traffic type's label for each record. Fig. 4 shows a sample of the dataset's structure, including the following key features: duration, protocols, packet rates, and flag counts. However, the initial dataset was class-imbalanced and redundant, and as it was a real-world network traffic dataset, further analysis and preprocessing were required.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	
1	flow_durat	Header_Le	Protocol Ty	Duration	Rate	Srate	Drate	fin_flag_nu	syn_flag_nı	rst_flag_nu	psh_flag_n	ack_flag_n	ece_flag_n	cwr_flag_n	ack_count	syn_count	fin_count	urg_count	rst_count	HTTP	HTTPS	DNS	Telnet	SN
2	0	54	6	64	0.329807	0.329807	0	1	0	1	0	0	0	0	1	0	1	0	0	0	0	0	0	
3	0	57.04	6.33	64	4.290556	4.290556	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	
4	0	0	1	64	33.3968	33.3968	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
5	0.328175	76175	17	64	4642.133	4642.133	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
6	0.11732	101.73	6.11	65.91	6.202211	6.202211	0	0	1	0	0	0	0	0	0	1.01	0.04	0	0.02	0	0	0	0	
7	0	0	47	64	1.954123	1.954123	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
8	1.052463	108	6	64	1.902353	1.902353	0	0	1	0	0	0	0	0	0	2	0	0	0	0	0	0	0	
9	0.142555	2322.79	6.66	79.77	493.2836	493.2836	0	0	0	0	0	1	0	0	0	0.02	0	1	30.41	0	1	0	0	
10	0.002135	192.52	16.89	65.73	16.88324	16.88324	0	0	0	0	0	0	0	0	0.01	0	0	0.01	0.02	0	0	0	0	
11	0	54.2	6	64	11.24355	11.24355	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	
12	0.223192	61.54	6.11	64.64	9.087882	9.087882	0	0	0	0	1	1	0	0	0	0	0.11	0.99	0.99	0	0	0	0	
13	0	54	6	64	17.33318	17.33318	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
14	0	0	1	75.46	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
15	6.745006	108	6	64	0.296516	0.296516	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
16	0	0	1	64	1.507148	1.507148	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
17	0.019979	58.56	6.11	64.64	1.62796	1.62796	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
18	0.088335	29216	17	64	7752.316	7752.316	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
19	0.009798	11833.5	16.68	67.82	24151.4	24151.4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
20	0	54	6	64	50.09919	50.09919	0	0	1	0	0	0	0	0	0	1	0	0	0	0	1	0	0	
21	0.008513	9025	17	64	21095.83	21095.83	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
22	0.09217	16692.5	17	64	4124.115	4124.115	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
23	0	0	1	64	14.32837	14.32837	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
24	37.88703	1687747	9.3	63.5	34.04822	34.04822	0	0	0	0	0	1	0	0	0	0	0	58.6	1610.8	0	1	0	0	
25	0	81	6	64	37.23935	37.23935	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
26	0	0	1	64	4972.5	4972.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Fig 4. Sample snapshot from the CIC IoT 2023 dataset

4.3 Data Preprocessing

Algorithm 1 shows the several preprocessing steps needed for the dataset before model training.

```
Algorithm 1. Dataset Preprocessing Procedure
Input:
    •  Raw dataset $D_{raw}$ with 275,235 rows and 47 columns.
    •  Sampling ratio $r = 50\%$.
Output:
    •  Final preprocessed dataset with 110,094 samples, 30 normalized features.
Step 1: Load the raw dataset $D_{raw}$.
Step 2: Randomly sample 50% of the data to reduce computational load.
Step 3: Apply a Random Forest classifier to perform feature selection.
    o  Select the top 30 most important features.
Step 4: Encode the categorical target labels using LabelEncoder.
    o  Convert class labels to numeric values.
Step 5: Apply SMOTE to the sampled dataset to generate synthetic samples for minority
classes.
Step 6: Split the dataset into training and testing sets using an 80:20 stratified split.
Step 7: Normalize feature values using MinMaxScaler to scale all features to the range
        [0,1].
Step 8: End.
```

4.4 The proposed Algorithm of CGAN_AWOA

The CGAN-AWOA framework's first step is to design a Generator and a Discriminator that work adversarially to produce high-quality synthetic network traffic samples. The Generator accepts the concatenated vector of random noise z, generated from N(0,1), and a one-hot-encoded class label. This will learn the underlying distribution of the original dataset and generate feature vectors similar to the original real traffic patterns.

