http://dx.doi.org/10.18623/rvd.v23.5674

ENHANCED GREY WOLF OPTIMIZER WITH DYNAMIC ENCIRCLING AND FITNESS-BASED BLENDING FOR CYBERATTACK DETECTION USING SELF-ORGANIZING FUZZY NEURAL NETWORK

OTIMIZADOR GREY WOLF APRIMORADO COM CERCO DINÂMICO E MISTURA BASEADA NA APTIDÃO PARA DETECÇÃO DE CIBERATAQUES UTILIZANDO REDE NEURAL FUZZY AUTO-ORGANIZADA

Article received on: 12/1/2025
Article accepted on: 2/27/2026

Sharaf Aldeen Abdulkadhum Abbas*
* Department of Electrical and Computer Engineering, Altinbas University, MahmutbeyYerleşkesi, Bağcılar, Turkey
sharafabeed@gmail.com

The authors declare that there is no conflict of interest

Abstract
Intrusion Detection Systems (IDS) has an intrinsic role in protecting contemporary networks against developing cyberattacks. Still, current IDSs suffer from high false positives, low resistance to novel attacks, and inferior feature selection in heavy traffic. This paper introduces a new and improved Grey Wolf Optimizer (GWO) with adaptive mechanisms that enhance exploration and exploitation capabilities in complex search spaces. A sigmoid adaptive decay function and leader-spiral motion boost search efficiency and rate of convergence. A dynamic encircling mechanism weighted by fitness adjusts the search based on wolf fitness, increasing diversity and accuracy in the search. A Fitness Difference-Based Blending Mechanism (FDBM) enhances position updates by utilizing elite wolves' positional differences, maximizing flexibility and accuracy. The chosen features are treated with a Self-Organizing Stacked Fuzzy Neural Network (SOSFNN) to cope better with the dynamics and for better accuracy of detection in dynamic environments. The model provides 98.22% accuracy, 94.92% precision, 94.48% recall, and 94.67% F1-score, with a minimal false positive (0.0007 for UNSW-NB15 and 1.14% for CICIDS-2018). Our proposed improvements illustrate better detection efficiency and lower temporal complexity (27.10s for CICIDS-2018 and 15.10s for UNSW-NB15) compared to state-of-the-art studies, thus being highly accurate, adaptable, and effective for real-time IDS applications.

Keywords: Cybersecurity. Wolf Optimization Algorithm. Stacked Fuzzy Neural Network. Precision. Recall.

Resumo
Os Sistemas de Detecção de Intrusão (IDS) desempenham um papel fundamental na proteção das redes contemporâneas contra os crescentes ataques cibernéticos. No entanto, os IDSs atuais sofrem com altos índices de falsos positivos, baixa resistência a novos ataques e seleção de características inferior em condições de tráfego intenso. Este artigo apresenta um novo e aprimorado Otimizador do Lobo Cinza (GWO) com mecanismos adaptativos que melhoram as capacidades de exploração e aproveitamento em espaços de busca complexos. Uma função de decaimento adaptativa sigmoide e um movimento em espiral do líder aumentam a eficiência da busca e a taxa de convergência. Um mecanismo de cerco dinâmico ponderado pela aptidão ajusta a busca com base na aptidão dos lobos, aumentando a diversidade e a precisão na busca. Um Mecanismo de Mistura Baseado na Diferença de Adequação (FDBM) aprimora as atualizações de posição utilizando as diferenças posicionais dos lobos de elite, maximizando a flexibilidade e a precisão. As características escolhidas são processadas por uma Rede Neural Fuzzy Empilhada Auto-Organizada (SOSFNN) para lidar melhor com a dinâmica e obter maior precisão de detecção em ambientes dinâmicos. O modelo oferece 98,22% de precisão, 94,92% de exatidão, 94,48% de recall e 94,67% de F1-score, com um mínimo de falsos positivos (0,0007 para UNSW-NB15 e 1,14% para CICIDS-2018). As melhorias propostas demonstram maior eficiência de detecção e menor complexidade temporal (27,10 s para CICIDS-2018 e 15,10 s para UNSW-NB15) em comparação com estudos de ponta, sendo, portanto, altamente precisas, adaptáveis e eficazes para aplicações de IDS em tempo real.

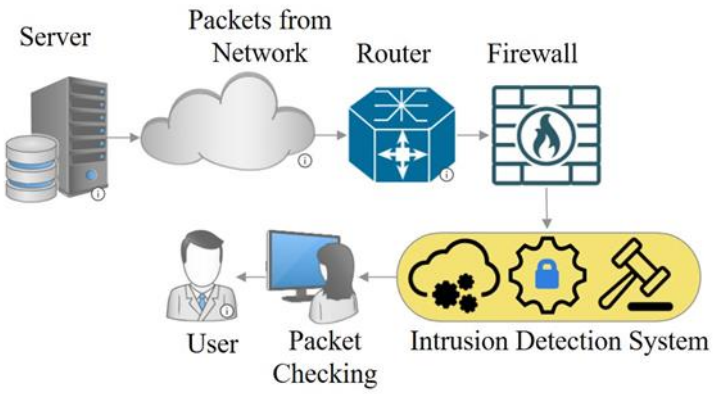
Palavras-chave: *Segurança Cibernética. Algoritmo de Otimização de Wolf. Rede Neural Difusa Empilhada. Precisão. Recuperação.*

1 INTRODUCTION

The accelerated growth of interconnected systems such as IoT, cloud computing, and cyber-physical systems has reshaped contemporary infrastructure, facilitating unobstructed communication, automation, and data sharing among different industries [1]. Nevertheless, the interconnectivity also poses risks, with cybercriminals using these systems to launch sophisticated attacks. Figure 1 shows the working flow of an Intrusion Detection System (IDS) within a network infrastructure. Incoming packets from the network are routed through a firewall, which filters traffic based on predefined rules. The IDS then analyzes the packets for suspicious activities, allowing legitimate traffic to reach the user while identifying and mitigating potential threats. With the increase of sophistication and commonality of threats, research became essential to protect networks from uninvited intruders carrying malicious actions and abnormalities.

Figure 1

Workflow depiction of and IDS process for threat detection and network traffic monitoring.



Modern cyberattacks made classic IDS approaches, which are signature or rule based, incapable of catching up to its techniques [2]. Confidence in rule-based methods

is compromised due to the large volume of false positives produced, and Signature based methods lack the ability to defend against zero-day attacks [3]. The high growth of networks increased their complexity making scalability impossible.

The models of Deep Learning (DL) enables solutions that can give promising results as they can detect subtle patters in large volume of data [4]. Nevertheless, these DL models have some impairments posed by high power consumptions to perform computations in addition to challenges posed by unbalanced data and overfitting [5]. These problems are further amplified by the increasing growth of network dimentionality leading to what know as the "curse of dimensionality," where the DL models begin to have difficulty in deriving high-efficiency patters to be useful [6]. Efficient feature selection, dimensionality reduction, and hybrid methods are necessary to improve efficiency and scalability. Solving these problems is crucial in coming up with IDS solutions that are accurate and adequate for real-time, resource-limited environments. In recent times, bio-inspired optimization algorithms have emerged as effective approaches for enhancing search space exploration and improving solution accuracy in complex optimization problems [7][8][9]. They demonstrate high computational efficiency and effectively navigate very large solution spaces [10].

In contrast, bio-inspired algorithms, such as the Grey Wolf Optimization (GWO), offer robust mechanisms for global optimization [11]. However, the conventional GWO suffers from various limitations, such as premature convergence, imbalance between exploration and exploitation, and sensitivity to initial parameter settings [12]. It is very challenging to solve high-dimensional and complex problems, like intrusion detection, with a vast and multimodal search space [13]. In addition, there is also an imbalance between exploration (exploration of new solutions) and exploitation (exploitation of current solutions), where the algorithm could wander without being directed or converge too quickly without fully searching the search space [14]. Moreover, the absence of an adaptive mechanism to escape from local minima limits WOA's effectiveness in intrusion detection scenarios that necessarily demand continuous learning and adaptability.

In this paper, we introduce a novel mathematical framework to enhance the exploration and exploitation tradeoff of GWO and mitigate its disadvantages in high-dimensional and complex spaces. The standard GWO is prone to premature convergence and insufficient diversity because the coefficient vectors are fixed, and the encircling

mechanism is static. To address these problems, we propose a number of adaptive adjustments, such as a sigmoid-based adaptive decay function for the coefficient vectors, a leader-guided spiral motion for improved search efficiency, and a fitness-weighted dynamic encircling mechanism to adapt the search behavior according to the differences in fitness among the wolves. These enhancements are designed to achieve a better balance between exploration and exploitation, thus enhancing the convergence speed and solution accuracy of the algorithm. In addition, we introduce a Fitness Difference-Based Blending Mechanism (FDBM) to improve the position update step by taking advantage of positional differences between elite wolves. The mechanism enhances search diversity and flexibility through the incorporation of controlled randomness depending on fitness differences. Moreover, to address discrete search spaces, we introduce a sigmoid-based binary thresholding scheme, thus rendering the algorithm applicable to feature selection and other discrete optimization tasks. The suggested changes aim to strengthen the algorithm, enabling it to get out of local optima and more appropriately handle complex search problems with high dimensions.

Finally, the optimally selected features were fed into a Self-Organizing Stacked Fuzzy Neural Network (SOSFNN) [15]. This integration is crucial in dealing with the inherent uncertainty, ambiguity, and complexity found in real-world cyberattack scenarios. The continuous adaptation of self-organizing neural structures is realized by SOSFNN against the evolving network environment, such that learning attack signatures occurs at every new emergence of patterns. The stacked architecture of SOSFNN allows for multiple hierarchical layers that can represent features at different abstraction levels, capturing the subtle characteristics of various attack behaviors progressively. The designed framework is verified on two benchmark datasets- UNSW-NB15 [16] and CICIDS 2018 [17] using a five-fold cross-validation method [18]. It provides a strong evaluation by dividing each dataset into five subsets, training on four, and testing on one for every fold. The outcomes are subsequently compared with the best available IDS methods along with essential performance metrics, such as detection accuracy, false positive ratio (FPR), false negative ratio (FNR), true positive ratio (TPR), F1-score, precision, recall, the area under the ROC curve (AUC), and log loss. Apart from these measures, temporal complexity is evaluated to compare the response time of the system and understand the real-time feasibility and scalability of IDS systems.

Our key contributions can be summarized into the following five points:

1. We introduce a sigmoid-based adaptive decay function and leader-following spiral motion to enhance the exploration-exploitation tradeoff of GWO while boosting convergence rate and precision;
2. Dynamic encircling, weighted by fitness, modifies the search behavior according to differences in fitness, enhancing search diversity and evading local optima;
3. A newly developed FDBM scheme boosts the position update step by including positional differences of elite wolves, enhancing search flexibility and solution precision;
4. The identified features are analyzed using a Self-Organizing Stacked Fuzzy Neural Network (SOSFNN), enhancing adaptability and detection accuracy in dynamically changing cyberattack scenarios;
5. We validate the performance of the proposed approach on two benchmark datasets, UNSW-NB15 and CICIDS 2018, through five-fold cross-validation.

The remaining parts of the paper is structured as follows: Section 2 is related to the related work based on the previously developed intrusion detection systems and existing limitations in processing complex cyberattacks. Section 3 presents the envisaged methodology that combines the enhanced GWO with adaptive coefficient vectors, spiral motion with a leader, and fitness-dependent dynamic encircling, as well as the application of the SOSFNN. Section 4 combines the experimental setup and results. Additionally, the experimental results are presented and compared against existing IDS approaches, and the findings are discussed. Finally, the conclusion of the paper is in Section 5 along with future potential paths of research that can be followed by this research.

2 RELATED WORK

The advancement of cyber attacks and the increased complexity of network infrastructures have prompted the research community to propose sophisticated intrusion detection systems. The conventional IDS methodologies have been unable to cope with changing attack signatures. As a result, optimization techniques, bio-inspired techniques, and fuzzy neural networks have been adopted to improve detection capabilities and flexibility. Recent research has shown promising development in feature selection, hybrid

classification models, and smart optimization methods to enhance intrusion detection performance (Table 1).

For example, Srilatha and Shyam (2021)[19] proposed a novel IDS as they combined an optimal type-2 fuzzy neural network (OT2FNN) with a kernel fuzzy c-mean clustering (KFCM). The T2FNN's parameters selection process was optimized by utilizing Lion Optimization Algorithm (LOA). The proposed IDS system was proven to have better results which were a result of the simulation on the NSLKDD dataset, and these results were superior to the existing IDS systems in terms of precision, recall, and F-measure. Kunhare *et al*. (2022) [20] a genetic algorithm (GA) to find the best subsets of features from the NSL-KDD dataset. In addition, hybrid methods were used to classify data using logistic regression (LR) and decision tree (DT) models to achieve the highest detection rates (DR) and accuracies (ACC). From the experimental results, it was shown that the grey wolf optimization (GWO) yielded an accuracy of 99.44% and a DR of 99.36% after reducing the number of features from 41 down to 20. Ghanbarzadeh *et al*. (2023) [21] dealt with enhancing IDS precision using the Horse Optimization Algorithm (HOA), which showed higher adaptability in different attack patterns. However, a major challenge in their method was the use of supervised datasets, showing the importance of semi-supervised approaches to decrease the cost of label acquisition and improve practical applicability.

