A Hybrid Binary and Multi-Class Classification Model for Network Intrusion Detection

Karrar Mohsin Alwan[1]*, Ahmed Saad Mohammed[2], A. S. Abohamama [3], Altameemi Ali Najm Abdullah [4]and Walaa Khalil Abrahem[5]

[1]Department of Electromechanical Systems Technologies, Baquba Technical College, Middle Technical University, Iraq
[2]Department of Computer, College of Basic Education, Mustansiriya University, Iraq
[3]Department of Computer Science, Faculty of Computers and Information, Mansoura University, Mansoura, Egypt
[3]Department of Computer Science, Arab East Colleges, Riyadh, Saudi Arabia.
[4]Department of Accounting Techniques, Baquba Technical College, Middle Technical University, Iraq
[5]Department of Computer Science, College of Science, University of Diyala, Iraq

Article Info

Article history:

Received: 11,10,2025
Revised: 12 ,08, 2025
Accepted: 16, 11, 2025
Published: 30, 01, 2026

Keywords:

Intrusion Detection,
Feature Selector,
NSL-KDD dataset,
Firefly Algorithm,
Support Vector Machine,
K-Nearest Neighbour.

ABSTRACT

Intrusion detection is a cornerstone in computer networks, maintaining privacy and ensuring availability and security. However, the larger the number of features involved in the intrusion detection process, the more complex it becomes. Therefore, reducing the number of features is necessary. Feature selection techniques can effectively enhance the classifiers' performance by eliminating redundant or irrelevant features. Two powerful models were introduced for anomaly-based intrusion detection based on a binary classifier and a multi-classifier, which both depend on a modified firefly algorithm (FFA) for feature selection. Support Vector Machine (SVM) and K-Nearest Neighbour classifiers have been used to evaluate both models over the NSL-KDD dataset. The first and second models have been used for attack classification to distinguish between normal and abnormal traffic, and between four types of attacks, including Denial of Service Attack (DoS), User to Root Attack (U2R), Remote to Local Attack (R2L), Probing Attack, and the normal case, respectively. The models were evaluated for classification accuracy and the number of features. The first model achieved 98% accuracy with 7 selected features, while the second achieved 97% accuracy with 11 selected features.

Corresponding Author:

Karrar Mohsin Alwan
Department of Electromechanical Systems Technologies, Baquba Technical College, Middle Technical University, Iraq.

Email: karrar.mohsin@mtu.edu.iq

1. INTRODUCTION

Computer and network security, along with cybersecurity, is an urgent and pivotal trouble that needs attention. Every day, the systems and networks of organizations across various sectors, including business, medicine, technology, engineering, and education, are subjected to numerous state-of-the-art cyber-attacks. We have to take this hazard seriously and put into effect sturdy safety features to defend our crucial statistics and infrastructure [1]. As a result, the frequency of assaults continues to rise, inflicting economic losses, denial of offerings, and numerous terrible impacts for countries and organizations[2]. Intrusion detection techniques are a defense against pc attacks threatening community security. These measures come into play after establishing a robust network structure, enforcing firewalls, and carrying out thorough personnel screenings. Despite the availability of superior intrusion prevention methods, the truth remains that successful attacks on PC systems continue to occur with alarming frequency. This ongoing assignment underscores the essential role of intrusion detection systems (IDSs) in enhancing and reinforcing typical network security, acting as vigilant sentinels that reveal suspicious activities and potential breaches [3]. The primary cause of an Intrusion Detection System (IDS) is to discover intrusions within normal audit records, which may be considered a type of hassle.

One extensive undertaking with IDSs is the potential for excessive overhead, which can become prohibitively expensive. IDSs are normally labeled into three principal kinds: Host-based Intrusion Detection Systems (HIDS), Network or Distributed Intrusion Detection Systems (NIDS or DIDS), and Hybrid Intrusion Detection Systems (HYIDS). This taxonomy is prepared via the operational method of every IDS. HIDS attempts to ensure the safety of a standalone computer node on which it operates. On the alternative hand, DIDS analyses visitors throughout a distributed pc community, monitoring for suspicious sports at the network degree. HYIDS combines the capabilities of each DIDS and HIDS, integrating the blessings of each strategy [4].

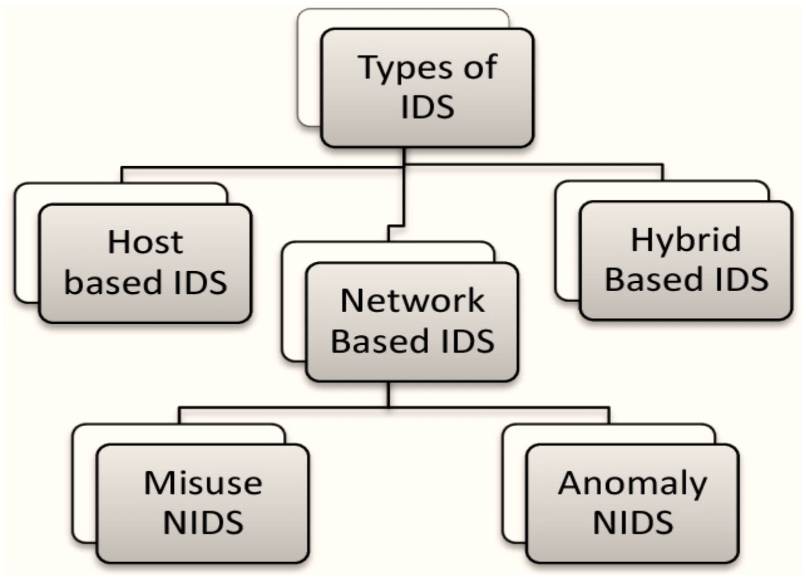


Figure 1. Classification of IDS

As shown in Figure 1, all sorts of Intrusion Detection Systems (IDS) can be categorized into three predominant classes: misuse-based, anomaly-based, and hybrid-based. A signature-based IDS detects intrusions through the use of a database of previously recorded attacks. On the alternative side, an anomaly-based IDS specializes in the conduct of the community. It continuously monitors sports deviating from normal behaviour and marks it as suspicious sports. Eventually, a hybrid-based IDS attempts to benefit from the advantages of both anomaly-based and signature-based systems. [5]. Due to the diverse protocols and services involved, community packets comprise numerous features. Some of these capabilities are redundant or beside the point. It has been determined that redundant functions contribute appreciably to a multiplied False Alarm Rate (FAR) and a decreased detection rate.

Feature Selection (FS) is an important approach for figuring out and retaining the simplest and most relevant capabilities in a data set. This technique significantly enhances the reliability of a Network Intrusion Detection System (NIDS) by decisively getting rid of noisy and redundant features. Additionally, it successfully reduces the computational time required for enforcing NIDS. [6]. This paper provides two progressive models designed for class obligations. The first adopts a binary classifier, while the second adopts a multi-classifier. To optimize the Intrusion Detection Systems (IDS), we utilize a hybrid metaheuristic method known as the Firefly Algorithm (FA), which is adept at appearing characteristic discount. This process is critical in identifying and selecting the maximum applicable functions, in the end leading to greater correct detection skills.

In addition to the standard FA, a mutation operator is employed to increase the standard algorithm's efficiency and effectiveness. This modification allows for greater solution space exploration, facilitating better optimization results. To validate the efficacy of our proposed method, we conduct extensive experiments using two prominent classifiers: the Support Vector Machine (SVM) and K-Nearest Neighbour (KNN). We apply these classifiers to the “NSL-KDD” dataset, a well-known benchmark in the field, and evaluate our models against a range of suitable performance metrics. Through this comprehensive approach, we aim to demonstrate significant improvements in classification accuracy and reliability for IDS applications.

2. RELATED WORK

Many models have been created to improve intrusion detection, especially to address the shortcomings of anomaly detection. This section looks at traditional methods of intrusion detection. Because network traffic has many dimensions, many intrusion detection models use feature selection to prepare data. Sharma et al.[7] presented a hybrid method to classify attacks and identify intrusions. The NSLKDD dataset was categorized into two classes using FGSVM: regular and attack classes. Notable findings from FGSVM show that 99.03% of samples can accurately detect DDoS, probe, U2R, and R2L attacks. Aberrant patterns identified by FGSVM were then activated using ANFIS. This method classified the data and enhanced the accuracy and efficiency of the systems; however, it was limited to a specific type of deep learning-based classifier. Advanced AI and machine learning technologies are enabling more precise intrusion detection. Amiri et al.[8] proposed a feature selection algorithm that effectively utilizes the mutual information method to assess feature relationships.

The resulting optimal feature set was subsequently employed to train the LS-SVM classifier, contributing to the development of the Intrusion Detection System (IDS). Horng et al.[9] suggested an SVM-based IDS, which employs both hierarchical clustering and an SVM-based classifier. The hierarchical clustering algorithm attempts to obtain a reduced set of high-quality training data, which helps in reducing both training and testing time, in addition to increasing the classification accuracy. In the experiments conducted on the corrected labels of the KDD Cup 99 dataset, which has several new attack types, the proposed system achieved 95.75% accuracy and 0.7% false positive rate. Khammassi et al.[10] used a Genetic Algorithm (GA) with Logistic Regression (LR) for feature selection on the UNSW-N15 and KDDCup99 datasets. He worked with the Weka simulation tool. After running multiple tests, the obtained results revealed that the GA-LR combined with a Decision Tree-based classifier has an 81.42% detection rate and a 6.39% false alarm rate based on 20 features rather than 42 features, which are contained in the UNSW-NB15 dataset. For the KDDCup99 dataset, the GA-LR with the DT classifier reached a detection rate of 99.90% and a false alarm rate of 0.105% while using 18 features. Osanaiye et al.[11] presented a method for detecting Distributed Denial of Service (DDoS) attacks using several filters. The filters included Information Gain, Chi-Square, Gain Ratio, and ReliefF. The researchers used the NSL-KDD attack detection dataset to show how well this system works. For classification, they used the Decision Tree (DT) algorithm and trained it with a method called k-fold cross-validation (with (k = 10)). The results showed that the DT classifier achieved a detection accuracy of 99.67% by using only 13 out of the 42 features available. The false alarm rate (FAR) was 0.42%. However, the research did not thoroughly explore the multiclass classification problem in the NSL-KDD dataset.

