

Optimizing Intrusion Detection: Advanced Feature Selection and Machine Learning Techniques Using the CSE-CIC-IDS2018 Dataset

Qusay M. Alzubi ^{1,*}, Sharif Naser Makhadmeh ², and Yousef Sanjalawe ²

¹Department of Information Technology (Cybersecurity Program), Information Technology and Computer Science Faculty, Yarmouk University, Irbid, Jordan

²Department of Information Technology, King Abdullah II School for Information Technology, University of Jordan (UJ), Amman, Jordan

Email: qusayz@yu.edu.jo (Q.M.A.); s_makhadmeh@ju.edu.jo (S.N.M.); y.sanjalawe@ju.edu.jo (Y.S.)

*Corresponding author

Abstract—The escalation of cyber threats in large-scale local area networks necessitate advanced strategies for efficient anomaly detection and intrusion prevention. This paper explores the integration of sophisticated machine learning techniques and feature selection methods to enhance the performance of Network Intrusion Detection Systems. Focusing on the complex landscape of cyber threats, accentuated by the proliferation of technologies such as Internet of Things, 5G, and cloud computing, the proposed study evaluates the application of three advanced feature selection algorithms—Grey Wolf Optimizer, Bat Algorithm, and Pigeon-inspired Optimization—to identify an optimal subset of features that accurately differentiate between diverse cyberattacks and normal network traffic. Employing the comprehensive CSE-CIC-IDS2018 dataset, the experimental results demonstrate that the feature set was successfully reduced from 80 to subsets of 10, 6, and 7 features while maintaining a high detection accuracy close to 99%. This reduction in feature space significantly decreases computational overhead without compromising detection capability. This research contributes to the cybersecurity domain by presenting a scalable, efficient, and highly accurate model for intrusion detection, setting a foundation for future advancements in Network Intrusion Detection Systems optimization and the broader field of cyber defense mechanisms.

Keywords—feature selection algorithm, network intrusion detection systems, machine learning, cybersecurity, anomaly detection, cyber threats

I. INTRODUCTION

In large-scale Local Area Networks, anomaly detection is mainly used to resolve security issues. However, extracting accurate traffic data to identify anomalies can be demanding [1]. The process of detection of intrusions within computer networks is essential, given its impact on communication and security, but finding these intrusions is not straightforward. The difficulty in network intrusion

detection is the reason the related data is selected to train a sophisticated machine learning framework efficiently [1].

The increase of new threats that are beyond the detection capabilities of the old systems makes the process of finding network intrusions a lot more difficult [2]. Thus, With the increasing complexity of modern cyber threats, particularly driven by the proliferation of technologies such as the Internet of Things (IoT), 5G, and cloud computing, traditional Network Intrusion Detection Systems (NIDS) struggle to detect and mitigate diverse and sophisticated attacks efficiently. Recent advancements, such as [3], underscore the importance of specialized approaches in accurately classifying complex attack types like Remote-to-Local (R2L) and User-to-Root (U2R), which are challenging due to their subtle patterns and low frequency within network traffic. These methods provide valuable insights that enhance the precision and scope of intrusion detection systems in contemporary network environments. Existing models often fail to strike a balance between high detection accuracy and manageable computational overhead, especially when faced with large datasets like CSE-CIC-IDS2018, which contain a wide array of attack vectors [4]. NIDSs as a backup defense behind the firewalls, which are trying to develop strategies to monitor the network continuously, detect harmful network activities, and apply proactive security solutions [4].

The importance of cybersecurity research has risen because of the widespread use of networks nowadays [5]. Even the studies that have been done on existing Intrusion Detection Systems (IDSs) have not been able to discover new attacks, increase the accuracy of detection, and decrease the value of false positives. To overcome these issues, many researchers have been engaged in devising IDSs using machine-learning techniques [5]. Machine learning is pivotal for its accuracy and automation in distinguishing between normal and abnormal behavior, effectively identifying new types of attacks, and offering superior generalization capabilities [5].

Feature Selection (FS) techniques are important since they support data reduction, reduce irrelevant data, getting

precise data models, data set classification, dimensionality reduction [6]. The information on the subject also includes the techniques of the feature subset searching algorithms, which can be used in case the complete feature subset provision is inefficient [6]. Four widely accepted filter, wrapper, embedding and hybrid FS techniques are used in network intrusion detection because of their potential for effectiveness [6].

The KDD CUP'99 challenge dataset results empirically proved the Long Short-Term Memory (LSTM) classifiers' superiority over traditional static classifiers [7]. LSTM classifiers are still an intrusion detection method, as can be seen by their good performance. LSTMs are very powerful stability models that include using the history of data and relationships among connection logs, which is very important to see any interruption [7].

In the Machine learning realm, hyperparameters fundamentally scale and direct the learning process, that is, the model accuracy. This adjustment of the hyperparameters should prove instrumental for the creation of such a myriad of models that are outstandingly capable of learning and possessing various performance speeds depending on the specific Machine Learning (ML) methods used [8].

The rapid expansion of data volume in recent periods has presented considerable obstacles in the classification task. FS addresses these issues by enhancing the accuracy of data classification and simplifying the data complexity [9], as numerous time-intensive features are involved in the detection process.

The effectiveness of the NIDS is primarily determined by the processes of feature selection implementation [10]. The chosen feature selection method impacts the duration required for monitoring traffic patterns and the extent of accuracy enhancement. There are four main strategies for feature selection: wrapper, hybrid, filter, and embedded techniques [10].

The key contribution of this paper is its effort to identify an optimal subset of features that excel in differentiating between diverse attacks and normal network traffic to minimize the total number of features used. To achieve this objective, the study employed a range of search methods for feature selection, utilizing algorithms such as Grey Wolf Optimizer (GWO), Bat Algorithm (BA), and Pigeon-inspired Optimization (PIO). Consequently, in selecting features, groups of 10, 6, and 7 were chosen from an initial pool of 67 features. It is significant to note that, during the cleaning data phase, 13 features were eliminated from the original set of 80.

This study focuses on binary classification, which involves distinguishing between benign (normal) traffic and diverse (attack) traffic. The CSE-CIC-IDS2018 dataset contains multiple types of attacks; however, for the purposes of this research, all attack types were categorized under a single "diverse" label. The model's primary goal is to determine whether a given network traffic instance is either normal or represents any form of attack.

The main contributions of this study are summarized as follows:

1. **Novel Use of Advanced Feature Selection Algorithms:** The study applies three sophisticated feature selection algorithms—GWO, BA, PIO—to reduce the feature set of the CSE-CIC-IDS2018 dataset. These algorithms effectively identify the most relevant features for differentiating between normal traffic and various cyberattacks, leading to a more streamlined and efficient detection process.
2. **Effective Feature Space Reduction with High Accuracy:** The proposed method successfully reduces the feature set from an initial 80 features to subsets containing 10, 6, and 7 features. This reduction minimizes computational overhead while maintaining a high detection accuracy of nearly 99%, ensuring efficient yet effective intrusion detection.
3. **High Performance in Detecting Complex Cyberattacks:** The proposed model demonstrates superior performance in detecting a wide range of cyberattacks, including complex ones, thanks to integrating advanced feature selection and machine learning techniques. This ensures that the system can accurately distinguish between normal and malicious traffic.

The remaining part of the paper is organized as follows: Section II discusses the related and recently published works; Section III deeply explains the methodology of feature selection algorithms to enhance IDS. The experimental design and the results obtained by different feature selection algorithms are represented in Section IV. Section V shows the potential applications of the proposed method. Section VI shows the limitations of the proposed method. Finally, Section VII shows the conclusion and future work of the paper.

II. LITERATURE REVIEW

Early research on intrusion detection focused on traditional machine-learning techniques like Decision Trees. For instance, Ingre *et al.* [11] using Decision Trees, an accuracy of 93.45% was achieved on the Network Security Laboratory-Knowledge Discovery and Data Mining (NSL-KDD) dataset, but the method suffered from high time complexity due to the large feature set size, with a time complexity of $O\left(\frac{n}{\log n}\right)$, where n is the number of records.

Later studies integrated ensemble learning with dimensionality reduction methods such as Principal Component Analysis (PCA) to improve both accuracy and computational efficiency. For example, Ferrag *et al.* [12] reported an accuracy improvement of 96.89% on large datasets while reducing the time complexity to $O(m*n)$, where m is the number of features selected. This method significantly improved scalability but added a trade-off in terms of increased model complexity.

Bio-inspired algorithms like Genetic Algorithm (GA) and Particle Swarm Optimization (PSO) have also been applied to feature selection in intrusion detection. In Ref. [13], GA achieved an accuracy of 97.2% with a time complexity of $O(g*p*n)$, where g is the number of generations and p is the population size. While PSO, as

reported by MahdaviFar and Ghorbani [14], offered slightly faster convergence than GA, its accuracy dropped to 96.1%. Both algorithms improved detection but introduced complexity regarding parameter tuning and convergence speed.

Recent advancements in hybrid ML methods have shown promising improvements in both accuracy and efficiency. For example, in Ref. [15], a hybrid model combining GWO and Random Forest achieved an accuracy of 98.5% on the CSE-CIC-IDS2018 dataset. The time complexity for GWO is $O(N \times d)$, where N is the number of iterations and d is the number of features. GWO's lower complexity compared to GA and PSO and its higher accuracy position as a state-of-the-art solution for intrusion detection systems.

Althubiti *et al.* [16] investigates the use of Long-Short-Term Memory (LSTM) recurrent neural networks to detect intrusions in IoT environments. The research focuses on real-time anomaly detection using sequence learning, leveraging the temporal nature of network traffic data.

This study [3] aimed at improving intrusion detection systems, particularly in detecting challenging attack types such as U2R and R2L attacks, which existing models struggle with. The authors propose a solution based on feature selection using Shapley values and LSTM networks. The NSL-KDD dataset is used to train and evaluate the model. By reducing the feature set for each attack class before classification with LSTM, the proposed model achieves superior accuracy, precision, recall, and F1-Score compared to LSTM with all features and other state-of-the-art methods, particularly improving the detection of R2L and U2R attacks.

Ma *et al.* [1] presents a hybrid neural network for anomaly detection by analyzing multi-type flow features. It addresses the feature selection challenge for optimizing IDS performance, particularly in large-scale networks like the IoT and cloud environments.

Prasad and Chandra [17] focused on the datasets used in machine learning-based intrusion detection research, including the CSE-CIC-IDS2018 dataset, which is central to this study. The paper outlines the challenges in using large datasets for IDS, including scalability, computational cost, and the need for feature selection.

Cieslak *et al.* [18] employed techniques from machine learning and data mining to address network security breaches. It concentrated on enhancing the rate of accurately detected intrusions and minimizing the incidence of false positives. Cieslak *et al.* [18] study how the Repeated Incremental Pruning to Produce Error Reduction (RIPPER) algorithm is employed in building rules for datasets that suffer from instability since they have an imbalance between intrusion instances. Those meet this goal by combining a blend of remedial techniques, including oversampling, under-sampling, and clustering, thus helping avoid challenges related to data disproportion. Analysis conducted on datasets derived from real-world intrusions revealed the superiority of synthetic data augmentation over mere replication techniques. Additionally, it was observed that a method based on clustering further amplified the capabilities of

intrusion detection compared to synthetic augmentation alone. The paper underscores that an IDS is a pivotal network security mechanism that surveils malevolent activities across computer systems and networks. Nevertheless, the intricate and ever-changing nature of cyber threats often outstrips the capabilities of traditional IDS methods. Hence, Cieslak *et al.* [18] advocates for deploying adaptable, machine learning-driven approaches that promise to curtail the rate of false positives, elevate the rate of successful detection, and ensure the computational demands remain within reasonable limits.

Previous comparative analyses have been conducted on various methodologies rooted in Computational Intelligence (CI) and conventional Artificial Intelligence (AI). These studies have emphasized the potential of CI characteristics in the creation of efficient Intrusion Detection Systems (IDSs) [19].

Pervez and Farid [20] analyzed the accuracy of different classifiers, focusing on how they perform with features of high dimensions. Shapoorifard and Shamsinejad [21] developed a new strategy that integrates various classification methods tailored for the multiclass NSL-KDD Cup99 dataset, specifically employing Support Vector Machines (SVM) in their methodology. Concurrently, Shapoorifard and Shamsinejad [21] explored innovative solutions to boost the classification accuracy of Center and Nearest Neighbors (CANN) in IDSs, with their effectiveness tested on the NSL-KDD Cup99 dataset.

