ISSN: 2093-5374 / E-ISSN: 2093-5382

Feature Selection Model-Based Intrusion Detection System for Cyberattacks on the Internet of Vehicles Using Cat and Mouse Optimizer

Deepthi Reddy Dasari[1*], and Dr.G. Hima Bindu[2]

[1*]Research Scholar, Department of Computer Science and Engineering, GITAM University, Hyderabad, India. deepthi.phd20@gmail.com, https://orcid.org/0000-0003-2201-6452

[2]Assistant Professor, Department of Computer Science and Engineering, GITAM University, Hyderabad, India. hgottumu@gitam.in, https://orcid.org/0000-0003-4996-9050

Received: February 22, 2024; Revised: April 18, 2024; Accepted: May 21, 2024; Published: June 29, 2024

Abstract

The Internet of Vehicles (IoV) is an environment that is changing quickly. To protect user data and car safety, cyber security is very important. Choosing the right features is important for making intruder detection systems (IDS) more useful and efficient in Internet of Things (IoT) situations. This paper shows a new way to use optimization methods for feature selection-based intruder detection in IoV. Optimization methods are used in the proposed framework for selecting the most important attributes. This makes detection more accurate and reduces the cost of computing. The population-based optimization technique formula is a linked, important, and helpful way to resolve optimization issues. The Cat and Mouse Optimizer (CMO) is a new way to optimize based on how mice and cats naturally act. The proposed CMO acts like cats attacking mice and mice running away to safe places. Popular methods like Kernel Linear Discriminant Analysis (KDA) and the Cuckoo technique are used to compare the CMBO's results. The research showed that the CMO-Bi-LSTM model was very accurate on the CICIDS-2018 and Car hacking datasets, with scores of 99.36% and 99.80%, respectively. Testing and analysis show that the proposed approach successfully finds attacks while reducing false positives. The flexible and expandable nature of the system makes it perfect for securing IoV settings from new hacking attacks.

Keywords: Internet of Vehicles, Bi-LSSTM, CMO, Intrusion Detection System, Feature Selection.

1 Introduction

The advancement of the IoV has revolutionized transportation networks by delivering connectivity, safety, and efficiency. However, the fact that IoV includes a lot of different networked devices and contact routes makes security very worried. Intrusion detection systems (IDS) are a big part of keeping IoV networks safe from illegal attacks, data theft, and denial-of-service attacks (Mohammed, 2019; Gharkan & Abdulrahman, 2023). The "Internet of Vehicles" (IoV) is a term for cars that can connect to the internet and other networks for contact. IoV makes a lot of different uses possible, such as managing traffic, self-driving cars, remote repairs, and video services. IDS are safety systems that are meant to find attacks or illegal breaches on a system. Within the Internet of Vehicles, intrusion detection systems

Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA), volume: 15, number: 2 (June), pp. 251-269. DOI: *10.58346/JOWUA.2024.I2.017*

*Corresponding author: Research Scholar, Department of Computer Science and Engineering, GITAM University, Hyderabad, India.

(IDS) are very important for keeping the system safe from hacking. Traditional IDS systems often have a lot of extra work to do and a lot of fake alarms (Gou et al., 2023; Sreenivasu et al., 2022) because IoV settings make a lot more data. To solve these issues, feature selection techniques have become a well-known way to make IDS work better and more efficiently in IoV (Ali et al., 2023). Feature selection is the process of finding the most reliable and relevant group of features from a large amount of data. This makes computing easier and improves the accuracy of classification (Jegan et al., 2022). Feature selection is the best way to choose a group of similar characteristics for a model. The goal of feature selection is to make the data more efficient. Choosing the right features is important in many areas (Win & Kham, 2019; Liu & Motoda, 2012; Al-Tashi et al., 2020), such as intrusion detection systems (Sahoo & Chandra, 2017), wrapper (Emary et al., 2016), filter (Al-Tashi et al., 2019), and embedding methods (Sahoo & Chandra, 2017; Alamer et al., 2023) for selecting the right features. As part of the training process, the built-in method uses the feature selection method for the chosen learning algorithm. Several learning algorithms are able to predict characteristics when they use the wrapper-based method. Feature selection methods and optimization techniques have been examined in the context of IDS creation for IoV settings in previous years (Fung., 2011; Camgözlü et al., 2023). The study looked at PCA, information gain, and recursive feature reduction in IoV settings. The increase in computing complexity may be caused by more complicated data, longer working times, and less reliable classification. Utilizing optimization techniques, the most similar attributes from a large data set has been chosen. Its lowering the number of dimensions and raising the efficiency of the IDS. Focusing on the most important characteristics that gather on both normal and attack behavior in IDS detection works better with optimization methods (Srinivasareddy et al., 2021; Sinha et al., 2013; Aswathy et al., 2023). Optimization algorithms can extract the most useful features of intruder detection models to make them more accurate, sensitive, and specific (Yagız et al., 2022). To make the parameters of breach detection models work better, researchers have used optimization methods. One method is hyperparameter tuning of machine learning and deep learning model design (Asl et al., 2022). As researchers work in this area, their major goal has been to make IDS for IoV more effective, accurate, and scalable (Nobles et al., 2011). The goal of optimisation techniques is to find the best solution to a given issue within a predetermined set of restrictions. In IDS for IoV, optimisation methods can be used to improve feature selection, parameter values, or the structure of the detection model. Some common optimisation strategies are simulated annealing, genetic algorithms, differential evolution, ant colony optimisation, and particle swarm optimization (Babenko et al., 2021; Padmanabhan et al., 2019).

In this paper, CMO employs two populations, representing cats (predators) and mice (prey), to balance exploration and exploitation. Cats explore the search space to discover potential feature subsets, while mice exploit promising regions to refine feature selection. The interaction between cats and mice reveals a revamped search approach that adapts to the evolving characteristics of a search space. Cats chase mice to exploit valuable features, whereas mice evade cats to explore new regions, fostering a diverse exploration of the feature space. While minimizing the number of selected features, CMO can handle multi-objective optimisation objectives, such as maximizing detection accuracy. Cats perform global searches by exploring the entire feature space to identify promising subsets. Mice conduct local searches by exploiting local information to refine feature subsets and improve detection performance. We design an intelligent IDS for both the internet of vehicles and general networks by using random forest, decision tree, and Bi-LSTM methods to enhance the efficiency of the proposed IDS and perform an analysis of network attacks. We conduct experimental evaluations on the intrusion detection and IoV datasets. By using the special features of CMO, IDS for IoV can choose informative features while keeping detection accuracy high and reducing the amount of work that needs to be done on the computer.

CMO-based feature selection enables IDS to adapt to the dynamic nature of vehicular networks, increase resilience to cyber threats, and improve overall security and reliability in IoV environments.

2 Related Work

Given its independence in decision-making, which could determine life and death, the IoV's unique features, including its fast accessibility and regular node operations, raise safety concerns. The advancement of safer interconnected vehicles has sparked significant interest among individuals.

In (Escorcia-Gutierrez et al., 2023), the authors proposed the Sea Turtle Foraging Algorithm to identify and categories intrusions using a hybrid deep learning-based intrusion detection system for an IoT environment (STFA-HDLID) (Priyanka et al., 2023). This suggests that pre-processing the data using min-max normalisation is necessary. Additionally, they used the STFA for feature selection. Finally, the final step is classifying the data by a Deep Belief Network (DBN) with the tandem operation of the Sparrow Search Optimisation (SSO) method. Successful tests on a standardized dataset depict the accuracy of STFA-HDLID, at 99.51%, TON_IoT at 99.51%, and UNSW-NB15 at 98.85% compared with the existing solutions. In (Al Tawil & Sabri, 2021), the authors described a wrapper feature selection which employs this technique. The moth flame optimisation method (MFO) is a new algorithm derived from the natural moth's lateral location and navigation ability. They tested a suggested approach on the CIC-2017 dataset. They compare the proposed approach to a different wrapper technique: attribute-correlation-based feature selection, which uses a best-first search scenario to consider all features without selection. Adopting a MFO feature reduction technique from 78 features to 4 features and a DT classifier led to a 100% detection rate, a 99.9% accuracy rate, and a decreased false alarm rate, according to research results.