Ingre et al.[12] developed an Intrusion Detection System (IDS) using a filter-based methodology to reduce the number of input attributes (features) necessary for training and testing the model. The Decision Tree (DT) classifier was employed alongside a correlation input selection technique. The dataset used within the experiments became the NSL-KDD. After making use of the filter out to the function area, 14 features were selected. Additionally, the author considered each multiclass and binary category configuration, encompassing all five training attacks inside the NSL-KDD dataset. Based on the experimental results, the system was able to obtain an accuracy of 90.30% in the binary classification approach and 83.66% in the multiclass approach. Sung et al.[13] removed one feature at a time to experiment with Support Vector Machines (SVM) and neural networks. The KDD Cup 1999 dataset was applied to assess this technique.

In the category of 5-magnitude classification, it was noted that employing only 19 of the most significant features, rather than the complete set of 41, did not produce a statistically significant change in the performance of intrusion detection. Selvakumar et al.[14] implemented a method that combines filtering and wrapping techniques using a Firefly algorithm for feature selection. The functions that were decided on have been evaluated using C4.5 and Bayesian Network (BN)-based classifiers on the KDD CUP 99 dataset. The experimental results verified that using just 10 features was sufficient for effective intrusion detection, resulting in improved accuracy. Ghanem et al.[15] adeptly utilized a feature selection technique grounded in a multi-objective BAT algorithm (MOBBAT) during the initial phase. In the subsequent phase, they employed the selected features to categorize network traffic utilizing an upgraded BAT algorithm (EBAT) specifically formulated for training a multilayer perceptron (EBATMLP). This method markedly improves the efficacy of the Intrusion Detection System (IDS) and is designated as MOB-EBATMLP. The effectiveness of this approach has been evaluated using typical datasets for testing IDS, such as NLS-KDD, ISCX2012, UNSW-NB15, KDD CUP 1999, and CICIDS2017. This method has resulted in a decrease from 41 to 12 features, with a high accuracy of 99%.

To address the issue of data imbalance, Bakro et al.[16] implemented a more robust cloud intrusion detection system (IDS) that makes use of the Synthetic Minority Over-sampling Technique (SMOTE). With the use of three different methods—Information Gain (IG), Chi-Square (CS), and Particle Swarm Optimization—a hybrid approach to feature selection has been proposed. The attack type categorization was achieved by using the Random Forest (RF) model. Using this strategy reduced the number of features from 41 to 21, while maintaining a high degree of accuracy of 98%. The proposed system has been validated using the UNSW-NB15 and Kyoto datasets. Faizin et al.[17] suggested an Intrusion Detection System (IDS) using mutual information, threshold-based feature selection, and XGBoost. The relationship between input and target features is assessed using mutual information.

After mutual information determines the amount of information, thresholding determines the ideal number of features for classification. Finally, XGBoost features are used to classify the data. Two key variables were compared: the number of characteristics selected for classification and classification accuracy. The optimum feature selection approach and thresholding value combinations were examined using UNSW-NB15 as the primary dataset. The proposed method was further tested utilizing NSL-KDD and CIC-IDS2017 datasets to compare performance to earlier studies. This approach utilized all 41 features, resulting in a low accuracy level of 80%.

3. MATERIAL AND METHODS

Before, a must add Material and methods in this paper, conclude different algorithms such as:

3.1 Firefly Algorithm

The Firefly algorithm (FFA) is an innovative optimization technique created by Yang, inspired by the mesmerizing behaviours of natural fireflies. This algorithm harnesses the enchanting patterns of these glowing insects to solve complex problems, making it a fascinating blend of nature and technology[18]. The Firefly Algorithm (FFA) is a biologically inspired global optimization method. It is a population-based metaheuristic, where each firefly in the population represents a potential solution within the search space. The algorithm simulates the behaviour of fireflies as they communicate through flashing lights during mating rituals. These flashes attract potential prey and act as a warning mechanism to others. Yang [18] Formulated the Firefly Algorithm (FA) based on three principles that describe the behaviour of fireflies:

a. All fireflies are unisex, which means that all fireflies will be attracted to one another.
b. Attractiveness is relative to brightness; thus, the less bright one will be attracted to the brighter one among any two fireflies. However, this attractiveness decreases as the distance between the two fireflies increases.
c. The brightness of a firefly is linked to its fitness function. If no firefly is brighter than the current one, it will randomly attract other fireflies.

```
BEGIN Firefly Algorithm
Initialize population of fireflies (solutions)
Define the objective function
Define parameters (light intensity, attractiveness, randomization)
WHILE termination condition is not met DO
  FOR each firefly i DO
    FOR each firefly j DO
      IF i! = j THEN
        IF (fitness of i < fitness of j) THEN
          Move firefly i towards firefly j
          Update light intensity
        ENDIF
      ENDIF
    ENDFOR
    Evaluate the fitness of firefly i
  ENDFOR
ENDWHILE
Return the best solution found
END Firefly Algorithm
```

Figure 2. Firefly Algorithm (FFA)

3.2 Support Vector Machine (SVM)

SVMs were initially proposed by Vapnik (1995) for solving problems of classification and regression analysis [19]. SVM is a supervised learning technique that is trained to classify different categories of data from various disciplines. These have been used for two-class classification problems and are applicable to both linear and non-linear data classification tasks. SVM creates a hyperplane or multiple hyperplanes in a high-dimensional space, and the best hyperplane among them is the one that optimally divides data into different classes with the largest separation between the classes. A non-linear classifier uses various kernel functions to estimate the margins. The main objective of these kernel functions (i.e., linear, polynomial, radial basis, and sigmoid) is to maximize margins between hyperplanes. Recently, many highly promising applications have been developed by researchers because of the increasing interest in SVMs [20]. SVM has been widely used in image processing and pattern recognition applications. One of the primary advantages of making use of Support Vector Machines (SVM) in Intrusion Detection Systems (IDS) is their remarkable speed and efficiency. Within the field of cybersecurity, the potential to apprehend and respond to potential intrusions in real-time is critical to safeguarding sensitive data and keeping system integrity.

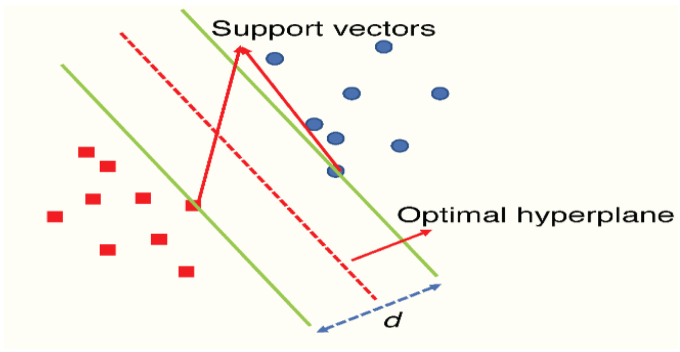


Figure 3. Support Vector Machine

SVMs are particularly adept at processing and gaining knowledge from widespread datasets, letting them identify a broader variety of patterns related to everyday and malicious behaviour. In addition to these blessings, SVMs own a completely unique functionality to evolve dynamically. Whenever they come upon new patterns for the duration of the type manner, they could include those experiences in their education framework. This flexibility permits SVMs to adapt continuously, enhancing their effectiveness in identifying novel intrusions, especially vital in an ever-changing threat panorama. Overall, those attributes make Support Vector Machines an effective device within the combat against cyber threats, ensuring timely detection and a higher defense mechanism for systems under surveillance [21]. Figure (3) shows the process of SVM.

3.3 K-Nearest Neighbor (KNN)

The k-nearest neighbour (k-NN) is a simple and effective technique for object classification according to the closest training examples in the feature space[22]. Consider a set of observations and targets (x_1, y_1), . . ., (x_n, y_n), where observations $x_i \in R^d$ and targets $y_i \in \{0, 1\}$; then for a given i, k-NN rates the neighbors of a test sequence among the training sample, and uses the class labels of the nearest neighbors to predict the test vector class. So, K-NN takes the new points and classifies them according to the majority of the votes obtained for the K nearest points in the training data.