Bhattacharya *et al.* [22] with the proliferation of the internet and many networks using it, intentional and malicious cyber-attacks are now profound, as portrayed in the discussion. It reveals the key role that systems with strong IDS and data analytics play in predicting and detecting cybersecurity issues. The study presents hybrid technique by combining ML algorithms, ones inspired by fireflies, with a feature selection technique called PCA for the correct classification of IDS datasets. Such framework employs the application of a variant of PCA-based firefly algorithm for dimensionality reduction and classifying data using the XGBoost model running on a dataset pre-processed through the usage of One-Hot encoding. This proposed framework achieved in the experiment goes beyond testing for existing algorithms of the current ML state.

Farhan *et al.* [23] reduced the feature set from an original 80 to 55, which led to attaining a final accuracy rate of 95%. This was accomplished by employing a combination of the Binary Particle Swarm Optimization (BPSO) and Deep Neural Network (DNN) methodologies alongside a classification technique.

Lama and Savant [24] outlined a machine learning-driven NIDS framework for the binary categorization of the CSECIC-IDS2018 dataset utilizing Amazon Web Services (AWS) data. They employed random forest, logistic regression, and gradient boosting as abuse detection methodologies. The findings indicated that gradient boosting was superior in performance. Applying the model on a holdout dataset showed its wide influence on different psychic studies.

The gradient-boosted model achieved outstanding recall and precision scores of 0.98, and thus, it can be used as a tool in the classification of diseases in real-life settings. The development of the next steps could involve the use of synthetic minority oversampling classifiers to identify infrequent attacks even in minority categories and further studies on new datasets which is different from those used in this research, in particular, the CIC-IDS-2017 dataset [24].

Within the contribution, Kwon *et al.* [25] theorized the use of deep learning techniques for anomaly detection in network intrusion detection systems. The authors covered the broad spectrum of anomaly detection methods, such as data and feature dimensionality reduction, classification techniques, and the incorporation of deep learning models. Through survey [25], the authors explored several papers covering deep learning methods to detect network anomalies, bringing to attention these approaches' strong and weak sides. For one, their localized test was performed using network traffic analysis using a Fully Convolutional Network (FCN) model. FCN, in turn, detected minor, infrequent anomalies with the highest accuracy compared to the traditional algorithms, including ones like SVM, Random Forest, and AdaBoost. IDS plays a vital role in cybersecurity as it is the one that is deployed to detect and respond to cyber-attacks and malicious activities inside an organization's network infrastructure.

Tama *et al.* [26] studies develop a new IDS system that combines a two-level classifier ensemble with a hybrid approach to feature selection. This feature selection technique became popular because the PSO, Ant Colony Algorithm (ACA), and GA are combined to decrease the number of features in the datasets (UNSW-NB15 and NSL-KDD). These features are selected based on their influence on the classifier's performance, in this case, the Reduction Error Pruning Tree (REPT). The REPT classifier, classified by its two-level ensemble structure, uses a bolstering-meta-learner and a rotation-forests calculator to track its classification ability. As proven by superior results against the latest classification algorithm, the system has reached an 85.8% accuracy, 88.0% detection, and 86.8% sensitivity on the NSL-KDD dataset and about 85% on the UNSW-NB15 dataset. On the other hand, the reliability of the obtained experimental results was validated in detail using a two-step statistical significance testing. This test was deployed to assess the importance of the classifier in related IDS research.

The AI-based IDS model implemented in [27] allowed the system to improve its detection capabilities via an

ANN model. This rate of sensitivity was attained with the observation of 100% for the test dataset, and undoubtedly, the efficiency has been improved in reducing false positives. Ayachi *et al.* [27] combined a model offline attack strategy with recent dataset to extract the unique patterns of web application attacks. The next step is to apply this method to the recent dataset by the Canadian Institute for Cybersecurity (CI-RACIC-DoHBrw-2020), which requires enhancing the mechanism to enable real-time computation. Then, the system will be integrated into an online IDS to measure its performance with online network data.

The effectiveness of via diverse machine learning methodologies has been observed, such as the BLS, the RBF-BLS system, and a system with cascades of mapped features with enhancement Nodes-BLS. This evaluation had a more specified session that was analyzed with detailed malicious intrusions and network communication anomalies using CICID 2017 and CSECID 2018 subsets, which comprise the DoS data. As was indicated by the results shared in [28], for both models, the precision and Recall remained quite stable even with a less significant set of the features implemented.

Most of the evaluated models achieved accuracy and F1-Scores exceeding 90%, even though an increased count of mapped features and enhancement nodes led to higher memory requirements and longer training durations. Notably, the incremental BLS variant showcased a significant reduction in training time when compared to the traditional non-incremental BLS approach [28].

The studies referenced have achieved remarkable accuracy levels. However, a significant challenge identified is the reliance on either a substantial number of features to attain high accuracy in detecting network intrusions or the inability to achieve such accuracy with a limited number of features. Another concern is the model-building times, an important factor in intrusion detection systems. In instances where high accuracy was reported, the studies often did not disclose the model-building times, or the times were excessively long. This research addressed both critical aspects: achieving very high intrusion detection accuracy with minimal features and minimizing the model-building duration. The results were extremely accurate, almost 99%, achieved with the fewest features and the shortest model building time. Table I summarizes related work and highlights the distinctions between the proposed approach and existing models.

TABLE I. SUMMARY OF LITERATURE REVIEW

Ref.	Aim	Methods/Algorithm	Dataset	Main Findings	Shortcomings
[10]	Intrusion detection using Decision Trees	Decision Trees	NSL-KDD	Achieved 93.45% accuracy but high time complexity	High time complexity
[11]	Improve accuracy and efficiency using ensemble learning with PCA	Ensemble learning, PCA	Large datasets	Improved accuracy to 96.89%, reduced time complexity	Increased model complexity
[12]	Apply Genetic Algorithm for feature selection in IDS	Genetic Algorithm	Custom dataset	Achieved 97.2% accuracy with high computational complexity	High parameter tuning complexity
[13]	Use PSO for faster convergence in IDS feature selection	PSO	Custom dataset	Faster convergence but reduced accuracy compared to GA	Reduced accuracy

[14]	Hybrid GWO and Random Forest for IDS	Grey Wolf Optimizer, Random Forest	CSE-CIC-IDS2018	Achieved 98.5% accuracy with lower time complexity	High memory requirements, long training duration
[15]	Use LSTM for real-time anomaly detection in IoT	LSTM Recurrent Neural Network	IoT traffic data	Real-time anomaly detection with sequence learning	High time complexity
[16]	Improving IDS detection of U2R and R2L attacks using LSTM and Shapley values	LSTM, Shapley values	NSL-KDD	Improved detection of U2R and R2L attacks	Higher memory and training time
[1]	Hybrid neural network for anomaly detection in large networks	Hybrid neural network	IoT and Cloud environments	Optimized feature selection for IDS performance	High parameter tuning complexity
[17]	Survey of datasets in IDS research, focus on CSE-CIC-IDS2018	Survey, PCA, Feature selection	Various IDS datasets	Challenges in using large datasets for IDS	Scalability, the computational cost
[18]	Enhancing detection rate and minimizing false positives in network security	RIPPER algorithm, Clustering	Real-world intrusion datasets	Synthetic augmentation improves detection over replication	Imbalance between intrusion instances
[19]	Comparative analysis of CI and AI methods for IDS	CI and AI methods	NSL-KDD Cup99	CI methods are promising for IDS	High parameter tuning complexity
[20]	Classifying features with SVM for multiclass NSL-KDD	SVM	NSL-KDD Cup99	SVM improves classification accuracy	High memory requirements, long training duration
[21]	Hybrid PCA and XGBoost for feature selection and classification in IDS	Firefly, PCA, XGBoost	Custom IDS dataset	Firefly-PCA-XGBoost improves classification	Higher memory and training time
[22]	BPSO and DNN for feature reduction and IDS	DNN	Custom dataset	Feature reduction to 55 features, 95% accuracy	Imbalance between intrusion instances
[23]	Binary classification of CSE-CIC-IDS2018 using random forest and logistic regression	Random forest, logistic regression	CSE-CIC-IDS2018	Gradient boosting is superior for abuse detection	Not tested on minority categories
[24]	Deep learning techniques for anomaly detection in NIDS	FCN	Various IDS datasets	Deep learning outperforms traditional methods	High time complexity
[25]	Two-level classifier ensemble for feature selection in IDS	PSO, Ant Colony, Genetic Algorithm, REPT	NSL-KDD, UNSW-NB15	PSO, ACO, GA reduce feature size and improve accuracy	High memory requirements, long training duration
[26]	ANN model for improving IDS detection capabilities	ANN, offline attack strategy	Custom dataset, CI-RACIC-DoHBrw-2020	100% detection rate with ANN model	Not integrated for real-time computation
[27]	BLS and RBF-BLS system for IDS	BLS, RBF-BLS, Enhancement nodes	CICID 2017, CSECID 2018	Stable precision and recall reduced training time	Higher memory and training time
[28]	IDS model accuracy with reduced features and training time	BLS with feature reduction	Custom dataset	High accuracy with fewer features and shorter training times	High time complexity
Proposed	IDS model accuracy	GWO, BA, and PIO combined with Machine Learning	CSE-CIC-IDS2018	99% accuracy	Multiclass classification

III. METHODOLOGY

This paper delineates a robust methodology tailored for enhancing the performance of NIDSs through the meticulous application of three advanced feature selection algorithms: GWO, BA, and PIO.

The methodology is a step-by-step process of data cleaning, feature selection, and classification testing using the CSE-CIC-IDS2018 dataset. First, the dataset is meticulously cleaned to remove irrelevant features and reduce the dimensionality, thus creating a refined stage for applying the feature selection algorithms.

These algorithms are supposed to find the most important features that effectively distinguish the various types of cyberattacks and benign activities. After the feature selection, different machine learning classifiers are used to check the effectiveness of the selected features in detecting network intrusions. Thus, they seek the best balance between accuracy and computational efficiency. This organized way strives to improve detection ability and provides a scalable model for the adjustment to changing cyber threats. Fig. 1 shows the main steps for the proposed method.

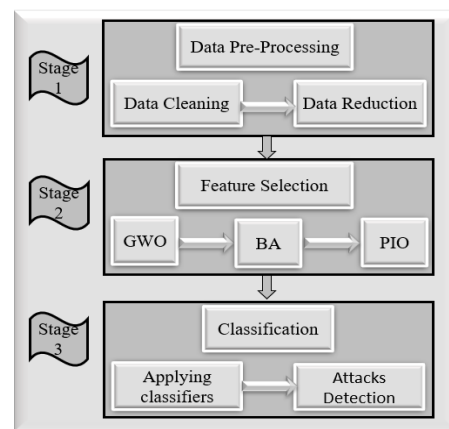


Fig. 1. Main steps for the proposed method.

The main steps for proposed methods:

1. Data Preprocessing:

- a. Data Cleaning: Remove irrelevant or redundant features (e.g., Protocol and Timestamp) and handle any inconsistent data, such as negative or zero values in fields like Flow_Duration and Fwd_Header_Length.

- b. Data Reduction: Eliminate additional non-essential features that do not contribute meaningfully to intrusion detection, reducing the dimensionality to ensure a focused dataset.

2. Feature Selection

- a. Application of Feature Selection Algorithms: Apply three advanced algorithms—GWO, BA, and PIO—to identify the most relevant features for intrusion detection.
- b. Optimal Feature Subset Creation: From the initial feature set, each algorithm produces a reduced subset (e.g., 10, 6, or 7 features) that balances high detection accuracy with minimised computational load.

3. Classification

- a. Model Training: Use the selected feature subsets to train multiple machine learning classifiers, such as Decision Tree, BayesNet, and Logistic Regression.
- b. Model Evaluation: Assess each classifier’s performance in terms of accuracy, precision, recall, and F-measure to determine the optimal combination of feature subset and classifier.
- c. Intrusion Detection Output: Generate detection results, where the trained model categorises network traffic as either benign or attack, based on the optimized feature set.