In (Hussein et al., 2023), the authors developed a new type of algorithm that was called the improved version, known as GMSMFO, to manage the tradeoff between exploration and exploitation activities, and also create diverse population variants. The system uses a modified approach in which a Gaussian method causes the bacteria's lethal mutation. The study compares the GMSMFO algorithm's results with datasets from the 2017 Imbecile Competition and 20 datasets that include a sophisticated intrusion prevention system, with the last being the focus. Statistical approaches, for example, the Friedman and Wilcoxon rank-sum tests, help to investigate the selection of the best methods for dealing with challenging meta-heuristics. This algorithm is superior and almost always the best of its kind, as stated in the research. As a result, it may be overfitting, but it might be capable of optimizing the feature subset, increasing the classification precision level while reducing the feature definitiveness index. One of the significant impacts of the Scholar paper is its exploration of the balance between the discoverer and the exploiter, as well as its focus on a broader region. Genomes and mutations contribute to soil differentiation and movement. This study employs the GMSMFO algorithm to score axial and classical versions of similar algorithms for 29 test functions. They also use it to evaluate the 20 (binary) spaces for all possible feature selections for intrusion detection systems. However, they also use these statistics tests, like the Wilcoxon rank-sum and Friedman test, to evaluate the effectiveness of GMSMFO, which incorporates various algorithms.

In (AlGhamdi, 2023), the authors' suggested Network Intrusion Detection System (NIDS-LOFSDL) method includes both deep learning and Lion optimisation feature selection. The NIDS-LOFSDL technique for intrusion detection utilises a hyperparameter-tuned deep learning model in conjunction with the FS principle. The NIDS-LOFSDL technique is useful because it applies the LOFS methodology to FS, resulting in higher classification rates. In addition, they used a process known as attention-based

bidirectional long short-term memory (ABiLSTM) to identify intrusions. In order to make the ABiLSTM algorithm better at finding intrusions, the gorilla troop optimizer (GTO) has to change hyperparameters. Because manually setting hyperparameters by trial and error takes a lot of time, this work stands out using the GTO-based hyperparameter tuning method. They did a lot of tests to make sure that the NIDS-LOFSDL system's improved intrusion detection answer worked. The simulations show that the NIDS-LOFSDL system is more accurate than other deep learning systems. It achieved a maximum accuracy of 96.88% for the UNSW-NB15 dataset and 96.92% for the AWID dataset.

In (Balasaraswathi et al., 2022), the authors proposed three crucial steps during their discussion: data preprocessing, feature selection, and classification. The data preparation stage entails removing input data, which may include noisy signals, high-dimensional and redundant data, other unnecessary features, and so on. The second phase involves selecting features using a search-based learning technique that is both cooperative and competitive (C2). The classification step uses the B-HkNN method, which is based on Bonferroni, to make sure that the returned features are categorized as correctly as possible. In the end, this leads to the creation of a good intruder detection system. Furthermore, they demonstrate how well the suggested method works by utilising the standard CICIDS 2017 and ADFA-LD data sets, as well as an intrusion monitoring system.

In (Ren et al., 2023), the author stated that MAFSIDS, which was short for Multi-Agent Feature Selection Intrusion Detection System, was what the writers termed it. This strategy's criterion is deep Q-learning. Both parts of MAFSIDS use Deep Reinforcement Learning (DRL): a model capable of detecting attacks, and a feature self-selection technique. Humanization: The method of feature self-selection is a new type of generation of feature space with N agents' N representations. This task is handled using the multi-agent reinforcement learning framework method. The multi-agent reinforcement learning framework method already displays many features in a list format, simplifying the final decision-making process. This model makes the search method smart enough to select features while leaving a simple design, thus serving the purpose. They adopt the GCN (Graph Convolutional Network) method to identify the more specific features of the data. The feature selection process is expedited by utilizing criteria functions for the features. The detection DRL tool also uses the Minit-Batch method to encode the data. In a feasible situation, reinforcing learning serves this important purpose. These combinations thus increase the efficiency and accuracy of the search process. To save computational power, this module attempts to shrink the policy network that models how humans act under different circumstances. Then, using the CSE-CIC-IDS2018 and NSL-KDD data sets, the team performed Python simulations. The model was very accurate, with 96.8% and 99.1% success rates, and received F1 scores of 96.3% and 99.1%. The modified feature group, acting as a selection group, was able to remove approximately eighty percent of the rejected properties. They compared it with the most extensively studied and used machine learning models.

3 Materials and Methods

This suggested approach to feature selection is meant to help IDS improve IoV performance. In the past few years, from analyzing big data to using machine learning approaches, some experts have addressed these problems and successfully improved the system's operation. This study has applied the latter approach and decreased the total number of features in order to enhance the performance of IDS in IoV. Figure 1 depicts the building blocks of the suggested model. The following section will outline the proposed model's stages in detail.

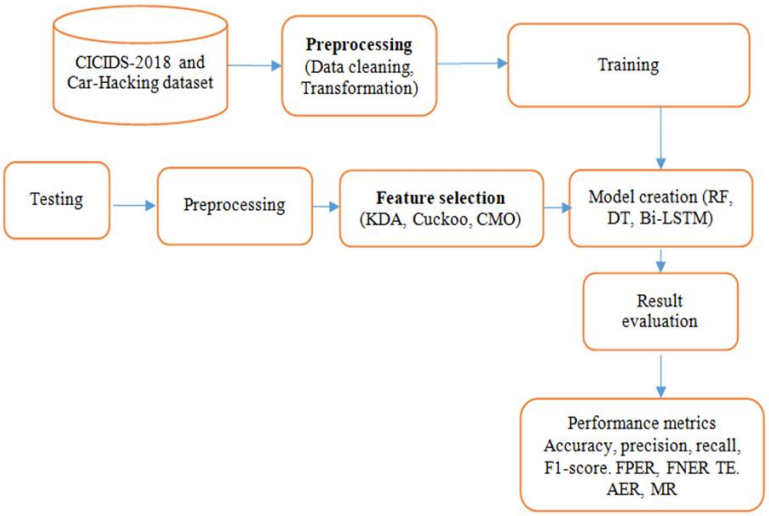

Figure 1: Overall Process of Intelligent Feature Selection with Optimization Algorithm

Dataset

The dataset plays a critical role in testing and monitoring an IDS's performance. Recent decades have seen the introduction of a large number of IDS and IoV datasets. Examples of such datasets are the KDDCup99 Dataset, UNSW-NB15, NSL-KDD, CICIDS-2018, CICIDS-2017, and Car-Hacking. A dataset often consists of many characteristics. We refer to these properties as classes and features. Most investigations that highlight insight on IDSs used KDDCup99 and NSL-KDD. This paper employed the CICIDS-2018 and Car-Hacking datasets since the other datasets did not match the study's criteria. This is due to fast advancements in network and vehicle security, as well as the necessity to satisfy operational needs. While the dataset has inherent weaknesses, it does not include typical modern-day traffic or novel attack tactics. The present study employed the CICIDS-2018 and Car-Hacking datasets in this paper. This study uses the CICIDS2018 benchmark dataset to reflect BENIGN and current network traffic attacks. This dataset includes protocols like email, HTTP, and HTTPS. In addition, it also contains up-to-date attacks on the network. The CICIDS2018 dataset puts the proposed framework to the test through DDoS attacks, web attacks, and infiltration. The other dataset is car hacking, the malicious manipulation of a vehicle's computer systems, which has become a growing concern in recent years. To develop effective defenses against these attacks, researchers rely on car-hacking datasets that simulate real-world scenarios and provide valuable insights into attacker techniques. Korea University developed this dataset, which collects data from a real vehicle during various attack types. It includes normal CAN bus traffic as well as four attack categories: flooding, spoofing, replay, and fuzzing. Each data point comprises a timestamp, CAN identifier, data length, and data bytes.