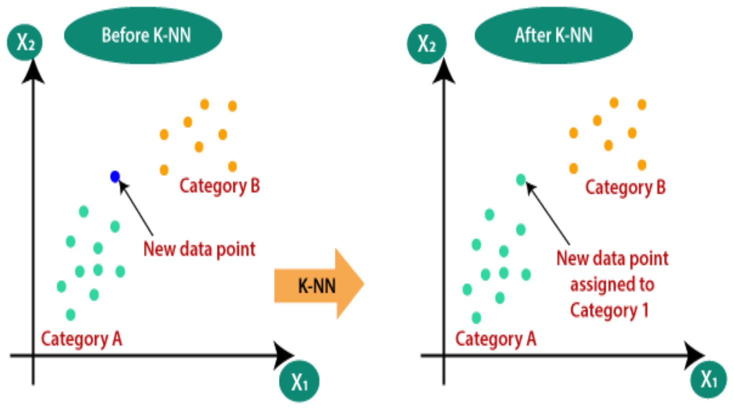


Figure 4. K-Nearest Neighbor (KNN)

In k-NN, the Euclidean distance is often used as the distance metric to measure the similarity between two vectors (points):

$$d^2\left(x_{i,x_j}\right) = \left\|x_i - x_j\right\|^2 = \sum_{k=1}^{d}(x_{ik}, y_{jk})^2 \qquad (1)$$

where $(x_i x_j) \in R^d$, $x_i = (x_{i1}, x_{i2} \dots \dots, x_{id})$

The k parameter of k-NN classifiers represents the number of neighbors in a set of training observations that are nearest to the given observation in the validation or testing data set. Variation of this parameter will affect the accuracy of each binary classifier inside an expert.The KNN (K-Nearest Neighbors) algorithm classifies objects based on their proximity to training examples within the characteristic area. The handiest form of KNN is known as the Nearest Neighbor rule (NN), which occurs when K is set to ten. In this technique, each sample is assessed based on its nearby samples. If a pattern's category is unknown, it is able to be expected through looking at the classifications of its nearest neighbor samples[23]. Figure (4) shows the process of KNN

3.4 NSL-KDD Dataset

The NSL-KDD (Network Security Laboratory Knowledge Discovery and Data Mining) dataset is an outstanding upgrade to the unique KDDCup'99 dataset, offering superior features and stepped forward performance for researchers inside the subject of community intrusion detection [24]. The dataset used to evaluate the performance of the proposed model encompasses a variety of network intrusions, categorized into four primary types:

- Denial of Service (DoS): Attackers overwhelm a network resource, rendering it unavailable.
- Probe: Attempts to gather information about the target system.
- User to Root (U2R): An unauthorised user gains elevated access rights.
- Remote to Local (R2L): An attacker exploits vulnerabilities to access a local account.

The NSL-KDD dataset consists of two awesome subsets: KDDTrain+, that's used for education the model, and KDDTest +, which serves because the trying out set. A super feature of the check dataset is that it carries assault sorts not blanketed in the schooling dataset. This component is vital because it demanding situations the classifier to identify and reply to unknown attacks, as a consequence assessing its potential to generalize and locate novel threats. Each sample in the NSL-KDD dataset is described by 41 attributes followed by a class label. The attributes, which are discrete and continuous can be categorized into (i) Basic features (ii) Contents features (iii) Traffic features. The specific characteristics and statistical details of the NSL-KDD datasets are outlined in Table (1) [24].

Table 1. The Characteristics of the NSL-KDD Dataset

CATEGORY	INDEX	FEATURES	DESCRIPTION
BASIC FEATURES (9)	1	Duration	Length of connection
	2	protocol type	Type of protocol (TCP, UDP...)
	3	Service	Destination service (ftp, telnet...)
	4	Flag	Status of connection
	5	source bytes	No. of B from source to destination
	6	destination bytes	No. of B from destination to source
	7	Land	If the source and destination address are the same land=1/if not, then 0
	8	wrong fragments	No. of wrong fragments
	9	Urgent	No. of urgent packets
CONTENT FEATURES (13)	10	Hot	No. of hot indicators
	11	Num_failed_logins	No. of unsuccessful attempts at login
	12	logged in	If logged in=1/if login failed 0
	13	Num_compromised	No. of compromised states
	14	Root_shell	If a command interpreter with a root account is running root shell=1/if not, then 0
	15	Su_attempted	If an su command was attempted, su attempted=1/if not, then 0 (temporary login to the system with other user credentials)
	16	Num_root	No. of root accesses
	17	Num_file_creations	No. of operations that create new files
	18	Num_shells	No. of active command interpreters
	19	Num_access_files	No. of file creation operations
	20	Num_outbound_cmds	No. of outbound commands in an ftp session
	21	is host login	is host login=1 if the login is on the host login list/if not, then 0
	22	is guest login	If a guest is logged into the system, is guest login=1/if not, then 0
	23	Count	No. of connections to the same host as the current connection at a given interval

TRAFFIC FEATURES (19)	24	Srv_count	No. of connections to the same service as the current connection at a given interval
	25	Serror_rate	% of connections with SYN errors
	26	Srv_error_rate	% of connections with SYN errors
	27	rerror rate	% of connections with REJ errors
	28	Srv_rerror_rate	% of connections with REJ errors
	29	Same_srv_rate	% of connections to the same service
	30	diff srv rate	% of connections to different services
	31	srv_diff_host_rate	% of connections to different hosts
	32	dst_host_count	No. of connections to the same destination
	33	dst_host_srv_count	No. of connections to the same destination that use the same service
	34	dst_host_same_src_rate	% of connections to the same destination that use the same service
	35	dst_host_srv_rate	% of connections to different hosts on the same system
	36	dst_host_same_srv_port rate	% of connections to a system with the same source port
	37	dst_host_srv_diff_host_rate	% of connections to the same service coming from different hosts
	38	dst_host_serror_rate	%_of connections to a host with an S0 error
	39	dst_host_srv_serror_rate	% of connections to a host and specified service with an S0 error
	40	dst_host_serror_rate	% of connections to a host with an RST error
	41	dst_host_srv_serror_rate	% of connections to a host and specified service with an RST error
	42	Class Label	

Any network behavior that deviates from "Normal" is considered to be an attack (class label). We used a dataset with 10,000 records to evaluate the proposed technique. This dataset comes from a famous intrusion detection framework. Each document has forty-one capabilities that help capture important details about network traffic. The records are labelled as either "normal" or one of four intrusion types: Denial of Service (DoS), Probe, User to Root (U2R), and Remote to Local (R2L).
The features are thoughtfully organized into three distinct groups to facilitate analysis:

- Basic features (9 total) these include intrinsic properties such as connection duration and protocol type.
- Content features (13 total) These encompass indicators derived from the packets' payload, providing
- Insights into the content being transmitted.
- Traffic features (19 total): These are based on the statistical properties of network traffic over time, helping to identify patterns and anomalies.

The assessment outcomes for the entire NSL-KDD dataset were specifically based on the most effective features, which were carefully selected within the previous phase of the analysis. This established method guarantees that the proposed technique is both rigorous and capable of yielding meaningful insights into intrusion detection.

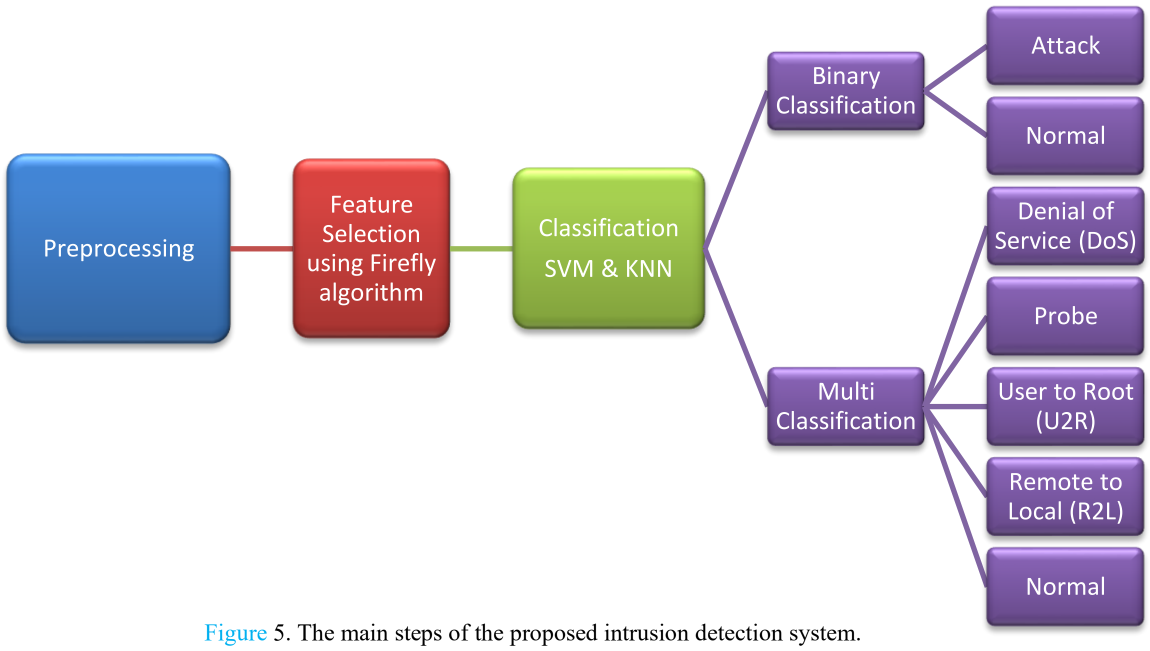


Figure 5. The main steps of the proposed intrusion detection system.

4. The proposed model

Two awesome fashions had been developed, one designed for binary classification tasks and the other tailored for multi-class type demanding situations. Both models leverage a more desirable model of the firefly algorithm, which has been thoughtfully modified to include a mutation operator. This modern integration targets to improve the set of rules' exploratory capabilities.
The Support Vector Machine (SVM) and K-Nearest Neighbor (KNN) classifiers are used to assess the effectiveness of the chosen features, offering robust assessment metrics. Within the framework of the genuine firefly algorithm (FFA), the pleasant firefly stands out tremendously; it stays desk-bound even as all other fireflies gravitate in the direction of it. This specific behaviour is important, as any failure of the set of rules to identify an advanced role within a few iterations can significantly diminish the overall performance of the firefly algorithm (FFA). As shown in Figure (5), the first and second classification models comprise three main steps: data preprocessing, feature selection, and classification.

