International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 18(2s) 2026 pp. 1221-1254
https://doi.org/10.70917/ijcisim-2026-2225
Article

ADAPTIVE MULTI-OBJECTIVE FEATURE OPTIMIZATION USING RECURSIVE MONTE CARLO TREE SEARCH FOR NETWORK INTRUSION DETECTIONS

Radharani Akula[1]*, G. S. Naveen Kumar[2], A. V. L. N. Sujith[3]

[1] Department of Computer Science and Engineering, Malla Reddy University, Hyderabad – 500100, Telangana, India
[2] Department of Data Science, Malla Reddy University, Hyderabad – 500100, Telangana, India
[3] Department of Computer Science and Engineering, Malla Reddy University, Hyderabad – 500100, Telangana, India
Corresponding Author: Radharani Akula (Email: aradharani786@gmail.com)

Abstract: Intrusion detection in modern network environments is challenged by high-dimensional data, imbalanced attack classes, and constantly evolving cyber threats. Traditional intrusion detection methods often rely on fixed or single-objective feature selection techniques, limiting their ability to achieve high detection performance while maintaining computational efficiency. This paper proposes an adaptive feature optimization framework that combines recursive Monte Carlo Tree Search (MCTS) with multi-objective optimization for effective intrusion detection. The proposed Adaptive Recursive Multi-Objective Feature Selection (ARMFS) framework progressively refines feature subsets through recursive search, enabling efficient exploration of the feature space. Feature subsets are evaluated using two objectives: maximizing classification accuracy and minimizing the number of selected features. This strategy identifies compact and informative feature sets while reducing computational complexity. The optimized features are used to train a deep neural network classifier, with its hyperparameters tuned using a nature-inspired optimization algorithm to improve convergence and generalization. Experimental results on benchmark intrusion detection datasets demonstrate that the proposed framework achieves high detection accuracy with significantly fewer features than the original datasets. The method also improves the detection of minority attack classes, maintains low false positive rates, and produces consistent performance across multiple validation runs. These results demonstrate that combining recursive search with multi-objective feature optimization provides an efficient and scalable solution for intelligent intrusion detection in dynamic and high-dimensional cybersecurity environments.

Keywords: Intrusion Detection System, Feature Selection, Monte Carlo Tree Search, Deep Learning, Dense Neural Network, Network Security..

1. INTRODUCTION

With the rapid expansion of digital communication and interconnected infrastructures, ensuring network security has become increasingly critical across both academic and industrial domains. The continuous growth in network traffic volume and complexity has significantly increased the difficulty of accurately detecting malicious activities in real time. Intrusion Detection Systems (IDS) serve as a key defense mechanism by monitoring and analyzing network behavior; however, their effectiveness is often constrained by evolving attack patterns, high-dimensional feature spaces, and the presence of imbalanced data distributions. These challenges necessitate the development of intelligent and scalable IDS frameworks capable of adapting to dynamic network environments. Traditional IDS approaches, including signature-based and anomaly-based methods, exhibit inherent limitations. Signature-based systems are effective in identifying known attack patterns but fail to detect previously unseen threats,

whereas anomaly-based methods are capable of identifying unknown intrusions but often suffer from high false positive rates. To overcome these limitations, data-driven techniques have been widely explored to enable automated learning of complex traffic patterns and improve detection performance.

Machine learning techniques have been extensively applied in intrusion detection due to their ability to model nonlinear relationships and generalize from historical data [1], [2]. However, conventional machine learning models rely heavily on feature engineering and are sensitive to redundant or irrelevant features, which can degrade performance and increase computational cost. This issue becomes more pronounced in high-dimensional datasets such as NSL-KDD, where unnecessary attributes can negatively affect model efficiency. Feature selection has therefore been recognized as a critical preprocessing step for improving model performance and generalization capability [3]. Existing feature selection techniques are broadly categorized into filter, wrapper, and embedded methods, each offering different trade-offs between computational efficiency and predictive performance [4]–[6]. Wrapper-based approaches can improve classification accuracy by evaluating feature subsets using predictive models [7], whereas filter-based methods provide computational efficiency for high-dimensional data [5], [8]. Several studies have explored optimization-based feature selection techniques, including genetic algorithms and hybrid neural network approaches, to enhance intrusion detection performance [10], [11]. Multi-objective optimization methods have also been investigated to identify optimal feature subsets by balancing competing criteria [12], while information-theoretic approaches have demonstrated effectiveness in reducing dimensionality and improving classification efficiency [13].

Recent research has further explored the integration of intelligent search mechanisms with feature selection. Monte Carlo-based strategies and mutual information techniques have been employed to guide feature exploration and improve detection accuracy in complex environments [14], [15]. In parallel, deep learning-based IDS models have gained significant attention due to their ability to automatically learn hierarchical feature representations from large datasets. Convolutional neural networks and auto encoder-based architectures have demonstrated strong performance in intrusion detection tasks [16]–[18], while deep neural networks have been effectively applied for anomaly detection in network security scenarios [19].

Metaheuristic optimization techniques have also played a significant role in enhancing feature selection and model performance. Approaches based on ant colony optimization, particle swarm optimization, genetic algorithms, and artificial bee colony methods have been widely used to identify optimal feature subsets and improve classification accuracy [20]–[24]. Additionally, gradient boosting and hybrid learning frameworks have contributed to improved detection capability in recent studies [25]-[29]. Among search-based approaches, Monte Carlo Tree Search (MCTS) has emerged as a promising technique for solving complex decision-making problems with large search spaces, owing to its ability to balance exploration and exploitation effectively [30]-[33]. Recent works have also explored hybrid deep learning and optimization frameworks for improving IDS performance and robustness [34]–[38], while modern AI-driven approaches continue to focus on scalability and real-time adaptability [39]–[41].

Despite these advancements, several challenges remain unresolved. Many existing feature selection methods rely on static or single-stage optimization strategies, which limit their ability to adaptively refine feature subsets during the search process. Furthermore, most approaches prioritize classification accuracy as the primary objective, without explicitly considering the trade-off between feature compactness and computational efficiency. This often results in models that are either computationally expensive or lack generalization capability, particularly in high-dimensional and imbalanced datasets. To address these limitations, this study proposes an adaptive feature optimization framework based on recursive Monte Carlo Tree Search. The proposed Dynamic Feature Selection with Recursive Monte Carlo Tree Search (DFS-RMT) framework introduces an iterative search mechanism that progressively refines feature subsets across multiple stages. Unlike conventional MCTS-based approaches, the proposed method incorporates a multi-objective evaluation strategy that jointly considers classification performance and feature compactness, enabling the identification of efficient and discriminative feature subsets. The recursive structure further reduces the effective search space in subsequent iterations, improving both efficiency and convergence stability.

The optimized feature subsets are integrated with a deep neural network architecture for multi-class intrusion detection. To enhance model performance and generalization, a nature-inspired optimization technique is employed for hyperparameter tuning. This integrated framework aims to achieve a balanced trade-off between detection accuracy, feature efficiency, and computational complexity, while improving the detection of minority attack classes.

The main contributions of this work are summarized as follows:

A recursive feature optimization framework based on Monte Carlo Tree Search is proposed, enabling progressive refinement of feature subsets across multiple search iterations.

A multi-objective evaluation strategy is introduced to jointly optimize classification performance and feature subset compactness.

A deep neural network-based classification model is integrated with adaptive hyperparameter optimization to enhance convergence and generalization.

The proposed framework improves detection performance for minority attack classes, addressing class imbalance challenges in intrusion detection.

Extensive experimental analysis demonstrates the robustness and scalability of the proposed approach in high-dimensional environments.

The remainder of this paper is organized as follows. Section 2 presents the proposed DFS-RMT-based framework, including feature selection and model architecture. Section 3 describes the recursive optimization strategy and hyperparameter tuning approach. Section 4 discusses the experimental setup, results, and performance evaluation. Finally, Section 5 concludes the paper and outlines directions for future research.

2. PROPOSED ADAPTIVE DFS-RMT FRAMEWORK

This section presents the proposed adaptive feature optimization framework based on recursive Monte Carlo Tree Search (DFS-RMT). The objective of the framework is to efficiently identify compact and discriminative feature subsets in high-dimensional intrusion detection datasets. Unlike conventional feature selection approaches, the proposed method adopts a recursive search strategy combined with multi-objective evaluation, enabling progressive refinement of feature subsets while balancing classification performance and feature compactness.

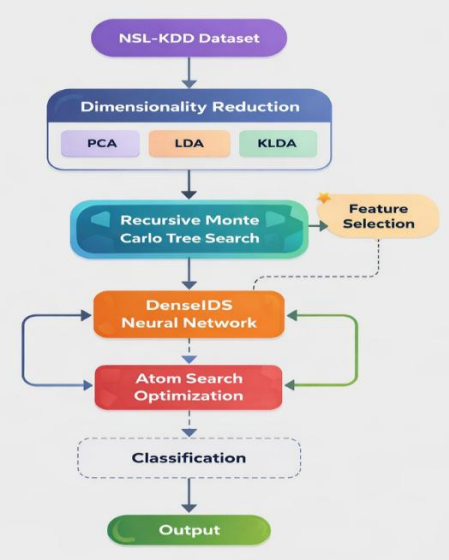


Figure 1. Overview of Proposed Adaptive DFS-RMT Framework.

The next stage involves building a deep neural network architecture that can determine the complex phenomena and relationships between the selected features. This neural network is supplemented by a nature-inspired metaheuristic technique to optimize the model parameters and improve the prediction capabilities. Through this optimization phase, the network reaches a balance between model complexity and generalization; hence, potential overfitting is prevented, and robustness is improved. The effectiveness of the DenseIDS framework on a benchmark high-dimensional dataset representative of real-world intrusion detection scenarios. The results demonstrate the framework's ability to effectively handle high-dimensional features, optimize feature subsets, and improve performance in intrusion detection tasks. DenseIDS is an adaptable and expandable technology that offers a feasible solution to modern network security challenges in academic research and industrial enterprises. Figure 1 shows a schematic of the proposed methodology. We conducted nine non-parametric and statistical tests against the data divided into random sets of ten chunks to establish the consistency of the system.

Multi-Objective Optimization Strategy

The proposed DFS-RMT framework incorporates a multi-objective optimization strategy to ensure a balanced trade-off between detection performance and feature efficiency. Unlike conventional feature selection methods that optimize a single objective (typically classification accuracy), the proposed approach simultaneously considers multiple criteria during the feature subset evaluation process. This enables the identification of compact and discriminative feature subsets while avoiding unnecessary computational complexity.

Let $S \subseteq F$ denote a candidate feature subset selected from the original feature set F, where $|S|$ represents the number of selected features and $|F|$ is the total number of features. The feature selection process is formulated as a bi-objective optimization problem.

Objective 1: Maximize classification performance

Objective 2: Minimize feature subset size

The first objective is defined in terms of classification effectiveness, which can be measured using accuracy or F1-score depending on the evaluation setting. The second objective is represented using the feature selection ratio, defined as:

$$FSR = \frac{|S|}{|F|} \tag{1}$$

To integrate both objectives into a unified evaluation framework, a weighted aggregation function is employed:

$$J(S) = \alpha \cdot Perf(S) - \beta \cdot FSR \tag{2}$$

where:

Perf(S) denotes the classification performance of subset S,

FSR is the feature selection ratio,

α and β are weighting coefficients such that $\alpha+\beta=1$, controlling the trade-off between accuracy and feature compactness.

This formulation approximates a Pareto trade-off between performance and feature compactness. The optimization process aims to maximize J(S), thereby selecting feature subsets that achieve high classification performance while maintaining minimal dimensionality. In this work, the weighting parameters are chosen empirically to ensure that performance is prioritized while still enforcing feature reduction. Within the DFS-RMT framework, this multi-objective evaluation is embedded into the Monte Carlo Tree Search process. During each iteration, candidate feature subsets generated through the tree expansion and simulation phases are evaluated using the objective function J(S). The resulting value is treated as the reward signal for guiding the search process.

The recursive structure of DFS-RMT further enhances the optimization process by progressively refining the search space. At each recursion level, the best-performing feature subset identified in the previous iteration is used as the input for constructing a new search tree. This iterative reduction in the search space improves convergence efficiency while preserving high-quality feature combinations. Additionally, the multi-objective formulation inherently improves model generalization by preventing over-reliance on large feature subsets. By penalizing excessive feature selection, the framework encourages the discovery of minimal yet informative feature representations, which are particularly beneficial in high-dimensional and imbalanced intrusion detection scenarios.

Overall, the integration of multi-objective optimization within the recursive MCTS framework enables a systematic and efficient exploration of the feature space, achieving a robust balance between detection accuracy, feature efficiency, and computational cost.