Preprocessing Stage

A typical preprocessing procedure for the CIDCIDS2018 dataset and the car hacking dataset entails various steps to ensure the data is clean, transformed, and ready for further analysis or modelling.

Data Cleaning: “Data cleansing," or its other name, is an unavoidable step in any data processing chain. Its implementation is done by detecting and correcting the errors, inconsistencies, and missing values in the dataset, which is the mandatory prep-step before using the dataset for any meaningful

analysis. Furthermore, duplicate entries may inflate the results and confuse the analysis. It does this by eliminating the possibility of inadvertent manipulation, which in turn facilitates the correctness of the representational data. Anecdotal techniques are quite omnipresent and include the following: loss (when data about a variable is missing), imputation (replacing a missing value with an assumed value), or encoding a missing value as a new characteristic.

Data Transformation: To fix the problem of convergence and performance, ensure the normalisation of numerical attributes that have a standard deviation of one with a zero mean. For skewness in the data, use the log transformation of the Box-Cox transformation for a more symmetrical distribution. In this paper, the Box-Cox transformation is employed for the CICIDS 2018 and car hacking datasets. The Box-Cox transformation is a statistical method that is applied to stabilize the variance and make data absolutely normally distributed. It is particularly useful when dealing with nonlinearity influence, heteroscedasticity (unambiguous variance ranges between values), or non-normality. The transformation is defined by a parameter, usually denoted as λ, which determines the type of transformation applied to the data. The Box-Cox transformation is defined as: The transformation is defined by a parameter, usually denoted as λ, which determines the type of transformation applied to the data. The Box-Cox transformation is defined in equation (1)**:**

$$y(\lambda) = \begin{matrix} \frac{y^2-1}{\lambda}, if\ \lambda \neq 0 \\ \log(y), if\ \lambda = 0 \end{matrix} \qquad (1)$$

Where: $y(\lambda)$ is the transformed data of original data y. The transformation parameter () λ can take any real value. When

λ=1, the transformation is equivalent to the natural logarithm. When

λ=0, the transformation is equivalent to a logarithmic transformation with base 10.

Maximum likelihood estimation or other optimisation techniques are often used to determine the optimal value of λ so as to achieve the best normalisation of the data.

Data Splitting: In this, we split the dataset into two parts. Around 70–80% for training and 30–20% for testing are typical values for splitting datasets. It is done to evaluate the model's performance.

Feature Selection Stage

This is crucial in intrusion detection to reduce computational costs and improve model performance. Traditional methods struggle with incomplete data and redundant features, leading to suboptimal results. A range of general network attributes are typically present in intrusion detection datasets. Identify relevant features: Determine which features are relevant for the analysis or modelling task. This may involve domain knowledge, exploratory data analysis, or feature selection techniques. Remove irrelevant features. Remove features that are redundant, irrelevant, or have low variance. This can help reduce dimensionality and improve model performance. Four subsets were extracted based on the KLDA, CSA, and CMO algorithms through the proposed model.

Kernel Linear Discriminant Analysis (KLDA)

The KLDA follows two primary techniques. First, observations are mapped into a higher-dimensional feature space $\left(\varphi: R^n \to X, y \to (y)\right)$ using an RBF kernel (formula described in the GRBF section). We then apply the normal LDA to the new space to determine the optimal dimension. The approach begins with centering the features $\left(\sum_{i=1}^{n} y_j = 0\right)$. The LDA is fine-tuned by the vector that maximises $J(v)$,

where $J(v) = \frac{|V^T F_b V|}{|V^T F_b V|}$.. In equations (2) and (3), F_b represents within-group scattering, while F_w represents between-group scattering matrices. The variable V represents the transformation matrix calculated by $F_b v_f = \gamma_j F_w v_j$, where γ is the eigenvalue (Chang & Lin, 2011).

$$F_b = \sum_{j=1}^{k} \frac{1}{K_j} \sum_{l,m=1}^{n_j} \left(O_l^{(j)}\right)\left(O_m^j\right)^T \quad (2)$$

$$F_w = \sum_{k=1}^{k} \sum_{;=1}^{n_j} \left(O_l^{(j)} - \frac{1}{K_j} \sum_{l,m=1}^{n_j} \left(O_l^{(j)}\right)\left(O_l^j\right) - \frac{1}{K_j} \sum_{l,m=1}^{n_j} \left(O_l^j\right)^T \right) \quad (3)$$

where K is the number of trials, $O_l^{(j)}$ is the feature of the lth trial from the jth class, and T is the transpose operator. Higher-dimensional feature spaces are more efficient at separability. Therefore, we classify classes using an RF m DT and Bi-LSTM classifiers.

Cuckoo Search Algorithm

We selected CSA to narrow down the most relevant nonlinear parameters and uncover anomaly observations (Sarvari et al., 2020). This paper will dwell on a new CSA-based technique that aspires to deal with nonlinear functions and personify discrimination between valid and unacceptable acts, together with the classification methods. To imitate the brood parasitism of non-obligate cuckoos, the CSA combines host egg recognition with host egg acceptance. In some cases, the cuckoos realize that the eggs laid and hatched in the nests are not of their ovary, while when the novelty is revealed, the eggs drop to the ground or the whole nest is deserted. Three rules are the basics of the CS algorithm:

(1) Every cuckoo deposit their eggs singly, in other cuckoo's nests.
(2) The eggs in nests of the best quality will be mostly used to generate new generations of eagles.
(3) There are a fixed number of available host's nests; therefore, the cuckoos have pa probability that could vary from 0 to 1 alternatively.

CSA might turn global random walk and local random walk into a moderate search path, using which CSA would be able to explore larger features of the information space. From a mathematical point of view, the global multi-dimensional random walk is represented by a system of equations.

$$x_i^{t+1} = x_i^t + a\ell(s,\lambda) \quad (4)$$

where, x_i^t the current position of the i^{th} cuckoo while t being current iteration, x_i^{t+1} is the next position and s is the step size, L (s, λ) is the levy distributions used to determine the size of the step of random walk, and a is a positive scaling factor. The local random walk is defined as follows:

$$x_i^{t+1} = x_i^t + as \otimes H(p_t - \varepsilon) \otimes \left(x_i^t - x_k^t\right) \quad (5)$$

where, H is a heavy-side function, N indicates the entry-wise multiplication operator, ⊗ is a random number generated based on the normal distribution, and x_i^t and x_k^t are two random positions chosen randomly from the current population. t_{max} indicates the maximum number of iterations. The steps of CSA are shown in Algorithm1.

Algorithm 1

1. Create an initial population of N solutions.
2. Initialize $a, pa,$ and $t = 0$;
3. Calculate the fitness for each solution .
4. while $(t < tmax)$

5. *C*reate a new population using Equation (4) and insert better ones into the current population.
6. $t = t + 1$;
7. Create a new population using Equation (5) and add into the current the best ones.
8. $t = t + 1$;
9. end while

Objective Function

The objective function for CMOA in feature selection can be formulated as minimizing the number of selected features, to maximize the detection accuracy.

Let $f(X)$ be the objective function, where X represents the selected features.