Algorithm 2. CGAN-AWOA Training Loop

```
Input:
  • Dataset $X_{real}$
  • CGAN parameters: Noise vector z, label vector y, generator and discriminator weights
  • WOA parameters: population size n, maximum iterations T, random coefficients $\vec{A}$, $\vec{C}$,
    probability threshold p
  • Training hyperparameters: number of epochs, max_epochs, batch size.
Output:
  • Trained the CGAN-IWOA model (updated generator parameters θ).

Step1: Initialize generator and discriminator weights.
Step 2: Pretrain the discriminator using Adam
    FOR pretrain_epoch = 1 to E_p DO:
        FOR each batch in training data DO:
            o Generate fake data using noise z and labels
            o Compute discriminator outputs for real and fake data
            o Calculate loss using Binary Cross-Entropy  // see eq. (4)
            o Backpropagate and update only the discriminator weights.
        End FOR
    End FOR

Step 3: Train the generator using AWOA
    FOR epoch = 1 to max_epochs DO
        FOR each batch in the training data DO
          o Generate noise and one-hot encode class labels
          o Generate synthetic data using the generator
          o Compute discriminator outputs for real and fake data
          o Update the discriminator using BCE loss and backpropagation. // see Eq.(4)
          o Measure discriminator accuracy before generator update
          o WOA Feedback Optimization (Generator Update)
            o Initialize population of candidate generator updates $X_i$
            o Evaluate each candidate using discriminator feedback as a fitness score
            o Iterate for t=1 to T:
                ▪ Update the position of each agent using:
                  If p<0.5:
                    If $|\vec{A}|$< 1:
                        $X_{\text{new}} = X_{\text{best}} - \overline{A} \cdot |\overline{C} \cdot (X_{\text{best}} - X_i)|$
                    Else:
                        $X_{\text{new}} = X_{\text{rand}} - \overline{A} \cdot |\overline{C} \cdot (X_{rand} - X_i)|$
                  If p≥0.5:
                    Update the search agent by Eq. (6)
                    Select the best update candidate
        o Adaptive Alpha Update (Relaxation)
            o Compare new accuracy with previous accuracy:
              IF improved:
                    Set alpha = 0.7
              ELSE:
                    Set alpha = 0.3
            o Compute weighted update:
              $\Delta WOA^{n+1} = \alpha \cdot \Delta WOA^{n+1} + (1-\alpha) \cdot \Delta WOA^{n}$ // relaxation equation
            o Apply delta to update generator weights.
        o Save the new discriminator accuracy for the next iteration.
      END FOR
  END FOR
```

Due to the careful design of the training procedure, the CGAN-AWOA framework was trained with stable, progressive adversarial learning. Standard CGAN training is known to be particularly unstable when the Discriminator and Generator performance is not smooth and exhibits sharp oscillations. The training addresses this using two strategies: optimizing the Discriminator with the Adam optimizer and updating the Generator using the AWOA with a relaxing mechanism, as shown in Algorithm 2.

4.4.1 Adam Optimizer for Discriminator

A discriminator is essential for controlling the training dynamics of the CGAN-AWOA framework. Before adversarial training, a pre-training phase was introduced to ensure that the Discriminator starts from a stable point (see Algorithm 2). In this phase, the Discriminator was trained on real samples and on outputs from an untrained Generator to establish an initial decision boundary between authentic and synthetic data. The pretraining of this method reduced instability during the initial training period and provided helpful feedback to the Generator at the start of adversarial training. The Adam optimizer trained the discriminative model throughout both pretraining and adversarial training.

4.4.2 Generator Optimization via Adaptive WOA

In the optimization process, a population of candidate solutions is initialized. A diverse set of search agents is created by applying small random perturbations to the generator's current parameters, thereby generating each candidate. The complete process of evaluating and selecting the best candidate solution follows the Whale Optimization search strategy detailed in Algorithm 2. With this diversification, the algorithm gains access to a more effective parameter space exploration and retains the original generator structure to support stable, progressive optimization.

4.4.2.1 Fitness Function

In the proposed CGAN-AWOA framework, the Adaptive Whale Optimization Algorithm is employed to optimize the generator parameters by minimizing a mini-batch adversarial objective. For a given batch of noise vectors z and conditional labels y, the fitness of a candidate generator is defined as the binary cross-entropy loss between the discriminator output on generated samples and a smoothed real target label:

$$f(\theta_G) = E_{z,y}[\, BCE\ (D(G(z,y;\ \theta_G), y)\ y_{real})]$$

Where θ_G denotes the generator parameters, z is a random noise vector, y represents the conditional class label, D (.) is the discriminator, BCE denotes the binary cross-entropy loss, and y_{real} is the smoothed real target label. By minimizing this fitness function, AWOA guides the generator toward producing samples that the discriminator classifies as real while maintaining stable adversarial training dynamics.

4.4.3 Relaxation Mechanism

After defining the value of the best candidate solution obtained from the Whale Optimization process, the generator's parameter update is not applied directly. Instead, dynamic relaxation is used to control the update's impact. In particular, an adaptive relaxation coefficient is selected based on the change in the Discriminator's performance. If the Discriminator's classification accuracy improves over the previous iteration, a larger relaxation coefficient (α=0.7) is used, enabling a more aggressive integration of the new update. Conversely, a smaller coefficient (α=0.3) smooths the update and stabilizes the Discriminator's performance. The final update of the Generator is a weighted combination of the current evolutionary update and the previous one. The detailed steps of the generator's update mechanism, including the adaptive relaxation strategy, are formally described in Algorithm 2. This adaptive relaxation strategy aims to facilitate smoother parameter transitions and maintain stable adversarial dynamics throughout training.

5 Results, Analysis, and Discussions

This section provides a detailed analysis of the experimental training results and evaluates the proposed CGAN-AWOA. The experiments aim to assess the effects of incorporating a dynamically adaptive optimization approach into the generator training process and compare its efficiency against traditional methods. Several experimental cases were designed to thoroughly validate the proposed model. First, a baseline CGAN is trained using the Adam optimizer, followed by a variant CGAN using the standard WOA and the final proposed CGAN-AWOA framework, which adds to the standard WOA by tuning the relaxation factor α adaptively to increase the generator's stability and convergence. This comparison will compare these models across various critical metrics using 5-fold cross-validation [20]. In addition to comparisons of model-level performance, several relaxation strategies were examined within the AWOA-based training pipeline.

These strategies include:

- Discrete static values of α.
- Continuous values within a specified range.