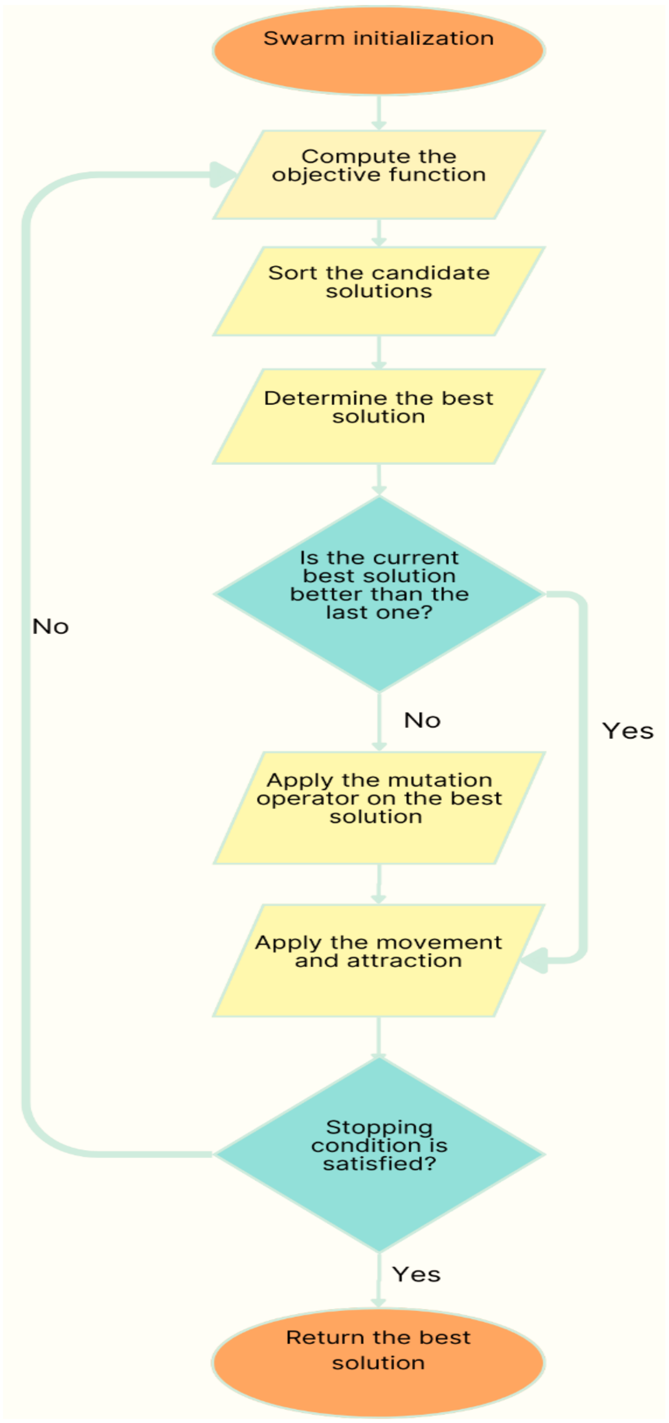

Swarm initialization
Compute the objective function
Sort the candidate solutions
Determine the best solution
Is the current best solution better than the last one?
No
No
Yes
Apply the mutation operator on the best solution
Apply the movement and attraction
Stopping condition is satisfied?
Yes
Return the best solution

Figure (6) illustrates a comprehensive methodology and process flow of the modified Firefly algorithm with an implementing flowchart.

To address this limitation, a Genetic Algorithm (GA) mutation operator is employed to introduce a random element that enhances the exceptional firefly's movement capabilities compared to the rest of the population. This technique facilitates the random repositioning of the excellent-discovered answer, taking into consideration clean, exploratory moves which could lead to improved positions in the search area. Consequently, this not only enhances the firefly algorithm's ability to traverse through various potential solutions but also significantly boosts its overall performance. The major steps of the proposed feature selection methods are listed below:

I. Swarm initialization

is a key step where each firefly in the population is randomly assigned a position within a continuous domain. This random positioning utilizes a uniform distribution, as detailed in Eq. (2). By ensuring a diverse starting point for each firefly, this method enhances the potential for effective exploration and optimization in subsequent iterations.

$$X_i = (UB - LB) * Rand \tag{2}$$

In this context, let Rand represent a random variable that is bounded within the interval [0, 1]. The upper bound (UB) is defined as (1, 0), and the lower bound (LB) is defined as (0, 0). To tackle the issue of feature selection, every firefly is initially encoded using a binary representation. To convert continuous values into binary values (0s and 1s), the sigmoid function, defined in Eq. (3), is utilized. In this encoding, the features selected are represented as 1, while non-selected features are described as 0.

$$\mathrm{B_i} = \begin{cases} 1 & \frac{1}{1+\mathrm{e}^{-\mathrm{X_i}}} > Random(0,1) \\ 0 & \text{Otherwise} \end{cases} \tag{3}$$

where X_i *and* B_iDenotes the continuous and binary values of the firefly positions, respectively.

II. Fitness function calculation

The fitness function guides the search by determining the quality of the possible solutions. The objective function evaluates these solutions based on their accuracy and number of features, using Eq. (4).

$$\min \mathrm{f(x)} = (100 - \mathrm{Accuracy}) \tag{4}$$

We calculate accuracy using K-Nearest Neighbor (KNN) and Support Vector Machine (SVM) classifiers. After finding the error rate, we determine the light intensity for each firefly by Eq. (5).

$$\mathrm{I}(\mathrm{F}_i) = \frac{1}{1+error^2} \tag{5}$$

III. Distance calculation

The distance from f_i to f_j is denoted by f_{ij} and computed by Eq. (6).

$$f_{ij} = \| X_i - X_j \| = \sqrt{\sum_{d=1}^{D}(X_{id} - X_{jd})^2} \tag{6}$$

where X_{id} is the position of a firefly i. The Euclidean distance is adopted to compute the distance between any two fireflies. In the proposed models, the variable Dindicates the number of features involved in the network intrusion detection process, which is equal to 41.

IV. Attractiveness

For each firefly, the attractiveness β is calculated by Eq. (7).

$$\beta_f = \beta_0 e^{-\gamma f^2} \tag{7}$$

Where f denotes the range between two fireflies and ($\beta_0 = 1$)indicates the attractiveness at $r = 0$ (first attractiveness).

V. Fireflies position update

The fireflies in the population are attracted to other fireflies that have a higher degree of light using Eq. (8), which means that the position of every firefly is updated continuously. Therefore, Eq. (2) is used to binarize the values of fireflies.

$$X_{new} = X_{old} + \beta * (X_j - X_i) + \alpha\,(rand - 0.5) \tag{8}$$

Eq. (8) has three main parts. The first part denotes the current position. On the other side, the second part includes the attractiveness between the positions of the firefly F_i and firefly F_j.

Finally, the third part denotes the random movement where α is the randomization parameter and $rand$ is a random number based on a uniform distribution that ranges between 0 and 1. Therefore, the value of $(rand - 0.5)$ In the third part of Eq. (8), the range is between -0.5 and 0.5 to support both positive and negative changes.

VI. Mutation Operator for Best Firefly

As previously mentioned, the best firefly remains in its original position in the standard Firefly Algorithm (FFA), which can slow down the search process and make the algorithm more susceptible to getting trapped in local maxima, whilst, in the suggested variant of the Genetic Algorithm-Firefly Algorithm (GA-FFA), a mutation operator from the GA is used to change the position of the best firefly by randomly exchanging certain features and variables. To implement this operator, several steps need to be taken. First, a random even number is selected to determine how many features will be swapped within the best firefly; this number is referred to as the removed features (RF). Next, half of these removed features are exchanged with the other half. For example, if RF is set to 2, it means that two random positions (r1 and r2) are swapped, as depicted in Figure (7).

The updated solution is then evaluated using the evaluation function (as shown in Equation (3)). If this new solution performs better than the original, it is retained; otherwise, the algorithm reverts to the previous solution. The mutation operator plays a crucial role in enhancing the search efficiency of firefly algorithms, allowing them to navigate more effectively through complex search spaces. By introducing random variations, this operator significantly diminishes the chances of the algorithm becoming trapped in local optima. It operates on both binary and continuous values. In other words, the random positions selected can influence the outcome across diverse types of data. This dual capability allows for a more robust exploration of potential solutions, leading to greater success and optimal results in the end.

.