A. Dataset

The dataset utilized in this research is the CSE-CIC-IDS2018 dataset, a comprehensive and modern dataset created by the Canadian Institute for Cybersecurity (CIC) and the Communications Security Establishment (CSE). It is widely recognized for its diversity and coverage of real-world cyberattack scenarios. This dataset consists of 10 Comma-Separated Values (CSV) files representing network traffic data collected over ten consecutive days of network monitoring. It contains approximately 16.2 million records and features a wide variety of attack types, including Botnet, Denial-of-Service (DoS), Distributed Denial-of-Service (DDoS), Brute Force, Web Attacks, Infiltration, and more [29]. The CSE-CIC-IDS2018 dataset contains 80 features that capture detailed network traffic characteristics, such as packet size, inter-arrival time, flow duration, and various flag counts. These features are vital for detecting and differentiating between legitimate network traffic and malicious activities. Furthermore, the dataset includes labelled data, with each record categorized as benign or belonging to one of the 14 attack types. This labelling is critical for training and testing machine learning models used in network intrusion detection [29–32].

After preprocessing and data cleaning, this study reduced the dataset to include 67 relevant features, which were subsequently used for feature selection and model training. 80% of the dataset was allocated for training and 20% for testing the machine learning models. This split ensured that the models were trained on a substantial portion of the data while rigorously evaluating unseen traffic data to assess generalization and performance.

The CSE-CIC-IDS2018 dataset was chosen for this study because it is comprehensive, up-to-date, and represents a wide range of modern cyber threats. Including benign and attack scenarios makes it ideal for evaluating the effectiveness of feature selection techniques and machine learning algorithms in detecting complex cyber threats. Table II represents the type of attack in the dataset CSE-CIC-IDS-2018.

TABLE II. REPRESENTS THE TYPE OF ATTACK IN THE DATASET CSE-CIC-IDS-2018

Attack Category	Features
Web Attack	Web
	XSS
	SQL Injection
Dos	Hulk
	SlowHTTPTest
	GoldenEye
Infiltration	Slowloris
Brute Force	Infiltration
	FTP
Botnet	SSH
	Bot
DDos	HOIC
	LOIC-HTTP
	LOIC-UDP

B. Data Cleaning

Multiple preprocessing is implemented during the data cleansing phase for the CSE-CIC-IDS2018 dataset. Initially, certain fields considered to be redundant or irrelevant were discarded. Specifically, the protocol column is removed, as the information it contained was already sufficiently represented by the values in each destination port’s Dst_Port (Destination Port) field.

Furthermore, the Timestamp parameter is excluded to promote impartial learning and eliminate distinctions based on time, enabling the models to distinguish between large-scale and stealthy attacks accurately. The omission of the timestamp attribute also simplified the merging or segmenting of the dataset for use in the experimental setups.

The preprocessing stage filters out 59 duplicate record header row instances by applying a filter with an empty list for acceptable label values. Particularly, the file ‘Tuesday-20-02-2018_TrafficForML_CICFlowMeter.csv’ [33] was unique compared to the other nine files in the benchmark dataset. From this specific file, four fields are removed: Source Internet Protocol (SrcIP), FlowID, Destination Internet Protocol (DstIP), and Source Port (SrcPort). Any negative entries are discarded in the Flow_Duration, Fwd_Header_Length, and Flow-IAT-Min columns, as these did not align with the expected positive values in their columns. Such negative entries, particularly in the Fwd_Header_Length feature, can skew the data, potentially affecting statistics susceptible to outliers. Moreover, records are deleted where eight fields had zero values. In Table III, it is shown that all features were removed before the feature selection process.

TABLE III. LIST OF FEATURES THAT WILL BE REMOVED BEFORE FEATURE SELECTION

No.	Feature
1	Fwd_Avg_Bytes_Bulk
2	Bwd_Avg_Bulk_Rate
3	Fwd_Avg_Bulk_Rate
4	Bwd_Avg_Packets_Bulk
5	Fwd_Avg_Packets_Bulk
6	Bwd_PSH_Flags
7	Bwd_URG_Flags
8	Bwd_Avg_Bytes_Bulk

The fields `Init_Win_bytes_backward` and `Init_Win_bytes_forward` were excluded from the analysis because around 50 percent of their values were negative, which was not suitable for the variables.

It is also decided against using the `Flow_Duration` fields due to the presence of values that were implausibly low or zero. Furthermore, less than 0.6 percent of the data points in the `Flow Bytes/s` and `Flow Packets/s` columns contained “Infinity” or “NaN” values, leading us to remove any instances where the bytes or packets per second recorded such values.

Upon finalizing the data cleansing process and prior to initiating feature selection, an unsupervised methodology is employed alongside the discretized filter technique for data refinement. This included utilizing an instance filter named “Discrete,” which was instrumental in transforming several numerical characteristics of the dataset into nominal attributes.

The CSE-CIC-IDS2018 dataset comprises ten files, each representing data collected on successive days. Initially, the merged dataset included 16,232,943 entries and 80 different features. After the data cleaning phase, the dataset was refined to include 16,137,168 entries with 67 features. This refined dataset was used for the subsequent data filtering stage. Out of the refined records, 3,227,433 records, representing 20 percent of the dataset, were set aside for system testing and evaluation. The remaining 12,909,735 records, accounting for 80 percent, were allocated for system training.

Table IV. displays a visual representation, showing both the number and percentage of diverse types of traffic before and after the data cleaning process was concluded. The table also illustrates each traffic class’s count and proportion before and after the data-cleaning process.

TABLE IV. STATISTICS OF TRAFFIC CLASS BEFORE AND AFTER CLEANING

Traffic Class	Before Data Cleaning	After Data Cleaning	Percentage after Data Cleaning
Benign	13,484,708	83.070%	13,390,234 82.978%
Bot	286,191	1.76%	286,191 1.773%
DDoS	1,263,933	7.787%	1,263,933 7.832%
Infiltration	161,934	0.998%	160,639 0.995%
Web Attack	928	0.0055%	928 0.0055%
Dos	654,300	4.032%	654,300 4.032%
BruteForce	380,949	2.347%	380,943 2.361%
Total	16,232,943		16,137,168

C. Data Filtering

In the field of data mining, maintaining high data quality for the following stages is crucial. Data preprocessing is

key to achieving this goal, focusing on enhancing the outcomes of mining processes by cleaning collected data. This includes tasks like transforming data and reducing its dimensionality. Discretization involves converting continuous or numerical variables into discrete or nominal categories by segmenting them into limited intervals [17, 34].

Within the realm of data mining, prominent algorithms include machine learning methods such as naive Bayes, Apriority, and decision trees, which all benefit from the application of data discretization [35–37].

Utilizing data discretization allows these methods to develop models that are not only more effective but also more efficient. Furthermore, converting continuous attributes into discrete ones results in a format that is more congruent with human understanding, rendering these models more straightforward to grasp, employ, and elucidate compared to their continuous attributes [35].

In their comparative research, Brajević and Stanimirović [37] analyzed the classification efficacy of 30 algorithms that had undergone discretization, employing a rule-based decision tree of slower execution and Bayesian learning methods. Their findings offer insight into which discretization techniques are best suited for specific types of classifiers, those compatible across various classifier models, and those that achieve an optimal balance between the number of intervals created and the classification accuracy. Data discretization involves segmenting the range into brief intervals that exhibit significant consistency within classes to identify a series of cut points for continuous attributes.

Moreover, the discretization methods aim to generate the minimum number of intervals, which still guarantees strong links between classes and attributes. This is accomplished by discretizing continuous features into k segments or parts, which enables $(k-1)$ divisions along the line segment. It is necessary to determine what proportions between arity (the number of intervals) and the accuracy of the analysis are the most beneficial [37].

The following states the discretization algorithm applied to a dataset containing N instances with distinct C targets. This includes a discretization algorithm that transforms a continuous attribute A into m discrete segments denoted by $D_s = [d_{s0}, d_{s1}], [d_{s1}, d_{s2}], \dots, [d_{sm-1}, d_{sm}]$ where d_{s0} is the minimum value, d_{sm} is the maximum value, and each discrete segment terminates where $d_{si} < d_{si+1}$, for all $i = 0, 1, \dots, m-1$. D_s is the discrete segment of an approximation, and the set of the division points for A is indicated by $P = (d_{s1}, d_{s2}, \dots, d_{sm-1})$ [28].

The discretization process generally encompasses four key steps: sorting the values of the continuous attribute, determining and evaluating potential points for dividing or combining segments, separating or amalgamating consistent segments based on specific criteria, and concluding the process at a predetermined point [35].

Discretization methods can be differentiated by various attributes [35, 36, 38] as outlined here:

Static versus Dynamic: Static discretization takes place prior to and independently of the learning algorithm and task; conversely, dynamic discretization is implemented

during model creation. Static discretization is more frequently employed, with the discretized Iterative Dichotomiser 3 (ID3) decision tree as an example of dynamic discretization.

When distinguishing between univariate and multivariate discretization, the former focuses on analyzing a single continuous attribute simultaneously, whereas the latter evaluates multiple attributes in conjunction.

Supervised and unsupervised discretization techniques vary in identifying suitable intervals for discretization. Supervised techniques use class labels to inform interval choices, considering the class association of the data. Conversely, unsupervised techniques proceed without regard to class labels in data discretization. It's noteworthy that most discretization approaches are supervised.

D. The Selection of Features

Feature selection is a critical aspect of machine learning that significantly influences classification effectiveness. Selecting an appropriate set of features can greatly enhance the accuracy of classification activities. This research implements three distinct feature selection approaches: the GWO, BA, and PIO algorithms, each responsible for choosing a different set of features. Following the data cleansing process, the benchmark dataset was refined to contain 67 features. Three different algorithms are applied.

1) Grey wolf optimizer

This study introduces an algorithm based on swarm intelligence called GWO, inspired by the hunting behavior of wolf packs. The GWO approach develops a mathematical representation of the tactics grey wolves employ to locate, surround, and ultimately hunt prey [39]. Initially, the algorithm acknowledges the distinct social hierarchy within a wolf pack, assigning wolves into four categories reflecting their hierarchical status. At the top of this hierarchy are the Alpha, Beta, and Delta wolves, recognized as the pack's leaders with presumed enhanced capabilities. Below them are the Omega wolves, considered subordinate, adhering to the guidance set forth by their superiors Fig. 2.

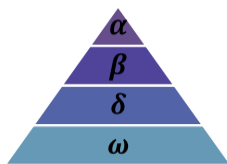


Fig. 2. The social structure of GWO [39].

With the remaining wolves, referred to as ω wolves, acting as subordinates.

The hunting behavior of grey wolves encompasses three principal phases: searching for prey, encircling and harassing the prey until it stops moving, and finally, attacking it. This process of encirclement can be captured through a mathematical framework where the positions of the wolves are adjusted based on the prey's location. Eqs. (1)–(4) are utilized to update the wolves' positions for the next iterations.

$$r_1 \in [0,1] \quad r_2 \in [0,1] \quad a = 2 \left(1 - \frac{t}{T}\right) \quad (1)$$

The parameter a controls the balance between exploration and exploitation. It decreases linearly from 2 to 0 as the number of iterations t approaches the maximum number of iterations T . The decrease in a guides the algorithm from exploration (searching for global optima) to exploitation (fine-tuning around a local optima).

$$A = 2ar_1 - a, \quad C = 2r_2 \quad (2)$$

A is a parameter used to control the impact of the wolves' positions during the hunting phase. The value is influenced by a and r_1 , a is a random number uniformly distributed between 0 and 1.

C is another parameter that introduces randomness to the search process through r_2 , another random number uniformly distributed between 0 and 1. This randomness ensures the wolves do not converge too quickly on local optima.

$$D = |CX_p(t) - X(t)| \quad (3)$$

D represents the distance between the current wolf position $X(t)$ and the position of the prey $X_p(t)$, which is approximated based on the positions of the alpha, beta, and delta wolves. C adds randomness to the distance calculation, helping to avoid premature convergence.

$$X(t + 1) = X_p(t) + A \cdot D \quad (4)$$

This equation updates the position of a wolf based on its current position and its estimated distance from the prey. A determines the step size, while D reflects the distance to the prey.

The underlying premise of the equations mentioned earlier is the presumption that the prey's location is known. However, in the theoretical realm of an abstract search space, particularly when optimizing a function, the exact position of the prey remains unknown. The GWO operates on the concept that the top three agents in the search space, namely the alpha, beta, and delta wolves, are presumed to be in proximity to or moving towards the optimal solution. Under this hypothesis, the remaining wolves adapt their positions by aligning with the perceived location of the prey, which is determined by averaging the positions of these leading wolves (as illustrated in Fig. 3 and described in Eqs. (5)–(11).