5.1 Training parameters

Table 1: Model architecture and training hyperparameters

Parameters	Value
Batch size	64
Learning rate	1e-5
Epochs (Generator)	50
Pre-training Epochs (Discriminator)	5
Noise dimension	100
Hidden dimension (Generator)	128
Hidden dimension (Discriminator)	64

Loss function	Binary Cross Entropy
Optimizer (Discriminator)	Adam
WOA - Population Size	30
WOA - Max Iterations	15
Relax-Alpha Strategy	Discrete [0.3, 0.7]

We set the batch size to 64 to balance computational efficiency and generalization. After preliminary tuning, we selected a small learning rate (1e−5) to stabilize adversarial training, which is sensitive to optimizer dynamics. Mode collapse or instability was observed for a larger learning rate. The model was trained for 50 epochs. Furthermore, the discriminator was initially trained for five epochs using both real and fake data to develop a robust border that was confident enough to distinguish between real and counterfeit images (adversarial training). The generator provided a 100-unit dimensional noise vector (latent space), which was sufficient to ensure diversity in synthetic data generation. The generator and discriminator also had moderately sized hidden layers (128 and 64 units, respectively), which were expressive enough to learn the data distribution without overfitting. The discriminator used an Adam optimizer. In the AWOA variants, the generator parameters were updated by a metaheuristic population-based search. The relax-alpha mechanism was introduced in the AWOA framework. Through this approach, α ∈ [0.3, 0.7] adaptively changes the updates of the generator's parameters across epochs. As the discriminator's performance improves, α increases to take larger steps or decreases to promote stability and gradual learning.

5.2 Model Evaluation

Several performance metrics were used to comprehensively evaluate the proposed CGAN-AWOA model and its counterparts. The metrics were chosen to provide insight into the discriminative capability and to assess the quality of the synthetic data generated. This evaluation is divided into two main domains. Performance in Classifying: Accuracy, Precision, Recall, F1Score, AUCROC Curve, and Confusion Matrix [6]. Generation Quality: Fréchet Inception Distance (FID) [21] and Generator Mean Squared Error (GMSE) [24].

5.2.1 Model-Level Comparison

- **Accuracy**

Accuracy measures the overall correctness of the model's predictions and is defined as:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{7}$$

where:

- TP = True Positives
- TN = True Negatives
- FP = False Positives
- FN = False Negatives

As shown in Tables 2–3 and Fig. 5, CGAN-AWOA outperformed the baselines, reaching ≈ 99% accuracy with low variance across folds—evidence of improved generalization and stable learning.

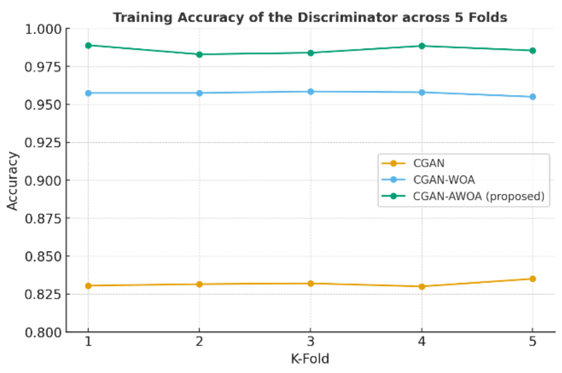
Table 2: Training Accuracy of the Discriminator across 5 Folds

Model	K-Folds				
	1	2	3	4	5
CGAN	0.8305	0.8315	0.8320	0.8300	0.8350
CGAN-WOA	0.9575	0.9575	0.9585	0.9580	0.9550
CGAN-AWOA (proposed model)	0.9890	0.9830	0.9840	0.9885	0.9855

Table 3: Testing Accuracy of the Discriminator across 5 Folds

Model	K-Folds				
	1	2	3	4	5
CGAN	0.8459	0.8443	0.8483	0.8497	0.8384
CGAN-WOA	0.9548	0.9569	0.9538	0.9573	0.9546
CGAN-AWOA (proposed model)	0.9857	0.9865	0.9859	0.9861	0.9852

(a) **(b)**



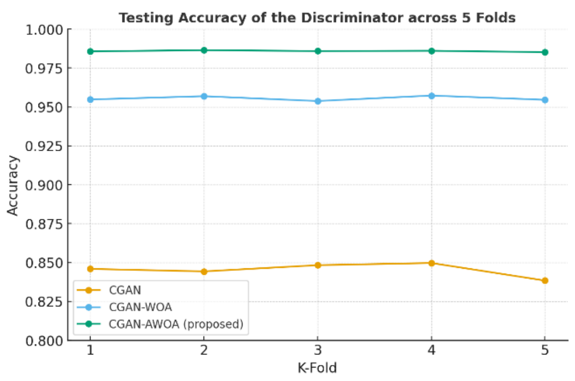


Fig 5. (a) Comparison of Training Accuracy across 5 Folds for All Models, (b) Comparison of Testing Accuracy across 5 Folds for All Models

The results reported in Tables 2 and 3 provide a comparative evaluation of the proposed CGAN-AWOA framework against its baseline variants. In the proposed design, the relaxation mechanism is tightly integrated with the AWOA-based generator update and is applied directly to the weight adjustments produced by the search process. Accordingly, Relax-Alpha is introduced as a stabilization strategy rather than as an independent optimization component.

The effect of WOA alone is captured through the CGAN-WOA baseline, while the combined impact of AWOA and the relaxation mechanism is evaluated using the proposed CGAN-AWOA model.