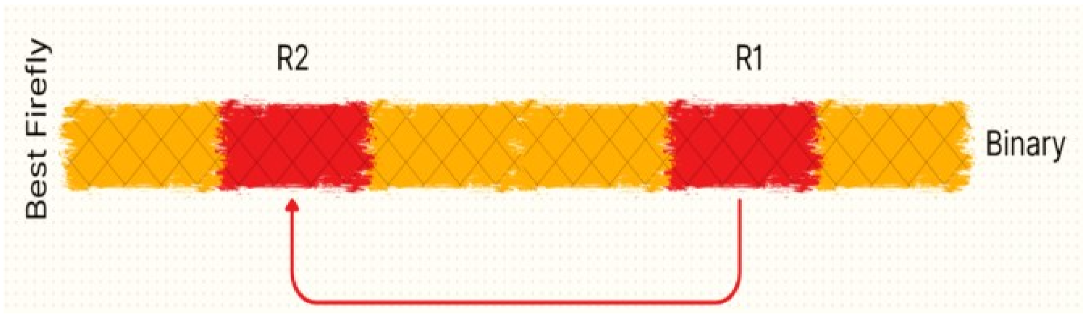


Figure 7.A schematic view of the mutation operator

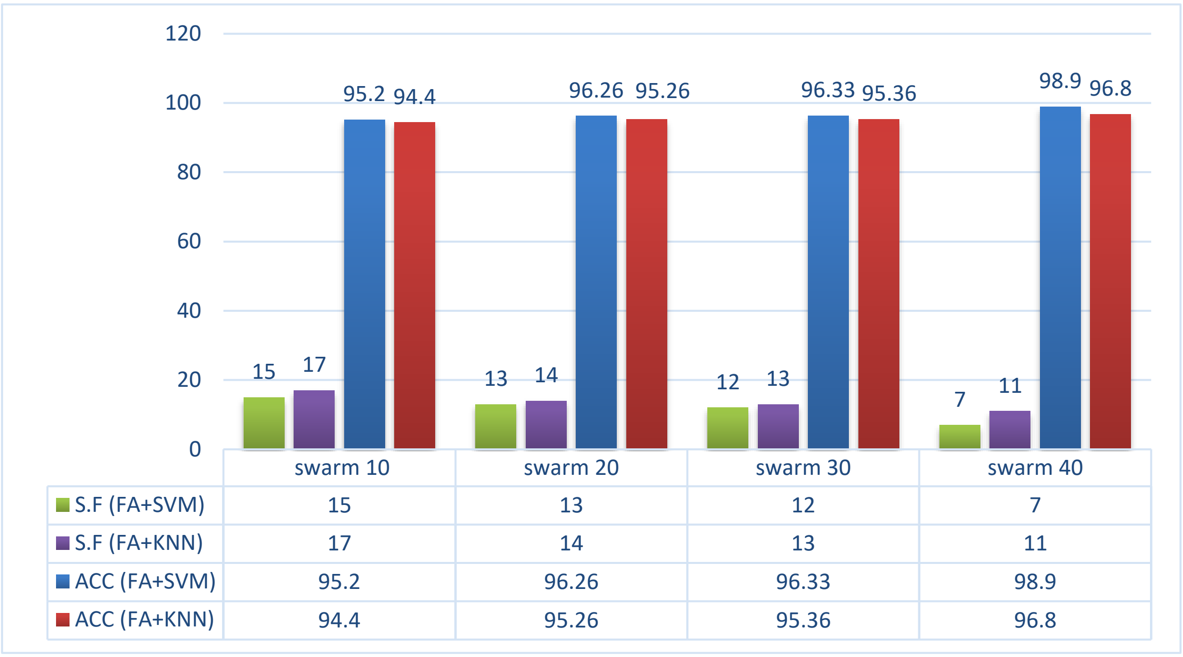
5. RESULTS AND DISCUSSIONS

Two common performance metrics have been used in the literature to assess the performance of intrusion detection models: the accuracy of classification and the number of selected features. The accuracy metric is formulated as shown below.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \qquad (9)$$

- **TP**: a shorthand for True Positives that indicates the number of positive cases that are correctly classified.
- **TN**: a shorthand for True Negatives that indicates negative cases that are correctly identified, while positive cases are incorrectly identified.
- **FP**: a shorthand for False Positives that indicates the negative cases that are mistakenly classified as positive.
- **FN**: a shorthand for False Negatives that indicates the positive cases that are mistakenly classified as negative.

All the experiments are executed on a PC that runs a 64-bit Windows 10 Pro operating system. It has a computing power of 3 GHZ i5 and a memory capacity of 8 GB RAM. Two performance metrics are mainly used to assess the proposed intrusion detection models: the selected features and the accuracy of classification. The best features subset has been used to train and test the proposed models. 70% of the dataset has been used in the training process, and 30% has been used in the test process. In this experiment, the proposed models are evaluated for achieving binary and multi-class classification to differentiate between the normal case and four classes of attacks: DOS, Probe, R2L, and U2R. For the first binary model, different population sizes have been used, including 10, 20, 30, and 40 agents, while the number of iterations has been set to 500, as shown in Figure (8).



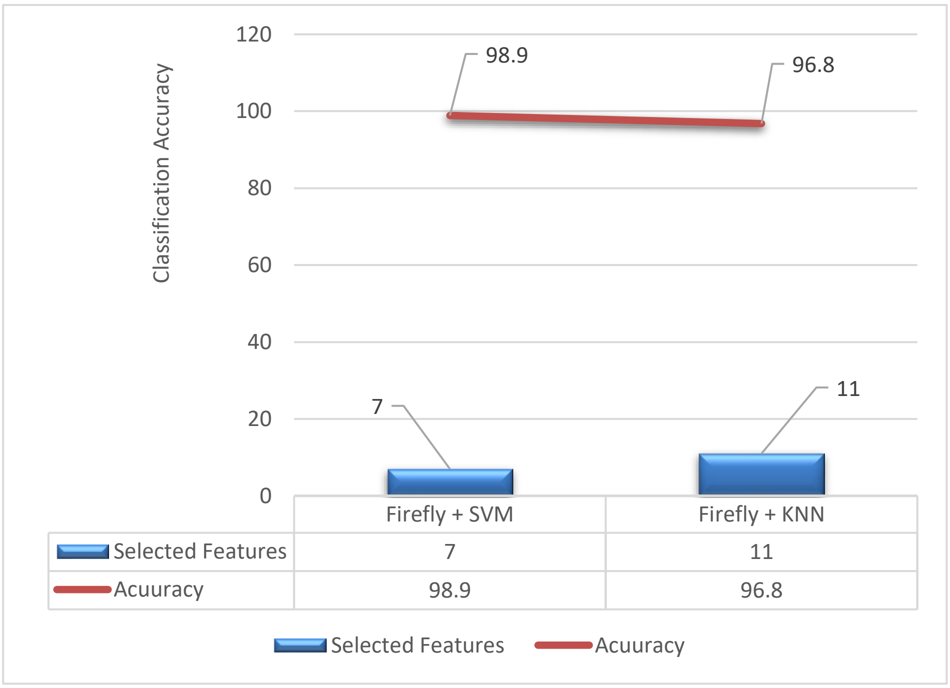
	swarm 10	swarm 20	swarm 30	swarm 40
🟩 S.F (FA+SVM)	15	13	12	7
🟪 S.F (FA+KNN)	17	14	13	11
🟦 ACC (FA+SVM)	95.2	96.26	96.33	98.9
🟥 ACC (FA+KNN)	94.4	95.26	95.36	96.8

S.F(Selected Feature), F**A**(Firefly), **SVM** (Support Vector Machine), **KNN** (K-Nearest Neighbour)

Figure 8. Results for the binary classification model with different population sizes

Each configuration population size aims to assess how the number of agents influences the overall performance of the system. This visual representation shows the outcomes and patterns observed for each population size throughout the 500 iterations, and the performance of the proposed model is rigorously evaluated for binary classification, focusing on its ability to differentiate between normal and abnormal cases accurately .The illustration above indicates that a swarm size of 40 yields excellent results when compared to alternative options. The accurateness of the outcomes correlates with the swarm size, a larger swarm yields higher quality outcomes.

These results are comprehensively illustrated in Figure (9) and Table (2), which compare model performance and highlight the optimal configurations for effective classification.



	Firefly + SVM	Firefly + KNN
🟦 Selected Features	7	11
🟥 Acuuracy	98.9	96.8

Figure 9. Proposed binary classification model

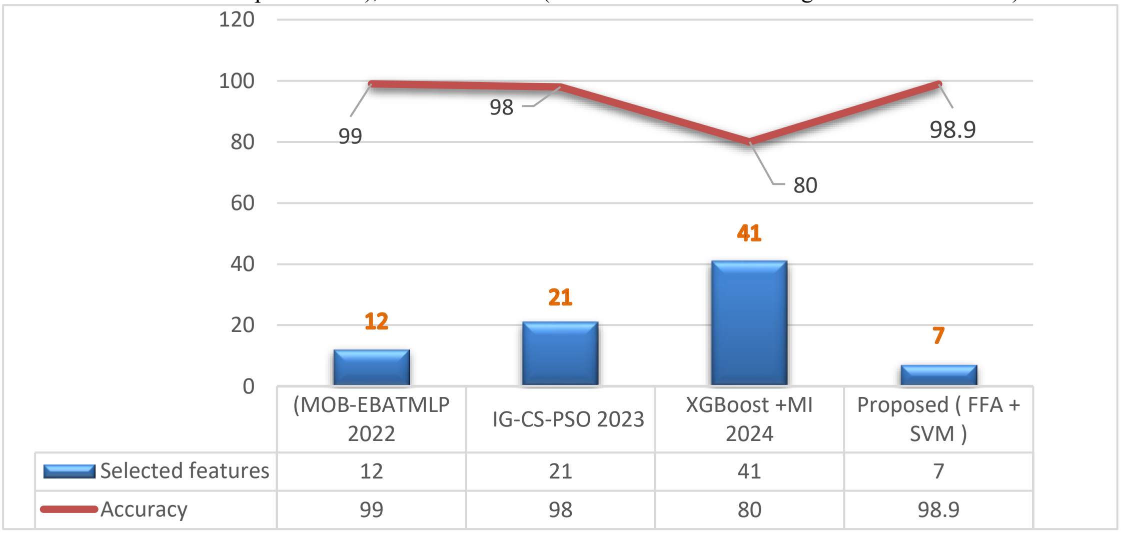
Table 2. Chosen features for the binary classification model