Detection Accuracy−×Number of Features Selected

$f(X)$ =Detection Accuracy – λ × Number of Features Selected

Where λ is a weight parameter to balance between accuracy and the number of features.

This section explains the theory and mathematical model of CMO, which may be used to optimize different issues. Inspired by the natural behaviours of a cat chasing a mouse and the mouse escaping to the refuge, a population-based algorithm called the CMO was introduced. The suggested method divides the search agents into two groups: cats and mice, who wander randomly around the issue search space. There are two stages to the proposed approach for updating members of the population. In the first stage, we see cats and mice interacting, and in the second, we see mice running away to safety. In mathematical terms, every individual in the population stands in for a potential solution to the problem. As a result, the variables in the problem are defined by the values of the vectors that comprise the population. Equation (6) determines the algorithm's population using a matrix known as the population matrix.

$$\alpha = \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_j \\ \vdots \\ \alpha_P \end{bmatrix}_{P\times q} = \begin{bmatrix} \alpha_{1,1} & \cdots & \alpha_{1,d} & \cdots & \alpha_{1,1} \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ \alpha_{j,1} & \cdots & \alpha_{j,d} & \cdots & \alpha_{j,q} \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ \alpha_{P\times q} & \cdots & \alpha_{P,d} & \cdots & \alpha_{P,q} \end{bmatrix}_{P\times q} \quad (6)$$

The variable α represents the matrix of the population, α_j stands for the *i*th search agent, $\alpha_{j,d}$ for the value of the *d*th problem variable acquired by the *i*th search agent, P stands for the "population members", and q for the "number of problem variables". It can be added that each of the community members decides the values for the proposed variables to solve the problem. As a result, each species member's object function value is defined. The vector in Equation (7) shows the target function values.

$$F = \begin{bmatrix} F_1 \\ \vdots \\ F_i \\ \vdots \\ F_N \end{bmatrix}_{N\times 1} \quad (7)$$

Where, and F_i is the "objective function" and F is function vector.

Comparing the obtained functions' values iteratively ranks the original population from best to worst, with the population member having the lowest objective function value. Equations (8) and (9) state that the same technique determines both the "sorted objective function" and the "population matrix".

$$X^S = \begin{bmatrix} X_1^S \\ \vdots \\ X_i^S \\ \vdots \\ X_N^S \end{bmatrix}_{N\times m} = \begin{bmatrix} X_{1,1}^S & \cdots & X_{1,d}^S & \cdots & X_1^S \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ X_{i,1}^S & \cdots & X_{i,d}^S & \cdots & X_{i,m}^S \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ X_{N,1}^S & \cdots & X_{M,d}^S & \cdots & X_{N,m}^S \end{bmatrix}_{N\times m} \quad (8)$$

$$F^S = \begin{bmatrix} F_1^S & \min(F) \\ \vdots & \vdots \\ F_N^S & \max(F) \end{bmatrix}_{N\times 1} \quad (9)$$

The objective function's sorted vector, X_i^S, is the *i*th member of the "sorted population matrix", X^S, based on the objective function value, and $X_{i,d}^S$, the *d*th deliberate variable.

The CMO setup creates two groups of cats and mice in the population matrix. It is proposed in CMO that essentially fifty percent of the population members that were maximizing the objective function are mice, whereas the remaining half of the population members that were minimizing the objective function are cats. Equations (10) and (11) respectively evaluate the mice and cat populations.

$$M = \begin{bmatrix} M_1 = X_1^S \\ \vdots \\ M_1 = X_i^S \\ \vdots \\ M_{N_M} = X_{N_m}^S \end{bmatrix}_{N_m\times m} = \begin{bmatrix} X_{1,1}^S & \cdots & X_{1,d}^S & \cdots & X_{1,m}^S \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ X_{i,1}^S & \cdots & X_{i,d}^S & \cdots & X_{i,m}^S \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ X_{N_m,1}^S & \cdots & X_{N_m,d}^S & \cdots & X_{N_m,m}^S \end{bmatrix}_{N_m\times m} \quad (10)$$

$$C = \begin{bmatrix} C_1 = X_{N_m+1}^S \\ \vdots \\ C_j = X_{N_m+j}^S \\ \vdots \\ C_{N_c} = X_{N_m+N_c}^S \end{bmatrix}_{N_c\times m} = \begin{bmatrix} X_{N_m+1,1}^S & \cdots & X_{N_m+1,d}^S & \cdots & X_{N_m+1,m}^S \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ X_{N_m+j,1}^S & \cdots & X_{N_m+j,d}^S & \cdots & X_{N_m+j,m}^S \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ X_{N_m+,1}^S & \cdots & X_{N_m+N_c,d}^S & \cdots & X_{N_m+N_c,m}^S \end{bmatrix}_{N_c\times m} \quad (11)$$

Where N_m is the number of mice, the population matrix of mice is represented by *M*, and the individual mice are referred to as M_i. The cat population is represented by the symbol *C*, the number of cats is represented by the symbol N_c, and the associated cats are C_i. To perform a changeover, in the initial phase, the movement of the cat is reproduced; as such, these represent felines and the actions of the mouse. This is the form in which the mathematical modelling update of the proposed CMO has been made to use Equations (12)–(14) to perform such tasks.

$$\alpha_j^{new}: \alpha_{j,d}^{new} = \alpha_{j,d} + r \times \left(q_{k,d} - I \times \ \alpha_{j,d}\right) \& \ j = 1{:}N_c, d = 1{:}m, k \in 1{:}N_m \quad (12)$$

$$I = round(1 + rand) \quad (13)$$

$$\alpha_j = \begin{cases} \alpha_j^{new}, |F_j^{\alpha,new} < F_j^{\alpha} \\ \alpha_j, |else \end{cases} \quad (14)$$

The objective function value $F_j^{\alpha,new}$ is determined by the new status of the *j*th cat, while the solution variable for the *k*th mouse is $q_{k,d}$ over a random number generation interval [0, 1].

The escape of mice from havens happens in the second phase. In CMO, it is proposed that the mice's escape from havens (the given haven, safe area, or refuge) is random, and they ran towards the given refuge whether there were mice there or not. The algorithm's randomness according to the patterning characters of its members creates a haven in the search space. Only this stage's reevaluation of mice's angular velocity is creatively expressed by Equations (15)–(17).

$$H_i: h_{i,d} = x_{l,d} \& \ i = 1{:}N_m, d = 1{:}m. \ l \in 1{:}N. \quad (15)$$

$$M_i^{new}: m_{i,d}^{new} = h_{j,d} + r \times \left(q_{k,d} - I \times \ \alpha_{j,d}\right) \times sign\left(F_i^{m,new} - F_i^m\right)\& \quad (16)$$
$$j = 1:N_c, d = 1:m,$$

$$M_i = \begin{cases} M_i^{new}, |F_i^{m,new} < F_i^m \\ M_i, |else \end{cases}, \quad (17)$$

In the instance above, the denotation H_i is the haven of the *i*th mouse and F_i^m is the value of its function to be maximized. M_i^{new} denotes the new state of mouse *i*, and $F_i^{m,new}$ is its objective function value. Bringing the algorithm to the next iteration is done by updating those steps that were performed over all individuals' populations. And until the stop criterion is met, iterations of the algorithm will continue. Conditions can be the number of iterations, or by accurately specifying the error that will satisfy as a solution in different loops for stopping optimization algorithms, *c*. In addition, stopping in conducting the algorithm may take a definite period of time. By implementing the iterations consecutively and then the subsequent application of the algorithm to the demands of the optimisation problem, CMO supplies an optimal quasi-solution. The procedure will be explained with the pseudocode that is present in Algorithm 2.