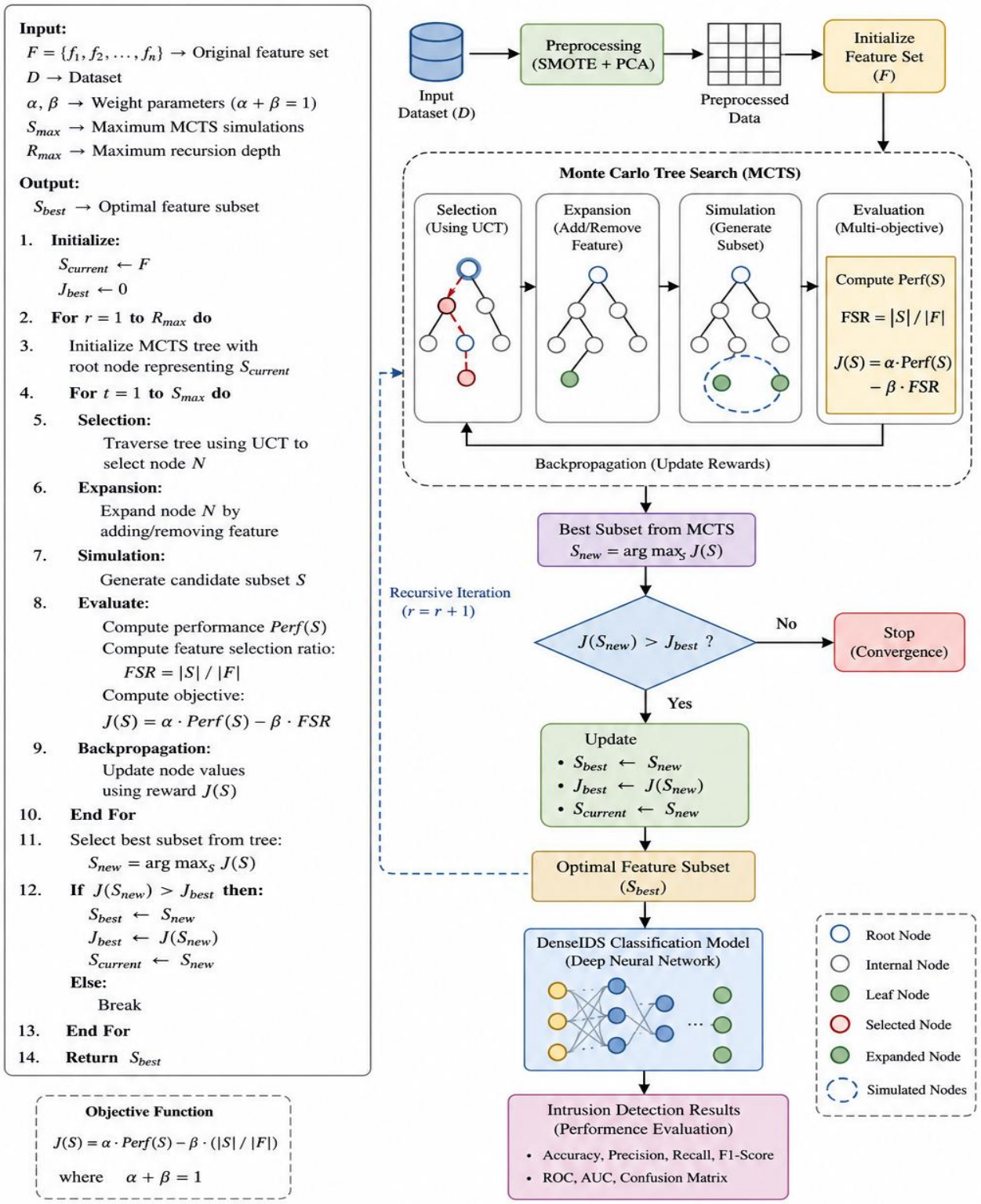
Figure 2 presents the overall architecture of the proposed DFS-RMT-based framework for adaptive multi-objective feature optimization in intrusion detection. The process begins with the input dataset, which is first subjected to preprocessing to handle class imbalance and reduce redundancy in the feature space. Techniques such as synthetic

oversampling and dimensionality reduction are applied to ensure that the data are both balanced and computationally manageable before feature optimization. Following preprocessing, the refined feature set is used to initialize the search process. The core of the framework is built around a Monte Carlo Tree Search (MCTS) mechanism, which systematically explores the feature space. At each iteration, the search proceeds through four stages. In the selection phase, the algorithm identifies promising nodes based on a balance between prior performance and exploration potential. The expansion phase introduces new candidate feature combinations by modifying the current subset. During simulation, these candidate subsets are evaluated under stochastic conditions to approximate their effectiveness. The outcomes of these simulations are then used in the backpropagation phase to update the search tree, reinforcing feature combinations that yield better results.

A distinguishing aspect of the proposed framework is the integration of a multi-objective evaluation strategy within the search process. Instead of focusing solely on classification accuracy, the method evaluates each feature subset by jointly considering predictive performance and subset compactness. This balance is achieved through an objective function that penalizes unnecessary feature inclusion while rewarding improved detection capability. As a result, the search process naturally favors solutions that are both accurate and efficient. The framework operates recursively, where the best-performing feature subset identified in one iteration is used to guide the search in subsequent iterations. This progressive refinement reduces the effective search space and allows the algorithm to converge toward an optimal subset without exhaustive exploration. The recursion continues until no further improvement is observed, ensuring both stability and efficiency in the selection process.

Once the optimal feature subset is obtained, it is provided as input to a deep neural network-based classifier. This model is responsible for performing the final intrusion detection task, producing predictions across multiple attack categories. The performance of the system is evaluated using standard metrics, including accuracy, precision, recall, F1-score, and ROC-based measures. Overall, the figure illustrates how the proposed framework integrates recursive search, multi-objective optimization, and deep learning into a unified pipeline. This design enables effective handling of high-dimensional data while maintaining a balance between detection performance and computational efficiency.

Figure 2. Workflow of the Proposed DFS-RMT-Based Multi-Objective Feature Optimization Framework.



The proposed DFS-RMT framework operates through an iterative and recursive optimization process as shown in Figure 2. Initially, the complete feature set is provided as input to the MCTS-based search structure. During each iteration, candidate feature subsets are generated through selection, expansion, and simulation phases. These subsets are evaluated using a multi-objective function that balances classification performance and feature compactness. The evaluation score is propagated back through the tree to guide subsequent exploration. At the end of each recursion, the best-performing feature subset is selected and used as the input for the next iteration. This recursive refinement continues until no further improvement is observed, resulting in an optimal and compact feature subset.

Preprocessing and Data Preparation

To address class imbalance in the dataset, a preprocessing step is applied using the Synthetic Minority Oversampling Technique (SMOTE). This approach generates synthetic samples for minority classes by interpolating between existing instances in the feature space, thereby improving class distribution without simple duplication. The application of SMOTE ensures that minority attack classes are adequately represented during training, which is critical for improving detection performance. However, excessive oversampling may introduce synthetic bias. To mitigate this, the proposed framework evaluates performance across multiple validation folds, ensuring that the model generalizes beyond artificially generated samples.

$$X_{new} = X_i + \gamma\left(X_k - X_i\right) \quad (3)$$

Where Xi denotes the minority class sample; X_k denotes k nearest neighbor in the class; γ is a random number between 0,1; and finally X_{new} is the new synthetic sample. In the dataset, the minority classes are balanced using SMOTE to achieve a comparable distribution with majority classes, ensuring improved learning without excessive duplication.

Dimensionality Reduction

Dimensionality reduction is applied to eliminate redundant information and improve computational efficiency before feature selection. In this work, multiple techniques, including PCA, LDA, and KLDA, are evaluated to analyze their impact on feature representation. Among these, PCA is selected due to its ability to retain maximum variance while reducing dimensionality, making it suitable for subsequent feature optimization.

Moreover, dimensionality reduction increases the interpretability of the data. By concentrating only on the most relevant features or changes, the dataset is easier to interpret for human analysts. This added clarity helps to keep sight of the basic associations in the data, making for stronger conclusions and better choices. Additionally, dimensionality reduction is a resolution for the curse of dimensionality, which is a phenomenon observed in high-dimensional datasets that are often sparse and challenge algorithms to generalize effectively. Lower dimensionality leads to reduced sparsity, which, in turn, aids models in learning stronger patterns more efficiently. In most cases, removing irrelevant or redundant features improved the model performance. By focusing specifically on the most important information, noise is reduced, generalization is increased, and ultimately, accuracy on the prediction task is improved. Overall, dimensionality reduction has many benefits that highlight its use in building effective and efficient machine learning models. Amongst existing methods, filter methods are popular for dimensionality reduction, specifically feature selection, as they assess the importance of individual features in reference to the target variable via statistical metrics such as correlation coefficients, mutual information, chi-square tests, or variance thresholds. For instance, features that demonstrate minimal variance can be eliminated, as they offer little to no usage in differentiating between groups or explaining the variances within the target variable. This process, called feature selection, reduces datasets to remove redundant or noisy features, keeping only the most informative ones.

$$X_{standardised} = \frac{X - \mu}{\sigma} \quad (4)$$

Following standardization, PCA calculates the covariance matrix C of the data to analyze the interrelationships among the characteristics of the data. The covariance matrix is represented by the following equation:

$$C = \left(\frac{1}{n-1}\right) X^T X \quad (5)$$

where ‘n’ denotes the number of samples or rows in the data. This captures the covariance between each feature pair-wise. PCA continues to find eigenvectors and eigenvalues from this covariance matrix. These are the directions with the highest possible variance. The principal components are also called this, and the eigenvalue relates to how much that is measured by taking it across this direction. The eigenvectors are sorted in order of the descending magnitude of their corresponding eigenvalues. The first eigenvector is along the direction of the greatest variance, the second is along the second greatest variance, and so on. The principal components (PCs) are the eigenvectors, and the eigenvalues indicate the variance captured by each PC. Finally, the data were projected onto the selected principal components to form a new set of uncorrelated features. If we select the top k eigenvectors (principal components), the transformed dataset is

$$X_{reduced} = X_{standardised} W_k \quad (6)$$

Here, is the matrix of ‘k’ eigenvectors.

DFS-RMT Feature Selection

The DFS-RMT (Dynamic Feature Selection with Recursive Monte Carlo Tree Search) framework constitutes the core component of the proposed methodology, enabling efficient exploration and optimization of feature subsets in high-dimensional intrusion detection datasets. Unlike conventional feature selection approaches that rely on static or single-stage optimization, the proposed framework introduces a recursive search mechanism integrated with a multi-objective evaluation strategy. This combination allows for progressive refinement of feature subsets while maintaining a balance between classification performance and feature compactness.

In the DFS-RMT framework, feature selection is formulated as a sequential decision-making process, where each candidate subset is represented as a node within a search tree. The exploration of this tree is guided by the Monte Carlo Tree Search (MCTS) algorithm, which iteratively constructs and evaluates feature subsets through four key phases: selection, expansion, simulation, and backpropagation. During the selection phase, promising nodes are identified based on their historical performance and exploration potential. The expansion phase generates new candidate subsets by adding or removing features, while the simulation phase evaluates these subsets under stochastic conditions. The results obtained from simulation are then propagated back through the tree to update node statistics, guiding future exploration.

A key distinguishing aspect of the proposed approach is the integration of the multi-objective evaluation function within the DFS-RMT process. Instead of optimizing solely for classification accuracy, each candidate feature subset is evaluated using the objective function J(S), which jointly considers classification performance and feature subset size. This ensures that the search process does not favour overly large feature subsets and instead promotes compact and efficient representations without compromising detection capability.

The recursive structure of DFS-RMT further enhances its effectiveness. At each iteration, the best-performing feature subset identified from the MCTS process is selected and used as the input for the subsequent iteration. This recursive refinement reduces the effective search space in a progressive manner, allowing the algorithm to focus on increasingly promising regions of the feature space. As a result, the framework achieves faster convergence compared to traditional search-based feature selection methods, which typically operate over a fixed and often large search space.

The iterative process continues until a stopping criterion is satisfied, such as the absence of improvement in the objective function or reaching a predefined recursion limit. The final output of the DFS-RMT framework is an optimal feature subset that achieves a desirable balance between predictive performance and dimensionality reduction.

By integrating recursive search, multi-objective optimization, and guided exploration through MCTS, the DFS-RMT framework provides a robust and scalable solution for feature selection in intrusion detection systems. This approach not only improves classification accuracy but also significantly reduces computational complexity by eliminating redundant and irrelevant features, making it suitable for real-world high-dimensional cybersecurity applications.

Deep Neural Network Model (DenseIDS)

To further enhance model performance, a nature-inspired optimization technique is employed to tune the hyperparameters of the neural network. The selected feature subsets are evaluated using a deep neural network model designed for multi-class intrusion detection. The architecture consists of multiple dense layers with batch normalization and dropout regularization, enabling effective learning while preventing overfitting.

$$S_W = \Sigma_{i=1}^{k} \Sigma_{\{x \in C_i\}} \left(x - \mu_i\right)\left(x - \mu_i\right)^2 \quad (7)$$

where represents the i[th] class, is the mean of the class, and k is the number of classes.

The main idea of LDA is to find the linear combinations of features that maximize the ratio of the determinant of, which is formally written as:

$$\frac{J\left(W\right) = \det\left(S_B\right)}{\det\left(S_W\right)} \quad (8)$$

This objective function is maximized by solving the generalized eigenvalue problem as follows:

$$S_B v = \lambda S_W v \quad (9)$$

where v is the eigenvector (direction of maximum discrimination), and is the eigenvalue. Once the eigenvectors are computed, those with the highest eigenvalues are chosen to form a transformation matrix. The data were then projected onto these eigenvectors to reduce their dimensionality.

Hyperparameter Optimization (ASO)

Kernel Linear Discriminant Analysis, an extension of the traditional LDA to address data that are not linearly separable using kernel approaches. It uses a transformation from a low-dimensional input space into a possibly higher dimensional space via the use of a kernel function when data are not separable within a linear feature space. The main advantage of KLDA over LDA is that it can handle complex decision boundaries between classes, which improves the classification performance in cases where datasets have more complex patterns. The key idea behind KLDA is to apply a nonlinear transformation using a kernel function. Instead of directly working with the input data, KLDA first maps the data from the original feature space X into a higher-dimensional space using a mapping function

$$\varphi\left(X\right) \rightarrow H \quad (10)$$

Where *H* is the Hilbert space. The kernel used in this model was a sigmoid kernel.

Similar to LDA, KLDA seeks to maximize class separation and minimize variance within each class. The optimization problem of LDA is defined in terms of scatter matrices, the within-class scatter matrix, and the between-class scatter matrix. For KLDA, the same scatter matrices are used, but these scatter matrices are calculated in the kernel space using kernelized data. This is achieved by utilizing two measures.

