Journal of Information Systems Engineering and Management
2025, 10(51s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Towards Secure and Resilient MANETs: A SHIELD-Based Strategy for Application-Layer DDoS Prevention and Quality of Service Optimization

Prof. Kiran M. Salunke[1]*, Dr. Suresh Kurumbanshi[2]
[1,2] Department of Computer Engineering, MPSTME, Shirpur Campus, India
** Corresponding Author: Kiran.salunke@nmims.edu*

ARTICLE INFO	ABSTRACT

Received: 29 Dec 2024
Revised: 15 Feb 2025
Accepted: 24 Feb 2025

Mobile Ad Hoc Networks (MANETs) offer flexible infrastructure-free communication which makes them suitable for dynamic environments. MANETs experience extreme vulnerability to application-layer Distributed Denial of Service (DDoS) attacks because attackers can easily replicate normal traffic growth patterns during flash events. The enhanced SHIELD framework uses machine learning and optimization techniques to identify and counteract the detected problem. The Xavier Soft-plus Convolutional Neural Network (CNN) achieves an accuracy rate of 98.96% when detecting anomalies with high precision. The Brownian Motion-enhanced Harris Hawks Optimization (BM-HHO) algorithm uses optimization techniques to select features in limited resource scenarios. The framework employs an I-CMIWO method which combines Improved Crossover Mutation to achieve intelligent load balancing and adaptive traffic control. The framework implements Dynamic Spectrum Resource Control (DSRC) as a mechanism to perform real-time threat mitigation after threat detection for network recovery purposes. Pearson correlation serves as the basis for analyzing traffic patterns to distinguish between malicious activity and genuine flash crowds because it provides high reliability. The proposed system shows superior performance in maintaining Quality of Service (QoS) while reducing false positives and ensuring network resilience. Experimental results demonstrate that the framework detects application-layer DDoS attacks in real-time operation across dynamic MANET environments and performs efficient flash event differentiation and mitigation.

Keywords: Tunnel Infrastructure, Misalignments, Detection Techniques, Safety, Operational Efficiency.

INTRODUCTION

The main security concern in Mobile Ad Hoc Networks (MANETs) exists because there is no central control point as shown in Figure 1 [11]. The absence of fixed infrastructure makes it challenging to handle node connections and network monitoring activities [6] The network faces dual threats from internal malicious nodes that drop packets and external denial-of-service attacks. The accumulation of these threats throughout time leads to substantial degradation of network performance. The most destructive attacks in MANETs include black hole, wormhole, jellyfish and DoS and DDoS attacks. The system becomes overwhelmed by request floods until genuine users lose access to services [7][8].

Machine learning has helped in some ways. Researchers have implemented RNNs, ANNs and CNNs as models to determine user normal behavior. The implemented models demonstrate effective capabilities in detecting malicious network traffic. Detection systems present a major problem because they do not solve the issue of load balancing. The system's inability to distribute traffic evenly leads to node overload which results in performance degradation or system failures even when attacks are detected in time [9].

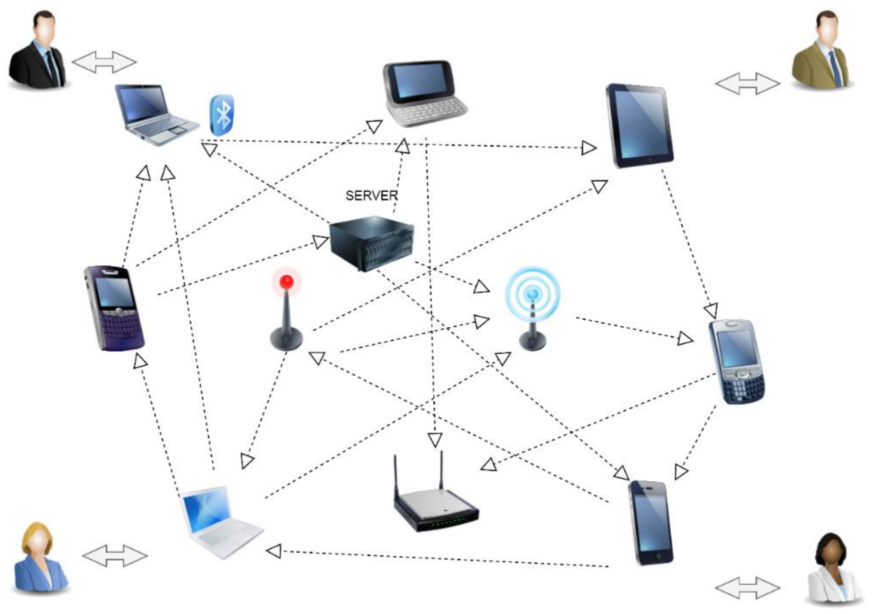

Figure 1. MANET

The identification of DDoS attacks in MANETs proves to be more complicated than people might think. The improper distribution of legitimate user requests creates system failures that resemble actual attacks even when no attack exists. The main challenge emerges from identifying authentic network traffic among fake data. The fast-changing nature of MANET environments makes it difficult to distinguish between actual DDoS attempts and normal traffic spikes because fixed thresholds fail to provide reliable results. Our solution combines multiple concepts to solve this problem. Our approach begins with Xavier Soft-Plus Convolutional Neural Network which performs the initial classification task.

The system combines IP spoofing detection with this feature to eliminate fake source traffic during the early stages. Our customized Invasive Weed Optimization method tracks user activity changes through linear-layer crossover mutation while following Pearson correlation guidance. The model adjusts instantly through this approach which improves its ability to distinguish between actual flash crowds and attack traffic. The system enhances IP traceback accuracy through spoofed address reduction which enables better resource management [10] [31].

This paper follows a specific organization in its remaining sections. Section 1 asylums the introduction. Section 2 covers the most relevant related work. Section 3 describes our detection and load-balancing methodology step by step. Section 4 demonstrates the model's performance across various operational scenarios. Section 5 summarizes the research conclusions and research opportunities and potential enhancements to this work.

LITERATURE REVIEW

This section includes a review of existing approaches in this field. The development of MANET stanches from recent advancements in wireless communication technology and portable devices [1]. MANET functions as a wireless network of mobile devices which automatically configures itself for self-organization. The nodes of MANETs maintain built-in routing capabilities to support automation and disaster recovery and reliable military communication [2]. MANET nodes actively participate in discovering neighboring nodes dynamically to establish a dynamic network for packet movement between sources and destinations [3]. The security of mobile node communication requires essential attention. The absence of centralized access points makes it essential to manage node membership and monitor node behavior within MANETs [4]. Security issues in MANETs occur because of malicious and selfish node attacks which lead to discarded data-carrying packets. Network nodes experience communication disruptions as a result of this situation [5].

We have investigated deep learning-based approaches for detecting DDoS attacks in different network settings and these methods will be tested against proposed X-SPCNN for accuracy. Yousuf and Mir [26] used recurrent neural networks (RNN) to detect DDoS attacks in IoT systems. Shen [27] developed an intrusion detection system through the combination of Deep Belief Networks (DBN) with three-way decision theory. Kumar et al. [28] presented a lightweight CNN model which works for DDoS detection in distributed settings. Makuvaza et al. [29] created a real-time DNN-based solution which is specifically designed for SDN architectures to improve DDoS threat response. The most recent advancements in intrusion and DDoS detection systems combine bio-inspired optimization techniques with deep learning methods to improve both detection precision and feature selection capabilities. Mohammed [30] applied invasive weed optimization to detect ransomware threats in cloud systems. Singh and Jayakumar [31] introduced an optimized deep CNN model with feature refinement for detecting DDoS in SDNs. Xu et al. [32] used improved butterfly and black widow optimization algorithms to select features efficiently for intrusion detection systems. Khare et al. [33] created a hybrid SMO-DNN model which combined spider monkey optimization with deep neural networks to achieve robust intrusion classification and these methods will be tested against projected I-CMIWO for Optimized QoS Improvement.

Table 1 illustrates the examined approaches together with their operational areas and limitations. Recent approaches have implemented alternative optimization strategies for attack detection yet these methods produce different performance trade-offs. Hussain Mohammed [41] used invasive weed optimization to detect cloud ransomware in his research. The method demonstrated potential application but system size scalability became a major issue when the system expanded. Hui Xu et al. [43] tested Butterfly optimization and Black Widow optimization as methods for selecting intrusion features in their study. The model achieved successful parameter identification yet its real-time response capabilities remained limited. SMO-DNN represents a new approach by Albakri, et al. [44] which merges spider monkey optimization techniques with deep neural networks. The detection rates were solid, but performance dropped off once the network became large and more complex. N. Sivanesan and his team [45] developed an Extra Tree Classifier system with randomized search capabilities to detect different types of DDoS attacks in MANETs. The model achieved 98.89% accuracy through hyper parameter tuning but required complex tracking of malicious node activity in AODV routing. Gautam et al. [39] implemented an anomaly-based traffic system which monitored networks live and used threshold parameters for detection. The system achieved better accuracy and specificity results through its classification methods but the evaluation only included synthetic data which creates uncertainty about its real-time deployment performance.

The recent developments in DDoS detection and mitigation strategies employ multiple intelligent techniques which are specifically designed for dynamic and decentralized environments such as SDN, cloud networks and MANETs. Zhang and Jinsong [15] developed a hybrid detection system which combined entropy features with SSAE-SVM to detect DDoS attacks in SDN. Aydın et al. [16] developed an LSTM-based framework to detect and defend against DDoS attacks in public cloud networks. Alam and Raj [17] created an SVM-based DEHO classifier which improves detection efficiency in cloud environments. Fouladi et al. [18] developed a DWT and autoencoder-based mechanism to detect and counter DDoS threats in SDN. Batchu and Seetha [19] developed a generalized machine learning model that combined hybrid feature selection with hyperparameter optimization for robust DDoS detection. Salunke and Ragavendran [20] performed a detailed evaluation of shield techniques which focus on application-layer DDoS threats in MANETs. Shalini et al. [21] introduced DOCUS which is a modified CUSUM-based detection system that distinguishes flash traffic from DDoS in SDN to enhance mitigation efficiency.