In the field of cloud security, Jain *et al*. (2023) [22] proposed a Fuzzy Deep Neural Network (FDNN) with the Honey Badger Algorithm (HBA), known as FDNN-HBAID, for detecting intrusions in cloud environments. Their approach was shown to have high classification accuracy using NSL-KDD and CICIDS-2017 datasets. Ninu (2023) [23] proposed an effective Mobile Adhoc Networks (MANET) intrusion detection method by a Deep Neuro-Fuzzy Network optimized by Exponential-Henry Gas Solubility Optimization (EHGSO). The proposed EHGSO approach merged the Henry Gas Solubility Optimization (HGSO) with the Exponential Weighted Moving Average (EWMA) and recognized a high recall score of 0.924 and a precision score of 0.950. Ishaque *et al*. (2023) [24] proposed a hybrid algorithm for the removal of uncertainty and the prediction of outcomes. Fuzzy logic was utilized in the process of removing uncertainty, whereas neural networks were utilized for prediction. The genetic algorithm was utilized to improve the accuracy of prediction results. Experimental results concluded

an overall detection accuracy of 99.12 %. Subramani and Selvi (2023) [25] proposed a fuzzy CNN model using spatiotemporal constraints that were induced from the Schrödinger equation and Feynman Path Integral to vastly enhance the detection accuracy and packet delivery ratio, along with decreased latency in WSNs. Wu *et al*. (2023) [26] introduced a Quantum Walks Classification Model to identify malicious activity within Cloud Computing Systems. This model utilizes Principal Component Analysis (PCA) and Quantum Walks techniques while employing a training-free clustering methodology. Experimental evaluations conducted using four Benchmark Data Sets (Accuracy Results: InSDN - 99.4%, NSL-KDD - 95.8%, UNSW-NB15 - 98%, CICIDS-2018 - 96.4%) illustrate the benefits of being able to detect and respond to attacks against SDN-based Cloud Computing Systems using this classification scheme.

According to the work by Maazalahi & Hosseini (2024) [27], the authors developed Atom Search Optimization (ASO), Equilibrium Optimization (EO) for feature selection, and also developed the Firefly Algorithm (FA) integrated with the Elitism Method for clustering. The results of the evaluation indicated that the proposed methodology produced the highest accuracy and the lowest error rates for three datasets, NSL-KDD and UNSW_NB15 KDD-CUP99, with accuracy rates of 0.998, 0.995, and 0.995, respectively. Subramanian and Chinnadurai (2024) [28] proposed a centralized approach by integrating a spatiotemporal attention network with a quantum-inspired federated averaging optimization procedure for cyber-attack detection. This model was validated on the UNSWNB15 dataset with a maximum precision of 98.2%, recall of 98.5%, f1-score of 98.35%, specificity of 98.2%, and accuracy of 98.34%. Singh *et al*. (2024) [29] proposed a Fuzzy-based teaching-learning-based Optimisation regression algorithm (F-TLBO-ID) for IDS. In this work, synthetically generated pertinent features are used to determine the concerned region's area, effective transmission range, effective sensing range, number of sensor nodes, and fading parameters. The results exhibited a strong correlation coefficient ($R = 0.84$), acceptable RMSE (36.24), and minimal bias (-7.17). Binthiya and Ravindran (2025) [30] introduced a novel Fuzzy Logic-based Intrusion Detection System with a Hidden Markov Model (FIDS-HMM) to identify malicious nodes and mitigate zero-day attacks. Moreover, an HMM was employed in the proposed protocol to monitor the energy levels of the nodes in order to detect malicious nodes effectively. The global

results concluded that the protocol improved the Quality of Service (QoS) parameters, such as packet delivery ratio, delay, and throughput, in the network with efficiency.

Table 1

Summary of Related Works Performed in Intrusion Detection

No.	Reference	Limitations Addressed	Proposed Methodology	Limitations
1	Srilatha&Shyam (2021) [19]	High computational complexity in cloud-based IDS	Kernel Fuzzy C-Means Clustering (KFCM) with Optimal Type-2 Fuzzy Neural Network (OT2FNN) using Lion Optimization Algorithm (LOA)	Limited scalability and computational efficiency
2	Kunhare *et al*. (2022) [20]	Inefficient feature selection in high-dimensional datasets	Genetic Algorithm (GA) for feature selection with hybrid classification using Logistic Regression (LR) and Decision Tree (DT)	High dependency on dataset quality and feature extraction process
3	Ghanbarzadeh *et al*. (2023) [22]	Low accuracy in IDS optimization approaches	Horse Optimization Algorithm (HOA) for improved intrusion detection	Dependency on supervised datasets and the need for semi-supervised learning
4	Jain *et al*. (2023) [23]	Privacy concerns in cloud-based IDS	Honey Badger Algorithm (HBA) in combination with Fuzzy Deep Neural Network (FDNN) with	Computational overhead due to deep learning integration
5	Ninu (2023) [24]	Intrusion Detection in MANET	Deep Neuro-Fuzzy Network optimized by Exponential-Henry Gas Solubility Optimization (EHGSO)	Increased computational complexity due to optimization methods
6	Ishaque *et al*. (2023) [25]	Uncertainty removal and prediction accuracy in IDS	Hybrid fuzzy logic and neural network with Genetic Algorithm (GA) for enhanced prediction	Requires high computational power for real-time applications
7	Subramani&Selvi (2023) [26]	Spatial and temporal constraints in IDS for WSN	Fuzzy CNN with Feynman Path Integral for spatial constraints and Schrödinger equation for temporal constraints	Increased model complexity and high training time
8	Wu *et al*. (2023) [27]	Intrusion detection in SDN-based cloud environments	Quantum Walks-based classification model with PCA and training-free clustering algorithm	Lack of interpretability of quantum transformations
9	Maazalahi&Hosseini (2024) [28]	High error rates in feature selection and Clustering for IDS	Atom Search Optimization (ASO) and Equilibrium Optimization (EO) with Firefly Algorithm (FA) for Clustering	High dependency on optimization hyperparameters
10	Subramanian&Chinnadurai (2024) [29]	Cyber-attack detection and	Spatiotemporal attention network with quantum-	Computational overhead in federated learning environments

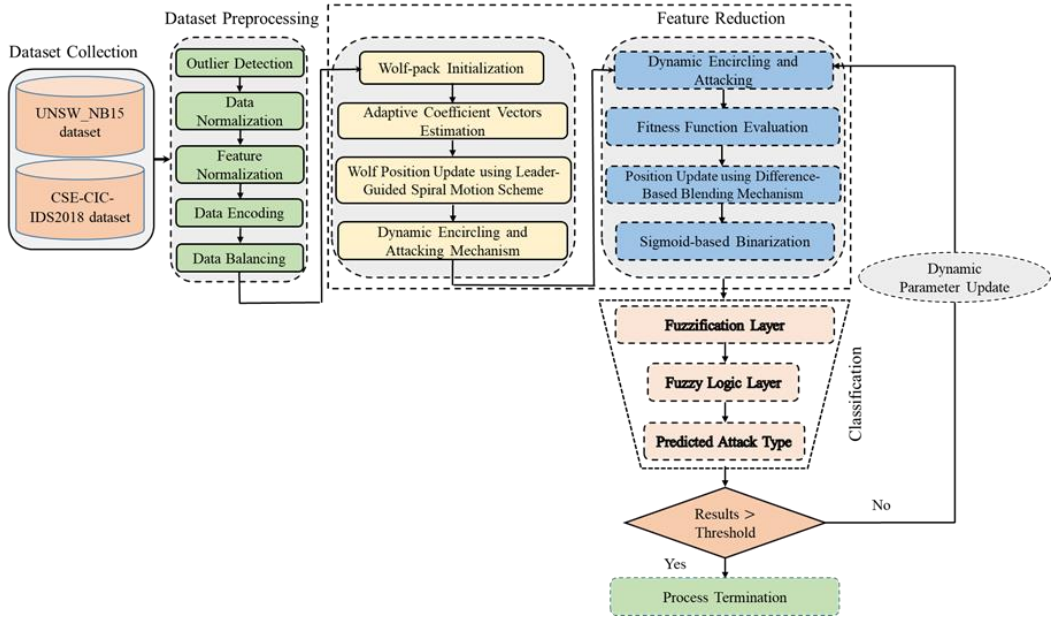
		federated learning optimization	inspired federated averaging	
11	Singh *et al.* (2024) [30]	Optimization challenges in IDS feature selection and parameter tuning	Fuzzy-based Teaching-Learning-Based Optimization (F-TLBO-ID) for regression-based IDS	Limited real-world deployment and validation
12	Binthiya&Ravindran (2025) [31]	Black hole attack detection in MANET	Fuzzy Logic-based IDS with Hidden Markov Model (FIDS-HMM) for node energy monitoring	Increased computational complexity for real-time IDS monitoring

3 PROPOSED METHODOLOGY

Our framework enhances the search efficiency and solution quality by integrating adaptive updates, spiral search, fitness-based blending, and dynamic encircling in GWO, combined with a Self-Organizing Stacked Fuzzy Neural Network (SOSFNN) for robust intrusion detection. The methodology has four key phases: (1) Data Collection, (2) Data Preprocessing, (3) Feature Reduction using Improved GWO, and (4) Classification using SOSFNN (Figure 2). These steps are briefly discussed in the subsequent subsections.

Figure 2

Flowchart of the Proposed Intrusion Classification Framework using Enhanced Grey Wolf Optimizer with Dynamic Encircling and Fitness-Based Blending



3.1 Data preprocessing

Original datasets have several redundant, missing, and irrelevant attributes that could degrade the general performance of the intrusion detection system. To ensure data quality and improve learning efficiency, a comprehensive preprocessing pipeline is employed with steps such as outlier detection, data normalization, feature encoding, and data balancing.

3.1.1 Outlier detection

There is normally anomalous data in intrusion detection datasets, as they are outside the normal distribution of traffic. These could result from packet loss, logging errors, or some form of manipulation done in an adversarial manner. Our work leverages the concept of Mahalanobis Distance (MD) [32] to detect and remove outliers where the MD of a point deviates from a multivariate mean distribution. The mathematical representation of this step is formulated in Equation 1.

$$D_M(x) = \sqrt{(x-\mu)^T \sum^{-1} (x-\mu)} \quad (1)$$

where:

- x is the feature vector of a given network traffic instance,
- μ is the mean of all instances,
- Σ is the covariance matrix of the dataset.

A threshold τ is determined using the Chi-Square distribution [33]:

$$\tau = \chi^2_{d,\alpha} \quad (2)$$

where:

d is the number of features, and α is the confidence level (0.99). Any instance with $D_M(x) > \tau$ is classified as an outlier and removed.

3.1.2 Data normalization

Since network traffic features exhibit diverse ranges (e.g., packet sizes can be in bytes while time-based features are in milliseconds), normalization is applied to bring all attributes into a uniform scale. The applied Min-Max Normalization technique [34] is defined in Equation 3**.**

$$\mathbf{X_{norm}} = \frac{\mathbf{X} - \mathbf{X_{min}}}{\mathbf{X_{max}} - \mathbf{X_{min}}} \quad (3)$$

where:

- $\mathbf{X_{norm}}$ is the normalized feature,
- $\mathbf{X_{min}}$ are the minimum and maximum values of $\mathbf{X}$ in the dataset.

This transformation ensures that all features lie within the range [0,1], facilitating faster convergence of optimization algorithms.

3.1.3 Feature encoding

We applied the One-Hot Encoding (OHE) procedure [35] to encode categorical variables such as TCP, UDP, ICMP, service type, and attack labels into respective numerical representations. This step is formulated in Equation 4.

$$\boldsymbol{OHE(X)} = \begin{cases} \mathbf{0}, & \boldsymbol{if\ the\ feature\ belongs\ to\ category\ C} \\ \mathbf{1} & \boldsymbol{Otherwise} \end{cases}$$

(4)

For instance, Table 2 below shows the Protocol Type categorically categorical feature Protocol Type with three values {TCP, UDP, ICMP} is encoded in Table 2.

Table 2

One Hot Encoding transformation of three protocols

Protocol Type	TCP	UDP	ICMP
TCP	1	0	0
UDP	0	1	0
ICMP	0	0	1

3.1.4 Data balancing

The intrusion detection datasets are typically plagued with class imbalance, in which benign traffic samples outnumber attack instances. In order to avoid model biasing toward the majority class, the Synthetic Minority Oversampling Technique (SMOTE) [36] is used to generate synthetic attack samples. The mathematical representation is given in Equation (5).

$$X_{NEW} = X_i + \lambda\,(X_j - X_i) \quad (5)$$

where:

- X_i and X_j are two randomly selected minority class samples,
- $\lambda \sim U(0,1)$ is a random interpolation factor.

After balancing the dataset, a stratified train-test split is performed with 70% for training, 15% for validation, and 15% for testing to ensure a fair evaluation.