Kernelized Within-Class Scatter Matrix calculated by

$$S_W = \Sigma_{i=1}^{k} \Sigma_{x \in C_i} K\left(x, x_i\right) \quad (11)$$

Kernelized Between-Class Scatter Matrix calculated by

$$S_B = \Sigma_{i=1}^{k} N_i \left(K\left(\mu_i, \mu\right)\right) \quad (12)$$

Where N_i is the data point count in the c_i class, and is the mean of the class, while the total mean is.

Computational Complexity

The recursive structure of DFS-RMT reduces the effective search space at each iteration, enabling efficient exploration without exhaustive enumeration. This results in a significant reduction in computational cost compared to traditional feature selection methods. Feature selection is a very important first step in any complex problem in machine learning and data mining. One way to do this is through feature selection, which is the process of removing

unimportant or irrelevant features from a dataset and retaining only the most important and relevant features. Some of the major benefits of feature selection are improved performance (higher prediction accuracy), reduced computational cost, and more comprehensible models. In feature selection, identifying and evaluating feature subsets typically consists of a paired search component (the process of exploring the space of possible features) and an evaluator component shown in Figure 3. The Search Component is the method employed to traverse the space of potential feature subsets. Forward selection and backward elimination are some of them, as well as more complex techniques, such as genetic algorithms or greedy methods.

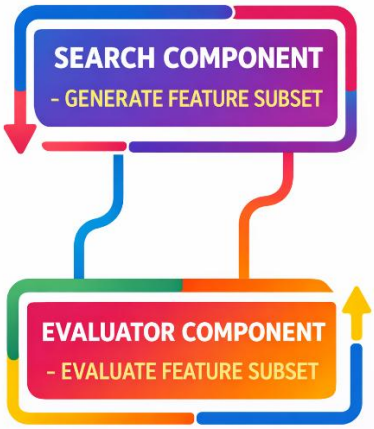


Figure 3. Components of feature selection.

The proposed model is named DFS-RMT (Dynamic Feature Selection with Recursive Monte Carlo Tree Search) as shown in Figure 3. The main evaluation metric used to assess the DFS-RMT was the feature selection ratio. The complexity can be approximated as O(R × S × C), where R is recursion depth, S is number of simulations, and C is model evaluation cost.

Monte Carlo Tree Search

Monte Carlo Tree Search (MCTS) is an adaptive search strategy designed for solving complex decision-making problems with large and uncertain search spaces. In this work, MCTS is employed to identify an optimal subset of features by formulating feature selection as a sequential decision process. Each node in the search tree represents a partial feature subset, while branches correspond to decisions regarding the inclusion or exclusion of individual features. The algorithm iteratively constructs the search tree using stochastic sampling and performance-driven feedback. During each iteration, candidate feature subsets are generated and evaluated using a classification model, where predictive multi-objective function J(S) is used as the reward signal. This allows the search process to progressively favor feature combinations that improve classification performance.

To maintain efficiency, the proposed approach integrates both filter-based ranking and wrapper-based evaluation. Initially, candidate subsets are refined using lightweight statistical criteria to eliminate irrelevant features. These refined subsets are then assessed through model-based evaluation, ensuring that feature interactions are captured effectively. By balancing exploration of new feature combinations with exploitation of previously successful subsets, MCTS enables efficient navigation of the high-dimensional feature space. The iterative refinement continues until a predefined stopping condition is satisfied, such as convergence in performance or reaching the maximum number of iterations. This strategy ensures a systematic and scalable approach to feature selection without requiring exhaustive search.

2.8.1 Selection

The selection phase determines the path to be followed within the existing search tree based on prior exploration outcomes. Starting from the root node, the algorithm traverses through child nodes by selecting those that exhibit a favorable balance between historical performance and exploration potential. This decision-making process is guided by a tree policy that evaluates each node using a reward-based criterion. Nodes associated with higher classification performance are more likely to be selected, while less explored nodes are still considered to maintain diversity in the search process. This balance prevents premature convergence to suboptimal feature subsets.

At each level of the tree, the decision to include or exclude a feature is represented by branching to corresponding child nodes. The traversal continues until a node is reached that has not been fully expanded or requires further exploration. The selected path thus represents a promising candidate subset that will be further refined in subsequent phases. Through this mechanism, the selection phase ensures that the search process remains both performance-driven and exploratory, enabling efficient identification of informative feature combinations in high-dimensional datasets.

$$UCT = \left(\frac{x_i}{m_i} \right) + \delta \, sqrt \left(\frac{\ln M}{m_i} \right) \qquad (12)$$

Where x_i is the total reward of node i

m_i is the total number of nodes i has visited

M is the total visits to the parent node

δ is the exploration parameter

The UCT algorithm identifies the node to be selected at each level of the tree. Selecting node Si at level i signifies the inclusion of feature Si in the current feature subset, whereas opting for results in its exclusion. The choice of Si is based on the rationale that its incorporation yielded substantial rewards in prior iterations, indicating that it should be included in the current feature subset. Conversely, the non-selection of Si suggests that it did not substantially enhance the rewards previously, rendering its elimination advantageous.

2.8.2 Expansion

Considering the sequence of actions, the existing new child nodes are added, and the expansion of the tree occurs. The urgent node, which is the final node selected in the selection process, gains a new child node during expansion. The incorporation of a new child node at node is similarly based on the UCT function. If ‘exceeds, then the child node is included, hence including feature in the current feature subset. In contrast, the child node is incorporated, but feature is excluded from the present feature subset.

2.8.3 Simulation

On a child node, a simulation is used to compute an approximate outcome. The default policy incorporates randomness into the creation of feature subsets throughout the simulation phase. It randomly selects attributes from the set of unexpanded attributes, with each attribute having an equal probability of selection. Upon the recent expansion of node, a random path is chosen from to a leaf node. When the current expanded node is, the selection and expansion phases determine which features from S1 to Si are included in the current feature subset. In the simulation phase, a random decision is made regarding the inclusion of the remaining features from S_i+1 to Sn in the current subset. This methodology combines the use of a tree search with a random search to obtain feature subsets, thus identifying optimal combinations of features in far fewer iterations without the need to expand the complete search tree.

2.8.4 Backpropagation

The simulation reward is then justified by the selected nodes to update the tree statistics. The classifier was further used to determine the merit of the feature subset. The multi-objective function J(S) is used as the reward signal for simulation, "Simulation," of the now chosen nodes that then updates itself backwards to update the tree.

$$Q = Accuracy_{classifier} \left(S_{subset} \right) \qquad (13)$$

If the accuracy of the present subset of features exceeds that of the prior best, the current subset of features is designated as the optimal feature subset. This iteration continues until the stopping criteria are satisfied.

3. PROPOSED DFS-RMT (DYNAMIC FEATURE SELECTION WITH RECURSIVE MONTE CARLO TREE SEARCH)

The proposed DFS-RMT selects a number of feature trees recursively, which are built recursively in an iterative manner. Successive feature selection trees possess a smaller state space than their predecessors; thus, the efficiency of the tree search in finding ideal features is enhanced while maintaining a consistent number of MCTS simulations. A

collection of characteristics F is used as the first input to build several trees in succession. Each tree produces an ideal selection of features as output after running the S-MCTS simulations. The output of each tree acts as the input to the following tree in the chain. This iterative procedure continues until the aforementioned condition is met, finally yielding the best set of characteristics.

3.1 Recursion method in DFS-RMT

The goal of this method is to find the best feature set (S_{best}) for 'm' recursions. In each recursion step, the DFS-RMT simulation constructs a feature selection tree to search for the best subset of features.

The predetermined criteria of DFS-RMT assume that there are 0 to m recursions such that r= 0,1,2…i, j, .m. Initially, the input is the feature set Si for building the search tree. After the DFS-RMT simulation, is identified, where. When the condition that s_{best} is better than s_i, it is concluded that S_{best} is reassigned as $S_{optimal}$, which is the optimal feature subset. Subsequently, the next recursion (j_{th}) takes the input of S_{best}. The condition now satisfies the condition. It subsequently creates another tree of feature subsets that produces. The DFS-RMT continues iteratively until the stopping criterion is met. In this research we define the stop criterion as

$$Accuracy\left(S_{best}^{j}\right) < Accuracy\left(S_{best}^{j-1}\right) \qquad (14)$$

The above equation implies that the child tree generates an accuracy that is less than the parent tree, the stop criterion is met, and the recursive process ends at that point. A visual representation of the DFS-RMT model is presented in Figure 4.

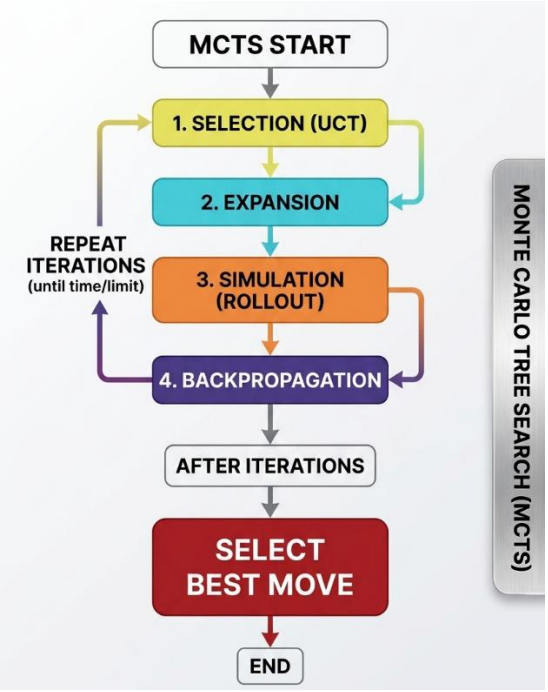


Figure 4. Proposed DFS-RMT.

In the next phase, the feature tree is generated, as shown in Figure 5. There are two predefined conditions that the tree must satisfy to achieve the best subset (S):

This implies that the feature has not yet been selected.

If i is the level of the tree, it has only two children S_i.

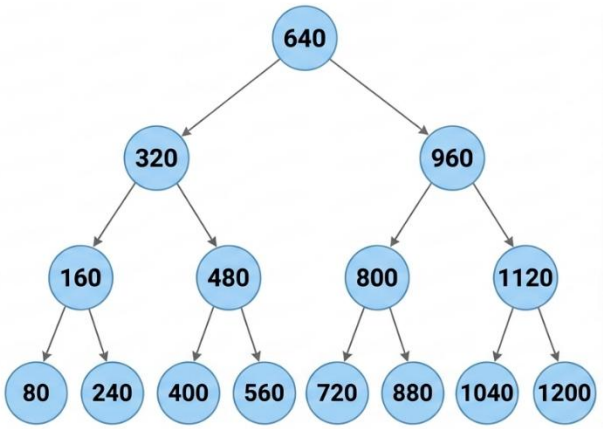


Figure 5. Feature sub tree generation.

The feature states are represented by S_i. denotes the selection and non-selection of a feature from the subset. A subset is defined as the path followed by the feature to the leaf node from the root node. The ultimate goal was to identify the path with the best accuracy. The DFS-RMT traverses all nodes and extracts the most accurate route. The traversed features in the selected path are collectively referred to as the accurate feature subset S_{best}. Figure showcases the tree constructed by the DFS-RMT. Subsequently, a feature subset is generated based on the principle of progressively adding features as nodes, which in turn builds the search tree. The steps involved in feature set generation are selection, expansion, and simulation. The inclusion or exclusion of a feature into the subset is determined by the same equation given in the selection section, that is, Upper Confidence Bounds for Trees:

$$UCT = \left(\frac{x_i}{m_i} \right) + \delta \, sqrt \left(\frac{\ln M}{m_i} \right) \qquad (15)$$

A stochastic simulation was performed from the top-level node to the terminal node, integrating the characteristics into the subset according to the standard protocol. This unique method expedites the discovery of the optimal feature subset by integrating tree search with random sampling, eliminating the need to traverse the entire feature selection tree.

3.1.1 Dense Neural Network for classification

The DenseIDS architecture is a systematic multi-class classification feedforward neural network. It combines several dense layers, batch normalization, and dropout regularization methods to determine the optimal trade-off between complexity, capacity, and generalization. This results in a trained network with a total of 309279 parameters to learn, striking an interesting balance between the depth of the model to learn high-level representations and normalizing approaches to avoid overfitting. First, this model started with an input layer connected to the first dense layer with 256 nodes. This layer acts with a Rectified Linear Unit (ReLU) as its activation function, which is a conventional nonlinear activation, providing sparsity in the activation space while eliminating vanishing gradient problems. It consists of 2,304 trainable parameters (weights and biases) that connect the input features to the next-level representations. This is followed by a batch normalization layer, which helps stabilize the optimization process. It is an internal covariate shift and batch normalization that standardizes the input to the activation function at each batch. It has 1,024 learnable parameters corresponding to the scaling and shifting parameters of each feature. A dropout layer is applied, which disables the neurons in the network during training to mitigate overfitting and enhance generalization. Another dense layer expands the feature representation to 512 nodes, capturing more abstract and higher-level features of the input. This layer has 131584 trainable parameters, indicating the increased capacity that the model must have to handle this dimensional explosion. Again, batch normalization is employed in this layer to normalize the activations and facilitate optimization, adding 2,048 parameters to our network. Again, dropout is applied, which reinforces the regularization strategy and allows the model to be robust to a larger number of parameters.

A third layer is then added, which is a dense layer that compresses the feature depth back to 256 nodes, enabling the system to extract intermediate-level representations. Despite this contribution, it comprises 131,328 trainable parameters, a deliberate architectural design selection chosen to reduce the utilized feature space while preserving the critical learned information along the way. After the activation function, a batch normalization layer is added that accounts for 1,024 parameters, normalizing how each channel is scaled, and therefore reducing inconsistencies that cause instability during training. To mitigate the risk of relying too much on a neuron, dropout is introduced at this stage. The fourth dense layer reduces the feature space to 128 nodes to further improve the feature representations. The part that abstracts and compresses the learned features (32,896 parameters). Next is batch normalization (512 parameters), which also serves as a stabilizer, and dropout regularization, which guarantees that the model does not memorize the training data while maintaining good generalization. The fifth layer reduces the dimensionality of the features to 64, with 8,256 trainable parameters, capturing crisp high-level features. At this point, a dropout layer is applied, which aligns with the general design principle of regularization.