- **Precision**

  Precision quantifies the number of true positives among all predicted positives:

$$Precision = \frac{TP}{TP+FP} \tag{8}$$

  High precision indicates a low false-positive rate.

As shown in Tables 4 and 5 and Figs. 6a and 6b, CGAN-AWOA had the highest CGAN precision (>98.5%) across both training and testing, outperforming CGAN-WOA and CGAN-Adam, with better generalization and a lower false-positive rate.

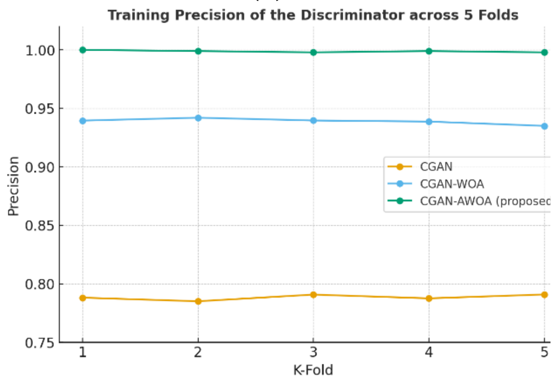
Table 4: Training Precision of the Discriminator across 5 Folds

Model	K-Folds				
	1	2	3	4	5
CGAN	0.7881	0.7850	0.7907	0.7875	0.7908
CGAN-WOA	0.9395	0.9420	0.9396	0.9387	0.9350
CGAN-AWOA (proposed model)	1.0000	0.9990	0.9979	0.9990	0.9979

Table 5: Testing Precision of the Discriminator across 5 Folds

Model	K-Folds				
	1	2	3	4	5
CGAN	0.8478	0.8468	0.8498	0.8515	0.8406
CGAN-WOA	0.9556	0.9577	0.9547	0.9581	0.9554
CGAN-AWOA (proposed model)	0.9861	0.9867	0.9862	0.9864	0.9855

(a)



(b)

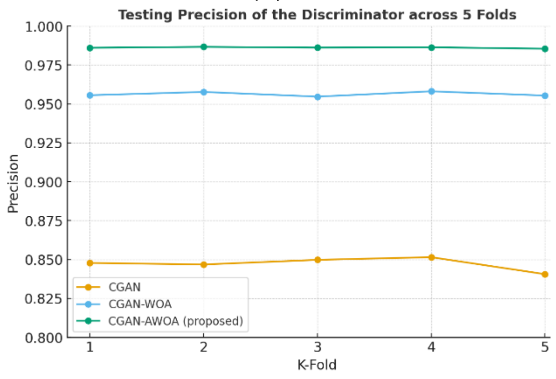


Fig 6. (a) Precision per Fold – Training, (b) Precision per Fold – Testing

- **Recall**

Recall, or sensitivity, is the ratio of correctly predicted positive samples to all actual positives:

$$Recall = \frac{TP}{TP+FN} \quad (9)$$

It reflects how well the model captures actual positive cases.

As illustrated in Tables 6 and 7 and Fig.7a and Fig.7b, CGAN-AWOA was able to achieve the highest and most stable recall (>98.5%) across all folds in both training and testing of CGAN-WOA and CGAN-Adam, with little deviation from these values.

Table 6: Training Recall of the Discriminator across 5 Folds

Model	K-Folds				
	1	2	3	4	5
CGAN	0.9040	0.9130	0.9030	0.9040	0.9110
CGAN-WOA	0.9780	0.9750	0.9800	0.9800	0.9780
CGAN-AWOA (proposed model)	0.9780	0.9670	0.9700	0.9780	0.9730

Table 7: Testing Recall of the Discriminator across 5 Folds

Model	K-Folds				
	1	2	3	4	5
CGAN	0.8459	0.8443	0.8483	0.8384	0.8453
CGAN-WOA	0.9548	0.9569	0.9538	0.9573	0.9546
CGAN-AWOA (proposed model)	0.9857	0.9865	0.9859	0.9861	0.9852

(a) **(b)**

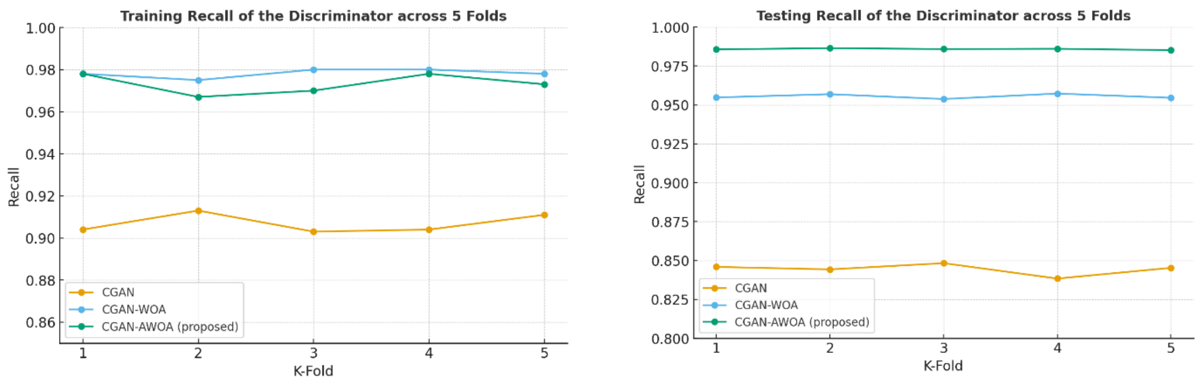


Fig 7. (a) Recall per Fold – Training, (b) Recall per Fold – Testing

- **F1-Score**

F1-score is the harmonic mean of precision and recall, offering a balance between the two:

$$F1 - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \qquad (10)$$

It is beneficial in imbalanced datasets.