Algorithm 2 Pseudo code of CMO

Start CMO.
Input:
− Variables: X
− Objective function: $f(X)$
− Constraints: g(X) <= 0
− Number of search agents (N)
− Number of iterations (T)
Generate an initial population matrix at random.
Evaluate the objective function.
for t = 1 to T:
Sort population matrix based on objective function value.
Select population of mice (M).
Select population of cats (C).
Phase 1: Update status of cats.
for j = 1 to Nc:
Update status of the jth cat.
Phase 2: Update status of mice.
for i = 1 to Nm:
Create haven for the ith mouse.
Update status of the ith mouse.
Output: Best quasi − optimal solution obtained with the CMO.
end CMO

Classification

Classification methods are used to classify the category of object samples from the set of feature vectors. Machine learning works on building models that can adapt to past labelled dataset experiences and be utilized to anticipate the same for unforeseen data points. The classifier module, as such, ensures that normal and abnormal data are classified into the incoming data flow. Here, DT, RF, and Bi-LSTM learners are applied. Artificial intelligence and machine learning methods include information

classification. As a consequence, the output of the feature set after the optimisation algorithms that the classifier is based on is used as the input for the classifier.

Decision Trees

The decision tree is the simplest classifier method with the minimum concept, and it is easy to grasp and view. They run their classification process with the "if, then" rules based on the pattern of feature values from the sample training set. The leaves of this tree are then used for each class label. The branch leads home; for every node, only one feature is selected; no itinerary has two features in common across all routes. There might be another component of the decision tree that would let us also know the level of confidence in the classification quality. One of the most prominent advantages of the classifier is that it constructs its tree recursively (Charbuty & Abdulazeez, 2021). It is a repetitive process based on data partitioning in which the set of data is divided randomly into partitions, which are split on every branch afterward. Data can be broken down with statistical methods such as information gain or the Gini index to choose the features that will be taken at each node. The last feature becomes a leaf when it is completely used up and there is a mixture of more than one class in the rest of the samples, then the class with the majority wins the game.

Random Forest (RF)

The Random Forest (RF) approach (Breiman, 2001) was created to address a drawback of decision trees: the propensity to overfit the sample data. RF solves this issue by using a statistical approach known as bootstrapping, which creates numerous models and integrates their findings to get a final choice. RF's primary principle is to reduce the impact of individual mistakes by combining predictions from numerous classifiers. During bootstrapping, numerous smaller samples are randomly taken from the training dataset and replaced. We use each sample to train an individual classifier. Thus, when classifying a new observation, we calculate the final class prediction by aggregating the data from the separate models. Most of the time, we use a majority voting approach.

Bi-LSTM

A LSTM bidirectional neural network is employed to deduce long-term relationships along with the setting of the data. Unlike the vanilla LSTM, which processes data in the forward direction only, a bi-LSTM will move the sequential data in both forward and backward directions, which will increase the comprehension of temporal relationships. The training network feeds the selected features to the Bi-LSTM for training. While the network learns to detect relationships and patterns between features and a desired product, the Bi-LSTM has the ability to classify new sequences after the training stage by using the extracted features and learned patterns shows in Figure 2.

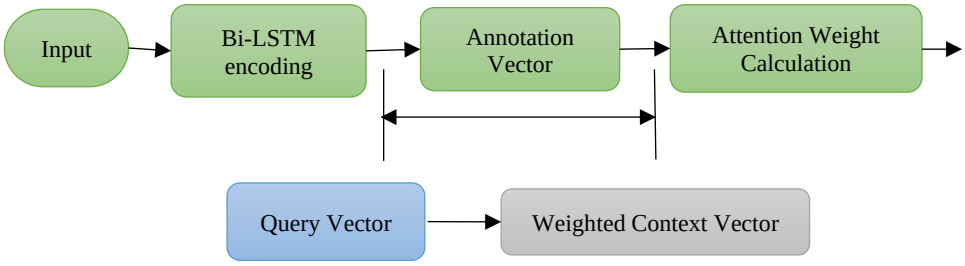

Figure 2: Proposed Model Block Diagram for Intrusion Detection in IoV

Step 1: To capture contextual information from both past and future elements, pass the input sequence through a bi-LSTM network. For each time step, generate an annotation vector (usually denser than the hidden state) that captures the relevant information for attention.

Step 2: Define a query vector that represents the focus of attention at each time step. This can be the hidden state of the Bi-LSTM at a specific step or a different learned vector. Compute the dot product between the query vector and each annotation vector to assess their relevance. Apply softmax activation to the dot product results, normalizing them into probability weights between 0 and 1. These weights represent the "attention scores," indicating how much each element in the sequence contributes to the final output.

Step 3: For each time step, multiply each element in the input sequence by its corresponding attention score and sum them up. This creates a weighted context vector that summarizes the most relevant information from the entire sequence based on the query vector's focus.

Step 4: Concatenate the weighted context vector with the hidden state of the Bi-LSTM at the current time step to generate a more comprehensive representation of the sequence. Additional layers, such as fully-connected layers, can receive the combined representation for classification.

The first LSTM computational layer is bidirectional (bi-LSTM), with H being the number of hidden units and inputs of L units. The function expresses this in Equation (18). A bi-directional LSTM network, together with an LSTM layer, keeps two values as its hidden layer. A solves in the forward direction, while B solves in the reverse direction. The final result is y=Axe, which is our product matrix A, together with A transpose.

$$a_h^t = \sum_{l=1}^{L} x_l^t w_{lh} + \sum_{h^t, t>0}^{H} b_{h^t}^{t-1} w_{h^t h} \quad (18)$$

$$a_h^t = \theta_h(a_h^t)$$

4 Results and Discussion

The Intel Core i5 8th generation laptop was the tool of choice to develop the proposed scheme. Python 3 simulation environments were used to perform experiments to examine the efficiency of the presented algorithm. We reported the main performance indicators, which are recall, precision, accuracy (ACC), and F1-score. For monitoring and optimisation of model performance, multiple types of performance metrics should be chosen, as some are relative to particular models, which makes it vital to select an ideal one. The class that usually holds the majority of a real system's processes cannot be seen and is often misunderstood. The other side of the story is that most of the real system processes cannot be observed and are rather hard to interpret, but logs, metrics, and telemetry are important tools for monitoring system performance. The TP, TN, FP, and FN indicators are determined by evaluating the measures.

In Equation (19) The false positive error rate (FPER), also known as the Type I error rate, is a statistical concept primarily used in hypothesis testing and classification tasks. It represents the probability of incorrectly rejecting a true null hypothesis or incorrectly identifying a negative instance as positive.

$$\text{FPER} = \frac{\text{FP}}{\text{FP} + \text{TN}} \quad (19)$$

A Statistical concept false negative error rate, also known as Type II error rate, is used in hypothesis testing and classification tasks. It represents the probability of incorrectly failing to reject a false null hypothesis or incorrectly identifying a positive instance as negative in equation (20).

$$\text{FNER} = \frac{\text{FN}}{\text{FN} + \text{TP}} \tag{20}$$

Total error, in the context of classification tasks, refers to the overall rate of incorrect predictions made by a model. It measures of the overall accuracy and it encompasses both false negatives and false positives in equation (21).

$$\text{TE} = \text{FP} + \text{FN} \tag{21}$$

The average error rate is not a standard term in statistics or machine learning. However, it could refer to the mean error rate across multiple models or multiple iterations of a single model in equation (22).

$$\text{AER} = \frac{\text{FPER} + \text{FNER}}{2} \tag{22}$$

The misclassification rate, also known as the error rate, represents from total number of instances, the proportion of incorrectly classified instances in equation (23)-(27).

$$\text{MR} = \frac{\text{FP} + \text{FN}}{\text{FP} + \text{TP} + \text{FN} + \text{TN}} \tag{23}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{24}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{25}$$

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FN} + \text{TN} + \text{FP}} \tag{26}$$

$$\text{F1} - Score = 2 \times \frac{\text{Precision} + \text{Recall}}{\text{Precision} \times \text{Recall}} \tag{27}$$

Performance Indices Comparison of IDS in IoV

In Tables 1 and 2, we compare the proposed architecture with other methods against different attacks obtained from the CICIDS-2018 and car hacking datasets. Our architecture achieves a higher performance value than other methods, though the difference is less. We have bolded the highest values for each performance criterion.