The output layer has five nodes, each corresponding to a target class (classification task). The final layer employs a softmax activation function, which converts the raw logits into a probability distribution across the classes. The softmax function normalizes the raw scores into probability values; therefore, the interpretation of the outputs of the model is straightforward — the normalized scores sum to one. The last layer of the model, or the output layer, consists, in this specific case, of 325 trainable parameters: the weights and biases that connect the last hidden layer to the output. This architecture illustrates a meticulously organized neural network design that utilizes dense layers to progressively enhance feature representations and integrates batch normalization to stabilize training dynamics. The intentional implementation of dropout layers across the network guarantees a resilient model that mitigates overfitting and generalizes effectively to novel inputs. The integration of ReLU with softmax activation algorithms enhances computing efficiency and classification. This model is adept at multi-class classification problems on high-dimensional datasets because of its balanced complexity and scalability of architecture. The modular design facilitates expansion and adaptability for several practical machine learning applications. The architecture is shown in Table 1.

Table 1: Architecture Details of Proposed DenseIDS Model.

Layer Type	Activation Function	Output Shape	Parameters	No. of Neurons
Dense	ReLU	(None, 256)	2,304	256
Batch Normalization	—	(None, 256)	1,024	—
Dropout	—	(None, 256)	0	—
Dense	ReLU	(None, 512)	131,584	512
Batch Normalization	—	(None, 512)	2,048	—
Dropout	—	(None, 512)	0	—
Dense	ReLU	(None, 256)	131,328	256
Batch Normalization	—	(None, 256)	1,024	—
Dropout	—	(None, 256)	0	—
Dense	ReLU	(None, 128)	32,896	128
Batch Normalization	—	(None, 128)	512	—
Dense	ReLU	(None, 64)	8,256	64
Dropout	—	(None, 64)	0	—
Dense (Output)	Softmax	(None, 5)	325	5

Optimization in deep learning is the process of updating the parameters of a neural network, such as its weights and biases, to minimize the error or loss function, which estimates the difference between the predictions of the network and the real target values. In general, optimization aims to improve the accuracy and generalization capability of the model to ensure that it performs well on new data. The optimized function is the key part of the deep learning model training. They are initialized randomly, and optimization is the tool that brings continual updates to discover those ordered and useful bindings between input and output. The model did not converge to a state capable of making

reliable predictions without optimization. DenseIDS 1,2 were optimized with a nature-based meta-heuristic optimizer called the atom search optimization algorithm. It is described below:

Atom search optimizer

The Atom Search Optimization Algorithm (ASOA) is a nature-inspired meta-heuristic optimization algorithm that simulates the behavior of atoms in nature. It is designed for current populations, and some of the key dynamics from atomic physics, such as attraction and repulsion, are used to dynamically search for the best possible solution to problems. The number of points representing possible solutions was generated randomly and then operated by attraction/repulsion based on one another. The algorithm in each iteration assesses the solutions and determines the attraction force between them. The solutions with the strongest attraction forces are combined to form a new solution. This continues until they find a solution with which they are satisfied. The geometric constraint was maintained at 0.0001, and the interaction force was calculated as follows:

$$F_{ij(t)} = \left(24\frac{\varepsilon(t)}{\sigma(t)}\right)\left(2\left(\frac{\sigma(t)}{r_{ij(t)}}\right)^{13} - \left(\frac{\sigma(t)}{r_{ij(t)}}\right)^{7}\right) \qquad (16)$$

A high learning rate can accelerate the learning process and reduce the risk of settling into a local optimum. Once the model's accuracy was established, the learning rate was adjusted to make the learning process more precise. Weighted cross-entropy was applied to address the discrepancy in the sample size by adjusting the contribution to the loss of each sample according to its size, thereby ensuring an equal representation of samples in the loss calculation.

$$Loss = \Sigma_{i=1}^{M} w_i \left(y_i \log\left(its_i\right) + \left(1 - y_i\right)\log\left(1 - its_i\right)\right) \quad (17)$$

where $w_i = \frac{N_{total}}{\left(M\, N_i\right)}$, w is the weight and M is the class number.

3.2.1 Selection of parameters of the CNN model using atom search optimization

The Atom Search Optimization Algorithm is a nature-inspired optimization technique modelled with atomic behavior. It applies physical principles for atom molecular dynamics to provide solutions for optimization problems. It simulates the interaction forces in atoms to move the search patterns towards the global optimum solution. Each solution to the problem is represented as an atom, whose position corresponds to a potential solution. Interatomic forces are computed based on principles of attraction and repulsion, such as Van der Waals forces and the Lennard-Jones potential, so that the atoms are guided towards better solutions. These forces determine the acceleration and velocity of the atom within the search space; therefore, its movement is determined by it. By iteratively updating the positions, the ASO explores the solution space to optimize the objective function. The algorithm checks the fitness of the atoms, retains the best solutions, and continuously refines the search until convergence or a number of iterations. Dynamically balancing exploration and exploitation is the key to making ASO highly effective in solving optimization problems that are highly dimensional and complex. The DenseIDS 1,2 architecture used in this study comprises a variety of hyper parameters, such as learning rate, number of neurons in the dense layer, batch size, loss, activation layer, and dropout rate. The Enhanced atom search optimizer (EASO) was employed to determine a few hyper parameter values. The enhanced Atom Search Optimizer (ASO) illustrated in Figure 6 presents the workflow adopted for adaptive hyper parameter tuning within the proposed framework. The optimization process begins with the initialization of a population of candidate solutions, where each atom represents a possible configuration of model parameters. These atoms interact through attraction–repulsion mechanisms inspired by molecular dynamics, allowing the search process to explore the solution space efficiently. During each iteration, the fitness of candidate solutions is evaluated based on the classification performance obtained from the DFS-RMT framework. The interaction forces guide atoms toward promising regions, while controlled randomness prevents premature convergence.

As the optimization progresses, the algorithm gradually shifts from exploration to exploitation, enabling fine-tuning of the best-performing solutions. The enhanced strategy incorporates adaptive parameter control to improve convergence stability and avoid stagnation in local optima. The iterative update of positions and velocities ensures that candidate solutions are refined based on both individual performance and collective behavior. This results in improved hyper parameter selection, which directly contributes to better generalization and classification accuracy.

The integration of the enhanced ASO within the proposed framework thus provides a systematic approach for optimizing model performance without excessive computational overhead.

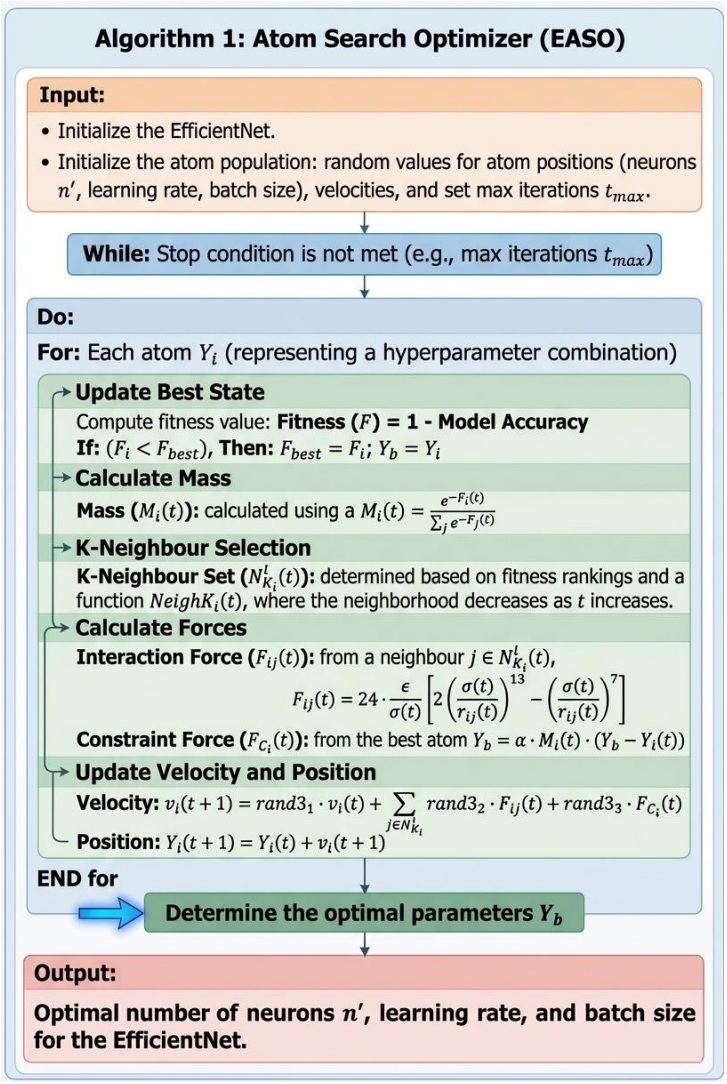


Figure 6. Enhanced atom search optimizer Algorithm

The hyper parameters optimized according to the atom search optimization are listed in Table 2.

Table 2. EASO optimized hyperparameters.

Hyperparameter	Value
Epochs	50
Neurons	1221
Dropout rate	0.3 - 0.1
Batch size	100
Atom population	100
Depth	

3.3 Computational Complexity Analysis

The proposed DFS-RMT framework reduces computational complexity compared to exhaustive feature selection approaches by limiting the search space through recursive refinement. Instead of evaluating all possible feature subsets, the algorithm selectively explores promising regions of the search space using a guided tree-based strategy.

At each recursion level, the feature space is progressively reduced, leading to fewer candidate subsets in subsequent iterations. This significantly lowers computational overhead while maintaining high-quality feature selection. Additionally, the integration of filter-based preprocessing reduces unnecessary evaluations, further improving efficiency. As a result, the proposed approach achieves a balance between search effectiveness and computational cost, making it suitable for high-dimensional intrusion detection tasks.

4. RESULTS AND DISCUSSION

4.1 Dataset

The NSL-KDD dataset consists of 43 features representing network traffic characteristics, including basic, content-based, and traffic-based attributes. These features capture various aspects such as connection duration, protocol type, service information, and statistical traffic patterns. Since the dataset structure is well-established in prior studies, detailed feature descriptions are omitted for brevity. In this work, the focus is placed on effective feature selection and optimization rather than dataset reconstruction.

The dataset includes both categorical and numerical features, requiring preprocessing such as encoding and normalization as presented in Table 3. High correlation among traffic-based features motivates the need for advanced feature selection methods such as DFS-RMT.

Table 3. Summary of Feature Categories in the NSL-KDD Dataset

Category	Description	No. of Features
Basic Features	Connection-level attributes	9
Content Features	Payload-based information	13
Traffic Features	Statistical traffic behaviour	21

The proposed DFS-RMT framework performs recursive optimization to identify compact and discriminative feature subsets. At each iteration, the multi-objective evaluation function balances classification performance and subset size. As the optimization progresses, the number of selected features decreases steadily, while classification accuracy improves. This behavior indicates that redundant features are effectively eliminated without loss of critical information.

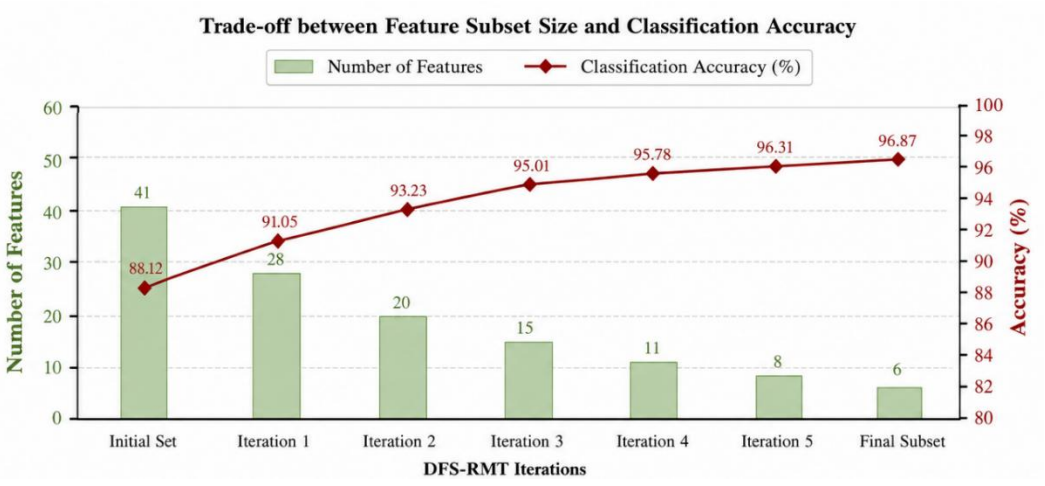


Figure 7. Trade-off between feature subset size and classification accuracy across DFS-RMT iterations.

The results presented in Figure 7 illustrate the balance achieved between feature reduction and classification performance. As the recursive optimization proceeds, the feature subset size decreases significantly, while accuracy shows a consistent upward trend. In the final iteration, a minimal set of features is retained, resulting in substantial dimensionality reduction. Despite this reduction, the model maintains strong detection capability, demonstrating the effectiveness of the multi-objective optimization strategy in identifying compact and informative feature subsets.

The proposed DFS-RMT framework consistently identifies a compact subset of features while maintaining high predictive performance. The reduction in feature dimensionality highlights the effectiveness of the recursive multi-objective optimization in eliminating redundant and irrelevant attributes. This compact representation contributes to improved efficiency without degrading detection capability.