4 FEATURE REDUCTION WITH IMPROVED WOLF-OPTIMIZATION ALGORITHM

4.1 Conventional Gray Wolf optimization algorithm

Gray Wolf Optimization Algorithm (WOA) is a novel metaheuristic technique proposed by Mirjalili *et al.* (2014) [11]. It mimics the hunting behavior and leadership ability of gray wolves. In this context, the social structure of the wolves is modeled by classifying them into four roles: alpha (α), beta (β), delta (δ), and omega (ω). The α wolf is responsible for hunting, choosing sleeping grounds, and determining when to move the pack. The second-highest wolf is the β, which assists the alpha in decision-making, reinforcement, and other activities. The β takes over as leader if the alpha gets ill or dies. The δ wolves serve as reporters to both the alpha and beta wolves and are in charge of keeping the omega wolf in line. The ω wolf warns the pack of threats or injuries and tends to the most vulnerable members of the pack. Omega wolves are lower-ranking than the others and tend to be scapegoats in the case of trouble. The hierarchical organization of the GWO can be represented in four stages: (1) Encircling prey, (2) Hunting prey, and (3) Attacking prey. The description of each stage is given as follows:

a) **Social Hierarchy:** Alpha (α) wolves are superior in the hierarchy and represent the most optimal result. Beta (β) wolves are the second best and delta (δ) Wolves are the third-best available solution. The omega (ω) wolves are deemed the last available solution. The optimum solution is acquired by a set of α, β, δ, and a group of ω wolves taking the rest of the wolves;

b) **Encircling Prey:** A wolf in the encircling stage finds the prey and tells all the other wolves to surround the prey. Because of this, other wolves need to update their locations as per the instructions. The process of prey encircling is mathematically depicted in Eqs (6) and (7).

$$\vec{D} = |\vec{V}.\vec{P}_p(t) - \vec{W}.\vec{P}(t)| \quad (6)$$

$$\vec{p}(t+1) = \vec{p}_p(t) - \vec{V}.\vec{D} \quad (7)$$

where:

t refers to the current iteration, $\overrightarrow{V}$, and $\overrightarrow{W}$ are the coefficient vectors, $\vec{p}_p(t)$ represents the positional vector of the prey, and $\vec{P}$ represents the position vector of the grey wolf. The random vector $\vec{V}$ and $\overrightarrow{W}$ are described in Eqs (8) and (9).

$$\vec{V} = 2\vec{a} * rand_1 * \vec{a} \quad (8)$$

$$\overrightarrow{W} = 2rand_2 \quad (9)$$

where:

$\vec{a}$ is a parameter, which is decreased by 2 to 0 in each cycle, and $rand_1$ and $rand_2$ represent random vectors in [0, 1].

c) Process of Hunting: In the hunting phase, α supervises β and δ. α refers to the fitness score of the most suitable position, β, and δ represent the second and third best positions, respectively. ω wolves update their position according to α, β, and δ. The hunting process is described in Eqs. (10), (11), and (12).

$$\vec{D}_\alpha = |\overrightarrow{W}_1\vec{P}_\alpha - \vec{P}| \quad (10)$$

$$\vec{D}_\beta = |\overrightarrow{W}_2\vec{P}_\beta - \vec{P}| \quad (11)$$

$$\vec{D}_\delta = |\overrightarrow{W}_3\vec{P}_\delta - \vec{P}| \quad (12)$$

where:

$\vec{D}_\alpha$, $\vec{D}_\beta$, and $\vec{D}_\delta$ are the displacement vectors of α, β, and δ wolves, respectively.

$$\vec{P}_1 = \vec{P}_\alpha - \vec{V}_1\vec{D}_\alpha \quad (13)$$

$$\vec{P}_2 = \vec{P}_\beta - \vec{V}_2\vec{D}_\beta \quad (14)$$

$$\vec{P}_3 = \vec{P}_\delta - \vec{V}_3\vec{D}_\delta \quad (15)$$

$$\overrightarrow{P}(t+1) = \frac{\vec{P}_1+\vec{P}_2+\vec{P}_3}{3} \quad (14)$$

where:

$\vec{P}_1$, $\vec{P}_2$, and $\vec{P}_3$ refer to the three initial solutions for populations. However, $\overrightarrow{P}(t+1)$ is not useful for feature selection because it is a discrete optimization problem. Therefore, a discrete and efficient variant of the GWO algorithm is required.

d) Attacking Prey: Grey wolves only attack their prey when they have stopped moving. In the attacking phase (exploitation), $|A| < 1$, the wolves are compelled to attack their prey. During the iteration, $\vec{a}$ is reduced from 2 to 0 at a random value within the range [2a, −2a]. In searching for prey (exploration), every solution (wolf) changes its location based on the location of α, β, δ. When $|A| > 1$ or $|A| < 1$, then avoid the prey and attempt to get a better one.

4.2 Improved wolf optimization algorithm with fitness difference-based blending mechanism

This section outlines the Modified Grey Wolf Optimizer (MGWO) and how it is combined with the Fitness Difference-Based Blending Mechanism (FDBM) to improve the process of solution search. MGWO brings in adaptive compensation, perturbation-based exploration, and mutation-based fine-tuning to enhance the tradeoff between exploration and exploitation. Following the production of candidate solutions through MGWO, FDBM is utilized to blend the best-performing solutions and thus produce better solutions and hasten convergence. The detailed working of MGWO is discussed in the subsequent subsections:

4.2.1 Position update in modified Grey Wolf optimizer

The update mechanism of position in GWO is inspired by the hunting process of grey wolves, in which the alpha, beta, and delta wolves are the leaders in search. The position of every wolf at iteration is updated using (15).

$$\vec{P}(t+1) = \vec{P}(t) + V_1(W_1.\vec{P}_\alpha - \vec{P}_t) + V_2(W_2.\vec{P}_\beta - \vec{P}_t) + V_3(W_3.\vec{P}_\delta - \vec{P}_t) \quad (15)$$

where

- $\vec{P}(t)$ refers to the current position of the wolf at iterations t.
- $\vec{P}_\alpha, \vec{P}_\beta, \vec{P}_\delta$ shows the positions of the α, β, δ wolves, respectively.
- V_i and W_i are adaptive coefficients, computed using Eqs (8) and (9).

4.2.2 Adaptive coefficient vectors for enhanced exploration and exploitation

In the traditional GWO, the coefficient vectors V and W are set by random values, and the parameter a, which linearly diminishes from 2 to 0 along iterations. Nevertheless, this constant decrease may lead to premature convergence or poor exploration, particularly in high-dimensional search spaces. We therefore propose an adaptive modification of the coefficient vectors with a sigmoid-based decay function:

$$V = 2ar_1\left(\frac{1}{1+e^{-k(\frac{t}{T}-0.5)}}\right) \quad (16)$$

$$W = 2ar_2\left(\frac{1}{1+e^{-k(\frac{t}{T}-0.5)}}\right) \quad (17)$$

where:

- r_1 and r_2 are random values in the range [0, 1]
- k = Steepness parameter controlling the rate of decay
- t = Current iteration
- T = Maximum number of iterations

4.2.3 Hybrid search strategy using leader-guided spiral motion

To improve the search effectiveness and avert the algorithm from getting trapped in local optima, we incorporate a leader-directed spiral movement based on the search behavior of marine predators. The new position of a wolf during the hunting process is calculated as:

$$P_i(t+1) = P_\alpha(t) + B.e^{c\theta}.Cos(\theta) \quad (18)$$

where:

B = Constant defining the spiral size
c = Convergence factor controlling the spiral tightness
θ= Angular component increasing with each iteration

This spiral-shaped search mechanism enhances local exploitation without losing effective exploration, thus enhancing convergence precision and preventing premature stagnation.

4.2.4 Dynamic encircling and attacking mechanism

In the traditional GWO, an encircling mechanism is based on static distance-based encircling, which may cause poor convergence in high-dimensional spaces. For improvement, we propose a fitness-weighted distance calculation that adaptively adjusts the encircling mechanism according to the difference between the best and worst wolves' fitness:

$$D_i = |W_i.P_{best}(t) - P_i(t)|.(1 + \frac{f_{best} - f_i}{f_{best}}) \quad (19)$$

where:

f_{best} represents best fitness value and f_i indicates the fitness value of the current wolf computed using Eq. (20).

$$f(x) = \alpha . Class_{Error} + \beta . \frac{No.of\ selected\ features}{Original\ features} \quad (20)$$

where:

$Class_{Error}$ refers to errors in classification accuracy, and α and β are the weights balancing the tradeoff between classification accuracy and selected features. The sum of both weights should be equal to 1. This correction raises the impact of the dominant wolves, enhancing the search process to be more adaptive and enhancing the algorithm's capacity to escape the trap of local optima.

4.2.5 Enhanced position update using the difference-based blending mechanism

In the conventional GWO, the position of a new wolf is usually calculated as the weighted mean of the positions of the alpha (P_α), beta (P_β), and delta (P_δ) wolves. Although this mechanism facilitates convergence to promising areas of the search space, it can restrict the diversity of candidate solutions and enhance the likelihood of premature convergence, especially in high-dimensional, complex search spaces. To overcome this drawback, we propose a new Fitness Difference-Based Blending Mechanism (FDBM), which improves the position update process by including the positional differences among the elite wolves (α, β, and δ). This mechanism allows adaptive exploration and exploitation by combining the advantages of elite-based learning with difference-based learning.

$$P_i(t+1) = w_1 P_\alpha + w_2 P_\beta + w_3 P_\delta + \lambda D_{blend} \quad (21)$$

where:

$$w_1 = \frac{f_\alpha}{f_\alpha + f_\beta + f_\delta}, w_2 = \frac{f_\beta}{f_\alpha + f_\beta + f_\delta}, w_3 = \frac{f_\delta}{f_\alpha + f_\beta + f_\delta} \quad (22),$$

- $P_i(t+1)$ is the updated position of the i^{th} wolf at iteration $t+1$.

The rightmost added term, λD_{blend} leverages the positional differences between the elite wolves to inject controlled randomness and augment search diversity. This component is termed as:

$$D_{blend} = \eta_1.(P_\alpha - P_\beta) + \eta_2.(P_\beta - P_\gamma) + \eta_3.(P_\gamma - P_\alpha) \quad (23)$$

where:

- η_1, η_2, η_3 = Random blending factors in the range [0,1] to introduce variability and adaptability in the search.
- λ = Blending intensity factor that controls the contribution of the difference-based term, ensuring a balanced tradeoff between exploration and exploitation.

The blending component enables the search to dynamically adjust according to the disparities among the elite wolves' locations, motivating the algorithm to explore novel regions of the solution space without neglecting good-performing areas.

4.2.6 Discrete mapping for feature selection

Feature selection is an optimization problem with a discrete nature; hence, the updated positions of the wolves need to be mapped to discrete values. We propose a sigmoid-based binary thresholding [37] approach to transform continuous position values into binary values:

$$P_i(t) = \begin{cases} 1, & if\ \sigma(P_i(t)) > 0.5 \\ 0, & Otherwise \end{cases} \quad (23)$$

where

$$\sigma(P_i(t)) = \frac{1}{1+e^{-P_i(t)}} \quad (24)$$

This mapping allows the algorithm to handle binary search spaces, making it suitable for feature selection tasks. The detailed work of the improved GWO is illustrated in Figure 3. The pseudocode for the proposed feature selection is given in Algorithm 1.

