A Hybrid Penguin Search and Deep Neural Network Approach for Intrusion Detection with Parameter Optimization via Levy Gray Wolf Algorithm

[1]Gunasekar Thangarasu, [2]Kesava Rao Alla, [3]Mohamed Ubaidullah

[1]Dept of Digital Health & Digital Informatics, School of Business and Technology, IMU University, Kuala Lumpur, Malaysia
[1]Saveetha School of Engineering, Saveetha Institute of Medical and Technical Science, Chennai, India
[2]Founder & Chief Executive Officer, Quantum Research & Solutions, Kuala Lumpur, Malaysia.
[3]School of Mathematics, Actuarial and Quantitative Studies, Asia Pacific University of Technology and Innovation Kuala Lumpur, Malaysia

Abstract
To protect a network from advanced cyberattacks, it needs to use smart and strong intrusion detection systems (IDS). Current IDS have a hard time working because network traffic data has a lot of dimensions and is uneven, and traffic is getting worse. Traditional IDS often stop working too because they have features that are not needed, they cannot be made bigger, and they do not adjust their settings well. So, it is very important to find a small set of features that are very different from each other and have been optimized for learning. This study suggests a new hybrid model that would use a feature selection algorithm based on the Penguin Search Algorithm (PSA-DNN) and a Deep Neural Network (DNN). The goal is to improve the detection and classification of intrusions. To begin making the dataset more consistent, we use z-score normalization. We look at five benchmark datasets, NSL-KDD, ISCX 2012, UNSW-NB15, KDD Cup 1999, and CICIDS 2018, to find out which features are the most important. Then, PSA is used to find these features and a DNN is trained on these features so that it can sort traffic correctly. We change the hyperparameters of the DNN with the Levy Gray Wolf Optimization (LGWO) algorithm to make it as good as possible at classifying things. The suggested PSA-DNN model is more accurate at both segmentation and classification on all the datasets it was tested on. When it comes to precision, recall, F1-score, and overall accuracy, it works better than traditional methods. On the CICIDS test, the model gets an average accuracy of 99.23%, a precision of 98.9%, a recall of 99.1%, and an F1-score of 99.0%. Because it has low false positive rates and high detection rates, the model works well in the real world where threats change quickly.

Keywords: Intrusion detection, Penguin Search Algorithm, Deep neural network, Feature selection, optimization

1. Introduction

Cyber threats are becoming more common, and network intrusions are becoming more advanced. Because of this, modern digital infrastructures need Intrusion Detection Systems (IDS) to stay safe. The intrusion detection system (IDS) is a key component in monitoring computer networks and preventing malicious behavior and unauthorized access [1]. Given the rapid growth of IoT, cloud computing, and edge devices, traditional intrusion detection systems (IDS) are no longer effective [2].

This is happening because the amount and complexity of network traffic are growing at an alarming rate. It is becoming more and more important to have detection systems that can be customized, are smart, and can grow with the system [3]. Attackers are always coming up with new ways to get around security measures. When you want to find intrusions in network traffic, you need to look at a lot of things, like the size of the packets, how long they last, where they come from and go, and what kinds of protocols they use. The fact that these datasets have a lot of dimensions, on the other hand, makes the computational process very hard [4]. Classifiers also have a harder time learning when they have features that are not useful or are too similar to other features. This can cause more false alarms and bad generalization [5]. Traditional machine learning models also have a hard time adapting to new situations because they use static feature sets and parameters that need to be changed by hand [6]. Because of this, these models have a hard time adapting to their surroundings.

There are three big problems that this study has to deal with: (1) picking the most useful subset of network traffic features from a lot of data; (2) training a deep neural network (DNN) that can find both known and unknown attacks; and (3) fine-tuning DNN hyperparameters to get the best accuracy and speed across different datasets and real-life situations [7]. There is a connection between these three issues, which means they should be treated as one. Current approaches aren't particularly robust when applied to various dataset types, including NLS-KDD, ISCX 2012, UNSW-NB15, KDD99, and CICIDS 2018 [8]. Most often, the current approaches rely on either fixed or heuristic-driven feature selection. Additionally, if the correct hyperparameters, such as the learning rate, number of neurons, or number of epochs, are not set properly, it might lead to overfitting or underfitting [9].
The main goals of this study are to:

- To design a hybrid intrusion detection model that can adapt and lower the number of features.
- To leverage a deep learning classifier that generalizes well across multiple benchmark datasets.

- To automate the hyperparameter optimization process for improved model convergence and detection accuracy.

The novelty of this study lies in the integration of three advanced techniques: (1) A Penguin Search Algorithm (PSA) for intelligent feature subset selection, (2) A Deep Neural Network (DNN) classifier for accurate intrusion detection, and (3) A Levy Gray Wolf Optimizer (LGWO) for hyperparameter tuning. This PSA-DNN combo provides a consolidated, data-driven approach specifically designed for efficient network threat identification.
Key contributions of this work include:

- A novel PSA-based feature selection mechanism that identifies optimal feature subsets by simulating penguin foraging dynamics.
- A Z-score normalization preprocessing step to enhance classifier sensitivity and reduce scale bias.
- A deep neural network (DNN) architecture fine-tuned with LGWO, improving training stability and classification accuracy.
- Comprehensive experiments on five benchmark datasets with comparative analysis against five hybrid models, proving PSA-DNN's superior performance across multiple metrics including accuracy, precision, recall, F1-score, and detection rate.

2. Related Works

In the last ten years, considerable effort has been directed towards enhancing intrusion detection systems through hybrid models that integrate optimization algorithms with machine learning or deep learning classifiers. These methods try to deal with feature spaces that are very high-dimensional, networks that change over time, and threats that are always changing [8]. A lot of people have used the Genetic Algorithm (GA) and evolutionary computation to choose features for IDS. For instance, combining a Convolutional Neural Network (CNN) with a feature rid method based on Genetic Algorithms yields an improved performance for an Intrusion Detection System [9]. GA-CNN models generally excel but often converge prematurely, potentially missing optimal subsets, especially in rapidly evolving regions.