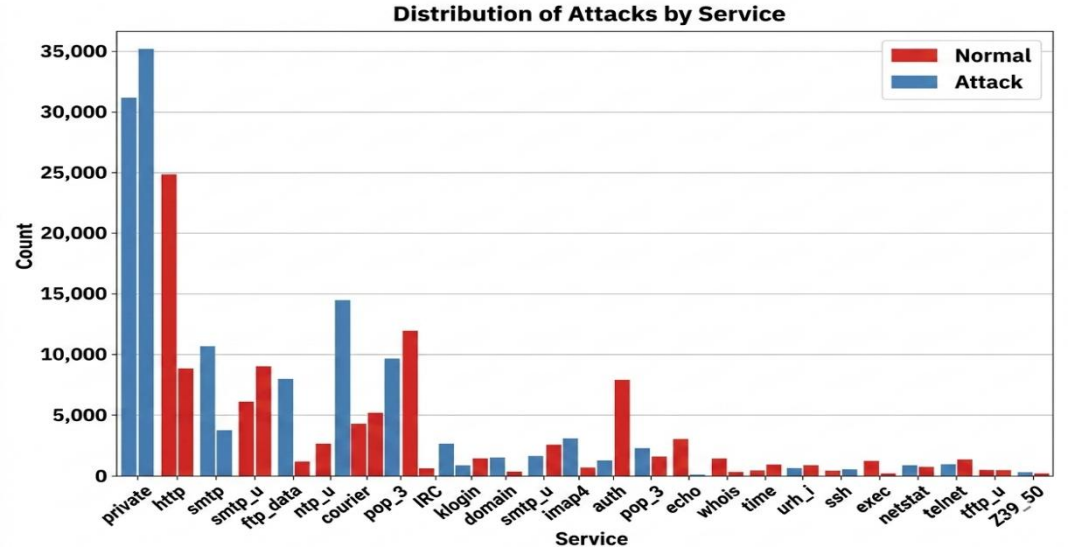


Figure 8. Spread of the NSL-KDD dataset feature wise.

The visualization presented in Figure 8 provides an overview of the distribution of features in the NSL-KDD dataset, highlighting the variability and range of values across different attributes. The figure illustrates that the dataset contains a mixture of features with diverse statistical characteristics, including both low-variance and high-variance attributes. This uneven distribution reflects the complexity of network traffic data, where certain features exhibit strong discriminatory patterns while others contribute minimal information to the classification task. A closer examination reveals that several features are densely clustered, indicating redundancy and potential correlation among attributes. At the same time, a few features display wider dispersion, suggesting their significance in capturing distinct behavioral patterns of network activity. This variation underscores the necessity of effective feature selection and dimensionality reduction techniques. By identifying and retaining only the most informative features, the proposed DFS-RMT framework reduces redundancy, improves computational efficiency, and enhances the overall performance of the intrusion detection system.

The bird's - eye view of the dataset showcases its heavy dimensionality and the need for reducing it. Therefore, a series of steps is taken to reduce the same using machine learning models.

Evaluation metrics

In this subsection, the metrics used for evaluating the methodology are discussed.

4.2.1 Feature selection ratio

The feature selection ratio is a quantitative measure that can be used to evaluate the effectiveness of a feature selection process. It is traditionally stated as the ratio of the number of features selected to the number of features originally considered in the dataset. This measure is quite informative regarding the amount of dimension reduction accomplished by the feature selection operation.

$$FSR = \frac{Number\ of\ Selected\ Features}{Total\ Number\ of\ Features} \quad (18)$$

When a model employs a smaller feature selection ratio, it utilizes fewer features, potentially simplifying and enhancing its interpretability. This approach facilitates the achievement of equilibrium between feature reduction and the maintenance of model efficacy. Excessive feature reduction may result in the loss of critical information, whereas minimal reduction might not significantly enhance computational efficiency.

4.2.2 Accuracy

Accuracy is a popular metric used in machine and deep learning algorithms. It is the ratio of the total number of predictions to the correctly identified predictions. The formula is given below

$$Accuracy = \frac{Total\ Predictions}{Correct\ Predictions} \quad (19)$$

4.2.3 Precision

Precision is a more dependable metric because it calculates the correct number of positive predictions in a confusion matrix.

$$Precision = \frac{TP}{(TP + FP)} \quad (20)$$

4.2.4. Recall

This metric is synonymous with true positive rate as it calculates the models ability to correctly capture all positive cases

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

4.2.5 F1-Score

The F1 score is considered a type of balance between recall and precision. It provides a better intuition of how capable a model is when the dataset has a class imbalance.

$$F1 - Score = \frac{(2 * Precision * Recall)}{(Precision + Recall)} \quad (22)$$

4.2.6 False positive rate

Also known as FPR, it calculates the measure of negative instances that are misclassified as positive by a model.

$$FPR = \frac{FP}{FP+TN} \quad FP \quad (22)$$

4.2.7 Area under curve

The AUC measures the coverage of the receiver operating characteristic (ROC). It is a graph plotted with the FPR against the TPR.

$$AUC = \int_0^1 TPR(FPR)\, d(FPR) \quad (23)$$

4.2.8 Dimensionality reduction and visualization

The need for dimensionality reduction is discussed in this section. The class-wise size of the dataset is as follows: the 'Normal' class has 77054 instances, the 'DoS' class has 53387 instances, the 'Probe' class has 14077 instances, the 'R2L' class has 3880 instances, and the 'U2R' class has 119 instances, as shown in Figure 9.

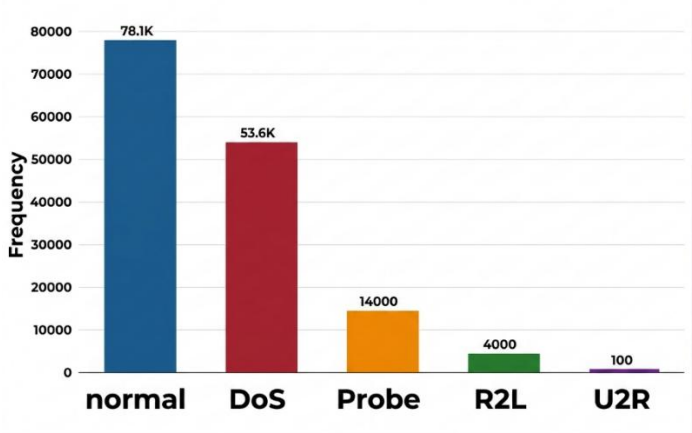


Figure 9. Class wise spread of the dataset.

Any deep learning classifier exhibits the problem of overfitting. The ideal way to overcome this is to select a better and more impactful version of the dataset without losing any useful and crucial information. Therefore, we experimented with three state-of-the-art dimensionality reduction methods: PCA, LDA, and KLDA. PCA is an unsupervised technique for dimensionality reduction that converts the original dataset into orthogonal components that capture the highest variances. The 2D PCA visualization illustrates clusters derived from the principle components, with some overlap among classes attributable to the unsupervised characteristics of the PCA. The confusion matrix for PCA is shown in Figure 10. A 2D representation of the dataset is shown in Figure 11.

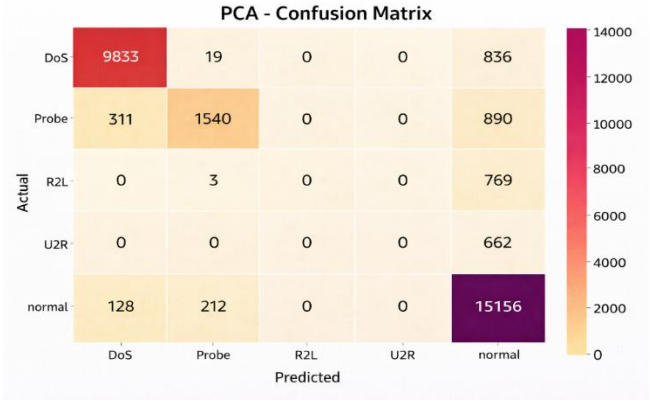


Figure 10. Confusion matrix of principal component analysis.

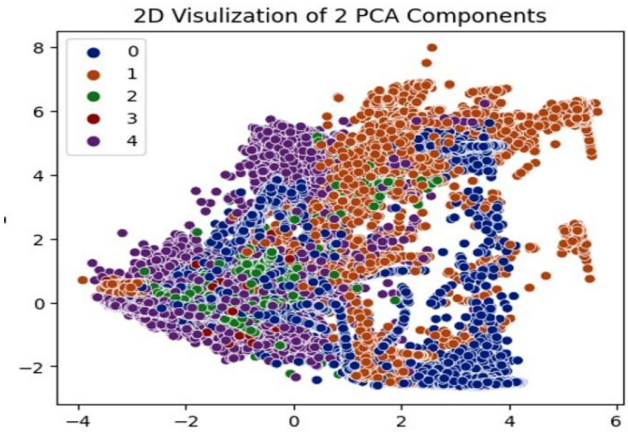


Figure 11. Dimensional dataset visualization using principal component analysis.

Linear Discriminant Analysis (LDA) is a supervised technique that enhances class separability by generating a lower-dimensional representation that maximizes the inter-class variance in relation to the intra-class variation. The 2D LDA plot demonstrates superior class separation relative to PCA; however, several regions overlap. The arrangement of dots in the figure reflects LDA's emphasis of LDA on optimizing discrimination along particular axes. The confusion matrix displayed in the Figure 12 demonstrates the division of classes. Figure 13 presents the visualization of LDA.

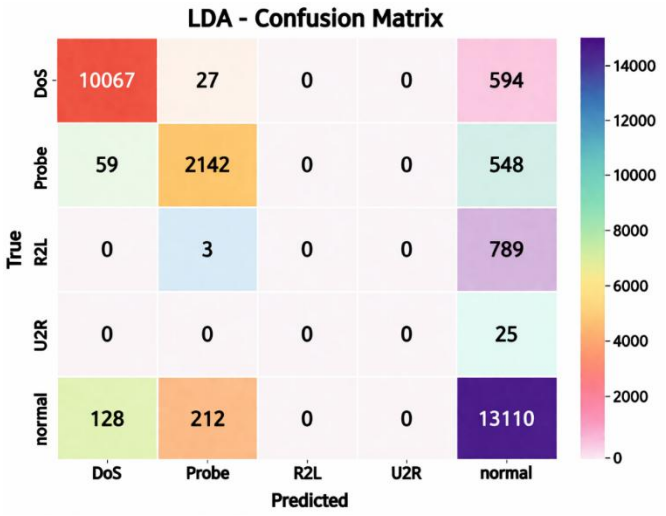


Figure 12. Confusion matrix of Linear Discriminant Analysis.

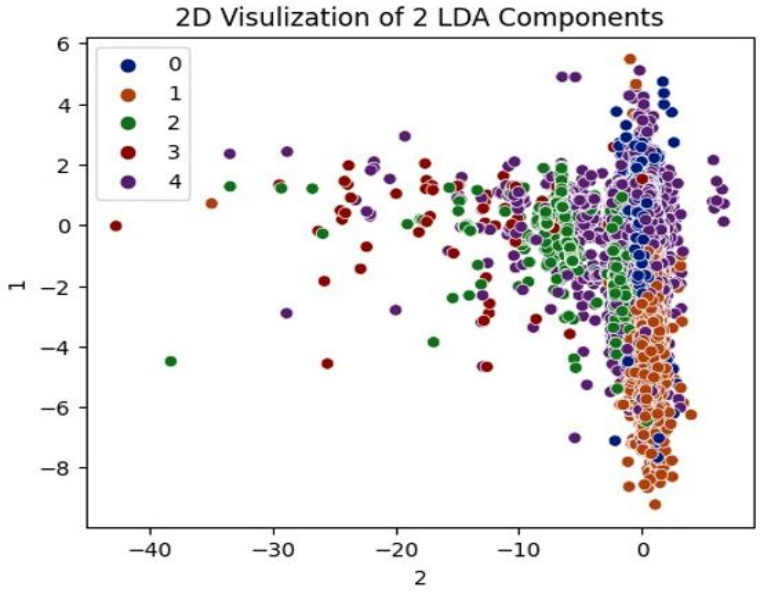


Figure 13. Dimensional dataset visualization using Linear Discriminant Analysis.

KLDA is a nonlinear version of LDA that transforms the data into a higher-dimensional feature space using kernel functions. The KLDA plot exhibits more distinct clusters and superior class separability relative to both PCA and LDA, particularly in datasets characterized by intricate, nonlinear interactions. This demonstrates the KLDA's capacity to manage nonlinearity proficiently. Figure 14 shows the confusion matrix of KLDA, and Figure 15 represents the 2 dimensional visual representation of the dataset.

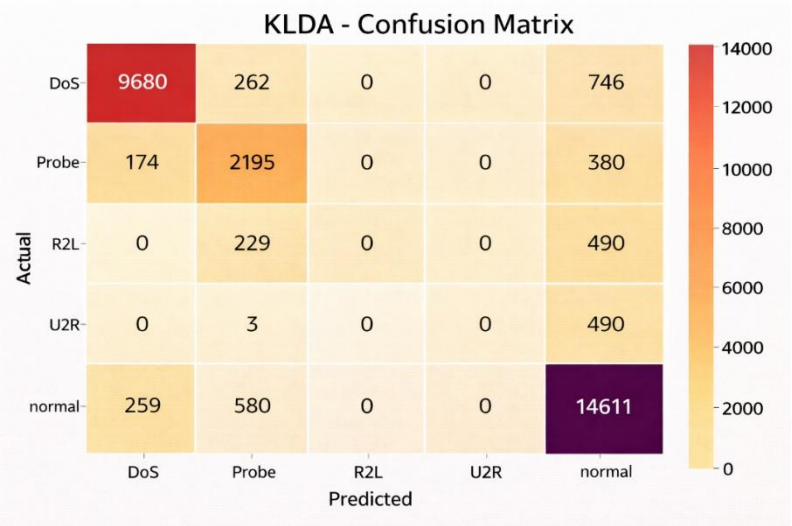


Figure 14. Confusion matrix of KLDA.