Model	chosen Features		Accuracy
Firefly + support vector machine	7	0,**2**,**3**,0,0,0,0,**8**,0,0,0,**12**,0,0,0,0,0,0,0,0,0,0,0,**24**,0,0,0,**28**,0,0,0,0,0,0,0,0,**37**,0, 0,0,0	98.9
		protocol_type, Service, wrong_fragment, logged_in, srv_count, srv_rerror_rate, dst_host_srv_diff_host _rate.	
Firefly +K-Nearest Neighbour	11	0,**2**,**3**,0,0,**6**,**7**,**8**,0,0,0,**12**,**13**,0,0,0,0,0,0,0,**21**,0,0,0,0,0,0,**28**,0,0,0,0,0,0,0,0,0,**38** ,0,0,**41**	96.8
		protocol_type, Service, dst_bytes, Land, wrong_fragmentlogged_in,num_compromised, is_host_login, srv_rerror_rate, dst_host_serror_rate, dst_host_srv_rerror_rate	

Based on the analysis in Figure (9), the Support Vector Machine (SVM) classifier confirmed the best performance in opposition to the K-Nearest Neighbour (KNN). It achieved an outstanding accuracy rate of 98.9% and identified 7 selected features, indicating its effectiveness in efficiently classifying record points. Furthermore, Table (2) provides a summary of the features that were chosen for the binary classification model. These features were extracted by employing the Firefly algorithm with a support vector machine, and k-nearest neighbors respectively. This highlights the effectiveness of the approach in selecting significant variables while keeping a high level of predicted accuracy.
This aggregate of overall performance metrics underscores the SVM classifier's robustness within the given undertaking. The proposed model integrates a modified firefly algorithm with support vector machines (SVM) to enhance binary classification performance.

This model was compared against several existing binary classification approaches, specifically those developed by Ghanem et al.[15], Bakro et al.[16], and Faizin et al.[17]. Ghanem et al. [15] Employed a sophisticated wrapper approach-based feature selection algorithm in the first stage, built on a multi-objective BAT algorithm (MOBBAT). In the second stage, the features obtained from the first stage are used to classify traffic using an enhanced BAT algorithm (EBAT) specifically designed for training a multilayer perceptron (EBATMLP). This approach aims to improve the performance of the Intrusion Detection System (IDS). The resulting methodology is referred to as MOB-EBATMLP. Bakro et al.[16] Presented an improved cloud intrusion detection system (IDS) that incorporates the Synthetic Minority Over-sampling Technique (SMOTE) to address the issue of imbalanced data. For feature selection, A hybrid approach has been proposed that combines three techniques: Information Gain (IG), Chi-Square (CS), and Particle Swarm Optimization (PSO). Finally, the Random Forest (RF) model has been utilized to detect and classify various types of attacks. Faizin et al.[17] Proposed an Intrusion Detection System (IDS) that integrates mutual information with threshold-based feature selection and the XGBoost classification algorithm. Mutual information is utilized to assess the dependency between each input feature and the target features. Once the amount of information is determined through mutual information, thresholding is applied to identify the optimal number of features for the classification process. Finally, the data is classified using the features selected by XGBoost. The comparison evaluated two critical metrics: the number of features selected for classification and the accuracy of the classification results. Figure (10), presents in detail the findings of this comprehensive evaluation, highlighting each model's strengths and weaknesses. This analysis provides valuable insights into the effectiveness of the modified Firefly algorithm in optimizing feature selection and improving classification outcomes.

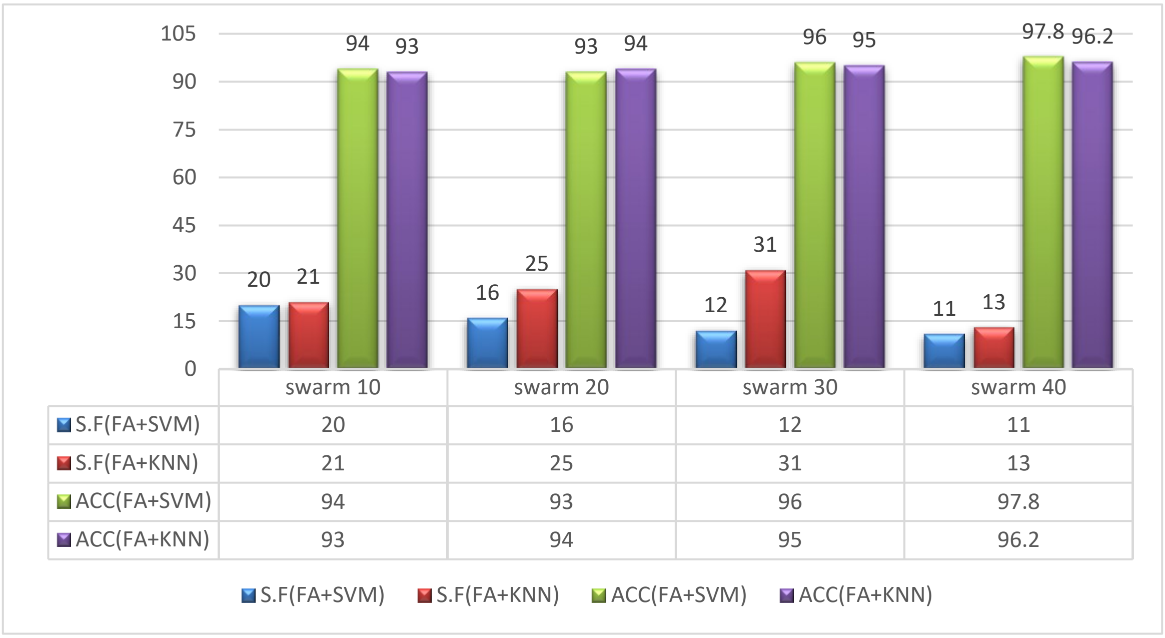
MOB-EBATMLP (Multi-Objective-Enhanced Bat Multilayer Perceptron), IG-CS-PSO (Information Gain, Chi Square, Particle Swarm Optimization), XGBOOST-MI (Extreme Gradient Boosting-Mutual Information)



	(MOB-EBATMLP 2022	IG-CS-PSO 2023	XGBoost +MI 2024	Proposed (FFA + SVM)
▬ Selected features	12	21	41	7
▬ Accuracy	99	98	80	98.9

Figure 10. Comparison between the best-proposed model and other binary models

According to Figure (10), the firefly algorithm combined with the SVM classifier gives the best results by reducing the number of selected features from 41 to just 7. In comparison, the results gathered from the studies conducted by Ghanem et al.[15], Bakro et al.[16], and Faizin et al. [17] Indicate poor performances in the context of selected features. In the second proposed model, we evaluate performance within a multi-class classification scenario aimed at distinguishing between normal behaviour and four distinct types of cyber attacks: Denial of Service (DOS), Probe, Remote to Local (R2L), and User to Root (U2R). To ensure a fair and rigorous comparison among the various classification models employed, different population swarm sizes have been used: 10, 20, 30, and 40. We established a maximum of 500 iterations for the testing process, as shown in Figure (11).



	swarm 10	swarm 20	swarm 30	swarm 40
🟦 S.F(FA+SVM)	20	16	12	11
🟥 S.F(FA+KNN)	21	25	31	13
🟩 ACC(FA+SVM)	94	93	96	97.8
🟪 ACC(FA+KNN)	93	94	95	96.2

S.F(Selected Feature), **FA**(Firefly), **SVM** (Support Vector Machine), **KNN** (K-Nearest Neighbour)
Figure 11. Results for the multi-classification model with different population sizes

Furthermore, each classification model was thoroughly assessed based on the number of features selected and the overall classification accuracy achieved, as illustrated in Figure (12) and Table (3):

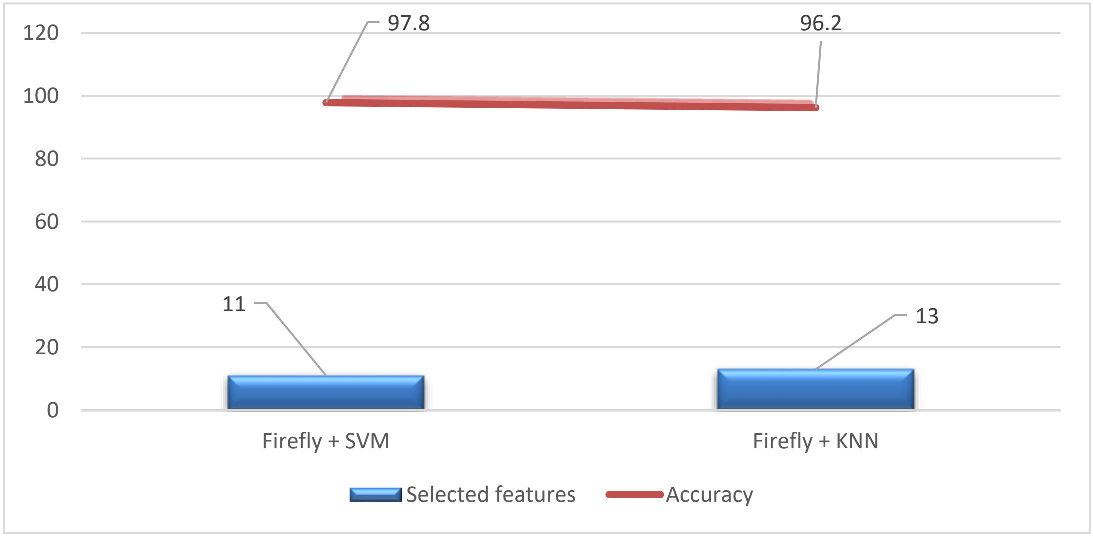


Figure 12. Proposed multi-classification model

As illustrated in Figure (12), the Support Vector Machine (SVM) classifier demonstrates outstanding performance with a classification accuracy of 97.8%, making it the leading choice among the various classifiers evaluated in this study. This high level of accuracy indicates the SVM's robust ability to categorize data points and minimize prediction errors accurately.