Figure 3

Flowchart of the Improved Wolf Optimization Algorithm with Fitness Difference-Based Blending Mechanism

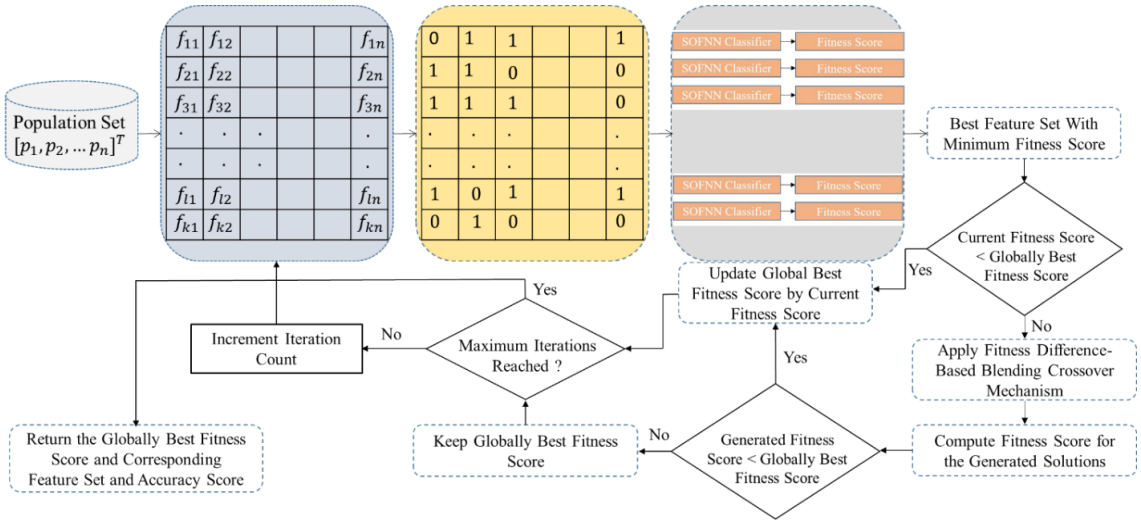


Chart 1

Algorithm 1. Pseudocode of the proposed Improved Wolf Optimization Algorithm with FDBM

BEGIN Improved Wolf Optimization Algorithm (IWOF-DBM)

1. **Initialize Parameters**

- N = Number of wolves, T = Maximum number of iterations, a = Control parameter (initially set to 2),
- B, c = Spiral motion constants, λ = Blending intensity factor, k = Sigmoid steepness parameter, α, β= Tradeoff weights for classification and feature reduction
- Initialize the position of wolves P_i randomly in the search space

FOR each wolf i

Compute fitness f_i using Eq. (20)

END FOR

- Identify α, β, δ wolves based on best fitness values

FOR t = 1 to T

a = 2 - $(2 * \frac{t}{T})$

FOR each wolf i

Compute Adaptive Coefficient Vectors

$V = 2ar_1(\frac{1}{1+e^{-k(\frac{t}{T}-0.5)}})$

$W = 2ar_2(\frac{1}{1+e^{-k(\frac{t}{T}-0.5)}})$

- Position Update Using Modified Grey Wolf Optimizer (MGWO)

$\vec{P}(t+1) = \vec{P}(t) + V_1(W_1.\vec{P}_\alpha - \vec{P}_t) + V_2(W_2.\vec{P}_\beta - \vec{P}_t) + V_3(W_3.\vec{P}_\delta - \vec{P}_t)$

- Spiral-Based Leader-Guided Search

  θ = Random angle

  $P_i(t+1) = P_\alpha(t) + B.e^{c\theta}.Cos(\theta)$

- Fitness-Weighted Encircling and Attacking

  $D_i = |W_i.P_{best}(t) - P_i(t)|.(1 + \frac{f_{best} - f_i}{f_{best}})$

- Fitness Difference-Based Blending Mechanism (FDBM)

  $w_1 = \frac{f_\alpha}{f_\alpha + f_\beta + f_\delta}, w_2 = \frac{f_\beta}{f_\alpha + f_\beta + f_\delta}, w_3 = \frac{f_\delta}{f_\alpha + f_\beta + f_\delta}$

  $D_{blend} = \eta_1.(P_\alpha - P_\beta) + \eta_2.(P_\beta - P_\gamma) + \eta_3.(P_\gamma - P_\alpha)$

  $P_i(t+1) = w_1P_\alpha + w_2P_\beta + w_3P_\delta + \lambda D_{blend}$

  $\sigma(P_i(t))$= 1 / (1 + exp(- $P_i(t)$))

  IF $\sigma(P_i(t)) > 0.5$ ***THEN***

  $P_i(t)$=1

  ELSE

  $P_i(t)$= 0

  END IF

END FOR

Evaluate fitness f_i for each updated position

Update α, β, and δ wolves based on best fitness values

END FOR

Return position and fitness of the alpha wolf

END

4.3 Classification using Self-Organizing Stacked Fuzzy Neural Network

The Self-Organizing Stacked Fuzzy Neural Network (SOSFNN) includes a Self-Organizing Map (SOM) [38] to improve classification by topological Clustering and adaptive learning. SOM is an unsupervised learning algorithm that maps high-dimensional input data onto a lower-dimensional representation in a way that maintains feature relationships. It enables the network to adaptively change its structure according to the distribution of chosen features derived from the Improved Wolf Optimization Algorithm.

For an input feature vector $X = [x_1, x_2, .. x_n]$, the SOM arranges neurons in a grid-like topology, where each neuron i has an associated weight vector $W_i = [w_{i1}, w_{i2}, \dots, w_{1,n}]$. The best-matching unit (BMU) is determined by minimizing the Euclidean distance between the input vector and the neuron weight vectors using Eq. (24)

$$BMU = \arg\min_i ||X - W_i|| \quad (24)$$

After the BMU is determined, weight updates are done through a neighborhood function $h(i, BMU)$, which ensures that not only the BMU but also its surrounding neurons are adapted. The weight update rule is defined as

$$W_i(t+1) = W_i(t) + \eta h(i, BMU)(X - W_i(t)) \quad (24)$$

where:

η is the learning rate, and $h(i, BMU)$ is a Gaussian neighborhood function [39], defined as

$$h(i, BMU) = e^{-\frac{||r_i - r_{BMU}||^2}{2\sigma^2(t)}} \quad (25)$$

where:

(r_i - r_{BMU}) reflects the Euclidean distance between the currently selected feature vector and the previous best feature set.

This adaptation makes the neurons adapt to the data distribution, and clusters are formed, representing different attack patterns. After structuring the data with the SOM, the correlation-based fuzzy rule generation module uses the mapped feature clusters to form fuzzy rules. The input x_i is converted to a fuzzy variable using the membership function defined in Eq. (26).

$$\mu_{A_i}(x_i) = e^{-\frac{(x_i - c_i)^2}{2\sigma_i^2}} \quad (26)$$

where:

c_i and σ_i represent the cluster center and dispersion, respectively. The correlation between features x_i and x_j determines rule strength as

$$\sigma(x_i, x_j) = \frac{\sum(x_i - \bar{x}_i)(x_j - \bar{x}_j)}{\sqrt{\sum(x_i - \bar{x}_i)^2}\sqrt{\sum(x_j - \bar{x}_j)^2}} \quad (27)$$

Rules are generated where correlation exceeds a threshold (ι), defined as:

$$IF\ x_i \text{ is } A_i \text{ AND } x_j \text{ is } A_j \text{ THEN } y_k \text{ is } B_k \quad (28)$$

The self-organizing hierarchical structure further refines classification through stacked neural layers. Each layer processes extracted fuzzy-enhanced features using Equation (29);

$$h_{l+1} = \sigma(W_l h_l + b_l) \quad (29)$$

where:

W_l and b_l are weight and bias parameters, and σ is the Relu activation function.

$$\sigma(x) = \max(0, x) \quad (30)$$

The classification decision is made based on a Softmax activation function using the deepest stacked layer output of the SOSFNN, which is a linearly separable and highly fine-grained feature vector.

$$P(y = k|X) = \frac{e^{W_k h_{final} + b_k}}{\sum_j e^{W_j h_{final} + b_j}} \quad (31)$$

where

$P(y = k|X)$ represents the probability of class k given the input X, and w_k and b_k are the classification layer's trainable parameters.

5 RESULTS AND DISCUSSION

This section analyzes the performance of the proposed intrusion detection framework. Initially, the Q-WOA selected the most relevant features, which were then

classified using SOSFNN. The model is evaluated on benchmark intrusion detection datasets and compared with recently published studies.

5.1 Experimental setup

To facilitate the implementation of and determine the effectiveness of the proposed Self-Organizing Stacked Fuzzy Neural Network (SOSFNN) in detecting intrusions through an experimental design consisting of hardware and software configurations, two separate configurations were created to allow for optimal training/testing of the proposed SOSFNN model. The hardware configuration consisted of a high-performance computer built on an Intel Core i9-13900K processor, running on 64 GB of RAM and an NVIDIA RTX 4090 GPU for quick data processing capabilities. To provide a high-performance data storage capability for the training/testing of the model, each of the hardware configurations contained a 1 TB NVMe solid-state drive (SSD). In addition, the operating system was installed as Ubuntu 22.04 LTS in order to provide a reliable operating environment and so that the system has the ability to continue support for deep-learning frameworks for the foreseeable future.

The software configuration was established so that the main programming language used to write code for the SOSFNN model was Python 3.10. The trained model was executed using TensorFlow 2.12 and Keras, while other modules of the SOSFNN were created using the SciPy and Skfuzzy libraries. To provide reproducibility of the software configurations, Docker containers were utilized during the implementation of the SOSFNN, while all software dependencies were managed by Anaconda. The SOSFNN model training and hyperparameter tuning were accomplished with CUDA 12.1 and cuDNN to take advantage of the accelerated processing capabilities of the GPUs.

5.2 Dataset description

Two publicly available datasets, (1) UNSW_NB15 [16] and (2) CICIDS-2018 [17], are used to demonstrate the effectiveness of the proposed intrusion detection model. The crisp details of these datasets are given below.

5.2.1 UNSW_NB15 dataset

UNSW-NB15 was created from the IXIA PerfectStorm tool by the UNSW Canberra Cyber Range Lab to simulate a hybrid of contemporary artificial attack behavior and real-world normal behavior. Raw network traffic amounting to 100 GB of Pcap files was collected with the aid of the tcpdump tool and involves nine classes of attacks, i.e., Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode, and Worms. Argus and Bro-IDS tools were utilized, in conjunction with twelve algorithms, to mine 49 features with class labels, described in the UNSW-NB15_features.csv file.

5.2.2 CICIDS-2018 dataset

CICIDS-2018 consists of seven attack scenarios: brute force, Heartbleed, Botnet, DoS, DDoS, Web attacks, and network intrusion. The attacking infrastructure consists of 50 machines, and the victim organization consists of five departments, comprising 420 machines and 30 servers. The dataset comprises each machine's captured network traffic and system logs, and 80 features extracted from the captured traffic with CICFlowMeter-V3. These data were selected based on the specifications of the attack cases, the forms of attacks used within these pages, and under which conditions this network is functional.

5.3 Performance evaluation and comparative analysis

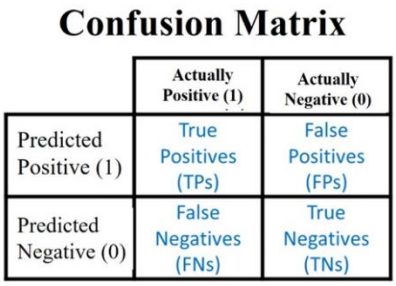
We compared the performance of our algorithm with existing state-of-the-art approaches in terms of ten measures: (1) Confusion Matrix, (2) Average Classification Accuracy (ACC), (3) Precision, (4) Recall, (5) F1-score, (6) False Positive Rate, (7) Fitness Score, (8) False Negative Rate (FNR), (9) Area Under the Curve (AUC), (10) Feature Reduction Rate. The description of these metrics is discussed in the subsequent points:

[1] Confusion Matrix: The confusion matrix provides an organized manner for assessing the classification performance through comparisons of actual versus predicted class labels (Figure 4) [39]. It defines the four most important metrics: True Positives (TP), which are accurate instances of a particular class; False

Positives (FP), in which instances of other classes are classified as the particular class; True Negatives (TN), which are accurate instances not belonging to the particular class; and False Negatives (FN), in which instances of a particular class are classified as another.

Figure 4

The layout of the confusion matrix for the bi-classification problem



[2] **Average Classification Accuracy:** ACC is an important performance measure that shows the discrimination ability of the applied classification approach [40]. It is computed as:

$$ACC = \frac{TP+TN}{TP+TN+FP+FN} \quad (32)$$

[3] **Precision:** Precision is the ratio of the number of positive instances correctly predicted to the total number of instances that were predicted as positive [41]. Precision measures how reliable the model's positive predictions are and is quite crucial in situations where false positives must be avoided at all costs. Mathematically, it is represented as:

$$Precision = \frac{TP}{TP+FP} \quad (33)$$

High Precision ensures that there are fewer false positives and that the model can identify positive cases accurately with little misclassification.

[4] Recall, Sensitivity, or True Positive Rate (TPR), calculates the model's capacity to identify real positive instances correctly [41]. It measures how many of the true positive instances were accurately picked up by the classifier. Mathematically, it is represented as:

$$Recall = \frac{TP}{TP+FN} \tag{34}$$

A high Recall value means that the model successfully picks up most of the real positive cases with a low rate of false negatives.

[5] F1-Score: The F1-score is the harmonic mean between Precision and Recall, which helps to balance the false positive-false negative tradeoff [41]. F1-score comes in handy whenever class distributions are uneven, with the aim that precision and recall should both be equally weighted when arriving at a final score. F1-score is given as:

$$F = \frac{2*Precison*Recall}{Precison+Recall} \tag{35}$$

A high F1 score implies a robust classification model that works well to label positive instances correctly and avoids most misclassifications. It is an important measure of performance when assessing the robustness of intrusion detection systems.

[6] False Positive Rate: The False Positive Rate measures the rate of negative samples wrongly labeled as positive [42]. A smaller FPR signifies a better classifier with fewer false alarms. It is calculated as:

$$FPR = \frac{FP}{FP+TN} \tag{36}$$

The Fitness Score measures the optimization performance of feature selection in balancing classification accuracy and feature subset size. It aids in finding the best tradeoff between dimensionality reduction and classification performance. The precise formulation relies on the optimization function applied to the algorithm.

[7] **False Negative Rate:** The False Negative Rate is the ratio of the actual positive instances that are predicted as negative, reflecting the model's inability to identify true attacks [43]. It is represented by:

$$FNR = \frac{FN}{FN+TP} \tag{37}$$

Lower FNR indicates a detection-strong model.