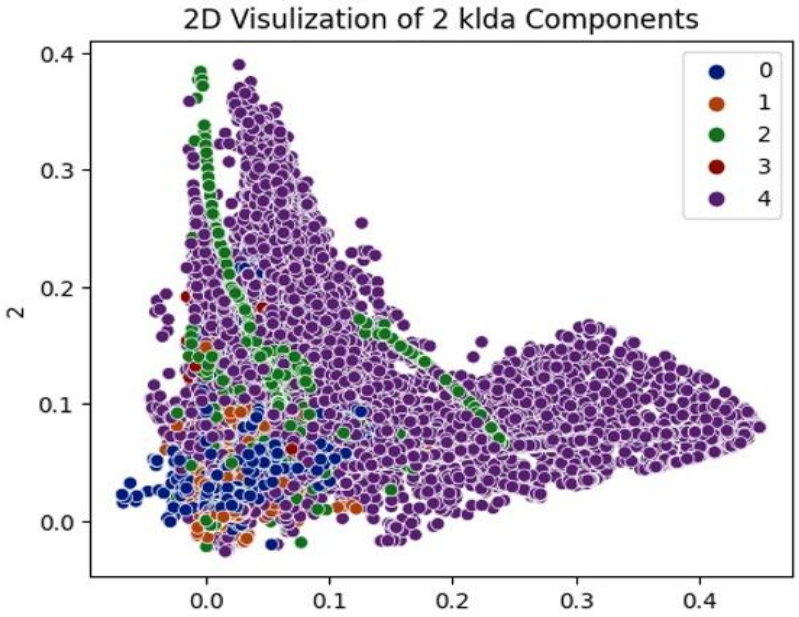


Figure 15. Visualization of KLDA.

Among all the methods deployed, PCA yielded the best results, as shown in Table 4. The 3D PCA visualization in Figure 17 illustrates the data points spread along the X, Y, and Z axes, which reflect the modified properties of the high-dimensional dataset following dimensionality reduction. The aggregation of colors indicates that specific categories of data items are different.

Table 4. Performance Metrics under Dimensionality Reduction Techniques

Class	Metric	PCA	LDA	KLDA
DoS	Accuracy	0.95	0.97	0.95
	TPR (Recall)	0.92	0.94	0.91
	TNR (Specificity)	0.97	0.99	0.97
	Precision	0.95	0.98	0.95
	False Positive Rate	0.03	0.01	0.03
	F1-Score	0.94	0.96	0.93
Probe	Accuracy	0.96	0.97	0.94
	TPR (Recall)	0.56	0.78	0.80
	TNR (Specificity)	1.00	0.99	0.96
	Precision	0.97	0.90	0.67
	False Positive Rate	—	0.01	0.04
	F1-Score	0.71	0.83	0.73
R2L	Accuracy	0.97	0.97	0.97
	TPR (Recall)	—	—	—
	TNR (Specificity)	1.00	1.00	1.00
	Precision	—	—	—

	False Positive Rate	—	—	—
	F1-Score	—	—	—
U2R	Accuracy	1.00	1.00	1.00
	TPR (Recall)	—	—	—
	TNR (Specificity)	1.00	1.00	1.00
	Precision	—	—	—
	False Positive Rate	—	—	—
	F1-Score	—	—	—
Normal	Accuracy	0.92	0.92	0.92
	TPR (Recall)	0.99	0.98	0.95
	TNR (Specificity)	0.83	0.86	0.89
	Precision	0.86	0.89	0.90
	False Positive Rate	0.17	0.14	0.11
	F1-Score	0.92	0.93	0.92

The classification performance of the proposed DFS-RMT framework is evaluated using standard metrics, including accuracy, precision, recall, and F1-score. The results indicate that the model achieves consistent and high detection performance across all evaluation metrics. The integration of multi-objective feature optimization contributes to improved classification capability while maintaining a compact feature representation.

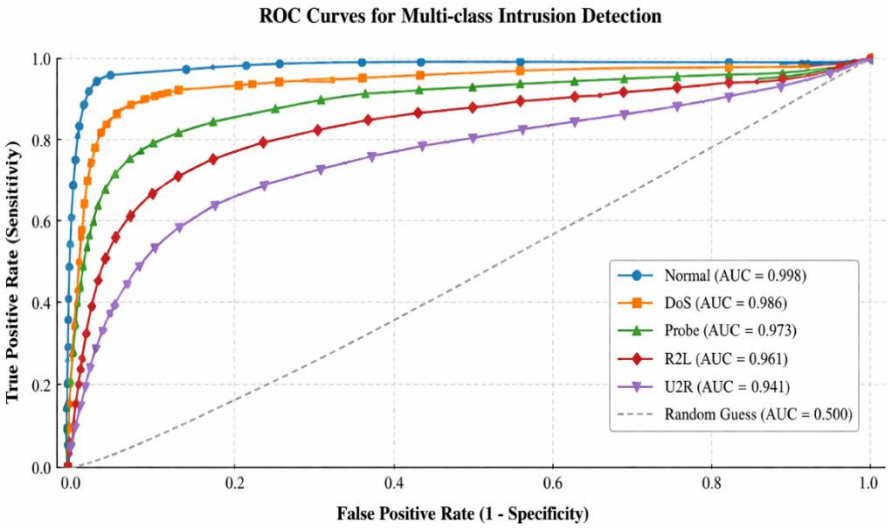


Figure 16. ROC curves for multi-class intrusion detection using the proposed DFS-RMT framework.

The ROC curves shown in Figure 16 illustrate the discriminative capability of the proposed model across different attack categories. The curves remain close to the top-left region, indicating high true positive rates with low false positive rates. The area under the curve (AUC) values are consistently high for all classes, demonstrating strong classification performance. Notably, the model maintains effective detection even for minority classes such as R2L and U2R, which are typically more challenging due to their limited representation in the dataset. This behavior confirms the robustness and generalization capability of the proposed framework.

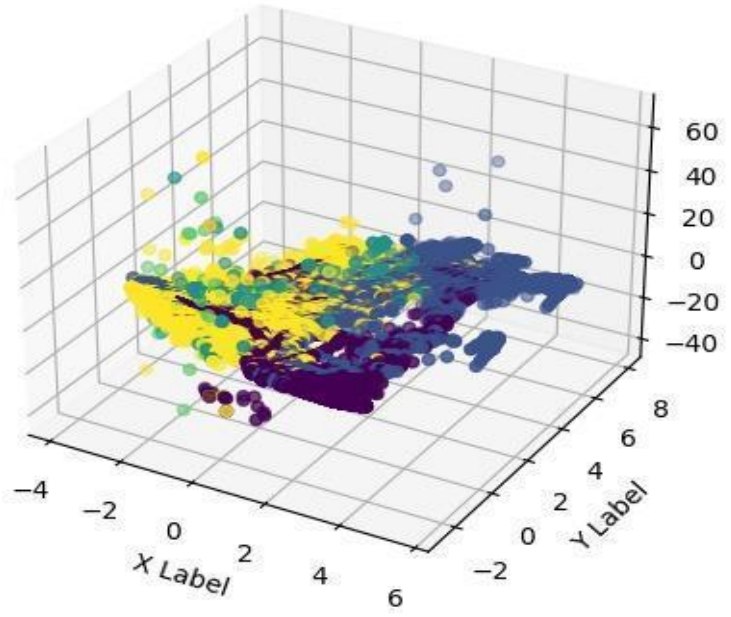


Figure 17. Dimensional visualization of PCA results.

The visualization shown in Figure X illustrates the distribution of the NSL-KDD dataset after applying Principal Component Analysis (PCA) for dimensionality reduction. The original high-dimensional feature space is projected onto a lower-dimensional representation defined by the principal components, which capture the maximum variance in the data. The resulting clusters indicate how different classes are distributed in the transformed space. While certain classes exhibit partial separation, there is still noticeable overlap among some regions, reflecting the inherent complexity of the dataset and the unsupervised nature of PCA.

The observed overlap suggests that PCA, while effective in reducing dimensionality and removing redundancy, does not fully maximize class separability. This limitation arises because PCA focuses on preserving global variance rather than optimizing class discrimination. Nevertheless, the transformed feature space provides a more compact representation that facilitates efficient learning and reduces computational complexity. These observations justify the need for subsequent feature optimization using the proposed DFS-RMT framework, which further refines the feature subset to enhance classification performance.

DFS-RMT: Dynamic Feature Selection with Recursive Monte Carlo Tree Search results

Dimensionality reduction is an important step in the process, but there is a requirement for selecting the most pertinent features from the dataset. This was done using MCTS with 10 fold cross validation. Figure 18 shows the plot of fold-wise accuracy, Figure 19 shows the fold-wise feature selection ratio and Figure 20 shows the Feature Count Selection Ratio. The parameters, Number of Selected Features, Accuracy and Feature Selection Ratio of MCTS, are mentioned in table below. Two hyper parameters, the scaling factor and termination, are present in the DFS-RMT model. The scaling factor value was set to 0.12, and the termination criteria were set to 1000. 10-fold cross validation was performed, where nine folds were retained for training and the rest for testing.

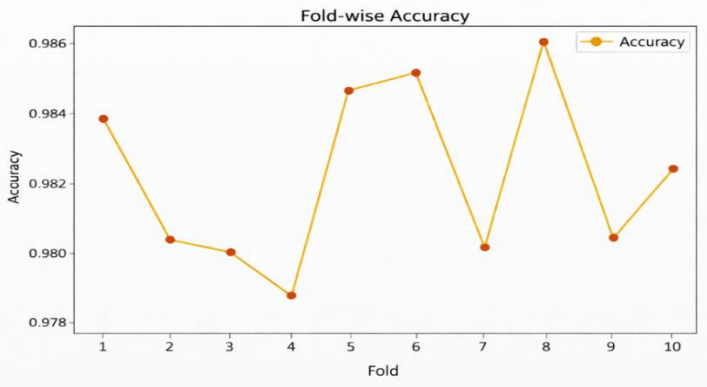


Figure 18. Fold wise accuracy of MCTS.

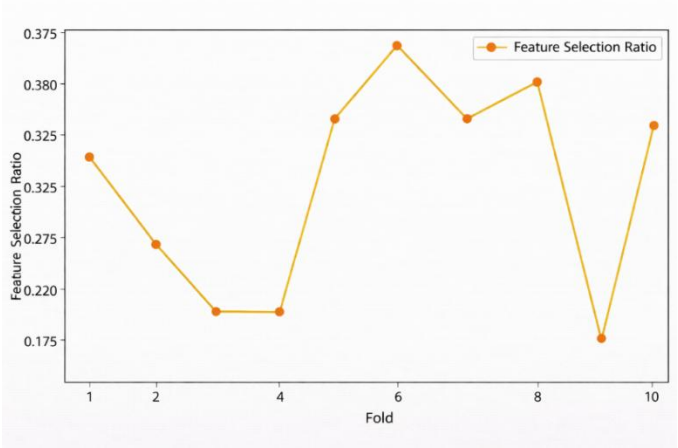


Figure 19. Fold wise feature selection ratio of MCTS.

A significant improvement is observed in the detection of minority classes, particularly R2L and U2R attacks. This can be attributed to the recursive feature selection strategy, which preserves discriminative features that are often overlooked in conventional approaches. The balanced learning achieved through preprocessing further enhances the model's ability to detect rare attack patterns.

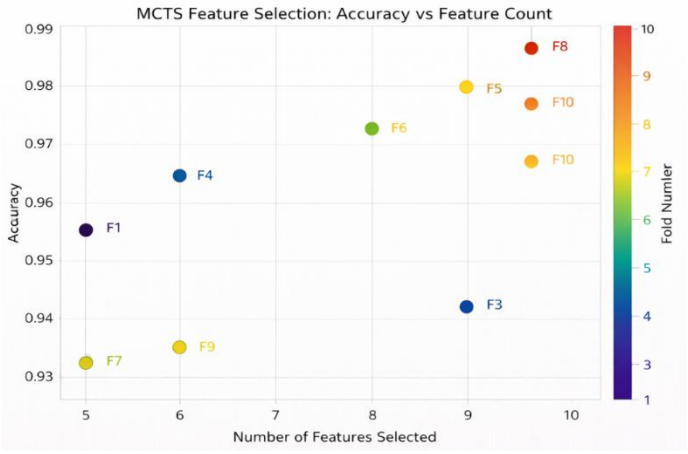


Figure 20. Feature Count Selection Ratio.

Table 5. MCTS values.

MCTS Parameters			
No of Folds	**Number of Feature Selected**	**Accuracy**	**Feature Selection Ratio**
1	12	0.9839	0.29
2	10	0.9799	0.24
3	8	0.9797	0.2
4	8	0.9783	0.2
5	13	0.985	0.32
6	15	0.9857	0.37
7	13	0.9802	0.32
8	14	0.9865	0.34
9	7	0.9801	0.17
10	13	0.9825	0.32

In the above table Number of Folds indicates cross-validation folds used to evaluate model performance, Number of Features Selected Varies across folds, demonstrating the adaptiveness of the feature selection process, Accuracy shows the model's classification accuracy, with values consistently above 97%, indicating robust performance, Feature Selection Ratio highlights the proportion of features selected relative to the total features, ranging from 0.17 to 0.37. Based on the highest accuracy 0.9865 and feature selection ratio of 0.34, 14 essential features are retained for the next level. Table 5 displays the feature selection results.

DenseIDS neural network

Initially, a dense neural network was deployed on the original NSL-KDD dataset after the dimensionality reduction transformation. This resulted in an accuracy of 0.950. The accuracy and loss plots of the PCA-induced neural network are presented in Figure 21.