Tables 8 and 9 and Fig.8a and Fig.8b show that CGAN-AWOA consistently obtained the best and the most stable F1-scores (always above 98.5% both in training and testing) with significantly higher scores than CGAN-WOA (roughly 95.5%) and the baseline CGAN-Adam (almost always lower than 85%). 'This reflects substantial precision concerning recall.

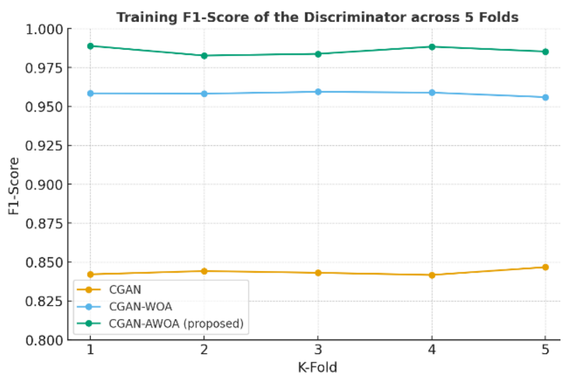
Table 8: Training F1-Score of the Discriminator across 5 Folds

Model	K-Folds				
	1	2	3	4	5
CGAN	0.8421	0.8442	0.8431	0.8417	0.8467
CGAN-WOA	0.9584	0.9582	0.9594	0.9589	0.9560
CGAN-AWOA (proposed model)	0.9889	0.9827	0.9838	0.9884	0.9853

Table 9: Testing F1-Score of the Discriminator across 5 Folds

Model	K-Folds				
	1	2	3	4	5
CGAN	0.8456	0.8440	0.8482	0.8495	0.8381
CGAN-WOA	0.9547	0.9568	0.9537	0.9573	0.9546
CGAN-AWOA (proposed model)	0.9857	0.9865	0.9859	0.9861	0.9852

(a)



(b)

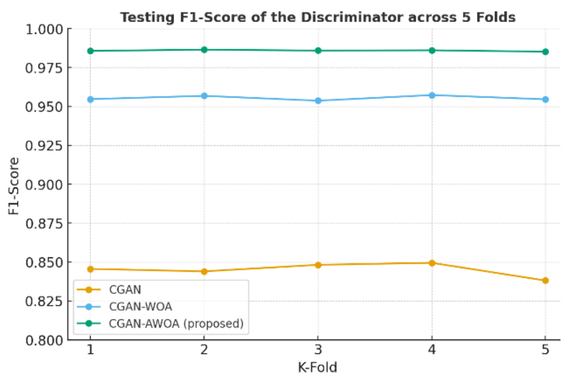


Fig 8. (a) F1-Score per Fold – Training, (b) F1-Score per Fold – Testing

- **AUC-ROC (Area under the Curve – Receiver Operating Characteristic)**

This metric assesses the model's ability to separate classes at different threshold levels. Because the AUC is higher, the discriminator can better separate real and fake samples.

The results shown in Tables 10 and 11 and Fig.9a and Fig.9b indicate that CGAN-AWOA obtained the highest AUC ROC score (~99.6%), as compared to CGAN-WOA and CGAN-Adam. It confirms its firm decision boundaries and reliable class separation.

Table 10: Training AUC-ROC of the Discriminator across 5 Folds

Model	K-Folds				
	1	2	3	4	5
CGAN	0.9375	0.9404	0.9437	0.9400	0.9399
CGAN-WOA	0.9911	0.9911	0.9929	0.9914	0.9889
CGAN-AWOA (proposed model)	0.9959	0.9963	0.9973	0.9966	0.9966

Table 11: Testing AUC-ROC of the Discriminator across 5 Folds

Model	K-Folds				
	1	2	3	4	5
CGAN	0.9061	0.9018	0.9056	0.9057	0.9013
CGAN-WOA	0.9910	0.9910	0.9901	0.9913	0.9908
CGAN-AWOA (proposed model)	0.9967	0.9962	0.9959	0.9968	0.9965