[8] **Area Under the Curve (AUC):** AUC is the area under the Receiver Operating Characteristic (ROC) curve, which captures the model's discrimination power between positive and negative samples [44]. AUC scores range between 0 and 1, with higher AUC reflecting better classification performance regardless of threshold values;

[9] **Feature Reduction Rate (FRR):** This measure evaluates the performance of the feature selection operation by measuring the percentage of features removed with the preservation of classification accuracy [45]. It is calculated as:

$$FRR = 1 - \frac{Number\ of\ selected\ features}{Total\ number\ of\ features} \tag{39}$$

We implemented a five-fold cross-validation methodology to divide the datasets into training, validation, and testing sets to estimate model performance [18]. We divided the data into five equal-sized sections or folds; we then used four of the folds as a training and validation set and assigned one fold to use as a test set. In particular, 80% of the data (the four training and validation folds) and approximately 8% of the complete data set (the part set aside for validation during training) were allocated for use as the validation set during training (i.e., 10% of the training set).

5.4 Hyperparameter tuning

Hyperparameter tuning is a necessary step for fine-tuning the proposed QETC-WOA algorithm and SOSFNN. The population size, tunneling probability, energy threshold, and learning rate are tuned as key parameters to achieve exploration-

exploitation balance for effective feature selection and classification. The SOM neighborhood function, neuron pruning threshold, and fuzzy rule weights are tuned to enhance adaptability and decision-making. A grid-based hyperparameter tuning approach [46] was implemented to compute these optimal values. The detailed list of hyperparameters used in our work is tabulated in Tables 3 and 4.

Table 3

Details of hyperparameters used in the improved GWO algorithm-based feature selection approach

Parameter	Description	Range
Population Size (N)	Number of wolves in the population. Affects the exploration and exploitation balance.	20 – 100
Maximum Iterations (T)	Number of iterations for the algorithm to run.	100 – 1000
Adaptive Coefficient Steepness	Controls the rate of decay in adaptive coefficients.	0.1 – 5
Random Coefficient (r_1, r_2)	Random values that adjust the influence of alpha, beta, and delta wolves.	[0, 1]
Convergence Factor (c)	Controls the tightness of the spiral motion.	0.01 – 1
Spiral Size Constant (B)	Defines the amplitude of the spiral-based search.	0.1 – 2
Blending Intensity Factor (λ)	Controls the contribution of the blending term in the position update.	0.1 – 1
Blending Factors (η_1, η_2, η_3)	Adjusts the influence of position differences between α, β, and δ wolves.	[0, 1]
Classification Error Weight (α)	Weight for classification accuracy in the fitness function.	0.1 – 1 (α + β = 1)
Feature Selection Weight (β)	Weight is used for feature reduction in the fitness function.	0.1 – 1 (α + β = 1)
Binary Threshold Value	Sigmoid threshold for converting continuous positions to binary values.	0.5
Learning Rate for Position Update	The rate at which the wolves adjust positions.	0.01 – 0.1

Table 4

Details of hyperparameters used in SOSFNN classifier

Parameter	Symbol	Description	Tuning Strategy
Learning Rate	η	Controls weight updates in SOM.	0.007
Neighborhood Function Decay	σ	The decay rate of the Gaussian neighborhood function.	1 (Initialization)
Feature Clustering Threshold	ζ	Correlation threshold for rule generation.	0.66
Neuron Significance	$\psi\ (N_j)$	Measures the importance of neurons.	0.5
Activation Function	Relu	Function used in stacked layers.	Relu $(\sigma(x))$ =$\max(0, x)$
Error Threshold	θ	The threshold for adding new neurons is when the local error exceeds it.	0.05

Rule Weighting Factor	w_k	Importance of fuzzy rules in classification.	Optimized using training data statistics.

5.5 Baseline research studies used for results comparison

The average performance of the proposed approach is compared with five recently published articles: (1) Convoluted Bi-LSTM model, (2) Xg-Boost with machine learning classifiers, (3) SMOTE with ML classifiers, (4) PCA with ML techniques, and (5) Positional Embedding with Autoencoder transformation scheme. In Convoluted BiLSTM architecture, Hnamte and Hussian (2023) [47] integrated deep CNN with a bi-directional LSTM model to extract spatiotemporal features from streamed network packets. The model has been trained with real-time traffic datasets, CICIDS-2018, and Edge_IIoT datasets. The performance of the model is investigated using multiclass classification, and a 100% and 99.64% accuracy rate was achieved, respectively, when trained and tested with the datasets.

In the second experiment, Talukder *et al*. (2024) [48] incorporated SMOTE for data balancing, XGBoost for important feature selection, and multiple classification approaches. The proposed architecture produced excellent results when tested on two datasets, KDDCUP'99 and CIC-MalMem-2022, with an accuracy of 99.99% and 100% for KDDCUP'99 and CIC-MalMem-2022, respectively. In the third study, Asif *et al*. (2022) [49] merged the MapReduce approach with artificial neural networks for Intrusion Detection to automate intrusion detection procedures. The resultant security system's detection accuracy is 97.6% in training and 95.7% in validation on the NSL-KDD dataset. Saheed *et al*. (2022) [50] used principal component analysis (PCA) and six ML classifiers for discriminating true and attack network packets. The final IDS model realized a 99.99% classification accuracy on the UNSW-NB15 dataset. As presented in Wu and others [51] (2022), a position embedding method has been proposed for linking sequential data among attribute dimensions; a type of stacked encoder-decoder NN was used to extract low-dimensional representations of high-dimensional raw data; the model utilized self-attention techniques to classify network traffic types. These models were evaluated against two publicly available datasets called CICIDS-2017 and CIC DDoS 2019,

resulting in F1 scores of 99.17% & 98.48%, respectively. A comparison between their performance on these datasets is provided in Table 1 below.

5.5.1 Performance on Dataset 1 (UNSW_NB15 Dataset)

The proposed methodology has produced excellent classification results across all classes of the UNSW-NB15 Dataset and achieved an overall average accuracy of 99.48% (refer to Figure 5). The high precision (0.9643) and recall (0.9947) scores reflect the model's capability to reduce false negatives and false positives regardless of which attack classes are used to measure them. The F1-score of 0.9790 indicates that the system has been able to establish itself as robust in distinguishing malicious from legitimate traffic. The accuracy for Class 0 (Normal Traffic) is 99.35% and the system is able to accurately separate benign activity from attacks. The accuracy for Class 8 (Attack Category 8) is 100%, meaning that every instance in this class was perfectly classified. While the impressive classification rates shown above suggest that further investigation of the potential for bias within the dataset is warranted, Classes 1 and 7 showed slightly lower precision (0.9192 and 0.9349, respectively), which indicates that some amount of misclassification occurred relative to the other attack types due to feature overlap. The FPR is less than 0.0011 for every class, keeping benign traffic exceptionally infrequently classified as an attack, and the FNR is, on average, 0.0052, verifying little missed detection (Table 5). The analysis of the confusion matrix also proves these findings, as the percentage-based confusion matrix indicates a high diagonal dominance, and hence, most classifications are accurate. Class 5, for example, has a 99.38% accuracy, with only 0.62% misclassification, to ensure that the model accurately labels intricate attack patterns. The low rates of misclassification between attacks establish the system's capability to identify subtle patterns in network activity to minimize indecisiveness in decision-making.

The high recall measures (over 0.99 for all except Class 1, which is 0.9918) prove the strong ability of the model in the detection of attack traffic without the presence of huge false negatives to ensure that constantly changing cyber threats are well detected. The plot of the fitness function further explains how the number of chosen features (x-axis) correlates with the resultant classification accuracy (y-axis) (Figure 6). In the

beginning, as the number of chosen features is more (48 features), the accuracy of the model is also moderate at 44.18%, suggesting that the occurrence of redundant or less useful features degrades the performance of the model. With the increased refinement of feature selection, the accuracy is greatly enhanced. As the number of features decreases to 42, the accuracy in classification improves to 66.33%, pointing out that the removal of irrelevant features improves decision boundaries. A further decrease to 33 features leads to an accuracy of 71.08%, validating that dimensionality reduction creates a more efficient and discriminative feature space. As the number of chosen features diminishes to 24, accuracy increases to 79.56%, indicating that the most representative attributes start influencing the learning process.

The most profound improvements were achieved when there were only 19 features used, which produced an improved classification accuracy of over 91.06%, allowing the model to be able to classify the attack patterns using very few features. The model's accuracy improved significantly (96.42%) for 16 features, which confirmed that a smaller number of features was all that was needed to provide a high-performance Intrusion Detection System. Finally, the model produced impressive classification accuracy (99.48%) with 14 features extracted, which demonstrated that the model based on the proposed feature selection technique can extract the most discriminative features, resulting in a near-perfect classification accuracy.

The training/validation accuracy curves (as seen in the left plot of Figure 7) indicate that as each epoch passes, the performance is gradually improving over time due to the continued improvement in identifying patterns from training data. In other words, the model continues to recognize patterns as more examples are presented. During the first 40 epochs, there was a quick increase in the accuracy of training and validation, reaching about 80% accuracy within 40 total epochs. After this point, the amount of increase levels off, and both training/validation accuracies reach very close to 99%. This indicates that there was no overfitting of the model to the training set and that there is good ability of the model to generalize to the validation (or unseen) set. The small difference between training/validation accuracy curves at the end indicates that this model can generalize well and remain robust on new examples (i.e., unseen examples). The training and validation loss curve (Figure 7, right plot) demonstrates a regular downward trend in loss values, with training and validation losses decreasing drastically within the

first 50 epochs. The loss function is around 1.75 and below 0.25 at the progression of training. Validation loss tracks with training loss tightly, which reaffirms that there is no overfitting in the model but rather an effective reduction of error in unseen samples. The slight volatility in later epochs shows that the optimizer fine-tunes the parameters of the model while balancing the stability of learning and flexibility toward intricate patterns.

Figure 5

Confusion matrix for the UNSW-NB15 dataset, showing the classification performance of the proposed model. In the left figure, the confusion matrix shows classwise counts (numbers), while the confusion matrix shown in the right visualizes the classwise counts (in %)

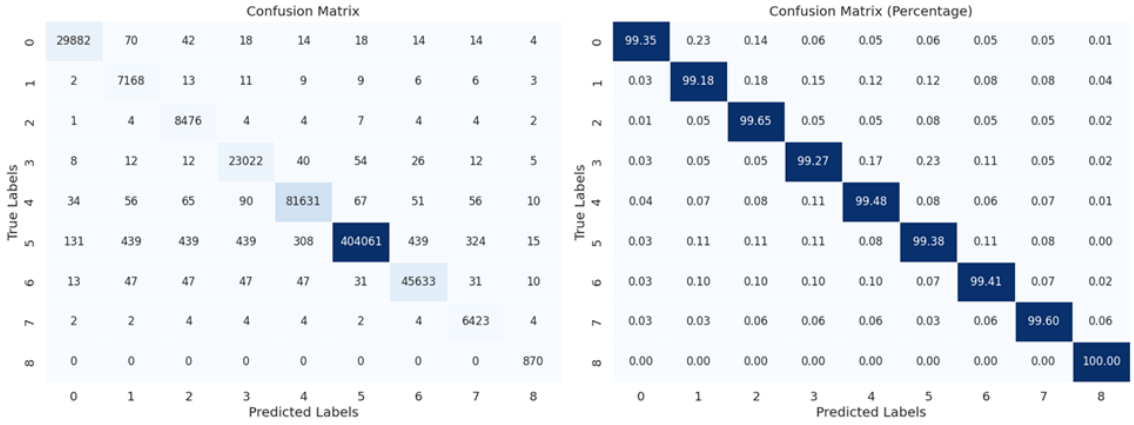


Table 5

Performance evaluation using different metrics on the UNSW-NB15 dataset

	Accuracy (%)	Precision	Recall	F1-score	FPR	FNR	Fitness Score
Class 0	99.35	0.9936	0.9935	0.9935	0.00032	0.00645	0.80375
Class 1	99.18	0.9192	0.9918	0.9541	0.00104	0.00816	0.80375
Class 2	99.65	0.9316	0.9964	0.9629	0.00103	0.00352	0.80375
Class 3	99.27	0.9740	0.9927	0.9833	0.00104	0.00728	0.80375
Class 4	99.48	0.9948	0.9947	0.9947	0.00080	0.00522	0.80375
Class 5	99.38	0.9995	0.9937	0.9966	0.00092	0.00623	0.80375
Class 6	99.41	0.9882	0.9940	0.9911	0.00096	0.00594	0.80375
Class 7	99.60	0.9349	0.9959	0.9644	0.00074	0.00403	0.80375
Class8	100	0.9425	1.0000	0.9704	0.00008	0.00000	0.80375
Average	99.48	0.9643	0.9947	0.9790	0.00077	0.00520	0.80375

Figure 6

Visualization of the fitness function in terms of the number of selected features and corresponding classification accuracy over 150 epochs