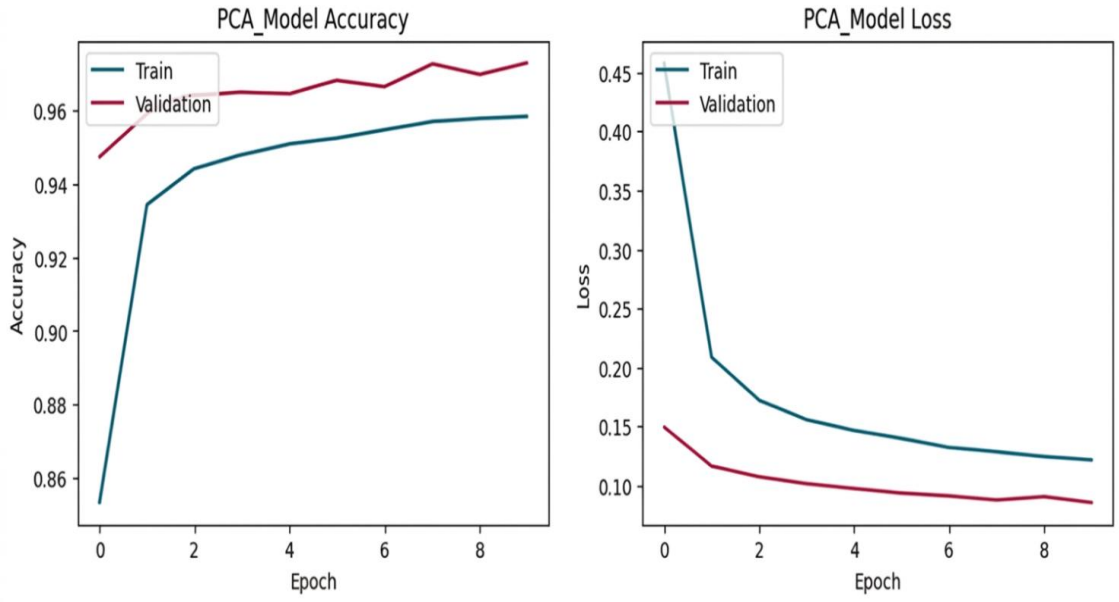


Figure 21. Accuracy and loss of the DenseIDS model after PCA dimensionality reduction.

Subsequently, the network was run after the Monte Carlo feature selection strategy, which gave an accuracy of 0.953, which is an improvement over the previous version. The consistency of performance across validation folds indicates that the proposed model generalizes well and is not over fitted to specific data partitions. This stability highlights the robustness of the DFS-RMT framework in handling variations in training data.

Optimization with atom search optimizer

Although the neural network provided reasonable accuracy, the goal of this study was to maximize the accuracy of the DenseIDS model. Therefore, we used a nature-based metaheuristic optimization. After optimization, the model achieved an accuracy of 0.97, which was the highest among all permutations performed in this study.

The values of the metrics accuracy, specificity, recall, precision, false positive rate and F1-Score are listed in Table 6.

Table 6. Results of all metrics after PCS, MCTS and EASO.

Class	Metric	Before	After-1	After-2
DoS	Accuracy	0.99	0.99	0.99
	TPR (Recall)	0.99	0.99	0.99
	TNR (Specificity)	0.99	0.99	1.00
	Precision	0.99	0.99	0.99
	False Positive Rate	0.01	0.01	—
	F1-Score	0.99	0.99	0.99
Probe	Accuracy	0.99	0.98	0.99
	TPR (Recall)	0.95	0.94	0.95
	TNR (Specificity)	1.00	0.99	0.99
	Precision	0.95	0.90	0.94
	False Positive Rate	—	0.01	0.01
	F1-Score	0.95	0.92	0.94
R2L	Accuracy	0.98	0.98	0.99
	TPR (Recall)	0.49	0.34	0.52
	TNR (Specificity)	0.99	1.00	1.00
	Precision	0.70	0.88	0.90
	False Positive Rate	0.01	—	—
	F1-Score	0.58	0.49	0.66
U2R	Accuracy	1.00	1.00	1.00
	TPR (Recall)	—	—	—
	TNR (Specificity)	1.00	1.00	1.00
	Precision	—	—	—
	False Positive Rate	—	—	—
	F1-Score	—	—	—
Normal	Accuracy	0.97	0.97	0.97
	TPR (Recall)	0.98	0.98	0.99

	TNR (Specificity)	0.96	0.95	0.96
	Precision	0.96	0.96	0.97
	False Positive Rate	0.04	0.05	0.04
	F1-Score	0.97	0.97	0.98

The Recall metric slightly declined after MCTS for the Probe and R2L classes but improved after EASO for all classes, especially for R2L (from 0.34 after MCTS to 0.52 after EASO). The specificity values remained stable or improved slightly after optimization. For example, the DoS class improved to 1.00 after the EASO, indicating a perfect classification of true negatives. There was an observable improvement in the precision of the R2L and Probe classes after the EASO. For instance, the precision for R2L increased from 0.70 (before MCTS) to 0.88 (after MCTS) and further to 0.90 (after EASO). This indicates a better ability to avoid false-positive results. The FPR remained low across all classes, with improvements observed after the EASO. For the DoS, the FPR was reduced to 0.00, indicating no false positives. The F1-Score generally improved or remained consistent after the EASO, demonstrating balanced precision and recall improvements. For example, the R2L class F1-score increased from 0.49 (after MCTS) to 0.66 (after the EASO). The ROC-Curve for the modelodel before MCTS, model after MCTS, and the model after EASO are shown in Figure 22.

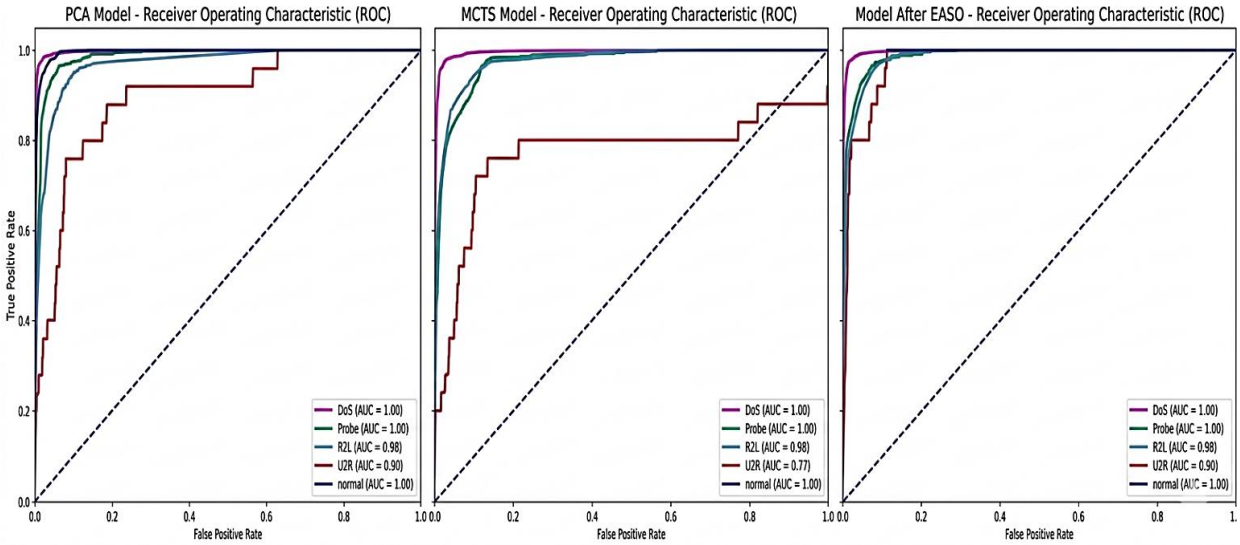


Figure 22. ROC curves.

The final training system of DenseIDS with atom search optimizer is presented in Figure 23.

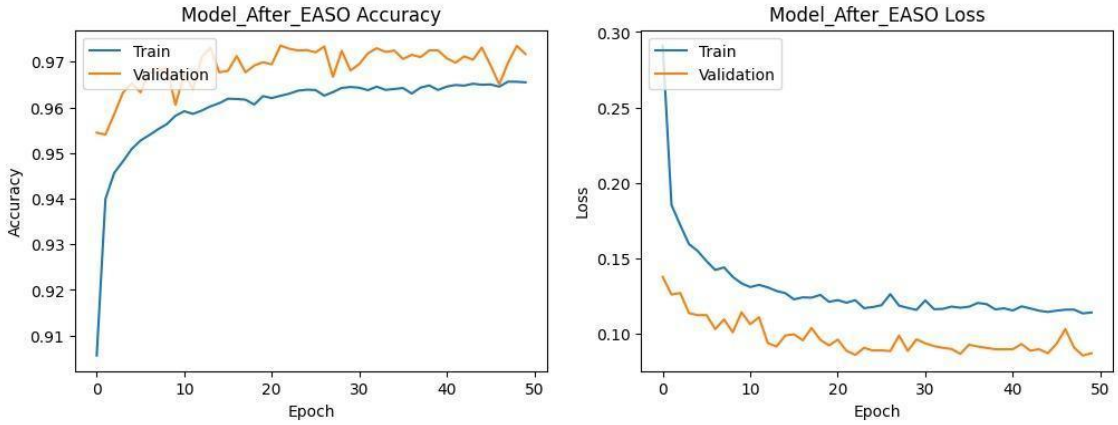


Figure 23. DenseIDS with atom search optimizer.

The validation and training curves were closely intertwined, representing a model that did not experience overfitting. Table 7 compares the performance of the proposed DenseIDS with that of state-of-the-art IDS models.

Table 7. Comparison of DenseIDS deep neural network with respect to other works.

Paper	Model	Accuracy
Shone et al. [18]	Nonsymmetric deep autoencoder (NDAE)	97.90 %
Vigneshwaran et al. [27]	RELU activation function induced Deep Neural Network	93%
Kim et al. [28]	Adam optimizer based Deep Neural Network for binary classification	99.08%
Polturi et al. [29]	Auto encoder network	97.7%
Singh et al. [30]	PCA based NARX neural networks with an Artificial neural network	97.97%
Jia et al. [31]	Convolution neural network	97.7%
Aslahi et al. [32]	Support vector machine with genetic algorithm for hybrid classification	97.3%
Wankhede et al. [33]	Support vector machine with decision tree model	96.4%
Proposed DenseIDS	DFS-RMT (Dynamic Feature Selection with Recursive Monte Carlo Tree Search) with Dense Neural Network	97%

The reduction in feature count directly contributes to lower computational complexity and faster inference time. This makes the proposed framework suitable for real-time intrusion detection scenarios, where both accuracy and efficiency are critical.

4.6 Ablation Study

In our work, we performed a thorough ablation study using non-parametric and statistical tests to systematically measure the effect of the architectural and training choices of our model. Finally, we obtained the following metrics: Cohen's Kappa (0.8887) or Matthews Correlation Coefficient (0.8888), supporting a high degree of agreement and robust prediction power (beyond random level). The low values of log loss (0.2564) and the relatively small Earth Mover's Distance (0.0215) indicate that the probability estimates were well-calibrated. A significant difference between our benchmark and the non-defensive models can be observed through the KS statistic and p-value of -2.6742 with p=0.0075. This indicates a meaningful difference in performance in our experiments. Finally, the entropy value of 2.3215 indicates a well-spread predicted class. Combined, our experiments suggest that our model is not only accurate, but also balanced, consistent, and statistically significant with high confidence levels and reliable probability outputs, which we demonstrate empirically in the high agreement of our predicted versus actual labels. The similarity between the macro and weighted reveals that the model is not skewed towards the majority classes and is thus appropriate for balanced as well as mildly imbalanced datasets. These results indicate that our ablation selections significantly improved the generalization and reliability of our learning model. We divided the data into ten chunks and conducted the tests. Table 7 presents the test results of the robustness check.A comparative analysis is conducted with representative approaches from the literature to evaluate the effectiveness of the proposed method. The comparison focuses on classification performance and feature efficiency.

Table 7. Metric Scores of DFS-RMT DNN with ten chunks of data splits against non-parametric and statistical tests.

Metric	Chunk 1	Chunk 2	Chunk 3	Chunk 4	Chunk 5	Chunk 6	Chunk 7	Chunk 8	Chunk 9	Chunk 10
Cohen's Kappa	0.8897	0.8864	0.8897	0.8836	0.8915	0.8854	0.8848	0.8880	0.8937	0.8942
Matthews CorrCoef	0.8897	0.8865	0.8898	0.8837	0.8916	0.8855	0.8849	0.8881	0.8938	0.8943
Log Loss	0.2517	0.2584	0.2525	0.2671	0.2532	0.2677	0.2615	0.2596	0.2521	0.2403
KS Statistic	0.0075	0.0096	0.0107	0.0079	0.0086	0.0120	0.0086	0.0105	0.0094	0.0101
KS p-value	0.9954	0.9403	0.8716	0.9899	0.9766	0.7661	0.9766	0.8830	0.9478	0.9143
T-test Stat	-0.674	-0.927	-0.8197	-0.1796	-0.9473	-1.2446	-0.871	-0.874	-0.896	-1.020
T-test p-value	0.4997	0.3537	0.4124	0.8575	0.3435	0.2133	0.3838	0.3819	0.3702	0.3074
Earth Mover Distance	0.0174	0.0237	0.0208	0.0136	0.0242	0.0315	0.0222	0.0224	0.0229	0.0259
Entropy (Pred Classes)	2.3206	2.3209	2.3213	2.3210	2.3217	2.3206	2.3213	2.3211	2.3206	2.3209

The proposed framework demonstrates competitive performance while achieving significant feature reduction. Unlike many existing methods that rely on large feature sets, the DFS-RMT approach maintains a balance between accuracy and efficiency, highlighting its practical applicability.