In contrast, the K-Nearest Neighbors (KNN) classifier, while achieving a commendable accuracy of 96.2%, falls short of the SVM's performance, highlighting potential limitations in its predictive capabilities. Table (3) outlines the features selected throughout the analysis process. The Support Vector Machine (SVM) classifier demonstrated an outstanding ability to recognize 11 essential features, which are crucial for enhancing the model's predictive performance. This option improves the model's accuracy while simplifying its complexity, resulting in a more efficient and understandable outcome. The K-Nearest Neighbors (KNN) classifier, on the other hand, selected a total of 13 features. The difference in the number of features selected by the two classifiers underscores the distinct approaches used in feature selection and their effects on model performance.

The proposed model has outperformed other multi-class classification models in comparative analyses, including those by Choobdar et al.[25], Almutairi et al.[26], and Ferrão et al.[27]. Choobdar et al.[25] proposed an intrusion detection system that utilizes the Software-Defined Network (SDN) model and operates as an application module within the controller. This system consists of three distinct phases. In the first phase, sparse stacked autoencoders are employed for pre-training, allowing the model to learn features in an unsupervised manner. The second phase involves training the system using a SoftMax classifier. Finally, in the third phase, the system parameters are optimized for improved performance.

Various machine learning algorithms have been utilized by Almutairi et al.[26] to assess Network Intrusion Detection Systems (NIDS), including Support Vector Machine, J48, Random Forest, and Naïve Bayes. These evaluations have been conducted using both binary and multi-class classification methods. Ferrão et al.[27] Designed a Multi-Attack Intrusion Detection System (MAIDS) for Software-Defined IoT Networks(SDN-IoT). The proposed system utilizes two machine learning algorithms to enhance detection efficiency and establish a mechanism that minimizes false alarms. Initially, a comparative analysis of the most commonly used machine learning algorithms for securing SDN is conducted to identify the most suitable algorithms for the proposed scheme and for protecting SDN-IoT systems. The evaluation algorithms include: Extreme Gradient Boosting (XGBoost), K-Nearest Neighbors (KNN), Random Forest (RF), Support Vector Machine (SVM), and Logistic Regression (LR).

Table 3. Chosen features for the multi-classification model

Model	chosen Features		Accuracy
Firefly + support vector machine	11	**1**,0,0,0,0,0,0,0,**9**,**10**,0,0,**13**,0,0,0,**17**,0,0,0,0,**22**,0,0,0,0,0,**28**,0,0,0,**32**,**33**,0,0,0,0,0,0,0,**40**,**41**	97.8
		Duration, Urgent, Hot, num_compromised, num_file_creations, is_guest_login, srv_rerror_rate, dst_host_count, dst_host_srv_count, dst_host_rerror_rate, dst_host_srv_rerror_rate	
Firefly +K-Nearest Neighbour	13	0,0,**3**,0,**5**,**6**,0,0,**9**,0,0,0,**13**,0,0,0,0,0,**19**,0,0,0,**23**,0,**25**,0,0,**28**,0,**30**,0,0,0,0,**35,36**,0,0,0,0,**41**	96.2
		Service, src_bytes, dst_bytes, Urgent, num_compromised, num_access_files, Count, serror_rate, srv_rerror_rate, diff_srv_rate, dst_host_diff_srv_rate, dst_host_same_src_port_rate, dst_host_srv_rerror_rate	

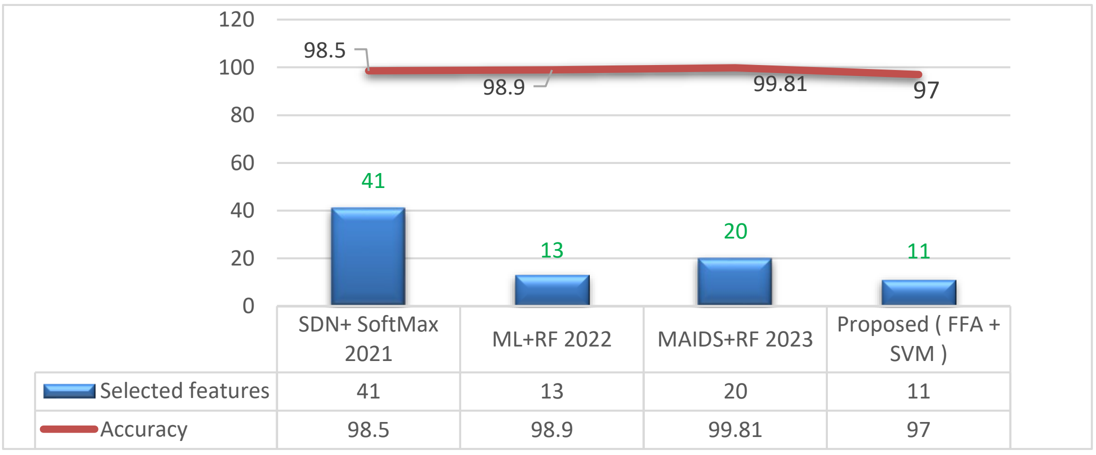


Figure (13) demonstrates the comparison results concerning selected features and the accuracy of classification.

The models proposed in Choobdar et al.[25], Almutairi et al.[26], and Ferrão et al.[27] Have the worst results on feature selection. Conversely, the models in Choobdar et al.[25], Almutairi et al.[26], and Ferrão et al.[27] Have better multi-classification accuracy. Additionally, Figure (14) gives the accuracy of the proposed models concerning the different attacks, including DoS, Probe, U2R, and R2L, as well as the normal case.

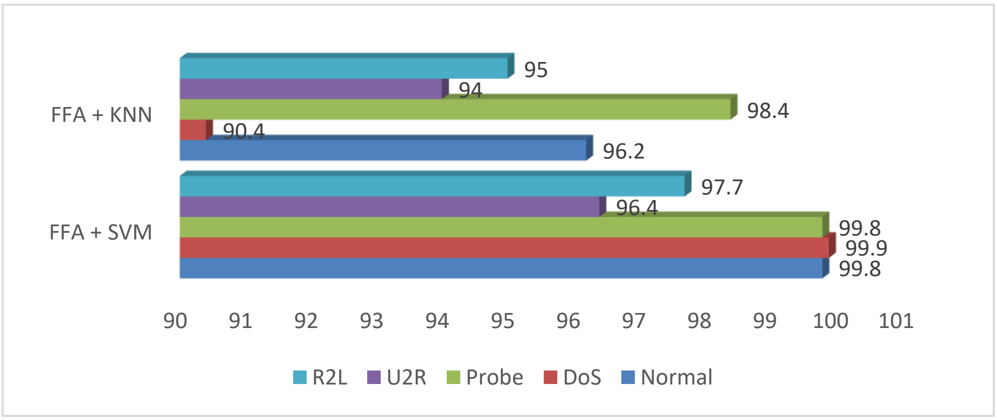


Figure 14. Classification accuracy for each attack class

According to the information presented in Figure (14), the Support Vector Machine (SVM) classifier demonstrates the highest level of performance among the various methodologies evaluated. This suggests that the SVM is especially powerful in appropriately distinguishing between the one-of-a-kind classes within the dataset. In comparison, the K-Nearest Neighbors (KNN) classifier indicates the least favorable results, indicating that it struggles more with category accuracy than the other analyzed procedures. This disparity highlights the SVM's superiority in handling the unique characteristics of the information. At the same time, the KNN's performance can be hindered by its reliance on local data points for classification.

6. CONCLUSION

This study presents two novel approaches to network intrusion detection by combining a modified Firefly Algorithm with a mutation operator (FFA) for feature selection with two classic classifiers: Support Vector Machine (SVM), and K-Nearest Neighbor (KNN). The NSL-KDD dataset has been utilized to assess the two models. The suggested methodologies utilize a wrapper approach to the selection of relevant features. This reduced dataset will increase the performance and detection accuracy of the SVM and KNN-based detection model. Moreover, the training and testing time will also be reduced with a reduced set of features.

Binary classification (normal vs. attack): The proposed FFA with SVM achieved the best performance, reaching an accuracy of 98.9% using only 7 features. The FFA variant with KNN also performed well (96.8% with 11 features), but SVM consistently provided superior accuracy and better generalization in the binary setting.

Multi-class classification (DoS, Probe, U2R, R2L, Normal): For distinguishing among five classes, FFA with SVM delivered the highest overall accuracy of 97.8%, using 11 features. The KNN counterpart achieved 96.2%, using 13 features. These results underscore the robustness of SVM when paired with the FFA feature subset in handling multiple attack classes.
In summary, the study demonstrates that a modified Firefly Algorithm for feature selection, when coupled with SVM, yields high-accuracy intrusion detection with a substantially reduced feature set. This contributes a practical, efficient, and effective method for enhancing IDS performance in modern network environments.