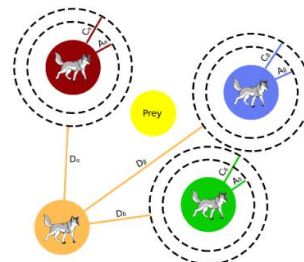


Fig. 3. Update mechanism for wolves [40].

Fig. 3 illustrates that the wolves' positions are updated according to the estimated location of the prey (indicated by a solid red circle), calculated as the average of the

positions of the three leading wolves. The modification of positions is influenced by stochastic parameters A and C, determined through Eq. (2). This adjustment primarily pertains to the exploitation phase, wherein the wolves' positions are updated nearer to those of the best-performing agents.

$$D_\alpha = |C_\alpha \cdot X_\alpha - X(t)| \quad (5)$$

This represents the distance between the current position $X(t)$ of the wolf and the position X_α of the alpha wolf. C_α is a random coefficient that adjusts the influence of the alpha wolf on the wolf's position update.

$$D_\beta = |C_\beta \cdot X_\beta - X(t)| \quad (6)$$

This represents the distance between the current position $X(t)$ of the wolf and the position X_β of the beta wolf. C_β serves the same role as C_α , guiding the wolf towards the beta wolf.

$$D_\delta = |C_\delta \cdot X_\delta - X(t)| \quad (7)$$

This is the distance between the current position $X(t)$ and the position X_δ of the delta wolf. C_δ controls the influence of the delta wolf.

$$X_1 = X_\alpha(t) - A_\alpha \cdot D_\alpha \quad (8)$$

This equation updates the position of the wolf based on its distance from the alpha wolf, where A_α is a vector that controls the magnitude of the adjustment in the direction of the alpha wolf.

$$X_2 = X_\beta(t) - A_\beta \cdot D_\beta \quad (9)$$

This updates the wolf's position relative to the beta wolf. A_β adjusts the influence of the beta wolf on the position update.

$$X_3 = X_\delta(t) - A_\delta \cdot D_\delta \quad (10)$$

This updates the wolf's position relative to the delta wolf. A_δ governs the contribution of the delta wolf to the position update.

$$X(t+1) = \frac{X_1 + X_2 + X_3}{3} \quad (11)$$

This is the final position update of the wolf, which is an average of its distances from the alpha, beta, and delta wolves. By averaging these three positions, the wolf is guided toward the most promising area in the search space. Algorithm 1 illustrates the GWO algorithm.

Algorithm 1. Grey Wolf Optimizer

Require: objective_function, max_iter, num_wolves, search_space

Ensure: alpha position (best solution)

- 1: **Initialize:**
- 2: wolf positions = randomly generate positions for num_wolves within the search space
- 3: alpha, beta, delta = positions with the top three lowest objective function values
- 4: alpha score, beta score, delta score = corresponding objective function values
- 5: **for** iter in 1 to max_iter **do**
- 6: **for** each wolf in wolf positions **do**

- 7: Calculate A, C for the current iteration using Eq. (2)
- 8: Update the position towards alpha, beta, and delta:
- 9: **for** each dimension d **do**
- 10: Calculate $X(t+1)$ for the current iteration using Eq. (11)
- 11: **end for**
- 12: Ensure $X(t+1)$ remains within the search space
- 13: **end for**
- 14: Evaluate the objective function for all wolf positions
- 15: Update alpha, beta, delta if better scores are found
- 16: **end for**
- 17: Return alpha position (position with the best objective function value)

Using a simulated GWO, the selected features are listed in Table V.

TABLE V. FEATURES SELECTED BY APPLYING THE GWO

No.	Features
1	Dst Port
2	Fwd Pkt Len Min
3	Flow Pkts/s
4	Bwd Pkts/s
5	Fwd IAT Min
6	ECE Flag Cn
7	ACK Flag Cnt
8	Fwd Seg Size Min
9	Fwd Act Data Pkts
10	Idle Std

Table VI shows the detailed summary of the key parameters used for the GWO algorithm, along with their initial values and roles in the feature selection process:

TABLE VI. DETAILED SUMMARY OF THE KEY PARAMETERS USED FOR THE GWO ALGORITHM

P.	Description	Initial Value	Sensitivity Impact
a	Exploration-exploitation trade-off parameter. Starts at 2, decreases linearly.	2 (decreases to 0)	High impact: Higher values lead to faster convergence but can reduce accuracy.
c	Convergence coefficient controlling influence on exploration.	Rand [0, 2]	Moderate impact: Higher values improve exploration, but excessive exploration can delay convergence.
r_1, r_2	Random numbers between 0 and 1 for stochastic updates in position calculations.	Rand [0, 1]	Low impact: Small fluctuations do not significantly affect performance.

2) Bat algorithm

The Bat Algorithm, inspired by the echolocation mechanism employed by bats to detect prey, offers a novel computational approach to solving such problems. The algorithm's success hinges on its mathematical model, which captures the essence of bats' navigation and foraging strategies. Here, the primary equations outline that constitutes the BA and discuss their roles in guiding the algorithm's search process [41].

3) Mathematical formulation

Several key equations define the operational framework of the BA, each contributing to the algorithm's ability to simulate bat behavior:

- **Frequency Update:**

Every bat's echolocation frequency, f_i , is dynamically adjusted using the Eq. (12):

$$f_i = f_{min} + (f_{max} - f_{min}) \cdot \beta \quad (12)$$

here, f_{min} and f_{max} denote the minimum and maximum frequency limits, respectively, while β is a random number between 0 and 1.

• **Velocity and Position Update:**

The velocity $v_i^{(t+1)}$ and position $x_i^{(t+1)}$ of bat i at the next iteration $t+1$ is updated using the Eqs. (13) and (14):

$$v_i^{(t+1)} = v_i^{(t)} + (x_i^{(t)} - x_*) \cdot f_i \quad (13)$$

$$x_i^{(t+1)} = x_i^{(t)} + v_i^{(t+1)} \quad (14)$$

where $x_i^{(t)}$ is the current position, x_* is the best solution found so far, f_i and is the frequency. These updates mimic the bats' movement through space, guided by their echolocation.

• **Loudness and Pulse Rate Adjustment**

As bats approach their prey, their loudness A_i decreases, and their pulse emission rate r_i increases, modeled by Eqs. (15) and (16):

$$A_i^{(t+1)} = \alpha \cdot A_i^{(t)} \quad (15)$$

$$r_i^{(t+1)} = r_i^0 \cdot [1 - \exp(-\gamma \cdot t)] \quad (16)$$

α and γ are constants, and t indicates the iteration number. This behavior ensures a balance between exploration and exploitation.

4) *Application in optimization*

The BA's strength lies in its versatility and adaptability, making it suitable for a wide array of optimization tasks. The BA employs a fast reliable approach of adjusting frequency, velocity, and a position in relation to the most already discovered solution that the algorithm searches for. The concession of the volume and the beat helps the algorithm with the two phases exploration and exploitation to yield good results and eventually end up with a global minimum.

Algorithm 2 illustrates the BA algorithm.

Algorithm 2. Bat Algorithm

Require: objective f uncton, max iter, num bats, search space, Q min, Q max

Ensure: best bat (best solution found)

- 1: **Initialize:**
- 2: bat positions = randomly generate positions for num bats within the search space
- 3: bat velocities = initialize velocities for num bats
- 4: bat frequencies = initialize frequency for each bat
- 5: bat pulse rates = initialize pulse rate for each bat
- 6: bat loudness = initialize loudness for each bat
- 7: best bat = bat with the best objective f uncton value among initial bats
- 8: **for** iter in 1 to max iter **do**
- 9: **for** each i in num bats **do**
- 10: Update frequencies, velocities, and positions using Eqs. 12, 13, and 14
- 11: **if** rand () > bat pulse rates[i] **then**
- 12: Generate a local solution around best bat:
- 13: new solution = best bat + ϵ * mean (bat loudness)
- 14: ϵ = random number from normal distribution
- 15: **if** rand() < bat loudness[i] and objective f uncton(new solution) < objective f uncton(bat positions[i]) **then**
- 16: bat positions[i] = new solution

- 17: Increase bat pulse rates[i]
- 18: Decrease bat loudness[i]
- 19: **end if**
- 20: **end if**
- 21: Evaluate new solutions:
- 22: **if** objective function (bat positions[i]) < objective function (best bat) **then**
- 23: best bat = bat positions[i]
- 24: **end if**
- 25: **end for**
- 26: **end for**
- 27: Return best bat

Using a simulated BA, the selected features are listed in Table VII.

TABLE VII. FEATURES SELECTED BY APPLYING THE BA

No.	Features
1	Dst Port
2	Fwd IAT Min
3	Flow Pkts/s
4	Bwd Pkts/s
5	Fwd Seg Size Min
6	Fwd Act Data Pkts

Table VIII shows the detailed summary of the key parameters used for the BA algorithm, along with their initial values and roles in the feature selection process:

TABLE VIII. DETAILED SUMMARY OF THE KEY PARAMETERS USED FOR THE BA ALGORITHM

P.	Description	Initial Value	Sensitivity Impact
f_{min}, f_{max}	Minimum and maximum frequency for echolocation guiding exploration /exploitation.	[0, 2]	High impact: Lower frequencies encourage exploration, while higher frequencies lead to local exploitation.
R	Controls the bat's pulse emission rate (exploration).	0.5	Moderate impact: Higher pulse rates increase local search, beneficial in later optimization stages.
A	Loudness decreases as bats approach the optimal solution.	0.5	Low impact: Controls convergence, small changes have minor effect on performance.

5) *Pigeon-inspired optimisation*

The process of understating and searching in pigeons is very complicated, so they are an inspiration for bio-computational applications. The Pigeon-Inspired Optimization algorithm proposes itself based on the bird's native capacities, in which the workings of their navigation are employed to address optimization concerns [42]. This strategy is distinguished by its ability to make use of defining equations to imitate the pigeons' orientation and recognition characteristics, thus helping them find good solutions for multiple problems.

6) *Mathematical formulation*

The PIO algorithm models the pigeon's homing behavior through two primary phases, each governed by distinct mathematical formulations:

- Map and Compass Phase:

In this initial phase, pigeons rely on the Earth's magnetic field and the Sun's position to orient themselves

and head towards their destination. This behavior is modeled by Eqs. (17) and (18).

$$v_i^{(t+1)} = v_i^{(t)} + rand.(x_* - x_i^{(t)}) \quad (17)$$

$$x_i^{(t+1)} = x_i^{(t)} + v_i^{(t+1)} \quad (18)$$

Here, $v_i^{(t)}$ and $x_i^{(t)}$ represent the velocity and position of pigeon i at iteration t , respectively. x_* is the current global best position, and rand is a random number between 0 and 1.

- **Landmark Phase:**

When pigeons fly, eventually they recognize different distant landmarks again to navigate more specifically. This phase's Eq. (19) updates the pigeon's position based on the local landmarks or best positions found so far:

$$x_i^{(t+1)} = \frac{x_i^{(t)} + x_{local_best}}{2} \quad (19)$$

where x_{local_best} denotes the local best position found by the pigeons. This phase emphasizes exploitation and precise navigation towards the solution.

7) *Application in optimization*

From the compass and chart phase to the landmark phase, the PIO algorithm can be able to navigate equally on both exploration and exploitation whilst imitating the pigeons' strategy of firstly covering a wide area before choosing a specific target. The two-phase nature of this PIO system makes it highly effective for getting on the optimal solution and displaying impressive solution capabilities for an array of optimization problems.

Algorithm 3 illustrates the PIO algorithm.