Recent frameworks have advanced the field but their current limitations remain in operating within highly mobile or constrained environments. Some contemporary studies also focused on DDoS detection in specific network environments through various algorithmic frameworks. Anyanwu et al. [34] created an RBF-SVM kernel-based model which specifically targets DDoS detection in vehicular networks that use SDN integration. Sharma et al. [35] developed an anomaly detection framework for fog-enabled IoT networks to fight against DDoS threats. Kalathiripi and Venkatram [36] presented the RCTFM approach which uses traffic flow metrics regression coefficients for IoT ecosystem DDoS defense. The main problem with current approaches is their failure to unite lightweight pre-processing with adaptive mitigation techniques that maintain low latency during real-time operations. The majority of feature selection approaches fail to address scalability concerns. The increase in node density leads to processing unnecessary data which results in wasted resources. There's also the question of mobility. MANET networks require better responses to changing conditions through adaptive metrics and PCC-guided thresholds yet these concepts remain underdeveloped for fast-moving MANET scenarios. The main research gap consists of insufficient testing under simulated link failures.

Testing without this standard makes it difficult to determine actual model resilience levels. The solution of these problems would result in significant improvements for building MANET frameworks that maintain scalability and energy efficiency during network pressure. Our proposed model unites an enhanced adaptive system for MANET DDoS attack detection and mitigation which includes protection against flash event misclassification. The research focuses on three main areas to enhance detection precision and improve traffic distribution and fix previous model weaknesses during network changes and resource limitations.

Table 1. Prevailing Approaches for Safeguarding MANET

Authors	Approach	Strength	Weakness
Sokkalingam et al. [22]	SVM parameters are optimized for an IDS to detect DDoS.	Improved SVM performance.	Lack of optimization for scalability in large networks.
Raveendranadh et al. [23]	Attack detection using exponential polynomial kernel-centered deep neural network.	Good performance in resource-constrained environments.	High computational complexity due to the depth of learning.
Kucukkara et al. [43]	QNN-based model for DDoS attack detection.	Terminated training with only 7 features for classification.	Limited accuracy due to constraints in the simulator.
Albakri et al. [44]	A swarm optimization algorithm, Rock Hyrax, for improving the effectiveness of IDS in detecting DDoS.	Improves the optimization of DDoS detection.	Has a problem with local optimization in dynamic networks.
Kaviarasan et al. [24]	Monarch Butterfly Optimization for load balancing in the cloud.	Good cloud load balancing.	Not very suitable for real-time applications.
H. Beitollahi et al. [25]	Cuckoo search algorithm-trained radial basis function for identification of application-layer DDoS attacks.	Highly robust detection at the application layer.	Does not perform well in highly mobile node environments.
Abdullah Emir Cilet et al. [7]	DNN-based model trained on CICDDoS2019 dataset.	Detection rate is 93.99%.	Needs a large amount of training data.
Neha Agrawal & Shashikala Tapaswi [8]	A frequency-domain approach using Power Spectral Density (PSD) for the identification of LDDoS.	3.7% false positive rate, 4.9% false negative rate.	High computational complexity.
P. Chandra Sekar & H. Mangalam [9]	Load balancing in MANET with the help of 3rd-generation mobility.	Good for MANET networks.	Necessary to recompile the source code when the user's code is changed.

Sengathir Janakiraman et al. [10]	Proposed an integrated context-centered mitigation framework (ICMF) for the protection of MANET.	Enhanced network performance with GTIDF and ICMF.	Failed to counter attacks on root nodes.
Bhuvaneswari et al. [11]	A method for detecting DoS attacks using fictitious node verification in OLSR-based MANET.	Good for low-mobility networks.	Inapplicable to real-time and high-mobility networks.
M. Anbarasan et al. [12]	A secured DDoS attack prevention technique based on encryption for MANETs.	Decreased memory requirement in trusted MANETs.	Does not extend well to larger networks.
M. Islabudeen & M.K. Kavitha Devi [13]	SA-IDPS: Intrusion Detection and Prevention System based on Security Awareness and Machine Learning for DoS, Probe, U2R, and R2L attacks.	All-around detection and prevention.	The complexity of the model increases with the amount of data.
Mukul Shukla et al. [14]	Elliptic Curve Cryptography (ECC) for the protection of MANET against Wormhole/Black Hole attacks.	High security with a limited number of nodes.	The computational cost is high, especially in large networks.
Toklu & Simsek [42]	Dual-layer High-Rate detection system.	Effective at distinguishing valid from attack traffic.	Fails when multiple attackers are involved.

METHODOLOGY

This phase integrates the fundamental elements of our X-SPCNN classification model with DSRC mitigation capabilities and I-CMIWO optimizer tuned by Pearson correlation for detecting and responding to DDoS attacks. K-Means clustering joins the system to enhance the separation of flash events from actual attacks. The system operates through two primary stages that start with abnormal behavior detection followed by mitigation activation. X-SPCNN operates as the detection tool to analyze incoming user requests and identify them as valid or invalid. After processing the results through PCC-guided I-CMIWO optimizer we use DSRC to execute response and mitigation actions. The simplified system architecture appears in Figure 2. A random mobility model became part of the framework to improve the matching of MANETs' dynamic characteristics. The simulation aimed to duplicate actual node movements together with their associated network disruptions including link failures and intermittent connections. The simulation included multiple active nodes with link failures affecting approximately 10% of nodes simultaneously. The dataset received a new “Node Mobility” feature which applied to both training and testing phases. The system received evaluation for its response capabilities to network topology changes. The addition of this feature enhanced the model's resistance to real-world MANET network conditions where nodes frequently enter or exit and links form or break suddenly.

3.1 DDOS Attack Detection Phase

In the classifier training step, a DDoS dataset is used to train the model to classify whether a particular request by the user can be classified as legitimate or non-legitimate. The training phase involves several processes such as pre-processing, feature extraction, feature selection, and classification. Data from the DDoS dataset were pre-processed and prepared for the training classifier. The input data acquired as of the DDOS dataset is provided by,

$$\boldsymbol{S} = \{\boldsymbol{x_1}, \boldsymbol{x_2}, \dots, \boldsymbol{x_M}\} \tag{1}$$

The initial dataset S contains M data samples from $\boldsymbol{x_1}$ **to** $\boldsymbol{x_M}$, according to this equation. It lists all input records that will undergo subsequent pre-processing steps in the DDoS detection pipeline.

$$\boldsymbol{U} = \{\boldsymbol{x'_1}, \boldsymbol{x2'}, \dots, \boldsymbol{x_{Mu}}'\} \tag{2}$$

The dataset U contains Mu unique records $\boldsymbol{x'_1}, \boldsymbol{x2'}, \dots, \boldsymbol{x_{Mu}}'$. The number of remaining non-duplicate data points is denoted by $\boldsymbol{M_u} \le \boldsymbol{M_M}$, which represents the effect of the deduplication process.

$$\boldsymbol{V} = \{\boldsymbol{y_1}\ , \boldsymbol{y_2}, \dots, \boldsymbol{y_{Mu}}\} \tag{3}$$

All non-numeric attributes are converted, yielding the numeralized dataset $\boldsymbol{V}$ with $\boldsymbol{M_u}$ elements. The $\boldsymbol{Y_i}$ corresponds to a preprocessed record expressed entirely in numerical form, making the data suitable for input into the classification model.

$$\boldsymbol{z_{ij}} = \frac{\boldsymbol{v_{ij}} - \boldsymbol{\mu_j}}{\boldsymbol{\sigma_j}} \tag{4}$$

Each numeric feature value $\boldsymbol{V_{ij}}$ is standardized to $\boldsymbol{Z_{ij}}$ by subtracting the mean µj of feature j and dividing by its standard deviation $\boldsymbol{\sigma_j}$. This Z-score normalization centers the feature distribution (zero mean) and scales it to unit variance, which helps accelerate and stabilize the training process.

$$\boldsymbol{F} = \{\boldsymbol{f_1}, \boldsymbol{f_2}, \dots, \boldsymbol{f_K}\} \tag{5}$$

The set F represents the collection of extracted features. The set F contains K features $\boldsymbol{f_1}, \boldsymbol{f_2}, \dots, \boldsymbol{f_{K,}}$ which were determined as the most relevant after feature extraction to create a compact representation of the data for the learning algorithm.

Case 1

$$\boldsymbol{H_i}^{(t+1)} = \boldsymbol{R}^{(t)} - \boldsymbol{r_1}\ \boldsymbol{R}^{(t)} - 2.\boldsymbol{r_2}\boldsymbol{H_i}(\boldsymbol{t}) \tag{6}$$

Case 2

$$\boldsymbol{H_i^{(t+1)}} = \left(\boldsymbol{Z}^{(t)} - \boldsymbol{H}^{(t)}\right) - \boldsymbol{r3} \cdot [\boldsymbol{L} + \boldsymbol{r4} \cdot (\boldsymbol{U} - \boldsymbol{L})] \tag{7}$$

In Equation (6), the hawk update rule has two forms depending on a random probability $\boldsymbol{q}$.

(Case 1), For $\boldsymbol{q} \ge \boldsymbol{0.5}$ the new position $\boldsymbol{H_i}^{(t+1)}$ is computed based on a random family member's position $\boldsymbol{R}^{(t)}$, adjusted by random factors $\boldsymbol{r_1}, \boldsymbol{r_2}$.

(Case 2), For $\boldsymbol{q} < \boldsymbol{0.5}$, the update is guided by the prey's position $Z^{(t)}$ and the flock's average position $\overline{H}^{(t)}$ with a random displacement scaled by r_3 and the range between lower bound $\boldsymbol{L}$ and upper bound $\boldsymbol{U}$.

$$\overline{\boldsymbol{H}}^{(t)} = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{H_i^{(t)}} \tag{8}$$

The average position $\bar{H}^{(t)}$ of the hawks at iteration t is defined as the mean of the positions of all n hawks. It provides a central reference point for the swarm's state by aggregating all individual positions, which is useful for guiding collective search strategies.

$$E_{esc} = 2\, E_{init} \left(1 - \frac{t}{T}\right) \tag{9}$$

The equation shows that E_{esc} represents the prey's escape energy, which decreases linearly with the iteration number t. The prey starts with E_{init} at the beginning of the hunt ($t = 0$) and the energy decreases to 0 at the end of the hunt (T) because the prey weakens throughout the chase.

$$H_i^{t+1} = Z^t \; - E_{esc}\, |Z^{(t)} \; - H_i^{(t)}| \tag{10}$$

When the prey's escape energy is low, the hawk moves directly closer to the prey's current position $Z^{(t)}$, the update subtracts a fraction of the current distance $Z^{(t)} \; - H_i^{(t)}$ scaled by E_{esc} from the prey's position. This represents a **hard besiege** strategy in which a weakened prey cannot effectively increase the gap, allowing the hawk to significantly shrink the distance in one step.