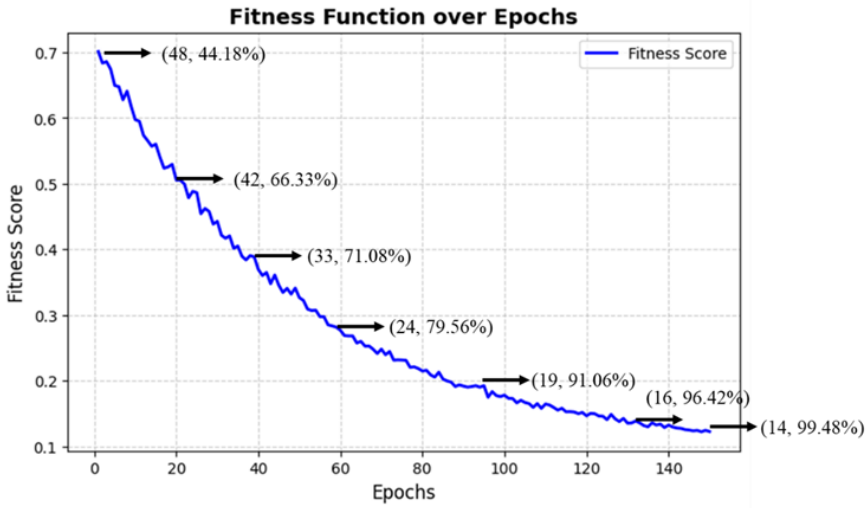
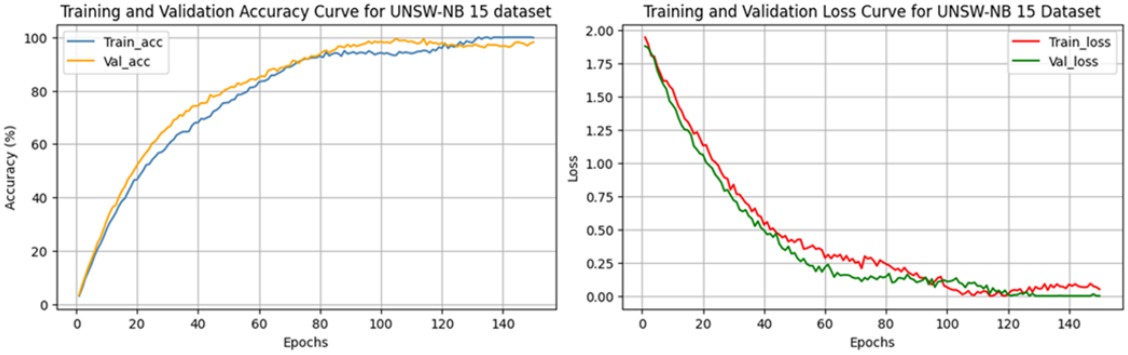


Figure 7

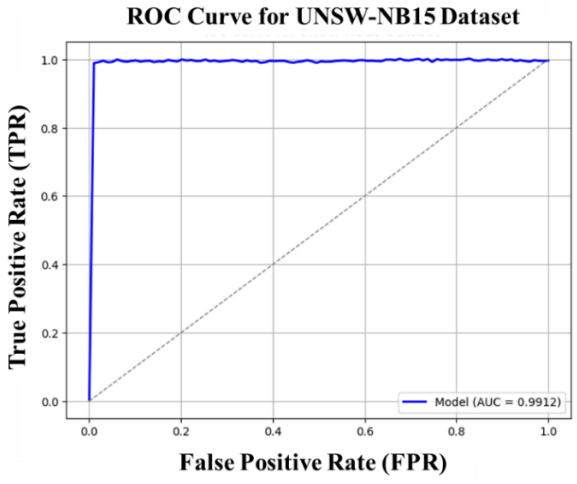
Training and validation accuracy curve (left) and loss curves for the UNSW-NB Dataset



The ROC curve of the UNSW-NB15 dataset measures how well the model can classify attack versus normal traffic (Figure 8). The false positive rate is plotted on the x-axis, and the true positive rate is on the y-axis. The diagonal line shows a random classifier with zero predictive ability. The ROC curve of the model remains close to the top-left corner, which implies high classification capability. A value of 0.9912 for AUC indicates that the detection ability is high, while false positives are minimal. The almost horizontal top curve indicates consistent performance across different thresholds. These outcomes validate the efficiency of the model in intrusion detection.

Figure 8

ROC curve for UNSW-NB15 Dataset



5.5.2 Performance on Dataset 1 (CICIDS-2018 Dataset)

The classification outcomes of the suggested intrusion detection model on the CICIDS-2018 dataset are high-performance. The confusion matrix shows that the model successfully discriminates between various types of network attacks with a high overall true positive rate and few misclassifications. The percentage matrix ensures that the majority of the attack categories are categorized with more than 95% correctness, albeit Class 1 (DDoS) has relatively poorer accuracy owing to its proximity to other types of attacks (Figure 9). The statistical performance measures also ensure the strength of the model. The overall accuracy of the model stands at 98.22%, which indicates its performance in classifying both normal and attack traffic (Table 6). The precision, recall, and F1-score for each class show a well-balanced precision and recall with an average precision of 94.92%, recall of 94.48%, and F1-score of 94.67%. At the class level, Class 3 (DoS) has the highest recall of 98.08% with the least false negatives, and Class 4 (Normal) has the highest precision of 97.30% with the least false positives. Yet, Class 1 (DDoS) has a slightly lower recall (85.48%), which reflects a higher false negative rate (14.52%), and this could be due to overlapping feature distributions between DDoS and DoS traffic. The average FPR stands at 1.14%, meaning that the model has a relatively low false identification of benign traffic as an attack. In addition, the FNR is just higher at 5.52%, indicating a limited percentage of attacks are going to be left unidentified. The

best FNR is recorded in Class 1 (DDoS) at 14.52%, consistent with its lower recall score. The fitness score of the model is constant at 1.32 in all the classes, which reflects consistent performance on the basis of feature selection as well as classification. The validation and training accuracy and loss curves for 200 epochs further establish the efficiency of the model (Figure 10). The accuracy curves show a very steep improvement in classification performance within the first 50 epochs, followed by smooth convergence with around 98% accuracy. The accuracy of validation tracks the training accuracy closely, verifying that the model learns well without extreme overfitting. Similarly to the learning curve, there were immediate drops in the fitness curve before stabilising near zero, indicating that the algorithms had good learning and converged very quickly to a solution.

It is evident how the feature selection succeeded in enhancing the Grey Wolf Optimization Algorithm, as shown in Figure 11 of the fitness function. The number of features selected reduces as iterations increase, while classification accuracy increases. First, for 72 features, the accuracy is 40.09%, whereas with the optimized feature set, the model achieves 98.22% accuracy using 19 features alone. This reduction in the number of features, without any tradeoff in the classification performance, reflects the power of the new approach in enhancing computational efficiency as well as detection accuracy. The ROC curve for the CICIDS-2018 dataset indicates the performance of the model in intrusion detection (Figure 12). The curve is close to the top-left, reflecting high classification accuracy. An AUC of 0.9894 implies good discrimination between attack and normal traffic. The small deviation in the curve reflects stable detection across varying thresholds.

Figure 9

Confusion matrix for the CICIDS-2018 dataset, showing the classification performance of the proposed model. In the left figure, the confusion matrix shows classwise counts (numbers), while the confusion matrix shown in the right visualizes the classwise counts (in %)

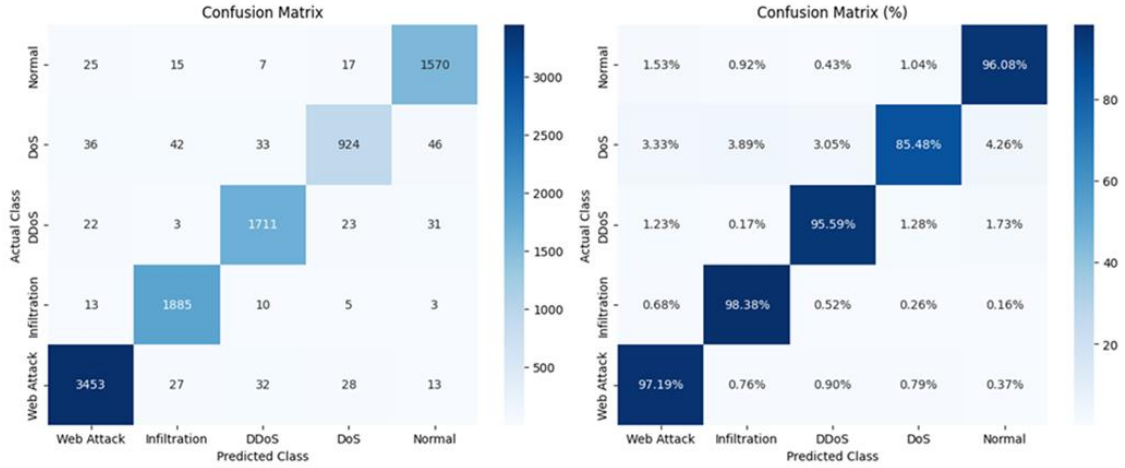


Table 6

Performance evaluation using different metrics on the CICIDS-2018 dataset

	Accuracy (%)	Precision	Recall	F1-score	FPR	FNR	Fitness Score
Class 0	98.38	94.41	96.08	95.24	1.16	3.92	1.32
Class 1	97.62	92.68	85.48	88.93	0.85	14.52	1.32
Class 2	98.34	95.43	95.59	95.51	1.04	4.41	1.32
Class 3	98.78	94.80	98.08	96.41	1.08	1.92	1.32
Class 4	97.97	97.30	97.19	97.24	1.57	2.81	1.32
Average	98.22	94.92	94.48	94.67	1.14	5.52	1.32

Figure 10

Training and validation accuracy curve (left) and loss curves for the CICIDS-2018 dataset

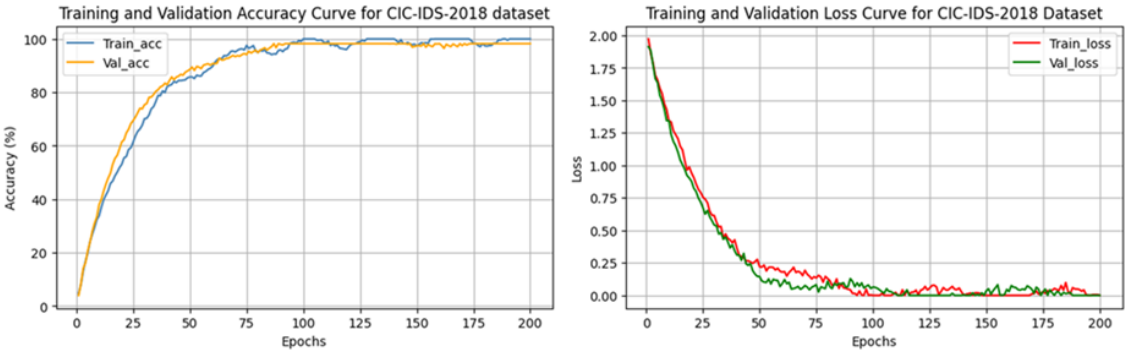


Figure 11

Visualization of the fitness function in terms of the number of selected features and corresponding classification accuracy over 200 epochs

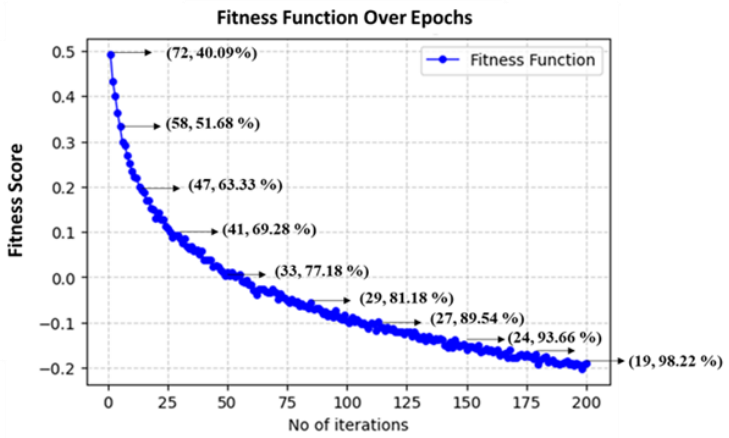
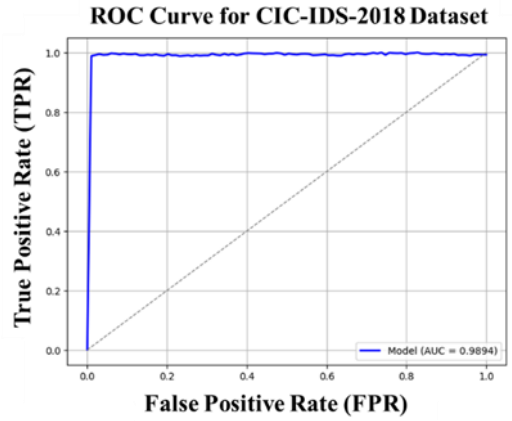


Figure 12

ROC curve for UNSW-NB15 Dataset



5.5.3 Comparison with State-of-the-Art Methods

The improved performance of the presented method outperforms all five state-of-the-art solutions. It successfully discriminates between normal and attack-type queries on the network with the best accuracy of 99.48% on the UNSW-NB15 dataset (Table 7). In addition, the proposed tunneling mechanism facilitates adaptive learning against evolving attack patterns, and it minimizes overfitting with strong generalization ability. The use of an SOSFNN increases dynamic feature refinement. As opposed to traditional neural networks dependent on static-layer mappings, SOSFNN self-organizes according to the

complexity of intrusions, enhancing decision boundaries as well as classification accuracy. The model's precision score of 0.9643 reflects its ability to detect attack events with fewer false alarms, outperforming Convoluted Bi-LSTM (0.9215) and Positional Embedding with Autoencoder (0.9140). It is vital to minimize the number of false alarms when performing intrusion detection monitoring in the actual environment since high false alarm rates may cause unnecessary use of resources and execution of security responses. The model's success with respect to its ability to identify the true intrusion attempts has been shown by its recall rate of 0.9947 and minimal FNR of 0.0052. Compared with XGBoost using ML classifiers (recall: 0.9655, FNR: 0.0345) and SMOTE-based ML classifiers (recall: 0.9570, FNR: 0.0430), the proposed model offers a guarantee of fewer intrusions going undetected for the sake of greater security dependability.