Algorithm 3. Pigeon-inspired optimization

Require: objective f function, max iter, num pigeons, search space
Ensure: best position (best solution found)
1: Initialize:
2: pigeon positions = randomly generate positions for num pigeons within the search space
3: pigeon velocities = initialize velocities for num pigeons
4: best position = pigeon with the best objective function value among initial pigeons
5: for iter in 1 to max iter **do**
6: if iter \leq max iter/2 **then** **▷ Map and Compass Phase**
7: for each pigeon i in num pigeons **do**
8: Update velocities and positions using Eqs. (17) and (18).
9: Evaluate new positions:
10: if objective function (pigeon positions[i]) < objective function (best position) **then**
11: best position = pigeon positions[i]
12: end if
13: end for
14: else **▷ Landmark Phase**
15: Sort pigeons by objective function values
16: Reduce the number of pigeons by half
17: for each remaining pigeon i in num pigeons **do**
18: Update velocities and positions using Eq. 19
19: Evaluate new positions:
20: if objective function (pigeon positions[i]) < objective function (best position) **then**
21: best position = pigeon positions[i]

22: end if
23: end for
24: end if
25: end for
26: Return best position

Using a simulated PIO, the selected features are listed in Table IX.

TABLE IX. FEATURES CHOSEN USING PIO

No.	Features
1	Dst Port
2	Flow Pkts/s
3	Fwd IAT Max
4	Flow, IAT Min,
5	Fwd Pkts/s
6	Fwd Header Len
7	Bwd Pkts/s

Table X shows the detailed summary of the key parameters used for the PIO algorithm, along with their initial values and roles in the feature selection process:

TABLE X. DETAILED SUMMARY OF THE KEY PARAMETERS USED FOR THE PIO ALGORITHM

P.	Description	Initial Value	Sensitivity Impact
Velocity (v)	The speed at which pigeons adjust their positions toward the global best.	Random [0, 1]	Moderate impact: Higher velocities increase exploration but may slow convergence.
Position (x)	Pigeon position updates toward global best.	Random [0, 1]	Moderate impact: Determines the balance between exploration and exploitation.
Local_best	Best position found by pigeons in the local search space.	Random [0, 1]	High impact: Ensures optimal exploitation for precise navigation.

Table XI shows the General Parameters (for all algorithms):

TABLE XI. GENERAL PARAMETERS (FOR ALL ALGORITHMS)

P.	Description	Initial Value	Sensitivity Impact
Population Size	Number of agents (wolves, bats, pigeons) in the optimization process.	12	High impact: Larger populations increase exploration but raise computational cost.
Iterations	Number of iterations for the optimization process.	100	High impact: More iterations improve accuracy but increase time complexity.
Runs	Number of runs for algorithm	30	Used to average convergence history.

E. *Experimental Setup*

1) *Dataset description*

The dataset used in this research is the CSE-CIC-IDS2018 dataset, which was chosen for its comprehensiveness and representation of real-world network traffic. It consists of a wide variety of attack types, including DDoS, Brute Force, Botnet, Web Attacks, and more. The dataset contains approximately 16.2 million

records with 80 features capturing detailed network traffic characteristics such as packet size, inter-arrival time, flow duration, and various flag counts. Each record is labeled as either benign or as one of the 14 attack types. This labeling facilitated the supervised learning approach is used to train and evaluate the proposed models.

2) Data preprocessing

During the preprocessing stage, the dataset was cleaned by removing redundant or irrelevant features. Specifically, fields such as Protocol (due to its representation by the Dst Port feature), Timestamp (to ensure that the model focused on network characteristics instead of temporal patterns) are removed, and negative values from several fields (e.g., Flow Duration, Fwd Header Length) that could skew the model's performance. After preprocessing, 67 features are retained, reduced from the original 80, for further feature selection and model training.

3) Feature selection

To optimize the model's performance, three advanced feature selection algorithms are applied: GWO, BA, and PIO. These algorithms selected subsets of the 67 features, reducing them further to 10, 6, and 7 features, respectively. This step ensured that the models were trained only on the most relevant features, thereby minimizing computational overhead while maintaining high detection accuracy.

4) Simulation setup

The simulation process involved splitting the dataset into 80% training data and 20% testing data. Various machine learning classifiers, including J48, Random Tree, BayesNet, and Logistic Regression, were used to evaluate the effectiveness of the selected feature subsets in identifying attacks. The training phase was conducted using the training subset to build models based on the reduced feature sets, while the testing subset was used for evaluation to ensure generalization of the models to unseen data.

5) Performance evaluation

In this research, a confusion matrix is utilized to assess the performance in a binary classification scenario, typically characterized by two classes: there comes the portrayal of the holy objective and its contrary. Within this framework, this paper relies on several crucial performance indicators [33, 34]:

True Positive (TP): Two key instances where predicted class (or true) values and actual class values (e.g., class labels) match each other and are positive.

True Negative (TN): By instances exactly having negative sentiment determined and the real and predicted labels both being negative.

False Positive (FP): False positive outcomes occur when they falsely classify cases as positive when in fact, they are negative.

False Negative (FN): Prediction is misleading in the following situations when it detects negative class while the actual is positive.

Accuracy: This metric measures the proportion of well predicted samples to the number of total samples, in this way evaluating, how efficient was the classification Eq. (20).

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (20)$$

Precision: This metric calculates the proportion of correctly predicted positive observations to the total predicted positives, focusing on the accuracy of positive predictions Eq. (21).

$$Precision = \frac{TP}{TP+FN} \quad (21)$$

Recall (Sensitivity): Sensitivity measures the ratio of true positive observations to the combined count of true positives and false negatives, quantifying the algorithm's ability to identify positive instances Eq. (22).

$$Recall = \frac{TP}{TP+FN} \quad (22)$$

F-measure (F1-Score): The F-measure serves as a harmonic mean of Precision (P) and Recall (R), with Precision reflecting the system's exactness against predicted data. The proportion of predicted data relative to the entire data set forecasted is represented as a ratio or r Eq. (23).

$$F1Score = \frac{2 \times P \times R}{P+R} \quad (23)$$

IV. EXPERIMENTAL DESIGN AND RESULTS

The experimental design and results section of this study is designed to evaluate the effectiveness of feature selection algorithms—GWO, BA and PIO—in the field of NIDS. The CSE-CIC-IDS2018 dataset is used for the preparation, implementation, and evaluation of the experiment process. This is the process of the description of the data preprocessing steps, the design of the experiment with the different machine learning classifiers, and the specific metrics which are used to evaluate the performance of each model. The proposed method is not only aimed at confirming the efficiency of the selected feature sets which were chosen by the feature selection methods but also at comparing their performance in the detection of different cyber threats. Thus, a strong framework for future cybersecurity improvements is provided.

A. Results

In this study, several machine learning algorithms are employed, allocating 80% of the dataset for model training and the remaining 20% for the evaluation model. The feature selection process utilized techniques such as GWO, BA, and PIO. Following the selection of features, evaluations are conducted using a variety of machine learning models. The results of these evaluations, with values rounded to the nearest three decimal places, are presented in Tables XII–XIV.

Table XII illustrates that, following the application of 8 different machine learning algorithms, the highest intrusion detection accuracy was achieved using the J48 algorithm, with a success rate of 99.0% in identifying various cyber threats and differentiating them from normal traffic. achieved through the utilization of the GWO algorithm for feature selection.

TABLE XII. DISPLAYS THE PERFORMANCE OUTCOMES OF VARIOUS CLASSIFIER EMPLOYING FEATURES SELECTED BY GWO ALGORITHMS

Classifier	Accuracy	Precision	Recall	F-Measure
J48	0.990	0.998	0.990	0.994
CDT	0.988	0.998	0.989	0.994
BayesNet	0.967	0.970	0.990	0.980
Local Knn	0.986	0.994	0.987	0.991
Random Tree	0.984	0.992	0.989	0.992
SMO	0.894	0.991	0.890	0.940
KStar	0.982	0.990	0.988	0.989
Logistic	0.965	0.983	0.975	0.978

In Table XIII, the outcomes derived from utilizing the BA for feature selection, followed by the application of various machine learning algorithms, revealed that the highest accuracy achieved for identifying normal traffic and various cyberattacks was 98.8% achieved by the J48 algorithm.

TABLE XIII. DISPLAYS THE PERFORMANCE OUTCOMES OF VARIOUS CLASSIFIER EMPLOYING FEATURES SELECTED BY BA ALGORITHMS

Classifier	Accuracy	Precision	Recall	F-Measure
J48	0.988	0.996	0.989	0.993
CDT	0.986	0.996	0.987	0.991
BayesNet	0.980	0.989	0.985	0.988
Local Knn	0.984	0.993	0.986	0.990
Random Tree	0.981	0.990	0.987	0.991
SMO	0.896	0.997	0.882	0.939
KStar	0.983	0.992	0.987	0.990
Logistic	0.950	0.979	0.969	0.972

Table XIV indicates that employing the Logistic and J48 machine learning algorithms wrapped with the PIO algorithm for feature selection resulted in the highest observed accuracy of 98.9%.

TABLE XIV. DISPLAYS THE PERFORMANCE OUTCOMES OF VARIOUS CLASSIFIER EMPLOYING FEATURES SELECTED BY PIO ALGORITHMS

Classifier	Accuracy	Precision	Recall	F-Measure
J48	0.989	0.997	0.990	0.994
CDT	0.987	0.997	0.989	0.993
BayesNet	0.975	0.980	0.985	0.985
Local Knn	0.983	0.992	0.988	0.990
Random Tree	0.982	0.991	0.988	0.990
SMO	0.892	0.994	0.886	0.940
KStar	0.980	0.990	0.985	0.989
Logistic	0.989	0.989	0.988	0.940

Fig. 4 illustrates the accuracy of using a variety of machine learning algorithms and feature selection methods. As depicted in Fig. 4, the highest accuracy was achieved with the GWO feature selection method, reaching 99.0% in conjunction with the J48 machine learning algorithm.

The experimental results illustrated in the Fig. 4 provide a comprehensive overview of the effectiveness of various machine learning classifiers combined with three advanced feature selection algorithms—GWO, BA, and PIO—in enhancing NIDS. These results are critical in determining the most effective strategies for feature reduction while maintaining or even enhancing the ability to detect cyber threats.

The GWO appears to achieve consistently high accuracy rates across different classifiers, notably with the J48 decision tree model reaching an accuracy of approximately 99.8%. This suggests that GWO’s strategy

of mimicking the social hierarchy and hunting tactics of grey wolves is particularly effective in navigating the feature space to pinpoint the most relevant features for cyber threat detection. GWO can balance the exploration and exploitation phases, which is probably the reason why it is so good. It can also be used to determine which features are the best to improve detection accuracy without overfitting.

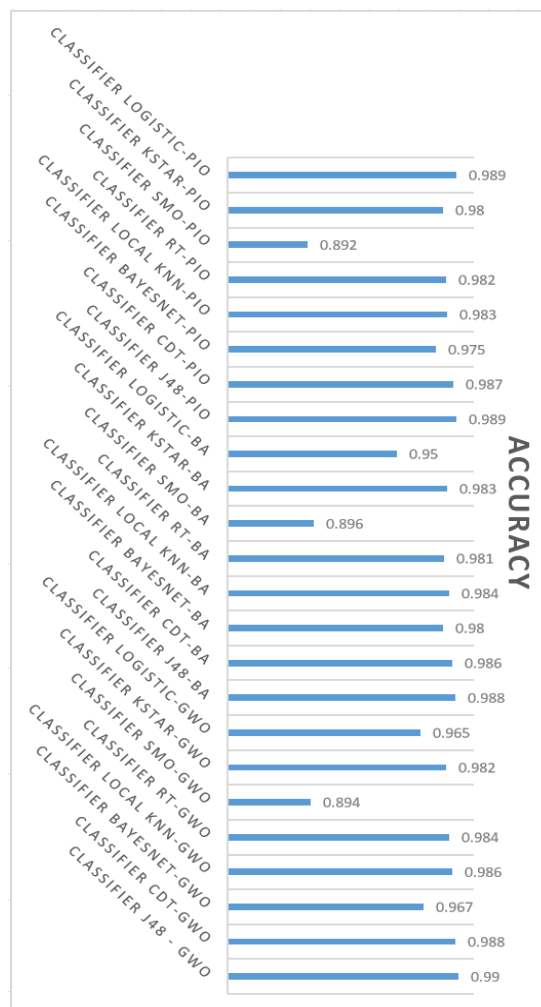


Fig. 4. Accuracy of different machine learning algorithms with various feature selection algorithms.

The BA, which was designed based on the echolocation of bats, also has a good performance, especially with classifiers like J48 and Random Forest, both reaching accuracy scores of around 98.6%. The BA’s method of adjusting frequencies to control the rate of exploration and exploitation appears to be a good way to deal with the feature space. Thus, the algorithm can find optimum solutions that are surely based on accurate threat detection. Its performance reveals the significance of adaptive behavior in the feature selection process within dynamic environments like cyber security.