$$H_i^{(t+1)} = \left(Z^{(t)} \; - H_i^{(t)}\right) - E_{esc}\, J\, Z^{(t)} \; - H_i^{(t)} \tag{11}$$

Under a **soft besiege** strategy, the hawk's position update includes a term J to account for a sudden random jump of the prey. The expression $J\, Z^{(t)} \; - H_i^{(t)}$ represents the prey's unpredictable movement (with J as a random jump strength factor), and the hawk's advance is reduced by E_{esc} times this jump distance. This cautious update means the hawk closes in on the prey more carefully, anticipating potential evasion.

$$J = C\sqrt{S}.A \tag{12}$$

The jump strength J is defined as a function of a constant C, a scale parameter S, and a diffusion factor A.. This structure captures the idea of a Brownian random walk: the jump size increases with the square root of the scale S (e.g. time or number of small steps) and is influenced by the constants C and A. It ensures that the prey's random jump behaviour has the right scale, with C and A. controlling the magnitude of these stochastic movements.

$$H_i^{t+1} = Z^{(t)} + |E_{esc}|\,.\; |Z^{(t)} \; - H_i^{(t)}| \tag{13}$$

If the prey still has substantial energy (represented by a negative E_{esc} in the model), the hawk may overshoot the prey's current position. In this update, the term $+|E_{esc}|\,.\; |Z^{(t)} \; - H_i^{(t)}|$ adds to the hawk's movement, effectively pushing the hawk slightly beyond $Z^{(t)}$. This models a scenario where the hawk anticipates the prey will keep moving and positions itself ahead, thereby maintaining pressure on a fast-moving prey.

$$H_i^{t+1} + \; E \; |J.Z^{(t)} \; - H_i^{(t)}| = \; Z^{(t)} \; - H_i^{(t)} \tag{14}$$

Rearranging the hawk's update equation highlights the balance between the hawk's advance and the prey's escape jump. The left-hand side shows the hawk's new position plus the portion of the prey's jump that was counteracted ($E \; |J.Z^{(t)} \; - H_i^{(t)}|$), and it equals the original gap $Z^{(t)} - H_i^{(t)}$on the right-hand side. This makes it explicit that the hawk's movement (when added to the remaining gap after the prey's jump) accounts for the entire initial distance, illustrating how the escape energy E and jump factor J together determine how much of that gap is closed.

$$H_i{}^{(t+1)} = H_i^{(t)} + L\, v \tag{15}$$

If a direct dive at the prey is not executed, the hawk updates its position using a Lévy flight step. Here $L\, v$ represents a random jump vector: L is the step size drawn from a Lévy distribution and v is a random unit direction. This allows the hawk to explore the search space more broadly, as Lévy flights produce occasional long jumps that help escape local optima, while still making smaller movements most of the time.

$$\boldsymbol{\sigma} = \left(\frac{\Gamma(1+\beta)\sin\left(\frac{\pi\beta}{2}\right)}{\Gamma\frac{(1+\beta)}{2}\,\beta\, 2^{\left(\beta-\frac{1}{2}\right)}} \right)^{\frac{1}{\beta}} \tag{16}$$

The formula determines the scale parameter $\boldsymbol{\sigma}$ which defines the Levy flight step-length distribution. The formula uses the stability parameter $\boldsymbol{\beta}$ (usually set to 1.5 for Levy flights) and the gamma function $\boldsymbol{\Gamma}(\cdot)$ to calculate $\boldsymbol{\sigma}$. The equation uses these constants to determine σ which produces step sizes that follow a heavy-tailed Levy distribution. The heavy tail property enables the hawk to occasionally make very large jumps, which is essential for global search capability.

$$\boldsymbol{L} = \frac{\boldsymbol{u}}{|\boldsymbol{v}|^{\frac{1}{\beta}}} \tag{17}$$

The step length $\boldsymbol{L}$ in the Lévy flight is generated by the random ratio shown, where $\boldsymbol{u}$ and $\boldsymbol{v}$ are independent random variables (e.g., drawn from normal distributions $\mathrm{N}(0, \sigma^2)$ and N(0,1) respectively). Because $\boldsymbol{v}$ appears in the denominator raised to the power $\frac{1}{\beta}$, this formula yields a **heavy-tailed distribution** for $\boldsymbol{L}$: most values of $\boldsymbol{L}$ will be small (when $\boldsymbol{v}$ is not too tiny), but occasional samples will produce a very large $\boldsymbol{L}$ (when $\boldsymbol{v}$ is extremely small). Such a distribution enables the hawk to perform mostly local searches with intermittent long-range explorations.

$$\boldsymbol{H}_i^{(t+1)} = \begin{cases} \boldsymbol{Y}_i, & \textbf{if } \boldsymbol{f}(\boldsymbol{Y}_i) \geq \boldsymbol{f}(\boldsymbol{Q}_i) \\ \boldsymbol{Q}_i, & \textbf{otherwise}. \end{cases} \tag{18}$$

Equation (17) implements a selection between two candidate solutions $\boldsymbol{Y}_i$ and $\boldsymbol{Q}_i$ based on their fitness values $\boldsymbol{f}(\boldsymbol{Y}_i)$ and $\boldsymbol{f}(\boldsymbol{Q}_i)$. The hawk will adopt position $\boldsymbol{Y}_i$ for the next iteration if and only if $\boldsymbol{Y}_i$ is at least as fit as $\boldsymbol{Q}_i$ (for example, yielding higher classification accuracy); otherwise, it will choose $\boldsymbol{Q}_i$. This elitist strategy ensures that the hawk moves to the more promising of the two dive outcomes, thereby improving the optimization result.

$$\mathbf{Y_i}' = \boldsymbol{Z}^{(t)} - \boldsymbol{E}_{esc}\, |\boldsymbol{j}\, \boldsymbol{Z}^{(t)} - \overline{\boldsymbol{H}}_{\boldsymbol{i}}^{(t)}| \tag{19}$$

The candidate position $\boldsymbol{Y_i}'$ is calculated using the average hawk position $\overline{\boldsymbol{H}}^{(t)}$ in the escape term. In this hard besiege with rapid dive scenario, the hawk's update is based on the prey's position $\boldsymbol{Z}^{(t)}$ and the swarm's mean position $\overline{\boldsymbol{H}}^{(t)}$. The distance between the prey and the swarm center $|\boldsymbol{J}\, \boldsymbol{Z}^{(t)} - \overline{\boldsymbol{H}}^{(t)}|$, is reduced by a factor of $\boldsymbol{E}_{esc}$. This represents a cooperative strategy where the hawk leverages the collective knowledge of the flock (via $\overline{\boldsymbol{H}}^{(t)}$) to intercept the prey more effectively.

$$\mathbf{Q_i}' = \boldsymbol{Z}^{(t)} - \boldsymbol{E}_{esc}\, |\boldsymbol{j}\, \boldsymbol{Z}^{(t)} - H_i^{(t)}| \tag{20}$$

The alternative candidate $Q_i{}'$ uses the hawk's own current position $H_i^{(t)}$ in the update formula. Here the distance from the prey to this hawk, $j\, Z^{(t)} - \overline{H}_i{}^{(t)}$, determines the adjustment. By computing both Y_i' and $Q_i{}'$ (from Eq. 18 and 19), the algorithm can compare a cooperative move (using the swarm's average position) versus an individual move (using the hawk's personal position) and then choose the better one as per Eq. (17). This ensures robust exploitation by considering different pursuit tactics for the prey.

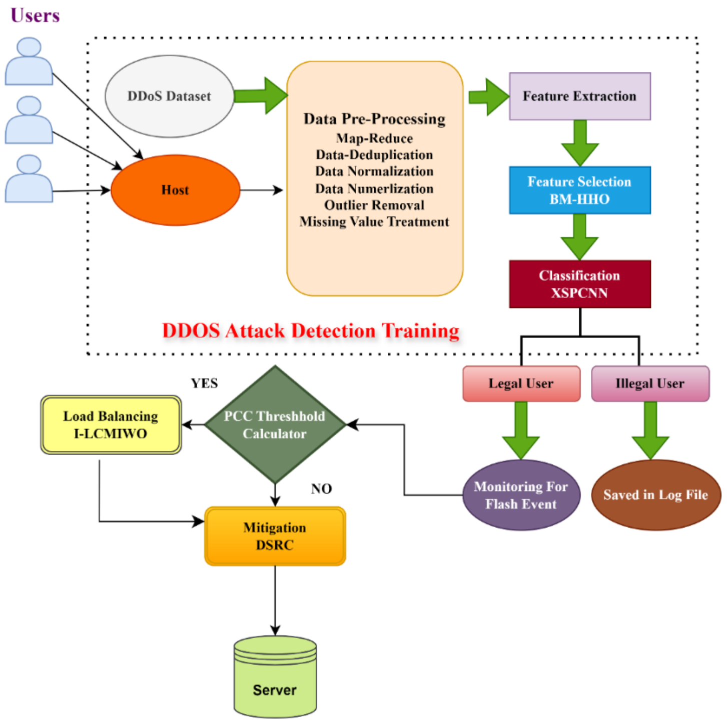

Figure 2. Block diagram of the proposed model

BM-HHO served as an algorithm to automate feature selection by determining the relationship between each feature and the target variable. The approach was basic: The system maintained only the features which delivered performance benefits regardless of the data environment changes such as traffic fluctuations or noise introduction. The method delivered consistent results. The method consistently picked high-impact features regardless of the amount of variability present in the data. Such adaptability becomes essential when working with datasets that lack uniformity and predictability. After feature selection the X-SPCNN model received the most effective features for training. The X-SPCNN model addresses standard deep model problems through its combination of Xavier initialization and Soft Plus activation function. The development of CNNs has led to better performance but they face ongoing issues with gradient vanishing and unstable scaling problems. The Xavier Soft Plus combination helps address both. Figure 3 illustrates the complete X-SPCNN model structure along with its modifications for this particular implementation.