(a) **(b)**

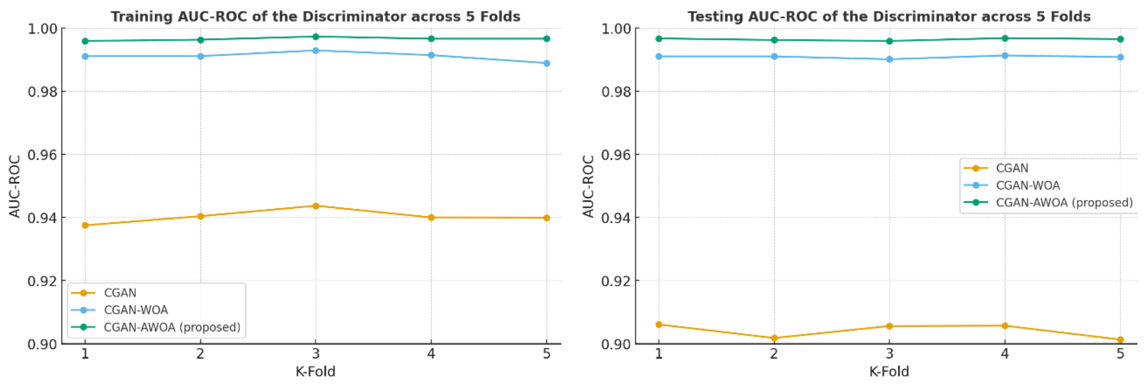


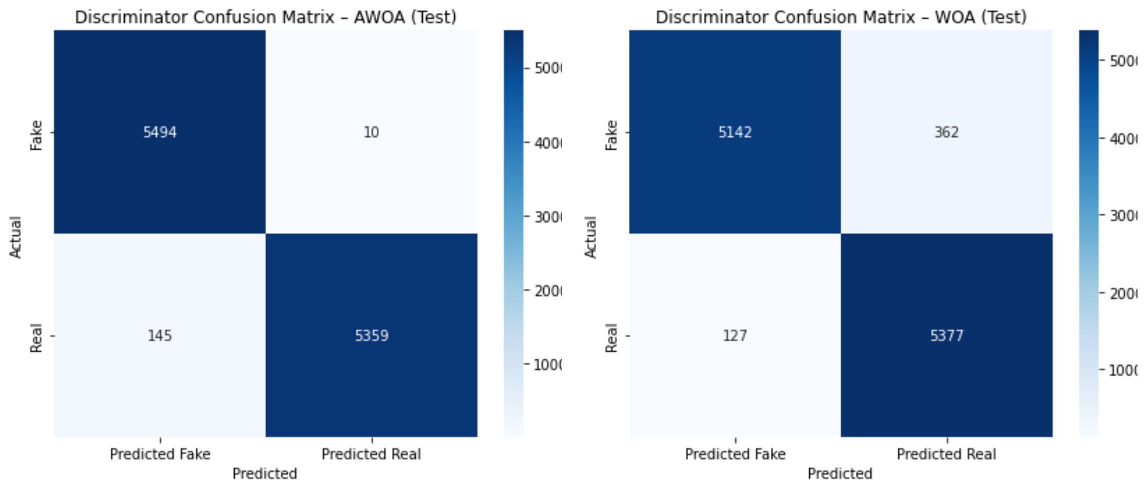
Fig 9. (a) AUC-ROC per Fold – Training, (b) AUC-ROC per Fold – Testing

- **Confusion Matrix**

A confusion matrix is a complete summary of classification performance. Showing the distribution of predictions across actual vs. predicted classes is necessary for understanding model bias.

The performance of each model on both the training and testing datasets is shown in Figures 10a, 10 b, and 10c. CGAN-AWOA provided near-perfect class separation, as shown in Figures 10a and 10 b, indicating that it generalized very well. On the other hand, the baseline CGAN showed higher rates of false positives and negatives, as shown in Fig. 10c, indicating lower classification confidence and reliability.

(a) **(b)**



(c)

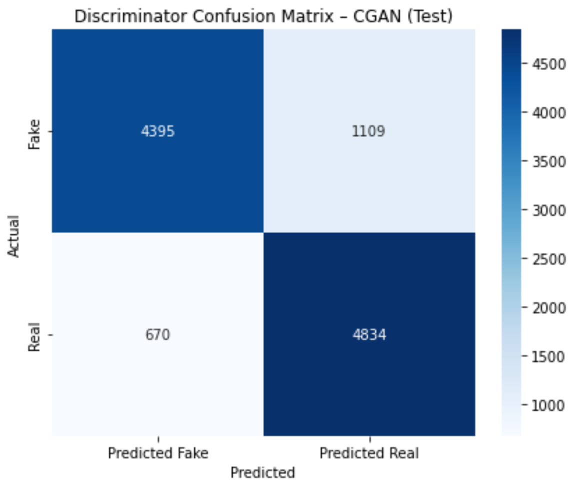


Fig 10. Confusion Matrix – (a) CGAN-AWOA (Test), (b) CGAN-WOA, (c) CGAN (Test)

- **Overall Average Performance**

However, to consolidate the model comparison, average performance metrics were calculated across 50 training epochs for all evaluation metrics, as shown in Table 12. The testing will evaluate peak performance, training stability, convergence behavior, and overall generative quality. In all measured metrics, the proposed model, CGAN-AWOA, outperformed both CGAN-Adam and CGAN-WOA. With an average test accuracy of 98.59%, precision of 98.62%, recall of 98.59%, and F1-score of 98.59%, the AUC-ROC was close to perfect at 99.64%.

In addition to classification performance, the CGAN-AWOA produced the most realistic and diverse samples, as evidenced by its lowest FID score (27.68) and GMSE (0.056), both significantly lower than those of other models.

Table 12: Average Metrics over 50 Epochs

Model	CGAN	CGAN-WOA	CGAN-AWOA (The proposed model)
		Train	
Average D-Accuracy	0.8318	0.9573	0.9890
Average D- Precision	0.7884	0.9390	1.0000
Average D-Recall	0.9070	0.9782	0.9780
Average D-F1-Score	0.8436	0.9582	0.9889
Average D-AUC-ROC	0.9403	0.9911	0.9959
		Test	
Average D –Accuracy	0.8453	0.9555	0.9859
Average D –Precision	0.8473	0.9563	0.9862
Average D -Recall	0.8453	0.9555	0.9859
Average D-F1-Score	0.8451	0.9554	0.9859
Average D-AUC- ROC	0.9041	0.9908	0.9964
FID score – Generator	83.6286	160.3236	27.6820
Generator MSE score	2.1046	4.2262	0.0564

5.2.2 Alpha Adaptation Variants - Discrete vs Continuous

Two α-adaptation strategies were evaluated in this experiment: a fixed discrete approach using 0.3 and 0.7, and a continuous adaptation with a range of 0.3 to 0.7. Table 13 shows that the discriminator accuracy during training (98.90%) and testing (98.59%) was slightly higher for the discrete strategy. Additionally, perfect training precision (1.000) was achieved, along with strong recall and AUC-ROC scores. Additionally, the continuous strategy performed well in terms of generative quality, with lower FID (25.03 vs. 27.68) and GMSE (0.0412 vs. 0.0564). As seen in Fig. 11a and Fig. 11b, the accuracy progression during training was smoother, indicating better learning stability.