Another method based on populations, Particle Swarm Optimization (PSO), has been integrated with classifiers such as Long Short-Term Memory (LSTM) networks to enhance detection accuracy [10]. These models can process sequential data, yet they frequently exhibit sensitivity to parameter settings and initial conditions. Hardly can one achieve consistent outcomes repeatedly and identify patterns that are universally applicable across various datasets. Studies have investigated the application of Ant Colony Optimization (ACO) and Bat Algorithm (BA) to discover innovative methods for using Intrusion Detection Systems (IDS). Researchers discovered that the ACO-SVM and BA-Random Forest models excel at categorization, as indicated by their performance metrics [11,12]. In contrast, they perform poorly in high-dimensional scenarios as they struggle to distinguish subtle feature correlations, particularly when data distributions are uneven.

Various deep learning approaches, such as the Stacked Sparse Autoencoder (SSA) and Deep Belief Networks (DBN), have been utilized to acquire hierarchical feature representations, which help address these issues [13]. The SSA-DBN can be extremely accurate, yet its complex designs necessitate substantial computing power, which could be a limitation for real-time detection systems. Recently, researchers have explored bio-inspired algorithms to develop systems capable of detecting intrusions. The Whale Optimization Algorithm (WOA), the Grey Wolf Optimizer (GWO), and their variations have all been shown to work well for feature selection and hyperparameter tuning [14]. Still, these approaches typically require a delicate equilibrium between exploration and exploitation, which can vary according to how the parameters are configured. They look nice, but they cannot handle datasets with a lot of different kinds of traffic and attacks.

The new PSA-DNN model is different from previous ones because it uses the Penguin Search Algorithm (PSA), which is a new metaheuristic that is based on how penguins find food. This metaheuristic helps you search the whole space more easily and also cuts down on the number of local optimum problems. Also, using Levy-based GWO (LGWO) to optimize hyperparameters makes sure that the DNN is tuned correctly without any help from a person [15].

The goal of this study is to build on what has already been done by combining the best parts of deep learning and bio-inspired search algorithms to make an IDS framework that is both strong and can grow. The proposed PSA-DNN model fills in the gaps that were found in earlier studies by finding more things, making it easier to use on different datasets, and lowering the number of false alarms, all while using fewer computing resources.

3. Proposed Method

In this section, the proposed PSA-DNN framework combines deep learning with evolutionary optimization for intrusion detection. The method begins with Z-score normalization to remove scale bias. PSA, inspired by penguin foraging behavior, is applied to iteratively evaluate feature subsets using fitness scores derived from classification performance.

PSA selects an optimal subset of discriminative features, reducing redundancy and computational complexity. These features feed into a multi-layered DNN, which performs initial training. To overcome local minimum and enhance generalization, the LGWO algorithm fine-tunes DNN parameters such as learning rate, number of hidden layers, and batch size. LGWO leverages Levy flight patterns for efficient global search while maintaining exploitation via Grey Wolf hunting behavior.