Table 1: Results Reached based on CICIDS-2018 Dataset

Method	Accuracy	Precision	Recall	F1-Score
KLDA-RF	0.9552	0.9052	0.9852	0.9752
KLDA-DT	0.9852	0.9132	0.9653	0.9357
KLDA-Bi-LSTM	0.9878	0.9201	0.9702	0.9808
Cuckoo - RF	0.9895	0.9899	0.9803	0.9818
Cuckoo - DT	0.9899	0.9898	0.9878	0.9854
Cuckoo - Bi-LSTM	0.9899	0.9895	0.9895	0.9895
CMO- RF	0.9900	0.9898	0.9899	0.9898
CMO- DT	0.9900	0.9889	0.9899	0.9899
CMO- Bi-LSTM	**0.9910**	**0.9912**	**0.9900**	**0.9905**

With other techniques achieving higher precision or recall in the case of a DoS attack, it must be noted that the accuracy and F1 ratings reviewed are still lower than our proposed model. The IDS that are being proposed performs better with fuzzy attacks than the other strategies that have been stated

earlier. To wrap it up, it is not an effortless mission to make comparisons of existing methodologies, with the suggested IDS outshining the others by ensuring a better score in the quantitative comparison. The experimental results demonstrate that the suggested IDS can indeed differentiate between data that is legitimate and that is malicious for the network CAN bus shows in Figure 3 and 4.

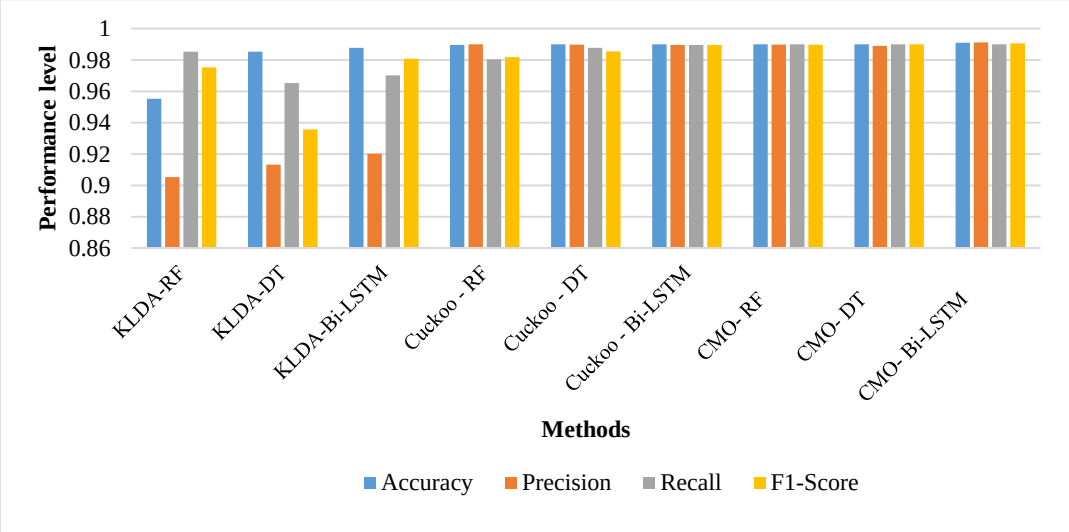

Figure 3: Performance Comparison of Different Feature Selection Techniques on the CICIDS-2018 Dataset

Table 2: Results Reached based on Car-Hacking Dataset

Method	Accuracy	Precision	Recall	F1-Score
KLDA-RF	0.9752	0.9754	0.9747	0.9750
KLDA-DT	0.9789	0.9801	0.9789	0.9792
KLDA-Bi-LSTM	0.9801	0.9807	0.9799	0.9500
Cuckoo - RF	0.9823	0.9844	0.9844	0.9823
Cuckoo - DT	0.9853	0.9878	0.9865	0.9865
Cuckoo - Bi-LSTM	0.9899	0.9895	0.9895	0.9895
CMO- RF	0.9899	0.9877	0.9878	0.9835
CMO- DT	0.9865	0.9869	0.9899	0.9899
CMO- Bi-LSTM	**0.9920**	**0.9924**	**0.9901**	**0.9905**

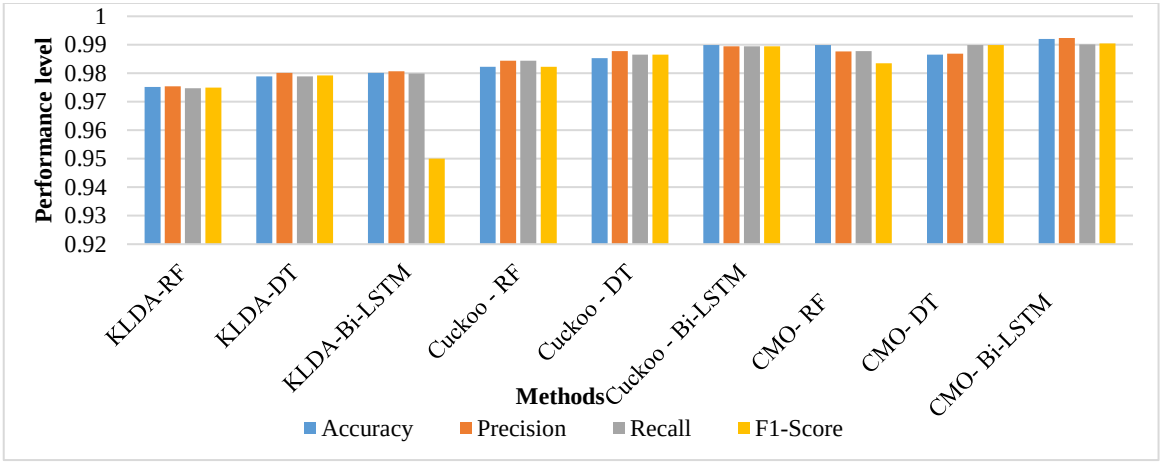

Figure 4: Performance Comparison of Different Feature Selection Techniques on the Car Hacking Dataset

Testing Response Time Analysis

We use the car hacking dataset to evaluate the model's response time. For evaluating the model from the dataset, 20% of the pre-processed data was used. The time-testing technique was applied to three models: RF, DT, and Bi-LSTM. The dataset had 1,001,720 instances that were split into 31,304 batches of size 32, which we used to train the CMO-Bi-LSTM model. This entire testing operation lasted 48.98 seconds and can be seen in Figure 5. We executed each batch in 1.53 milliseconds. The number of instances in one batch was 32. KLDA-RF gives a low (85.33 s) test time response. As we can see from the response time, the suggested model (CMO-Bi-LSTM) is faster than others. Also, for the CICIDS-2018 dataset, the testing time is 38.39 seconds, and the most sophisticated model (CMO-Bi-LSTM) will be utilised.