Table 13: Comparison of Discrete vs. Continuous α Based on Discriminator Accuracy (Fake Samples Only)

Model	AWOA – Discrete [0.3,0.7]	AWOA –Continuous [0.3-0.7]
Train		
Average D-Accuracy	0.9890	0.9706
Average D- Precision	1.0000	0.9719
Average D-Recall	0.9780	0.9692
Average D-F1-Score	0.9889	0.9706
Average D-AUC-ROC	0.9959	0.9958
Test		
Average D – Accuracy	0.9859	0.9709
Average D – Precision	0.9862	0.9709
Average D -Recall	0.9859	0.9709
Average D-F1-Score	0.9859	0.9709
Average D-AUC-ROC	0.9964	0.9957
FID score	27.6820	25.0362
GMSE score	0.0564	0.0412

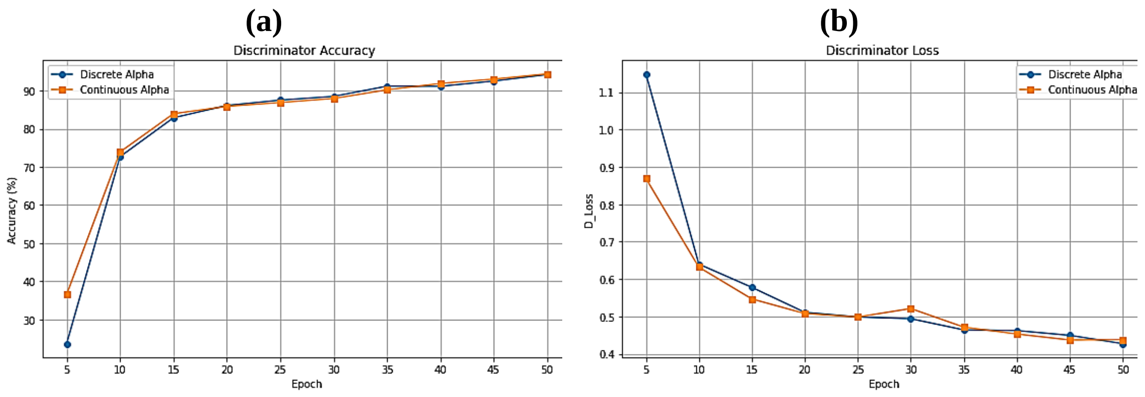


Fig 11. (a) Discriminator Accuracy Curve Using Discrete vs. Continuous Alpha, (b) Discriminator Loss Curve Using Discrete vs. Continuous Alpha

To evaluate the practical utility of the generated data, an additional IDS-level experiment was conducted. An IDS classifier was trained using synthetic samples generated by the proposed CGAN-AWOA framework and evaluated on a held-out real test set. The results indicate stable and balanced detection performance, with improvements observed in macro-averaged F1-score, particularly for classes with sufficient support. These findings suggest that the generated data preserves discriminative characteristics relevant for intrusion detection tasks.

6 Conclusion

This study proposed a novel approach for generating high-quality synthetic data working within the adversarial learning framework for the problem of IDS. A CGAN with an AWOA model was proposed, introducing a relaxation-based update mechanism that dynamically updates the generator with real-time feedback from the discriminator. Compared to traditional GAN training methods, which often rely on static optimizers or blind mutation strategies, such as the Adam optimizer, the proposed AWOA framework leverages the discriminator's accuracy trends over fake samples to determine the direction and intensity of the generator updates. Through extensive experimentation, the results showed that the superior CGAN-AWOA model outperformed baseline approaches (standard CGAN and CGAN-WOA). The classification performance was significantly higher, the generation error was lower, and the adversarial behavior was better in AWOA: Training Accuracy reached up to 98.90%, and Testing Accuracy up to 98.59% F1-Score remained high across training and testing, with a top score of 98.89% AUC-ROC values exceeded 99.6%, reflecting strong separation between real and fake data FID and GMSE scores were notably low, indicating high-quality, realistic data generation As a practical and key achievement besides numerical performance, the proposed framework also provided training stability. Unlike baseline models, the AWOA model's discriminator performance was smooth and consistent rather than erratic, and it oscillated sharply as training progressed: Discriminator accuracy increased gradually and reliably over time, and Discriminator loss decreased steadily without significant fluctuations. This behavior must make the learning process predictable and trackable. Now, we can estimate the model’s future behavior, be confident in its progress, and determine the optimal stopping point during training, which was missing in traditional GANs. In conclusion, this work demonstrated that feedback-driven evolutionary updates under a dynamic relaxation mechanism can significantly enhance stability and improve adversarial data generation. The CGAN-AWOA framework can be a reliable and intelligent solution for generating synthetic attack data, making it a strong candidate for enhancing IDS training in practical, real-world scenarios.