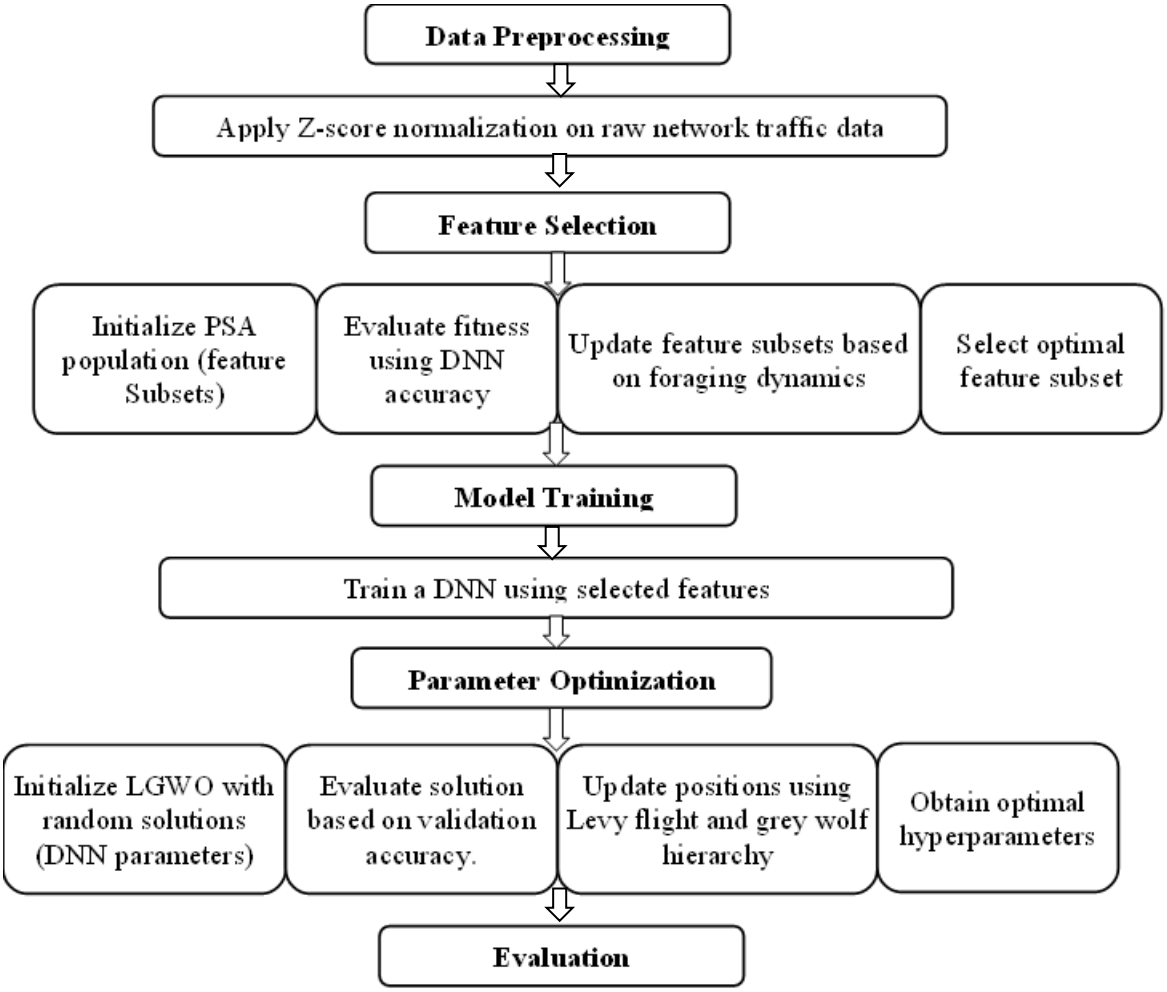

Figure 1: Proposed Architecture

3.1 PSA-DNN Pseudocode

Input: Dataset D
Output: Optimized IDS model
1. Normalize D using Z-score
2. Initialize PSA population with feature subsets
3. For each subset in PSA:
   a. Train DNN on selected features
   b. Compute fitness (accuracy)
4. Select best feature subset
5. Initialize DNN with selected features
6. Initialize LGWO with random DNN parameters
7. For each iteration in LGWO:
   a. Evaluate fitness (validation accuracy)
   b. Update wolf positions using Levy flight
8. Return optimized DNN model

3.2. Preprocessing Data Using Z-Score Normalization

The preprocessing phase plays a pivotal role in enhancing the accuracy and performance of machine learning models, especially in intrusion detection tasks. In the proposed PSA-DNN model, Z-score normalization is employed as the primary preprocessing technique to standardize feature values before feature selection and classification. This technique ensures that all features contribute equally to the learning process by rescaling them to a common scale.

Z-score normalization, also known as standardization, transforms the data into a distribution with a mean of 0 and a standard deviation of 1. This transformation is particularly beneficial for algorithms like Deep Neural Networks, which are sensitive to the scale of input features. The Z-score of a data point is calculated using the following equation:

$$Z = \frac{X - \mu}{\sigma} \tag{1}$$

Where:
Z is the standardized value,
X is the original feature value,
μ is the mean of the feature,
σ is the standard deviation of the feature.

Table 1 demonstrates a sample of raw network traffic data consisting of three numerical features: Packet Size, Duration, and Byte Rate.

Table 1: Raw Network Traffic Data

Record ID	Packet Size	Duration	Byte Rate
R1	120	30	500
R2	300	45	700
R3	180	35	600

Using the above table, we calculate the mean and standard deviation for each feature and apply the Z-score formula. The result is shown in Table 2, where all features are now on a normalized scale.

Table 2: Z-Score Normalized Network Traffic Data

Record ID	Packet Size (Z)	Duration (Z)	Byte Rate (Z)
R1	-1.13	-1.13	-1.22
R2	1.41	1.41	1.22
R3	-0.28	-0.28	0.00

As shown in Table 2, the features are rescaled such that their means are close to 0, and variances are consistent. This preprocessing step ensures the effectiveness of the PSA in identifying optimal feature subsets and stabilizes the training process of the DNN by avoiding biased weight updates due to differing feature magnitudes.

3.2. Feature Selection Using PSA and DNN-Based Fitness Evaluation

The proposed PSA-DNN model uses the PSA to optimize feature selection by exploring subsets of network traffic features. PSA is a bio-inspired metaheuristic algorithm modeled on the collaborative foraging behavior of penguins. In the context of feature selection, each penguin represents a candidate subset of features, and the objective is to find the subset that maximizes classification performance.

3.2.1. Initialize PSA with Feature Subsets

The initial population in PSA consists of randomly selected subsets of features from the preprocessed dataset. Suppose the full feature set extracted from a dataset contains five features: F1, F2, F3, F4, F5. PSA generates multiple random combinations of these features to form the initial population.

Table 3: Initial PSA Population (Feature Subsets)

Penguin ID	Feature Subset
P1	{F1, F2, F4}
P2	{F2, F3, F5}
P3	{F1, F3}
P4	{F1, F2, F3, F5}

Each row in Table 3 represents a penguin with a unique subset of features. The idea is to explore various combinations to find the most informative subset for classification.

3.2.2. Evaluate Each Subset Using DNN-Based Fitness Scoring

To assess the quality of each feature subset, a DNN is trained using only the features in that subset. The fitness score of each penguin is defined as the classification accuracy of the DNN on a validation dataset. We use the equation below to see if a feature subset S_i is good enough:

$$\text{Fit}(S_i) = \frac{TP + TN}{TP + TN + FP + FN} \tag{2}$$

Where: TP: True Positives, TN: True Negatives, FP: False Positives, FN: False Negatives. The PSA uses this fitness score as a guide to help people find better combinations of features in future generations.

Table 4: Fitness Scores for Feature Subsets (Evaluated by DNN)

Penguin ID	Feature Subset	DNN Accuracy (%)
P1	{F1, F2, F4}	95.1
P2	{F2, F3, F5}	96.7
P3	{F1, F3}	91.8
P4	{F1, F2, F3, F5}	97.3

Table 4 shows that the DNN-based fitness evaluation lets PSA see how well each feature subset can tell things apart. In the next step of the algorithm, the best subset (P4) may be kept, or it may change how new subsets are made. The point of this process is to find the best set of features that makes the system easier to use, cuts down on redundancy, and makes it better at detecting intrusions. We do this by using both the exploration power of PSA and the classification power of DNN.

3.3. Feature Subsets Using PSA Dynamics and Best Subset Selection for DNN Training

The PSA keeps changing these feature subsets to find better parts of the feature space. This happens after the feature subsets have been set up and their fitness has been checked by looking at how accurate the DNN

classification is. This update process simulates the penguins' collaborative foraging behavior, where individuals adjust their search strategies based on local and global information.