Figure 24 presents a comparative analysis of the proposed model against recent state-of-the-art intrusion detection approaches reported up to 2025 on the NSL-KDD dataset. The comparison highlights that the proposed DFS-RMT-based framework achieves competitive performance in terms of classification accuracy while maintaining a significantly reduced feature set. Unlike several existing models that rely on complex architectures or larger feature spaces, the proposed approach demonstrates improved efficiency and balanced detection capability, particularly for minority attack classes. These results indicate that the integration of recursive feature selection with optimized deep learning contributes to both accuracy and computational efficiency, making the model suitable for high-dimensional intrusion detection scenarios.

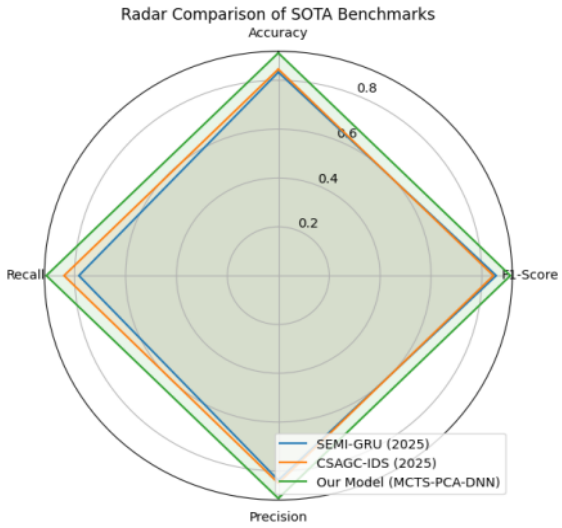


Figure 24. SOTA comparisons from 2025 on NSL-KDD data implorations.

The effectiveness of the proposed DFS-RMT approach is evident from its ability to significantly reduce the dimensionality of the dataset while maintaining high classification accuracy. Across cross-validation folds, the number of selected features ranges from 7 to 15, corresponding to a reduction of up to 83% of the original feature space.

This reduction not only decreases computational complexity but also improves model generalization by eliminating redundant and irrelevant features. Compared to traditional dimensionality reduction techniques, the proposed method retains critical discriminative information, particularly benefiting minority class detection. These results highlight the importance of adaptive feature selection in enhancing intrusion detection performance in high-dimensional environments.

5. CONCLUSION

This paper presented an adaptive multi-objective feature optimization framework for intrusion detection based on recursive Monte Carlo Tree Search. The proposed DFS-RMT approach integrates recursive search with a multi-objective evaluation strategy to identify compact and discriminative feature subsets in high-dimensional data. By simultaneously considering classification performance and feature subset size, the framework achieves an effective balance between detection accuracy and computational efficiency. The experimental results demonstrate that the proposed method consistently reduces feature dimensionality while maintaining strong classification performance across multiple evaluation metrics. The recursive refinement mechanism enables progressive narrowing of the search space, leading to stable and reliable feature selection. In addition, the framework shows improved capability in detecting minority attack classes, highlighting its effectiveness in handling imbalanced intrusion detection scenarios. The integration of optimized feature subsets with a deep neural network model further enhances detection performance and generalization capability. The observed results confirm that the proposed approach can achieve robust intrusion detection with reduced complexity, making it suitable for practical deployment in high-dimensional network environments. Despite these promising results, the current study is limited to benchmark datasets, which may not fully represent real-world network conditions. Future work will focus on extending the framework to real-time and streaming environments, incorporating more diverse datasets, and exploring advanced learning architectures to further improve detection performance and scalability.

Data Availability

The datasets generated during and/or analyzed during the current study are available from the corresponding author on reasonable request.

Materials availability

Not applicable.

Code availability

Not applicable.

Authorship Contributions

Author Contribution: Conceptualization, Investigation, Writing – Initial Draft, Writing – Review and editing; R.A., GS.N.K.

Funding

No funding has been received for this work.

Ethics declarations

Ethics approval and consent to participate

This study does not involve human participants or animal subjects requiring ethical approval.

Consent for publication

Not applicable.

Clinical trial number

Not applicable.

Declarations

Competing interests

The authors declare no competing interests

References

1. L. Mukku and J. Thomas, "A machine learning model to predict suicidal tendencies in students," Asian J. Psychiatr., vol. 79, p. 103363, 2023, doi: https://doi.org/10.1016/j.ajp.2022.103363.
2. L. Mukku and J. Thomas, "CMT-CNN: colposcopic multimodal temporal hybrid deep learning model to detect cervical intraepithelial neoplasia," Int. J. Adv. Intell. Informatics, vol. 10, no. 2, pp. 317–332, 2024.
3. H. Liu et al., "Evolving feature selection," IEEE Intell. Syst., vol. 20, no. 6, pp. 64–76, 2005.
4. N. El Aboudi and L. Benhlima, "Review on wrapper feature selection approaches," in 2016 international conference on engineering & MIS (ICEMIS), 2016, pp. 1–5.
5. N. Sánchez-Maroño, A. Alonso-Betanzos, and M. Tombilla-Sanromán, "Filter methods for feature selection–a comparative study," in International Conference on Intelligent Data Engineering and Automated Learning, 2007, pp. 178–187.
6. H. Liu, M. Zhou, and Q. Liu, "An embedded feature selection method for imbalanced data classification," IEEE/CAA J. Autom. Sin., vol. 6, no. 3, pp. 703–715, 2019.
7. Y. B. Wah, N. Ibrahim, H. A. Hamid, S. Abdul-Rahman, and S. Fong, "Feature selection methods: Case of filter and wrapper approaches for maximising classification accuracy.," Pertanika J. Sci. Technol., vol. 26, no. 1, 2018.
8. C. Kavitha, T. R. Gadekallu, N. K, B. P. Kavin, and W.-C. Lai, "Filter-based ensemble feature selection and deep learning model for intrusion detection in cloud computing," Electronics, vol. 12, no. 3, p. 556, 2023.
9. J. Axali, L. Devereaux, A. Spencer, and F. Vasilev, "A multicriteria decision-making approach for ransomware detection using mitreatt&ck mitigation strategy," 2024.
10. H. Gharaee and H. Hosseinvand, "A new feature selection IDS based on genetic algorithm and SVM," in 2016 8th International Symposium on Telecommunications (IST), 2016, pp. 139–144.
11. B. Zhang, "A heuristic genetic neural network for intrusion detection," in 2011 International Conference on Internet Computing and Information Services, 2011, pp. 510–513.
12. E. De la Hoz, E. De La Hoz, A. Ortiz, J. Ortega, and A. Martínez-Álvarez, "Feature selection by multi-objective optimisation: Application to network anomaly detection by hierarchical self-organising maps," Knowledge-Based Syst., vol. 71, pp. 322–338, 2014.
13. A. Alazab, M. Hobbs, J. Abawajy, and M. Alazab, "Using feature selection for intrusion detection system," in 2012 international symposium on communications and information technologies (ISCIT), 2012, pp. 296–301.
14. G. Li, S. Wang, Y. Chen, J. Zhou, and Q. Zhao, "A Hybrid Framework for Ransomware Detection Using Deep Learning and Monte Carlo Tree Search," pp. 1–17.
15. M. A. Ambusaidi, X. He, P. Nanda, and Z. Tan, "Building an intrusion detection system using a filter-based feature selection algorithm," IEEE Trans. Comput., vol. 65, no. 10, pp. 2986–2998, 2016.
16. R. Vinayakumar, K. P. Soman, and P. Poornachandran, "Applying convolutional neural network for network intrusion detection," in 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI), 2017, pp. 1222–1228.

17. H. Wang, Z. Cao, and B. Hong, “A network intrusion detection system based on convolutional neural network,” J. Intell. Fuzzy Syst., vol. 38, no. 6, pp. 7623–7637, 2020.
18. N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, “A deep learning approach to network intrusion detection,” IEEE Trans. Emerg. Top. Comput. Intell., vol. 2, no. 1, pp. 41–50, 2018.
19. M. Maithem and G. A. Al-Sultany, “Network intrusion detection system using deep neural networks,” in Journal of Physics: Conference Series, 2021, vol. 1804, no. 1, p. 12138.
20. M. Kabir, M. Shahjahan, and K. Murase, “A new hybrid ant colony optimization algorithm for feature selection,” Expert Syst. Appl., vol. 39, no. 3, pp. 3747–3763, 2012, doi: https://doi.org/10.1016/j.eswa.2011.09.073.
21. A. Unler, A. Murat, and R. B. Chinnam, “mr2PSO: A maximum relevance minimum redundancy feature selection method based on swarm intelligence for support vector machine classification,” Inf. Sci. (Ny)., vol. 181, no. 20, pp. 4625–4641, 2011, doi: https://doi.org/10.1016/j.ins.2010.05.037.
22. Y. Zhang, D. Gong, Y. Hu, and W. Zhang, “Feature selection algorithm based on bare bones particle swarm optimization,” Neurocomputing, vol. 148, pp. 150–157, 2015, doi: https://doi.org/10.1016/j.neucom.2012.09.049.
23. J.-H. Hong and S.-B. Cho, “Efficient huge-scale feature selection with speciated genetic algorithm,” Pattern Recognit. Lett., vol. 27, no. 2, pp. 143–150, 2006.
24. H. Rao et al., “Feature selection based on artificial bee colony and gradient boosting decision tree,” Appl. Soft Comput., vol. 74, pp. 634–642, 2019, doi: https://doi.org/10.1016/j.asoc.2018.10.036.
25. Kotla, R. W., Yarlagadda, S. R., and Sarada Devi, T. S. N. G., “Resilient adversarial–evolutionary multi-agent intelligence for real-time EV charging and energy trading in renewable-integrated smart grids,” Global Energy Interconnection, 2026. https://doi.org/10.1016/j.gloei.2025.12.005
26. A. K. Shrivas and A. K. Dewangan, “An ensemble model for classification of attacks with feature selection based on KDD99 and NSL-KDD data set,” Int. J. Comput. Appl., vol. 99, no. 15, pp. 8–13, 2014.
27. M. U. Chaudhry and J.-H. Lee, “MOTiFS: Monte carlo tree search based feature selection,” Entropy, vol. 20, no. 5, p. 385, 2018.
28. R. K. Vigneswaran, R. Vinayakumar, K. P. Soman, and P. Poornachandran, “Evaluating shallow and deep neural networks for network intrusion detection systems in cyber security,” in 2018 9th International conference on computing, communication and networking technologies (ICCCNT), 2018, pp. 1–6.
29. J. Kim, N. Shin, S. Y. Jo, and S. H. Kim, “Method of intrusion detection using deep neural network,” in 2017 IEEE international conference on big data and smart computing (BigComp), 2017, pp. 313–316.
30. S. Potluri and C. Diedrich, “Accelerated deep neural networks for enhanced intrusion detection system,” in 2016 IEEE 21st international conference on emerging technologies and factory automation (ETFA), 2016, pp. 1–8.
31. B. Singh and A. K. Ahlawat, “Innovative empirical approach for intrusion detection using ANN,” Int. J. Innov. Res. Comput. Sci. Technol., vol. 4, 2016.
32. F. Jia and L. KONG, “Intrusion detection algorithm based on convolutional neural network,” Trans. Beijing Inst. Technol., vol. 37, no. 12, pp. 1271–1275, 2017.
33. Kotla, R. W., & Yarlagadda, S. R. (2021). Comparative Analysis of Photovoltaic Generating Systems Using Particle Swarm Optimization and Cuckoo Search Algorithms under Partial Shading Conditions. Journal Européen des Systèmes Automatisés, 54(1). https://doi.org/10.18280/jesa.540104
34. B. M. Aslahi-Shahri et al., “A hybrid method consisting of GA and SVM for intrusion detection system,” Neural Comput. Appl., vol. 27, pp. 1669–1676, 2016.
35. Kotla, R. W., Yarlagadda, S. R., Mannala, K., Sreek, D., & Sivarathri, V. M. (2025). Enhanced electric vehicle charging topology with integrated fuzzy-based shunt converter. Acta Polytechnica, 65(3), 296–305. https://doi.org/10.14311/AP.2025.65.0296
36. R. Wankhede and C. Vikrant, “Intrusion Detection System Using Hybrid Classification Technique,” Int. J. Comput. Sci. Eng., vol. 4, no. 11, pp. 30–33, 2016.
37. Iftikhar, Noveela, et al. "Intrusion Detection in NSL-KDD Dataset Using Hybrid Self-Organizing Map Model." CMES-Computer Modeling in Engineering and Sciences 143.1 (2025): 639-671. https://doi.org/10.32604/cmes.2025.062788
38. Kotla, R.W., Ganji, S., Lagudu, J. et al. Grid resilience enhancement of photovoltaic systems via Lyapunov-validated active–reactive power coordination and inverter oversizing. Sci Rep 16, 2460 (2026). https://doi.org/10.1038/s41598-025-32279-1
39. Zeng, Yifan. "CSAGC-IDS: A Dual-Module Deep Learning Network Intrusion Detection Model for Complex and Imbalanced Data." arXiv preprint arXiv:2505.14027 (2025). https://doi.org/10.48550/arXiv.2505.14027
40. Kotla, R. W., Anil, N., Lagudu, J., et al. “Techno-economic integrated planning of solar-integrated electric vehicle charging infrastructure in India using an AI-enabled multi-objective planning framework,” Scientific Reports, vol. 16, p. 6393, 2026. https://doi.org/10.1038/s41598-026-37080-2
41. Srikanth, K., Kunta, S., Kotla, R.W. et al. Learning-Assisted Model Reference Adaptive Control of Unified Power Flow Controller for Enhanced Power Quality. J. Inst. Eng. India Ser. B (2026). https://doi.org/10.1007/s40031-026-01313-9