This work is to be the intrusion detection system, mainly to attenuate both the false positive rate and the false negative rate. However, the primary focus of this project is the False Negative (FN) parameter. As shown on tables 3 and 4 below, crabs were sold more during the day than in the evening. The confusion matrix visualizes FP and FN parameters to draw the conclusion that the binary classifier gives the highest FN value of 0.093. Thus, we get the majority of attacks classified as normal packets. On the other side, the decision table classifier is characterized by the lowest number of false negatives at 0.002. Based on this evidence, the final decision tree classifier reported the biggest number of FP packets with a rate of 0.073, which corresponds to an overwhelming number of innocent packets identified as malicious. In the case of MR, the considered person must notice the various fraction errors produced due to his incorrect prediction of the condition. The MR may be defined as the merging of all prediction errors corresponding to the number of instances by its count. This study explores the accuracy of a MR of nine methods applied to data conducted on the CICIDS-2018 and car hacking datasets in both datasets. The next experiment would focus on the trait of MR performance across solely different feature selection processes versus two sets of data. CMO-Bi-LSTM returns an MR value of 0.036 against the CICIDS-2018 dataset.

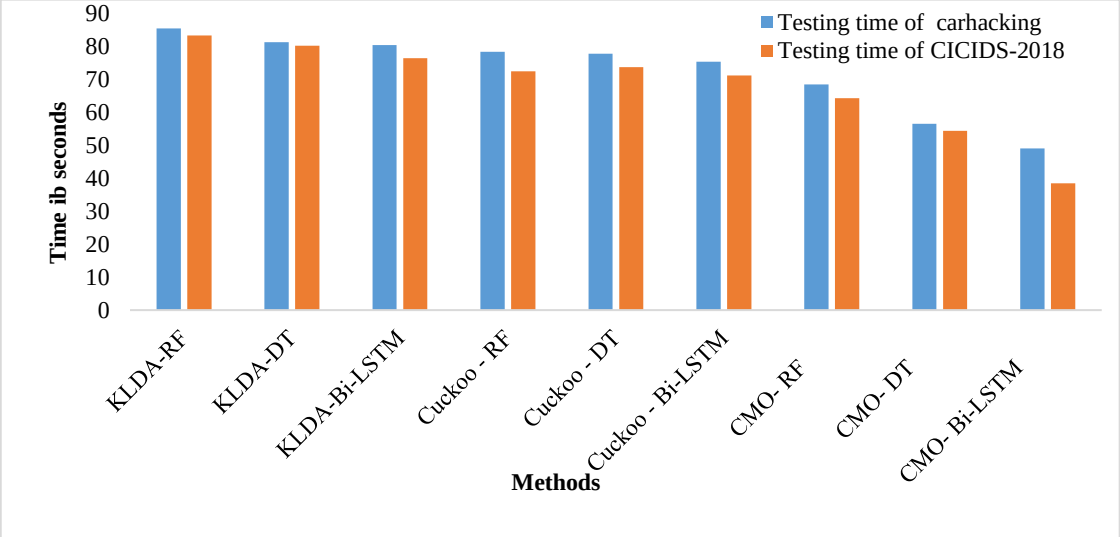

Figure 5: Testing Response Time on the Car Hacking and CICIDS-2018 Datasets

In contrast, KLDA-RF has an MR value of 0.0223 for the same dataset. The accuracy of SVM shows a high value of 0.984, but it gradually degrades to 0.969 when it comes to the CMO-Bi-LSTM algorithm against the car hacking dataset. This is well-shown in the performance of the CMO-Bi-LSTM in the latter case, as it is better compared with the other methods.

Table 3: Performance Evaluation of Error Rates on the CICIDS-2018 Dataset

Method	FPER	FNER	TE	AER	MR
KLDA-RF	0.0036	0.0055	185	0.0038	0.0223
KLDA-DT	0.0032	0.0051	172	0.0043	0.0081
KLDA-Bi-LSTM	0.0030	0.0048	160	0.0052	0.0072
Cuckoo - RF	0.0028	0.0043	143	0.0058	0.0069
Cuckoo - DT	0.0026	0.0041	120	0.0062	0.0062
Cuckoo - Bi-LSTM	0.0025	0.0038	110	0.0068	0.0058
CMO- RF	0.0021	0.0032	98	0.0072	0.0051
CMO- DT	0.0018	0.0021	88	0.0082	0.0048
CMO- Bi-LSTM	0.0008	0.0013	52	0.0127	0.0036

Table 4: Performance Evaluation of Error Rates on the Car Hacking Dataset

Method	FPER	FNER	TE	AER	MR
KLDA-RF	0.0046	0.0065	224	0.0041	0.0058
KLDA-DT	0.0042	0.0061	198	0.0037	0.0081
KLDA-Bi-LSTM	0.0038	0.0058	178	0.0032	0.0072
Cuckoo - RF	0.0032	0.0047	135	0.0027	0.0069
Cuckoo - DT	0.0022	0.0039	117	0.0062	0.0062
Cuckoo - Bi-LSTM	0.0018	0.0028	105	0.0068	0.0058
CMO- RF	0.0016	0.0026	98	0.0072	0.0051
CMO- DT	0.0012	0.0018	21	0.0082	0.0048
CMO- Bi-LSTM	0.0005	0.0007	3	0.0127	0.0003

5 Conclusion

The study proposes a robust intrusion detection system based on feature selection, specifically tailored for the dynamic environment of the Internet of Vehicles. The proposed system uses optimisation techniques to efficiently identify and prioritize relevant features, improving detection accuracy and lowering computing costs. The strategy's effectiveness in identifying intrusions was confirmed after extensive testing, ensuring reliable detection and minimal false alarms. This work offers a new optimizer called Cat and Mouse Optimizer (CMO), which mimics the natural behaviour of cats and mice. The CMO proposes a mathematical model based on cats attacking mice and mice fleeing to shelters. The suggested technique was thoroughly tested using the CSE-CIC-IDS 2018 and Car hacking datasets. An examination of the proposed model demonstrates that the combined DDoS dataset and automobile hacking dataset have an accuracy rate of 99.5% and 99.9%, respectively. The future of connected vehicles necessitates the development of scalable intrusion detection system (IDS) solutions. Optimisation methods can guarantee that the system expands efficiently without sacrificing performance. Putting feature-selection-based intrusion detection systems (IDS) on edge devices in vehicles lets them find intrusions in real time, which cuts down on latency and speeds up reaction times.

References

[1] Al Tawil, A., & Sabri, K. E. (2021). A feature selection algorithm for intrusion detection system based on moth flame optimization. *In IEEE International Conference on Information Technology (ICIT)*, 377-381. https://doi.org/10.1109/ICIT52682.2021.9491690