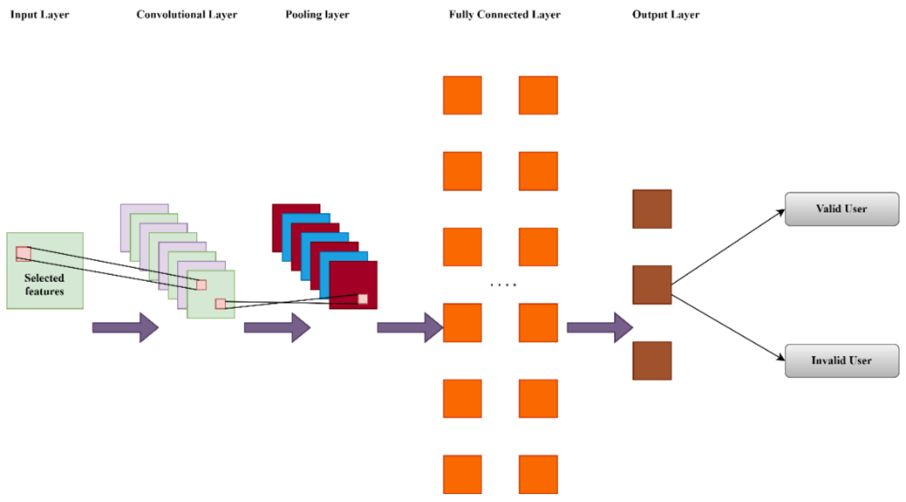

Figure 3. Architecture of X-SPCNN

$$O_{CL}^{(c)} = S\left(\sum_{i,j} W_{ij}^{(c)} X_{ij}\right) \tag{21}$$

This equation defines the output of the convolutional layer $O_{CL}^{(c)}$ for channel (c). It is obtained by applying the Soft-plus activation function S(·) to the weighted sum of inputs X_{ij} with convolutional weights $W_{ij}^{(c)}$ over all positions i, j. In other words, for feature map ccc, each output is the Soft-plus of a convolution between the input patch X_{ij} and the corresponding kernel W^c .

$$S(x) = ln\,(1 + ex) \tag{21}$$

The Softplus activation function $S(x)$ is defined as the natural logarithm of $(1 + ex)$. This smooth approximation of ReLU ensures the output is always positive and differentiable. In the context of the X-SPCNN model, using Softplus helps avoid issues like vanishing gradients by providing a gentler nonlinear transformation of the convolutional responses.

$$O_{PL}^{(c)}(m, n) = max_{(i,j)\,\epsilon\,\Omega_{m,n}}\ O_{CL}^{(c)}(i, j) \tag{22}$$

Equation (22) describes the max-pooling operation.

$O_{PL}^{(c)}(m, n)$ is the output of the pooling layer at spatial location (m, n) for channel (c), defined as the maximum value of the convolutional feature map $O_{CL}^{(c)}(i, j)$ within the region $\Omega_{m,n}$. Here $\Omega_{m,n}$ denotes the set of indices i, j in the receptive field (or "pooling window") corresponding to output (m, n). This down-sampling step retains the most prominent activation in each region, reducing feature map size while preserving important features.

$$W_{i,j} \sim U\left(-\sqrt{6/(n_i + n_o}\right), \sqrt{\frac{6}{n_i + n_o}} \tag{23}$$

The weight initialization follows the Xavier (Glorot) scheme. Each weight $W_{i,j}$ in the fully connected layer is drawn from a uniform distribution $U(-a, +a)$ where $=\ 6/(n_i + n_o)$. Here n_i and n_o are the number of input and output neurons for that weight. By using this symmetric range $\pm\sqrt{6/(n_i + n_o)}$, the initial weights have variance scaled to the network's size, which helps maintain stable gradients through the network and improves convergence during training.

$$O_{FCL} = X_{flat}W + b \tag{24}$$

The output O_{FCL} of the fully-connected layer is given by a linear combination of the flattened input and the network weights. In this expression, X_{flat} is the flattened output from the previous layer $(a\ 1 \times N\ feature\ vector)$, W is the weight matrix, and b is the bias term. The product $X_{flat}W + b$ yields a vector of raw scores (logits) that will be passed to the final classification stage.

$$P(y = k) = \frac{exp(z_k)}{\sum_j exp(z_j)} \tag{25}$$

The Soft-max function transforms the logits z_j into a probability distribution across classes. The predicted probability $(y = k)$ that the input belongs to class k is calculated through this equation. The predicted probability is calculated by taking the exponential of (z_k) followed by a division operation with the total exponential sum of all logits. The output probabilities remain non-negative and add up to 1, which is suitable for multi-class classification results.

3.2 Differentiation Segment

The request is observed for flash events before mitigation. The Pearson Correlation Coefficient (PCC) is used to detect flash events if a large number of valid requests are acknowledged. The Pearson Correlation Coefficient Threshold (PCC) is utilized to check for flash events if the system receives a large number of legitimate requests.

Journal of Information Systems Engineering and Management
2025, 10(51s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

$$\rho = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_i (x_i - \bar{x})^2 (y_i - \bar{y})^2}} \qquad (26)$$

The Pearson correlation coefficient ρ between two sequences $\{x_i\}$ and $\{y_i\}$ is calculated by this formula. The numerator represents the covariance between x and y through the sum of their deviations from $\bar{x}$and $\bar{y}$. The denominator represents the product of the standard deviations of x and . A higher ρ indicates a stronger linear relationship. The flash event detection scenario uses x_i and y_i to represent successive time window request counts, so a large ρ value indicates a correlated surge (potential flash event).

We achieved better flash event detection through the integration of K-means clustering with Pearson Correlation Coefficient (PCC) analysis. The approach involved grouping similar genuine requests together for subsequent correlation checks within each cluster. The method enables detection of unusual patterns such as traffic surges that could potentially mimic flash events but prove to be non-attack activities. The PCC analysis applied at a local level minimizes unnecessary alerts while detecting unusual network traffic patterns during critical periods. Our load balancing approach concentrated on distributing traffic for preventing network congestion. The Interpolation–Crossover Mutation Invasive Weed Optimization (I-CMIWO) algorithm serves as the solution to address this problem. The IWO method serves as the foundation for this algorithm which draws its inspiration from weed colonization patterns but includes modifications to enhance its response to changing bandwidth requirements. The I-CMIWO system adjusts its traffic distribution through current bandwidth usage measurements instead of using fixed thresholds to determine available data movement capacity. The standard deviation shifts in traditional IWO optimization methods create a major problem. We added an interpolation crossover mutation system to make the process more stable. The four-stage process of this enhanced system starts with initialization followed by reproduction then spatial dispersal and finishes with competitive selection.

The algorithm begins by priming a population of weeds for legitimate requests. This population is randomly distributed across the search space.

$$S_i = \left| \frac{(f_i - f_{min})}{(f_{max} - f_{min})} \cdot N_{max} \right| \qquad (27)$$

This interpolation formula computes the number of seeds S_i produced by weed i in the I-CMIWO load balancing algorithm. The value (f_i) is the fitness of the current weed (for instance, inversely related to request response time), while f_{min} and f_{max} are the minimum and maximum fitness in the weed population. By taking the ratio $\frac{(f_i - f_{min})}{(f_{max} - f_{min})}$ we scale $i's$ fitness to a 0–1 range; multiplying by N_{max} (the maximum number of seeds possible) and flooring the result yields an integer number of seeds. This means weeds with higher fitness (better performance) will reproduce more, mirroring natural selection where fitter individuals have more offspring.

After reproduction, the seeds were randomly scattered in the search space. Instead of using the standard deviation process to generate new seed positions, Crossover Mutation (CM) is applied to enhance exploration. Crossover combines position vectors from multiple weeds to generate new, optimised positions for the seeds. The seed placement is optimized with merging vectors as shown in Figure 4

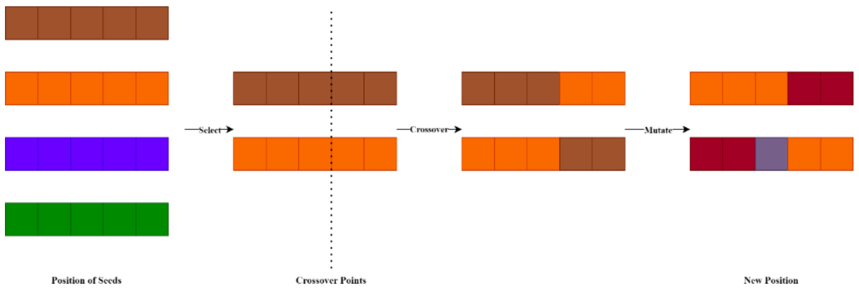

Figure 4. Crossover Mutation Method

The competition for survival begins between weeds at this point. The survival chances of weeds depend on their fitness level because advanced fitness standards give them an advantage over inferior fitness values. The procedure continues its iterative cycle until either a predetermined maximum number of repetitions is completed or the best resolution becomes apparent. The optimal solution came from weeds that demonstrated the highest fitness value. The load-balancing system schedules user requests according to their bandwidth requirements. The system uses a priority-based approach that gives higher bandwidth requests the same level of importance as fitness-based weed selection does.

3.3 Mitigation Segment

Mitigation strategies aim to minimise the impact of malicious requests. A common technique is to alter packets that use false-source IP addresses. In this model, we employ the double-shift right-and-complement (DSRC) technique to handle legitimate requests that fall below the bandwidth threshold or are part of balanced legitimate requests.

$$\boldsymbol{B = bin_8(d1) \parallel bin_8(d2) \parallel bin_8(d3) \parallel bin_8(d4)} \tag{28}$$

An IP address is transformed into binary form as shown in Equation (28). Each octet d_k (for k=1,2,3,4) is converted to its 8-bit binary representation $bin_8(dk)$ These four 8-bit binary strings are then concatenated (denoted by the symbol "‖") to form a single 32-bit binary number BBB. This represents the original IP address in binary format, which is necessary for applying bit-level operations in the mitigation phase.

$$\boldsymbol{C = \left\lfloor \frac{B}{4} \right\rfloor} \tag{29}$$

After converting the IP to binary, a double right-shift operation is performed. Dividing B by $2^2 = 4$ and taking the floor is equivalent to shifting all bits of B two places to the right (with the two rightmost bits dropped). The result is C, the shifted binary number. In effect, if B was a 32-bit binary, C corresponds to that IP's binary value with the last two bits set to 0, implementing the required two-bit right shift in the DSRC technique.

$$\boldsymbol{D = \overline{C}} \tag{30}$$

Finally, the one's complement of the shifted binary address is obtained in Equation (30).

$\boldsymbol{\overline{C}}$ denotes that every bit of C is flipped (0 becomes 1 and 1 becomes 0), yielding D. This transformed binary D is the mitigated IP address after applying the double-shift right and complement operations. In practice, these altered binary bits would be placed back into the IP header, so that malicious requests use the modified source D, a technique which helps in mitigating DDoS attacks by invalidating the attacker's source address information.

This DSRC technique helps prevent malicious requests from taking control of the system, thereby ensuring that legitimate traffic is prioritised and securely managed.