The highest F1-score of 0.9790 among all the techniques confirms its recall-precision tradeoff. It outperforms Convoluted Bi-LSTM (0.9490), XGBoost (0.9363), and Positional Embedding with Autoencoder (0.9427) and is more resistant to cyber-attacks. The second advantage of the proposed technique is that its False Positive Rate (FPR) of 0.0007 is extremely low in comparison with Positional Embedding with Autoencoder (0.0032) and Convoluted Bi-LSTM (0.0025). It ensures that normal network traffic is rarely falsely reported as an attack, and the system is better suited for real-time security applications. At the CICIDS-2018 dataset, the model hits 98.22% accuracy, much higher than that of Convoluted Bi-LSTM (96.85%), XGBoost with ML classifiers (95.72%), SMOTE with ML classifiers (94.88%), PCA with ML techniques (94.21%), and Positional Embedding with Autoencoder (96.20%) (Table 8). Such performance indicates the method's capacity to learn complicated attack models and generalize over diverse network threats. A precision of 94.92% supports its ability to minimize false alarms, surpassing Convoluted Bi-LSTM (92.30%), XGBoost (90.45%), SMOTE (89.20%), and PCA (87.95%). By reducing the amount of false-alarms produced, the proposed model improves the efficiency of an organisation's security personnel, as they will respond only to credible threats. As such, this will help increase the overall efficiency of operations. In regards to recall, the proposed methodology achieved a recall of 94.48%, which is higher than that achieved by the Convoluted BiLSTM at 93.15%, the XGBoost at 91.80%, and the Positional Embedding with Autoencoder method at 92.00%. Furthermore, the model's

very low False Negative Rate (FNR) of 5.52% indicates that it can be relied upon to detect intrusions consistently, when compared to BiLSTM (6.85%), XGBoost (8.20%), and PCA (10.90%), all of which have much higher FNRs. As such, the low FNR means there will be fewer missed intrusions, resulting in an enhanced level of security, thus making the model suitable for use in real-time Cybersecurity. Finally, the F1-score of 94.67% represents the model's ability to strike the proper balance between reducing False Alarms and the accurate identification of legitimate attacks.

The methodology introduced in this study produces a FPR of only 1.14%, which is lower than that produced by the Convoluted BiLSTM (2.10%), the XGBoost (2.85%), and PCA (3.60%). A lower FPR means that the model generates/produces actual traffic examples much less frequently and allows a reduction of unnecessary manual evaluations of traffic, which leads to a more efficient Security Response. Thus, having a lower FPR will allow Security Professionals to focus their investigative efforts on legitimate threats and avoid spending time addressing/monitoring false alerts, and will help pave the way for real-world implementation.

Table 7

Comparative performance between baseline methods and the proposed framework of the UNSW-NB15 dataset

Methods	Accuracy (%)	Precision	Recall	F1-score	FPR	FNR
Convoluted Bi-LSTM model [47]	97.85	0.9215	0.9780	0.9490	0.0025	0.0220
Xg-Boost with machine learning classifiers [48]	96.72	0.9087	0.9655	0.9363	0.0041	0.0345
SMOTE with ML classifiers [49]	96.05	0.8953	0.9570	0.9251	0.0052	0.0430
PCA with ML techniques [50]	95.48	0.8832	0.9501	0.9154	0.0060	0.0499
Positional Embedding with Autoencoder Transformation [51]	97.10	0.9140	0.9732	0.9427	0.0032	0.0268
Proposed Method	99.48	0.9643	0.9947	0.9790	0.0007	0.0052

Table 8

Comparative performance between baseline methods and the proposed framework of the CICIDS-2018 dataset

Methods	Accuracy (%)	Precision	Recall	F1-score	FPR	FNR
Convoluted Bi-LSTM model [47]	96.85	92.30	93.15	92.72	2.10	6.85
Xg-Boost with machine learning classifiers [48]	95.72	90.45	91.80	91.12	2.85	8.20
SMOTE with ML classifiers [49]	94.88	89.20	90.50	89.84	3.20	9.50
PCA with ML techniques [50]	94.21	87.95	89.10	88.52	3.60	10.90

Positional Embedding with Autoencoder Transformation [51]	96.20	91.10	92.00	91.55	2.40	8.00
Proposed Method	98.22	94.92	94.48	94.67	1.14	5.52

5.6 Time complexity comparison

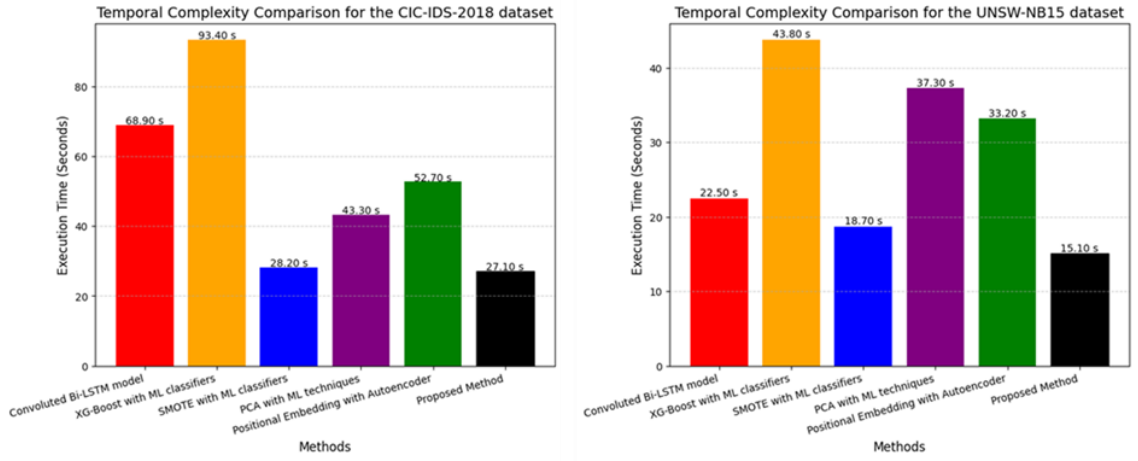
The temporal efficiency results in Figure 13 demonstrate the execution time of the approach on CICIDS-2018 and UNSW-NB15 datasets over current state-of-the-art strategies. Execution time is a valuable measurement in real-time cyber security domains where the rapid detection of intrusions is essential for effective threat control. On CICIDS-2018, the strategy yields a minimum execution time of 27.10 seconds to defeat other methodologies handily. On the other hand, the XGBoost with ML classifiers method boasts the highest run time of 93.40 seconds, proving its computationally expensive nature for performing exhaustive feature selection and complex tree-based decision modeling. The Convoluted Bi-LSTM model is completed within 68.90 seconds, illustrating the heavy computational nature of recurrent operations. Similarly, Positional Embedding with Autoencoder and PCA employing ML algorithms both take 52.70 seconds and 43.30 seconds, respectively, to indicate their moderate but adequately high computational demands. The significantly shorter run time of the suggested model is a result of the success of the proposed improved WOA in optimizing feature selection by tunneling through laborious search spaces, with the effect of discarding unnecessary computations.

The UNSW-NB15 datas*et al*so follows the same trends as the aforementioned dataset where the technique presented has the lowest execution time of 15.10 seconds. Additionally, XGBoost with machine learning classifiers has the longest execution time at 43.80 seconds, which is a result of its complex configuration of decision trees. The other techniques that took more than 15 seconds in execution time were positional embedding using Autoencoder (37.30 seconds) and PCA used in conjunction with ML classifiers (33.20 seconds). While convoluted Bi-LSTM took a moderate amount of time in executing (22.50 seconds) compared to the other techniques, it is substantially slower than the model proposed. The model proposed has a shorter execution time relative to other models because of the way it significantly reduces classification steps while also effectively optimizing them.. As compared to SMOTE-based ML classifiers (18.70

seconds), which attempt to speed training up by oversampling but contribute additional processing overhead, the scheme in this article enhances feature selection by proposing improvements directly without compromising accuracy for a total reduction in computational expense. Overall, the findings point out that the suggested method persistently records the lowest execution time on both datasets. The considerably low computational load guarantees that the model can run smoothly without sacrificing detection accuracy.

Figure 13

Temporal complexity comparison between state-of-the-art methods and proposed method on CICIDS-2018 (left) and UNSW-NB15 (right) datasets



5.7 Potential limitations of the experiment

The integration of improved GWO with a self-organizing fuzzy neural framework enhances detection accuracy, but certain limitations exist.

[1]Processing high-dimensional traffic data adds inference time and memory usage through the combined impact of GWO-based search and deep fuzzy networks;

[2]Relying on knowledge-based attack signatures enhances susceptibility to overfitting, lowering the framework's responsiveness to zero-day attacks;

[3]The fluidity of the hybrid model dampens interpretability, which may hinder its entry into security-vital environments in need of clarity;

[4]The dependency of the system on learned attack signatures can lead to overfitting common attack patterns, lowering its detection of zero-day attacks that are significantly different from regular patterns;

[5]The nature of the hybrid method brings in several hyperparameters, and as such, optimization can be quite challenging. Improper hyperparameter tuning might result in poor performance, slow convergence, or higher false positive rates;

[6]The Accurate Detection Systems can rely on the well-established nature of the different types of training datasets and the quality of how each one has been labelled. Poorly located data that may have been skewed by the way in which it was labelled or distributed will produce a lower level of accurate detection than that which has been well labelled and well protected by the quality of data that has been used in their construction;

[7]Due to the use of deep learning components and the self-organizing nature of this particular model, it is very problematic to interpret how the system functions; therefore, it is likely that the system will not be widely used in security-critical deployments where a high level of explainability is required.

6 CONCLUSION AND FUTURE SCOPE

The IDS Framework presents a new adaptive paradigm for building resilient Intrusion Detection Systems (IDS) in a constantly changing Cyber environment. By utilizing the improved Grey Wolf Optimizer through Adaptive Updates, Spiral Search and Dynamic Encircling to allow the algorithm to navigate through a high dimensional search space efficiently without prematurely converging as well as improving its ability to optimize globally, An Adaptive Search Process creates a Natural Balance Between Exploitation&Exploration. An Adaptive Search Process improves the ability to detect New Cyber Threats in Challenging Network Traffic Patterns. In addition to the Improved Grey Wolf Optimizer, the Self-Organizing Fuzzy Neural Network provides a threefold benefit in that it provides Intelligent Pattern Recognition, Adaptive Decision Making, and Greater Resilience to Complex Attack Patterns. The synergy of the Improved Grey Wolf Optimizer and the Self-Organizing Fuzzy Neural Network greatly increases the

Accuracy, Scalability, and Adaptability of Both Solutions for Real-World Cybersecurity Implementations.

In spite of these improvements, future work can be directed toward further enhancing computational efficiency to allow real-time deployment in high-speed, large-scale network environments. The framework can be expanded to include reinforcement learning-based adaptive tuning for hyperparameter optimization [52], minimizing manual intervention, and enhancing generalization across varied threat scenarios. Moreover, the inclusion of multimodal security analytics like behavioral biometrics [53] and blockchain-based threat intelligence [54] can also enhance the intrusion detection framework's resilience against zero-day attacks and adversarial tampering, including attention-based hierarchical representations [55] and evolutionary feature selection methods [56], may further enhance the system's capability to identify complex attack signatures with less computational overhead. The use of neuromorphic computing and quantum machine learning paradigms can also be investigated to improve the system's processing speed and adaptive capabilities. Through ongoing adaptation to the new generation of cyber threats, this study opens the door to future intelligent security paradigms, guaranteeing forward-looking and robust defense mechanisms for contemporary network infrastructures.