References

[1] Adam, M., Hammoudeh, M., Alrawashdeh, R., & Alsulaimy, B. (2024). A survey on security, privacy, trust, and architectural challenges in IoT systems. *IEEE Access*.

[2] Stojmenovic, I., & Wen, S. (2014, September). The fog computing paradigm: Scenarios and security issues. In *2014 federated conference on computer science and information systems* (pp. 1-8). IEEE.

[3] Khan, S., Parkinson, S., & Qin, Y. (2017). Fog computing security: a review of current applications and security solutions. *Journal of Cloud Computing*, *6*, 1-22..

[4] Neto, E. C. P., Dadkhah, S., Ferreira, R., Zohourian, A., Lu, R., & Ghorbani, A. A. (2023). CICIoT2023: A real-time dataset and benchmark for large-scale attacks in IoT environment. *Sensors*, *23*(13), 5941.

[5] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2020). Generative adversarial networks. *Communications of the ACM*, 63, 139–144.

[6] Guo, X., & Mounjid, O. (2024). GANs training: A game and stochastic control approach. *Mathematical Finance*, *34*(2), 522-556.

[7] Brodzicki, A., Piekarski, M., & Jaworek-Korjakowska, J. (2021). The whale optimization algorithm approach for deep neural networks. *Sensors*, *21*(23), 8003.

[8] Nadimi-Shahraki, M. H., Zamani, H., Asghari Varzaneh, Z., & Mirjalili, S. (2023). A systematic review of the whale optimization algorithm: theoretical foundation, improvements, and hybridizations. *Archives of Computational Methods in Engineering*, *30*(7), 4113-4159..

[9] Vallabhaneni, R., Vaddadi, S. A., Pillai, S. E. V. S., Addula, S. R., & Ananthan, B. (2024). Detection of cyberattacks using bidirectional generative adversarial network. *Indonesian Journal of Electrical Engineering and Computer Science*, *35*(3), 1653-1660.

[10] Kotal, A., Luton, B., & Joshi, A. (2024, July). KiNETGAN: Enabling Distributed Network Intrusion Detection through Knowledge-Infused Synthetic Data Generation. In *2024 IEEE 44th International Conference on Distributed Computing Systems Workshops (ICDCSW)* (pp. 140-145). IEEE.

[11] Chu, H. C., & Lin, Y. J. (2023). Improving the IoT Attack Classification Mechanism with Data Augmentation for Generative Adversarial Networks. *Applied Sciences*, *13*(23), 12592.

[12] Alabsi, B. A., Anbar, M., & Rihan, S. D. A. (2023). Conditional tabular generative adversarial based intrusion detection system for detecting ddos and dos attacks on the internet of things networks. *Sensors*, *23*(12), 5644.

[13] Alabdulwahab, S., Kim, Y. T., Seo, A., & Son, Y. (2023). Generating synthetic dataset for ml-based ids using ctgan and feature selection to protect smart iot environments. Applied Sciences, 13(19), 10951.

[14] Wang, J., Yan, X., Liu, L., Li, L., & Yu, Y. (2022). CTTGAN: traffic data synthesizing scheme based on conditional GAN. *Sensors*, *22*(14), 5243..

[15] Shahriar, M. H., Haque, N. I., Rahman, M. A., & Alonso, M. (2020, July). G-ids: Generative adversarial networks assisted intrusion detection system. In 2020 *IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC)* (pp.

376-385). IEEE.

[16] Mahmood, S., Bawany, N. Z., & Tanweer, M. R. (2023). A comprehensive survey of whale optimization algorithm: modifications and classification. *Indonesian Journal of Electrical Engineering and Computer Science*, *29*(2), 899.

[17] Ullah, I., & Mahmoud, Q. H. (2021). A framework for anomaly detection in IoT networks using conditional generative adversarial networks. *IEEE Access*, *9*, 165907-165931.

[18] Han, F., Zhang, H., Chatterjee, S., Guo, Q., & Wan, S. (2019). A modified generative adversarial nets integrated with stochastic approach for realizing super-resolution reservoir simulation. *IEEE Transactions on Geoscience and Remote Sensing*, *58*(2), 1325-1336..

[19] Mirjalili, S., & Lewis, A. (2016). The whale optimization algorithm. *Advances in engineering software*, 95, 51-67.

[20] Lumumba, V. W., Kiprotich, D., Makena, N. G., Kavita, M. D., & Mpaine, M. L. (2024). Comparative Analysis of Cross-Validation Techniques: LOOCV, K-Folds Cross-Validation, and Repeated K-Folds Cross-Validation in Machine Learning Models. *Am. J. Theor. Appl. Stat*, *13*, 127-137.

[21] Chan, D. A., & Sithungu, S. P. (2024, November). Evaluating the Suitability of Inception Score and Fréchet Inception Distance as Metrics for Quality and Diversity in Image Generation. In *Proceedings of the 2024 7th International Conference on Computational Intelligence and Intelligent Systems* (pp. 79-85).

[22] Zhao, X., Fok, K. W., & Thing, V. L. (2024). Enhancing network intrusion detection performance using generative adversarial networks. Computers & Security, 145, 104005.

[23] Rahman, S., Pal, S., Mittal, S., Chawla, T., & Karmakar, C. (2024). SYN-GAN: A robust intrusion detection system using GAN-based synthetic data for IoT security. *Internet of Things*, *26*, 101212.