[2] Alamer, L., & Shadadi, E. (2023). DDoS Attack Detection using Long-short Term Memory with Bacterial Colony Optimization on IoT Environment. *Journal of Internet Services and Information Security, 13*(1), 44-53.
[3] AlGhamdi, R. (2023). Design of Network Intrusion Detection System Using Lion Optimization-Based Feature Selection with Deep Learning Model. *Mathematics*, *11*(22), 4607. https://doi.org/10.3390/math11224607
[4] Ali, I., Chen, Y., Li, J., Wakeel, A., Pan, C., & Ullah, N. (2023). Efficient Offline/Online Heterogeneous-Aggregated Signcryption Protocol for Edge Computing-Based Internet of Vehicles. *IEEE Transactions on Intelligent Transportation Systems*. https://doi.org/10.1109/TITS.2023.3296974
[5] Al-Tashi, Q., Kadir, S. J. A., Rais, H. M., Mirjalili, S., & Alhussian, H. (2019). Binary optimization using hybrid grey wolf optimization for feature selection. *IEEE Access*, *7*, 39496-39508. https://doi.org/10.1109/ACCESS.2019.2907069
[6] Al-Tashi, Q., Md Rais, H., Abdulkadir, S. J., Mirjalili, S., & Alhussian, H. (2020). A review of grey wolf optimizer-based feature selection methods for classification. *Evolutionary machine learning techniques: algorithms and applications*, 273-286.
[7] Asl, T. M., & Asl, T. S. (2022). Strategy Optimization for Responding to Primary, Secondary and Residual Risks Considering Cost and Time Dimensions in Petrochemical Projects. *Archives for Technical Sciences, 2*(27), 33-48.
[8] Aswathy, R. H., Srithar, S., Roslin Dayana, K., Padmavathi., & Suresh, P. (2023). MIAS: An IoT based Multiphase Identity Authentication Server for Enabling Secure Communication. *Journal of Internet Services and Information Security, 13*(3), 114-126.
[9] Babenko, V., Danilov, A., Vasenin, D., & Krysanov, V. (2021). Parametric Optimization of the Structure of Controlled High-voltage Capacitor Batteries. *Archives for Technical Sciences, 1*(24), 9–16.
[10] Balasaraswathi, V. R., Mary Shamala, L., Hamid, Y., Pachhaiammal Alias Priya, M., Shobana, M., & Sugumaran, M. (2022). An efficient feature selection for intrusion detection system using B-HKNN and C2 search based learning model. *Neural Processing Letters*, *54*(6), 5143-5167. https://doi.org/10.1007/s11063-022-10854-1
[11] Breiman, L. (2001). Random forests. *Machine Learning, 45*, 5–32. https://doi.org/10.1023/A:1010933404324
[12] Camgözlü, Y., & Kutlu, Y. (2023). Leaf Image Classification Based on Pre-trained Convolutional Neural Network Models. *Natural and Engineering Sciences, 8*(3), 214-232.
[13] Chang, C. C., & Lin, C. J. (2011). LIBSVM: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, *2*(3), 1-27. https://doi.org/10.1145/1961189.1961199
[14] Charbuty, B., & Abdulazeez, A. (2021). Classification based on decision tree algorithm for machine learning. *Journal of Applied Science and Technology Trends*, *2*(01), 20-28.
[15] Emary, E., Zawbaa, H. M., & Hassanien, A. E. (2016). Binary grey wolf optimization approaches for feature selection. *Neurocomputing*, *172*, 371-381. https://doi.org/10.1016/j.neucom.2015.07.085
[16] Escorcia-Gutierrez, J., Gamarra, M., Leal, E., Madera, N., Soto, C., Mansour, R. F., & Gupta, D. (2023). Sea turtle foraging algorithm with hybrid deep learning-based intrusion detection for the internet of drones environment. *Computers and Electrical Engineering*, *108*, 108704. https://doi.org/10.1016/j.compeleceng.2023.108704
[17] Fung, C. J. (2011). Collaborative Intrusion Detection Networks and Insider Attacks. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications, 2*(1), 63-74.
[18] Gharkan, D., & Abdulrahman, A. (2023). Construct an efficient distributed denial of service attack detection system based on data mining techniques. *Indonesian Journal of Electrical Engineering and Computer Science, 29*, 591. https://doi.org/10.11591/ijeecs.v29.i1.pp591-597

[19] Gou, W., Zhang, H., & Zhang, R. (2023). Multi-Classification and Tree-Based Ensemble Network for the Intrusion Detection System in the Internet of Vehicles. *Sensors, 23*, 8788. https://doi.org/10.3390/s23218788

[20] Hussein, N. K., Qaraad, M., Amjad, S., Farag, M. A., Hassan, S., Mirjalili, S., & Elhosseini, M. A. (2023). Enhancing feature selection with GMSMFO: A global optimization algorithm for machine learning with application to intrusion detection. *Journal of Computational Design and Engineering*, *10*(4), 1363-1389. https://doi.org/10.1093/jcde/qwad053

[21] Jegan, M. S., & Balasubramanian, P. (2022). Influencing Features and Factors of Reference Management Software among University Research Scholars in Tamil Nadu, India. *Indian Journal of Information Sources and Services, 12*(2), 1–9.

[22] Liu, H., & Motoda, H. (2012). *Feature selection for knowledge discovery and data mining, 454.* Springer science & business media.

[23] Mohammed, S. A. (2019). Designing rules to implement reconnaissance and unauthorized access attacks for intrusion detection system. *Iraqi Journal of Information and Communication Technology*, *2*(2), 25-43. https://doi.org/10.31987/ijict.2.2.67

[24] Nobles, P., Ali, S., & Chivers, H. (2011). Improved Estimation of Trilateration Distances for Indoor Wireless Intrusion Detection. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications, 2*(1), 93-102.

[25] Padmanabhan, B., & Premalatha, L. (2019). Adaptive bacterial foraging algorithm to green energy supported non smooth dynamic power dispatch optimization. *Journal of Green Engineering, 9*(3), 402-410.

[26] Priyanka, J., Ramya, M., & Alagappan, M. (2023). IoT Integrated Accelerometer Design and Simulation for Smart Helmets. *Indian Journal of Information Sources and Services, 13*(2), 64–67.

[27] Ren, K., Zeng, Y., Zhong, Y., Sheng, B., & Zhang, Y. (2023). MAFSIDS: a reinforcement learning-based intrusion detection model for multi-agent feature selection networks. *Journal of Big Data*, *10*(1), 137. https://doi.org/10.1186/s40537-023-00814-4

[28] Sahoo, A., & Chandra, S. (2017). Multi-objective grey wolf optimizer for improved cervix lesion classification. *Applied Soft Computing, 52*, 64-80. https://doi.org/10.1016/j.asoc.2016.12.039

[29] Sarvari, S., Sani, N. F. M., Hanapi, Z. M., & Abdullah, M. T. (2020). An efficient anomaly intrusion detection method with feature selection and evolutionary neural network. *IEEE Access, 8*, 70651-70663. https://doi.org/10.1109/ACCESS.2020.2982283

[30] Sinha, P. (2013). Particle swarm optimization for solving economic load dispatch problem. *International Journal of Communication and Computer Technologies (IJCCTS), 1*(2), 100-105.

[31] Sreenivasu, M., Kumar, U. V., & Dhulipudi, R. (2022). Design and Development of Intrusion Detection System for Wireless Sensor Network. *Journal of VLSI Circuits and Systems, 4*(2), 1-4.

[32] Srinivasareddy, S., Narayana, Y. V., & Krishna, D. (2021). Sector beam synthesis in linear antenna arrays using social group optimization algorithm. *National Journal of Antennas and Propagation (NJAP), 3*(2), 6-9.

[33] Win, T. Z., & Kham, N. S. M. (2019). Information Gain Measured Feature Selection to Reduce High Dimensional Data. *In Proceedings of the 17th International Conference on Computer Applications (ICCA)*, 68–73. Novotel hotel, Yangon, Myanmar.

[34] Yağız, E., Ozyilmaz, G., & Ozyilmaz, A. T. (2022). Optimization of graphite-mineral oil ratio with response surface methodology in glucose oxidase-based carbon paste electrode design. *Natural and Engineering Sciences, 7*(1), 22-33.

Authors Biography

Deepthi Reddy Dasari, Research scholar from the Department of Computer Science and Engineering, GITAM School of Technology, GITAM Deemed to be University and working as Assistant Professor, Department of Computer Science and Engineering had 7 years of teaching experience in various technologies. Published 6 research papers from various reputed journals and attended conferences. Successfully completed 3 NPTEL courses. Apart from these attended faculty development programs to learn new domains.

Dr.G. Hima Bindu, Assistant Professor, Department of Computer Science and Engineering, GITAM School of Technology, GITAM Deemed to be University has 15 years of teaching experience. Published around 10 research papers in various reputed Scopus and Science citation indexed journals and guiding around 4 research scholars. The areas of research include Image processing, Information security and optimization in AI.