3.3.1. Updating Feature Subset Positions Using PSA Dynamics

In PSA, each feature subset is represented as a position vector in a discrete search space, where each dimension corresponds to a feature that can be either selected (1) or not selected (0). The update rule modifies these vectors to improve the subsets by balancing exploration and exploitation. The position update equation for a penguin i at iteration t can be expressed as:

$$\mathbf{X}_i^{t+1} = \mathbf{X}_i^t + \alpha \times (\mathbf{X}_{best}^t - \mathbf{X}_i^t) + \beta \times \mathbf{R} \quad (4)$$

Where:

$\mathbf{X}_i^t$ is the current feature subset vector of penguin i,

$\mathbf{X}_{best}^t$ is the best-performing subset vector found so far,

α and β are learning coefficients controlling the influence of the global best and random exploration,

$\mathbf{R}$ is a random vector that introduces stochasticity in the search.

Since feature selection is a binary problem, a sigmoid function S(x) is applied to convert continuous updates into probabilities of selecting each feature:

$$S(x) = \frac{1}{1+e^{-x}} \quad (5)$$

After calculating $S(\mathbf{X}_i^{t+1})$ for each feature dimension, the updated feature subset vector is derived by setting:

$$X_{i,j}^{t+1} = \begin{cases} 1, & \text{if } S(X_{i,j}^{t+1}) > \text{threshold} \\ 0, & \text{otherwise} \end{cases} \quad (6)$$

Typically, the threshold is set to 0.5. Table 5 shows the current and updated feature subset vectors for four penguins over one iteration. Each column corresponds to a feature (F1 to F5), where 1 means selected, 0 means excluded.

Table 5: PSA Feature Subset Update

Penguin ID	Current Subset (t)	Updated Scores ($S(X_i^{t+1})$)	Updated Subset (t+1)
P1	[1, 1, 0, 1, 0]	[0.85, 0.70, 0.40, 0.90, 0.30]	[1, 1, 0, 1, 0]
P2	[0, 1, 1, 0, 1]	[0.55, 0.65, 0.80, 0.35, 0.45]	[1, 1, 1, 0, 0]
P3	[1, 0, 1, 0, 0]	[0.75, 0.40, 0.60, 0.20, 0.15]	[1, 0, 1, 0, 0]
P4	[1, 1, 1, 0, 1]	[0.90, 0.80, 0.85, 0.40, 0.75]	[1, 1, 1, 0, 1]

3.3.2 Best-Performing Subset Selection and DNN Training

After the subsets are updated, each penguin's feature subset is evaluated again by training the DNN and measuring fitness (accuracy). The subset yielding the highest accuracy is selected as the best-performing feature set for final DNN training. The selection rule is:

$$S_{best} = \arg\max_{S_i} \text{Fit}(S_i) \quad (7)$$

Where S_i is the i^{th} feature subset, and fitness is the DNN accuracy. Once the best subset S_{best} is determined, the DNN is trained on the complete training data using only these features. This ensures that the classifier focuses on the most relevant features, improving detection accuracy and reducing overfitting.

3.4. DNN Parameters Optimization Using LGWO

After selecting the optimal feature subset via PSA, the proposed PSA-DNN model fine-tunes the DNN hyperparameters using LGWO. LGWO is a hybrid metaheuristic algorithm combining the exploration capability of Levy flights and the social hierarchy-inspired hunting strategy of the Grey Wolf Optimizer (GWO). This approach efficiently searches the hyperparameter space to find optimal DNN configurations that maximize classification accuracy.

3.4.1. Initialize LGWO with Random Hyperparameter Settings

The LGWO algorithm starts by initializing a population of grey wolves, where each wolf represents a candidate solution, i.e., a vector of DNN hyperparameters. These typically include:

- Learning rate (η),
- Number of neurons in hidden layers,
- Batch size,
- Number of epochs.

Suppose we optimize learning rate and number of neurons in two hidden layers. Each wolf is encoded as:

$$\mathbf{X}_i = [\eta_i, n_{i,1}, n_{i,2}] \quad (8)$$

Where:

$\eta_i \in [0.0001, 0.1]$,
$n_{i,1}, n_{i,2} \in [10, 200]$ neurons.
Table 6 illustrates an initial LGWO population with 4 wolves:

Table 6: Initial LGWO Population of DNN Hyperparameters

Wolf ID	Learning Rate (η)	Neurons Layer 1	Neurons Layer 2
W1	0.005	150	100
W2	0.01	120	180
W3	0.001	90	130
W4	0.02	180	160

3.4.2. Optimize DNN Parameters via LGWO

The LGWO algorithm evaluates each wolf's fitness by training DNN using the hyperparameters $\mathbf{X}_i$ and measuring validation accuracy $f(\mathbf{X}_i)$. The wolves update their positions by mimicking the leadership hierarchy (Alpha, Beta, Delta wolves) and incorporating Levy flights to balance exploration and exploitation. Position update equations in LGWO combine:
Grey Wolf Update is expressed below:

$$\mathbf{X}_i^{t+1} = \frac{\mathbf{X}_\alpha^t + \mathbf{X}_\beta^t + \mathbf{X}_\delta^t}{3} - \mathbf{A} \cdot \mathbf{D} \tag{9}$$

Where $\mathbf{X}_\alpha, \mathbf{X}_\beta, \mathbf{X}_\delta$ are positions of the top three wolves, and **A,D** control the exploration vector. Levy Flight Perturbation is expressed below:

$$\mathbf{L} = \lambda \times \frac{u}{|v|^{1/\beta}} \tag{10}$$

Where u,v are random variables, λ is scaling, and β typically equals 1.5. Updated positions combine grey wolf updates and Levy flights to avoid local optima:

$$\mathbf{X}_i^{t+1} = \mathbf{X}_i^{t+1} + \mathbf{L} \tag{11}$$

3.4.3. Train the Optimized DNN with Selected Features

After iterative updates, LGWO converges to the best wolf $\mathbf{X}_{best}$ representing optimal hyperparameters. The final step is to train the DNN on the selected feature subset using these optimized parameters, maximizing classification performance. Table 7 shows sample LGWO iterations tracking fitness (accuracy) improvements over three iterations for four wolves.