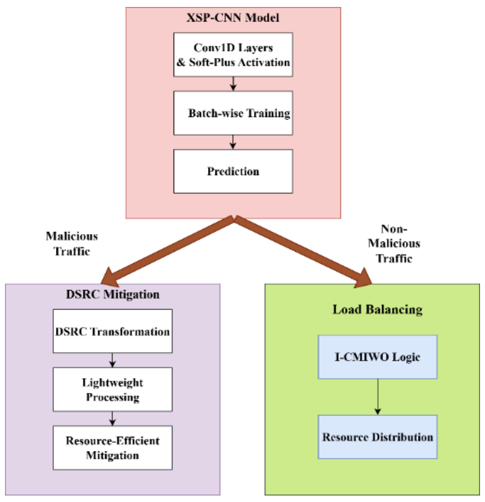

Figure 5. Process Flow Diagram

Fig. 5 depicts process flow diagram which represents a management of network traffic. After incorporating an XSP-CNN model which has convolutional layers and Soft-Plus activation function for the traffic classification, the model is trained in batches and classifies the traffic as malicious. If the traffic is malicious, then the traffic is diverted through DSRC mitigation process which includes transformation and lightweight processing. Otherwise, if the traffic is identified as not aggressive, then it is forwarded to load balancing mechanisms that use I-CMIWO approach to allocate resources in the best possible manner.

RESULTS

Here, an experiment on the proposed model is performed empirically and compared with existing methods. This evaluation is performed using a public dataset to validate the performance of the proposed model.

4.1 Performance Analysis

Performance evaluation considered metrics such as DDoS attack detection classification and fitness versus iteration analysis for feature selection (FS) etc.

The employed CIC-Darknet 2020 dataset includes data from darknet network traffic, which is utilized to launch application-layer DDoS Attacks. For the CIC-Darknet 2020 dataset, an orchestration of two layers is used to generate benign and darknet traffic in the primary layer. The subsequent layer creates the chat, email, audio stream, video stream, browsing, p2p transfer, audio stream, browsing, and VOIP on the Internet. Data from the Canadian Institute for Cybersecurity at the University of New Brunswick were collected to assess new techniques for Internet traffic classification.

We evaluated the performance of the Brownian Motion-based Harris Hawks Optimization (BM-HHO) algorithm using a fitness-versus-iteration analysis. Figure 6 shows how BM-HHO influenced feature selection specifically how many features were retained versus removed.

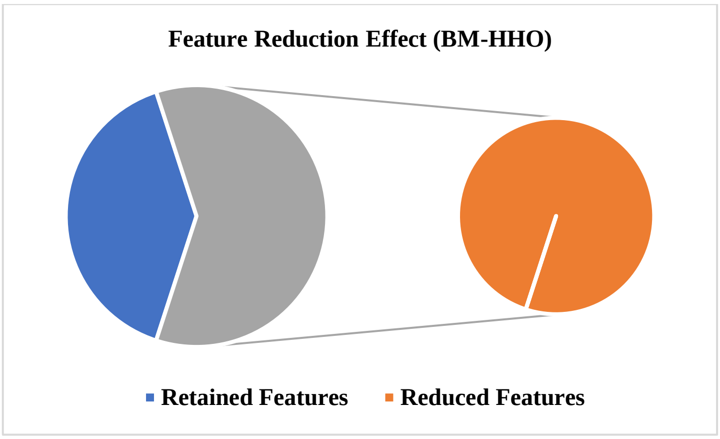

Fig. 6 Performance analysis of BM-HHO algorithm

Initially, the dataset included a large number of features, but once BM-HHO was applied, the feature set was significantly reduced. What's important is that this reduction didn't compromise model performance. In fact, around 40% of the features were kept, meaning about 60% were removed mostly those that had little or no impact on classification.

Table 2. Performance investigation of projected XP-CNN in contrast with the existing algorithms

Metrics/ Techniques	Recurrent Neural Network (RNN)	Deep Belief Network (DBN)	Convolutional Neural Network (CNN)	Deep Neural Network (DNN)	Proposed XSP-CNN
Accuracy	95.7671	93.3450	91.5492	90.6574	98.9609

Precision	95.8477	93.4256	91.6955	90.6574	98.5778
Recall	94.8756	93.8476	91.8653	90.5478	98.8745
F-Measure	95.4578	93.6875	91.1548	90.6849	98.3545
CPU Memory Usage (in kb)	590221	638194	789666	888244	415068
Training Time (in ms)	23107	32107	43208	55107	18007
Sensitivity	95.8477	93.4256	91.6955	90.6574	98.5778
Specificity	95.6834	93.2624	91.3978	90.6574	98.5438

This reduction brought down the size of the dataset, which helped improve training speed and overall efficiency something especially important in resource-constrained environments like MANETs. The way BM-HHO balances which features to keep and which to drop shows that it can consistently identify the most meaningful subset for training. That balance helps maintain accuracy while cutting down on noise and unnecessary computation.

To further evaluate the system, we compared our approach against several well-known models. Metrics like accuracy, precision, recall, F-measure, and training time were used to assess performance. The recall of the DDoS detection, in particular, was measured through our Xavier Soft-Plus CNN framework to see how it handled classification under real traffic patterns.

As shown in Table 2, the Xavier SoftPlus CNN (XSP-CNN) outperforms the other deep learning models tested in this study, including RNN, DBN, CNN, and DNN. It leads across all key performance metrics. What really gives it the edge is the combination of Xavier initialization and the SoftPlus activation function. Together, they stabilize the model and improve both training speed and accuracy. The Xavier method helps distribute weights more evenly, which prevents issues like vanishing or exploding gradients. Because of this, the model converges faster training time dropped to 18,007 ms compared to 55,107 ms for the DNN. The SoftPlus activation also plays a big role. Unlike ReLU, which can completely deactivate neurons, SoftPlus has a soft slope that keeps neurons active during training. That's one reason XSP-CNN hits 98.96% accuracy, outperforming RNN at 95.77% and CNN at 91.55%.

But it's not just about accuracy—XSP-CNN is lighter, too. It only used 415,068 KB of memory during training, which is significantly lower than CNN (789,666 KB) and DNN (888,244 KB). That makes it a better fit for real-time use where system resources are limited. The model also scored high in sensitivity and specificity 98.57% and 98.54%, respectively which shows it's capable of catching subtle patterns without overfitting. Altogether, these results make a strong case for XSP-CNN as the most reliable choice for complex classification tasks, especially when both performance and efficiency matter.

Figure 7 illustrates how the model performs under dynamic conditions, where the input data is intentionally scrambled for each run. This setup mimics the kind of unpredictability you'd expect in real-world network traffic—like what happens in MANETs or during actual DDoS attacks. By reshuffling the dataset at every cycle, the model is forced to adapt and learn from different traffic distributions each time. The idea is to break the static assumptions and see whether the XSP-CNN can still hold up when the environment shifts. After each run, we record the model's accuracy, and then plot the results to evaluate both performance and consistency. What this approach really tests is how well the system handles uncertainty, something that's essential in adaptive networks. The goal isn't just accuracy in ideal conditions, but stable behaviour across a range of unpredictable inputs.

The resource consumption analysis measures the computational efficiency of the XSP-CNN model when it performs inference operations. The framework measures the model's real-time performance through inference time logging while processing the scaled test dataset. The model requires 5.43 seconds to produce predictions which reveals its speed and responsiveness. The evaluation holds essential importance for MANET environments because it determines the model's performance regarding latency and power usage. The logged inference time proves the XSP-CNN model's readiness for real-world deployment through its ability to perform lightweight high-performance computations. The model becomes more computationally efficient when its inference time is shorter thus meeting the requirements of DDoS attack detection systems that need low latency. The development of scalable flexible frameworks requires truncated logged inference time to handle growing network demands while preserving high performance in dynamic high-mobility networks such as MANET.

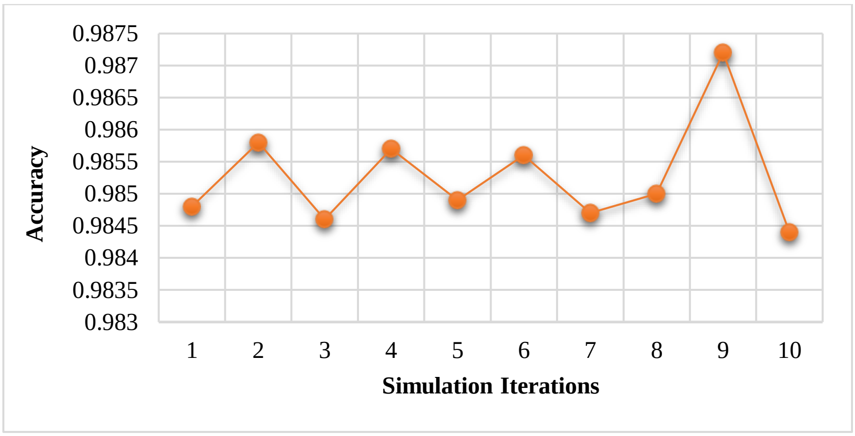

Figure 7. XSP-CNN Performance Under Dynamic Scenarios

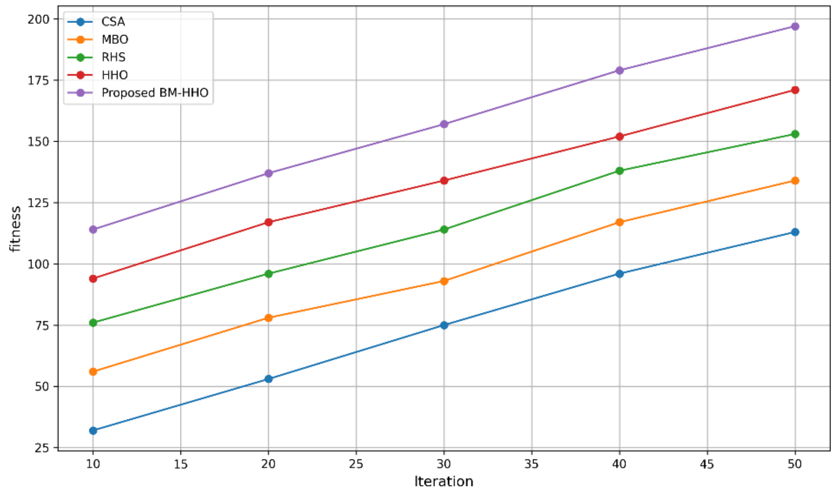

Figure 8. Fitness vs Iteration Study

Figure 8 shows a graphical representation of fitness versus iteration analysis. The graphical representation illustrates how the fitness value changes with each iteration, thus depicting the convergence tendencies. This makes it easier to comprehend the performance and efficiency of the proposed algorithm in the optimization of resource allocation.