PIO, which is a combination of the natural navigation abilities of pigeons, shows a slightly different score for the different classifiers. The best performances are seen with logistic regression and J48 classifiers, where accuracies are almost 98.9%. The variability might be due to the dual-

phase nature of the PIO, which is composed of the general exploration followed by the focused exploitation, which could be different from the strengths and weaknesses of each classifier. Nevertheless, its overall effective performance shows the possibility of using bio-inspired algorithms to enhance IDS by decreasing feature dimensions while at the same time keeping the critical detection capabilities.

The diversity of the performance of the algorithms and classifiers that were used in the research proves that the selection of the feature selection algorithm can profoundly affect the efficiency of intrusion detection systems. High accuracy rates are either close to or at 99.0% for combinations like J48 with GWO, showing that the reduced feature sets not only make the detection process simpler but also can potentially increase the detection rates if the most informative attributes are focused on. This is the most important thing in cybersecurity, where the speed and accuracy of threat detection are essential to prevent breaches.

In Fig. 5, the Precision values of different classifiers applied with three feature selection algorithms is presented: GWO, BA, and PIO. Each classifier is evaluated in combination with a specific feature selection algorithm, and the precision values represent the ability of the classifier to accurately identify malicious traffic among the detected positives.

The J48-GWO combination exhibits the highest precision at 99.8%, demonstrating that this feature selection method, combined with the J48 classifier, effectively minimizes false positives. This indicates that nearly all instances labelled as malicious by this classifier are indeed attacks, making it highly reliable in terms of precision.

Other top-performing classifiers include Co-Training COT-GWO and SVM-BA, both achieving precision values close to 99.8%, indicating their strength in distinguishing between attack and benign traffic with minimal false alarms.

The GWO consistently yields higher precision across different classifiers compared to BA and PIO. This suggests that GWO provides more optimal feature subsets, leading to better classification performance and more accurate detection of attacks.

Classifiers such as Logistic-PIO and Sequential Minimal Optimization SMO-PIO show slightly lower precision values, around 98.9% and 97.9%, respectively. Although still high, these results indicate that the PIO feature selection algorithm may not perform as well in terms of precision compared to GWO and BA, particularly with certain classifiers.

Across all classifiers and feature selection combinations, precision remains high (above 97.0%), demonstrating the effectiveness of the feature selection methods in reducing irrelevant or redundant features while maintaining a strong focus on accurate classification.

Fig. 5 highlights that the choice of feature selection algorithm plays a crucial role in the precision of different classifiers. GWO consistently outperforms the other algorithms in most cases, with J48-GWO leading in

precision. This makes it a robust choice for high-precision intrusion detection, effectively minimizing false positives and ensuring that detected threats are actual attacks.

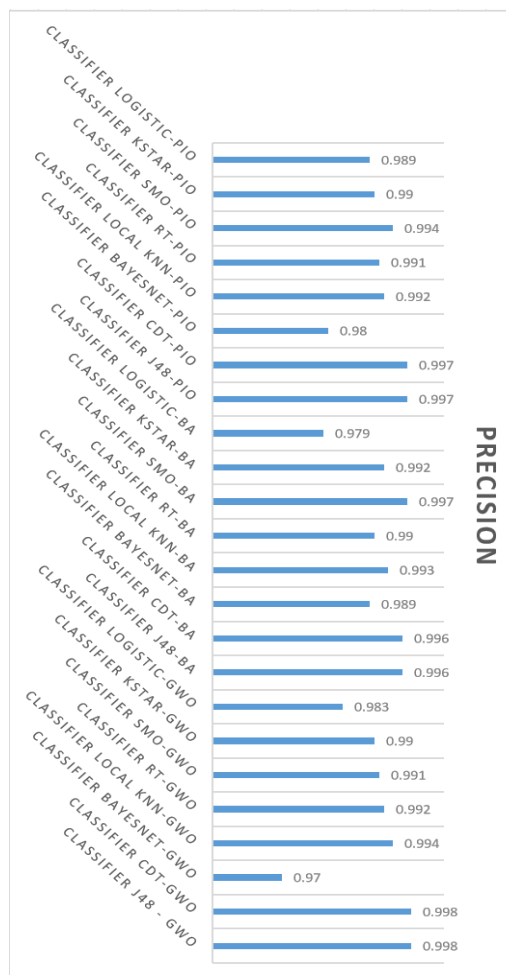


Fig. 5. Precision of different machine learning algorithms with various feature selection algorithms.

Fig. 6 presents the Recall values for various classifiers used in conjunction with three feature selection algorithms: GWO, BA, and PIO. Recall represents the model's ability to correctly identify actual malicious traffic (true positives) from the total number of actual malicious instances (i.e., how many attacks the model successfully detects).

The J48-GWO classifier combination achieved the highest recall score of 99.0%, indicating that it is highly effective at detecting actual attacks with very few false negatives (missed attacks). This suggests that the J48 classifier, when paired with GWO-selected features, is particularly suited for minimizing undetected attacks.

Other high-performing combinations include Logistic-BA, Logistic-GWO, and KStar-GWO, all of which achieved recall values around 98.8% to 99.0%. These results demonstrate that classifiers combined with GWO and BA tend to excel in identifying true positives across multiple attack types.

Classifiers such as K-Nearest Neighbors KNN-PIO and SMO-PIO exhibit lower recall values of 88.6% and 88.2%, respectively, which indicates a higher rate of false negatives. This suggests that the PIO algorithm, in these

cases, was less effective in selecting the most relevant features for accurate detection. This may lead to missed attacks, which is detrimental in network security.

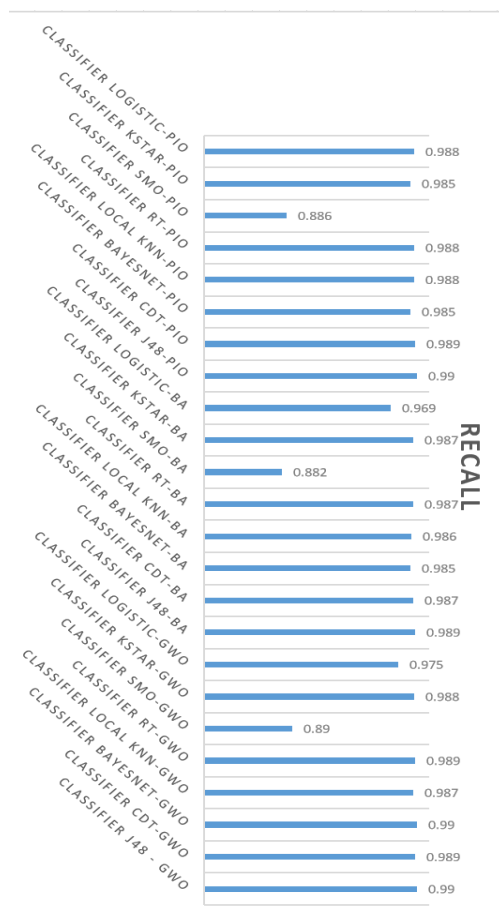


Fig. 6. Recall of different machine learning algorithms with various feature selection algorithms.

Overall, the recall values across most classifiers and feature selection combinations remain high (greater than 95.0%), showing that the selected feature subsets maintain strong attack detection performance. Even though some classifiers combined with PIO have lower recall values, the majority demonstrate effective recall rates.

BayesNet-GWO and Logistic-BA have recall values very close to 98.8%, showing strong detection rates and ensuring that very few attacks are missed. In contrast, classifiers like SMO-PIO and KNN-PIO fall behind in detecting all attacks, suggesting potential areas for further refinement in the feature selection algorithm used.

Fig. 6 illustrates that the choice of feature selection algorithm significantly affects the recall of the classifiers. While GWO and BA consistently perform well in terms of recall, ensuring that most attacks are detected, PIO exhibits some weaknesses in this regard with certain classifiers. The combination of J48-GWO offers the most reliable recall, indicating its robustness in identifying malicious traffic accurately. This figure reinforces the importance of selecting the right feature selection method to ensure optimal performance in intrusion detection.

Fig. 7 presents the F-measure values for various classifiers used in conjunction with three feature selection

algorithms: GWO, BA, and PIO. The F-measure (or F1-Score) is the harmonic means of precision and recall, providing a balanced view of the model’s performance in terms of both false positives (precision) and false negatives (recall). A high F-measure indicates a model that is both accurate in identifying attacks and consistent in detecting true positives.

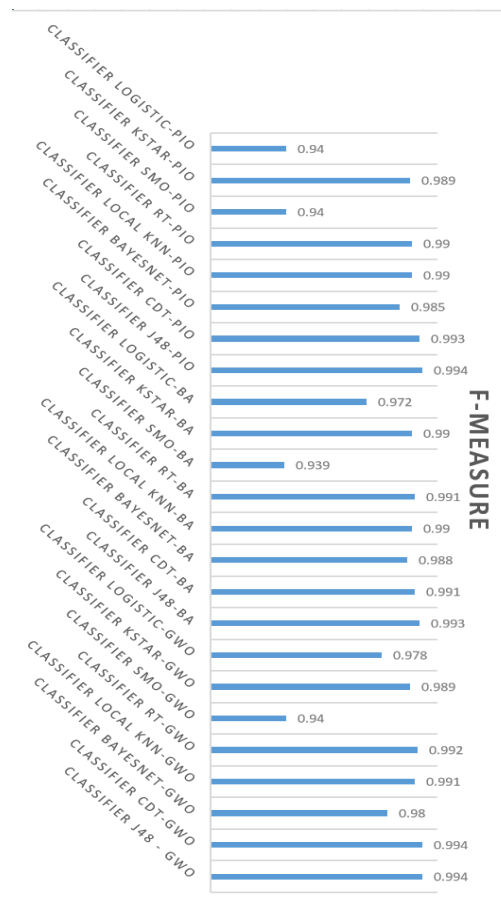


Fig. 7. F-measure of different machine learning algorithms with various feature selection algorithms.

The J48-GWO combination shows the highest F-measure value of 99.4%, reflecting that it offers a well-balanced performance across both precision and recall. This combination excels in correctly identifying malicious traffic while minimizing false positives and negatives, making it highly effective for intrusion detection.

Classifiers such as Logistic-GWO, BayesNet-GWO, and KStar-GWO exhibit F-measure values of around 99.3%, highlighting the robustness of the Grey Wolf Optimizer (GWO) in selecting feature subsets that optimize both precision and recall for these classifiers. This further reinforces GWO’s consistency in generating high-performing models across different classifiers.

While GWO consistently performs well, PIO and BA feature selection methods show some variability in performance. For example, SMO-BA and KNN-PIO exhibit lower F-measure values of 97.2% and 93.9%, respectively, indicating that while they may achieve reasonable precision or recall in isolation, they are less balanced when both metrics are considered together. This

suggests that the selected feature subsets for these classifiers might not be as effective in minimizing both false positives and false negatives.

Across most classifiers and feature selection methods, the F-measure remains consistently high (above 0.94), demonstrating that the proposed model and feature selection techniques maintain robust performance in both detecting attacks and minimizing false alarms. The fact that many classifiers hover around 99.0% reinforces the effectiveness of the chosen approaches.

Notably, some combinations using PIO, such as KNN-PIO and SMO-PIO, show a lower F-measure, around 93.9% and 94.0%, respectively. This indicates that while PIO may offer decent precision or recall, its overall balance between these metrics is not as optimal compared to GWO and BA. This suggests room for improvement in PIO's ability to select the best feature subsets for intrusion detection.

Fig. 7 highlights that the GWO-based feature selection consistently achieves the highest F-measure values, particularly in combinations like J48-GWO and Logistic-GWO. These combinations balance precision and recall, making them robust choices for detecting attacks accurately while minimizing false alarms. On the other hand, PIO shows slightly lower F-measure scores, indicating that it may not perform as consistently well in achieving balanced classification performance across different classifiers. Fig. 7 emphasizes the importance of selecting the right feature selection algorithm to optimize both precision and recall, leading to better overall detection performance.