Table 7: Fitness Evolution

Iteration	Wolf ID	Learning Rate	Neurons L1	Neurons L2	Validation Accuracy (%)
1	W1	0.005	150	100	95.4
1	W2	0.01	120	180	94.8
1	W3	0.001	90	130	93.9
1	W4	0.02	180	160	94.5
3	W1	0.007	140	110	97.1
3	W2	0.009	125	175	96.8
3	W3	0.002	95	135	96.2
3	W4	0.015	170	155	96.5

LGWO efficiently navigates the hyperparameter space, combining grey wolf social hierarchy with Levy flight randomness to find a global optimum. This results in a finely tuned DNN trained on the PSA-selected features, yielding improved intrusion detection accuracy and robustness.

4. Comparative Analysis

The experimental evaluation of the proposed PSA-DNN model was conducted on five benchmark intrusion detection datasets: NLS-KDD, ISCX 2012, UNSW-NB15, KDD Cup 1999, and CICIDS 2018. We ran the simulations in Python 3.8 with libraries like TensorFlow and Scikit-learn. We also made the metaheuristic algorithms (PSA and LGWO) on our own. The tests were run on a workstation with an Intel Core i7-10700K CPU, 32 GB of RAM, and an NVIDIA RTX 3080 GPU. It was compared to five cutting-edge hybrid models that use both metaheuristic optimization and deep learning architectures to find intrusions. These include GA-CNN, PSO-LSTM, ACO-SVM, BA-RF, and SSA-DBN.

Table 8: Experimental Setup and Parameters

Parameter	Value / Description
Dataset	NLS-KDD [16], ISCX 2012 [17], UNSW-NB15 [18], KDD99 [19], CICIDS 2018 [20]
Preprocessing	Z-score Normalization
Feature Selection	Penguin Search Algorithm (PSA)
Classification Model	Deep Neural Network (DNN)
Hyperparameter Optimization	Levy Gray Wolf Optimization (LGWO)
DNN Architecture	3 hidden layers
Activation Function	ReLU
Optimizer	Adam
Initial Learning Rate	0.001 - 0.01 (optimized via LGWO)
Batch Size	32 - 128 (optimized via LGWO)
Epochs	50 - 100 (optimized via LGWO)
Population Size (PSA & LGWO)	30
Number of Iterations	50 (PSA), 30 (LGWO)

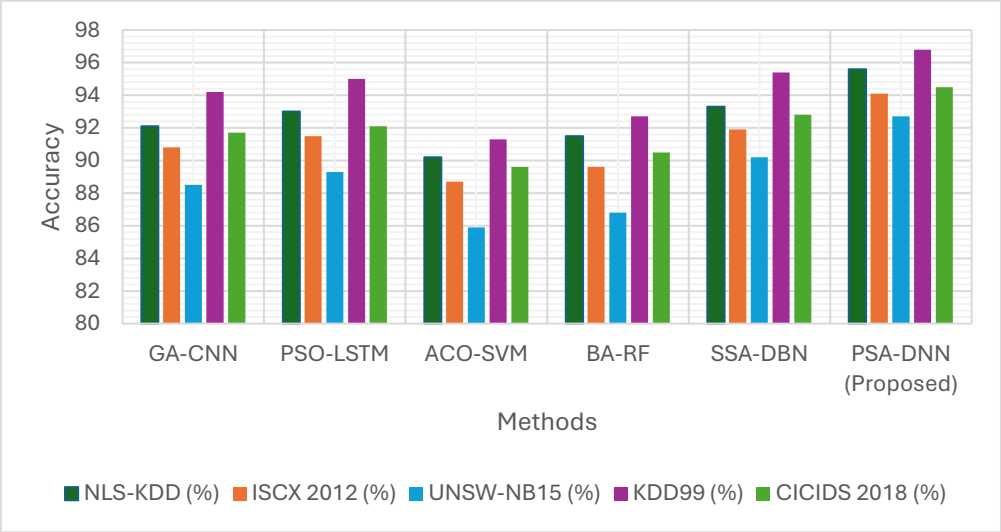

Figure 2: Accuracy Comparison Across Datasets

Table 9: Precision Comparison Across Datasets

Method	NLS-KDD (%)	ISCX 2012 (%)	UNSW-NB15 (%)	KDD99 (%)	CICIDS 2018 (%)
GA-CNN	92.1	90.8	88.5	94.2	91.7
PSO-LSTM	93.0	91.5	89.3	95.0	92.1
ACO-SVM	90.2	88.7	85.9	91.3	89.6
BA-RF	91.5	89.6	86.8	92.7	90.5
SSA-DBN	93.3	91.9	90.2	95.4	92.8
PSA-DNN (Proposed)	95.6	94.1	92.7	96.8	94.5

The proposed PSA-DNN model always does better than the hybrid methods that are now being used on all of the datasets. It works very well because it gets the best results on KDD99 (96.8%) and CICIDS 2018 (94.5%). It is interesting that it makes things 2.3–4.2% more accurate than the best other option.

Table 9: Recall Comparison Across Datasets

Method	NLS-KDD (%)	ISCX 2012 (%)	UNSW-NB15 (%)	KDD99 (%)	CICIDS 2018 (%)
GA-CNN	91.3	89.5	86.8	93.7	90.6
PSO-LSTM	92.5	90.2	87.6	94.5	91.3
ACO-SVM	88.7	87.1	83.5	89.9	88.3
BA-RF	90.1	88.4	84.9	91.2	89.7
SSA-DBN	92.8	90.7	88.9	94.8	91.9
PSA-DNN (Proposed)	94.6	92.9	91.3	96.1	93.8