We tested our model on the "Friday-WorkingHours-Afternoon-DDos.pcap_ISCX.csv" dataset from CIC-IDS2017. It's one of those benchmark sets that includes actual traffic like real-world highway data mixed with different kinds of attacks, including botnet and PDoS. The model held up well and scored over 98.78% accuracy, which was honestly better than we expected at this stage. to make it closer to real-world behaviour, we replayed the. pcap data using tools like Wireshark, and in some cases, converted it into structured features—stuff like flow stats and header-level metadata. That part helped a lot. It gives the model more to work with, and that's what makes it spot strange behaviours or low-key attack patterns that aren't easy to catch otherwise. It's not just about raw accuracy—it's also about being able to handle unexpected stuff as it comes.

One thing we made sure of was that the system stays flexible. It uses a few different thresholds and learning layers so that it doesn't mess up on edge cases. The PCC-based detection paired with K-means clustering helped a lot here. That combination made it easier to tell apart DDoS spikes from legit flash traffic. You can see the improvement clearly in Figure 9, it's not perfect, but the distinction got a lot sharper.

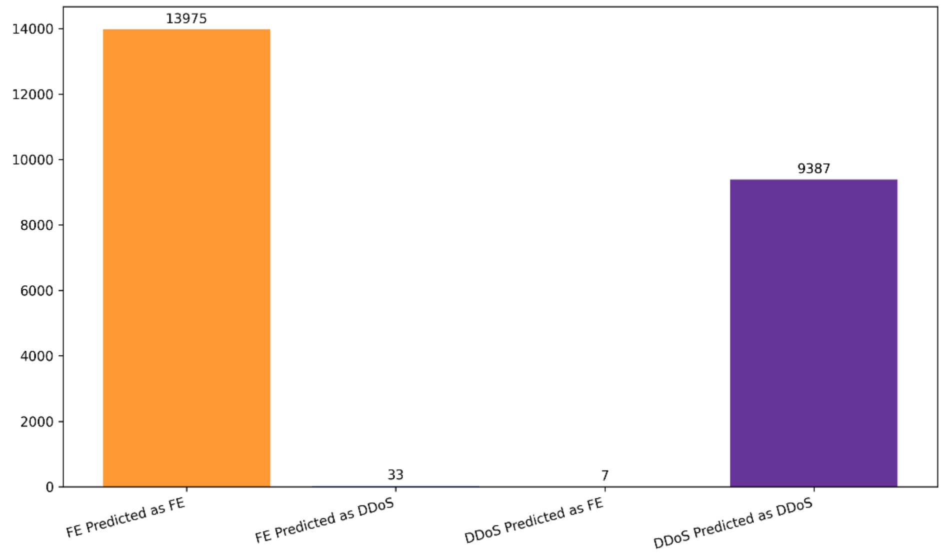

Figure 9. Flash Events Differentiation Analysis

We assessed the ability of the model to distinguish between flash events and actual DDoS traffic. To be honest, the accuracy of 99.82% was a bit of a surprise to us. It wasn't just a clean win, what really made it stand out was how well it handled weird edge cases, where the traffic patterns looked similar. Real attacks were flagged consistently without any misclassification of legitimate surges. When we broke it down further, the precision was 99.65%, recall hit 99.93%, and F1-score came out to 99.79%. High numbers across the board, but the real takeaway was how steady it was over multiple runs. The model didn't just do well once it kept performing under different traffic setups. That kind of consistency is what matters most if this is going to be useful in an actual network, not just in testing. Figure 10 shows the performance of the XSPCNN model for different MANET sizes ranging from 50 to 500 nodes. The system replicates real-world dynamic conditions by sampling subsets of the CIC-DDoS2019 dataset and introducing synthetic "Node Mobility" features. The CNN model, optimized with Conv1D layers, maintains efficiency and accuracy despite increasing node density. The results show high accuracy (100% for 50 nodes, 99% for 100 nodes, and 98% for 200 and 500 nodes), which proves the model's robustness and scalability. This lightweight, power-efficient approach highlights its suitability for real-time MANET deployments under resource-constrained environments.

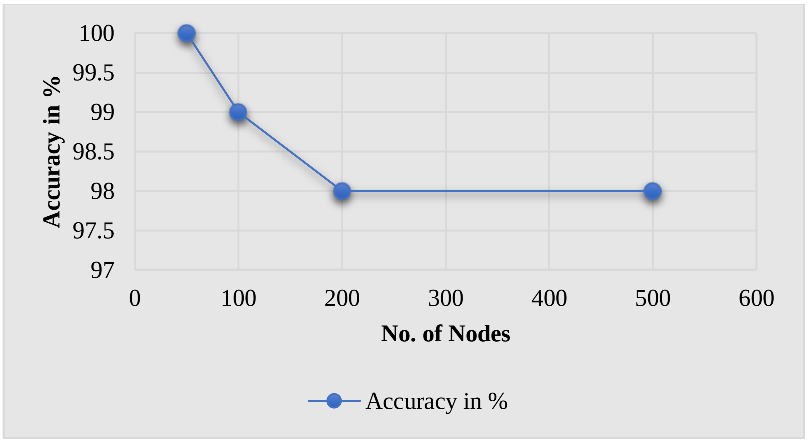

Figure 10. Scalability Analysis of XSP-CNN algorithm

4.2. Load Balancing Performance Inspection

This section looks at how the proposed I-CMIWO algorithm holds up against existing approaches across several performance metrics—things like average waiting time, throughput, latency, processing time, and how well each method balances the load. Based on the data, our interpolation-based crossover mutation model consistently outperforms the others in nearly every category.

The system stands out because of its scheduling management. The average waiting time along with process time and response time experience significant reductions through I-CMIWO. The system cuts waiting time by 34% and enhances response time by 28% better than BOA, SMO, ROA and the classical IWO as shown in figure 11 and table 3. The hybrid crossover-mutation strategy of this system leads to enhanced population diversity which results in faster convergence. The system controls weed distribution through dynamic adjustments which results in better load balancing. The system achieves better Quality of Service indicators through its latency and throughput improvements. The system delivers a 32% improvement in these performance metrics according to Figure 12 and table 4. The performance of I-CMIWO under pressure makes it an effective solution for real-time systems that require fast responses and stable operation.

Proposed dynamic thresholding method functions within MANET environments as illustrated in Figure 13. The K-Means clustering algorithm grouped scaled features from the dataset into distinct clusters which appear in different colors in the plot. The cluster centers appear as red 'X' symbols. The clustering step functions as a fundamental component. Real-time traffic patterns enable us to set adaptive thresholds which observe how each group behaves. The constantly changing conditions of MANETs make this feature essential. The approach enables better detection of application-layer DDoS attacks while preventing excessive reactions to typical traffic fluctuations.

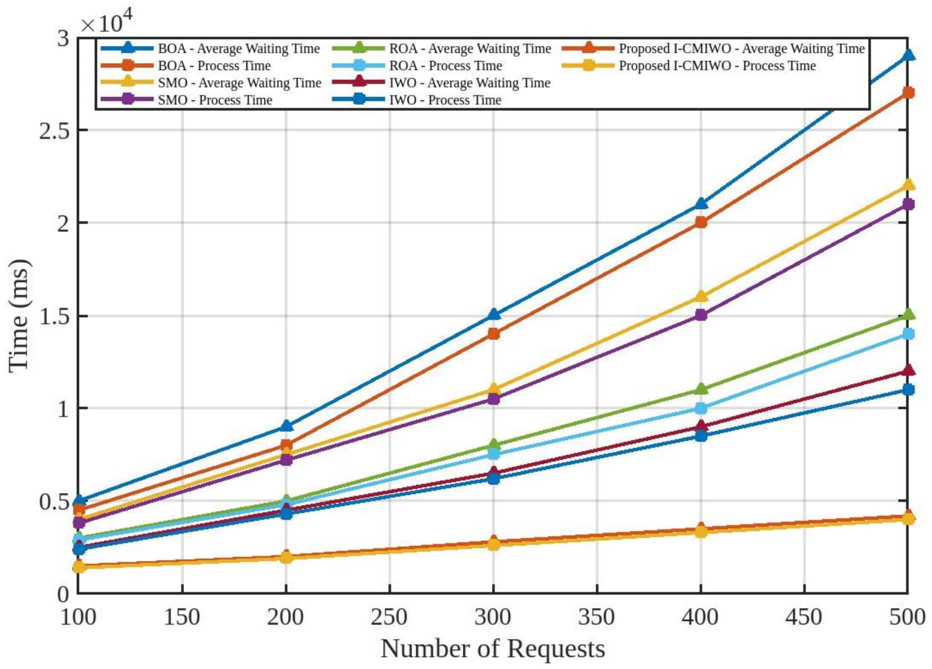

Figure 11. Performance investigation of projected I-CMIWO in contrast with the existing algorithms

The adaptable structure enables the detection model to adapt to network fluctuations which occur due to node movement and unexpected traffic patterns. Clustering enables significant reduction of misclassifications because it operates within the energy and scalability constraints of resource-limited MANETs. The cluster identification in the dataset holds multiple applications beyond detection purposes. The process of grouping nodes according to their resource profiles enables better resource distribution. The optimization of cluster-based routing helps decreases network congestion while enhancing total network performance. Monitoring cluster node behaviour provides an additional advantage for detecting security threats and anomalies at an early stage. The process requires feature scaling as an essential step. The absence of feature scaling allows features with extensive numerical ranges to control clustering results which produces biased or misleading outcomes. Scaling ensures that all features contribute equally.

Table 3. Performance investigation of projected I-CMIWO in contrast with the existing algorithms (Time in ms)

No. of Requests	Metric	BOA	SMO	ROA	IWO	Proposed I-CMIWO
100	Avg. Waiting Time	281	243	214	184	164
	Process Time	6523	6071	5714	5237	4855
	Response Time	4236	3784	3085	2784	2128
200	Avg. Waiting Time	438	408	376	351	307
	Process Time	9214	8624	7214	6819	5744
	Response Time	5517	4925	4467	4068	3716
300	Avg. Waiting Time	661	602	534	476	418
	Process Time	13412	10217	9134	8129	6324
	Response Time	6127	5617	5049	4571	4162
400	Avg. Waiting Time	874	786	703	621	543
	Process Time	21475	17514	14268	12713	10386
	Response Time	6758	6234	5871	5496	5037
500	Avg. Waiting Time	1129	1002	914	842	746
	Process Time	29511	25117	21477	19227	16847
	Response Time	7329	6874	6473	6178	5772

Feature 1 (Scaled) in this configuration shows network node mobility while Feature 2 (Scaled) indicates total network traffic volume. The original dataset provides these values which underwent normalization to ensure clustering based on actual data characteristics instead of numerical magnitude.