Through the use of UNSW-NB15 and CICIDS-2018 datasets, the model validation concluded with a mean accuracy of 98.22%, a mean precision of 94.92%, a mean recall of 94.48%, and a mean F1-score of 94.67%. The results indicated that the false-positive rate for the UNSW-NB15 dataset is 0.0007, and for the CICIDS-2018 data, it is 1.14%. The time taken to detect both datasets was also fast (15.10 s and 27.10 s, respectively), thus showing that this system can perform real-time monitoring with very low latency. Additionally, this new model requires significantly less processing resources due to reduced features while providing greater accuracy and detection capability than previous models. While this model has made several improvements over previous approaches, future studies may focus on the deployment of this system onto lower-end devices as well as updating the model regularly for the addition of new attack types. Ultimately, the structure of this intrusion detection framework will give businesses flexibility and will allow them to adapt to their ever-changing network environment.

REFERENCES

1. Heidari, A., & Jabraeil Jamali, M. A. (2023). Internet of Things intrusion detection systems: a comprehensive review and future directions. Cluster Computing, 26(6), 3753-3780.

2. Sindiramutty, S. R., Prabagaran, K. R. V., Jhanjhi, N. Z., Murugesan, R. K., Brohi, S. N.,&Masud, M. (2025). Generative AI in Network Security and Intrusion Detection. In Reshaping CyberSecurity With Generative AI Techniques (pp. 77-124). IGI Global.

3. Ahmed, U., Nazir, M., Sarwar, A., Ali, T., Aggoune, E. H. M., Shahzad, T., & Khan, M. A. (2025). Signature-based intrusion detection using machine learning and deep learning approaches empowered with fuzzy Clustering. Scientific Reports, 15(1), 1726.

4. Dasgupta, D., Akhtar, Z., & Sen, S. (2022). Machine learning in cybersecurity: a comprehensive survey. The Journal of Defense Modeling and Simulation, 19(1), 57-106.

5. Pourkamali-Anaraki, F.,&Hariri-Ardebili, M. A. (2021). Neural networks and imbalanced learning for data-driven scientific computing with uncertainties. IEEE Access, 9, 15334-15350.

6. Dey, A. K., Gupta, G. P., & Sahu, S. P. (2023). A metaheuristic-based ensemble feature selection framework for cyber threat detection in IoT-enabled networks. Decision Analytics Journal, 7, 100206.

7. Devika, G.,&Karegowda, A. G. (2023). Bio-inspired optimization: algorithm, analysis and scope of application. In Swarm Intelligence-Recent Advances and Current Applications. IntechOpen.

8. Darvishpoor, S., Darvishpour, A., Escarcega, M., & Hassanalian, M. (2023). Nature-inspired algorithms from oceans to space: A comprehensive review of heuristic and meta-heuristic optimization algorithms and their potential applications in drones. Drones, 7(7), 427.

9. Pham, T. H., & Raahemi, B. (2023). Bio-inspired feature selection algorithms with their applications: a systematic literature review. IEEE Access, 11, 43733-43758.

10. Ramalingam, R., Shobana, J., Arthi, K., Elangovan, G., Radha, S., & Priyanka, N. (2024). An Extensive Investigation of Meta-Heuristics Algorithms for Optimization Problems. In Metaheuristics Algorithm and Optimization of Engineering and Complex Systems (pp. 223-241). IGI Global.

11. Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Grey wolf optimizer. Advances in engineering software, 69, 46-61.

12. Alzaqebah, A., Aljarah, I., Al-Kadi, O., & Damaševičius, R. (2022). A modified grey wolf optimization algorithm for an intrusion detection system. Mathematics, 10(6), 999.

13. Liu, L., Li, L., Nian, H., Lu, Y., Zhao, H., & Chen, Y. (2023). Enhanced grey wolf optimization algorithm for mobile robot path planning. Electronics, 12(19), 4026.

14. Yu, M., Xu, J., Liang, W., Qiu, Y., Bao, S., & Tang, L. (2024). Improved multi-strategy adaptive Grey Wolf Optimization for practical engineering applications and high-dimensional problem solving. Artificial Intelligence Review, 57(10), 277.

15. Kwan, H. K., & Cai, Y. (1994). A fuzzy neural network and its application to pattern recognition. IEEE transactions on Fuzzy Systems, 2(3), 185-193.

16. Moustafa, N.,&Slay, J. (2015, November). UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In 2015 military communications and information systems conference (MilCIS) (pp. 1-6). IEEE.

17. Stiawan, D., Idris, M. Y. B., Bamhdi, A. M., & Budiarto, R. (2020). CICIDS-2017 dataset feature analysis with information gain for anomaly detection. IEEE Access, 8, 132911-132921.

18. Anguita, D., Ghelardoni, L., Ghio, A., Oneto, L.,&Ridella, S. (2012, April). The'K'in K-fold Cross Validation. In ESANN (Vol. 102, pp. 441-446).

19. Srilatha, D., & Shyam, G. K. (2021). Cloud-based intrusion detection using kernel fuzzy clustering and optimal type-2 fuzzy neural network. Cluster Computing, 24(3), 2657-2672.

20. Kunhare, N., Tiwari, R.,&Dhar, J. (2022). Intrusion detection system using hybrid classifiers with meta-heuristic algorithms for the optimization and feature selection by genetic algorithm. Computers and Electrical Engineering, 103, 108383.

21. Saritha, A., Reddy, B. R., & Babu, A. S. (2022). QEMDD: Quantum inspired ensemble model to detect and mitigate DDoS attacks at various layers of SDN architecture. Wireless Personal Communications, 127(3), 2365-2390.

22. Ghanbarzadeh, R., Hosseinalipour, A.,&Ghaffari, A. (2023). A novel network intrusion detection method based on metaheuristic optimisation algorithms. Journal of ambient intelligence and humanized computing, 14(6), 7575-7592.

23. Jain, D. K., Ding, W., & Kotecha, K. (2023). Training fuzzy deep neural network with honey badger algorithm for intrusion detection in cloud environment. International Journal of Machine Learning and Cybernetics, 14(6), 2221-2237.

24. Ninu, S. B. (2023). An intrusion detection system using exponential Henry gas solubility optimization based deep neuro fuzzy network in MANET. Engineering Applications of Artificial Intelligence, 123, 105969.

25. Ishaque, M., Johar, M. G. M., Khatibi, A., & Yamin, M. (2023). A novel hybrid technique using fuzzy logic, neural networks and genetic algorithm for intrusion detection system. Measurement: Sensors, 30, 100933.

26. Subramani, S., & Selvi, M. (2023). Intelligent IDS in wireless sensor networks using deep fuzzy convolutional neural network. Neural Computing and Applications, 35(20), 15201-15220.

27. Wu, X., Jin, Z., Zhou, J., & Duan, C. (2023). Quantum walks-based classification model with resistance for cloud computing attacks. Expert Systems with Applications, 232, 120894.

28. Maazalahi, M., & Hosseini, S. (2024). K-means and meta-heuristic algorithms for intrusion detection systems. Cluster Computing, 1-43.

29. Subramanian, G., & Chinnadurai, M. (2024). Hybrid Quantum enhanced federated learning for cyber attack detection. Scientific Reports, 14(1), 32038.

30. Bhadani, V., Singh, A., Kumar, V., & Gaurav, K. (2024). Nature-inspired optimal tuning of input membership functions of fuzzy inference system for groundwater level prediction. Environmental Modelling & Software, 175, 105995.

31. Ravindran, S. (2024). Intelligent fuzzy logic based intrusion detection system for effective detection of black hole attack in wsn. Peer-to-Peer Networking and Applications, 1-17.

32. De Maesschalck, R., Jouan-Rimbaud, D., & Massart, D. L. (2000). The mahalanobis distance. Chemometrics and intelligent laboratory systems, 50(1), 1-18.

33. Lancaster, H. O., & Seneta, E. (2005). Chi-square distribution. Encyclopedia of biostatistics, 2.

34. Patro, S. G. O. P. A. L., & Sahu, K. K. (2015). Normalization: A preprocessing stage. arXiv preprint arXiv:1503.06462.

35. Ul Haq, I., Gondal, I., Vamplew, P., & Brown, S. (2019). Categorical features transformation with compact one-hot encoder for fraud detection in distributed environment. In Data Mining: 16th Australasian Conference, AusDM 2018, Bahrurst, NSW, Australia, November 28–30, 2018, Revised Selected Papers 16 (pp. 69-80). Springer Singapore.

36. Larsen, B. S. (2022). Synthetic minority over-sampling technique (SMOTE). GitHub (https://github. com/dkbsl/matlab_smote/releases/tag/1.0).

37. Kyurkchiev, N., & Markov, S. (2015). Sigmoid functions: some approximation and modelling aspects. LAP LAMBERT Academic Publishing, Saarbrucken, 4, 34.

38. Vesanto, J., & Alhoniemi, E. (2000). Clustering of the self-organizing map. IEEE Transactions on neural networks, 11(3), 586-600.

39. Amin, F., & Mahmoud, M. (2022). Confusion matrix in binary classification problems: A step-by-step tutorial. Journal of Engineering Research, 6(5), 0-0.

40. Aronoff, S. (1982). Classification accuracy: a user approach. Photogrammetric Engineering and Remote Sensing, 48(8), 1299-1307.

41. Yacouby, R., & Axman, D. (2020, November). Probabilistic extension of precision, recall, and f1 score for more thorough evaluation of classification models. In Proceedings of the first workshop on evaluation and comparison of NLP systems (pp. 79-91).

42. Pietraszek, T. (2004). Using adaptive alert classification to reduce false positives in intrusion detection. In Recent Advances in Intrusion Detection: 7th International Symposium, RAID 2004, Sophia Antipolis, France, September 15-17, 2004. Proceedings 7 (pp. 102-124). Springer Berlin Heidelberg.

43. Ho, C. Y., Lai, Y. C., Chen, I. W., Wang, F. Y., & Tai, W. H. (2012). Statistical analysis of false positives and false negatives from real traffic with intrusion detection/prevention systems. IEEE Communications Magazine, 50(3), 146-154.

44. Turner, J. R. (2020). Area under the curve (AUC). Encyclopedia of Behavioral Medicine, 146-146.

45. Tiwari, A. (2023). A hybrid feature selection method using an improved binary butterfly optimization algorithm and adaptive β–hill climbing. IEEE Access, 11, 93511-93537.

46. Shekar, B. H., & Dagnew, G. (2019, February). Grid search-based hyperparameter tuning and classification of microarray cancer data. In 2019 second international conference on advanced computational and communication paradigms (ICACCP) (pp. 1-8). IEEE.

47. Hnamte, V., & Hussain, J. (2023). DCNNBiLSTM: An efficient hybrid deep learning-based intrusion detection system. Telematics and Informatics Reports, 10, 100053.

48. Talukder, M. A., Sharmin, S., Uddin, M. A., Islam, M. M., & Aryal, S. (2024). MLSTL-WSN: machine learning-based intrusion detection using SMOTETomek in WSNs. International Journal of Information Security, 23(3), 2139-2158.

49. Asif, M., Abbas, S., Khan, M. A., Fatima, A., Khan, M. A.,&Lee, S. W. (2022). MapReduce based intelligent model for intrusion detection using machine learning technique. Journal of King Saud University-Computer and Information Sciences, 34(10), 9723-9731.

50. Saheed, Y. K., Abiodun, A. I., Misra, S., Holone, M. K.,&Colomo-Palacios, R. (2022). A machine learning-based intrusion detection for detecting internet of things network attacks. Alexandria Engineering Journal, 61(12), 9395-9409.

51. Wu, Z., Zhang, H., Wang, P., & Sun, Z. (2022). RTIDS: A robust transformer-based approach for intrusion detection system. IEEE Access, 10, 64375-64387.

52. Zhang, H., Sun, J., Wang, Y., Shi, J., & Xu, Z. (2022). Variational reinforcement learning for hyper-parameter tuning of adaptive evolutionary algorithm. IEEE Transactions on Emerging Topics in Computational Intelligence, 7(5), 1511-1526.

53. Alzubaidi, A.,&Kalita, J. (2016). Authentication of smartphone users using behavioral biometrics. IEEE Communications Surveys&Tutorials, 18(3), 1998-2026.

54. Cha, J., Singh, S. K., Pan, Y., & Park, J. H. (2020). Blockchain-based cyber threat intelligence system architecture for sustainable computing. Sustainability, 12(16), 6401.

55. Islam, M. M., & Iqbal, T. (2021). Multi-gat: A graphical attention-based hierarchical multimodal representation learning approach for human activity recognition. IEEE Robotics and Automation Letters, 6(2), 1729-1736.

56. Abd-Alsabour, N. (2014, October). A review on evolutionary feature selection. In 2014 European Modelling Symposium (pp. 20-26). IEEE.

Authors' Contribution

All authors contributed equally to the development of this article.

Data availability

All datasets relevant to this study's findings are fully available within the article.

How to cite this article (APA)

Abbas, S. A. A. (2026). ENHANCED GREY WOLF OPTIMIZER WITH DYNAMIC ENCIRCLING AND FITNESS-BASED BLENDING FOR CYBERATTACK DETECTION USING SELF-ORGANIZING FUZZY NEURAL NETWORK. *Veredas Do Direito,* 23(5), e235674. https://doi.org/10.18623/rvd.v23.5674