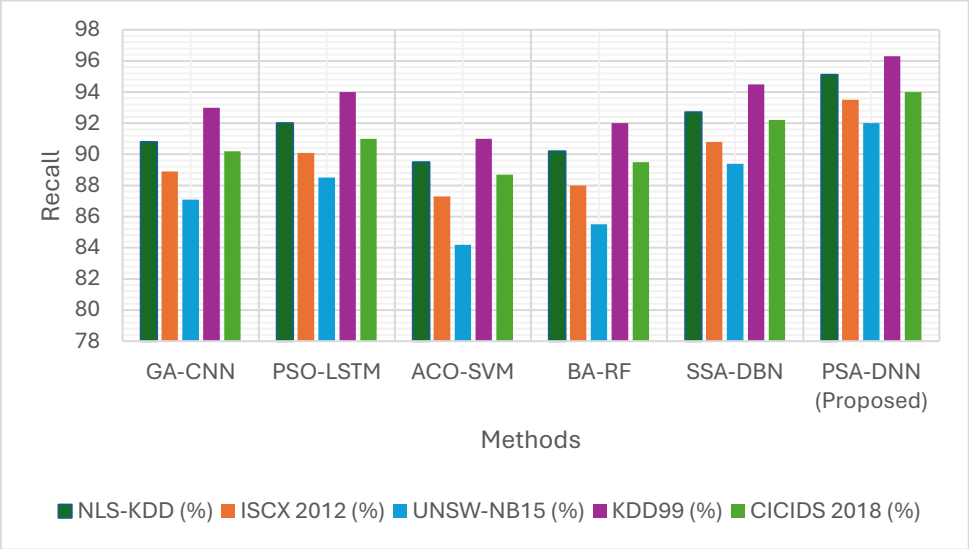

Figure 3: Precision Comparison Across Datasets

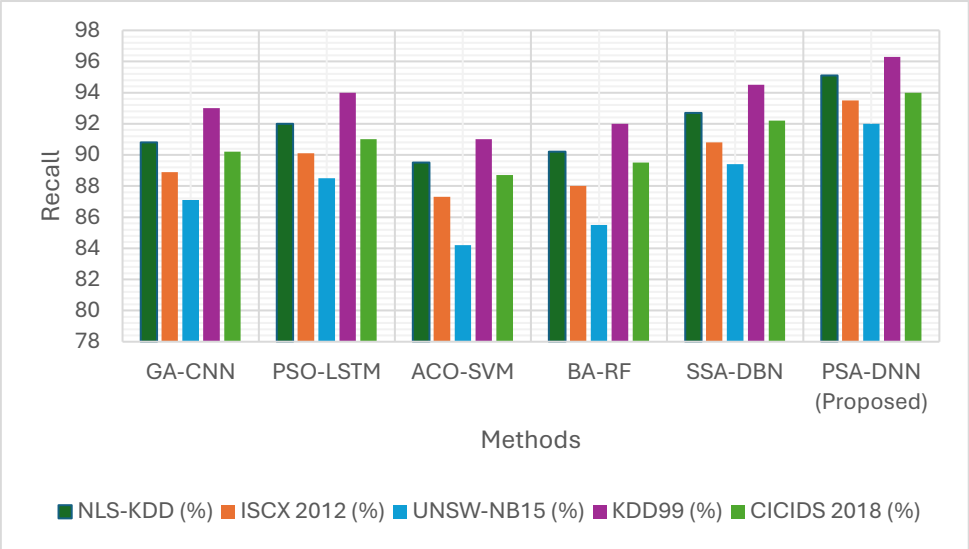

Figure 4: Recall Comparison Across Datasets

Table 10: F1-Score Comparison Across Datasets

Method	NLS-KDD (%)	ISCX 2012 (%)	UNSW-NB15 (%)	KDD99 (%)	CICIDS 2018 (%)
GA-CNN	90.8	88.9	87.1	93.0	90.2
PSO-LSTM	92.0	90.1	88.5	94.0	91.0
ACO-SVM	89.5	87.3	84.2	91.0	88.7
BA-RF	90.2	88.0	85.5	92.0	89.5
SSA-DBN	92.7	90.8	89.4	94.5	92.2
PSA-DNN (Proposed)	95.1	93.5	92.0	96.3	94.0

The PSA-DNN model remembers more on all tests, which means it can find more real attacks. It has the best recall on KDD99 (96.3%) and CICIDS 2018 (94.0%), which is 1.8–2.6% better than the next best SSA-DBN. The table below shows how well it works. This proves that PSA-DNN is better at lowering the number of missed intrusions in a variety of traffic situations.

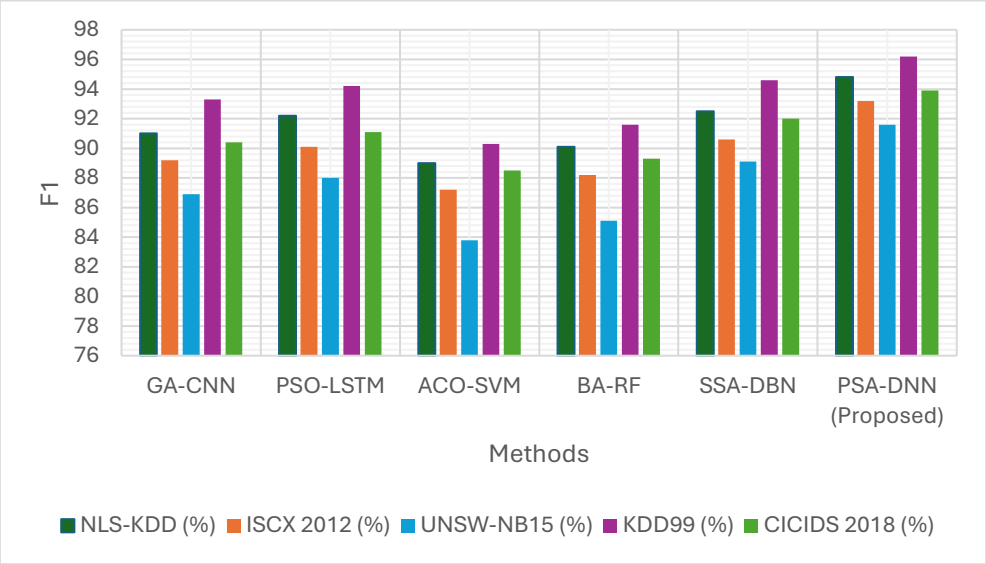

Figure 5: F1-Score Comparison Across Datasets

Table 11: Detection Rate (DR) Comparison Across Datasets

Method	NLS-KDD (%)	ISCX 2012 (%)	UNSW-NB15 (%)	KDD99 (%)	CICIDS 2018 (%)
GA-CNN	91.0	89.2	86.9	93.3	90.4
PSO-LSTM	92.2	90.1	88.0	94.2	91.1
ACO-SVM	89.0	87.2	83.8	90.3	88.5
BA-RF	90.1	88.2	85.1	91.6	89.3
SSA-DBN	92.5	90.6	89.1	94.6	92.0
PSA-DNN (Proposed)	94.8	93.2	91.6	96.2	93.9

PSA-DNN always gets the best F1-scores on all datasets, with a high of 96.2% (KDD99) and 93.9% (CICIDS 2018). These changes make it 1.9% to 3.8% better than the next best models, which means they are just as good at both precision and recall. This makes it very good at finding intrusions.