Table 4. Performance investigation of projected I-CMIWO for QoS Improvement

No. of Requests	Metric	BOA	SMO	ROA	IWO	Proposed I-CMIWO
100	Latency (ms)	1054	845	635	487	214
	Throughput (Kb/s)	2157	1856	1428	1135	952
	DSRC Time	0.532214	0.482753	0.431251	0.387415	0.348522
200	Latency (ms)	2086	1648	1254	875	468
	Throughput (Kb/s)	2954	2415	1936	1497	1247
	DSRC Time	0.587412	0.547521	0.497285	0.451475	0.412477
300	Latency (ms)	3067	2415	1847	1275	684
	Throughput (Kb/s)	3324	2765	2382	1873	1486
	DSRC Time	0.635211	0.598475	0.554744	0.512477	0.475623
400	Latency (ms)	4074	3215	2468	1675	847
	Throughput (Kb/s)	3762	3357	2912	2318	1865
	DSRC Time	0.679854	0.623574	0.578456	0.536587	0.485214

Journal of Information Systems Engineering and Management
2025, 10(51s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/
Research Article

500	Latency (ms)	5023	4067	3086	2057	1049
	Throughput (Kb/s)	4215	3862	3175	2651	2375
	DSRC Time	0.756234	0.712478	0.674455	0.632478	0.574123

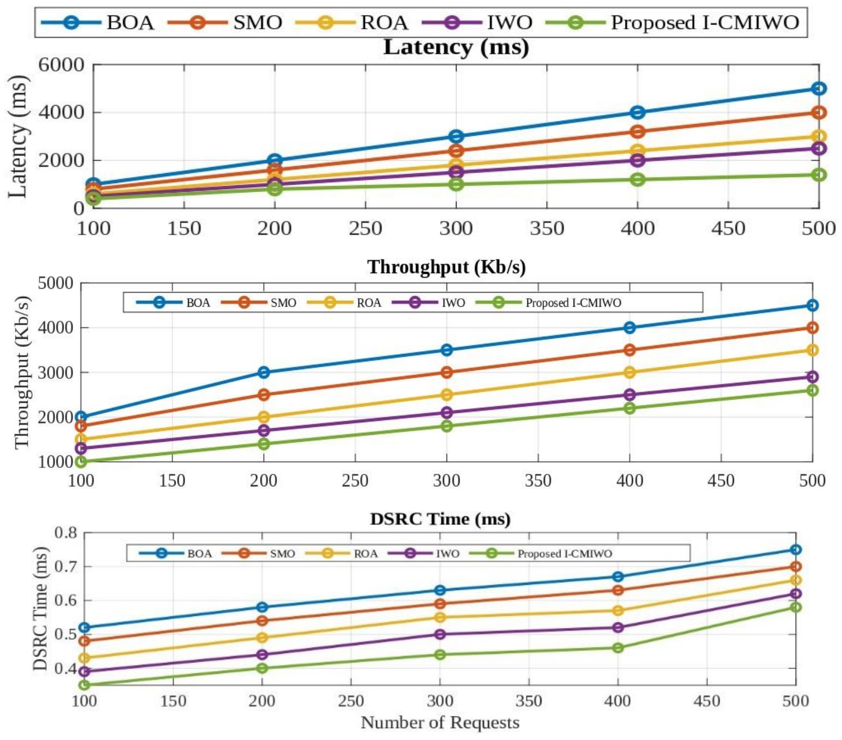

Figure 12. Performance investigation of projected I-CMIWO for QoS Improvement

We ran an analysis on the DSRC (Double Shift Right Complement) operation to see how well it performs when used for lightweight data processing. The focus was on how long it takes to apply the transformation across a given dataset. Since MANET applications are often power-sensitive and delay-critical, we wanted to make sure the method was fast enough to be practical. To do that, we applied the DSRC transformation across the dataset and recorded the time it took for each instance. The results showed that the operation is genuinely lightweight as it processes data quickly and doesn't put much strain on system resources. That kind of low-latency behaviour is exactly what you want in real-time systems.

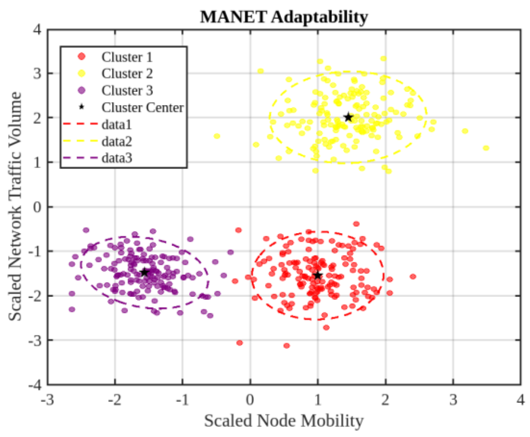

Figure 13. Dynamic Thresholding Analysis for MANET compliance

The limited processing power of devices in MANET setups makes DSRC an appropriate solution. The transformation happens quickly and the approach maintains low power usage which enhances both scalability and efficiency. The proposed model demonstrates its performance in accuracy measurements through Figure 14 when compared to other popular methods. It highlights the same methods which considered the same dataset for evaluation [37] [38] [39] [40].

The system delivers more than accurate detection because it includes smart classification and active mitigation features as part of its full-stack protection system. The system detects application-layer DDoS attacks and separates flash events from legitimate traffic spikes so neither are reported as threats. The Pearson Correlation Coefficient (PCC) acts as the solution to prevent false positives by distinguishing actual attacks from sudden legitimate traffic bursts. The XP-CNN enhances precision through its ability to detect spatial and temporal patterns in traffic flow which leads to better classification results.

The I-CMIWO algorithm takes over mitigation responsibilities in this system. The system handles resource management and performs automatic load distribution to combat changing security threats effectively. The system components work together to form a complete system. The system detects threats early while responding effectively to maintain network stability under pressure.

This framework brings together several pieces that haven't really been integrated before low-power optimization, adaptive mobility handling, and resource-conscious processing, all aimed at making MANETs more practical in real-world conditions. What makes it work is the way everything fits together? The lightweight pre-processing steps like deduplication, scaling, and imputation keep the load off the edge devices, which are often running on limited power.

The core detection engine, XSP-CNN, uses Conv1D layers along with the Soft-Plus activation function to strike a balance between fast convergence and smooth gradient flow, without draining resources. On the optimization side, BM-HHO helps shrink the feature space without losing key data. Some of those features, like synthetic node mobility metrics, capture the dynamic behaviour of the nodes, while PCC-based adaptive thresholds help the model adjust to topological changes as they happen.

We also accounted for real-world events like link failures. Here, I-CMIWO plays a role by redistributing tasks and helping the network stay balanced even when things go sideways. For protection, the DSRC mechanism adds a lightweight defense layer that works well with power-constrained nodes. Finally, the batch-wise training and tests across different node densities showed the model holds up in both low-power and high-mobility conditions. Altogether, the system shows what a next-generation MANET framework might look like one that's responsive, scalable, and ready for real-world demands.

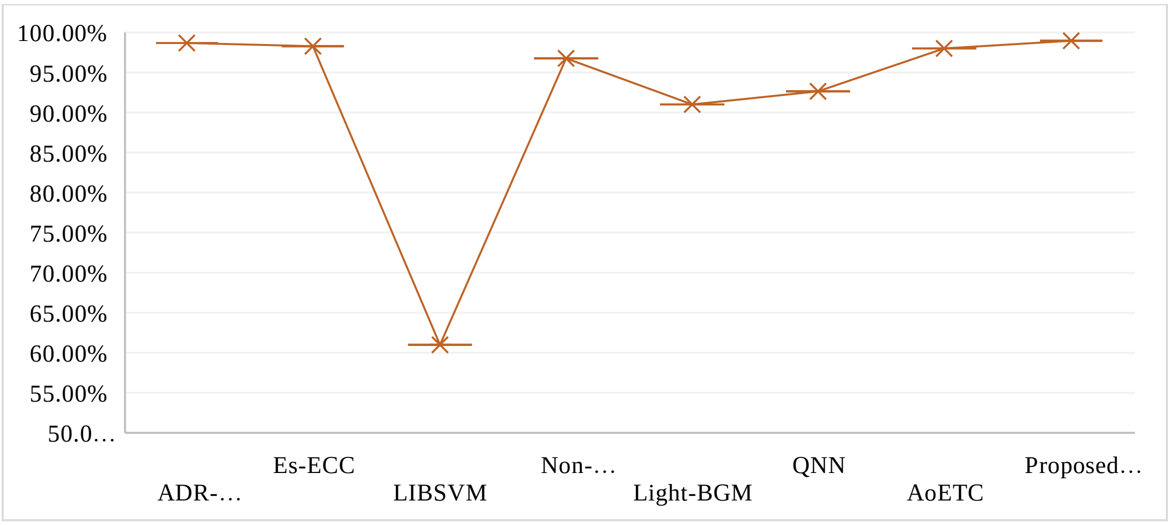

Figure 14. Accuracy Investigation with Existing Proposed Techniques

DISCUSSION

The system's integrated XSP-CNN algorithm demonstrated superior effectiveness for identifying application-layer DDoS attack traffic between malicious and harmless traffic. XSP-CNN demonstrated superior performance than previous models through all important evaluation metrics. The system achieved an accuracy of 98.96% that surpassed conventional methods by a significant margin. The model performed well in static and dynamic scenarios of MANET environments because it did not experience any performance degradation. The implementation of BM-HHO as a feature selection method delivered distinct performance benefits to the system. The dataset reduction process with BM-HHO selected the most vital features which resulted in lower computational requirements without compromising accuracy. The I-CMIWO algorithm enhanced multiple QoS parameters through better load distribution since it decreased processing time and response time and latency. The average waiting time decreased which demonstrated that tasks processed at a faster rate with minimal delays. The DSRC mechanism established a lightweight yet reliable defense system that worked well for nodes with constrained resources. The PCC-based adaptive thresholding system improved identification of actual attacks against flash events which proves difficult for most real-world networks. The complete system functions as a complete protection system against application-layer DDoS attacks in MANETs. The system demonstrates scalability and efficiency while responding to unpredictable network behavior that occurs regularly in real-world networks. The DSRC approach demonstrates excellent efficiency but experiences scalability problems when working with node densities reaching 1000 or more. The PCC-based adaptive thresholds need additional optimization to work with heterogeneous complex networks that have different traffic patterns since the network model assumes homogeneity. Extension plans to address these limitations through additional scalability tests with elevated node density levels. The MANET environment can benefit from XSP-CNN model advancements through complex architecture improvements which would enhance system flexibility and performance across different operational conditions.