Source of funding: The source of funding is self-funding.
Conflict of interest: There is no conflict of interest for the authors.
Ethical clearance: Ethical approval was not required for this review.

7. REFRENSES

[1] K. Scarfone and P. Mell, “NIST Special Publication 800-94: Guide to Intrusion Detection and Prevention Systems (IDPS). Recommendations of the National Institute of Standards and Technology,” *Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology, Gaithersburg, MD, United States*, pp. 20899–28930, 2007.
[2] M. Ammar, G. Russello, and B. Crispo, “Internet of Things: A survey on the security of IoT frameworks,” *Journal of Information Security and Applications*, vol. 38, pp. 8–27, 2018.

[3] K. P. Hiba Fathima and P. P. Anugraha, “A Review on Network Intrusion Detection,” *International Journal of Scientific Research and Technology*, vol. 2, no. 12, p. 1, 2025, doi: 10.5281/zenodo.14598998.

[4] S. M. Kasongo and Y. Sun, “A deep gated recurrent unit based model for wireless intrusion detection system,” *ICT Express*, vol. 7, no. 1, pp. 81–87, 2021, doi: 10.1016/j.icte.2020.03.002.

[5] N. T. T. Van and T. N. Thinh, “Accelerating anomaly-based IDS using neural network on GPU,” in *2015 international conference on Advanced Computing and Applications (ACOMP)*, IEEE, 2015, pp. 67–74. doi: 10.1109/ACOMP.2015.30.

[6] T. Whare W-ananga, W. Hamilton, and M. A. Hall, “Correlation-based Feature Selection for Machine Learning,” 1999.

[7] P. Sharma, J. Sengupta, and P. K. Suri, “Intrusion Detection Using Data Mining in Cloud Computing Environment.” [Online]. Available: http://www.publishingindia.com/ijdcc

[8] 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, doi: 10.1016/j.jnca.2011.01.002.

[9] S.-J. Horng *et al.*, “A novel intrusion detection system based on hierarchical clustering and support vector machines,” *Expert Syst Appl*, vol. 38, no. 1, pp. 306–313, 2011, doi: 10.1016/j.eswa.2010.06.066.

[10] C. Khammassi and S. Krichen, “A GA-LR wrapper approach for feature selection in network intrusion detection,” *Comput Secur*, vol. 70, pp. 255–277, 2017, doi: 10.1016/j.cose.2017.06.005.

[11] O. Osanaiye, H. Cai, K.-K. R. Choo, A. Dehghantanha, Z. Xu, and M. Dlodlo, “Ensemble-based multi-filter feature selection method for DDoS detection in cloud computing,” *EURASIP J Wirel Commun Netw*, vol. 2016, pp. 1–10, 2016, doi: 10.1186/s13638-016-0623-3.

[12] B. Ingre and A. Yadav, “Performance analysis of NSL-KDD dataset using ANN,” in *2015 international conference on signal processing and communication engineering systems*, IEEE, 2015, pp. 92–96. doi: 10.1109/SPACES.2015.7058223.

[13] A. H. Sung and S. Mukkamala, “Identifying important features for intrusion detection using support vector machines and neural networks,” in *2003 Symposium on Applications and the Internet, 2003. Proceedings.*, IEEE, 2003, pp. 209–216. doi: 10.1109/SAINT.2003.1183050.

[14] B. Selvakumar and K. Muneeswaran, “Firefly algorithm based feature selection for network intrusion detection,” *Comput Secur*, vol. 81, pp. 148–155, 2019, doi: 10.1016/j.cose.2018.11.005.

[15] W. A. H. M. Ghanem *et al.*, “Cyber Intrusion Detection System Based on a Multiobjective Binary Bat Algorithm for Feature Selection and Enhanced Bat Algorithm for Parameter Optimization in Neural Networks,” *IEEE Access*, vol. 10, pp. 76318–76339, 2022, doi: 10.1109/ACCESS.2022.3192472.

[16] M. Bakro *et al.*, “An Improved Design for a Cloud Intrusion Detection System Using Hybrid Features Selection Approach With ML Classifier,” *IEEE Access*, vol. 11, pp. 64228–64247, 2023, doi: 10.1109/ACCESS.2023.3289405.

[17] M. A. Faizin, D. T. Kurniasari, N. Elqolby, M. A. R. Putra, and T. Ahmad, “Optimizing Feature Selection Method in Intrusion Detection System Using Thresholding,” *International Journal of Intelligent Engineering and Systems*, vol. 17, no. 3, pp. 214–226, 2024, doi: 10.22266/ijies2024.0630.18.

[18] X.-S. Yang, “Firefly Algorithms for Multimodal Optimization,” Mar. 2010, doi: 10.1007/978-3-642-04944-6_14.

[19] J. Jha and L. Ragha, “Intrusion detection system using support vector machine,” 2013. [Online]. Available: www.ijais.org

[20] S. M. H. Bamakan, H. Wang, T. Yingjie, and Y. Shi, “An effective intrusion detection framework based on MCLP/SVM optimized by time-varying chaos particle swarm optimization,” *Neurocomputing*, vol. 199, pp. 90–102, 2016, doi: 10.1016/j.neucom.2016.03.031.

[21] S. Peddabachigari, A. Abraham, C. Grosan, and J. Thomas, “Modeling intrusion detection system using hybrid intelligent systems,” *Journal of network and computer applications*, vol. 30, no. 1, pp. 114–132, 2007, doi: 10.1016/j.jnca.2005.06.003.

[22] W. Wang, X. Zhang, and S. Gombault, “Constructing attribute weights from computer audit data for effective intrusion detection,” *Journal of Systems and Software*, vol. 82, no. 12, pp. 1974–1981, 2009, doi: 10.1016/j.jss.2009.06.040.

[23] S. A. Dudani, “The distance-weighted k-nearest-neighbor rule,” *IEEE Trans Syst Man Cybern*, no. 4, pp. 325–327, 1976, doi: 10.1109/TSMC.1976.5408784.

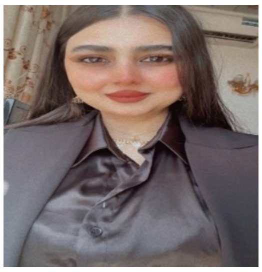
[24] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, “A detailed analysis of the KDD CUP 99 data set,” in *2009 IEEE symposium on computational intelligence for security and defense applications*, Ieee, 2009, pp. 1–6. doi: 10.1109/CISDA.2009.5356528.

[25] P. Choobdar, M. Naderan, and M. Naderan, “Detection and Multi-Class Classification of Intrusion in Software Defined Networks Using Stacked Auto-Encoders and CICIDS2017 Dataset,” *Wirel Pers Commun*, vol. 123, no. 1, pp. 437–471, Mar. 2022, doi: 10.1007/s11277-021-09139-y.

[26] Y. S. Almutairi, B. Alhazmi, and A. A. Munshi, “Network Intrusion Detection Using Machine Learning Techniques,” *Advances in Science and Technology Research Journal*, vol. 16, no. 3, pp. 193–206, 2022, doi: 10.12913/22998624/149934.

[27] T. Ferrão, F. Manene, and A. A. Ajibesin, "Multi-Attack Intrusion Detection System for Software-Defined Internet of Things Network," *Computers, Materials and Continua*, vol. 75, no. 3, pp. 4985–5007, 2023, doi: 10.32604/cmc.2023.038276.

BIOGRAPHIES OF AUTHORS

	Karrar Mohsin Alwan is Asst. Lecturer at Baquba Technical College, Middle Technical University, Iraq. He received the B.Sc. degree in computer science from College of Science, University of Diyala and M.Sc. degree from Faculty of Computers and Artificial Intelligence, Benha University, Arab Republic of Egypt. His research area are networking and Wireless communication, Artificial Intelligence. He can be contacted at email: karrar.mohsin@mtu.edu.iq.
	Ahmed Saad Mohammed received his bachelor's degree in Software Engineering from the Baghdad College of Economic Sciences University, Iraq, in 2007, and a master's degree from the Faculty of Computers & Informatics, Benha University, Egypt in 2019. He is currently an Assistant Lecturer at Al-Mustansiriya University, College of Basic Education, in the Computer Department. His research interests include machine learning, data mining, and Artificial Intelligence. He can be contacted at email: ahmed.saad@uomustansiriyah.edu.iq
	A. S. Abohamama was born in Egypt in 1985. He received B.Sc., M.Sc., and PhD. in Computer Sciences from Mansoura University in 2006, 2012, 2018, respectively. He worked as an Assistant Professor at the Department of Computer Sciences, University of Mansoura until 2022. Now, he works as an Associate Professor at the Department of Computer Sciences, Arab East Colleges, KSA. His current research interests include Distributed Systems, Cloud Computing, IoT and Big data analytics, Image Processing, and Biometrics. He can be contacted at email: abohamama@mans.edu.eg
	Altameemi Ali Najm is an Assistant Lecturer at the Technical College in Baqubah. He holds a Master's degree in Software Engineering from Bucharest, Romania, where his thesis focused on Vehicular Ad Hoc Networks (VANET). He can be contacted at Academic email: mailto:ali.najm@mtu.edu.iq
	**Walaa Khalil** Abrahem is an Assistant Lecturer at the College of Science, University of Diyala, Iraq. She holds a Bachelor's degree in Computer Engineering from the College of Engineering, University of Diyala, and a Master's degree from the College of Engineering, Iraqi University, Baghdad. She can be contacted via email at: walaa.khalil.abrahem@gmail.com.