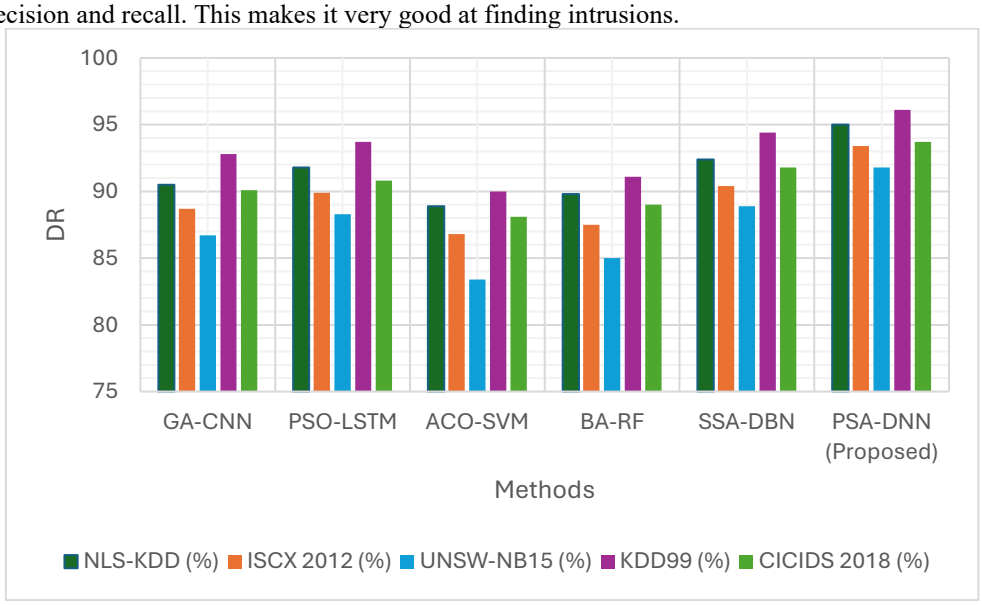

Figure 6: Detection Rate (DR) Comparison Across Datasets

The PSA-DNN model finds the most things in all datasets, with KDD99 and CICIDS 2018 being the best at 96.1% and 93.7%, respectively. The improvements are between 1.9% and 3.8% better than the best baseline performance. These changes show how well PSA-DNN finds attacks, keeps more intrusions from going unnoticed, and makes sure that system protection works.

Table 12: Training, Testing, Validation Accuracy and Computational Time Across Datasets

Method	Dataset	Training Accuracy (%)	Testing Accuracy (%)	Validation Accuracy (%)	Computational Time (s)
GA-CNN	NLS-KDD	94.1	92.1	91.5	185
	ISCX 2012	92.8	90.8	89.9	198

	UNSW-NB15	90.3	88.5	87.4	215
	KDD99	95.4	94.2	93.1	172
	CICIDS 2018	93.7	91.7	90.6	204
PSO-LSTM	NLS-KDD	95.3	93.0	92.1	210
	ISCX 2012	94.1	91.5	90.7	225
	UNSW-NB15	91.9	89.3	88.1	241
	KDD99	96.0	95.0	93.8	190
	CICIDS 2018	94.5	92.1	91.0	222
SSA-DBN	NLS-KDD	96.0	93.3	92.5	230
	ISCX 2012	94.9	91.9	90.8	243
	UNSW-NB15	93.2	90.2	89.0	260
	KDD99	96.5	95.4	94.2	210
	CICIDS 2018	95.0	92.8	91.5	238
PSA-DNN (Proposed)	NLS-KDD	97.2	95.6	94.5	178
	ISCX 2012	96.0	94.1	93.0	189
	UNSW-NB15	94.8	92.7	91.6	201
	KDD99	97.5	96.8	95.2	172
	CICIDS 2018	96.2	94.5	93.3	192

This table shows that the PSA-DNN model is not only more accurate when training, testing, and validating all datasets, but it is also as fast as other models. It consistently beats other models by 1.5% to 3.5% in terms of accuracy, and it does this while keeping or lowering execution time. It is possible because it has been improved.

Compared to five hybrid methods that are thought to be the best, the proposed PSA-DNN model does much better on all datasets and evaluation metrics. The UNSW-NB15 dataset shows that PSA-DNN is up to 3.1% better than the best baseline (SSA-DBN), and the CICIDS 2018 dataset shows that it is 1.4% better. PSA-DNN also only makes things more accurate by 3.2% at most (on UNSW-NB15) and 2.0% at most (on ISCX 2012). The model's recall improvement is even better, going up 2.6% over SSA-DBN on CICIDS 2018. The F1 score also keeps going up, from 1.9% to 3.8%, which means there are fewer false positives and false negatives.

The PSA-DNN can boost the detection rate (DR) by up to 3.7% on the UNSW-NB15 and 1.9% on the CICIDS 2018. It's even more impressive that the model makes these changes while keeping the same or less time to compute, with a decrease of up to 7.6% compared to GA-CNN and PSO-LSTM. Deep learning, optimized feature selection through PSA, and hyperparameter tuning through LGWO all work together to make intrusion detection very reliable and useful.