Conflict of interest

On behalf of all the authors, I declare that there are no conflicts of interest.

REFRENCES

[1] N. Singh, A., Dumka, R. Sharma, “A novel technique to defend DDOS attack in MANET,” Journal of Computer Engineering & Information Technology, vol. 7, pp. 1-5, 2018.

[2] S. Khan, F. Hashim, M. F. A. Rasid, and T. Perumal, “Reducing the severity of black hole and DDoS attacks in MANETs by modifying AODV protocol using MAC authentication and symmetric encryption,” in 2nd International Conference on Telematics and Future Generation Networks (TAFGEN), 24-26 July 2018, Kuching, Malaysia, 2018.

[3] Deepa, K. S. Dhindsa, and B. Bhushan, Clustering-based Technique to Defend DDoS Attacks in Mobile Ad Hoc Networks, 1st ed. Cham: Springer, 2020.

[4] Deepa, K. S. Dhindsa, and K. Singh, “Entropy-based DDoS attack detection in cluster-based mobile Ad Hoc networks,” Ad Hoc & Sensor Wireless Networks, vol. 49, pp. 269-288, 2021.

[5] M. N. Anjum, C. Chowdhury, and S. Neogy, “Implementing mobile agent based secure load balancing scheme for MANET”, in International Conference on Opto-Electronics and Applied Optics (Optronix), 18-20 March 2019, Kolkata, India, 2019.

[6] D. Ramphull, et al., "A review of mobile ad hoc NETwork (MANET) protocols and their applications," in 2021 5th International Conference on Intelligent Computing and Control Systems (ICICCS), IEEE, 2021.

[7] A. E. Cil, K. Yildiz, and A. Buldu, “Detection of DDoS attacks with feed-forward based deep neural network model,” Expert Systems with Applications, vol. 169, no. 4, pp. 1-8, 2021.

[8] N. Agrawal and S. Tapaswi, “Low-rate cloud DDoS attack defense method based on power spectral density analysis”, Information Processing Letters, vol. 138, pp. 44-50, 2018.

[9] P. C. Sekar and H. Mangalam, “Third generation memetic optimization technique for energy efficient routing stability and load balancing in MANET”, Cluster Computing, vol. 22, no. 1, pp. 1-8, 2018.

[10] S. Janakiraman, D. Priya, and C. Jebamalar, “Integrated context-based mitigation framework for enforcing security against rendezvous point attacks in MANETs,” Wireless Personal Communications, vol. 119, no. 3, pp. 2147-2163, 2021.

[11] R. Bhuvaneswari and R. Ramachandran, "Denial of service attack solution in OLSR based MANET by varying number of fictitious nodes," Cluster Computing, vol. 22, no. 4, pp. 1-11, 2018.
[12] M. Anbarasan, S. Prakash, A. Antonidoss, and M. Anand, "Improved encryption protocol for secure communication in trusted MANETs against denial of service attacks," Multimedia Tools and Applications, vol. 79, no. 1, pp. 1-21, 2018.
[13] M. Islabudeen and M. K. Kavitha Devi, "A smart approach for intrusion detection and prevention system in mobile Ad Hoc networks against security attacks," Wireless Personal Communications, vol. 112, no. 55, pp. 1-32, 2020.
[14] M. Shukla, B. K. Joshi, and U. Singh, "Mitigate wormhole attack and blackhole attack using elliptic curve cryptography in MANET," Wireless Personal Communications, vol. 121, no. 11, pp. 1-24, 2021.
[15] L. Zhang and W. Jinsong, "A hybrid method of entropy and SSAE-SVM based DDoS detection and mitigation mechanism in SDN," Computers & Security, vol. 115, 2022.
[16] H. Aydın, Z. Orman, and M. A. Aydın, "A long short-term memory (LSTM)-based distributed denial of service (DDoS) detection and defense system design in public cloud network environment," Computers & Security, vol. 118, 2022.
[17] M. M. Gowthul Alam and M. Raj, "An efficient SVM-based DEHO classifier to detect DDoS attack in a cloud computing environment," Computer Networks, vol. 215, pp. 109138, 2022.
[18] R. F. Fouladi, O. Ermiş, and E. Anarim, "A DDoS attack detection and countermeasure scheme based on DWT and auto-encoder neural network for SDN," Computer Networks, vol. 214, pp. 109140, 2022.
[19] R. K. Batchu and H. Seetha, "A generalized machine learning model for DDoS attacks detection using hybrid feature selection and hyperparameter tuning," Computer Networks, vol. 200, pp. 108498, 2021.
[20] K. Salunke and U. Ragavendran, "Shield techniques for application layer DDoS attack in MANET: A methodological review," Wireless Personal Communications, vol. 120, no. 4, pp. 2773-2799, 2021.
[21] P. V. Shalini, V. Radha, and S. G. Sanjeevi, "DOCUS-DDoS detection in SDN using modified CUSUM with flash traffic discrimination and mitigation", Computer Networks, vol. 217, pp. 109361, 2022.
[22] S. Sokkalingam and R. Ramakrishnan, "An intelligent intrusion detection system for distributed denial of service attacks: A support vector machine with hybrid optimisation algorithm-based approach," Concurrency and Computation: Practice and Experience, vol. 34, no. 27, pp. e7334, 2022.
[23] B. Raveendranadh and S. Tamilselvan, "An accurate attack detection framework based on exponential polynomial kernel-centered deep neural networks in the wireless sensor network," Transactions on Emerging Telecommunications Technologies, pp. e4726.
[24] R. Kaviarasan, P. Harikrishna, and A. Arulmurugan, "Load balancing in cloud environment using enhanced migration and adjustment operator based monarch butterfly optimization," Advances in Engineering Software, vol. 169, pp. 103128, 2022.
[25] H. Beitollahi, D. M. Sharif, and M. Fazeli, "Application layer DDoS attack detection using cuckoo search algorithm-trained radial basis function," IEEE Access, vol. 10, pp. 63844-63854, 2022.
[26] O. Yousuf and R. N. Mir, "DDoS attack detection in Internet of Things using recurrent neural network", Computers and Electrical Engineering, vol. 101, pp. 108034, 2022.
[27] Y. Shen, "An intrusion detection algorithm for DDoS attacks based on DBN and three-way decisions", in Journal of Physics: Conference Series, vol. 2356, no. One IOP Publishing 2022.
[28] H. Kumar, et al., "Light weighted CNN model to detect DDoS attack over distributed scenario", Security and Communication Networks, vol. 2022, pp. 1-11, 2022.
[29] A. Makuvaza, D. S. Jat, and A. M. Gamundani, "Deep neural network (DNN) solution for real-time detection of distributed denial of service (DDoS) attacks in software-defined networks (SDNs)," SN Computer Science, vol. 2, pp. 1-10, 2021.
[30] A. H. Mohammed, "Invasive weed optimization based ransomware detection in a cloud environment," 2021.
[31] S. Singh and S. K. V. Jayakumar, "DDoS attack detection in SDN: Optimized deep convolutional neural network with optimal feature set," Wireless Personal Communications, vol. 125, no. 3, pp. 2781-2797, 2022.
[32] H. Xu, Y. Lu, and Q. Guo, "Application of improved butterfly optimization algorithm combined with black widow optimization in feature selection of network intrusion detection," Electronics, vol. 11, no. 21, pp. 3531, 2022.
[33] N. Khare, et al., "Smo-dnn: Spider monkey optimization and deep neural network hybrid classifier model for

intrusion detection," Electronics, vol. 9, no. 4, pp. 692, 2020.
[34] G. O. Anyanwu, et al., "RBF-SVM kernel-based model for detecting DDoS attacks in SDN integrated vehicular network," Ad Hoc Networks, vol. 140, pp. 103026, 2023.
[35] D. K. Sharma, et al., "Anomaly detection framework to prevent DDoS attack in fog empowered IoT networks," Ad Hoc Networks, vol. 121, pp. 102603, 2021.
[36] R. Kalathiripi and N. Venkatram, "Regression coefficients of traffic flow metrics (RCTFM) for DDOS defense in IoT networks", International Journal of Communication Systems, vol. 34, no. 6, pp. e4330, 2021.
[37] S. Saravanan, et al., "Enhancing IoT security in MANETs: A novel adaptive defense reinforcement approach," Peer-to-Peer Networking and Applications, 2024.
[38] V. A. Vuyyuru, et al., "EsECC_SDN: Attack detection and classification model for MANET," Computers, Materials & Continua, vol. 74, no. 3, pp. 1-10, 2023.
[39] D. Gautam and V. Tokekar, "A novel approach for detecting DDoS attacks in MANET," Materials Today: Proceedings, vol. 29, pp. 674-677, 2020.
[40] S. Joardar, et al., "Mitigating DoS attack in MANETs considering node reputation with AI," Journal of Network and Systems Management, vol. 31, no. 3, pp. 50, 2023.
[41] L. V. Legashev et al.: "Development of a model for detecting network traffic anomalies in distributed wireless ad hoc networks", Scientific and Technical Journal of Information Technologies, Mechanics and Optics, vol. 22, no. 4, pp. 699–707, 2022.
[42] M. Şimşek and M. Toklu, "Two-layer approach for mixed high-rate and low-rate distributed denial of service (DDoS) attack detection and filtering," Arabian Journal for Science and Engineering, vol. 43, no. 12, pp. 7923-7931, 2018.
[43] M. Y. Kuçukkara, F. Atban, and C. Bayılmış, "Quantum-neural network model for platform-independent DDoS attack classification in cybersecurity. Advanced Quantum Technologies, 2024.
[44] A. Albakri, et al., "Metaheuristics with deep learning model for cybersecurity and Android malware detection and classification", Applied Sciences, vol. 13, no. 4, pp. 2172, 2023.
[45] Sivanesan, N., et al. "Detecting distributed denial of service (DDoS) in MANET using Ad Hoc on-demand distance vector (AODV) with extra tree classifier (ETC)." Iranian Journal of Science and Technology, Transactions of Electrical Engineering 48.2 (2024): 645-659.