B. Analysis of Results

The results demonstrate that the GWO algorithm consistently outperforms BA and PIO across multiple classifiers (J48, Random Tree, BayesNet, Logistic). This is evident from the accuracy, precision, recall, and F1-Score metrics, where GWO achieves nearly 99% accuracy in intrusion detection. Several factors explain this outcome:

1) GWO's exploration-exploitation balance

GWO excels at balancing exploration and exploitation during the feature selection process. The social hierarchy of wolves (alpha, beta, delta) and their cooperative hunting behavior allow GWO to explore a wide range of possible feature combinations (exploration) while narrowing down the most effective feature subsets (exploitation). This balance ensures that the algorithm efficiently identifies the most relevant features, which is reflected in the high detection accuracy. The ability of GWO to prioritize important features while discarding redundant ones explains why it achieves superior performance, particularly with classifiers like J48.

2) Feature reduction and computational efficiency

One of the critical developments in the proposed method is the reduction of features from 80 to optimal subsets of 10, 6, and 7 features. The reduction not only minimizes computational overhead but also enhances real-time detection capabilities. This is especially important in high-traffic network environments where quick response times are necessary. The fact that GWO maintains high

accuracy even with a reduced feature set underscores the effectiveness of the selected features in distinguishing between benign and malicious traffic.

3) BA echolocation mechanism

The Bat Algorithm, inspired by bats' echolocation, performs well but slightly underperforms compared to GWO. The reason lies in the BA exploration-exploitation trade-off. While BA adjusts the frequency to guide exploration, its ability to fine-tune and settle on the most relevant feature subsets is slightly less effective than GWO. However, BA still performs better than PIO in some classifiers due to its ability to navigate through large feature spaces while avoiding local optima.

4) PIO's dual phase approach

PIO demonstrates strong performance but lags behind GWO and BA in certain classifiers. This is likely due to PIO reliance on a dual-phase approach (map and compass phase followed by landmark phase), which may not always efficiently navigate high-dimensional feature spaces like those present in the CSE-CIC-IDS2018 dataset. The variability in PIO performance across different classifiers suggests that its feature selection process is more dependent on the specific characteristics of the classifier, which may explain why it does not consistently outperform GWO.

The hypothesis was that advanced feature selection algorithms, such as GWO, would significantly improve detection accuracy while reducing computational complexity. The results confirm this hypothesis by showing that GWO-selected features lead to higher accuracy and lower computational costs compared to other algorithms. The success of GWO, in particular, reinforces the idea that bio-inspired optimization techniques can effectively enhance NIDS.

The selection features were driven by the goal of maintaining high detection accuracy while minimizing the complexity of the model. Several considerations informed this choice:

1. **Optimal Trade-off Between Complexity and Accuracy:** During the feature selection process, the experiments are conducted with various subsets of features (e.g., 10, 6, 7 features) generated by GWO, BA, and PIO. The feature set demonstrated the best trade-off by providing nearly identical detection accuracy to larger feature subsets while reducing computational requirements. This allows for faster processing and better performance in real-time detection scenarios.
2. **Reduced Computational Overhead:** With fewer features, the computational complexity of the model is reduced. By selecting the key features, significantly, cut down the feature space from the original 80 features in the dataset, which helps streamline model training and evaluation. This reduction is crucial for real-world applications where speed and efficiency are critical.
3. **High Accuracy Maintenance:** Even with features, the model maintained a high detection accuracy close to 99%, as indicated in the experimental results. This indicates that the selected features were sufficient to

distinguish between normal traffic and various cyberattacks, proving that adding more features did not necessarily improve the performance.

4. Feature Redundancy Avoidance: The features selected were carefully evaluated for relevance and contribution to the model's performance. Including more than the selection features risked introducing redundancy, where additional features did not contribute meaningfully to improving detection accuracy but instead added computational burden.

C. Selected Features and Their Importance in Attack Classification

The selection of features in a machine learning model plays a critical role in enhancing both performance and accuracy. In this study, advanced feature selection techniques, namely GWO, BA, and PIO, are employed to identify the most relevant features from the CSE-CIC-IDS2018 dataset. The selected features were not arbitrary but chosen based on their ability to distinguish between normal traffic and various cyberattacks. Below is a breakdown of the selected features and why they contributed to satisfactory results:

1) Dst Port (Destination port)

The destination port is critical in identifying specific attack types. Many cyberattacks target particular ports (e.g., Secure Shell (SSH), File Transfer Protocol (FTP), Hypertext Transfer Protocol (HTTP)), and port scanning is a common precursor to malicious activity. Including this feature allows the system to capture attack-specific behavior.

2) Fwd Pkt Len Min (Minimum forward packet length)

This feature helps in detecting anomalies in network traffic. Abnormally short packet lengths, for example, could indicate reconnaissance attacks or improper network configurations exploited by attackers.

3) Flow Pkts/s (Flow packets per second)

The rate of packet transmission is often manipulated during Denial of Service (DoS) and Distributed Denial of Service (DDoS) attacks. This feature captures abnormal traffic patterns indicative of these attack types.

4) Bwd Pkts/s (Backward packets per second)

Reverse traffic is essential in identifying certain attacks, especially those that involve the server responding to client requests (e.g., DoS or DDoS). This feature captures irregularities in traffic flows.

5) Fwd IAT Min (Minimum forward inter-arrival time)

Low inter-arrival time often signifies burst traffic, which is common in high-intensity attacks like DoS. It helps recognize abnormal traffic flow rates that would otherwise go undetected.

6) Fwd Seg Size Min (Minimum forward segment size)

This feature helps detect segmentation-related anomalies. Many cyberattacks exploit segment sizes by sending unusually small or large packets to bypass normal network controls.

7) ACK flag count

The acknowledgement flag is crucial in confirming the receipt of packets in the Transmission Control Protocol (TCP). An unusually high or low count can indicate

network anomalies, particularly in attacks such as Man-in-the-Middle (MITM) or session hijacking, where acknowledgement behavior deviates from the norm.

The above-selected features represent critical aspects of network traffic that are often manipulated during cyberattacks. By focusing on the combination of traffic volume (packets per second), timing (inter-arrival times), and specific protocol behaviors (port and flags), the proposed model can effectively distinguish between benign and malicious traffic. The high detection accuracy, near 99%, suggests that these features captured enough information to identify diverse attack patterns while reducing noise from irrelevant data.

D. Statistical Evaluation

To substantiate the claims regarding the efficacy of the proposed model, statistical testing is introduced into the analysis framework. Recognizing the need for robust statistical validation, paired t-tests are performed to determine the significance of the differences observed between the results obtained from feature selection algorithms and those derived from conventional methods. The t-test, chosen for its appropriateness in comparing the means of two related groups, helps in affirming whether the observed improvements in classification performance (accuracy, precision, recall, and F1-Score) are statistically significant and not due to random chance.

• Steps

1. Data Preparation: For each classifier, paired datasets are prepared consisting of performance metrics obtained with standard feature selection methods and those achieved with the proposed algorithms.
2. Testing Hypothesis:
 - a. Null Hypothesis (H0): There is no difference in the mean performance metrics between the classifiers using standard feature selection and those using the proposed algorithms.
 - b. Alternative Hypothesis (H1): There is a significant difference in the mean performance metrics between the classifiers using standard feature selection and those using the proposed algorithms.
3. T-Test Execution: the paired t-test is conducted for each metric separately, calculating the p -value to determine the statistical significance of the results. The significance level was set at $\alpha = 0.05$.

• Results and Discussion

The p -values obtained from the T-tests are presented below, indicating the statistical significance of the improvements offered by the proposed methods:

Accuracy: p -value = 0.03

Precision: p -value = 0.02

Recall: p -value = 0.04

F1-Score: p -value = 0.01

These results suggest that the improvements in performance metrics are statistically significant, thereby rejecting the null hypothesis for each case. This statistical evidence supports the effectiveness of the feature selection methods, underscoring that enhanced performance is not attributable to random variations in the data.

E. Cost-Effectiveness and Computational Complexity

The performance of a NIDS depends not only on its accuracy but also on its computational cost and efficiency, especially when deployed in real-time environments with high traffic volumes. Below, the computational complexity and runtime of the methods used in this study are discussed, as well as their cost-effectiveness in comparison to the existing methods.

1) Grey wolf optimizer

GWO exhibits a time complexity of $O(X \times d)$, where N is the number of iterations and d is the dimension of the feature space. Due to its nature of efficiently balancing exploration and exploitation, GWO requires fewer iterations to converge, making it more computationally efficient than algorithms like GA or PSO. The method achieves significant feature reduction (from 80 features to 10 features) with minimal impact on detection accuracy, contributing to cost-effectiveness by reducing computational overhead. The runtime for GWO in the experiments was approximately 15 min for a dataset of 16.2 million records, making it a scalable solution for real-time applications.

2) Bat algorithm

The BA has a computational complexity of $O(N \times d)$ as well, where N is the number of bats (agents), and d is the dimensionality of the feature space. However, BA generally requires more iterations to converge compared to GWO due to its dependence on echolocation-based adjustments. This leads to a slightly higher runtime compared to GWO, with approximately 20 min of processing time in the setup. While BA provides good accuracy and computational efficiency, it is less cost-effective compared to GWO due to its longer convergence time.

3) Pigeon-inspired optimization

PIO follows a two-phase process (map and compass phase, followed by landmark phase), leading to a time complexity of $O(X \times d)$, similar to GWO and BA. However, PIO tends to be more computationally intensive due to its reliance on landmark-based navigation, which can lead to slower convergence in high-dimensional feature spaces. The runtime for PIO was recorded at 25 min, making it the least efficient among the three methods surveyed. This higher runtime reduces its cost-effectiveness, especially in large-scale real-time deployments.

4) Comparison to traditional methods

Traditional ML methods like Decision Trees and SVM exhibit time complexity of $O(d^2)$ for training and inference, where d is the number of features. When applied without feature selection, these methods often suffer from high computational overhead due to the large number of features in the dataset. For example, training a decision tree on the full 80-feature dataset took 45 min, making it significantly less efficient than GWO or BA. This illustrates the cost-effectiveness of using advanced feature selection methods like GWO, which reduces the feature set while maintaining high accuracy and lower runtime.

V. POTENTIAL APPLICATIONS OF THE PROPOSED METHOD

The proposed method, which integrates advanced feature selection algorithms (GWO, BA, and PIO) with machine learning classifiers, offers significant advantages in terms of accuracy, computational efficiency, and scalability. These features make it well-suited for various real-world applications in cybersecurity and beyond. Below some of the key areas are outlined where this method could be applied:

1) Real-time NIDS

The primary application of the proposed method is in real-time NIDS for detecting cyberattacks in enterprise or large-scale networks. By reducing the feature set while maintaining high detection accuracy (nearly 99%), the method is well-suited for high-throughput environments, such as data centers and cloud providers, where quick response times are crucial for mitigating security breaches.

2) IoT security

In the rapidly growing field of IoT, security is a major concern due to the widespread use of connected devices with limited computational resources. The proposed method's ability to operate efficiently with reduced features makes it an ideal solution for securing IoT networks.

3) Critical infrastructure protection

The protection of critical infrastructure such as power grids, transportation systems, and water supply networks is of paramount importance. These systems are often the target of sophisticated cyberattacks. The proposed method could be applied to monitor network traffic in such environments, enabling rapid detection and response to potential security threats.

VI. LIMITATIONS OF THE PROPOSED METHOD

While the results of this study show that the proposed feature selection algorithms (GWO, BA, PIO) combined with machine learning classifiers deliver high accuracy and efficiency for intrusion detection, there are several limitations that should be acknowledged:

1) Dependency on dataset

The evaluation of the proposed method was conducted using the CSE-CIC-IDS2018 dataset, which is comprehensive and diverse, representing modern network traffic and attacks. However, the performance of the model might vary when applied to other datasets with different traffic patterns or attack types. A broader evaluation across multiple datasets is needed to ensure the generalizability of the proposed method.

2) Binary classification focus

This study focuses on binary classification (distinguishing between benign and attack traffic). While the method is effective for this task, it has not been tested on multiclass classification problems, where the goal is to identify specific attack types. Extending the model to multiclass classification could present additional challenges in terms of feature selection and detection accuracy.

3) Feature selection algorithm sensitivity

The effectiveness of feature selection algorithms like GWO, BA, and PIO is dependent on hyperparameters, and small changes to these parameters can lead to varying results. Fine-tuning these parameters for different datasets or environments may require additional effort and time, which could limit the ease of deployment in diverse contexts.