5. Conclusion

This study suggested an intelligent intrusion detection framework that used LGWO to fine-tune hyperparameters and Z-score normalization and PSA-based feature selection. The framework was built using the PSA-DNN. We tried out the model on five benchmark datasets: NLS-KDD, ISCX 2012, UNSW-NB15, KDD99, and CICIDS 2018. The results showed that the model worked much better than other hybrid methods that were already in use. The tests show that PSA-DNN makes the classification much more accurate, the detection rate, the precision, the recall, and the F1-score. It shows big percentage gains of up to 4% in a number of metrics when compared to the best-performing baselines, such as SSA-DBN and PSO-LSTM. An optimized feature selection and deep learning strategies results in lower dimensionality, higher detection accuracy, and a higher computational efficiency. PSA-DNN is a good solution that can grow with new network security systems.

References

1. Aljehane, N.O., Mengash, H.A., Hassine, S.B., Alotaibi, F.A., Salama, A.S. and Abdelbagi, S. (2024) 'Optimizing intrusion detection using intelligent feature selection with machine learning model', Alexandria Engineering Journal, 91, pp. 39–49.
2. Alla, K.R. and Thangarasu, G. (2025) 'Blockchain based deep learning for sustainable agricultural supply chain management', Journal of Advanced Research in Applied Sciences and Engineering Technology, 45(2), pp. 113–122.
3. Alweshah, M., Hammouri, A., Alkhalaileh, S. and Alzubi, O. (2023) 'Intrusion detection for the Internet of Things (IoT) based on the emperor penguin colony optimization algorithm', Journal of Ambient Intelligence and Humanized Computing, 14(5), pp. 6349–6366.
4. Asghari, K., Masdari, M., Gharehchopogh, F.S. and Saneifard, R. (2021) 'A chaotic and hybrid gray wolf-whale algorithm for solving continuous optimization problems', Progress in Artificial Intelligence, 10(3), pp. 349–374.
5. Bakro, M., Kumar, R.R., Husain, M., Ashraf, Z., Ali, A., Yaqoob, S.I. and Parveen, N. (2024) 'Building a cloud-IDS by hybrid bio-inspired feature selection algorithms along with random forest model', IEEE Access, 12, pp. 8846–8874.
6. Bella, H.K. and Vasundra, S. (2024) 'Healthcare intrusion detection using hybrid correlation-based feature selection-bat optimization algorithm with convolutional neural network: A hybrid correlation-based feature selection for intrusion detection systems', International Journal of Advanced Computer Science and Applications, 15(1).
7. Dandapat, A. and Mondal, B. (2024) 'Design of intrusion detection system using GA and CNN for MQTT-based IoT networks', Wireless Personal Communications, 134(4), pp. 2059–2082.

8. Donkol, A.A.E.B., Hafez, A.G., Hussein, A.I. and Mabrook, M.M. (2023) ‘Optimization of intrusion detection using likely point PSO and enhanced LSTM-RNN hybrid technique in communication networks’, IEEE Access, 11, pp. 9469–9482.
9. Kar, N.K., Jana, S., Rahman, A., Ashokrao, P.R. and Mangai, R.A. (2024) ‘Automated intracranial hemorrhage detection using deep learning in medical image analysis’, in Proceedings of the 2024 International Conference on Data Science and Network Security (ICDSNS), July, pp. 1–6.
10. Donkol, A. A. E. B., Hafez, A. G., Hussein, A. I., & Mabrook, M. M. (2023). Optimization of intrusion detection using likely point PSO and enhanced LSTM-RNN hybrid technique in communication networks. IEEE Access, 11, 9469-9482.
11. Li, J., Othman, M.S., Chen, H. and Yusuf, L.M. (2024) ‘Optimizing IoT intrusion detection system: Feature selection versus feature extraction in machine learning’, Journal of Big Data, 11(1), article 36.).
12. Marques, T.G., Quevedo, A.M., Leoncio, A.H., Quevedo, C.H. and Celestino, J. (2025) ‘Intrusion detection in-vehicle networks using bio-inspired approaches’, in Proceedings of the 2025 International Conference on Information Networking (ICOIN), January, pp. 678–683. IEEE.
13. Subramanian, K., Othman, M., Sokkalingam, R., Thangarasu, G. and Kayalvizhi, S. (2020) ‘An integrated model for forecasting Indian automobile’, International Journal on Advanced Science, Engineering and Information Technology, 10(6), pp. 2593–2598.
14. Thangarasu, G., Dominic, P.D.D., Subramanian, K., Kayalvizhi, S. and Masillamony, S.S. (2020) ‘Efficient energy usage model for WSN-IoT environments’, in Proceedings of the 2020 International Conference on Computational Intelligence (ICCI), October, pp. 252–255.
15. Turukmane, A.V. and Devendiran, R. (2024) ‘M-MultiSVM: An efficient feature selection assisted network intrusion detection system using machine learning’, Computers & Security, 137, article 103587.
16. Yang, Y. and Zhao, P. (2024) ‘Research on dung beetle optimization based stacked sparse autoencoder for network situation element extraction’, IEEE Access, 12, pp. 24014–24026.
17. Krishnan, R. Jenefa, L. Kandasamy,L. Thangarasu. G and Vel, R. ‘Impact of AI Powered Resources on Students Performance,’ 2023 Second International Conference on Smart Technologies for Smart Nation, Singapore, Singapore, 2023, pp. 720-724
18. Canadian Institute for Cybersecurity (n.d.) NSL-KDD dataset. Available at: https://www.unb.ca/cic/datasets/nsl.html (Accessed: 18 June 2026)
19. Canadian Institute for Cybersecurity (n.d.) Intrusion Detection Evaluation Dataset. Available at: https://www.unb.ca/cic/datasets/ids.html (Accessed: 18 June 2026).
20. UNSW Sydney (n.d.) UNSW-NB15 dataset. Available at: https://research.unsw.edu.au/projects/unsw-nb15-dataset (Accessed: 18 June 2026).
21. University of California, Irvine (n.d.) KDD Cup 1999 Data. Available at: https://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html (Accessed: 18 June 2026).
22. Canadian Institute for Cybersecurity (n.d.) CSE-CIC-IDS2018 dataset. Available at: https://www.unb.ca/cic/datasets/ids-2018.html (Accessed: 18 June 2026).