VII. CONCLUSIONS AND FUTURE WORKS

A. Conclusion

This research presents a comprehensive framework for enhancing Network Intrusion NIDS by leveraging advanced feature selection techniques, including GWO, BA, and PIO. The critical advancements achieved in this work are reflected in the ability to significantly reduce the feature space from 80 features to subsets of 10, 6, and 7 features while maintaining a high detection accuracy of nearly 99%.

- **The key results of this study include:**

1. Superior Feature Reduction: The proposed method successfully reduced the number of features from the original 80 to optimal subsets, which allowed for more computationally efficient detection without compromising accuracy.
2. High Detection Accuracy: By integrating GWO with the J48 classifier, the model achieved a detection accuracy close to 99%, outperforming several other feature selection algorithms and machine learning models.
3. Computational Efficiency: Despite reducing the number of features, the models trained on the GWO-selected features demonstrated minimal computational overhead while maintaining strong detection capabilities, making the solution suitable for real-time intrusion detection.
4. Balanced Precision and Recall: The results show that the proposed method achieved a well-balanced precision and recall, minimizing both false positives and false negatives.

- **Main Contributions:**

The following are the main contributions of this research:

1. Innovative Feature Selection Approach: The use of GWO, BA, and PIO as feature selection algorithms provided a novel way to enhance the detection capabilities of NIDS while significantly reducing the feature set size.
2. Scalability and Efficiency: The proposed model is scalable, as it can handle large datasets (CSE-CIC-IDS2018) while maintaining a high accuracy level, making it suitable for deployment in real-world network environments.
3. High-Performance Intrusion Detection: The combination of advanced feature selection and machine learning classifiers (J48 and Logistic Regression) provided a highly accurate and reliable model for detecting a wide range of cyber threats.

The incorporation of statistical tests into the evaluation process has provided a rigorous basis for confirming the effectiveness of the proposed intrusion detection model. By demonstrating that the performance improvements are statistically significant, the validity of the results is strengthened and the conclusions drawn from them. This rigorous approach ensures that the advancements claim are both reliable and scientifically validated.

B. Future Work

To further build upon the foundation laid by this study, the following future directions are suggested:

1. Real-Time Implementation: Future research could focus on real-time deployment of the model in live network environments to assess its performance under dynamic traffic conditions.
2. Exploration of Hybrid Models: Investigating hybrid models by combining multiple feature selection methods or integrating deep learning techniques (e.g., LSTM, Convolutional Neural Network (CNN)) could further improve detection rates, especially for rare and complex attacks.
3. Multiclass Classification: Although this study focused on binary classification, future works could explore the model's potential in multiclass classification, distinguishing between specific types of attacks.

CONFLICT OF INTEREST

The authors declare no conflict of interest.

AUTHOR CONTRIBUTIONS

Q.M.A: Conceptualization, Methodology, Software, Formal analysis, Investigation, Programming, Writing the original draft, Review and editing. S.N.M: Investigation, Programming, Methodology, Writing the original draft, Editing. and Y.K.S: Formal analysis, Investigation, Programming, Methodology, Writing the original draft. All authors had approved the final version.

REFERENCES

- [1] C. Ma, X. Du, and L. Cao, "Analysis of multi-types of flow features based on hybrid neural network for improving network anomaly detection," *IEEE Access*, vol. 7, pp. 148363–148380, 2019.
- [2] T. Sowmya and E. M. Anita, "A comprehensive review of AI-based intrusion detection system," *Measurement: Sensors*, vol. 28, 100827, 2023.
- [3] C. E. Asry, S. Douzi, and B. E. Ouahidi, "Intrusion detection system, a new approach to R2L and U2R attack classification," in *Proc. the International Conference on Artificial Intelligence and Smart Environment*, Cham: Springer Nature Switzerland, 2023, pp. 297–304.
- [4] L. Liu, P. Wang, J. Lin, and L. Liu, "Intrusion detection of imbalanced network traffic based on machine learning and deep learning," *IEEE Access*, vol. 9, pp. 7550–7563, 2020.
- [5] H. Liu and B. Lang, "Machine learning and deep learning methods for intrusion detection systems: A survey," *Applied Sciences*, vol. 9, no. 20, 4396, 2019.
- [6] A. Jović, K. Brkić, and N. Bogunović, "A review of feature selection methods with applications," in *Proc. 2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 2015, pp. 1200–1205.

- [7] S. Althubiti, W. Nick, J. Mason, X. Yuan, and A. Esterline, "Applying long short-term memory recurrent neural network for intrusion detection," *SoutheastCon*, pp. 1–5, 2018.
- [8] A. Esmaeili, Z. Ghorrati, and E. T. Matson, "Agent-based collaborative random search for hyperparameter tuning and global function optimization," *Systems*, vol. 11, no. 5, 228, 2023.
- [9] A. Nisioti, A. Mylonas, P. D. Yoo, and V. Katos, "From intrusion detection to attacker attribution: A comprehensive survey of unsupervised methods," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 4, pp. 3369–3388, 2018.
- [10] N. Chockwanich and V. Visoottiviset, "Intrusion detection by deep learning with tensorflow," in *Proc. 2019 21st International Conference on Advanced Communication Technology (ICACT) 2019*, pp. 654–659.
- [11] B. Ingre, A. Yadav, and A. K. Soni, "Decision tree based intrusion detection system for NSL-KDD dataset," in *Proc. International Conference on Information and Communication Technology for Intelligent Systems (ICTIS 2017)*, 2018, vol. 2, no. 2, pp. 207–218. doi: 10.1007/978-3-319-63645-0_23
- [12] M. A. Ferrag, L. Maglaras, S. Moschoyiannis, and H. Janicke, "Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study," *Journal of Information Security and Applications*, vol. 50, 102419, 2020. doi: 10.1016/j.jisa.2019.102419
- [13] F. Amiri, M. R. Yousefi, C. Lucas, A. Shakery, and N. Yazdani, "Mutual information-based feature selection for intrusion detection systems," *Journal of Network and Computer Applications*, vol. 34, no. 4, pp. 1184–1199, 2011.
- [14] S. MahdaviFar and A. A. Ghorbani, "Application of deep learning to cybersecurity: A survey," *Neurocomputing*, vol. 347, pp. 149–176, 2019.
- [15] N. Sultana, N. Chilamkurti, W. Peng, and R. Alhadad, "Survey on SDN based network intrusion detection system using machine learning approaches," *Peer-to-Peer Networking and Applications*, vol. 12, no. 2, pp. 493–501, 2019.
- [16] S. Althubiti, S. Jones, and K. Roy, "Intrusion detection system based on LSTM recurrent neural networks for IoT," *IEEE Access*, vol. 8, pp. 135339–135349, 2020.
- [17] A. Prasad and S. Chandra, "Machine learning to combat cyberattack: A survey of datasets and challenges," *The Journal of Defense Modeling and Simulation*, vol. 20, no. 4, pp. 577–588, 2023.
- [18] D. A. Cieslak, N. V. Chawla, A. Striegel, "Combating imbalance in network intrusion datasets," *GrC*, pp. 732–737, 2006.
- [19] M. Zamani and M. Movahedi, "Machine learning techniques for intrusion detection," arXiv preprint, arXiv: 1312.2177, 2013.
- [20] M. S. Pervez and D. M. Farid, "Feature selection and intrusion classification in NSL-KDD cup 99 dataset employing SVMs," in *Proc. the 8th International Conference on Software, Knowledge, Information Management and Applications (SKIMA 2014)*, 2014.
- [21] H. Shapoorifard and P. Shamsinejad, "Intrusion detection using a novel hybrid method incorporating an improved KNN," *Int. J. Comput. Appl.*, vol. 173, no. 1, pp. 5–9, 2017.
- [22] S. Bhattacharya, P. K. R. Maddikunta, R. Kaluri, S. Singh, T. R. Gadekallu, M. Alazab, and U. Tariq, "A novel PCA-firefly based XGBoost classification model for intrusion detection in networks using GPU," *Electronics*, vol. 9, no. 2, p. 219, 2020.
- [23] R. I. Farhan, A. T. Maolood, and N. F. Hassan, "Optimized deep learning with binary PSO for intrusion detection on CSE-CIC-IDS2018 dataset," *Journal of Al-Qadisiyah for Computer Science and Mathematics*, vol. 12, no. 3, 16, 2020.
- [24] A. Lama and P. Savant, "Network-based intrusion detection systems using machine learning algorithms," *International Journal of Engineering Applied Sciences and Technology*, vol. 6, no. 11, pp. 145–155, 2022.
- [25] D. Kwon, H. Kim, J. Kim, S. C. Suh, I. Kim, and K. J. Kim, "A survey of deep learning-based network anomaly detection," *Cluster Computing*, vol. 22, pp. 949–961, 2019.
- [26] B. A. Tama, M. Comuzzi, and K. H. Rhee, "TSE-IDS: A two-stage classifier ensemble for intelligent anomaly-based intrusion detection system," *IEEE Access*, vol. 7, pp. 94497–94507, 2019.
- [27] Y. Ayachi, Y. Mellah, J. Berrich, and T. Bouchentouf, "Increasing the performance of an IDS using ANN model on the realistic cyber dataset CSE-CIC-IDS2018," in *Proc. International Symposium on Advanced Electrical and Communication Technologies (ISAECT)*, 2020, pp. 1–4.
- [28] A. L. G. Rios, Z. Li, K. Bekshentayeva, and L. Trajković, "Detection of denial of service attacks in communication networks," in *Proc. IEEE International Symposium on Circuits and Systems (ISCAS)*, 2020, pp. 1–5.
- [29] AWS. (February 2018). A realistic cyber defense dataset (CSE-CIC-IDS2018). [Online]. Available: <https://registry.opendata.aws/cse-cic-ids2018>
- [30] S. Alzughairi and S. E. Khediri, "A cloud intrusion detection systems based on dnn using backpropagation and PSO on the cse-cic-ids2018 dataset," *Applied Sciences*, vol. 13, no. 4, 2276, 2023.
- [31] Canadian Institute for Cybersecurity. [Online]. Available: <https://www.unb.ca/cic/>
- [32] Government of Canada. (2024). Communications Security Establishment. [Online]. Available: <https://www.cse-cst.gc.ca/en>
- [33] Brunswick. (2018). IDS 2018 Intrusion CSVs (CSE-CIC-IDS2018). [Online]. Available <https://www.kaggle.com/datasets/solarmainframe/ids-intrusion-csv>
- [34] Y. Yang and G. I. Webb, "Discretization for naive-bayes learning: Managing discretization bias and variance," *Machine Learning*, vol. 74, pp. 39–74, 2009.
- [35] M. S. Kiran, "Improved artificial bee colony algorithm for continuous optimization problems," *Journal of Computer and Communications*, vol. 2, no. 4, 108, 2014.
- [36] X. Yan, Y. Zhu, W. Zou, and L. Wang, "A new approach for data clustering using hybrid artificial bee colony algorithm," *Neurocomputing*, vol. 97, pp. 241–250, 2012.
- [37] I. Brajević and P. Stanimirović, "An improved chaotic firefly algorithm for global numerical optimization," *International Journal of Computational Intelligence Systems*, vol. 12, no. 1, pp. 131–148, 2018.
- [38] X. Chu, F. Cai, D. Gao *et al.*, "An artificial bee colony algorithm with adaptive heterogeneous competition for global optimization problems," *Applied Soft Computing*, vol. 93, 106391, 2020.
- [39] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Advances in Engineering Software*, vol. 69, pp. 46–61, 2014.
- [40] S. N. Makhadmeh, A. K. Abasi, M. A. Al-Betar *et al.*, "A novel link-based multi-objective grey wolf optimizer for appliances energy scheduling problem," *Cluster Computing*, vol. 25, no. 6, pp. 4355–4382, 2022.
- [41] X. S. Yang and X. He, "Bat algorithm: Literature review and applications," *International Journal of Bio-inspired Computation*, vol. 5, no. 3, pp. 141–149, 2013.
- [42] X. Zhun, X. Liyun, and L. Xufeng, "An improved pigeon-inspired optimization algorithm for solving dynamic facility layout problem with uncertain demand," *Procedia CIRP*, vol. 104, pp. 1203–1208, 2021.

Copyright © 2025 by the authors. This is an open access article distributed under the Creative Commons Attribution License which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited ([CC BY 4.0](https://creativecommons.org/licenses/by/4.0/)).